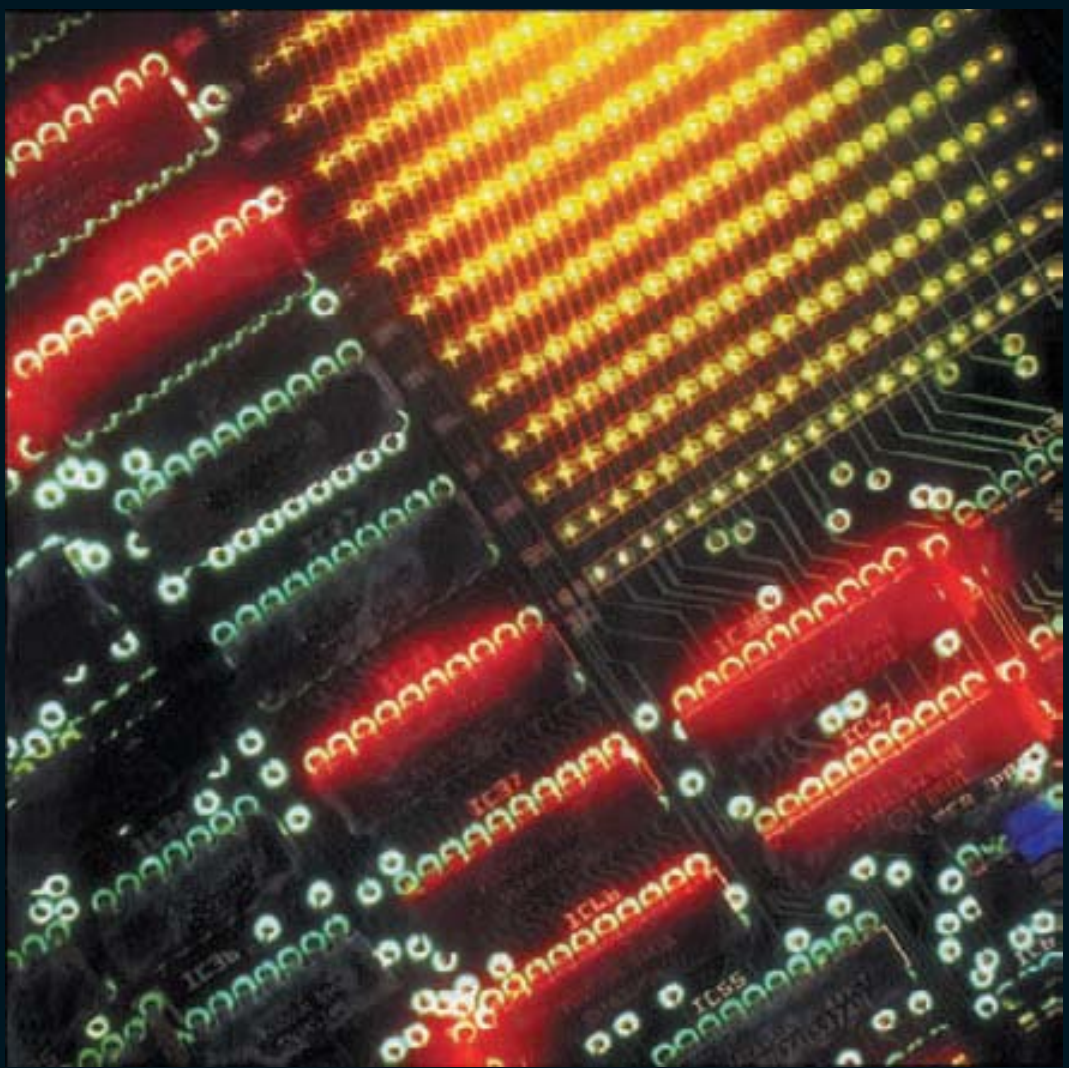
DÉCIMA EDICIÓN

INCLUYE CD-ROM

SISTEMAS DIGITALES

PRINCIPIOS Y APLICACIONES



RONALD J. TOCCI
NEAL S. WIDMER
GREGORY L. MOSS

PEARSON
Prentice Hall ®

Sistemas digitales

Principios y aplicaciones

DÉCIMA EDICIÓN

Sistemas digitales

Principios y aplicaciones

Ronald J. Tocci
Monroe Community College

Neal S. Widmer
Purdue University

Gregory L. Moss
Purdue University

Traducción

Alfonso Vidal Romero Elizondo
Ingeniero en Electrónica y Comunicación
Instituto Tecnológico y de Estudios Superiores de Monterrey - Campus Monterrey

Revisión técnica

Reynaldo Félix Acuña
Profesor investigador
Departamento de Ingeniería Eléctrica y Electrónica
Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Estado de México

Marcos de Alba
Profesor investigador
Departamento de Ingeniería Eléctrica y Electrónica
Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Estado de México

PEARSON Educación ®

México • Argentina • Brasil • Colombia • Costa Rica • Chile • Ecuador
España • Guatemala • Panamá • Perú • Puerto Rico • Uruguay • Venezuela

Datos de catalogación bibliográfica

TOCCI, RONALD J., NEAL S. WIDMER, GREGORY L. MOSS

Sistemas digitales.
Principios y aplicaciones
Décima edición

Pearson Educación, México, 2007
ISBN: 978-970-26-0970-4
Área: Ingeniería

Formato: 21 × 27 cm Páginas: 968

Authorized traslation from the English language edition, entitled *Digital systems: principles and applications* by *Ronald J. Tocci*, *Neal S. Widmer* and *Gregory L. Moss*, published by Pearson Education, Inc., publishing as Prentice Hall, Inc., Copyright ©2007, All rights reserved.

ISBN 0131725793

Traducción autorizada de la edición en idioma inglés titulada *Digital systems: principles and applications* por *Ronald J. Tocci*, *Neal S. Widmer* and *Gregory L. Moss*, publicada por Pearson Education, Inc., publicada como Prentice Hall Inc., Copyright ©2007. Todos los derechos reservados.

Edición en español

Editor:	Luis Miguel Cruz Castillo
	e-mail: luis.cruz@pearsoned.com
Editor de desarrollo:	Bernardino Gutiérrez Hernández
Supervisor de producción:	Adriana Rida Montes

DÉCIMA EDICIÓN, 2007

D.R. © 2007 por Pearson Educación de México, S.A. de C.V.
Atlacomulco 500-5o. piso
Industrial Atoto
53519, Naucalpan de Juárez, Edo. de México

Cámara Nacional de la Industria Editorial Mexicana. Reg. Núm. 1031.

Prentice Hall es una marca registrada de Pearson Educación de México, S.A. de C.V.

Reservados todos los derechos. Ni la totalidad ni parte de esta publicación pueden reproducirse, registrarse o transmitirse, por un sistema de recuperación de información, en ninguna forma ni por ningún medio, sea electrónico, mecánico, fotoquímico, magnético o electroóptico, por fotocopia, grabación o cualquier otro, sin permiso previo por escrito del editor.

El préstamo, alquiler o cualquier otra forma de cesión de uso de este ejemplar requerirá también la autorización del editor o de sus representantes.

ISBN 10: 970-26-0970-4

ISBN 13: 978-970-26-0970-4

Impreso en México. *Printed in Mexico.*
1 2 3 4 5 6 7 8 9 0 - 10 09 08 07



A ti, Cap, por amarme tanto tiempo; y por las mil y una maneras de iluminar las vidas de todos los que tocas.

—RJT

A mi esposa Kris y a nuestros hijos John, Brad, Blake, Matt y Katie: por prestarme el tiempo y atención que les quité al realizar esta revisión.

—NSW

A mi familia: Marita, David y Ryan.

—GLM



PREFACIO

Este libro es un estudio completo sobre los principios y las técnicas de los sistemas digitales modernos. Enseña los principios fundamentales sobre los sistemas digitales y cubre con amplitud los métodos tradicional y moderno para aplicar las técnicas de diseño y desarrollo digital, incluyendo cómo administrar un proyecto a nivel de sistemas. El libro está planeado para utilizarse en programas de dos y cuatro años relacionados con tecnología, ingeniería y ciencias computacionales. Aunque sería de utilidad tener conocimientos sobre electrónica básica, la mayor parte del material no lo requiere. Se pueden omitir las partes del texto en las que se utilizan conceptos sobre electrónica sin peligro de afectar la comprensión de los principios de lógica.

Mejoras generales

La décima edición de *Sistemas digitales* refleja los puntos de vista de los autores en relación con la dirección de la electrónica digital moderna. En la industria actual podemos ver la importancia de liberar un producto en el mercado lo más rápido posible. El uso de herramientas de diseño modernas, dispositivos lógicos programables complejos, CPLD, y arreglos de compuertas programables por campo (FPGA) permite a los ingenieros progresar de los conceptos al silicón funcional con mucha rapidez. Los microcontroladores se están haciendo cargo de muchas aplicaciones que antes se implementaban mediante circuitos digitales, por lo que se ha estado usando la DSP para sustituir muchos circuitos análogos. Es sorprendente que los microcontroladores, la DSP y todos los componentes lógicos necesarios puedan ahora consolidarse en una sola FPGA mediante el uso de un lenguaje de descripción de hardware con herramientas de desarrollo avanzadas. Ahora, los estudiantes deben estar expuestos a estas modernas herramientas, inclusive hasta en un curso introductorio. Es responsabilidad de cada educador encontrar la mejor forma de preparar a los graduandos para el trabajo que encontrarán en su vida profesional.

Las piezas SSI y MSI estándar que han servido como “ladrillos y mortero” en la construcción de sistemas digitales durante más de 40 años ahora se están volviendo obsoletas. Muchas de las técnicas que se han enseñado se enfocan en la optimización de circuitos que se crean a partir de estos dispositivos fuera de moda. Los temas que se adaptan en forma única a la aplicación de la antigua tecnología *pero que no contribuyen a una comprensión de la nueva tecnología* deben eliminarse del currículum. No obstante y desde un punto de vista educativo, estos pequeños

circuitos integrados (CI) ofrecen una forma de estudiar los circuitos digitales simples, además de que el cableado de circuitos mediante el uso de "breadboards" (tarjetas experimentales o tableros experimentales) es un ejercicio pedagógico valioso. Estos ejercicios nos ayudan a reforzar conceptos tales como las entradas y salidas binarias, la operación de un dispositivo físico y las limitaciones prácticas, mediante el uso de una plataforma muy simple. En consecuencia hemos optado por seguir presentando las descripciones conceptuales de los circuitos digitales y ofrecer ejemplos en los que se utilicen piezas lógicas estándar convencionales. Para los instructores que deseen seguir enseñando los fundamentos mediante el uso de circuitos SSI y MSI, este libro retiene esas cualidades que han provocado una amplia aceptación de las ediciones pasadas. Incluso muchas herramientas de diseño de hardware proporcionan una técnica introductoria de diseño fácil de usar, en la que se emplea la funcionalidad de las piezas estándar convencionales con la flexibilidad de los dispositivos lógicos programables. Un diseño digital puede describirse mediante el uso de un dibujo esquemático con bloques de construcción predefinidos, los cuales son equivalentes a las piezas estándar convencionales, que pueden compilarse y después programarse directamente en un PLD de destino con la capacidad adicional de poder simular con facilidad el diseño dentro de la misma herramienta de desarrollo.

Creemos que los graduandos podrán aplicar los conceptos que se presentan en este libro mediante el uso de métodos de descripción de nivel superior y dispositivos programables más complejos. El cambio más importante en el campo es una mayor necesidad de comprender los métodos de descripción, en vez de enfocarse en la arquitectura de un dispositivo específico. Las herramientas de software han evolucionado a un punto en el que hay muy poca necesidad de preocuparse por el funcionamiento interno del hardware y es mucho más necesario enfocarse en qué es lo que entra, qué es lo que sale y cómo puede el diseñador describir lo que se supone debe hacer el dispositivo. También creemos que los graduandos se involucrarán con proyectos en los que se utilice lo más nuevo en herramientas de diseño y soluciones de hardware.

Este libro ofrece una ventaja estratégica para enseñar el nuevo y vital tema de los lenguajes de descripción de hardware a los principiantes en el campo digital. En estos momentos, VHDL es sin duda un lenguaje estándar en la industria, pero también es muy complejo y con una curva de aprendizaje bastante pronunciada. A menudo los estudiantes principiantes se desalientan debido a los rigurosos requerimientos de diversos tipos de datos, y luchan por comprender los eventos de disparo por borde que se utilizan en VHDL. Por fortuna Altera ofrece el AHDL, un lenguaje menos exigente que utiliza los mismos conceptos básicos que el VHDL pero más sencillo de dominar para los principiantes. Así, los instructores pueden optar por utilizar AHDL para enseñar a los estudiantes principiantes, o VHDL para las clases más avanzadas. Esta edición ofrece más de 40 ejemplos con AHDL, otros tantos con VHDL y muchos ejemplos de prueba de simulación. Todos estos archivos de diseño están disponibles en el CD-ROM que acompaña al libro.

El sistema de desarrollo de software más reciente de Altera es Quartus II. El software MAX+ PLUS II que se ha utilizado durante muchos años aún sigue siendo popular en la industria y cuenta con soporte por parte de Altera. Su principal desventaja es que no programa los dispositivos más recientes. El material de este texto no está enfocado a enseñar una plataforma específica de hardware ni los detalles acerca del uso de un sistema de desarrollo de software. Las nuevas revisiones de software aparecen con tanta frecuencia que un libro de texto no podría mantenerse actualizado si tratara de describir todos los detalles. Hemos tratado de demostrar lo que esta herramienta puede hacer, en vez de enseñar al lector cómo utilizarla. Sin embargo, en el CD-ROM que viene con este libro hemos incluido tutoriales para facilitar el aprendizaje de cualquiera de estos dos paquetes de software. Los ejemplos de AHDL y VHDL son compatibles con los sistemas Quartus o MAX+ PLUS. Las simulaciones de temporización se desarrollaron mediante el uso de MAX+ PLUS, pero también pueden realizarse con Quartus.

Hay muchas opciones de hardware de laboratorio disponibles para los usuarios de este libro. Existen muchas tarjetas de desarrollo CPLD y FPGA para que



FIGURA P-1 La tarjeta de desarrollo UP3 de Altera.

los estudiantes puedan usarlas en el laboratorio. También se dispone de varias tarjetas de generaciones anteriores que son similares a la UP2 de Altera y que contienen CPLDs de la familia MAX7000. Un ejemplo más reciente de una de las tarjetas disponibles es la tarjeta UP3 del programa universitario de Altera (vea la figura P-1), que contiene una FPGA más grande, proveniente de la familia Cyclone. La tarjeta DE2 es una de las más reciente de Altera (vea la figura P-2), la cual cuenta con una nueva y poderosa FPGA Cyclone II de 672 terminales y una variedad de características básicas tales como interruptores, LEDs y pantallas, así como muchas características adicionales para proyectos más avanzados. Cada año entran al mercado más tarjetas de desarrollo, y muchas de ellas tienen un costo realmente bajo. Estas tarjetas, junto con el poderoso software educativo, ofrecen una excelente manera de enseñar y demostrar la implementación práctica de los conceptos que presentamos en este texto.

Las mejoras más considerables en esta décima edición se encuentran en el capítulo 7. Aunque los contadores asincrónicos (ondulación) ofrecen una buena introducción a los circuitos secuenciales, en el mundo real se utilizan los circuitos contadores sincrónicos. Hemos rediseñado el capítulo 7 y los ejemplos subsiguientes

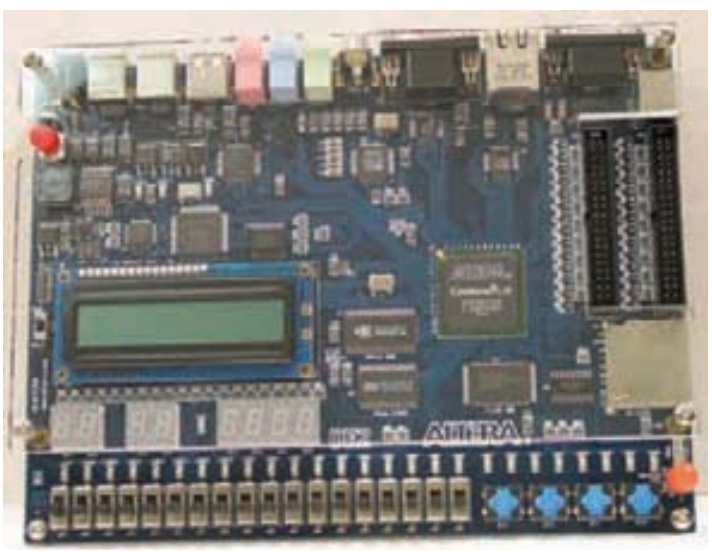


FIGURA P-2 La tarjeta de desarrollo DE2 de Altera.

para enfatizar los CIs de contadores sincrónicos e incluimos técnicas para analizarlos, conectarlos en cascada y utilizar HDL para describirlos. También se ha agregado una sección para mejorar la cobertura de las máquinas de estado y las características de HDL que se utilizan para describirlas. Otras de las mejoras son: la inclusión de técnicas de análisis para los circuitos combinacionales, la cobertura expandida de aplicaciones del temporizador 555 y una mejor cobertura de los números binarios con signo.

Nuestro enfoque sobre el HDL y los PLDs proporciona a los instructores varias opciones:

1. El material relacionado con el HDL puede omitirse por completo sin afectar la continuidad del texto.
2. El HDL puede enseñarse como un tema separado si se omite el material inicial y después se regresa a las últimas secciones de los capítulos 3, 4, 5, 6, 7 y 9, para después cubrir el 10.
3. Los temas relacionados con el HDL y el uso de los PLDs pueden cubrirse a medida que se desarrolle el curso (capítulo por capítulo) y pueden irse entremezclando en las prácticas de laboratorio o las lecturas.

De entre todos los lenguajes de descripción de hardware específicos, es evidente que VHDL es el estándar en la industria, por lo cual es muy probable que los graduandos lo vayan a utilizar en sus profesiones. No obstante, siempre hemos creído que es muy pesado tratar de enseñar VHDL en un curso introductorio. La naturaleza de la sintaxis, las sutiles distinciones en los tipos de objetos y los mayores niveles de abstracción pueden convertirse en obstáculos para un principiante. Por esta razón hemos incluido el AHDL de Altera como el lenguaje de introducción recomendado para los cursos de primero y segundo años. También hemos incluido el VHDL como lenguaje recomendado para clases más avanzadas o cursos introductorios que se ofrecen a estudiantes con más madurez. No recomendamos tratar de cubrir ambos lenguajes en el mismo curso. Las secciones del texto que cubren los detalles específicos de un lenguaje se identifican claramente con una barra gris en el margen. Sin duda hemos creado un libro que puede usarse en múltiples cursos y que servirá como una excelente referencia después de que se gradúen los estudiantes.

Organización de los capítulos

Es raro que un instructor utilice los capítulos de un libro de texto en la secuencia en la que se presentan. Este libro se escribió de manera que, en su mayor parte, cada capítulo se basa en el material anterior, pero es posible alterar la secuencia de capítulos hasta cierto punto. La primera parte del capítulo 6 puede cubrirse justo después del capítulo 2, aunque esto significa que habrá un largo intervalo antes de llegar a los circuitos aritméticos del capítulo 6. La mayor parte del material del capítulo 8 puede cubrirse antes (por ejemplo, después del capítulo 4 o 5) sin problemas graves.

Este libro puede utilizarse ya sea en un curso de un solo ciclo escolar o en una secuencia de dos ciclos. En un curso de un ciclo escolar tal vez haya que omitir algunos temas, debido a los límites en cuanto a las horas de clase disponibles. Es obvio que la elección de los temas a eliminar dependerá de factores tales como los objetivos del programa o del curso y los antecedentes de los estudiantes. A continuación se muestra una lista de las secciones y capítulos que pueden eliminarse con la menor probabilidad de trastornos:

- Capítulo 1: Todo.
- Capítulo 2: Sección 6.
- Capítulo 3: Secciones 15 a 20.
- Capítulo 4: Secciones 7, 10 a 13.

- Capítulo 5: Secciones 3, 23 a 27.
- Capítulo 6: Secciones 5 a 7, 11, 13, 16 a 23.
- Capítulo 7: Secciones 9 a 14, 21 a 24.
- Capítulo 8: Secciones 10, 14 a 19.
- Capítulo 9: Secciones 5, 9, 15 a 20.
- Capítulo 10: Todo.
- Capítulo 11: Secciones 7, 14 a 17.
- Capítulo 12: Secciones 17 a 21.
- Capítulo 13: Todo.

CONJUNTOS DE PROBLEMAS En esta edición se incluyen seis categorías de problemas: básicos (B), avanzados (A), diagnóstico de fallas (F), nuevos (N), diseño (D) y HDL (H). Los problemas sin designación se consideran como de dificultad intermedia, entre los básicos y los avanzados. Los problemas para los cuales se imprime la solución al final del libro o en el CD-ROM incluido están marcados con un asterisco (vea la figura P-3).

ADMINISTRACIÓN DE PROYECTOS Y DISEÑO A NIVEL DE SISTEMA En el capítulo 10 se incluyen varios ejemplos reales para describir las técnicas que se utilizan para administrar proyectos. Por lo general, estas aplicaciones son familiares para la mayoría de los estudiantes de electrónica, y el primer ejemplo de un reloj digital es familiar para todos. Muchos textos hablan sobre el diseño de arriba-abajo, pero este texto demuestra las características clave de este método y cómo utilizar las modernas herramientas para desarrollarlo.

HOJAS TÉCNICAS Hemos eliminado el CD-ROM que contenía hojas técnicas de Texas Instruments de la novena edición. La información que contenía este CD-ROM se encuentra ahora disponible en línea.

ARCHIVOS DE SIMULACIÓN En esta edición también se incluyen archivos de simulación que pueden cargarse en el programa Electronics Workbench Multisim®. Los diagramas esquemáticos de los circuitos de muchas de las figuras que se muestran en el texto se han capturado como archivos de entrada para esta popular herramienta de simulación. Cada archivo demuestra en cierta forma la operación del circuito o refuerza un concepto. En muchos casos se adjuntan instrumentos al circuito y se aplican secuencias de entrada para demostrar el concepto presentado en una de las figuras del texto. Estos circuitos pueden modificarse según se desee para profundizar sobre los temas o crear asignaturas y tutoriales para los estudiantes.

FIGURA P-3 Las letras denotan categorías de problemas, y los asteriscos indican que se proporcionan las soluciones correspondientes al final del texto.

PROBLEMAS

SECCIÓN 9-1

B 9-1. Consulte la figura 9-3. Determine los niveles en la salida de cada decodificador para los siguientes conjuntos de condiciones de entrada.
(a)* Todas las entradas en BAJO.
(b)* Todas las entradas en bajo, excepto E_3 = ALTO.
(c) Todas las entradas en ALTO excepto $\overline{E}_1 = \overline{E}_2$ = BAJO.
(d) Todas las entradas en ALTO.

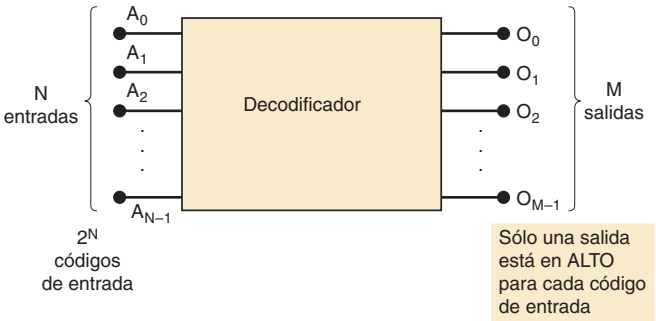
B 9-2.* ¿Cuál es el número de entradas y salidas de un decodificador que acepta 64 distintas combinaciones de entrada?

* Encontrará las respuestas a los problemas marcados con un asterisco al final del libro.

FIGURA P-4 El icono denota que la figura tiene su correspondiente archivo de simulación en el CD-ROM.

FIGURA 9-1 Diagrama general de un decodificador.





Todas las figuras en el texto que tienen su correspondiente archivo de simulación en el CD-ROM se identifican mediante el icono que se muestra en la figura P-4.

TECNOLOGÍA DE CI Esta nueva edición continúa la práctica que empezamos con las últimas tres ediciones que se refiere a dar más prominencia a la tecnología CMOS como principal tecnología de CI en aplicaciones de integración de pequeña y media escala. Hemos logrado esta profundidad en la cobertura, al mismo tiempo que pudimos retener una amplia cobertura de la lógica TTL.

Cambios específicos

A continuación se muestran los principales cambios en la cobertura de los temas:

- **Capítulo 1** Hemos actualizado y mejorado muchas explicaciones que tratan sobre cuestiones análogas/digitales.
- **Capítulo 2** Eliminamos el sistema numérico octal y agregamos el código de Gray. Incluimos una tabla de códigos ASCII estándar completa, junto con nuevos ejemplos relacionados con los caracteres ASCII, la representación hexadecimal y los archivos de transferencia de código objeto de las computadoras. También agregamos nuevo material sobre las tramas de caracteres ASCII para la transferencia asincrónica de datos.
- **Capítulo 3** Además de algunos nuevos ejemplos prácticos de funciones lógicas, la principal mejora en el capítulo 3 es una nueva técnica de análisis que utiliza tablas para evaluar puntos intermedios en el circuito lógico.
- **Capítulo 4** En el capítulo 4 sólo fue necesario realizar unos cuantos cambios.
- **Capítulo 5** Hay una nueva sección que trata acerca de los pulsos digitales y las definiciones asociadas, tales como: anchura de pulso, periodo, tiempo de elevación y tiempo de caída. Se modificó la terminología utilizada para las entradas de los circuitos de latch, de Clear (Borrar) a Reset (Restablecer) para que fuera compatible con las descripciones de los componentes de Altera. También se eliminó la definición de un flip-flop maestro/esclavo. Se mejoró la explicación sobre las aplicaciones de circuitos disparadores de Schmitt (Schmitt trigger) para enfatizar su papel en cuanto a la eliminación de los efectos del ruido. Ahora se explica el funcionamiento interno del temporizador 555 y se proponen ciertos circuitos de temporización mejorados que hacen que el dispositivo sea más versátil. Se rediseñó la cobertura sobre el HDL para los latches SR y D, para utilizar una descripción más descriptiva de su comportamiento, y se modificó la cobertura de los contadores para enfocarse en las técnicas estructurales para interconectar bloques de flip-flops.
- **Capítulo 6** En esta edición se cubren con más detalle los números con signo, en especial todo lo relacionado con la extensión de signo en los números con complemento a 2 y el desbordamiento aritmético. Una nueva sugerencia de calculadora simplifica la negación de números binarios representados en hexadecimal. Se utiliza un modelo de círculo de números para comparar los formatos

de números con y sin signo, lo cual ayuda a los estudiantes para que visualicen la operación de suma/resta mediante el uso de ambos formatos.

- **Capítulo 7** Este capítulo se revisó con mucho detalle para enfatizar los circuitos de contadores sincrónicos. Se siguen introduciendo los contadores de ondulación simple para proporcionar una comprensión básica del concepto de conteo y la conexión en cascada asincrónica. Después de examinar las limitaciones de los contadores de ondulación en la sección 2, se presentan los contadores sincrónicos en la sección 3 y se utilizan en todos los ejemplos subsiguientes a lo largo del texto. Los contadores de CI que se presentan son: 74160, '161, '162 y '163. Estos dispositivos comunes ofrecen una excelente diversidad de características que enseñan la diferencia entre las entradas de control sincrónicas y asincrónicas, y las técnicas de conexión en cascada. Los circuitos 74190 y '191 se utilizan como ejemplo de un CI de contador sincrónico arriba/abajo, con lo cual se refuerzan aún más las técnicas requeridas para la conexión sincrónica en cascada. Hay una nueva sección dedicada a las técnicas de análisis para circuitos sincrónicos mediante el uso de flip-flops JK y D. Las técnicas de diseño sincrónico ahora incluyen también el uso de registros de flip-flop D que representan la mejor manera en que se implementan los circuitos secuenciales en la tecnología PLD moderna. Se mejoraron las secciones sobre HDL para demostrar la implementación de las operaciones sincrónicas/asincrónicas de cargar, borrar y conectar en cascada. Otra de las novedades es un énfasis en la simulación y la prueba de los módulos de HDL. Ahora las máquinas de estado se presentan como un tema, se definen los modelos tradicionales de Mealy y Moore y se presenta un nuevo sistema de control de semáforos como ejemplo. También se hicieron pequeñas mejoras en la segunda mitad del capítulo 7 y se rediseñaron todos los problemas al final del capítulo 7 para reforzar los conceptos.
- **Capítulo 8** Este capítulo sigue siendo una descripción muy técnica de la tecnología disponible en las familias lógicas estándar y los componentes digitales. Se mejoraron las secciones de interfaz de voltaje mixto para cubrir la tecnología de bajo voltaje. La curva de ciclo de vida más reciente de Texas Instruments muestra el historial y la posición actual de diversas series lógicas, desde su introducción hasta su obsolescencia. También se presenta la señalización diferencial de bajo voltaje (LVDS).
- **Capítulo 9** Este capítulo trata todavía acerca de los fundamentos de los sistemas digitales y su demostración mediante el uso de HDL. También se introducen muchas otras técnicas de HDL, como las salidas de tres estados y diversas estructuras de control de HDL. Se describe un circuito 74ALS148 como otro ejemplo de codificador. Se actualizaron todos los ejemplos de sistemas que utilizan contadores para la operación sincrónica. En especial, se mejoró el sistema de transmisión en serie mediante el uso de MUX y DEMUX. Se explica la técnica de cómo utilizar un MUX para implementar expresiones SOP en una forma más estructurada, como un ejercicio de estudio independiente en los problemas al final del capítulo.
- **Capítulo 10** Este capítulo, que se introdujo como novedad en la novena edición, no sufrió cambios considerables.
- **Capítulo 11** Se mejoró el material sobre DACs bipolares, y se presentó un ejemplo sobre el uso de DACs como control de amplitud digital para formas de onda analógicas. En esta edición se explica la especificación de precisión de convertidores A/D más común en forma de +/− LSB.
- **Capítulo 12** Se realizaron pequeñas mejoras en este capítulo para consolidar y comprimir algo del material sobre tecnologías antiguas de memoria, tales como la UV EPROM. Aún se presenta la tecnología flash mediante el uso de un ejemplo de primera generación, pero también se describen las mejoras más recientes, así como algunas de las aplicaciones de la tecnología flash en los dispositivos modernos disponibles para el consumidor.
- **Capítulo 13** Este capítulo, nuevo en la novena edición, se actualizó para introducir la nueva familia Cyclone de PLDs.

Características que se retuvieron

Esta edición retiene todas las características que contribuyeron a la amplia aceptación de las ediciones anteriores. Utiliza un enfoque de diagrama de bloques para enseñar las operaciones lógicas básicas sin confundir al lector con los detalles de su operación interna. Se conservan todas las características eléctricas de los CIs lógicos, excepto las más básicas, hasta que el lector cuente con una sólida comprensión de los principios lógicos. En el capítulo 8 se introducen los CI internos. En ese punto, el lector podrá interpretar las características de entrada y salida de un bloque lógico y "ajustarlo" en forma apropiada a un sistema completo.

Por lo general, el tratamiento de cada nuevo tema o dispositivo sigue estos pasos: se presenta el principio de operación; se presentan ejemplos y aplicaciones con una explicación detallada, a menudo mediante el uso de CIs reales; se plantean preguntas cortas de repaso al final de la sección; y por último se presentan problemas a profundidad al final del capítulo. Estos problemas, que van desde lo más simple hasta lo más complejo, ofrecen a los instructores una amplia elección para los estudiantes. El principal objetivo de estos problemas es reforzar el material pero no sólo con repetir los principios, ya que requieren que los estudiantes demuestren la comprensión de estos principios al aplicarlos a distintas situaciones. Este enfoque también ayuda a los estudiantes a desarrollar confianza y a expandir su conocimiento sobre el material.

El material relacionado con los PLDs y HDLs se distribuye a lo largo del texto, con ejemplos en los que se enfatizan las características clave en cada aplicación. Estos temas aparecen al final de cada capítulo, lo que facilita relacionar cada tema con el texto general al principio del capítulo, o se puede analizar por separado el material que trata sobre los conceptos de PLD/HDL.

La extensa cobertura relacionada con la detección y solución de problemas se extiende desde el capítulo 4 hasta el 12, incluyendo la presentación de los principios y las técnicas de detección y solución de problemas, ejemplos prácticos, 25 ejemplos de detección y solución de problemas, y además, 60 problemas *reales* relacionados con este mismo tema. Si se complementa con los ejercicios prácticos de laboratorio, este material puede ayudar a impulsar el desarrollo de buenas habilidades para la detección y solución de problemas.

La décima edición ofrece más de 200 ejemplos resueltos, más de 400 preguntas de repaso y más de 450 problemas/ejercicios de capítulo. Algunos de estos problemas son aplicaciones que muestran cómo se utilizan en un típico sistema de microcomputadora los dispositivos lógicos presentados en el capítulo. Después del Glosario están las respuestas a la mayoría de los problemas. El Glosario provee definiciones concisas de todos los términos que están resaltados en el texto, en negritas.

Al final del libro hay un índice de CIs para ayudar a que los lectores localicen con facilidad el material sobre cualquier CI citado o utilizado en el texto. Las últimas hojas del libro contienen tablas de los teoremas de álgebra Booleana más usados, síntesis de compuertas lógicas y tablas de verdad de flip-flops para una rápida referencia al resolver problemas o trabajar en el laboratorio.

Suplementos (en inglés)

Para acompañar a este libro de texto se ha desarrollado un detallado complemento con herramientas de enseñanza y aprendizaje. Cada componente provee una función única, y cada uno de ellos puede usarse ya sea en forma independiente o en conjunto con los demás.

CD-ROM Cada libro incluye un CD-ROM, con lo siguiente:

- **MAX+PLUS® II, software versión Educativa de Altera.** Éste es un entorno integrado de desarrollo completamente funcional, de calidad profesional para sistemas digitales, el cual se ha utilizado durante muchos años y sigue teniendo soporte por parte de Altera. Los estudiantes pueden utilizarlo para escribir,

compilar y simular sus diseños en casa, antes de ir al laboratorio. Pueden usar el mismo software para programar y probar un CPLD de Altera.

- **Quartus II, software Versión Web de Altera.** Éste es el software de sistema de desarrollo más reciente de Altera, el cual ofrece características más avanzadas y soporta los nuevos dispositivos PLD, tales como la familia Cyclone de FPGAs, que se encuentran en muchas de las tarjetas educacionales más recientes.
- **Tutoriales.** Gregory Moss ha desarrollado tutoriales que se han utilizado con éxito durante varios años para enseñar a los estudiantes principiantes a utilizar el software MAX+PLUS II de Altera. Estos tutoriales están disponibles en los formatos PDF y PPT (presentación de Microsoft® PowerPoint®), y también se han adaptado para enseñar Quartus II. Con la ayuda de estos tutoriales, cualquiera puede aprender a modificar y probar todos los ejemplos que se presentan en este texto, así como a desarrollar sus propios diseños.
- **Archivos de diseño de las figuras del libro de texto.** Hay más de 40 archivos de diseño en cada lenguaje que se presenta en figuras a lo largo del texto. Los estudiantes pueden cargar estos archivos en el software de Altera y probarlos.
- **Soluciones a problemas selectos: archivos de diseño de HDL.** Algunas de las soluciones a los problemas al final del capítulo se encuentran disponibles (en inglés) para los estudiantes. (Todas las soluciones de HDL están disponibles para los instructores en el *Manual para el instructor*.) Las soluciones para los problemas del capítulo 7 incluyen algunos archivos de gráficos extensos y de HDL que no se publican en el libro, pero están disponibles en el CD-ROM.
- **Circuitos del texto modelizados en Multisim®.** Es posible abrir y trabajar en forma interactiva con cerca de 100 circuitos para incrementar su comprensión de los conceptos y prepararse para las actividades de laboratorio. Se proporcionan archivos de circuitos Multisim para cualquiera que tenga este software. Quienes no tengan el software Multisim y deseen comprarlo para usar los archivos de circuitos pueden ordenarlo en el sitio Web www.prenhall.com/ewb.
- **Material suplementario de introducción a los microprocesadores y los microcontroladores.** Para tener flexibilidad al dar servicio a las diversas necesidades de las distintas escuelas, se presenta una introducción a este tema como una conveniente interfaz entre un curso de sistemas digitales y un curso de introducción a los microprocesadores/microcontroladores.

RECURSOS PARA LOS ESTUDIANTES (en inglés)

- **Sitio Web complementario (www.pearsoneducacion.net/tocci).** Este sitio ofrece a los estudiantes una guía de estudio en línea gratuita, con la cual pueden revisar el material tratado en el texto y comprobar su comprensión de los temas clave.

RECURSOS PARA EL INSTRUCTOR (en inglés)

- ***Manual para el instructor.*** Este manual contiene las soluciones para todos los problemas de final de capítulo que vienen en este libro de texto.
- ***Manual de soluciones de laboratorio.*** Se incluyen los resultados de ambos manuales de laboratorio.
- **Presentaciones en PowerPoint®.** Las figuras del texto, además de Notas para el orador para cada capítulo.
- **TestGen.** Hay un banco de pruebas computarizado.

Para acceder a los materiales suplementarios en línea, los instructores deberán solicitar un código de acceso, luego ir a www.pearsoneducacion.net/tocci, haga clic en el vínculo **Instructor Resource Center** y después haga clic en **Register** para obtener un código de acceso de instructor. Después de registrarse, en un plazo no mayor a 48 horas recibirá un correo electrónico de confirmación en el que se incluirá el código de acceso. Cuando haya recibido su código, vaya al sitio e inicie sesión para obtener las instrucciones completas sobre cómo descargar los materiales que desee utilizar.

AGRADECIMIENTOS

Agradecemos a todas aquellas personas que evaluaron la novena edición y proporcionaron respuestas para un extenso cuestionario: Ali Khabari, Wentworth Institute of Technology; Al Knebel, Monroe Community College; Rex Fisher, Brigham Young University; Alan Niemi, LeTourneau University; y Roger Sash, University of Nebraska. Se tomaron muy en cuenta sus comentarios, críticas y sugerencias, cuyo incalculable valor nos fue de utilidad para determinar el formato final de la décima edición.

También estamos muy agradecidos con el profesor Frank Ambrosio, Monroe Community College, por su usual labor de alta calidad en los índices y en el *Manual de recursos para el instructor*; con Thomas L. Robertson, Purdue University, por proveer su sistema de levitación magnética como ejemplo; y con los profesores Russ Aubrey y Gene Harding, Purdue University, por la revisión técnica de los temas y muchas sugerencias para realizar mejoras. Apreciamos la cooperación de Mike Phipps y de Altera Corporation por su apoyo al otorgar el permiso para usar su paquete de software y las figuras de sus publicaciones técnicas.

Un proyecto de esta magnitud requiere de un soporte editorial exhaustivo y profesional, y una vez más Prentice Hall logró su cometido con excelencia. Agradecemos al personal en Prentice Hall y TechBooks/GTS por su ayuda para hacer que esta publicación sea un éxito.

Por último, queremos hacer saber a nuestras esposas e hijos cuánto apreciamos su apoyo y su comprensión. Esperamos poder reponer gradualmente todas las horas que pasamos alejados de ellos mientras trabajábamos en esta revisión.

Ronald J. Tocci
Neal S. Widmer
Gregory L. Moss



RESUMEN DE CONTENIDO

CAPÍTULO 1	Conceptos introductorios	2
CAPÍTULO 2	Sistemas y códigos numéricos	24
CAPÍTULO 3	Descripción de los circuitos lógicos	54
CAPÍTULO 4	Circuitos lógicos combinacionales	118
CAPÍTULO 5	Flip-flops y dispositivos relacionados	208
CAPÍTULO 6	Aritmética digital: operaciones y circuitos	296
CAPÍTULO 7	Contadores y registros	360
CAPÍTULO 8	Familias lógicas de circuitos integrados	488
CAPÍTULO 9	Circuitos lógicos MSI	576
CAPÍTULO 10	Proyectos de sistemas digitales mediante el uso de HDL	676
CAPÍTULO 11	Interface con el mundo analógico	718
CAPÍTULO 12	Dispositivos de memoria	784
CAPÍTULO 13	Arquitecturas de los dispositivos lógicos programables	868
	Glosario	898
	Respuestas a los problemas seleccionados	911
	Índice de CIs	919
	Índice	922



CONTENIDO

CAPÍTULO 1 Conceptos introductorios 2

1-1 Representaciones numéricas 4
1-2 Sistemas digitales y analógicos 5
1-3 Sistemas numéricos digitales 10
1-4 Representación de cantidades binarias 13
1.5 Circuitos digitales/circuitos lógicos 15
1-6 Transmisión en paralelo y en serie 17
1-7 Memoria 18
1-8 Computadoras digitales 19

CAPÍTULO 2 Sistemas y códigos numéricos 24

2-1 Conversiones de binario a decimal 26
2-2 Conversiones de decimal a binario 26
2-3 Sistema numérico hexadecimal 29
2-4 Código BCD 33
2-5 Código gray 35
2-6 Integración de los sistemas numéricos 37
2-7 Byte, nibble y palabra 37
2-8 Códigos alfanuméricos 39
2-9 Método de paridad para la detección de errores 41
2-10 Aplicaciones 44

CAPÍTULO 3 Descripción de los circuitos lógicos 54

3-1 Constantes y variables booleanas 57
3-2 Tablas de verdad 57
3-3 Operación OR con compuertas OR 58
3-4 Operación AND con compuertas AND 62
3-5 Operación NOT 65
3-6 Descripción de circuitos lógicos en forma algebraica 66
3-7 Evaluación de las salidas de circuitos lógicos 68
3-8 Implementación de circuitos a partir de expresiones booleanas 71
3-9 Compuertas NOR y NAND 73
3-10 Teoremas booleanos 76
3-11 Teoremas de DeMorgan 80
3-12 Universalidad de las compuertas NAND y NOR 83
3-13 Representaciones alternas de compuertas lógicas 86
3-14 Cuál representación de compuerta se debe usar 89
3-15 Símbolos lógicos del estándar IEEE/ANSI 95
3-16 Resumen de los métodos para describir circuitos lógicos 96
3-17 Comparación entre lenguajes de descripción y lenguajes de programación 98
3-18 Implementación de circuitos lógicos con PLDS 100
3-19 Formato y sintaxis del HDL 102
3-20 Señales intermedias 105

CAPÍTULO 4 Circuitos lógicos combinacionales 118

4-1 Forma de suma de productos 120
4-2 Simplificación de circuitos lógicos 121
4-3 Simplificación algebraica 121
4-4 Diseño de circuitos lógicos combinacionales 127
4-5 Método de mapas de Karnaugh 133
4-6 Circuitos OR exclusivo y NOR exclusivo 144
4-7 Generador y comprobador de paridad 149
4-8 Circuitos de habilitación/deshabilitación 151
4-9 Características básicas de los CIS digitales 153
4-10 Diagnóstico de fallas de sistemas digitales 160
4-11 Fallas internas en los circuitos integrados digitales 162
4-12 Fallas externas 166
4-13 Ejemplo práctico de diagnóstico de fallas 168
4-14 Dispositivos lógicos programables 170
4-15 Representación de datos en HDL 177
4-16 Tablas de verdad mediante el uso de HDL 181
4-17 Estructuras de control de decisiones en HDL 184

CAPÍTULO 5 Flip-flops y dispositivos relacionados 208

5-1 Latch de compuerta NAND 211
5-2 Latch de compuerta NOR 216
5-3 Ejemplo práctico de diagnóstico de fallas 219
5-4 Pulsos digitales 220
5-5 Señales de reloj y flip-flops sincronizados por reloj 221
5-6 Flip-flop sincronizado por reloj en S-R 224
5-7 Flip-flop sincronizado por reloj en J-K 227
5-8 Flip-flop sincronizado por reloj en D 230
5-9 Latch *D* (latch transparente) 232
5-10 Entradas asíncronas 233
5-11 Símbolos IEEE/ANSI 236
5-12 Consideraciones de sincronización de los flip-flops 238
5-13 Problema potencial de sincronización en circuitos con FF 241
5-14 Aplicaciones de los flip-flops 243
5-15 Sincronización de los flip-flops 243
5-16 Detección de una secuencia de entrada 244
5-17 Almacenamiento y transferencia de datos 245
5-18 Transferencia de datos en serie: registros de desplazamiento 247
5-19 División y conteo de frecuencia 250
5-20 Aplicación de microcomputadora 254
5-21 Dispositivos disparadores de Schmitt 256
5-22 One-shot (multivibrador monoestable) 256
5-23 Circuitos generadores de reloj 260
5-24 Diagnóstico de fallas en circuitos con flip-flops 264
5-25 Circuitos secuenciales mediante el uso de HDL 268
5-26 Dispositivos disparados por flanco 272
5-27 Circuitos de HDL con varios componentes 277

CAPÍTULO 6 Aritmética digital: operaciones y circuitos 296

6-1 Suma binaria 298
6-2 Representación de números con signo 299
6-3 Suma en el sistema de complemento a 2 306
6-4 Resta en el sistema de complemento a 2 307
6-5 Multiplicación de números binarios 310
6-6 División binaria 311
6-7 Suma BCD 312
6-8 Aritmética hexadecimal 314
6-9 Circuitos aritméticos 317
6-10 Sumador binario en paralelo 318
6-11 Diseño de un sumador completo 320

6-12 Sumador completo en paralelo con registros 323
6-13 Propagación del acarreo 325
6-14 Sumador en paralelo de circuito integrado 326
6-15 Sistema de complemento a 2 328
6-16 Circuitos integrados tipo ALU 331
6-17 Ejemplo práctico de diagnóstico de fallas 335
6-18 Uso de las funciones de la biblioteca TTL con Altera 337
6-19 Operaciones lógicas con arreglos de bits 338
6-20 Sumadores en HDL 340
6-21 Expansión de la capacidad de bits de un circuito 343

CAPÍTULO 7 Contadores y registros 360

7-1 Contadores asíncronos (de rizo) 362
7-2 Retraso de propagación en contadores de rizo 365
7-3 Contadores síncronos (en paralelo) 367
7-4 Contadores con números mod < 2^N 370
7-5 Contadores síncronos descendentes y ascendentes/descendentes 377
7-6 Contadores preajustables 379
7-7 Contadores síncronos de CI 380
7-8 Decodificación de un contador 389
7-9 Análisis de los contadores síncronos 393
7-10 Diseño de un contador síncrono 396
7-11 Contadores básicos mediante el uso de HDL 405
7-12 Contadores con todas las características en HDL 412
7-13 Cómo alambrar módulos de HDL en conjunto 417
7-14 Máquina de estados 425
7-15 Registros de circuito integrado 437
7-16 Entrada en paralelo/salida en paralelo: 74ALS174/74HC174 437
7-17 Entrada en serie/salida en serie: 74ALS166/74HC166 439
7-18 Entrada en paralelo/salida en serie: 74ALS165/74HC165 441
7-19 Entrada en serie/salida en paralelo: 74ALS164/74HC164 443
7-20 Contadores de registro de desplazamiento 445
7-21 Diagnóstico de fallas 450
7-22 Registros en HDL 452
7-23 Contadores de anillo en HDL 459
7-24 Monoestables en HDL 461

CAPÍTULO 8 Familias lógicas de circuitos integrados 488

8-1 Terminología de CIS digitales 490
8-2 La familia lógica TTL 498
8-3 Hojas técnicas TTL 502
8-4 Características de las series TTL 506

8-5 Capacidad de carga de la familia TTL 509
8-6 Otras características de la familia TTL 514
8-7 Tecnología MOS 518
8-8 Lógica de MOS complementario 521
8-9 Características de las series CMOS 523
8-10 Tecnología de bajo voltaje 530
8-11 Salidas de colector abierto/drenador abierto 533
8-12 Salidas lógicas triestado (tres estados) 538
8-13 Lógica de interfase de bus de alta velocidad 541
8-14 La familia ECL de CIS digitales 543
8-15 Compuerta de transmisión CMOS (interruptor bilateral) 546
8-16 Interfase de CIS 548
8-17 Interfase de voltaje mixto 553
8-18 Comparadores de voltaje analógico 554
8-19 Diagnóstico de fallas 556

CAPÍTULO 9 Circuitos lógicos MSI 576

9-1 Decodificadores 577
9-2 Decodificadores/controladores de BCD a 7 segmentos 584
9-3 Pantallas de cristal líquido 587
9-4 Codificadores 591
9-5 Diagnóstico de fallas 597
9-6 Multiplexores (selectores de datos) 599
9-7 Aplicaciones de los multiplexores 604
9-8 Demultiplexores (distribuidores de datos) 610
9-9 Más diagnóstico de fallas 617
9-10 Comparador de magnitud 621
9-11 Convertidores de código 624
9-12 Buses de datos 628
9-13 El registro triestado 74ALS173/HC173 629
9-14 Operación del bus de datos 632
9-15 Decodificadores mediante el uso de HDL 638
9-16 El decodificador/controlador de 7 segmentos en HDL 642
9-17 Codificadores mediante el uso de HDL 645
9-18 Multiplexores y demultiplexores en HDL 648
9-19 Comparadores de magnitud en HDL 652
9-20 Convertidores de código en HDL 653

CAPÍTULO 10 Proyectos de sistemas digitales mediante el uso de HDL 676

10-1 Administración de pequeños proyectos 678
10-2 Proyecto de controlador de motor de pasos 679
10-3 Proyecto de codificador de teclado numérico 687

10-4 Proyecto de reloj digital 693
10-5 Proyecto de contador de frecuencia 710

CAPÍTULO 11 Interface con el mundo analógico 718

11-1 Repaso de la comparación entre digital y analógico 719
11-2 Conversión digital-analógica (DAC) 721
11-3 Circuitos convertidores D/A 728
11-4 Especificaciones de un DAC 733
11-5 Un DAC de circuito integrado 735
11-6 Aplicaciones de los DACS 736
11-7 Diagnóstico de fallas en los DACS 738
11-8 Conversión analógica-digital (ADC) 739
11-9 ADC de rampa digital 740
11-10 Adquisición de datos 745
11-11 ADC de aproximaciones sucesivas 749
11-12 ADCS tipo flash 755
11-13 Otros métodos de conversión A/D 757
11-14 Circuitos de muestreo y retención 761
11-15 Multiplexaje 762
11-16 Osciloscopio de almacenamiento digital 764
11-17 Procesamiento digital de señales (DSP) 765

CAPÍTULO 12 Dispositivos de memoria 784

12-1 Terminología de memoria 786
12.2 Operación general de la memoria 790
12-3 Conexiones entre CPU y memoria 793
12-4 Memorias de sólo lectura 795
12-5 Arquitectura de la ROM 796
12-6 Sincronización de la ROM 799
12-7 Tipos de ROMS 800
12-8 Memoria flash 808
12-9 Aplicaciones de la ROM 811
12-10 RAM semiconductora 814
12-11 Arquitectura de la RAM 815
12-12 RAM estática (SRAM) 818
12-13 RAM dinámica (DRAM) 823
12-14 Estructura y operación de la RAM dinámica 824
12-15 Ciclos de lectura/escritura de la DRAM 829
12-16 Refresco de la DRAM 831
12-17 Tecnología de la DRAM 834
12-18 Expansión de tamaño de palabra y capacidad 836
12-19 Funciones especiales de la memoria 844

12-20 Diagnóstico de fallas en sistemas de RAM 847
12-21 Prueba de la ROM 852

CAPÍTULO 13 Arquitecturas de los dispositivos lógicos programables 868

13-1 Árbol familiar de los sistemas digitales 870
13-2 Fundamentos de los circuitos de un PLD 875
13-3 Arquitecturas de PLDS 877
13-4 El GAL 16V8 (matriz lógica genérica) 881
13-5 El CPLD EPM7128S de Altera 885
13-6 La familia FLEX10K de Altera 890
13-7 La familia Cyclone de Altera 894

Glosario 898

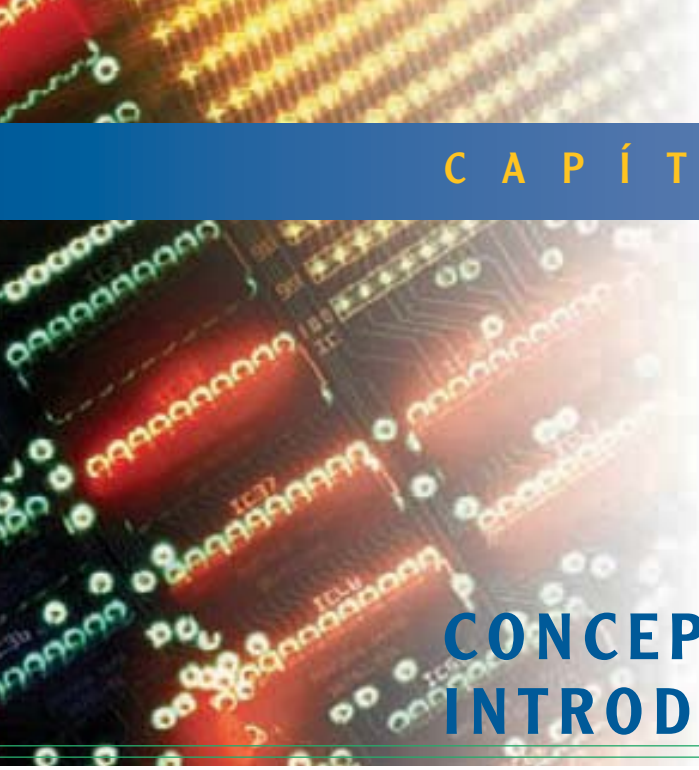
Respuestas a los problemas seleccionados 911

Índice de CIs 919

Índice 922

Sistemas digitales

Principios y aplicaciones



CAPÍTULO 1

CONCEPTOS INTRODUCTORIOS

CONTENIDO

1-1 Representaciones numéricas
1-2 Sistemas digitales y analógicos
1-3 Sistemas numéricos digitales
1-4 Representación de cantidades binarias
1-5 Circuitos digitales/circuitos lógicos
1-6 Transmisión en paralelo y en serie
1-7 Memoria
1-8 Computadoras digitales

OBJETIVOS

Al terminar este capítulo, usted podrá:

- Diferenciar entre la representación analógica y la digital.
- Citar las ventajas y desventajas de las técnicas digitales en comparación con las analógicas.
- Comprender la necesidad de los convertidores analógicos-digitales (ADC) y los convertidores digitales-analógicos (DAC).
- Reconocer las características básicas del sistema número binario.
- Convertir un número binario en su equivalente decimal.
- Contar en el sistema numérico binario.
- Identificar las señales digitales comunes.
- Identificar un diagrama de tiempos.
- Establecer las diferencias entre la transmisión en paralelo y la transmisión en serie.
- Describir las propiedades de las memorias.
- Describir las partes principales de una computadora digital y comprender sus funciones.
- Diferenciar entre microcomputadoras, microprocesadores y microcontroladores.

INTRODUCCIÓN

En el mundo actual, el término *digital* se ha convertido en parte de nuestro vocabulario común, debido a la dramática forma en que los circuitos y las técnicas digitales se han vuelto tan utilizados en casi todas las áreas de la vida: computadoras, automatización, robots, ciencia médica y tecnología, transporte, telecomunicaciones, entretenimiento, exploración en el espacio, etcétera. Usted está a punto de empezar un emocionante viaje educativo, en el cual descubrirá los principios fundamentales, conceptos y operaciones que son comunes para todos los sistemas digitales, desde el interruptor de encendido/apagado más simple hasta la computadora más compleja. Si el libro logra su cometido, usted deberá tener una comprensión detallada de la manera en que funcionan todos los sistemas digitales, y deberá ser capaz de aplicar este conocimiento en el análisis y la detección de fallas en cualquier sistema digital.

Primero presentaremos algunos conceptos subyacentes que forman una parte vital de la tecnología digital (estos conceptos se ampliarán a medida que se requieran más adelante en el libro). Además, presentaremos parte de la terminología que se necesita cuando uno inicia un nuevo campo de estudio, para agregarla a la lista de términos importantes de cada capítulo.

1-1 REPRESENTACIONES NUMÉRICAS

En los campos de la ciencia, la tecnología, los negocios y en la mayoría de los campos de trabajo, constantemente tratamos con *cantidades*. Estas cantidades se miden, se monitorean, se manipulan en forma aritmética, se observan, o de alguna otra forma se utilizan en los sistemas físicos. Es importante que al tratar con varias cantidades se puedan representar sus valores con eficiencia y precisión. Básicamente existen dos maneras de representar el valor numérico de las cantidades: **analógica** y **digital**.

Representaciones analógicas

En la **representación analógica** una cantidad se representa mediante un indicador proporcional que varía en forma continua. Un ejemplo es el velocímetro de los automóviles clásicos de las décadas de 1960 o 1970. La deflexión de la aguja es proporcional a la velocidad del automóvil y sigue cualquier cambio que se produzca a medida que el vehículo aumente o reduzca su velocidad. En los automóviles antiguos se utilizaba un eje mecánico flexible para conectar la transmisión con el velocímetro en el tablero. Es interesante observar que en automóviles recientes, por lo general, se prefiere la representación analógica, aun y cuando la velocidad ahora se mide en forma digital.

Los termómetros anteriores a la revolución digital utilizaban la representación analógica para medir la temperatura, y muchos de ellos todavía se usan actualmente. Esos termómetros utilizan una columna de mercurio, cuya altura es proporcional a la temperatura. Estos dispositivos están desapareciendo del mercado debido a problemas con el medio ambiente, pero sin duda son un excelente ejemplo de la representación analógica. Otro ejemplo es el termómetro de exteriores, en el cual la posición del apuntador gira alrededor de un disco a medida que un serpentín de metal se expande y se contrae con base en los cambios de temperatura. La posición del apuntador es proporcional a la temperatura. Sin importar qué tan pequeño sea el cambio en la temperatura, habrá un cambio proporcional en el indicador.

En estos dos ejemplos las cantidades físicas (velocidad y temperatura) se acoplan a un indicador a través de un medio mecánico solamente. En los sistemas analógicos eléctricos, la cantidad física que se mide o se procesa se convierte en un voltaje o corriente proporcional (señal eléctrica). Entonces el sistema utiliza este voltaje o corriente para fines de visualización, procesamiento o control.

El sonido es un ejemplo de una cantidad física que puede representarse mediante una señal analógica eléctrica. Un micrófono es un dispositivo que genera un voltaje de salida proporcional a la amplitud de las ondas sonoras que lo golpean. Las ondas sonoras producen variaciones en el voltaje de salida del micrófono. De esta manera, las grabaciones en cinta pueden almacenar ondas sonoras mediante el uso del voltaje de salida del micrófono para cambiar en forma proporcional el campo magnético en la cinta.

Las cantidades analógicas como las antes mencionadas tienen una importante característica: sin importar cómo se representen: *pueden variar a través de un intervalo continuo de valores*. La velocidad de un automóvil puede tener *cualquier* valor entre 0 y 100 km/hora, por ejemplo. De manera similar, la salida del micrófono podría tener cualquier valor dentro del intervalo de 0 a 10 mV (por ejemplo: 1 mV, 2.3724 mV, 9.9999 mV).

Representaciones digitales

En la **representación digital** las cantidades se representan no mediante indicadores que varían en forma continua, sino mediante símbolos llamados *dígitos*. Considere como ejemplo el reloj digital, que indica la hora del día en forma de dígitos decimales que representan horas y minutos (y algunas veces segundos). Como es sabido, la hora del día cambia en forma continua pero la lectura del reloj digital no cambia así, sino que cambia en intervalos de uno por minuto (o por segundo). En otras palabras, esta representación digital de la hora del día cambia en incrementos *discretos*,

en comparación con la representación de la hora que proporciona un reloj de pared operado mediante corriente alterna analógica, en donde la lectura de la carátula cambia en forma continua.

Así, la principal diferencia entre las cantidades analógicas y digitales puede plantearse de la siguiente manera:

analógica ≡ continua
digital ≡ discreta (paso por paso)

Debido a la naturaleza discreta de las representaciones digitales, no existe ambigüedad cuando se lee el valor de una cantidad digital, mientras que el valor de una cantidad analógica, por lo general, se deja abierto a la interpretación. En la práctica, cuando se mide una cantidad analógica, siempre se "redondea" a un nivel de precisión conveniente. En otras palabras, se digitaliza la cantidad. La representación digital es el resultado de asignar un número de precisión limitada a una cantidad que varía en forma continua. Por ejemplo, cuando usted toma su temperatura con un termómetro analógico, es común que la columna de mercurio se encuentre entre dos líneas de graduación, pero usted elije la línea más cercana y le asigna un número, por decir, 36.7 °C (98.6 °F).

EJEMPLO 1-1

¿Cuáles de las siguientes cantidades son analógicas y cuáles son digitales?

(a) Un interruptor de diez posiciones.
(b) La corriente que fluye a través de un contacto eléctrico.
(c) La temperatura de una habitación.
(d) Granos de arena en la playa.
(e) El medidor de combustible de un automóvil.

Solución

(a) Digital.
(b) Analógica.
(c) Analógica.
(d) Digital, ya que el número de granos sólo puede tener ciertos valores discretos (enteros) y no cualquier valor posible en un rango continuo.
(e) Analógico, en caso de ser del tipo con aguja; digital, si tiene indicador numérico o de barra de gráficos.

PREGUNTAS DE REPASO *

1. Describa en forma concisa la principal diferencia entre las cantidades analógicas y digitales.

1-2 SISTEMAS DIGITALES Y ANALÓGICOS

Un **sistema digital** es la combinación de dispositivos diseñados para manipular información lógica o cantidades físicas que se representan en forma digital; es decir, las cantidades sólo pueden tener valores discretos. Estos dispositivos, por lo general, son

* Las respuestas a las preguntas de repaso se encuentran al final del capítulo correspondiente.

electrónicos, pero también pueden ser mecánicos, magnéticos o neumáticos. Algunos de los sistemas digitales más comunes son las computadoras y las calculadoras digitales, los equipos de audio y video digital, y el sistema telefónico (el sistema digital más grande del mundo).

Un **sistema analógico** contiene dispositivos que manipulan cantidades físicas que se representan en forma analógica. En un sistema analógico, las cantidades pueden variar sobre un intervalo continuo de valores. Por ejemplo, la amplitud de la señal de salida a la bocina en un receptor de radio puede tener cualquier valor entre cero y su límite máximo. Otros sistemas analógicos comunes son los amplificadores de audio, los equipos de grabación y reproducción de cintas magnéticas, y un simple interruptor regulador de luz.

Ventajas de las técnicas digitales

Cada vez hay más aplicaciones en la electrónica, al igual que en la mayoría de las otras tecnologías, que utilizan técnicas digitales para hacer operaciones que antes se realizaban mediante métodos analógicos. Las razones principales del cambio hacia la tecnología digital son:

1. *Generalmente los sistemas digitales son más fáciles de diseñar*. Los circuitos que se utilizan en los sistemas digitales son *circuitos de conmutación*, en donde no importan los valores *exactos* de voltaje o de corriente, sino sólo el intervalo (**ALTO** o **BAJO**) en el que se encuentren.
2. *Es fácil almacenar información*. Esto se logra mediante dispositivos y circuitos especiales que pueden fijar la información digital y almacenarla durante el tiempo que sea necesario, y las técnicas de almacenamiento masivo pueden guardar miles de millones de bits de información en un espacio físico relativamente pequeño. En contraste, la capacidad de almacenamiento de las técnicas analógicas es extremadamente limitada.
3. *Es más fácil mantener la precisión y la exactitud en todo el sistema*. Una vez que se digitaliza una señal, la información que contiene no se deteriora a medida que se procesa. En los sistemas analógicos, las señales de voltaje y de corriente tienden a distorsionarse debido a los efectos de temperatura, humedad y por variaciones de tolerancia de los componentes en los circuitos que procesan la señal.
4. *La operación puede programarse*. Es bastante sencillo diseñar sistemas digitales cuya operación esté controlada por un conjunto de instrucciones almacenadas, a lo cual se le conoce como *programa*. Los sistemas analógicos también pueden *programarse*, pero la variedad y la complejidad de las operaciones disponibles son muy limitadas.
5. *Los circuitos digitales son más resistentes al ruido*. Las fluctuaciones espurias en el voltaje (ruido) no son tan críticas en los sistemas digitales, ya que el valor exacto del voltaje no es importante, siempre y cuando el ruido no sea tan fuerte como para evitar que podamos distinguir entre un nivel ALTO y un nivel BAJO.
6. *Pueden fabricarse más circuitos digitales en los chips de CI*. Es cierto que los circuitos analógicos también se han beneficiado con el desarrollo de la tecnología de los circuitos integrados, pero su complejidad relativa y uso de dispositivos que no pueden integrarse de manera económicamente conveniente (capacitores de alto valor, resistencias de precisión, inductancias, transformadores) han hecho imposible alcanzar el mismo grado de integración en los sistemas analógicos.

Limitaciones de las técnicas digitales

En realidad existen muy pocas desventajas al utilizar técnicas digitales. Los dos problemas principales son:

El mundo real es analógico.
El procesamiento de las señales digitales lleva tiempo.

La mayoría de las cantidades físicas son analógicas por naturaleza, y a menudo estas cantidades son las entradas y salidas que un sistema monitorea, opera o controla. Algunos ejemplos son: temperatura, presión, posición, velocidad, nivel de líquido y flujo, entre otros. Estamos habituados a expresar estas cantidades en forma *digital*, como cuando decimos que la temperatura es de 32° (31.8° si queremos ser más precisos), pero en realidad estamos haciendo una aproximación digital de una cantidad que es sin duda analógica.

Para aprovechar las técnicas digitales al procesar entradas y salidas analógicas, se deben seguir cuatro pasos:

1. Convertir la variable física en una señal eléctrica (analógica).
2. Convertir la señal eléctrica (analógica) a su forma digital.
3. Procesar (operar con) la información digital.
4. Convertir las salidas digitales nuevamente a la forma analógica del mundo real.

Se podría escribir todo un libro únicamente para explicar el paso 1. Existen muchos tipos de dispositivos que convierten diversas variables físicas en señales eléctricas analógicas (sensores). Éstos se utilizan para medir cosas que se encuentran en nuestro mundo analógico "real". Tan sólo en un automóvil hay sensores de nivel de líquido (tanque de gasolina), temperatura (control del clima y del motor), velocidad (velocímetro), aceleración (detección de colisión con bolsa de aire), presión (aceite, múltiple de admisión) y flujo (combustible), por mencionar unos cuantos.

Para ilustrar un sistema típico que utiliza este enfoque, la figura 1-1 describe un sistema de precisión para regulación de temperatura. Un usuario oprime botones para aumentar o reducir la temperatura en incrementos de 0.1° (representación digital). Un sensor de temperatura en el espacio que se está calentando convierte la temperatura medida en un voltaje proporcional. Este voltaje analógico se convierte en una cantidad digital mediante un **convertidor analógico-digital (ADC)**. Después, este valor se compara con el valor deseado y se utiliza para determinar un valor digital que representa cuánto calor se necesita. El valor digital se convierte en una cantidad analógica (voltaje) mediante un **convertidor digital-analógico (DAC)**. Este voltaje se aplica a un elemento de calentamiento, el cual producirá el calor correspondiente al voltaje aplicado y afectará la temperatura del espacio.

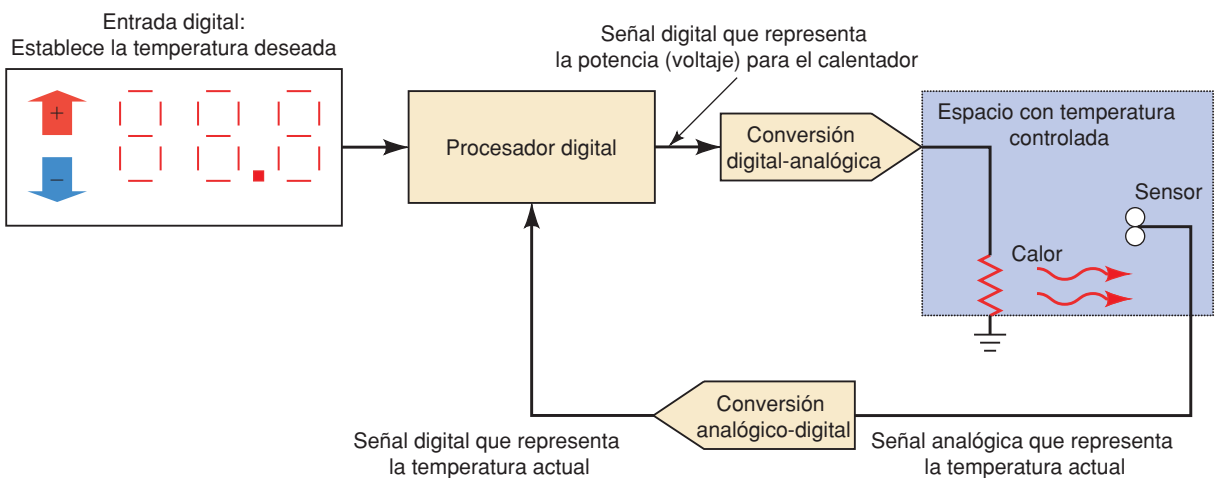


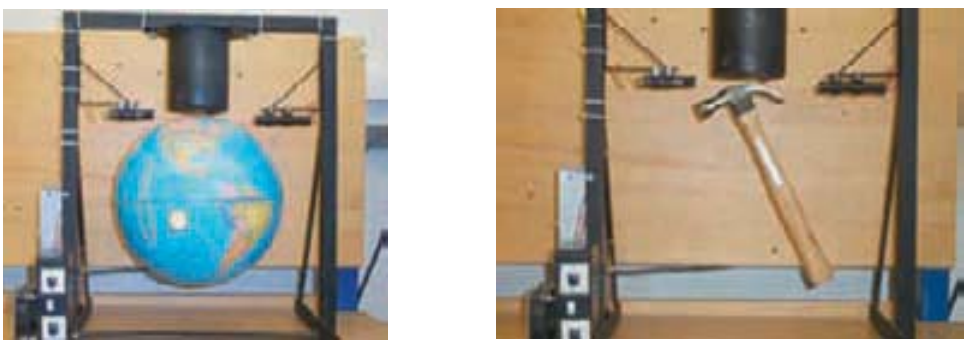
FIGURA 1-1 Diagrama de bloques de un sistema digital de precisión para control de temperatura.

Otro buen ejemplo en donde se lleva a cabo la conversión entre valores analógicos y digitales es en la grabación de audio. Los discos compactos (CD) han sustituido a las cintas magnéticas debido a que proporcionan un medio mucho más conveniente

para la grabación y reproducción de música. El proceso funciona más o menos así: (1) los sonidos provenientes de los instrumentos y de las voces humanas producen una señal analógica de voltaje en un micrófono; (2) esta señal analógica se convierte en un formato digital mediante el uso de un proceso de conversión analógico-digital; (3) la información digital se almacena en la superficie del CD; (4) durante la reproducción, el reproductor de CD toma la información digital de la superficie del CD y la convierte en una señal analógica, que a su vez se amplifica y se alimenta a una bocina, en donde puede ser detectada por el oído humano.

La segunda desventaja de los sistemas digitales es que se requiere tiempo para procesar las señales digitalizadas (listas de números). Además es necesario realizar conversiones entre las formas analógica y digital de la información, lo cual hace que el sistema sea más complejo y costoso. Entre más precisos necesiten ser los números, más tiempo se requerirá para procesarlos. En muchas aplicaciones, estos aspectos se minimizan con las numerosas ventajas del uso de las técnicas digitales, por lo cual la conversión entre cantidades analógicas y digitales se ha vuelto bastante común en la tecnología actual.

No obstante, existen situaciones en las que es más sencillo o económico utilizar técnicas analógicas. Por ejemplo, hace varios años un colega (Tom Robertson) decidió crear una demostración de un sistema de control para los grupos de visitantes. Planeaba suspender un objeto metálico en un campo magnético, como se muestra en la figura 1-2. Para implementar un electroimán enredó alambre en una bobina, al tiempo que controlaba la cantidad de corriente que pasaba a través de ésta. Para medir la posición del objeto metálico pasaba un rayo de luz infrarroja a través del campo magnético. A medida que el objeto se acercaba al electroimán, empezaba a bloquear el rayo de luz. Para controlar el campo magnético se medían los pequeños cambios en el nivel de luz, logrando así mantener el objeto metálico flotando y estacionario, sin usar cordones. Todos los intentos por utilizar una microcomputadora para medir estos cambios tan pequeños, realizar los cálculos de control y excitar el electroimán resultaban demasiado lentos, aún y cuando se utilizaba la PC más rápida y potente disponible en ese entonces. En la solución final sólo utilizó un par de amplificadores operacionales y otros componentes que costaron sólo unos cuantos dólares: un método completamente analógico. Hoy en día tenemos acceso a procesadores que son lo suficientemente veloces y a técnicas de medición lo bastante precisas como para lograr esta hazaña, pero la solución más simple sigue siendo analógica.



(a) (b)

FIGURA 1-2 Un sistema de levitación magnético suspendiendo: (a) un globo terráqueo con una placa de acero incrustada, y (b) un martillo.

Es común ver que se emplean técnicas tanto analógicas como digitales dentro del mismo sistema para beneficiarse de las ventajas de ambas. En estos sistemas *híbridos*, una de las partes más importantes de la fase de diseño implica determinar cuáles partes del sistema deben ser analógicas y cuáles digitales. La tendencia en

la mayoría de los sistemas es digitalizar la señal lo más pronto posible y convertirla de nuevo en analógica lo más tarde posible, a medida que las señales fluyen a través del sistema.

El futuro es digital

Los avances en la tecnología digital durante las últimas tres décadas han sido fenomenales, y hay razones fuertes para creer que todavía hay más por venir. Piense en los artículos de uso diario que han cambiado del formato analógico al digital en el transcurso de su vida. Ahora puede comprar un termómetro digital inalámbrico para interiores/exteriores por menos de 10.00 dólares. Los automóviles han pasado de tener muy pocos controles electrónicos a ser vehículos controlados en su mayor parte por la tecnología digital. El audio digital nos ha llevado a usar el disco compacto y el reproductor MP3. El video digital trajo consigo al DVD. Las cámaras de video y fotográficas digitales para el hogar; la grabación digital con sistemas como TiVo; los teléfonos celulares digitales y el tratamiento digital de imágenes en los rayos X; el tratamiento de imágenes de resonancia magnética (MRI) y los sistemas de ultrasonido en los hospitales son sólo unas cuantas de las aplicaciones que han sido invadidas por la revolución digital. Tan pronto como se cuente con la infraestructura apropiada, los sistemas de teléfono y televisión se harán digitales. La tasa de crecimiento en el ámbito digital continúa sorprendiendo. Tal vez su automóvil esté equipado con un sistema tal como On Star de GM, el cual convierte el tablero en un centro de comunicación, información y navegación inalámbrica. Tal vez ya esté utilizando comandos de voz para enviar o recibir correo electrónico, pedir un reporte de tráfico, revisar las necesidades de mantenimiento del automóvil o sólo cambiar de estación de radio o de CD; todo sin necesidad de quitar sus manos del volante o su vista del camino. Los automóviles pueden reportar su posición exacta en caso de emergencia o de una falla mecánica. En los años por venir la comunicación inalámbrica seguirá expandiendo su cobertura para ofrecerle conectividad en donde quiera que se encuentre. Los teléfonos podrán recibir, ordenar y tal vez responder a las llamadas entrantes, al igual que una secretaria altamente capacitada. La revolución de la televisión digital le proveerá no sólo de una mayor definición de la imagen, sino también mucha más flexibilidad en la programación. Usted podrá seleccionar el programa que desea ver y cargarlo en la memoria de su televisor, con lo cual podrá congelar la imagen o reproducir escenas a placer, como si estuviera viendo un DVD. A medida que la realidad virtual continúe mejorando, usted podrá interactuar con la materia que esté estudiando. Tal vez esto no suene tan emocionante cuando se estudia electrónica, pero imagine estudiar la historia como si fuera un participante, o aprender las técnicas apropiadas para todo lo relacionado con temas que vayan desde el atletismo hasta la cirugía, por medio de simulaciones basadas en su desempeño actual.

La tecnología digital continuará su incursión de alta velocidad en las actividades diarias de nuestras vidas, además de que abrirá nuevas fronteras en formas que tal vez nunca hayamos imaginado. Estas aplicaciones (y muchas más) se basan en los principios que presentamos en este libro. Las herramientas de software para desarrollar sistemas complejos se actualizan constantemente y están disponibles para cualquier persona a través de la Web. También estudiaremos los fundamentos técnicos necesarios para comunicarnos con cualquiera de estas herramientas y lo prepararemos para que pueda disfrutar de una carrera fascinante y llena de satisfacciones.

PREGUNTAS DE REPASO

1. ¿Cuáles son las ventajas de las técnicas digitales, en comparación con las analógicas?
2. ¿Cuál es la principal limitación en cuanto al uso de las técnicas digitales?

1-3 SISTEMAS NUMÉRICOS DIGITALES

En la tecnología digital se utilizan muchos sistemas numéricos. Los más comunes son los siguientes: decimal, binario, octal y hexadecimal. Evidentemente el sistema decimal es el más conocido, ya que es el que utilizamos a diario. Analicemos algunas de sus características para ayudarnos a comprender los demás sistemas numéricos.

Sistema decimal

El **sistema decimal** está compuesto de *10* números o símbolos. Estos 10 símbolos son: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9; al utilizar estos símbolos como *dígitos* de un número, podemos expresar cualquier cantidad. El sistema decimal se conoce también como sistema de *base 10* ya que tiene 10 dígitos, y ha evolucionado en forma natural debido al hecho de que las personas tenemos 10 dedos. De hecho, la palabra *dígito* se deriva de la palabra "dedo".

El decimal es un *sistema de valor posicional*, en el cual el valor de un dígito depende de su posición. Por ejemplo, considere el número decimal 453. Por la posición de los dígitos sabemos que el 4 en realidad representa 4 *centenas*, el 5 representa 5 *decenas* y el 3 representa 3 *unidades*. En esencia, el 4 lleva el mayor peso de los tres dígitos y se le conoce como el *dígito más significativo (MSD)* mientras que el 3 lleva el menor peso y se le conoce como el *dígito menos significativo (LSD)*.

Considere otro ejemplo, 27.35. Este número en realidad es igual a 2 decenas más 7 unidades más 3 décimas más 5 centésimas, o bien $2 \times 10 + 7 \times 1 + 3 \times 0.1 + 5 \times 0.01$. El punto decimal se utiliza para separar las partes entera y fraccional del número.

Dicho de forma más rigurosa, las diversas posiciones relativas al punto decimal llevan pesos que pueden expresarse como potencias de 10. Esto se ilustra en la figura 1-3, en donde se representa el número 2745.214. El punto decimal separa las potencias positivas de 10 de las potencias negativas. El número 2745.214 es, por lo tanto, igual a:

$$(2 \times 10^{+3}) + (7 \times 10^{+2}) + (4 \times 10^{1}) + (5 \times 10^{0}) + (2 \times 10^{-1}) + (1 \times 10^{-2}) + (4 \times 10^{-3})$$

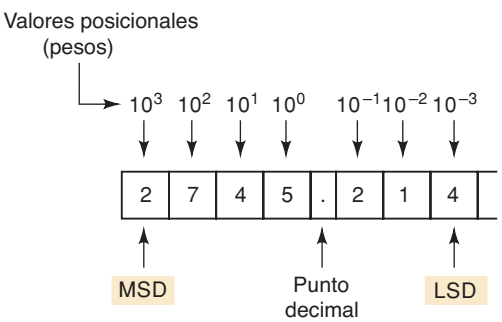


FIGURA 1-3 Valores de posición decimal como potencias de 10.

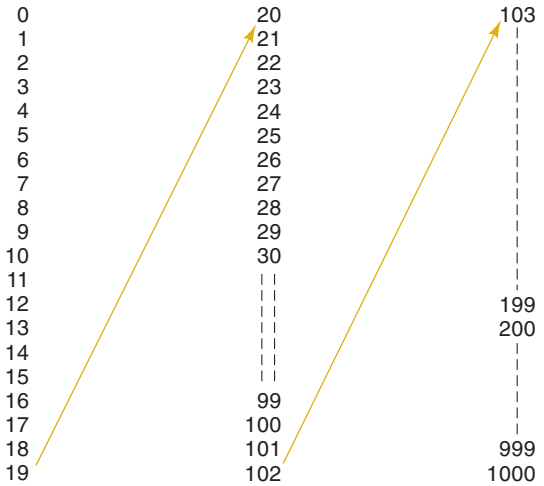
En general, cualquier número es sólo la suma de los productos del valor de cada dígito y su valor posicional.

Conteo decimal

Al contar en el sistema decimal, empezamos con 0 en la posición de las unidades y tomamos cada símbolo (dígito) en forma progresiva hasta llegar al 9. Luego agregamos un 1 a la siguiente posición más alta y empezamos de nuevo con 0 en la prime-

ra posición (vea la figura 1-4). Este proceso continúa hasta llegar al conteo de 99. Después agregamos un 1 a la tercera posición y continuamos de nuevo con 0s en las primeras dos posiciones. El mismo patrón se sigue hasta llegar al número que deseamos contar.

FIGURA 1-4 Conteo decimal.



Es importante observar que en el conteo decimal, la posición de las unidades (LSD) cambia hacia arriba con cada paso en el conteo, la posición de las decenas cambia hacia arriba cada 10 pasos en el conteo, la posición de las centenas cambia hacia arriba cada 100 pasos en el conteo, y así sucesivamente.

Otra característica del sistema decimal es que si utilizamos sólo dos lugares decimales, podemos contar hasta $10^2 = 100$ números distintos (0 a 99).* Con tres lugares podemos contar hasta $10^3 = 1000$ números (0 a 999), y así sucesivamente. En general, con N lugares o dígitos podemos contar hasta 10^N números distintos. Empezando con cero, el número más grande siempre será $10^N - 1$.

Sistema binario

Desafortunadamente, el sistema numérico decimal no se presta para una implementación conveniente en los sistemas digitales. Por ejemplo, es muy difícil diseñar equipo electrónico de manera que pueda trabajar con 10 niveles de voltaje distintos (cada uno representando un carácter decimal, del 0 al 9). Por otro lado, es muy sencillo diseñar circuitos electrónicos simples y precisos que operen sólo con dos niveles de voltaje. Por esta razón casi cualquier sistema digital utiliza el sistema numérico binario (base 2) como el sistema numérico básico de sus operaciones. Frecuentemente se utilizan otros sistemas numéricos para interpretar o representar cantidades binarias, para ayudar a las personas que trabajan con estos sistemas digitales y los utilizan.

En el **sistema binario** sólo hay dos símbolos o posibles valores de dígitos: 0 y 1. Aún así, este sistema de base 2 puede usarse para representar cualquier cantidad que pueda representarse en decimal o en otros sistemas numéricos. Sin embargo, se requeriría de un mayor número de dígitos binarios para expresar una determinada cantidad.

* El cero se cuenta como número.

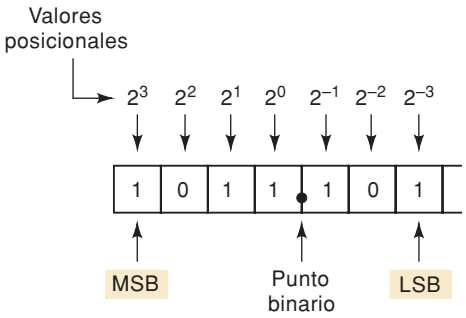


FIGURA 1-5 Los valores posicionales binarios como potencias de 2.

Todas las aseveraciones mencionadas con respecto al sistema decimal pueden aplicarse de igual forma al sistema binario. Este sistema también es de valor posicional, en donde cada dígito binario tiene su propio peso expresado como potencia de 2. Esto se ilustra en la figura 1-5, donde los lugares a la izquierda del *punto binario* (contraparte del punto decimal) son potencias positivas de 2, y los lugares a la derecha son potencias negativas de 2. El número 1101.101 se representa en la figura. Para encontrar su equivalente en el sistema decimal, sólo tomamos la suma de los productos del valor de cada dígito (0 o 1) y su valor posicional.

$$\begin{aligned} 1011.101_2 &= (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) \\ &\quad + (1 \times 2^{-1}) + (0 \times 2^{-2}) + (1 \times 2^{-3}) \\ &= 8 + 0 + 2 + 1 + 0.5 + 0 + 0.125 \\ &= 11.625_{10} \end{aligned}$$

En la operación anterior podemos observar que se utilizaron subíndices (2 y 10) para indicar la base en la cual se expresa el número. Esta convención evita la confusión que se crea siempre que se emplea más de un sistema numérico.

En el sistema binario es muy común que el término *dígito binario* se abrevie con el término **bit**, el cual utilizaremos a partir de ahora. Por ende, en el número expresado en la figura 1-5 hay cuatro bits a la izquierda del punto binario, los cuales representan la parte entera del número, y tres bits a la derecha del punto binario, los cuales representan la parte fraccional. El bit más significativo (MSB) es el bit más a la izquierda (de mayor peso). El bit menos significativo (LSB) es el bit más a la derecha (de menor peso). Éstos se indican en la figura 1-5. Aquí, el MSB tiene un peso de 2^3 y el LSB tiene un peso de 2^{-3}.

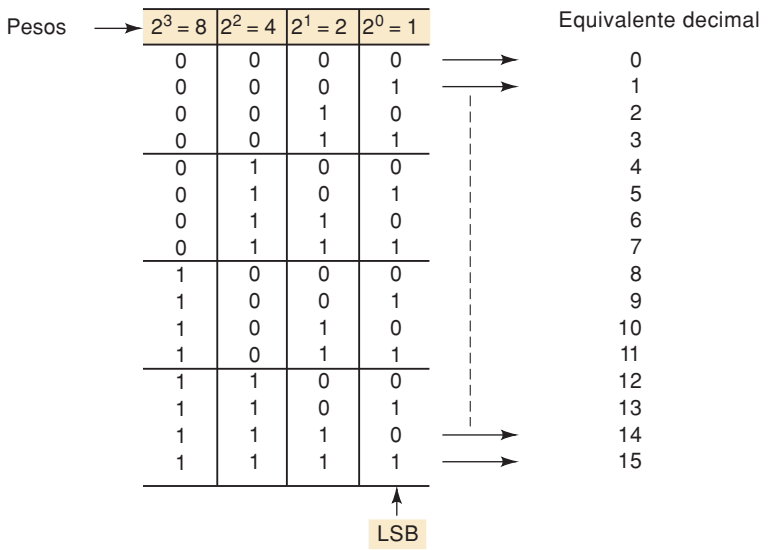
Conteo binario

Al tratar con números binarios, por lo general, nos restringimos a un número específico de bits. Esta restricción se basa en los circuitos que utilicemos para representar los números binarios. Utilicemos números binarios de cuatro bits para ilustrar el método para contar en binario.

La secuencia que se muestra en la figura 1-6, comienza con todos los bits en 0; a esto se le conoce como *cuenta cero*. Para cada cuenta sucesiva se *alterna* la posición de las unidades (2^0); es decir, cambia de un valor binario al otro. Cada vez que el bit de las unidades cambie de 1 a 0 se alternará la posición de los dos (2^1), es decir, cambiará de estado. Cada vez que la posición de los dos cambie de 1 a 0, se alternará la posición de los cuatros (2^2) (cambiará de estado). De igual forma, cada vez que la posición de los cuatros pase de 1 a 0, se alternará la posición de los ochos (2^3). Si el número binario tuviera más de cuatro bits, este mismo proceso continuaría para las posiciones de los bits de mayor orden.

La secuencia de conteo binario tiene una característica importante, como se muestra en la figura 1-6. El bit de las unidades (LSB) cambia ya sea de 0 a 1 o de 1 a 0 con *cada* conteo. El segundo bit (posición de los dos) permanece en 0 durante dos conteos, después en 1 durante dos conteos, después en 0 durante dos conteos, y así sucesivamente. El tercer bit (posición de los cuatros) permanece en 0 durante

FIGURA 1-6 Secuencia de conteo binario.



cuatro conteos, después en 1 durante cuatro conteos, y así sucesivamente. El cuarto bit (posición de los ochos) permanece en 0 durante ocho conteos, después en 1 durante ocho conteos. Si quisiéramos contar aún más, agregaríamos más lugares y este patrón continuaría con los 1s y los 0s alternando en grupos de 2^{N-1}. Por ejemplo, si utilizamos un quinto lugar binario, el quinto bit alternaría dieciséis 0s, después dieciséis 1s, y así en lo sucesivo.

Al igual que para el sistema decimal, para el sistema binario también aplica que si se utilizan N bits o lugares, podemos realizar 2^N conteos. Por ejemplo, con dos bits podemos realizar $2^2 = 4$ conteos (de 00_2 hasta 11_2); con cuatro bits podemos realizar $2^4 = 16$ conteos (de 0000_2 hasta 1111_2); y así sucesivamente. En la última cuenta todos los bits serán 1s y será igual a 2^N–1 en el sistema decimal. Por ejemplo, si se utilizan cuatro bits, el último número será $1111_2 = 2^4 - 1 = 15_{10}$.

EJEMPLO 1-2

¿Cuál es el mayor número que se puede representar si se utilizan ocho bits?

Solución

$2^N - 1 = 2^8 - 1 = 255_{10} = 11111111_2$.

Ésta fue una breve introducción al sistema numérico binario y su relación con el sistema decimal. En el siguiente capítulo invertiremos mucho más tiempo en estos dos sistemas numéricos y otros más.

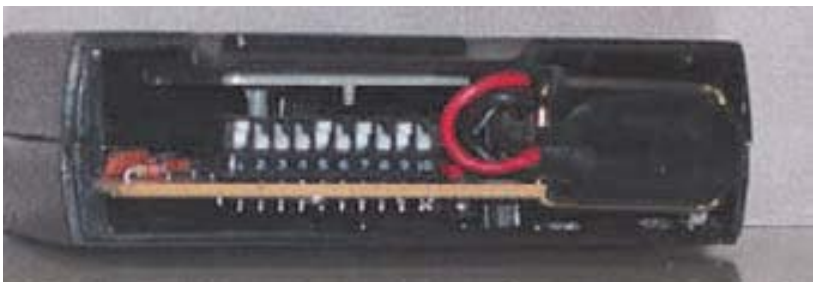
PREGUNTAS DE REPASO

1. ¿Cuál es el equivalente decimal de 1101011_2?
2. ¿Cuál es el binario que sigue al 10111_2 en la secuencia de conteo?
3. ¿Cuál es el mayor valor decimal que se puede representar si se utilizan 12 bits?

1-4 REPRESENTACIÓN DE CANTIDADES BINARIAS

En los sistemas digitales, la información que se procesa, por lo general, está presente en formato binario. Las cantidades binarias pueden representarse mediante cualquier dispositivo que sólo tenga dos estados de operación o condiciones posibles.

Por ejemplo, un interruptor sólo tiene dos estados: abierto o cerrado. De manera arbitraria podemos permitir que un interruptor abierto represente el 0 binario y que un interruptor cerrado represente el 1 binario. Con esta asignación podemos ahora representar cualquier número binario. La figura 1-7(a) muestra un número en código binario para un dispositivo de apertura de puertas de garaje. Los pequeños interruptores están ajustados para formar el número binario 1000101010. La puerta se abrirá sólo si coinciden los patrones de bits en el receptor y en el transmisor .



(a)



(b)

FIGURA 1-7
(a) Opciones de configuración binaria para un dispositivo de apertura de puertas de garaje. (b) Audio digital en un CD.

La figura 1-7(b) muestra otro ejemplo, en el cual se almacenan números binarios en un CD. La superficie interior (debajo de una capa de plástico transparente) se recubre con una capa de aluminio con alta capacidad de reflexión. Se queman hoyos a través de esta cubierta reflectora para formar "pozos" que no reflejan la luz de la misma manera que las áreas no quemadas. Estas áreas en las que se queman los pozos se consideran como "1s" y las áreas reflectoras son "0s".

Existen muchos otros dispositivos que sólo tienen dos estados de operación, o que pueden operarse en dos condiciones extremas. Entre ellos están: la bombilla de luz (brillante u oscura), el diodo (conductor o no conductor), el electroimán (energizado o desenergizado), el transistor (en corte o saturado), la fotocelda (iluminada u oscura), el termostato (abierto o cerrado), el embrague mecánico (enganchado o desenganchado), y un área en un disco magnético (magnetizada o desmagnetizada).

En los sistemas digitales electrónicos la información binaria se representa mediante voltajes (o corrientes) que están presentes en las entradas y salidas de los diversos circuitos. Por lo general, el 0 y el 1 binarios se representan mediante dos niveles de voltaje nominal. Por ejemplo, cero Volts (0 V) podrían representar el 0 binario y +5 V podrían representar el 1 binario. En realidad y debido a las variaciones en los circuitos, el 0 y el 1 se representan mediante intervalos de voltaje. Esto se ilustra en la figura 1-8(a), en donde cualquier voltaje entre 0 y 0.8 V representa un 0 y cualquier voltaje entre 2 y 5 V representa un 1. Por lo general, todas las señales de entrada y salida se encuentran dentro de alguno de estos intervalos, excepto durante las transiciones de un nivel a otro.

Ahora podemos notar otra diferencia importante entre los sistemas digitales y los analógicos. En los sistemas digitales, el valor exacto de un voltaje *no* es impor-

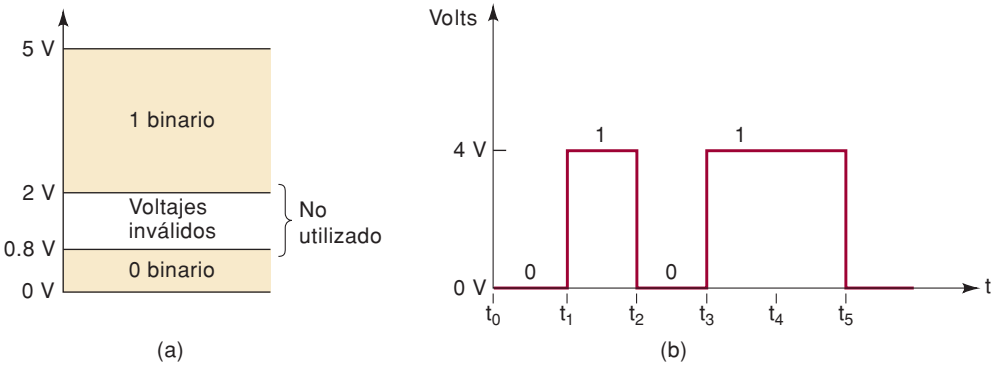


FIGURA 1-8 (a) Asignaciones comunes de voltaje en un sistema digital; (b) diagrama común de sincronización de una señal digital.

tante; por ejemplo, para las asignaciones de voltaje de la figura 1-8(a) un voltaje de 3.6 V significa lo mismo que un voltaje de 4.3 V. En los sistemas analógicos el valor exacto de un voltaje *es* importante. Por ejemplo, si el voltaje analógico es proporcional a la temperatura medida por un transductor, 3.6 V representaría una temperatura distinta que 4.3 V. En otras palabras, la magnitud del voltaje lleva información importante. Dicha característica significa que, por lo general, es más difícil diseñar circuitos analógicos precisos que circuitos digitales, debido a la manera en la que se ven afectados los valores exactos de voltaje por las variaciones en los valores de los componentes, la temperatura y el ruido (fluctuaciones aleatorias de voltaje).

Las señales digitales y los diagramas de tiempos

La figura 1-8(b) muestra una señal digital común y la forma en que ésta varía a través del tiempo. En realidad es un gráfico de voltaje contra tiempo (t) y se le conoce como **diagrama de tiempos**. La escala de tiempo horizontal está graduada en intervalos regulares que comienzan desde t_0 y avanzan hasta t_1, t_2 y así sucesivamente. Para el ejemplo del diagrama de tiempos que se muestra aquí, la señal empieza en 0 V (un 0 binario) en el tiempo t_0 y permanece ahí hasta el tiempo t_1. En t_1 la señal realiza una transición (salto) hasta 4 V (un 1 binario). En t_2 regresa a 0 V. En t_3 y t_5 ocurren transiciones similares. Observe que la señal no cambia en t_4, sino que permanece en 4 V desde t_3 hasta t_5.

Las transiciones en este diagrama de tiempos se dibujan como líneas verticales, por lo que parecen ser instantáneas cuando en realidad no lo son. Sin embargo, en muchas situaciones los tiempos de transición son tan cortos en comparación con los tiempos entre transiciones que podemos mostrarlos en el diagrama como líneas verticales. Más adelante nos encontraremos con situaciones en las que será necesario mostrar las transiciones con más precisión, en una escala de tiempo a mayor detalle.

Los diagramas de tiempos se utilizan mucho para mostrar cómo cambian las señales digitales con el tiempo, y en especial para mostrar la relación entre dos o más señales digitales en el mismo circuito o sistema. Al mostrar una o más señales digitales en un *osciloscopio* o *analizador lógico* podemos comparar las señales con sus diagramas de tiempos esperados. Ésta es una parte muy importante de los procedimientos de prueba y detección de fallas que se utilizan en los sistemas digitales.

1.5 CIRCUITOS DIGITALES/CIRCUITOS LÓGICOS

Los **circuitos digitales** están diseñados para producir voltajes de salida que se encuentran dentro de los intervalos de voltaje prescritos para 0 y 1, como los que se definen en la figura 1-8. De igual forma, los circuitos digitales están diseñados para responder en forma predecible a los voltajes de entrada que se encuentran

dentro de los intervalos definidos de 0 y 1. Esto significa que un circuito digital responderá de igual forma a todos los voltajes de entrada que se encuentren dentro de los valores permitidos para 0; de manera similar, no habrá distinción entre los voltajes de entrada que se encuentren dentro del intervalo permitido para 1.

La figura 1-9 representa un circuito digital típico con una entrada v_i y una salida v_o. La salida se muestra para dos formas de onda de señal de entrada distintas. Observe que v_o es igual para ambos casos, pues aunque las dos formas de onda de entrada difieren en sus niveles exactos de voltaje, se encuentran en los mismos niveles binarios.

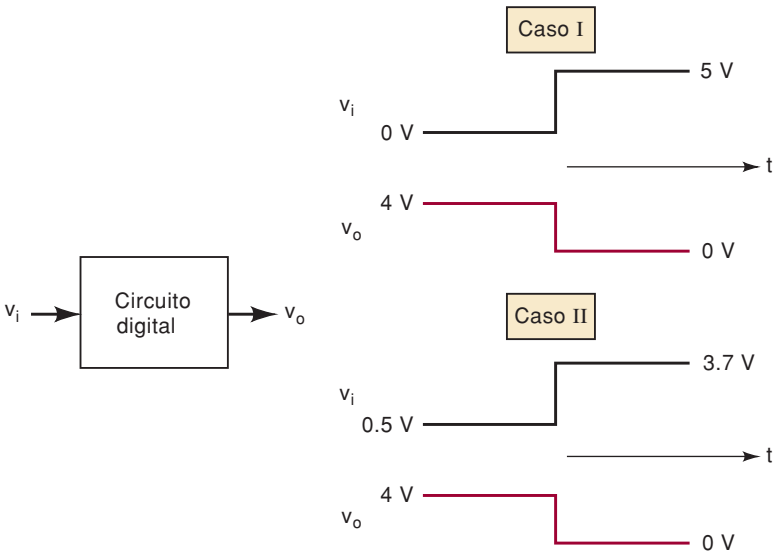


FIGURA 1-9 Un circuito digital responde a un nivel binario de entrada (0 o 1) y no a su voltaje real.

Circuitos lógicos

La forma en que un circuito digital responde a una entrada se conoce como *lógica* del circuito. Cada tipo de circuito digital obedece un cierto conjunto de reglas lógicas. Por esta razón a los circuitos digitales se les conoce también como **circuitos lógicos**. A lo largo del libro utilizaremos ambos términos sin distinción. En el capítulo 3 veremos con mayor claridad el significado del término "lógica" de un circuito.

Vamos a estudiar todos los tipos de circuitos lógicos que se utilizan en la actualidad en los sistemas digitales. Primero enfocaremos nuestra atención sólo en la operación lógica que realizan estos circuitos; es decir, la relación entre las entradas y las salidas del circuito. Dejaremos pendiente cualquier discusión acerca de la operación interna de los circuitos lógicos hasta que hayamos entendido completamente su operación lógica.

Circuitos digitales integrados

Casi todos los circuitos digitales que se utilizan en los sistemas digitales modernos son circuitos integrados (CI). La amplia variedad de circuitos integrados lógicos disponibles, ha hecho posible la construcción de sistemas digitales complejos que son más pequeños y confiables que sus contrapartes fabricados con componentes discretos.

Existen varias tecnologías de fabricación de circuitos integrados utilizadas para producir circuitos integrados digitales, de las cuales las más comunes son CMOS, TTL, NMOS y ECL. Cada una difiere en cuanto al tipo de circuito utilizado para proporcionar la operación lógica deseada. Por ejemplo, TTL (lógica de transistor-transistor) utiliza el transistor bipolar como el elemento principal en el circuito, mientras que CMOS (semiconductor de metal óxido complementario) utiliza el MOSFET en modo mejorado como el elemento principal del circuito. Aprenderemos sobre las diversas tecnologías de CIs, sus características, ventajas y desventajas a medida que vayamos dominando los tipos básicos de circuitos lógicos.

PREGUNTAS DE REPASO

1. *Verdadero* o *falso:* El valor exacto de un voltaje de entrada es imprescindible para un circuito digital.
2. ¿Puede un circuito digital producir el mismo voltaje de salida para distintos valores del voltaje de entrada?
3. Un circuito digital también se conoce como circuito _____.
4. Un gráfico que muestra cómo cambian una o más señales digitales a través del tiempo se llama _____.

1-6 TRANSMISIÓN EN PARALELO Y EN SERIE

Una de las operaciones más comunes que ocurre en cualquier sistema digital es la transmisión de información de un lugar a otro. La información puede transmitirse a través de una distancia tan pequeña como una fracción de un centímetro en la misma tarjeta de circuitos, o a través de muchos kilómetros cuando el operador de una terminal de computadora se comunica con una ubicada en otra ciudad. La información que se transmite se encuentra en formato binario y, por lo general, se representa como voltajes en las salidas de un circuito emisor, las cuales se conectan a las entradas de un circuito receptor. La figura 1-10 ilustra los dos métodos básicos para la transmisión de información digital: en **paralelo** y en **serie**.

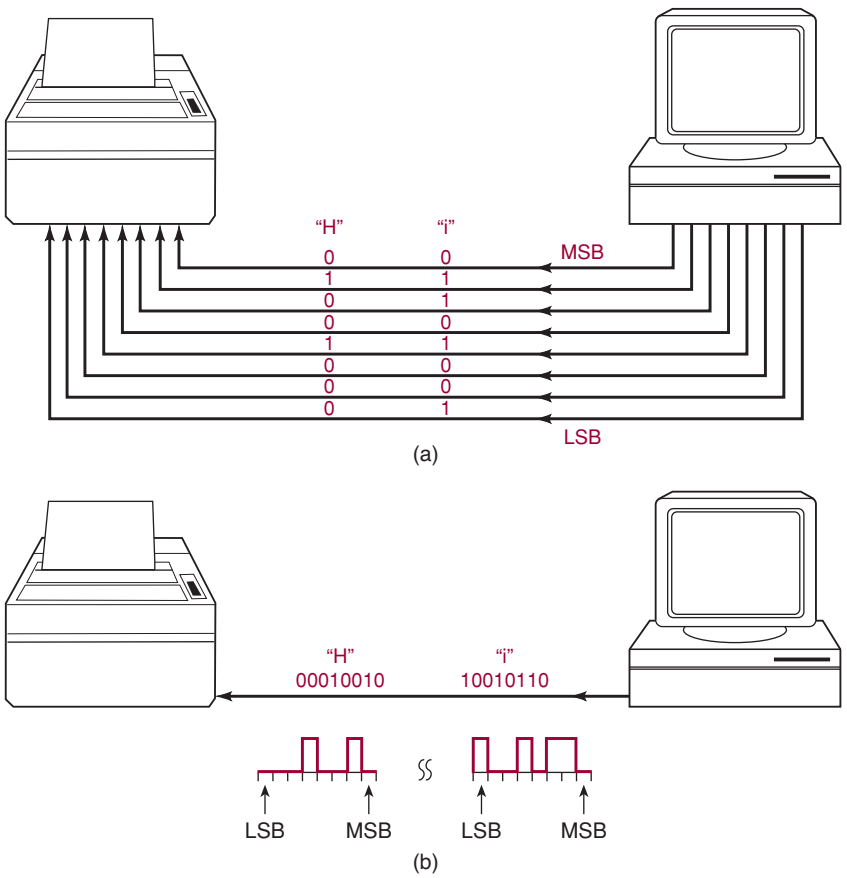


FIGURA 1-10 (a) La transmisión en paralelo utiliza una línea de conexión por bit, y todos los bits se transmiten al mismo tiempo; (b) la transmisión en serie utiliza sólo una línea de señal y los bits individuales se transmiten en forma consecutiva (uno a la vez).

La figura 1-10(a) demuestra el uso de la transmisión de datos en paralelo desde una computadora hacia una impresora mediante el uso del puerto paralelo de impresora (LPT1) de la computadora. En este escenario vamos a suponer que queremos imprimir

la palabra "Hi" en la computadora. El código binario para la "H" es 01001000 y el código binario para la "i" es 01101001. Cada carácter (la "H" y la "i") está formado de ocho bits. Mediante el uso de la transmisión en paralelo, los ocho bits se envían al mismo tiempo a través de ocho alambres. La "H" se envía primero, seguida de la "i".

La figura 1-10(b) demuestra el uso de la transmisión en serie como cuando se utiliza un puerto COM serial en la computadora para enviar datos a un módem, o cuando se utiliza un puerto USB (Bus Serie Universal) para enviar datos a una impresora. Aunque los detalles acerca de los formatos de los datos y la velocidad de la transmisión son bastante distintos entre un puerto COM y un puerto USB, los datos en sí se envían de la misma forma: un bit a la vez, a través de un solo alambre. Los bits se muestran en el diagrama como si en realidad se desplazaran sobre el alambre en el orden mostrado. El bit menos significativo de la "H" se envía primero y el bit más significativo de la "i" se envía al último. Desde luego que en la realidad sólo puede haber un bit en el alambre en cualquier instante y, por lo general, el tiempo se dibuja empezando por la izquierda y avanzando hacia la derecha. Esto produce un gráfico de bits lógicos contra tiempo de la transmisión en serie, al cual se le conoce como diagrama de tiempos. Observe que en esta presentación el bit menos significativo se muestra a la izquierda, ya que se envió primero.

La principal diferencia entre las representaciones en paralelo y en serie es la pérdida de velocidad a cambio de simpleza en el circuito. La transmisión de datos binarios desde una parte del sistema digital a otra puede realizarse con más rapidez si se utiliza el método en paralelo ya que todos los bits se transmiten al mismo tiempo, mientras que en el método serie se transmite un bit a la vez. Por otro lado, la transmisión en paralelo requiere más líneas de señal conectadas entre el emisor y el receptor de los datos binarios que la transmisión en serie. En otras palabras, la transmisión en paralelo es más rápida y la transmisión en serie requiere menos líneas de señal. A lo largo del libro encontraremos muchas veces esta comparación entre los métodos paralelo y serie para representar información binaria.

PREGUNTAS DE REPASO

1. Describa las ventajas de la transmisión de datos binarios en paralelo y en serie.

1-7 MEMORIA

Cuando se aplica una señal de entrada a la mayoría de los dispositivos o circuitos, la salida cambia de cierta forma en respuesta a la entrada, y cuando se elimina la señal de entrada la salida regresa a su estado original. Estos circuitos no exhiben la propiedad de la *memoria* debido a que sus salidas regresan a su estado normal. En los circuitos digitales, ciertos tipos de dispositivos y circuitos tienen memoria. Cuando se aplique una entrada a dicho circuito la salida cambiará su estado, pero permanecerá en el nuevo estado aún y cuando se elimine la entrada que lo provocó. Esta propiedad de retener su respuesta a una entrada momentánea se conoce como **memoria**. La figura 1-11 ilustra las operaciones con y sin memoria.

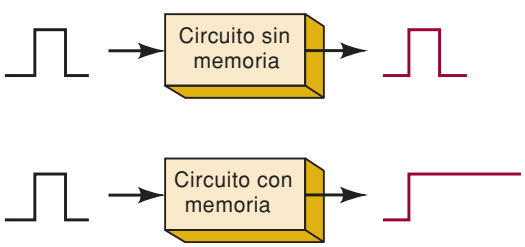


FIGURA 1-11
Comparación entre la operación sin memoria y con memoria.

Los circuitos y dispositivos de memoria juegan un importante papel en los sistemas digitales, ya que proporcionan los medios para almacenar números binarios ya sea en forma temporal o permanente, con la habilidad de modificar la información almacenada en determinado momento. Como veremos, los diversos elementos de memoria incluyen los de tipo magnético y óptico, y aquellos que utilizan circuitos de fijación electrónica (llamados *latches* y *flip-flops*).

1-8 COMPUTADORAS DIGITALES

Las técnicas digitales han encontrado aplicación en una innumerable variedad de áreas de tecnología, pero el de las **computadoras digitales** automáticas es hasta ahora la más notable y extensa. Aunque las computadoras digitales afectan en alguna forma la vida de todos nosotros, es poco probable que todos sepamos con exactitud qué es lo que hace una computadora. En términos simples, *una computadora es un sistema de hardware que realiza operaciones aritméticas, manipula datos (por lo general en formato binario) y realiza decisiones.*

En su mayoría los humanos podemos hacer todo lo que hacen las computadoras, pero éstas lo hacen con mucha más velocidad y precisión, a pesar del hecho de que las computadoras realizan todos sus cálculos y operaciones un paso a la vez. Por ejemplo, un ser humano puede tomar una lista de 10 números y encontrar la suma en una sola operación, con sólo colocar los números uno sobre otro y sumarlos columna por columna. Por otro lado, una computadora sólo puede sumar dos números a la vez, por lo que para sumar esa misma lista de números tendría que realizar nueve pasos. Desde luego el hecho de que la computadora sólo requiera de unos cuantos nanosegundos por cada paso es algo que compensa su aparente ineficiencia.

Una computadora es mucho más veloz y precisa que las personas pero, a diferencia de nosotros, debe recibir un conjunto completo de instrucciones que le indiquen *con exactitud* qué debe hacer en cada paso. Este conjunto de instrucciones, llamado **programa**, lo preparan una o más personas para cada trabajo que la computadora debe realizar. Los programas se colocan en la unidad de memoria de la computadora, en formato codificado en binario, y cada instrucción tiene un código único. La computadora toma estos códigos de instrucción de la memoria *uno por uno* y realiza la operación designada.

Principales partes de una computadora

Existen varios tipos de sistemas computacionales, pero cada uno puede descomponerse en las mismas unidades funcionales. Cada unidad realiza funciones específicas, y todas las unidades funcionan en conjunto para llevar a cabo las instrucciones que se dan en el programa. La figura 1-12 muestra las cinco partes funcionales prin-

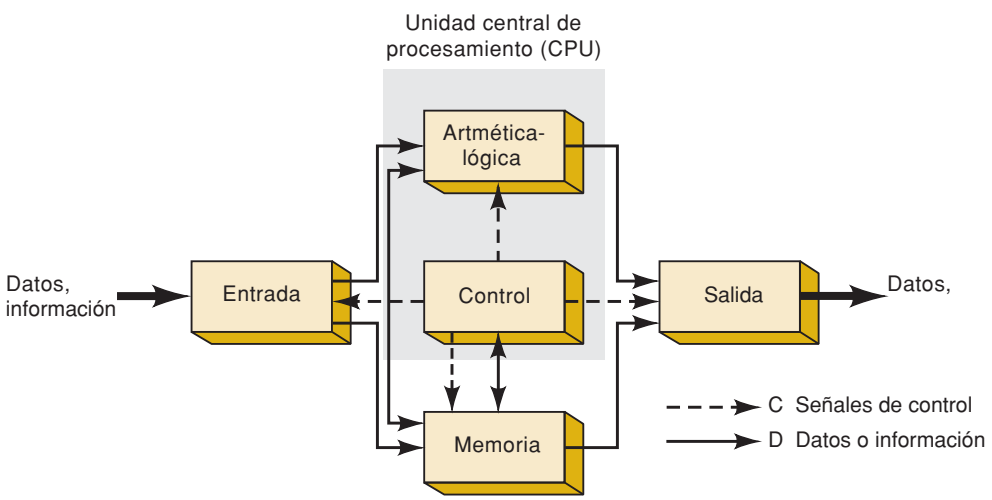


FIGURA 1-12 Diagrama funcional de una computadora digital.

cipales de una computadora digital y su interacción. Las líneas sólidas con flechas representan el flujo de datos e información. Las líneas punteadas con flechas representan el flujo de sincronización y las señales de control.

Las principales funciones de cada unidad son:

1. **Unidad de entrada.** A través de esta unidad se alimenta un conjunto completo de instrucciones y datos al sistema computacional y la unidad de memoria, para almacenarlos hasta que se requieran. Por lo general, la información se introduce a la unidad de entrada desde un teclado o un disco.
2. **Unidad de memoria.** La memoria almacena las instrucciones y los datos que se reciben de la unidad de entrada. Almacena los resultados de las operaciones aritméticas que se reciben de la unidad aritmética. También suministra información a la unidad de salida.
3. **Unidad de control.** Esta unidad recibe instrucciones de la unidad de memoria, una por una y las interpreta. Después envía las señales de sincronización y control apropiadas a todas las demás unidades para hacer que se ejecute una determinada instrucción.
4. **Unidad aritmética-lógica.** En esta unidad se realizan todos los cálculos aritméticos y las decisiones lógicas, para después almacenar los resultados en la unidad de memoria.
5. **Unidad de salida.** Esta unidad recibe los datos de la unidad de memoria y los imprime, los muestra en pantalla o los presenta de cualquier otra forma al operador (o proceso, en caso de una computadora de control de procesos).

Unidad central de procesamiento (CPU)

Como se muestra en el diagrama de la figura 1-12, por lo general, las unidades de control y aritmética-lógica se consideran como una sola unidad, a la cual se le conoce como **unidad central de procesamiento (CPU)**. La CPU contiene todos los circuitos requeridos para obtener e interpretar las instrucciones y para controlar y realizar las diversas operaciones que requieren las instrucciones.

TIPOS DE COMPUTADORAS Todas las computadoras están compuestas por las unidades básicas antes descritas, pero pueden diferir en cuanto al tamaño físico, la velocidad de operación, la capacidad de memoria y la capacidad de cómputo, entre otras características. Hoy en día los sistemas computacionales están configurados en muchas y muy diversas formas, con muchas características comunes y también diferencias. Tanto empresas como universidades utilizan sistemas computacionales grandes instalados de manera permanente en varios gabinetes, para el soporte de tecnología de información. Las computadoras personales de escritorio se utilizan en nuestros hogares y oficinas para ejecutar programas útiles que mejoran nuestras vidas y proporcionan comunicación con otras computadoras. En los PDA se utilizan computadoras portátiles, y en los sistemas de videojuegos se utilizan computadoras especializadas. La forma más prevalente de computadoras se encuentra realizando tareas de rutina dedicadas en electrodomésticos y demás sistemas a nuestro alrededor.

En la actualidad, todos los sistemas con excepción de los más grandes, utilizan tecnología que ha evolucionado a partir de la invención del **microprocesador**. Este componente es en esencia una unidad central de procesamiento (CPU) en un circuito integrado, el cual puede conectarse a los demás bloques de un sistema computacional. Por lo general, a las computadoras que utilizan un microprocesador como CPU se les denomina **microcomputadoras**. Las microcomputadoras de propósito general (por ejemplo: las PC, las PDA, etc.) realizan una variedad de tareas en una amplia gama de aplicaciones, dependiendo del software (programas) que ejecuten. En contraste tenemos a las computadoras dedicadas que realizan actividades tales como operar el motor de un automóvil, controlar el sistema de frenos antibloqueo u operar el horno de microondas. El usuario no puede programar estas computadoras ya que sólo realizan una tarea de control: se les conoce como **microcontroladores**.

Como estos microcontroladores son una parte integral de un sistema más grande y sirven un propósito dedicado, también se les llama *controladores embebidos*. Por lo general, los microcontroladores tienen todos los elementos de una computadora completa (CPU, memoria y puertos de entrada/salida), todo lo cual está contenido dentro de un solo circuito integrado. Los microcontroladores embebidos se encuentran en los electrodomésticos de su cocina, en el equipo de entretenimiento, en las fotocopiadoras, en los cajeros automáticos del banco, en el equipo automatizado de fabricación, en la instrumentación médica, entre muchas otras aplicaciones.

Como puede ver, aún las personas que no tienen una PC o que no utilizan una en la escuela o el trabajo utilizan a diario microcomputadoras, ya que muchos dispositivos electrónicos, electrodomésticos, equipo de oficina y muchos otros aparatos están fabricados con base en microcontroladores embebidos (integrados). En esta era digital, si usted trabaja, juega o va a la escuela, no podrá escapar: con toda seguridad en alguna parte estará utilizando una microcomputadora.

PREGUNTAS DE REPASO

1. Explique la diferencia ente un circuito digital con memoria y uno sin memoria.
2. Nombre las cinco unidades funcionales principales de una computadora.
3. ¿Cuáles son las dos unidades que conforman la CPU?
4. Un chip de CI que contiene un CPU se llama _____.

RESUMEN

1. Las dos formas básicas de representar el valor numérico de las cantidades físicas son la analógica (continua) y la digital (discreta).
2. La mayoría de las cantidades en el mundo real son analógicas, pero, por lo general, las técnicas digitales son superiores a las técnicas analógicas, y se pronostica que la mayor parte de los avances estarán en el ámbito digital.
3. El sistema numérico binario (0 y 1) es el sistema básico que se utiliza en la tecnología digital.
4. Los circuitos digitales o lógicos operan con voltajes que se encuentran dentro de intervalos prescritos, los cuales representan ya sea un 0 o un 1 binario.
5. Las dos formas básicas de transferir información digital son: en paralelo (todos los bits simultáneamente) y en serie (un bit a la vez).
6. Las partes principales de todas las computadoras son las unidades: de entrada, de control, de memoria, de aritmética-lógica y de salida.
7. La CPU es la combinación de la unidad aritmética-lógica y la unidad de control.
8. Por lo general, una microcomputadora tiene una CPU que se encuentra en un solo chip, y se le llama *microprocesador*.
9. Un microcontrolador es una microcomputadora diseñada de manera especial para aplicaciones de control dedicadas (no de propósito general).

TÉRMINOS IMPORTANTES*

bit
circuitos digitales/circuitos lógicos
computadoras digitales
microcomputadoras
microcontroladores
microprocesador
programa
sistema digital
transmisión en paralelo
transmisión en serie
unidad aritmética-lógica

* Encontrará estos términos en **negritas** a lo largo del capítulo; además están definidos en el Glosario al final del libro. Esto aplica en todos los capítulos.

convertidor analógico-digital (ADC)
convertidor digital-analógico (DAC)
diagrama de tiempos
memoria
representación analógica
representación digital
sistema analógico
sistema binario
sistema decimal
unidad central de procesamiento (CPU)
unidad de control
unidad de entrada
unidad de memoria
unidad de salida

PROBLEMAS

SECCIÓN 1-2

1-1.* ¿Cuáles de las siguientes cantidades son analógicas y cuáles digitales?
(a) El número de átomos en una muestra de material.
(b) La altitud de una aeronave.
(c) La presión en la llanta de una bicicleta.
(d) La corriente que pasa a través de una bocina.
(e) La configuración del temporizador en un horno de microondas.

1-2. ¿Cuáles de las siguientes cantidades son analógicas y cuáles digitales?
(a) La anchura de una pieza de madera.
(b) La cantidad de tiempo transcurrido antes de que se apague el timbre.
(c) La hora del día que se muestra en un reloj de cuarzo.
(d) La altitud por encima del nivel del mar, si se mide desde una escalera.
(e) La altitud por encima del nivel del mar, si se mide desde una rampa.

SECCIÓN 1-3

1-3.* Convierta los siguientes números binarios a sus valores decimales equivalentes.
(a) 11001_2
(b) 1001.1001_2
(c) 10011011001.10110_2

1-4. Convierta los siguientes números binarios a decimal.
(a) 10011_2
(b) 1100.0101
(c) 10011100100.10010

1-5.* Muestre la secuencia de conteo binario de 000 a 111, usando tres bits.

1-6. Muestre la secuencia de conteo binario de 000000 a 111111, usando seis bits.

1-7.* ¿Cuál es el máximo número que podemos contar si utilizamos 10 bits?

1-8. ¿Cuál es el máximo número que podemos contar si utilizamos 14 bits?

1-9.* ¿Cuántos bits se necesitan para contar hasta 511?

1-10. ¿Cuántos bits se necesitan para contar hasta 63?

SECCIÓN 1-4

1-11.* Dibuje el diagrama de tiempos para una señal digital que alterne en forma continua entre 0.2 V (0 binario) durante 2 ms y 4.4 V (1 binario) durante 4 ms.

1-12. Dibuje el diagrama de tiempos para una señal que alterne entre 0.3 V (0 binario) durante 5 ms y 3.9 V (1 binario) durante 2 ms.

* Encontrará las respuestas a los problemas marcados con un asterisco al final del libro.

SECCIÓN 1-6

1-13.* Suponga que se van a transmitir en binario los valores enteros decimales desde el 0 hasta el 15.

(a) ¿Cuántas líneas se necesitan si se utiliza la representación en paralelo?

(b) ¿Cuántas líneas se necesitan si se utiliza la representación en serie?

SECCIONES 1-7 Y 1-8

1-14. ¿Cuál es la diferencia entre un microprocesador y una microcomputadora?

1-15. ¿Cuál es la diferencia entre un microcontrolador y una microcomputadora?

RESPUESTAS A LAS PREGUNTAS DE REPASO DE LAS SECCIONES

SECCIÓN 1-1

1. Las cantidades analógicas pueden tomar *cualquier* valor sobre un intervalo continuo; las cantidades digitales sólo pueden tomar valores *discretos*.

SECCIÓN 1-2

1. Es más fácil de diseñar; es más fácil de almacenar información; tiene mayor precisión; capacidad de programación; se ve menos afectado por el ruido; tiene un mayor grado de integración.
2. Las cantidades físicas del mundo real son analógicas. El procesamiento digital requiere tiempo.

SECCIÓN 1-3

1. 107_{10} 2. 11000_2 3. 4095_{10}

SECCIÓN 1-5

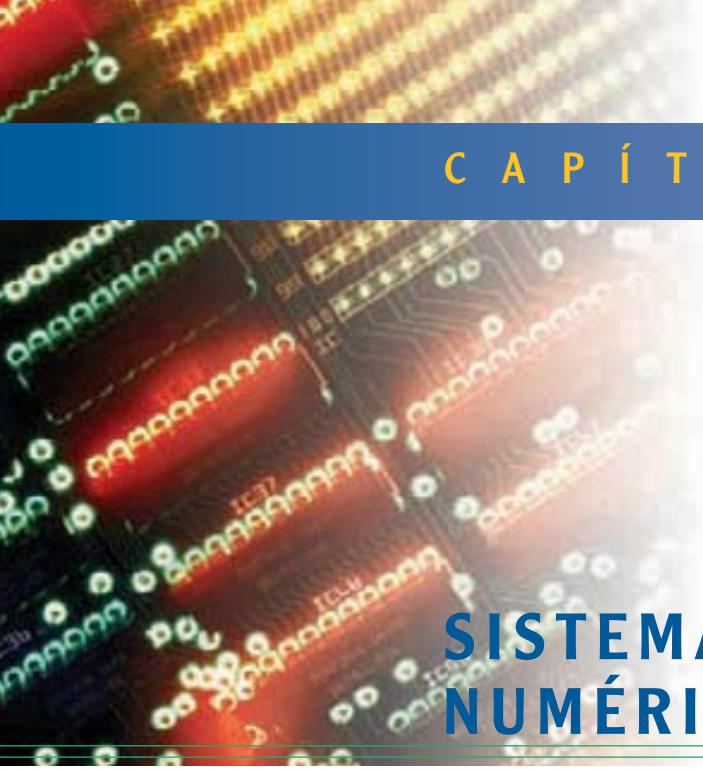
1. Falso. 2. Sí, siempre y cuando los dos voltajes de entrada estén dentro del mismo intervalo de niveles lógicos. 3. Lógico. 4. Diagrama de tiempos.

SECCIÓN 1-6

1. La transmisión en paralelo es más rápida; la transmisión en serie sólo requiere una línea de señal.

SECCIÓN 1-8

1. Si hay un cambio instantáneo en la señal de entrada, el circuito que tiene memoria cambiará su salida y *permanecerá* con ese cambio. 2. Entrada, salida, memoria, aritmética-lógica, de control. 3. Unidad de control y unidad aritmética-lógica.
4. Microprocesador.



CAPÍTULO 2

SISTEMAS Y CÓDIGOS NUMÉRICOS

CONTENIDO

2-1 Conversiones de binario a decimal
2-2 Conversiones de decimal a binario
2-3 Sistema numérico hexadecimal
2-4 Código BCD
2-5 Código Gray
2-6 Integración de los sistemas numéricos
2-7 Byte, nibble y palabra
2-8 Códigos alfanuméricos
2-9 Método de paridad para la detección de errores
2-10 Aplicaciones

OBJETIVOS

Al terminar este capítulo, usted podrá:

- Convertir un número de un sistema numérico (decimal, binario, hexadecimal) a su equivalente en uno de los otros sistemas numéricos.
- Citar las ventajas del sistema numérico hexadecimal.
- Contar en hexadecimal.
- Representar números decimales mediante el código BCD; y citar las ventajas y desventajas en cuanto al uso del código BCD.
- Comprender la diferencia entre BCD y binario directo.
- Entender el propósito de los códigos alfanuméricos tales como el código ASCII.
- Explicar el método de paridad para la detección de errores.
- Determinar el bit de paridad que se adjuntará a una cadena de datos digitales.

INTRODUCCIÓN

El sistema numérico binario es el más importante en los sistemas digitales, aunque también hay otros que son importantes. El sistema decimal es importante ya que se utiliza de manera universal para representar cantidades fuera de un sistema digital. Esto significa que habrá situaciones en las que los valores decimales deban convertirse en valores binarios antes de introducirse en el sistema digital. Por ejemplo, cuando oprime un número decimal en su calculadora de bolsillo (o computadora), los circuitos en el interior de la máquina convierten el número decimal en un valor binario.

De igual forma, habrá situaciones en las que los valores binarios en las salidas de un sistema digital deberán convertirse en valores decimales para presentarlos al mundo exterior. Por ejemplo, su calculadora (o computadora) utiliza números binarios para calcular la solución a un problema y después convierte los resultados en dígitos decimales antes de mostrarlos en pantalla.

Como verá más adelante, no es fácil ver un número binario y convertirlo en su valor decimal equivalente. Resulta muy tedioso tener que introducir una larga secuencia de unos (1s) y ceros (0s) en un teclado numérico, o escribir números binarios extensos en papel. En especial es difícil tratar de comunicar una cantidad binaria cuando se habla con otra persona. El sistema numérico hexadecimal (base 16) se ha convertido la forma estándar de comunicar valores numéricos en los sistemas digitales. La gran ventaja es que los números hexadecimales pueden convertirse con facilidad a binario y viceversa.

Se han ideado otros métodos para representar cantidades decimales con dígitos codificados en binario, los cuales no son en sí verdaderos sistemas numéricos, pero ofrecen la facilidad de conversión entre el código binario y el sistema numérico decimal. A esto se le conoce como decimal codificado en binario. Es muy importante que usted pueda interpretar los valores en cualquier sistema y realizar conversiones entre cualquiera de estas representaciones numéricas, ya

que las cantidades y los patrones de bits podrían estar representadas mediante cualquiera de estos métodos en cualquier sistema dado y en todo el material escrito de soporte para esos sistemas. También cubriremos otros códigos que utilizan 1s y 0s para representar cosas tales como caracteres alfanuméricos, ya que son muy comunes en los sistemas digitales.

2-1 CONVERSIONES DE BINARIO A DECIMAL

Como se explicó en el capítulo 1, el sistema numérico binario es un sistema posicional, en el cual cada dígito binario (bit) lleva un cierto peso basado en su posición relativa al LSB. Cualquier número binario puede convertirse en su equivalente decimal con sólo sumar todos los pesos de las diversas posiciones en el número binario que contengan 1. Para ilustrar esto, vamos a convertir el número 11011_2 a su equivalente decimal.

$$\begin{aligned} &1 \quad 1 \quad 0 \quad 1 \quad 1_2 \\ &2^4 + 2^3 + 0 + 2^1 + 2^0 = 16 + 8 + 2 + 1 \\ &\qquad\qquad\qquad\quad = 27_{10} \end{aligned}$$

Hagamos ahora otro ejemplo con un mayor número de bits:

$$\begin{aligned} &1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1_2 = \\ &2^7 + 0 + 2^5 + 2^4 + 0 + 2^2 + 0 + 2^0 = 181_{10} \end{aligned}$$

Observe que el procedimiento es encontrar los pesos (es decir, potencias de 2) para cada posición de bit que contenga un 1, y después hay que sumarlos. Observe también que el MSB tiene un peso de 2^7, aun y cuando es el octavo bit; esto se debe a que el LSB es el primer bit y tiene un peso de 2^0.

PREGUNTAS DE REPASO

1. Convierta el número 100011011011_2 en su equivalente decimal.
2. ¿Cuál es el peso del MSB de un número de 16 bits?

2-2 CONVERSIONES DE DECIMAL A BINARIO

Existen dos métodos para convertir un número *entero* decimal en su representación equivalente en el sistema binario. El primer método es el proceso inverso al que se describió en la sección 2-1. El número decimal tan sólo se expresa como una suma de potencias de 2, y después se escriben 1s y 0s en las posiciones de bit apropiadas. Para ilustrar lo anterior veamos lo siguiente:

$$\begin{aligned} 45_{10} = 32 + 8 + 4 + 1 &= 2^5 + 0 + 2^3 + 2^2 + 0 + 2^0 \\ &= 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1_2 \end{aligned}$$

Observe que se coloca un 0 en las posiciones 2^1 y 2^4, ya que todas las posiciones deben tomarse en cuenta. Veamos otro ejemplo:

$$\begin{aligned} 76_{10} = 64 + 8 + 4 &= 2^6 + 0 + 0 + 2^3 + 2^2 + 0 + 0 \\ &= 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0_2. \end{aligned}$$

División repetida

El segundo método para convertir enteros decimales a binario es el que utiliza la división entre 2. Para la conversión, que se muestra a continuación para el número 25_{10}, se requiere dividir en forma repetida el número decimal entre 2 y anotar el residuo después de cada división hasta que se obtenga un cociente de 0. El resultado binario se obtiene al escribir el primer residuo como el LSB y el último como el MSB. Este proceso, que se ilustra en el diagrama de flujo de la figura 2-1, puede usarse también para convertir de decimal a cualquier otro sistema numérico, como veremos más adelante.

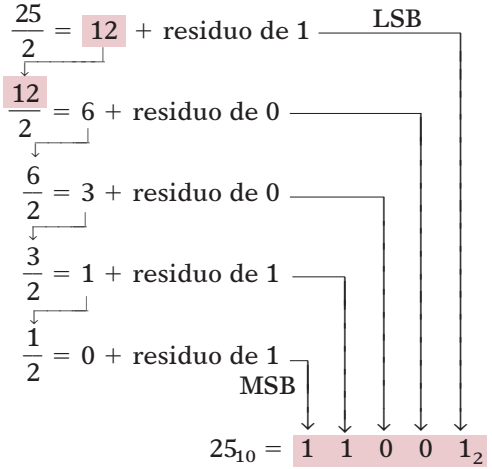
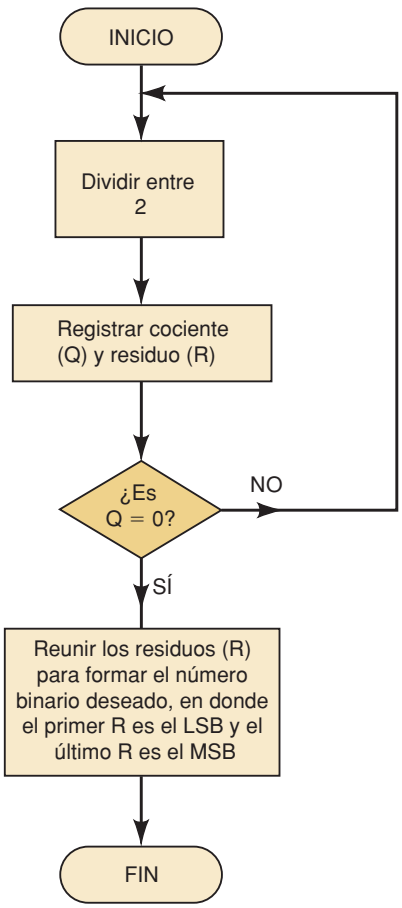


FIGURA 2-1 Diagrama de flujo para el método de división repetida de la conversión de enteros decimales a binarios. Puede usarse el mismo proceso para convertir un entero decimal a cualquier otro sistema numérico.



SUGERENCIA PARA EL USO DE CALCULADORA:

Si utiliza una calculadora para realizar las divisiones entre 2, podrá saber si el residuo es 0 o 1 y si el resultado tiene o no una parte fraccionaria. Por ejemplo, 25/2 produciría 12.5. Como hay una parte fraccionaria (.5), el residuo es un 1. Si no hubiera parte fraccionaria, como en 12/2 = 6, entonces el residuo sería 0. El siguiente ejemplo demuestra esto.

EJEMPLO 2-1

Convierta el número 37_{10} en binario. Trate de resolverlo por su cuenta antes de ver la solución.

Solución

$$\frac{37}{2} = 18.5 \longrightarrow \text{residuo de 1 (LSB)}$$

$$\frac{18}{2} = 9.0 \longrightarrow 0$$

$$\frac{9}{2} = 4.5 \longrightarrow 1$$

$$\frac{4}{2} = 2.0 \longrightarrow 0$$

$$\frac{2}{2} = 1.0 \longrightarrow 0$$

$$\frac{1}{2} = 0.5 \longrightarrow 1 \text{ (MSB)}$$

Por lo tanto, $37_{10} = \mathbf{100101}_2$.

Alcance de conteo

Recuerde que si utilizamos N bits, podemos contar hasta 2^N números decimales distintos que van desde 0 hasta $2^N - 1$. Por ejemplo, para $N = 4$ podemos contar desde 0000_2 hasta 1111_2, 0_{10} hasta 15_{10}, para un total de 16 números distintos. Aquí, el valor decimal más grande es $2^4 - 1 = 15$, y hay 2^4 números distintos.

Entonces, en general, podemos decir que:

Si utilizamos N bits, podemos representar números decimales que van desde 0 hasta $2^N - 1$, un total de 2^N números distintos.

EJEMPLO 2-2

(a) ¿Cuál es el intervalo de valores decimales que pueden representarse en ocho bits?

(b) ¿Cuántos bits se necesitan para representar valores decimales que van desde 0 hasta 12,500?

Solución

(a) Aquí tenemos que $N = 8$. Por ende, podemos representar números decimales desde 0 hasta $2^8 - 1 = \mathbf{255}$. Para verificar lo anterior podemos comprobar si 11111111_2 se convierte en 255_{10}.

(b) Con 13 bits podemos contar desde el 0 decimal hasta $2^{13} - 1 = 8191$. Con 14 bits podemos contar desde 0 hasta $2^{14} - 1 = 16{,}383$. Es evidente que no son suficientes 13 bits, pero 14 bits nos llevarán más allá de 12,500. Por ende, el número de bits requerido es **14**.

PREGUNTAS DE REPASO

1. Convierta 83_{10} en binario usando ambos métodos.
2. Convierta 729_{10} en binario usando ambos métodos. Revise su respuesta y convierta el resultado de vuelta en decimal.
3. ¿Cuántos bits se requieren para contar hasta 1 millón decimal?

2-3 SISTEMA NUMÉRICO HEXADECIMAL

El **sistema numérico hexadecimal** utiliza la base 16. En consecuencia, tiene 16 símbolos posibles para los dígitos. Utiliza los dígitos del 0 al 9 más las letras A, B, C, D, E y F como símbolos para los 16 dígitos. Las posiciones de los dígitos se ponderan como potencias de 16, como se muestra a continuación, en lugar de ponderarse como potencias de 10, como en el sistema decimal.

16^4	16^3	16^2	16^1	16^0	16^{-1}	16^{-2}	16^{-3}	16^{-4}

Punto hexadecimal

La tabla 2-1 muestra las relaciones entre hexadecimal, decimal y binario. Observe que cada dígito hexadecimal representa un grupo de cuatro dígitos binarios. Es importante recordar que los dígitos hex (abreviación de "hexadecimal") de la A a la F son equivalentes a los valores decimales del 10 al 15.

TABLA 2-1



Hexadecimal	Decimal	Binario
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Conversión de hexadecimal a decimal

Es posible convertir un número hexadecimal a su equivalente decimal gracias a que la posición de cada dígito hexadecimal tiene un peso equivalente a una potencia

de 16. El LSD tiene un peso de $16^0 = 1$; la siguiente posición de dígito tiene un peso de $16^1 = 16$; la siguiente tiene un peso de $16^2 = 256$; y así sucesivamente. En el siguiente ejemplo demostraremos el proceso de conversión.

SUGERENCIA PARA EL USO DE CALCULADORA:

Puede usar la función y^x de la calculadora para evaluar las potencias de 16.

$$\begin{aligned} 356_{16} &= 3 \times 16^2 + 5 \times 16^1 + 6 \times 16^0 \\ &= 768 + 80 + 6 \\ &= 854_{10} \end{aligned}$$

$$\begin{aligned} 2AF_{16} &= 2 \times 16^2 + 10 \times 16^1 + 15 \times 16^0 \\ &= 512 + 160 + 15 \\ &= 687_{10} \end{aligned}$$

Observe que en el segundo ejemplo, el valor 10 se sustituyó por A y el valor 15 por F en la conversión a decimal.

Para practicar, verifique que $1BC2_{16}$ sea igual a 7106_{10}.

Conversión de decimal a hexadecimal

Recuerde que para realizar la conversión de decimal a binario utilizamos la división repetida entre 2. De igual forma, la conversión de decimal a hexadecimal puede realizarse mediante el uso de la división repetida entre 16 (figura 2-1). El siguiente ejemplo contiene dos casos de esta conversión.

EJEMPLO 2-3

(a) Convierta 423_{10} a hexadecimal.

Solución

$$\frac{423}{16} = 26 + \text{residuo de } 7$$

$$\frac{26}{16} = 1 + \text{residuo de } 10$$

$$\frac{1}{16} = 0 + \text{residuo de } 1$$

$$423_{10} = 1A7_{16}$$

(b) Convierta 214_{10} a hexadecimal.

Solución

$$\frac{214}{16} = 13 + \text{residuo de } 6$$

$$\frac{13}{16} = 0 + \text{residuo de } 13$$

$$214_{10} = D6_{16}$$

Observe de nuevo que los residuos de los procesos de división forman los dígitos del número hexadecimal. Observe también que cualquier residuo mayor de 9 se representa por las letras de la A a la F.

SUGERENCIA PARA EL USO DE CALCULADORA:

Si utiliza una calculadora para realizar las divisiones en el proceso de conversión, los resultados incluirán una fracción decimal en vez de un residuo. Para obtener el residuo multiplique la fracción por 16. En el ejemplo 2-3(b) la calculadora habría producido lo siguiente:

$$\frac{214}{16} = 13.375$$

El residuo se convierte en $(0.375) \times 16 = 6$.

Conversión de hexadecimal a binario

El sistema numérico hexadecimal se utiliza principalmente como método "abreviado" para representar números binarios. En realidad es muy sencillo convertir un número hexadecimal en binario. *Cada* dígito hexadecimal se convierte en su equivalente binario de cuatro bits (tabla 2-1). Esto se ilustra a continuación para el número $9F2_{16}$.

$$\begin{aligned} 9F2_{16} &= \quad 9 \qquad\quad F \qquad\quad 2 \\ &\qquad\; \downarrow \qquad\; \downarrow \qquad\; \downarrow \\ &= 1\ 0\ 0\ 1 \quad 1\ 1\ 1\ 1 \quad 0\ 0\ 1\ 0 \\ &= 100111110010_2 \end{aligned}$$

Para practicar, verifique que $BA6_{16} = 101110100110_2$.

Conversión de binario a hexadecimal

Esta conversión es sólo el inverso del proceso antes mencionado. El número binario se separa en grupos de *cuatro* bits, y cada grupo se convierte en su dígito hexadecimal equivalente. Se agregan ceros (los que se muestran ensombrecidos) según sea necesario para completar un grupo de cuatro bits en el MSD.

$$\begin{aligned} 1\,1\,1\,0\,1\,0\,0\,1\,1\,0_2 &= \underbrace{0\ 0\ 1\ 1}_{3}\ \underbrace{1\ 0\ 1\ 0}_{A}\ \underbrace{0\ 1\ 1\ 0}_{6} \\ &= 3A6_{16} \end{aligned}$$

Para hacer las conversiones entre hexadecimal y binario es necesario conocer los números binarios de cuatro bits (del 0000 hasta el 1111) y sus dígitos hexadecimales correspondientes. Una vez que memorize, la conversión podrá realizarse con rapidez sin necesidad de hacer cálculos. Esto explica por qué el sistema hexadecimal es tan útil para representar números binarios extensos.

Para practicar, verifique que $101011111_2 = 15F_{16}$.

Conteo en hexadecimal

Al contar en hexadecimal, puede incrementarse (en 1) la posición de cada dígito, desde el 0 hasta la F. Una vez que la posición de un dígito llega al valor F, se restablece

a 0 y se incrementa la posición del siguiente dígito. En las siguientes secuencias de conteo hexadecimal se ilustra esto:

(a) 38, 39, 3A, 3B, 3C, 3D, 3F, 40, 41, 42

(b) 6F8, 6F9, 6FA, 6FB, 6FC, 6FD, 6FE, 6FF, 700

Observe que cuando hay un 9 en la posición de un dígito, se convierte en A cuando se incrementa.

Con N posiciones de dígitos hexadecimales podemos contar desde el 0 decimal hasta $16^N - 1$, para un total de 16^N valores distintos. Por ejemplo, con tres dígitos hexadecimales podemos contar desde 000_{16} hasta FFF_{16}, 0_{10} hasta 4095_{10}, para un total de $4096 = 16^3$ valores distintos.

Utilidad de los números hexadecimales

Los números hexadecimales se utilizan a menudo en un sistema digital como una manera "abreviada" de representar cadenas de bits. Al trabajar con las computadoras, es muy común usar cadenas de hasta 64 bits. Estas cadenas binarias no siempre representan un valor numérico, sino que, como veremos más adelante, pueden indicar algún tipo de código que conlleve información no numérica. Al trabajar con un número extenso de bits es más conveniente y menos errático escribir los números binarios en hexadecimal, ya que, como hemos visto, es bastante sencillo realizar conversiones entre hexadecimal y binario, o viceversa. Para ilustrar la ventaja de la representación hexadecimal de una cadena binaria, suponga que tiene en su poder una impresión del contenido de 50 ubicaciones de memoria, cada una de las cuales es un número de 16 bits y usted tiene que revisarlas comparándolas con una lista. ¿Qué preferiría revisar, 50 números como éste: 0110111001100111, o 50 números como éste: 6E67? ¿Y con cuál sería más probable equivocarse? No obstante, es importante tener en cuenta que todos los circuitos digitales trabajan en binario. Los números hexadecimales sólo se utilizan como una conveniencia para los humanos involucrados. Sería conveniente que memorizara el patrón binario de 4 bits para cada dígito hexadecimal. Sólo entonces se dará cuenta de la utilidad de esta herramienta en los sistemas digitales.

EJEMPLO 2-4

Convierta el número 378 decimal en un número binario de 16 bits, primero convirtiendo el número en hexadecimal.

Solución

$$\frac{378}{16} = 23 + \text{residuo de } 10_{10} = A_{16}$$

$$\frac{23}{16} = 1 + \text{residuo de } 7$$

$$\frac{1}{16} = 0 + \text{residuo de } 1$$

Por lo tanto, $378_{10} = 17A_{16}$. Este valor hexadecimal puede convertirse con facilidad en el número binario 000101111010. Por último, podemos expresar el número 378_{10} como un número de 16 bits si le agregamos cuatro 0s a la izquierda:

$$378_{10} = 0000 \quad 0001 \quad 0111 \quad 1010_2$$

EJEMPLO 2-5

Convierta el número $B2F_{16}$ en decimal.

Solución

$$\begin{aligned} B2F_{16} &= B \times 16^2 + 2 \times 16^1 + F \times 16^0 \\ &= 11 \times 256 + 2 \times 16 + 15 \\ &= 2863_{10} \end{aligned}$$

Resumen de las conversiones

En estos momentos es probable que su cabeza esté dando vueltas a medida que trata de mantener el sentido con todas estas distintas conversiones de un sistema numérico a otro. Tal vez se haya dado cuenta que muchas de estas conversiones pueden realizarse en forma *automática* en su calculadora con sólo oprimir una tecla, pero es importante que las domine para que pueda comprender el proceso. Además, ¿qué pasaría si su calculadora se quedara sin energía en un momento crucial y no tuviera un reemplazo a la mano? El siguiente resumen le ayudará, pero nada se compara con la práctica continua.

1. Al convertir de binario (o hexadecimal) a decimal, utilice el método de tomar la suma ponderada de la posición de cada bit.
2. Al convertir de decimal a binario (o hexadecimal), utilice el método de la división repetida entre 2 (o 16) y recolectar los residuos (figura 2-1).
3. Al convertir de binario a hexadecimal, divida el número en grupos de cuatro bits y convierta cada grupo en el dígito hexadecimal correcto.
4. Al convertir de hexadecimal a binario, convierta cada dígito en su equivalente de cuatro bits.

PREGUNTAS DE REPASO

1. Convierta el número $24CE_{16}$ en decimal.
2. Convierta el número 3117_{10} en hexadecimal y después de hexadecimal a binario.
3. Convierta el número 1001011110110101_2 en hexadecimal.
4. Escriba los siguientes cuatro números en esta secuencia de conteo hexadecimal: E9A, E9B, E9C, E9D, _____, _____, _____, _____.
5. Convierta el número 3527_{16} en binario.
6. ¿Qué rango de valores decimales puede representarse mediante un número hexadecimal de cuatro dígitos?

2-4 CÓDIGO BCD

Cuando se representan números, letras o palabras mediante un grupo especial de símbolos, decimos que están siendo codificados, y al grupo de símbolos se le llama *código*. Tal vez uno de los códigos más conocidos sea el Morse, en el cual una seria de puntos y rayas representan las letras del alfabeto.

Hemos visto que cualquier número decimal puede representarse mediante un número binario equivalente. El grupo de 1s y 0s en el número binario puede considerarse como un código que representa el número decimal. Cuando un número decimal se representa por su número binario equivalente, le llamamos **código binario directo**.

Todos los sistemas digitales utilizan cierta forma de números binarios para su operación interna, pero el mundo externo es decimal por naturaleza. Esto significa que con frecuencia se realizan conversiones entre los sistemas decimal y binario. Hemos visto que las conversiones entre decimal y binario pueden volverse extensas y complicadas cuando se manejan números grandes. Por esta razón, en ciertas situaciones se utiliza un medio para codificar números decimales que combina algunas características tanto del sistema decimal como del sistema binario.

Código decimal codificado en binario

Si *cada* dígito de un número decimal se representa mediante su equivalente binario, el resultado es un código que se conoce como **decimal codificado en binario** (que en lo sucesivo abreviaremos como BCD). Como un dígito decimal puede llegar hasta el 9, se requieren cuatro bits para codificar cada dígito (el código binario para el 9 es 1001).

Para ilustrar el código BCD, considere como ejemplo el número decimal 874. Cada *dígito* se cambia a su equivalente binario de la siguiente manera:

8	7	4	(decimal)
↓	↓	↓	
1000	0111	0100	(BCD)

Como segundo ejemplo, vamos a cambiar el número 943 a su representación en código BCD:

9	4	3	(decimal)
↓	↓	↓	
1001	0100	0011	(BCD)

Una vez más, cada dígito decimal se cambia a su equivalente binario directo. Observe que *siempre* se utilizan cuatro bits para cada dígito.

Así, el código BCD representa cada dígito del número decimal mediante un número binario de cuatro bits. Es evidente que sólo se utilizan los números binarios de cuatro bits del 0000 al 1001. El código BCD no utiliza los números 1010, 1011, 1100, 1101, 1110 y 1111. En otras palabras, sólo se utilizan 10 de los 16 posibles grupos de código binario de cuatro bits. Si llega a aparecer uno de los números "prohibidos" de cuatro bits en una máquina que utilice el código BCD, por lo general, es una indicación de que se produjo un error.

EJEMPLO 2-6

Convierta el número 0110100000111001 (BCD) en su equivalente decimal.

Solución

Divida el número BCD en grupos de cuatro bits y convierta cada grupo en decimal.

0110	1000	0011	1001
6	8	3	9

EJEMPLO 2-7

Convierta el número BCD 011111000001 en su equivalente decimal.

Solución

0111	1100	0001
7	↓	1
	El grupo con el código prohibido indica un error en el número BCD	

Comparación entre BCD y binario

Es importante entender que BCD no es otro sistema numérico como el binario, el decimal o el hexadecimal. De hecho, se utiliza el sistema decimal pero cada dígito está codificado en su equivalente binario. También es importante comprender que un número BCD *no* es lo mismo que un número binario directo. Un número binario directo toma el número decimal *completo* y lo representa en binario; el código BCD convierte *cada dígito* decimal en binario de manera individual. Para ilustrar esto, tome el número 137 y compare sus códigos binario directo y BCD:

$$137_{10} = 10001001_2 \quad \text{(binario)}$$

$$137_{10} = 0001\ \ 0011\ \ 0111 \quad \text{(BCD)}$$

Para representar el 137, el código BCD requiere 12 bits, mientras que el código binario directo sólo requiere de ocho bits. El código BCD requiere más bits que el binario directo para representar números decimales de más de un dígito, ya que no utiliza todos los grupos de cuatro bits posibles, como se indicó antes, y es, por lo tanto, algo ineficiente.

La principal ventaja del código BCD es la relativa facilidad de convertir BCD en decimal y viceversa. Sólo necesitan recordarse los grupos de código de cuatro bits para los dígitos decimales del 0 al 9. Esta facilidad de conversión es muy importante desde el punto de vista del hardware, ya que en un sistema digital son los circuitos lógicos los que realizan las conversiones hacia y desde decimal.

PREGUNTAS DE REPASO

1. Represente el valor decimal 178 mediante su equivalente binario directo. Luego codifique el mismo número decimal en BCD.
2. ¿Cuántos bits se requieren para representar un número decimal de ocho dígitos en BCD?
3. ¿Cuál es la ventaja de codificar un número decimal en BCD, en lugar de hacerlo en binario directo? ¿Cuál es la desventaja?

2-5 CÓDIGO GRAY

Los sistemas digitales operan a velocidades muy elevadas y responden a los cambios que se producen en las entradas digitales. Al igual que en la vida real, cuando varias condiciones de entrada están cambiando al mismo tiempo la situación puede malinterpretarse, con lo cual se puede llegar a producir una reacción errónea. Cuando se ven los bits en una secuencia de conteo binario, a menudo hay varios bits que deben cambiar estados al mismo tiempo. Por ejemplo, considere cuando el número binario de tres bits correspondiente al 3 decimal cambia a 4: los tres bits deben cambiar de estado.

Para reducir la probabilidad de que un circuito digital malinterprete una entrada cambiante, se desarrolló el **código Gray** como una manera de representar una secuencia de números. El aspecto único del código Gray es que, entre dos números sucesivos en la secuencia sólo un bit cambia. La tabla 2-2 muestra la traducción entre el valor del código binario de tres bits y el código Gray. Para convertir de binario a Gray sólo hay que empezar en el bit más significativo y usarlo como el MSB de Gray, como muestra la figura 2-2(a). Después se compara el MSB binario con el siguiente bit binario (B1). Si son iguales, entonces G1 = 0; si son distintos, entonces G1 = 1. Para encontrar G0 se compara B1 con B0.

TABLA 2-2
Equivalencia entre el código binario de tres bits y el código Gray.

B_2	B_1	B_0	G_2	G_1	G_0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

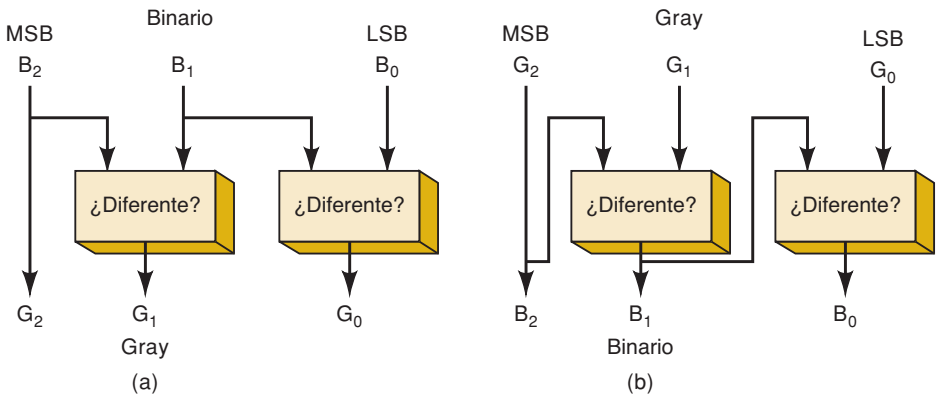
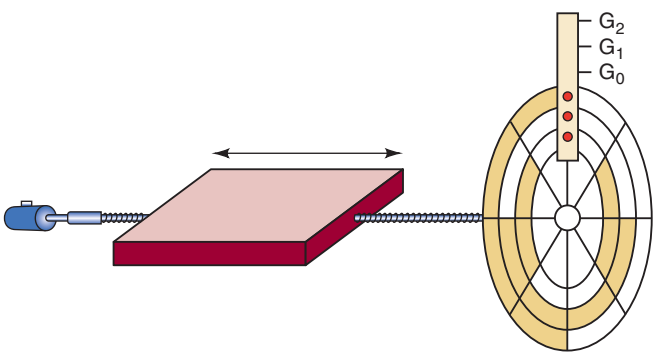


FIGURA 2-2 Conversión de (a) binario a Gray y de (b) Gray a binario.

La figura 2-2(b) muestra la conversión del código Gray a binario. Observe que el MSB en Gray siempre es el mismo que el MSB en binario. El siguiente bit binario se encuentra comparando el bit *binario* a la izquierda con el *bit correspondiente en código Gray*. Los bits similares producen un 0 y los bits distintos un 1. La aplicación más común del código Gray es en los codificadores de posición de eje, como muestra la figura 2-3. Estos dispositivos producen un valor binario que representa la posición de un eje mecánico giratorio. Un codificador de eje práctico utiliza mucho más de tres bits y divide la rotación en mucho más de ocho segmentos, por lo que puede detectar incrementos de rotación mucho más pequeños.

FIGURA 2-3 Un codificador de eje de ocho posiciones y tres bits.



PREGUNTAS DE REPASO

1. Convierta el número 0101 (binario) en su equivalente en código de Gray.
2. Convierta el número 0101 (código de Gray) en su número binario equivalente.

2-6 INTEGRACIÓN DE LOS SISTEMAS NUMÉRICOS

La tabla 2-3 muestra la representación de los números decimales del 1 al 15 en los sistemas numéricos binario y hexadecimal, y también en los códigos BCD y Gray. Examine esta tabla con cuidado y asegúrese de comprender de dónde proviene. Observe en especial cómo la representación en BCD siempre usa cuatro bits para cada dígito decimal.

TABLA 2-3

Decimal	Binario	Hexadecimal	BCD	GRAY
0	0	0	0000	0000
1	1	1	0001	0001
2	10	2	0010	0011
3	11	3	0011	0010
4	100	4	0100	0110
5	101	5	0101	0111
6	110	6	0110	0101
7	111	7	0111	0100
8	1000	8	1000	1100
9	1001	9	1001	1101
10	1010	A	0001 0000	1111
11	1011	B	0001 0001	1110
12	1100	C	0001 0010	1010
13	1101	D	0001 0011	1011
14	1110	E	0001 0100	1001
15	1111	F	0001 0101	1000

2-7 BYTE, NIBBLE Y PALABRA

Bytes

La mayoría de las microcomputadoras maneja y almacena datos binarios e información en grupos de ocho bits, por lo que una cadena de ocho bits tiene un nombre especial: **byte**. Un byte consiste de ocho bits y puede representar cualquier tipo de datos o de información. Los siguientes ejemplos ilustrarán este punto.

EJEMPLO 2-8

¿Cuántos bytes hay en una cadena de 32 bits?

Solución

32/8 = 4; por lo tanto, hay **cuatro** bytes en una cadena de 32 bits.

EJEMPLO 2-9

¿Cuál es el valor decimal más grande que puede representarse en binario si se utilizan dos bytes?

Solución

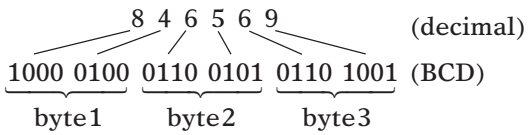
Dos bytes son 16 bits, por lo que el valor binario más grande será equivalente al número decimal $2^{16} - 1 = 65{,}535$.

EJEMPLO 2-10

¿Cuántos bytes se necesitan para representar el valor decimal 846,569 en BCD?

Solución

Cada dígito decimal se convierte en un código BCD de cuatro bits. Por ende, un número decimal de seis dígitos requiere 24 bits. Esos 24 bits corresponden a **tres** bytes. El diagrama para este caso se muestra a continuación:



Nibbles

A menudo los números binarios se descomponen en grupos de cuatro bits, como hemos visto con los códigos BCD y las conversiones a números hexadecimales. En los primeros días de los sistemas digitales surgió un término para describir un grupo de cuatro bits. Como abarca la mitad de un byte, se le denominó **nibble**. Los siguientes ejemplos ilustran el uso de este término.

EJEMPLO 2-11

¿Cuántos nibbles hay en un byte?

Solución

2

EJEMPLO 2-12

¿Cuál es el valor hexadecimal del nibble menos significativo del número binario 1001 0101?

Solución

1001 0101

El nibble menos significativo es 0101 = 5.

Palabras

Los términos bit, nibble y byte representan un número fijo de dígitos binarios. A medida que los sistemas han ido creciendo a través de los años, también ha crecido

su capacidad (¿apetito?) de manejar datos binarios. Una **palabra** es un grupo de bits que representa una cierta unidad de información. El tamaño de la palabra depende del tamaño de la ruta de datos en el sistema que utiliza la información. El **tamaño de palabra** puede definirse como el número de bits en la palabra binaria con el que opera un sistema digital. Por ejemplo, tal vez la computadora en su horno de microondas sólo pueda manejar un byte a la vez. Tiene un tamaño de palabra de ocho bits. Por otro lado, la computadora personal en su escritorio puede manejar ocho bytes a la vez, por lo que tiene un tamaño de palabra de 64 bits.

PREGUNTAS DE REPASO

1. ¿Cuántos bytes se necesitan para representar el número 235_{10} en binario?
2. ¿Cuál es el valor decimal más grande que puede representarse en BCD, si se utilizan dos bytes?
3. ¿Cuántos dígitos hexadecimales puede representar un nibble?
4. ¿Cuántos nibbles hay en un dígito BCD?

2-8 CÓDIGOS ALFANUMÉRICOS

Además de los datos numéricos, una computadora debe ser capaz de manejar información no numérica. En otras palabras, una computadora debe reconocer códigos que representen letras del alfabeto, signos de puntuación y otros caracteres especiales, además de los números. A estos códigos se les denomina **códigos alfanuméricos**. Un código alfanumérico completo debe incluir las 26 letras minúsculas, las 26 letras mayúsculas, los 10 dígitos numéricos, 7 signos de puntuación y de 20 a 40 caracteres adicionales, como +, /, #, %, *, y así sucesivamente. Podemos decir que un código alfanumérico representa a todos los caracteres y funciones que se encuentran en el teclado de una computadora.

Código ASCII

El código alfanumérico más utilizado es el **Código estándar estadounidense para el intercambio de información (ASCII)**. Este código es de siete bits, por lo cual tiene $2^7 = 128$ código posibles. Más que suficiente para representar todos los caracteres estándar del teclado, así como las funciones de control tales como retorno de carro (RETURN) y avance de línea (LINEFEED). La tabla 2-4 muestra un listado del código ASCII estándar de siete bits. La tabla proporciona los equivalentes en hexadecimal y decimal. Para obtener el código binario de siete bits para cada carácter hay que convertir el valor hexadecimal en binario.

EJEMPLO 2-13

Use la tabla 2-4 para encontrar el código ASCII de siete bits para el carácter de barra diagonal inversa (\).

Solución

El valor hexadecimal que aparece en la tabla 2-4 es 5C. Al traducir cada dígito hexadecimal en código binario de cuatro bits produce el valor 0101 1100. Los siete bits de menor peso representan el código ASCII para \, o 1011100.

TABLA 2-4 Código ASCII estándar.

Caracter	HEX	Decimal	Caracter	HEX	Decimal	Caracter	HEX	Decimal	Caracter	HEX	Decimal
NUL (nulo)	0	0	Espacio	20	32	@	40	64	`	60	96
Inicio del encabezado	1	1	!	21	33	A	41	65	a	61	97
Inicio del texto	2	2	“	22	34	B	42	66	b	62	98
Fin del texto	3	3	#	23	35	C	43	67	c	63	99
Fin de la transmisión	4	4	$	24	36	D	44	68	d	64	100
Consulta	5	5	%	25	37	E	45	69	e	65	101
Reconocimiento	6	6	&	26	38	F	46	70	f	66	102
Timbre	7	7	`	27	39	G	47	71	g	67	103
Retroceso	8	8	(	28	40	H	48	72	h	68	104
Tabulación horizontal	9	9	)	29	41	I	49	73	i	69	105
Avance de línea	A	10	*	2A	42	J	4A	74	j	6A	106
Tabulación vertical	B	11	+	2B	43	K	4B	75	k	6B	107
Avance de hoja de impresión	C	12	,	2C	44	L	4C	76	l	6C	108
Retorno de carro	D	13	-	2D	45	M	4D	77	m	6D	109
Mayúsculas desactivadas	E	14	.	2E	46	N	4E	78	n	6E	110
Mayúsculas activadas	F	15	/	2F	47	O	4F	79	o	6F	111
Escape de enlace de datos	10	16	0	30	48	P	50	80	p	70	112
Control directo 1	11	17	1	31	49	Q	51	81	q	71	113
Control directo 2	12	18	2	32	50	R	52	82	r	72	114
Control directo 3	13	19	3	33	51	S	53	83	s	73	115
Control directo 4	14	20	4	34	52	T	54	84	t	74	116
ACK (reconocimiento) negativo	15	21	5	35	53	U	55	85	u	75	117
Sincronía en estado inactivo	16	22	6	36	54	V	56	86	v	76	118
Fin de Bloque de Transmisión	17	23	7	37	55	W	57	87	w	77	119
Cancelar	18	24	8	38	56	X	58	88	x	78	120
Fin de medio	19	25	9	39	57	Y	59	89	y	79	121
Sustituir	1A	26	:	3A	58	Z	5A	90	z	7A	122
Escape	1B	27	;	3B	59	[	5B	91	{	7B	123
Separador de formas	1C	28	<	3C	60	\	5C	92	l	7C	124
Separador de grupos	1D	29	=	3D	61	]	5D	93	}	7D	125
Separador de registros	1E	30	>	3E	62	^	5E	94	~	7E	126
Separador de unidades	1F	31	?	3F	63		5F	95	Suprimir	7F	127

El código ASCII se utiliza para la transferencia de información alfanumérica entre una computadora y los dispositivos externos, tales como una impresora u otra computadora. La computadora también utiliza ASCII en forma interna para almacenar la información que escribe un operador en el teclado. El siguiente ejemplo ilustra lo anterior.

EJEMPLO 2-14

Un operador está escribiendo un programa en lenguaje C en el teclado de cierta microcomputadora, la cual convierte cada pulsación de tecla en su código ASCII, y lo almacena como un byte en memoria. Determine las cadenas binarias que se introducirán en memoria cuando el operador escriba la siguiente instrucción en C:

```
if (x>3)
```

Solución

Localice cada carácter (incluyendo el espacio) en la tabla 2-4 y escriba su código ASCII.

i	69	0110	1001
f	66	0110	0110
espacio	20	0010	0000
(	28	0010	1000
x	78	0111	1000
>	3E	0011	1110
3	33	0011	0011
)	29	0010	1001

Observe que se agregó un 0 al bit más a la izquierda de cada código ASCII, ya que los códigos deben almacenarse como bytes (ocho bits). A este proceso de agregar un bit adicional se le denomina *rellenar con 0s*.

PREGUNTAS DE REPASO

1. Codifique el siguiente mensaje en código ASCII utilizando la representación hexadecimal: "COSTO = $72."
2. El siguiente mensaje en código ASCII con bits de relleno se almacena en ubicaciones contiguas de memoria en una computadora:

   01010011 01010100 01001111 01010000

¿Cuál es el mensaje?

2-9 MÉTODO DE PARIDAD PARA LA DETECCIÓN DE ERRORES

El movimiento de datos binarios y códigos de un lugar a otro es la operación más frecuente que se realiza con los sistemas digitales. A continuación se listan solo unos cuantos ejemplos:

- La transmisión de voz digitalizada a través de un enlace de microondas.
- El almacenamiento de datos y la recuperación de los mismos desde dispositivos de memoria externos, como el disco magnético y el disco óptico.
- La transmisión de datos digitales desde una computadora hacia otra computadora remota, a través de líneas telefónicas (mediante el uso de un módem). Ésta es una de las principales formas de enviar y recibir información en Internet.

Siempre que se transmite información desde un dispositivo (el transmisor) hasta otro (el receptor), existe la posibilidad de que puedan producirse errores tales

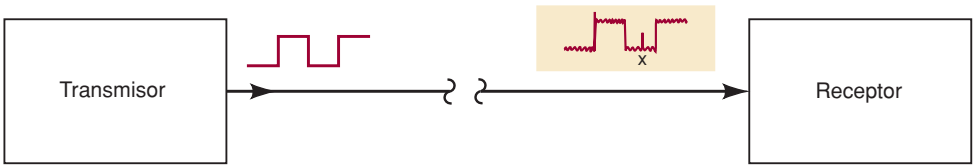


FIGURA 2-4 Ejemplo del ruido que produce un error en la transmisión de datos digitales.

que causen que el receptor no reciba la información idéntica tal y como la envió el transmisor. La principal causa de errores de transmisión es el *ruido eléctrico*, el cual consiste en fluctuaciones espurias en el voltaje o en la corriente, dichas señales de ruido están presentes en todos los sistemas electrónicos en diversos grados. La figura 2-4 es una ilustración simplificada de un tipo de error de transmisión.

El transmisor envía una señal digital en serie que está relativamente libre de ruido, a través de una línea de señal hacia un receptor. No obstante, para cuando la señal llega al receptor contiene un cierto grado de ruido acumulado a la señal original. En ocasiones, el ruido es tan grande en amplitud que altera el nivel lógico de la señal, como se ve en el punto *x*. Cuando esto ocurre, el receptor puede interpretar en forma incorrecta ese bit como un 1 lógico, que no corresponde con lo que el transmisor ha enviado.

La mayoría del equipo digital moderno está diseñado relativamente libre de errores, y la probabilidad de que ocurran errores como el que se muestra en la figura 2-4 es muy baja. No obstante, debemos considerar que los sistemas digitales frecuentemente transmiten miles, incluso millones de bits por segundo, por lo que incluso hasta una relación muy baja de ocurrencia de errores puede producir un error ocasional que podría llegar a ser molesto, si no es que desastroso. Por esta razón, muchos sistemas digitales emplean algún método para la detección (y en ocasiones corrección) de errores. Uno de los esquemas más simples y utilizados para este fin es el **método de paridad**.

Bit de paridad

Un **bit de paridad** es un bit que se agrega al grupo de bits del código que se está transfiriendo de un lugar a otro. El bit de paridad se hace 0 o 1, dependiendo del número de 1s que contenga el grupo de bits del código. Se utilizan dos métodos distintos.

En el método de *paridad par*, el valor del bit de paridad se elije de manera que el número total de 1s en el grupo de bits del código, incluyendo el bit de paridad, sea *par*. Por ejemplo, suponga que el grupo es 1000011. Éste es el carácter "C" en ASCII. El código tiene *tres* 1s. Por ende, agregaremos un bit de paridad de 1 para que el número total de 1s sea par. El *nuevo* código *incluyendo el bit de paridad* sería entonces:

1 1 0 0 0 0 1 1
↑ bit de paridad agregado*

Si el grupo de bits del código contiene un número par de 1s para empezar, el bit de paridad recibe un valor de 0. Por ejemplo, si el código fuera 1000001 (el código ASCII para la "A"), el bit de paridad asignado sería 0 y el nuevo código *incluyendo el bit de paridad* sería 01000001.

El método de *paridad impar* se utiliza de la misma forma, sólo que el bit de paridad se elije de manera que el número total de 1s, incluyendo el bit de paridad, sea *impar*. Por ejemplo, para el código 1000001, el bit de paridad asignado sería un 1. Para el código 1000011, el bit de paridad sería un 0.

* El bit de paridad puede colocarse en cualquier extremo del grupo de código, pero, por lo general, se coloca a la izquierda del MSB.

Ya sea que se utilice la paridad par o impar, el bit de paridad se convierte en parte de la palabra de código. Por ejemplo, al agregar un bit de paridad al código ASCII de siete bits se produce un código de ocho bits. Por lo tanto, el bit de paridad se trata justo igual que cualquier otro bit en el código.

El bit de paridad se transmite para detectar cualquier error de *un solo bit* que ocurra durante la transmisión de un código de un lugar a otro. Por ejemplo, suponga que el carácter "A" se va a transmitir y se va a utilizar paridad *impar*. El código transmitido sería

1 1 0 0 0 0 0 1

Cuando el circuito receptor reciba el código, verificará que éste contenga un número impar de 1s, incluyendo el bit de paridad. De ser así, el receptor supondrá que el código se ha recibido de manera correcta. Ahora suponga que debido a cierto ruido o falla el receptor en realidad recibe el siguiente código:

1 1 0 0 0 0 0 0

El receptor descubrirá que este código tiene un número *par* de 1s. Esto indica al receptor que debe haber un error en el código, ya que se presume que el transmisor y el receptor han acordado utilizar paridad impar. Sin embargo, no hay forma de que el receptor sepa cuál bit tiene error, ya que no sabe cuál se supone que va a ser el código.

Podemos asumir que este método de paridad no funciona si *dos* bits tienen error, ya que dos errores no cambiarían la característica de "par" o "impar" en el número de 1s en el código. En la práctica, el método de paridad se utiliza sólo en situaciones en las que la probabilidad de un solo error es muy baja y la probabilidad de doble error es prácticamente cero.

Cuando se utiliza el método de paridad, el transmisor y el receptor deben acordar antes de la transmisión si se va a utilizar la paridad par o impar. No hay ventaja de un método sobre el otro, aunque parece que la paridad par se utiliza más a menudo. El transmisor debe agregar el bit de paridad apropiado a cada unidad de información que transmita. Por ejemplo, si el transmisor está enviando datos codificados en ASCII, deberá agregar el bit de paridad a cada grupo de código ASCII de siete bits. Cuando el receptor examine los datos que reciba del transmisor, comprobará cada grupo de código para ver si el número total de 1s, incluyendo el bit de paridad, es consistente con el tipo de paridad acordado. A esto se le conoce comúnmente como *comprobar la paridad* de los datos. En caso de que detecte un error, el receptor puede enviar un mensaje al transmisor para pedirle que vuelva a transmitir el último conjunto de datos. El procedimiento a seguir cuando se detecta un error depende de cada sistema.

EJEMPLO 2-15

Es común que las computadoras se comuniquen con otros equipos remotos a través de líneas telefónicas. Así es como se lleva a cabo la comunicación por acceso telefónico a través de Internet. Cuando una computadora transmite un mensaje a otra, esa información, por lo general, se codifica en ASCII. ¿Cuáles serían las cadenas de bits que transmitiría una computadora para enviar el mensaje HOLA, utilizando ASCII con paridad par?

Solución

Primero, vea los códigos ASCII para cada carácter del mensaje. Después, para cada código cuente el número de 1s. Si es un número par añada un 0 como el MSB, o un 1

si es un número impar. Por ende, los códigos de ocho bits (bytes) resultantes tendrán un número par de 1s (incluyendo la paridad).

		bits de paridad agregados ↓							
H	=	0	1	0	0	1	0	0	0
O	=	1	1	0	0	0	1	0	1
L	=	1	1	0	0	1	1	0	0
A	=	0	1	0	0	0	0	0	1

PREGUNTAS DE REPASO

1. Agregue un bit de paridad impar al código ASCII para el símbolo $ y exprese el resultado en hexadecimal.
2. Adjunte un bit de paridad par al código BCD para el 69 decimal.
3. ¿Por qué el método de paridad no puede detectar un doble error en los datos transmitidos?

2-10 APLICACIONES

A continuación se presentan algunas aplicaciones que servirán como repaso de algunos conceptos que vistos en este capítulo. El objetivo de estas aplicaciones es que usted se dé una idea de cómo se utilizan los diversos sistemas y códigos numéricos en el mundo digital. En los problemas al final del capítulo presentaremos más aplicaciones.

APLICACIÓN 2-1

Un CD-ROM ordinario puede almacenar 650 megabytes de datos digitales. Como mega = 2^{20}, ¿cuántos bits de datos puede almacenar un CD-ROM?

Solución

Recuerde que un byte tiene ocho bits. Por lo tanto, 650 megabytes son $650 \times 2^{20} \times 8 =$ **5,452,595,200 bits**.

APLICACIÓN 2-2

Para poder programar la mayoría de microcontroladores, las instrucciones binarias se almacenan en un archivo de una computadora personal, de una manera especial que se conoce como Formato Hexadecimal Intel. La información hexadecimal se codifica en caracteres ASCII, de manera que pueda mostrarse con facilidad en la pantalla de una PC, imprimirse o transmitirse de una manera sencilla, carácter por carácter, a través de un puerto COM serial de la PC. A continuación se muestra una línea de un archivo en Formato Hexadecimal Intel:

:10002000F7CFFFCF1FEF2FEF2A95F1F71A95D9F7EA

El primer carácter enviado es el código ASCII correspondiente a los dos puntos, seguido de un 1. A cada uno se le adjunta un bit de paridad como el bit más significativo. Un instrumento de prueba captura el patrón binario de bits a medida que pasa a través del cable hacia el microcontrolador.

(a) ¿Qué apariencia debe tener el patrón binario de bits incluyendo la paridad? (MSB-LSB.)

(b) El valor 10, que va después de los dos puntos, representa en número hexadecimal el total de bytes que se van a cargar en la memoria del microcontrolador. ¿Cuál es el número decimal de bytes que se van a cargar?

(c) El número 0020 es un valor hexadecimal de cuatro dígitos que representa la dirección en la que se va a almacenar el primer byte. ¿Cuál es la mayor dirección posible? ¿Cuántos bits se requerirían para representar esta dirección?

(d) El valor del primer byte de datos es F7. ¿Cuál es el valor (en binario) del nibble menos significativo de este byte?

FFFF 1111 1111 1111 1111 16 bits

Solución

(a) Los códigos ASCII son 3A (para el :) y 31 (para el 1) 00111010 10110001
bit de paridad par

(b) 10 hex = $1 \times 16 + 0 \times 1 = 16$ bytes decimales.

(c) FFFF es el mayor valor posible. Cada dígito hexadecimal es de 4 bits, por lo que necesitamos 16 bits.

(d) El nibble menos significativo (4 bits) se representa mediante el 7 hexadecimal. En binario sería 0111.

APLICACIÓN 2-3

Una pequeña computadora de control de procesos utiliza códigos hexadecimales para representar sus direcciones de 16 bits de memoria.

(a) ¿Cuántos dígitos hexadecimales se requieren?

(b) ¿Cuál es el intervalo de direcciones en hexadecimal?

(c) ¿Cuántas localidades de memoria hay?

Solución

(a) Como 4 bits se convierten en un dígito hexadecimal, se necesitan 16/4 = 4 dígitos hexadecimales.

(b) El intervalo binario es de 0000000000000000_2 a 1111111111111111_2. En hexadecimal sería de 0000_{16} a $FFFF_{16}$.

(c) Con 4 dígitos hexadecimales, el número total de direcciones es $16^4 = \mathbf{65{,}536}$.

APLICACIÓN 2-4

En un sistema basado en microcontrolador, los números se introducen en BCD pero se almacenan en binario directo. Como programador, usted debe decidir si necesita una ubicación de almacenamiento de un byte o de dos bytes.

(a) ¿Cuántos bytes necesita si el sistema recibe una entrada decimal de dos dígitos?

(b) ¿Qué pasaría si tuviera que introducir tres dígitos?

Solución

(a) Con dos dígitos puede introducir valores hasta el 99 ($1001\ 1001_{BCD}$). En binario este valor es 01100011, el cual cabe en una ubicación de memoria de ocho bits. También puede utilizar un solo bit.

(b) Tres dígitos pueden representar hasta 999 (1001 1001 1001). En binario este valor es 1111100111 (10 bits). Por lo tanto, no puede usar un solo byte; necesita dos bytes.

APLICACIÓN 2-5

Cuando hay que transmitir caracteres ASCII entre dos sistemas independientes (como una computadora y un módem), debe haber una manera de indicar al receptor cuándo va a llegar un nuevo carácter. A menudo también se tiene la necesidad de detectar errores en la transmisión. El método de transferencia se llama comunicación asíncrona de datos. El estado normal de inactividad de la línea de transmisión es un 1 lógico. Cuando el transmisor envía un carácter ASCII, debe "encapsularse" para que el receptor sepa en dónde comienzan y terminan los datos. El primer bit debe ser siempre un bit de inicio (0 lógico). A continuación se envía el código ASCII, en donde el LSB va primero y el MSB al último. Después del MSB se adjunta un bit de paridad para comprobar errores en la transmisión. Para terminar la transmisión se envía un bit de paro (1 lógico). En la figura 2-5 se muestra una transmisión asíncrona ordinaria del código ASCII de siete bits para el signo # (23 Hex) con paridad par.

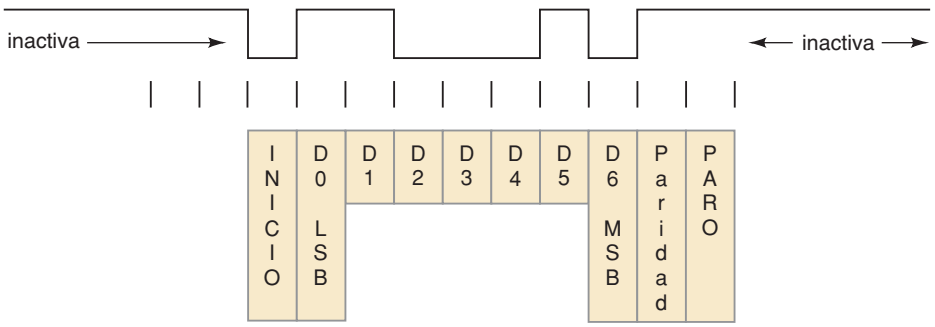


FIGURA 2-5 Datos seriales asíncronos con paridad par.

RESUMEN

1. El sistema numérico hexadecimal se utiliza en los sistemas y las computadoras digitales como una manera eficiente de representar cantidades binarias.
2. En las conversiones entre hexadecimal y binario, cada dígito hexadecimal corresponde a cuatro bits.
3. El método de división repetida se utiliza para convertir números decimales en binario o hexadecimal.
4. Mediante el uso de un número binario de N bits podemos representar valores decimales desde el 0 hasta $2^N - 1$.
5. Para formar el código BCD de un número decimal se convierte cada dígito del número decimal en su equivalente binario de cuatro bits.
6. El código Gray define una secuencia de patrones de bits, en los que sólo un bit cambia entre patrones sucesivos en la secuencia.
7. Un byte es una cadena de ocho bits. Un nibble es de cuatro bits. El tamaño de una palabra depende del sistema.
8. Un código alfanumérico utiliza grupos de bits para representar a todos los caracteres y funciones que forman parte del teclado ordinario de una computadora. El código ASCII es el código alfanumérico más utilizado.
9. En el método de paridad para la detección de errores se adjunta un bit de paridad especial a cada grupo de bits que se transmite.

TÉRMINOS IMPORTANTES

bit de paridad
byte
códigos alfanuméricos
código binario directo
código decimal codificado en binario (BCD)
Código estándar estadounidense para el intercambio de información (ASCII)
código Gray
método de paridad
nibble
palabra
sistema numérico hexadecimal
tamaño de palabra

PROBLEMAS

SECCIONES 2-1 Y 2-2

2-1. Convierta los siguientes números binarios en decimales.

(a)* 10110 (d) 01101011 (g)* 1111010111
(b) 10010101 (e)* 11111111 (h) 11011111
(c)* 100100001001 (f) 01101111

2-2. Convierta los siguientes valores decimales en binarios.

(a)* 37 (d) 1000 (g)* 205
(b) 13 (e)* 77 (h) 2133
(c)* 189 (f) 390 (i)* 511

2-3. ¿Cuál es el valor decimal más grande que puede representarse mediante (a)* un número binario de ocho bits? (b) un número de 16 bits?

SECCIÓN 2-4

2-4. Convierta cada número hexadecimal en su equivalente decimal.

(a)* 743 (d) 2000 (g)* 7FF
(b) 36 (e)* 165 (h) 1204
(c)* 37FD (f) ABCD

2-5. Convierta cada uno de los siguientes números decimales en hexadecimales.

(a)* 59 (d) 1024 (g)* 65,536
(b) 372 (e)* 771 (h) 255
(c)* 919 (f) 2313

2-6. Convierta cada uno de los valores hexadecimales del problema 2-4 en números binarios.

2-7. Convierta los números binarios del problema 2-1 en hexadecimales.

2-8. Liste los números hexadecimales en secuencia, desde 195_{16} hasta 280_{16}.

2-9. Cuando se va a convertir un número decimal grande en binario, algunas veces es más fácil convertirlo primero en hexadecimal y después en binario. Pruebe este procedimiento para el número 2133_{10} y compárelo con el procedimiento usado en el problema 2-2(h).

2-10. ¿Cuántos dígitos hexadecimales se requieren para representar los números decimales del 0 hasta el 20,000?

2-11. Convierta los siguientes valores hexadecimales en decimales.

(a)* 92 (d) ABCD (g)* 2C0
(b) 1A6 (e)* 000F (h) 7FF
(c)* 37FD (f) 55

* Encontrará las respuestas a los problemas marcados con un asterisco al final del libro.

2-12. Convierta los siguientes valores decimales en hexadecimales.

(a)* 75	(d) 24	(g)* 25,619
(b) 314	(e)* 7245	(h) 4095
(c)* 2048	(f) 498	

2-13. Tome cada número binario de cuatro bits en el orden en el que están escritos y escriba el dígito hexadecimal equivalente sin realizar ningún cálculo manual o mediante la calculadora.

(a) 1001	(e) 1111	(i) 1011	(m) 0001
(b) 1101	(f) 0010	(j) 1100	(n) 0101
(c) 1000	(g) 1010	(k) 0011	(o) 0111
(d) 0000	(h) 1001	(l) 0100	(p) 0110

2-14. Tome cada dígito hexadecimal y escriba su valor binario de cuatro bits sin realizar ningún cálculo manual ni mediante la calculadora.

(a) 6	(e) 4	(i) 9	(m) 0
(b) 7	(f) 3	(j) A	(n) 8
(c) 5	(g) C	(k) 2	(o) D
(d) 1	(h) B	(l) F	(p) 9

2-15.* Convierta los números binarios del problema 2-1 en hexadecimales.

2-16.* Convierta los valores hexadecimales del problema 2-11 en binarios.

2-17.* Liste los números hexadecimales en secuencia, desde 280 hasta 2A0.

2-18. ¿Cuántos dígitos hexadecimales se requieren para representar números decimales hasta 1 millón?

SECCIÓN 2-5

2-19. Codifique los siguientes números decimales en BCD.

(a)* 47	(d) 6727	(g)* 89,627
(b) 962	(e)* 13	(h) 1024
(c)* 187	(f) 529	

2-20. ¿Cuántos bits se requieren para representar los números decimales en el intervalo de 0 a 999 si se utiliza: (a) código binario directo, y (b) código BCD?

2-21. Los siguientes números están en BCD. Conviértalos en decimales.

(a)* 1001011101010010	(d) 0111011101110101
(b) 000110000100	(e)* 010010010010
(c)* 011010010101	(f) 010101010101

SECCIÓN 2-7

2-22.* (a) ¿Cuántos bits hay en ocho bytes?

(b) ¿Cuál es el número hexadecimal más grande que puede representarse en cuatro bytes?

(c) ¿Cuál es el valor decimal codificado en BCD más grande que puede representarse en tres bytes?

2-23. (a) Consulte la tabla 2-4. ¿Cuál es el nibble más significativo del código ASCII para la letra X?

(b) ¿Cuántos nibbles pueden almacenarse en una palabra de 16 bits?

(c) ¿Cuántos bytes se requieren para formar una palabra de 24 bits?

SECCIONES 2-8 Y 2-9

2-24. Represente la instrucción "X = 3 × Y" en código ASCII. Adjunte un bit de paridad impar.

2-25.* Adjunte un bit de paridad *par* a cada uno de los códigos ASCII del problema 2-24, y muestre los resultados en hexadecimal.

2-26. Los siguientes bytes (mostrados en hexadecimal) representan el nombre de una persona según como se almacenaría en la memoria de una computadora. Cada byte es código ASCII con relleno. Determine el nombre de cada persona.

(a)* 42 45 4E 20 53 4D 49 54 48

(b) 4A 6F 65 20 47 72 65 65 6E

2-27. Convierta los siguientes números decimales en código BCD y después adjunte un bit de paridad *impar*.

(a)* 74 (c)* 8884 (e)* 165

(b) 38 (d) 275 (f) 9201

2-28.* En cierto sistema digital, los números decimales del 000 al 999 se representan en código BCD. También se incluye un bit de paridad *impar* al final de cada grupo. Examine cada uno de los códigos que se muestran a continuación y suponga que cada uno acaba de transferirse de un lugar a otro. Algunos de los grupos contienen errores. Suponga que *no se han producido más de dos errores* en cada grupo. Determine cuáles de los siguientes casos tienen un solo error y cuáles tienen *en definitiva* un error doble. (*Sugerencia:* recuerde que éste es código BCD).

(a) 1001010110000



(b) 0100011101100

(c) 0111110000011

(d) 1000011000101

2-29. Suponga que el receptor recibió los siguientes datos del transmisor del ejemplo 2-16:

0 1 0 0 1 0 0 0
1 1 0 0 0 1 0 1
1 1 0 0 1 1 0 0
1 1 0 0 1 0 0 0
1 1 0 0 1 1 0 0

¿Qué errores puede determinar el receptor en los datos que recibió?

PREGUNTAS DE PRÁCTICA

2-30.* Realice cada una de las siguientes conversiones. Si desea puede probar varios métodos en algunas de ellas para ver con cuál se adapta mejor. Por ejemplo, una conversión de binario a decimal puede realizarse en forma directa, o también mediante una conversión de binario a hexadecimal seguida de una conversión de hexadecimal a decimal.

(a) 1417_{10} = _____$_2$

(b) 255_{10} = _____$_2$

(c) 11010001_2 = _____$_{10}$

(d) 1110101000100111_2 = _____$_{10}$

(e) 2497_{10} = _____16

(f) 511_{10} = _____ (BCD)

(g) 235_{16} = _____10

(h) 4316_{10} = _____16

(i) $7A9_{16}$ = _____10

(j) $3E1C_{16}$ = _____10

(k) 1600_{10} = _____16

(l) $38{,}187_{10}$ = _____16

(m) 865_{10} = _____ (BCD)

(n) 100101000111 (BCD) = _____10

(o) 465_{16} = _____2

(p) $B34_{16}$ = _____2

(q) 01110100 (BCD) = _____2

(r) 111010_2 = _____ (BCD)

2-31.* Represente el valor decimal 37 en cada una de las siguientes formas.

(a) Binario directo.

(b) BCD.

(c) Hexadecimal.

(d) ASCII (es decir, trate cada dígito como un carácter).

2-32.* Llene los espacios en blanco con la palabra o palabras correctas.

(a) Para convertir de decimal a _____ se requiere de la división repetida entre 16.

(b) Para convertir de decimal a binario se requiere de la división repetida entre _____.

(c) En el código BCD, cada _____ se convierte en su equivalente binario de cuatro bits.

(d) El código _____ tiene la característica de que sólo cambia un bit al avanzar de un paso al siguiente.

(e) Un transmisor adjunta un _____ a un código para permitir que el receptor detecte _____.

(f) El código _____ es código alfanumérico más común que se utiliza en los sistemas computacionales.

(g) _____ se utiliza a menudo como una manera conveniente de representar números binarios extensos.

(h) Una cadena de ocho bits se llama _____.

2-33. Escriba el número binario que se produce cuando cada uno de los siguientes números se incrementa en uno.

(a)* 0111 (b) 010011 (c) 1011

2-34. Decremente cada uno de los siguientes números binarios.

(a)* 1110 (b) 101000 (c) 1110

2-35. Escriba el número que se produce cuando se incrementa cada una de las siguientes cifras.

(a)* 7779_{16} (c)* $0FFF_{16}$ (e)* $9FF_{16}$

(b) 9999_{16} (d) 2000_{16} (f) $100A_{16}$

2-36.* Repita el problema 2-35 para la operación de decremento.

EJERCICIOS AVANZADOS

2-37.* En una microcomputadora, las *direcciones* de las localidades de memoria son números binarios que identifican cada uno de los circuitos de memoria en donde se almacena un byte. El número de bits que forman cada dirección depende de cuántas localidades de memoria haya. Como el número de bits puede ser muy extenso, a menudo las direcciones se especifican en hexadecimal, en lugar de binario.

(a) Si una microcomputadora utiliza una dirección de 20 bits, ¿cuántas localidades de memoria distintas hay?

(b) ¿Cuántos dígitos hexadecimales se necesitan para representar la dirección de una localidad de memoria?

(c) ¿Cuál es la dirección hexadecimal de la localidad de memoria número 256? (*Nota:* la primera dirección siempre es 0.)

2-38. En un CD de audio, la señal de voltaje de audio, por lo general, se muestrea aproximadamente 44,000 veces por segundo, y el valor de cada muestra se graba en la superficie del CD como número binario. En otras palabras, cada número binario que se graba representa un punto de voltaje individual en la forma de onda de la señal de audio.

(a) Si los números binarios tienen una longitud de seis bits, ¿cuántos valores de voltaje distintos pueden representarse mediante un solo número binario? Repita para ocho y diez bits.

(b) Si se utilizan números de diez bits, ¿cuántos bits se grabarán en el CD en un segundo?

(c) Si un CD puede almacenar, por lo general, 5 mil millones de bits, cuántos segundos de audio pueden grabarse si se utilizan diez bits?

2-39.* Una cámara digital en blanco y negro coloca una rejilla fina sobre una imagen para después medir y registrar un número binario que representa el nivel de gris que ve en cada celda de la rejilla. Por ejemplo, si se utilizan números de cuatro bits el valor de negro se establece en 0000 y el valor de blanco en 1111, y cualquier nivel de gris puede tener algún valor entre 0000 y 1111. Si se utilizan números de seis bits, el negro es 000000 y el blanco es 111111, y todos los grises se encuentran entre estos dos valores.

Suponga que queremos diferenciar entre 254 niveles de gris dentro de cada una de las celdas de la rejilla. ¿Cuántos bits necesitaríamos usar para representar estos niveles de gris?

2-40. Una cámara digital de 3 megapixeles almacena un número de ocho bits para el brillo de cada uno de los colores primarios (rojo, verde, azul) que se encuentran en cada elemento de imagen (pixel). Si se almacenan todos los bits, sin compresión de datos, ¿cuántas imágenes pueden almacenarse en una tarjeta de memoria de 128 Megabytes? (Nota: en los sistemas digitales, Mega significa 2^{20}).

2-41. Construya una tabla que muestre las representaciones en binario, hexadecimal y BCD de todos los números decimales del 0 al 15. Compare sus resultados con la tabla 2-3.

RESPUESTAS A LAS PREGUNTAS DE REPASO DE LAS SECCIONES

SECCIÓN 2-1

1. 2267 2. 32768

SECCIÓN 2-2

1. 1010011 2. 1011011001 3. 20 bits

SECCIÓN 2-3

1. 9422 2. C2D; 110000101101 3. 97B5 4. E9E, E9F, EA0, EA1
5. 11010100100111 6. 0 a 65,535

SECCIÓN 2-4

1. 101100109_2; 000101111000 (BCD) 2. 32 3. Ventaja: facilidad de conversión. Desventaja: el código BCD requiere más bits.

SECCIÓN 2-5

1. 0111 2. 0110

SECCIÓN 2-7

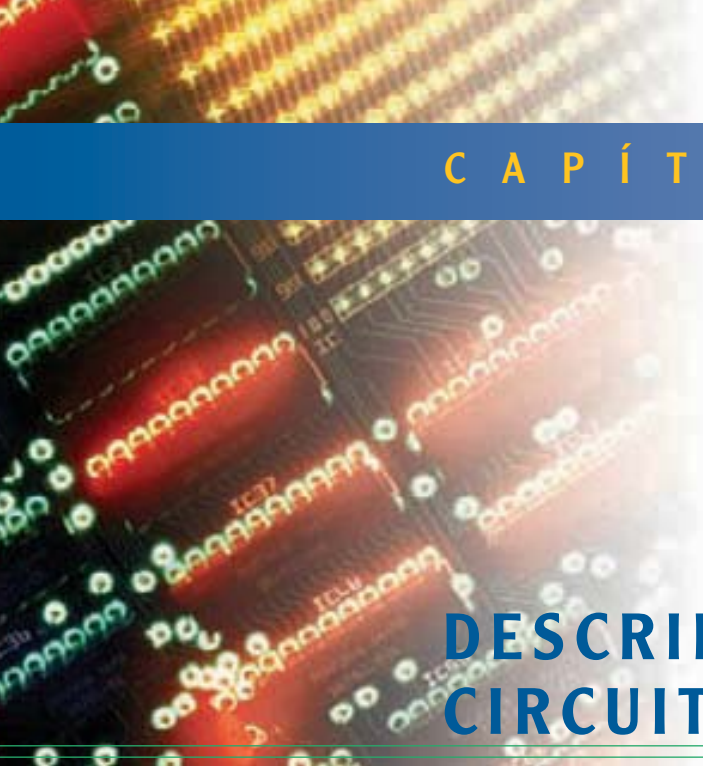
1. Uno 2. 9999 3. Uno 4. Uno

SECCIÓN 2-8

1. 43, 4F, 53, 54, 20, 3D, 20, 24, 37, 32 2. STOP

SECCIÓN 2-9

1. A4 2. 001101001 3. Dos errores en los datos no cambiarían la característica impar o par del número de 1s en los datos.



CAPÍTULO 3

DESCRIPCIÓN DE LOS CIRCUITOS LÓGICOS

CONTENIDO

3-1 Constantes y variables booleanas
3-2 Tablas de verdad
3-3 Operación OR con compuertas OR
3-4 Operación AND con compuertas AND
3-5 Operación NOT
3-6 Descripción de circuitos lógicos en forma algebraica
3-7 Evaluación de las salidas de circuitos lógicos
3-8 Implementación de circuitos a partir de expresiones booleanas
3-9 Compuertas NOR y NAND
3-10 Teoremas booleanos
3-11 Teoremas de DeMorgan
3-12 Universalidad de las compuertas NAND y NOR
3-13 Representaciones alternas de compuertas lógicas
3-14 Cuál representación de compuerta se debe usar
3-15 Símbolos lógicos del estándar IEEE/ANSI
3-16 Resumen de los métodos para describir circuitos lógicos
3-17 Comparación entre lenguajes de descripción y lenguajes de programación
3-18 Implementación de circuitos lógicos con PLDs
3-19 Formato y sintaxis del HDL
3-20 Señales intermedias

OBJETIVOS

Al terminar este capítulo, usted podrá:

- Realizar las tres operaciones lógicas básicas.
- Describir la operación y construir las tablas de verdad para las compuertas AND, NAND, OR y NOR, y el circuito NOT (INVERSOR).
- Dibujar diagramas de tiempos para las diversas compuertas de los circuitos lógicos.
- Escribir la expresión booleana para las compuertas lógicas y combinaciones de compuertas lógicas.
- Implementar circuitos lógicos mediante el uso de compuertas AND, OR y NOT.
- Apreciar el potencial del álgebra booleana para simplificar circuitos lógicos complejos.
- Usar los teoremas de DeMorgan para simplificar las expresiones lógicas.
- Usar cualquiera de las compuertas universales (NAND y NOR) para implementar un circuito representado por una expresión booleana.
- Explicar las ventajas de construir un diagrama de circuitos lógicos mediante el uso de los símbolos alternos de las compuertas, en comparación con los símbolos estándar de las compuertas lógicas.
- Describir el concepto de las señales lógicas, activa en BAJO y activa en ALTO.
- Dibujar e interpretar los símbolos de las compuertas lógicas del estándar IEEE/ANSI.
- Usar varios métodos para describir la operación de los circuitos lógicos.
- Interpretar circuitos simples definidos mediante un lenguaje de descripción de hardware (HDL).
- Explicar la diferencia entre un HDL y un lenguaje de programación de computadoras.
- Crear un archivo de HDL para una compuerta lógica simple.
- Crear un archivo de HDL para circuitos combinacionales con variables intermedias.

INTRODUCCIÓN

En los capítulos 1 y 2 presentamos los conceptos de niveles lógicos y circuitos lógicos. En circuitos lógicos sólo existen dos posibles condiciones para cualquier entrada o salida: verdadero o falso. El sistema numérico binario utiliza sólo dos dígitos: 1 y 0, por lo que es perfecto para representar relaciones lógicas. Los circuitos lógicos digitales utilizan intervalos de voltaje predefinidos para representar estos estados binarios. Si utilizamos estos conceptos podemos crear circuitos compuestos de algo más que arena de playa procesada y alambre, con lo cual se pueden realizar decisiones lógicas consistentes e inteligentes. Es muy importante que tengamos un método para describir las decisiones lógicas que realizan estos circuitos. En otras palabras, debemos describir la forma en que operan. En este capítulo estudiaremos muchas formas para describir su operación.

Cada uno de estos métodos de descripción es importante, ya que todos los métodos aparecen muchas veces en la literatura técnica y la documentación de sistemas, y se utilizan en conjunto con las herramientas modernas de diseño y desarrollo.

La vida está llena de ejemplos de casos que se encuentran en un estado o en otro. Por ejemplo, una criatura puede estar viva o muerta, una luz puede estar encendida o apagada, una puerta puede estar cerrada o abierta, y puede estar lloviendo o no. En 1854, un matemático llamado George boole escribió el libro *Una investigación sobre las leyes del pensamiento*, en donde describía la manera en que realizamos decisiones lógicas con base en circunstancias verdaderas o falsas. Los métodos que describió se conocen actualmente como lógica booleana, y el sistema de usar símbolos y operadores para describir estas decisiones se llama álgebra booleana. De la misma forma que utilizamos símbolos tales como x y y para representar valores numéricos desconocidos en el álgebra tradicional, el álgebra booleana utiliza símbolos para representar una expresión lógica que tiene uno de dos valores posibles: verdadero o falso. La expresión lógica podría ser *puerta está cerrada*, *botón está oprimido*, o *nivel de combustible está bajo*. Es muy tedioso escribir estas expresiones, por lo que se acostumbra sustituirlas por símbolos tales como A, B y C.

El propósito principal de estas expresiones lógicas es describir la relación entre la salida de un circuito lógico (la decisión) y sus entradas (las condiciones). En este capítulo estudiaremos los circuitos lógicos más básicos (*compuertas lógicas*), que son los elementos a partir de los cuales se construyen todos los demás circuitos lógicos y sistemas digitales. Veremos cómo la operación de las distintas compuertas lógicas y los circuitos más complejos formados por combinaciones de compuertas lógicas pueden describirse y analizarse mediante el uso del álgebra booleana. También veremos un breve adelanto sobre cómo puede usarse dicha álgebra para simplificar una expresión booleana de un circuito determinado, de manera que éste pueda reconstruirse con menos compuertas lógicas y/o menos conexiones. En el capítulo 4 veremos mucho más acerca de la simplificación de circuitos.

El álgebra booleana no sólo se utiliza como herramienta para analizar y simplificar sistemas lógicos. También puede usarse como herramienta para crear un circuito lógico con una relación de entrada/salida deseada. Por lo general, este proceso se conoce como síntesis de circuitos lógicos, en vez de análisis. Se han utilizado otras técnicas en el análisis, síntesis y documentación de sistemas y circuitos lógicos, como tablas de verdad, símbolos esquemáticos, diagramas de tiempos y (por último, pero no menos importante) el lenguaje. Para categorizar estos métodos, podríamos decir que el álgebra booleana es una herramienta matemática, las tablas de verdad son herramientas para organización de datos, los símbolos esquemáticos son herramientas de representación gráfica, los diagramas de tiempos son herramientas para graficar la respuesta y el lenguaje es la herramienta universal de descripción.

Actualmente puede usarse cualquiera de estas herramientas para proporcionar la entrada a una computadora. Éstas pueden utilizarse para simplificar y traducir entre estas diversas formas de descripción y, en última instancia, proveer una salida en la forma necesaria para implementar un sistema digital. Para aprovechar los poderosos beneficios del software computacional, primero debemos comprender por completo las formas aceptables para describir estos sistemas en términos que la computadora pueda entender. En este capítulo veremos las bases para un estudio más detallado de estas herramientas vitales para la síntesis y el análisis de los sistemas digitales.

Es evidente que las herramientas que describiremos aquí son invaluables para describir, analizar, diseñar e implementar circuitos digitales. El estudiante que tenga la expectativa de trabajar en el campo digital deberá esforzarse mucho por comprender y familiarizarse con el álgebra booleana (aunque no lo crea, es mucho más sencilla que el álgebra convencional) y con todas las demás herramientas. Resuelva *todos* los ejemplos, ejercicios y problemas, incluso los que no le asigne su instructor. Cuando acabe con todos ellos, haga los suyos. Bien valdrá la pena el tiempo que invierta en ello, ya que usted verá cómo mejoran sus habilidades y aumenta su confianza.

3-1 CONSTANTES Y VARIABLES BOOLEANAS

El álgebra booleana difiere en gran medida del álgebra ordinaria, ya que a las constantes y variables booleanas sólo se les permite tener dos valores posibles: 0 y 1. Una variable booleana es una cantidad que puede ser (en distintas ocasiones) igual a 0 o a 1. Las variables booleanas se utilizan a menudo para representar el nivel de voltaje presente en un alambre o en las terminales de entrada/salida de un circuito. Por ejemplo, en cierto sistema digital el valor booleano 0 podría asignarse a cualquier voltaje en el intervalo de 0 a 0.8 V, mientras que el valor booleano 1 podría asignarse a cualquier voltaje entre 2 y 5 V.*

Por lo tanto, el 0 y el 1 booleanos no representan números reales, sino el estado de una variable de voltaje, o lo que se conoce como su **nivel lógico**. Se dice que un voltaje en un circuito digital está en el nivel 0 lógico o en el nivel 1 lógico, dependiendo de su valor numérico actual. En la lógica digital se utilizan otros términos más como sinónimos de 0 y 1. La tabla 3-1 muestra algunos de los más comunes. La mayor parte del tiempo utilizaremos las designaciones 0/1 y BAJO/ALTO.

TABLA 3-1

0 lógico	1 lógico
Falso	Verdadero
Apagado	Encendido
Bajo	Alto
No	Sí
Interruptor abierto	Interruptor cerrado

Como dijimos en la introducción, el **álgebra booleana** es el medio para expresar la relación entre las entradas y las salidas de un circuito lógico. Las entradas se consideran variables lógicas cuyos niveles lógicos en cualquier momento determinan los niveles de salida. En todo el trabajo que veremos utilizaremos símbolos de letras para representar variables lógicas. Por ejemplo, la letra A podría representar una cierta entrada o salida de un circuito digital, y en un determinado momento debemos tener $A = 0$ o $A = 1$; alguno de los dos estados.

Como sólo dos valores son posibles, en realidad es muy sencillo trabajar con el álgebra booleana en comparación con el álgebra ordinaria. En el álgebra booleana no hay fracciones, decimales, números negativos, raíces cuadradas, raíces cúbicas, logaritmos, números imaginarios, etc. De hecho, en el álgebra booleana sólo hay *tres* operaciones básicas: *OR*, *AND* y *NOT*.

A estas operaciones básicas se les conoce como *operaciones lógicas*. Los circuitos digitales, llamados *compuertas lógicas,* pueden construirse a partir de diodos, transistores y resistencias conectados de manera que la salida del circuito sea el resultado de una operación lógica básica (*OR*, *AND*, *NOT*) que se lleva a cabo con las entradas. Utilizaremos primero el álgebra booleana para describir y analizar las compuertas lógicas básicas, y después para analizar y diseñar combinaciones de compuertas lógicas conectadas para formar circuitos lógicos.

3-2 TABLAS DE VERDAD

Una **tabla de verdad** es una herramienta para describir la forma en que la salida de un circuito lógico depende de los niveles lógicos presentes en las entradas del circuito. La figura 3-1(a) muestra una tabla de verdad para un tipo de circuito lógico de dos entradas. La tabla lista todas las posibles combinaciones de niveles lógicos

* Los voltajes entre 0.8 y 2 V están indefinidos (no son 0 ni 1) y bajo circunstancias normales no deben presentarse.

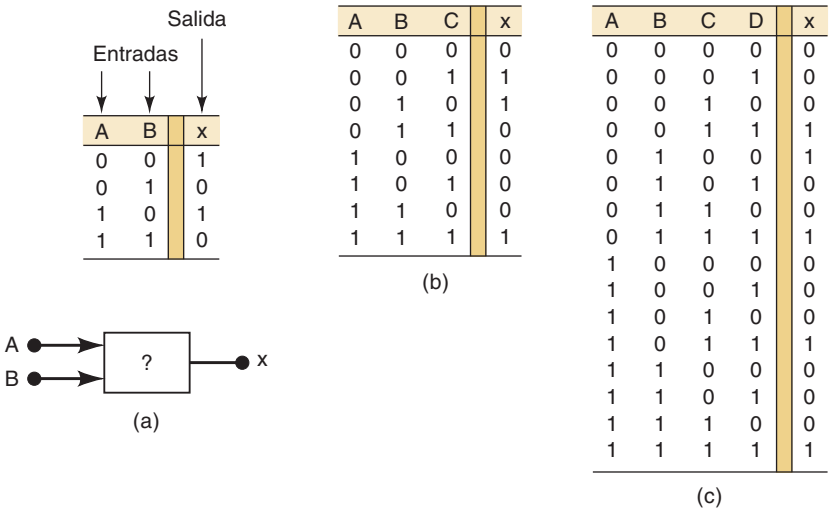


FIGURA 3-1 Ejemplo de tablas de verdad para circuitos de (a) dos entradas, (b) tres entradas y (c) cuatro entradas.

presentes en las entradas A y B, junto con el correspondiente nivel en la salida x. La primera entrada en la tabla muestra que cuando A y B se encuentran en el nivel 0, la salida x se encuentra en el nivel 1 o, de manera equivalente, en el estado 1. La segunda entrada muestra que cuando la entrada B se cambia al estado 1, de manera que $A = 0$ y $B = 1$, la salida x se vuelve un 0. De manera similar, la tabla muestra qué ocurre con el estado de salida para cualquier conjunto de condiciones de entrada.

Las figuras 3-1(b) y (c) muestran ejemplos de tablas de verdad para circuitos lógicos de tres y cuatro entradas. De nuevo, cada tabla enlista todas las posibles combinaciones de niveles lógicos de las entradas a la izquierda, con el nivel lógico resultante para la salida x a la derecha. Desde luego que los valores reales para x dependerán del tipo de circuito lógico.

Observe que hay 4 combinaciones para la tabla de verdad de dos entradas, 8 combinaciones para una tabla de verdad de tres entradas y 16 combinaciones para la tabla de verdad de cuatro entradas. El número de combinaciones de entrada será igual a 2^N para una tabla de verdad con N entradas. Observe también que la lista de todas las posibles combinaciones de entrada va de acuerdo con la secuencia de conteo binario, por lo que es fácil anotar todas las combinaciones sin que falte una.

PREGUNTAS DE REPASO

1. ¿Cuál es el estado de salida del circuito de cuatro entradas representado en la figura 3-1(c) cuando todas las entradas excepto B son 1?
2. Repita la pregunta 1 para las siguientes condiciones de entrada: $A = 1, B = 0, C = 1, D = 0$.
3. ¿Cuántas combinaciones se necesitan en la tabla para un circuito de cinco entradas?

3-3 OPERACIÓN OR CON COMPUERTAS OR

La **operación OR (O)** es la primera de las tres operaciones booleanas básicas que estudiaremos. El horno de cocina es un buen ejemplo. La luz dentro del horno debe encenderse si el *interruptor de la luz del horno está encendido* "*O*" si la *puerta está abierta*. La letra A podría usarse para representar la condición *interruptor de la luz del horno encendido* y B podría representar la condición *puerta abierta*. La letra x podría representar la condición *luz encendida*. La tabla de verdad de la figura 3-2(a) muestra lo que ocurre cuando se combinan dos entradas lógicas (A y B) mediante el uso de la operación OR para producir la salida x. La tabla muestras que x es un 1 lógico para cada una de las combinaciones de niveles de entrada en donde una *o* más entradas sea 1. El único caso en el que x es un 0 es cuando ambas entradas son 0.

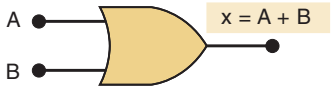
FIGURA 3-2 (a) Tabla de verdad que define la operación OR; (b) símbolo del circuito para una compuerta OR de dos entradas.



OR

A	B	x = A + B
0	0	0
0	1	1
1	0	1
1	1	1

(a)



Compuerta OR

(b)

La expresión booleana para la operación OR es

$$x = A + B$$

En esta expresión, el signo + no indica la suma ordinaria; indica la operación OR. Esta operación es similar a la suma ordinaria, excepto para el caso en el que tanto *A* como *B* son 1; la operación OR produce 1 + 1 = 1, no 1 + 1 = 2. En el álgebra booleana 1 es el valor más alto, por lo que nunca tendremos un resultado mayor que 1. Lo mismo aplica cuando se combinan tres entradas mediante el uso de la operación OR. Aquí tenemos que $x = A + B + C$. Si consideramos el caso en el que las tres entradas son 1, tenemos

$$x = 1 + 1 + 1 = 1$$

La expresión $x = A + B$ se lee como "*x* es igual a *A* OR *B*", lo cual significa que *x* será 1 cuando *A* o *B* o ambas sean 1. De igual forma, la expresión $x = A + B + C$ se lee como "*x* es igual a *A* OR *B* OR *C*", lo cual significa que *x* será 1 cuando *A* o *B* o *C* o cualquier combinación de ellas sean 1. Para describir este circuito en el idioma español podríamos decir que *x es verdadera (1)* ***CUANDO*** *A es verdadera (1)* ***OR*** *B es verdadera (1)* ***OR*** *C es verdadera (1)*.

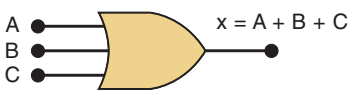
Compuerta OR

En los circuitos digitales, una **compuerta OR*** es un circuito que tiene dos o más entradas y cuya salida es igual a la combinación OR de las entradas. La figura 3-2(b) muestra el símbolo lógico para una compuerta **OR** de dos entradas. Las entradas *A* y *B* son niveles lógicos de voltaje y la salida *x* es un nivel lógico de voltaje cuyo valor es el resultado de la operación OR sobre *A* y *B*; es decir, $x = A + B$. En otras palabras, la compuerta OR opera de manera que su salida esté en ALTO, 1 lógico, si cualquiera de las entradas *A o B o ambas* se encuentran en el nivel 1 lógico. La salida de la compuerta OR estará en BAJO, 0 lógico, sólo si todas sus entradas están en 0 lógico.

Esta misma idea puede extenderse a más de dos entradas. La figura 3-3 muestra una compuerta OR de tres entradas y su tabla de verdad. Si examinamos esta tabla de verdad podremos ver de nuevo que la salida será 1 para cada caso en el que una o más entradas sean 1. Este principio general es el mismo para las compuertas OR con cualquier número de entradas.

FIGURA 3-3 Símbolo y tabla de verdad para una compuerta OR de tres entradas.





A	B	C	x = A + B + C
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

* El término *compuerta* proviene de la operación de inhibición/habilitación que se describe en el capítulo 4.

Si utilizamos el lenguaje del álgebra booleana, la salida x podemos expresarla como $x = A + B + C$, en donde hay que enfatizar otra vez que el símbolo + representa la operación OR. Así, la salida de cualquier compuerta OR puede expresarse como la combinación OR de sus diversas entradas. Pondremos esto en práctica a la hora de analizar los circuitos lógicos.

Resumen de la operación OR

Los puntos importantes que debemos recordar en relación con la operación OR y las compuertas OR son:

1. La operación OR produce un resultado (salida) de 1 siempre que *cualquiera* de sus entradas sea 1. En cualquier otro caso, la salida será 0.
2. Una compuerta OR es un circuito lógico que realiza una operación OR sobre las entradas del circuito.
3. La expresión $x = A + B$ se lee como "x es igual a A OR B".

EJEMPLO 3-1

En muchos sistemas de control industriales se requiere activar una función de salida cada vez que se activa una de varias entradas. Por ejemplo, en un proceso químico tal vez sea conveniente que se active una alarma cada vez que la temperatura del proceso exceda un valor máximo O cada vez que la presión pase cierto límite. La figura 3-4 es un diagrama de bloques del caso descrito. El circuito transductor de temperatura produce un voltaje de salida proporcional a la temperatura del proceso. Este voltaje V_T se compara con un voltaje de referencia de temperatura V_{TR} en un circuito comparador de voltajes. Por lo general, la salida del comparador T_H es de un voltaje bajo (0 lógico) pero cambia a un voltaje alto (1 lógico) cuando V_T excede a V_{TR}, lo cual indica que la temperatura del proceso es demasiado alta. Para la medición de presión se utiliza un arreglo similar, de manera que su salida asociada del comparador P_H cambie de BAJO a ALTO cuando la presión es demasiado alta.

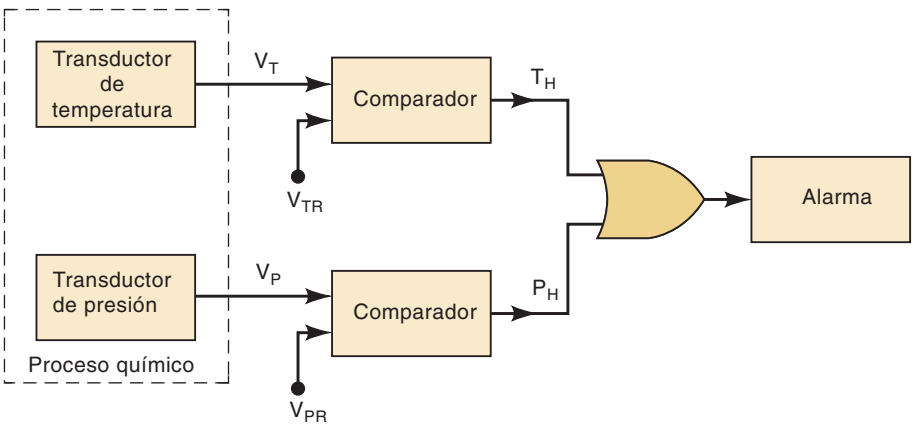


FIGURA 3-4 Ejemplo del uso de una compuerta OR en un sistema de alarma.



Dado que se quiere que la alarma se active cuando la temperatura o la presión estén demasiado altas, es indudable que las dos salidas del comparador pueden alimentarse a una compuerta OR. Por ende, la salida de la compuerta OR cambia a ALTO (1) para cualquiera de las dos condiciones de alarma activándola. Es obvio que esta misma idea puede extenderse a situaciones con más de dos variables del proceso.

EJEMPLO 3-2

Determine la salida de la compuerta OR de la figura 3-5. Las entradas *A* y *B* de la compuerta OR están variando de acuerdo con los diagramas de tiempos que se muestran. Por ejemplo, *A* empieza en nivel BAJO en el tiempo t_0, cambia a ALTO en t_1, regresa a BAJO en t_3 y así sucesivamente.

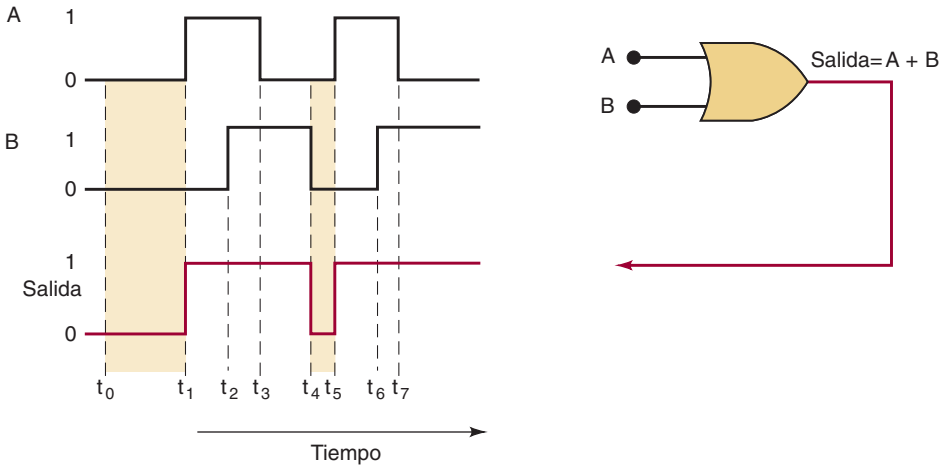


FIGURA 3-5 Ejemplo 3-2.

Solución

La salida de la compuerta OR estará en ALTO siempre que *cualquiera* de las entradas esté en ALTO. Entre los tiempos t_0 y t_1 ambas entradas están en BAJO, por lo cual SALIDA = BAJO. En t_1 la entrada *A* cambia a ALTO mientras que la entrada *B* permanece en BAJO. Esto hace que SALIDA cambie a ALTO en t_1 y permanezca así hasta t_4, ya que durante este intervalo una o ambas entradas están en ALTO. En t_4 la entrada *B* cambia de 1 a 0, por lo que ambas entradas están en BAJO haciendo que SALIDA cambie nuevamente a BAJO. En t_5 *A* cambia a ALTO, con lo cual SALIDA regresa a BAJO y permanece así durante el resto del tiempo mostrado.

EJEMPLO 3-3A

Para la situación que se describe en la figura 3-6, determine la forma de onda en la salida de la compuerta OR.

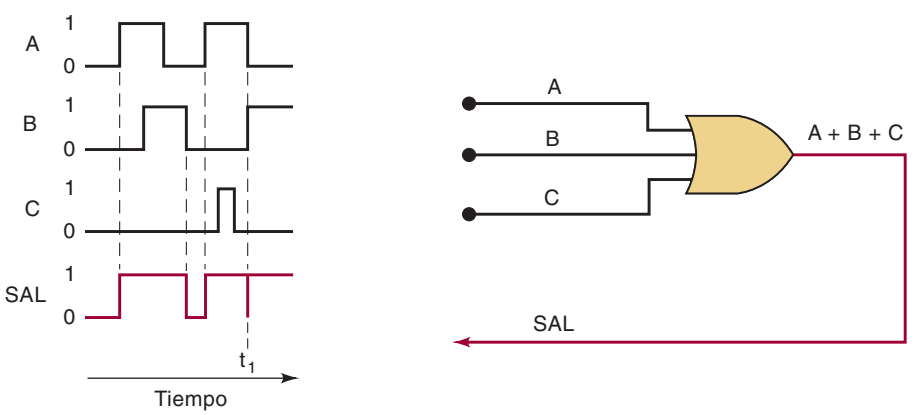


FIGURA 3-6 Ejemplos 3-3A y B.

Solución

Las tres entradas *A*, *B* y *C* de la compuerta OR están variando, según se muestra en sus diagramas de tiempos. La salida de la compuerta OR se determina concluyendo

que estará en ALTO cada vez que al menos *una* de las tres entradas esté en un nivel ALTO. Si utilizamos este razonamiento, la forma de onda de la salida de la compuerta OR es como se muestra en la figura. Se debe poner especial atención a lo que ocurre en el tiempo t_1. El diagrama muestra que, en ese instante, la entrada *A* está cambiando de ALTO a BAJO mientras que la entrada *B* está cambiando de BAJO a ALTO. Como estas entradas están realizando sus transiciones casi al mismo tiempo, y como estas transiciones tardan cierta cantidad de tiempo, hay un pequeño intervalo durante el cual estas entradas de la compuerta OR se encuentran en la región indefinida entre 0 y 1. Cuando esto ocurre, la salida de la compuerta OR también adquiere un valor en este intervalo, como se puede ver por la deformación o pico en la forma de onda de salida en t_1. La ocurrencia de esta deformación y su tamaño (amplitud y anchura) dependen de la velocidad con la que ocurren las transiciones de entrada.

EJEMPLO 3-3B

¿Qué le pasaría a la deformación en la salida de la figura 3-6 si la entrada *C* se asentara en el estado ALTO mientras que *A* y *B* estuvieran cambiando en el tiempo t_1?

Solución

Con la entrada *C* en nivel ALTO en t_1 la salida de la compuerta OR permanecerá en ese estado sin importar lo que ocurra en las otras entradas, ya que cualquier entrada en ALTO mantendrá la salida de la compuerta OR en ALTO. Por lo tanto, la deformación no aparecerá en la salida.

PREGUNTAS DE REPASO

1. ¿Cuál es el único conjunto de condiciones de entrada que producirá una salida en BAJO para cualquier compuerta OR?
2. Escriba la expresión booleana para una compuerta OR de seis entradas.
3. Si la entrada *A* en la figura 3-6 se mantiene de manera permanente en el nivel 1, ¿cuál será la forma de onda de salida?

3-4 OPERACIÓN AND CON COMPUERTAS AND

La **operación AND** es la segunda operación booleana básica. Como ejemplo del uso de la lógica AND, considere una secadora de ropa ordinaria: seca ropa (calienta y gira) sólo si el *temporizador está por encima de cero* "Y" la *puerta está cerrada*. Vamos a asignar *A* para representar cuando el *temporizador está activado*, *B* para representar cuando la *puerta está cerrada* y *x* puede representar cuando el *calentador y el motor están encendidos*. La tabla de verdad de la figura 3-7(a) muestra lo que ocurre cuando dos entradas lógicas *A* y *B* se combinan mediante el uso de la operación *AND* para producir la salida *x*. La tabla muestra que *x* es un 1 lógico sólo cuando *A* y *B* están en el nivel 1 lógico. Para cualquier caso en el que una de las entradas sea 0, la salida será 0.

La expresión booleana para la operación AND es

$$x = A \cdot B$$

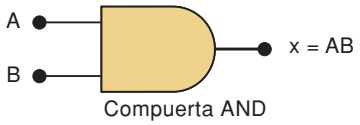
En esta expresión, el signo · indica la operación AND booleana y no la operación de multiplicación. No obstante, la operación AND sobre variables booleanas opera de la misma forma que la multiplicación ordinaria según nos muestra el análisis de la tabla de verdad, por lo que podemos considerarlas como iguales. Esta característica puede ser útil al evaluar expresiones lógicas que contengan operaciones AND.

FIGURA 3-7 (a) Tabla de verdad para la operación AND; (b) símbolo de la compuerta AND.

AND

A	B	x = A · B
0	0	0
0	1	0
1	0	0
1	1	1

(a)



(b)

La expresión $x = A \cdot B$ se lee como "x es igual a A AND B", lo cual significa que x será 1 sólo cuando A y B sean 1. Por lo general, se omite el signo · de manera que la expresión se vuelve $x = AB$. Para el caso en el que se aplica la operación AND con tres entradas, tenemos que $x = A \cdot B \cdot C = ABC$. Esto se lee como "$x$ es igual a A AND B AND C", lo cual significa que x será 1 sólo cuando A y B y C sean todas 1.

Compuerta AND

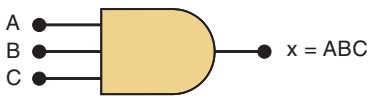
La figura 3-7(b) muestra el símbolo lógico para una **compuerta AND** de dos entradas. La salida de la compuerta AND es igual al producto AND de las entradas lógicas; es decir, $x = AB$. En otras palabras, la compuerta AND es un circuito que opera de manera que su salida esté en ALTO sólo cuando todas sus entradas se encuentren en ALTO. Para todos los demás casos, la salida de la compuerta AND estará en BAJO.

Esta misma operación es característica de compuertas AND con más de dos entradas. Por ejemplo, la figura 3-8 muestra una compuerta AND de tres entradas y su tabla de verdad correspondiente. Una vez más, observe que la salida de la compuerta es 1 sólo para el caso en el que $A = B = C = 1$. La expresión para la salida es $x = ABC$. Para una compuerta AND de cuatro entradas, la salida es $x = ABCD$, y así sucesivamente.

FIGURA 3-8 Tabla de verdad y símbolo para una compuerta AND de tres entradas.



A	B	C	x = ABC
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1



Observe la diferencia entre los símbolos para la compuerta AND y la compuerta OR. Cada vez que vea el símbolo AND en el diagrama de un circuito lógico, le indicará que la salida estará en ALTO *sólo* cuando *todas* las entradas estén en ALTO. Cada vez que vea el símbolo OR, le indicará que la salida estará en ALTO cuando *cualquiera* de sus entradas esté en ALTO.

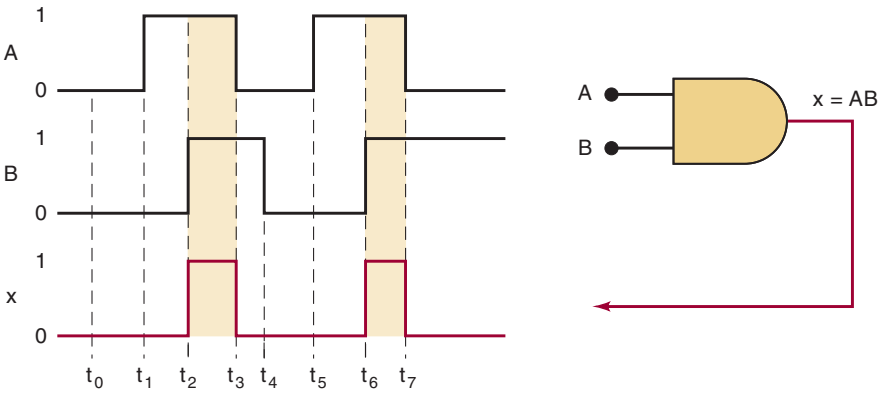
Resumen de la operación AND

1. La operación AND se realiza de la misma forma que la multiplicación ordinaria de 1s y 0s.
2. Una compuerta AND es un circuito lógico que realiza la operación AND sobre las entradas.
3. La salida de una compuerta AND será 1 *sólo* para el caso en el que *todas* las entradas sean 1; para todos los demás casos la salida será 0.
4. La expresión $x = AB$ se lee como "x es igual a A AND B".

EJEMPLO 3-4

Determine la salida x de la compuerta AND en la figura 3-9, para las formas de onda de entrada dadas.

FIGURA 3-9 EJEMPLO 3-4.



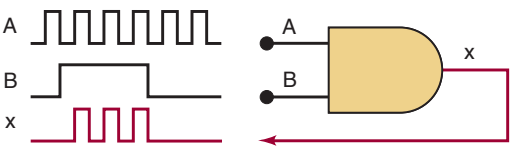
Solución

La salida de una compuerta AND se determina con base en la consideración de que estará en ALTO sólo cuando todas las entradas se encuentren en ALTO al mismo tiempo. Para las formas de onda de entrada dadas, la condición se cumple sólo durante los intervalos $t_2 - t_3$ y $t_6 - t_7$. En todos los demás intervalos, sólo una o más de las entradas son 0, con lo cual se produce una salida en BAJO. Observe que los cambios en el nivel de entrada que se producen mientras la otra entrada está en BAJO no tienen efecto sobre la salida.

EJEMPLO 3-5A

Determine la forma de onda de salida para la compuerta AND que se muestra en la figura 3-10.

FIGURA 3-10 Ejemplos 3-5A y 3-5B.



Solución

La salida x será 1 sólo cuando A y B se encuentren en ALTO al mismo tiempo. Si nos basamos en este hecho podemos determinar la forma de onda de x como se muestra en la figura.

Observe que la forma de onda de x es 0 siempre que B es 0, sin importar el nivel de la señal en A. Observe también que siempre que B es 1, la forma de onda de x es igual que la de A. En consecuencia podemos considerar la entrada B como una entrada de *control* cuyo nivel lógico determina si la forma de onda de A pasa o no hacia la salida x. En esta situación, la compuerta AND se utiliza como un *circuito inhibidor*. Podemos decir que $B = 0$ es la condición de inhibición que produce una salida de 0. Por otro lado, $B = 1$ es la condición de *habilitación*, la cual permite que A llegue a la salida. Esta operación de inhibición es una aplicación importante de las compuertas AND, que veremos más adelante con mayor detalle.

EJEMPLO 3-5B

¿Cómo se modifica la forma de onda de la salida x en la figura 3-10 si la entrada B se mantiene en el nivel 0?

Solución

Con B en BAJO, la salida x también permanecerá en BAJO. Esto puede razonarse de dos maneras distintas. Primero, con $B = 0$ tenemos que $x = A \cdot B = A \cdot 0 = 0$, ya que

cualquier cosa que se multiplique por 0 (AND) será igual a 0. Otra manera de ver esto es que una compuerta AND requiere que todas las entradas estén en ALTO para que la salida esté en ALTO, y esto no puede ocurrir si *B* se mantiene en BAJO.

PREGUNTAS DE REPASO

1. ¿Cuál es la única combinación de entrada que producirá un nivel ALTO en la salida de una compuerta AND de cinco entradas?
2. ¿Qué nivel lógico debe aplicarse a la segunda entrada de una compuerta AND de dos entradas si se desea inhibir la señal lógica en la primera entrada para evitar que llegue a la salida?
3. *Verdadero o falso:* la salida de una compuerta AND siempre será distinta de la salida de una compuerta OR para las mismas condiciones de entrada.

3-5 OPERACIÓN NOT

La **operación NOT** es distinta de las operaciones OR y AND, ya que puede realizarse sobre una sola variable de entrada. Por ejemplo, si la variable A está sujeta a la operación NOT, el resultado x puede expresarse así:

$$x = \overline{A}$$

En donde la barra superior representa la operación NOT. Esta expresión se lee como "x es igual a NOT A" o "x es igual al *inverso* de A" o "x es igual al *complemento* de A". Cada una de estas expresiones es de uso común y todas indican que el valor lógico de $x = \overline{A}$ *es el opuesto* del valor lógico de A. La tabla de verdad de la figura 3-11(a) aclara esto para los dos casos en que $A = 0$ y $A = 1$. Esto es,

$$0 = \overline{1} \quad \text{porque 0 no es 1}$$

y

$$1 = \overline{0} \quad \text{porque 1 no es 0}$$

La operación NOT también se conoce como **inversión** o **complementación**, términos que utilizaremos de manera indistinta en este libro. Aunque siempre utilizaremos el indicador de barra superior para representar la inversión, es importante mencionar que el símbolo primo (′) es otro indicador para la inversión. Esto es,

$$A' = \overline{A}$$

Ambos deben reconocerse como símbolos que indican la operación de inversión.

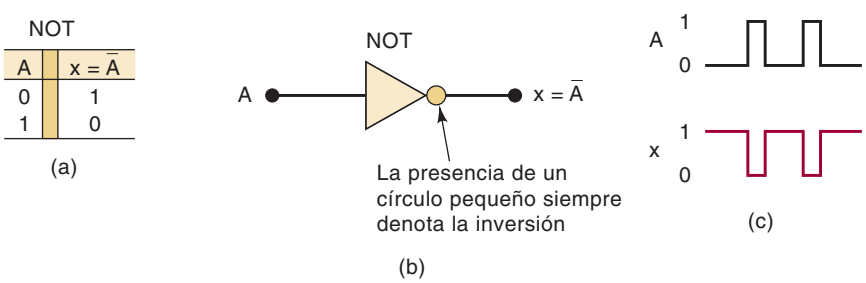


FIGURA 3-11 (a) Tabla de verdad; (b) símbolo para el INVERSOR (compuerta NOT); (c) ejemplos de formas de onda.

Circuito NOT (INVERSOR)

La figura 3-11(b) muestra el símbolo para un **circuito NOT**, al cual se le conoce más comúnmente como **INVERSOR**. Este circuito *siempre* tiene una sola entrada y su nivel lógico de salida siempre es opuesto al nivel lógico de esta entrada. La figura 3-11(c) muestra la forma en que el INVERSOR afecta a una señal de entrada. Invierte (complementa) la señal de entrada en todos los puntos de la forma de onda, por lo que siempre que la entrada = 0, la salida = 1, y viceversa.

APLICACIÓN 3-1

La figura 3-12 muestra una aplicación ordinaria de la compuerta NOT. El botón está cableado para producir un 1 lógico (verdadero) cuando se oprime. Algunas veces es conveniente saber si el botón no está oprimido, por lo que este circuito proporciona una expresión que es verdadera cuando el botón no está oprimido.

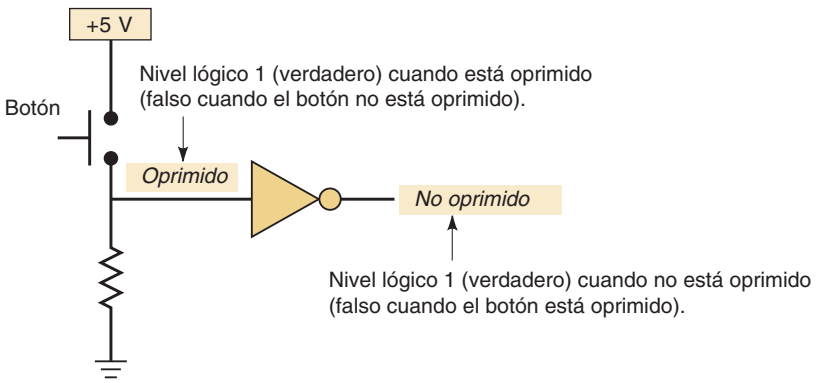


FIGURA 3-12 Una compuerta NOT que indica que un botón *no* está oprimido cuando su salida es verdadera.

Resumen de las operaciones booleanas

Las reglas para las operaciones OR, AND y NOT pueden resumirse de la siguiente manera:

OR	*AND*	*NOT*
$0 + 0 = 0$	$0 \cdot 0 = 0$	$\overline{0} = 1$
$0 + 1 = 1$	$0 \cdot 1 = 0$	$\overline{1} = 0$
$1 + 0 = 1$	$1 \cdot 0 = 0$	
$1 + 1 = 1$	$1 \cdot 1 = 1$	

PREGUNTAS DE REPASO

1. La salida del INVERSOR de la figura 3-11 está conectada a la entrada de un segundo INVERSOR. Determine el nivel de salida del segundo INVERSOR para cada nivel de entrada *A*.
2. La salida de la compuerta AND de la figura 3-7 está conectada a la entrada de un INVERSOR. Escriba la tabla de verdad que muestre la salida, *y*, del INVERSOR para cada combinación de entradas *A* y *B*.

3-6 DESCRIPCIÓN DE CIRCUITOS LÓGICOS EN FORMA ALGEBRAICA

Cualquier circuito lógico, sin importar qué tan complejo sea, puede describirse por completo mediante el uso de las tres operaciones booleanas básicas ya que las compuertas OR, AND y el circuito NOT son los bloques fundamentales para la construc-

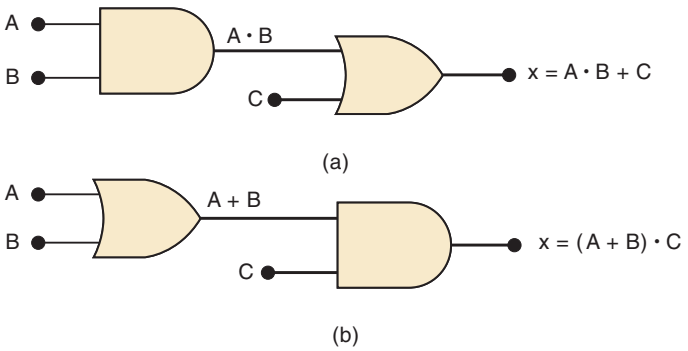


FIGURA 3-13 (a) Circuito lógico con su expresión booleana. (b) Circuito lógico cuya expresión requiere paréntesis.

ción de sistemas digitales. Por ejemplo, considere el circuito de la figura 3-13(a), el cual tiene tres entradas *A*, *B* y *C*, y una sola salida *x*. Si utilizamos la expresión booleana para cada compuerta podemos determinar con facilidad la expresión para la salida.

La expresión para la salida de la compuerta AND se escribe como $A \cdot B$. Esta salida AND está conectada como entrada para la compuerta OR junto con *C*, otra entrada. La compuerta OR opera sobre sus entradas de manera que su salida es la suma OR de las entradas. Por lo tanto, podemos expresar la salida OR como $x = A \cdot B + C$. Esta expresión final podría haberse escrito también como $x = C + A \cdot B$, ya que no importa cuál término de la suma OR se escriba primero.

Precedencia de operadores

En ocasiones puede haber confusión acerca de cuál operación debe llevarse a cabo primero en una expresión. La expresión $A \cdot B + C$ puede interpretarse de dos maneras: (1) Se aplica un OR entre $A \cdot B$ y el término *C*; o (2) Se aplica un AND entre *A* y el término $B + C$. Para evitar esta confusión debe quedar claro que si una expresión contiene las operaciones AND y OR, la operación AND se realiza primero, a menos que haya *paréntesis* en la expresión, en cuyo caso la operación encerrada entre paréntesis es la que se debe realizar primero. Esta regla es la misma que se utiliza en el álgebra ordinaria para determinar el orden de las operaciones.

Para ilustrar mejor este concepto, considere el circuito de la figura 3-13(b). La expresión para la salida de la compuerta OR es $A + B$. Esta salida sirve como entrada para la compuerta AND junto con otra entrada *C*. Por ende, expresamos la salida de la compuerta AND como $x = (A + B) \cdot C$. Observe el uso de los paréntesis aquí para indicar que primero se aplica la operación OR entre *A* y *B*, antes de que a su suma OR se le aplique un AND con *C*. Sin los paréntesis se interpretaría de manera incorrecta, ya que $A + B \cdot C$ significa que a la entrada *A* se le aplica un OR con el producto $B \cdot C$.

Circuitos que contienen INVERSORES

Siempre que haya un INVERSOR presente en el diagrama de un circuito lógico, la expresión de su salida es en sí igual a la expresión de la entrada con una barra sobre ella. La figura 3-14 muestra dos ejemplos que utilizan INVERSORES. En la figura 3-14(a) la entrada *A* se alimenta a través de un INVERSOR, cuya salida es, por lo tanto $\overline{A}$. La salida del INVERSOR se alimenta a una compuerta OR junto con *B*, de manera que la salida *OR* es igual a $A + B$. Observe que la barra sólo está sobre *A*, lo cual indica que *A* primero se invierte y después se alimenta a la compuerta OR junto con *B*.

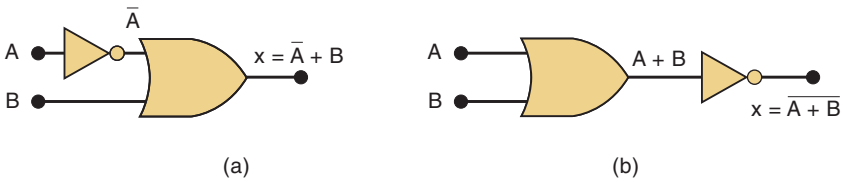


FIGURA 3-14 Circuitos que utilizan INVERSORES.

En la figura 3-14(b) la salida de la compuerta OR es igual a $A + B$ y se alimenta a través de un INVERSOR. Por lo tanto, la salida del INVERSOR es igual a $(\overline{A + B})$ ya que invierte *toda* la expresión de entrada. Observe que la barra cubre toda la expresión $(A + B)$. Esto es importante ya que, como veremos más adelante, las expresiones $(\overline{A + B})$ y $(\overline{A} + \overline{B})$ *no* son equivalentes. La expresión $(\overline{A + B})$ indica que primero se aplica un OR entre A y B, y después su suma OR se invierte, mientras que la expresión $(\overline{A} + \overline{B})$ indica que primero se invierten A y B, y después se aplica un OR a los resultados de las dos inversiones.

La figura 3-15 muestra dos ejemplos más, que deben estudiarse con cuidado. Note el uso de *dos* conjuntos separados de paréntesis en la figura 3-15(b). Observe además en la figura 3-15(a) que la variable de entrada A está conectada como entrada para dos compuertas distintas.

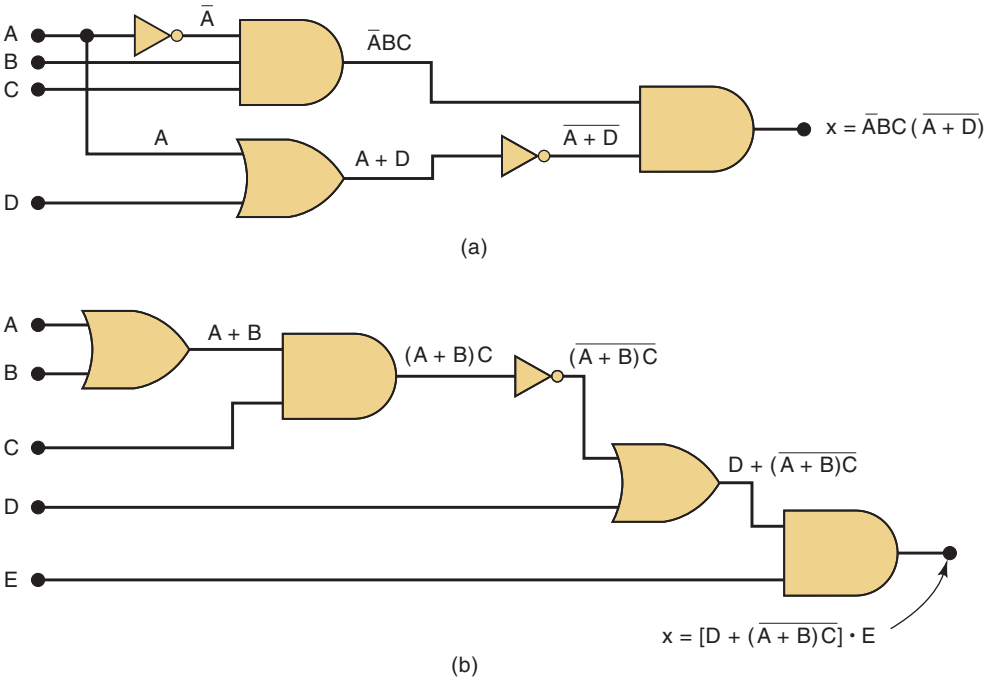


FIGURA 3-15 Más ejemplos.

PREGUNTAS DE REPASO

1. En la figura 3-15(a) cambie cada compuerta AND por una compuerta OR, y cambie la compuerta OR por una compuerta AND. Después escriba la expresión para la salida x.
2. En la figura 3-15(b) cambie cada compuerta AND por una compuerta OR y cada compuerta OR por una compuerta AND. Después escriba la expresión para la salida x.

3-7 EVALUACIÓN DE LAS SALIDAS DE CIRCUITOS LÓGICOS

Una vez que tengamos la expresión booleana para un circuito lógico, podremos obtener el nivel lógico de salida para cualquier conjunto de niveles de entrada. Por ejemplo, suponga que deseamos conocer el nivel lógico de la salida x del circuito de la figura 3-15(a), para el caso en el que $A = 0$, $B = 1$, $C = 1$ y $D = 1$. Como en el álgebra ordinaria, podemos encontrar el valor de x si "conectamos" los valores de

las variables en la expresión y realizamos las operaciones indicadas como se muestra a continuación:

$$\begin{aligned} x &= \overline{A}BC(\overline{A + D}) \\ &= \overline{0} \cdot 1 \cdot 1 \cdot (\overline{0 + 1}) \\ &= 1 \cdot 1 \cdot 1 \cdot (\overline{0 + 1}) \\ &= 1 \cdot 1 \cdot 1 \cdot (\overline{1}) \\ &= 1 \cdot 1 \cdot 1 \cdot 0 \\ &= 0 \end{aligned}$$

Como otro ejemplo, vamos a evaluar la salida del circuito de la figura 3-15(b) para $A = 0, B = 0, C = 0, D = 1$ y $E = 1$.

$$\begin{aligned} x &= [D + \overline{(A + B)C}] \cdot E \\ &= [1 + \overline{(0 + 0) \cdot 1}] \cdot 1 \\ &= [1 + \overline{0 \cdot 1}] \cdot 1 \\ &= [1 + \overline{0}] \cdot 1 \\ &= [1 + 1] \cdot 1 \\ &= 1 \cdot 1 \\ &= 1 \end{aligned}$$

En general, siempre debemos seguir las siguientes reglas cuando se evalúa una expresión booleana:

1. Primero realice todas las inversiones de términos individuales; es decir, $\overline{0} = \overline{1}$ o $1 = 0$.
2. Después realice todas las operaciones que estén encerradas entre paréntesis.
3. Realice una operación AND antes de una operación OR, a menos que haya paréntesis que indiquen lo contrario.
4. Si una expresión tiene una barra sobre ella, primero realice las operaciones dentro de la expresión y después invierta el resultado.

Como práctica, determine las salidas de ambos circuitos de la figura 3-15 para el caso en el que todas las entradas sean 1. Las respuestas son $x = 0$ y $x = 1$, respectivamente.

Análisis mediante el uso de una tabla

Siempre que tenga un circuito lógico combinacional y desee saber cómo funciona, la mejor manera de analizarlo es mediante el uso de una tabla de verdad. Las ventajas de este método son:

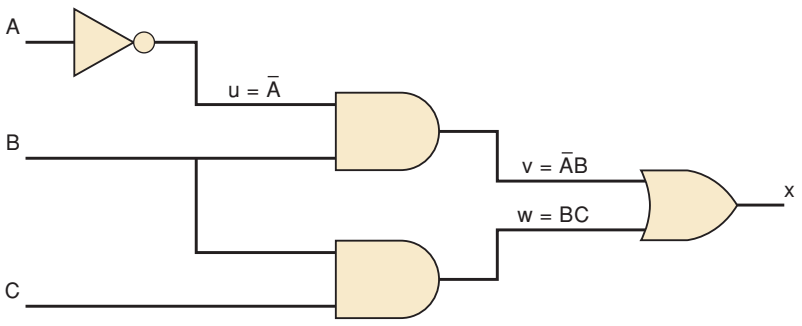
Le permite analizar una compuerta o combinación lógica a la vez.

Le facilita una segunda comprobación de su trabajo.

Cuando termine tendrá una tabla que será de un beneficio incalculable cuando se busquen fallas en el circuito lógico.

Recuerde que una tabla de verdad lista todas las posibles combinaciones de las entradas en orden numérico. Para cada combinación posible de entrada, podemos determinar el estado lógico en cada punto (nodo) del circuito lógico, incluyendo la salida. Consulte la figura 3-16(a). Hay varios nodos intermedios en este circuito, los cuales no son entradas ni salidas. Tan sólo son conexiones entre la salida de una compuerta y la entrada de otra. En este diagrama se han etiquetado como u, v y w. El primer paso después de listar todas las combinaciones de entradas es crear una columna en la tabla de verdad para cada señal intermedia (nodo), como se muestra en la figura 3-16(b). El nodo u se ha llenado como el complemento de A.

FIGURA 3-16 Análisis de un circuito lógico mediante el uso de tablas de verdad.



(a)

A	B	C	u= Ā	v= ĀB	w= BC	x= v+w
0	0	0	1			
0	0	1	1			
0	1	0	1			
0	1	1	1			
1	0	0	0			
1	0	1	0			
1	1	0	0			
1	1	1	0			

(b)

A	B	C	u= Ā	v= ĀB	w= BC	x= v+w
0	**0**	0	1	**0**		
0	**0**	1	1	**0**		
0	1	0	1	1		
0	1	1	1	1		
1	0	0	0	**0**		
1	0	1	0	**0**		
1	1	0	**0**	**0**		
1	1	1	**0**	**0**		

(c)

A	B	C	u= Ā	v= ĀB	w= BC	x= v+w
0	**0**	**0**	1	0	**0**	
0	**0**	1	1	0	**0**	
0	1	**0**	1	1	**0**	
0	1	1	1	1	1	
1	**0**	**0**	0	0	**0**	
1	**0**	1	0	0	**0**	
1	1	**0**	0	0	**0**	
1	1	1	0	0	1	

(d)

A	B	C	u= Ā	v= ĀB	w= BC	x= v+w
0	0	0	1	**0**	**0**	**0**
0	0	1	1	**0**	**0**	**0**
0	1	0	1	1	**0**	1
0	1	1	1	1	1	1
1	0	0	0	**0**	**0**	**0**
1	0	1	0	**0**	**0**	**0**
1	1	0	0	**0**	**0**	**0**
1	1	1	0	**0**	1	1

(e)

El siguiente paso es llenar los valores para la columna *v*, como se muestra en la figura 3-16(c). Del diagrama podemos ver que $v = \bar{A}B$. El nodo *v* debe estar en ALTO cuando $\bar{A}$ (nodo *u*) está en ALTO Y *B* está en ALTO. Esto ocurre cada vez que *A* está en BAJO Y *B* está en ALTO. El tercer paso es predecir los valores en el nodo *w*, que es el producto lógico de *BC*. Esta columna está en ALTO cada vez que *B* está en ALTO Y *C* está en ALTO, como se muestra en la figura 3-16(d). El último paso es combinar en forma lógica las columnas *v* y *w* para predecir la salida *x*. Como $x = v + w$, la salida *x* estará en ALTO cuando *v* esté en ALTO *O w* esté en ALTO, como se muestra en la figura 3-16(e).

Si usted construyera este circuito y no se produjera la salida correcta para *x* bajo todas las condiciones, podría utilizar esta tabla para encontrar el problema. El procedimiento general es probar el circuito bajo cada combinación de entradas. Si cualquier combinación de entradas produce una salida incorrecta (es decir, una falla), compare el estado lógico actual de cada nodo intermedio en el circuito con el valor teórico correcto en la tabla mientras aplica esa condición de entrada. Si el estado lógico para un nodo intermedio es *correcto*, el problema debe estar más a la derecha de ese nodo. Si el estado lógico de un nodo intermedio es *incorrecto*, el problema debe estar a la izquierda de ese nodo (o ese nodo está en corto con algo). En el capítulo 4 veremos con más detalle los procedimientos de detección de errores y las posibles fallas en los circuitos.

EJEMPLO 3-6

Analice la operación de la figura 3-15(a) mediante la creación de una tabla que muestre el estado lógico en cada nodo del circuito.

Solución

Para llenar la columna t, escriba un 1 para todas las entradas en las que $A = 0$ Y $B = 1$ y $C = 1$.

Para llenar la columna u, escriba un 1 para todas las entradas en las que $A = 1$ O $D = 1$.

Para llenar la columna v, complete todas las entradas de la columna u.

Para llenar la columna x, escriba un 1 para todas las entradas en las que $t = 1$ y $v = 1$.

A	B	C	D	$t = \overline{A}BC$	$u = A + D$	$v = \overline{A + D}$	$x = tv$
0	0	0	0	0	0	1	0
0	0	0	1	0	1	0	0
0	0	1	0	0	0	1	0
0	0	1	1	0	1	0	0
0	1	0	0	0	0	1	0
0	1	0	1	0	1	0	0
0	1	1	0	1	0	1	1
0	1	1	1	1	1	0	0
1	0	0	0	0	1	0	0
1	0	0	1	0	1	0	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	0	0
1	1	0	0	0	1	0	0
1	1	0	1	0	1	0	0
1	1	1	0	0	1	0	0
1	1	1	1	0	1	0	0

PREGUNTAS DE REPASO

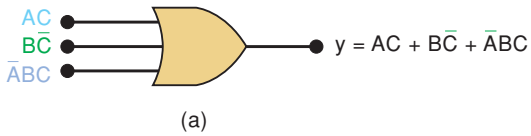
1. Use la expresión de x para determinar la salida del circuito de la figura 3-15(a), para las condiciones en que $A = 0, B = 1, C = 1$ y $D = 0$.
2. Use la expresión de x para determinar la salida del circuito de la figura 3-15(b), para las condiciones en que $A = B = E = 1$ y $C = D = 0$.
3. Determine las respuestas a las preguntas 1 y 2 encontrando los niveles lógicos presentes en la salida de cada compuerta mediante el uso de una tabla, como en la figura 3-16.

3-8 IMPLEMENTACIÓN DE CIRCUITOS A PARTIR DE EXPRESIONES BOOLEANAS

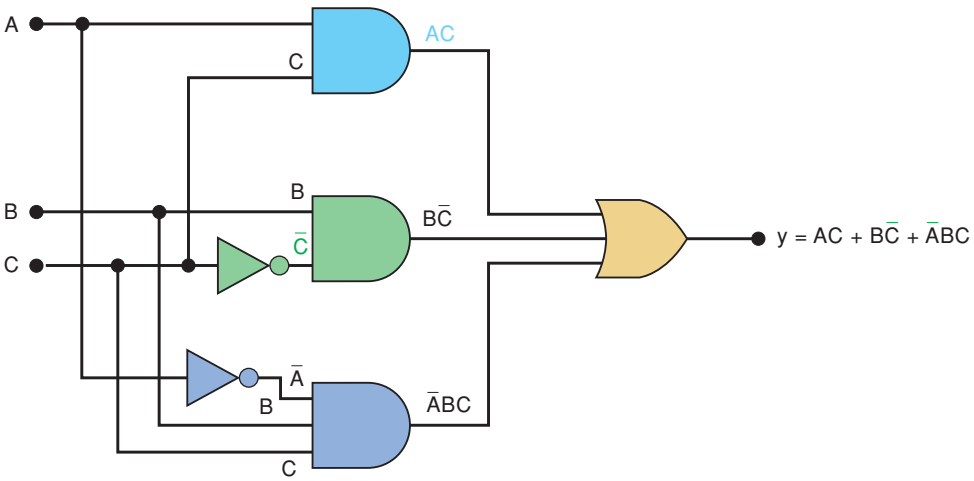
Cuando la operación de un circuito se define mediante una expresión booleana, podemos dibujar el diagrama de un circuito lógico de manera directa a partir de esa expresión. Por ejemplo, si necesitáramos un circuito que estuviera definido por $x = A \cdot B \cdot C$, de inmediato sabríamos que todo lo que se requiere es una compuerta AND de tres entradas. Si necesitáramos un circuito definido por $x = A + \overline{B}$, utilizaríamos una compuerta OR de dos entradas con un INVERSOR en una de ellas. El mismo razonamiento que se utiliza para estos casos puede extenderse hacia circuitos más complejos.

Suponga que deseamos construir un circuito cuya salida sea $y = AC + B\overline{C} + \overline{A}BC$. Esta expresión booleana contiene tres términos (AC, $B\overline{C}$, $\overline{A}BC$), a los cuales se les aplica una operación OR. Esto nos indica que se requiere una compuerta OR de tres entradas, en donde éstas son iguales a AC, $B\overline{C}$ y $\overline{A}BC$. En la figura 3-17(a) se ilustra esto, en donde está dibujada una compuerta OR de tres entradas etiquetadas como AC, $B\overline{C}$ y $\overline{A}BC$.

FIGURA 3-17
Construcción de un circuito lógico a partir de una expresión booleana.



(a)



(b)

Cada entrada de la compuerta OR es un término del producto de AND, lo cual significa que puede usarse una compuerta AND con las entradas apropiadas para generar cada uno de estos términos. Esto se muestra en la figura 3-17(b), que corresponde al diagrama del circuito final. Observe el uso de INVERSORES para producir los términos A y C requeridos en la expresión.

Siempre puede seguirse este mismo enfoque general, aunque más adelante veremos que existen técnicas más inteligentes y eficientes. No obstante, por ahora utilizaremos este método simple y directo para minimizar el número de elementos nuevos que deberá aprender.

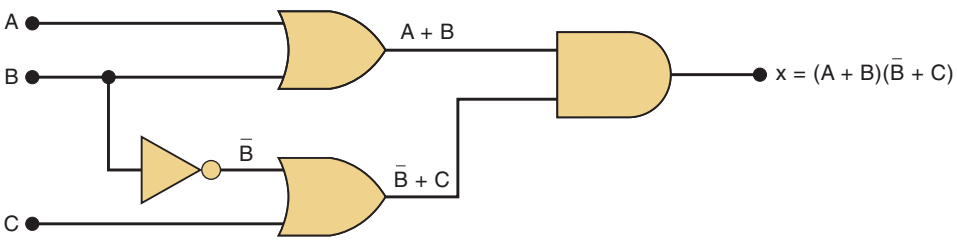
EJEMPLO 3-7

Dibuje el diagrama del circuito para implementar la expresión $x = (A + B)(\overline{B} + C)$.

Solución

Esta expresión muestra que los términos $A + B$ y $\overline{B} + C$ son entradas para una compuerta AND, y cada uno de estos términos se genera a partir de una compuerta OR separada. La figura 3-18 muestra un dibujo del resultado.

FIGURA 3-18 Ejemplo 3-7.



PREGUNTAS DE REPASO

1. Dibuje el diagrama del circuito que implemente la expresión $x = \overline{A}BC(\overline{A + D})$ mediante el uso de compuertas con no más de tres entradas.
2. Dibuje el diagrama del circuito para la expresión $y = AC + B\overline{C} + \overline{A}BC$.
3. Dibuje el diagrama del circuito para $x = [D + (\overline{A + B}\overline{C})] \cdot E$.

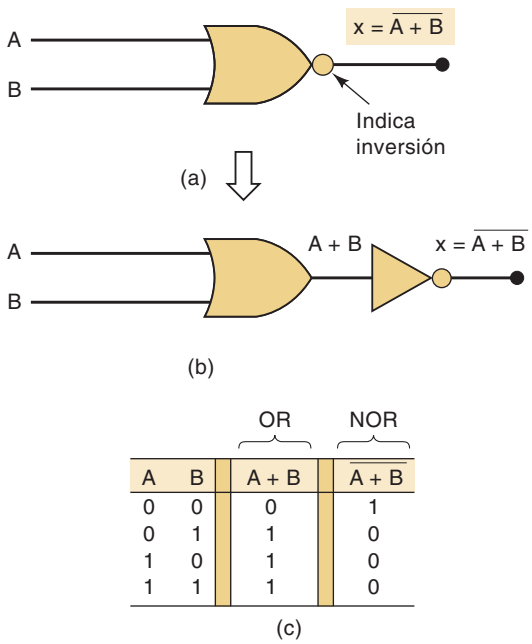
3-9 COMPUERTAS NOR Y NAND

En los circuitos digitales se utilizan mucho otros dos tipos de compuertas lógicas: NOR y NAND. Estas compuertas combinan las operaciones básicas AND, OR y NOT, por lo que es muy sencillo escribir sus expresiones booleanas.

Compuerta NOR

La figura 3-19(a) muestra el símbolo para una **compuerta NOR** de dos entradas. Es igual que el símbolo de la compuerta OR, sólo que tiene un pequeño círculo en la salida. El pequeño círculo representa la operación de inversión. Por ello, la compuerta NOR opera de manera similar a una compuerta OR seguida de un INVERSOR, de manera que los circuitos en las figuras 3-19(a) y (b) son equivalentes, y la expresión de salida para la compuerta NOR es $x = \overline{A + B}$.

FIGURA 3-19 (a) símbolo NOR; (b) circuito equivalente; (c) tabla de verdad.



		OR	NOR
A	B	A + B	$\overline{A + B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

(c)

La tabla de verdad de la figura 3-19(c) muestra que la salida de la compuerta NOR es el inverso exacto de la salida de la compuerta OR para todas las posibles condiciones de entrada. La salida de una compuerta OR cambia a ALTO cuando cualquiera de sus entradas se encuentra en ALTO; la salida de la compuerta NOR cambia a BAJO cuando cualquiera de sus entradas se encuentra en ALTO. Esta misma operación puede aplicarse a las compuertas NOR con más de dos entradas.

EJEMPLO 3-8

Determine la forma de onda en la salida de una compuerta NOR para las formas de onda de entrada que se muestran en la figura 3-20.



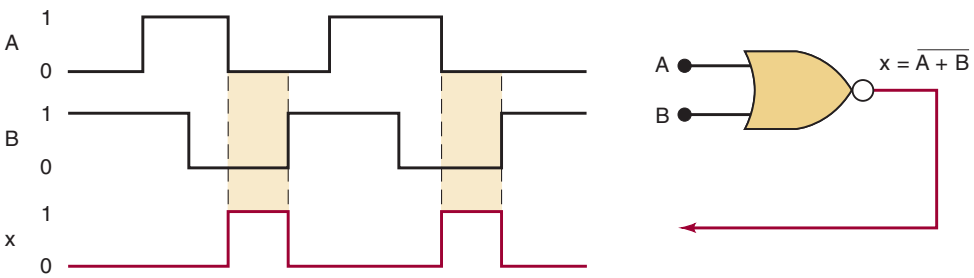


FIGURA 3-20 Ejemplo 3-8.

Solución

Una manera de determinar la forma de onda de salida NOR es encontrando primero la forma de onda de salida OR y luego invirtiéndola (cambiar todos los 1s por 0s y viceversa). Otro sistema utiliza el hecho de que la salida de una compuerta NOR estará en ALTO *sólo* cuando todas sus entradas estén en BAJO. Por lo tanto, puede examinar las formas de onda de entrada, encontrar esos intervalos en los que todas están en BAJO, y hacer que la salida de la compuerta NOR esté en ALTO para esos intervalos. La salida de la compuerta NOR estará en BAJO para todos los demás intervalos. La figura muestra la forma de onda de salida resultante.

EJEMPLO 3-9

Determine la expresión booleana para una compuerta NOR de tres entradas, seguida de un INVERSOR.

Solución

Consulte la figura 3-21, en donde se muestra el diagrama del circuito. La expresión en la salida de la compuerta NOR es $(\overline{A + B + C})$, la cual se alimenta después a través de un INVERSOR para producir

$$x = (\overline{\overline{A + B + C}})$$

La presencia de los signos de doble inversión indica que se invirtió la cantidad $(A + B + C)$ y se invirtió una vez más. Debe quedar claro que esto sólo hace que la expresión $(A + B + C)$ permanezca sin cambios. Esto es,

$$x = (\overline{\overline{A + B + C}}) = (A + B + C)$$

Siempre que hay dos barras de inversión sobre la misma variable o cantidad, una cancela a la otra, como en el ejemplo anterior. No obstante, en casos tales como $\overline{\overline{A} + \overline{B}}$ las barras de inversión no se cancelan. Esto se debe a que las barras de inversión más pequeñas invierten las variables individuales A y B, mientras que la barra larga invierte la cantidad $(\overline{A} + \overline{B})$. En consecuencia, $\overline{\overline{A} + \overline{B}} \neq A + B$. De manera similar, $\overline{\overline{A}\,\overline{B}} \neq AB$.

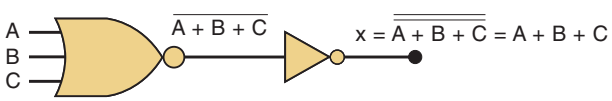
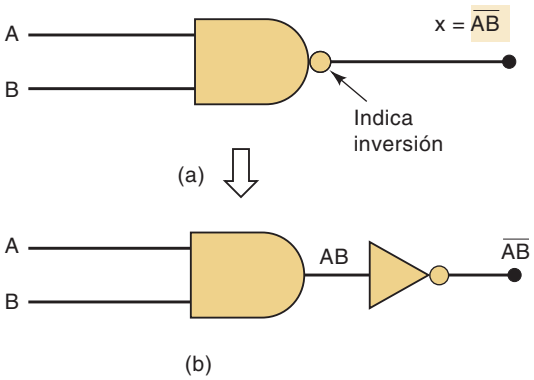


FIGURA 3-21 Ejemplo 3-9.

Compuerta NAND

La figura 3-22(a) muestra el símbolo para una **compuerta NAND** de dos entradas. Es el mismo que el de la compuerta AND más un pequeño círculo en la salida. De nuevo, este pequeño círculo denota la operación de inversión. Así, la compuerta NAND opera igual que una compuerta AND seguida de un INVERSOR, de manera que los circuitos de la figura 3-22(a) y (b) son equivalentes y la expresión de salida para la compuerta NAND es $x = \overline{AB}$.

FIGURA 3-22
(a) Símbolo NAND;
(b) circuito equivalente;
(c) tabla de verdad.



		AND	NAND
A	B	AB	$\overline{AB}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

(c)

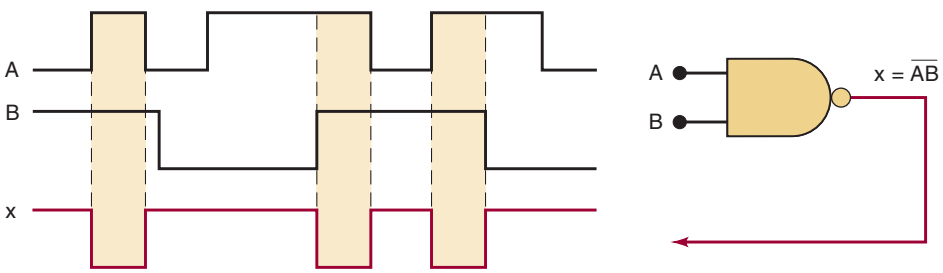
La tabla de verdad de la figura 3-22(c) muestra que la salida de la compuerta NAND es el inverso exacto de la compuerta AND para todas las posibles condiciones de entrada. La salida AND cambia a nivel ALTO sólo cuando todas las entradas se encuentran en ALTO, mientras que la salida NAND cambia a BAJO sólo cuando todas las entradas se encuentran en ALTO. Esta misma característica puede aplicarse a las compuertas NAND que tienen más de dos entradas.

EJEMPLO 3-10

Determine la forma de onda de salida de una compuerta NAND que tiene las entradas que se muestran en la figura 3-23.

FIGURA 3-23 Ejemplo 3-10.





Solución

Una manera es dibujar primero la forma de onda de salida para una compuerta AND y después invertirla. Otra técnica utiliza el hecho de que una salida NAND estará en BAJO sólo cuando todas las entradas estén en ALTO. De esta forma, puede encontrar todos esos intervalos durante los cuales todas las entradas están en ALTO y hacen que la salida NAND esté en BAJO para esos intervalos. La salida estará en ALTO en todos los demás tiempos.

EJEMPLO 3-11

Implemente el circuito lógico que tiene la expresión $x = \overline{AB \cdot (\overline{C + D})}$ utilizando sólo compuertas NOR y NAND.

Solución

El término $(\overline{C + D})$ es la expresión para la salida de una compuerta NOR. A este término se le aplica la operación NAND junto con A y B, y el resultado se invierte; ésta es, desde luego, la operación NAND. Así, el circuito se implementa como se muestra en la figura 3-24. Observe que la compuerta NAND primero aplica un AND a los términos A, B y $(\overline{C + D})$, y después invierte *todo* el resultado.

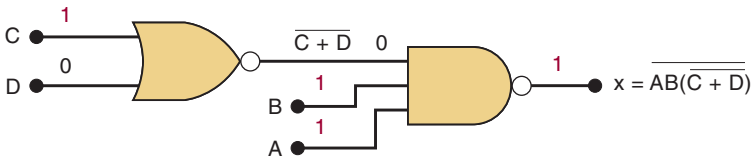


FIGURA 3-24
Ejemplos 3-11 y 3-12.

EJEMPLO 3-12

Determine el nivel de salida de la figura 3-24 cuando $A = B = C = 1$ y $D = 0$.

Solución

En el primer método utilizamos la expresión para x.

$$\begin{aligned} x &= \overline{AB(\overline{C + D})} \\ &= \overline{1 \cdot 1 \cdot (\overline{1 + 0})} \\ &= \overline{1 \cdot 1 \cdot (\overline{1})} \\ &= \overline{1 \cdot 1 \cdot 0} \\ &= \overline{0} = 1 \end{aligned}$$

En el segundo método, anotamos los niveles lógicos de entrada en el diagrama del circuito (los cuales se muestran en gris en la figura 3-24) y seguimos estos niveles a través de cada compuerta, hasta la salida final. La compuerta NOR tiene entradas de 1 y 0 para producir una salida de 0 (un OR hubiera producido una salida de 1). Entonces la compuerta NAND tiene niveles de entrada de 0, 1 y 1 para producir una salida de 1 (al igual que AND hubiera producido una salida de 0).

PREGUNTAS DE REPASO

1. ¿Cuál es el único conjunto de condiciones de entrada que producirá una salida en ALTO en una compuerta NOR de tres entradas?
2. Determine el nivel de salida en la figura 3-24 cuando $A = B = 1, C = D = 0$.
3. Cambie la compuerta NOR de la figura 3-24 por una compuerta NAND y cambie la compuerta NAND por una compuerta NOR. ¿Cuál es la nueva expresión para x?

3-10 TEOREMAS BOOLEANOS

Hemos visto cómo puede utilizarse el álgebra booleana para ayudar a analizar un circuito lógico y expresar su operación en forma matemática. Para continuar con nuestro estudio del álgebra booleana vamos a investigar los diversos **teoremas booleanos** (también conocidos como reglas booleanas) que pueden ayudarnos a simplificar las

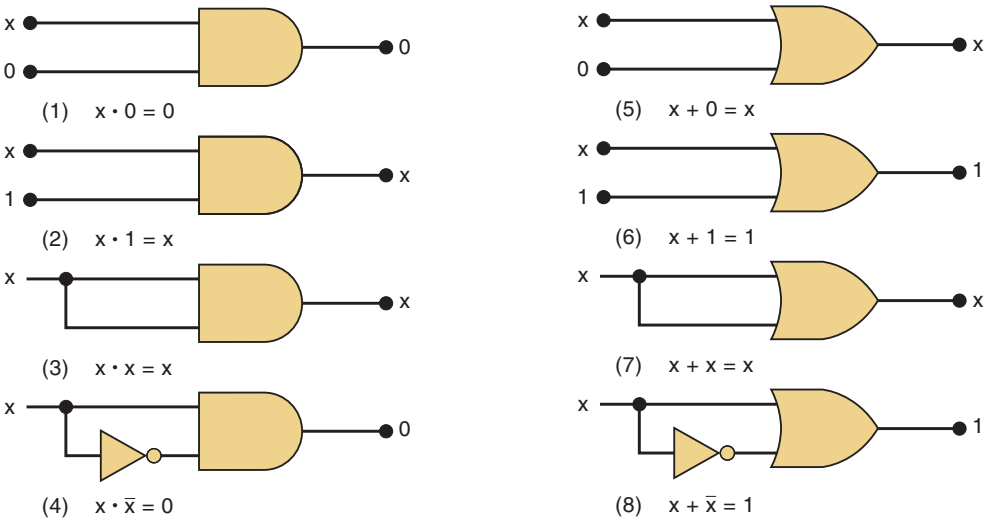


FIGURA 3-25 Teoremas con una sola variable.

expresiones lógicas y los circuitos lógicos. El primer grupo de teoremas se muestra en la figura 3-25. En cada teorema, x es una variable lógica que puede ser un 0 o un 1. Cada teorema se acompaña por el diagrama de un circuito lógico que demuestra su validez.

El teorema (1) establece que si se aplica un AND entre cualquier variable y 0, el resultado es 0. Esto es fácil de recordar ya que la operación AND es como la multiplicación ordinaria, en donde sabemos que cualquier cantidad multiplicada por 0 es igual a 0. También sabemos que la salida de una compuerta AND será 0 siempre que una entrada sea 0, sin importar el nivel en la otra entrada.

El teorema (2) también es obvio si se le compara con la multiplicación ordinaria.

El teorema (3) puede demostrarse si se prueba cada caso. Si $x = 0$, entonces $0 \cdot 0 = 0$; si $x = 1$ entonces $1 \cdot 1 = 1$. Por lo tanto, $x \cdot x = x$.

El teorema (4) puede probarse de la misma forma. No obstante, también podemos razonar que en cualquier momento ya sea x o su inverso, $\bar{x}$, deben estar en el nivel 0 y, por lo tanto, su producto AND siempre será 0.

El teorema (5) es bastante directo, ya que cualquier cantidad que se *sume* con 0 no afecta su valor, ya sea en la suma ordinaria o en la operación OR.

El teorema (6) establece que si se aplica un OR entre cualquier variable y 1, el resultado siempre será 1. Comprobamos esto para ambos valores de x: $0 + 1 = 1$ y $1 + 1 = 1$. También es importante recordar que la salida de una compuerta OR será 1 cuando *cualquiera* de sus entradas sea 1, sin importar el valor de la otra entrada.

El teorema (7) puede probarse si se comprueban ambos valores de x: $0 + 0 = 0$ y $1 + 1 = 1$.

El teorema (8) puede probarse de manera similar, o basta con razonar que en cualquier momento ya sea x o $\bar{x}$ estará en el nivel 1, de manera que siempre se esté aplicando un OR entre 0 y 1, lo cual siempre produce un 1.

Antes de presentar más teoremas, debemos recalcar que cuando se aplican los teoremas (1) a (8) la variable x puede llegar a representar una expresión que contiene más de una variable. Por ejemplo, si tenemos que $A\bar{B}(\overline{A\bar{B}})$, podemos invocar el teorema (4) si hacemos que $x = A\bar{B}$. Por ende, podemos decir que $A\bar{B}(\overline{A\bar{B}})$. Esta misma idea puede aplicarse al uso de cualquiera de los teoremas.

Teoremas con múltiples variables

Los teoremas que presentamos a continuación involucran más de una variable:

$$(9) \qquad x + y = y + x$$
$$(10) \qquad x \cdot y = y \cdot x$$

(11) $x + (y + z) = (x + y) + z = x + y + z$
(12) $x(yz) = (xy)z = xyz$
(13*a*) $x(y + z) = xy + xz$
(13*b*) $(w + x)(y + z) = wy + xy + wz + xz$
(14) $x + xy = x$
(15*a*) $x + \overline{x}y = x + y$
(15*b*) $\overline{x} + xy = \overline{x} + y$

Los teoremas (9) y (10) se conocen como *leyes conmutativas*, ya que indican que el orden en el que se aplican las operaciones OR y AND a dos variables no importa; el resultado es el mismo.

Los teoremas (11) y (12) son las *leyes asociativas*, las cuales establecen que podemos agrupar las variables en una expresión AND o en una expresión OR de cualquier forma que necesitemos.

El teorema (13) es la *ley distributiva*, la cual establece que para expandir una expresión se multiplica término por término, de igual forma que en el álgebra ordinaria. Este teorema también indica que podemos factorizar una expresión. Esto es, si tenemos una suma de dos (o más) términos, cada uno de los cuales contiene una variable común, esta variable común puede factorizarse de igual forma que en el álgebra ordinaria. Por ejemplo, si tenemos la expresión $A\overline{B}C + \overline{A}\,\overline{B}\,\overline{C}$ podemos factorizar la variable B:

$$A\overline{B}C + \overline{A}\,\overline{B}\,\overline{C} = \overline{B}(AC + \overline{A}\,\overline{C})$$

Como otro ejemplo, considere la expresión $ABC + ABD$. Aquí los dos términos tienen en común las variables A y B, por lo que $A \cdot B$ puede factorizarse de ambos términos. Esto es,

$$ABC + ABD = AB(C + D)$$

Los teoremas (9) a (13) son fáciles de recordar y de utilizar, ya que son idénticos a los del álgebra ordinaria. Por otro lado, los teoremas (14) y (15) no tienen contrapartes en el álgebra ordinaria. Para demostrar cada uno de ellos hay que probar todos los casos posibles para x y y. Para ilustrar lo anterior (para el teorema 14) vamos a crear una tabla de análisis para la ecuación $x + xy$, como se muestra a continuación:

x	y	xy	x + xy
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

Observe que el valor de toda la expresión $(x + xy)$ es siempre el mismo que para x.

El teorema (14) también puede demostrarse mediante la factorización y el uso de los teoremas (6) y (2), como se muestra a continuación:

$x + xy = x(1 + y)$
$= x \cdot 1$ [usando el teorema (6)]
$= x$ [usando el teorema (2)]

Todos estos teoremas booleanos pueden ser útiles para simplificar una expresión lógica; es decir, para reducir el número de términos en la expresión. Con esto, la expresión reducida produce un circuito que es menos complejo que el que habría producido la expresión original. Una buena porción del siguiente capítulo estará dedicada al proceso de simplificación de los circuitos. Por ahora, los siguientes

ejemplos servirán para ilustrar cómo pueden aplicarse los teoremas booleanos. **Nota:** encontrará todos los teoremas booleanos en la parte interior de la cubierta posterior del libro.

EJEMPLO 3-13

Simplifique la expresión $y = A\overline{B}D + A\overline{B}\,\overline{D}$.

Solución

Factorice las variables comunes AB mediante el uso del teorema (13):

$$y = A\overline{B}(D + \overline{D})$$

Si utilizamos el teorema (8), el término entre paréntesis es equivalente a 1. Así,

$$\begin{aligned} y &= A\overline{B} \cdot 1 \\ &= A\overline{B} \qquad \text{[utilizando el teorema (2)]} \end{aligned}$$

EJEMPLO 3-14

Simplifique la expresión $z = (\overline{A} + B)(A + B)$.

Solución

Podemos expandir la expresión si multiplicamos los términos [teorema (13)]:

$$z = \overline{A} \cdot A + \overline{A} \cdot B + B \cdot A + B \cdot B$$

Si invocamos el teorema (4), el término $\overline{A} \cdot A = 0$. Además, $B \cdot B = B$ [teorema (13)]:

$$z = 0 + \overline{A} \cdot B + B \cdot A + B = \overline{A}B + AB + B$$

Si factorizamos la variable B [teorema (13)] tenemos que:

$$z = B(\overline{A} + A + 1)$$

Por último, utilizando los teoremas (2) y (6),

$$z = B$$

EJEMPLO 3-15

Simplifique $x = ACD + \overline{A}BCD$.

Solución

Si factorizamos las variables comunes CD, tenemos que

$$x = CD(A + \overline{A}B)$$

Utilizando el teorema (15a) podemos sustituir $A + \overline{A}B$ por $A + B$, de manera que

$$\begin{aligned} x &= CD(A + B) \\ &= ACD + BCD \end{aligned}$$

PREGUNTAS DE REPASO

1. Use los teoremas (13) y (14) para simplificar $y = A\overline{C} + AB\overline{C}$.
2. Use los teoremas (13) y (8) para simplificar $y = \overline{A}\,\overline{B}\,C\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$.
3. Use los teoremas (13) y (15b) para simplificar $y = \overline{A}D + ABD$.

3-11 TEOREMAS DE DEMORGAN

Dos de los teoremas más importantes del álgebra booleana fueron aportación de un gran matemático apellidado DeMorgan. Los **teoremas de DeMorgan** son extremadamente útiles para simplificar expresiones en las cuales se invierte un producto o la suma de variables. Los dos teoremas son:

$$(16) \qquad \overline{(x + y)} = \overline{x} \cdot \overline{y}$$
$$(17) \qquad \overline{(x \cdot y)} = \overline{x} + \overline{y}$$

El teorema (16) establece que cuando se invierte la suma OR de dos variables, es lo mismo que invertir cada variable en forma individual y después aplicar una operación AND a estas dos variables invertidas. El teorema (17) establece que cuando se invierte el producto AND de dos variables, es lo mismo que invertir cada variable en forma individual y después aplicar una operación OR a estas dos variables. Cada uno de los teoremas de DeMorgan puede demostrarse con facilidad si se comprueban todas las posibles combinaciones para x y y. Dejaremos esta comprobación como un ejercicio de final de capítulo.

Aunque estos teoremas están declarados en términos de las variables individuales x e y, son también válidos en situaciones en las que x y/o y son expresiones que contienen más de una variable. Por ejemplo, vamos a aplicarlas a la expresión $\overline{(A\overline{B} + C)}$ como se muestra a continuación:

$$\overline{(A\overline{B} + C)} = \overline{(A\overline{B})} \cdot \overline{C}$$

Aquí utilizamos el teorema (16) y tratamos a $A\overline{B}$ como x y a C como y. El resultado puede simplificarse aún más ya que tenemos un producto $A\overline{B}$ que está invertido. Si utilizamos el teorema (17), la expresión se convierte en

$$\overline{A\overline{B}} \cdot \overline{C} = (\overline{A} + \overline{\overline{B}}) \cdot \overline{C}$$

Ahora, si sustituimos $\overline{\overline{B}}$ por B, al final nos queda

$$(\overline{A} + B) \cdot \overline{C} = \boldsymbol{\overline{A}\,\overline{C}} + \boldsymbol{B\overline{C}}$$

Este resultado final sólo contiene signos de inversión para invertir una sola variable.

EJEMPLO 3-16

Simplifique la expresión $z = \overline{(\overline{A} + C) \cdot (B + \overline{D})}$ en una en la que sólo haya variables individuales invertidas.

Solución

Si utilizamos el teorema (17) y tratamos a $(\overline{A} + C)$ como x y a $(B + \overline{D})$ como y, tenemos que

$$z = \overline{(\overline{A} + C)} + \overline{(B + \overline{D})}$$

En esta expresión se partió el signo inversor grande a la mitad y se cambió el signo AND (·) por un signo OR (+). Ahora el término $(\overline{\overline{A} + C})$ puede simplificarse mediante la aplicación del teorema (16). El término $(\overline{B + \overline{D}})$ también puede simplificarse:

$$\begin{aligned} z &= (\overline{\overline{A} + C}) + (\overline{B + \overline{D}}) \\ &= (\overline{\overline{A}} \cdot \overline{C}) + \overline{B} \cdot \overline{\overline{D}} \end{aligned}$$

Aquí se partieron los signos inversores grandes de cada expresión a la mitad y se cambiaron los (+) por (·). Cancelando las dobles inversiones nos queda por último

$$z = A\overline{C} + \overline{B}D$$

El punto clave en el ejemplo 3-16 es que cuando se utilizan los teoremas de DeMorgan para reducir una expresión, podemos descomponer un signo inversor en cualquier punto de la expresión y cambiar el signo del operador en ese punto por su opuesto (+ se cambia por ·, y viceversa). Este procedimiento se continúa hasta que la expresión se reduce a una en la que sólo haya variables individuales invertidas. A continuación se proporcionan dos ejemplos más.

Ejemplo 1

$$\begin{aligned} z &= \overline{A + \overline{B} \cdot C} \\ &= \overline{A} \cdot (\overline{\overline{B} \cdot C}) \\ &= \overline{A} \cdot (\overline{\overline{B}} + \overline{C}) \\ &= \overline{A} \cdot (B + \overline{C}) \end{aligned}$$

Ejemplo 2

$$\begin{aligned} \omega &= \overline{(A + BC) \cdot (D + EF)} \\ &= (\overline{A + BC}) + (\overline{D + EF}) \\ &= (\overline{A} \cdot \overline{BC}) + (\overline{D} \cdot \overline{EF}) \\ &= [\overline{A} \cdot (\overline{B} + \overline{C})] + [\overline{D} \cdot (\overline{E} + \overline{F})] \\ &= \overline{A}\overline{B} + \overline{A}\overline{C} + \overline{D}\overline{E} + \overline{D}\overline{F} \end{aligned}$$

Los teoremas de DeMorgan pueden extenderse fácilmente a más de dos variables. Por ejemplo, puede demostrarse que

$$\begin{aligned} \overline{x + y + z} &= \overline{x} \cdot \overline{y} \cdot \overline{z} \\ \overline{x \cdot y \cdot z} &= \overline{x} + \overline{y} + \overline{z} \end{aligned}$$

Aquí podemos ver que el signo inversor grande se descompone en *dos* puntos de la expresión y que el signo del operador se cambia por su opuesto. Esto puede extenderse a cualquier número de variables. De nuevo, debemos tener en cuenta que las variables en sí pueden ser expresiones, en lugar de variables individuales. A continuación se muestra otro ejemplo.

$$\begin{aligned} x &= \overline{\overline{AB} \cdot \overline{CD} \cdot \overline{EF}} \\ &= \overline{\overline{AB}} + \overline{\overline{CD}} + \overline{\overline{EF}} \\ &= AB + CD + EF \end{aligned}$$

Implicaciones de los teoremas de DeMorgan

Examinemos los teoremas (16) y (17) desde el punto de vista de los circuitos lógicos. Primero vamos a considerar el teorema (16):

$$\overline{x + y} = \overline{x} \cdot \overline{y}$$

El lado izquierdo de la ecuación puede considerarse como la salida de una compuerta NOR cuyas entradas son x y y. Por otra parte, el lado derecho de la ecuación es el

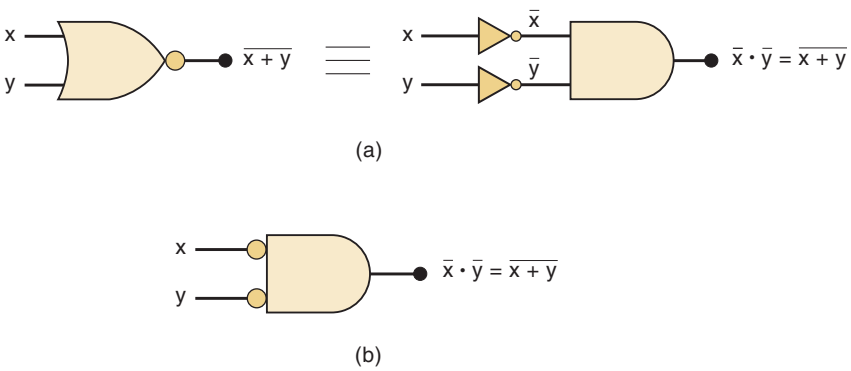


FIGURA 3-26
(a) Circuitos equivalentes implicados por el teorema (16); (b) símbolo alternativo para la función NOR.

resultado de primero invertir ambas variables x y y, y después pasarlas a través de una compuerta AND. Estas dos representaciones son equivalentes y se ilustran en la figura 3-26(a), lo que significa que una compuerta AND con INVERSORES en cada una de sus entradas es equivalente a una compuerta NOR. De hecho, ambos símbolos se utilizan para representar a la función NOR. Cuando se utiliza la compuerta AND con entradas invertidas para representar a la función NOR, por lo general, se dibuja en la forma que se muestra en la figura 3-26(b), en donde los pequeños círculos en las entradas representan la operación de inversión.

Ahora consideremos el teorema (17):

$$\overline{x \cdot y} = \bar{x} + \bar{y}$$

Podemos implementar el lado izquierdo de la ecuación mediante una compuerta NAND con entradas x y y. Para implementar primero el lado derecho se invierten las entradas x e y, y después se pasan a través de una compuerta OR. En la figura 3-27(a) se muestran estos dos símbolos equivalentes. La compuerta OR con INVERSORES en cada una de sus entradas es equivalente a la compuerta NAND.También ambas representaciones se utilizan para la función NAND. Cuando se utiliza la compuerta OR con entradas invertidas para representar a la función NAND, por lo general, se dibuja en la forma que se muestra en la figura 3-27(b), en donde los círculos en las entradas nuevamente representan la operación de inversión.

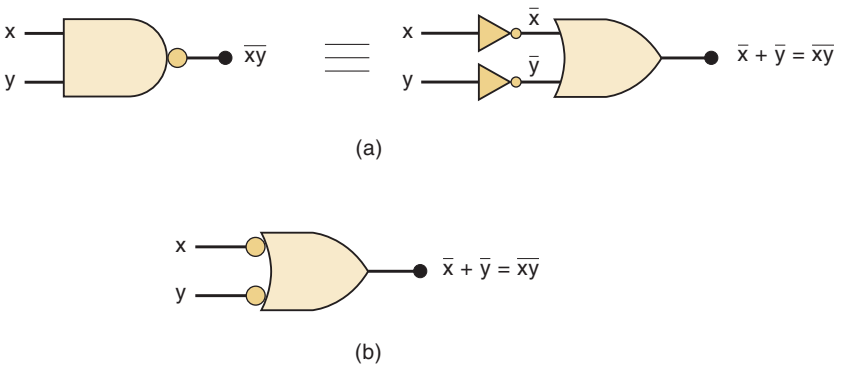


FIGURA 3-27
(a) Circuitos equivalentes implicados por el teorema (17); (b) símbolo alternativo para la función NAND.

EJEMPLO 3-17

Determine la expresión de salida para el circuito de la figura 3-28 y simplifíquelo mediante el uso de los teoremas de DeMorgan.

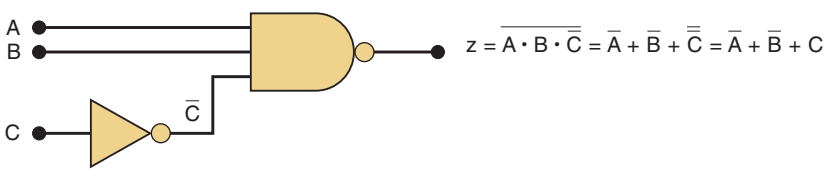


FIGURA 3-28 Ejemplo 3-17.



Solución

La expresión para z es $z = \overline{AB\overline{C}}$. Utilizamos el teorema de DeMorgan para descomponer el signo de inversión grande:

$$z = \overline{A} + \overline{B} + \overline{\overline{C}}$$

Ahora cancelamos la doble inversión en C para obtener

$$z = \overline{A} + \overline{B} + C$$

PREGUNTAS DE REPASO

1. Utilice los teoremas de DeMorgan para convertir la expresión $z = \overline{(A + B) \cdot \overline{C}}$ por una que sólo tenga inversiones de variables individuales.
2. Repita la pregunta 1 para la expresión $y = \overline{R\overline{S}T + \overline{Q}}$.
3. Implemente un circuito que tenga la expresión de salida $z = \overline{A}\,\overline{B}\,C$ utilizando sólo una compuerta NOR y un INVERSOR.
4. Use los teoremas de DeMorgan para convertir $y = \overline{A + \overline{B} + \overline{C}D}$ en una expresión que contenga sólo inversiones de variables individuales.

3-12 UNIVERSALIDAD DE LAS COMPUERTAS NAND Y NOR

Todas las expresiones booleanas consisten de varias combinaciones de las operaciones básicas de OR, AND e INVERSOR. Por lo tanto, cualquier expresión puede implementarse mediante el uso de combinaciones de compuertas OR, AND e INVERSOR. No obstante, es posible implementar cualquier expresión lógica utilizando *sólo* compuertas NAND. Esto se debe a que si las compuertas NAND se combinan en forma apropiada, pueden usarse para realizar cada una de las operaciones booleanas OR, AND e INVERSOR. La figura 3-29 demuestra este concepto.

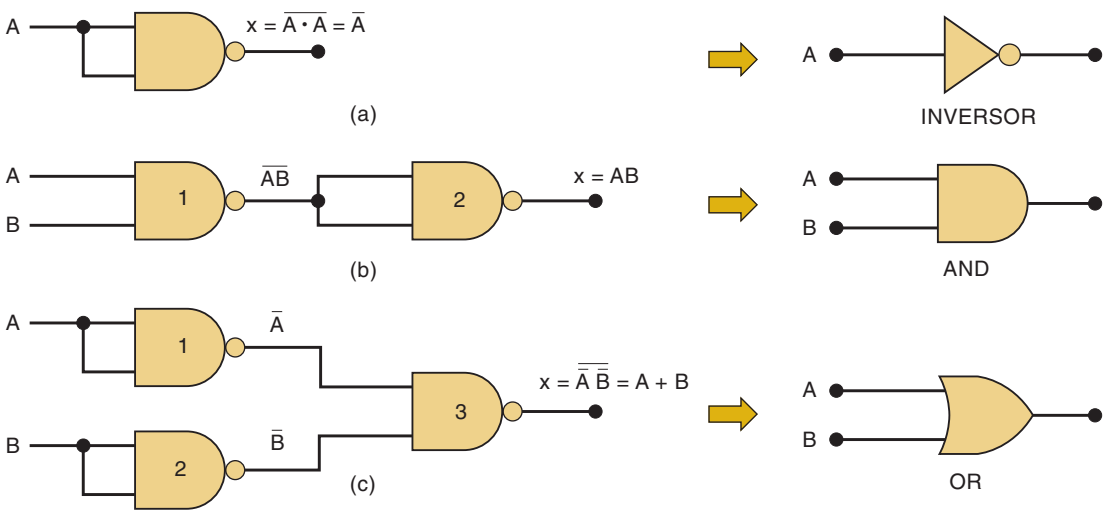


FIGURA 3-29 Las compuertas NAND pueden usarse para implementar cualquier función booleana.

En primer lugar, en la figura 3-29(a) tenemos una compuerta NAND de dos entradas, las cuales se conectaron juntas a propósito, de manera que la variable A se aplique a ambas. En esta configuración, la compuerta NAND actúa tan sólo como INVERSOR, ya que su salida es $x = \overline{A \cdot A} = \overline{A}$.

En la figura 3-29(b) tenemos dos compuertas NAND conectadas de manera que se realice la operación AND. La compuerta NAND 2 se utiliza como INVERSOR para cambiar $\overline{AB}$ por $\overline{\overline{AB}} = AB$, que es la función AND deseada.

Para implementar la operación OR se pueden utilizar compuertas NAND conectadas como se muestra en la figura 3-29(c). Aquí las compuertas NAND 1 y 2 se utilizan como INVERSORES para las entradas, de manera que la salida final sea $x = \overline{\overline{A} \cdot \overline{B}}$, lo cual puede simplificarse como $x = A + B$ mediante el uso del teorema de DeMorgan.

De una manera similar, podemos demostrar que se pueden crear arreglos de compuertas NOR para implementar cualquiera de las operaciones booleanas. Esto se ilustra en la figura 3-30. La parte (a) muestra que una compuerta NOR con sus entradas conectadas entre sí se comporta como un INVERSOR, debido a que la salida es $x = \overline{A + A} = \overline{A}$.

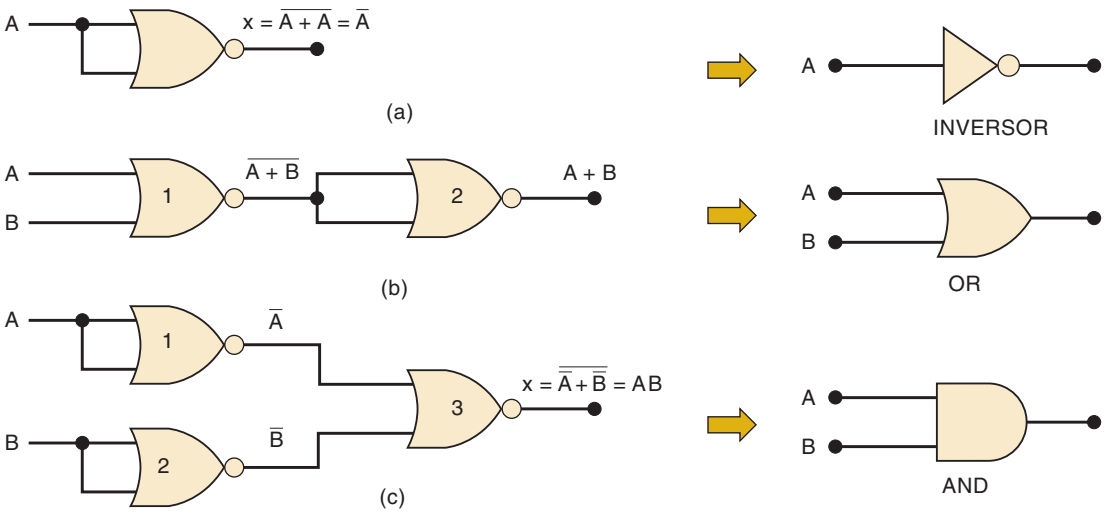


FIGURA 3-30 Las compuertas NOR pueden utilizarse para implementar cualquier operación booleana.

En la figura 3-30(b) hay dos compuertas NOR conectadas de tal forma que se lleve a cabo la operación OR. La compuerta NOR 2 se utiliza como un INVERSOR para cambiar $\overline{A + B}$ por $\overline{\overline{A + B}} = A + B$, que es la función OR deseada.

La operación AND puede implementarse con compuertas NOR, como se muestra en la figura 3-30(c). Aquí las compuertas NOR 1 y 2 se utilizan como INVERSORES para las entradas de manera que la salida final sea $x = \overline{\overline{A} + \overline{B}}$, que puede simplificarse, mediante el uso del teorema de DeMorgan, como $x = A \cdot B$.

Como cualquiera de las operaciones booleanas pueden implementarse con sólo utilizar compuertas NAND, puede construirse cualquier circuito si se utilizan sólo compuertas NAND. Lo mismo aplica para las compuertas NOR. Esta característica de las compuertas NAND y NOR puede ser muy útil en el diseño de circuitos lógicos, como se muestra en el ejemplo 3-18.

EJEMPLO 3-18

En cierto proceso de manufactura, una banda transportadora se apaga cada vez que ocurren determinadas condiciones, las cuales se supervisan y reflejan con base en

los estados de cuatro señales lógicas de la siguiente manera: la señal *A* estará en ALTO siempre que la velocidad de la banda transportadora sea demasiado alta; la señal *B* estará en ALTO cada vez que el recipiente recolector al final de la banda se encuentre lleno; la señal *C* estará en ALTO cuando la tensión de la banda esté demasiado alta; la señal *D* estará en ALTO cuando esté desconectado el sobrepaso manual.

Se necesita un circuito lógico para generar una señal *x* que cambie a ALTO siempre que las condiciones *A* y *B* se presenten al mismo tiempo, o cada vez que las condiciones *C* y *D* se presenten al mismo tiempo. Podemos deducir que la expresión lógica para *x* es $x = AB + CD$. El circuito debe implementarse con la menor cantidad de circuitos integrados que sea posible. Los circuitos integrados TTL que se muestran en la figura 3-31 están disponibles. Cada CI es *cuádruple*, lo que significa que contiene *cuatro* compuertas idénticas en un solo chip.

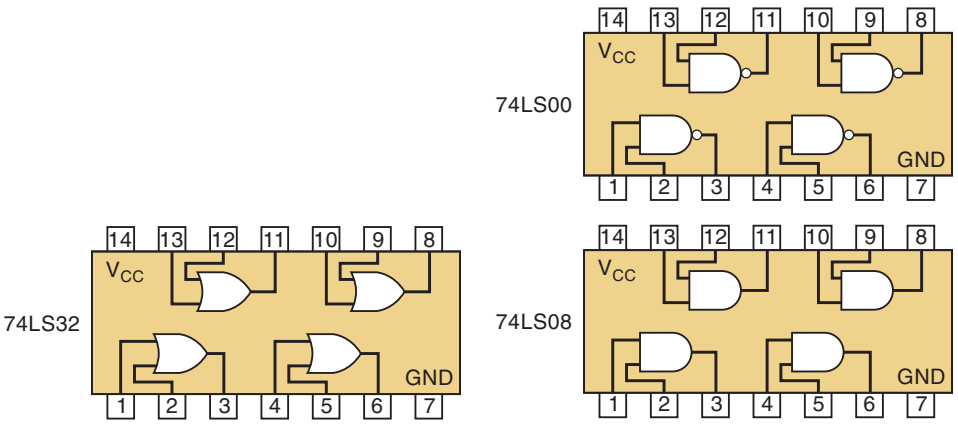


FIGURA 3-31 Circuitos integrados disponibles para el ejemplo 3-18.

Solución

El método directo para implementar la expresión obtenida utiliza dos compuertas AND y una compuerta OR, como se muestra en la figura 3-32(a). Esta implementación utiliza dos compuertas del CI 74LS08 y una sola compuerta del CI 74LS32. Los números entre paréntesis en cada entrada y salida son los números de terminal del CI respectivo, los cuales siempre se muestran en cualquier diagrama de cableado de circuitos lógicos. Para nuestros fines, la mayoría de los diagramas lógicos no mostrarán los números de terminal a menos que se necesiten en la descripción de la operación del circuito.

Se puede lograr otra implementación si se toma el circuito de la figura 3-32(a) y se sustituye cada compuerta AND y OR por la implementación equivalente con la compuerta NAND de la figura 3-29. El resultado se muestra en la figura 3-32(b).

A primera vista, este nuevo circuito parece requerir siete compuertas NAND. No obstante, las compuertas NAND 3 y 5 están conectadas como INVERSORES en serie y pueden eliminarse del circuito, ya que realizan una doble inversión de la señal que sale de la compuerta NAND 1. De manera similar, las compuertas NAND 4 y 6 pueden eliminarse. En la figura 3-32(c) se muestra el circuito final, después de eliminar los dobles INVERSORES.

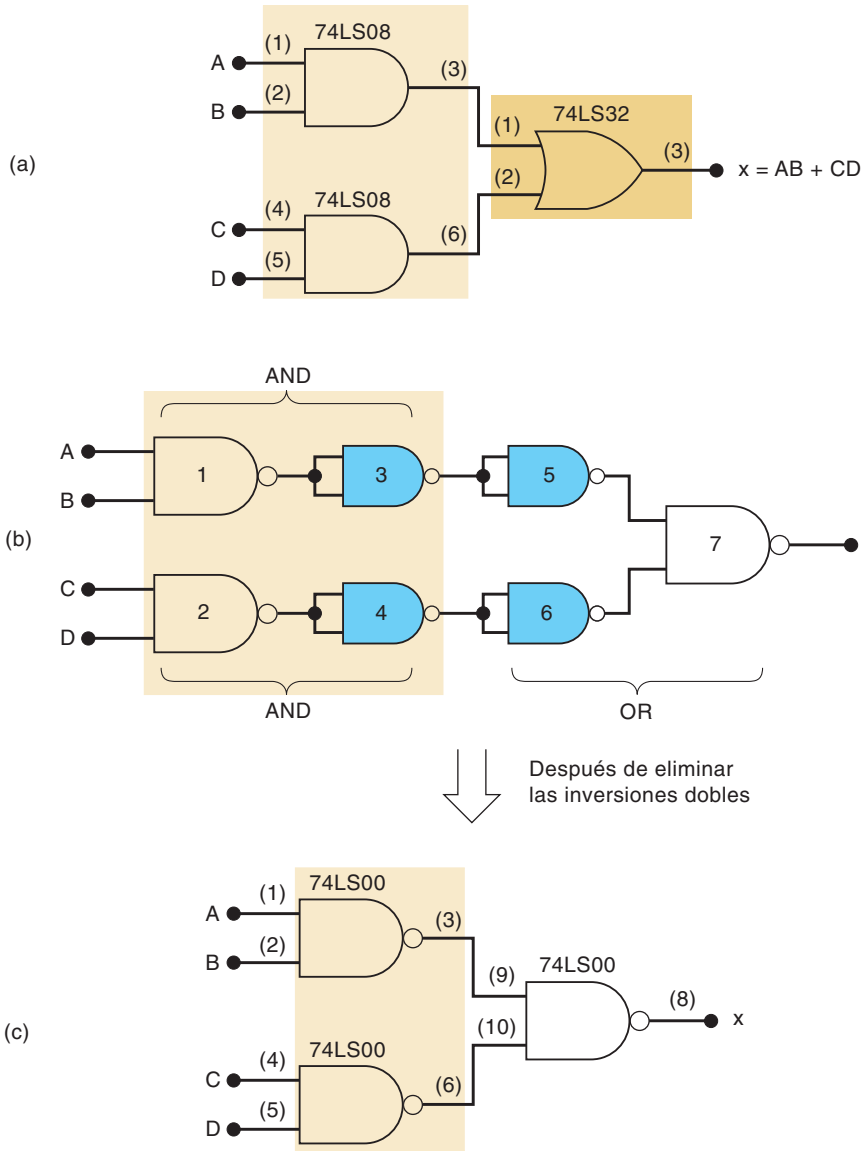
Este circuito final es más eficiente que el de la figura 3-32(a), ya que utiliza tres compuertas NAND de dos entradas que están incluidas en un solo CI, el 74LS00.







FIGURA 3-32 Posibles implementaciones para el ejemplo 3-18.



PREGUNTAS DE REPASO

1. ¿Cuántas maneras tenemos ahora para implementar la operación de inversión en un circuito lógico?
2. Implemente la expresión $x = (A + B)(C + D)$ mediante el uso de compuertas OR y AND. Implemente después la expresión utilizando sólo compuertas NOR, convirtiendo cada compuerta OR y AND a su implementación con NOR de la figura 3-30. ¿Cuál circuito es más eficiente?
3. Escriba la expresión de salida para el circuito de la figura 3-32(c) y utilice los teoremas de DeMorgan para demostrar que es equivalente a la expresión para el circuito de la figura 3-32(a).

3-13 REPRESENTACIONES ALTERNAS DE COMPUERTAS LÓGICAS

Hemos presentado las cinco compuertas lógicas básicas (AND, OR, INVERSOR, NAND y NOR) y los símbolos estándar que se utilizan para representarlas en los diagramas de circuitos lógicos. Aunque tal vez algunos diagramas de circuitos aún

utilicen estos símbolos estándar de manera exclusiva, cada vez es más común encontrar diagramas de circuitos en los que se emplean los **símbolos lógicos alternativos** *además* de los símbolos estándar.

Antes de hablar sobre las razones para utilizar un símbolo alternativo para una compuerta lógica, presentaremos los símbolos alternativos para cada compuerta y demostraremos que son equivalentes a los símbolos estándar. Consulte la figura 3-33; el lado izquierdo muestra el símbolo estándar para cada compuerta lógica y el lado derecho muestra el símbolo alternativo. El símbolo alternativo para cada compuerta se obtiene a partir del símbolo estándar mediante el siguiente proceso:

1. Invertir cada entrada y salida del símbolo estándar. Para ello se agregan burbujas (pequeños círculos) en las entradas y salidas que no tienen burbujas y se quitan las de las entradas y salidas que si tienen.
2. Cambiar el símbolo de la operación de AND a OR, o de OR a AND. En el caso especial del INVERSOR, el símbolo de la operación no se cambia.

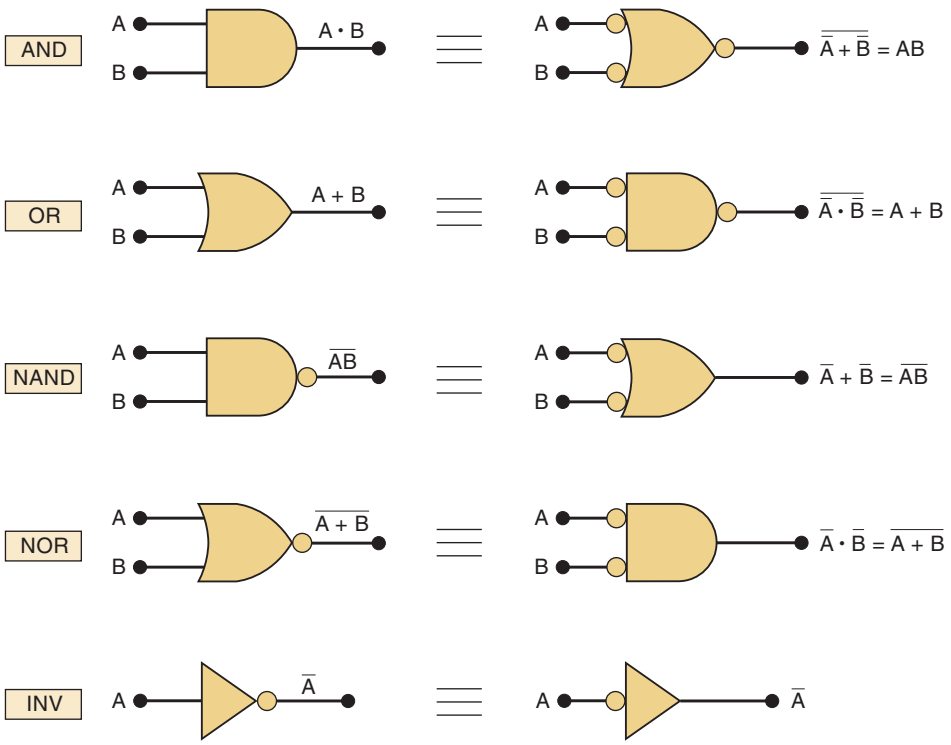


FIGURA 3-33
Símbolos estándar y alternativos para varias compuertas lógicas y para el inversor.

Por ejemplo, el símbolo NAND estándar es un símbolo AND con una burbuja en su salida. Siguiendo los pasos antes mencionados, eliminamos la burbuja de la salida y agregamos una en cada entrada. Después cambiamos el símbolo AND por un símbolo OR. El resultado es un símbolo OR con burbujas en sus entradas.

Podemos demostrar con facilidad que este símbolo alternativo es equivalente al símbolo estándar mediante el uso de los teoremas de DeMorgan y recordando que la burbuja representa una operación de inversión. La expresión de salida del símbolo NAND estándar es $\overline{AB} = \overline{A} + \overline{B}$, que es lo mismo que la expresión de salida para el símbolo alterno. Podemos seguir este mismo procedimiento para cada par de símbolos de la figura 3-33.

Debemos recalcar varios puntos en relación con las equivalencias de los símbolos lógicos:

1. Las equivalencias pueden extenderse a las compuertas con *cualquier* número de entradas.
2. Ninguno de los símbolos estándar tiene burbujas en sus entradas, y todos los símbolos alternativos sí.

3. Los símbolos estándar y alternativos para cada compuerta representan el mismo circuito físico; *no hay diferencia en los circuitos representados por los dos símbolos.*
4. Las compuertas NAND y NOR son inversoras, por lo que tanto los símbolos estándar como los alternativos para estas compuertas tendrán una burbuja *ya sea* en la entrada o en la salida. Las compuertas AND y OR son *no inversoras*, por lo que los símbolos alternativos para cada una de ellas tendrá burbujas *tanto en las* entradas *como en las* salidas.

Interpretación de los símbolos lógicos

Cada uno de los símbolos de las compuertas lógicas de la figura 3-33 ofrece una interpretación única de la manera en que opera la compuerta. Antes de demostrar estas interpretaciones debemos establecer el concepto de los **niveles lógicos activos**.

Cuando una línea de entrada o de salida en el símbolo de un circuito lógico *no tiene burbuja*, se dice que está **activa en ALTO**. Cuando una línea de entrada o de salida *tiene una burbuja*, se dice que está **activa en BAJO**. Por lo tanto, la presencia o ausencia de una burbuja determina el estado activo en BAJO /activo en ALTO respectivamente, de las entradas y la salida de un circuito, y se utiliza para interpretar la operación del mismo.

Para ilustrar lo anterior, la figura 3-34(a) muestra el símbolo estándar para una compuerta NAND. Este símbolo estándar tiene una burbuja en su salida y no tiene burbujas en sus entradas. Por lo tanto, tiene una salida activa en BAJO y entradas activas en ALTO. En consecuencia, la operación lógica que representa este símbolo puede interpretarse de la siguiente manera:

La salida cambia a BAJO sólo cuando *todas* las entradas están en ALTO.

Observe que esto dice que la salida cambiará a su estado activo sólo cuando *todas* las entradas se encuentren en sus estados activos. Se utiliza la palabra *todas* debido al símbolo AND.

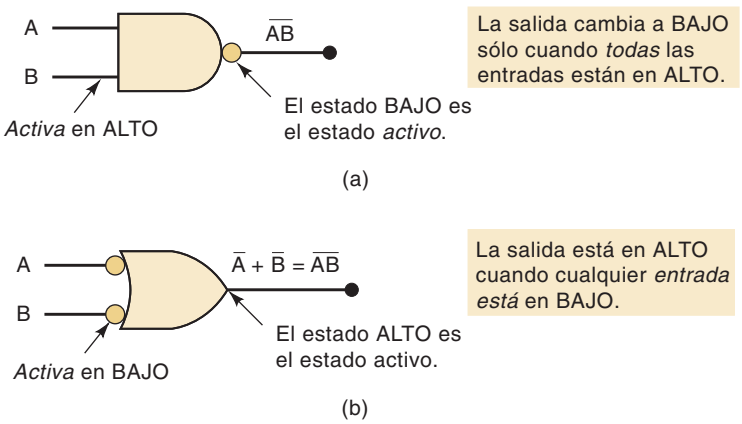


FIGURA 3-34
Interpretación de los dos símbolos de compuertas NAND.

El símbolo alterno para una compuerta NAND que se muestra en la figura 3-34(b) tiene una salida activa en ALTO y entradas activas en BAJO, por lo que su operación puede declararse así:

La salida cambia a ALTO cuando *cualquier entrada* está en BAJO.

Esto quiere decir que la salida estará en su estado activo cuando *cualquiera* de las entradas se encuentre en su estado activo. Se utiliza la palabra *cualquiera* debido al símbolo OR.

Si lo analiza un poco, podrá ver que las dos interpretaciones para los símbolos NAND de la figura 3-34 son maneras distintas de decir lo mismo.

Resumen

En estos momentos tal vez se esté preguntando por qué la necesidad de tener dos símbolos e interpretaciones distintas para cada una de las compuertas lógicas. Esperamos que las razones para ello se aclaren después de que lea la siguiente sección. Por ahora vamos a resumir los puntos importantes en relación con las representaciones de las compuertas lógicas.

1. Para obtener el símbolo alternativo para una compuerta lógica, tome el símbolo estándar y cambie su símbolo de operación (OR a AND, o AND a OR). Cambie también las burbujas tanto en las entradas como en la salida (es decir, elimine las burbujas que estén presentes y agregue otras en donde no haya).
2. Para interpretar la operación de una compuerta lógica, primero observe cuál estado lógico (0 o 1) es el activo para las entradas y cuál es el activo para la salida. Después tome en cuenta que el estado activo de la salida se produce al tener *todas* las entradas en su estado activo (si se utiliza un símbolo AND) o *cualquiera* de las entradas en su estado activo (si se utiliza un símbolo OR).

EJEMPLO 3-19

Proporcione la interpretación de los dos símbolos de compuerta OR.

Solución

Los resultados se muestran en la figura 3-35. Observe que se utiliza la palabra *cualquiera* cuando el símbolo de operación es un OR y la palabra *todas* cuando se usa un AND.

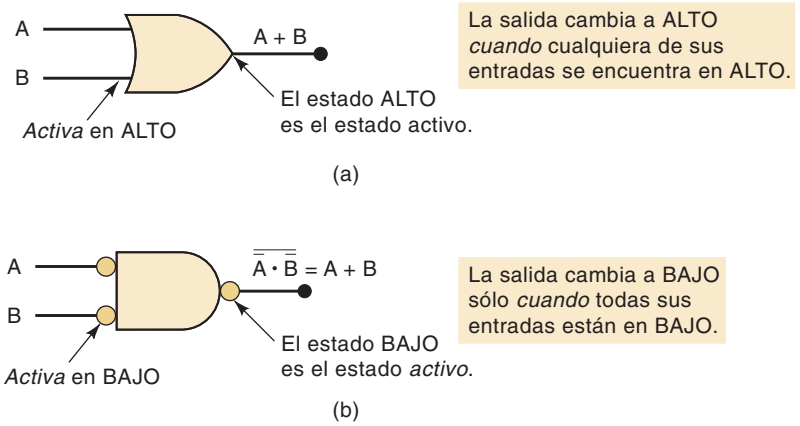


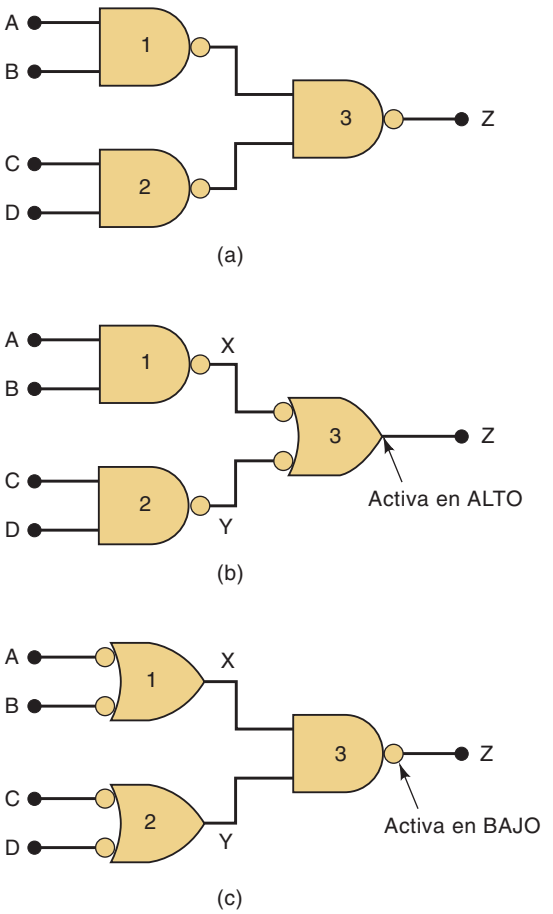
FIGURA 3-35
Interpretación de los dos símbolos de compuerta OR.

PREGUNTAS DE REPASO

1. Escriba la interpretación de la operación que realiza el símbolo estándar de la compuerta NOR de la figura 3-33.
2. Repita la pregunta 1 para el símbolo alternativo de la compuerta NOR.
3. Repita la pregunta 1 para el símbolo alternativo de la compuerta AND.
4. Repita la pregunta 1 para el símbolo estándar de la compuerta AND.

3-14 CUÁL REPRESENTACIÓN DE COMPUERTA SE DEBE USAR

Algunos diseñadores de circuitos lógicos y algunos libros de texto sólo utilizan los símbolos estándar de las compuertas lógicas en los diagramas esquemáticos de sus circuitos. Aunque esta práctica es correcta, no facilita el seguimiento de la operación del circuito. El uso apropiado de los símbolos de compuerta alternativos en el



A	B	C	D	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

(d)

FIGURA 3-36
(a) Circuito original que utiliza símbolos NAND estándar; (b) representación equivalente en la cual la salida Z es activa en ALTO; (c) representación equivalente en la cual la salida Z es activa en BAJO; (e) tabla de verdad.

diagrama puede hacer mucho más clara la operación del circuito. Para ilustrar lo anterior consideremos el ejemplo que se muestra en la figura 3-36.

El circuito de la figura 3-36(a) contiene tres compuertas NAND conectadas para producir una salida Z que depende de las entradas *A*, *B*, *C* y *D*. El diagrama del circuito utiliza el símbolo estándar para cada una de las compuertas NAND. Aunque la lógica de este diagrama es correcta, no facilita una comprensión de la manera en que funciona el circuito. Sin embargo, las representaciones que se muestran en las figuras 3-36(b) y (c) pueden analizarse con más facilidad para determinar la operación del circuito.

La representación de la figura 3-36(b) se obtiene a partir del diagrama del circuito original, sustituyendo la compuerta NAND 3 con su símbolo alternativo. En este diagrama la salida *Z* se toma del símbolo de una compuerta NAND que tiene una salida activa en ALTO. Por ende, podemos decir que *Z* cambiará a ALTO cuando *X* o *Y* estén en BAJO. Ahora, como *X* y *Y* aparecen cada una en la salida de símbolos NAND que tienen salidas activas en BAJO, podemos decir que *X* cambiará a BAJO sólo si $A = B = 1$, y que *Y* cambiará a BAJO sólo si $C = D = 1$. Ahora podemos describir la operación del circuito de la siguiente manera:

> **La salida *Z* cambiará a ALTO siempre que $A = B = 1$ o cuando $C = D = 1$ (o en ambos casos).**

Esta descripción puede traducirse al formato de tablas de verdad si se establece $Z = 1$ para aquellos casos en los que $A = B = 1$ y para aquellos casos en los que $C = D = 1$. Para todos los demás casos, *Z* se hace 0. La tabla de verdad resultante se muestra en la figura 3-36(d).

La representación de la figura 3-36(c) se obtiene a partir del diagrama del circuito original, sustituyendo las compuertas NAND 1 y 2 por sus símbolos alternativos.

En esta representación equivalente, la salida Z se toma de una compuerta NAND en la cual su salida sea activa en BAJO. Por lo tanto, podemos decir que Z cambiará a BAJO sólo cuando $X = Y = 1$. Como X y Y son salidas activas en ALTO, podemos decir que X cambiará a ALTO cuando A o B estén en BAJO, y que Y cambiará a ALTO cuando C o D estén en BAJO. De todo esto podemos concluir que la operación del circuito es la siguiente:

> **La salida *Z* cambiará a BAJO sólo cuando *A* o *B* estén en BAJO y *C* o *D* estén en BAJO.**

Esta descripción puede traducirse al formato de tablas de verdad si hacemos que $Z = 0$ para todos los casos en los que cuando menos una de las entradas A o B esté en BAJO al mismo tiempo que cuando menos una de las entradas C o D esté en BAJO. Para todos los demás casos, Z se hace 1. La tabla de verdad resultante es la misma que la que obtuvimos para el diagrama del circuito de la figura 3-36(b).

¿Cuál diagrama de circuito debe usarse?

La respuesta a esta pregunta depende de la función específica que vaya a realizar la salida del circuito. Si se va a utilizar para producir cierta acción (por ejemplo, encender un LED o activar otro circuito lógico) cuando la salida Z cambia al estado 1, entonces decimos que Z debe ser activa en ALTO, y debemos usar el diagrama del circuito de la figura 3-36(b). Por otro lado, si el circuito se va a utilizar para producir cierta acción cuando Z cambia al estado 0, entonces Z debe ser activa en BAJO, y debemos usar el diagrama de la figura 3-36(c).

Desde luego que habrá situaciones en las que se utilicen *ambos* estados de la salida para producir distintas acciones, y cualquiera de ellos podría considerarse como el estado activo. Para estos casos puede utilizarse cualquiera de las dos representaciones del circuito.

Colocación de las burbujas

Consulte la representación del circuito de la figura 3-36(b) y observe que se seleccionaron los símbolos para las compuertas NAND 1 y 2 de manera que tengan salidas activas en BAJO, para hacer que coincidan con las entradas activas en BAJO de la compuerta NAND 3. Consulte la representación del circuito de la figura 3-36(c) y observe que se seleccionaron los símbolos para las compuertas NAND 1 y 2 de manera que tengan salidas activas en ALTO, para hacer que coincidan con las entradas activas en ALTO de la compuerta NAND 3. Esto nos conduce a la siguiente regla general para preparar diagramas esquemáticos de circuitos lógicos:

> **Siempre que sea posible, seleccione símbolos de compuerta de manera que las salidas con burbuja se conecten a las entradas con burbuja, y las salidas sin burbuja se conecten a las entradas sin burbuja.**

Los siguientes ejemplos demostrarán cómo aplicar esta regla.

EJEMPLO 3-20

El circuito lógico de la figura 3-37(a) se utiliza para activar una alarma cuando la salida Z cambie a ALTO. Modifique el diagrama del circuito de manera que represente la operación del mismo de una manera más eficiente.

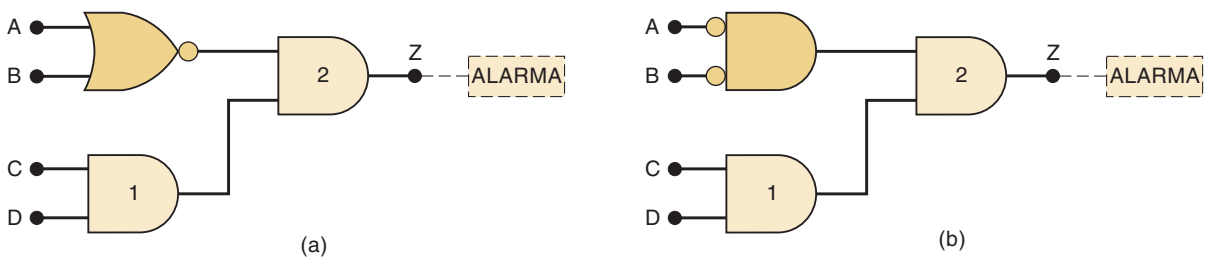


FIGURA 3-37 Ejemplo 3-20.

Solución

Como la condición $Z = 1$ activa la alarma, Z debe ser activa en ALTO. Por ende, el símbolo de la compuerta AND 2 no tiene que cambiarse. El símbolo de la compuerta NOR debe cambiarse por su símbolo alternativo con una salida sin burbuja (activa en ALTO) para que concuerde con la entrada sin burbuja de la compuerta AND 2, como se muestra en la figura 3-37(b). Observe que ahora el circuito tiene salidas sin burbuja conectadas a las entradas sin burbuja de la compuerta 2.

EJEMPLO 3-21

Cuando la salida del circuito lógico de la figura 3-38(a) cambia a BAJO, activa otro circuito lógico. Modifique el diagrama del circuito para representar la operación del mismo más eficazmente.

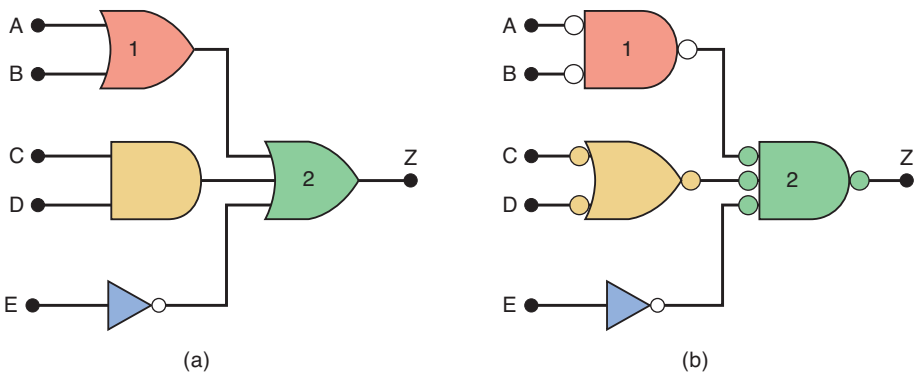


FIGURA 3-38 Ejemplo 3-21.

Solución

Como Z debe ser activa en BAJO, debemos cambiar el símbolo para la compuerta OR 2 por su símbolo alterno, como se muestra en la figura 3-38(b). El nuevo símbolo de la compuerta OR 2 tiene entradas con burbujas, por lo que los símbolos de la compuerta AND y de la compuerta OR 1 deben cambiarse por símbolos que tengan salidas con burbuja, como se muestra en la figura 3-38(b). El INVERSOR ya tiene una salida con burbuja. Ahora el circuito tiene todas sus salidas con burbuja conectadas a las entradas con burbuja de la compuerta OR 2.

Análisis de circuitos

Cuando se dibuja el diagrama esquemático de un circuito lógico utilizando las reglas que hemos seguido en estos ejemplos, es mucho más fácil para un ingeniero, técnico o estudiante seguir el flujo de la señal a través del circuito y determinar las condiciones de entrada necesarias para activar la salida. Ilustraremos esto en los siguientes ejemplos que, por coincidencia, utilizan diagramas de circuitos tomados de los diagramas esquemáticos lógicos de una microcomputadora real.

EJEMPLO 3-22

El circuito lógico de la figura 3-39 genera una salida llamada *MEM* que se utiliza para activar los circuitos integrados de memoria en cierta microcomputadora. Determine las condiciones de entrada necesarias para activar a *MEM*.

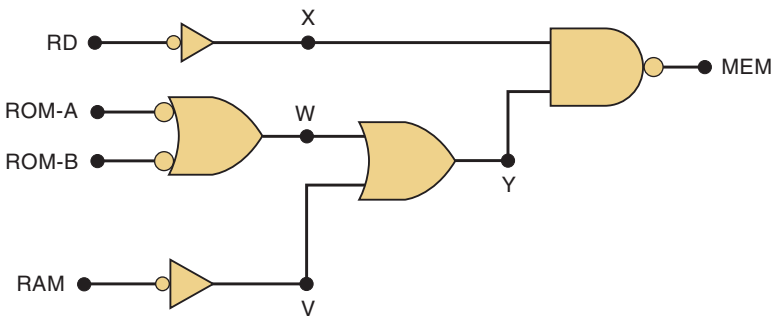


FIGURA 3-39 Ejemplo 3-22.

Solución

Una manera de hacerlo sería escribir la expresión para *MEM* en términos de las entradas *RD*, *ROM-A*, *ROM-B* y *RAM*, y evaluarla para las 16 combinaciones posibles de estas entradas. Aunque este método sí funcionaría, se requeriría mucho más trabajo del necesario.

Un método más eficiente es interpretar el diagrama del circuito utilizando las ideas que hemos desarrollado en las últimas dos secciones. Éstos son los pasos:

1. *MEM* es activa en BAJO y cambará a BAJO sólo cuando *X* y *Y* estén en ALTO.
2. *X* estará en ALTO sólo cuando *RD* = 0.
3. *Y* estará en ALTO cuando *W* o *V* estén en ALTO.
4. *V* estará en ALTO cuando *RAM* = 0.
5. *W* estará en ALTO cuando *ROM-A* o *ROM-B* = 0.
6. Con todo esto podemos deducir que *MEM* cambiará a BAJO sólo cuando *RD* = 0 y cuando menos una de las tres entradas *ROM-A*, *ROM-B* o *RAM* esté en BAJO.

EJEMPLO 3-23

El circuito lógico de la figura 3-40 se utiliza para controlar el motor de control del eje de una unidad de disco flexible cuando la microcomputadora envía o recibe datos hacia o desde el disco. El circuito encenderá el motor cuando *UNIDAD* = 1. Determine las condiciones necesarias de entrada para encender el motor.

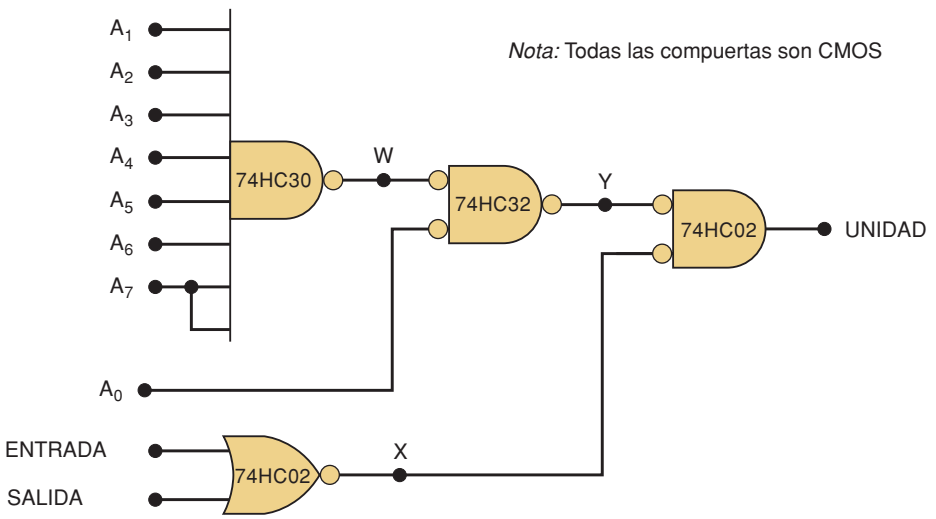


FIGURA 3-40 Ejemplo 3-23.

Solución

Una vez más, interpretaremos el diagrama paso por paso:

1. *UNIDAD* estará activa en BAJO y cambiará a ALTO sólo cuando $X = Y = 0$.
2. *X* estará en BAJO cuando *ENTRADA* o *SALIDA* estén en ALTO.
3. *Y* estará en BAJO sólo cuando $W = 0$ y $A_0 = 0$.
4. *W* estará en BAJO sólo cuando todas las entradas de la A_1 a la A_7 estén en ALTO.
5. De todo esto podemos deducir que *UNIDAD* estará en ALTO cuando $A_1 = A_2 = A_3 = A_4 = A_5 = A_6 = A_7 = 1$ y $A_0 = 0$, y cuando ni *ENTRADA* o *SALIDA* ni ambas estén en 1.

Observe el extraño símbolo para la compuerta NAND CMOS de ocho entradas (74HC30); observe además que la señal A_7 está conectada a dos de las entradas NAND.

Niveles asignados

Hemos estado describiendo las señales lógicas como activas en BAJO o activas en ALTO. Por ejemplo, la salida *MEM* en la figura 3-39 es activa en BAJO y la salida *UNIDAD* en la figura 3-40 es activa en ALTO, ya que son los estados de salida que hacen que ocurra algo. De manera similar, la figura 3-40 tiene las entradas de la A_0 a la A_7 activas en ALTO y la entrada A_0 activa en BAJO.

Cuando una señal lógica se encuentra en su estado activo, podemos decir que está **asignada**. Por ejemplo, cuando decimos que la entrada A_0 está asignada, significa que se encuentra en su estado de activa en BAJO. Cuando una señal no se encuentra en su estado activo, se dice que está **no asignada**. Por ende, cuando decimos que *UNIDAD* está no asignada, significa que se encuentra en su estado inactivo (bajo).

Es evidente que los términos *asignado* y *no asignado* son sinónimos de *activo* e *inactivo*, respectivamente:

asignado = activo

no asignado = inactivo

Ambos conjuntos de términos son de uso común en el campo digital, por lo que usted debe ser capaz de reconocer ambas formas de describir el estado activo de una señal lógica.

Etiquetado de señales lógicas activas en BAJO

El uso de una barra superior para etiquetar las señales activas en BAJO se ha convertido en práctica común. La barra superior sirve como otra indicación de que la señal es activa en BAJO; desde luego que la ausencia de una barra superior indica que la señal es activa en ALTO.

Como ejemplo, todas las señales de la figura 3-39 son activas en BAJO y pueden etiquetarse de la siguiente manera:

$$\overline{RD}, \quad \overline{ROM\text{-}A}, \quad \overline{ROM\text{-}B}, \quad \overline{RAM}, \quad \overline{MEM}$$

Recuerde que la barra superior es sólo una manera de enfatizar que estas señales son activas en BAJO. Emplearemos esta convención para etiquetar las señales lógicas siempre que sea apropiado.

Etiquetado de señales biestado

Es muy común que una señal de salida tenga dos estados activos; es decir, que tiene una función importante en el estado ALTO y otra en el estado BAJO. Es costumbre etiquetar dichas señales de manera que ambos estados activos sean aparentes. Un ejemplo común es la señal de lectura/escritura $RD/\overline{WR}$, la cual se interpreta de la siguiente manera: cuando esta señal está en ALTO se realiza la operación de lectura (*RD*); cuando está en BAJO se realiza la operación de escritura (*WR*).

PREGUNTAS DE REPASO

1. Use el método de los ejemplos 3-22 y 3-23 para determinar las condiciones de entrada necesarias para activar la salida del circuito de la figura 3-37(b).
2. Repita la pregunta 1 para el circuito de la figura 3-38(b).
3. ¿Cuántas compuertas NAND se muestran en la figura 3-39?
4. ¿Cuántas compuertas NOR se muestran en la figura 3-40?
5. ¿Cuál será el nivel de salida en la figura 3-38(b) cuando todas las entradas estén asignadas?
6. ¿Qué entradas se requieren para asignar la salida de alarma en la figura 3-37(b)?
7. ¿Cuál de las siguientes señales es activa en BAJO: *RD*, $\overline{W}$, $R/\overline{W}$?

3-15 SÍMBOLOS LÓGICOS DEL ESTÁNDAR IEEE/ANSI

Los símbolos lógicos que hemos utilizado hasta ahora en este capítulo son los símbolos estándar *tradicionales* utilizados en la industria digital durante muchos, muchos años. Estos símbolos tradicionales utilizan una forma distintiva para cada compuerta lógica. En 1984 se desarrolló un estándar más reciente para los símbolos lógicos; a éste se le conoce como Estándar **IEEE/ANSI** 91-1984 para símbolos lógicos. El estándar IEEE/ANSI utiliza símbolos rectangulares para representar todas las compuertas y circuitos lógicos. Una *notación de dependencia* especial dentro del símbolo rectangular indica cómo dependen las salidas del dispositivo de sus entradas. La figura 3-41 muestra los símbolos IEEE/ANSI a un lado de los símbolos tradicionales para las compuertas lógicas básicas. Observe los siguientes puntos:

1. Los símbolos rectangulares utilizan un pequeño triángulo rectángulo (◺) en lugar de la pequeña burbuja de los símbolos tradicionales para indicar la inversión del nivel lógico. La presencia o ausencia del triángulo también indica si una entrada o salida es activa en nivel BAJO o ALTO.

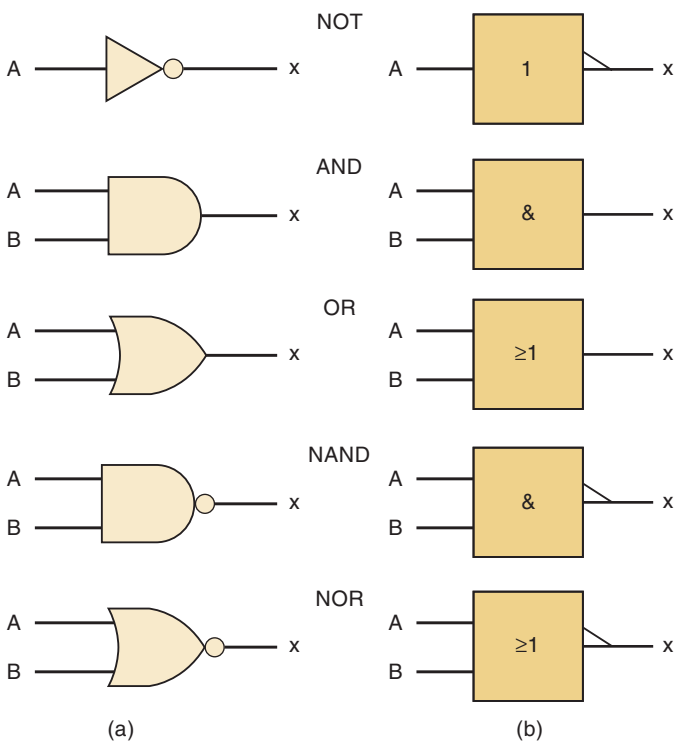


FIGURA 3-41
Símbolos lógicos estándar:
(a) tradicionales;
(b) IEEE/ANSI.

2. Una notación especial dentro de cada símbolo rectangular describe la relación lógica entra las entradas y la salida. El "1" dentro del símbolo INVERSOR denota un dispositivo con sólo *una* entrada; el triángulo en la salida indica que ésta cambiará al estado activo en BAJO cuando la entrada se encuentre en su estado activo en ALTO. El "&" dentro del símbolo AND significa que la salida cambiará a su estado activo en ALTO cuando *todas* las entradas se encuentren en el estado activo en ALTO. El "≥" dentro de la compuerta OR indica que la salida cambiará a su estado activo (ALTO) siempre que *cualquiera* de las entradas se encuentre en su estado activo (ALTO).
3. Los símbolos rectangulares para las compuertas NAND y NOR son los mismos que para las compuertas AND y OR, respectivamente, con la adición del pequeño triángulo inversor en la salida.

¿Tradicional o IEEE/ANSI?

El estándar IEEE/ANSI no ha sido muy aceptado todavía en el campo digital, aunque tal vez se encuentre con él en algunos diagramas esquemáticos de equipos recientes. La mayoría de los libros de datos de CI digitales incluyen los símbolos tradicionales y también los de IEEE/ANSI, ya que es posible que este estándar, más reciente, se utilice cada vez más. En la mayoría de los diagramas de circuitos de este libro emplearemos los símbolos tradicionales.

PREGUNTAS DE REPASO

1. Dibuje todas las compuertas lógicas básicas tanto con los símbolos tradicionales como con los símbolos IEEE/ANSI.
2. Dibuje el símbolo IEEE/ANSI para una compuerta NOR con la salida activa en ALTO.

3-16 RESUMEN DE LOS MÉTODOS PARA DESCRIBIR CIRCUITOS LÓGICOS

Los temas que hemos visto en este capítulo están centrados alrededor de sólo tres funciones lógicas simples que conocemos como AND, OR y NOT. El concepto no es nuevo, ya que todos usamos estas funciones lógicas en nuestra vida diaria cuando tomamos decisiones. He aquí algunos ejemplos lógicos. Si está lloviendo O (OR) el periódico dice que podría llover, entonces llevaré mi paraguas. Si recibo mi cheque de nómina hoy Y (AND) llego al banco, entonces tendré dinero para gastar esta tarde. Si tengo una calificación aprobatoria en teoría Y (AND) NO (NOT) he fallado en el laboratorio, entonces aprobaré mi clase de circuitos digitales. En estos momentos tal vez usted se pregunte por qué hemos invertido tantos esfuerzos para describir estos conceptos familiares. La respuesta puede resumirse en dos puntos clave:

1. Debemos ser capaces de representar estas decisiones lógicas.
2. Debemos ser capaces de combinar estas funciones lógicas y de implementar un sistema de toma de decisiones.

Hemos aprendido a representar cada una de las funciones lógicas básicas mediante el uso de:

Declaraciones lógicas en nuestro propio lenguaje.
Tablas de verdad.
Símbolos lógicos gráficos tradicionales.

Símbolos lógicos del estándar IEEE/ANSI.
Expresiones de álgebra booleana.
Diagramas de tiempos.

EJEMPLO 3-24

La siguiente expresión describe la manera en que un circuito lógico necesita operar para poder controlar un indicador de advertencia del cinturón de seguridad de un automóvil.

Si el conductor está presente Y (AND) el conductor NO (NOT) tiene el cinturón enganchado Y (AND) el interruptor de encendido está activado, ENTONCES encender la luz de advertencia.

Describa el circuito utilizando álgebra booleana, diagramas esquemáticos con símbolos lógicos, tablas de verdad y diagramas de tiempos.

Solución

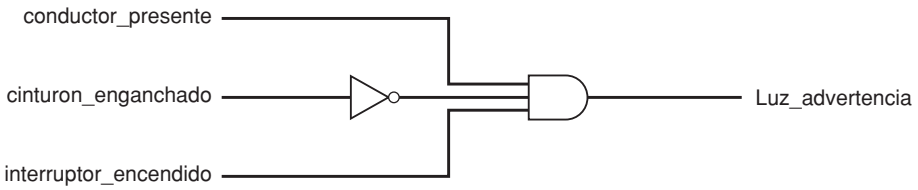
Vea la figura 3-42.

Expresión booleana

luz_advertencia 5 conductor_presente • $\overline{\text{cinturon_enganchado}}$ • interruptor_encendido

(a)

Diagrama esquemático



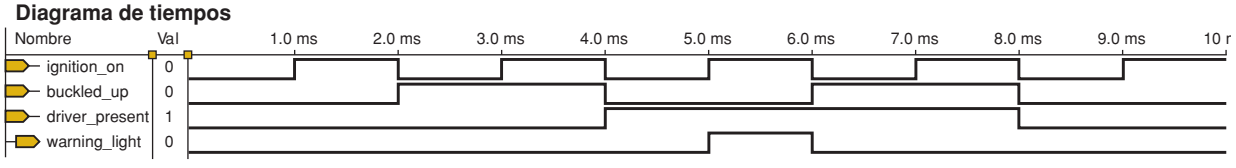
(b)

Tabla de verdad

conductor_presente	cinturon_enganchado	interruptor_encendido	Luz_advertencia
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

(c)

Diagrama de tiempos



(d)

FIGURA 3-42 Métodos para describir circuitos lógicos: (a) expresión booleana; (b) diagrama esquemático; (c) tablas de verdad; (d) diagrama de tiempos.

La figura 3-42 muestra cuatro maneras distintas de representar el circuito lógico que se describió verbalmente en el problema del ejemplo 3-24. Hay muchas otras formas en que podríamos representar la lógica para esta decisión. Como ejemplo podríamos idear todo un conjunto nuevo de símbolos gráficos, o enunciar la relación lógica en un idioma diferente como francés o japonés. Desde luego que no podemos cubrir todas las formas posibles de describir un circuito lógico, pero debemos comprender los métodos más comunes para poder comunicarnos con otras personas en esta profesión. Lo que es más, ciertas situaciones son más fáciles de describir si utilizamos un método en lugar de otro. En algunos casos una imagen vale más que mil palabras; en otros las palabras son lo bastante concisas y se comunican con más facilidad a los demás. El punto importante aquí es que necesitamos formas de describir y comunicar la operación de los sistemas digitales.

PREGUNTAS DE REPASO

1. Nombre cinco formas de describir la operación de los circuitos lógicos.

3-17 COMPARACIÓN ENTRE LENGUAJES DE DESCRIPCIÓN Y LENGUAJES DE PROGRAMACIÓN*

Las tendencias recientes en el campo de los sistemas digitales están a favor de la descripción de los circuitos digitales mediante el lenguaje basado en texto. Tal vez usted haya notado que cada método de descripción de la figura 3-42 presenta obstáculos para introducir los datos a la computadora, ya sea por las barras superiores, los símbolos, el formato o el dibujo de líneas. En esta sección empezaremos a conocer algunas de las herramientas más avanzadas que utilizan los profesionales en el área de sistemas digitales para describir los circuitos que implementan sus ideas. A estas herramientas se les conoce como **lenguajes de descripción de hardware (HDL)**. Aún con las poderosas computadoras que tenemos actualmente, no es posible describir un circuito lógico en lenguaje común (como español o inglés) y esperar que la computadora lo entienda. Las computadoras necesitan un lenguaje definido con más rigidez. En este libro nos enfocaremos en dos lenguajes: el **lenguaje de descripción de hardware de Altera (AHDL)** y el **lenguaje de descripción de hardware para circuitos integrados de muy alta velocidad (VHDL)**.

VHDL y AHDL

VHDL no es un lenguaje nuevo. El Departamento de Defensa de los Estados Unidos lo desarrolló a principios de la década de 1980 como una forma concisa de documentar los diseños en el proyecto de circuitos integrados de muy alta velocidad (VHSIC). Como era demasiado adjuntar HDL a este acrónimo (incluso hasta para los militares), el lenguaje se abrevió como VHDL. Se desarrollaron programas de computadora para tomar los archivos de lenguaje VHDL y simular la operación de los circuitos. Con el crecimiento de los dispositivos lógicos programables complejos en los sistemas digitales, VHDL ha evolucionado para convertirse en uno de los principales lenguajes de descripción de hardware de alto nivel para diseñar e implementar circuitos digitales (síntesis). El IEEE estandarizó este lenguaje, con lo cual se hizo atractivo para los ingenieros, así como para los fabricantes de herramientas de software que traducen los diseños a los patrones de bits utilizados para programar los dispositivos reales.

AHDL es un lenguaje desarrollado por Altera Corporation para ofrecer una forma conveniente de configurar los dispositivos lógicos que ofrecen. Altera fue una de las primeras compañías en introducir dispositivos lógicos que pueden reconfigurarse

* Se pueden omitir todas las secciones que hablan sobre los lenguajes de descripción de hardware sin perder la continuidad en el balance de los capítulos del 1 al 12.

mediante electrónica. A estos dispositivos se les conoce como **dispositivos lógicos programables (PLDs)**. A diferencia del VHDL, este lenguaje no está diseñado para usarse como un lenguaje universal para describir cualquier circuito lógico, sino para programar sistemas digitales complejos en PLDs de Altera, en un lenguaje que se perciba en general como más sencillo de aprender, pero que es muy similar al VHDL. También cuenta con características que aprovechan por completo la arquitectura de los dispositivos de Altera. En todos los ejemplos de este libro utilizaremos el software Altera MAX +PLUS II o Quartus II para desarrollar archivos de diseño en AHDL y VHDL. Podrá ver la ventaja de usar el sistema de desarrollo de Altera para ambos lenguajes cuando programe un dispositivo real. El sistema de Altera facilita el desarrollo de circuitos en forma considerable; además, contiene todas las herramientas necesarias para traducir el archivo de diseño de HDL a uno listo para cargarse en un PLD de Altera. También le permite desarrollar bloques de construcción mediante la introducción de diagramas esquemáticos, AHDL, VHDL y otros métodos, para después interconectar esos bloques y formar un sistema completo.

Hay otros HDLs disponibles que son más adecuados para programar dispositivos lógicos más simples. Después de que, con la ayuda de este libro, aprenda los fundamentos de AHDL o VHDL, le parecerá fácil utilizar cualquiera de los otros lenguajes.

Lenguajes de programación de computadoras

Es importante diferenciar entre los lenguajes de descripción de hardware, que están diseñados para describir la configuración de hardware de un circuito, y los lenguajes de programación que representan una secuencia de instrucciones que deben ser llevadas a cabo por una computadora para realizar cierta tarea. En ambos casos utilizamos un *lenguaje para programar* un dispositivo. No obstante, las computadoras son sistemas digitales complejos que se fabrican a partir de circuitos lógicos. Las computadoras operan mediante el seguimiento de una lista de tareas (es decir, instrucciones o "el programa"), cada una de las cuales debe realizarse en un orden secuencial. La velocidad de operación se determina con base en la rapidez de la computadora para ejecutar cada instrucción. Por ejemplo, si una computadora fuera a responder a cuatro entradas distintas, requeriría cuando menos cuatro instrucciones (tareas secuenciales) para detectar e identificar cuál entrada cambió de estado. Por otro lado, la velocidad de un circuito lógico digital está limitada sólo por la rapidez con la que la circuitería pueda cambiar las salidas en respuesta a los cambios en las entradas. Supervisa todas las entradas en forma **concurrente** (al mismo tiempo) y responde a cualquier cambio.

La siguiente analogía le ayudará a comprender la diferencia entre la operación de una computadora y la operación de un circuito lógico digital, junto con el papel de los elementos de lenguaje que se utilizan para describir lo que hacen los sistemas. Considere el reto de describir lo que se le hace a un auto de carreras durante una parada en los pits. Si una sola persona realizara todas las tareas necesarias, una por una, tendría que ser muy rápida. Ésta es la forma en la que opera una computadora: una tarea a la vez, pero con mucha rapidez. Desde luego que en las competencias de autos hay todo un equipo de mecánicos en los pits que invade el auto, y cada miembro realiza su tarea mientras los demás realizan la suya. Todos los miembros del equipo operan en forma concurrente, al igual que los elementos de un circuito digital. Ahora considere la forma en que usted describiría a alguien más lo que se está haciendo al auto de carreras durante la parada en los pits, utilizando (1) el enfoque del mecánico individual o (2) el enfoque del equipo de mecánicos. ¿No serían muy similares las dos descripciones verbales de lo que se está haciendo? Como veremos, los lenguajes que se utilizan para describir el hardware digital (HDL) son muy similares a los lenguajes que describen los programas de computadora (por ejemplo, BASIC, C, JAVA), aún y cuando la implementación resultante opera en formas muy distintas. No es necesario conocer cualquiera de estos lenguajes de programación para comprender el HDL. Lo importante es que cuando usted haya aprendido a usar tanto el HDL como un lenguaje de computadora, deberá comprender el papel particular de cada uno en los sistemas digitales.

EJEMPLO 3-25

Compare la operación de una computadora y un circuito lógico al realizar la operación lógica simple de $y = AB$.

Solución

El circuito lógico es una simple compuerta AND. La salida y estará en ALTO dentro de un lapso aproximado de 10 nanosegundos después del punto en el que A y B estén en ALTO al mismo tiempo. La salida y estará en BAJO dentro de un lapso aproximado de 10 nanosegundos después de que cualquiera de las entradas cambie a BAJO.

La computadora debe ejecutar un programa de instrucciones que realice decisiones. Suponga que cada instrucción toma 20 ns (bastante rápido). Cada figura del diagrama de flujo que se muestra en la figura 3-43 representa una instrucción. Es evidente que se requerirá un mínimo de dos o tres instrucciones (40-60 ns) para responder a los cambios en las entradas.

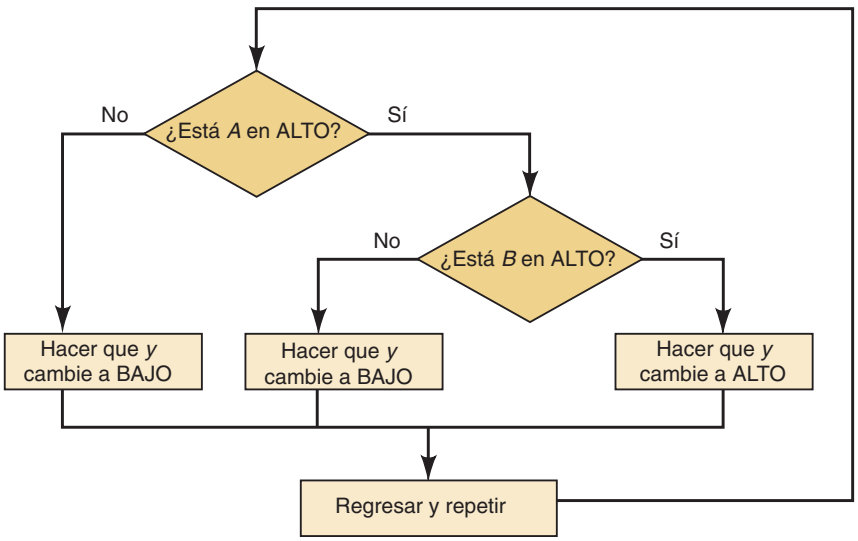


FIGURA 3-43 El proceso de decisiones de un programa de computadora.

PREGUNTAS DE REPASO

1. ¿Qué significa HDL?
2. ¿Cuál es el propósito de un HDL?
3. ¿Cuál es el propósito de un lenguaje de programación de computadoras?
4. ¿Cuál es la diferencia clave entre el HDL y los lenguajes de programación de computadoras?

3-18 IMPLEMENTACIÓN DE CIRCUITOS LÓGICOS CON PLDS

En la actualidad, muchos circuitos digitales se implementan mediante el uso de dispositivos lógicos programables (PLDs). Estos dispositivos no son como las microcomputadoras o los microcontroladores, los cuales "ejecutan" un programa de instrucciones. En vez de ello se configuran en forma electrónica y sus circuitos internos están "alambrados" entre sí en forma electrónica para formar un circuito lógico. Este alambrado programable puede considerarse como miles de conexiones que están conectadas (1) o no conectadas (0). La figura 3-44 muestra una pequeña área de conexiones programables. Cada cruce de una fila (alambre horizontal) y una columna (alambre vertical) es una conexión programable. Es fácil imaginar lo difícil

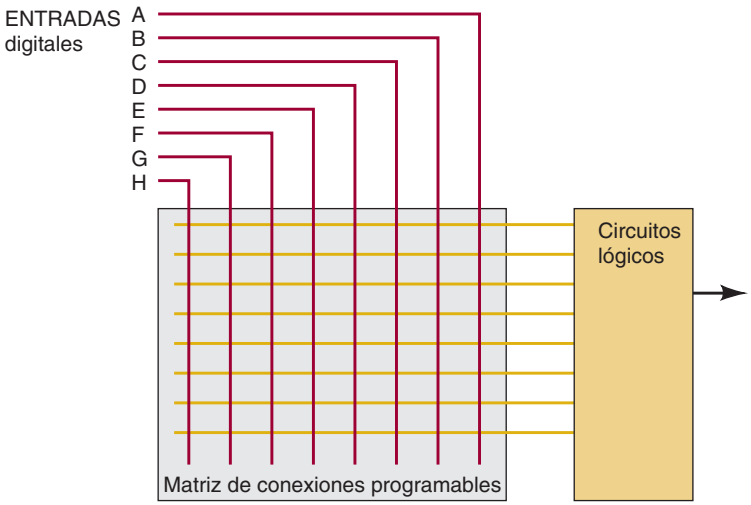


FIGURA 3-44
Configuración de las conexiones de hardware con los dispositivos lógicos programables.

que sería tratar de configurar estos dispositivos mediante la colocación de 1s y 0s en una matriz en forma manual (que es como se hacía en la década de 1970).

El papel del lenguaje de descripción de hardware es proporcionar una manera concisa y conveniente para que el diseñador describa la operación del circuito en un formato que una computadora personal pueda manejar y almacenar adecuadamente. La computadora ejecuta una aplicación de software especial llamada **compilador** para traducir el lenguaje de descripción de hardware en la matriz de 1s y 0s que pueden cargarse en el PLD. Si una persona puede dominar el lenguaje de descripción de hardware de mayor nivel, le será más fácil programar los PLDs que tratar de usar álgebra booleana, dibujos esquemáticos o tablas de verdad. En forma muy parecida a como aprendemos un idioma, empezaremos por expresar cosas simples y poco a poco iremos aprendiendo los aspectos más complicados de estos lenguajes. Nuestro objetivo es que aprenda lo suficiente sobre HDL como para que se comunique con los demás y realice tareas simples. La total comprensión de todos los detalles sobre estos lenguajes es algo que está fuera del alcance de este libro, ya que sólo puede dominarse mediante la práctica regular.

En las secciones de este libro que tratan acerca de los HDLs presentaremos tanto a AHDL como a VHDL en un formato que le permita omitir un lenguaje y concentrarse en el otro sin perderse de información importante. Desde luego que esto implica que habrá cierta información redundante presente si usted opta por leer sobre ambos lenguajes. Sentimos que esta redundancia vale el esfuerzo de proporcionarle la flexibilidad de enfocarse en cualquiera de los dos lenguajes, o de aprender ambos al comparar y contrastar ejemplos similares. La forma recomendada de utilizar el libro es enfocarse en un lenguaje. Es cierto que la manera más sencilla de volverse bilingüe y eficiente en ambos lenguajes es crecer en un entorno en el que ambos lenguajes se hablen de manera rutinaria. No obstante, también es muy fácil confundir los detalles, por lo que mantendremos los ejemplos específicos separados e independientes. Esperamos que este formato le brinde la oportunidad de aprender un lenguaje ahora y, más adelante, utilizar este libro como referencia, en caso de que necesite aprender el segundo lenguaje.

PREGUNTAS DE REPASO

1. ¿Qué significa PLD?
2. ¿Cómo se reconfiguran los circuitos en forma electrónica en un PLD?
3. ¿Qué hace un compilador?

3-19 FORMATO Y SINTAXIS DEL HDL

Todos los lenguajes tienen sus propiedades particulares, sus similitudes con otros lenguajes y su propia sintaxis. Cuando estudiamos gramática en la escuela, aprendemos convenciones tales como el orden de las palabras como elementos en una oración y la puntuación apropiada. A esto se le conoce como la **sintaxis** del lenguaje. Un lenguaje diseñado para que la computadora lo interprete debe seguir reglas estrictas de sintaxis. Una computadora es tan sólo un conjunto de alambre y arena de playa procesada (silicio) que no tiene idea de lo que uno "quiere" darle a entender, por lo que debemos presentar las instrucciones usando la sintaxis exacta que el lenguaje de computadora espera y comprende. El formato básico de la descripción de cualquier circuito de hardware, en cualquier lenguaje, implica dos elementos vitales:

1. La definición de lo que entra al circuito y lo que sale de éste, es decir las especificaciones de entradas/salidas.
2. La definición de la forma en que las salidas responden a las entradas, es decir su operación.

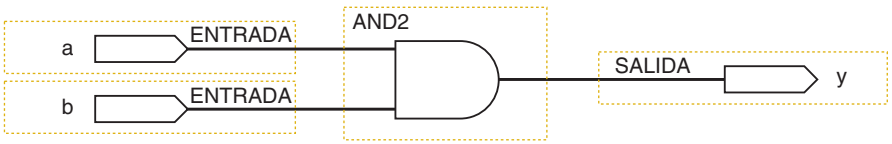


FIGURA 3-45 Descripción de un diagrama esquemático.

Un ingeniero o técnico competente podría leer un diagrama esquemático como el de la figura 3-45, ya que ambos comprenderían el significado de cada uno de los símbolos en el dibujo. Si usted comprende cómo funciona cada elemento y cómo se conectan los elementos entre sí, podrá comprender la forma en que opera el circuito. En el lado izquierdo del diagrama está el conjunto de entradas y a la derecha está el conjunto de salidas. Los símbolos del centro definen su operación. El lenguaje basado en texto debe transmitir la misma información. Todos los HDLs utilizan el formato que se muestra en la figura 3-46.

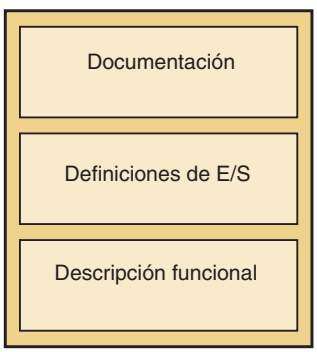


FIGURA 3-46 Formato de los archivos de HDL.

En un lenguaje basado en texto, el circuito que se describe debe tener un nombre. Se deben asignar nombres a las entradas y las salidas (algunas veces llamadas puertos), y se deben definir de acuerdo con la naturaleza del puerto. ¿Es un bit individual de un interruptor? ¿O es un número de cuatro bits que proviene de un teclado numérico? El lenguaje basado en texto debe transmitir de alguna forma la naturaleza de estas entradas y salidas. El **modo** de un puerto define si es de entrada, de salida o de entrada/salida. El **tipo** se refiere al número de bits y la manera en que éstos se agrupan y se interpretan. Si el *tipo* de entrada es de un solo bit, entonces sólo puede tener dos valores posibles: 0 y 1. Si el tipo de entrada es un número binario de cuatro bits que proviene de un teclado numérico, puede tener cualquiera de 16 valores distintos ($0000_2 - 1111_2$). El tipo determina el intervalo de valores posibles. En un lenguaje basado en texto, la definición de la operación del circuito está

contenida dentro de un conjunto de instrucciones que van después de la definición de entrada/salida (E/S) del circuito. En las siguientes dos secciones describiremos el circuito simple de la figura 3-45 e ilustraremos los elementos críticos del AHDL y del VHDL.

DESCRIPCIÓN BOOLEANA MEDIANTE EL USO DE AHDL

Consulte la figura 3-47. La palabra clave **SUBDESIGN** asigna un nombre al bloque del circuito, que en este caso es *compuerta_and*. El nombre del archivo también debe ser compuerta_and.tdf. Observe que la palabra clave SUBDESIGN está en mayúsculas. El software no lo requiere, pero el uso de un estilo consistente en cuanto a las mayúsculas y minúsculas facilita en forma considerable la lectura del código. La guía de estilo que se incluye con el compilador de Altera para AHDL sugiere el uso de letras mayúsculas para las palabras clave del lenguaje. Las variables que nombre el diseñador deben estar en minúsculas.

FIGURA 3-47 Elementos esenciales en AHDL.

```
SUBDESIGN compuerta and
(
   a, b      :INPUT;
   y         :OUTPUT;
)
BEGIN
    y = a & b;
END;
```

La sección SUBDESIGN define las entradas y salidas del bloque del circuito lógico. Algo debe encerrar al circuito que estamos tratando de describir, al igual que un diagrama de bloques encierra a todo lo que forma parte del diseño. En AHDL, esta definición de entrada/salida se encierra entre paréntesis. La lista de variables utilizadas como entradas para este bloque se separa mediante comas y va seguida de :INPUT;. En AHDL se asume el uso del tipo bit individual, a menos que la variable se designe como varios bits. El bit de salida individual se declara con el modo :OUTPUT;. Aprenderemos la forma correcta de describir otros tipos de entradas, salidas y variables a medida que lo vayamos necesitando.

El conjunto de instrucciones que describe la operación del circuito en AHDL está contenido en la sección lógica, entre las palabras clave BEGIN y END. En este ejemplo, la operación de hardware se describe mediante una ecuación de álgebra booleana muy simple, la cual establece que a la salida (y) se le debe asignar (=) el nivel lógico producido por a AND b. A esta ecuación de álgebra booleana se le conoce como **instrucción de asignación concurrente**. Cualquier instrucción (sólo hay una en este ejemplo) entre BEGIN y END se evaluará en forma constante y concurrente. El orden en el que se listen no tiene nada que ver. Los operadores booleanos básicos son:

&	AND
#	OR
!	NOT
$	XOR

PREGUNTAS DE REPASO

1. ¿Qué aparece dentro de los paréntesis ( ) después de SUBDESIGN?
2. ¿Qué aparece entre BEGIN y END?

DESCRIPCIÓN BOOLEANA MEDIANTE EL USO DE VHDL

Consulte la figura 3-48. La palabra clave **ENTITY** asigna un nombre al bloque del circuito, que en este caso es compuerta_and. Observe que la palabra ENTITY está en mayúsculas pero compuerta_and no. El software no requiere esto, pero el uso de un estilo consistente en cuanto a las mayúsculas y minúsculas facilita en forma considerable la lectura del código. La guía de estilo que se incluye con el compilador de Altera para VHDL sugiere el uso de letras mayúsculas para las palabras clave del lenguaje. Las variables que nombra el diseñador deben estar en minúsculas.

FIGURA 3-48 Elementos esenciales en VHDL.

```
ENTITY compuerta and IS
PORT (    a, b    :IN BIT;
          y       :OUT BIT);
END compuerta and;
ARCHITECTURE ckt OF compuerta and IS
BEGIN
        y <= a AND b;
END ckt;
```

La declaración ENTITY puede considerarse como la descripción de un bloque. Algo debe encerrar al circuito que estamos tratando de describir, al igual que un diagrama de bloques encierra a todo lo que forma parte del diseño. En VHDL, la palabra clave PORT indica al compilador que estamos definiendo entradas y salidas para el bloque de este circuito. Se listan los nombres utilizados para las entradas (separados por comas) y se termina con un punto y coma, junto con una descripción del modo y del tipo de entrada (:IN BIT). En VHDL, la descripción **BIT** indica al compilador que cada una de las variables en la lista es un bit individual. Aprenderemos la forma apropiada de describir otros tipos de entradas, salidas y variables a medida que lo vayamos necesitando. La línea que contiene END compuerta_and; termina la declaración ENTITY.

La declaración **ARCHITECTURE** se utiliza para describir la operación de todo lo que se encuentra dentro del bloque. El diseñador inventa un nombre para esta descripción arquitectónica del funcionamiento interno del bloque ENTITY (*ckt* en este ejemplo). Todo bloque ENTITY debe tener cuando menos una declaración ARCHITECTURE asociada a él. Las palabras OF e IS son palabras clave en esta declaración. El cuerpo de la descripción de la arquitectura va encerrado entre las palabras clave BEGIN y END. END va seguida del nombre que se asignó a esta arquitectura. Dentro del cuerpo (entre BEGIN y END) está la descripción de la operación del bloque. En este ejemplo, la operación del hardware se describe mediante una ecuación de álgebra booleana muy simple, la cual establece que a la salida (y) se le debe asignar (<=) el nivel lógico producido por a AND b. A esto se le conoce como una **instrucción de asignación concurrente**, lo cual significa que todas las instrucciones (sólo hay una en este ejemplo) entre BEGIN y END se evaluarán en forma constante y concurrente. El orden en el que se listen no es relevante.

PREGUNTAS DE REPASO

1. ¿Cuál es la función de la declaración ENTITY?
2. ¿Qué sección clave define la operación del circuito?
3. ¿Cuál es el operador de asignación que se utiliza para asignar un valor a una señal lógica?

3-20 SEÑALES INTERMEDIAS

En muchos diseños se tiene la necesidad de definir puntos de señal "dentro" del bloque del circuito. Son puntos en el circuito que no son entradas ni salidas para el bloque, pero que pueden ser útiles como punto de referencia. Puede ser una señal que necesite conectarse a muchos otros lugares dentro del bloque. En un diagrama esquemático analógico o digital se llamarían puntos de prueba o *nodos*. En un HDL se conocen como **nodos ocultos** o **señales locales**. La figura 3-49 muestra un circuito muy simple que utiliza una señal intermedia llamada *m*. En el HDL estos nodos (señales) no se definen con las entradas y salidas, sino en la sección que describe la operación del bloque. Las entradas y salidas están disponibles para otros bloques de circuito en el sistema, pero estas señales locales se reconocen sólo dentro de este bloque.

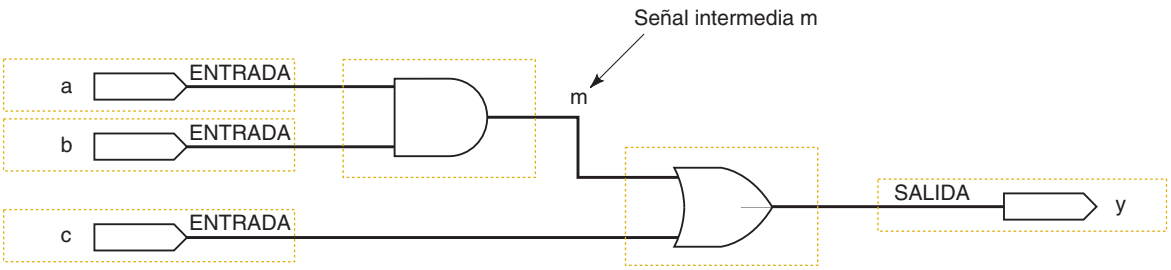


FIGURA 3-49
Diagrama de un circuito lógico con una variable intermedia.

En el código de ejemplo que viene a continuación, observe la información en la parte superior. El propósito de esta información es sólo con fines de documentación. En definitiva es imprescindible que el diseño se documente con todos los detalles posibles. Como mínimo, debe describir el proyecto en el que se está usando, quién lo escribió y la fecha. Por lo general, a esta información se le conoce como el encabezado. En este libro vamos a mantener nuestros encabezados breves para que sea más ligero cuando lo lleve a su clase, pero recuerde que el espacio en memoria casi no cuesta y la información es valiosa, así que *no tenga miedo de documentar sus proyectos con todo el detalle posible*. También hay comentarios enseguida de instrucciones en el código. Estos comentarios sirven al diseñador para recordar lo que estaba tratando de hacer, además ayudan a que cualquier otra persona comprenda el significado del código.

NODOS OCULTOS EN AHDL

El Código AHDL que describe el circuito de la figura 3-49 se muestra en la figura 3-50. Los **comentarios** en AHDL pueden ir encerrados entre caracteres %, como podemos ver entre las líneas 1 y 4. Esta sección del código permite al diseñador escribir muchas líneas de información que serán ignoradas por los programas de computadora que utilicen este archivo, pero para cualquier persona que trate de descifrar el código serán muy útiles. Observe que los comentarios al final de las líneas 9, 10, 13, 15 y 16 van precedidos por dos guiones cortos (--). El texto después de los guiones cortos es sólo para documentación. Puede usarse cualquiera de estos símbolos para comentarios, pero los signos de porcentaje deben usarse en pares para abrir y cerrar un comentario. Los dos guiones cortos indican un comentario que se extiende hasta el final de la línea.

En AHDL las señales locales se declaran en la sección VARIABLE, la cual se coloca entre la sección SUBDESIGN y la sección lógica. La señal intermedia *m* se define en la línea 11, después de la palabra clave **VARIABLE**. La palabra clave **NODE** designa la naturaleza de la variable. Observe que un signo de dos puntos

FIGURA 3-50 Las variables intermedias en AHDL, descritas en la figura 3-49.

```
1   % Variables intermedias en AHDL (Figura 3 49)
2     Sistemas digitales 10a ed
3     NS Widmer
4     MAYO 23, 2005       %
5   SUBDESIGN fig3 50
6   (
7     a,b,c      :INPUT;      define las entradas del bloque
8     y          :OUTPUT;     define la salida del bloque
9   )
10  VARIABLE
11    m        :NODE;         nombra una señal intermedia
12  BEGIN
13    m = a & b;              genera término de producto oculto
14    y = m # c;              genera suma en la salida
15  END;
```

separa el nombre de la variable de su designación de nodo. En la descripción de hardware de la línea 13, la variable intermedia se asigna (se conecta) a un valor (*m = a & b;*) y después *m* se utiliza en la segunda instrucción en la línea 14 para asignar (conectar) un valor a *y* (*y = m # c;*). Recuerde que las instrucciones de asignación son concurrentes y por ende no importa el orden en el que se proporcionan. Para que los humanos podamos leerlas, es más lógico asignar valores a variables intermedias antes de usarlas en otras instrucciones de asignación, tal y como se muestra en este ejemplo.

PREGUNTAS DE REPASO

1. ¿Cuál es la designación que se utiliza para las variables intermedias?
2. ¿En dónde se declaran estas variables?
3. ¿Importa si la ecuación *m* o *y* se declara primero?
4. ¿Qué carácter se utiliza para limitar un bloque de comentarios?
5. ¿Qué caracteres se utilizan para comentar una sola línea?

VHDL

SEÑALES LOCALES DE VHDL

El código de VHDL que describe el circuito de la figura 3-49 se muestra en la figura 3-51. Los **comentarios** en VHDL van después de dos guiones cortos (--). Al escribir dos guiones cortos sucesivos, el diseñador puede escribir información desde ese punto hasta el final de la línea. La información que vaya después de estos dos guiones será ignorada por los programas de computadora que utilicen este archivo, pero cualquier persona que trate de descifrar el código podrá leerla.

La señal intermedia *m* se define en la línea 13, después de la palabra clave SIGNAL. La palabra clave BIT designa el tipo de la señal. Observe que un signo de dos puntos separa el nombre de la señal de su designación de tipo. En la descripción de hardware de la línea 16, la señal intermedia se asigna (se conecta) a un valor (*m <= a* AND *b;*) y después *m* se utiliza en la instrucción de la línea 17 para asignar (conectar) un valor a *y* (*y <= m* OR *c;*). Recuerde que las instrucciones de asignación son

```
1     Variables intermedias en VHDL (Figura 3 49)
2     Sistemas digitales 10a ed
3     NS Widmer
4     MAYO 23, 2005
5
6   ENTITY  fig3 51 IS
7   PORT( a, b, c   :IN BIT;       define las entradas del bloque
8   y               :OUT BIT);     define la salida del bloque
9   END fig3 51;
10
11  ARCHITECTURE ckt OF fig3 51 IS
12
13     SIGNAL m     :BIT;          nombra una señal intermedia
14
15  BEGIN
16     m <= a AND b;             genera término de producto oculto
17     y <= m OR c;              genera suma en la salida
18  END ckt;
```

FIGURA 3-51 Señales intermedias en VHDL, descritas en la figura 3-49.

concurrentes y no importa el orden en el que se proporcionen. Para que los humanos podamos leerlas, es más lógico asignar valores a las señales intermedias antes de utilizarlas en otras instrucciones de asignación, como se muestra aquí.

PREGUNTAS DE REPASO

1. ¿Cuál es la designación utilizada para las señales intermedias?
2. ¿En dónde se declaran estas señales?
3. ¿Importa si la ecuación m o y se declara primero?
4. ¿Qué caracteres se utilizan para comentar una sola línea?

RESUMEN

1. El álgebra booleana es una herramienta matemática que se utiliza en el análisis y diseño de circuitos digitales.
2. Las operaciones booleanas básicas son OR, AND y NOT.
3. Una compuerta OR produce una salida en ALTO cuando cualquier entrada está en ALTO. Una compuerta AND produce una salida en ALTO sólo cuando todas las entradas están en ALTO. Un circuito NOT (INVERSOR) produce una salida que es el nivel lógico opuesto al de la entrada.
4. Una compuerta NOR es igual que una compuerta OR con su salida conectada a un INVERSOR. Una compuerta NAND es igual que una compuerta AND con su salida conectada a un INVERSOR.

5. Los teoremas y las reglas booleanas pueden usarse para simplificar la expresión de un circuito lógico y pueden producir una manera más simple de implementar el circuito.
6. Las compuertas NAND pueden usarse para implementar cualquiera de las operaciones booleanas básicas. Las compuertas NOR pueden usarse en forma similar.
7. Pueden usarse los símbolos alternativos o estándar para cada compuerta lógica, dependiendo de si la salida va a ser activa en ALTO o activa en BAJO.
8. El estándar IEEE/ANSI para los símbolos lógicos utiliza símbolos rectangulares para cada dispositivo lógico, con notaciones especiales dentro de los rectángulos para mostrar cómo dependen las salidas de las entradas.
9. Los lenguajes de descripción de hardware se han convertido en un método importante para describir circuitos digitales.
10. El código HDL siempre debe contener comentarios que documenten sus características más importantes, de manera que cualquier persona que lo lea después pueda comprender lo que hace.
11. Toda descripción de un circuito en HDL contiene una definición de las entradas y salidas, seguida de una sección que describe la operación del circuito.
12. Además de las entradas y salidas, pueden definirse conexiones intermedias inaccesibles desde fuera del circuito. A estas conexiones intermedias se les conoce como nodos ocultos o señales locales.

TÉRMINOS IMPORTANTES

activa en ALTO
activa en BAJO
AHDL –lenguaje de descripción de hardware de Altera
álgebra booleana
ARCHITECTURE
asignada
BIT
circuito NOT (INVERSOR)
compilador
compuerta AND
compuerta NAND
compuerta NOR
compuerta OR
concurrente
ENTITY
HDLs –lenguajes de descripción de hardware
IEEE/ANSI
instrucción de asignación concurrente
inversión (complementación)
modo
nivel lógico
niveles lógicos activos
no asignado
NODE
nodos ocultos (señales locales)
operación AND
operación NOT
operación OR
PLDs –dispositivos lógicos programables
símbolos lógicos alternativos
sintaxis
SUBDESIGN
tabla de verdad
teoremas booleanos
teoremas de DeMorgan
tipo
VARIABLE
VHDL –lenguaje de descripción de hardware para circuitos integrados de muy alta velocidad (VHSIC)

PROBLEMAS

Las letras en gris que van antes de algunos de los problemas se utilizan para indicar la naturaleza o el tipo de éstos, como se indica a continuación:

B problema básico.

F problema de diagnóstico de fallas.

D problema de diseño o modificación de circuito.

N nuevo concepto o técnica que no se cubre en el libro.

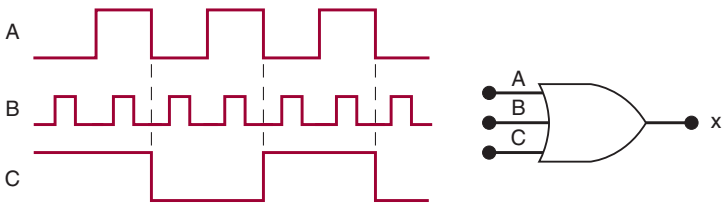
A problema avanzado.

H problema de HDL.

SECCIÓN 3-3

B 3-1.* Dibuje la forma de onda de salida para la compuerta OR de la figura 3-52.

FIGURA 3-52



B 3-2. Suponga que la entrada *A* en la figura 3-52 se conectó a tierra en forma inadvertida (es decir, $A = 0$). Dibuje la forma de onda de salida resultante.

B 3-3.* Suponga que la entrada *A* en la figura 3-52 se pone en corto en forma inadvertida con la línea de suministro de +5 V (es decir, $A = 1$). Dibuje la forma de onda de salida resultante.

C 3-4. Lea las siguientes aseveraciones con respecto a una compuerta OR. Al principio tal vez parezcan válidas, pero después de analizarlas se dará cuenta que ninguna es verdadera *siempre*. Demuestre esto mediante un ejemplo específico para refutar cada aseveración.

(a) Si la forma de onda de salida de una compuerta OR es igual que la forma de onda en una de sus entradas, la otra entrada se mantendrá de manera permanente en BAJO.

(b) Si la forma de onda de salida de una compuerta OR siempre está en ALTO, una de sus entradas se mantendrá de manera permanente en ALTO.

B 3-5. ¿Cuántas combinaciones distintas de condiciones de entrada producirán una salida en ALTO en una compuerta OR de cinco entradas?

SECCIÓN 3-4

B 3-6. Cambie la compuerta OR en la figura 3-52 por una compuerta AND.

(a)* Dibuje la forma de onda de salida.

(b) Dibuje la forma de onda de salida si la entrada *A* se conecta de manera permanente a tierra.

(c) Dibuje la forma de onda de salida si *A* se pone en corto de manera permanente con +5 V.

D 3-7.* Consulte la figura 3-4. Modifique el circuito de manera que la alarma se active sólo cuando la presión y la temperatura excedan sus límites máximos al mismo tiempo.

B 3-8.* Cambie la compuerta OR en la figura 3-6 por una compuerta AND y dibuje la forma de onda de la salida.

B 3-9. Suponga que tiene una compuerta desconocida de dos entradas, que puede ser OR o AND. ¿Qué combinación de niveles de entrada debe aplicar a las entradas para determinar qué tipo de compuerta es?

B 3-10. *Verdadero* o *falso:* sin importar cuántas entradas tenga, una compuerta AND producirá una salida en ALTO para sólo una combinación de niveles de entrada.

* Encontrará las respuestas a los problemas marcados con un asterisco al final del libro.

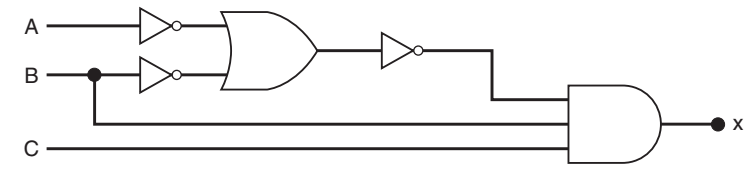
SECCIONES 3-5 A 3-7

B 3-11. Aplique la forma de onda *A* de la figura 3-23 a la entrada de un INVERSOR. Dibuje la forma de onda de salida. Repita el proceso para la forma de onda *B*.

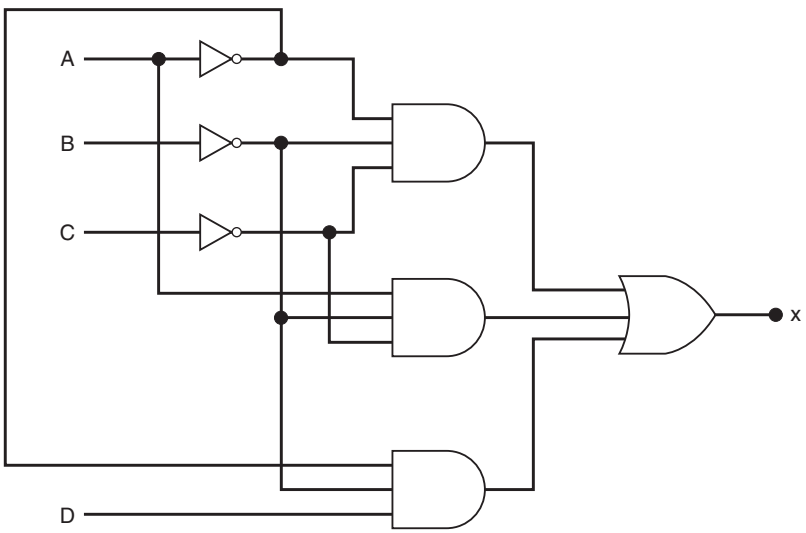
B 3-12. (a)* Escriba la expresión booleana para la salida *x* en la figura 3-53(a). Determine el valor de *x* para todas las posibles condiciones de entrada y liste los valores en una tabla de verdad.

(b) Repita el proceso para el circuito de la figura 3-53(b).

FIGURA 3-53



(a)



(b)

B 3-13.* Cree una tabla de análisis completa para el circuito de la figura 3-15(b); para ello encuentre los niveles lógicos presentes en la salida de cada compuerta, para todas las 32 posibles combinaciones de entrada.

B 3-14. (a)* Cambie cada compuerta OR por una compuerta AND y cada compuerta AND por una compuerta OR en la figura 3-15(b). Después escriba la expresión para la salida.

(b) Complete una tabla de análisis.

B 3-15. Cree una tabla de análisis completa para el circuito de la figura 3-16; para ello encuentre los niveles lógicos presentes en la salida de cada compuerta, para todas las 16 posibles combinaciones de niveles de entrada.

SECCIÓN 3-8

B 3-16. Para cada una de las siguientes expresiones, construya el circuito lógico correspondiente utilizando compuertas AND y OR e INVERSORES.

(a)* $x = \overline{AB(C + D)}$

(b)* $z = \overline{A + B + \overline{C}\overline{D}\overline{E}} + \overline{B}\overline{C}\overline{D}$

(c) $y = (\overline{\overline{M + N} + \overline{P}Q})$

(d) $x = \overline{W + P\overline{Q}}$

(e) $z = MN(P + \overline{N})$

(f) $x = (A + B)(\overline{A} + \overline{B})$

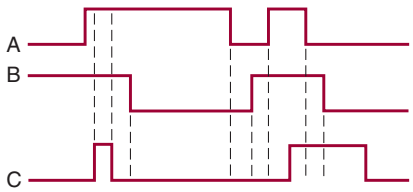
SECCIÓN 3-9

B 3-17.* (a) Aplique las formas de onda de entrada de la figura 3-54 a una compuerta NOR y dibuje la forma de onda de salida.

(b) Repita el procedimiento manteniendo *C* de manera permanente en BAJO.

(c) Repita el procedimiento manteniendo *C* en ALTO.

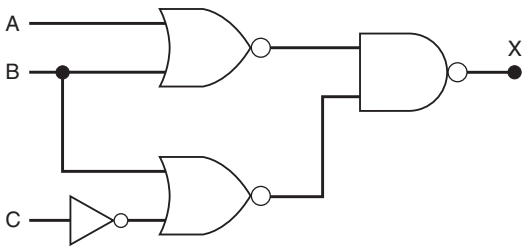
FIGURA 3-54



B 3-18. Repita el problema 3-17 para una compuerta NAND.

C 3.19.* Escriba la expresión para la salida de la figura 3-55 y utilícela para determinar la tabla de verdad completa. Después aplique las formas de onda de la figura 3-54 a las entradas del circuito y dibuje la forma de onda de salida resultante.

FIGURA 3-55



B 3-20. Determine la tabla de verdad para el circuito de la figura 3-24.

B 3-21. Modifique los circuitos que se construyeron en el problema 3-16 de manera que se utilicen compuertas NAND y NOR en donde sea apropiado.

SECCIÓN 3-10

C 3-22. Demuestre los teoremas (15a) y (15b) probando todos los casos posibles.

B 3-23.* PREGUNTA DE EJERCICIO

Complete cada una de las expresiones.

(a) $A + 1 =$ _____

(b) $A \cdot A =$ _____

(c) $B \cdot \overline{B} =$ _____

(d) $C + C =$ _____

(e) $x \cdot 0 =$ _____

(f) $D \cdot 1$ _____

(g) $D + 0 =$ _____

(h) $C + \overline{C} =$ _____

(i) $G + GF =$ _____

(j) $y + \overline{w}y =$ _____

C 3-24. (a)* Simplifique la siguiente expresión usando los teoremas (13b), (3) y (4):

$$x = (M + N)(\overline{M} + P)(\overline{N} + \overline{P})$$

(b) Simplifique la siguiente expresión utilizando los teoremas (13a), (8) y (6):

$$z = \overline{A}B\overline{C} + AB\overline{C} + B\overline{C}D$$

SECCIONES 3-11 Y 3-12

C 3-25. Demuestre los teoremas de DeMorgan probando todos los casos posibles.

B 3-26. Simplifique cada una de las siguientes expresiones usando los teoremas de DeMorgan.

(a)* $\overline{\overline{A}B\overline{C}}$ (d) $\overline{A + \overline{B}}$ (g)* $\overline{A(\overline{B + C})D}$

(b) $\overline{\overline{A} + \overline{B}C}$ (e)* $\overline{\overline{A}\overline{B}}$ (h) $\overline{(M + \overline{N})(\overline{M} + N)}$

(c)* $\overline{AB\overline{C}\overline{D}}$ (f) $\overline{\overline{A} + \overline{C} + \overline{D}}$ (i) $\overline{\overline{A}\overline{B}CD}$

B 3-27.* Use los teoremas de DeMorgan para simplificar la expresión de salida de la figura 3-55.

C 3-28. Convierta el circuito de la figura 3-53(b) en uno que utilice sólo compuertas NAND. Después escriba la expresión de salida para el nuevo circuito, simplifíquelo utilizando los teoremas de DeMorgan y compárelo con la expresión para el circuito original.

C 3-29. Convierta el circuito de la figura 3-53(a) en uno que utilice sólo compuertas NOR. Después escriba la expresión para el nuevo circuito, simplifíquelo utilizando los teoremas de DeMorgan y compárelo con la expresión para el circuito original.

B 3-30. Muestre cómo puede construirse una compuerta NAND de dos entradas a partir de dos compuertas NOR de dos entradas.

B 3-31. Muestre cómo puede construirse una compuerta NOR de dos entradas a partir de compuertas NAND de dos entradas.

C 3-32. Un jet emplea un sistema para monitorear los valores de revoluciones por minuto (rpm), presión y temperatura de sus motores mediante el uso de motores que operan de la siguiente manera:

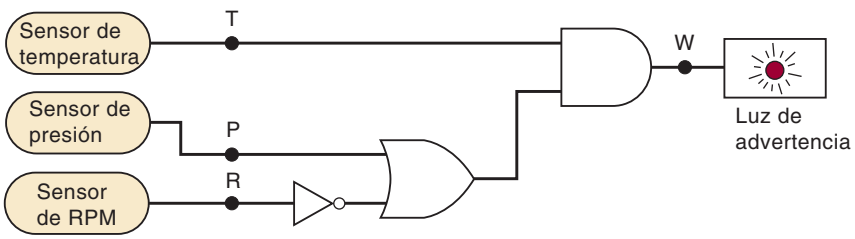
salida del sensor de $RPM = 0$ sólo cuando la velocidad < 4800 rpm
salida del sensor $P = 0$ sólo cuando la presión < 220 psi
salida del sensor $T = 0$ sólo cuando la temperatura $< 200°$ F

La figura 3-56 muestra el circuito lógico que controla una luz de advertencia en cabina para ciertas combinaciones de condiciones del motor. Suponga que un nivel ALTO en la salida W activa la luz de advertencia.

(a)* Determine qué condiciones del motor darán una advertencia al piloto.

(b) Cambie este circuito por uno que utilice sólo compuertas NAND.

FIGURA 3-56



SECCIONES 3-13 Y 3-14

B 3-33. Para cada una de las siguientes instrucciones, dibuje el símbolo de compuerta lógica (estándar o alternativo) apropiado para la operación dada.

(a) Una salida en ALTO ocurre sólo cuando las tres entradas están en BAJO.

(b) Una salida en BAJO ocurre cuando alguna de las cuatro entradas está en BAJO.

(c) Una salida activa en BAJO ocurre sólo cuando las ocho entradas están en ALTO.

B 3-34. Dibuje las representaciones estándar para cada una de las compuertas lógicas básicas. Después dibuje las representaciones alternativas.

C 3-35. Suponga que el circuito de la figura 3-55 es un candado de combinación digital simple, cuya salida generará una señal $\overline{\text{ABRIR}}$ activa en BAJO para sólo una combinación de entradas.

(a)* Modifique el diagrama del circuito de tal forma que represente la operación del circuito de una manera más efectiva.

(b) Use el nuevo diagrama del circuito para determinar la combinación de entradas que activará la salida. Para ello, trabaje regresando desde la salida y utilizando la información que proporcionan los símbolos de las compuertas, como se hizo en los ejemplos 3-22 y 3-23. Compare los resultados con la tabla de verdad que se obtuvo en el problema 3-19.

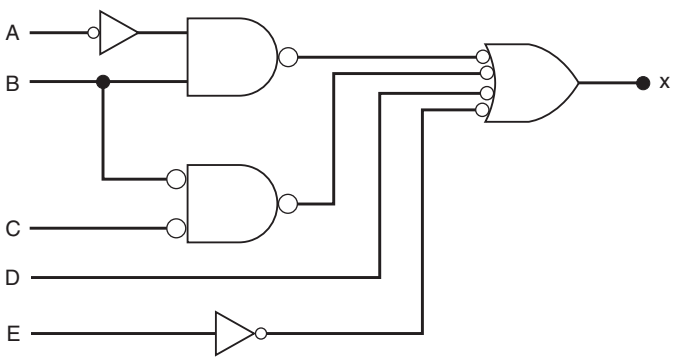
C 3-36. (a) Determine las condiciones de entrada necesarias para activar la salida Z en la figura 3-37(b). Para ello, trabaje regresando desde la salida, como se hizo en los ejemplos 3-22 y 3-23.

(b) Suponga que el estado BAJO de Z es el que debe activar la alarma. Cambie el diagrama del circuito para reflejar lo anterior y después utilice el diagrama revisado para determinar las condiciones de entrada necesarias para activar la alarma.

D 3-37. Modifique el circuito de la figura 3-40, de manera que se necesite que $A_1 = 0$ para producir *UNIDAD* = 1, en lugar de $A_1 = 1$.

B 3-38.* Determine las condiciones de entrada necesarias para hacer que la salida en la figura 3-57 cambie a su estado activo.

FIGURA 3-57



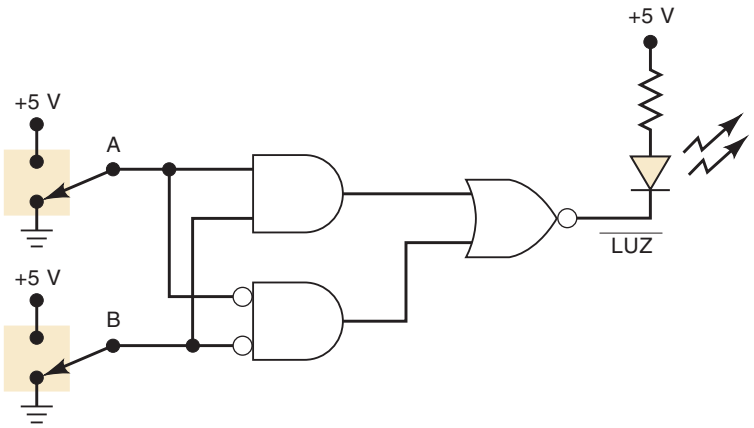
B 3-39.* ¿Cuál es el estado asignado para la salida de la figura 3-57?, ¿para la salida de la figura 3-36(c)?

B 3-40. Use los resultados del problema 3-38 para obtener la tabla de verdad completa para el circuito de la figura 3-57.

N 3-41.* La figura 3-58 muestra una aplicación de compuertas lógicas que simula un interruptor de dos vías, como los que utilizamos en nuestros hogares para encender o apagar una luz desde dos interruptores distintos. Aquí la luz

es un LED que estará ENCENDIDO (en conducción) cuando la salida de la compuerta NOR esté en BAJO. Observe que esta salida está etiquetada como *LUZ* para indicar que es activa en BAJO. Determine las condiciones de entrada necesarias para encender el LED. Después verifique que el circuito opere como un interruptor de dos vías, utilizando los interruptores *A* y *B*. (En el capítulo 4 aprenderá a diseñar circuitos como éste para producir una relación dada entre las entradas y las salidas.)

FIGURA 3-58



SECCIÓN 3-15

B 3-42. Vuelva a dibujar los circuitos de (a)* la figura 3-57 y (b) la figura 3-58 mediante el uso de los símbolos IEEE/ANSI.

SECCIÓN 3-17

PREGUNTAS DE PRÁCTICA DE HDL

H 3.-43.* *Verdadero* o *falso*:

(a) VHDL es un lenguaje de programación de computadoras.

(b) VHDL puede hacer lo mismo que AHDL.

(c) AHDL es un lenguaje del estándar IEEE.

(d) Cada intersección en una matriz de conmutación puede programarse como un circuito abierto o cerrado, entre un alambre de fila con uno de columna.

(e) El primer elemento que aparece en la parte superior de un listado de HDL es la descripción funcional.

(f) El tipo de un objeto indica si es una entrada o una salida.

(g) El modo de un objeto determina si es una entrada o una salida.

(h) Los nodos ocultos son nodos que se han eliminado y que nunca se utilizarán de nuevo.

(i) Las señales locales son otro nombre para las variables intermedias.

(j) El encabezado es un bloque de comentarios que documentan información vital sobre el proyecto.

SECCIÓN 3-18

B 3-44. Vuelva a dibujar la matriz de conexiones programable de la figura 3-44. Etiquete las señales de salida (líneas horizontales) de la matriz de conexión

(desde la fila superior hasta la fila inferior) de la siguiente manera: AAABADHE. Dibuje una X en las intersecciones apropiadas para poner en corto circuito una fila con una columna y crear estas conexiones hacia el circuito lógico.

H 3-45.* Escriba el código de HDL en el lenguaje de su preferencia, para que produzca las siguientes funciones de salida:

$$X = A + B$$
$$Y = AB$$
$$Z = A + B + C$$

H 3-46. Escriba el código de HDL en el lenguaje de su preferencia, para implementar el circuito lógico de la figura 3-39.

(a) Use una sola ecuación booleana.

(b) Use las variables intermedias V, W, X y Y.

APLICACIÓN DE MICROCOMPUTADORA

C 3-47.* Consulte la figura 3-40 en el ejemplo 3-23. Las entradas de A_7 a A_0 son entradas de *dirección* que se suministran a este circuito desde las salidas del microprocesador de una microcomputadora. El código de dirección de ocho bits $A_7 - A_0$ selecciona cuál dispositivo desea activar el microprocesador. En el ejemplo 3-23, el código de dirección requerido para activar la unidad de disco era A_7 a $A_0 = 11111110_2 = FE_{16}$.

Modifique el circuito de manera que el microprocesador deba enviar un código de dirección de $4A_{16}$ para activar la unidad de disco.

EJERCICIOS AVANZADOS

C 3-48. Muestre cómo puede implementarse $x = AB\overline{C}$ con una compuerta NOR de dos entradas y una compuerta NAND de dos entradas.

C 3-49.* Implemente $y = ABCD$ usando sólo compuertas NAND de dos entradas.

RESPUESTAS A LAS PREGUNTAS DE REPASO DE LAS SECCIONES

SECCIÓN 3-2

1. $x = 1$ 2. $x = 0$ 3. 32

SECCIÓN 3-3

1. Las entradas en BAJO. 2. $x = A + B + C + D + E + F$ 3. Nivel ALTO constante.

SECCIÓN 3-4

1. Todas las cinco entradas = 1. 2. Una entrada en BAJO mantendrá la salida en BAJO. 3. Falso; vea la tabla de verdad para cada compuerta.

SECCIÓN 3-5

1. La salida del segundo INVERSOR será igual que la entrada A. 2. y estará en BAJO sólo para $A = B = 1$.

SECCIÓN 3-6

1. $x = \overline{A} + B + C + \overline{AD}$ 2. $x = D(\overline{AB + C}) + E$

SECCIÓN 3-7

1. $x = 1$ 2. $x = 1$ 3. $x = 1$ para ambos casos.

SECCIÓN 3-8

1. Vea la figura 3-15(a). 2. Vea la figura 3-17(b). 3. Vea la figura 3-15(b).

SECCIÓN 3-9

1. Todas las entradas en BAJO. 2. $x = 0$ 3. $x = \overline{A + B + CD}$

SECCIÓN 3-10

1. $y = A\overline{C}$ 2. $y = \overline{A}\,\overline{B}\,\overline{D}$ 3. $y = \overline{A}D + BD$

SECCIÓN 3-11

1. $z = \overline{AB} + C$ 2. $y = (\overline{R} + S + \overline{T})Q$ 3. Igual que la figura 3-28, sólo que la compuerta NAND se sustituye por una compuerta NOR. 4. $y = \overline{A}B(C + \overline{D})$

SECCIÓN 3-12

1. Tres. 2. El circuito NOR es más eficiente, ya que puede implementarse con solo un CI 74LS02. 3. $x = \overline{(\overline{AB})(\overline{CD})} = \overline{\overline{AB}} + \overline{\overline{(CD)}} + AB + CD$

SECCIÓN 3-13

1. La salida cambia a BAJO cuando cualquier entrada está en ALTO. 2. La salida cambia a ALTO sólo cuando todas las entradas están en BAJO. 3. La salida cambia a BAJO cuando cualquier entrada está en BAJO. 4. La salida cambia a ALTO sólo cuando todas las entradas están en ALTO.

SECCIÓN 3-14

1. Z cambiará a ALTO cuando $A = B = 0$ y $C = D = 1$. 2. Z cambiará a BAJO cuando $A = B = 0, E = 1$ y C o D o ambas sean 1. 3. Dos. 4. Dos. 5. BAJO. 6. $A = B = 0, C = D = 1$ 7. $\overline{W}$

SECCIÓN 3-15

1. Vea la figura 3-41. 2. Rectángulo con & en su interior y triángulos en las entradas.

SECCIÓN 3-16

1. Ecuación booleana, tabla de verdad, diagrama lógico, diagrama de tiempos, lenguaje.

SECCIÓN 3-17

1. Lenguaje de descripción de hardware. 2. Para describir un circuito digital y su operación. 3. Para dar a una computadora una lista secuencial de tareas. 4. El HDL describe los circuitos concurrentes de hardware; las instrucciones de computadora se ejecutan una a la vez.

SECCIÓN 3-18

1. Dispositivo lógico programable. 2. Se cierran y se abren conexiones en una matriz de conmutación. 3. Traduce el código de HDL en un patrón de bits para configurar la matriz de conmutación.

SECCIÓN 3-19

AHDL

1. Las definiciones de entrada y de salida. 2. La descripción de la forma en que opera.

VHDL

1. Para dar un nombre al circuito y definir sus entradas y salidas. 2. La descripción ARCHITECTURE. 3. < =

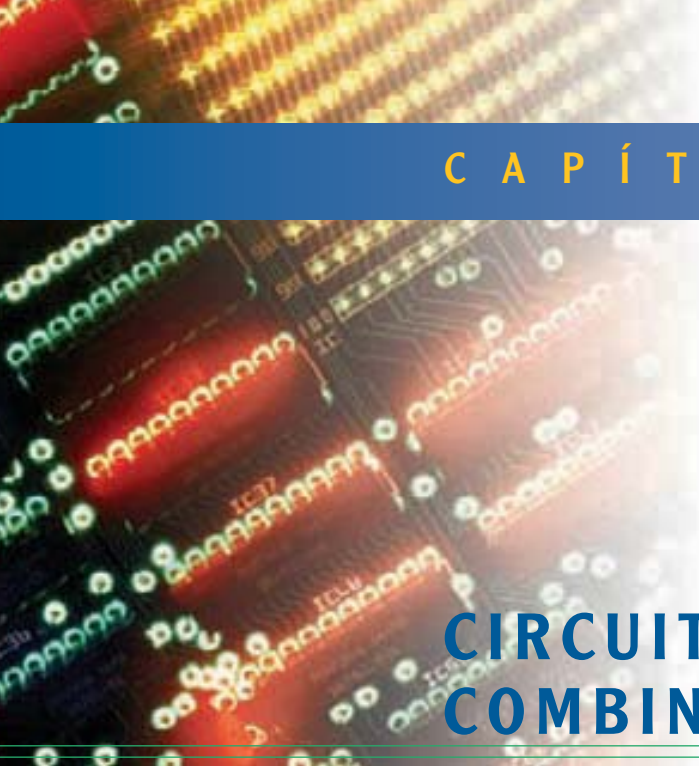
SECCIÓN 3-20

AHDL

1. NODE. 2. Después de la definición de E/S y antes de BEGIN. 3. No. 4. %. 5. --

VHDL

1. SIGNAL. 2. Dentro de ARCHITECTURE, antes de BEGIN. 3. No. 4. --



CAPÍTULO 4

CIRCUITOS LÓGICOS COMBINACIONALES

CONTENIDO

4-1 Forma de suma de productos
4-2 Simplificación de circuitos lógicos
4-3 Simplificación algebraica
4-4 Diseño de circuitos lógicos combinacionales
4-5 Método de mapas de Karnaugh
4-6 Circuitos OR exclusivo y NOR exclusivo
4-7 Generador y comprobador de paridad
4-8 Circuitos de habilitación/deshabilitación
4-9 Características básicas de los CIs digitales
4-10 Diagnóstico de fallas en sistemas digitales
4-11 Fallas internas en los circuitos integrados digitales
4-12 Fallas externas
4-13 Ejemplo práctico de diagnóstico de fallas
4-14 Dispositivos lógicos programables
4-15 Representación de datos en HDL
4-16 Tablas de verdad mediante el uso de HDL
4-17 Estructuras de control de decisiones en HDL

OBJETIVOS

Al terminar este capítulo, usted podrá:

- Convertir una expresión lógica en una expresión de suma de productos.
- Realizar los pasos necesarios para reducir una expresión de suma de productos a su forma más simple.
- Utilizar el álgebra booleana y el mapa de Karnaugh como herramientas para simplificar y diseñar circuitos lógicos.
- Explicar la operación de los circuitos OR exclusivo y NOR exclusivo.
- Diseñar circuitos lógicos simples sin la ayuda de una tabla de verdad.
- Implementar circuitos de habilitación.
- Citar las características básicas de los CIs digitales TTL y CMOS.
- Utilizar las reglas básicas de diagnóstico de fallas de los sistemas digitales.
- Deducir las fallas a partir de los resultados observados en circuitos lógicos combinacionales.
- Describir la idea fundamental de los dispositivos lógicos programables (PLDs).
- Describir los pasos implicados en la programación de un PLD para realizar una función lógica combinacional simple.
- Consultar los manuales de usuario de Altera para adquirir la información necesaria para realizar un experimento de programación simple en el laboratorio.
- Describir los métodos de diseño jerárquico.
- Identificar los tipos de datos apropiados para las variables con valores de un solo bit, arreglos de bits y numéricas.
- Describir los circuitos lógicos mediante el uso de las estructuras de control de HDL IF/ELSE, IF/ELSIF y CASE.
- Seleccionar la estructura de control apropiada para un problema dado.

INTRODUCCIÓN

En el capítulo 3 estudiamos la operación de todas las compuertas lógicas básicas, y utilizamos el álgebra booleana para describir y analizar circuitos formados de combinaciones de compuertas lógicas. Estos circuitos pueden clasificarse como circuitos lógicos *combinacionales* ya que, en cualquier momento, el nivel lógico de la salida depende de la combinación de los niveles lógicos presentes en las entradas. Un circuito combinacional no tiene característica de *memoria*, por lo que su salida depende *sólo* del valor actual de sus entradas.

En este capítulo continuaremos con nuestro estudio de los circuitos combinacionales. Para empezar, veremos más detalles sobre la simplificación de los circuitos lógicos. Analizaremos dos métodos: el primero utiliza los teoremas del álgebra booleana; el segundo utiliza una técnica de *mapeo*. Además,

estudiaremos técnicas simples de diseño de circuitos lógicos combinacionales para satisfacer un conjunto dado de requerimientos. El estudio completo del diseño de circuitos lógicos no es uno de nuestros objetivos, pero los métodos que presentaremos le brindarán una excelente introducción al diseño lógico.

Una buena porción de este capítulo está dedicada al diagnóstico de fallas de los circuitos combinacionales. Con esta primera exposición al diagnóstico de fallas, usted podrá empezar a desarrollar el tipo de habilidades analíticas necesarias para tener éxito al realizar sus diagnósticos de fallas. Para que este material sea lo más práctico posible, primero presentaremos algunas de las características básicas de los CIs de compuertas lógicas en las familias lógicas TTL y CMOS, junto con una descripción de los tipos más comunes de fallas que se encuentran en los circuitos integrados digitales.

En las últimas secciones de este capítulo ampliaremos nuestro conocimiento sobre los dispositivos lógicos programables y los lenguajes de descripción de hardware. Reforzaremos el concepto de las conexiones de hardware programables y brindaremos más detalles en relación con la función del sistema de desarrollo. Usted conocerá los pasos que se siguen actualmente para el diseño y desarrollo de los sistemas digitales. Le proporcionaremos suficiente información para que pueda elegir los tipos de datos correctos para usarlos en proyectos simples que presentaremos más adelante en este capítulo. Por último explicaremos varias estructuras de control, junto con algunas instrucciones relacionadas con su uso apropiado.

4-1 FORMA DE SUMA DE PRODUCTOS

Los métodos de simplificación y diseño de circuitos lógicos que estudiaremos requieren que la expresión lógica se encuentre en forma de **suma de productos (SOP)**. Algunos ejemplos de esta forma son:

1. $ABC + \overline{A}B\overline{C}$
2. $AB + \overline{A}B\overline{C} + \overline{C}\,\overline{D} + D$
3. $\overline{A}B + C\overline{D} + EF + GK + H\overline{L}$

Cada una de estas expresiones de suma de productos consiste de dos o más términos AND (productos) a los que se les aplica la operación OR. Cada término AND consiste de una o más variables que aparecen de manera *individual*, ya sea en forma complementada o no complementada. Por ejemplo, en la expresión de suma de productos $ABC + \overline{A}B\overline{C}$ el primer producto AND contiene las variables A, B y C en su forma no complementada (no invertida). El segundo término AND contiene a A y C en su forma complementada (invertida). En una expresión de suma de productos, un signo de inversión *no puede* cubrir más de una variable en un término (por ejemplo, no podemos tener $\overline{ABC}$ o $\overline{RST}$).

Producto de las sumas

Algunas veces se utiliza otra forma general para las expresiones lógicas en el diseño de circuitos lógicos. A esta forma se le llama **producto de las sumas (POS)** y consiste de dos o más términos OR (sumas) a los que se les aplica una operación AND. Cada término OR consiste de una o más variables en su forma complementada o no complementada. He aquí algunas expresiones de producto de sumas:

1. $(A + \overline{B} + C)(A + C)$
2. $(A + \overline{B})(\overline{C} + D)F$
3. $(A + C)(B + \overline{D})(\overline{B} + C)(A + \overline{D} + \overline{E})$

Los métodos de simplificación y diseño de circuitos que utilizaremos se basan en la forma de suma de productos (SOP), por lo que no veremos muchos ejemplos

con la forma de producto de sumas (POS). No obstante, de vez en cuando se presentará esta forma para algunos circuitos lógicos que tengan una cierta estructura.

PREGUNTAS DE REPASO

1. ¿Cuál de las siguientes expresiones se encuentra en la forma SOP?
   (a) $AB + CD + E$
   (b) $AB(C + D)$
   (c) $(A + B)(C + D + F)$
   (d) $\overline{MN} + PQ$
2. Repita la pregunta 1 para la forma POS.

4-2 SIMPLIFICACIÓN DE CIRCUITOS LÓGICOS

Una vez que se obtiene la expresión para un circuito lógico, podemos reducirla a una forma más simple que contenga menos términos, o menos variables en uno o más términos. Así, la nueva expresión puede utilizarse para implementar un circuito equivalente al circuito original, pero que contenga menos compuertas y conexiones.

Para ilustrar esto, el circuito de la figura 4-1(a) puede simplificarse para producir el circuito de la figura 4-1(b). Ambos circuitos realizan la misma lógica, por lo que debe ser obvio que el más simple es más conveniente, ya que contiene menos compuertas y, por lo tanto, será más pequeño y económico que el original. Lo que es más, la confiabilidad del circuito aumentará, ya que hay menos interconexiones que pueden provocar fallas potenciales en el circuito.

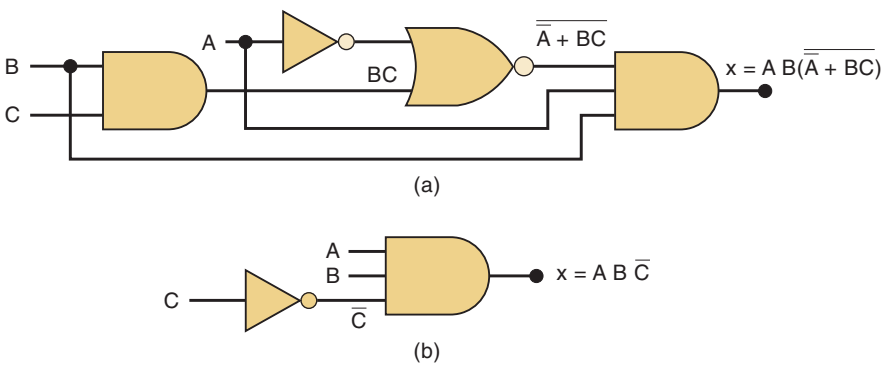


FIGURA 4-1 A menudo es posible simplificar un circuito lógico de tal forma que en la parte (a) se produzca una implementación más eficiente, la cual se muestra en (b).



En las siguientes secciones estudiaremos dos métodos para simplificar los circuitos lógicos. Uno de ellos utiliza los teoremas de álgebra booleana y, como veremos, depende mucho de la inspiración y la experiencia. El otro método (mapeo de Karnaugh) tiene un enfoque sistemático, paso a paso. Tal vez algunos instructores deseen omitir este último método debido a que es algo mecánico y es posible que no contribuya a una mejor comprensión del álgebra booleana. Esto puede hacerse sin afectar la continuidad o la claridad del resto del libro.

4-3 SIMPLIFICACIÓN ALGEBRAICA

Podemos utilizar los teoremas de álgebra booleana que estudiamos en el capítulo 3 para que nos ayuden a simplificar la expresión para un circuito lógico. Desafortunadamente, no siempre es obvio cuáles teoremas deben aplicarse para producir

el resultado más simple. Lo que es más, no hay una manera sencilla de saber si la expresión simplificada se encuentra en su forma más simple o si todavía puede simplificarse más. Por ende, la simplificación algebraica se vuelve a menudo un proceso de prueba y error. Sin embargo, con experiencia uno puede volverse un adepto para obtener resultados bastante razonables.

Los ejemplos que se muestran a continuación ilustrarán muchas de las maneras en las que pueden aplicarse los teoremas booleanos para tratar de simplificar una expresión. Estos ejemplos contienen dos pasos esenciales:

1. La expresión original se coloca en forma SOP mediante la aplicación repetida de los teoremas de DeMorgan y la multiplicación de los términos.
2. Una vez que la expresión original se encuentre en la forma SOP, se comprueba si hay factores comunes en los términos de productos y se lleva a cabo la factorización en donde sea posible. Este proceso de factorización deberá ayudar a eliminar uno o más términos.

EJEMPLO 4-1

Simplifique el circuito lógico que se muestra en la figura 4-2(a).

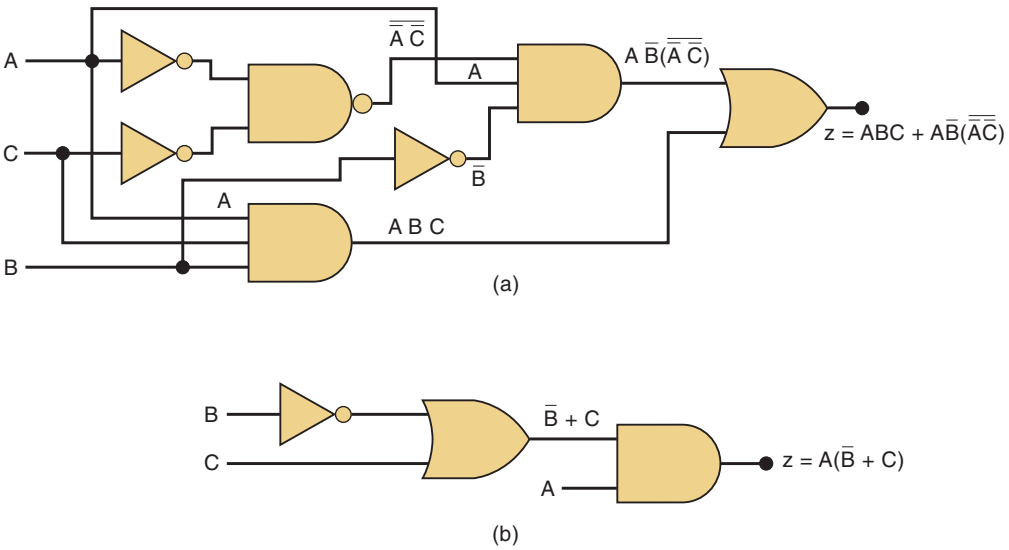


FIGURA 4-2 Ejemplo 4-1.

Solución

El primer paso es determinar la expresión para la salida, utilizando el método que se presentó en la sección 3-6. El resultado es

$$z = ABC + A\overline{B} \cdot (\overline{\overline{A}\,\overline{C}})$$

Una vez que se obtiene la expresión, por lo general, es conveniente descomponer todos los signos inversores grandes mediante el uso de los teoremas de DeMorgan, y después multiplicar todos los términos.

$$\begin{aligned} z &= ABC + A\overline{B}(\overline{\overline{A}} + \overline{\overline{C}}) && \text{[teorema (17)]} \\ &= ABC + A\overline{B}(A + C) && \text{[se cancelan las inversiones dobles]} \\ &= ABC + A\overline{B}A + A\overline{B}C && \text{[se realizan las multiplicaciones]} \\ &= ABC + A\overline{B} + A\overline{B}C && [A \cdot A = A] \end{aligned}$$

Ahora que la expresión se encuentra en la forma SOP, debemos buscar variables comunes entre los diversos términos con la intención de factorizar. Los términos primero y tercero de arriba tienen a AC en común, lo cual puede factorizarse:

$$z = AC(B + \overline{B}) + A\overline{B}$$

Como $B + \overline{B} = 1$, entonces

$$\begin{aligned} z &= AC(1) + A\overline{B} \\ &= AC + A\overline{B} \end{aligned}$$

Ahora podemos factorizar A, lo cual produce

$$z = A(C + \overline{B})$$

Este resultado ya no puede simplificarse más. En la figura 4-2(b) se muestra la implementación de su circuito. Es fácil observar que el circuito de la figura 4-2(b) es mucho más simple que el de la figura 4-2(a).

EJEMPLO 4-2

Simplifique la expresión $z = A\overline{B}\,\overline{C} + A\overline{B}C + ABC$.

Solución

La expresión ya se encuentra en la forma SOP.

Método 1: Los primeros dos términos de la expresión tienen el producto AB en común. Por lo tanto,

$$\begin{aligned} z &= A\overline{B}(\overline{C} + C) + ABC \\ &= A\overline{B}(1) + ABC \\ &= A\overline{B} + ABC \end{aligned}$$

Podemos factorizar la variable A de ambos términos:

$$z = A(\overline{B} + BC)$$

Utilizando el teorema (15b):

$$z = A(\overline{B} + C)$$

Método 2: La expresión original es $z = A\overline{B}\,\overline{C} + A\overline{B}C + ABC$. Los primeros dos términos tienen a $A\overline{B}$ en común. Los últimos dos tienen a AC en común. ¿Cómo podemos saber si debemos factorizar $A\overline{B}$ de los primeros dos términos, o AC de los últimos dos términos? En realidad podemos hacer ambas cosas si utilizamos el término $A\overline{B}C$ *dos veces*. En otras palabras, podemos reformular la expresión de la siguiente manera:

$$z = A\overline{B}\,\overline{C} + A\overline{B}C + A\overline{B}C + ABC$$

en donde hemos agregado un término $A\overline{B}C$. Esto es válido y no cambia el valor de la expresión, ya que $A\overline{B}C + A\overline{B}C = A\overline{B}C$ [(teorema (7)]. Ahora podemos factorizar $A\overline{B}$ de los primeros dos términos y AC de los últimos dos:

$$\begin{aligned} z &= A\overline{B}(C + \overline{C}) + AC(\overline{B} + B) \\ &= A\overline{B} \cdot 1 + AC \cdot 1 \\ &= A\overline{B} + AC = A(\overline{B} + C) \end{aligned}$$

Desde luego que este resultado es el mismo que se obtuvo con el método 1. Este truco de usar el mismo término dos veces puede usarse siempre. De hecho, el mismo término puede usarse más veces si es necesario.

EJEMPLO 4-3

Simplifique $z = \overline{A}C(\overline{\overline{A}BD}) + \overline{A}B\overline{C}\,\overline{D} + A\overline{B}C$.

Solución

Primero, utilizamos el teorema de DeMorgan en el primer término:

$$z = \overline{A}C(A + \overline{B} + \overline{D}) + \overline{A}B\overline{C}\,\overline{D} + A\overline{B}C \qquad \textbf{(paso 1)}$$

La multiplicación produce lo siguiente:

$$z = \overline{A}CA + \overline{A}C\overline{B} + \overline{A}C\overline{D} + \overline{A}B\overline{C}\,\overline{D} + A\overline{B}C \qquad \textbf{(2)}$$

Como $\overline{A} \cdot A = 0$, se elimina el primer término:

$$z = \overline{A}\,\overline{B}C + \overline{A}C\overline{D} + \overline{A}B\overline{C}\,\overline{D} + A\overline{B}C \qquad \textbf{(3)}$$

Ésta es la forma SOP deseada. Ahora debemos buscar factores comunes de entre los diversos términos de productos. La idea es buscar el factor común más grande entre dos o más términos de productos. Por ejemplo, los términos primero y último tienen el factor común $\overline{B}C$ y los términos segundo y tercero tienen el factor común $\overline{A}\,\overline{D}$. Podemos factorizar estos términos de la siguiente manera:

$$z = \overline{B}C(\overline{A} + A) + \overline{A}\,\overline{D}(C + B\overline{C}) \qquad \textbf{(4)}$$

Ahora, como $\overline{A} + A = 1$, y $C + B\overline{C} = C + B$ [teorema (15a)], tenemos que

$$z = \overline{B}C + \overline{A}\,\overline{D}(B + C) \qquad \textbf{(5)}$$

Este mismo resultado podría obtenerse si eligiéramos otros términos para la factorización. Por ejemplo, podríamos haber factorizado C de los términos de productos primero, segundo y cuarto, en el paso 3, para obtener:

$$z = C(\overline{A}\,\overline{B} + \overline{A}\,\overline{D} + A\overline{B}) + \overline{A}B\overline{C}\,\overline{D}$$

La expresión dentro de los paréntesis puede factorizarse aún más:

$$z = C(\overline{B}[\overline{A} + A] + \overline{A}\,\overline{D}) + \overline{A}B\overline{C}\,\overline{D}$$

Y como $\overline{A} + A = 1$, esta expresión se convierte en:

$$z = C(\overline{B} + \overline{A}\,\overline{D}) + \overline{A}B\overline{C}\,\overline{D}$$

Después de multiplicar, nos queda

$$z = \overline{B}C + \overline{A}C\,\overline{D} + \overline{A}B\overline{C}\,\overline{D}$$

Ahora podemos factorizar $\overline{A}\,\overline{D}$ de los términos segundo y tercero para obtener

$$z = \overline{B}C + \overline{A}\,\overline{D}(C + B\overline{C})$$

Si utilizamos el teorema (15a), la expresión entre paréntesis se convierte en $B + C$. Por lo tanto, queda

$$z = \overline{B}C + \overline{A}\,\overline{D}(B + C)$$

Este resultado es el mismo que obtuvimos antes, pero nos llevó muchos más pasos. Aquí se ilustra el porqué debemos buscar los factores comunes más grandes: por lo general, nos llevará a la expresión final en menos pasos.

El ejemplo 4-3 ilustra la frustración que se encuentra a menudo en la simplificación booleana. Como hemos llegado a la misma ecuación (que parece irreducible) por dos métodos distintos, podría parecer razonable concluir que esta ecuación final es la forma más simple. De hecho, la forma más simple de esta ecuación es

$$z = \overline{A}B\overline{D} + \overline{B}C$$

Pero no hay una manera aparente de reducir el paso (5) para llegar a esta versión más simple. En este caso nos faltó una operación en el proceso, la cual nos podría haber conducido a la forma más simple. La pregunta es, "¿Cómo hubiéramos podido saber que omitimos un paso?" Más adelante examinaremos una técnica de mapeo que siempre nos llevará a la forma SOP más simple.

EJEMPLO 4-4

Simplifique la expresión $x = (\overline{A} + B)(A + B + D)\overline{D}$.

Solución

La expresión puede colocarse en la forma de suma de productos mediante la multiplicación de todos los términos. El resultado es

$$x = \overline{A}A\overline{D} + \overline{A}B\overline{D} + \overline{A}D\overline{D} + BA\overline{D} + BB\overline{D} + BD\overline{D}$$

El primer término puede eliminarse, ya que $\overline{A}A = 0$. De igual forma, los términos tercero y sexto pueden eliminarse debido a que $D\overline{D} = 0$. El quinto término puede simplificarse a $B\overline{D}$, ya que $BB = B$. Esto nos da

$$x = \overline{A}B\overline{D} + AB\overline{D} + B\overline{D}$$

Podemos factorizar $B\overline{D}$ de cada término para obtener

$$x = B\overline{D}(\overline{A} + A + 1)$$

Es evidente que el término dentro de los paréntesis siempre es 1, por lo que al último nos queda

$$x = B\overline{D}$$

EJEMPLO 4-5

Simplifique el circuito de la figura 4-3(a).

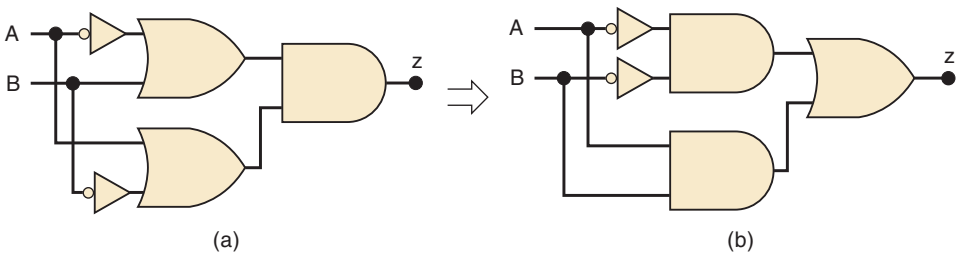


FIGURA 4-3 Ejemplo 4-5.

Solución

La expresión para la salida z es

$$z = (\overline{A} + B)(A + \overline{B})$$

Si multiplicamos para obtener la forma de suma de productos, obtenemos

$$z = \overline{A}A + \overline{A}\,\overline{B} + BA + B\overline{B}$$

Podemos eliminar $\overline{A}A = 0$ y $B\overline{B} = 0$ para terminar con

$$z = \overline{A}\,\overline{B} + AB$$

Esta expresión se implementa en la figura 4-3(b), y si la comparamos con el circuito original veremos que ambos circuitos contienen el mismo número de compuertas y conexiones. En este caso, el proceso de simplificación produjo un circuito equivalente pero más simple.

EJEMPLO 4-6

Simplifique $x = A\overline{B}C + \overline{A}BD + \overline{C}\,\overline{D}$.

Solución

Inténtelo, pero no podrá simplificar más esta expresión.

PREGUNTAS DE REPASO

1. Indique cuáles de las siguientes expresiones *no* se encuentran en la forma de suma de productos:
   (a) $RS\overline{T} + \overline{R}S\overline{T} + \overline{T}$
   (b) $A\overline{DC} + \overline{A}DC$
   (c) $MN\overline{P} + (M + \overline{N})P$
   (d) $AB + \overline{A}B\overline{C} + A\,\overline{B}\,\overline{C}\,D$
2. Simplifique el circuito de la figura 4-1(a) para obtener el circuito de la figura 4-1(b).
3. Cambie cada compuerta AND en la figura 4-1(a) por una compuerta NAND. Determine la nueva expresión para x y simplifíquela.

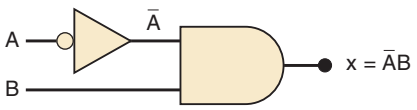
4-4 DISEÑO DE CIRCUITOS LÓGICOS COMBINACIONALES

Cuando se da el nivel de salida deseado de un circuito lógico para todas las posibles condiciones de entrada, los resultados pueden mostrarse de manera conveniente en una tabla de verdad. La expresión booleana para el circuito requerido puede entonces derivarse de la tabla de verdad. Por ejemplo, considere la figura 4-4(a), en donde se muestra la tabla de verdad para un circuito que tiene dos entradas A y B, y la salida x. La tabla muestra que la salida x estará en el nivel 1 *sólo* para el caso en el que $A = 0$ y $B = 1$. Ahora lo que resta es determinar qué circuito lógico producirá esta operación deseada. Debería ser evidente que una de las posibles soluciones es la que se muestra en la figura 4-4(b). Aquí se utiliza una compuerta AND con las entradas $\overline{A}$ y B, de manera que $x = \overline{A} \cdot B$. Es obvio que x será 1 *sólo* si ambas entradas de la compuerta AND son 1, a saber, $\overline{A} = 1$ (lo cual significa que $A = 0$) y $B = 1$. Para todos los demás valores de A y B, la salida x será 0.

FIGURA 4-4 Circuito que produce una salida de 1 sólo para la condición en la que $A = 0$ y $B = 1$.

A	B	x
0	0	0
0	1	1
1	0	0
1	1	0

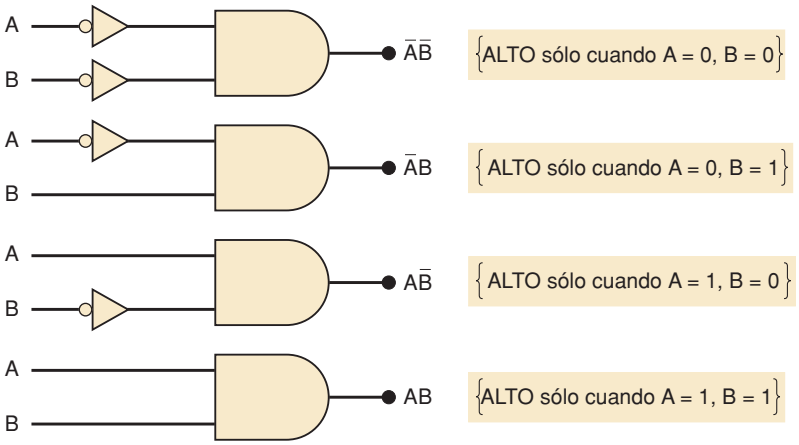
(a)



(b)

Puede usarse un enfoque similar para las demás condiciones de entrada. Por ejemplo, si x fuera a estar en nivel alto sólo para la condición $A = 1$, $B = 0$, el circuito resultante sería una compuerta AND con entradas A y $\overline{B}$. En otras palabras, para cualquiera de las cuatro posibles condiciones de entrada, podemos generar una salida x en nivel alto mediante el uso de una compuerta AND con las entradas apropiadas para generar el producto AND requerido. En la figura 4-5 se muestran los cuatro casos. Cada una de las compuertas AND que se muestran genera una salida que es 1 *sólo* para una condición de entrada dada y la salida es 0 para todas las demás condiciones. Hay que recalcar que las entradas AND son invertidas o no invertidas, dependiendo de los valores que tengan las variables para la condición dada. Si la variable es 0 para la condición dada, se invierte antes de entrar a la compuerta AND.

FIGURA 4-5 Una compuerta AND con las entradas apropiadas puede usarse para producir una salida de 1 para un conjunto específico de niveles de entrada.



Ahora consideremos el caso que se muestra en la figura 4-6(a), en donde tenemos una tabla de verdad que indica que la salida x debe ser 1 para dos casos distintos: $A = 0, B = 1$ y $A = 1, B = 0$. ¿Cómo puede implementarse esto? Sabemos que el término AND $\overline{A} \cdot B$ generará un 1 sólo para la condición $A = 0$, $B = 1$, y que el término AND $A \cdot \overline{B}$ generará un 1 para la condición $A = 1, B = 0$. Como x debe estar

en ALTO para *cualquiera* de esas condiciones, debe quedar claro que se debe aplicar una operación OR a estos términos para producir la salida x deseada. Esta implementación se muestra en la figura 4-6(b), en donde la expresión resultante para la salida es $x = \overline{A}B + A\overline{B}$.

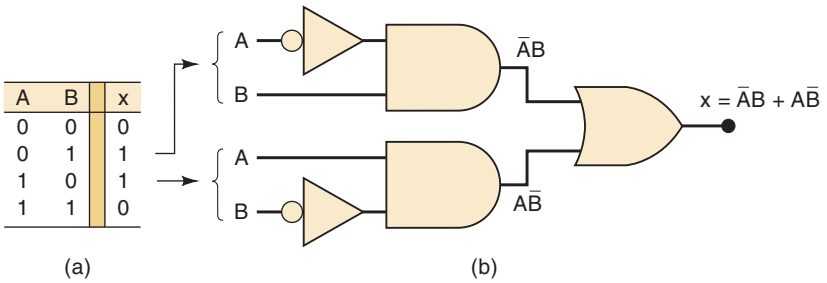


FIGURA 4-6 Cada conjunto de condiciones de entrada que debe producir una salida en ALTO se implementa mediante una compuerta AND separada. Se aplica una operación OR a las salidas de la compuerta AND para producir la salida final.

En este ejemplo se genera un término AND para cada caso en la tabla donde la salida x va a ser un 1. Después se aplica un OR a las salidas de la compuerta AND para producir la salida total x, que será 1 cuando cualquiera de los términos de la operación AND sea 1. Este mismo procedimiento puede extenderse a los ejemplos con más de dos entradas. Considere la tabla de verdad para un circuito de tres entradas (tabla 4-1). Aquí hay tres casos en donde la salida x debe ser 1. Se muestra el término AND requerido para cada uno de estos casos. Observe nuevamente que para cada caso en el que una variable es 0, ésta aparece invertida en el término AND. La expresión de suma de productos para x se obtiene aplicando una operación OR a los tres términos AND.

$$x = \overline{A}B\overline{C} + \overline{A}BC + ABC$$

TABLA 4-1

A	*B*	*C*	*x*	
0	0	0	0	
0	0	1	0	
0	1	0	1	$\rightarrow \overline{A}B\overline{C}$
0	1	1	1	$\rightarrow \overline{A}BC$
1	0	0	0	
1	0	1	0	
1	1	0	0	
1	1	1	1	$\rightarrow ABC$

Procedimiento completo de diseño

Cualquier problema lógico puede resolverse mediante el uso del siguiente procedimiento:

1. Interprete el problema y establezca una tabla de verdad para describir su operación.
2. Escriba el término AND (producto) para cada caso en el que la salida sea 1.
3. Escriba la expresión de suma de productos (SOP) para la salida.
4. Simplifique la expresión de salida, si es posible.
5. Implemente el circuito para la expresión final simplificada.

El siguiente ejemplo ilustra el procedimiento completo de diseño.

EJEMPLO 4-7

Diseñe un circuito lógico que tenga tres entradas A, B y C, y cuya salida esté en ALTO sólo cuando la mayoría de sus entradas estén en ALTO.

Solución

Paso 1. Establezca la tabla de verdad.

Con base en el enunciado del problema, la salida x deberá ser 1 siempre que dos o más entradas sean 1; para todos los demás casos, la salida deberá ser 0 (tabla 4-2).

TABLA 4-2

A	B	C	x	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\rightarrow \overline{A}BC$
1	0	0	0	
1	0	1	1	$\rightarrow A\overline{B}C$
1	1	0	1	$\rightarrow AB\overline{C}$
1	1	1	1	$\rightarrow ABC$

Paso 2. Escriba el término AND para cada caso en el que la salida sea un 1.

Hay cuatro casos así. Los términos AND se muestran enseguida de la tabla de verdad (tabla 4-2). Observe de nuevo que cada término AND contiene cada variable de entrada en su forma invertida o no invertida.

Paso 3. Escriba la expresión de suma de productos para la salida.

$$x = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

Paso 4. Simplifique la expresión de salida.

Esta expresión puede simplificarse de varias formas. Tal vez la más rápida sea considerar que el último término ABC tiene dos variables en común con cada uno de los otros términos. Por ende, podemos usar el término ABC para factorizarlo con cada uno de los demás términos. La expresión se reformula de manera que el término ABC ocurra tres veces (recuerde del ejemplo 4-2 que es posible hacer esto en el álgebra booleana):

$$x = \overline{A}BC + ABC + A\overline{B}C + ABC + AB\overline{C} + ABC$$

Si factorizamos los pares de términos apropiados, tenemos que

$$x = BC(\overline{A} + A) + AC(\overline{B} + B) + AB(\overline{C} + C)$$

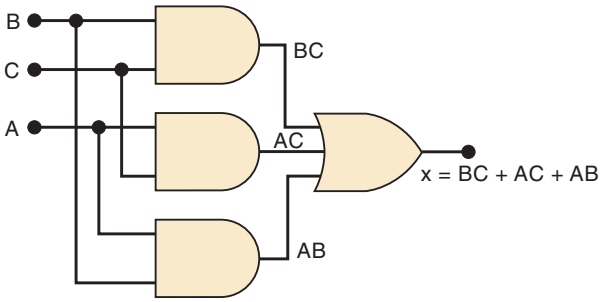
Cada término en paréntesis es igual a 1, por lo que nos queda

$$x = BC + AC + AB$$

Paso 5. Implemente el circuito para la expresión final.

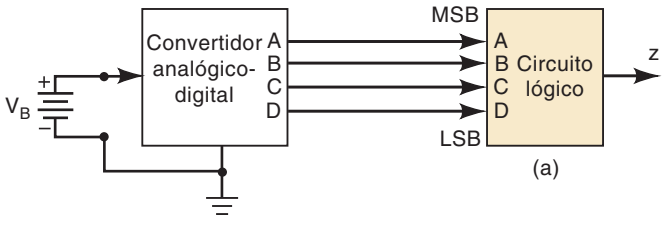
Esta expresión se implementa en la figura 4-7. Como la expresión se encuentra en la forma SOP, el circuito consiste de un grupo de compuertas AND conectadas con una sola compuerta OR.

FIGURA 4-7 Ejemplo 4-7.



EJEMPLO 4-8

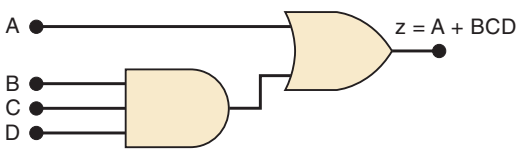
Consulte la figura 4-8(a), en donde un convertidor analógico-digital está monitoreando el voltaje de corriente directa de una batería de almacenamiento de 12 V en una nave espacial en órbita. La salida del convertidor es un número binario de 4 bits identificado como *ABCD*, que corresponde al voltaje de la batería en intervalos de 1 V, en donde *A* es el MSB. Las salidas binarias del convertidor se alimentan a un circuito lógico que debe producir una salida en ALTO siempre y cuando el valor binario sea mayor que $0110_2 = 6_{10}$; esto es, que el voltaje de la batería sea mayor que 6 V. Diseñe este circuito lógico.



(a)

	A	B	C	D	z
(0)	0	0	0	0	0
(1)	0	0	0	1	0
(2)	0	0	1	0	0
(3)	0	0	1	1	0
(4)	0	1	0	0	0
(5)	0	1	0	1	0
(6)	0	1	1	0	0
(7)	0	1	1	1	1→ $\bar{A}BCD$
(8)	1	0	0	0	1→ $A\bar{B}\bar{C}\bar{D}$
(9)	1	0	0	1	1→ $A\bar{B}\bar{C}D$
(10)	1	0	1	0	1→ $A\bar{B}C\bar{D}$
(11)	1	0	1	1	1→ $A\bar{B}CD$
(12)	1	1	0	0	1→ $AB\bar{C}\bar{D}$
(13)	1	1	0	1	1→ $AB\bar{C}D$
(14)	1	1	1	0	1→ $ABC\bar{D}$
(15)	1	1	1	1	1→ $ABCD$

(b)



(c)

FIGURA 4-8 Ejemplo 4-8.

Solución

La tabla de verdad se muestra en la figura 4-8(b). Para cada caso en la tabla de verdad hemos indicado el equivalente decimal del número binario representado por la combinación *ABCD*.

La salida *z* es igual a 1 para todos aquellos casos en los que el número binario sea mayor que 0110. Para todos los demás casos, *z* es igual a 0. Esta tabla de verdad nos da la siguiente expresión de suma de productos:

$$z = \bar{A}BCD + A\bar{B}\,\bar{C}\,\bar{D} + A\bar{B}\,\bar{C}D + A\bar{B}C\bar{D} + A\bar{B}CD + AB\bar{C}\,\bar{D} + AB\bar{C}D + ABC\bar{D} + ABCD$$

La simplificación de esta expresión será una formidable tarea, pero con un poco de cuidado puede lograrse. El proceso paso a paso implica factorizar y eliminar términos de la forma $A + \overline{A}$:

$$\begin{aligned} z &= \overline{A}BCD + A\overline{B}\,\overline{C}(\overline{D} + D) + A\overline{B}C(\overline{D} + D) + AB\overline{C}(\overline{D} + D) + ABC(\overline{D} + D) \\ &= \overline{A}BCD + A\overline{B}\,\overline{C} + A\overline{B}C + AB\overline{C} + ABC \\ &= \overline{A}BCD + A\overline{B}(\overline{C} + C) + AB(\overline{C} + C) \\ &= \overline{A}BCD + A\overline{B} + AB \\ &= \overline{A}BCD + A(\overline{B} + B) \\ &= \overline{A}BCD + A \end{aligned}$$

Esta expresión puede reducirse aún más si aplicamos el teorema (15a), el cual nos dice que $x + \overline{x}y = x + y$. En este caso $x = A$ y $y = BCD$. Por lo tanto,

$$z = \overline{A}BCD + A = BCD + A$$

Esta expresión final se implementa en la figura 4-8(c).

Como lo demuestra este ejemplo, el método de simplificación algebraica puede ser bastante largo cuando la expresión original contiene un gran número de términos. Ésta es una limitación que no comparte el método de mapeo de Karnaugh, como veremos más adelante.

EJEMPLO 4-9

Observe la figura 4-9(a). En una copiadora simple, se debe generar una señal de paro S para detener la operación de la máquina y encender una luz indicadora cada vez que exista una de las siguientes condiciones: (1) que no haya papel en la bandeja alimentadora; o (2) que se activen los dos microinterruptores en la ruta del papel, lo cual indica un atasco. La presencia de papel en la bandeja alimentadora se indica

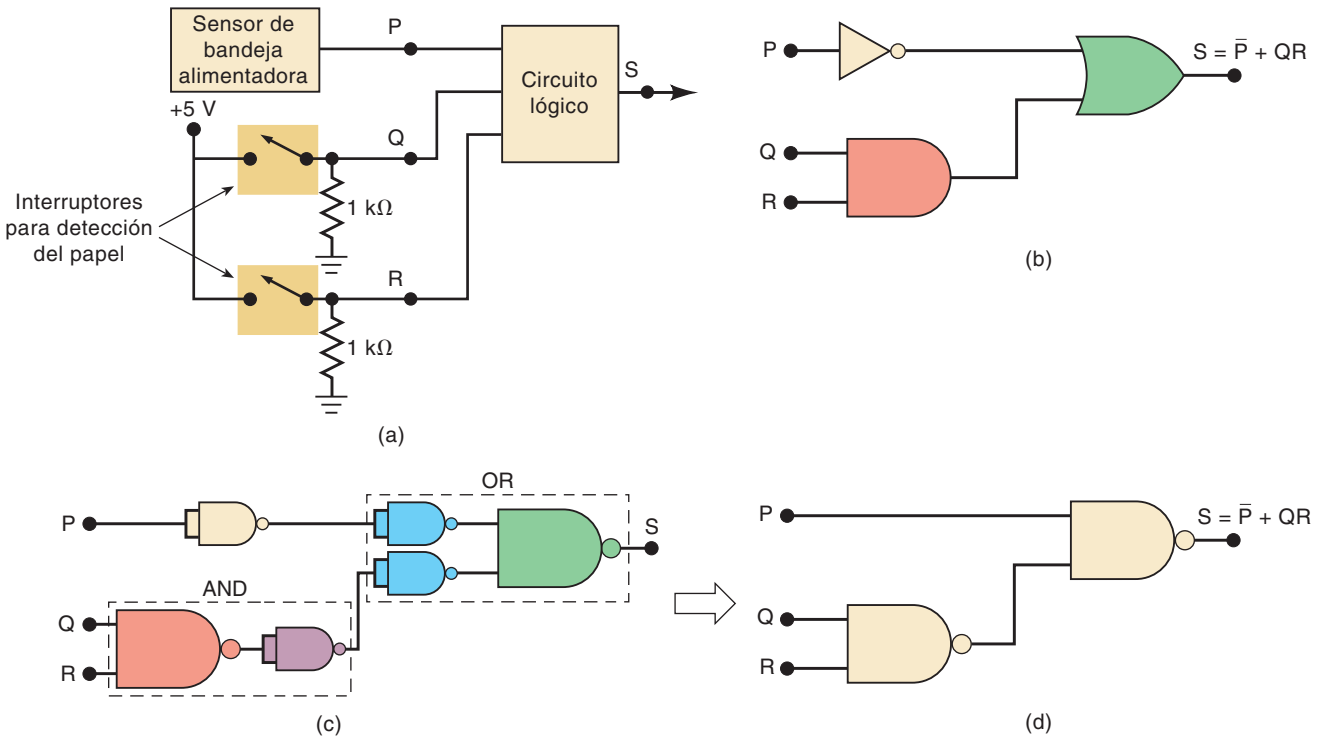


FIGURA 4-9 Ejemplo 4-9.

mediante un nivel ALTO en la señal lógica *P*. Cada microinterruptor produce una señal lógica (*Q* y *R*, *respectivamente*) que cambia a ALTO cada vez que el papel pasa por el interruptor y lo activa. Diseñe el circuito lógico para producir un nivel ALTO en la señal de salida *S* para las condiciones antes mencionadas, e impleméntelo utilizando el chip con cuatro compuertas NAND de dos entradas 74HC00 CMOS.

Solución

Utilizaremos el proceso de cinco pasos indicado en el ejemplo 4-7. La tabla de verdad se muestra en la figura 4-3. La salida *S* será un 1 lógico siempre que $P = 0$, ya que esto indica que no hay papel en la bandeja alimentadora. *S* también será un 1 para los dos casos en los que *Q* y *R* sean ambas 1, lo cual indica un atasco de papel. Como la tabla indica, hay cinco condiciones de entrada distintas que producen una salida en ALTO. **(Paso 1)**

TABLA 4-3

P	*Q*	*R*	*S*	
0	0	0	1	$\overline{P}\,\overline{Q}\,\overline{R}$
0	0	1	1	$\overline{P}\,\overline{Q}R$
0	1	0	1	$\overline{P}Q\overline{R}$
0	1	1	1	$\overline{P}QR$
1	0	0	0	
1	0	1	0	
1	1	0	0	
1	1	1	1	PQR

Se muestran los términos AND para cada uno de estos casos. **(Paso 2)**
La expresión de suma de productos sería

$$S = \overline{P}\,\overline{Q}\,\overline{R} + \overline{P}\,\overline{Q}R + \overline{P}Q\overline{R} + \overline{P}QR + PQR \qquad \textbf{(Paso 3)}$$

Podemos empezar la simplificación factorizando $\overline{P}\,\overline{Q}$ de los términos 1 y 2, y factorizando $\overline{P}Q$ de los términos 3 y 4:

$$S = \overline{P}\,\overline{Q}(\overline{R} + R) + \overline{P}Q(\overline{R} + R) + PQR \qquad \textbf{(Paso 4)}$$

Ahora podemos eliminar los términos $R + R$ ya que son iguales a 1:

$$S = \overline{P}\,\overline{Q} + \overline{P}Q + PQR$$

Si factorizamos $\overline{P}$ de los términos 1 y 2 podemos eliminar *Q*:

$$S = \overline{P} + PQR$$

Aquí, podemos aplicar el teorema (15b) ($\overline{x} + xy = \overline{x} + y$) para obtener

$$S = \overline{P} + QR$$

Como una comprobación adicional de esta ecuación booleana simplificada, veamos si concuerda con la tabla de verdad con la que comenzamos. Esta ecuación dice que la salida *S* estará en ALTO siempre que *P* esté en BAJO OR cuando *Q* AND *R* estén en ALTO. Consulte la tabla 4-3 y observe que la salida está en ALTO para los cuatro casos en los que *P* está en BAJO. *S* también está en ALTO cuando *Q* AND *R*

están ambas en ALTO, sin importar el estado de *P*. Lo anterior concuerda con la ecuación.

La implementación AND/OR para este circuito se muestra en la figura 4-9(b).

(Paso 5)

Para implementar este circuito usando el chip NAND de dos entradas 74HC00, debemos convertir cada una de las compuertas y el INVERSOR en sus compuertas NAND equivalentes (con base en la sección 3-12). Esto se muestra en la figura 4-9(c). Es evidente que podemos eliminar los inversores dobles para producir la implementación con compuertas NAND que se muestra en la figura 4-9(d).

El circuito alambrado final se obtiene mediante la conexión de dos de las compuertas NAND en el chip 74HC00. Este chip CMOS tiene la misma configuración de compuertas y los mismos números de terminales que el chip 74LS00 TTL de la figura 3-31. La figura 4-10 muestra el circuito alambrado con números de terminales, incluyendo las terminales +5 V y tierra (GND). También incluye un transistor excitador de salida y un LED para indicar el estado de la salida *S*.

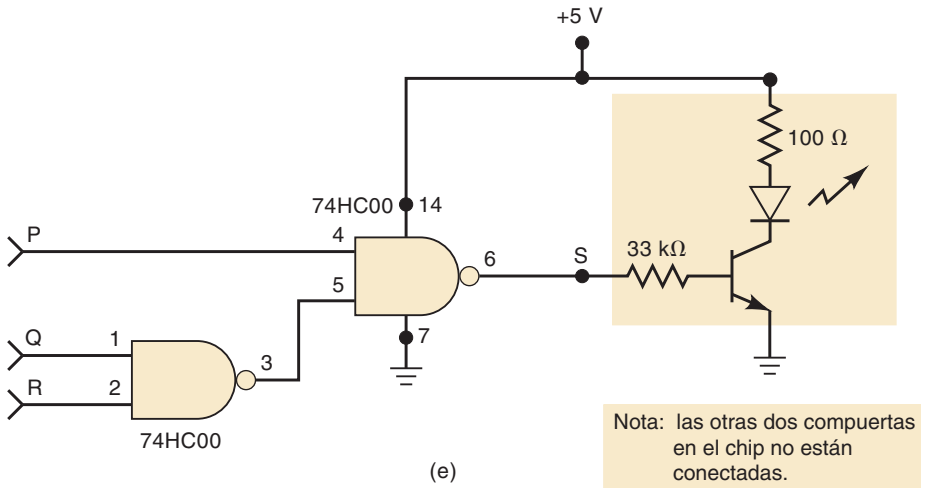


FIGURA 4-10 Circuito para la figura 4-9(d) implementado mediante el uso del chip NAND 74HC00.

PREGUNTAS DE REPASO

1. Escriba la expresión de suma de productos para un circuito con cuatro entradas y una salida que debe estar en ALTO sólo cuando la entrada *A* esté en BAJO al mismo tiempo que dos de las otras entradas estén en BAJO.
2. Implemente la expresión de la pregunta 1 utilizando sólo compuertas NAND de cuatro entradas. ¿Cuántas se requieren?

4-5 MÉTODO DE MAPAS DE KARNAUGH

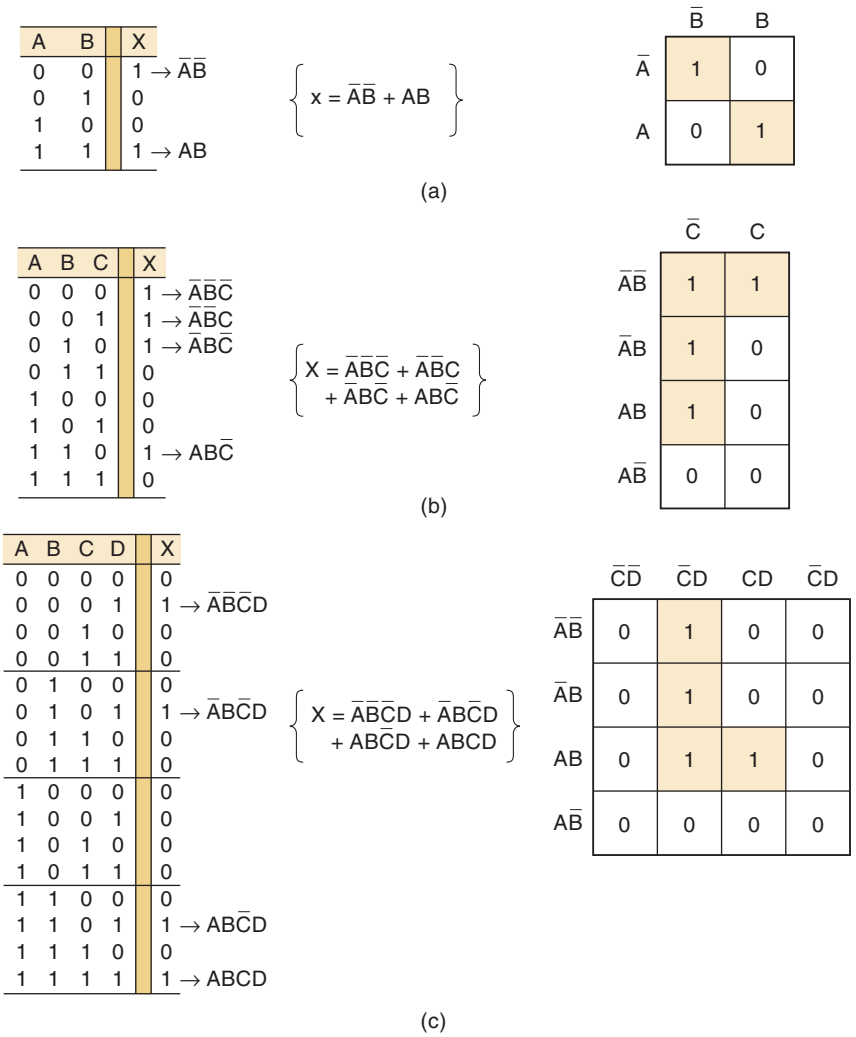
El **mapa de Karnaugh (mapa K)** es una herramienta gráfica que se utiliza para simplificar una ecuación lógica o convertir una tabla de verdad en su correspondiente circuito lógico mediante un proceso simple y ordenado. Aunque un mapa K puede usarse para problemas en los que se involucre cualquier número de variables de entrada, su utilidad práctica está limitada a cinco o seis variables. Los siguientes problemas estarán limitados a un máximo de cuatro entradas, ya que los problemas con cinco o más entradas son demasiado complicados y se resuelven mejor mediante el uso de un programa de computadora.

Formato del mapa de Karnaugh

Al igual que una tabla de verdad, el mapa K es un medio para mostrar la relación entre las entradas lógicas y la salida deseada. La figura 4-11 muestra tres ejemplos de mapas K para dos, tres y cuatro variables, junto con sus correspondientes tablas de verdad. Estos ejemplos ilustran los siguientes puntos importantes:

1. La tabla de verdad proporciona el valor de la salida X para cada combinación de valores de entrada. El mapa K proporciona la misma información en un formato distinto. Cada caso en la tabla de verdad corresponde a una casilla en el mapa K. Por ejemplo, en la figura 4-11(a) la condición $A = 0, B = 0$ corresponde la casilla $\overline{A}\,\overline{B}$ en el mapa K. Como la tabla de verdad muestra $X = 1$ para este caso, se coloca un 1 en la casilla $\overline{A}\,\overline{B}$ del mapa K. De manera similar, la condición $A = 1, B = 1$ en la tabla de verdad corresponde a la casilla AB del mapa K. Como $X = 1$ para este caso, se coloca un 1 en la casilla AB. Todas las demás casillas se llenan con 0s. Esta misma idea se utiliza en los mapas con tres y cuatro variables que se muestran en la figura.
2. Las casillas del mapa K se etiquetan de manera que las casillas adyacentes en forma horizontal difieran sólo por una variable. Por ejemplo, la casilla de la esquina superior izquierda en el mapa de cuatro variables es $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$, mientras que la casilla que se encuentra justo a su derecha es $\overline{A}\,\overline{B}\,\overline{C}\,D$ (sólo la variable D es distinta). De manera similar, las casillas adyacentes verticalles sólo difie-

FIGURA 4-11 Mapas de Karnaugh y tablas de verdad para (a) dos, (b) tres y (c) cuatro variables.



ren por una variable. Por ejemplo, la casilla de la esquina superior izquierda es $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$, mientras que la casilla que está justo debajo es $\overline{A}\,\overline{B}\,\overline{C}\,D$ (sólo la variable B es distinta).

Observe que cada casilla en la fila superior se considera como adyacente a una casilla correspondiente en la fila inferior. Por ejemplo, la casilla $\overline{A}\,\overline{B}CD$ en la fila superior es adyacente a la casilla $A\overline{B}CD$ en la fila inferior, ya que sólo difieren por la variable A. Podemos considerar que la parte superior del mapa se dobla para tocar su parte inferior. De manera similar, las casillas de la columna más a la izquierda son adyacentes a las correspondientes en la columna más a la derecha.

3. Para que las casillas adyacentes en forma vertical y horizontal difieran sólo por una variable, el etiquetado de arriba hacia abajo debe realizarse en el orden mostrado: $\overline{A}\,\overline{B}$, $\overline{A}B$, AB, $A\overline{B}$. Lo mismo aplica para el etiquetado de izquierda a derecha: $\overline{C}\,\overline{D}$, $\overline{C}D$, CD, $C\overline{D}$.
4. Una vez que se ha llenado un mapa K con 0s y 1s, puede obtenerse la expresión de suma de productos para la salida X mediante la aplicación de la operación OR a todas las casillas que contengan un 1. En el mapa de tres variables de la figura 4-11(b), las casillas $\overline{A}\,\overline{B}\,\overline{C}$, $\overline{A}\,\overline{B}C$, $\overline{A}B\overline{C}$ y $AB\overline{C}$ contienen un 1, de manera que $X = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + AB\overline{C}$.

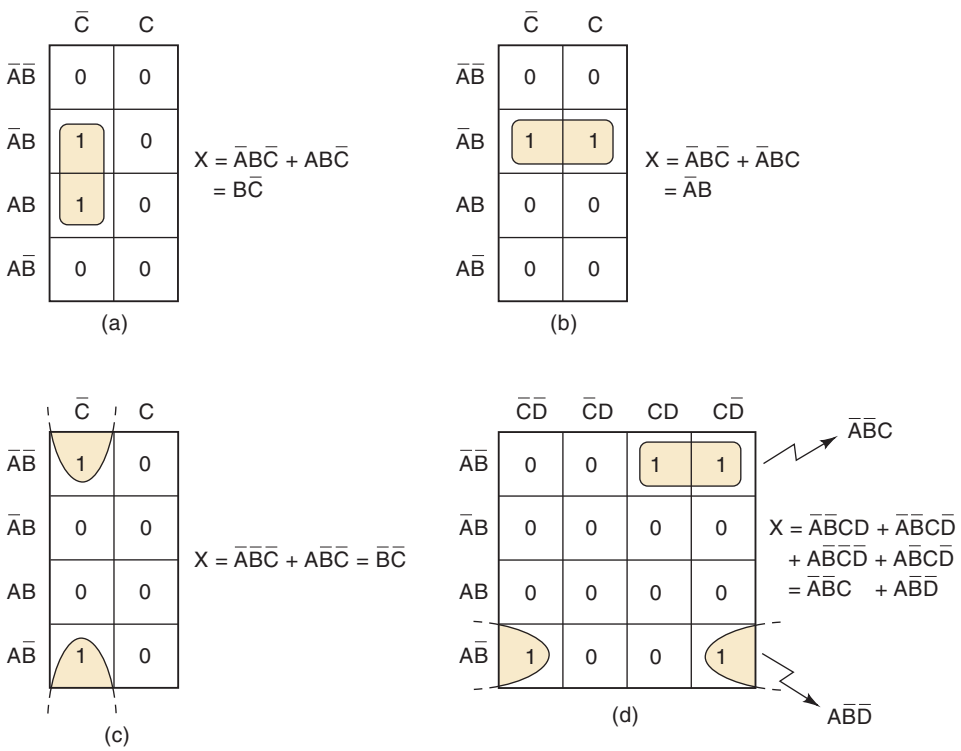
Agrupamiento

La expresión para la salida X puede simplificarse mediante la combinación apropiada de las casillas en el mapa K que contengan 1s. Al proceso para combinar estos 1s se le conoce como **agrupamiento**.

Agrupamiento de pares (grupos de dos)

La figura 4-12(a) es el mapa K para cierta tabla de verdad de tres variables. Este mapa contiene un par de 1s que son adyacentes en forma vertical; el primero representa a $\overline{A}B\overline{C}$ y el segundo a $AB\overline{C}$. Observe que en estos dos términos, sólo la variable A aparece

FIGURA 4-12
Ejemplos de agrupamientos de pares de 1s adyacentes.



tanto en forma normal como complementada (invertida), mientras que B y $\overline{C}$ permanecen sin cambios. Estos dos términos pueden agruparse (combinarse) para obtener un resultante que elimine la variable A, ya que aparece tanto en forma complementada como no complementada. Es fácil demostrarlo de la siguiente manera:

$$\begin{aligned} X &= \overline{A}B\overline{C} + AB\overline{C} \\ &= B\overline{C}(\overline{A} + A) \\ &= B\overline{C}(1) = B\overline{C} \end{aligned}$$

Este mismo principio se aplica para cualquier par de 1s adyacentes en forma vertical u horizontal. La figura 4-12(b) muestra un ejemplo de dos 1s adyacentes en forma horizontal; los cuales pueden agruparse, y la variable C puede eliminarse ya que aparece tanto en su forma no complementada como en su forma complementada, para obtener un resultante de $X = AB$.

La figura 4-12(c) muestra otro ejemplo. En un mapa K, la fila superior y la fila inferior de casillas se consideran adyacentes. Por ende, los dos 1s en este mapa pueden agruparse para obtener un resultante de $\overline{A}\,\overline{B}\,\overline{C} + A\overline{B}\,\overline{C} = \overline{B}\,\overline{C}$.

La figura 4-12(d) muestra un mapa K que tiene dos pares de 1s que pueden agruparse. Los dos 1s en la fila superior son adyacentes en forma horizontal. Los dos 1s en la fila inferior también son adyacentes ya que, en un mapa K la columna más a la izquierda y la columna más a la derecha se consideran adyacentes. Cuando se agrupa el par de 1s de la parte superior se elimina la variable D (ya que aparece como D y como $\overline{D}$) para producir el término $\overline{A}\,\overline{B}C$. Al agrupar el par de la parte inferior se elimina la variable C para producir el término $A\overline{B}\,\overline{D}$. Después se aplica una operación OR a estos dos términos para obtener el resultado final para X.

En resumen:

Al agrupar un par de 1s adyacentes en un mapa K se elimina la variable que aparece tanto en forma no complementada como en forma complementada.

Agrupamiento de cuádruples (grupos de cuatro)

Un mapa K puede contener un grupo de cuatro 1s que sean adyacentes. A este grupo se le conoce como *cuádruple*. La figura 4-13 muestra varios ejemplos de este tipo. En la figura 4-13(a), los cuatro 1s son adyacentes en forma vertical y en la figura 4-13(b) son adyacentes en forma horizontal. El mapa K de la figura 4-13(c) contiene cuatro 1s en una casilla y se consideran adyacentes entre sí. Los cuatro 1s de la figura 4-13(d) también son adyacentes, al igual que los de la figura 4-13(e) ya que, como dijimos antes, las filas superior e inferior se consideran como adyacentes entre sí, al igual que las columnas más a la izquierda y más a la derecha.

Cuando se agrupa un cuádruple, el término resultante sólo contendrá las variables que no cambian su forma en todas las casillas del cuádruple. Por ejemplo, en la figura 4-13(a) las cuatro casillas que contienen un 1 son $\overline{A}\,\overline{B}C$, $\overline{A}BC$, ABC y $A\overline{B}C$. Si examinamos estos términos descubriremos que sólo la variable C permanece sin cambios (tanto A como B aparecen en forma complementada y no complementada). Por ende, la expresión resultante para X es tan sólo $X = C$. Esto puede demostrarse de la siguiente manera:

$$\begin{aligned} X &= \overline{A}\,\overline{B}C + \overline{A}BC + ABC + A\overline{B}C \\ &= \overline{A}C(\overline{B} + B) + AC(B + \overline{B}) \\ &= \overline{A}C + AC \\ &= C(\overline{A} + A) = C \end{aligned}$$

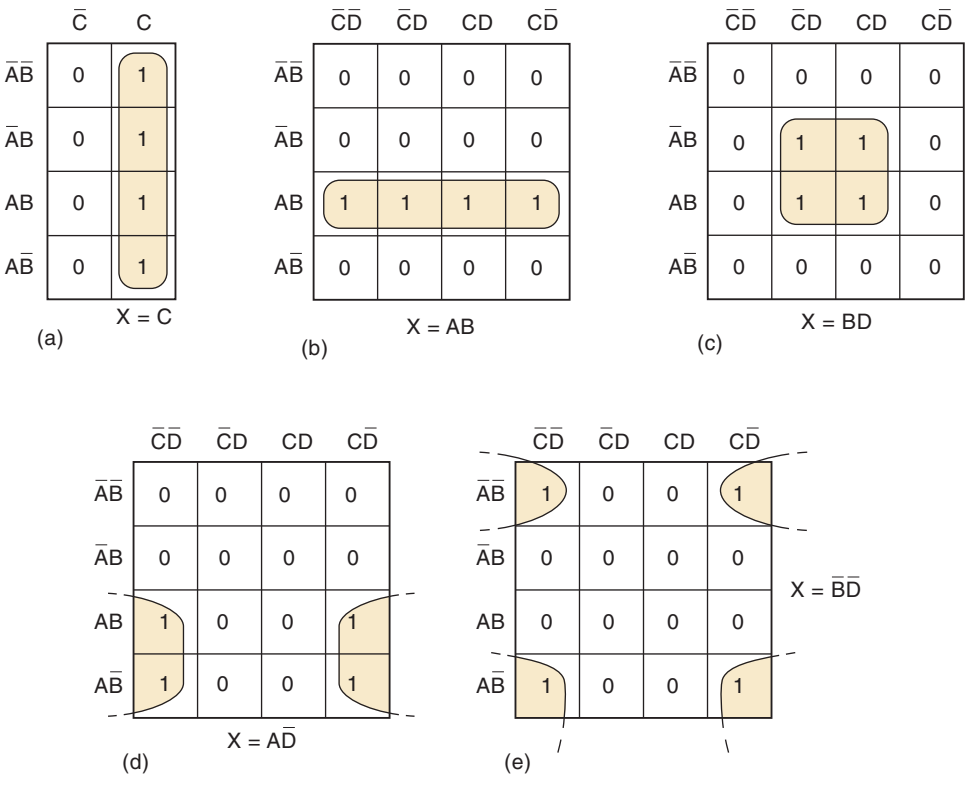


FIGURA 4-13
Ejemplos de agrupamiento de cuádruples.

Como otro ejemplo, considere la figura 4-13(d), en donde las cuatro casillas que contienen 1s son $ABC\overline{D}$, $A\overline{B}\,\overline{C}\,\overline{D}$, $ABC\overline{D}$ y $A\overline{B}C\overline{D}$. Al analizar estos términos podemos ver que sólo las variables A y D permanecen sin cambios, de manera que la expresión simplificada para X es

$$X = A\overline{D}$$

Lo que puede demostrarse de la misma forma que se hizo antes. El lector deberá comprobar cada uno de los otros casos de la figura 4-13 para verificar las expresiones indicadas para X.

En resumen:

Al agrupar un cuádruple de 1s adyacentes se eliminan las dos variables que aparecen tanto en forma complementada como en forma no complementada.

Agrupamiento de octetos (grupos de ocho)

A un grupo de ocho 1s adyacentes entre sí se le conoce como *octeto*. La figura 4-14 muestra varios ejemplos de octetos. Cuando se agrupa un octeto en un mapa de cuatro variables se eliminan tres de ellas, ya que sólo una permanece sin cambios. Por ejemplo, si examinamos las ocho casillas agrupadas en la figura 4-14(a) podremos ver que sólo la variable B se encuentra en la misma forma para las ocho casillas: las demás variables aparecen en su forma complementada y no complementada. En consecuencia, para este mapa $X = B$. El lector puede verificar los resultados para los demás ejemplos de la figura 4-14.

En resumen:

Al agrupar un octeto de 1s adyacentes se eliminan las tres variables que aparecen tanto en su forma complementada como en su forma no complementada.

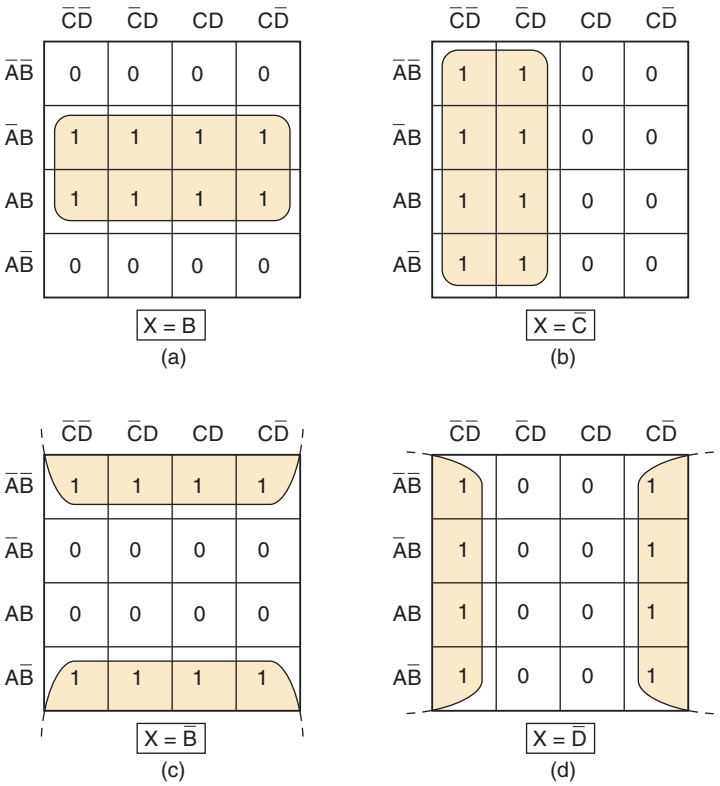


FIGURA 4-14
Ejemplos de agrupamiento de octetos.

Proceso completo de simplificación

Hemos visto cómo se puede utilizar el agrupamiento de pares, cuádruples y octetos en un mapa K para obtener una expresión simplificada. Podemos resumir la regla para los agrupamientos de *cualquier* tamaño, de la siguiente manera:

> **Cuando una variable aparece tanto en su forma complementada como no complementada dentro de un grupo, esa variable se elimina de la expresión. Las variables que son iguales para todas las casillas del grupo deben aparecer en la expresión final.**

Debe quedar claro que un agrupamiento mayor de 1s elimina más variables. Para ser exacto, un agrupamiento de dos elimina una variable, un agrupamiento de cuatro elimina dos variables y un agrupamiento de ocho elimina tres. Ahora utilizaremos este principio para obtener una expresión lógica simplificada a partir de un mapa K que contenga cualquier combinación de 1s y 0s.

Primero describiremos el procedimiento y después lo aplicaremos en varios ejemplos. Los siguientes pasos son el procedimiento mediante el uso del método del mapa K, para simplificar una expresión booleana:

Paso 1 Construya el mapa K y coloque 1s en las casillas que correspondan a los 1s en la tabla de verdad. Coloque 0s en las demás casillas.

Paso 2 Examine el mapa en busca de 1s adyacentes y marque los que *no* sean adyacentes con cualquier otro 1. A éstos se les conoce como 1s *aislados*.

Paso 3 A continuación busque los 1s que sean adyacentes sólo con otro 1. Agrupe *cualquier* par que contenga este tipo de 1s.

Paso 4 Agrupe cualquier octeto, aún y cuando contenga algunos 1s que ya se hayan agrupado.

Paso 5 Agrupe cualquier cuádruple que contenga uno o más 1s que no se hayan agrupado ya, *asegurándose de utilizar el número mínimo de grupos*.

Paso 6 Agrupe cualquier par necesario para incluir todos los 1 que no se hayan agrupado todavía, *asegurándose de utilizar el número mínimo de agrupamientos.*

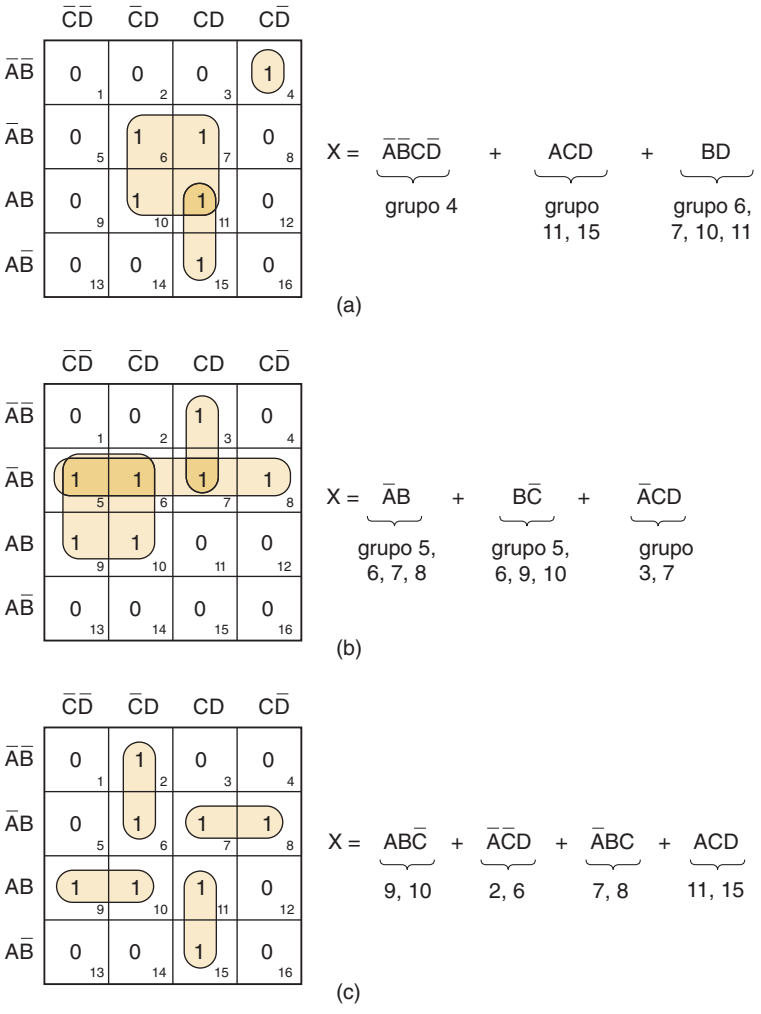
Paso 7 Forme la suma OR de todos los términos generados, uno por cada grupo.

En los siguientes ejemplos seguiremos al pie de la letra cada uno de estos pasos y haremos referencia a ellos. En cada caso, la expresión lógica resultante estará en su forma de suma de productos más simple.

EJEMPLO 4-10

La figura 4-15(a) muestra el mapa K para un problema con cuatro variables. Supondremos que el mapa se obtuvo de la tabla de verdad del problema (**paso 1**). Las casillas están numeradas para identificar cada uno de los grupos.

FIGURA 4-15
Ejemplos 4-10 al 4-12.



Paso 2 La casilla 4 es la única que contiene un 1 que no es adyacente con cualquier otro 1. Se agrupa y se identifica como grupo 4.

Paso 3 La casilla 15 es adyacente *sólo* con la casilla 11. Este par se agrupa y se identifica como grupo 11, 15.

Paso 4 No hay octetos.

Paso 5 Las casillas 6, 7, 10 y 11 forman un cuádruple. Este cuádruple se agrupa (grupo 6, 7, 10, 11). Observe que la casilla 11 se utiliza de nuevo, aun y cuando forma parte del grupo 11, 15.

Paso 6 Ya se han agrupado todos los 1s.

Paso 7 Cada grupo genera un término en la expresión para X. El grupo 4 es $\overline{A}\,\overline{B}C\overline{D}$. El grupo 11, 15 es ACD (se elimina la variable B). El grupo 6, 7, 10, 11 es BD (se eliminan A y C).

EJEMPLO 4-11

Considere el mapa K de la figura 4-15(b). Una vez más podemos suponer que ya se ha realizado el paso 1.

Paso 2 No hay 1s aislados.

Paso 3 El 1 en la casilla 3 es adyacente *sólo* con el 1 en la casilla 7. Al agrupar este par (grupo 3, 7) se produce el término $\overline{A}CD$.

Paso 4 No hay octetos.

Paso 5 Hay dos cuádruples: El primero lo forman los cuadros 5, 6, 7 y 8 . Al agrupar este cuádruple se produce el término $\overline{A}B$. El segundo está compuesto por las casillas 5, 6, 9 y 10. Se debe agrupar este cuádruple, ya que contiene dos casillas que no se han agrupado antes. Al agruparlo se produce $\overline{B}C$.

Paso 6 Ya se han agrupado todos los 1s.

Paso 7 Se aplica la operación OR a los términos generados por los tres grupos y se obtiene la expresión para X.

EJEMPLO 4-12

Considere el mapa K de la figura 4-15(c):

Paso 2 No hay 1s aislados.

Paso 3 El 1 en la casilla 2 es adyacente sólo para el 1 en la casilla 6. Este par se agrupa para producir $\overline{A}\,\overline{C}D$. De manera similar, la casilla 9 es adyacente sólo con la casilla 10. Al agrupar este par se produce $AB\overline{C}$. De igual forma, los grupos 7, 8 y 11, 15 producen los términos $\overline{A}BC$ y ACD, en forma correspondiente.

Paso 4 No hay octetos.

Paso 5 Sólo hay un cuádruple formado por las casillas 6, 7, 10 y 11. No obstante, este cuádruple *no* se agrupa debido a que todos los 1s que contiene ya se han incluido en otros grupos.

Paso 6 Ya se han agrupado todos los 1s.

Paso 7 La expresión para X se muestra en la figura.

EJEMPLO 4-13

Considere el mapa K de la figura 4-16(a).

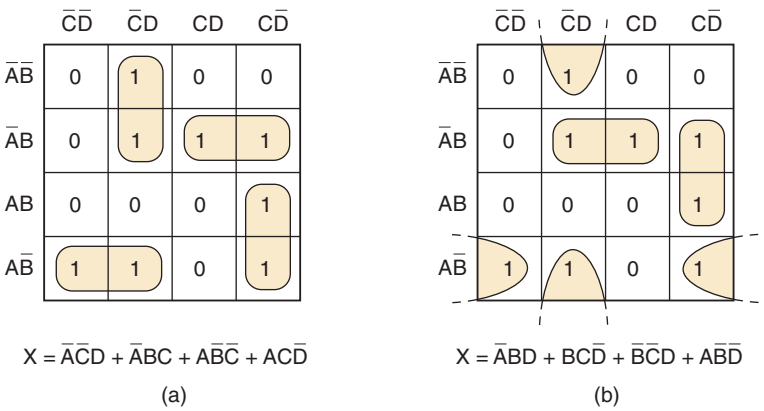


FIGURA 4-16 El mismo mapa K con dos soluciones correctas similares.

Paso 2 No hay 1s aislados.

Paso 3 No hay 1s que sean adyacentes sólo con otro 1.

Paso 4 No hay octetos.

Paso 5 No hay cuádruples.

Pasos 6 y 7 Hay muchos pares posibles. El agrupamiento debe utilizar el número mínimo de grupos para tomar en cuenta a todos los 1s. Para este mapa hay *dos* grupos posibles, los cuales requieren sólo cuatro pares agrupados. La figura 4-16(a) muestra una solución y su expresión resultante. La figura 4-16(b) muestra la otra. Observe que ambas expresiones son de la misma complejidad, por lo cual ninguna es mejor que la otra.

Cómo llenar un mapa K a partir de una expresión de salida

Cuando la salida deseada se presenta como expresión booleana en vez de tabla de verdad, el mapa K puede llenarse mediante el uso de los siguientes pasos:

1. Cambie la expresión a su forma SOP, en caso de que no se encuentre ya en esa forma.
2. Para cada término de productos en la expresión SOP, coloque un 1 en cada casilla del mapa K cuya etiqueta contenga la misma combinación de variables de entrada. Coloque un 0 en todas las demás casillas.

El siguiente ejemplo ilustra este procedimiento.

EJEMPLO 4-14

Use un mapa K para simplificar la expresión $y = \overline{C}(\overline{A}\,\overline{B}\,\overline{D} + D) + A\overline{B}C + \overline{D}$.

Solución

1. Multiplique el primer término para obtener $y = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{C}D + A\overline{B}C + \overline{D}$, que se encuentra ahora en la forma SOP.
2. Para el término $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$ sólo necesita colocar un 1 en la casilla $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$ del mapa K (figura 4-17). Para el término $\overline{C}D$ coloque un 1 en todas las casillas que tengan $\overline{C}D$ en sus etiquetas: $\overline{A}\,\overline{B}\,\overline{C}D$, $\overline{A}B\overline{C}D$, $AB\overline{C}D$, $A\overline{B}\,\overline{C}D$. Para el término $A\overline{B}C$ coloque un 1 en todas las casillas que tengan un $A\overline{B}C$ en sus etiquetas: $A\overline{B}C\overline{D}$, $A\overline{B}CD$. Para el término $\overline{D}$ coloque un 1 en todas las casillas que tengan una $\overline{D}$ en sus etiquetas: en todas las casillas de las columnas más a la izquierda y más a la derecha.

FIGURA 4-17 Ejemplo 4-14.

	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$	1	1	0	1
$\overline{A}B$	1	1	0	1
AB	1	1	0	1
$A\overline{B}$	1	1	1	1

$y = A\overline{B} + \overline{C} + \overline{D}$

Ahora el mapa K está lleno y puede agruparse para simplificar la expresión. Verifique que un agrupamiento apropiado produzca la expresión $y = A\overline{B} + \overline{C} + \overline{D}$.

Condiciones "No importa"

Algunos circuitos lógicos pueden diseñarse de manera que haya ciertas condiciones de entrada para las cuales no existan niveles de salida especificados, por lo general, debido a que estas condiciones de entrada nunca ocurrirán. En otras palabras, habrá ciertas combinaciones de niveles de entrada en las que "no importa" si la salida está en ALTO o en BAJO. Esto se ilustra en la tabla de verdad de la figura 4-18(a).

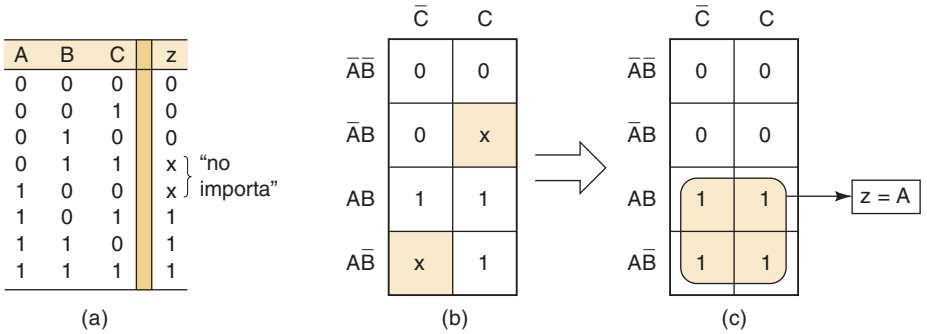


FIGURA 4-18 Las condiciones "No importa" deben cambiarse por 0 o 1 para producir un agrupamiento del mapa K que genere la expresión más simple.

Aquí la salida z no se especifica como 0 o 1 para las condiciones $A, B, C = 1, 0, 0$ y $A, B, C = 0, 1, 1$. En vez de ello se muestra una x para estas condiciones. La x representa la **condición de "no importa"**. Este tipo de condición puede surgir debido a varias razones; es la más común que en algunas situaciones nunca podrán ocurrir ciertas combinaciones de entradas, por lo que no hay una salida especificada para estas condiciones.

Un diseñador de circuitos tiene la libertad de hacer que la salida para cualquier condición de "no importa" sea un 0 o un 1 para producir la expresión de salida más simple. Por ejemplo, el mapa K para esta tabla de verdad se muestra en la figura 4-18(b), con una x en las casillas $A\overline{B}\,\overline{C}$ y $\overline{A}BC$. Aquí la mejor opción para el diseñador sería cambiar la x de la casilla $A\overline{B}\,\overline{C}$ por un 1 y la x de la casilla $\overline{A}BC$ por un 0, ya que esto produciría un cuádruple que puede agruparse para producir $z = A$, como se muestra en la figura 4-18(c).

Siempre que ocurran condiciones de "no importa", debemos decidir cuál x se va a cambiar por 0 y cuál por 1 para producir el mejor agrupamiento del mapa K (es decir, el grupo más grande que resulta en la expresión más simple). Esta decisión no siempre es fácil. Varios de los problemas al final del capítulo le ayudarán a aumentar su experiencia para tratar los casos de "no importa". He aquí otro ejemplo.

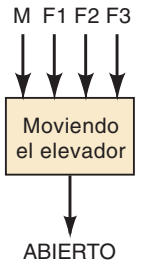
EJEMPLO 4-15

Vamos a diseñar un circuito lógico que controla la puerta de un elevador en un edificio de tres pisos. El circuito de la figura 4-19(a) tiene cuatro entradas. M es una señal lógica que indica cuando se mueve el elevador ($M = 1$) o cuando está detenido ($M = 0$). $F1$, $F2$ y $F3$ son señales indicadoras de cada piso que, por lo general, están en BAJO, y cambian a ALTO sólo cuando el elevador está posicionado en ese piso. Por ejemplo, cuando el elevador está alineado con el segundo piso, $F2 = 1$ y $F1 = F3 = 0$. La salida del circuito es la señal *ABIERTO* que, por lo general, está en BAJO y cambia a ALTO cuando se va a abrir la puerta del elevador.

Podemos llenar la tabla de verdad para la salida *ABIERTO* [Figura 4-19(b)] de la siguiente manera:

1. Como el elevador no puede alinearse con más de un piso a la vez, sólo una de las entradas de los pisos puede estar en ALTO en un momento dado. Esto significa que todos aquellos casos en la tabla de verdad en los que más de una entrada de piso esté en 1 serán condiciones de "no importa". Podemos colocar una x en la

FIGURA 4-19 Ejemplo 4-15.



(a)

M	F1	F2	F3	ABIERTO
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	X
0	1	0	0	1
0	1	0	1	X
0	1	1	0	X
0	1	1	1	X
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	X
1	1	0	0	0
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

(b)

	$\overline{F2}\,\overline{F3}$	$\overline{F2}F3$	$F2F3$	$F2\overline{F3}$
$\overline{M}\,\overline{F1}$	0	1	X	1
$\overline{M}F1$	1	X	X	X
$M F1$	0	X	X	X
$M\overline{F1}$	0	0	X	0

(c)

	$\overline{F2}\,\overline{F3}$	$\overline{F2}F3$	$F2F3$	$F2\overline{F3}$
$\overline{M}\,\overline{F1}$	0	1	1	1
$\overline{M}F1$	1	1	1	1
$M F1$	0	0	0	0
$M\overline{F1}$	0	0	0	0

$\text{OPEN} = \overline{M}\,(F1 + F2 + F3)$

(d)

columna de la salida *ABIERTO* para los ocho casos en los que más de una entrada *F* es 1.

2. Si analizamos los otros ocho casos, cuando $M = 1$ el elevador se está moviendo, por lo que *ABIERTO* debe ser un 0 ya que no deseamos que se abra la puerta del elevador. Cuando $M = 0$ (el elevador está detenido) queremos que *ABIERTO* = 1 siempre y cuando una de las entradas de piso sea 1. Cuando $M = 0$ y todas las entradas de piso son 0, el elevador está detenido pero no está alineado en forma apropiada con ninguno de los pisos, por lo que queremos que *ABIERTO* = 0 para mantener la puerta cerrada.

Ahora la tabla de verdad está completa y podemos transferir su información al mapa K de la figura 4-19(c). El mapa sólo tiene tres 1s, pero ocho condiciones de "no importa". Al cambiar cuatro de estas casillas de "no importa" por 1s, podemos producir grupos de cuádruples que contengan los 1s originales [Figura 4-19(d)]. Esto es lo mejor que podemos hacer en cuanto a minimizar la expresión de salida. Verifique que los agrupamientos produzcan la expresión de salida que se muestra para *ABIERTO*.

Resumen

El proceso de mapa K tiene varias ventajas en comparación con el método algebraico. El mapeo K es un proceso más ordenado, con pasos bien definidos en comparación con el proceso de prueba y error que se utiliza algunas veces en la simplificación algebraica. Por lo general, el mapeo K requiere menos pasos, en especial para las expresiones que contienen muchos términos, y siempre produce una expresión mínima.

Sin embargo, algunos instructores prefieren el método algebraico debido a que requiere un profundo conocimiento del álgebra booleana y no es tan sólo un procedimiento mecánico. Cada método tiene sus ventajas y, aunque la mayoría de los diseñadores lógicos son adeptos en ambos, ser proficiente en uno de ellos es todo lo que se necesita para producir resultados aceptables.

Existen otras técnicas más complejas que utilizan los diseñadores para minimizar circuitos lógicos con más de cuatro entradas. Estas técnicas se adecuan en forma especial a los circuitos con grandes cantidades de entradas, en donde no puede considerarse el método algebraico ni el mapeo K. La mayoría de estas técnicas puede traducirse a un programa de computadora que realizará la minimización con base en los datos de entrada que suministre la tabla de verdad o la expresión sin simplificar.

PREGUNTAS DE REPASO

1. Utilice el mapeo K para obtener la expresión del ejemplo 4-7.
2. Utilice el mapeo K para obtener la expresión del ejemplo 4-8. Aquí se debe enfatizar la ventaja del mapeo K para expresiones que contengan muchos términos.
3. Obtenga la expresión del ejemplo 4-9, utilizando un mapa K.
4. ¿Qué es una condición de "no importa"?

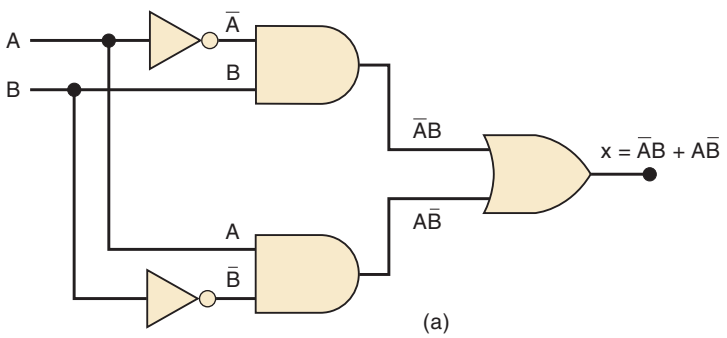
4-6 CIRCUITOS OR EXCLUSIVO Y NOR EXCLUSIVO

Dos de los circuitos lógicos especiales que se presentan con mucha frecuencia en los sistemas digitales son el *OR exclusivo* y el *NOR exclusivo*.

OR exclusivo

Considere el circuito lógico de la figura 4-20(a). La expresión de salida de este circuito es

$$x = \overline{A}B + A\overline{B}$$



A	B	x
0	0	0
0	1	1
1	0	1
1	1	0

Símbolos de compuerta XOR

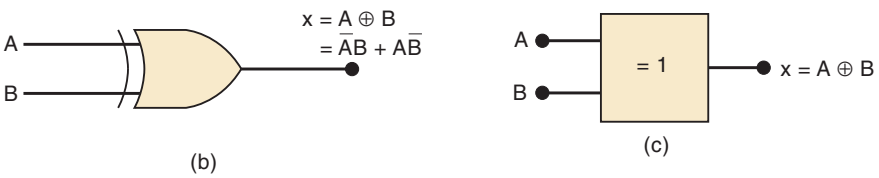


FIGURA 4-20
(a) Circuito OR exclusivo y su tabla de verdad; (b) símbolo tradicional de la compuerta XOR; (c) símbolo IEEE/ANSI para la compuerta XOR.

La tabla de verdad que acompaña a este circuito muestra que $x = 1$ para dos casos: $A = 0, B = 1$ (el término $\overline{A}B$) y $A = 1, B = 0$ (el término $A\overline{B}$). En otras palabras:

Este circuito produce una salida en ALTO siempre que las dos entradas se encuentran en los niveles opuestos.

Éste es el circuito **OR exclusivo**, que de aquí en adelante se abreviará como **XOR**.

Esta combinación específica de compuertas lógicas se produce con mucha frecuencia y es muy útil en ciertas aplicaciones. De hecho, al circuito XOR se le ha otorgado su propio símbolo, el cual se muestra en la figura 4-20(b). Se asume que este símbolo contiene toda la lógica dentro del circuito XOR y, por lo tanto, tiene la misma expresión lógica y la misma tabla de verdad. Por lo general, al circuito XOR se le conoce como *compuerta* XOR, y lo consideramos como otro tipo de compuerta lógica. En la figura 4-20(c) se muestra el símbolo IEEE/ANSI para una compuerta XOR. El símbolo de notación de dependencia (= 1) dentro del bloque indica que la salida será activa en ALTO *sólo* cuando una de las entradas esté en ALTO.

Una compuerta XOR sólo tiene *dos* entradas; no hay compuertas XOR de tres ni de cuatro entradas. Las dos entradas se combinan de manera que $x = \overline{A}B + A\overline{B}$. Una forma abreviada que se utiliza algunas veces para indicar la expresión de salida XOR es

$$x = A \oplus B$$

en donde el símbolo $\oplus$ representa la operación de la compuerta XOR.

A continuación se sintetizan las características de una compuerta XOR:

1. Sólo tiene dos entradas y su salida es

$$x = \overline{A}B + A\overline{B} = A \oplus B$$

2. Su salida está en *ALTO* sólo cuando las dos entradas se encuentran en niveles *distintos*.

Hay varios CIs disponibles que contienen compuertas XOR. Los que se listan a continuación son chips que contienen cuatro compuertas XOR.

74LS86	Chip con cuatro compuertas XOR (familia TTL)
74C86	Chip con cuatro compuertas XOR (familia CMOS)
74HC86	XOR (CMOS de alta velocidad)

NOR exclusivo

El circuito **NOR exclusivo** (que se abrevia como **XNOR**) opera en forma completamente opuesta al circuito XOR. La figura 4-21(a) muestra un circuito XNOR y su tabla de verdad correspondiente. La expresión de salida es

$$x = AB + \overline{A}\,\overline{B}$$

lo cual indica junto con la tabla de verdad que x será 1 para dos casos: $A = B = 1$ (el término AB) y $A = B = 0$ (el término $\overline{A}\,\overline{B}$). En otras palabras:

El circuito XNOR produce una salida en ALTO siempre que las dos entradas se encuentran en el mismo nivel.

De todo esto podemos deducir que la salida del circuito XNOR es el inverso exacto de la salida del circuito XOR. El símbolo tradicional para una compuerta

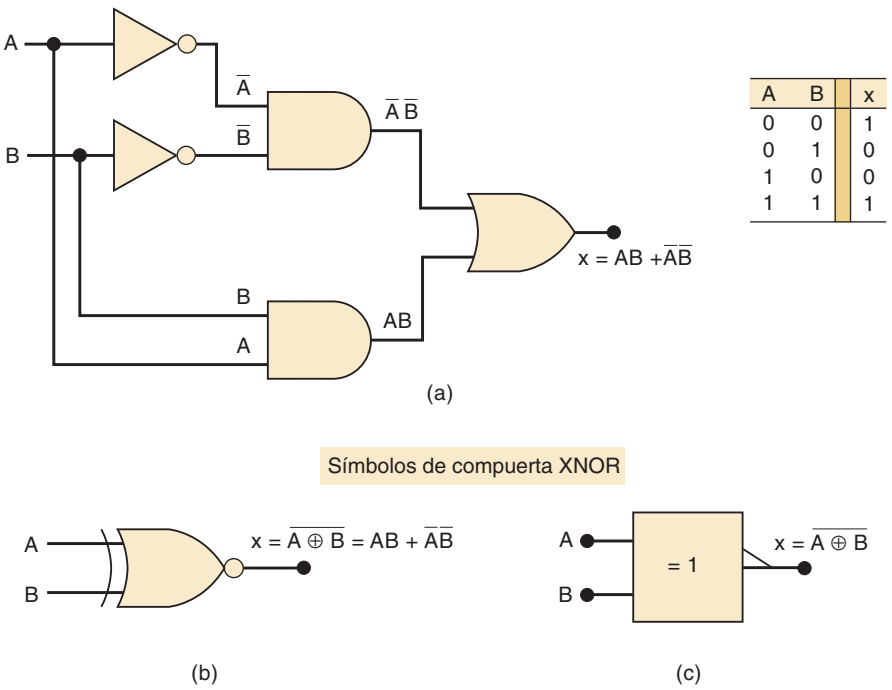


FIGURA 4-21
(a) Circuito NOR exclusivo; (b) símbolo tradicional para la compuerta XNOR; (c) símbolo IEEE/ANSI.

XNOR se obtiene con sólo agregar un pequeño círculo en la salida del símbolo XOR [figura 4-21(b)]. El símbolo IEEE/ANSI agrega el pequeño triángulo en la salida del símbolo XOR. Ambos símbolos indican una salida que cambia a su estado de activo en BAJO cuando *sólo una* de las entradas está en ALTO.

La compuerta XNOR también tiene *sólo dos* entradas, y las combina de manera que su salida sea

$$x = AB + \overline{A}\,\overline{B}$$

Una forma abreviada de indicar la expresión de salida de la compuerta XNOR es

$$x = \overline{A \oplus B}$$

la cual es el inverso de la operación XOR. La compuerta XNOR se puede sintetizar de la siguiente manera:

1. Sólo tiene dos entradas y su salida es

$$x = AB + \overline{A}\,\overline{B} = \overline{A \oplus B}$$

2. Su salida está en ALTO sólo cando las dos entradas se encuentran en el *mismo* nivel.

Hay varios CIs disponibles que contienen compuertas XNOR. Los que se listan a continuación son chips que contienen cuatro compuertas XNOR.

74LS266	Chip con cuatro compuertas XNOR (familia TTL)
74C266	Chip con cuatro compuertas XNOR (CMOS)
74HC266	Chip con cuatro compuertas XNOR (CMOS de alta velocidad)

Sin embargo, cada uno de estos chips consta de circuitos con salida especial que limita su uso a ciertos tipos especiales de aplicaciones. Es muy común que un diseñador lógico obtenga la función XNOR con sólo conectar la salida de una compuerta XOR a un INVERSOR.

EJEMPLO 4-16

Determine la forma de onda de salida para las formas de onda de entrada que se muestran en la figura 4-22.



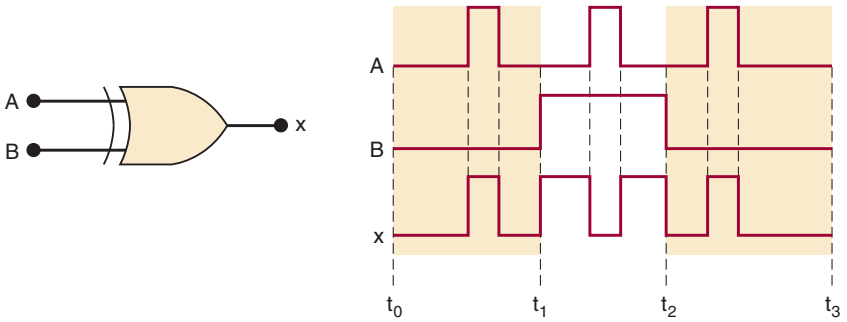


FIGURA 4-22 Ejemplo 4-16.

Solución

La forma de onda de salida se obtiene mediante el hecho de que la salida XOR estará en ALTO sólo cuando sus entradas se encuentren en distintos niveles. La forma de onda de salida resultante revela varios puntos interesantes:

1. La forma de onda de x concuerda con la forma de onda de entrada A durante los intervalos de tiempo en los que $B = 0$. Esto ocurre durante los intervalos de tiempo t_0 a t_1 y t_2 a t_3.
2. La forma de onda de x es el *inverso* de la forma de onda de entrada A durante los intervalos de tiempo en los que $B = 1$. Esto ocurre durante el intervalo t_1 a t_2.
3. Estas observaciones muestran que una compuerta XOR puede utilizarse como *INVERSOR CONTROLADO*; es decir, que una de sus entradas puede utilizarse para controlar si se va a invertir o no la señal de la otra entrada. Esta propiedad puede ser útil en ciertas aplicaciones.

EJEMPLO 4-17

La notación x_1x_0 representa un número binario de dos bits que puede tener cualquier valor (00, 01, 10 o 11); por ejemplo, cuando $x_1 = 1$ y $x_0 = 0$ el número binario es 10, y así sucesivamente. De manera similar, y_1y_0 representa otro número binario de dos bits. Diseñe un circuito lógico en el que utilice las entradas x_1, x_0, y_1 y y_0, y cuya salida esté en ALTO sólo cuando los dos números binarios x_1x_0 y y_1y_0 sean *iguales*.

Solución

El primer paso es construir una tabla de verdad para las 16 condiciones de entrada (tabla 4-4). La salida z debe estar en ALTO siempre que los valores de x_1x_0 concuerden con los valores de y_1y_0; esto es, siempre que $x_1 = y_1$ y $x_0 = y_0$. La tabla muestra que hay cuatro de esos casos. Ahora podríamos continuar con el procedimiento normal, que sería obtener una expresión de suma de productos para z, tratar de simplificarla y después implementar el resultado. No obstante, la naturaleza de este problema lo hace ideal para que se implemente mediante el uso de compuertas XNOR, y con un poco de pensamiento se producirá una solución simple con el mínimo esfuerzo. Consulte la figura 4-23; en este diagrama lógico, x_1 y y_1 se alimentan de

TABLA 4-4

x_1	x_0	y_1	y_0	z (Salida)
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

una compuerta XNOR y x_0 y y_0 se alimentan a otra compuerta XNOR. La salida de cada compuerta XNOR estará en ALTO sólo cuando sus entradas sean iguales. Por ende, para $x_0 = y_0$ y $x_1 = y_1$ las salidas de ambas compuertas XNOR estarán en ALTO. Ésta es la condición que estamos buscando, ya que significa que los dos números de dos bits son iguales. La salida de la compuerta AND estará en ALTO sólo para este caso, con lo cual se producirá el resultado deseado.

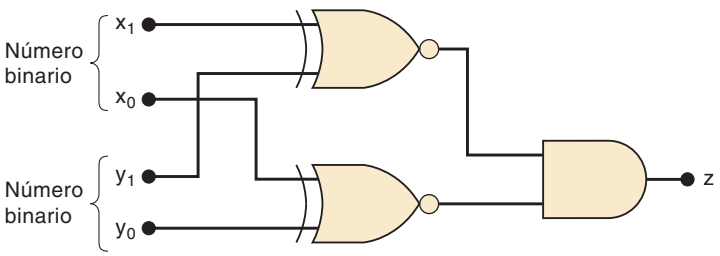


FIGURA 4-23 Circuito para detectar la igualdad de dos números binarios de dos bits.



EJEMPLO 4-18

Al simplificar la expresión para la salida de un circuito lógico combinacional, tal vez se encuentre con las operaciones XOR o XNOR cuando esté factorizando. A menudo esto nos lleva a utilizar compuertas XOR o XNOR en la implementación del circuito final. Para ilustrar lo anterior, simplifique el circuito de la figura 4-24(a).

Solución

La expresión sin simplificar para el circuito se obtiene como

$$z = ABCD + A\,\overline{B}\,\overline{C}\,D + \overline{A}\,\overline{D}$$

Podemos factorizar AD de los primeros dos términos:

$$z = AD(BC + \overline{B}\,\overline{C}) + \overline{A}\,\overline{D}$$

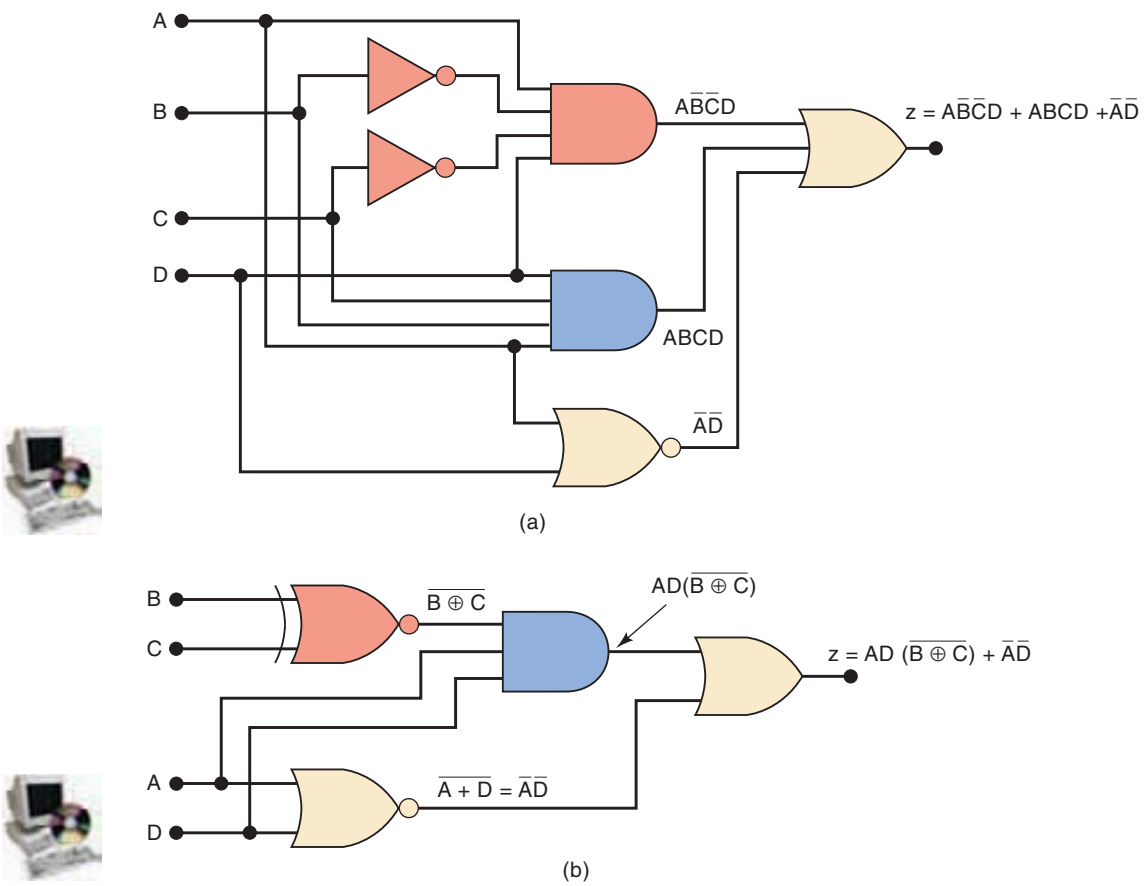


FIGURA 4-24 El ejemplo 4-18, que muestra cómo puede usarse una compuerta XNOR para simplificar la implementación de un circuito.

A primera instancia podría pensar que la expresión entre paréntesis puede sustituirse por un 1. Pero eso sería cierto sólo si la expresión fuera $BC + \overline{BC}$. Debería reconocer la operación entre paréntesis como la combinación XNOR de B y C. Este hecho puede usarse para volver a implementar el circuito como se muestra en la figura 4-24(b). Este circuito es mucho más simple que el original, ya que utiliza compuertas con menos entradas y se han eliminado dos INVERSORES.

PREGUNTAS DE REPASO

1. Use álgebra booleana para demostrar que la expresión de salida de la compuerta XNOR es el inverso exacto de la expresión de salida de la compuerta XOR.
2. ¿Cuál es la salida de una compuerta XNOR cuando se conectan a sus entradas una señal lógica y su inverso exacto?
3. Un diseñador lógico necesita un INVERSOR, y todo lo que hay disponible es una compuerta XOR de un chip 74HC86. ¿Necesita otro chip?

4-7 GENERADOR Y COMPROBADOR DE PARIDAD

En el capítulo 2 vimos que un transmisor puede adjuntar un bit de paridad a un conjunto de bits de datos antes de transmitirlos a un receptor. También vimos cómo esto permite al receptor detectar cualquier error de un solo bit que pueda haber ocurrido durante la transmisión. La figura 4-25 muestra un ejemplo de un tipo de

circuito lógico que se utiliza para la **generación de paridad** y la **comprobación de paridad**. Este ejemplo específico utiliza un grupo de cuatro bits como los datos que se van a transmitir, y utiliza un bit de paridad par. Puede adaptarse con facilidad para utilizar paridad impar y cualquier número de bits.

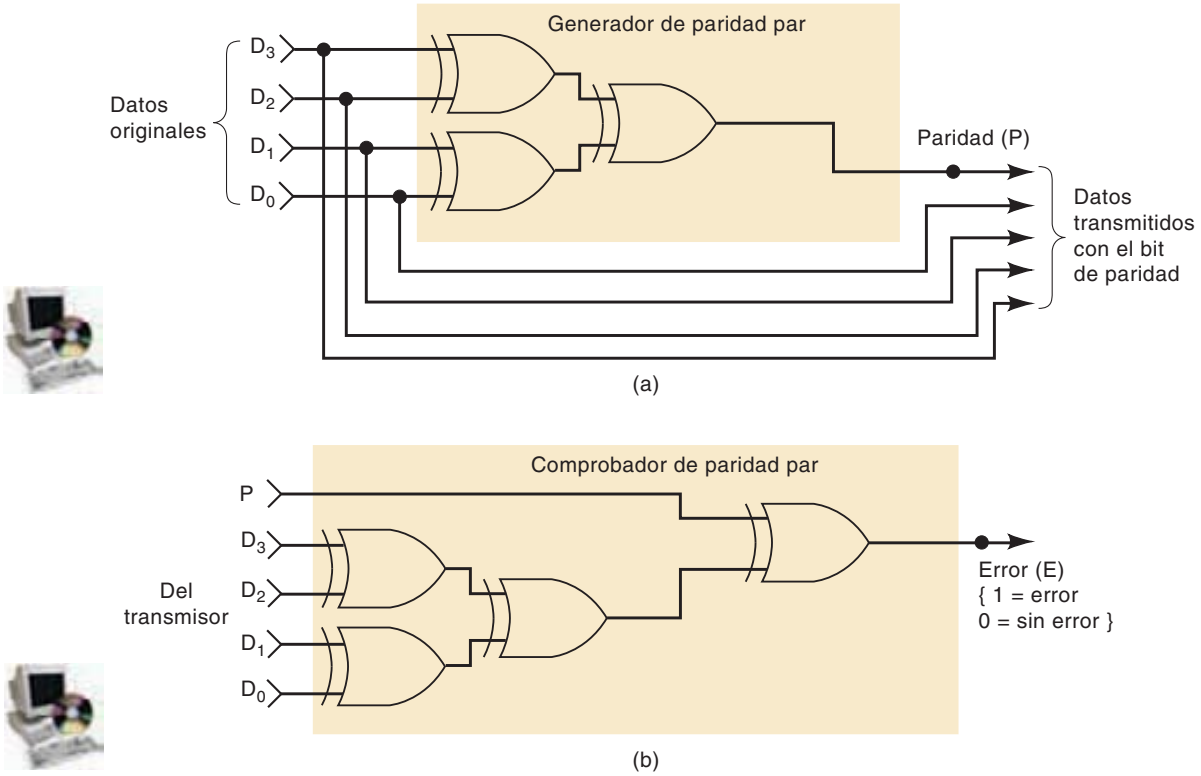


FIGURA 4-25 Compuertas XOR utilizadas para implementar (a) el generador de paridad y (b) el comprobador de paridad para un sistema con paridad par.

En la figura 4-25(a), el conjunto de datos que se van a transmitir se aplica al circuito generador de paridad, el cual produce el bit de paridad par *P* en su salida. Este bit de paridad se transmite al receptor junto con los bits de datos originales, formando un total de cinco bits. En la figura 4-25(b), estos cinco bits (datos + paridad) entran en el circuito comprobador de paridad del receptor, el cual produce una salida de error *E* que indica si ocurrió o no un error de un solo bit.

No debería sorprendernos demasiado que ambos circuitos empleen compuertas XOR, si consideramos que una sola compuerta XOR opera de manera que produce una salida de 1 si un número impar de sus entradas son 1, y una salida de 0 si un número par de sus entradas son 1.

EJEMPLO 4-19

Determine la salida del generador de paridad para cada uno de los siguientes conjuntos de datos de entrada, $D_3D_2D_1D_0$: (a) 0111; (b) 1001; (c) 0000; (d) 0100. Consulte la figura 4-25(a).

Solución

Para cada caso, aplique los niveles de datos a las entradas del generador de paridad y rastréelas a través de cada compuerta, hacia la salida *P*. Los resultados son: (a) 1; (b) 0; (c) 0; y (d) 1. Observe que *P* es 1 sólo cuando los datos originales contienen un número impar de 1s. Por ende, el número total de 1s que se envíen al receptor (datos + paridad) será par.

EJEMPLO 4-20

Determine la salida del comprobador de paridad [vea la figura 4-25(b)] para cada uno de los siguientes conjuntos de datos del transmisor:

	P	D_3	D_2	D_1	D_0
(a)	0	1	0	1	0
(b)	1	1	1	1	0
(c)	1	1	1	1	1
(d)	1	0	0	0	0

Solución

Para cada caso, aplique estos niveles a las entradas del comprobador de paridad y rastréelos a través de las compuertas hacia la salida *E*. Los resultados son: (a) 0; (b) 0; (c) 1; (d) 1. Observe que se produce un 1 en *E* sólo cuando aparece un número impar de 1s en las entradas que van hacia el comprobador de paridad. Esto indica que se ha producido un error, ya que se está utilizando la paridad par.

4-8 CIRCUITOS DE HABILITACIÓN/DESHABILITACIÓN

Cada una de las compuertas lógicas básicas puede utilizarse para controlar el paso de una señal lógica de entrada hacia la salida. Esto se describe en la figura 4-26, en donde se aplica una señal lógica *A* a una entrada de cada una de las compuertas lógicas básicas. La otra entrada de cada compuerta es la entrada de control *B*.

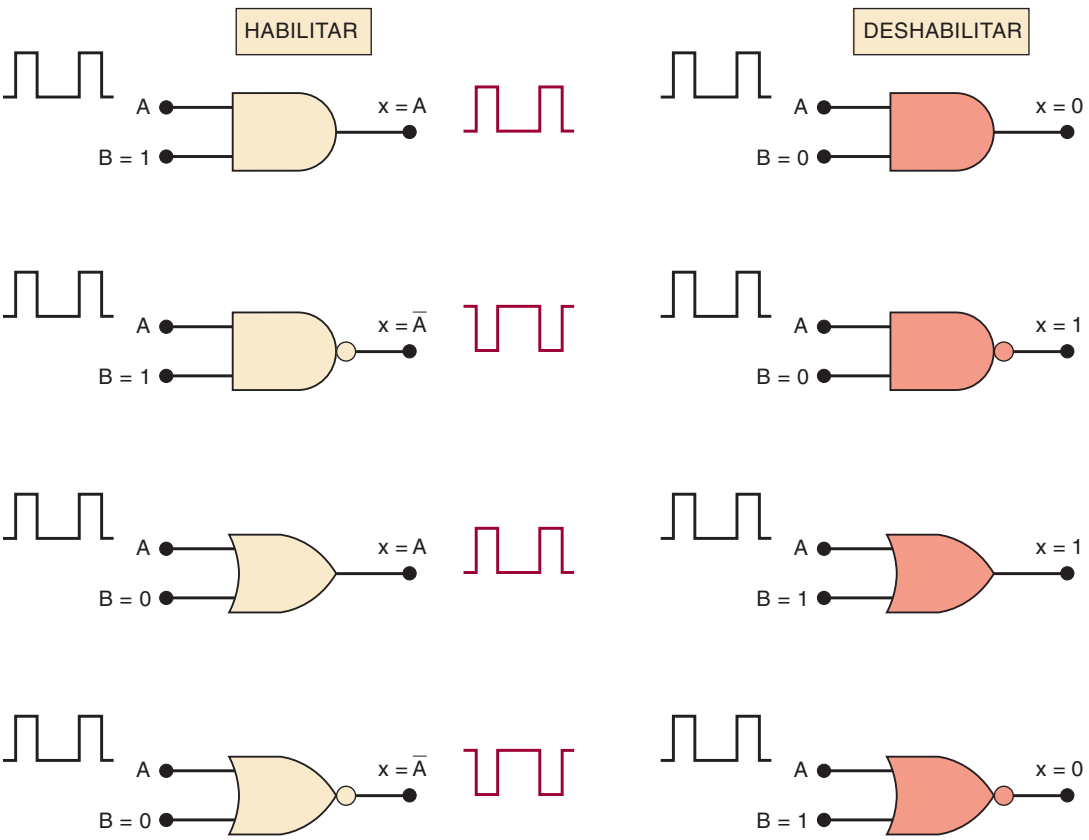


FIGURA 4-26 Las cuatro compuertas básicas pueden habilitar o deshabilitar el paso de una señal de entrada *A*, por medio del nivel lógico en la entrada de control *B*.

El nivel lógico en esta entrada de control determinará si la señal de entrada está **habilitada** para llegar a la salida o **deshabilitada** para que no pueda llegar a la salida. Esta acción de control explica por qué a estos circuitos se les empezó a llamar *compuertas*.

Si examina la figura 4-26 verá que cuando las compuertas no inversoras (AND, OR) están habilitadas, la salida sigue a la señal *A* de una manera exacta. En contraste, cuando las compuertas inversoras (NAND, NOR) están habilitadas, la salida es el inverso exacto de la señal *A*.

Observe también que en la figura las compuertas AND y NOR producen una salida constante en BAJO cuando se encuentran en la condición deshabilitada. En contraste, las compuertas NAND y OR producen una salida constante en ALTO cuando están deshabilitadas.

En el diseño de circuitos digitales se encontrará con muchas situaciones en las que se habilite o deshabilite el paso de una señal lógica, dependiendo de las condiciones presentes en una o más entradas de control. Los siguientes ejemplos muestran varias de estas situaciones.

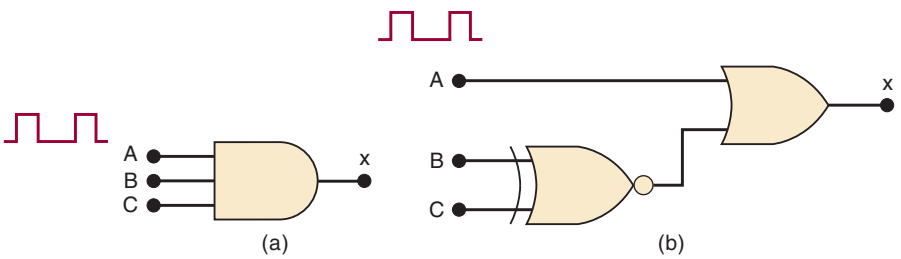
EJEMPLO 4-21

Diseñe un circuito lógico que permita que una señal pase a la salida sólo cuando las entradas de control *B* y *C* estén ambas en ALTO; en caso contrario, la salida deberá permanecer en BAJO.

Solución

Debe usarse una compuerta AND, ya que la señal debe pasarse sin invertir y la condición de salida de deshabilitación es un nivel BAJO. Como la condición de habilitación debe ocurrir sólo cuando $B = C = 1$ se debe usar una compuerta AND de tres entradas, como muestra la figura 4-27(a).

FIGURA 4-27
Ejemplos 4-21 y 4-22.



EJEMPLO 4-22

Diseñe un circuito lógico que permita que una señal pase hacia la salida sólo cuando una (pero no ambas) de sus entradas de control esté en ALTO; en caso contrario, la salida permanecerá en ALTO.

Solución

El resultado se dibuja en la figura 4-27(b). Se utiliza una compuerta OR porque queremos que la condición de deshabilitación de la salida sea un nivel ALTO, y no queremos invertir la señal. Las entradas de control *B* y *C* se combinan en una compuerta XNOR. Cuando *B* y *C* son distintas, la compuerta XNOR envía un nivel BAJO para habilitar la compuerta OR. Cuando *B* y *C* son iguales, la compuerta XNOR envía un nivel ALTO para deshabilitar la compuerta OR.

EJEMPLO 4-23

Diseñe un circuito lógico con la señal de entrada *A*, la entrada de control *B* y las salidas *X* y *Y*, que opere de la siguiente manera:

1. Cuando $B = 1$, la salida *X* seguirá a la entrada *A* y la salida *Y* será 0.
2. Cuando $B = 0$, la salida *X* será 0 y la salida *Y* seguirá a la entrada *A*.

Solución

Las dos salidas serán 0 cuando estén deshabilitadas y seguirán a la señal de entrada cuando estén habilitadas. Por lo tanto, debe utilizarse una compuerta AND para cada salida. Como X se debe habilitar cuando $B = 1$, su compuerta AND debe estar controlada por B, como muestra la figura 4-28. Como Y se debe habilitar cuando $B = 0$, su compuerta AND está controlada por $\overline{B}$. Al circuito de la figura 4-28 se le conoce como *circuito de dirección de pulso*, ya que dirige el pulso de entrada hacia una salida o la otra, dependiendo de B.

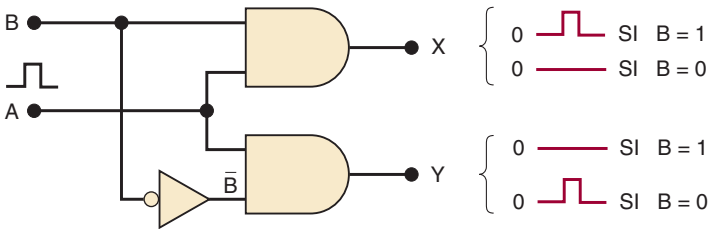


FIGURA 4-28 Ejemplo 4-23.



PREGUNTAS DE REPASO

1. Diseñe un circuito lógico con tres entradas A, B, C y una salida que cambie a BAJO sólo cuando A esté en ALTO mientras que B y C sean distintas.
2. ¿Cuáles compuertas lógicas producen una salida de 1 en el estado deshabilitado?
3. ¿Cuáles compuertas lógicas pasan el inverso de la señal de entrada cuando se habilitan?

4-9 CARACTERÍSTICAS BÁSICAS DE LOS CIS DIGITALES

Los CIs digitales son una colección de resistencias, diodos y transistores fabricados en una sola pieza de material semiconductor (por lo general, silicio), al cual se le conoce como *sustrato*, que por lo común se le denomina *chip*. El chip está encerrado en un encapsulado de plástico o cerámica protectora del cual salen terminales para conectar el CI con otros dispositivos. Uno de los tipos más comunes es el **encapsulado dual en línea (DIP)**, el cual se muestra en la figura 4-29(a), y se le llama así debido a que contiene dos filas paralelas de terminales. Estas terminales se numeran en sentido contrario al de las manecillas del reloj, viéndolas desde la parte superior del encapsulado con respecto a una muesca o punto de identificación en un extremo del encapsulado [vea la figura 4-29(b)]. El DIP que se muestra aquí es un encapsulado de 14 terminales que mide 0.75 pulg por 0.25 pulg; también se utilizan encapsulados de 16, 20, 24, 28, 40 y 64 terminales.

La figura 4-29(c) muestra que el chip de silicio es mucho más pequeño que el DIP; por lo general, es de 0.05 pulgadas cuadradas. El chip de silicio se conecta a las terminales del DIP mediante alambres muy finos [con diámetro de una milésima de pulgada (1 mil)].

El DIP es tal vez el encapsulado de CI digital más común que se encuentra en el equipo digital antiguo, pero actualmente se han hecho más populares otros tipos de encapsulados. El CI que se muestra en la figura 4-29(d) es sólo uno de los muchos encapsulados comunes en los circuitos digitales modernos. Este tipo específico utiliza puntas en forma de J que se encorvan por debajo del CI. En el capítulo 8 veremos otros tipos de encapsulados de CI.

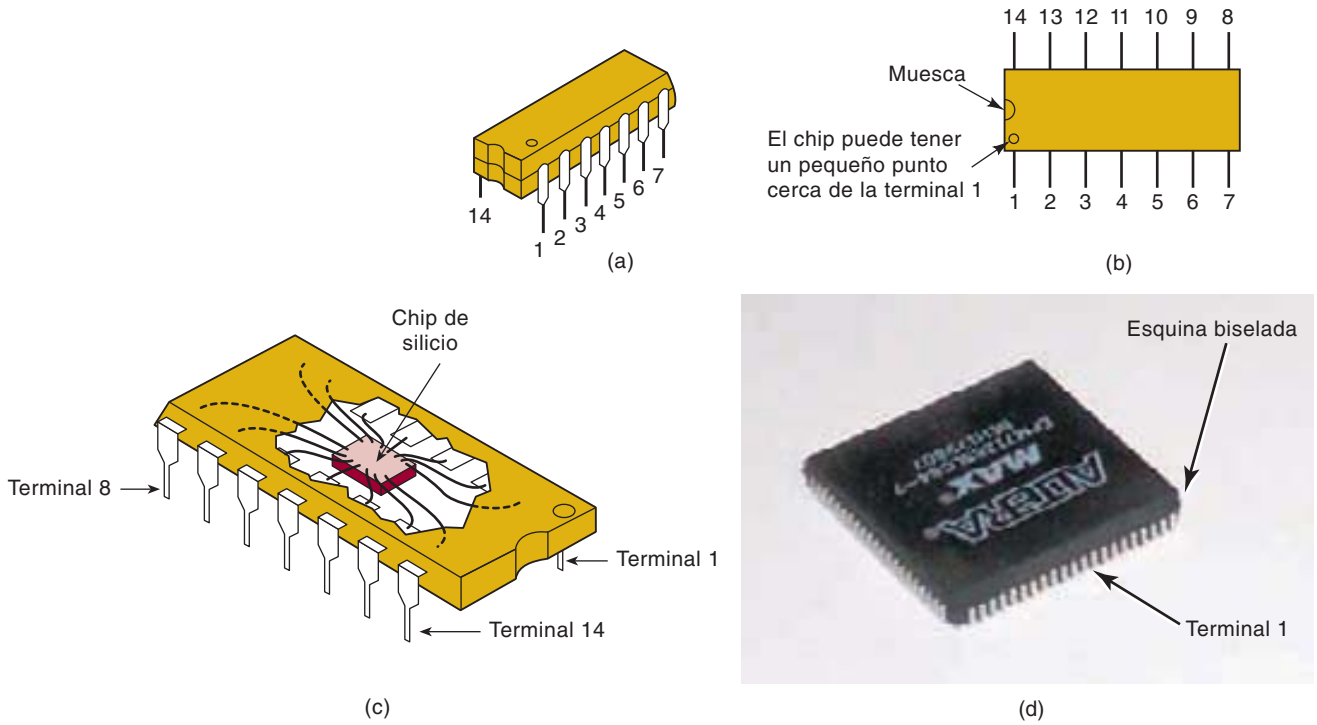


FIGURA 4-29 (a) Encapsulado dual en línea (DIP); (b) vista superior; (c) el chip de silicio es mucho más pequeño que el encapsulado protector; (d) encapsulado PLCC.

A menudo los CIs digitales se clasifican de acuerdo con la complejidad de sus circuitos, con base en el número de compuertas lógicas equivalentes en el sustrato. En la actualidad existen seis niveles de complejidad que, por lo común, se definen como se muestra en la tabla 4-5.

TABLA 4-5

Complejidad	Compuertas por chip
Integración a pequeña escala (SSI)	Menos de 12
Integración a mediana escala (MSI)	De 12 a 99
Integración a gran escala (LSI)	De 100 a 9999
Integración a muy grande escala (VLSI)	De 10,000 a 99,999
Integración a ultra gran escala (ULSI)	De 100,000 a 999,999
Integración a giga escala (GSI)	1,000,000 o más

Todos los CIs específicos a los que se hizo referencia en el capítulo 3 y en este capítulo son chips **SSI** con un número pequeño de compuertas. En los sistemas digitales modernos, los dispositivos con integración a mediana escala (**MSI**) e integración a gran escala (**LSI, VLSI, ULSI, GSI**) realizan la mayor parte de las funciones que alguna vez requirieron de varios tableros de circuitos llenos de dispositivos SSI. No obstante, los chips SSI se siguen utilizando como "interfaz" o "pegamento" entre estos chips más complejos. Los CIs de pequeña escala facilitan el aprendizaje de los fundamentos de los sistemas digitales. En consecuencia, muchos cursos basados en laboratorio utilizan estos CIs para construir y probar pequeños proyectos.

Ahora el mundo industrial de la electrónica digital se ha concentrado en los dispositivos lógicos programables (PLDs) para implementar un sistema digital de cualquier tamaño. Algunos PLDs simples están disponibles en encapsulados DIP,

pero los dispositivos lógicos programables más complejos requieren muchas más terminales de las que están disponibles en los DIPs. Los circuitos integrados más grandes que tal vez necesiten extraerse de un circuito para sustituirlos, por lo general, se fabrican en un encapsulado de soporte de chip de plástico con contactos (PLCC). La figura 4-29(d) muestra el EPM 7128SLC84 de Altera en un encapsulado PLCC, el cual es un PLD muy popular que se utiliza en muchos cursos de laboratorio. Las características clave de este chip son más terminales, un tamaño más compacto, y terminales alrededor de todo su perímetro. Observe que la terminal 1 no está "en la esquina" como en el DIP, sino en medio de la parte superior del encapsulado.

CIs bipolares y unipolares

Los CIs digitales también pueden clasificarse de acuerdo con el tipo principal de componente electrónico utilizado en sus circuitos. Los *CIs bipolares* se fabrican mediante el uso del transistor de unión bipolar (NPN y PNP) como elemento principal del circuito. Los *CIs unipolares* utilizan el transistor unipolar de efecto de campo (MOSFETs de canal P y N) como su elemento principal.

La familia **lógica de transistor/transistor (TTL)** ha sido la familia principal de CIs digitales bipolares durante más de 30 años. La serie 74 estándar fue la primera serie de CIs TTL, pero ya no se utiliza en diseños nuevos debido a que fue sustituida por varias series TTL de mejor desempeño, aunque su arreglo básico de circuitos forma la base para todos los CIs de las series TTL. Este arreglo de circuitos se muestra en la figura 4-30(a) para el INVERSOR TTL estándar. Observe que el circuito contiene varios transistores bipolares como elementos principales del circuito.

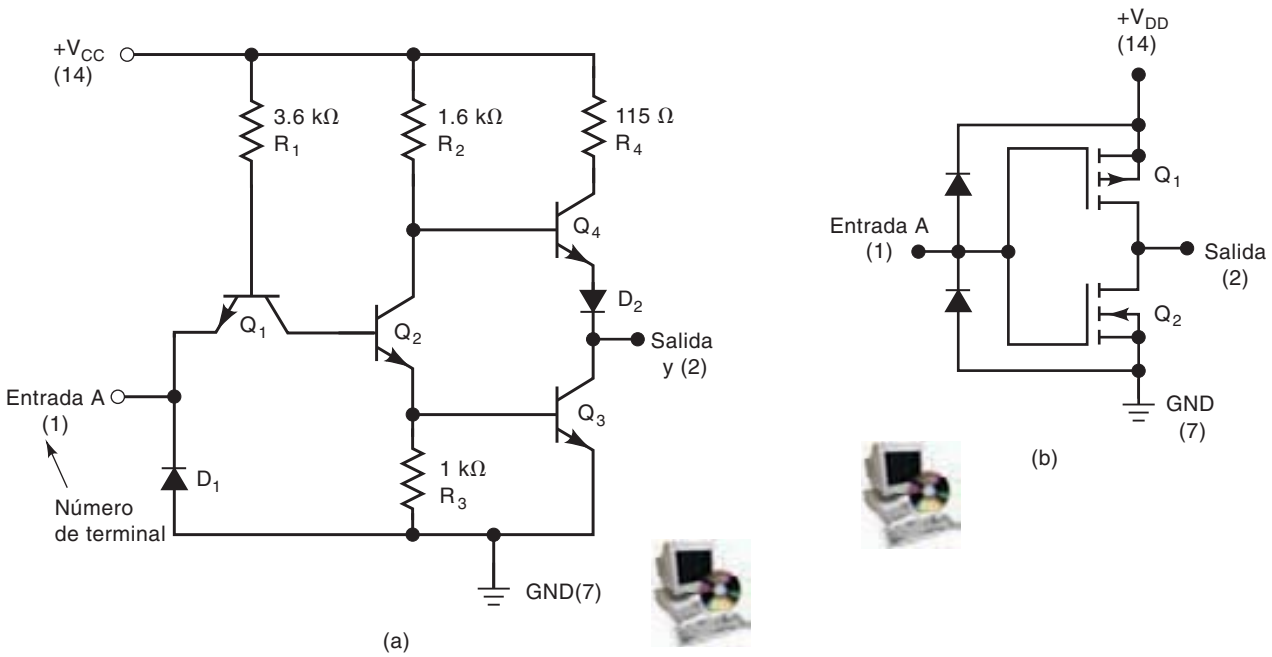


FIGURA 4-30 (a) Circuito INVERSOR TTL; (b) Circuito INVERSOR CMOS. Los números de terminales se muestran entre paréntesis.

La familia TTL fue la familia líder de CIs en las categorías SSI y MSI hasta hace aproximadamente 12 años. Desde entonces la familia de **semiconductor metal-óxido complementario (CMOS)** ha desplazado gradualmente a la familia TTL de esa posición. La familia CMOS pertenece a la clase de CIs digitales unipolares, ya que utiliza MOSFETs de canal P y N como elementos principales del circuito. La figura 4-30(b) es un circuito INVERSOR CMOS estándar. Si comparamos los circuitos TTL y CMOS de la figura 4-30, es evidente que la versión CMOS utiliza menos componentes. Ésta es una de las principales ventajas de CMOS en comparación con TTL.

Debido a la simpleza y tamaño reducido, así como de otros atributos superiores de los circuitos CMOS, los CIs modernos a gran escala se fabrican utilizando en su mayor parte la tecnología CMOS. Los cursos de laboratorio que utilizan dispositivos SSI y MSI a menudo utilizan TTL debido a su resistencia, aunque algunos utilizan CMOS también. En el capítulo 8 veremos un estudio detallado de los circuitos y las características de TTL y CMOS. Por ahora necesitamos conocer sólo algunas de sus características básicas para poder hablar sobre el diagnóstico de fallas en los circuitos combinacionales simples.

Familia TTL

La familia lógica TTL consiste en realidad de varias subfamilias o series. La tabla 4-6 lista el nombre de cada serie TTL, junto con la designación de prefijo que se utiliza para identificar los distintos CIs como parte de esa serie. Por ejemplo, los CIs que forman parte de la serie TTL estándar tienen un número de identificación que comienza con 74. Los CIs 7402, 7438 y 74123 son de esta serie. De igual forma, los CIs que forman parte de la serie TTL Schottky de bajo consumo de energía tienen un número de identificación que comienza con 74LS. Los CIs 74LS02, 74LS38 y 74LS123 son ejemplos de dispositivos de ésta.

TABLA 4-6 Varias series dentro de la familia lógica TTL.

Serie TTL	Prefijo	CI de ejemplo
TTL estándar	74	7404 (INVERSOR hex)
TTL Schottky	74S	74S04 (INVERSOR hex)
TTL Schottky de bajo consumo de energía	74LS	74LS04 (INVERSOR hex)
TTL Schottky avanzado	74AS	74AS04 (INVERSOR hex)
TTL Schottky avanzado de bajo consumo de energía	74ALS	74ALS04 (INVERSOR hex)

Las principales diferencias en las diversas series TTL tienen que ver con sus características eléctricas tales como la disipación de potencia y la velocidad de conmutación. No difieren en cuanto a la distribución de las terminales o las operaciones lógicas realizadas por los circuitos en el chip. Por ejemplo, los CIs 7404, 74S04, 74LS04, 74AS04 y 74ALS04 son circuitos de INVERSORES, cada uno de los cuales contiene *seis* INVERSORES en un solo chip.

Familia CMOS

Actualmente existen varias series CMOS, algunas de ellas se listan en la tabla 4-7. La serie 4000 es la más antigua. Esta serie contiene muchas de las mismas funciones lógicas que la familia TTL, pero no se diseñó para ser *compatible con las terminales* de los dispositivos TTL. Por ejemplo, el chip NOR cuádruple 4001 contiene cuatro compuertas NOR de dos entradas, al igual que el chip 7402 TTL, pero las entradas y las salidas de las compuertas en el chip CMOS no tienen los mismos números de terminales que las señales correspondientes en el chip TTL.

Las series CMOS 74C, 74HC, 74HCT, 74AC y 74ACT son más recientes. Las primeras tres son compatibles con las terminales de los dispositivos TTL con numeraciones correspondientes. Por ejemplo, los dispositivos 74C02, 74HC02 y 74HCT02 tienen la misma distribución de terminales que el 7402, 74LS02, y así en lo sucesivo. Las series 74HC y 74HCT operan a una mayor velocidad que los dispositivos 74C. La serie 74HCT está diseñada para ser *eléctricamente compatible* con los dispositivos TTL; esto es, un circuito integrado 74HCT puede conectarse en forma directa con los dispositivos TTL sin necesidad de circuitos que actúen como interfaz. Las series 74AC y 74ACT son CIs de desempeño avanzado. Ninguno es compatible con las terminales de los circuitos TTL. Los dispositivos 74ACT son eléctricamente compatibles con los circuitos TTL. En el capítulo 8 exploraremos las diversas series TTL y CMOS con más detalle.

TABLA 4-7 Varias series dentro de la familia lógica CMOS.

Serie CMOS	Prefijo	CI de ejemplo
CMOS de compuerta de metal	40	4001 (cuatro compuertas NOR)
Compuerta de metal, compatible con las terminales de TTL	74C	74C02 (cuatro compuertas NOR)
Compuerta de silicio de alta velocidad, compatible con las terminales TTL	74HC	74HC02 (cuatro compuertas NOR)
Compuerta de silicio de alta velocidad, compatible con las terminales y eléctricamente compatible con la familia TTL	74HCT	74HCT02 (cuatro compuertas NOR)
CMOS con desempeño avanzado, no es compatible con las terminales ni es eléctricamente compatible con la familia TTL	74AC	74AC02 (cuatro compuertas NOR)
CMOS con desempeño avanzado, no es compatible con las terminales de TTL, pero es eléctricamente compatible	74ACT	74ACT02 (cuatro compuertas NOR)

Alimentación y tierra

Para usar CIs digitales es necesario realizar las conexiones apropiadas a las terminales del CI. Las conexiones más importantes son: *alimentación de corriente directa(cd)* y *tierra*. Estas conexiones son requeridas para que los circuitos en el chip operen en forma correcta. En la figura 4-30 podemos ver que tanto el circuito TTL como CMOS tienen un voltaje de alimentación de *cd* conectado a una de sus terminales, y tierra en la otra. La terminal de voltaje de alimentación se etiqueta como V_{CC} para el circuito TTL y como V_{DD} para el circuito CMOS. Muchos de los circuitos integrados CMOS más recientes que están diseñados para ser compatibles con los circuitos integrados TTL también utilizan la designación V_{CC} en dicha terminal.

Si el CI no se conecta al voltaje de alimentación o a tierra, las compuertas lógicas en el chip no responderán en forma apropiada a las entradas lógicas y las compuertas no producirán los niveles lógicos de salida esperados.

Intervalos de voltaje de niveles lógicos

Para los dispositivos TTL, el valor nominal de V_{CC} es +5 V. Para los circuitos integrados CMOS, V_{DD} puede variar de +3 a +18 V, aunque el valor más común es +5 V cuando los circuitos CMOS se utilizan en la misma placa con circuitos integrados TTL.

Para los dispositivos TTL estándar, los intervalos de voltaje de entrada aceptables para los niveles de 0 lógico y de 1 lógico se definen como muestra la figura 4-31(a). Un 0 lógico es cualquier voltaje en el intervalo de 0 a 0.8 V; un 1 lógico es cualquier voltaje de 2 a 5 V. Los voltajes que no se encuentran en ninguno de estos intervalos se consideran como **indeterminados** y no deben utilizarse como entradas para un dispositivo TTL. Los fabricantes de circuitos integrados no pueden garantizar la manera en que responderá un circuito TTL a los niveles de entrada que se encuentren en el intervalo indeterminado (entre 0.8 y 2.0 V).

La figura 4-31(b) muestra los intervalos de voltaje de entrada lógicos para los circuitos integrados CMOS que operan con $V_{DD} = +5$ V. Los voltajes entre 0 y 1.5 V se definen como un 0 lógico, y los voltajes desde 3.5 hasta 5 V se definen como un 1 lógico. El intervalo indeterminado comprende los voltajes entre 1.5 y 3.5 V.

Entradas desconectadas (flotantes)

¿Qué ocurre cuando la entrada a un circuito integrado digital se deja desconectada? Por lo general, a una entrada desconectada se le conoce como entrada **flotante**. La respuesta a esta pregunta será distinta para TTL y para CMOS.

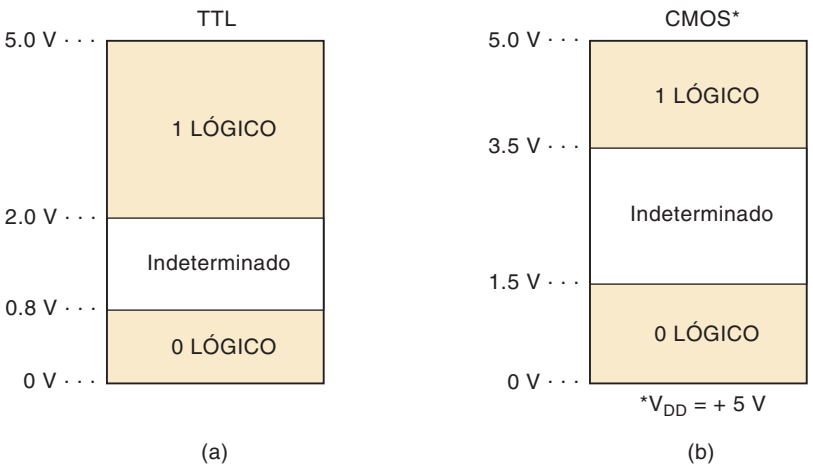


FIGURA 4-31
Intervalos de voltajes de entrada de los niveles lógicos para los CIs digitales (a) TTL y (b) CMOS.

Una entrada TTL flotante actúa justo igual que un 1 lógico. En otras palabras, el CI responderá como si se le hubiera aplicado un nivel lógico ALTO. Esta característica se utiliza a menudo cuando se prueba un circuito TTL. Un técnico descuidado podría dejar ciertas entradas desconectadas en vez de conectarlas a un nivel lógico ALTO. Aunque esto es "lógicamente" correcto no es una práctica recomendada, en especial cuando se trata de diseños de circuitos finales, ya que la entrada TTL flotante es muy susceptible de recoger señales de ruido que podrían afectar en forma adversa la operación del dispositivo.

En algunas compuertas TTL, una entrada flotante puede indicar un nivel de corriente directa de entre 1.4 y 1.8 V, si se comprueba con un voltímetro o un osciloscopio. Aun y cuando estos valores se encuentran en el intervalo indeterminado para TTL, producirá la misma respuesta que un 1 lógico. Es importante tener en cuenta esta característica de una entrada TTL flotante al diagnosticar fallas en circuitos TTL ya que puede ser de gran ayuda.

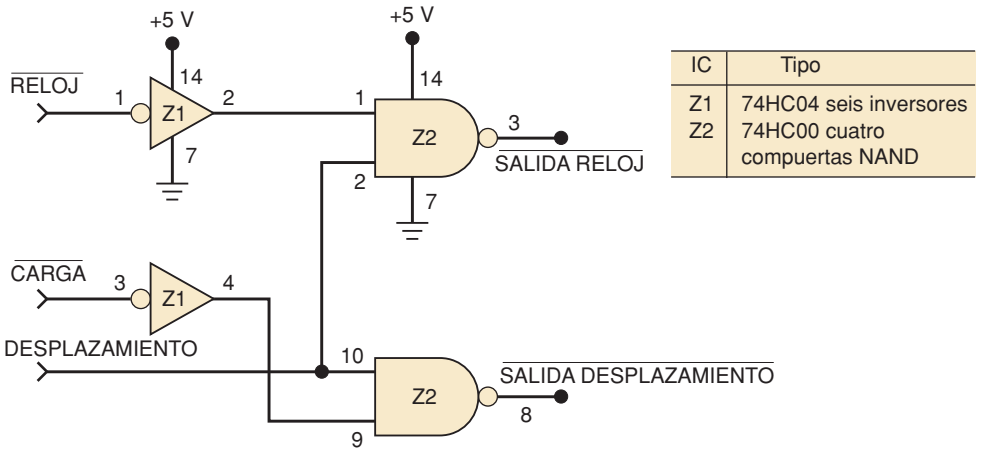
Si una entrada CMOS se deja flotante, pueden producirse resultados desastrosos. El CI podría sobrecalentarse y hasta dañarse. Por esta razón, todas las entradas de un circuito integrado CMOS deben conectarse a un nivel ALTO o BAJO, o a la salida de otro CI. Una entrada CMOS flotante no se medirá como un voltaje específico de corriente directa, sino que fluctuará en forma aleatoria a medida que recoja ruido. Por ende, no actúa como 1 ni como 0 lógico y su efecto sobre la salida es impredecible. Algunas veces la salida oscilará como resultado del ruido que recoja la salida flotante.

Muchos de los CIs CMOS más complejos tienen circuitos integrados en las entradas, los cuales reducen la probabilidad de cualquier reacción destructiva para una entrada abierta. Con estos circuitos, a la hora de experimentar con ellos, no es necesario aterrizar cada una de las terminales que no se utilicen en estos CIs grandes. No obstante, es una buena práctica conectar las entradas no utilizadas al nivel ALTO o BAJO (lo que sea apropiado) en la implementación del circuito final.

Diagramas de conexiones de circuitos lógicos

Un diagrama de conexiones muestra *todas* las conexiones eléctricas, números de terminal, números de CI, valores de los componentes, nombres de las señales y voltajes de alimentación. La figura 4-32 muestra un diagrama de conexiones común para un circuito lógico simple. Examínelo con cuidado y observe los siguientes puntos importantes:

1. El circuito utiliza compuertas lógicas de dos CIs distintos. Los dos INVERSORES forman parte de un chip 74HC04, designado como Z1. El 74HC04 contiene seis INVERSORES; dos de ellos se utilizan en este circuito y cada uno está etiquetado como parte del chip Z1. De manera similar, las dos compuertas NAND son parte de un chip 74HC00 que contiene cuatro compuertas NAND. Todas



IC	Tipo
Z1	74HC04 seis inversores
Z2	74HC00 cuatro compuertas NAND

FIGURA 4-32
Diagrama de conexiones de un circuito lógico común.

las compuertas en este chip están designadas con la etiqueta Z2. Al numerar cada compuerta como Z1, Z2, Z3 etcétera, podemos llevar el registro de cuál compuerta forma parte de cada chip. Esto es muy útil en los circuitos más complejos que contienen muchos CIs con varias compuertas por chip.

2. El número de terminal de las entradas y salidas de cada compuerta se indica en el diagrama. Estos números de terminales y las etiquetas de los CIs se utilizan para facilitar la referencia a cualquier punto en el circuito. Por ejemplo, la terminal 2 de Z1 se refiere a la terminal de salida del INVERSOR superior. De manera similar, podemos decir que la terminal 4 de Z1 está conectada a la terminal 9 de Z2.
3. Las conexiones de alimentación y de tierra para cada CI (no para cada compuerta) se muestran en el diagrama. Por ejemplo, la terminal 14 de Z1 está conectada a +5 V, y la terminal 7 de Z1 está conectada a tierra. Estas conexiones proveen de energía a los seis INVERSORES que forman parte de Z1.
4. Para el circuito mostrado en la figura 4-32, las señales que son entradas están a la izquierda. Las señales que son salidas están a la derecha. La barra sobre el nombre de la señal indica que ésta es activa en BAJO. Las burbujas se colocan también en los símbolos del diagrama para indicar el estado activo en BAJO. En este caso, es obvio que cada señal es un solo bit.
5. Las señales se definen en forma gráfica en la figura 4-32 como entradas y salidas, y la relación entre ellas (la operación del circuito) se describe en forma gráfica mediante el uso de símbolos lógicos interconectados.

Por lo general, los fabricantes de equipo electrónico suministran diagramas esquemáticos detallados que utilizan un formato similar al de la figura 4-32. Estos diagramas de conexiones son virtualmente necesarios cuando se diagnostican fallas en un circuito defectuoso. Hemos optado por identificar los CIs individuales como Z1, Z2, Z3, etcétera. Otras designaciones que se utilizan con frecuencia son IC1, IC2, IC3, … o U1, U2, U3, ….

Para dibujar circuitos lógicos pueden utilizarse computadoras personales con software para diagramas esquemáticos. Las aplicaciones computacionales que pueden interpretar estos símbolos gráficos y las conexiones de las señales pueden traducirlos en relaciones lógicas, a las cuales por lo común se les conoce como herramientas de captura de diagramas esquemáticos. El sistema de desarrollo MAX +PLUS de Altera para lógica programable permite al usuario introducir archivos de diseño gráfico (.gdf) mediante el uso de técnicas de captura de diagramas esquemáticos. Por ende, el diseño del circuito es tan sencillo como dibujar el diagrama esquemático en la pantalla de la computadora. Observe que en la figura 4-33 no hay números de terminales ni designaciones de chips en los símbolos lógicos. Los circuitos no se implementarán mediante el uso de chips SSI o MSI, sino que la funcionalidad de la lógica equivalente se “programará” en un PLD. Más adelante en este capítulo explicaremos esto con más detalle.

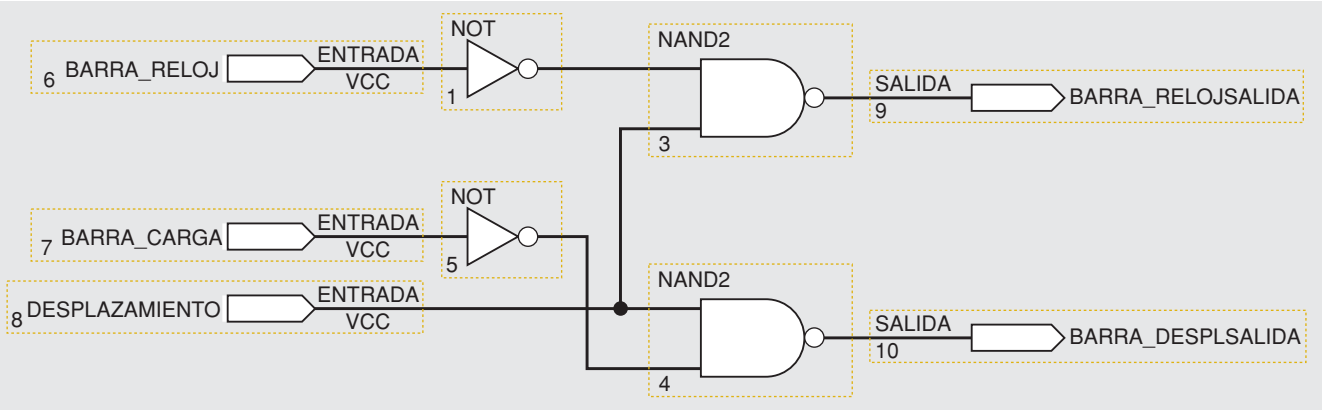


FIGURA 4-33 Diagrama lógico que utiliza captura de diagramas esquemáticos.

PREGUNTAS DE REPASO

1. ¿Cuál es el tipo más común de encapsulado de CI digital?
2. Nombre las seis categorías comunes de CIs digitales, de acuerdo con su complejidad.
3. *Verdadero o falso:* un chip 74S74 contiene la misma lógica y distribución de terminales que el 74LS74.
4. *Verdadero o falso:* un chip 74HC74 contiene la misma lógica y distribución de terminales que el 74AS74.
5. ¿Cuál serie CMOS no es compatible con las terminales de TTL?
6. ¿Cuál es el intervalo de voltaje de entrada aceptable de un 0 lógico para TTL? ¿Para un 1 lógico?
7. Repita la pregunta 6 para un CMOS que opera a $V_{DD} = 5$ V.
8. ¿Cómo responde un circuito integrado TTL a una entrada flotante?
9. ¿Cómo responde un circuito integrado CMOS a una entrada flotante?
10. ¿Cuál serie CMOS puede conectarse en forma directa a un TTL sin necesidad de circuitos de interfaz?
11. ¿Cuál es el propósito de los números de terminales en el diagrama de conexiones de un circuito lógico?
12. ¿Cuáles son las similitudes clave de los archivos de diseño gráfico que se utilizan para los diagramas de conexiones de circuitos lógicos tradicionales y de lógica programable?

4-10 DIAGNÓSTICO DE FALLAS DE SISTEMAS DIGITALES

Existen tres pasos básicos para corregir un circuito o sistema digital que tenga una falla:

1. *Detección de fallas.* Observe la operación del circuito/sistema y compárela con la operación correcta esperada.
2. *Aislamiento de fallas.* Realice pruebas y mediciones para aislar la falla.
3. *Corrección de fallas.* Sustituya el componente defectuoso, repare la conexión defectuosa, elimine el corto, o realice la acción pertinente.

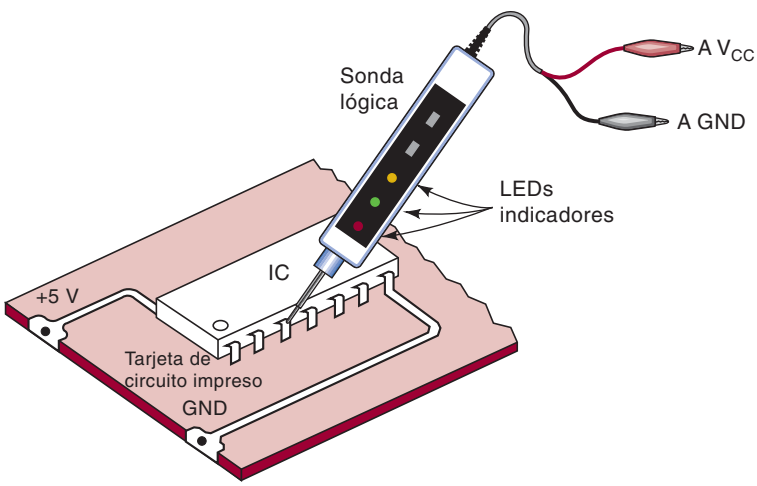
Aunque estos pasos pueden parecer bastante obvios y simples, el procedimiento real de diagnóstico de fallas que se siga dependerá en gran parte del tipo y la com-

plejidad del circuito, y de los tipos de herramientas de diagnóstico de fallas y de la documentación disponible.

Las buenas técnicas de diagnóstico de fallas sólo pueden aprenderse en un entorno de laboratorio, por medio de la experimentación y el diagnóstico de fallas real en circuitos y sistemas defectuosos. No existe en absoluto una forma de convertirse en un técnico de diagnóstico de fallas eficiente que tratar de diagnosticar fallas en todos los circuitos que sea posible; además, por más libros de texto que lea no podrá obtener de ellos ese tipo de experiencia. No obstante, podemos ayudarle a desarrollar las habilidades analíticas que forman la parte más esencial de una técnica eficiente de diagnóstico de fallas. Describiremos los tipos de fallas comunes en los sistemas compuestos en su mayor parte de CIs digitales y le indicaremos cómo reconocerlas. Después le presentaremos ejemplos prácticos triviales para ilustrar los procesos analíticos implicados en el diagnóstico de fallas. Además, habrá problemas al final del capítulo para que usted tenga la oportunidad de pasar por estos procesos analíticos para obtener sus propias conclusiones acerca de los circuitos digitales defectuosos.

Para los análisis y ejercicios sobre diagnóstico de fallas que realizaremos en este libro, supondremos que el técnico de diagnóstico de fallas tiene a su disposición las herramientas básicas para este propósito: *sonda lógica, osciloscopio* y *generador de pulsos lógico*. Desde luego que la herramienta de diagnóstico de fallas más importante y efectiva es el técnico, y ésa es la herramienta que esperamos desarrollar mediante la presentación de los principios y las técnicas de diagnóstico de fallas, ejemplos y problemas, tanto aquí como en los capítulos posteriores.

En las siguientes tres secciones sobre diagnóstico de fallas utilizaremos sólo nuestro cerebro y una **sonda lógica** tal como la que se muestra en la figura 4-34. La sonda lógica tiene una punta de metal afilada con la que se toca el punto específico que deseamos probar. En la figura 4-34 se muestra probando la terminal 3 de un CI. También puede tocar el trazo de una tarjeta de circuitos impresos, un alambre sin aislamiento, la terminal de un conector, la terminal de un componente discreto tal como un transistor o cualquier otro punto conductor en un circuito. El nivel lógico presente en la punta de la sonda se indicará mediante el estado de sus LEDs indicadores. En la tabla de la figura 4-34 se muestran las cuatro posibilidades. Un nivel lógico *indeterminado* no produce luz en los indicadores. Esto incluye la condición en la que la punta de la sonda toca un punto en un circuito que esté abierto o flotante; esto es, que no esté conectado a ninguna fuente de voltaje. Este tipo de sonda también cuenta con un LED amarillo para indicar la presencia de un tren de pulsos. Cualquier transición (de BAJO a ALTO o de ALTO a BAJO) provocará que el LED amarillo destelle durante una fracción de segundo y después se apague. Si las transiciones ocurren con frecuencia, el LED continuará destellando a una frecuencia



LEDs			
Rojo	Verde	Amarillo	Condición lógica
APAG	ENC	APAG	BAJO
ENC	APAG	APAG	ALTO
APAG	APAG	APAG	INDETERMINADO*
X	X	DESTELLANDO	PULSANDO

* Incluye condición abierta o flotante

FIGURA 4-34 Una sonda lógica se utiliza para monitorear la actividad de los niveles lógicos en la terminal de un CI o en cualquier punto accesible en un circuito lógico.

aproximada de 3 Hz. Mediante la observación de los LEDs rojo y verde, junto con el LED amarillo destellante, podrá saber si la señal está la mayor parte del tiempo en ALTO o en BAJO.

4-11 FALLAS INTERNAS EN LOS CIRCUITOS INTEGRADOS DIGITALES

Las fallas internas más comunes en los CIs digitales son:

1. Fallas en los circuitos internos.
2. Entradas o salidas cortocircuitadas a tierra o a V_{CC}.
3. Entradas o salidas sin conectar (circuito abierto).
4. Corto entre dos terminales (que no sean tierra ni V_{CC}).

Ahora describiremos cada uno de estos tipos de fallas.

Fallas en los circuitos internos

Por lo general, estas fallas son ocasionadas debido a que uno de los componentes internos falla por completo u opera fuera de sus especificaciones. Cuando esto ocurre, las salidas del CI no responden de manera apropiada a las entradas. No hay forma de predecir lo que harán las salidas, ya que esto depende del componente interno que esté fallando. Algunos ejemplos de este tipo de fallas serían un corto tipo base-emisor en el transistor Q_4 o un valor de resistencia demasiado grande para R_2 en el INVERSOR TTL de la figura 4-30(a). Este tipo de falla interna del CI no es tan común como las otras tres.

Entrada internamente cortocircuitada a tierra o a la fuente de alimentación

Este tipo de falla interna ocasionará que la entrada de un CI se quede atascada en el estado BAJO o ALTO. La figura 4-35(a) muestra la terminal de entrada 2 de una compuerta NAND cortocircuitada a tierra dentro del CI. Esto hará que la terminal 2 siempre esté en el estado BAJO. Si esta terminal de entrada se excita mediante una señal lógica B, sin duda aterrizará B a tierra. Por ende, este tipo de falla afectará la salida del dispositivo que está generando la señal B.

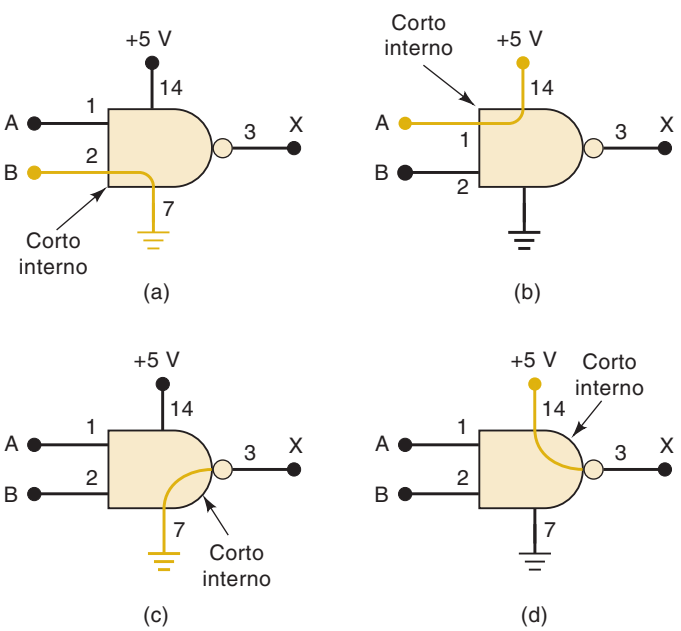


FIGURA 4-35
(a) Entrada de un CI cortocircuitada a tierra en forma interna; (b) entrada de un CI en corto con la fuente de voltaje, de manera interna. Estos dos tipos de fallas obligan a que la señal de entrada en la terminal cortocircuitada se quede en el mismo estado. (c) Salida del CI cortocircuitada a tierra de manera interna; (d) la salida en corto con la fuente de voltaje, de manera interna. Estas dos fallas no afectan a las señales en las entradas del CI.

De manera similar, la terminal de entrada de un CI podría estar en corto interno con +5 V, como en la figura 4-35(b). Esto mantendría a esa terminal atascada en el estado ALTO. Si esta terminal de entrada es excitada por una señal lógica *A*, sin duda pondría en corto a *A* con +5 V.

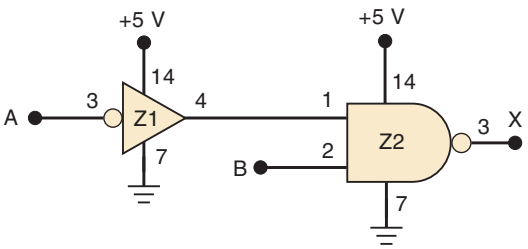
Salida internamente cortocircuitada a tierra o a la fuente de alimentación

Este tipo de falla interna provocará que la terminal de salida se quede atascada en el estado BAJO o ALTO. La figura 4-35(c) muestra a la terminal 3 de la compuerta NAND cortocircuitada a tierra dentro del CI. Esta salida se queda atascada en BAJO y no responderá a las condiciones que se apliquen a las terminales de entrada 1 y 2. En otras palabras, las entradas lógicas *A* y *B* no tendrán efecto sobre la salida *X*.

La terminal de salida de un CI también puede quedar en corto con +5 V dentro del CI, como se muestra en la figura 4-35(d). Esto obliga a que la terminal de salida 3 se quede atascada en ALTO, sin importar el estado de las señales en las terminales de entrada. Observe que este tipo de falla no tiene efecto sobre las señales lógicas en las entradas del CI.

EJEMPLO 4-24

Consulte el circuito de la figura 4-36. Un técnico utiliza una sonda lógica para determinar las condiciones de las diversas terminales del CI. Los resultados están registrados en la figura. Examine estos resultados y determine si el circuito está trabajando en forma apropiada. En caso contrario, sugiera algunas de las posibles fallas.



Terminal	Condición
Z1-3	Pulsando
Z1-4	BAJO
Z2-1	BAJO
Z2-2	ALTO
Z2-3	ALTO

FIGURA 4-36 Ejemplo 4-24.

Solución

La terminal de salida 4 del INVERSOR debe estar pulsando, ya que su entrada está pulsando. No obstante, los resultados registrados muestran que la terminal 4 está atascada en BAJO. Como esta terminal está conectada a la terminal 1 de Z2, la salida de la compuerta NAND se mantiene en ALTO. De nuestra discusión anterior podemos listar tres posibles fallas que podrían producir esta operación.

En primer lugar, podría existir una falla en un componente interno en el INVERSOR que no le permita responder en forma apropiada a su entrada. En segundo lugar, la terminal 4 del INVERSOR podría estar cortocircuitada a tierra en forma interna para Z1, con lo cual se mantendría en BAJO. En tercer lugar, la terminal 1 de Z2 podría estar cortocircuitada a tierra en forma interna para Z2. Esto evitaría que cambiara la terminal de salida del INVERSOR.

Además de estas posibles fallas, puede haber cortos externos a tierra en cualquier parte de la ruta conductora entre la terminal 4 de Z1 y la terminal 1 de Z2. En uno de los siguientes ejemplos veremos cómo aislar la verdadera falla.

Entrada o salida sin conectar (circuito abierto)

Algunas veces el alambre conductor tan fino que conecta la terminal de un CI con los circuitos internos del mismo se rompe, lo cual produce un circuito abierto. La figura 4-37 del ejemplo 4-25 muestra esta situación para una entrada (terminal 13) y una salida (terminal 6). Si se aplica una señal a la terminal 13, no llegará a la

entrada de la compuerta NAND-1 y, por lo tanto, no tendrá efecto sobre su salida. La entrada abierta de la compuerta estará en el estado flotante. Como dijimos antes, los dispositivos TTL responderán como si esta entrada flotante fuera un 1 lógico y los dispositivos CMOS responderán en forma errática, y tal vez podrían dañarse debido al sobrecalentamiento.

La abertura en la salida de la compuerta NAND-4 evita que la señal llegue a la terminal 6 del IC, por lo que no habrá un voltaje estable presente en esa terminal. Si esta terminal se conecta a la entrada de otro CI, producirá una condición flotante en esa entrada.

EJEMPLO 4-25

¿Qué indicaría una sonda lógica en la terminal 13 y en la terminal 6 de la figura 4-37?

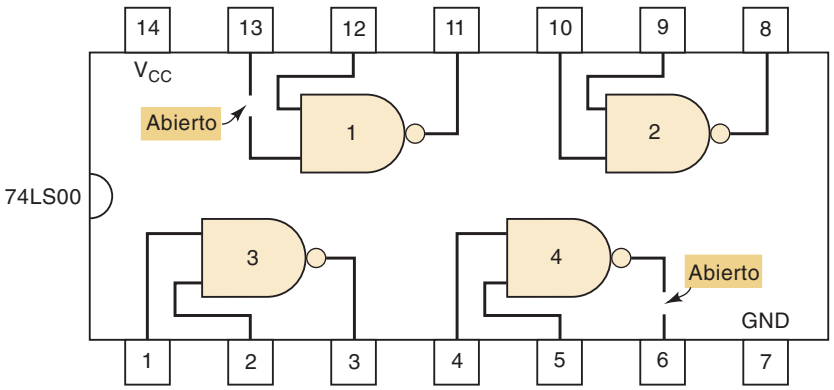


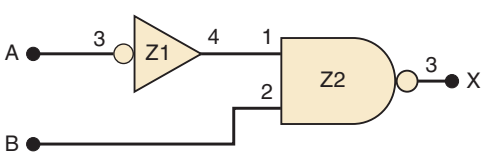
FIGURA 4-37 Un CI con una entrada abierta en forma interna no responderá a las señales que se apliquen a esa terminal de entrada. Una salida abierta en forma interna producirá un voltaje impredecible en esa terminal de salida.

Solución

En la terminal 13, la sonda lógica indicará el nivel lógico de la señal externa que se conecta a la terminal 13 (la cual no se muestra en este diagrama). En la terminal 6, la sonda lógica no tendrá ningún LED encendido para un nivel lógico indeterminado, ya que el nivel de salida de la compuerta NAND nunca llegará a la terminal 6.

EJEMPLO 4-26

Consulte el circuito de la figura 4-38 y las indicaciones registradas de la sonda lógica. ¿Cuáles son algunas de las posibles fallas que producirían esos resultados? Suponga que los CIs son TTL.



Nota: no se muestran las conexiones a V_{CC} ni a tierra de los CI

Terminal	Condición
Z1-3	ALTO
Z1-4	BAJO
Z2-1	BAJO
Z2-2	Pulsando
Z2-3	Pulsando

FIGURA 4-38 Ejemplo 4-26.

Solución

Un análisis de los resultados registrados nos indica que el INVERSOR parece estar funcionando en forma correcta, pero la salida de la compuerta NAND es inconsistente con sus entradas. La salida NAND debería estar en ALTO, ya que su terminal de entrada 1 está en BAJO. Este nivel BAJO debería evitar que la compuerta NAND respondiera a los pulsos en la terminal 2. Es probable que este nivel BAJO no esté llegando a los circuitos internos de la compuerta NAND debido a una abertura interna.

Como el CI es TTL, este circuito abierto produciría el mismo efecto que un nivel ALTO lógico en la terminal 1. Si el CI hubiera sido CMOS, el circuito abierto interno en la terminal 1 podría haber generado una salida indeterminada, lo cual podría provocar un sobrecalentamiento y la destrucción del chip.

De lo antes expuesto en relación con las entradas TTL abiertas, usted podría haber esperado que el voltaje de la terminal 1 de Z2 estuviera entre 1.4 y 1.8 V, y que la sonda lógica debería haberlo registrado como indeterminado. Esto hubiera sido cierto si el circuito abierto fuera *externo* para el chip NAND. Como no hay circuito abierto entre la terminal 4 de Z1 y la terminal 1 de Z2, el voltaje en la terminal 4 de Z1 sí llega a la terminal 1 de Z2, pero se desconecta *dentro* del chip NAND.

Corto entre dos terminales

Un corto interno entre dos terminales del CI obligará a que las señales lógicas en esas terminales siempre sean idénticas. Cada vez que dos señales que se supone deben ser distintas muestran las mismas variaciones de niveles lógicos, existe una buena posibilidad de que las señales estén en corto.

Considere el circuito de la figura 4-39, en donde las terminales 5 y 6 de la compuerta NOR están en corto de manera interna. Este corto hace que las dos terminales de salida del INVERSOR se conecten entre sí, de manera que las señales en la terminal 2 de Z1 y la terminal 4 de Z1 deben ser idénticas, aún y cuando las dos señales de entrada del INVERSOR están tratando de producir diferentes salidas. Para ilustrar esto, considere las formas de onda de entrada que se muestran en el diagrama. Aún cuando estas formas de onda de entrada son distintas, las salidas Z1-2 y Z1-4 son iguales.

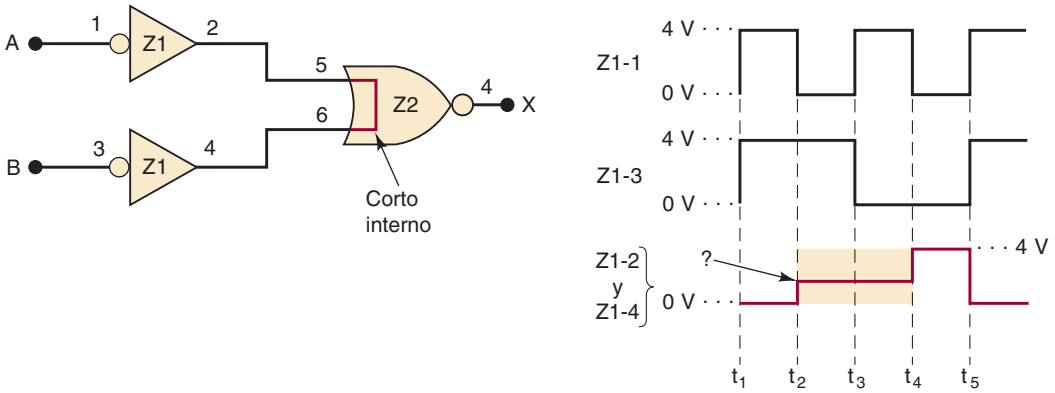


FIGURA 4-39 Cuando se ponen en corto dos terminales de entrada en forma interna, se obliga a las señales que excitan estas terminales a ser idénticas y, por lo general, se produce una señal con tres niveles distintos.

Durante el intervalo de t_1 a t_2, ambos INVERSORES tienen una entrada en ALTO y están tratando de producir una salida en BAJO, por lo que estar en corto no hace ninguna diferencia. Durante el intervalo de t_4 a t_5, ambos INVERSORES tienen una entrada en BAJO y están tratando de producir una salida en ALTO, así que el estar en corto de nuevo no tiene ningún efecto. No obstante, durante los intervalos de t_2 a t_3 y de t_3 a t_4 un INVERSOR está tratando de producir una salida en ALTO, mientras que el otro está tratando de producir una salida en BAJO. A esto se le conoce como **colisión** de señales, ya que las dos señales están "luchando" una con la otra. Cuando esto ocurra, el nivel de voltaje real que aparezca en las salidas en corto dependerá de los circuitos internos del CI. En los dispositivos TTL, por lo general, será un voltaje en el extremo superior del intervalo del 0 lógico (es decir, cerca de 0.8 V), aunque también podría estar en el intervalo indeterminado. En los dispositivos CMOS, por lo general, será un voltaje en el intervalo indeterminado.

Siempre que vea una forma de onda como la señal de Z1-2, Z1-4 en la figura 4-39 con tres niveles distintos, será motivo para sospechar que dos señales de salida pueden estar en corto.

PREGUNTAS DE REPASO

1. Liste las distintas fallas internas en los circuitos integrados.
2. ¿Cuál falla interna de un CI puede producir señales que muestren tres niveles de voltaje distintos?
3. ¿Qué indicaría una sonda lógica en Z1-2 y Z1-4 de la figura 4-39, si $A = 0$ y $B = 1$?
4. ¿Qué es la colisión de señales?

4-12 FALLAS EXTERNAS

Hemos visto cómo reconocer los efectos de diversas fallas internas para los CIs digitales. Hay muchas cosas más que pueden salir mal y que son externas para los CIs; en esta sección describiremos las más comunes.

Líneas de señal abiertas

Esta categoría incluye cualquier falla que produzca una interrupción o discontinuidad en la ruta conductora, de tal forma que se evite que un nivel de voltaje o señal pase de un punto a otro. Algunas de las causas de las líneas de señal abiertas son:

1. Alambre roto.
2. Conexión soldada defectuosamente; conexión de alambre enrollado floja.
3. Grieta o interrupción en la línea de conexión de un circuito impreso (algunas de éstas son del grueso de un cabello, y se pueden ver sólo con una lupa).
4. Terminal doblada o rota en un CI.
5. Zócalo de CI defectuoso, de tal forma que el CI no haga buen contacto con el zócalo.

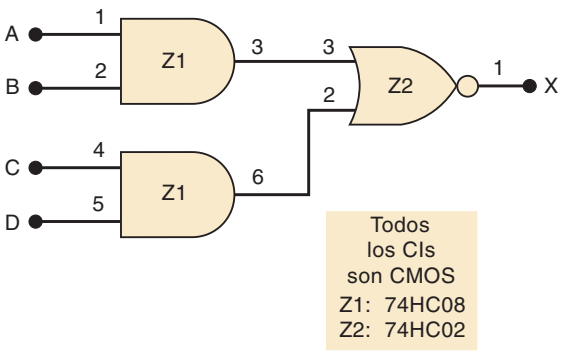
Con frecuencia, este tipo de fallas en los circuitos pueden detectarse mediante una inspección visual cuidadosa y después verificarse desconectando la alimentación del circuito y comprobando si hay continuidad (es decir, una ruta de baja resistencia) con un óhmetro entre los dos puntos en cuestión.

EJEMPLO 4-27

Considere el circuito CMOS de la figura 4-40 y las indicaciones de la sonda lógica que se incluyen. ¿Cuál es la falla más probable del circuito?

Solución

Es probable que el nivel indeterminado en la salida de la compuerta NOR se deba a la entrada indeterminada en la terminal 2. Como hay un nivel BAJO en Z1-6, también debe haber un nivel BAJO en Z2-2. Es evidente que el BAJO de Z1-6 no está llegando a Z2-2, por lo que debe haber un circuito abierto en la ruta de señal entre estos dos puntos. La ubicación de este circuito abierto puede determinarse empezando desde Z1-6 con la sonda lógica y rastreando el nivel BAJO a lo largo de la ruta de la señal hacia Z2-2, hasta que cambie a un nivel indeterminado.



Terminal	Condición
Z1-1	Pulsando
Z1-2	ALTO
Z1-3	Pulsando
Z1-4	BAJO
Z1-5	Pulsando
Z1-6	BAJO
Z2-3	Pulsando
Z2-2	Indeterminado
Z2-1	Indeterminado

FIGURA 4-40 Ejemplo 4-27.

Líneas de señal en corto

Este tipo de falla tiene el mismo efecto que un corto interno entre las terminales de un CI. Provoca que dos señales sean exactamente la misma (colisión de señales). Una línea de señal puede estar cortocircuitada a tierra o a V_{CC} en vez de estar en corto con otra línea de señal. En esos casos se obliga a que la señal permanezca en el estado BAJO o ALTO. Las principales causas de cortos inesperados entre dos puntos en un circuito son las siguientes:

1. *Alambrado mal instalado*. Un ejemplo es cuando se quita demasiado aislante de los extremos de los cables que están muy cerca uno del otro.
2. *Puentes de soldadura*. Son salpicaduras de soldadura que ponen en corto dos o más puntos. Por lo común ocurren entre puntos que están muy cercanos, como las terminales adyacentes en un chip.
3. *Desbaste incompleto*. El cobre entre las rutas conductoras adyacentes en una tarjeta de circuitos impresos no se desbasta de forma adecuada.

De nuevo, con frecuencia una cuidadosa inspección visual puede descubrir este tipo de fallas, y la comprobación con un óhmetro puede verificar que los dos puntos en el circuito están en corto.

Fuente de alimentación defectuosa

Todos los sistemas digitales tienen una o más fuentes de alimentación de corriente directa, las cuales suministran los voltajes V_{CC} y V_{DD} requeridos por los chips. Una fuente defectuosa o una sobrecargada (que suministra más de su valor nominal de corriente) proporcionará voltajes de suministro mal regulados a los CIs, y éstos no operarán u operarán en forma incorrecta.

Una fuente de alimentación puede dejar de regular el voltaje debido a una falla en sus circuitos internos, o porque los circuitos que está alimentando consumen más corriente de la que puede suministrar la fuente. Esto puede ocurrir si un chip o componente tiene una falla que hace que consuma mucha más corriente de la normal.

Una buena práctica de diagnóstico de fallas es comprobar los niveles de voltaje en cada una de las fuentes de alimentación en el sistema, para ver si se encuentran dentro de sus intervalos especificados. También es una buena idea comprobarlas en un osciloscopio para verificar que no haya una cantidad considerable de rizo de corriente alterna en los niveles de corriente directa, y para verificar que los niveles de voltaje permanezcan regulados durante la operación del sistema.

Uno de los signos más comunes de falla en la fuente de alimentación es que uno o más chips operen en forma incorrecta, o que no operen en lo absoluto. Algunos CIs son más tolerantes a las variaciones de la fuente de alimentación y pueden operar en forma apropiada, mientras que otros no. Usted siempre debe comprobar los niveles de voltaje de alimentación y de tierra en cada CI que parezca operar en forma incorrecta.

Carga de salida

Cuando un CI digital tiene su salida conectada a demasiadas entradas de un CI, se excede el valor nominal de su corriente de salida y el voltaje de salida puede caer dentro del intervalo indeterminado. A este efecto se le conoce como *cargar* la señal de salida (en realidad es sobrecargar la señal de salida) y, por lo general, es el resultado de un mal diseño o de una conexión incorrecta.

PREGUNTAS DE REPASO

1. ¿Cuáles son los tipos más comunes de fallas externas?
2. Liste algunas de las causas de circuitos abiertos en la ruta de la señal.
3. ¿Qué síntomas ocasiona una fuente de alimentación defectuosa?
4. ¿Cómo podría afectar la carga a un nivel de voltaje de salida de un CI?

4-13 EJEMPLO PRÁCTICO DE DIAGNÓSTICO DE FALLAS

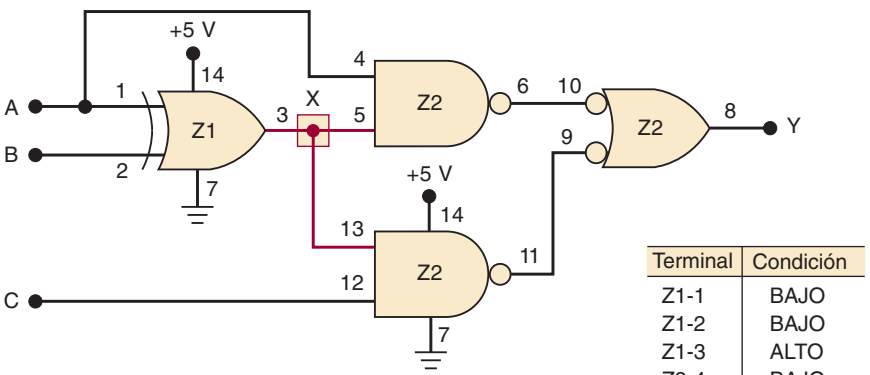
El siguiente ejemplo ilustrará los procesos analíticos involucrados en el diagnóstico de fallas en los circuitos digitales. Aunque el ejemplo es un circuito lógico combinacional bastante simple, el razonamiento y los procedimientos de diagnóstico de fallas pueden aplicarse a los circuitos digitales más complejos que encontraremos en los siguientes capítulos.

EJEMPLO 4-28

Considere el circuito de la figura 4-41. Se supone que la salida Y cambiará a ALTO en cualquiera de las siguientes condiciones:

1. $A = 1, B = 0$ sin importar el nivel en C
2. $A = 0, B = 1, C = 1$

Tal vez desee verificar estos resultados por su cuenta.



Terminal	Condición
Z1-1	BAJO
Z1-2	BAJO
Z1-3	ALTO
Z2-4	BAJO
Z2-5	ALTO
Z2-6,10	ALTO
Z2-13	ALTO
Z2-12	ALTO
Z2-9,11	BAJO
Z2-8	ALTO

FIGURA 4-41 Ejemplo 4-28.

Cuando se prueba el circuito, el técnico observa que la salida *Y* cambia a ALTO cada vez que *A* está en ALTO o que *C* está en ALTO, sin importar el nivel en *B*. Entonces toma mediciones con la sonda lógica para la condición en la que $A = B = 0$, $C = 1$ y obtiene las indicaciones registradas en la figura 4-41.

Examine los niveles registrados y liste las posibles causas del mal funcionamiento. Después desarrolle un procedimiento paso a paso para determinar la falla exacta.

Solución

Todas las salidas de la compuerta NAND son correctas para los niveles presentes en sus entradas. Sin embargo, la compuerta XOR debería producir un nivel BAJO en la terminal 3 de salida, ya que dos de sus entradas se encuentran en el mismo nivel BAJO. Parece que Z1-3 se queda en ALTO, aún y cuando sus entradas deberían producir un nivel BAJO. Hay varias causas posibles de esto:

1. Una falla en un componente interno en Z1 que evite que su salida cambie a BAJO.
2. Un corto externo con V_{CC} desde cualquier punto a lo largo de los conductores conectados al nodo *X* (sombreado en el diagrama de la figura).
3. La terminal 3 de Z1 en corto interno con V_{CC}.
4. La terminal 5 de Z2 en corto interno con V_{CC}.
5. La terminal 13 de Z2 en corto interno con V_{CC}.

Todas estas posibilidades (excepto la primera) pondrán en corto directo el nodo *X* (y todas las terminales del CI que estén conectadas a él) con V_{CC}.

El siguiente procedimiento puede usarse para aislar la falla. Este procedimiento no es el único método que puede utilizarse y, como dijimos antes, el verdadero procedimiento de diagnóstico de fallas que utiliza un técnico es muy dependiente del equipo de prueba que tenga disponible.

1. Compruebe los niveles de V_{CC} y de tierra en las terminales apropiadas de Z1. Aunque es muy poco probable que la ausencia de cualquiera de estas dos señales pueda hacer que Z1-3 se quede en ALTO, es conveniente realizar esta comprobación en cualquier CI que esté produciendo una salida incorrecta.
2. Desconecte la alimentación del circuito y utilice un óhmetro para comprobar si hay un corto (resistencia menor de 1 Ω) entre el nodo *X* y cualquier punto conectado a V_{CC} (tal como Z1-14 o Z2-14). Si no se indica un corto, podemos eliminar las últimas cuatro posibilidades en nuestra lista. Esto significa que es muy probable que Z1 tenga una falla interna y deba sustituirse.
3. Si el paso 2 muestra que hay un corto del nodo *X* a V_{CC}, realice un examen visual detallado de la tarjeta del circuito y busque puentes de soldadura, lengüetas de cobre sin desbastar, alambres sin aislamiento que se toquen unos con otros, y cualquier otra causa posible de un corto externo con V_{CC}. Un punto probable para un puente de soldadura sería entre las terminales adyacentes 13 y 14 de Z2. La terminal 14 se conecta a V_{CC} y la 13 al nodo *X*. Si se encuentra un corto externo, elimínelo y realice una comprobación con un óhmetro para verificar que el nodo *X* ya no esté en corto con V_{CC}.
4. Si el paso 3 no revela un corto externo, las tres posibilidades que quedan son cortos internos con V_{CC} y Z1-3, Z2-13 o Z2-5. Una de éstas está poniendo en corto el nodo *X* con V_{CC}.

Para determinar cuál de estas terminales del CI es la culpable, debemos desconectar cada una de ellas del nodo *X* *una a la vez* y debemos volver a comprobar si hay un corto con V_{CC} después de cada desconexión. Cuando se desconecte la terminal que esté en corto interno con V_{CC}, el nodo *X* ya no estará en corto con V_{CC}.

El proceso de desconectar cada una de las terminales sospechosas del nodo *X* puede ser fácil o difícil, dependiendo de la manera en que esté construido el circuito. Si los CIs están en zócalos, todo lo que necesitamos hacer es extraer el CI de su zócalo, doblar la terminal sospechosa y volver a insertar el CI en su zócalo. Si los CIs están soldados en una tarjeta de circuitos impresos, tendrá que cortar la línea que está conectada con la terminal y reparar la línea cortada cuando termine.

Aunque es bastante simple, el ejemplo 4-28 le muestra el tipo de razonamiento que debe emplear un técnico de diagnóstico de fallas para aislar una falla. Usted tendrá la oportunidad de comenzar a desarrollar sus propias habilidades de diagnóstico de fallas al trabajar en muchos problemas de final de capítulo que hemos designado con una **F** para identificarlos como problemas de diagnóstico de fallas.

4-14 DISPOSITIVOS LÓGICOS PROGRAMABLES*

En las secciones anteriores vimos un poco acerca de las clases de CIs conocidas como dispositivos lógicos programables. En el capítulo 3 presentamos el concepto de describir la operación de un circuito mediante el uso de un lenguaje de descripción de hardware. En esta sección exploraremos aún más estos temas y nos prepararemos para utilizar las herramientas implicadas en el desarrollo y la implementación de sistemas digitales mediante el uso de PLDs. Desde luego que es imposible comprender todos los detalles complejos de cómo funciona un PLD antes de abarcar los fundamentos de los circuitos digitales. A medida que examinemos nuevos conceptos fundamentales, expandiremos nuestro conocimiento sobre los PLDs y los métodos de programación. Presentaremos este material de tal forma que cualquiera que no esté interesado en los PLDs pueda omitir sin problemas estas secciones, sin perder la continuidad en la cobertura de los principios básicos.

Vamos a repasar el proceso que tratamos antes, en relación con el diseño de los circuitos digitales combinacionales. Los dispositivos de entrada se identifican y se les asigna un nombre algebraico tal como *A*, *B*, *C* o *CARGA*, *DESPLAZAMIENTO*, *RELOJ*. De igual forma, los dispositivos de salida reciben nombres tales como X, *Z* o *SALIDA_RELOJ*, *SALIDA_DESPLAZAMIENTO*. Después se crea una tabla de verdad que lista todas las posibles combinaciones de entradas e identifica el estado requerido de las salidas bajo cada condición de entrada. La tabla de verdad es una manera de describir la forma en que va a operar el circuito. Otra manera de describir la operación del circuito es mediante una expresión booleana. A partir de este punto, el diseñador debe encontrar la relación algebraica más simple y seleccionar CIs digitales que puedan alambrarse en conjunto para implementar el circuito. Tal vez usted haya experimentado que estos últimos pasos son los más tediosos, los que consumen más tiempo y los que están más propensos a errores.

Los dispositivos lógicos programables permiten automatizar la mayoría de estos tediosos pasos mediante una computadora y *software de desarrollo* para PLDs. El uso de la lógica programable mejora la eficiencia del proceso de diseño y desarrollo. En consecuencia, la mayoría de los sistemas digitales modernos se implementan de esta forma. El trabajo del diseñador de circuitos es identificar entradas y salidas, especificar la relación lógica de la manera más conveniente y seleccionar un dispositivo programable que sea capaz de implementar el circuito al costo más bajo. El concepto detrás de los dispositivos lógicos programables es simple: poner muchas compuertas lógicas en un solo CI y controlar la interconexión de estas compuertas mediante electrónica.

Hardware de un PLD

En el capítulo 3 vimos que muchos circuitos digitales de la actualidad se implementan mediante el uso de dispositivos lógicos programables (PLDs). Estos dispositivos se configuran en forma electrónica y sus circuitos internos están "alambrados"

* Se pueden omitir todas las secciones que tratan acerca de los PLDs sin perder la continuidad en el balance de los capítulos 1-12.

entre sí mediante electrónica, para formar un circuito lógico. Este alambrado programable puede considerarse como miles de conexiones que están conectadas (1) o desconectadas (0). Es muy tedioso tratar de configurar estos dispositivos en forma manual, colocando 1s y 0s en una rejilla, por lo que la siguiente pregunta lógica es: "¿Cómo controlamos la interconexión de compuertas en un PLD por medio de electrónica?"

Un método común para conectar una de muchas señales que entran en una red a una de muchas líneas de señal que salen de la red es mediante una matriz de conmutación. Consulte la figura 3-44, en donde se introdujo este concepto. Una matriz es tan sólo una rejilla de conductores (alambres) ordenados en filas y columnas. Las señales de entrada se conectan a las columnas de la matriz y las salidas se conectan a las filas de la misma. En cada intersección de una fila y una columna hay un interruptor que puede conectar por medios electrónicos esa fila con esa columna. Los interruptores que conectan filas con columnas pueden ser interruptores mecánicos, enlaces de fusibles, interruptores electromagnéticos (relevadores) o transistores. Ésta es la estructura general que se utiliza en muchas aplicaciones, la cual exploraremos con más detalle cuando estudiemos los dispositivos de memoria en el capítulo 12.

Los PLDs también utilizan una matriz de conmutación, la cual se conoce comúnmente como arreglo programable. Al decidir cuáles intersecciones están conectadas y cuáles no, podemos "programar" la forma en que las entradas se conectan a las salidas del arreglo. En la figura 4-42 se utiliza un arreglo programable para seleccionar las entradas para cada compuerta AND. Observe que en esta simple matriz podemos producir cualquier combinación de productos lógicos de las variables *A*, *B* en cualquiera de las salidas de la compuerta AND. Una matriz o arreglo programable tal como la que se muestra en la figura también puede utilizarse para conectar las salidas de la compuerta AND con compuertas OR. En el capítulo 13 veremos con detalle todo lo relacionado con varias arquitecturas de los PLDs.

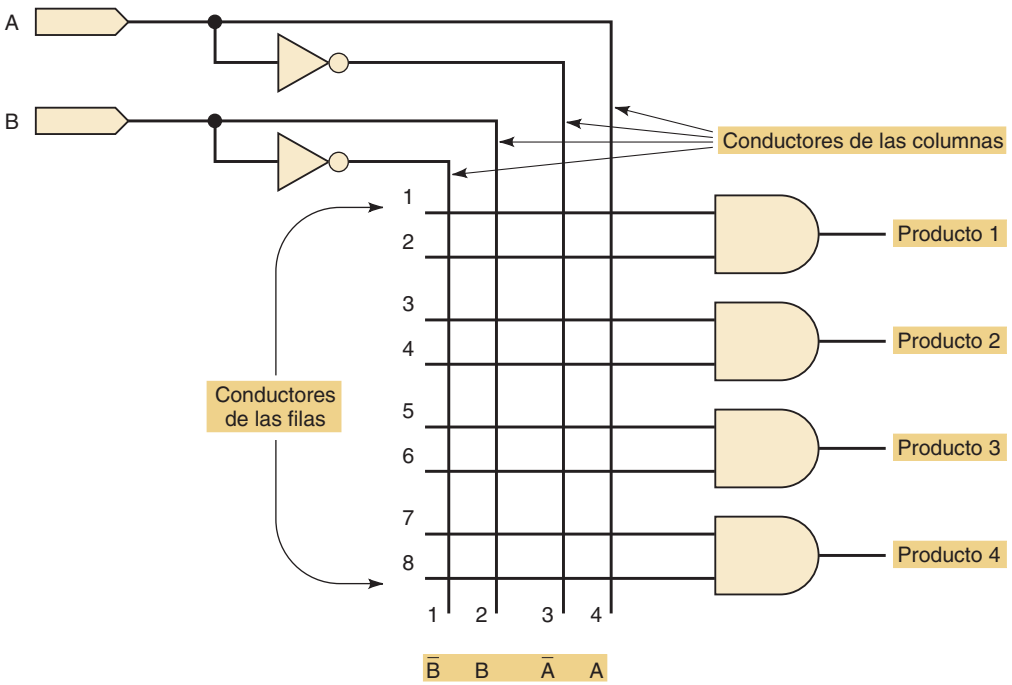


FIGURA 4-42 Un arreglo programable para seleccionar entradas como términos de productos.

Programación de un PLD

Existen dos maneras de "programar" un CI de PLD. Programar significa realizar las conexiones reales en el arreglo. En otras palabras, significa determinar cuáles

de esas conexiones se supone deben estar abiertas (0) y cuáles se supone deben estar cerradas (1). El primer método implica remover el chip de CI del PLD de su tarjeta de circuito. Después el chip se coloca en un dispositivo especial conocido como **programador**, el cual se muestra en la figura 4-43. La mayoría de los programadores modernos se conectan a una computadora personal que ejecuta software que contiene bibliotecas de información acerca de los diversos tipos de dispositivos programables disponibles.

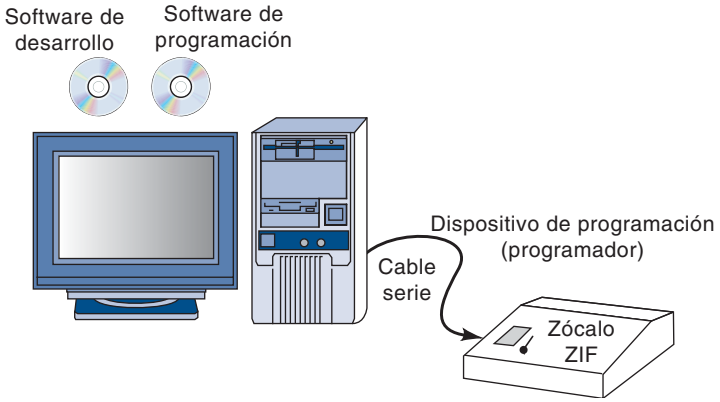


FIGURA 4-43 Un sistema de desarrollo de PLDs.

El software de programación se invoca (se llama y se ejecuta) en la PC para establecer la comunicación con el programador. Este software permite al usuario configurar el programador para el tipo de dispositivo que se va a programar, comprobar si el dispositivo está en blanco, leer el estado de cualquier conexión programable en el dispositivo y proveer las instrucciones para que el usuario programe un chip. En última instancia, la pieza se coloca en un zócalo especial que nos permite insertar el chip y después sujetar los contactos en las terminales. A éste se le conoce como zócalo de **cero esfuerzo de inserción (ZIF)**. Diversos fabricantes ofrecen los *programadores universales* que pueden manejar cualquier tipo de dispositivo programable.

Por fortuna, y a medida que las piezas programables empezaron a proliferar, los fabricantes vieron la necesidad de estandarizar las asignaciones de terminales y los métodos de programación. Como resultado, se formó el Consejo Común de Ingeniería de Dispositivos Electrónicos (**JEDEC**). Uno de los resultados fue el estándar 3 de JEDEC, un formato para transferir datos de programación para PLDs, independiente del fabricante del PLD o del software de programación. También se estandarizaron las asignaciones de terminales para varios encapsulados de CIs, con lo cual los programadores universales se hicieron menos complicados. En consecuencia, los dispositivos de programación pueden programar muchos tipos de PLDs. El software que permite al diseñador especificar una configuración para un PLD sólo necesita producir un archivo de salida que se adapte a los estándares de JEDEC. Después, este archivo JEDEC puede cargarse en cualquier programador de PLDs compatible con JEDEC que sea capaz de programar el tipo deseado de PLD.

El segundo método se conoce como programación en sistema (ISP). Como su nombre implica, el chip no necesita extraerse de su circuito para su programación. El Grupo Común de Acción de Pruebas (**JTAG**) desarrolló una interfaz estándar para probar los CIs sin tener que conectar el equipo de prueba a todas las terminales del CI. También permite la programación interna. Cuatro terminales en el CI se utilizan como un portal para almacenar datos y recuperar información acerca de la condición interna del CI. Muchos CIs, incluyendo los PLDs y los microcontroladores, se fabrican hoy en día para incluir la interfaz JTAG. Un cable de interfaz conecta las cuatro terminales JTAG en el CI a un puerto de salida (como el puerto de la impresora) de una computadora personal. El software que se ejecuta en la PC establece el contacto con el CI y carga la información en el formato apropiado.

Software de desarrollo

Hasta ahora hemos analizado varios métodos para describir circuitos lógicos, incluyendo la captura de diagramas esquemáticos, las ecuaciones lógicas, las tablas de verdad y el HDL. También describimos los métodos fundamentales para almacenar 1s y 0s en un CI de PLD para conectar los circuitos lógicos de la manera deseada. El mayor reto en cuanto a programar un PLD es realizar la conversión desde cualquier forma de descripción hacia el arreglo de 1s y 0s. Por fortuna, esta tarea se logra con bastante facilidad mediante una computadora que ejecute un software de desarrollo. El software al que haremos referencia y que estaremos utilizando en nuestros ejemplos es propiedad de Altera, y permite al diseñador introducir la descripción de un circuito en cualquiera de las distintas formas que hemos visto: archivos de diseño gráfico (diagramas esquemáticos), AHDL y VHDL. También permite el uso de otro HDL conocido como Verilog, y la opción de describir el circuito mediante diagramas de tiempos. Los bloques de circuitos descritos por cualquiera de estos métodos también pueden "conectarse" entre sí para implementar un sistema digital mucho más grande, como se muestra en la figura 4-44. Cualquier diagrama lógico que se presente en este libro podrá redibujarse mediante el uso de las herramientas de introducción de diagramas esquemáticos en el software Altera para crear un archivo de diseño gráfico. En este libro no nos enfocaremos en la introducción mediante el diseño gráfico, ya que es bastante sencillo obtener estas habilidades en el laboratorio. Enfocaremos nuestros ejemplos en los métodos que nos permiten utilizar el HDL como un medio alternativo para describir un circuito. Para obtener más información sobre el software de Altera, vea el CD que se incluye en este libro y los manuales de usuario del sitio Web de Altera (http://www.altera.com).

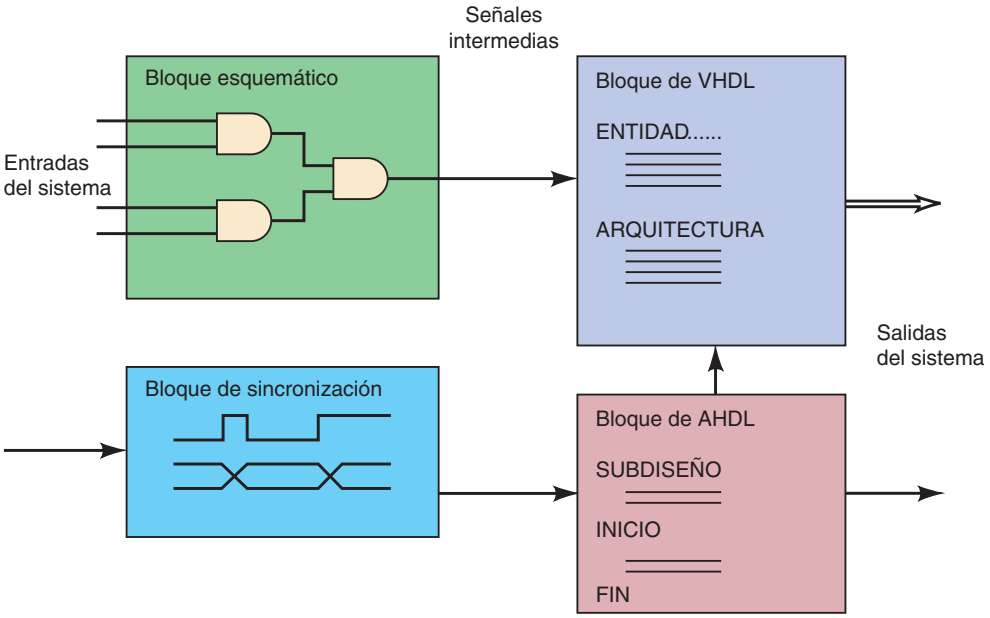


FIGURA 4-44
Combinación de los bloques desarrollados mediante el uso de distintos métodos de descripción.

A este concepto de utilizar bloques de construcción de circuitos se le conoce como **diseño jerárquico**. Pueden definirse circuitos lógicos pequeños y útiles de la manera que sea más conveniente (gráfico, HDL, diagrama de tiempos, etc.) y después se pueden combinar con otros circuitos para formar una selección extensa de un proyecto. Las secciones pueden combinarse y conectarse con otras secciones para formar el sistema completo. La figura 4-45 muestra la estructura jerárquica de un reproductor de CDs mediante el uso de un diagrama de bloques. El cuadro exterior encierra a todo el sistema. Las líneas punteadas identifican cada una de las subsecciones principales y cada subsección contiene circuitos individuales. Aunque no se muestra en este diagrama, cada circuito puede estar compuesto de bloques de cons-

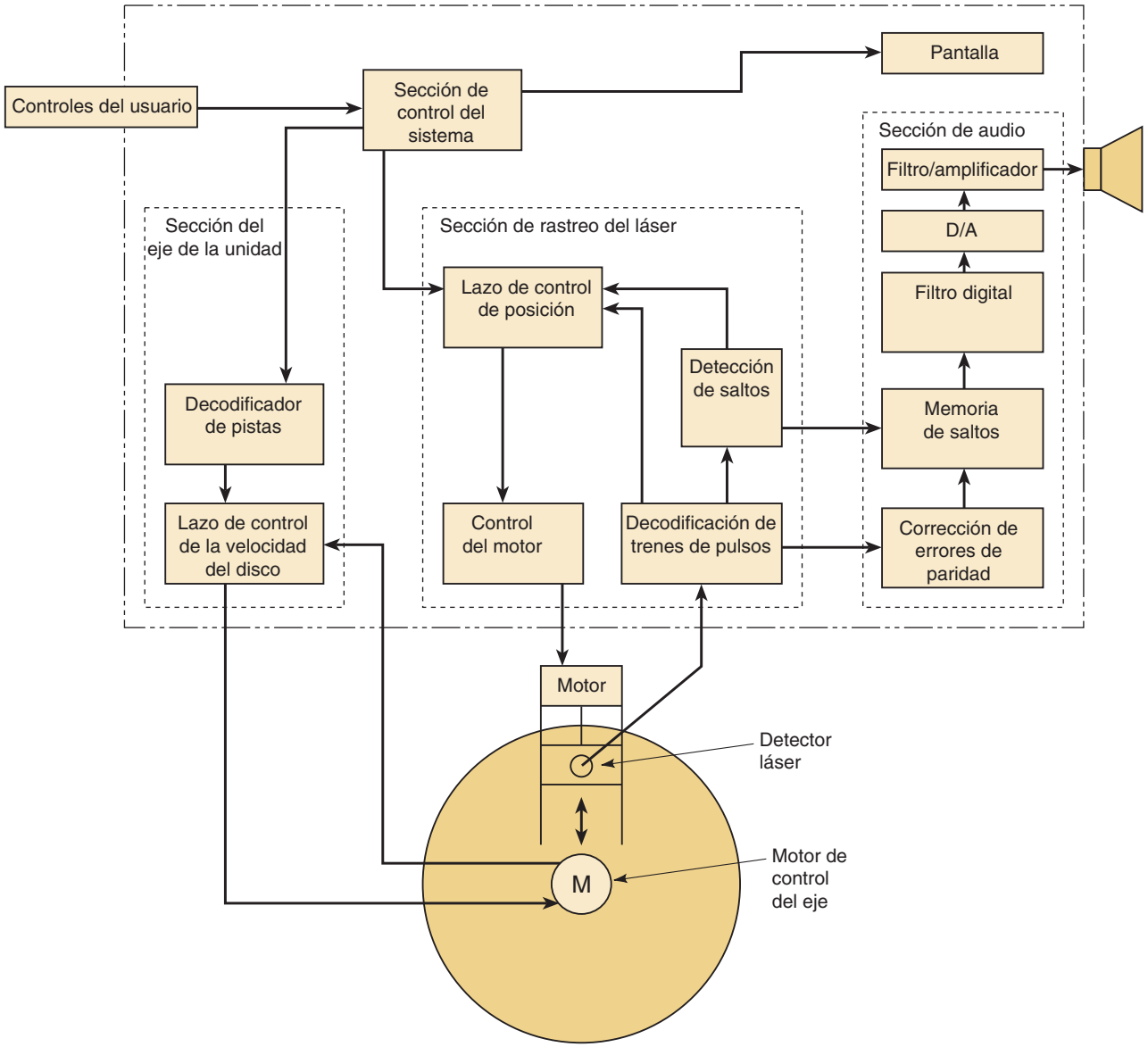


FIGURA 4-45 Diagrama de bloques de un reproductor de CDs.

trucción más pequeños de circuitos digitales comunes. El software de desarrollo de Altera hace que este tipo de diseño y desarrollo modular y jerárquico sea fácil de lograr.

Proceso de diseño y desarrollo

En la figura 4-46 se muestra otra forma en la que podríamos ver la jerarquía de un sistema como el reproductor de CDs que acabamos de describir. El nivel superior representa a todo el sistema completo. Está compuesto de tres subsecciones, cada una de las cuales está a su vez compuesta de los circuitos más pequeños que se muestran. Observe que este diagrama no muestra cómo fluyen las señales en todo el sistema, sino que identifica con claridad los diversos niveles de la estructura jerárquica del proyecto.

Este tipo de diagrama condujo al nombre de uno de los métodos más comunes de diseño: **arriba-abajo**. Con este enfoque de diseño se comienza con la descripción general de todo el sistema, como muestra el cuadro superior de la figura 4-46. Después se definen varias subsecciones que conformarán el sistema. Las subsecciones

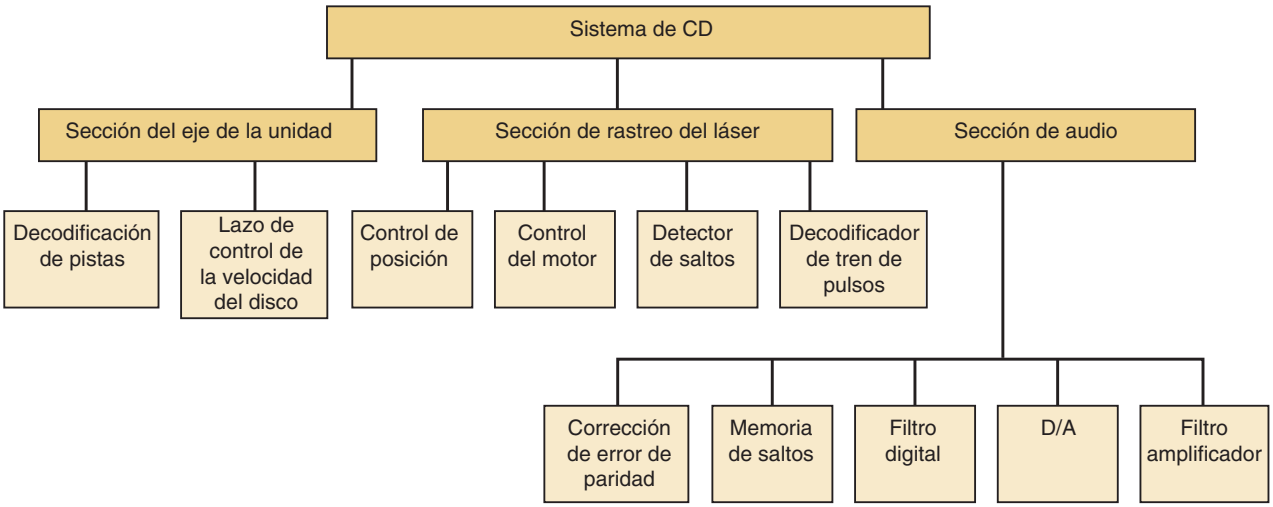


FIGURA 4-46 Un diagrama de jerarquía organizacional.

se refinan aún más en circuitos individuales conectados entre sí. Cada uno de estos niveles jerárquicos tiene definidas las entradas, las salidas y su comportamiento. Cada nivel puede probarse en forma individual, antes de conectarlo a los demás.

Después de definir los bloques de arriba-abajo, el sistema se construye desde abajo hacia arriba. Cada bloque en el diseño de este sistema tiene un archivo de diseño que lo describe. Para diseñar los bloques de los niveles se abre un archivo de diseño y se escribe una descripción de su operación. Después el bloque diseñado se compila mediante el uso de las herramientas de desarrollo. El proceso de compilación determina si usted cometió errores en su sintaxis. La computadora no podrá traducir su descripción en la forma apropiada sino hasta que la sintaxis esté correcta. Una vez que se haya compilado sin errores de sintaxis, deberá probarse para ver si opera en forma correcta. Los sistemas de desarrollo ofrecen programas simuladores que se ejecutan en la PC y simulan la manera en que su circuito responde a las entradas. Un simulador es un programa de computadora que calcula los estados lógicos de salida correctos, con base en una descripción del circuito lógico y las entradas actuales. Se desarrolla un conjunto de entradas hipotéticas y sus correspondientes salidas correctas, las cuales demostrarán que el bloque funciona de la manera esperada. A menudo a estas entradas hipotéticas se les llama **vectores de prueba**. Los procedimientos detallados de prueba durante la simulación incrementan en forma considerable la probabilidad de que el sistema final funcione de manera confiable. La figura 4-47 muestra el archivo de simulación para el circuito descrito en la figura 3-13(a) del capítulo 3. Las entradas *a*, *b* y *c* se introdujeron como vectores de prueba y la simulación produjo la salida *y*.

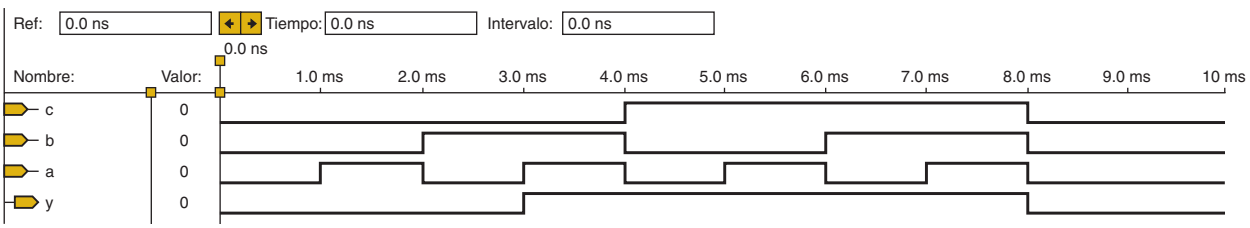
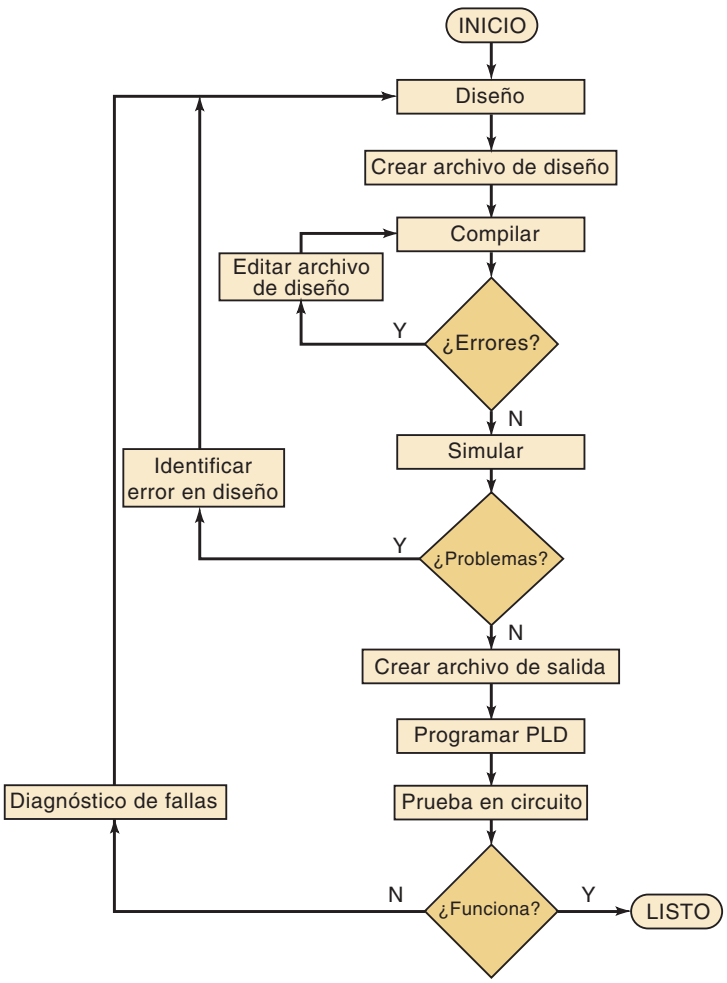


FIGURA 4-47 Una simulación de sincronización de un circuito descrito en HDL.

Cuando el diseñador está satisfecho de que el diseño funciona, éste puede verificarse mediante la programación de un chip y la prueba. Para un PLD complejo, el diseñador puede dejar que el sistema de desarrollo asigne terminales y después se distribuye la tarjeta del circuito final de manera acorde, o puede especificar las

terminales para cada señal mediante el uso de las características del software. Si el compilador asigna las terminales, las asignaciones se pueden encontrar en el archivo de reporte o en el archivo de distribución de terminales, el cual proporciona muchos detalles acerca de la implementación del diseño. Si el diseñador especifica las terminales, es importante conocer las restricciones y limitaciones de la arquitectura del chip. En el capítulo 13 cubriremos estos detalles. El diagrama de flujo de la figura 4-48 sintetiza el proceso de diseño para desarrollar cada bloque.

FIGURA 4-48
Diagrama de flujo del ciclo de desarrollo de un PLD.



Después de probar cada uno de los circuitos en una subsección pueden combinarse todos y se puede probar la subsección siguiendo el mismo proceso utilizado para los circuitos pequeños. Después se combinan las subsecciones y se prueba el sistema. Este método se presta muy bien para un típico entorno de proyecto, en donde un equipo de personas trabajan en conjunto y cada uno es responsable de sus propios circuitos y secciones que al final se reunirán para conformar el sistema.

PREGUNTAS DE REPASO

1. ¿Qué es lo que se "programa" en un PLD?
2. ¿Qué bits (columna, fila) en la figura 4-42 deben conectarse para hacer que el Producto 1 = AB?
3. ¿Qué bits (columna, fila) en la figura 4-42 deben conectarse para hacer que el Producto 3 = $A\overline{B}$?

4-15 REPRESENTACIÓN DE DATOS EN HDL

Los datos numéricos pueden representarse de varias formas. Hemos estudiado el uso del sistema numérico hexadecimal como una forma conveniente de representar patrones de bits. Por naturaleza preferimos usar el sistema numérico decimal para los datos numéricos, pero las computadoras y los sistemas digitales sólo pueden operar con información binaria, como vimos en capítulos anteriores. Cuando escribimos en HDL, a menudo necesitamos utilizar diversos formatos numéricos, y la computadora debe ser capaz de comprender cuál sistema numérico estamos usando. Hasta ahora, en este libro, hemos utilizado un subíndice para indicar el sistema numérico. Por ejemplo, 101_2 es binario, 101_{16} es hexadecimal y 101_{10} es decimal. Cada lenguaje de programación y el HDL tienen su propia manera única de identificar los diversos sistemas numéricos; por lo general, esto se hace mediante un prefijo para indicar el sistema numérico. En la mayoría de los lenguajes un número sin prefijo se considera como decimal. Cuando leemos una de estas designaciones numéricas, debemos considerarla como un símbolo que representa a un patrón binario de bits. Estos valores numéricos se conocen como escalares o **literales**. La tabla 4-8 sintetiza los métodos para especificar valores en binario, hexadecimal y decimal para AHDL y VHDL.

TABLA 4-8 Designación de sistemas numéricos en HDL.

Sistema numérico	AHDL	VHDL	Patrón de bits	Equivalente decimal
Binario	B"101"	B"101"	101	5
Hexadecimal	H"101"	X"101"	100000001	257
Decimal	101	101	1100101	101

EJEMPLO 4-29

Exprese el valor numérico del siguiente patrón de bits en binario, hexadecimal y decimal mediante el uso de la notación de AHDL y VHDL:

11001

Solución

El binario se designa de la misma forma en AHDL y en VHDL: **B "11001"**.

Si convertimos el binario en hexadecimal, tendremos 19_{16}.

En AHDL: **H "19"**

En VHDL: **X "19"**

Si convertimos el binario en decimal, tendremos 25_{10}.

El decimal se designa de la misma forma en AHDL y VHDL: 25.

Arreglos de bits/vectores de bits

En el capítulo 3 declaramos nombres para las entradas y las salidas de un circuito lógico muy simple. Éstas se definieron como bits, o dígitos binarios individuales. ¿Qué pasaría si quisiéramos representar una entrada, salida o señal compuesta por varios bits? En un HDL debemos definir el tipo de la señal y su intervalo de valores válidos.

Para comprender los conceptos utilizados en los HDLs, consideremos primero algunas convenciones para describir bits de palabras binarias en los sistemas digitales comunes. Suponga que tenemos un número de ocho bits que representa la temperatura actual, y que el número está llegando a nuestro sistema digital a través de un puerto de entrada que hemos identificado como P1, como se muestra en la figura 4-49. Podemos referirnos a los bits individuales de este puerto como el bit 0 de P1 para el bit menos significativo, y hasta el bit 7 de P1 para el bit más significativo.

También podemos describir este puerto diciendo que su nombre es P1 y sus bits están numerados del 7 al 0. Los términos **arreglo de bits** y **vector de bits** se utilizan con frecuencia para describir este tipo de estructura de datos. Lo que esto significa es que la estructura de datos en general (puerto de ocho bits) tiene un nombre (P1) y que cada elemento individual (bit) tiene un número de **índice** único (0-7) para describir la posición de cada bit (y tal vez su peso numérico) en la estructura general. Los HDLs y los lenguajes de programación de computadora sacan provecho de esta notación. Por ejemplo, el tercer bit a partir de la derecha se designa como P1[2] y puede conectarse a otro bit de señal mediante el uso de un operador de asignación.

EJEMPLO 4-30

Suponga que hay un arreglo de ocho bits llamado P1 como se muestra en la figura 4-49, y que hay otro arreglo de cuatro bits llamado P5.

(a) Escriba la designación para el bit más significativo de P1.
(b) Escriba la designación para el bit menos significativo de P5.
(c) Escriba una expresión que haga que el bit menos significativo de P5 controle el bit más significativo de P1.

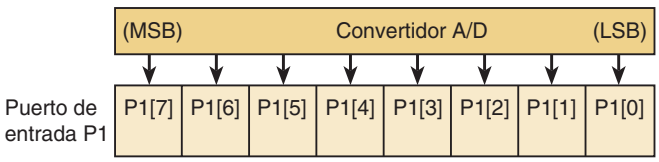


FIGURA 4-49
Notación de arreglo de bits.

Solución

(a) El nombre del puerto es P1 y el bit más significativo es el bit 7. La designación apropiada para el bit 7 de P1 es P1[7].
(b) El nombre del puerto es P5 y el bit menos significativo es el bit 0. La designación apropiada para el bit 0 de P5 es P5[0].
(c) La señal controladora se coloca en el lado derecho del operador de asignación y la señal controlada se coloca a la izquierda: P1[7] = P5[0];.

AHDL

DECLARACIÓN DE ARREGLOS DE BITS EN AHDL

En AHDL, el puerto *p1* de la figura 4-49 se define como un puerto de entrada de ocho bits y para hacer referencia a el valor en este puerto se puede utilizar cualquier sistema numérico tal como hexadecimal, binario, decimal, etc. La sintaxis para AHDL utiliza un nombre para el vector de bits, seguido por el intervalo de las designaciones de los índices, las cuales van encerradas entre corchetes. Esta declaración se incluye en la sección SUBDESIGN. Por ejemplo, para declarar un puerto de entrada de ocho bits llamado *p1*, escribiríamos lo siguiente:

```
p1 [7..0] : INPUT;     define un puerto de entrada de 8 bits
```

EJEMPLO 4-31

Declare una entrada de cuatro bits llamada *teclado* mediante el uso de AHDL.

Solución

```
teclado [3..0]  :  INPUT;
```

Las variables intermedias también pueden declararse como un arreglo de bits. Al igual que los bits individuales, se declaran justo después de las declaraciones de E/S en SUBDESIGN. Como ejemplo, el puerto de temperatura de ocho bits llamado *p1* puede asignarse (conectarse) a un nodo llamado *temp*, de la siguiente manera:

```
VARIABLE temp [7..0] :NODE;
BEGIN
    temp[] = p1[];
END;
```

Observe que los datos se aplican al puerto de entrada *p1* y que está controlando los alambres de la señal llamada *temp*. Podemos considerar el término a la derecha del signo de igual como el origen de los datos, y el término a la izquierda como la designación. Los corchetes vacíos [ ] indican que cada uno de los bits correspondientes en los dos arreglos están conectados. También pueden "conectarse" bits individuales si se especifican los bits dentro de los corchetes. Por ejemplo, para conectar sólo el bit menos significativo de *p1* con el LSB de *temp*, la instrucción sería temp[0] = p1[0];.

DECLARACIONES DE VECTORES DE BITS EN VHDL

En VHDL, el puerto *p1* de la figura 4-49 se define como un puerto de entrada de ocho bits y sólo se puede hacer referencia al valor de este puerto mediante el uso de literales binarias. La sintaxis para VHDL utiliza un nombre para el vector de bits, seguido del modo (:IN), el tipo (**BIT_VECTOR**) y el intervalo de las designaciones de los índices, que van encerradas entre paréntesis. Esta declaración se incluye en la sección ENTITY. Por ejemplo, para declarar un puerto de entrada de ocho bits llamado *p1*, escribiríamos lo siguiente:

```
PORT (p1  : IN   BIT VECTOR  (7 DOWNTO 0);
```

EJEMPLO 4-32

Declare una entrada de cuatro bits llamada *teclado* mediante el uso de VHDL.

Solución

```
PORT(teclado  : IN  BIT VECTOR  (3 DOWNTO 0);
```

También pueden declararse señales intermedias como un arreglo de bits. Al igual que los bits individuales, se declaran justo dentro de la definición ARCHITECTURE. Como ejemplo, la temperatura de ocho bits en el puerto *p1* puede asignarse (conectarse) a una señal llamada *temp*, como se muestra a continuación:

```
SIGNAL     temp  : BIT VECTOR (7 DOWNTO 0);
BEGIN
     temp <5 p1;
END;
```

Observe que los datos se aplican al puerto de entrada *p1* y que está controlando los alambres de la señal llamada *temp*. No se especifican elementos en el vector de bits, lo cual significa que todos los bits están conectados. También pueden "conectarse" bits individuales mediante el uso de asignaciones de señales y especificando los números de los bits entre paréntesis. Por ejemplo, para conectar sólo el bit menos significativo de *p1* con el LSB de *temp*, la instrucción sería temp(0) < = p1(0);.

VHDL es muy específico con respecto a las definiciones de cada tipo de datos. El tipo "bit_vector" describe a un arreglo de bits individuales. Esto se interpreta de manera distinta que un número binario de ocho bits (llamado cantidad escalar), el cual tiene el tipo **integer**. Por desgracia, VHDL no nos permite asignar un valor entero a una señal BIT_VECTOR de forma directa. Los datos pueden representarse mediante cualquiera de los tipos que se muestran en la figura 4-9, pero las asignaciones de datos y demás operaciones deben realizarse entre objetos del mismo tipo. Por ejemplo, el compilador no le permitirá recibir un número de un teclado declarado como entero y conectarlo a cuatro LEDs que estén declarados como salidas BIT_VECTOR. En la tabla 4-9 bajo Posibles valores podrá observar que los **objetos** individuales de datos BIT y STD_LOGIC (por ejemplo: señales, variables, entradas y salidas) se designan mediante comillas, mientras que los valores asignados a los tipos BIT_VECTOR y STD_LOGIC_VECTOR son cadenas de valores de bits válidos encerradas entre comillas dobles.

TABLA 4-9 Tipos de datos comunes en VHDL.

Tipo de datos	Declaración de ejemplo	Posibles valores	Uso
BIT	y :OUT BIT;	'0' '1'	y <= '0';
STD LOGIC	controlador: STD LOGIC	'0' '1' 'z' 'x' '-'	controlador <= 'z';
BIT VECTOR	datos bcd :BIT VECTOR (3 DOWNTO 0);	"0101" "1001" "0000"	digito <= datos bcd;
STD LOGIC VECTOR	dbus :STD LOGIC VECTOR (3 DOWNTO 0);	"0Z1X"	IF rd = '0' THEN dbus <= "zzzz";
INTEGER	SIGNAL z:INTEGER RANGE − 32 TO 31;	−32..−2, −1, 0, 1, 2 . . . 31	IF z > 5 THEN . . .

VHDL también ofrece ciertos tipos de datos estandarizados que se necesitan al utilizar funciones lógicas contenidas en las **bibliotecas**. Como habrá imaginado, las bibliotecas son simples colecciones de pequeñas piezas de código de VHDL, las cuales puede usar en sus descripciones de hardware sin necesidad de empezar desde cero. A menudo estas bibliotecas ofrecen funciones de uso común conocidas como **macrofunciones**, al igual que muchos de los dispositivos TTL estándar que se describen en este libro. En vez de escribir la nueva descripción de un dispositivo TTL conocido, podemos tan sólo sacar su macrofunción de la biblioteca y utilizarla en nuestro sistema. Desde luego que necesita que las señales entren y salgan de estas macrofunciones; además los tipos de las señales en su código deben concordar con los tipos en las funciones (que alguien más escribió). Esto significa que todos deben utilizar los mismos tipos de datos estándar.

Cuando el VHDL se estandarizó a través del IEEE, se crearon muchos tipos de datos a la vez. Lo dos que utilizaremos en este libro son **STD_LOGIC**, que es equivalente al tipo BIT, y **STD_LOGIC_VECTOR**, que es equivalente a BIT_VECTOR. Como podrá recordar, el tipo BIT sólo puede tener los valores '0' y '1'. Los tipos lógicos estándar vienen definidos en la biblioteca IEEE y tienen un intervalo más amplio de valores posibles que sus contrapartes integradas en la biblioteca. Los posibles valores para un tipo STD_LOGIC o para cualquier elemento en un STD_LOGIC_VECTOR se muestran en la tabla 4-10. Los nombres de estas categorías tendrán mucho más sentido una vez que estudiemos las características de los circuitos lógicos en el capítulo 8. Por ahora mostraremos ejemplos con el uso de los valores '1' y '0' solamente.

TABLA 4-10
Valores de STD_LOGIC.

'1'	1 lógico (justo igual que el tipo BIT)
'0'	0 lógico (justo igual que el tipo BIT)
'z'	Alta impedancia*
'-'	No importa (justo igual que como se utilizó en los mapas K)
'U'	Sin inicializar
'X'	Desconocido
'W'	Desconocido débil
'L'	'0' débil
'H'	'1' débil

* En el capítulo 8 estudiaremos la lógica de tres estados.

PREGUNTAS DE REPASO

1. ¿Cómo declararía un arreglo de entrada de seis bits llamado botones_pulsar en (a) AHDL o en (b) VHDL?
2. ¿Qué instrucción utilizaría para extraer el MSB del arreglo de la pregunta 1 y colocarlo en un puerto de salida de un solo bit llamado *z*? Use (a) AHDL o (b) VHDL.
3. En VHDL, ¿cuál es el tipo estándar IEEE que es equivalente al tipo BIT?
4. En VHDL, ¿cuál es el tipo estándar IEEE que es equivalente al tipo BIT_VECTOR?

4-16 TABLAS DE VERDAD MEDIANTE EL USO DE HDL

Ya hemos aprendido que una tabla de verdad es otra forma de expresar la operación de un bloque de circuitos. Relaciona la salida del circuito con cada una de las posibles combinaciones de sus entradas. Como vimos en la sección 4-4, una tabla de verdad es el punto inicial para que un diseñador defina la manera en que debe operar el circuito. Después se deriva una expresión booleana de la tabla de verdad y se simplifica mediante el uso de los mapas K o del álgebra booleana. Por último, el circuito se implementa a partir de la ecuación booleana final. ¿No sería grandioso si pudiéramos partir de la tabla de verdad y llegar en forma directa al circuito final, sin todos esos pasos intermedios? Si utilizamos HDL para introducir la tabla de verdad, podremos hacer justo eso.

TABLAS DE VERDAD MEDIANTE EL USO DE AHDL

El código de la figura 4-50 utiliza AHDL para implementar un circuito y utiliza una tabla de verdad para describir su operación. La tabla de verdad para este diseño se presentó en el ejemplo 4-7. El punto clave de este ejemplo es el uso de la palabra clave TABLE en AHDL. Esta palabra permite al diseñador especificar la operación del circuito, igual que como se llena una tabla de verdad. En la primera línea después de TABLE se listan las variables de entrada (*a*, *b*, *c*) de la misma forma como se crearía un encabezado de columna en una tabla de verdad. Al incluir las tres variables binarias entre paréntesis, indicamos al compilador que deseamos utilizar estos tres bits como un grupo y que nos referiremos a ellos como si fueran un número binario o un patrón de tres bits. Los valores específicos para este patrón de bits se listan debajo del grupo y se les denomina literales binarias. El operador especial (= >) se utiliza en las tablas de verdad para separar las entradas de la salida (*y*).

AHDL

FIGURA 4-50 Archivo de diseño de AHDL para la figura 4-7.

```
%     Figura 4 7 en AHDL
      Sistemas digitales 10a ed
      Neal Widmer
      MAYO 23, 2005                    %
SUBDESIGN FIG4 50
(
      a,b,c :INPUT;           define las entradas del bloque
      y     :OUTPUT;          define la salida del bloque
)
BEGIN
      TABLE
            (a,b,c)          =>    y;     encabezados de columna
            (0,0,0)          =>    0;
            (0,0,1)          =>    0;
            (0,1,0)          =>    0;
            (0,1,1)          =>    1;
            (1,0,0)          =>    0;
            (1,0,1)          =>    1;
            (1,1,0)          =>    1;
            (1,1,1)          =>    1;
      END TABLE;
END;
```

La instrucción TABLE en la figura 4-50 es para mostrar la relación entre el código de HDL y una tabla de verdad. Una manera más común de representar los encabezados de los datos de entrada es mediante el uso de un arreglo de bits para representar el valor en *a*, *b*, *c*. Este método requiere que se declare el arreglo de bits en la línea antes de BEGIN, como se muestra a continuación:

```
VARIABLE  bits ent[2..0]          :NODE;
```

Justo antes de la palabra clave TABLE, los bits de entrada pueden asignarse al arreglo *bits_ent[ ]*:

```
bits ent[ ] = (a, b, c);
```

Al proceso de agrupar tres bits independientes en un orden como el anterior se le conoce como **concatenación**; este proceso se lleva a cabo con frecuencia para conectar bits individuales con un arreglo de bits. En este caso, el encabezado de la tabla en los conjuntos de bits de entrada puede representarse mediante *bits_ent[ ]*. Observe que, a medida que listamos las posibles combinaciones de las entradas, tenemos varias opciones. Podemos crear un grupo de 1s y 0s entre paréntesis, como se muestra en la figura 4-50, o podemos representar el mismo patrón de bits utilizando el número equivalente en binario, hexadecimal o decimal. El diseñador es el que decidirá cuál formato es el más apropiado, dependiendo de lo que representen las variables de entrada.

VHDL

TABLAS DE VERDAD MEDIANTE EL USO DE VHDL: ASIGNACIÓN DE SEÑAL SELECCIONADA

El código de la figura 4-51 utiliza VHDL para implementar un circuito mediante una **asignación de señal seleccionada** para describir su operación. Esto le permite al diseñador especificar la operación del circuito, de la misma forma como se llena una tabla de verdad. En el ejemplo 4-7 se presentó la tabla de verdad para este diseño. El objetivo principal de este ejemplo es el uso de la instrucción WITH nombre_señal

```
      Figura 4 7 en VHDL
      Sistemas digitales 10a ed
      Neal Widmer
      MAYO 23, 2005
ENTITY fig4 51 IS
PORT(
      a,b,c :IN BIT;            declara bits de entrada individuales
      y     :OUT BIT);
END fig4 51;

ARCHITECTURE verdad OF fig4 51 IS
      SIGNAL bits ent  : BIT VECTOR(2 DOWNTO 0);
      BEGIN
      bits ent <5 a & b & c;   concatena los bits de entrada en bit vector
            WITH bits ent  SELECT
            y     <=    '0'  WHEN "000",     Tabla de verdad
                        '0'  WHEN "001",
                        '0'  WHEN "010",
                        '1'  WHEN "011",
                        '0'  WHEN "100",
                        '1'  WHEN "101",
                        '1'  WHEN "110",
                        '1'  WHEN "111";
END verdad;
```

FIGURA 4-51 Archivo de diseño de VHDL para la figura 4-7.

SELECT en VHDL. Uno de los objetivos secundarios es mostrar cómo se colocan los datos en un formato que pueda ser utilizado de manera conveniente con la asignación de la señal seleccionada. Observe que las entradas están definidas en la declaración ENTITY como tres bits independientes *a*, *b* y *c*. Nada en esta declaración hace que uno de estos bits sea más significativo que los demás. El orden en el que se listan no importa. Queremos comparar el valor actual de estos bits con cada una de las posibles combinaciones de entrada que podrían presentarse. Si trazáramos una tabla de verdad, decidiríamos cuál bit colocar a la izquierda (MSB) y cuál a la derecha (LSB). En VHDL esto se logra mediante la **concatenación** (conectar en orden) de las variables de bit para formar un vector de bits. El operador de concatenación es "&". Se declara una señal como BIT_VECTOR para recibir el conjunto ordenado de bits de entrada y se utiliza para comparar el valor de entrada con las literales de cadena encerradas entre comillas. A la salida (*y*) se le asigna (<=) un valor de bit ('0' o '1') cuando (WHEN) *bits_ent* contiene el valor que se lista entre comillas dobles.

VHDL es muy estricto en cuanto a la forma en que nos permite asignar y comparar objetos tales como señales, variables, constantes y literales. La salida *y* es un BIT, por lo que se le debe asignar un valor de '0' o de '1'. La señal *bits_ent* es un BIT_VECTOR de tres bits, por lo que debe compararse con un valor literal de cadena de tres bits. VHDL no permitirá que *bits_ent* (un BIT_VECTOR) se compare con un número hexadecimal tal como X "5", o con un número decimal tal como 3. Estas cantidades escalares serían válidas para la asignación o comparación con enteros.

EJEMPLO 4-33

Declare tres señales en VHDL que sean bits individuales de nombre *demasiado_caliente*, *demasiado_frio* y *muy_bien*. Combine (concatene) estos tres bits en una señal de tres bits llamada *estado_temp*, en donde lo caliente estará a la izquierda y lo frío a la derecha.

Solución

1. Declare primero las señales en la arquitectura (ARCHITECTURE).

```
SIGNAL  demasiado caliente, demasiado frio, muy bien  :BIT;
SIGNAL  estado temp  : BIT VECTOR (2 DOWNTO 0);
```

2. Escriba instrucciones de asignación concurrentes entre BEGIN y END.

```
estado temp <5 demasiado caliente & muy bien & demasiado frio;
```

PREGUNTAS DE REPASO

1. ¿Cómo concatenaría los tres bits *x*, *y* y *z* en un arreglo de tres bits llamado *omega*? Use AHDL o VHDL.
2. ¿Cómo se implementan las tablas de verdad en AHDL?
3. ¿Cómo se implementan las tablas de verdad en VHDL?

4-17 ESTRUCTURAS DE CONTROL DE DECISIONES EN HDL

En esta sección examinaremos métodos que nos permiten indicar al sistema digital cómo realizar decisiones "lógicas", en forma muy similar al proceso que utilizamos para hacer decisiones en nuestra vida diaria. En el capítulo 3 vimos que las instrucciones de asignación concurrentes se evalúan de tal forma que el orden en el que se escriben no tiene efecto sobre el circuito que se está describiendo. Cuando utilizamos **estructuras de control de decisiones**, el orden en el que hacemos las preguntas sí es importante. Para resumir este concepto en los términos utilizados en la documentación del HDL, a las instrucciones que se pueden escribir en cualquier secuencia se les llama **concurrentes**, y a las instrucciones que se evalúan en la secuencia en la que se escriben se les llama **secuenciales**. La secuencia de las instrucciones secuenciales afecta a la operación del circuito.

Los ejemplos que hemos visto hasta ahora implican el uso de varios bits individuales. Muchos sistemas digitales requieren entradas que representan un valor numérico. Consulte de nuevo el ejemplo 4-8, en el cual el objetivo del circuito lógico es monitorear el voltaje de la batería mediante un convertidor A/D. El valor digital se representa mediante un número de cuatro bits que proviene del convertidor A/D y que pasa al circuito lógico. Estas entradas no son variables binarias independientes, sino cuatro dígitos binarios de un número que representa el voltaje de la batería. Necesitamos dar a estos datos el tipo correcto que nos permita utilizarlos como un número.

IF/ELSE

Las tablas de verdad son estupendas para listar todas las posibles combinaciones de variables independientes, pero hay mejores formas de manejar los datos numéricos. Como ejemplo, cuando una persona parte hacia la escuela o su trabajo en la mañana, debe realizar una decisión lógica para saber si se va a llevar o no un abrigo. Supongamos que esta persona decide sobre esta cuestión basándose únicamente en la temperatura actual. ¿Cuántos de nosotros razonaríamos de la siguiente manera?

Utilizaré un abrigo si la temperatura es 0.
Utilizaré un abrigo si la temperatura es 1.
Utilizaré un abrigo si la temperatura es 2....
Utilizaré un abrigo si la temperatura es 13.

No utilizaré un abrigo si la temperatura es 14.
No utilizaré un abrigo si la temperatura es 15.
No utilizaré un abrigo si la temperatura es 16….
No utilizaré un abrigo si la temperatura es 40.

Este método es similar a aquél en el que se utiliza la tabla de verdad para describir la decisión. Para cada una de las posibles entradas, esta persona decide cuál debe ser la salida. En realidad, lo que haría para decidirse sería lo siguiente:

Usaré un abrigo si la temperatura es menor de 14 grados.
En caso contrario, *no* utilizaré un abrigo.

Un HDL nos permite describir los circuitos lógicos mediante este tipo de razonamiento. Primero debemos describir las entradas como un *número dentro de un intervalo dado*, y después podemos escribir instrucciones que decidan lo que se debe hacer en las salidas con base en el *valor* del número entrante. Al igual que en los HDLs, en la mayoría de los lenguajes de programación de computadoras estos tipos de decisiones se llevan a cabo mediante el uso de una estructura de control IF/THEN/ELSE. Cada vez que la decisión está entre realizar una acción o no realizarla, se utiliza la instrucción **IF/THEN**. La palabra clave IF va seguida de una instrucción que es verdadera o falsa. Si (IF) es verdadera, entonces (THEN) se hace lo que esté especificado. En el caso en el que la instrucción sea falsa, no se realiza ninguna acción. La figura 4-52(a) muestra en forma gráfica la manera como funciona esta decisión. La figura de diamante representa la decisión que se está llevando a cabo mediante la evaluación de la instrucción contenida dentro del diamante. Toda decisión tiene dos posibles resultados: verdadero o falso. En este ejemplo, si la instrucción es falsa no se realiza ninguna acción.

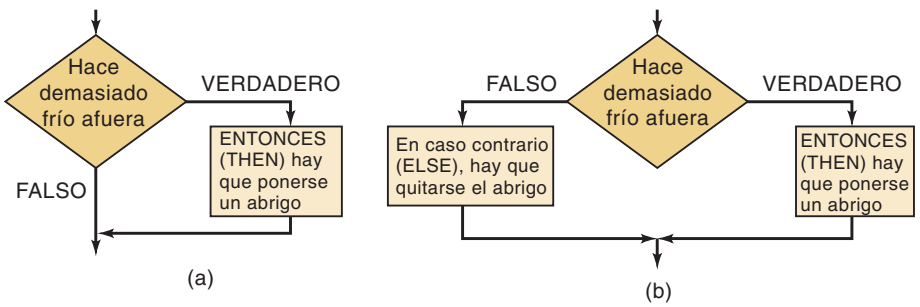


FIGURA 4-52
Flujo lógico de las instrucciones (a) IF/THEN y (b) IF/THEN/ELSE.

En algunos casos no sólo basta con decidir entre actuar y no actuar, sino que debemos elegir entre dos acciones distintas. Por ejemplo, en nuestra analogía sobre la decisión de usar un abrigo, si la persona ya tiene su abrigo puesto cuando llegue a esta decisión, no se lo quitará. El uso de la lógica IF/THEN supone que de inicio no está usando su abrigo.

Cuando las decisiones demandan dos posibles acciones, se utiliza la estructura de control IF/THEN/**ELSE** como se muestra en la figura 4-52(b). Aquí se evalúa de nuevo la instrucción como verdadera o falsa. La diferencia es que, cuando la instrucción es falsa se realiza una acción distinta. Debe realizarse una de las dos acciones con esta instrucción. Podemos describirla en forma verbal como, "SI (IF) la instrucción es verdadera, ENTONCES (THEN) hay que hacer esto. EN CASO CONTRARIO (ELSE) hay que hacer esto otro". En nuestra analogía del abrigo esta estructura de control funcionaría, sin importar que la persona haya traído puesto su abrigo o no desde un principio.

En el ejemplo 4-8 se mostró un ejemplo simple de un circuito lógico que tiene un valor numérico como entrada, el cual representa el voltaje de la batería de un convertidor A/D. Las entradas *A*, *B*, *C*, *D* son en realidad dígitos binarios en un número de 4 bits, en donde *A* es el MSB y *D* es el LSB. La figura 4-53 muestra el mismo cir-

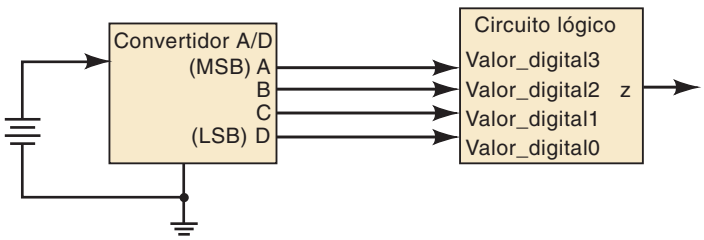


FIGURA 4-53 Circuito lógico similar al del ejemplo 4-8.

cuito con las entradas etiquetadas en forma de un número de cuatro bits, de nombre *valor_digital*. La relación entre los bits es la siguiente:

A	*valor_digital[3]*	bit 3 del valor digital (MSB)
B	*valor_digital[2]*	bit 2 del valor digital
C	*valor_digital[1]*	bit 1 del valor digital
D	*valor_digital[0]*	bit 0 del valor digital (LSB)

La entrada puede considerarse como un número decimal entre 0 y 15, si especificamos el tipo correcto de la variable de entrada.

AHDL

IF/THEN/ELSE MEDIANTE EL USO DE AHDL

En AHDL, las entradas pueden especificarse como un número binario formado de varios bits mediante la asignación de un nombre de variable, seguido de una lista de las posiciones de los bits, como se muestra en la figura 4-54. El nombre es *valor_digital* y las posiciones de los bits varían desde 3 hasta 0. Observe lo simple que se vuelve el código al utilizar este método junto con una instrucción IF/ELSE. La palabra clave IF va seguida de una instrucción que hace referencia al valor completo de la variable de entrada de cuatro bits y lo compara con el número 6. Desde luego que 6 es la forma decimal de una cantidad escalar y *valor_digital[ ]* representa en realidad a un número binario. Como el compilador puede interpretar números en cualquier sistema, crea un circuito lógico que compara el valor binario de *valor_digital* con el número binario correspondiente al 6 decimal y decide si esta instrucción es verdadera o falsa. Si es verdadera, ENTONCES (THEN) se utiliza la siguiente instrucción (z = VCC) para asignar un valor a z. Observe que en AHDL debemos usar VCC para un 1 lógico y GND para un 0 lógico cuando asignamos un nivel lógico a un bit individual. Cuando *valor_digital* es 6 o menos, va después de la instrucción que sigue de ELSE (z = GND). La instrucción END IF termina la estructura de control.

FIGURA 4-54 Versión en AHDL.

```
SUBDESIGN FIG4 54
(
   valor digital[3..0]  :INPUT;     define las entradas del bloque
   z                    :OUTPUT;    define la salida del bloque
)
BEGIN
   IF valor digital[ ] > 6 THEN
         z = VCC;                    la salida es 1
   ELSE  z = GND;                    la salida es 0
   END IF;
END;
```

IF/THEN/ELSE MEDIANTE EL USO DE VHDL

En VHDL la cuestión importante es la declaración del tipo de entradas. (Consulte la figura 4-55.) La entrada se trata como una variable individual llamada *valor_digital*. Como su tipo se declara como INTEGER, el compilador sabe que debe tratarla como un número. Al especificar un intervalo de 0 a 15, el compilador sabe que es un número de cuatro bits. Observe que RANGE no especifica el número de índice de un vector de bits, sino los límites del valor numérico del entero. En VHDL los enteros se tratan de manera distinta a los arreglos de bits (BIT_VECTOR). Un entero puede compararse con otros números mediante el uso de operadores de desigualdad. Un BIT_VECTOR no puede usarse con operadores de desigualdad.

FIGURA 4-55
Versión en VHDL.

```
ENTITY fig4 55 IS
PORT(  valor digital :IN INTEGER RANGE 0 TO 15;    entrada de 4 bits
       z             :OUT BIT);
END fig4 55;

ARCHITECTURE decision OF fig4 55 IS

BEGIN
   PROCESS (valor digital)
      BEGIN
         IF (valor digital > 6) THEN
            z <= '1';
         ELSE
            z <= '0';
      END IF;
END PROCESS ;
END decision;
```

Para utilizar la estructura de control IF/THEN/ELSE, VHDL requiere que el código se coloque dentro de un proceso (PROCESS). Las instrucciones que ocurren dentro de un proceso son *secuenciales*, lo cual significa que el orden en el que se escriben afecta a la operación del circuito. La palabra clave **PROCESS** va seguida de una lista de variables, a la cual se le llama **lista de sensibilidad**, que es una lista de variables a las cuales el código del proceso debe responder. Cada vez que *valor_digital* cambia, hace que se vuelva a evaluar el código del proceso. Aún y cuando sabemos que *valor_digital* es en realidad un número binario de cuatro bits, el compilador lo evaluará como un número entre los valores decimales equivalentes de 0 y 15. Si (IF) la instrucción entre paréntesis es verdadera, entonces (THEN) se aplica la siguiente instrucción (a *z* se le asigna el valor de 1 lógico). Si esta instrucción no es verdadera, la lógica sigue la cláusula ELSE y asigna a z un valor de 0. La instrucción END IF; termina la estructura de control y END PROCESS; termina la evaluación de las instrucciones secuenciales.

ELSIF

A menudo tenemos que elegir de entre muchas posibles acciones, dependiendo de la situación. La instrucción IF decide si se va a realizar o no un conjunto de acciones. La instrucción IF/ELSE selecciona una de dos posibles acciones. Mediante la combinación de las decisiones con IF y ELSE podemos crear una estructura de control que

se conoce como **ELSIF**, la cual selecciona uno de muchos resultados posibles. En la figura 4-56 se muestra la estructura de decisión en forma gráfica.

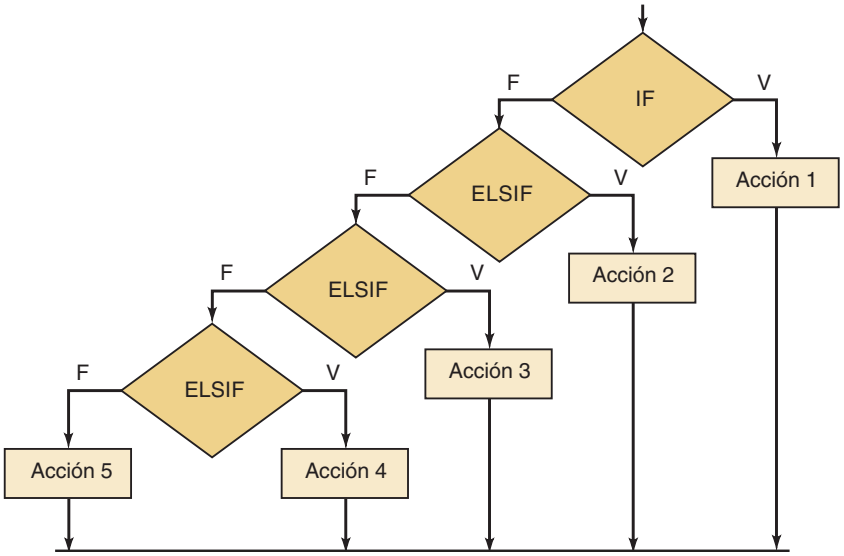


FIGURA 4-56
Diagrama de flujo para decisiones múltiples en las que se utiliza IF/ELSIF.

Observe que, a medida que se evalúa cada condición, se realiza una acción si es verdadera o en caso contrario pasa a evaluar la siguiente condición. Cada acción está asociada con una condición y no existe la probabilidad de seleccionar más de una acción. Observe también que las condiciones que se utilizan para decidir la acción apropiada pueden ser cualquier expresión que se evalúe como verdadera o como falsa. Este hecho permite al diseñador utilizar los operadores de desigualdad para seleccionar una acción con base en un intervalo de valores de entrada. Como ejemplo de esta aplicación, consideremos el sistema de medición de temperatura que utiliza un convertidor A/D, como se describe en la figura 4-57. Suponga que deseamos indicar cuando la temperatura se encuentra en cierto Intervalo, al cual denominaremos como Demasiado frío, Templado y Demasiado caliente.

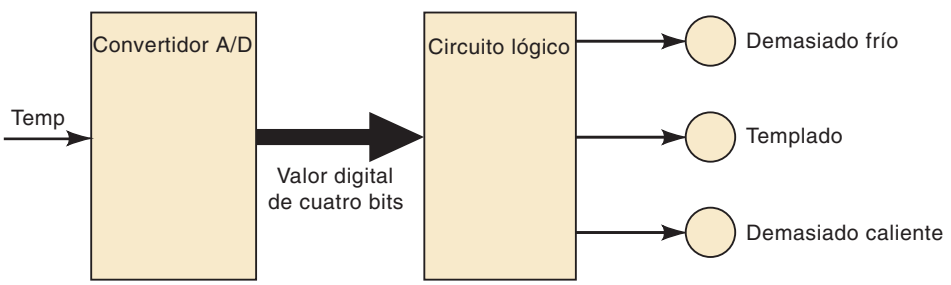


FIGURA 4-57
Circuito indicador de un intervalo de temperatura.

La relación entre los valores digitales para la temperatura y las categorías es

Valores digitales	*Categoría*
0000-1000	Demasiado frío
1001-1010	Templado
1011-1111	Demasiado caliente

Podemos expresar el proceso de toma de decisiones para este circuito lógico de la siguiente manera:

> Si (IF) el valor digital es menor o igual a 8, entonces (THEN) hay que encender sólo el indicador "Demasiado frío".
>
> En caso contrario, si (ELSE IF) el valor digital es mayor que 8 y (AND) menor que 11, entonces (THEN) hay que encender sólo el indicador "Templado".
>
> En caso contrario (ELSE) hay que encender sólo el indicador "Demasiado caliente".

AHDL

ELSIF MEDIANTE EL USO DE AHDL

El código de AHDL de la figura 4-58 define las entradas en forma de un número binario de cuatro bits. Las salidas son tres bits individuales que controlan los tres indicadores de Intervalo. En este ejemplo se utiliza una variable intermedia (*estado*), la cual nos permite asignar un patrón de bits que representa las tres condiciones de *demasiado_frio*, *templado* y *demasiado_caliente*. La sección secuencial del código utiliza las instrucciones IF, ELSIF, ELSE para identificar el intervalo en el cual se encuentra la temperatura y asigna a *estado* el patrón de bits correcto. En la última instrucción, los bits de *estado* se conectan a los bits del puerto de salida real. Estos bits están ordenados en un grupo que se relaciona con los patrones de bits asignados a *estado[ ]*. Esto también podría haberse escrito mediante tres instrucciones concurrentes: demasiado_frio = estado[2]; templado = estado[1]; demasiado_caliente = estado[0];

```
SUBDESIGN fig4 58
(
   valor digital[3..0]           :INPUT;    define las entradas del bloque
   demasiado frio, templado, demasiado caliente :OUTPUT;    define las salidas
)
VARIABLE
estado[2..0] :NODE;   guarda el estado de demasiado frio, templado, demasiado caliente
BEGIN
   IF       valor digital[] <= 8 THEN estado[] = b"100";
   ELSIF    valor digital[] > 8 AND valor digital[] < 11 THEN
         estado[] = b"010";
   ELSE  estado[] = b"001";
   END IF;
   (demasiado frio, templado, demasiado caliente) 5 estado[ ];    actualiza los bits de
    salida
END;
```

FIGURA 4-58 Ejemplo de Intervalos de temperatura en AHDL mediante el uso de ELSIF.

VHDL

ELSIF MEDIANTE EL USO DE VHDL

El código de VHDL en la figura 4-59 define las entradas como un entero de cuatro bits. Las salidas son tres bits individuales que controlan los tres indicadores de Intervalo. En este ejemplo se utiliza una señal intermedia (*estado*), la cual nos permite asignar un patrón de bits que representa a las tres condiciones *demasiado_frio*, *muy_bien* y *demasiado_caliente*. La sección de proceso del código utiliza las instrucciones IF, ELSIF y ELSE para identificar el intervalo en el cual se encuentra la temperatura y asigna a *estado* el patrón de bits correcto. En las últimas tres instrucciones, cada bit de *estado* se conecta al bit del puerto de salida correspondiente.

```
ENTITY fig4 59  IS
PORT(valor digital: IN INTEGER RANGE 0 TO 15;       declara la entrada de 4 bits
      demasiado frio, templado, demasiado caliente : OUT BIT);
END fig4 59 ;

ARCHITECTURE quetancaliente  OF fig4 59 IS
SIGNAL estado   :BIT VECTOR (2 downto 0);
BEGIN
   PROCESS (valor digital)
      BEGIN
         IF (valor digital <5 8) THEN  estado <5 “100”;
         ELSIF (valor digital > 8 AND valor digital < 11) THEN
               estado <= "010";
         ELSE  estado <= "001";
         END IF;
      END PROCESS ;
   demasiado frio    <= estado(2);        asigna los bits de estado a la salida
   templado  <= estado(1);
   demasiado caliente     <= estado(0);
END quetancaliente;
```

FIGURA 4-59 Ejemplo de Intervalos de temperatura en VHDL mediante el uso de ELSIF.

CASE

Existe otra estructura de control que es útil para elegir acciones con base en las condiciones actuales. Tiene varios nombres dependiendo del lenguaje de programación, pero casi siempre se utiliza la palabra **CASE**. Esta instrucción determina el valor de una expresión u objeto y después analiza una lista de posibles valores (casos) para la expresión u objeto que se está evaluando. Cada caso tiene una lista de acciones que deben llevarse a cabo. Una instrucción CASE es distinta a una instrucción IF/ELSIF debido a que un caso correlaciona un valor único de un objeto con un conjunto de acciones. Recuerde que una instrucción IF/ELSIF correlaciona un conjunto de acciones con una aseveración verdadera. Sólo puede haber una coincidencia para una instrucción CASE, mientras que una instrucción IF/ELSIF puede tener más de una, pero entonces (THEN) realizará solo la acción asociada con la primera aseveración verdadera que evalúe.

Otro punto importante en los ejemplos que siguen a continuación es la necesidad de combinar cierto número de variables independientes en un conjunto de bits, al cual se le conoce como vector de bits. Recuerde que a esta acción de enlazar varios bits en un orden específico se le conoce como *concatenación* y nos permite considerar el patrón de bits como un grupo ordenado.

AHDL

CASE MEDIANTE EL USO DE AHDL

El ejemplo de AHDL en la figura 4-60 muestra el uso de una instrucción CASE para implementar el circuito de la figura 4-9 (vea también la tabla 4-3). Utiliza bits individuales como sus entradas. En la primera instrucción después de BEGIN, estos bits se concatenan y se asignan a la variable intermedia llamada *estado*. La instrucción CASE evalúa a la variable *estado* y busca el patrón de bits (que sigue después de la palabra clave WHEN) que concuerde con el valor de *estado*. Después realiza la acción descrita después del signo =>. En este ejemplo, sólo asigna un 0 lógico a la salida para cada uno de los tres casos especificados. Los *demás* casos producen un 1 lógico en la salida.

FIGURA 4-60
Representación de la figura 4-9 en AHDL.

```
SUBDESIGN fig4 60
(
   p, q, r          :INPUT;          define las entradas del bloque
   s                :OUTPUT;         define las salidas
)
VARIABLE
  estado[2..0]     :NODE;
BEGIN
   estado[]= (p, q, r);      enlaza los bits de entrada en orden
   CASE estado[] IS
      WHEN b"100"       => s = GND;
      WHEN b"101"       => s = GND;
      WHEN b"110"       => s = GND;
      WHEN OTHERS       => s = VCC;
   END CASE;
END;
```

CASE MEDIANTE EL USO DE VHDL

El ejemplo de VHDL en la figura 4-61 demuestra el uso de la instrucción CASE para implementar el circuito de la figura 4-9 (vea también la tabla 4-3). Utiliza bits individuales como entradas. En la primera instrucción después de BEGIN, estos bits se concatenan y se asignan a la variable intermedia llamada *estado* mediante el uso del operador &. La instrucción CASE evalúa la variable *estado* y busca el patrón de bits (que va después de la palabra clave WHEN) que concuerde con el valor de *estado*. Después realiza la acción descrita después del signo =>. En este ejemplo simple, sólo asigna un 0 lógico a la salida para cada uno de los tres casos especificados. Los *demás* casos producen un 1 lógico en la salida.

FIGURA 4-61
Representación de la figura 4-9 en VHDL.

```
ENTITY fig4 61 IS
PORT( p, q, r      :IN bit;              declara 3 bits de entrada
      s            :OUT BIT);
END fig4 61;

ARCHITECTURE copia OF fig4 61 IS
SIGNAL estado      :BIT VECTOR (2 downto 0);
BEGIN
   estado <= p & q & r;                 enlaza los bits en orden.
   PROCESS (estado)
      BEGIN
         CASE estado IS
            WHEN "100" => s <= '0';
            WHEN "101" => s <= '0';
            WHEN "110" => s <= '0';
            WHEN OTHERS =>s <= '1';
         END CASE;
      END PROCESS;
END copia;
```

EJEMPLO 4-34

Un detector en una máquina despachadora acepta monedas de 25, 10 y 5 centavos, y activa la señal digital correspondiente (*Q, D, N*) sólo cuando está presente la moneda correcta. Físicamente es imposible que haya varias monedas presentes al mismo tiempo. Un circuito digital debe utilizar las señales *Q*, *D* y *N* como entradas y debe producir un número binario que represente el valor de la moneda según se muestra en la figura 4-62. Escriba el código en AHDL y VHDL.

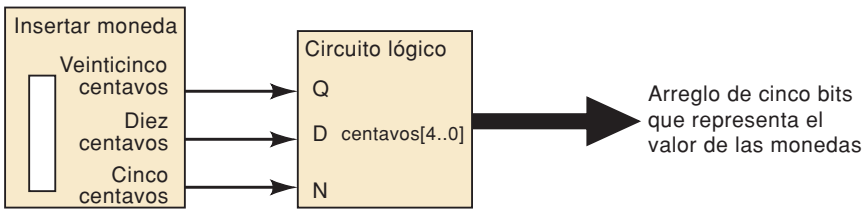


FIGURA 4-62 Un circuito detector de monedas para una máquina despachadora.

Solución

Ésta es una aplicación ideal de la instrucción CASE para describir la operación correcta. Las salidas deben declararse como números de cinco bits para poder representar hasta 25 centavos. La figura 4-63 muestra la solución en AHDL y la figura 4-64 la solución en VHDL.

AHDL

```
SUBDESIGN         fig4_63
(
      q, d, n         :INPUT;      -- define veinticinco, diez y cinco centavos
      centavos[4..0]  :OUTPUT;     -- define el valor binario de las monedas
)
BEGIN
      CASE   (q, d, n)  IS         --agrupa las monedas en un conjunto ordenado
        WHEN b"001" => centavos[ ] = 5;
        WHEN b"010" => centavos[ ] = 10;
        WHEN b"100" => centavos[ ] = 25;
        WHEN OTHERS => centavos[ ] = 0;
      END CASE;
END;
```

FIGURA 4-63 Un detector de monedas en AHDL.

VHDL

```
ENTITY  fig4_64  IS
PORT( q, d, n: IN BIT;                              --veinticinco, diez y cinco centavos
        centavos :OUT INTEGER RANGE 0 TO 25); -- valor binario de las monedas
END fig4_64;
ARCHITECTURE detector OF fig4_64 IS
      SIGNAL  monedas :BIT_VECTOR(2 DOWNTO 0); -- agrupa los sensores de monedas
      BEGIN
         monedas <=  (q & d & n);               --asigna sensores al grupo
         PROCESS  (monedas)
            BEGIN
               CASE  (centavos)  IS
                  WHEN "001"  => centavos <= 5;
                  WHEN "010"  => centavos <= 10;
                  WHEN "100"  => centavos <= 25;
                  WHEN OTHERS => centavos <= 0;
               END CASE;
            END PROCESS;
      END detector;
```

FIGURA 4-64 Un detector de monedas en VHDL.

PREGUNTAS DE REPASO

1. ¿Cuál estructura de control decide qué hacer o qué no hacer?
2. ¿Cuál estructura de control decide hacer esto o lo otro?
3. ¿Cuál(es) estructura(s) de control decide(n) la acción específica, entre varias, que se va a realizar?
4. Declare una entrada llamada *conteo* que pueda representar una cantidad numérica tan grande como 205. Use AHDL o VHDL.

RESUMEN

1. Las dos formas generales para las expresiones lógicas son la forma de suma de productos y la forma de producto de sumas.
2. Un método para el diseño de un circuito lógico combinacional es (1) construir su tabla de verdad, (2) convertir la tabla de verdad en una expresión de suma de productos, (3) simplificar la expresión mediante álgebra booleana o mapeo K, (4) implementar la expresión final.
3. El mapa K es un método gráfico para representar la tabla de verdad de un circuito y generar una expresión simplificada para la salida del circuito.
4. Un circuito OR exclusivo tiene la expresión $x = A\overline{B} + \overline{A}B$. Su salida x estará en ALTO sólo cuando las entradas A y B estén en niveles lógicos opuestos.
5. Un circuito NOR exclusivo tiene la expresión $x = \overline{A}\,\overline{B} + AB$. Su salida x estará en ALTO sólo cuando las entradas A y B estén en el mismo nivel lógico.
6. Cada una de las compuertas básicas (AND, OR, NAND, NOR) pueden usarse para habilitar o deshabilitar el paso de una señal de entrada hacia su salida.
7. Las principales familias de CIs digitales son las familias TTL y CMOS. Los CIs digitales están disponibles en una amplia gama de densidades (compuertas por chip), desde las funciones lógicas básicas hasta las de alta complejidad.
8. Para el diagnóstico de fallas básico se requiere (como mínimo) una comprensión de la operación del circuito, un conocimiento de los tipos de posibles fallas, un diagrama de conexiones del circuito lógico completo y una sonda lógica.
9. Un dispositivo lógico programable (PLD) es un CI que contiene un extenso número de compuertas lógicas cuyas interconexiones pueden ser programadas por el usuario para generar la relación lógica deseada entre las entradas y las salidas.
10. Para programar un PLD se necesita un sistema de desarrollo, el cual consiste de una computadora, software para desarrollo de PLDs y un dispositivo programador que se encarga de la programación física del chip PLD.
11. El sistema Altera permite técnicas convenientes de diseño jerárquico mediante el uso de cualquier forma de descripción de hardware.
12. El tipo de los objetos de datos debe especificarse, de manera que el compilador del HDL conozca el intervalo de números que van a representarse.
13. Las tablas de verdad pueden introducirse de manera directa en el archivo fuente mediante el uso de las características del HDL.
14. Pueden utilizarse las estructuras de control lógicas tales como IF, ELSE y CASE para describir la operación de un circuito lógico, con lo cual se simplifica aún más el código y la solución al problema.

TÉRMINOS IMPORTANTES

agrupamiento
arreglo de bits
arriba-abajo
asignación de señal selecta
bibliotecas
BIT_VECTOR
CASE
cero esfuerzo de inserción (ZIF)
colisión
comprobación de paridad
concatenación
concurrentes
condición de "no importa"
diseño jerárquico
ELSE
ELSIF
encapsulado dual en línea (DIP)
entero
estructuras de control de decisiones
flotante
generación de paridad
habilitar/deshabilitar
IF/THEN
Índice
indeterminado
JEDEC
JTAG
lista de sensibilidad
literales
lógica de transistor/-transistor (TTL)
macrofunción
mapa de Karnaugh (mapa K)
NOR exclusivo (XNOR)
objetos
OR exclusivo (XOR)
PROCESS
producto de sumas (POS)
programador
secuenciales
Semiconductor Metal-Óxido-complementario (CMOS)
sonda lógica
SSI, MSI, LSI, VLSI, ULSI, GSI
STD_LOGIC
STD_LOGIC_VECTOR
suma de productos (SOP)
vector de bits
vectores de prueba

PROBLEMAS

SECCIONES 4-2 Y 4-3

B 4-1.* Simplifique las siguientes expresiones mediante el uso del álgebra booleana.

(a) $x = ABC + \overline{A}C$
(b) $y = (Q + R)(\overline{Q} + \overline{R})$
(c) $w = ABC + A\overline{B}C + \overline{A}$
(d) $q = \overline{RST}(\overline{R + S + T})$
(e) $x = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}BC + ABC + A\,\overline{B}\,\overline{C} + A\overline{B}C$
(f) $z = (B + \overline{C})(\overline{B} + C) + \overline{\overline{A} + B + \overline{C}}$
(g) $y = (\overline{C + D}) + \overline{A}C\overline{D} + A\overline{B}\,\overline{C} + \overline{A}\,\overline{B}CD + AC\overline{D}$
(h) $x = AB(\overline{\overline{C}D}) + \overline{A}BD + \overline{B}\,\overline{C}\,\overline{D}$

B 4-2. Simplifique el circuito de la figura 4-65 mediante el uso del álgebra booleana.

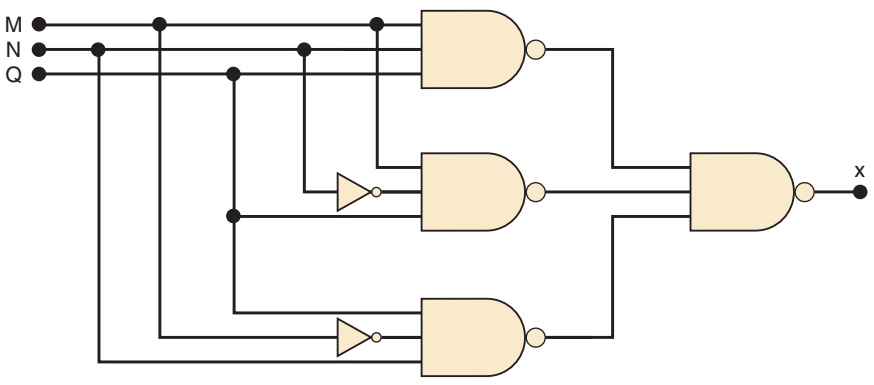


FIGURA 4-65
Problemas 4-2 y 4-3.

* Encontrará las respuestas a los problemas marcados con un asterisco al final del libro.

B 4-3.* Cambie cada una de las compuertas del problema 4-2 por compuertas NOR y simplifique el circuito mediante álgebra booleana.

SECCIÓN 4-4

B, D 4-4.* Diseñe el circuito lógico que corresponde a la tabla de verdad que se muestra en la tabla 4-11.

TABLA 4-11

A	*B*	*C*	*x*
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

B, D 4-5. Diseñe un circuito lógico cuya salida esté en ALTO *sólo* cuando la mayoría de las entradas *A*, *B* y *C* estén en BAJO.

D 4-6. Una planta de manufactura necesita tener un sonido de bocina para indicar la hora de salida. La bocina deberá activarse cuando se cumpla cualquiera de las siguientes condiciones:

1. Es después de las 5 en punto y todas las máquinas están apagadas.
2. Es viernes, se completó la producción del día y todas las máquinas están apagadas.

Diseñe un circuito lógico que controle la bocina. (*Sugerencia:* use cuatro variables lógicas de entrada para representar las diversas condiciones; por ejemplo, la entrada *A* estará en ALTO sólo cuando sean las 5 en punto o más tarde.)

D 4-7.* Un número binario de cuatro bits se representa como $A_3\,A_2\,A_1A_0$, en donde A_3, A_2, A_1 y A_0 representan los bits individuales y A_o es igual al LSB. Diseñe un circuito lógico que produzca una salida en ALTO cada vez que el número binario sea mayor que 0010 y menor que 1000.

D 4-8. La figura 4-66 muestra un diagrama para un circuito de alarma de automóvil que se utiliza para detectar ciertas condiciones indeseables. Los tres inte-

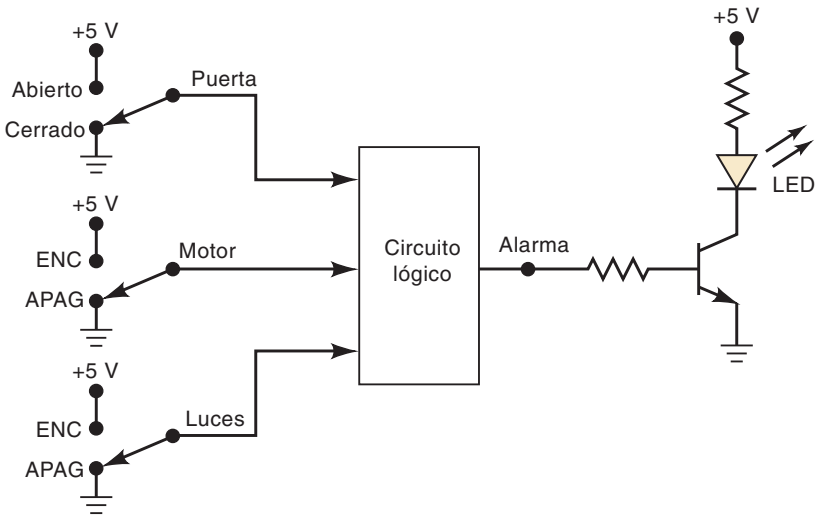


FIGURA 4-66
Problema 4-8.

rruptores se utilizan para indicar el estado de la puerta del lado del conductor, el motor y las luces, en forma respectiva. Diseñe el circuito lógico con estos tres interruptores como entradas, de manera que la alarma se active cada vez que exista cualquiera de las siguientes condiciones:

- Las luces estén encendidas mientras que el motor esté apagado.
- La puerta esté abierta mientras que el motor esté encendido.

4-9.* Implemente el circuito del problema 4-4, utilizando sólo compuertas NAND.

4-10. Implemente el circuito del problema 4-5, utilizando sólo compuertas NAND.

SECCIÓN 4-5

B 4-11. Determine la expresión mínima para cada uno de los mapas K en la figura 4-67. Ponga especial atención al paso 5 para el mapa en (a).

FIGURA 4-67
Problema 4-11.

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	1	1	1
$\bar{A}B$	1	1	0	0
AB	0	0	0	1
$A\bar{B}$	0	0	1	1

(a)*

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	0	1	1
$\bar{A}B$	1	0	0	1
AB	0	0	0	0
$A\bar{B}$	1	0	1	1

(b)

	$\bar{C}$	C
$\bar{A}\bar{B}$	1	1
$\bar{A}B$	0	0
AB	1	0
$A\bar{B}$	1	X

(c)

B 4-12. Para la tabla de verdad que se muestra a continuación, cree un mapa K de 2×2, agrupe los términos y simplifique. Después analice de nuevo la tabla de verdad para ver si la expresión es verdadera para todas las entradas en la tabla.

A	*B*	*y*
0	0	1
0	1	1
1	0	0
1	1	0

B 4-13. Empezando con la tabla de verdad en la tabla 4-11, utilice un mapa K para encontrar la ecuación SOP más simple.

B 4-14. Simplifique la expresión en (a)* el problema 4-1(e), usando un mapa K. (b) el problema 4-1(g), usando un mapa K. (c)* el problema 4-1(h), usando un mapa K.

B 4-15.* Obtenga la expresión de salida para el problema 4-7, usando un mapa K.

C,D 4-16. La figura 4-68 muestra un *contador BCD* que produce una salida de cuatro bits, la cual representa el código BCD para el número de pulsos que se han aplicado a la entrada del contador. Por ejemplo, después de haberse producido cuatro pulsos, las salidas del contador son $DCBA = 0100_2 = 4_{10}$. El contador se restablece a 0000 en el décimo pulso y empieza a contar de nuevo. En otras palabras, las salidas *DCBA* nunca representarán a un número mayor de $1001_2 = 9_{10}$.

(a)* Diseñe el circuito lógico que produzca una salida en ALTO cada vez que el conteo sea 2, 3 o 9. Use el mapeo K y aproveche las condiciones "no importa".

(b) Repita el proceso para $x = 1$ cuando $DCBA = 3, 4, 5, 8$.

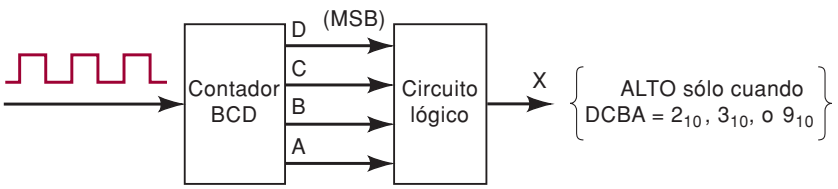


FIGURA 4-68
Problema 4-16.

D 4-17.* La figura 4-69 muestra cuatro interruptores que forman parte de los circuitos de control en una máquina copiadora. Los interruptores están en varios puntos a lo largo de la ruta del papel, a medida que éste pasa a través de la máquina. Cada interruptor está, por lo general, abierto, y a medida que el papel pasa a través de un interruptor, éste se cierra. Es imposible que los interruptores SW1 y SW4 estén cerrados al mismo tiempo. Diseñe el circuito lógico para producir una salida en ALTO cada vez que *dos o más* interruptores estén cerrados al mismo tiempo. Use el mapeo K y aproveche las condiciones "no importa".

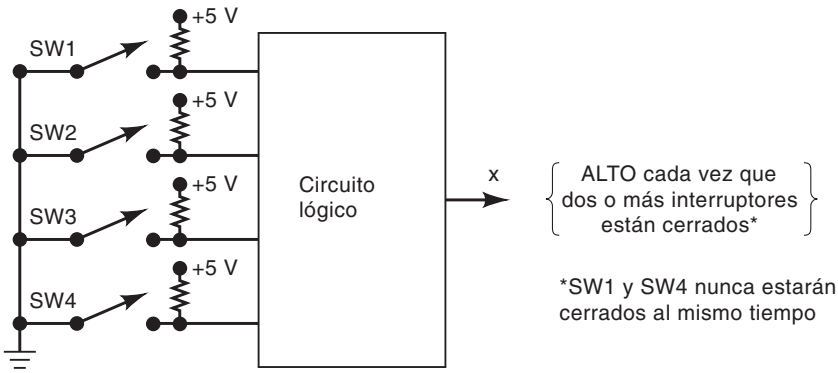


FIGURA 4-69
Problema 4-17.

B 4-18. El ejemplo 4-3 demostró la simplificación algebraica. El paso 3 produjo la ecuación SOP $z = \overline{A}\,\overline{B}\,C + \overline{A}\,C\,\overline{D} + \overline{A}\,B\,\overline{C}\,\overline{D} + A\overline{B}C$. Use un mapa K para demostrar que esta ecuación puede simplificarse aún más que la respuesta que se muestra en el ejemplo.

C 4-19. Utilice el álgebra booleana para llegar al mismo resultado que se obtuvo mediante el método del mapa K del problema 4-18.

SECCIÓN 4-6

B 4-20. (a) Determine la forma de onda de salida para el circuito de la figura 4-70.

(b) Repita el proceso con la entrada *B* mantenida en BAJO.

(c) Repita el proceso con la entrada *B* mantenida en ALTO.

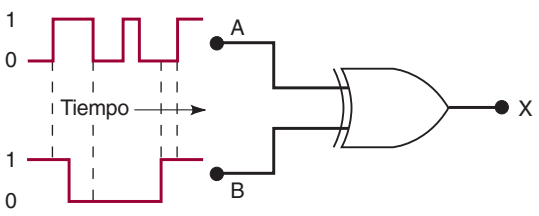
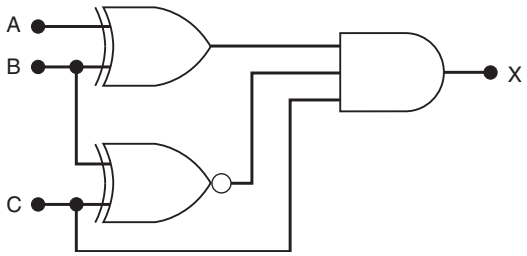


FIGURA 4-70 Problema 4-20.

B 4-21.* Determine las condiciones de entrada necesarias para producir $x = 1$ en la figura 4-71.

FIGURA 4-71 Problema 4-21.



B 4-22. Diseñe un circuito que produzca una salida en ALTO sólo cuando las tres entradas estén en el mismo nivel.

(a) Use una tabla de verdad y un mapa K para producir la solución SOP.

(b) Use compuertas XOR de dos entradas y otras compuertas para encontrar una solución. (*Sugerencia:* recuerde la propiedad transitiva del álgebra… si $a = b$ y $b = c$ entonces $a = c$.)

B 4-23.* Un chip 7486 contiene cuatro compuertas XOR. Muestre cómo hacer una compuerta XNOR utilizando sólo un chip 7486. *Sugerencia:* vea el ejemplo 4-16.

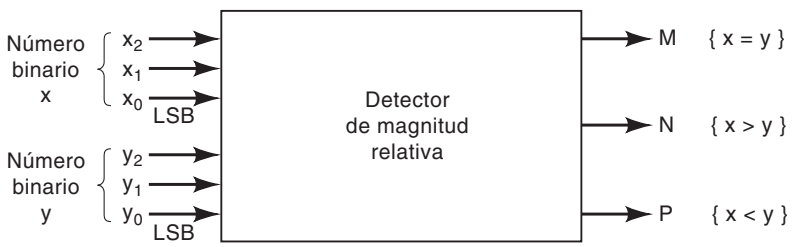
B 4-24.* Modifique el circuito de la figura 4-23 para comparar dos números de cuatro bits y producir una salida en ALTO cuando los dos números concuerden de manera exacta.

B 4-25. La figura 4-72 representa un *detector de magnitud relativa* que toma dos números binarios de tres bits ($x_2x_1x_0$ y $y_2y_1y_0$) y determine si son iguales; en caso de no ser así, que determine cuál es más grande. Hay tres salidas, que se definen de la siguiente manera:

1. $M = 1$ sólo si los dos números de entrada son iguales.
2. $N = 1$ sólo si $x_2x_1x_0$ es mayor que $y_2y_1y_0$.
3. $P = 1$ sólo si $y_2y_1y_0$ es mayor que $x_2x_1x_0$.

Diseñe los circuitos lógicos para este detector. El circuito tiene *seis* entradas y *tres* salidas, por lo que es demasiado complejo como para manejarlo mediante el método de la tabla de verdad. Consulte el ejemplo 4-17 como una sugerencia sobre cómo podría empezar a resolver este problema.

FIGURA 4-72
Problema 4-25.



MÁS PROBLEMAS DE DISEÑO

C,D 4-26.* La figura 4-73 representa un circuito multiplicador que toma dos números binarios de dos bits (x_1x_0 y y_1y_0) y produce un número binario de salida $z_3z_2z_1z_0$ que es igual al producto aritmético de los dos números de entrada. Diseñe el circuito lógico para el multiplicador. (*Sugerencia:* el circuito lógico tendrá cuatro entradas y cuatro salidas.)

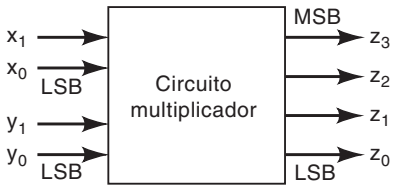


FIGURA 4-73 Problema 4-26.

D 4-27. Un código BCD se está transmitiendo a un receptor remoto. Los bits son A_3, A_2, A_1 y A_0, en donde A_3 es el MSB. Entre los circuitos del receptor incluye un circuito *detector de errores BCD*, el cual examina el código recibido para ver si es un código de BCD legal (es decir, ≤1001). Diseñe este circuito para producir un nivel ALTO para cualquier condición de error.

D 4-28.* Diseñe un circuito lógico cuya salida esté en ALTO cada vez que *A* y *B* estén ambas en ALTO, siempre y cuando *C* y *D* estén ambas en BAJO o ambas en ALTO. Trate de hacer esto sin utilizar una tabla de verdad. Después compruebe su resultado construyendo una tabla de verdad a partir de su circuito, para ver si concuerda con la declaración del problema.

D 4-29. Cuatro tanques grandes en una planta química que contienen distintos líquidos se están calentando. Se utilizan sensores de nivel de líquido para detectar cuando el tanque *A* o el tanque *B* se eleva por encima de un nivel predeterminado. Los sensores de temperatura en los tanques *C* y *D* detectan cuando la temperatura en cualquiera de estos tanques cae por debajo de un límite prescrito. Suponga que las salidas *A* y *B* del sensor de nivel de líquido están en BAJO cuando el nivel es satisfactorio y en ALTO cuando el nivel es demasiado alto. Además, las salidas *C* y *D* del sensor de temperatura están en BAJO cuando la temperatura es satisfactoria y en ALTO cuando la temperatura es demasiado baja. Diseñe un circuito lógico que detecte cada vez que el nivel en el tanque *A* o en el tanque *B* es demasiado alto, al mismo tiempo que la temperatura en el tanque *C* o en el tanque *D* sea demasiado baja.

C,D 4-30.* La figura 4-74 muestra la intersección de una autopista principal con un camino de acceso secundario. Se colocaron sensores de detección de vehículos a lo largo de los carriles *C* y *D* (camino principal) y de los carriles *A* y *B* (camino de acceso). Las salidas de estos sensores están en BAJO (0) cuando

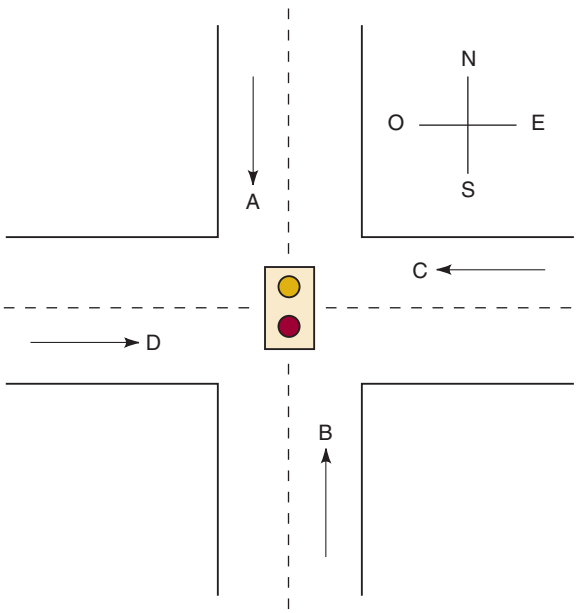


FIGURA 4-74
Problema 4-30.

no hay vehículos presentes, y en ALTO (1) cuando hay vehículos presentes. El semáforo de la intersección debe controlarse de acuerdo con la siguiente lógica:

1. El semáforo este-oeste (E-O) se pondrá en verde cada vez que estén ocupados ambos carriles *C* y *D*.
2. El semáforo E-O estará en verde cada vez que *C o D* estén ocupados, pero cuando *A* y *B* no estén ambos ocupados.
3. El semáforo norte-sur (N-S) se pondrá en verde cada vez que *ambos* carriles *A* y *B* estén ocupados, pero cuando *C* y *D* no estén *ambos* ocupados.
4. El semáforo N-S también se pondrá en verde cuando *A* o *B* estén ocupados, mientras que *C* y *D* estén *ambos* vacantes.
5. El semáforo E-O cuando *no* haya vehículos presentes.

Utilizando las salidas del sensor *A*, *B*, *C* y *D* como entradas, diseñe un circuito lógico para controlar el semáforo. Debe haber dos salidas, N-S y E-O, que cambien a ALTO cuando la luz correspondiente se vaya a poner en *verde*. Simplifique el circuito lo más que se pueda y muestre *todos* los pasos.

SECCIÓN 4-7

D 4-31. Rediseñe el generador y comprobador de paridad de la figura 4-25 para que (a) opere usando paridad impar. (*Sugerencia:* ¿cuál es la relación entre un bit de paridad impar y un bit de paridad par para el mismo conjunto de bits de datos?) (b) Opere con ocho bits de datos.

SECCIÓN 4-8

B 4-32. (a) ¿Bajo qué condiciones permitirá una compuerta OR que una señal lógica pase hacia su salida sin modificarla?

(b) Repita el inciso (a) para una compuerta AND.

(c) Repita el proceso para una compuerta NAND.

(d) Repita el proceso para una compuerta NOR.

B 4-33.* (a) ¿Puede utilizarse un INVERSOR como un circuito de habilitación/deshabilitación? Explique.

(b) ¿Puede usarse una compuerta XOR como un circuito de habilitación/deshabilitación? Explique.

D 4-34. Diseñe un circuito lógico que permita que la señal de entrada *A* pase hasta la salida sólo cuando la entrada de control *B* esté en BAJO, mientras que la entrada de control *C* esté en ALTO; en caso contrario, la salida debe estar en BAJO.

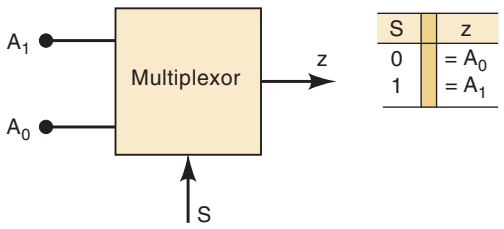
D 4-35.* Diseñe un circuito que *deshabilite* el paso de una señal de entrada sólo cuando las entradas de control *B*, *C* y *D* estén todas en ALTO; la salida deberá estar en ALTO para la condición deshabilitada.

D 4-36. Diseñe un circuito lógico que controle el paso de la señal *A*, de acuerdo con los siguientes requerimientos:

1. La salida *X* será igual a *A* cuando las entradas de control *B* y *C* sean iguales.
2. *X* permanecerá en ALTO cuando *B* y *C* sean distintas.

D 4-37. Diseñe un circuito lógico que tenga dos señales de entrada A_1 y A_0, y una entrada de control *S* de manera que funcione de acuerdo con los requerimientos descritos en la figura 4-75. (A este tipo de circuito se le conoce como *multiplexor*, el cual veremos en el capítulo 9.)

FIGURA 4-75 Problema 4-37.



D 4-38.* Use el mapeo K para diseñar un circuito que cumpla con los requerimientos del ejemplo 4-17. Compare este circuito con la solución en la figura 4-23. Aquí se recalca que el método del mapa K no puede aprovechar la lógica de las compuertas XOR y XNOR. El diseñador debe ser capaz de determinar cuándo pueden utilizarse estas compuertas.

SECCIONES 4-9 A 4-13

F 4-39. (a) Un técnico que está probando un circuito lógico se da cuenta de que la salida de cierto INVERSOR está atascada en BAJO mientras que su entrada esté pulsando. Liste todas las posibles razones que pueda para esta operación defectuosa.

(b) Repita el inciso (a) para el caso en el que la salida del INVERSOR se quede atascada en un nivel lógico indeterminado.

F 4-40.* Las señales que se muestran en la figura 4-76 se aplican a las entradas del circuito de la figura 4-32. Suponga que hay un circuito abierto interno en Z1-4.

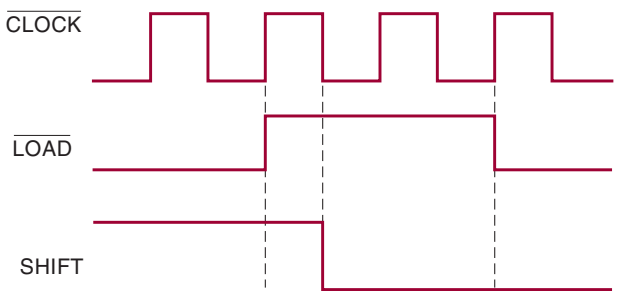
(a) ¿Qué indicará una sonda lógica en Z1-4?

(b) ¿Qué lectura de voltaje de corriente directa esperaría en Z1-4? (Recuerde que los CIs son TTL.)

(c) Haga un bosquejo de cuál cree usted que será la apariencia de las señales $\overline{CLKOUT}$ y $\overline{SHIFTOUT}$.

(d) En vez del circuito abierto en Z1-4, suponga que las terminales 9 y 10 de Z2 están en corto interno. Haga un bosquejo de las probables señales en Z2-10, $\overline{CLOCKOUT}$ y $\overline{SHIFTOUT}$.

FIGURA 4-76 Problema 4-40.



F 4-41. Suponga que los CIs de la figura 4-32 son CMOS. Describa cómo se vería afectada la operación del circuito debido a un circuito abierto en el conductor que conecta a Z2-2 y Z2-10.

F 4-42. En el ejemplo 4-24 listamos tres posibles fallas para la situación de la figura 4-36. ¿Qué procedimiento seguiría usted para determinar cuál de las fallas es la que se está produciendo en realidad?

F 4-43.* Consulte el circuito de la figura 4-38. Suponga que los dispositivos son CMOS. Suponga además que la indicación de la sonda lógica en Z2-3 es

* Recuerde que F indica un ejercicio de diagnóstico de fallas.

"indeterminado", en vez de "pulsando". Liste las posibles fallas y escriba un procedimiento a seguir para determinar la verdadera falla?

F 4-44.* Consulte el circuito lógico de la figura 4-41. Recuerde que se supone que la salida Y debe estar en ALTO para cualquiera de las siguientes condiciones:

1. $A = 1, B = 0$, sin importar C
2. $A = 0, B = 1, C = 1$

Al probar el circuito, el técnico observa que Y cambia a ALTO sólo para la primera condición, pero permanece en BAJO para todas las demás condiciones de entrada. Considere la siguiente lista de posibles fallas. Para cada una de ellas, escriba sí o no para indicar si podría o no ser la verdadera falla. Explique su razonamiento para cada una de las opciones en las que conteste que no.

(a) Un corto interno a tierra en Z2-13.
(b) Un circuito abierto en la conexión a Z2-13.
(c) Un corto interno con V_{CC} en Z2-11.
(d) Un circuito abierto en la conexión de V_{CC} con Z2.
(e) Un circuito abierto interno en Z2-9.
(f) Un circuito abierto en la conexión de Z2-11 a Z2-9.
(g) Un puente de soldadura entre las terminales 6 y 7 de Z2.

F 4-45. Desarrolle un procedimiento para aislar la falla que esté produciendo el funcionamiento defectuoso descrito en el problema 4-44.

F 4-46.* Suponga que todas las compuertas en la figura 4-41 son CMOS. Cuando el técnico prueba el circuito descubre que opera en forma correcta, excepto para las siguientes condiciones:

1. $A = 1, B = 0, C = 0$
2. $A = 0, B = 1, C = 1$

Para estas condiciones, la sonda lógica indica niveles indeterminados en Z2-6, Z2-11 y Z2-8. ¿Cuál cree usted que sea la probable falla en el circuito? Explique su razonamiento.

F 4-47. La figura 4-77 es un circuito lógico combinacional que opera una alarma en un automóvil, cada vez que están ocupados los asientos del conductor y/o del pasajero y que los cinturones de seguridad no están abrochados cuando el automóvil arranca. Las señales *CONDUC* y *PASAJ* activas en ALTO indican la presencia del conductor y del pasajero en forma respectiva, y se obtienen mediante interruptores operados por presión en los asientos. La señal *ENC* es activa en ALTO cuando el interruptor de encendido está activado. La señal $\overline{CINTC}$ es activa en BAJO e indica que el cinturón de seguridad

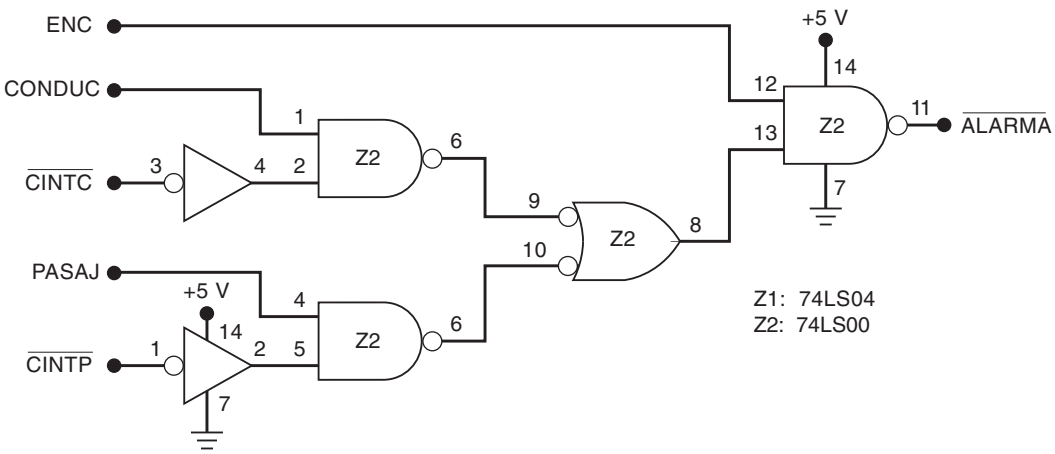


FIGURA 4-77 Problemas 4-47, 4-48 y 4-49.

del conductor *no está abrochado*; $\overline{CINTP}$ es la señal correspondiente para el cinturón de seguridad del pasajero. La alarma se activará (BAJO) cada vez que se encienda el automóvil, que cualquiera de los asientos delanteros esté ocupado y que su cinturón de seguridad no esté abrochado.

(a) Verifique que el circuito funcione de la manera descrita.

(b) Describa cómo operaría este sistema de alarma si Z1-2 estuviera en corto interno con tierra.

(c) Describa cómo operaría si hubiera una conexión abierta de Z2-6 a Z2-10.

F 4-48.* Suponga que el sistema de la figura 4-77 está funcionando de manera que la alarma se active tan pronto como el conductor o algún pasajero estén sentados y el automóvil esté encendido, sin importar el estado de los cinturones de seguridad. ¿Cuáles son las posibles fallas? ¿Qué procedimiento seguiría usted para encontrar la verdadera falla?

F 4.49.* Suponga que el sistema de alarma de la figura 4-77 está operando de manera que la alarma se encienda en forma continua tan pronto como se encienda el automóvil, sin importar el estado de las demás entradas. Liste las posibles fallas y escriba un procedimiento para aislar la falla.

PREGUNTAS DE PRÁCTICA SOBRE PLDS (50 A 55)

4-50.* *Verdadero o falso*:

(a) El diseño de arriba hacia abajo comienza con una descripción general de todo el sistema y sus especificaciones.

(b) Un archivo JEDEC puede usarse como archivo de entrada para un programador.

(c) Si un archivo de entrada se compila sin errores, significa que el circuito PLD funcionará en forma correcta.

(d) Un compilador puede interpretar código a pesar de los errores de sintaxis.

(e) Los vectores de prueba se utilizan para simular y probar un dispositivo.

H,B 4-51. ¿Qué son los caracteres % que se utilizan para el archivo de diseño de AHDL?

H,B 4-52. ¿Cómo se indican los comentarios en un archivo de diseño de VHDL?

B 4-53. ¿Qué es un zócalo ZIF?

B 4-54.* Enliste tres modos de entrada utilizados para introducir la descripción de un circuito en el software de desarrollo de PLDs.

B 4-55. ¿Qué significan JEDEC y HDL?

SECCIÓN 4-15

H,B 4-56. Declare los siguientes objetos de datos en AHDL o VHDL.

(a)* Un arreglo de ocho bits de salida llamados *aparatos*.

(b) Un bit individual de salida llamado *chicharra*.

(c) Un puerto de entrada numérico de 16 bits llamado *altitud*.

(d) Un bit individual intermedio dentro de un archivo de descripción de hardware llamado *alambre2*.

H,B 4-57. Exprese los siguientes números literales en hexadecimal, binario y decimal, utilizando la sintaxis de AHDL o VHDL.

(a)* 152_{10}

(b) 1001010100_2

(c) $3C4_{16}$

H,B 4-58.* La siguiente definición de E/S que se da para AHDL y para VHDL. Escriba cuatro instrucciones de asignación concurrentes que conecten las entradas con las salidas, como se muestra en la figura 4-78.

```
SUBDESIGN hw
(
    bitsent[3..0]   :INPUT;
    bitssal[3..0]   :OUTPUT;
)
```

```
ENTITY hw IS
PORT (
     bitsent     :IN BIT VECTOR (3 downto 0);
     bitssal     :OUT BIT VECTOR (3 downto 0)
     );
END hw;
```

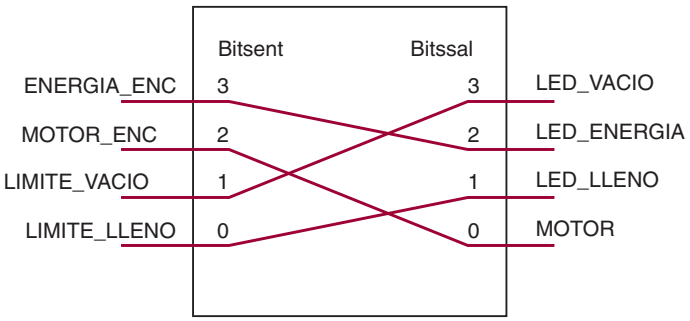


FIGURA 4-78
Problema 4-58.

SECCIÓN 4-16

H,D 4-59. Modifique la tabla de verdad de AHDL de la figura 4-50 para implementar la ecuación $AB + A\overline{C} + \overline{A}B$.

H,D 4-60.* Modifique el diseño de AHDL en la figura 4-54, de manera que $z = 1$ sólo cuando el valor digital sea menor que 1010_2.

H,D 4-61. Modifique la tabla de verdad de AHDL de la figura 4-51 para implementar $AB + A\overline{C} + \overline{A}B$.

H,D 4-62.* Modifique el diseño de VHDL de la figura 4-55, de manera que $z = 1$ sólo cuando el valor digital sea menor que 1010_2.

H,B 4-63. Modifique el código de (a) la figura 4-54 o (b) la figura 4-55 de tal forma que la salida z esté en BAJO sólo cuando valor_digital se encuentre entre 6 y 11 (inclusivo).

H,D 4-64. Modifique (a) el diseño de AHDL de la figura 4-60 para implementar la tabla 4-1. (b) el diseño de VHDL de la figura 4-61 para implementar la tabla 4-1.

H,D 4-65.* Escriba la ecuación booleana del archivo de diseño de descripción de hardware para implementar el ejemplo 4-9.

4-66. Escriba la ecuación booleana del archivo de diseño de descripción de hardware para implementar un generador de paridad de cuatro bits, como se muestra en la figura 4-25(a).

PREGUNTA DE PRÁCTICA

B 4-67. Defina cada uno de los siguientes términos.

(a) Mapa de Karnaugh.
(b) Forma de suma de productos.
(c) Generador de paridad.
(d) Octeto.

(e) Circuito de habilitación.
(f) Condición "no importa".
(g) Entrada flotante.
(h) Nivel de voltaje indeterminado.
(i) Colisión.
(j) PLD.
(k) TTL.
(l) CMOS.

APLICACIONES DE MICROCOMPUTADORA

C 4-68. En una microcomputadora, la unidad del microprocesador (MPU) siempre se está comunicando con uno de los siguientes elementos: (1) memoria de acceso aleatorio (RAM), la cual almacena los programas y datos que pueden modificarse con facilidad; (2) memoria de sólo lectura (ROM), la cual almacena programas y datos que nunca se modifican; y (3) dispositivos externos de entrada/salida (E/S) tales como teclados, pantallas de video, impresoras y unidades de disco. Al ejecutar un programa, la MPU genera un código de dirección que selecciona el tipo de dispositivo (RAM, ROM o E/S) con el que desea comunicarse. La figura 4-79 muestra un arreglo común en donde la MPU produce como salida un código de dirección de ocho bits ($A_{15} - A_8$). En realidad la MPU produce como salida un código de dirección de 16 bits, pero los bits de menor orden ($A_7 - A_0$) no se utilizan en el proceso de selección de dispositivos. El código de dirección se aplica a un circuito lógico que lo utiliza para generar las señales de selección de dispositivos: $\overline{RAM}$, $\overline{ROM}$ y $\overline{E/S}$.

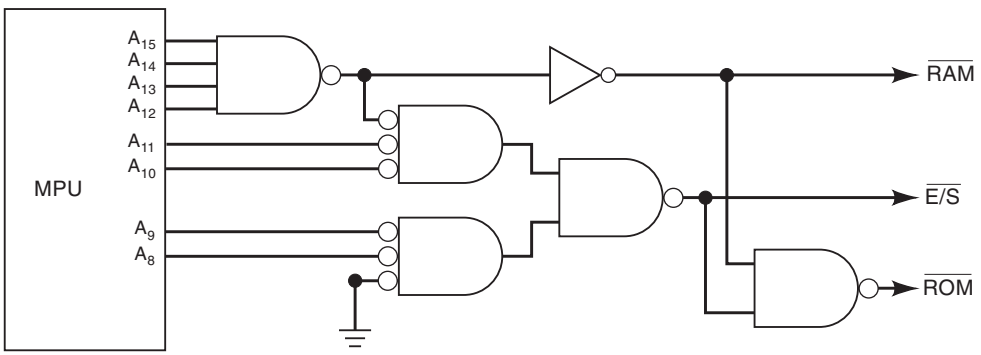


FIGURA 4-79
Problema 4-68.

Analice este circuito y determine lo siguiente:.

(a)* El intervalo de direcciones de A_{15} hasta A_8 que activa la señal $\overline{RAM}$.
(b) El intervalo de direcciones que activa la señal $\overline{E/S}$.
(c) El intervalo de direcciones que activa la señal $\overline{ROM}$.

Exprese las direcciones en binario y en hexadecimal. Por ejemplo, la respuesta para (a) es: A_{15} a $A_8 = 00000000_2$ a $11101111_2 = 00_{16}$ a EF_{16}.

C,D 4-69. En algunas microcomputadoras la MPU puede *deshabilitarse* por breves periodos mientras que otro dispositivo controla los dispositivos de RAM, ROM y E/S. Durante estos intervalos, la MPU activa una señal de control especial ($\overline{DMA}$), la cual se utiliza para deshabilitar (desactivar) la lógica de selección de dispositivos, de manera que las señales $\overline{RAM}$, $\overline{ROM}$ y $\overline{E/S}$ se encuentren en su estado inactivo. Modifique el circuito de la figura 4-79, de tal forma que $\overline{RAM}$, $\overline{ROM}$ y $\overline{E/S}$ se desactiven cada vez que la señal $\overline{DMA}$ esté activa, sin importar el estado del código de dirección.

RESPUESTAS A LAS PREGUNTAS DE REPASO DE LAS SECCIONES

SECCIÓN 4-1

1. Sólo (a). 2. Sólo (c).

SECCIÓN 4-3

1. La expresión (b) no se encuentra en la forma de suma de productos debido al signo de inversión sobre las variables C y D (es decir, el término $A\overline{CD}$). La expresión (c) no se encuentra en la forma de suma de productos debido al término $(M + \overline{N})P$.
3. $x = \overline{A} + \overline{B} + \overline{C}$

SECCIÓN 4-4

1. $x = \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}C\overline{D} + \overline{A}B\overline{C}\,\overline{D}$ 2. Ocho

SECCIÓN 4-5

1. $x = AB + AC + BC$ 2. $x = A + BCD$ 3. $S = \overline{P} + QR$ 4. Una condición de entrada para la cual no haya una condición de salida requerida específica; es decir, tenemos la libertad de hacerla 1 o 0.

SECCIÓN 4-6

2. Un nivel BAJO constante. 3. No; la compuerta XOR disponible puede utilizarse como un INVERSOR si se conecta una de sus entradas a un nivel ALTO constante (vea el ejemplo 4-16).

SECCIÓN 4-8

1. $x = \overline{A(B \oplus C)}$ 2. OR, NAND 3. NAND, NOR

SECCIÓN 4-9

1. DIP 2. SSI, MSI, LSI, VLSI, ULSI, GSI 3. Verdadero. 4. Verdadero.
5. series 40, 74AC y 74ACT 6. De 0 a 0-8 V; de 2.0 a 5-0 V 7. De 0 a 1.5 V; de 3.5 a 5.0 V 8. Como si la entrada estuviera en ALTO. 9. Es impredecible; podría sobrecalentarse y destruirse. 10. 74HCT y 74ACT 11. Describen la manera exacta de cómo interconectar los chips para distribuir el circuito y diagnosticar fallas.
12. Se definen las entradas y las salidas, y se describen las relaciones lógicas.

SECCIÓN 4-11

1. Entradas o salidas abiertas; entradas o salidas en corto con V_{CC}; entradas o salidas cortocircuitadas a tierra; terminales en corto; fallas internas del circuito. 2. Terminales en corto. 3. Para TTL, un nivel BAJO; para CMOS, indeterminado. 4. Dos o más salidas conectadas entre sí.

SECCIÓN 4-12

1. Líneas de señal abiertas; líneas de señal en corto; fuente de poder defectuosa; carga en la salida. 2. Alambres rotos; conexiones de soldadura pobres; grietas o cortes en la tarjeta de circuito impreso; terminales del CI dobladas o rotas; zócalos de CI defectuosos.
3. Los CIs operan en forma errática o no operan. 4. Nivel lógico indeterminado.

SECCIÓN 4-14

1. Las conexiones controladas por electricidad se programan como abiertas o cerradas. 2. (4, 1)(2, 2) o (2, 1)(4, 2) 3. (4, 5)(1, 6) o (4, 6)(1, 5)

SECCIÓN 4-15

1. (a) botones_pulsar[5..0] :INPUT; (b) botones_pulsar :IN BIT_VECTOR (5 DOWNTO 0);
2. (a) z = botones_pulsar[5]; (b) z <= botones_pulsar(5); 3. STD_LOGIC
4. STD_LOGIC_VECTOR

SECCIÓN 4-16

1. (AHDL) omega[ ] = (x, y, z); (VHDL) omega <= x & y & z; 2. Mediante el uso de la palabra clave TABLE. 3. Mediante el uso de las asignaciones de señal selecta.

SECCIÓN 4-17

1. IF/THEN 2. IF/THEN/ELSE 3. CASE o IF/ELSIF
4. (AHDL) conteo[7..0] :INPUT; (VHDL) conteo: IN INTEGER RANGE 0 TO 205

CAPÍTULO 5

FLIP-FLOPS Y DISPOSITIVOS RELACIONADOS

CONTENIDO

5-1 Latch de compuerta NAND
5-2 Latch de compuerta NOR
5-3 Ejemplo práctico de diagnóstico de fallas
5-4 Pulsos digitales
5-5 Señales de reloj y flip-flops sincronizados por reloj
5-6 Flip-flop sincronizado por reloj en S-R
5-7 Flip-flop sincronizado por reloj en J-K
5-8 Flip-flop sincronizado por reloj en D
5-9 Latch *D* (latch transparente)
5-10 Entradas asíncronas
5-11 Símbolos IEEE/ANSI
5-12 Consideraciones de sincronización de los flip-flops
5-13 Problema potencial de sincronización en los circuitos con FF
5-14 Aplicaciones de los flip-flops
5-15 Sincronización de los flip-flops
5-16 Detección de una secuencia de entrada
5-17 Almacenamiento y transferencia de datos
5-18 Transferencia de datos en serie: registros de desplazamiento
5-19 División y conteo de frecuencia
5-20 Aplicación de microcomputadora
5-21 Dispositivos disparadores de Schmitt
5-22 One-Shot (Multivibrador monoestable)
5-23 Circuitos generadores de reloj
5-24 Diagnóstico de fallas en circuitos con flip-flops
5-25 Circuitos secuenciales mediante el uso de HDL
5-26 Dispositivos disparados por flanco
5-27 Circuitos de HDL con varios componentes

OBJETIVOS

Al terminar este capítulo, usted podrá:

- Construir y analizar la operación de un flip-flop tipo match, compuesto a partir de compuertas NAND o NOR.
- Describir la diferencia entre los sistemas síncronos y asíncronos.
- Comprender la operación de los flip-flops disparados por flanco.
- Analizar y aplicar los diversos parámetros de sincronización de los flip-flops especificados por los fabricantes.
- Comprender las principales diferencias entre las transferencias de datos en paralelo y en serie.
- Dibujar las formas de onda de sincronización de salida de varios tipos de flip-flops en respuesta a un conjunto de señales de entrada.
- Reconocer los diversos símbolos IEEE/ANSI para los flip-flops.
- Utilizar los diagramas de transición de estado para describir la operación de un contador.
- Utilizar flip-flops en circuitos de sincronización.
- Conectar registros de desplazamiento como circuitos de transferencia de datos.
- Emplear los flip-flops como circuitos de división y conteo de frecuencia.
- Comprender las características comunes de los disparadores de Schmitt.
- Aplicar dos tipos distintos de monoestables (one-shots) en el diseño de circuitos.
- Diseñar un oscilador de carrera libre mediante el uso de un temporizador 555.
- Reconocer y predecir los efectos del desfase de reloj en los circuitos síncronos.
- Diagnosticar fallas en diversos tipos de circuitos con flip-flops.
- Escribir código en HDL para los latches.
- Utilizar primitivas, componentes y bibliotecas lógicas en el código de HDL.
- Construir circuitos a nivel estructural a partir de ciertos componentes.

INTRODUCCIÓN

Los circuitos lógicos que hemos considerado hasta ahora han sido combinacionales, cuyos niveles de salida en cualquier instante son dependientes de los niveles presentes en las entradas en ese instante. Cualquier condición anterior en relación con los niveles de entrada no tiene efecto alguno sobre las salidas actuales, ya que los circuitos lógicos combinacionales no tienen memoria. La mayoría de los sistemas digitales consisten tanto de circuitos combinacionales como de elementos de memoria.

La figura 5-1 muestra un diagrama de bloques de un sistema digital general que combina las compuertas lógicas combinacionales con los dispositivos de memoria. La porción combinacional acepta las señales lógicas de las entradas externas y de las salidas de los elementos de memoria. El circuito combinacional opera sobre estas entradas para producir varias salidas, algunas de las cuales se utilizan para determinar los valores binarios que se van a almacenar en los elementos de memoria.

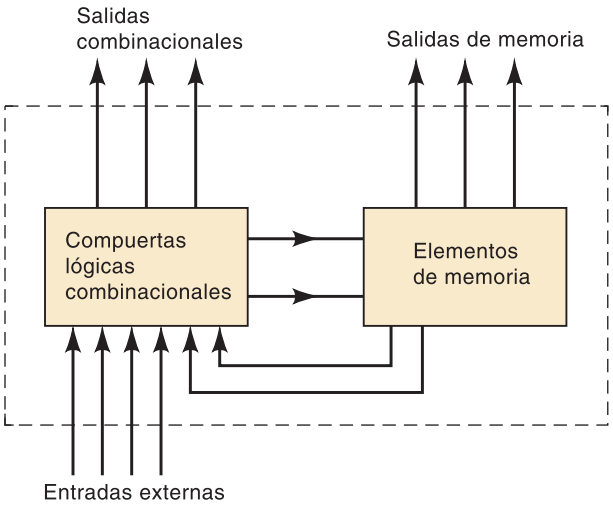


FIGURA 5-1 Diagrama de un sistema digital general.

A su vez, las salidas de algunos de los elementos de memoria van a las entradas de las compuertas lógicas en los circuitos combinacionales. Este proceso indica que las salidas externas de un sistema digital son funciones tanto de sus entradas externas como de la información almacenada en sus elementos de memoria.

El elemento de memoria más importante es el **flip-flop**, el cual está formado por un conjunto de compuertas lógicas. Una compuerta lógica por sí sola no tiene capacidad de almacenamiento, sin embargo, varias de ellas pueden interconectarse de cierta forma logrando que permiten almacenar información. Para producir estos flip-flop (que se abrevian como FF) se utilizan varios arreglos distintos de compuertas.

La figura 5-2(a) es el tipo general de símbolo que se utiliza para un flip-flop. Ahí se muestran dos salidas identificadas como Q y $\overline{Q}$, que son el inverso una de otra. $Q/\overline{Q}$ son las designaciones más comunes que se utilizan para las salidas de un FF. De vez en cuando utilizaremos otras designaciones tales como $X/\overline{X}$ y $A/\overline{A}$, por conveniencia, para identificar los distintos FFs en un circuito lógico.

La salida Q es la salida *normal* del FF, y $\overline{Q}$ es la salida *invertida* del FF. Cada vez que nos referimos al estado de un FF, estamos haciendo referencia al estado de su salida normal (Q); se sobreentiende que su salida invertida ($\overline{Q}$) está en el estado opuesto. Por ejemplo, si decimos que un FF se encuentra en el estado ALTO (1), significa que $Q = 1$; si decimos que un FF se encuentra en el estado BAJO (0), significa que $Q = 0$. Desde luego que el estado de $\overline{Q}$ siempre será el inverso de Q.

La figura 5-2(b) sintetiza los dos estados de operación posibles para un FF. Observe que al estado ALTO o 1 ($Q = 1/\overline{Q} = 0$) también se le conoce como **SET (establecer)**. Cada vez que las entradas a un FF provocan que cambie al estado $Q = 1$, le llamamos *establecer* el FF; es decir, el FF ha sido establecido. De manera similar, al estado BAJO o 0 ($Q = 0/\overline{Q} = 1$) también se le conoce como **CLEAR (borrar)** o **RESET (restablecer)**. Cada vez que las entradas de un FF hacen que cambie al estado $Q = 0$, le llamamos *borrar* o *restablecer* el FF; es decir, el FF ha sido

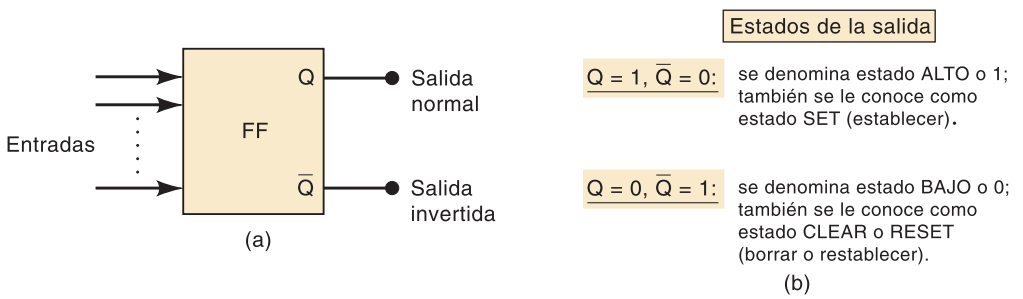


FIGURA 5-2 Símbolo general de un flip-flop y la definición de sus dos posibles estados de salida.

borrado (restablecido). Como veremos, muchos FFs tienen una entrada **SET** y/o una entrada **CLEAR** o **RESET**, la cual se utiliza para excitar el FF y llevarlo a un estado de salida específico.

Como indica el símbolo en la figura 5-2(a), un FF puede tener una o más entradas, las cuales se utilizan para hacer que el FF alterne ("flip-flop") entre sus posibles estados de salida. Más adelante veremos que la mayoría de las entradas de un FF necesitan activarse sólo en forma momentánea (pulsos) para provocar un cambio en el estado de salida del FF, y que la salida permanecerá en ese nuevo estado incluso hasta después de que se acabe el pulso de entrada. Ésta es la característica de *memoria* del FF.

Al flip-flop se le conoce también con otros nombres, incluyendo *latch* y *multivibrador biestable*. El término *latch* se utiliza para ciertos tipos de flip-flops que describiremos más adelante. El término *multivibrador biestable* es el nombre técnico más adecuado en español para un flip-flop, pero es demasiado largo como para utilizarlo regularmente.

5-1 LATCH DE COMPUERTA NAND

El circuito de FF más básico puede crearse a partir de dos compuertas NAND o de dos compuertas NOR. En la figura 5-3(a) se muestra la versión con compuertas NAND, a la cual se le conoce como **latch de compuerta NAND** o simplemente **latch**. Las dos compuertas NAND están retroalimentadas en forma transversal, de manera que la salida de la compuerta NAND-1 está conectada a una de las entradas de la compuerta NAND-2 y viceversa. Las salidas de las compuertas, identificadas como Q y $\overline{Q}$ respectivamente, son las salidas del latch. Bajo condiciones normales, una salida siempre será el inverso de la otra. Existen dos entradas para el latch: la entrada SET es la que *establece* Q en el estado 1; la entrada RESET es la que *restablece* Q al estado 0.

Por lo general, las entradas SET y RESET permanecen en el estado ALTO, y una de ellas cambiará a BAJO mediante un pulso cada vez que se quiera cambiar el estado de las salidas del latch. Comenzaremos nuestro análisis mostrando que hay dos estados de salida con la misma probabilidad cuando SET = RESET = 1. La figura 5-3(a) muestra una de esas posibilidades, en donde tenemos que $Q = 0$ y $\overline{Q} = 1$. Cuando $Q = 0$, las entradas para la compuerta NAND-2 son 0 y 1, lo cual produce $\overline{Q} = 1$. El 1 de Q hace que la compuerta NAND-1 tenga un 1 en ambas entradas para producir una salida de 0 en $\overline{Q}$. En efecto, lo que tenemos es el nivel BAJO en la salida de la compuerta NAND-1 que produce un nivel ALTO en la salida de la compuerta NAND-2, lo cual a su vez mantiene la salida de la compuerta NAND-1 en BAJO.

La segunda posibilidad se muestra en la figura 5-3(b), en donde $Q = 1$ y $\overline{Q} = 0$. El nivel ALTO de la compuerta NAND-1 produce un nivel BAJO en la salida de la compuerta NAND-2, el cual a su vez mantiene la salida de la compuerta NAND-1 en ALTO. De esta forma, hay dos posibles estados de salida cuando SET = RESET = 1; como veremos más adelante, la que se dé en un momento dado, dependerá de lo que haya ocurrido antes en las entradas.

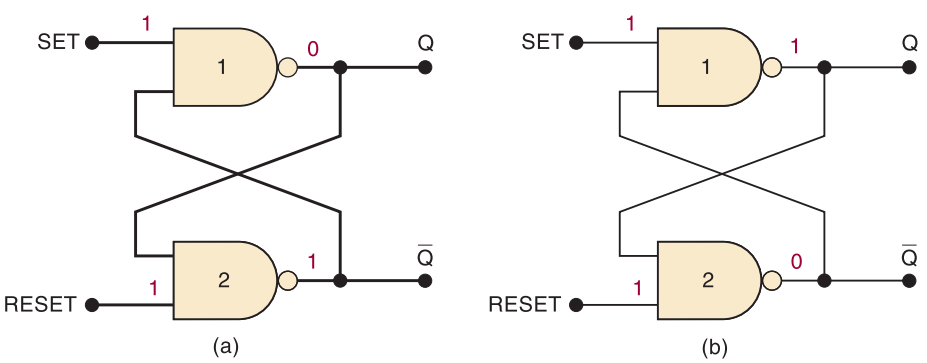


FIGURA 5-3 Un latch NAND tiene dos posibles estados en los que se puede mantener cuando SET = RESET = 1.

Cómo establecer el latch (FF)

Ahora vamos a investigar lo que ocurre cuando se aplica un pulso a la entrada SET para mantenerla unos momentos en nivel BAJO mientras que RESET se mantiene en ALTO. La figura 5-4(a) muestra lo que ocurre cuando $Q = 0$ antes de que se produzca el pulso. A medida que se aplica un pulso a SET para que cambie a nivel BAJO en el tiempo t_0, Q cambiará a ALTO y este nivel ALTO obligará a $\overline{Q}$ a cambiar a BAJO, de manera que la compuerta NAND-1 ahora tiene dos entradas en BAJO. Por lo tanto, cuando SET regrese al estado 1 en t_1, la salida de la compuerta NAND-1 *permanecerá* en ALTO, lo cual a su vez mantendrá la salida de la compuerta NAND-2 en BAJO.

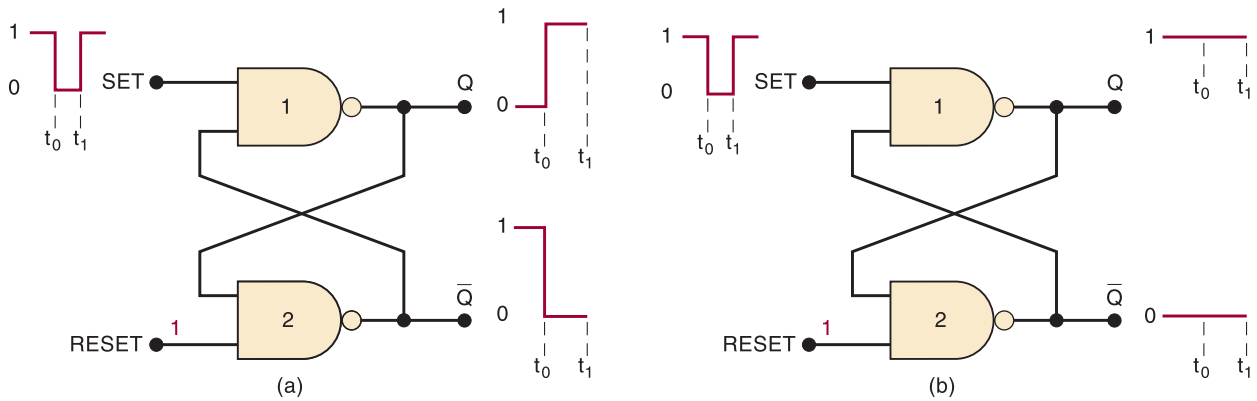


FIGURA 5-4 Aplicación de un pulso a la entrada SET para que cambie al estado 0 cuando (a) Q = 0 antes del pulso en SET; (b) Q = 1 antes del pulso en SET. Observe que, en ambos casos, Q termina en ALTO.

La figura 5-4(b) muestra lo que ocurre cuando $Q = 1$ y $\overline{Q} = 0$ antes de la aplicación del pulso en SET. Como $\overline{Q} = 0$ ya está manteniendo la salida de la compuerta NAND-1 en ALTO, el pulso BAJO en SET no cambiará nada. Así, cuando SET regrese a ALTO, las salidas del latch seguirán en el estado $Q = 1$, $\overline{Q} = 0$.

Para sintetizar la figura 5-4 diremos que un pulso BAJO en la entrada SET siempre hará que el latch termine en el estado $Q = 1$. A esta operación se le conoce como *establecer* el latch o FF.

Cómo restablecer el latch (FF)

Ahora consideremos lo que ocurre cuando se aplica un pulso a la entrada RESET para que cambie a nivel BAJO mientras que SET se mantiene en ALTO. La figura

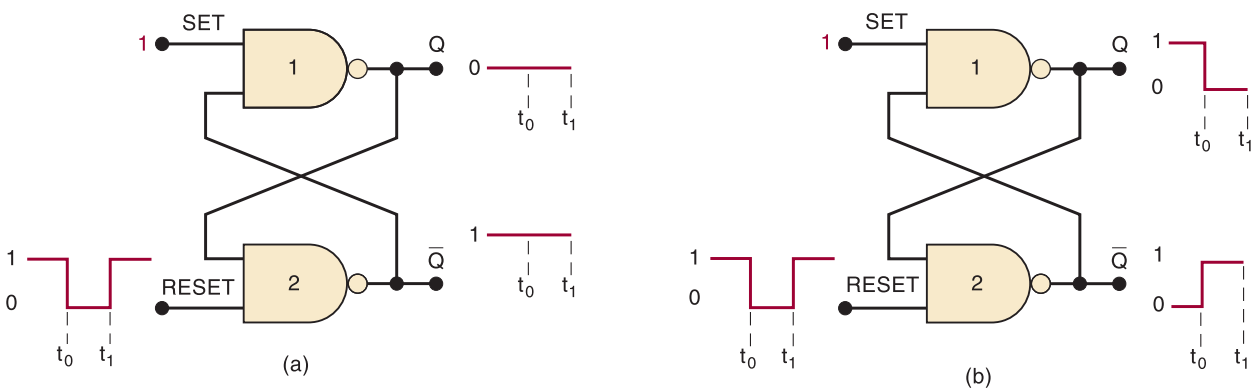


FIGURA 5-5 Aplicación de un pulso a la entrada RESET para que cambie al estado BAJO cuando (a) Q = 0 antes del pulso en RESET; (b) Q = 1 antes del pulso en RESET. En cada caso, Q termina en BAJO.

5-5(a) muestra lo que ocurre cuando $Q = 0$ y $\overline{Q} = 1$, antes de aplicar el pulso. Como $Q = 0$ ya está manteniendo la salida de la compuerta NAND-2 en ALTO, el pulso BAJO en RESET no tendrá ningún efecto. Cuando RESET regrese a ALTO, las salidas del latch seguirán en el estado $Q = 0$ y $\overline{Q} = 1$.

La figura 5-5(b) muestra la situación en donde $Q = 1$ antes de la ocurrencia del pulso en RESTABLECER. A medida que se aplica un pulso en dicha terminal, para que cambie a BAJO en t_0, $\overline{Q}$ cambiará a ALTO y este nivel ALTO obligará a Q a cambiar a BAJO, de manera que la compuerta NAND-2 ahora tiene dos entradas en BAJO. Así, cuando RESET regrese a ALTO en t_1, la salida de la compuerta NAND-2 *permanecerá* en ALTO, lo cual a su vez mantendrá la salida de la compuerta NAND-1 en BAJO.

Para sintetizar la figura 5-5 diremos que un pulso BAJO en la entrada RESET siempre hará que el latch termine en el estado $Q = 0$. A esta operación se le conoce como *borrar* o *restablecer* el latch.

Establecer y restablecer en forma simultánea

El último caso a considerar es cuando se aplica un pulso simultáneo a las entradas SET y RESET para que cambien a BAJO. Esto producirá niveles en ALTO en ambas salidas de las compuertas NAND, de manera que $Q = \overline{Q} = 1$. Es evidente que ésta es una condición indeseable, ya que se supone que las dos salidas deben ser el inverso una de la otra. Además, cuando las entradas SET y RESET regresen al nivel ALTO, el estado de salida resultante dependerá de cuál sea la entrada que regrese a ALTO primero. Las transiciones simultáneas al estado 1 producirán resultados impredecibles. Por estas razones, no es común utilizar la condición SET = RESET = 0 en el latch NAND.

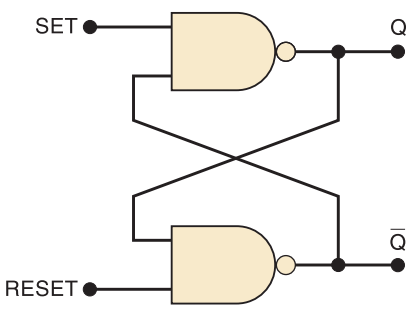
Resumen del latch NAND

La operación antes descrita puede colocarse de manera conveniente en una tabla de funciones (figura 5-6) y se sintetiza de la siguiente manera:

1. SET = RESET = 1. Esta condición es el estado normal de reposo, y no tiene efecto sobre el estado de la salida. Las salidas Q y $\overline{Q}$ permanecerán en el estado en el que se hayan encontrado antes de esta condición de entrada.
2. SET = 0, RESET = 1. Esta condición siempre provocará que la salida cambie al estado $Q = 1$, en donde permanecerá aún después de que SET regrese a ALTO. A esto se le conoce como *establecer* el latch.
3. SET = 1, RESET = 0. Esta condición siempre producirá el estado $Q = 0$, en donde permanecerá la salida aún después de que RESET regrese a ALTO. A esto se le conoce como *borrar* o *restablecer* el latch.
4. SET = RESET = 0. Esta condición trata de establecer y borrar el latch al mismo tiempo, y produce $Q = \overline{Q} = 1$. Si las entradas se regresan a 1 en forma simultánea, el estado resultante será impredecible. Esta condición de entrada no debe utilizarse.



FIGURA 5-6 (a) latch NAND; (b) tabla de funciones.



(a)

Set	Reset	Salida
1	1	Sin cambio
0	1	Q = 1
1	0	Q = 0
0	0	Inválido*

* Produce $Q = \overline{Q} = 1$.

(b)

Representaciones alternativas

De la descripción de la operación del latch NAND debe quedar claro que las entradas SET y RESET son activas en BAJO. La entrada SET establecerá $Q = 1$ cuando SET cambie a BAJO; la entrada RESET borrará $Q = 0$ cuando RESET cambie a BAJO. Por esta razón, a menudo el latch NAND se dibuja utilizando la representación alternativa para cada compuerta NAND, como se muestra en la figura 5-7(a). Las burbujas en las entradas, así como el etiquetado de las señales como $\overline{SET}$ y $\overline{RESET}$, indican el estado activo en BAJO de estas entradas. (Tal vez quiera repasar las secciones 3-13 y 3-14 que hablan sobre este tema.)

La figura 5-7(b) muestra una representación en bloque simplificada, la cual utilizaremos algunas veces. Las etiquetas *S* y *R* representan las entradas SET y RESET, y las burbujas indican la naturaleza activa en BAJO de estas entradas. Cada vez que utilicemos este símbolo de bloque, representará a un latch NAND.

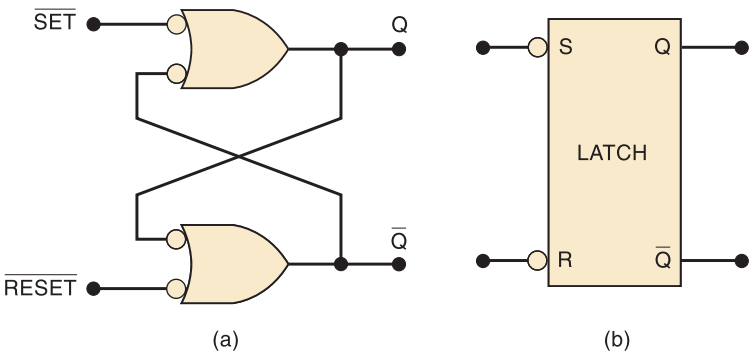


FIGURA 5-7
(a) representación equivalente del latch NAND; (b) símbolo de bloque simplificado.

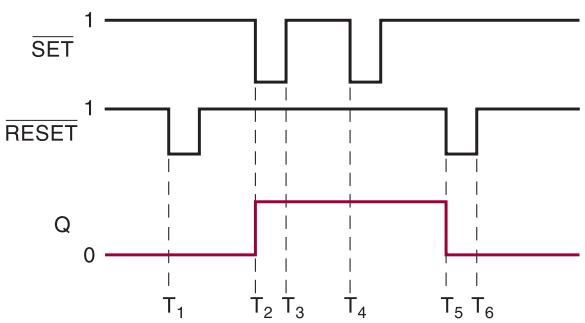
Terminología

A la acción de *restablecer* un FF o un latch también se le conoce como *borrar*; ambos términos se utilizan sin distinción en el campo digital. De hecho, una entrada RESET puede llamarse también entrada CLEAR, y un latch SET-RESET puede llamarse latch SET-CLEAR.

EJEMPLO 5-1

Las formas de onda de la figura 5-8 se aplican a las entradas del latch de la figura 5-7. Suponga que al inicio $Q = 0$; determine la forma de onda de *Q*.

FIGURA 5-8 Ejemplo 5-1.



Solución

Al principio $\overline{SET} = \overline{RESET} = 1$, por lo que *Q* permanecerá en el estado 0. El pulso BAJO que ocurre en la entrada $\overline{RESET}$ en el tiempo T_1 no tendrá efecto, ya que *Q* se encuentra de antemano en el estado borrado (0).

La única forma en que Q puede pasar al estado 1 es mediante un pulso BAJO en la entrada $\overline{\text{SET}}$. Esto ocurre en el tiempo T_2, cuando $\overline{\text{SET}}$ primero cambia a BAJO. Cuando $\overline{\text{SET}}$ regresa al nivel ALTO en T_3, Q permanecerá en su nuevo estado ALTO.

En el tiempo T_4, cuando $\overline{\text{SET}}$ cambia a BAJO de nuevo, no habrá efecto sobre Q ya que se encuentra de antemano en el estado 1.

La única forma de regresar a Q de nuevo al estado 0 es mediante un pulso BAJO en la entrada $\overline{\text{RESET}}$. Esto ocurre en el tiempo T_5. Cuando $\overline{\text{RESET}}$ regresa a 1 en el tiempo T_6, Q permanece en el estado BAJO.

El ejemplo 5-1 muestra que la salida del latch "recuerda" la última entrada que se activó, y no cambiará de estado sino hasta que se active la entrada opuesta.

EJEMPLO 5-2

Es casi imposible obtener una transición de voltaje "limpia" de un interruptor mecánico, debido al fenómeno conocido como **rebotes de contacto**. Esto se ilustra en la figura 5-9(a), en donde la acción de mover el interruptor de la posición de contacto 1 a la 2 produce varias transiciones en el voltaje de salida, a medida que el interruptor rebota (hace contacto y lo interrumpe con el contacto 2 varias veces) antes de detenerse en el contacto 2.

Por lo general, las transiciones múltiples en la señal de salida no duran más de unos cuantos milisegundos, pero podrían ser inaceptables en muchas aplicaciones. Un latch NAND puede utilizarse para evitar que la presencia del rebote de contactos afecte la salida. Describa la operación del circuito para "eliminar rebotes en el interruptor" de la figura 5-9(b).

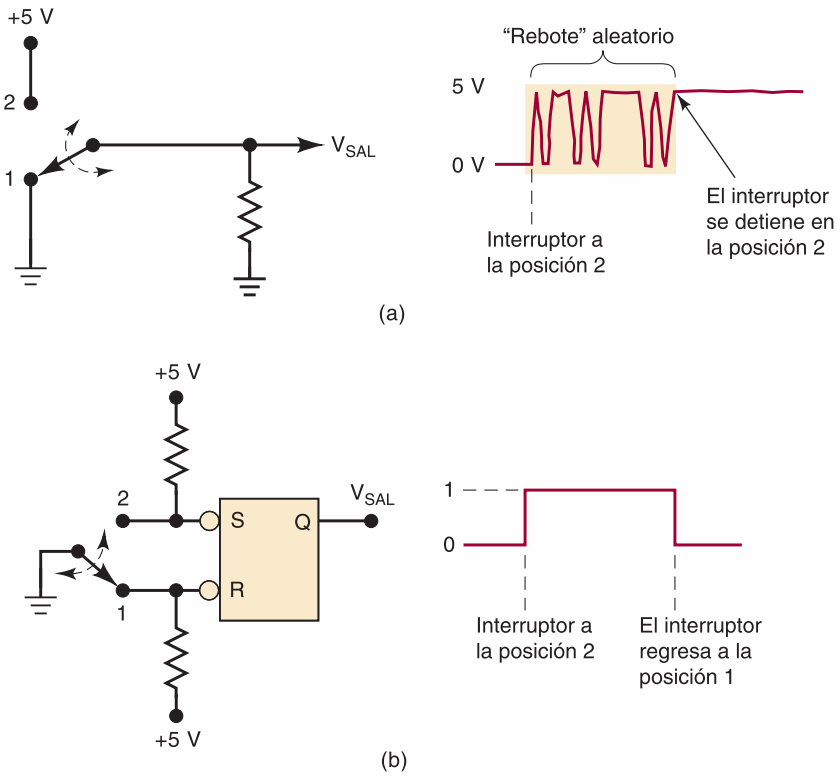


FIGURA 5-9 (a) El rebote mecánico de los contactos producirá transiciones múltiples; (b) latch NAND utilizado para eliminar los rebotes en un interruptor mecánico.

Solución

Suponga que el interruptor está detenido en la posición 1, de manera que la entrada $\overline{\text{RESET}}$ está en BAJO y $Q = 0$. Cuando el interruptor se mueve a la posición 2, $\overline{\text{RESET}}$ cambiará al nivel ALTO y aparecerá un nivel BAJO en la entrada $\overline{\text{SET}}$ cuando el interruptor haga el primer contacto. Esto hará que $Q = 1$ en cuestión de unos cuantos nanosegundos (el tiempo de respuesta de la compuerta NAND). Ahora, si el interruptor rebota del contacto 2, tanto $\overline{\text{SET}}$ como $\overline{\text{RESET}}$ estarán en ALTO y Q no se verá afectada; permanecerá en ALTO. Por lo tanto, nada ocurrirá en Q a medida que el interruptor rebote del contacto 2 antes de que por fin se detenga en la posición 2.

De igual forma, cuando el interruptor se mueva de la posición 2 a la posición 1, se colocará un BAJO en la entrada $\overline{\text{RESET}}$ cuando haga el primer contacto. Esto pondrá a Q en el estado BAJO, en donde permanecerá aún y cuando el interruptor rebote en el contacto 1 varias veces antes de detenerse.

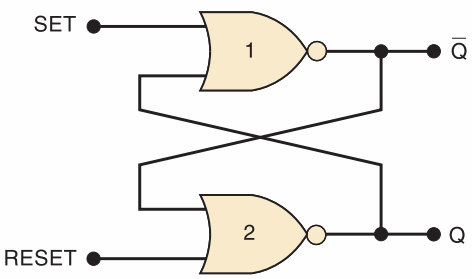
De esta forma, la salida en Q consistirá de una sola transición cada vez que el interruptor se mueva de una posición a la otra.

PREGUNTAS DE REPASO

1. ¿Cuál es el estado normal de reposo para las entradas $\overline{\text{SET}}$ y $\overline{\text{RESET}}$? ¿Cuál es el estado activo de cada una de estas entradas?
2. ¿Cuáles serán los estados de Q y $\overline{Q}$ después de que se haya restablecido (borrado) un FF?
3. *Verdadero o falso:* la entrada $\overline{\text{SET}}$ nunca puede usarse para hacer que $Q = 0$.
4. Cuando se aplica energía por primera vez a cualquier circuito FF, es imposible predecir los estados iniciales de Q y $\overline{Q}$. ¿Qué puede hacerse para asegurar que un latch NAND siempre empiece en el estado $Q = 1$?

5-2 LATCH DE COMPUERTA NOR

Pueden usarse dos compuertas NOR acopladas en forma transversal para formar lo que se conoce como **latch de compuerta NOR**. El arreglo, que se muestra en la figura 5-10(a), es similar al latch NAND a excepción de que las salidas Q y $\overline{Q}$ aparecen en posiciones invertidas.

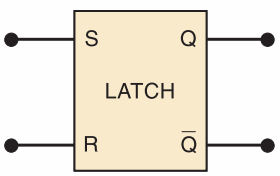


(a)

Set	Reset	Salida
0	0	Sin cambio
1	0	Q = 1
0	1	Q = 0
1	1	Inválido*

* Produce $Q = \overline{Q} = 0$.

(b)



(c)



FIGURA 5-10 (a) Latch de compuerta NOR; (b) tabla de funciones; (c) símbolo de bloque simplificado.

Podemos realizar el análisis de la operación del latch NOR exactamente de la misma forma que lo hicimos para el latch NAND. Los resultados se proporcionan en la tabla de funciones de la figura 5-10(b) y se sintetizan de la siguiente manera:

1. SET = RESET = 0. Éste es el estado normal de reposo para el latch NOR y no tiene efecto sobre el estado de la salida. Q y $\overline{Q}$ permanecerán en el estado que tenían antes de que se produjera esta condición de entrada.

2. SET = 1, RESET = 0. Esta condición siempre establecerá $Q = 1$, en donde permanecerá aún después de que SET regrese a 0.
3. SET = 0, RESET = 1. Esta condición siempre borrará $Q = 0$, en donde permanecerá aún después de que RESET regrese a 0.
4. SET = 1, RESET = 1. Esta condición trata de establecer y restablecer el latch al mismo tiempo, y produce $Q = \overline{Q} = 0$. Si las entradas se regresan a 0 en forma simultánea, el estado de salida resultante será impredecible. Esta condición de entrada no debe utilizarse.

El latch de compuerta NOR opera exactamente de la misma forma que el latch NAND, sólo que las entradas SET y RESET son activas en ALTO, en vez de activas en BAJO, y el estado detenido normal es SET = RESET = 0. Q cambiará a nivel ALTO mediante un pulso ALTO en la entrada SET, y cambiará a BAJO mediante un pulso ALTO en la entrada RESET. El símbolo de bloque simplificado para el latch NOR en la figura 5-10(c) se muestra sin burbujas en las entradas S y R; esto indica que las entradas son activas en ALTO.

EJEMPLO 5-3

Suponga que al principio $Q = 0$ y determine la forma de onda de Q para las entradas del latch NOR de la figura 5-11.

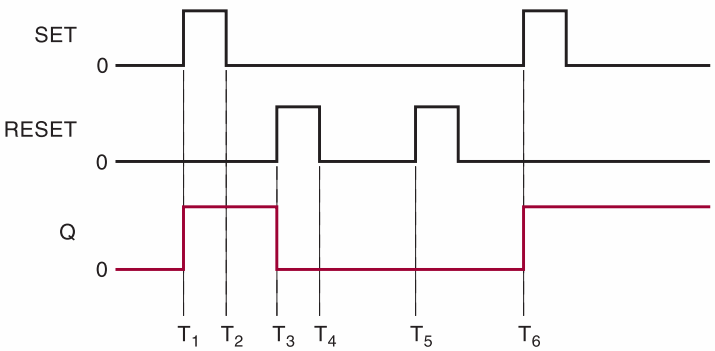


FIGURA 5-11 Ejemplo 5-3.

Solución

En un principio, SET = RESET = 0, lo cual no tiene efecto sobre Q, que permanece en BAJO. Cuando SET cambie a nivel ALTO en el tiempo T_1, Q se establecerá en 1 y permanecerá ahí aún después de que SET regrese a 0 en T_2.

En T_3 la entrada RESET cambia a nivel ALTO y borra a Q para que quede en el estado 0, en donde permanecerá aún después de que RESET regrese a BAJO en T_4.

El pulso RESET en T_5 no tiene efecto sobre Q, ya que de antemano se encuentra en BAJO. El pulso en SET en el tiempo T_6 regresa a Q de nuevo a 1, en donde permanecerá por tiempo indefinido.

El ejemplo 5-3 muestra que el latch "recuerda" la última entrada que se activó, y no cambiará de estado sino hasta que se active la entrada opuesta.

EJEMPLO 5-4

La figura 5-12 muestra un circuito simple que puede utilizarse para detectar la interrupción de un haz de luz. La luz se enfoca en un fototransistor que está conectado en la configuración de emisor común para operar como un interruptor. Suponga que el latch se ha borrado previamente para quedar en el estado 0, para lo cual se abrió el interruptor SW1 durante unos instantes, y describa lo que ocurre si se interrumpe por un momento el haz de luz.

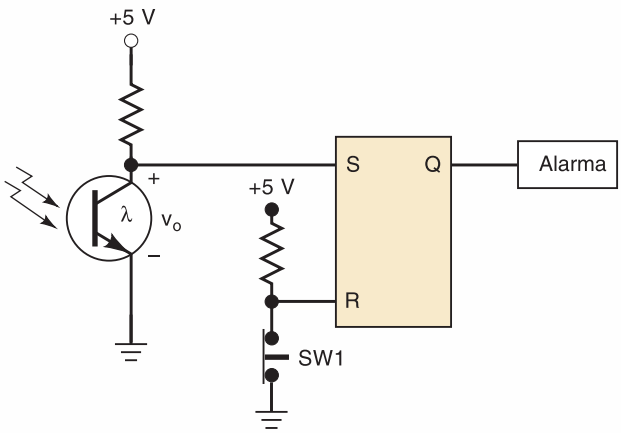


FIGURA 5-12 Ejemplo 5-4.

Solución

Cuando hay luz en el fototransistor, podemos suponer que está conduciendo al 100%, de manera que la resistencia entre el colector y el emisor es muy pequeña. Por ende, v_0 estará cerca de 0 V. Esto coloca un BAJO en la entrada SET del latch, de manera que SET = RESET = 0.

Cuando se interrumpe el haz de luz, el fototransistor se apaga y su resistencia de colector-emisor se hace muy alta (es decir, en esencia es un circuito abierto). Esto hace que v_0 se eleve hasta un valor aproximado de 5 V; con esto se activa la entrada SET, la cual establece a Q en ALTO y desactiva la alarma.

Por otro lado, Q permanecerá en ALTO y la alarma permanecerá encendida aún cuando v_0 regrese a 0 V (es decir, el haz de luz se interrumpió sólo por un momento), ya que SET y RESET estarán ambas en BAJO, lo cual no producirá cambio alguno en Q.

En esta aplicación se utiliza la característica de memoria del latch para convertir una ocurrencia momentánea (interrupción del haz) en una salida constante.

Estado de un flip-flop al momento de aplicarle energía

Cuando se aplica energía a un circuito, no es posible predecir el estado inicial de la salida de un flip-flop si sus entradas SET y RESET se encuentran en su estado inactivo (por ejemplo, $S = R = 1$ para un latch NAND, $S = R = 0$ para un latch NOR). Se tiene la misma probabilidad de que el estado inicial sea $Q = 0$ que $Q = 1$. Esto dependerá de ciertos factores tales como los tiempos de propagación internos, la capacitancia parásita y la carga externa. Si un latch o FF debe empezar en cierto estado específico para asegurar la operación apropiada de un circuito, entonces debe colocarse en ese estado mediante la activación momentánea de la entrada SET o RESET al principio de la operación del circuito. A menudo, esto se logra aplicando un pulso en la entrada apropiada.

PREGUNTAS DE REPASO

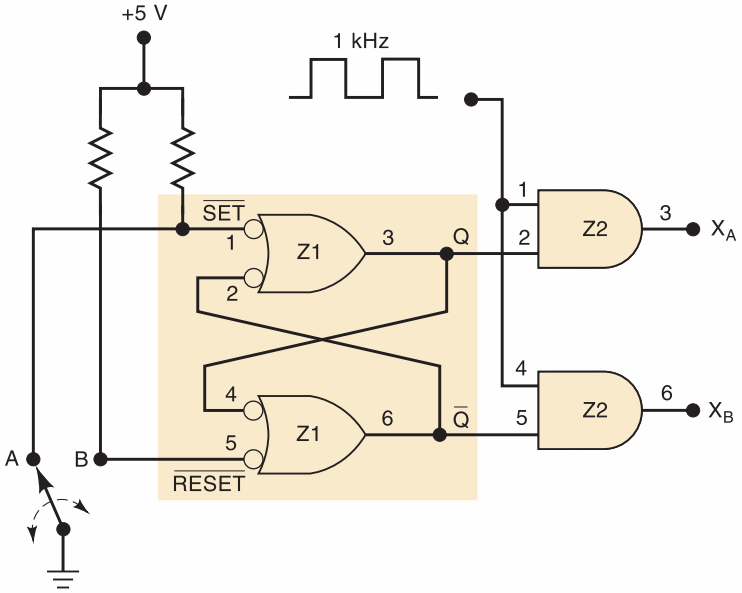
1. ¿Cuál es el estado normal en reposo de las entradas del latch NOR? ¿Cuál es el estado activo?
2. Cuando un latch se establece, ¿cuáles son los estados de Q y $\overline{Q}$?
3. ¿Cuál es la única manera de hacer que la salida Q de un latch NOR cambie de 1 a 0?
4. Si el latch NOR de la figura 5-12 se sustituyera por un latch NAND, ¿por qué el circuito no funcionaría en forma correcta?

5-3 EJEMPLO PRÁCTICO DE DIAGNÓSTICO DE FALLAS

Los siguientes dos ejemplos presentarán una ilustración de los tipos de razonamiento utilizados al diagnosticar fallas en un circuito que contenga un latch.

EJEMPLO 5-5

Analice y describa la operación del circuito de la figura 5-13.



Posición del interruptor	X_A	X_B
A	Pulsos	BAJO
B	BAJO	Pulsos

FIGURA 5-13 Ejemplos 5-5 y 5-6.

Solución

El interruptor se utiliza para establecer o borrar el latch NAND y producir señales limpias, libres de rebotes en Q y $\overline{Q}$. Estas salidas del latch controlan el paso de la señal de pulso de 1-kHz hacia las salidas X_A y X_B de la compuerta AND.

Cuando el interruptor se mueve hacia la posición A, el latch se establece para que $Q = 1$. Esto permite que los pulsos de 1-kHz pasen hacia X_A, mientras que el nivel BAJO en $\overline{Q}$ mantiene a $X_B = 0$. Cuando el interruptor se mueve hacia la posición B, el latch se borra para que $Q = 0$, lo cual mantiene a $X_A = 0$, mientras que el nivel ALTO en $\overline{Q}$ permite que los pulsos pasen hacia X_B.

EJEMPLO 5-6

Un técnico evalúa el circuito de la figura 5-13 y registra las observaciones, que se muestran en la tabla 5-1. Él observa que cuando el interruptor se encuentra en la posición B el circuito funciona de manera correcta, pero en la posición A el latch no se establece para quedar en el estado $Q = 1$. ¿Cuáles son las posibles fallas que podrían ocasionar este funcionamiento incorrecto?

Solución

Existen varias posibilidades:

1. Una conexión abierta interna en Z1-1, lo cual evitaría que Q respondiera a la entrada $\overline{\text{SET}}$.

TABLA 5-1

Posición del interruptor	$\overline{\text{SET}}$ (Z1-1)	$\overline{\text{RESET}}$ (Z1-5)	Q (Z1-3)	$\overline{Q}$ (Z1-6)	X_A (Z2-3)	X_B (Z2-6)
A	BAJO	ALTO	BAJO	ALTO	BAJO	Pulsos
B	ALTO	BAJO	BAJO	ALTO	BAJO	Pulsos

2. Una falla interna en un componente de la compuerta NAND Z1 que evite que responda en forma apropiada.
3. La salida Q se mantiene en BAJO, lo cual podría ser ocasionado por:
   (a) Z1-3 en corto interno a tierra
   (b) Z1-4 en corto interno a tierra
   (c) Z2-2 en corto interno a tierra
   (d) El nodo Q en corto externo a tierra

   La comprobación con un óhmetro de Q a tierra determinará si cualquiera de estas condiciones está presente. Una revisión visual deberá revelar cualquier corto externo.

¿Podría $\overline{Q}$ estar en corto interno o externo con V_{CC}? Al realizar el análisis nos llevará a la conclusión de que ésta no podría ser la falla. Si $\overline{Q}$ estuviera en corto con V_{CC}, esto no evitaría que la salida Q cambiara al nivel ALTO cuando $\overline{\text{SET}}$ cambiar a BAJO. Como Q *no cambia a* ALTO, ésta no puede ser la falla. La razón por la cual $\overline{Q}$ parece permanecer en ALTO es que Q permanece en BAJO, y eso mantiene a Q en ALTO mediante la compuerta NAND inferior.

5-4 PULSOS DIGITALES

Como se puede ver de nuestra disertación sobre los latches SR, hay situaciones en los sistemas digitales en las que una señal cambia de un estado normal inactivo al estado opuesto (activo), provocando que ocurra algo en el circuito. Después, la señal regresa a su estado inactivo mientras que el efecto de la señal que se activó recientemente permanece en el sistema. A estas señales se les conoce como **pulsos** y es muy importante comprender la terminología asociada con los pulsos y las formas de onda de los mismos. A un pulso que realiza su adecuada función cuando cambia a ALTO se le conoce como pulso *positivo*, y a un pulso que realiza su debida función cuando cambia a BAJO se le conoce como pulso *negativo*. En los circuitos reales, a una forma de onda de pulso le toma tiempo cambiar de un nivel al otro. A estos tiempos de transición se les conoce como tiempo de subida (t_r) y tiempo de caída (t_f), y se les define como el tiempo que tarda el voltaje en cambiar entre el 10 y el 90% del voltaje de nivel ALTO, como se muestra en el pulso positivo de la figura 5-14(a). A la transición al principio del pulso se le conoce como flanco positivo (de subida) y a la transición al final del pulso se le conoce como flanco negativo (de caída). La duración (ancho) del pulso (t_w) se le define como el tiempo entre los puntos cuando los flancos de subida y de caída se encuentran al 50% del nivel de voltaje ALTO. La figura 5-14(b) muestra un pulso activo en BAJO o negativo.

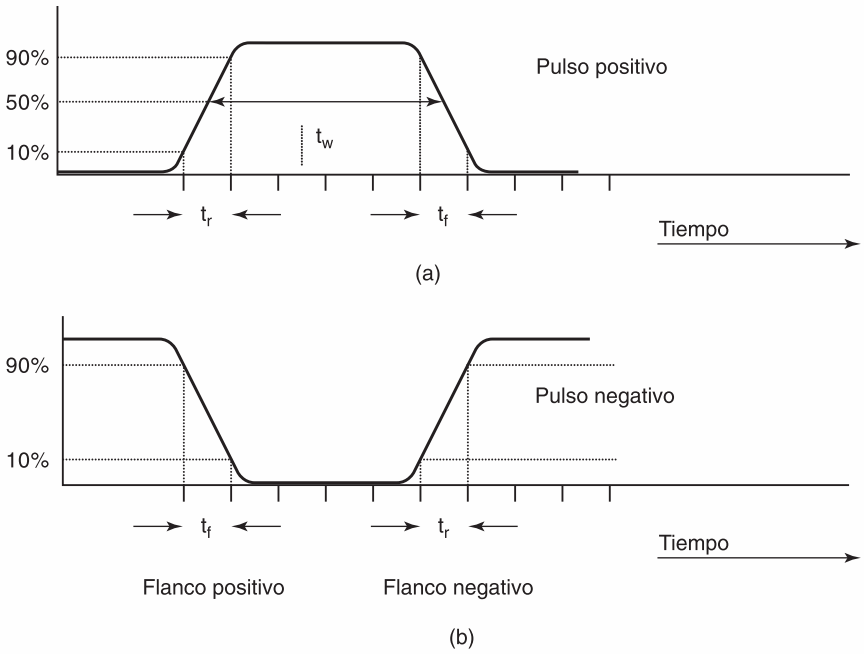


FIGURA 5-14 (a) Un pulso positivo y (b) un pulso negativo.

EJEMPLO 5-7

Cuando un microcontrolador desea acceder a los datos en su memoria externa, activa una terminal de salida activa en BAJO conocida como $\overline{RD}$ (lectura). La hoja de datos dice que, por lo general, el pulso $\overline{RD}$ tiene un ancho (t_w) de 50 ns, un tiempo de subida t_r de 15 ns y un tiempo de caída t_f de 10 ns. Haga un dibujo en escala del pulso $\overline{RD}$.

Solución

La figura 5-15 muestra el dibujo del pulso. El pulso $\overline{RD}$ es activo en BAJO, por lo que el flanco anterior es un flanco de caída medido en base a t_f y el flanco posterior es el flanco de subida medido en base t_r.

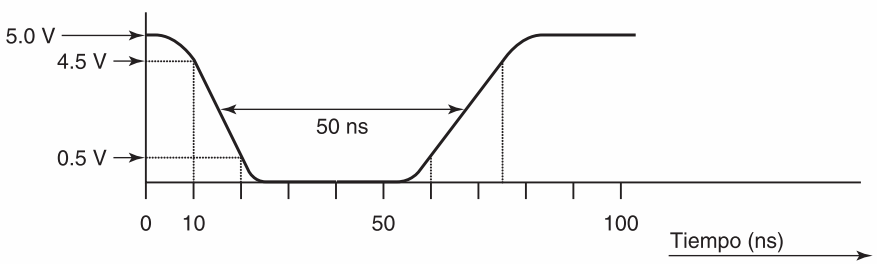


FIGURA 5-15 Ejemplo 5-7.

5-5 SEÑALES DE RELOJ Y FLIP-FLOPS SINCRONIZADOS POR RELOJ

Los sistemas digitales pueden operar en forma *asíncrona* o *síncrona*. En los sistemas asíncronos, las salidas de los circuitos lógicos pueden cambiar de estado en cualquier momento en el que una o más de las entradas cambien. Un sistema asíncrono es, por lo general, más difícil de diseñar que un sistema síncrono; de igual forma el proceso de diagnóstico de fallas es más sencillo para sistemas síncronos.

En los sistemas síncronos, los tiempos exactos en los que cualquier entrada puede cambiar de estados se determinan con base en una señal que se conoce comúnmente como el **reloj**. Por lo común, esta señal de reloj es un tren de pulsos rectangulares o una onda cuadrada, como se muestra en la figura 5-16. La señal del

reloj se distribuye en todas las partes del sistema, y la mayoría (si no es que todas) de las salidas del sistema pueden cambiar de estado sólo cuando el reloj hace una transición. La figura 5-16 muestra las transiciones (también llamadas *flancos*). Cuando el reloj cambia de un 0 a un 1 se le llama **transición de pendiente positiva (PGT)**; cuando el reloj cambia de 1 a 0 se le llama **transición de pendiente negativa (NGT)**. Utilizaremos las abreviaciones PGT y NGT ya que estos términos aparecen con mucha frecuencia a lo largo del texto.

La mayoría de los sistemas digitales son principalmente síncronos (aunque siempre hay algunas partes asíncronas), ya que es más fácil diseñar y diagnosticar fallas en los circuitos síncronos. El diagnóstico de fallas se facilita debido a que las salidas del circuito sólo pueden cambiar en instantes específicos. En otras palabras, casi todo se sincroniza con las transiciones de la señal de reloj.

La acción de sincronización de las señales del reloj se logra a través del uso de **flip-flops sincronizados por reloj**, los cuales están diseñados para cambiar de estado en una de las dos transiciones del reloj.

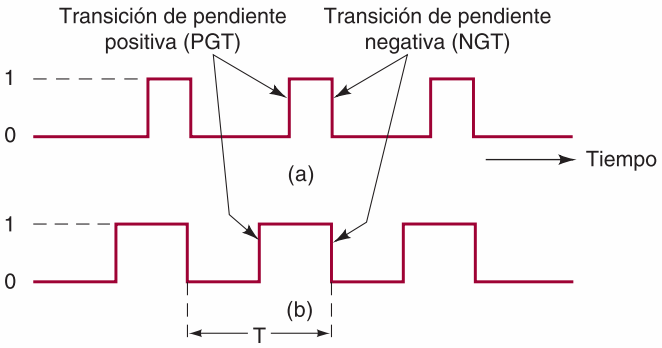


FIGURA 5-16 Señales de reloj.

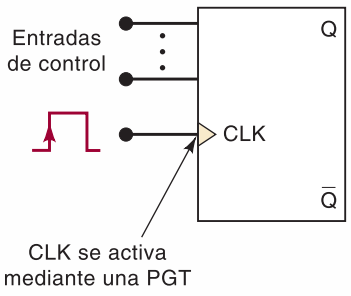
La velocidad a la que opera un sistema digital síncrono depende de la frecuencia con la que ocurren los ciclos del reloj. Un ciclo de reloj se mide desde una PGT hasta la siguiente PGT, o desde una NGT hasta la siguiente NGT. Al tiempo que se requiere para completar un ciclo (segundos/ciclo) se le conoce como **periodo (T)**, como se muestra en la figura 5-16(b). Por lo general, se hace referencia a la velocidad de un sistema digital con base en el número de ciclos de reloj que ocurren en 1 s (ciclos/segundo), a lo cual se le conoce como **frecuencia (F)** del reloj. La unidad estándar para la frecuencia son los hertz. Un hertz (1Hz) = 1 ciclo/segundo.

Flip-flops sincronizados por reloj

Varios tipos de FFs sincronizados por reloj se utilizan en una amplia variedad de aplicaciones, pero antes de comenzar nuestro estudio al respecto, describiremos las ideas principales que son comunes para todos ellos.

1. Los FFs sincronizados por reloj tienen una entrada de reloj que, por lo general, se identifica como *CLK*, *CK* o *CP*. Aquí utilizaremos *CLK*, como muestra la figurar 5-17. En la mayoría de los FFs sincronizados por reloj la entrada *CLK* es **disparada por flanco**, lo cual significa que se activa mediante una transición de señal; esto se indica por la presencia de un pequeño triángulo en la entrada *CLK*, lo cual contrasta con los latches, que se disparan por nivel.

   La figura 5-17(a) es un FF con un pequeño triángulo en su entrada *CLK* para indicar que esta entrada se activa *sólo* cuando ocurre una transición de pendiente positiva (PGT); ninguna otra parte del pulso de entrada tendrá efecto sobre la entrada *CLK*. En la figura 5-17(b) el símbolo FF tiene una burbuja, así como un triángulo en su entrada *CLK*. Esto significa que la entrada *CLK* se activa *sólo* cuando ocurre una transición de pendiente negativa; ninguna otra parte del pulso de entrada tendrá efecto sobre la entrada *CLK*.



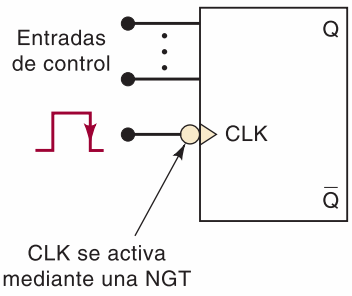


FIGURA 5-17 Los FFs sincronizados por reloj tienen una entrada de reloj (CLK) que se activa ya sea en (a) la PGT o (b) la NGT. Las entradas de control determinan el efecto de la transición activa del reloj.





2. Los FFs sincronizados por reloj tienen una o más **entradas de control** que pueden tener varios nombres, dependiendo de su operación. Las entradas de control no tendrán efecto sobre *Q* sino hasta que ocurra la transición activa del reloj. En otras palabras, su efecto está sincronizado con la señal que se aplica a *CLK*. Por esta razón se les conoce como **entradas de control síncronas**.

   Por ejemplo, las entradas de control del FF de la figura 5-17(a) no tendrán efecto sobre *Q* sino hasta que ocurra la PGT de la señal del reloj. De igual forma, las entradas de control en la figura 5-17(b) no tendrán efecto sino hasta que ocurra la NGT de la señal del reloj.
3. En resumen podemos decir que las entradas de control preparan a las salidas de FF para cambiar, mientras que la transición activa en la entrada *CLK dispara* el cambio. Las entradas de control controlan el QUE (es decir, a qué estado cambiará la salida); la entrada *CLK* determina el CUANDO.

Tiempos de estabilización y de retención

Deben cumplirse dos requerimientos de sincronización si un FF sincronizado por reloj va a responder en forma confiable a sus entradas de control cuando ocurra la transición activa en *CLK*. Estos requerimientos se ilustran en la figura 5-18 para un FF que se dispara en una PGT.

El **tiempo de estabilización (t_S)** es el tiempo que va justo antes de la transición activa de la señal *CLK*, durante la cual la entrada de control debe mantenerse en el nivel apropiado. Por lo general, los fabricantes de circuitos integrados especifican el tiempo de estabilización mínimo permisible t_s (min). Si no se cumple este requerimiento de tiempo, el FF tal vez no responda en forma confiable cuando ocurra el flanco del reloj.

El **tiempo de retención (t_H)** es el tiempo que sigue justo después de la transición activa de la señal *CLK*, durante la cual la entrada de control síncrona debe mantenerse en el nivel apropiado. Por lo general, los fabricantes de circuitos inte-

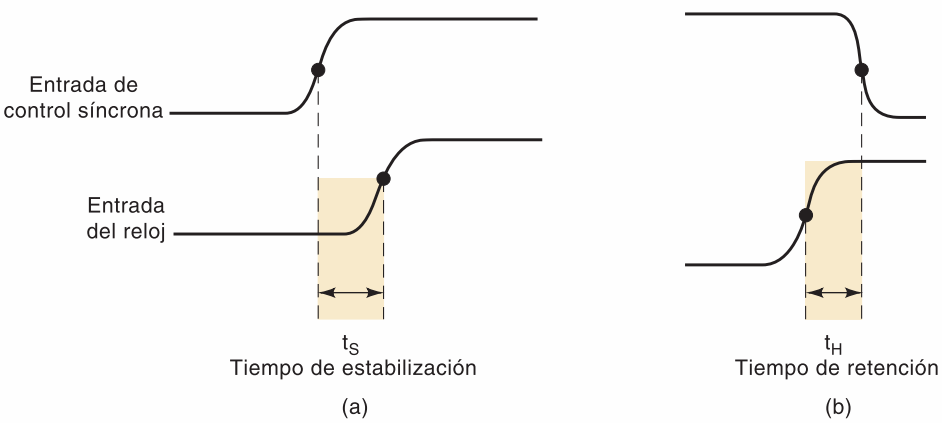


FIGURA 5-18 Las entradas de control deben mantenerse estables durante (a) un tiempo t_S antes de la transición activa del reloj y (b) durante un tiempo t_H después de la transición activa del reloj.

grados especifican el valor mínimo aceptable para el tiempo de retención t_H (min). Si no se cumple este requerimiento, el FF no se disparará en forma confiable.

Por lo anterior, para asegurar que un FF sincronizado por reloj responderá en forma apropiada cuando ocurra la transición activa del reloj, las entradas de control deben estar estables (sin cambios) durante cuando menos un tiempo igual a t_S (min) *antes* de la transición del reloj, y durante cuando menos un tiempo igual a t_H (min) *después* de la transición del reloj.

Los flip-flops de CI tendrán valores mínimos permisibles para t_S y t_H en el orden de los nanosegundos. Por lo general, los tiempos de estabilización se encuentran en el intervalo de 5 a 50 ns, mientras que los de retención son de 0 a 10 ns. Observe que estos tiempos se miden entre los puntos del 50 por ciento en las transiciones.

Estos requerimientos de sincronización son muy importantes en los sistemas síncronos ya que, como veremos, habrá muchas situaciones en las que las entradas de control síncronas para un FF estén cambiando a un tiempo casi igual que la entrada *CLK*.

PREGUNTAS DE REPASO

1. ¿Cuáles son los dos tipos de entradas que tiene un FF sincronizado por reloj?
2. ¿Qué significa el término *disparado por flanco*?
3. *Verdadero o falso:* la entrada CLK afectará a la salida del FF sólo cuando ocurra la transición activa de la entrada de control.
4. Defina los requerimientos para el tiempo de estabilización y el tiempo de retención de un FF sincronizado por reloj.

5-6 FLIP-FLOP SINCRONIZADO POR RELOJ EN S-R

La figura 5-19(a) muestra el símbolo lógico para un **flip-flop sincronizado por reloj en S-R** que se dispara por el flanco de pendiente positiva de la señal del reloj. Esto significa que el FF puede cambiar de estado *sólo* cuando una señal que se aplica a su entrada de reloj realiza la transición de 0 a 1. Las entradas *S* y *R* controlan el estado del FF de la misma forma como se describió antes para el latch de compuerta NOR, pero el FF no responde a estas entradas sino hasta que ocurre la PGT en la señal del reloj.

La tabla de funciones de la figura 5-19(b) muestra cómo responderá la salida del FF a la PGT en la entrada *CLK* para las diversas combinaciones de las entradas *S* y *R*. Esta tabla de funciones utiliza cierta nomenclatura nueva. La flecha hacia arriba (a) indica que se requiere una PGT en *CLK*; la etiqueta Q_0 indica el nivel en *Q* antes de la PGT. Esta nomenclatura se utiliza a menudo por los fabricantes de circuitos integrados en las hojas de datos de los CIs.

Las formas de onda de la figura 5-19(c) ilustran la operación del flip-flop S-R sincronizado por reloj. Si suponemos que se están cumpliendo los requerimientos para el tiempo de estabilización y el tiempo de retención en todos los casos, podremos analizar estas formas de onda de la siguiente manera:

1. Al principio todas las entradas son 0 y se asume que la salida *Q* es 0; esto es, $Q_0 = 0$.
2. Cuando ocurre la PGT del primer pulso de reloj (punto *a*) las entradas *S* y *R* son ambas 0, por lo que el FF no se ve afectado y permanece en el estado $Q = 0$ (es decir, $Q = Q_0$).
3. Al ocurrir la PGT del segundo pulso de reloj (punto *c*) la entrada *S* ahora está en nivel ALTO, mientras que *R* sigue en BAJO. Por ende, el FF se establece para quedar en el estado 1 durante el flanco de subida de este pulso de reloj.
4. Cuando el tercer pulso de reloj produce su transición positiva (punto *e*), encuentra que $S = 0$ y $R = 1$, lo cual hace que el FF se borre para quedar en el estado 0.

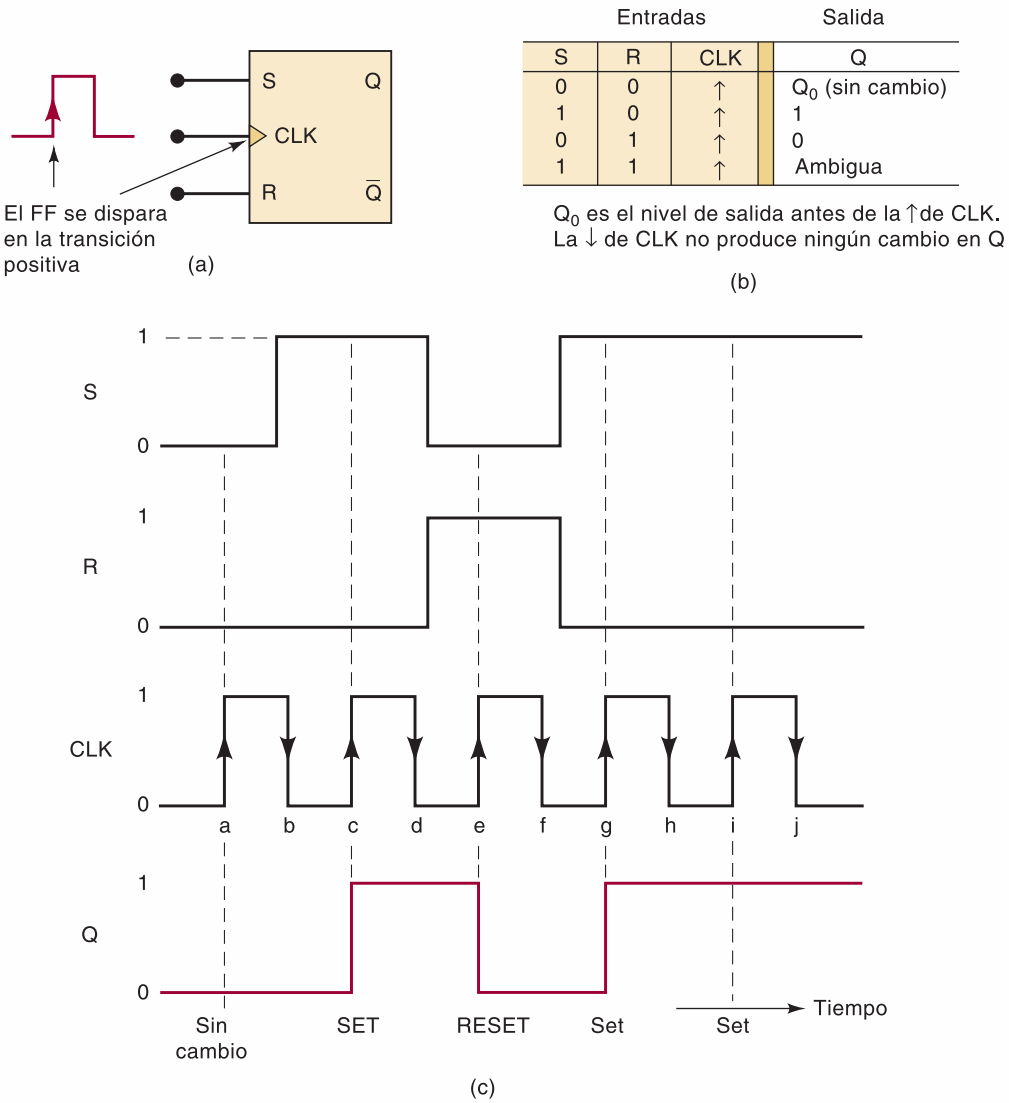
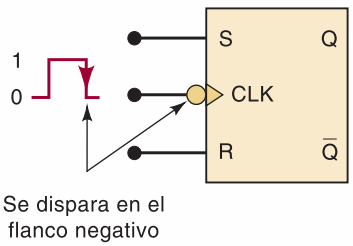


FIGURA 5-19 (a) Flip-flop S-R sincronizado por reloj que responde sólo al flanco positivo de un pulso de reloj; (b) tabla de funciones; (c) formas de onda comunes.

5. El cuarto pulso establece el FF una vez más para quedar en el estado $Q = 1$ (punto *g*), ya que $S = 1$ y $R = 0$ cuando ocurre el flanco positivo.
6. El quinto pulso también encuentra que $S = 1$ y $R = 0$ cuando realiza su transición de pendiente positiva. No obstante, Q ya se encuentra en nivel ALTO por lo que permanece en ese estado.
7. La condición $S = R = 1$ no debe utilizarse, ya que produce una condición ambigua.

Hay que recalcar de estas formas de onda que el FF no se ve afectado por las transiciones de pendiente negativa de los pulsos del reloj. Además, observe que los niveles de *S* y de *R* no tienen efecto sobre el FF, excepto al momento en que ocurre una transición de pendiente positiva en la señal del reloj. Las entradas *S* y *R* son entradas de *control* síncronas; controlan el estado al que cambiará el FF cuando ocurra el pulso de reloj. La entrada *CLK* es la entrada de **disparo**, la cual hace que el FF cambie de estado de acuerdo con el nivel de las entradas *S* y *R* cuando ocurre la transición activa del reloj.

La figura 5-20 muestra el símbolo y la tabla de funciones para un flip-flop sincronizado por reloj en S-R que se dispara con la transición de pendiente *negativa* en su entrada *CLK*. El pequeño círculo y el pequeño triángulo en la entrada *CLK*



Entradas			Salida
S	R	CLK	Q
0	0	↓	Q_0 (sin cambio)
1	0	↓	1
0	1	↓	0
1	1	↓	Ambigua

FIGURA 5-20 Flip-flop sincronizado por reloj en S-R que se dispara sólo en las transiciones de pendiente negativa.

indican que este FF se disparará sólo cuando la entrada *CLK* cambie de 1 a 0. Este FF opera de la misma forma que el FF de flanco positivo, con la diferencia de que la salida puede cambiar de estado sólo en el flanco negativo de los pulsos de reloj (puntos *b*, *d*, *f*, *h* y *j* en la figura 5-19). Tanto el FF de disparo en flanco positivo como el de disparo en flanco negativo se utilizan en los sistemas digitales.

Circuitos internos del flip-flop sincronizado por reloj en S-R

No es necesario un análisis detallado de los circuitos internos de un FF sincronizado por reloj, ya que todos los tipos se encuentran disponibles como CIs. Aunque nuestro interés principal es en la operación externa del FF, nuestra comprensión de esta operación externa puede mejorar si analizamos una versión simplificada de los circuitos internos del FF. La figura 5-21 muestra lo anterior para un flip-flop S-R disparado por flanco.

El circuito contiene tres secciones:

1. Un latch de compuerta NAND básico, formado por las compuertas NAND-3 y NAND-4.
2. Un **circuito de conducción de pulso**, formado por las compuertas NAND-1 y NAND-2.
3. Un **circuito detector de flancos.**

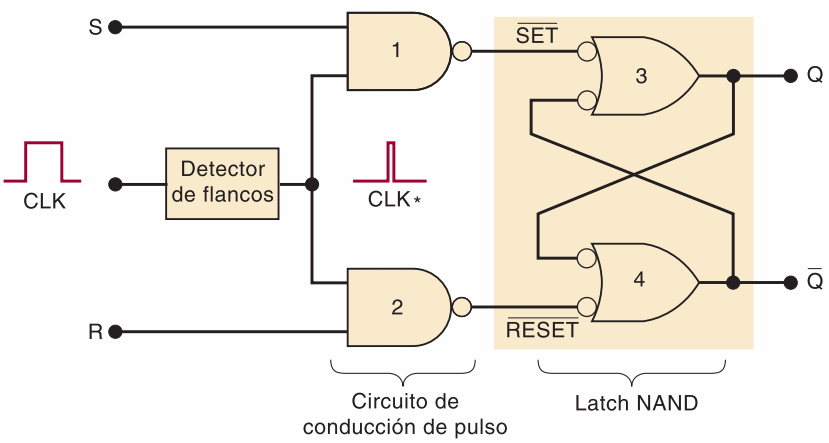


FIGURA 5-21 Versión simplificada de los circuitos internos para un flip-flop sincronizado por reloj en S-R.

Como muestra la figura 5-21, el detector de flancos produce un pico estrecho de pendiente positiva (*CLK*⋆), el cual ocurre en forma coincidente con la transición activa del pulso de la entrada *CLK*. El circuito de conducción de pulso "guía" el pico a través de la entrada SET o de la entrada RESET del latch, de acuerdo con los niveles presentes en *S* y *R*. Por ejemplo, con $S = 1$ y $R = 0$ la señal *CLK*⋆ se invierte y se pasa a través de la compuerta NAND-1 para producir un pulso BAJO en la entrada SET del latch, la cual establece $Q = 1$. Con $S = 0$, $R = 1$ la señal *CLK*⋆ se invierte y se pasa a través de la compuerta NAND-2 para producir un pulso BAJO en la entrada RESET del latch, la cual restablece a $Q = 0$.

La figura 5-22(a) muestra cómo se genera la señal *CLK*⋆ para los FFs disparados por flanco que se disparan con una PGT. El INVERSOR produce un retraso de unos cuantos nanosegundos, de manera que las transiciones de $\overline{CLK}$ ocurran un poco

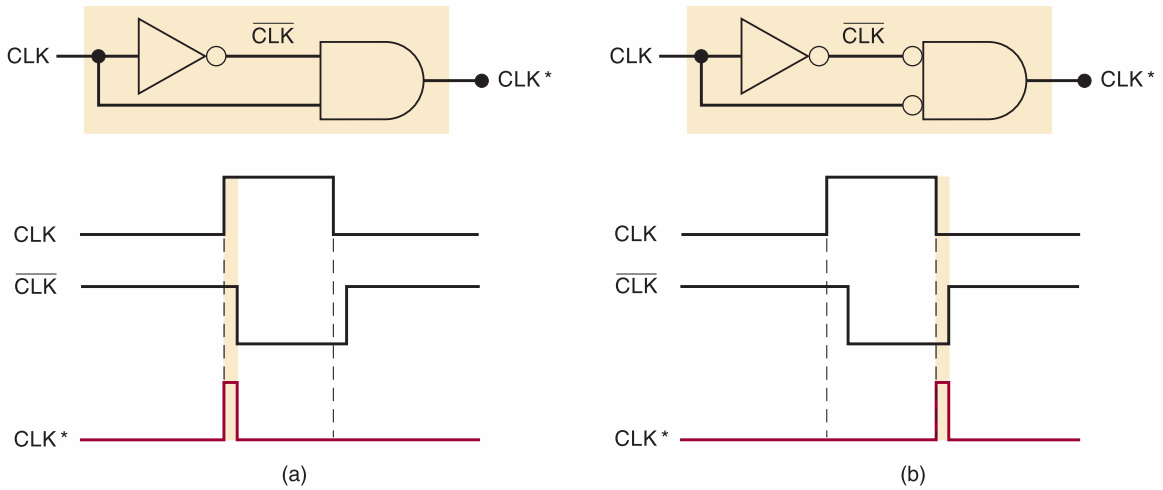


FIGURA 5-22 Implementación de circuitos detectores de flancos que se utilizan en flip-flops disparados por flanco: (a) PGT; (b) NGT. La duración de los pulsos en CLK* es por lo general de 2 a 5 ns.

después de las de $\overline{CLK}$. La compuerta AND produce un pulso de salida que está en nivel ALTO sólo durante los pocos nanosegundos cuando *CLK* y $\overline{CLK}$ se encuentran ambas en ALTO. El resultado es un pulso estrecho en *CLK**, que ocurre en la PGT de *CLK*. El arreglo de la figura 5-22(b) produce de igual forma a *CLK** en la NGT de *CLK* para los FFs que se disparan con una NGT.

Como la señal *CLK** está en ALTO durante unos cuantos nanosegundos, *Q* se ve afectada por los niveles en *S* y *R* sólo durante un corto intervalo, durante y después de la ocurrencia del flanco activo de *CLK*. Esto es lo que proporciona al FF su propiedad de dispararse por flancos.

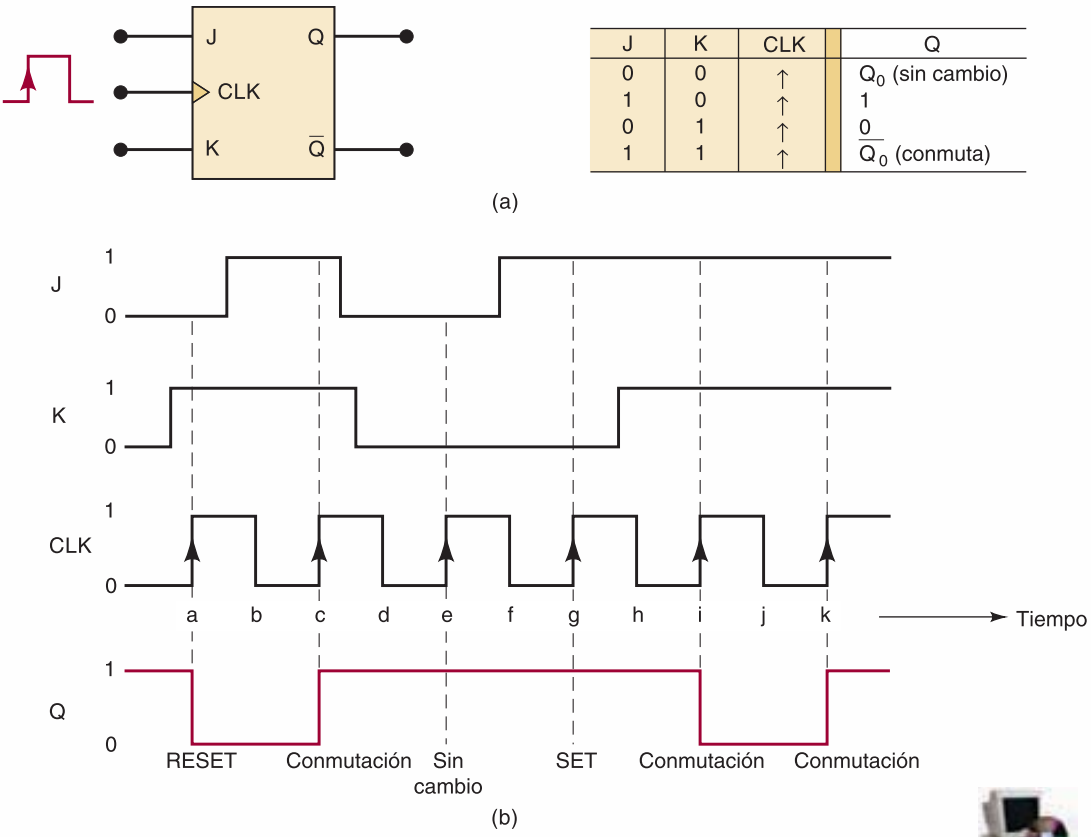
PREGUNTAS DE REPASO

1. Suponga que las formas de onda de la figura 5-19(c) se aplican a las entradas del FF de la figura 5-20. ¿Qué ocurrirá con *Q* en el punto *b*? ¿En el punto *f*? ¿En el punto *h*?
2. Explique por qué las entradas *S* y *R* afectan a *Q* sólo durante la transición activa de *CLK*.

5-7 FLIP-FLOP SINCRONIZADO POR RELOJ EN J-K

La figura 5-23(a) muestra un **flip-flop sincronizado por reloj en J-K**, el cual se dispara por el flanco de pendiente positiva de la señal de reloj. Las entradas *J* y *K* controlan el estado del FF de la misma forma que las entradas *S* y *R* controlan el flip-flop S-R, sólo por una gran diferencia: *la condición J = K = 1 no produce una salida ambigua*. Para esta condición de 1, 1, el FF siempre cambiará a su estado opuesto al momento en que ocurra la transición positiva de la señal de reloj. A este modo de operación se le conoce como el **modo de conmutación**. En este modo, si tanto *J* como *K* se dejan en ALTO, el FF cambiará de estado (conmutará) para cada PGT del reloj.

La tabla de funciones de la figura 5-23(a) sintetiza la forma en que el flip-flop J-K responde a la PGT para cada combinación de *J* y de *K*. Observe que la tabla de funciones es la misma que para el flip-flop sincronizado por reloj en S-R (figura 5-19), excepto por la condición $J = K = 1$. Esta condición produce $Q = \overline{Q}_0$, lo cual significa que el nuevo valor de *Q* será el inverso del valor que tenía antes de la PGT; ésta es la operación de conmutación.



J	K	CLK	Q
0	0	↑	Q_0 (sin cambio)
1	0	↑	1
0	1	↑	0
1	1	↑	$\overline{Q}_0$ (conmuta)

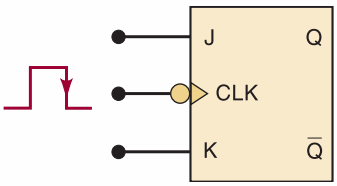
FIGURA 5-23 (a) Flip-flop sincronizado por reloj en J-K que responde sólo al flanco positivo del reloj; (b) formas de onda.

La operación de este FF se ilustra en la figura 5-23(b) mediante las formas de onda. Una vez más, suponemos que se han cumplido los requerimientos para el tiempo de estabilización y el de retención.

1. Al principio todas las entradas son 0 y se asume que la salida Q es 1; esto es, $Q_0 = 1$.
2. Cuando ocurre el flanco de pendiente positiva del primer pulso de reloj (punto *a*), existe la condición $J = 0, K = 1$. Por ende, el FF se restablecerá para quedar en el estado $Q = 0$.
3. El segundo pulso de reloj encuentra que $J = K = 1$ cuando realiza su transición positiva (punto *c*). Esto hace que el FF *conmute* a su estado opuesto, $Q = 1$.
4. En el punto *e* en la forma de onda del reloj, J y K son ambas 0 por lo que el FF no cambia de estado en esta transición.
5. En el punto *g*, $J = 1$ y $K = 0$. Ésta es la condición que establece Q para quedar en el estado 1. No obstante, ya se encuentra en 1 por lo que permanecerá ahí.
6. En el punto *i*, $J = K = 1$ por lo que el FF conmuta a su estado opuesto. Lo mismo ocurre en el punto *k*.

De estas formas de onda podemos observar que el FF no se ve afectado por el flanco de pendiente negativa de los pulsos de reloj. Además, los niveles de las entradas J y K no tienen efecto, excepto cuando ocurre la PGT de la señal del reloj. Las entradas J y K por sí solas no pueden hacer que el FF cambie de estado.

La figura 5-24 muestra el símbolo para un flip-flop sincronizado por reloj en J-K que se dispara con las transiciones de pendiente negativa de la señal del reloj. El pequeño círculo en la entrada *CLK* indica que este FF se disparará cuando la



J	K	CLK	Q
0	0	↓	Q_0 (sin cambio)
1	0	↓	1
0	1	↓	0
1	1	↓	$\overline{Q}_0$ (conmuta)

FIGURA 5-24 Flip-flop J-K que se dispara sólo con las transiciones de pendiente negativa.



entrada *CLK* cambie de 1 a 0. Este FF opera de la misma forma que el FF de flanco positivo de la figura 5-23, excepto que la salida puede cambiar de estado sólo en las transiciones de pendiente negativa de la señal del reloj (puntos *b*, *d*, *f*, *h* y *j*). Ambas polaridades de los flip-flops J-K disparados por flanco son de uso común.

El flip-flop J-K es mucho más versátil que el flip-flop S-R, ya que no tiene estados ambiguos. La condición $J = K = 1$, que produce la operación de conmutación, tiene un uso extenso en todos los tipos de contadores binarios. En esencia, el flip-flop J-K puede hacer cualquier cosa que hace el flip-flop S-R, *además* de operar en el modo de conmutación.

Circuitos internos del flip-flop J-K disparado por flanco

En la figura 5-25 se muestra una versión simplificada de los circuitos internos de un flip-flop J-K disparado por flanco. Contiene las mismas tres secciones que el flip-flop S-R disparado por flanco (figura 5-21). De hecho, la única diferencia entre los dos circuitos es que las salidas Q y $\overline{Q}$ se alimentan de vuelta a las compuertas NAND de conducción de pulso. Esta conexión de retroalimentación es lo que proporciona al flip-flop J-K su operación de conmutación para la condición $J = K = 1$.

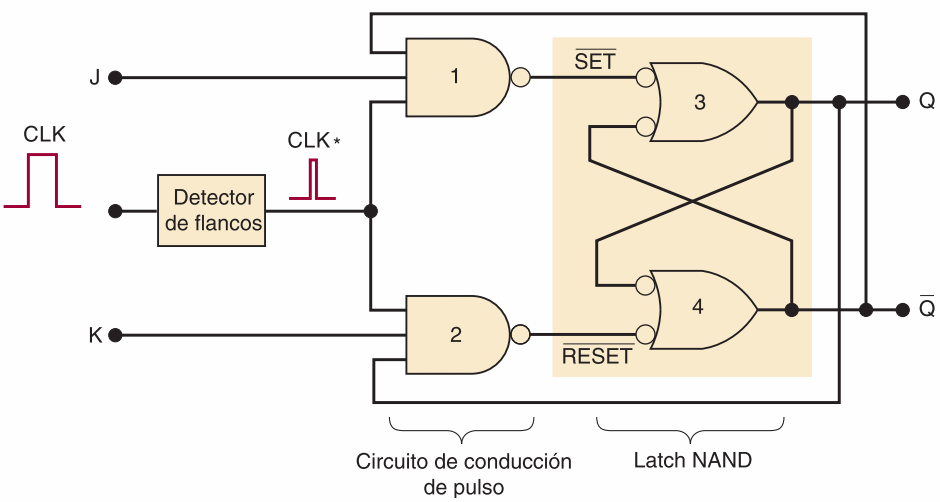


FIGURA 5-25 Circuito interno del flip-flop J-K disparado por flanco.

Examinemos ahora esta condición de conmutación más de cerca, asumiendo que $J = K = 1$ y que Q permanece en el estado BAJO cuando ocurre un pulso en *CLK*. Con $Q = 0$ y $\overline{Q} = 1$, la compuerta NAND 1 dirigirá a *CLK*⋆ (invertida) hacia la entrada $\overline{\text{SET}}$ del latch NAND para producir $Q = 1$. Si asumimos que Q está en ALTO cuando ocurre un pulso en *CLK*, la compuerta NAND 2 dirigirá a *CLK*⋆ (invertida) hacia la entrada $\overline{\text{RESET}}$ del latch para producir $Q = 0$. Por ende, Q siempre terminará en el estado opuesto.

Para que la operación de conmutación funcione como se describió antes, el pulso en *CLK*⋆ debe ser muy estrecho. Debe regresar a 0 antes de que las salidas Q y $\overline{Q}$ conmuten a sus nuevos valores; de no ser así, los nuevos valores de Q y $\overline{Q}$ harán que el pulso *CLK*⋆ conmute de nuevo las salidas del latch.

PREGUNTAS DE REPASO

1. *Verdadero o falso:* un flip-flop J-K puede usarse como un flip-flop S-R, pero un flip-flop S-R no puede usarse como un flip-flop J-K.
2. ¿Tiene un flip-flop J-K condiciones de entrada ambiguas?
3. ¿Qué condición de entrada en J-K siempre establecerá Q=1 cuando ocurra la transición activa en *CLK*?

5-8 FLIP-FLOP SINCRONIZADO POR RELOJ EN D

La figura 5-26(a) muestra el símbolo y la tabla de funciones para un **flip-flop sincronizado por reloj en D** que se dispara con una PGT. A diferencia de los flip-flops S-R y J-K, este flip-flop sólo tiene una entrada de control síncrona D (la cual significa *datos*). La operación del flip-flop D es muy simple: Q cambiará al mismo estado que esté presente en la entrada D cuando ocurra una PGT en *CLK*. En otras palabras, el nivel presente en D se *almacenará* en el flip-flop en el instante en que ocurra la PGT. Las formas de onda de la figura 5-26(b) ilustran esta operación.

Suponga que al principio Q está en nivel ALTO. Cuando ocurre la primera PGT en el punto *a*, la entrada D está en BAJO; por ende, Q cambiará al estado 0. Aún y cuando el nivel de la entrada D cambie entre los puntos *a* y *b*, no tiene efecto sobre Q; Q está almacenando el nivel BAJO que se encontraba en D en el punto *a*. Cuando ocurre la PGT en *b*, Q cambia a ALTO porque D está en ALTO en ese momento. Q almacena este nivel ALTO hasta que la PGT en el punto *c* hace que Q cambie a BAJO, ya que D se encuentra en BAJO en ese momento. De una manera similar, la salida Q toma los niveles presentes en D cuando ocurren las PGTs en los puntos *d*, *e*, *f* y *g*. Observe que Q permanece en ALTO en el punto *e*, porque D aún está en ALTO.

De nuevo, es importante recordar que Q puede cambiar sólo cuando ocurre una PGT. La entrada D no tiene efecto entre las PGTs.

Un flip-flop D disparado por flanco negativo opera de la misma forma antes descrita, sólo que Q tomará el valor de D cuando ocurra una NGT en *CLK*. El símbolo para el flip-flop D que se dispara con las NGTs tendrá una burbuja en la entrada *CLK*.



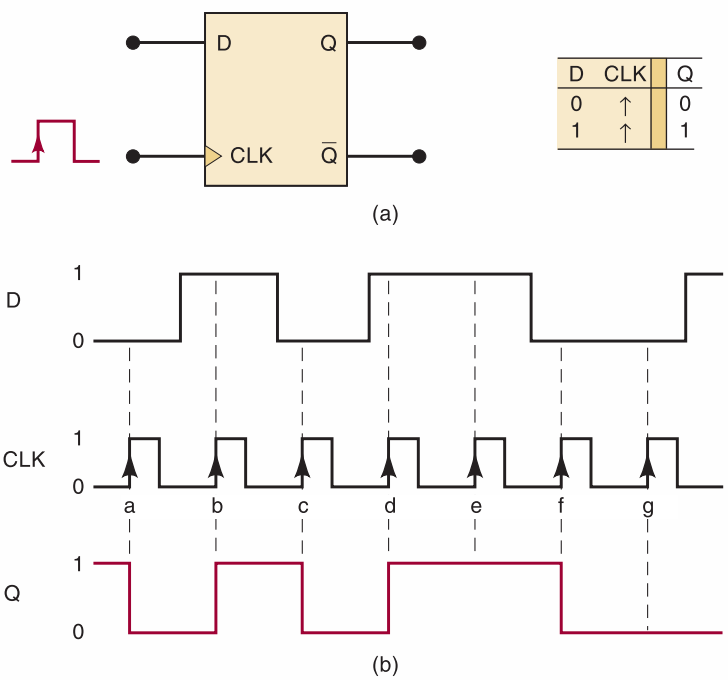


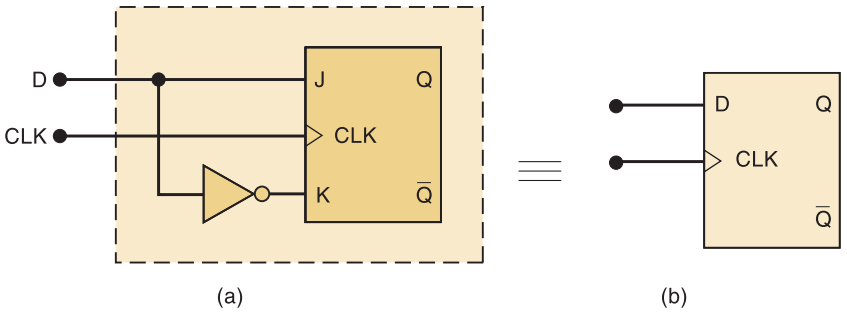
FIGURA 5-26 (a) Flip-flop D que se dispara sólo en las transiciones de pendiente positiva; (b) formas de onda.

Implementación del flip-flop D

Un flip-flop D disparado por flanco se implementa con facilidad con sólo agregar un INVERSOR al flip-flop J-K disparado por flanco, como se muestra en la figura 5-27. Si usted prueba ambos valores de *D*, podrá ver que *Q* toma el nivel presente en *D* cuando ocurre una PGT. Lo mismo puede hacerse para convertir un flip-flop S-R en un flip-flop D.

FIGURA 5-27
Implementación de un flip-flop D disparado por flanco, a partir de un flip-flop J-K.



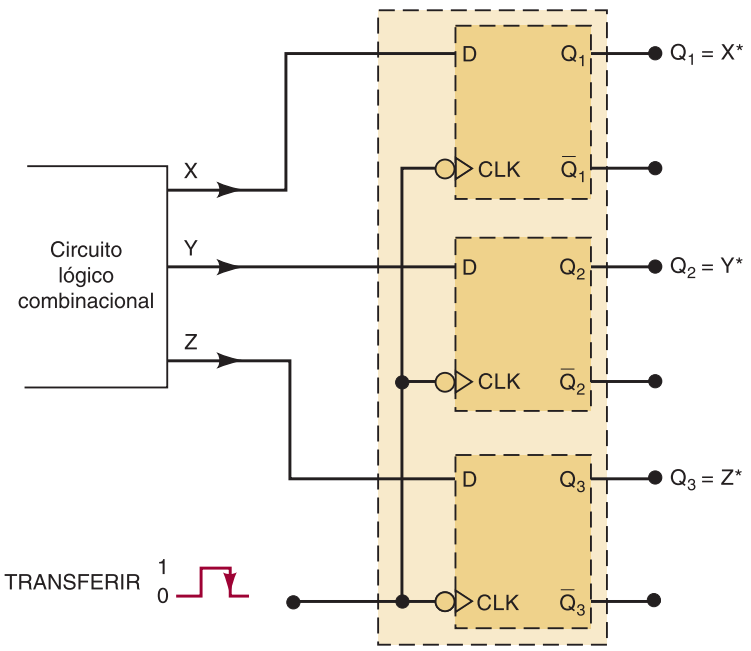


Transferencia de datos en paralelo

En este punto tal vez se esté preguntando acerca de la utilidad del flip-flop D, ya que parece que la salida *Q* es la misma que la entrada *D*. Pero no es exactamente así; recuerde que *Q* toma el valor de *D* sólo en ciertos intervalos de tiempo, por lo cual no es idéntica a *D* (por ejemplo, vea las formas de onda de la figura 5-26).

En la mayoría de las aplicaciones del flip-flop D, la salida *Q* debe tomar el valor de su entrada *D* sólo en tiempos definidos con precisión. Un ejemplo de esto se ilustra en la figura 5-28. Las salidas *X*, *Y*, *Z* de un circuito lógico van a transferirse a los FFs Q_1, Q_2 y Q_3 para su almacenamiento. Si utilizamos los flip-flops D, los niveles presentes en *X*, *Y* y *Z* se transferirán a Q_1, Q_2 y Q_3 en forma respectiva, al momento de aplicar un pulso en TRANSFERIR a las entradas *CLK* comunes. Los FFs pueden almacenar estos valores para su posterior procesamiento. Éste es un ejemplo de la **transferencia de datos en paralelo** para datos binarios; los tres bits *X*, *Y* y *Z* se transfieren en forma *simultánea*.

FIGURA 5-28
Transferencia en paralelo de datos binarios mediante el uso de flip-flops D.



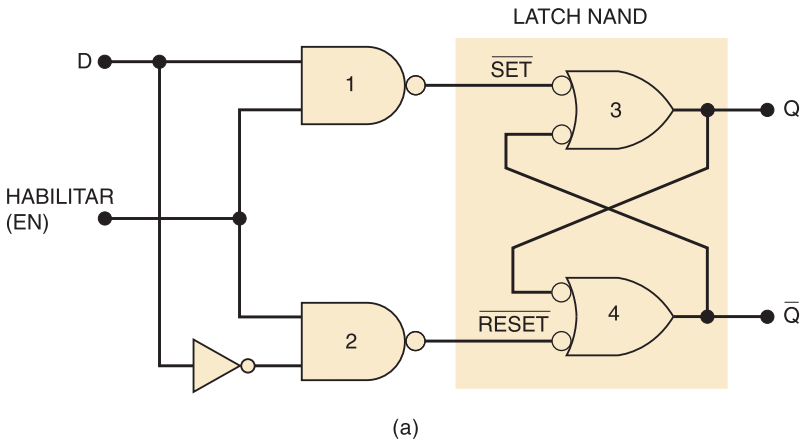
*Después de que ocurre la NGT

PREGUNTAS DE REPASO

1. ¿Qué ocurrirá con la forma de onda de *Q* en la figura 5-26(b) si la entrada *D* se mantiene en BAJO de manera permanente?
2. *Verdadero o falso:* la salida *Q* será igual al nivel en la entrada *D* en todo momento.
3. ¿Pueden usarse los FFs J-K para la transferencia de datos en paralelo?

5-9 LATCH *D* (LATCH TRANSPARENTE)

El flip-flop D disparado por flanco utiliza un circuito detector de flancos para asegurar que la salida responda a la entrada *D sólo* cuando ocurre la transición activa del reloj. Si no se utiliza este detector de flancos, el circuito resultante opera en forma un poco distinta. A éste se le conoce como **latch *D*** y tiene el arreglo que se muestra en la figura 5-29(a).

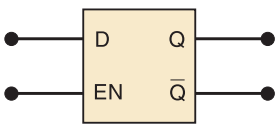


Entradas		Salida
EN	D	Q
0	X	Q_0 (sin cambio)
1	0	0
1	1	1

"X" indica "no importa"
Q_0 es el estado Q justo antes de que EN cambie a BAJO.

(b)





(c)

FIGURA 5-29 Latch *D*: (a) estructura; (b) tabla de funciones; (c) símbolo lógico.

El circuito contiene el latch NAND y las compuertas NAND de dirección 1 y 2 *sin* el circuito detector de flancos. La entrada común para las compuertas de dirección se llama entrada de *habilitación* (se abrevia como *EN*) en vez de una entrada de reloj, ya que su efecto sobre las salidas *Q* y $\overline{Q}$ no se restringe a ocurrir sólo en sus transiciones. La operación del latch *D* se describe a continuación:

1. Cuando *EN* está en ALTO, la entrada *D* producirá un nivel BAJO ya sea en la entrada $\overline{\text{SET}}$ o en la entrada $\overline{\text{RESET}}$ del latch NAND, para hacer que *Q* tenga el mismo nivel que *D*. Si *D* cambia mientras que *EN* está en ALTO, *Q* seguirá los mismos cambios. En otras palabras, mientras que *EN* = 1 la salida *Q* se verá justo igual que *D*; en este modo, se dice que el latch *D* es "transparente".
2. Cuando *EN* cambia a BAJO, la entrada *D* queda inhibida y no puede afectar al latch NAND debido a que las salidas de ambas compuertas de dirección se mantendrán en ALTO. Por ende, las salidas *Q* y $\overline{Q}$ permanecerán en el nivel que tenían justo antes de que *EN* cambiara a BAJO. En otras palabras, las salidas se "enganchan" a su nivel actual y no pueden cambiar mientras que *EN* se encuentre en BAJO, aún si *D* cambia.

Esta operación se sintetiza en la tabla de funciones de la figura 5-29(b). El símbolo lógico para el latch *D* se muestra en la figura 5-29(c). Observe que aún cuando la entrada *EN* opera en forma muy parecida a la entrada *CLK* de un FF disparado por flanco, no hay un pequeño triángulo en la entrada *EN*. Esto se debe a que el símbolo del pequeño triángulo se utiliza estrictamente para entradas que pueden provocar un cambio en la salida sólo cuando ocurre una transición. *El latch D no se dispara por flanco.*

EJEMPLO 5-8

Determine la forma de onda de *Q* para un latch *D* con las entradas *EN* y *D* de la figura 5-30. Asuma que al principio $Q = 0$.

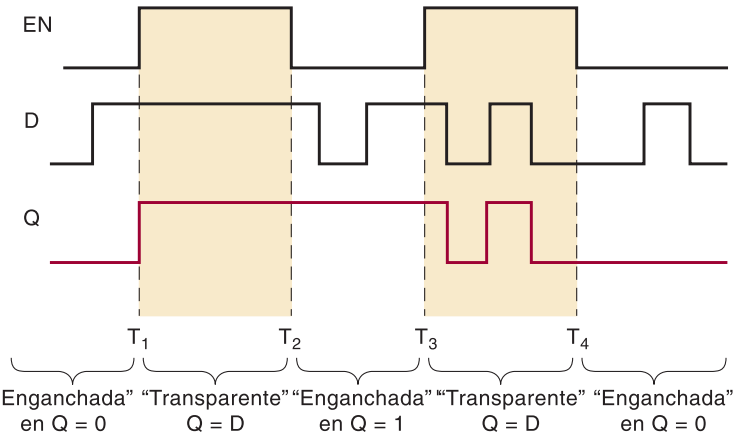


FIGURA 5-30 Formas de onda para el ejemplo 5-8, las cuales muestran los dos modos de operación del latch *D* transparente.

Solución

Antes del tiempo T_1 *EN* está en BAJO, por lo que *Q* está "enganchada" en su nivel 0 actual y no puede cambiar aunque cambie *D*. Durante el intervalo de T_1 a T_2, *EN* está en ALTO, de manera que *Q* seguirá la señal presente en *D*. Por ende, *Q* cambia a ALTO en T_1 y permanece ahí porque *D* no está cambiando. Cuando *EN* regrese a BAJO en T_2, *Q* se enganchará en el nivel ALTO que tiene en T_2 y permanecerá ahí hasta que *EN* cambie a BAJO.

En T_3, cuando *EN* cambia de nuevo a ALTO, *Q* seguirá los cambios en la entrada *D* hasta T_4, cuando *EN* regrese a BAJO. Durante el intervalo de T_3 a T_4, el latch *D* es "transparente" ya que las variaciones en *D* pasan hasta la salida *Q*. En T_4, cuando *EN* cambia a BAJO, *Q* se enganchará en el nivel 0 debido a que es su nivel en T_4. Después de T_4 las variaciones en *D* no tendrán efecto sobre *Q*, ya que está enganchada (es decir, $EN = 0$).

PREGUNTAS DE REPASO

1. Describa cómo un latch *D* opera en forma distinta a un flip-flop *D* disparado por flanco.
2. *Verdadero o falso:* un latch *D* se encuentra en su modo transparente cuando $EN = 0$.
3. *Verdadero o falso:* en un latch *D*, la entrada *D* sólo puede afectar a *Q* cuando $EN = 1$.

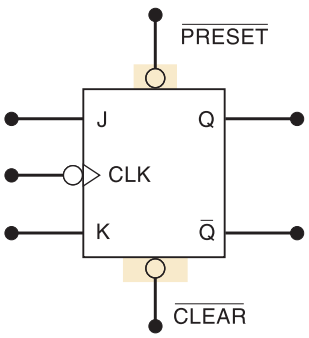
5-10 ENTRADAS ASÍNCRONAS

Para los flip-flops sincronizados por reloj que hemos estudiado, las entradas *S*, *R*, *J*, *K* y *D* se han designado como entradas de *control*. A estas entradas también se les conoce como entradas síncronas, ya que su efecto sobre la salida del FF está sincronizado con la entrada *CLK*. Como hemos visto, las entradas de control síncronas deben usarse en conjunto con una señal de reloj para disparar el FF.

La mayoría de los FFs sincronizados por reloj tienen también una o más **entradas asíncronas** que operan de manera independiente a las entradas síncronas y a la entrada del reloj. Estas entradas asíncronas pueden usarse para establecer el FF al estado 1 o borrar (RESET) el FF al estado 0 *en cualquier momento, sin importar las condiciones en las otras entradas*. Dicho de otra forma, las entradas asíncronas son **entradas predominantes**, las cuales pueden usarse para ignorar todas las demás entradas y colocar el FF en un estado u otro.

La figura 5-31 muestra un flip-flop J-K con dos entradas asíncronas designadas como $\overline{\text{PRESET}}$ y $\overline{\text{CLEAR}}$, que son entradas activas en BAJO, como lo indican las burbujas en el símbolo del FF. La tabla de funciones que se muestra sintetiza la manera en que afectan a la salida del FF. Vamos a examinar los diversos casos.





J	K	Clk	$\overline{\text{PRE}}$	$\overline{\text{CLR}}$	Q
0	0	↓	1	1	Q (sin cambio)
0	1	↓	1	1	0 (RESET síncrona)
1	0	↓	1	1	1 (SET síncrona)
1	1	↓	1	1	$\bar{Q}$ (Conmutación síncrona)
x	x	x	1	1	Q (sin cambio)
x	x	x	1	0	0 (CLEAR asíncrona)
x	x	x	0	1	1 (PRESET síncrona)
x	x	x	0	0	(Inválida)

FIGURA 5-31 Flip-flop sincronizado por reloj en J-K con entradas asíncronas.

- $\overline{\text{PRESET}} = \overline{\text{CLEAR}} = 1$. Las entradas asíncronas están inactivas y el FF es libre de responder a las entradas *J*, *K* y *CLK*; en otras palabras, puede llevarse a cabo la operación con sincronización por reloj.
- $\overline{\text{PRESET}} = 0$; $\overline{\text{CLEAR}} = 1$. La señal $\overline{\text{PRESET}}$ está activa y *Q* se establece en forma *inmediata* en 1, sin importar qué condiciones estén presentes en las entradas *J*, *K* y *CLK*. La entrada *CLK* no puede afectar al FF mientras que $\overline{\text{PRESET}} = 0$.
- $\overline{\text{PRESET}} = 1$; $\overline{\text{CLEAR}} = 0$. La señal $\overline{\text{CLEAR}}$ se activa y *Q* se borra *de inmediato* para quedar en 0, sin importar las condiciones en las entradas *J*, *K* o *CLK*. La entrada *CLK* no tiene efecto mientras que $\overline{\text{CLEAR}} = 0$.
- $\overline{\text{PRESET}} = \overline{\text{CLEAR}} = 0$. Esta condición no debe utilizarse, ya que puede producir una respuesta ambigua.

Es importante tener en cuenta que estas entradas asíncronas responden a los niveles de corriente directa. Esto significa que si se mantiene un 0 constante en la entrada $\overline{\text{PRESET}}$, el FF permanecerá en el estado $Q = 1$ sin importar lo que ocurra en las otras entradas. De manera similar, un BAJO constante en la entrada $\overline{CLEAR}$ mantendrá al FF en el estado $Q = 0$. Por ende, las entradas asíncronas pueden utilizarse para mantener el FF en un estado específico durante cualquier intervalo de tiempo deseado. No obstante, es más común utilizar las entradas asíncronas para establecer o borrar el FF para que quede en el estado deseado, mediante la aplicación de un pulso momentáneo.

Muchos FFs sincronizados por reloj que están disponibles como CIs tendrán ambas entradas asíncronas; otros sólo tendrán la entrada $\overline{\text{CLEAR}}$. Algunos FFs tendrán entradas asíncronas que sean activas en ALTO, en lugar de ser activas en BAJO. Para estos FFs, el símbolo del FF no tendrá una burbuja en las entradas asíncronas.

Designaciones para las entradas asíncronas

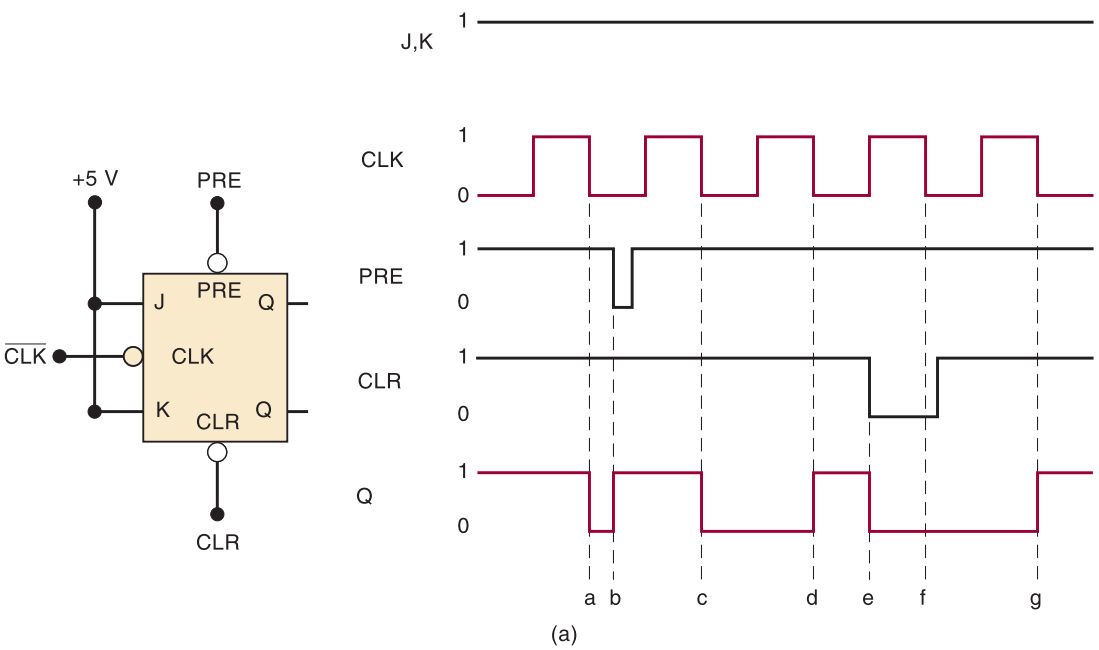
No todos los fabricantes de circuitos integrados están de acuerdo en la nomenclatura a usar para estas entradas asíncronas. Las designaciones más comunes son *PRE* (abreviación de PRESET) y *CLR* (abreviación de CLEAR). Estas etiquetas las

distinguen con claridad de las entradas SET y RESET síncronas. También se utilizan otras etiquetas tales como S_D (SET directo) y R_D (RESET directo). De ahora en adelante, utilizaremos las etiquetas *PRE* y *CLR* para representar las entradas asíncronas, ya que parecen ser las que se utilizan con más frecuencia. Cuando estas entradas asíncronas sean activas en BAJO, como casi siempre, utilizaremos la barra superior para indicar su estado de activas en BAJO; esto es, $\overline{PRE}$ y $\overline{CLR}$.

Aunque la mayoría de los flip-flops de CI tienen cuando menos una o más entradas asíncronas, hay ciertas aplicaciones de circuitos en las que no se utilizan. En tales casos se mantienen de forma permanente en su nivel inactivo. A menudo, en nuestro uso de los FFs a lo largo del resto de este libro, no mostraremos las entradas asíncronas de un FF que no se utilicen; asumiremos que se mantienen conectadas en forma permanente a su nivel lógico inactivo.

EJEMPLO 5-9

La figura 5-32(a) muestra el símbolo para un FF J-K que responde a una NGT en su entrada de reloj y que tiene entradas asíncronas activas en BAJO. Antes de seguir con el ejemplo, tome nota de la manera en que están etiquetadas las entradas. Primero observe que la señal de reloj que se aplica al FF está etiquetada como *CLK* (la barra superior indica que esta señal se activa en la NGT), mientras que en el otro lado de la burbuja (dentro del bloque) se etiqueta como $\overline{CLK}$. De igual forma,



(a)

Punto	Operación
a	Conmutación síncrona en NGT de $\overline{CLK}$
b	SET asíncrona en $\overline{PRE}$ = 0
c	Conmutación síncrona
d	Conmutación síncrona
e	CLEAR asíncrona en $\overline{CLR}$ = 0
f	$\overline{CLR}$ predomina sobre la NGT de $\overline{CLK}$
g	Conmutación síncrona

(b)



FIGURA 5-32 Formas de onda para el ejemplo 5-9 que muestran cómo responde un flip-flop sincronizado por reloj a sus entradas asíncronas.

las entradas asíncronas externas activas en BAJO se etiquetan como $\overline{PRE}$ y $\overline{CLR}$, mientras que dentro del bloque al otro lado de la burbuja se etiquetan como *PRE* y *CLR*. El punto importante a recordar es que la presencia de la burbuja en una entrada indica que esa entrada responde a una señal de nivel BAJO.

Las entradas *J* y *K* se muestran enlazadas al nivel ALTO en este ejemplo. Determine la salida *Q* en respuesta a las formas de onda de entrada que se muestran en la figura 5-32(a), Suponga que al principio *Q* está en ALTO.

Solución

Al principio, $\overline{PRE}$ y $\overline{CLR}$ se encuentran en su estado ALTO inactivo, por lo que no tendrán efecto sobre *Q*. Por ende, cuando ocurra la primera NGT de la señal $\overline{CLK}$ en el punto *a*, *Q* conmutará a su estado opuesto; recuerde que $J = K = 1$ produce la operación de conmutación.

En el punto *b*, se aplica un pulso a la entrada $\overline{PRE}$ para que cambie a su estado de activa en BAJO. Esto hará que $Q = 1$ *de inmediato*. Observe que $\overline{PRE}$ produce $Q = 1$ sin esperar una NGT en $\overline{CLK}$. Las entradas asíncronas operan en forma independiente de $\overline{CLK}$.

En el punto *c*, la NGT de $\overline{CLK}$ hará de nuevo que *Q* conmute a su estado opuesto. Observe que $\overline{PRE}$ ha regresado a su estado inactivo antes del punto *c*. De igual forma, la NGT de $\overline{CLK}$ en el punto *d* conmutará a *Q* de vuelta al nivel ALTO.

En el punto *e*, se aplica un pulso a la entrada $\overline{CLR}$ para que cambie a su estado de activa en BAJO y $Q = 0$ *de inmediato*. De nuevo, esto se hace en forma independiente de $\overline{CLK}$.

La NGT de $\overline{CLK}$ en el punto *f no conmutará a Q*, ya que la entrada $\overline{CLR}$ aún está activa. El nivel BAJO en $\overline{CLR}$ predomina sobre la entrada $\overline{CLK}$ y mantiene a $Q = 0$.

Cuando ocurra la NGT de $\overline{CLK}$ en el punto *g*, conmutará a *Q* al estado ALTO, ya que ninguna de las entradas asíncronas está activa en ese punto.

Estos pasos se sintetizan en la figura 5-32(b).

PREGUNTAS DE REPASO

1. ¿Cómo difiere la operación de una entrada asíncrona en comparación con una entrada síncrona?
2. ¿Puede un flip-flop D responder a sus entradas *D* y *CLK* mientras $\overline{PRE} = 1$?
3. Liste las condiciones necesarias para un flip-flop J-K disparado por flanco positivo, con entradas asíncronas activas en BAJO para conmutarlo a su estado opuesto.

5-11 SÍMBOLOS IEEE/ANSI

La figura 5-33(a) muestra el símbolo IEEE/ANSI para un flip-flop J-K disparado por flanco con entradas asíncronas. Observe el triángulo recto en la entrada CLK para indicar que se activa mediante una NGT. Recuerde que en los símbolos IEEE/ANSI un triángulo recto tiene el mismo significado que la pequeña burbuja en los símbolos tradicionales. Observe además que la entrada de reloj se etiqueta como "C" para denotar cualquier entrada que *controla* cuando otras entradas van a afectar a la salida. Las entradas $\overline{PRE}$ y $\overline{CLR}$ son activas en BAJO, según lo indican los triángulos rectos en estas entradas. Los símbolos IEEE/ANSI también utilizan las etiquetas "S" y "R" dentro del rectángulo para denotar las operaciones SET y RESET asíncronas, que son las mismas que PRESET y CLEAR respectivamente.

La figura 5-33(b) muestra el símbolo lógico IEEE/ANSI para un CI que forma parte de la serie 74LS de dispositivos TTL. El 74LS112 es un CI con dos flip-flop J-K disparados por flanco negativo, con capacidades de prestablecer (PRESET) y borrar (CLEAR). Contiene dos flip-flops J-K, igual al que se simboliza en la figura 5-33(a).

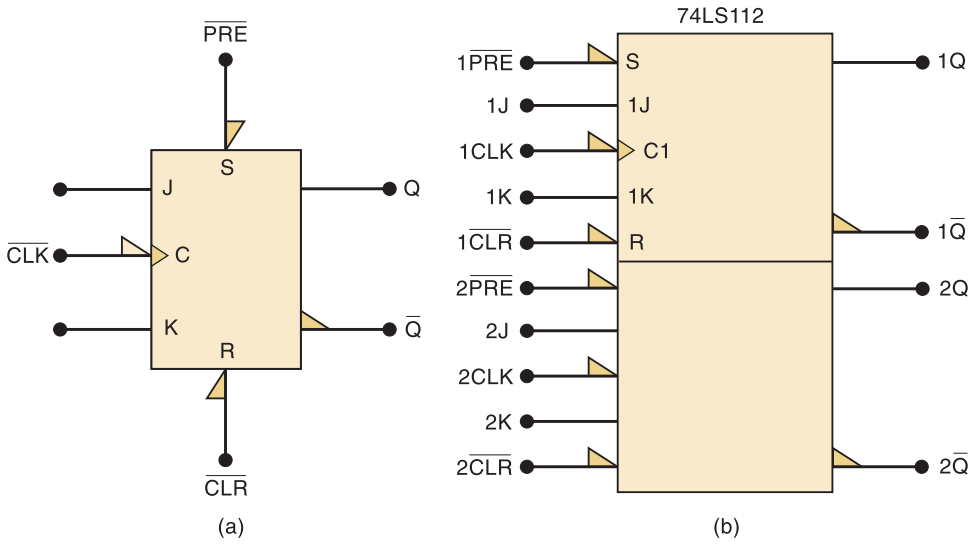


FIGURA 5-33
Símbolos IEEE/ANSI para (a) un solo flip-flop J-K disparado por flanco y (b) un CI real (dos flip-flop J-K disparados por flanco negativo 74LS112).

Observe cómo las entradas y salidas están numeradas. Observe además que las etiquetas de las entradas dentro de los rectángulos se muestran sólo para el FF superior. Se sobreentiende que las entradas para el FF inferior tienen el mismo arreglo que el FF superior. Este mismo símbolo de CI se aplica para el CMOS 74HC112.

La figura 5-34(a) es el símbolo IEEE/ANSI para un flip-flop D disparado por flanco positivo con entradas asíncronas. No hay un triángulo recto en la entrada del reloj debido a que este FF se sincroniza mediante las PGTs del reloj.

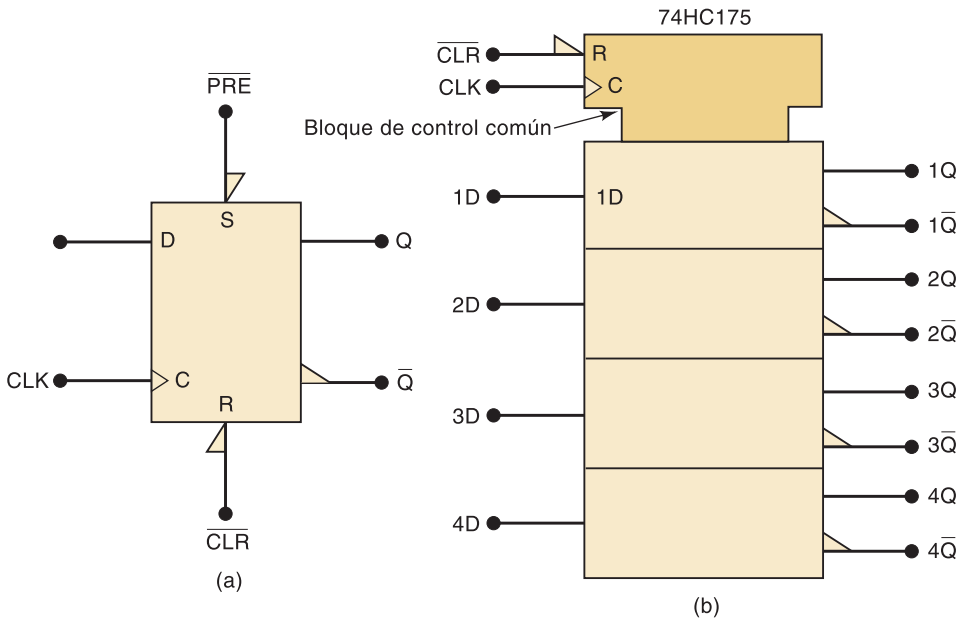


FIGURA 5-34
Símbolos IEEE/ANSI para (a) un solo flip-flop D disparado por flanco y (b) un CI actual (flip-flop cuádruple 74HC175 con señales de reloj y CLEAR comunes).

La figura 5-34(b) es el símbolo IEEE/ANSI para un CI 74HC175, el cual contiene cuatro flip-flops D que comparten una entrada *CLK* común y una entrada $\overline{CLR}$ común. Los FFs no tienen entrada $\overline{PRE}$. Este símbolo contiene un rectángulo separado para representar cada FF y un **bloque de control común** especial, identificado por el rectángulo con muescas en la parte superior. El bloque de control común se utiliza siempre que un CI tiene una o más entradas que son comunes para más de uno de los circuitos dentro del chip. Para el 74HC175, las entradas *CLK* y $\overline{CLR}$ son

comunes para los cuatro flip-flops D en el CI. Esto significa que una PGT en *CLK* hará que cada salida *Q* tome el nivel presente en su entrada *D*; también significa que un nivel BAJO en $\overline{CLR}$ borrará todas las salidas *Q* para que queden en el estado BAJO.

PREGUNTAS DE REPASO

1. Explique el significado de los dos triángulos distintos que pueden formar parte de la simbología IEEE/ANSI en una entrada de reloj.
2. Describa el significado del bloque de control común.

5-12 CONSIDERACIONES DE SINCRONIZACIÓN DE LOS FLIP-FLOPS

Los fabricantes de flip-flops en circuitos integrados especificarán varios parámetros de sincronización y características importantes que deben considerarse antes de utilizar un FF en cualquier aplicación de circuitos. Describiremos las más importantes y después proporcionaremos algunos ejemplos reales de flip-flops en CIs específicos de las familias lógicas TTL y CMOS.

Tiempos de estabilización y de retención

Ya hemos hablado sobre estos tiempos, y tal vez recuerde de la sección 5-5 que representan requerimientos que deben cumplirse para que un FF pueda dispararse en forma confiable. La hoja técnica del fabricante del CI siempre especificará los valores *mínimos* para t_S y t_H.

Tiempos de propagación

Cada vez que una señal va a cambiar el estado de la salida de un FF, hay un retraso a partir del tiempo en el que se aplica la señal hasta el tiempo en el que la salida realiza su cambio. La figura 5-35 ilustra los **retrasos por tiempos de propagación** que se producen en respuesta a una transición positiva en la entrada *CLK*. Observe que estos retrasos se miden entre los puntos del 50 por ciento en las formas de onda de entrada y de salida. Los mismos tipos de retrasos ocurren en respuesta a las señales en las entradas asíncronas de un FF (PRESET y CLEAR). Las hojas técnicas de los fabricantes, por lo general, especifican los tiempos de propagación en respuesta a todas las entradas, y es común que especifiquen los valores *máximos* para t_{PLH} y t_{PHL}.

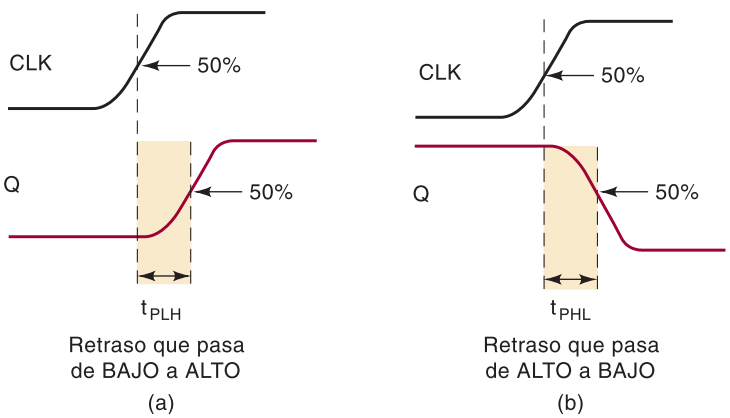


FIGURA 5-35 Tiempos de propagación de un FF.

Los flip-flops de CI modernos tienen tiempos de propagación que varían desde unos cuantos nanosegundos hasta un valor aproximado de 100 ns. Por lo general, los valores de t_{PLH} y t_{PHL} no son iguales, y se incrementan en proporción directa al número de cargas que controla la salida Q. Los tiempos de propagación de los FFs juegan un papel importante en ciertas situaciones que veremos más adelante.

Frecuencia máxima de reloj, $f_{MÁX}$

Ésta es la frecuencia más alta que puede aplicarse a la entrada *CLK* de un FF y que éste pueda dispararse en forma confiable. El límite de $f_{MÁX}$ variará de un FF a otro, incluso entre FFs que tengan el mismo número de dispositivo. Por ejemplo, el fabricante del CI 7470 con flip-flops J-K prueba muchos de estos FFs y puede llegar a encontrar que los valores de $f_{MÁX}$ caen en el intervalo de 20 a 35 MHz. Entonces especificará el valor *mínimo* de $f_{MÁX}$ como de 20 MHz. Esto puede parecer confuso, pero un poco de análisis nos dejará claro que lo que el fabricante está diciendo es que no puede garantizar que el FF 7470 que usted ponga en su circuito vaya a trabajar por encima de 20 MHz; la mayoría de ellos lo hará pero algunos no. No obstante, si usted los opera por debajo de 20 MHz, el fabricante garantiza que todos funcionarán.

Tiempos de pulso de reloj en ALTO y en BAJO

El fabricante también especificará la *mínima* duración de tiempo que la señal *CLK* debe permanecer en BAJO antes de cambiar a ALTO; a esto algunas veces se le conoce como $t_W(L)$. Al tiempo *mínimo* que *CLK* debe permanecer en ALTO antes de regresar a BAJO se le conoce algunas veces como $t_W(H)$. Estos tiempos se definen en la figura 5-36(a). No cumplir con estos requerimientos de tiempo mínimo puede provocar un disparo no confiable. Observe que estos valores de tiempo se miden entre los puntos intermedios en las transiciones de la señal.

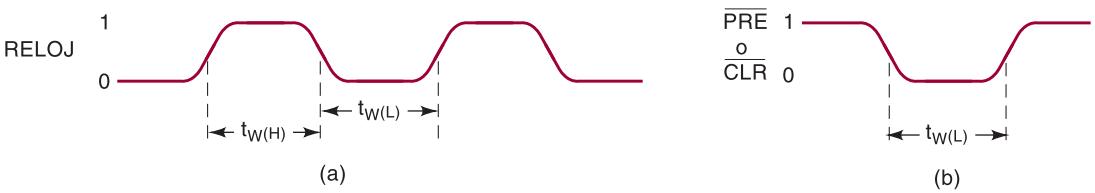


FIGURA 5-36 (a) Tiempos para los niveles BAJO y ALTO del reloj; (b) anchura de pulso asíncrono.

Anchura de pulso activo asíncrono

El fabricante también especificará la *mínima* duración de tiempo que debe mantenerse una entrada PRESET o CLEAR en su estado activo, para poder establecer o borrar el FF en forma confiable. La figura 5-36(b) muestra el valor de $t_W(L)$ para las entradas asíncronas activas en BAJO.

Tiempos de transición del reloj

Para un disparo confiable, los tiempos de transición de la forma de onda del reloj (tiempos de subida y de caída) deben mantenerse muy cortos. Si la señal del reloj tarda demasiado en hacer las transiciones de un nivel al otro, el FF podría dispararse en forma errática o tal vez no se dispararía. Por lo general, los fabricantes no listan un requerimiento de tiempo de transición mínimo para cada circuito integrado de FF. En vez de ello se proporciona como un requerimiento general para todos los CIs dentro de una familia lógica dada. Por ejemplo, los tiempos de transición deben ser, por lo general, ≤ 50 ns para los dispositivos TTL y ≤ 200 ns para los dispositivos CMOS. Estos requerimientos variarán entre los distintos fabricantes y entre las diversas subfamilias dentro de las extensas familias lógicas TTL y CMOS.

CIs reales

Como ejemplos prácticos de estos parámetros de sincronización, veamos un análisis de varios FFs de circuitos integrados reales. En especial veremos los siguientes CIs:

7474	Dos Flip-flop D, disparados por flanco (TTL estándar).
74LS112	Dos Flip-flop J-K, disparados por flanco (Schottky de baja potencia TTL).
74C74	Dos Flip-flop D, disparados por flanco (CMOS de compuerta metálica).
74HC112	Dos Flip-flop J-K, disparados por flanco (CMOS de alta velocidad).

La tabla 5-2 lista los diversos valores de sincronización para cada uno de estos FFs, a medida que aparecen en los libros de datos del fabricante. Todos los valores listados son *mínimos*, excepto para los tiempos de propagación, que son valores *máximos*. El análisis de la tabla 5-2 nos revela varios puntos interesantes.

TABLA 5-2
Valores de sincronización de los flip-flops (en nanosegundos)

		TTL		CMOS	
		7474	74LS112	74C74	74HC112
t_S		20	20	60	25
t_H		5	0	0	0
t_{PHL}	de *CLK* a *Q*	40	24	200	31
t_{PLH}	de *CLK* a *Q*	25	16	200	31
t_{PHL}	de $\overline{CLR}$ a *Q*	40	24	225	41
t_{PLH}	de $\overline{PRE}$ a *Q*	25	16	225	41
$t_W(L)$	tiempo en BAJO para *CLK*	37	15	100	25
$t_W(H)$	Tiempo en ALTO para *CLK*	30	20	100	25
$t_W(L)$	en $\overline{PRE}$ o $\overline{CLR}$	30	15	60	25
$f_{MÁX}$	en MHz	15	30	5	20

1. Todos los FFs tienen requerimientos muy bajos para t_H; esto es común en la mayoría de los FFs modernos disparados por flanco.
2. La serie 74HC de dispositivos CMOS tiene valores de sincronización que se comparan con los de los dispositivos TTL. La serie 74C es mucho más lenta que la serie 74HC.

EJEMPLO 5-10

De la tabla 5-2 determine lo siguiente.

(a) Suponga que $Q = 0$. ¿Cuánto tiempo puede tardar Q en cambiar a ALTO cuando ocurre una PGT en la entrada *CLK* de un 7474?

(b) Suponga que $Q = 1$. ¿Cuánto tiempo puede tardar Q en cambiar a BAJO, en respuesta a la entrada $\overline{CLR}$ de un 74HC112?

(c) ¿Cuál es el pulso más estrecho que debe aplicarse a la entrada $\overline{CLR}$ del FF 74LS112 para borrar a Q de manera confiable?

(d) ¿Cuál FF en la tabla 5-2 requiere que las entradas de control permanezcan estables *después* de que ocurra la transición activa del reloj?

(e) ¿Para cuáles FFs deben mantenerse estables las entradas de control durante un tiempo mínimo, antes de la transición activa del reloj?

Solución

(a) La PGT hará que Q cambie de BAJO a ALTO. El retraso de *CLK* a Q se lista como $t_{PLH} = 25$ ns para el 7474.

(b) Para el 74HC112, el tiempo requerido para que Q cambie de ALTO a BAJO en respuesta a la entrada $\overline{CLR}$ se lista como $t_{PHL} = 41$ ns.

(c) Para el 74LS112, el pulso más estrecho en la entrada $\overline{CLR}$ se lista como $t_W(L) = 15$ ns.

(d) El 7474 es el único FF en la tabla 5-2 que tiene un requerimiento de tiempo de retención distinto de cero.

(e) Todos los FFs tienen un requerimiento de tiempo de estabilización distinto de cero.

PREGUNTAS DE REPASO

1. ¿Cuáles parámetros de sincronización de los FFs indican el tiempo que tarda la salida Q en responder a una entrada?
2. *Verdadero o falso:* un FF que tiene una clasificación de $f_{MÁX}$ de 25 MHz puede dispararse en forma confiable mediante cualquier forma de onda de pulso en *CLK* con una frecuencia menor a 25 MHz.

5-13 PROBLEMA POTENCIAL DE SINCRONIZACIÓN EN CIRCUITOS CON FF

En muchos circuitos digitales, la salida de un FF se conecta ya sea en forma directa o a través de compuertas lógicas a la entrada de otro FF, y ambos FFs se disparan mediante la misma señal de reloj. Esto presenta un problema potencial de sincronización. Una situación común se ilustra en la figura 5-37, en donde la salida de Q_1 se conecta a la entrada J de Q_2 y ambos FFs se sincronizan por la misma señal en sus entradas *CLK*.

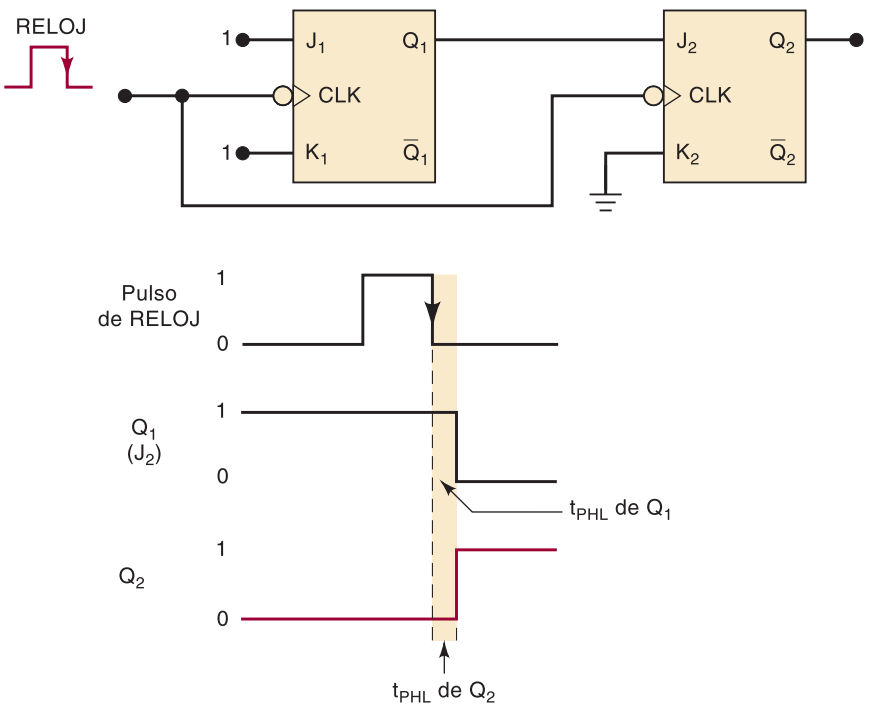


FIGURA 5-37 Q_2 responderá en forma apropiada al nivel presente en Q_1 antes de la NGT de CLK, siempre y cuando el requerimiento de tiempo de retención de Q_2 (t_H) sea menor que el tiempo de propagación de Q_1.

El problema potencial de sincronización es el siguiente: como Q_1 cambiará en la NGT del pulso de reloj, la entrada J_2 de Q_2 cambiará cuando reciba la misma NGT. Esto podría producir una respuesta impredecible en Q_2.

Vamos a suponer que al principio $Q_1 = 1$ y $Q_2 = 0$. Por ende, el FF de Q_1 tiene $J_1 = K_1 = 1$ y Q_2 tiene $J_2 = Q_1 = 1, K_2 = 0$ antes de la NGT del pulso de reloj. Cuando se produce la NGT, Q_1 conmutará al estado BAJO pero no cambiará a BAJO sino hasta después de su retraso de propagación t_{PHL}. La misma NGT cambiará de forma confiable a Q_2 a su estado ALTO siempre y cuando t_{PHL} sea mayor que el requerimiento de tiempo de retención de Q_2, t_H. Si no se cumple esta condición, la respuesta de Q_2 será impredecible.

Por fortuna, todos los FFs modernos disparados por flanco tienen requerimientos de tiempo de retención de 5 ns o menos; la mayoría tiene $t_H = 0$, lo cual significa que no tienen requerimiento de tiempo de retención. Para estos FFs, las situaciones como la de la figura 5-37 no serán problema.

A menos que se indique lo contrario, en todos los circuitos de FF que nos encontremos a lo largo del libro supondremos que el requerimiento de tiempo de retención del FF es lo bastante corto como para responder en forma confiable, de acuerdo con la siguiente regla:

> **La salida del FF cambiará a un estado determinado por los niveles lógicos presentes en sus entradas de control síncronas justo antes de la transición activa del reloj.**

Si aplicamos esta regla a la figura 5-37, nos dice que Q_2 cambiará a un estado determinado por la condición $J_2 = 1, K_2 = 0$, que esté presente justo antes de la NGT del pulso de reloj. El hecho de que J_2 esté cambiando en respuesta a la misma NGT no tiene efecto.

EJEMPLO 5-11

Determine la salida Q para un flip-flop J-K disparado por flanco negativo, para las formas de onda de entrada que se muestran en la figura 5-38. Suponga que al principio $t_H = 0$ y que $Q = 0$.

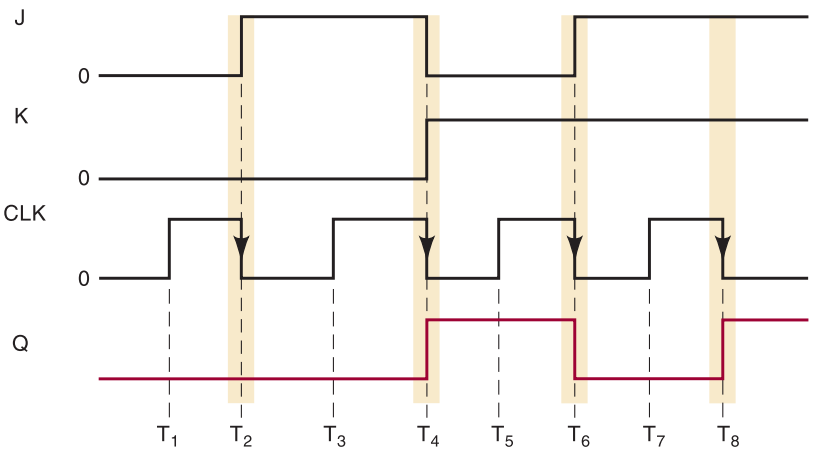


FIGURA 5-38 Ejemplo 5-11.

Solución

El FF responderá sólo en los tiempos T_2, T_4, T_6 y T_8. En T_2, Q responderá a la condición $J = K = 0$ presente justo antes de T_2. En T_4, Q responderá a la condición $J = 1$, $K = 0$ presente justo antes de T_4. En T_6, Q responderá a la condición $J = 0$, $K = 1$ presente justo antes de T_6. Q responde a $J = K = 1$.

5-14 APLICACIONES DE LOS FLIP-FLOPS

Los flip-flops disparados por flanco (sincronizados por reloj) son dispositivos versátiles que pueden usarse en una amplia variedad de aplicaciones, incluyendo el conteo, el almacenamiento de datos binarios, la transferencia de datos binarios desde un lugar hacia otro y muchas más. Casi todas estas aplicaciones utilizan la operación sincronizada del FF. Muchas de ellas entran en la categoría de los **circuitos secuenciales**. En un circuito secuencial las salidas siguen una secuencia predeterminada de estados, en donde un nuevo estado ocurre cada vez que se produce un pulso de reloj. En las siguientes secciones presentaremos algunas de las aplicaciones básicas y las veremos con más detalle en los siguientes capítulos.

5-15 SINCRONIZACIÓN DE LOS FLIP-FLOPS

La mayoría de los sistemas digitales son principalmente síncronos en su operación, ya que la mayoría de las señales cambiará de estado en sincronía con las transiciones del reloj. No obstante, en muchos casos habrá una señal externa que no esté sincronizada con el reloj; en otras palabras, será asíncrona. Las señales asíncronas se presentan con frecuencia, como resultado de la participación de un operador humano que activa un interruptor de entrada durante cierto tiempo aleatorio, relativo a la señal del reloj. Esta característica aleatoria puede producir resultados impredecibles e indeseables. El siguiente ejemplo ilustra cómo puede usarse un FF para sincronizar el efecto de una entrada asíncrona.

EJEMPLO 5-12

La figura 5-39(a) muestra una situación en donde la señal de entrada *A* se genera a partir de un interruptor sin rebotes controlado por un operador (en el ejemplo 5-2 se introdujo por primera vez un interruptor sin rebotes). *A* cambia a ALTO cuando el operador activa el interruptor y cambia a BAJO cuando el operador libera el interruptor. Esta entrada *A* se utiliza para controlar el paso de la señal de reloj hacia la compuerta AND, de manera que aparezcan pulsos de reloj en la salida *X* sólo durante el tiempo que *A* esté en ALTO.

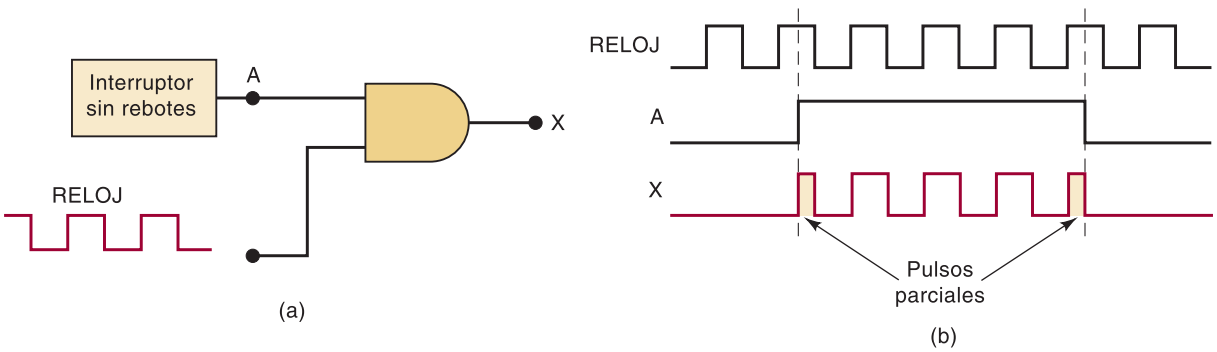


FIGURA 5-39 La señal asíncrona *A* puede producir pulsos parciales en *X*.

El problema con este circuito es que *A* es asíncrona; puede cambiar de estado en cualquier tiempo relativo a la señal del reloj, ya que los tiempos exactos cuando el operador actúa o libera el interruptor son en esencia aleatorios. Esto puede producir pulsos de reloj *parciales* en la salida *X* si cualquiera de las transiciones de *A* ocurre mientras la señal de reloj esté en ALTO, como se muestra en las formas de onda de la figura 5-39(b).

A menudo este tipo de salida no es aceptable, por lo que debemos desarrollar un método para evitar que aparezcan pulsos parciales en *X*. Una solución se muestra en la figura 5-40(a). Describa la manera en que este circuito resuelve el problema y dibuje la forma de onda de *X* para la misma situación que se presenta en la figura 5-39(b).



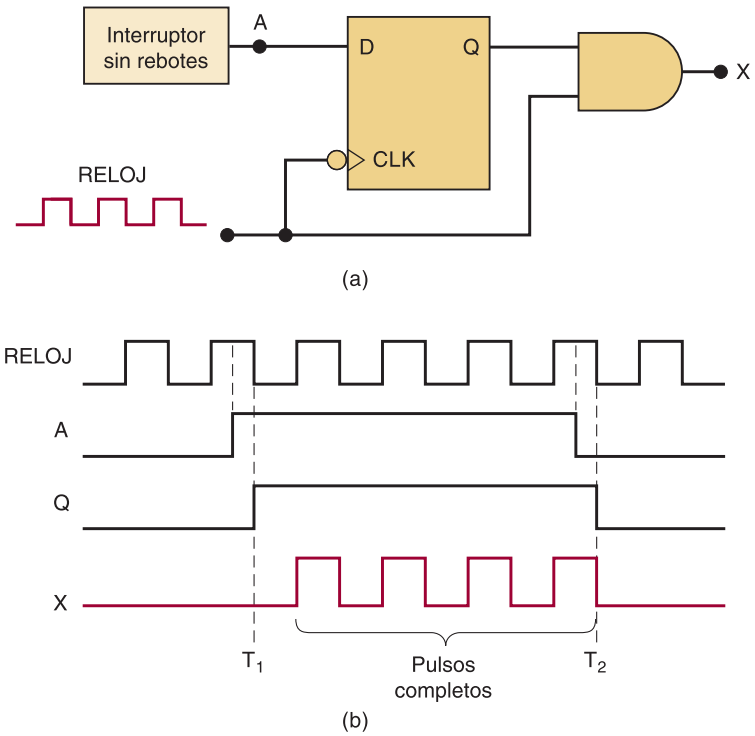


FIGURA 5-40 Un flip-flop D disparado por flanco se utiliza para sincronizar la habilitación de la compuerta AND con las NGTs del reloj.

Solución

La señal *A* se conecta a la entrada *D* del FF *Q*, el cual está sincronizado por la NGT de la señal del reloj. Por ende, cuando *A* cambia a ALTO *Q* cambiará a ALTO hasta la siguiente NGT del reloj en el tiempo T_1. Este nivel ALTO en *Q* habilitará a la compuerta AND para que pase los siguientes pulsos de reloj *completos* a *X*, como muestra la figura 5-40(b).

Cuando *A* regrese a BAJO, *Q* no cambiará a BAJO hasta la siguiente NGT del pulso de reloj en T_2. Por ende, la compuerta AND no inhibirá los pulsos de reloj sino hasta que haya pasado a *X* el pulso de reloj que termina en T_2. En consecuencia, la salida *X* sólo contiene pulsos completos.

Hay un problema potencial con este circuito. Como *A* podría cambiar a ALTO en cualquier momento, habría una probabilidad aleatoria de que violara los requerimientos de tiempo de estabilización del reloj. En otras palabras, la transición de *A* podría ocurrir tan cerca del flanco del reloj como para producir una respuesta inestable (deformación) de la salida *Q*. Para evitar esto se requeriría un circuito de sincronización más complejo.

5-16 DETECCIÓN DE UNA SECUENCIA DE ENTRADA

En muchas situaciones una salida debe activarse sólo cuando las entradas se activan en cierta secuencia. Esto no puede lograrse si se utiliza sólo la lógica combinacional, ya que requiere de la característica de almacenamiento de los FFs.

Por ejemplo, una compuerta AND puede utilizarse para determinar cuando dos entradas *A* y *B* se encuentran (ambas) en ALTO, pero su salida responderá igual sin importar cuál entrada cambie primero a este estado. Pero suponga que deseamos generar una salida en ALTO *sólo* si *A* cambia a ALTO y *B* lo hace también cierto tiempo después. La figura 5-41(a) muestra una forma de lograr esto.

Las formas de onda de las figuras 5-41(a) y (c) muestran que *Q* cambiará a ALTO sólo si *A* cambia a ALTO antes de que *B* lo haga. Esto se debe a que *A* primero debe estar en ALTO para que *Q* pueda cambiar a ALTO con la PGT de *B*.

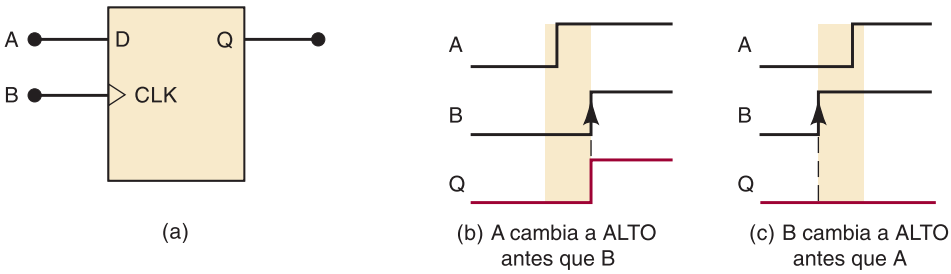


FIGURA 5-41 Flip-flop sincronizado por reloj en D, utilizado para responder a una secuencia específica de entradas.

Para que este circuito pueda funcionar en forma apropiada, *A* debe cambiar a ALTO antes que *B* durante cuando menos un lapso igual al requerimiento de tiempo de estabilización del FF.

5-17 ALMACENAMIENTO Y TRANSFERENCIA DE DATOS

Hasta ahora, el uso más común de los flip-flops es para el almacenamiento de datos o información. Los datos pueden representar valores numéricos (por ejemplo, números binarios, números decimales codificados en BCD) o cualquiera de una amplia variedad de tipos de datos que hayan sido codificados en binario. Por lo general, estos datos se almacenan en grupos de FFs, llamados **registros**.

La operación que se realiza con más frecuencia sobre los datos que se almacenan en un FF o un registro es la **transferencia de datos**; lo que implica la transferencia de datos de un FF o registro hacia otro. La figura 5-42 muestra cómo puede lograrse la transferencia de datos entre dos FFs mediante el uso de flip-flops sincronizados por reloj en S-R, J-K y D. En cada caso, el valor lógico que está almacenado en *A* del FF se transfiere a *B* del FF cuando ocurre la NGT del pulso TRANSFERIR. Por ende, después de esta NGT la salida *B* será la misma que la salida *A*.

Las operaciones de transferencia en la figura 5-42 son ejemplos de **transferencia síncrona**, ya que se utilizan las entradas de control y *CLK* síncronas para realizar la transferencia. También puede obtenerse una operación de transferencia si se utilizan las entradas asíncronas de un FF. La figura 5-43 muestra cómo puede lograrse una **transferencia asíncrona** mediante las entradas PRESET y CLEAR de cualquier tipo de FF.

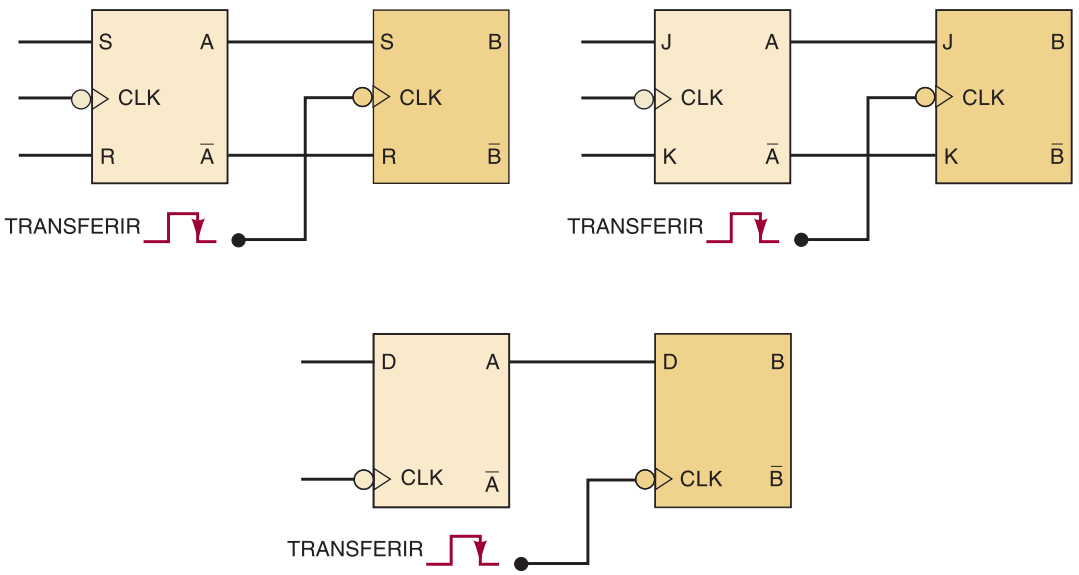
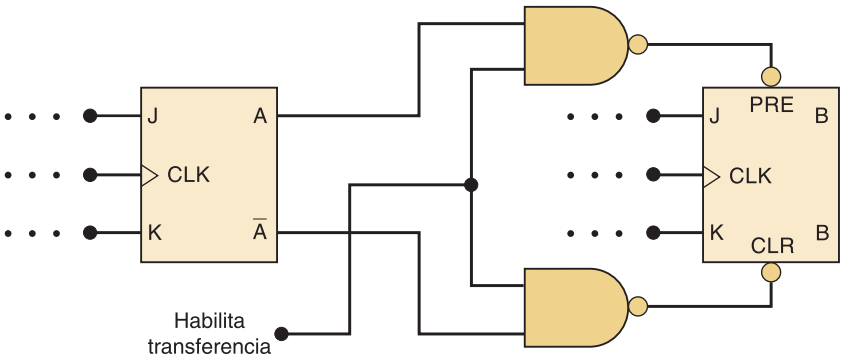


FIGURA 5-42 Operación de transferencia de datos síncrona, realizada por varios tipos de FFs sincronizados por reloj.

FIGURA 5-43
Operación de transferencia de datos asíncrona.

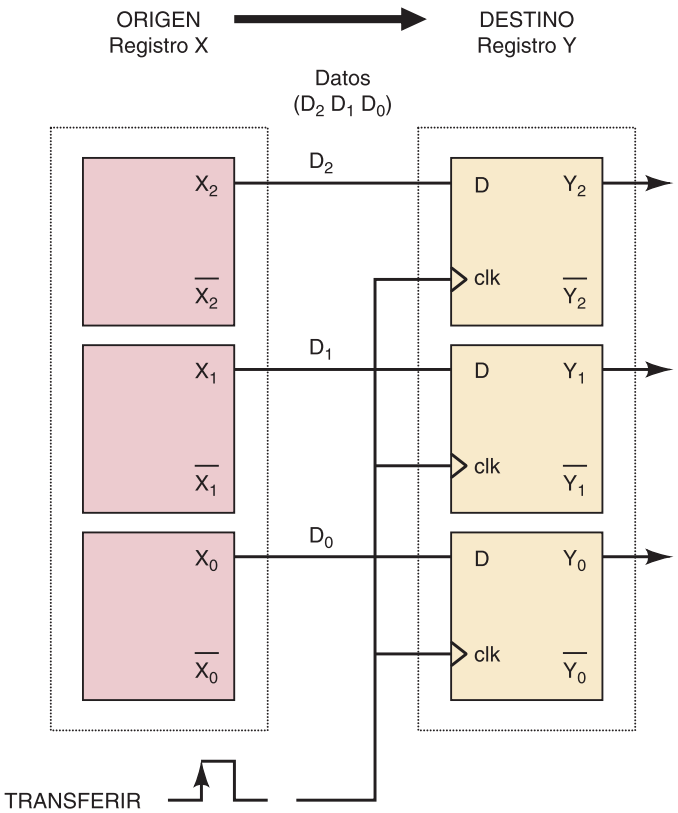


Aquí, las entradas asíncronas responden a los niveles BAJO. Cuando la línea HABILITA TRANSFERENCIA se mantiene en BAJO, las dos salidas de las compuertas NAND se mantienen en ALTO, sin efecto alguno sobre las salidas del FF. Cuando la línea HABILITA TRANSFERENCIA se cambia a ALTO, una de las salidas de las compuertas NAND cambiará a BAJO, dependiendo del estado de las salidas *A* y $\overline{A}$. Este nivel BAJO establecerá o borrará el FF *B* para que quede en el mismo estado que el FF *A*. Esta transferencia asíncrona se realiza de manera independiente de las entradas síncronas y *CLK* del FF. A la transferencia asíncrona también se le conoce como **transferencia atascada** debido a que los datos pueden "atascarse" en el FF *B* aún cuando sus entradas síncronas estén activas.

Transferencia de datos en paralelo

La figura 5-44 ilustra la transferencia de datos de un registro a otro mediante el uso de varios FF tipo D. El registro *X* consiste de los FFs X_2, X_1 y X_0; el registro *Y* consiste de los FFs Y_2, Y_1 y Y_0. Al momento de aplicar la PGT del pulso TRANSFERIR, el nivel almacenado en X_2 se transfiere a Y_2, X_1 a Y_1 y X_0 a Y_0. La transferencia del contenido del registro *X* hacia el registro *Y* es una transferencia síncrona. También

FIGURA 5-44
Transferencia en paralelo del contenido del registro X hacia el registro Y.



se le conoce como transferencia en paralelo, ya que el contenido de X_2, X_1 y X_0 se transfiere *de manera simultánea* hacia Y_2, Y_1 y Y_0, respectivamente. Si se fuera a realizar una **transferencia de datos en serie**, el contenido del registro *X* pasaría hacia el registro *Y* bit por bit. En la siguiente sección examinaremos este proceso.

Es importante comprender que la transferencia en paralelo no cambia el contenido del registro que actúa como el origen de los datos. Por ejemplo, en la figura 5-44 si $X_2X_1X_0 = 101$ y $Y_2Y_1Y_0 = 011$ antes de que ocurra el pulso en TRANSFERIR, entonces ambos registros almacenarán el valor de 101 después del pulso TRANSFERIR.

PREGUNTAS DE REPASO

1. *Verdadero o falso:* la transferencia asíncrona de datos utiliza la entrada *CLK*.
2. ¿Qué tipo de FF se adapta mejor para la transferencia síncrona debido a que requiere la menor cantidad de interconexiones de un FF al otro?
3. Si se utilizaran flip-flops J-K en los registros de la figura 5-44, en total ¿cuántas interconexiones se requerirían del registro *X* al registro *Y*?
4. *Verdadero o falso:* la transferencia síncrona de datos requiere menos circuitos que la transferencia asíncrona.

5-18 TRANSFERENCIA DE DATOS EN SERIE: REGISTROS DE DESPLAZAMIENTO

Antes de describir la operación de transferencia de datos en serie debemos examinar el arreglo básico de un **registro de desplazamiento;** es decir, un grupo de FFs ordenados de manera que los números binarios almacenados en ellos se desplacen de un FF al siguiente durante cada uno de los pulsos de reloj. Sin duda habrá usted visto en acción los registros de desplazamiento en dispositivos tales como una calculadora electrónica, en la cual los dígitos que se muestran en la pantalla se desplazan un lugar cada vez que usted introduce un nuevo dígito. Es la misma acción que se lleva a cabo en un registro de desplazamiento.

La figura 5-45(a) muestra una manera de ordenar los flip-flops J-K para que operen como un registro de desplazamiento de cuatro bits. Observe que los FFs están conectados de manera que la salida de X_3 se transfiera hacia X_2 hacia X_1, y X_1 hacia X_0. Lo que significa que al momento en que ocurre la NGT de un pulso de desplazamiento, cada FF toma el valor almacenado a su izquierda. El flip-flop X_3 toma un valor determinado por las condiciones presentes en sus entradas *J* y *K* cuando ocurre la NGT. Por ahora supondremos que las entradas *J* y *K* de X_3 se alimentan mediante la forma de onda de DATOS ENT que se muestra en la figura 5-45(b). También supondremos que todos los FFs se encuentran en el estado 0 antes de aplicar los pulsos de desplazamiento.

Las formas de onda en la figura 5-45(b) muestran cómo se desplazan los datos de entrada de izquierda a derecha, de FF en FF a medida que se aplican los pulsos de desplazamiento. Cuando se produce la primera NGT en T_1, cada uno de los FFs X_2, X_1 y X_0 tendrán la condición $J = 0$, $K = 1$ presente en sus entradas, debido al estado del FF a su izquierda. El flip-flop X_3 tendrá la condición $J = 1$, $K = 0$ debido a DATOS ENT. Por ende, en T_1 sólo X_3 cambiará a ALTO mientras todos los demás FFs permanecerán en BAJO. Cuando ocurra la segunda NGT en T_2, el flip-flop X_3 tendrá la condición $J = 0$, $K = 1$ debido a DATOS ENT. El flip-flop X_2 tendrá $J = 1$, $K = 0$ debido al nivel ALTO actual en X_3. Los flip-flops X_1 y X_0 tendrán todavía la condición $J = 0$, $K = 1$. Por ende, en T_2 sólo el FF X_2 cambiará a ALTO, el FF X_3 cambiará a BAJO y los FFs X_1 y X_0 permanecerán en BAJO.

Podemos usar un razonamiento similar para determinar cómo cambian las formas de onda en T_3 y en T_4. Observe que en cada NGT de los pulsos de desplazamiento la salida de cada FF toma el nivel que estaba presente en la salida del FF a su izquierda, justo *antes* de la NGT. Desde luego que X_3 toma el nivel que estaba presente en DATOS ENT justo antes de la NGT.

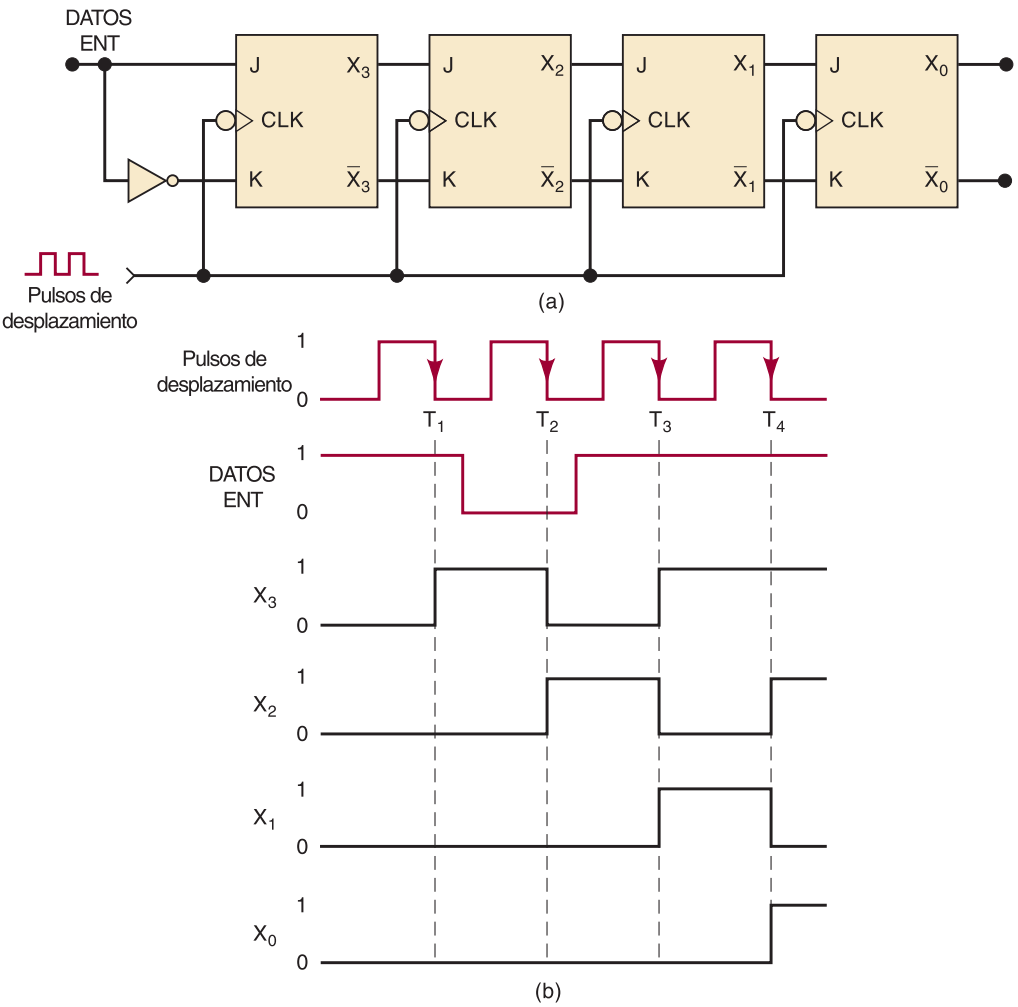


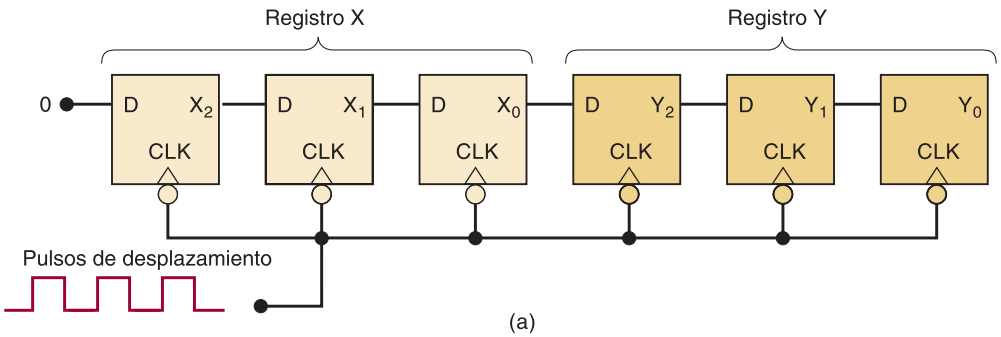
FIGURA 5-45 Registro de desplazamiento de cuatro bits.

Requerimiento de tiempo de retención

En este arreglo de registros de desplazamiento es necesario que los FFs tengan un requerimiento de tiempo de retención muy corto, debido a que hay veces en las que las entradas *J*, *K* están cambiando casi al mismo tiempo que la transición en *CLK*. Por ejemplo, la salida X_3 cambia de 1 a 0 en respuesta a la NGT en T_2, lo cual provoca que las entradas *J*, *K* de X_2 cambien mientras su entrada *CLK* está cambiando. En realidad, debido al retraso de propagación de X_3 las entradas *J*, *K* de X_2 no cambiarán sino hasta un corto tiempo después de la NGT. Por esta razón, un registro de desplazamiento debe implementarse utilizando FFs disparados por flanco que tengan un valor de t_H menor que un retraso de propagación de *CLK* a la salida. Este último requerimiento lo cumplen con facilidad la mayoría de los FFs disparados por flanco.

Transferencia en serie entre registros

La figura 5-46(a) muestra dos registros de desplazamiento de tres bits, conectados de forma tal que el contenido del registro *X* se transferirá en serie (se desplazará) hacia el registro *Y*. Estamos usando flip-flops D para cada registro de desplazamiento, ya que se requieren menos conexiones que con los flip-flops J-K. Observe cómo X_0, el último FF del registro *X*, está conectado a la entrada *D* de Y_2, el primer FF del registro *Y*. Por ende, a medida que se aplican los pulsos la transferencia de información se realiza de la siguiente manera: $X_2 \rightarrow X_1 \rightarrow X_0 \rightarrow Y_2 \rightarrow Y_1 \rightarrow Y_0$. El



(a)

X_2	X_1	X_0	Y_2	Y_1	Y_0	
1	0	1	0	0	0	← Antes de aplicar los pulsos
0	1	0	1	0	0	← Después del primer pulso
0	0	1	0	1	0	← Después del segundo pulso
0	0	0	1	0	1	← Después del tercer pulso

(b)

FIGURA 5-46 Transferencia en serie de información, del registro *X* al registro *Y*.

FF X_2 cambiará a un estado determinado con base en su entrada *D*. Por ahora, *D* se mantendrá en BAJO, de manera que X_2 cambiará a BAJO en el primer pulso y permanecerá ahí.

Para ilustrar lo anterior supongamos que antes de aplicar cualquier pulso de desplazamiento el contenido del registro *X* es 101 (es decir, $X_2 = 1$, $X_1 = 0$, $X_0 = 1$) y que el registro *Y* está en 000. Consulte la tabla en la figura 5-46(b), la cual muestra cómo cambian los estados de cada FF a medida que se aplican pulsos de desplazamiento. Hay que recalcar los siguientes puntos:

1. En la NGT de cada pulso, cada FF toma el valor que estaba almacenado en el FF a su izquierda, antes de que ocurriera el pulso.
2. Después de *tres* pulsos, el 1 que estaba al principio en X_2 está ahora en Y_2, el 0 que estaba al principio en X_1 se encuentra en Y_1 y el 1 que estaba al principio en X_0 está en Y_0. En otras palabras, el 101 que estaba almacenado en el registro *X* se ha desplazado ahora hacia el registro *Y*. El registro *X* está en 000; ha perdido sus datos originales.
3. La transferencia completa de los *tres* bits de datos requiere *tres* pulsos de desplazamiento.

EJEMPLO 5-13

Asuma el mismo contenido inicial de los registros *X* y *Y* en la figura 5-46. ¿Cuál será el contenido de cada FF después de que ocurra el sexto pulso de desplazamiento?

Solución

Si continuamos el proceso que se muestra en la figura 5-46(b) para tres pulsos de desplazamiento más, encontraremos que todos los FFs estarán en el estado 0 después del sexto pulso. Otra forma de llegar a este resultado es razonar de la siguiente manera: el nivel 0 constante en la entrada *D* de X_2 desplaza un nuevo 0 en cada pulso, de manera que después de seis pulsos los registros se llenan con 0s.

Operación de desplazamiento a la izquierda

Los FFs en la figura 5-46 pueden conectarse con igual facilidad de manera que la información se desplace de derecha a izquierda. No hay alguna ventaja particular de desplazar hacia una dirección o hacia la otra; como veremos más adelante la dirección que elija el diseñador lógico a menudo dependerá de la naturaleza de la aplicación.

Comparación entre transferencia en paralelo y en serie

En la transferencia en paralelo, toda la información se transfiere en forma simultánea cuando ocurre un pulso de comando de transferencia *individual* (figura 5-44), sin importar cuántos bits se estén transfiriendo. En la transferencia en serie, según el ejemplo de la figura 5-46, la transferencia completa de N bits de información requiere N pulsos de reloj (tres bits requieren tres pulsos, cuatro bits requieren cuatro pulsos, etc.). Es obvio que la transferencia en paralelo es mucho más rápida que la transferencia en serie cuando se utilizan registros de desplazamiento.

En la transferencia en paralelo, la salida de cada FF en el registro X está conectada a la correspondiente entrada FF en el registro Y. En la transferencia en serie, sólo el último FF en el registro X está conectado al registro Y. En general, la transferencia en paralelo requiere más interconexiones entre el registro emisor (X) y el receptor (Y) que la transferencia en serie. Esta diferencia se vuelve crítica cuando se transfiere un número mayor de bits de información. Ésta es una importante consideración cuando los registros emisor y receptor están alejados uno del otro, ya que determina cuántas líneas (alambres) se necesitan para la transición de la información.

La elección de transmisión en serie o en paralelo depende de la aplicación de un sistema en particular y de sus especificaciones. A menudo se utiliza una combinación de los dos tipos para aprovechar la *velocidad* de la transferencia en paralelo y la *economía y simpleza* de la transferencia en serie. Más adelante veremos otros detalles acerca de la transferencia de información.

PREGUNTAS DE REPASO

1. *Verdadero o falso:* el método más rápido para transferir datos de un registro a otro es la transferencia en paralelo.
2. ¿Cuál es la principal ventaja de la transferencia en serie, en comparación con la transferencia en paralelo?
3. Consulte la figura 5-46. Suponga que el contenido inicial de los registros es $X_2 = 0$, $X_1 = 1$, $X_0 = 0$, $Y_2 = 1$, $Y_1 = 1$, $Y_0 = 0$. Suponga además que la entrada D de X_2 se mantiene en ALTO. Determine el valor de la salida de cada FF después de que se produce el cuarto pulso de desplazamiento.
4. ¿En cuál de las formas de transferencia de datos el registro de origen no pierde su información?

5-19 DIVISIÓN Y CONTEO DE FRECUENCIA

Consulte la figura 5-47(a). Cada FF tiene sus entradas J y K en el nivel 1, por lo que cambiará de estado (conmutará) cada vez que la señal en su entrada *CLK* cambie de ALTO a BAJO. Los pulsos de reloj sólo se aplican a la entrada *CLK* del FF Q_0. La salida Q_0 está conectada a la entrada *CLK* del FF Q_1, y la salida Q_1 está conectada a la entrada *CLK* del FF Q_2. Las formas de onda de la figura 5-47(b) muestran cómo los FFs cambian de estado a medida que se aplican los pulsos. Hay que recalcar los siguientes puntos importantes:

1. El flip-flop Q_0 conmuta en la transición de pendiente negativa de cada pulso de reloj de entrada. Por ende, la forma de onda de la salida Q_0 tiene una frecuencia exactamente igual a la mitad de la frecuencia del pulso de reloj.



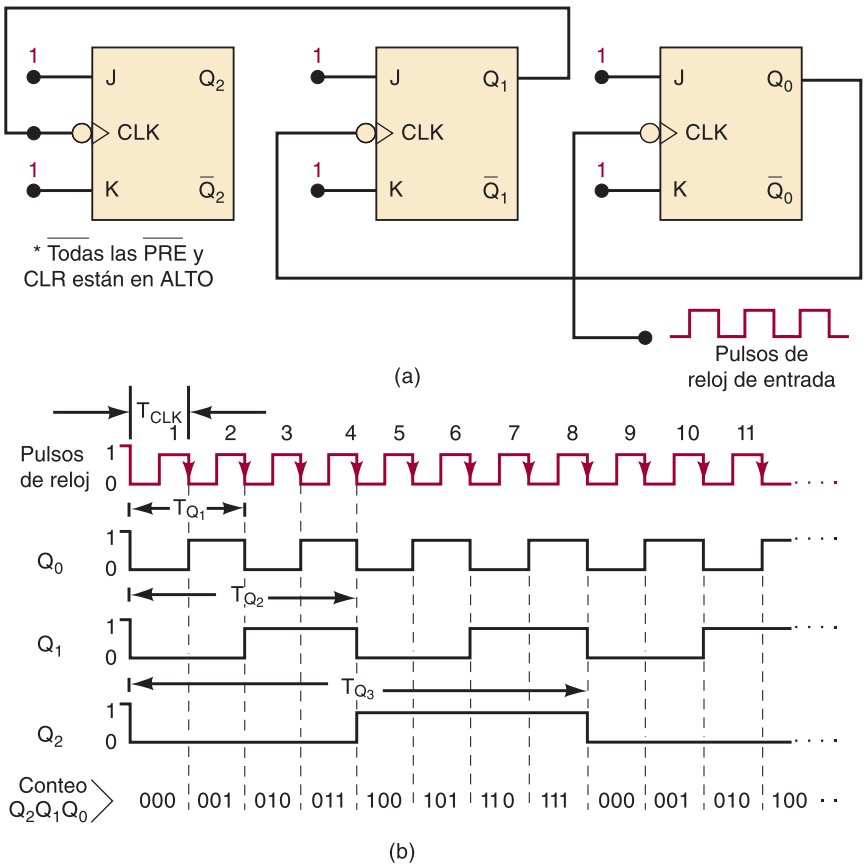


FIGURA 5-47 Flip-flops J-K conectados como un contador binario de tres bits (MOD-8).

2. El flip-flop Q_1 conmuta cada vez que la salida Q_0 cambia de ALTO a BAJO. La forma de onda de Q_1 tiene una frecuencia exactamente igual a la mitad de la frecuencia de la salida Q_0 y que, por lo tanto, es igual a un cuarto del valor de la frecuencia del reloj.
3. El flip-flop Q_2 conmuta cada vez que la salida Q_1 cambia de ALTO a BAJO. Por ende, la forma de onda de Q_2 tiene la mitad de la frecuencia de Q_1 y en consecuencia, un octavo de la frecuencia del reloj.
4. La salida de cada FF es una onda cuadrada (ciclo de trabajo del 50%).

Como se describió antes, cada FF divide la frecuencia de su entrada por 2. De esta forma, si agregáramos un cuarto FF a la cadena, éste tendría una frecuencia igual a 1/16 de la frecuencia del reloj, y así en lo sucesivo. Si utilizamos el número de FFs apropiados, este circuito podría dividir una frecuencia por cualquier potencia de 2. De manera específica, si se utilizan N flip-flops se producirá una frecuencia de salida del último FF, la cual equivale a $1/2^N$ de la frecuencia de entrada.

A esta aplicación de los flip-flops se le conoce como **división de frecuencia**. Muchas aplicaciones requieren una división de frecuencia. Por ejemplo, su reloj de pulso es sin duda un reloj de "cuarzo". El término *reloj de cuarzo* significa que se utiliza un cristal de cuarzo para generar una frecuencia de oscilación muy estable. La frecuencia natural resonante del cristal de cuarzo en su reloj tal vez sea de 1 MHz o más. Para poder avanzar el indicador de los "segundos" una vez cada segundo, la frecuencia de oscilación se *divide* entre un valor que producirá una frecuencia de salida de 1 Hz bastante estable y precisa.

Operación de conteo

Además de funcionar como divisor de frecuencia, el circuito de la figura 5-47 también opera como **contador binario**. Para demostrarlo podemos examinar la secuencia

2^2	2^1	2^0		
Q_2	Q_1	Q_0		
0	0	0	Antes de aplicar los pulsos del reloj	
0	0	1	Después del pulso #1	
0	1	0	#2	
0	1	1	#3	
1	0	0	#4	
1	0	1	#5	
1	1	0	#6	
1	1	1	#7	
0	0	0	#8	000
0	0	1	#9	
0	1	0	#10	
0	1	1	#1	
.	.	.	. . .	
.	.	.	. . .	
.	.	.	. . .	

FIGURA 5-48 La tabla de estados del flip-flop muestra la secuencia de conteo binario.

de estados de los FFs después de que ocurre cada pulso de reloj. La figura 5-48 presenta los resultados en una **tabla de estados**. Dejemos que los valores $Q_2Q_1Q_0$ representen un número binario, en donde Q_2 está en la posición 2^2, Q_1 está en la posición 2^1 y Q_0 está en la posición 2^0. Los primeros ocho estados de $Q_2Q_1Q_0$ en la tabla deben reconocerse como la secuencia de conteo binario de 000 a 111. Después de la primera NGT, los FFs se encuentran en el estado 001 ($Q_2 = 0$, $Q_1 = 0$, $Q_0 = 1$), la cual representa a 001_2 (que equivale al 1 decimal); después de la segunda NGT los FFs representan el valor 010_2, que equivale a 2_{10}; después de tres pulsos, $011_2 = 3_{10}$; después de cuatro pulsos $100_2 = 4_{10}$ y así en lo sucesivo, hasta que después de siete pulsos $111_2 = 7_{10}$. En la octava NGT, los FFs regresan al estado 000, y la secuencia binaria se repite a sí misma para los siguientes pulsos.

Por ende, para los primeros siete pulsos de entrada el circuito funciona como un contador binario, en el cual los estados de los FFs representan a un número binario equivalente al número de pulsos que se han producido. Este contador tiene una capacidad de hasta $111_2 = 7_{10}$ antes de regresar a 000.

Diagrama de transición de estados

Otra manera de mostrar cómo cambian los estados de los FFs con cada pulso de reloj que se aplica es mediante el uso de un **diagrama de transición de estados**, como se muestra en la figura 5-49.

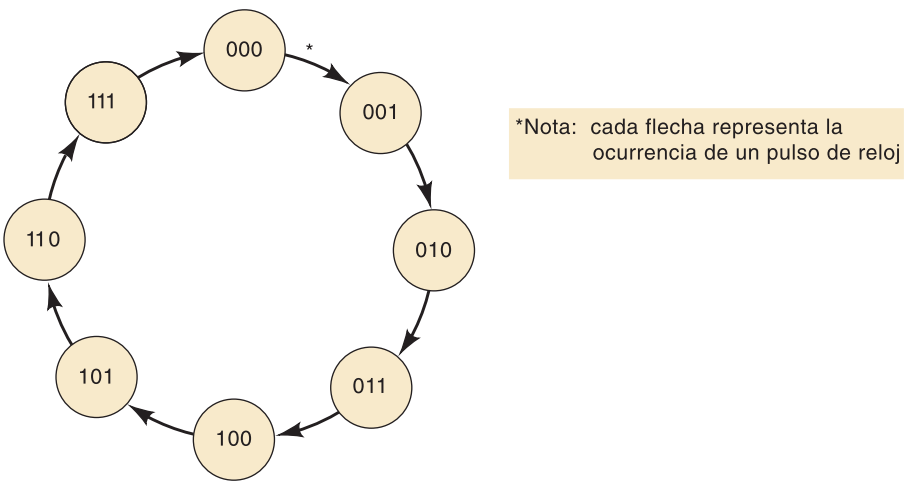


FIGURA 5-49 El diagrama de transición de estados muestra cómo cambian los estados de los flip-flops contadores con cada pulso de reloj que se aplica.

Cada círculo representa uno de los posibles estados, según lo indica el número binario dentro del círculo. Por ejemplo, el círculo que contiene el número 100 representa el estado 100 (es decir, $Q_2 = 1$, $Q_1 = Q_0 = 0$).

Las flechas que conectan un círculo con otro muestra la forma en que cambia un estado a otro, a medida que se aplica un pulso de reloj. Si analizamos un círculo de estado específico, podremos ver cuál estado va antes y cuál va después. Por ejemplo, si analizamos el estado 000 podemos ver que se llega a él cada vez que el contador se encuentra en el estado 111 y se aplica un pulso de reloj. De igual forma, podemos ver que el estado 000 siempre va antes del estado 001.

Utilizaremos los diagramas de transición de estados para ayudar a describir, analizar y diseñar contadores y otros circuitos secuenciales.

Número MOD

El contador de la figura 5-47 tiene $2^3 = 8$ estados distintos (del 000 al 111). A este contador se le denomina *contador MOD-8*, en donde el **número MOD** indica la cantidad de estados en la secuencia de conteo. Si se agregara un cuarto FF, la secuencia de estados contaría en binario desde 0000 hasta 111, un total de 16 estados. A esto se le llamaría *contador MOD-16*. En general, si se conectan N flip-flops en el arreglo de la figura 5-47, el contador tendrá 2^N estados distintos, por lo cual será un contador MOD-2^N. Sería capaz de contar hasta $2^N - 1$ antes de regresar a 0.

El número MOD de un contador también indica la división de frecuencia que se obtiene del último FF. Por ejemplo, un contador de cuatro bits tiene cuatro FFs, cada uno de los cuales representa un dígito binario (bit) y, por lo tanto, es un contador MOD-2^4 = MOD-16. Debido a esto, puede contar hasta 15 ($= 2^4 - 1$). También puede usarse para dividir la frecuencia del pulso de entrada por un factor de 16 (el número MOD).

Hasta ahora sólo hemos visto el contador binario de FF básico. En el capítulo 7 examinaremos los contadores con mucho más detalle.

EJEMPLO 5-14

Suponga que el contador MOD-8 de la figura 5-47 se encuentra en el estado 101. ¿Cuál será el estado (cuenta) después de aplicar 13 pulsos?

Solución

Localice el estado 101 en el diagrama de transición de estados. Proceda alrededor del diagrama de estados a través de ocho cambios de estado, y deberá encontrarse de vuelta en el estado 101. Ahora continúe a través de cinco cambios de estado más (para un total de 13) y deberá terminar en el estado 010.

Como éste es un contador MOD-8 con ocho estados, requiere de ocho transiciones de estado para realizar una excursión completa alrededor del diagrama y de vuelta al estado inicial.

EJEMPLO 5-15

Considere un circuito contador que contiene seis FFs conectados como se muestra en la figura 5-47 (es decir, Q_5, Q_4, Q_3, Q_2, Q_1, Q_0).

(a) Determine el número MOD del contador.

(b) Determine la frecuencia en la salida del último FF (Q_5) cuando la frecuencia del reloj de entrada es de 1 MHz.

(c) ¿Cuál es el intervalo de estados de conteo para este contador?

(d) Suponga un estado inicial (conteo) de 000000. ¿Cuál será el estado del contador después de 129 pulsos?

Solución

(a) Número MOD = $2^6 = 64$.

(b) La frecuencia en el último FF será igual a la frecuencia del reloj de entrada, dividida entre el número MOD. Esto es,

$$f(\text{en } Q_5) = \frac{1\text{ MHz}}{64} = 15.625\text{ kHz}$$

(c) El contador contará desde 000000_2 hasta 111111_2 (de 0 a 63_{10}) para un total de 64 estados. Observe que el número de estados es el mismo que el número MOD.

(d) Como éste es un contador MOD-64, cada 64 pulsos de reloj regresarán el contador de vuelta a su estado inicial. Por lo tanto, después de 128 pulsos el conteo se regresa a 000000. El pulso 129 lleva el contador a la cuenta 000001.

PREGUNTAS DE REPASO

1. Una señal de reloj de 20 kHz se aplica a un flip-flop J-K, en donde $J = K = 1$. ¿Cuál es la frecuencia de la forma de onda de salida del FF?
2. ¿Cuántos FFs se requieren para un contador que cuente de 0 a 255_{10}?
3. ¿Cuál es el número MOD del contador en la pregunta 2?
4. ¿Cuál es la frecuencia de la salida del octavo FF cuando la frecuencia de reloj es de 512 kHz?
5. Si este contador empieza en 00000000, ¿cuál será su estado después de 520 pulsos?

5-20 APLICACIÓN DE MICROCOMPUTADORA

Su estudio sobre los sistemas digitales aún se encuentra en una etapa relativamente temprana y, por lo tanto, no ha aprendido mucho todavía sobre los microprocesadores y las microcomputadoras. No obstante, puede darse una idea básica acerca de cómo se emplean los FFs en una aplicación ordinaria controlada por microprocesador, sin preocuparse por todos los detalles que veremos después.

La figura 5-50 muestra una unidad de microprocesador (MPU) en la cual su salida se utiliza para transferir datos binarios al registro X, el cual consiste de cuatro flip-flops D identificados como X_3, X_2, X_1, X_0. Un conjunto de salidas de la MPU es el *código de dirección*, que está compuesto por las ocho salidas A_{15}, A_{14}, A_{13}, A_{12}, A_{11}, A_{10}, A_9, A_8. La mayoría de las MPUs tienen cuando menos 16 salidas de dirección

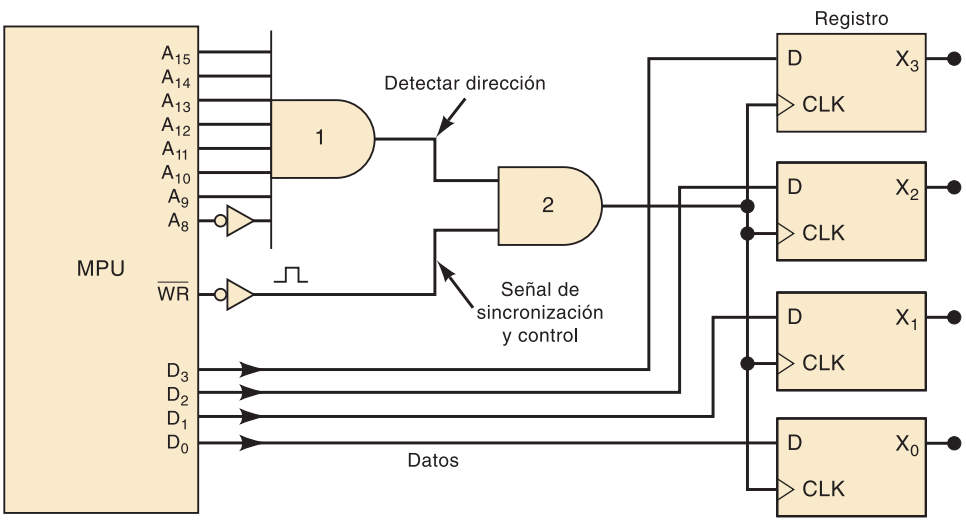


FIGURA 5-50 Ejemplo de un microprocesador que transfiere datos binarios a un registro externo.

disponibles, pero no siempre se utilizan todas. Un segundo conjunto de salidas de la MPU consiste de las cuatro *líneas de datos* D_3, D_2, D_1, D_0. La mayoría de las MPUs tienen cuando menos ocho líneas de datos disponibles. La otra salida de la MPU es una señal de control de sincronización $\overline{WR}$, la cual cambia a BAJO cuando la MPU está lista para escribir.

Recuerde que la MPU es la unidad central de procesamiento de una microcomputadora, y su función principal es ejecutar un programa de instrucciones almacenadas en la memoria de la computadora. Una de las instrucciones que podría ejecutar sería la que indique a la MPU que transfiera un número binario desde un registro de almacenamiento dentro de la MPU, hacia el registro externo *X*. A esto se le conoce como *ciclo de escritura*. Al ejecutar esta instrucción, la MPU realizaría los siguientes pasos:

1. Colocar el número binario en sus líneas de salida de datos D_3 a D_0.
2. Colocar el código de dirección apropiado en sus líneas de salida A_{15} a A_8 para seleccionar el registro *X* como el receptor de los datos.
3. Una vez que se hayan estabilizado las salidas de datos y de dirección, la MPU genera el pulso de escritura *WR* para sincronizar el registro y completar la transferencia de datos en paralelo hacia *X*.

Existen muchas situaciones en las que una MPU, bajo el control de un programa, enviará datos a un registro externo para poder controlar eventos externos. Por ejemplo, los FFs individuales en el registro pueden controlar el estado ENCENDIDO/APAGADO de dispositivos electromecánicos tales como solenoides, relevadores, motores, etcétera (a través de circuitos de interfaz apropiados, desde luego). Los datos que se envíen de la MPU al registro determinarán cuáles dispositivos deben estar ENCENDIDOS y cuáles APAGADOS. Otro ejemplo común es cuando el registro se utiliza para almacenar un número binario para enviarlo como entrada a un convertidor digital-analógico (DAC). La MPU envía el número binario al registro y el DAC lo convierte en un voltaje analógico que puede usarse para controlar algo tal como la posición de un haz de electrones en una pantalla CRT, o la velocidad de un motor.

EJEMPLO 5-16

(a) ¿Qué código de dirección debe generar la MPU para que los datos puedan transferirse hacia *X*?

(b) Suponga que $X_3 - X_0 = 0110$, $A_{15} - A_8 = 1111111$ y $D_3 - D_0 = 1011$. ¿Qué habrá en *X* después de que ocurra un pulso en $\overline{WR}$?

Solución

(a) Para poder transferir los datos hacia *X*, el pulso de reloj debe pasar a través de la compuerta AND 2 y hacia las entradas *CLK* de los FFs. Esto ocurrirá sólo si la entrada superior de la compuerta AND 2 se encuentra en ALTO. Esto significa que todas las entradas para la compuerta AND 1 deben estar en ALTO; es decir, desde A_{15} hasta A_9 deben ser 1 y A_8 debe ser 0. Por ende, se necesita la presencia del código de dirección 11111110 para permitir la transferencia de los datos hacia *X*.

(b) Con $A_8 = 1$, el nivel BAJO de la compuerta AND 1 inhibirá a *WR* de manera que no pueda pasar hacia la compuerta AND 2, y los FFs no se sincronizarán. Por lo tanto, el contenido del registro *X* no cambiará de 0110.

PREGUNTAS DE REPASO

1. Muestre cómo puede usarse el CI 74HC175 de la figura 5-34 para el registro *X* de la figura 5-50.

5-21 DISPOSITIVOS DISPARADORES DE SCHMITT

Un **circuito disparador de Schmitt** no se clasifica como un flip-flop, pero exhibe un tipo de memoria característico, que lo hace útil en ciertas situaciones especiales. Una de estas situaciones se muestra en la figura 5-51(a). Aquí un INVERSOR estándar está siendo controlado por una entrada lógica que tiene tiempos de transición relativamente lentos. Cuando estos tiempos de transición se exceden de los valores máximos permitidos (esto depende de la familia lógica específica), las salidas de la compuertas lógicas y de los INVERSORES pueden producir oscilaciones a medida que pasa la señal de entrada a través del intervalo indeterminado. Las mismas condiciones de entrada también pueden producir un disparo errático de los FFs.

Un dispositivo que tiene un tipo de entrada de disparador de Schmitt está diseñado para aceptar señales ruidosas con cambios lentos, y producir una salida con transiciones libres de oscilaciones. Por lo general, la salida tendrá tiempos de transición muy rápidos (por lo general de 10 ns) que son independientes de las características de la señal de entrada. La figura 5-51(b) muestra un INVERSOR disparador de Schmitt y su respuesta a una entrada con cambios lentos.

Si analiza las formas de onda en la figura 5-51(b), podrá observar que la salida no cambia de ALTO a BAJO sino hasta que la entrada se excede del voltaje *umbral de pendiente positiva*, V_{T+}. Una vez que la salida cambia a BAJO, permanecerá ahí incluso aunque la entrada vuelva a caer por debajo de V_{T+} (ésa es su característica de memoria), hasta que caiga por debajo del voltaje *umbral de pendiente negativa*, V_{T-}. Los valores de los dos voltajes de umbral variarán de una familia lógica a otra, pero V_{T-} siempre será menor que V_{T+}.

El INVERSOR disparador de Schmitt y todos los demás dispositivos con entradas de disparador de Schmitt utilizan el símbolo distintivo que se muestra en la figura 5-51(b) para indicar que pueden responder en forma confiable a las señales de entrada con cambios lentos. Los diseñadores lógicos utilizan CIs con entradas tipo disparador de Schmitt para convertir las señales con cambios lentos en señales limpias con cambios rápidos que puedan controlar entradas de CIs estándar.

Hay varios CIs disponibles con entradas tipo disparador de Schmitt. Los CIs 7414, 74LS14 y 74HC14 son CIs INVERSORES hexadecimales con entradas tipo disparador de Schmitt. Los CIs 7413, 74LS13 y 74HC13 son dos compuertas NAND de cuatro entradas tipo disparador de Schmitt.

PREGUNTAS DE REPASO

1. ¿Qué ocurre cuando se aplica una señal con cambios lentos a un CI lógico estándar?
2. ¿Cómo opera un dispositivo lógico disparador de Schmitt en forma distinta de un dispositivo lógico estándar?

5-22 ONE-SHOT (MULTIVIBRADOR MONOESTABLE)

Un circuito digital que está en parte relacionado con el FF es el **one-shot (OS)**. Al igual que el FF, el OS tiene dos entradas Q y $\overline{Q}$ que son el inverso una de otra. A diferencia del FF, el OS sólo tiene un estado de salida *estable* (por lo general $Q = 0$, $\overline{Q} = 1$), en donde permanece hasta que se dispara mediante una señal de entrada. Una vez disparado, las salidas del OS cambian al estado opuesto ($Q = 1$, $\overline{Q} = 0$). Permanece en este **estado casi estable** durante un periodo fijo (t_P), que, por lo general, se determina mediante una constante de tiempo RC que resulta de los valores de los componentes externos conectados al OS. Después de un tiempo t_P, las salidas del OS regresan a su estado inactivo hasta que se vuelven a disparar.

La figura 5-52(a) muestra el símbolo lógico para un OS. A menudo el valor de t_P se indica en alguna parte del símbolo del OS. En la práctica, t_P puede variar de varios nanosegundos hasta varias décadas de segundos. El valor exacto de t_P es variable y se determina mediante los valores de los componentes externos R_T y C_T.

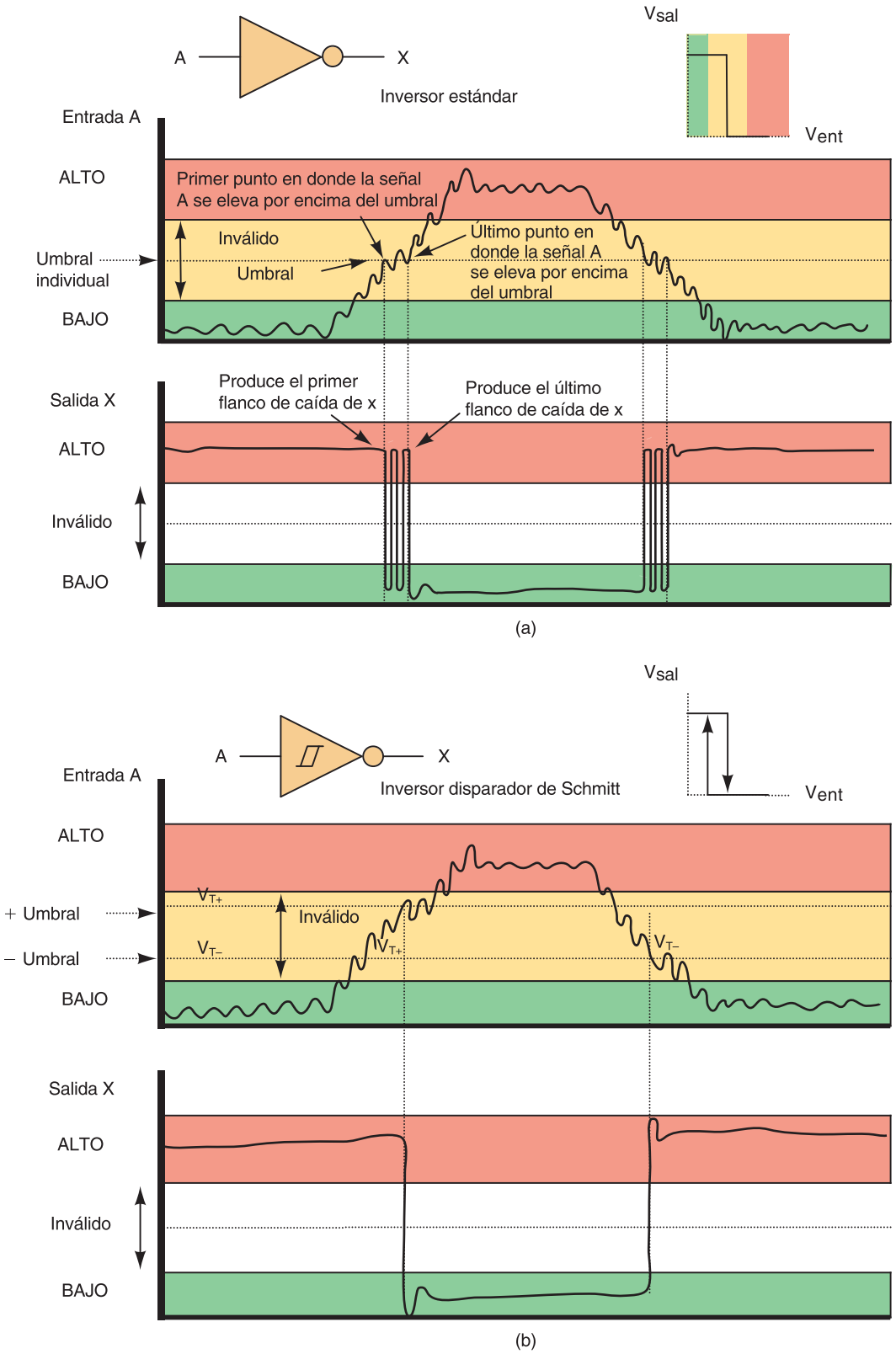


FIGURA 5-51 (a) Respuesta del inversor estándar a una entrada ruidosa y lenta, y (b) respuesta del disparador de Schmitt a una entrada ruidosa y lenta.

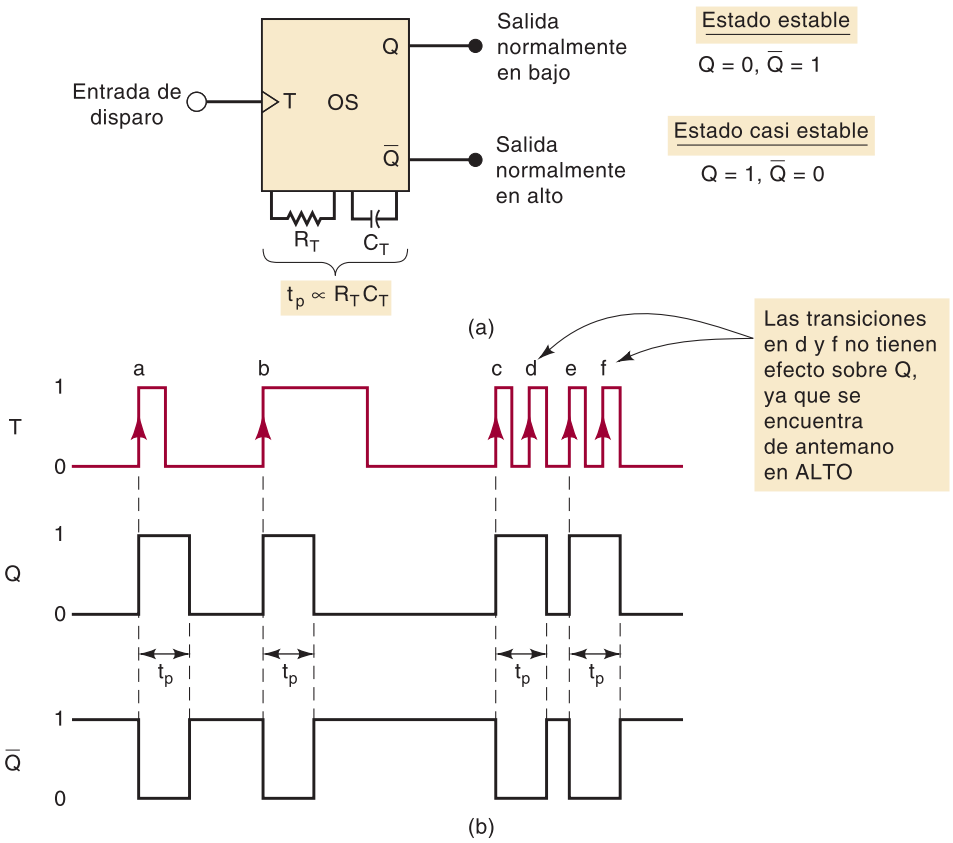


FIGURA 5-52 Símbolo del OS y formas de onda típicas para la operación no redisparable.

Hay dos tipos de one-shot disponibles en forma de CI: el **OS no redisparable** y el **OS redisparable**.

One-shot no redisparable

Las formas de onda de la figura 5-12(b) ilustran la operación de un OS no redisparable que se dispara en las transiciones de pendiente positiva en su entrada de disparo (*T*). Los puntos importantes a observar son:

1. Las PGTs en los puntos *a*, *b*, *c* y *e* dispararán el OS a su estado casi estable durante un tiempo t_p, después del cual regresará en forma automática a su estado estable.
2. Las PGTs en los puntos *d* y *f* no tienen efecto sobre el OS, ya que de antemano se ha disparado al estado casi estable. El OS debe regresar al estado estable antes de que pueda dispararse.
3. La duración del pulso de salida del OS es siempre la misma, sin importar la duración de los pulsos de entrada. Como se dijo antes, t_p, sólo depende de R_T y de C_T, además de los circuitos internos del OS. Un OS ordinario puede tener un t_p, dado por $t_p = 0.693\ R_T C_T$.

One-shot redisparable

El OS redisparable opera en forma muy parecida al OS no redisparable, con una diferencia importante: *puede redispararse mientras se encuentra en el estado casi estable, y comenzará un nuevo intervalo* t_p. La figura 5-53(a) compara la respuesta de ambos tipos de OS mediante el uso de un t_p de 2 ms. Examinemos estas formas de onda.

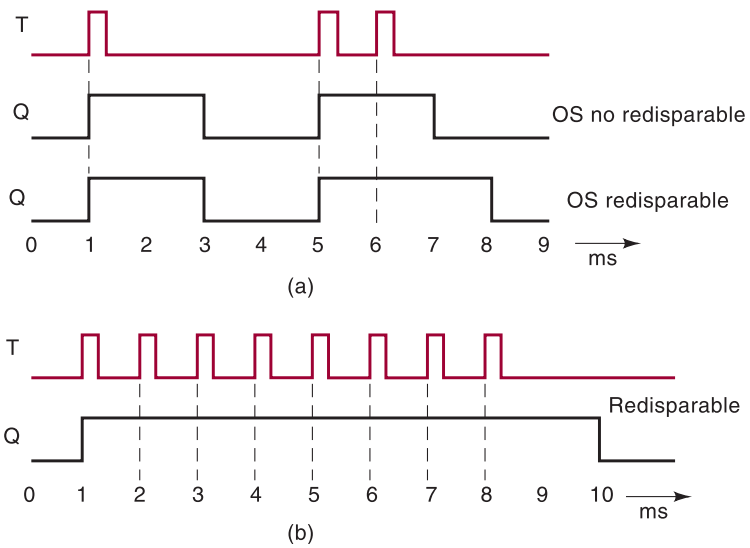


FIGURA 5-53
(a) Comparación de las respuestas del OS no redisparable y del OS redisparable para t_p = 2 ms. (b) El OS redisparable comienza un nuevo intervalo t_p cada vez que recibe un pulso de disparo.

Ambos tipos de OS responden al primer pulso de disparo en t = 1 ms, cambiando a nivel ALTO durante 2 ms y después regresando a BAJO. El segundo pulso de disparo en t = 5 ms dispara ambos monoestables al estado ALTO. El tercer pulso de disparo en t = 6 ms no tiene efecto sobre el OS no redisparable, ya que se encuentra de antemano en su estado casi estable. No obstante, este pulso de disparo *redisparará* el OS redisparable para comenzar un nuevo intervalo t_p = 2 ms. Por ende, permanecerá en ALTO durante 2 ms *después* de este tercer pulso de disparo.

En efecto, entonces un OS redisparable comienza un nuevo intervalo t_p cada vez que se aplica un pulso de disparo, sin importar el estado actual de su salida Q. De hecho, pueden aplicarse pulsos de disparo a una velocidad lo bastante rápida como para que el OS siempre se redispare antes de que termine el intervalo t_p y Q permanecerá en ALTO. Esto se muestra en la figura 5-53(b), en donde se aplican ocho pulsos cada 1 ms. Q no regresa a BAJO sino hasta 2 ms después del último pulso de disparo.

Dispositivos reales

Hay varios CIs monoestables disponibles tanto en la versión redisparable como en la no redisparable. El 74121 es un CI con one-shot individual no redisparable; el 74221, 74LS221 y 74HC221 son CIs con dos one-shot no redisparables; el 74122 y 74LS122 son CIs con one-shot individual redisparable; el 74123, 74LS123 y 74HC123 son CIs con dos one-shot redisparables.

La figura 5-54(a) muestra el símbolo tradicional para el CI 74121 con one-shot no redisparable. Observe que contiene compuertas lógicas internas para permitir que las entradas A_1, A_2 y B disparen el OS en una variedad de formas. La entrada B es tipo disparador de Schmitt y se le permite tener tiempos de transición lentos sin que pierda la capacidad de disparar en forma confiable el OS. Las terminales etiquetadas como R_{INT}, R_{EXT}/C_{EXT} y C_{EXT} se utilizan para conectar una resistencia y un capacitor externos para lograr la duración deseada en el pulso de salida. La figura 5-54(b) utiliza el símbolo IEEE/ANSI para el OS no redisparable 74121. Observe cómo este símbolo representa las compuertas lógicas. Observe además la presencia de un pequeño pulso con 1 enfrente del mismo. Esto indica que el dispositivo es un OS no redisparable. El símbolo IEEE/ANSI para un OS redisparable no tendría el 1 enfrente del pulso.

Multivibrador monoestable

Otro nombre para el one-shot es *multivibrador monoestable*, ya que sólo tiene un estado estable. Los monoestables encuentran una aplicación limitada en la mayoría

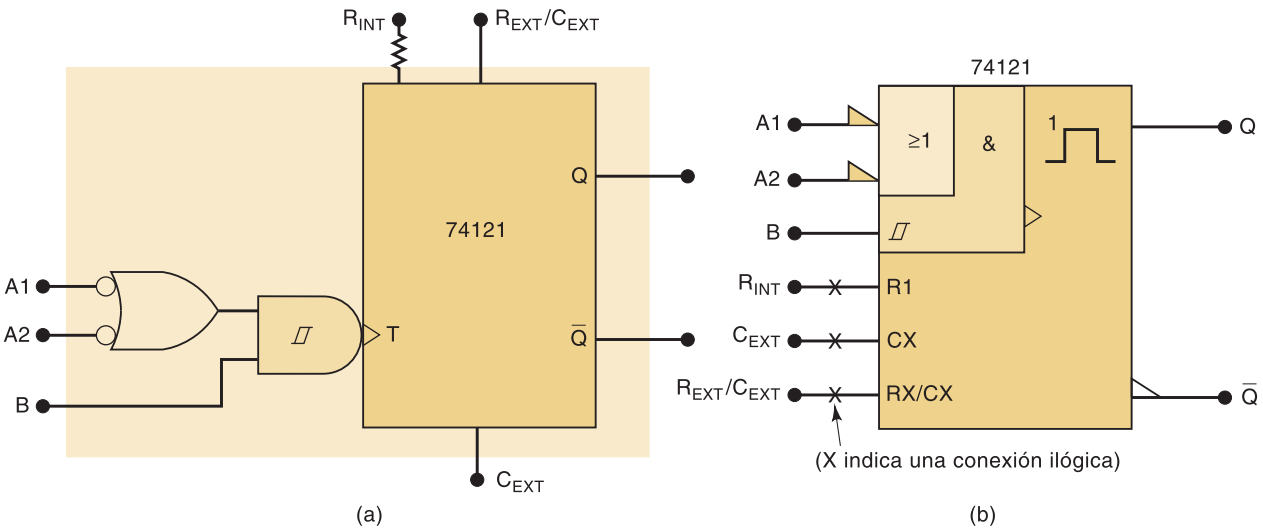


FIGURA 5-54 Símbolos lógicos para el monoestable no redisparable 74121: (a) tradicional; (b) IEEE/ANSI.

de los sistemas controlados por reloj; los diseñadores experimentados evitan usarlos ya que están propensos a falsos disparos debido al ruido espurio. Cuando se llegan a utilizar, por lo general, es en aplicaciones de sincronización simples que utilizan el intervalo t_p predeterminado. Varios de los problemas al final del capítulo ilustrarán la manera en que se utiliza un OS.

PREGUNTAS DE REPASO

1. En la ausencia de un pulso de disparo, ¿cuál será el estado de la salida de un OS?
2. *Verdadero o falso:* cuando se aplica un pulso a un OS no redisparable mientras se encuentra en su estado casi estable, la salida no se ve afectada.
3. ¿Qué es lo que determina el valor de t_p para un OS?
4. Describa cómo opera un OS redisparable en forma distinta de un OS no redisparable.

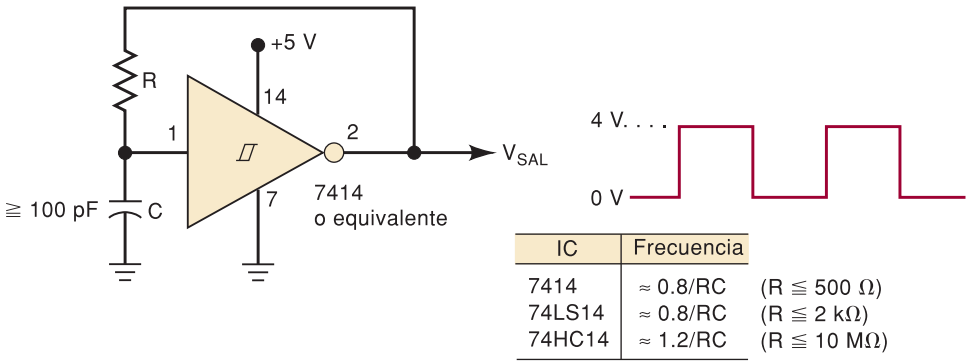
5-23 CIRCUITOS GENERADORES DE RELOJ

Los flip-flops tienen dos estados estables; por lo tanto, podemos decir que son *multivibradores biestables*. Los monoestables (one-shot) tienen sólo un estado estable, por lo que podemos llamarlos *multivibradores monoestables*. Un tercer tipo de multivibrador no tiene estados estables; se le llama **multivibrador astable** o **de funcionamiento libre**. Este tipo de circuito lógico alterna (oscila) entre dos estados de salida inestables. Es útil para generar señales de reloj para circuitos digitales síncronos.

Hay varios tipos de multivibradores astables de uso común. Presentaremos tres de ellos sin tratar de analizar su operación. Los presentamos aquí para que usted pueda construir un circuito generador de reloj si lo necesita para un proyecto o para probar circuitos digitales en el laboratorio.

Oscilador disparador de Schmitt

La figura 5-55 muestra cómo puede conectarse un INVERSOR disparador de Schmitt como un oscilador. La señal en V_{SAL} es una onda cuadrada aproximada con una frecuencia que depende de los valores de R y de C. La relación entre la frecuencia y



IC	Frecuencia	
7414	≈ 0.8/RC	(R ≦ 500 Ω)
74LS14	≈ 0.8/RC	(R ≦ 2 kΩ)
74HC14	≈ 1.2/RC	(R ≦ 10 MΩ)

FIGURA 5-55
Oscilador disparador de Schmitt que utiliza un INVERSOR 7414. También puede usarse una compuerta NAND tipo disparador de Schmitt 7413.

los valores *RC* se muestra en la figura 5-55 para tres INVERSORES disparadores de Schmitt distintos. Observe los límites máximos en el valor de resistencia para cada dispositivo. El circuito no oscilará si *R* no se mantiene por debajo de estos límites.

Temporizador 555 utilizado como multivibrador astable

El **CI temporizador 555** es un dispositivo compatible con TTL que puede operar en varios modos. La figura 5-56 muestra cómo pueden conectarse componentes externos a un 555 de manera que opere como un oscilador de funcionamiento libre. Su salida es una forma de onda rectangular repetitiva que alterna entre dos niveles

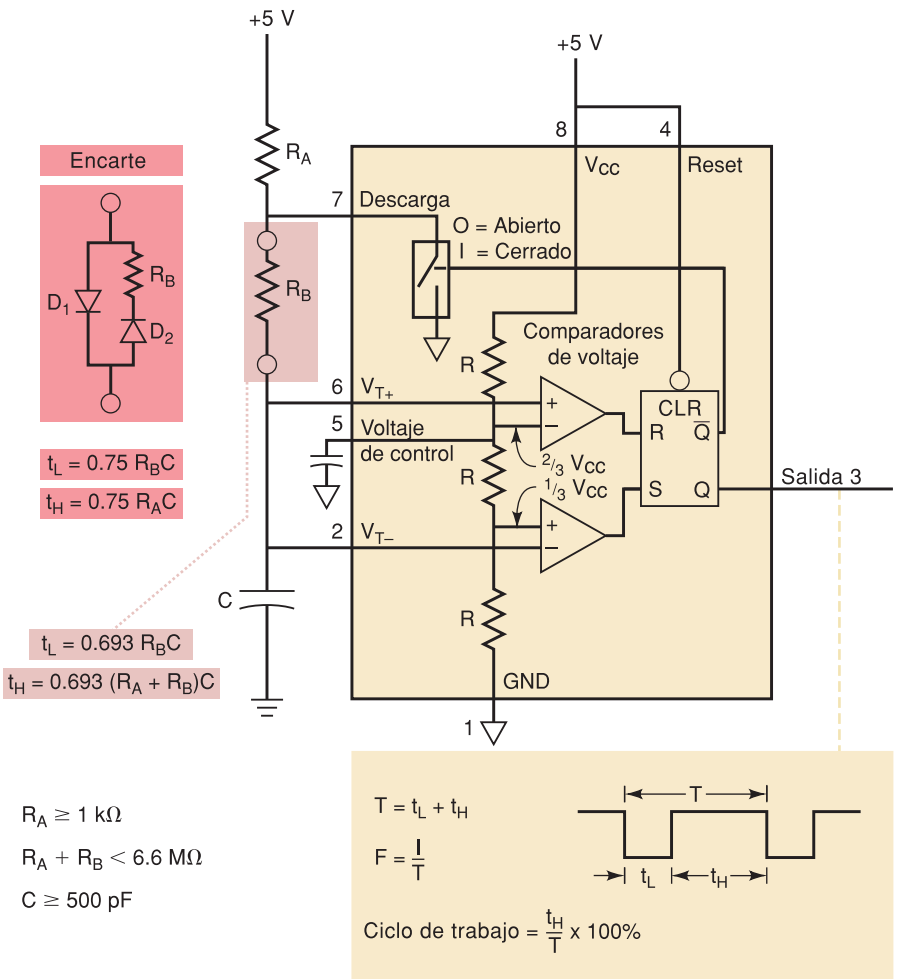


FIGURA 5-56 CI temporizador 555 utilizado como multivibrador astable.

lógicos, en donde los espacios de tiempo en cada nivel lógico se determinan con base en los valores de R y de C.

El corazón del temporizador 555 está compuesto de dos comparadores de voltaje y un latch SR, como se muestra en la figura 5-56. Los comparadores de voltaje son dispositivos que producen una salida en ALTO cada vez que el voltaje en la entrada + es mayor que el voltaje en la entrada −. El capacitor externo (C) se carga hasta que su voltaje excede de $^2/_3 \times V_{CC}$, según lo determinado por el comparador de voltaje superior que monitorea a V_{T+}. Cuando la salida de este comparador cambia a ALTO, restablece el latch SR, provocando que la terminal de salida (3) cambie a BAJO. Al mismo tiempo, $\overline{Q}$ cambia a ALTO y cierra el interruptor de descarga, haciendo que el capacitor comience a descargarse a través de R_B. Continuará descargándose hasta que su voltaje caiga por debajo de $^1/_3 \times V_{CC}$, según lo determinado por el comparador de voltaje inferior que monitorea a V_{T-}. Cuando la salida de este comparador cambia a ALTO establece el latch SR, haciendo que la terminal de salida cambie a ALTO y se abre el interruptor de descarga, con lo cual el capacitor puede empezar a cargarse otra vez, a medida que se repite el ciclo.

Las fórmulas para estos intervalos (t_L y t_H) y para el periodo total de las oscilaciones (T) se proporcionan en la figura. Desde luego que la frecuencia de las oscilaciones es el recíproco de T. Como indican las fórmulas en el diagrama, los intervalos t_L y t_H no pueden ser iguales a menos que R_A se haga cero. Esto no puede hacerse sin producir corriente en exceso a través del dispositivo. Esto significa que con este circuito es imposible producir una salida de onda cuadrada perfecta con un ciclo de trabajo del 50 por ciento. No obstante, es posible llegar muy cerca del 50 por ciento del ciclo de trabajo si hacemos que $R_B >> R_A$ (mientras R_A se mantenga mayor que 1 kΩ), de manera que $t_L \approx t_H$.

EJEMPLO 5-17

Calcule la frecuencia y el ciclo de trabajo de la salida del multivibrador astable 555 para $C = 0.001\ \mu F$, $R_A = 2.2\ k\Omega$ y $R_B = 100\ k\Omega$.

Solución

$$t_L = 0.693(100\ k\Omega)(0.001\ \mu F) = 69.3\ \mu s$$
$$t_H = 0.693(102.2\ k\Omega)(0.001\ \mu F) = 70.7\ \mu s$$
$$T = 69.3 + 70.7 = 140\ \mu s$$
$$f = 1/140\ \mu s = 7.29\ kHz$$
$$\text{ciclo de trabajo} = 70.7/140 = 50.5\%$$

Observe que el ciclo de trabajo está cerca del 50 por ciento (onda cuadrada) ya que R_B es mucho mayor que R_A. Puede hacerse aún más cercano al 50 por ciento si R_B se hace aún más grande en comparación con R_A. Por ejemplo, debería verificar que si cambiamos R_A a 1 kΩ (su valor mínimo permitido), los resultados son $f = 7.18$ kHz y el ciclo de trabajo = 50.3 por ciento.

Puede hacerse una simple modificación a este circuito para permitir un ciclo de trabajo de menos del 50 por ciento. La estrategia es permitir que el capacitor se llene (cargue) con partículas cargadas que fluyan sólo a través de R_A y se vacíe (descargue) a medida que las partículas cargadas fluyen sólo a través de R_B. Para lograr esto basta con conectar un diodo (D_2) en serie con R_B y otro diodo (D_1) en paralelo con R_B y D_2, como muestra el encarte de la figura 5-56. El circuito del encarte sustituye a R_B en el dibujo. Los diodos son dispositivos que permiten que las partículas cargadas fluyan a través de ellos sólo en una dirección, según lo que indique la punta de la flecha. El diodo D_1 permite que toda la corriente de carga que ha pasado a través de R_A se desvíe de R_B, y D_2 asegura que la corriente de carga no pueda fluir a través de R_B. Toda la corriente de descarga fluye a través

de D_2 y de R_B cuando se cierra el interruptor de descarga. Las ecuaciones para el tiempo alto y el tiempo bajo para este circuito son

$$t_L = 0.75\, R_B C$$
$$t_H = 0.75\, R_A C$$

Nota: La constante 0.75 es correcta sólo para $V_{CC} = 5$ V.

EJEMPLO 5-18

Utilice los diodos junto con R_B, como se muestra en la figura 5-56, para calcular los valores de R_A y R_B necesarios para obtener una forma de onda de 1 kHz con un ciclo de trabajo del 25 por ciento como salida de un 555. Suponga que C es un capacitor de 0.1 μF.

Solución

$$T = \frac{1}{F} = \frac{1}{1000} = 0.001 \text{ s} = 1 \text{ ms}$$
$$t_H = 0.25 \times T = 0.25 \times 1 \text{ ms} = 250\ \mu\text{s}$$
$$R_A = \frac{250\ \mu\text{s}}{0.75 \times C} = \frac{250\ \mu\text{s}}{0.75 \times 0.1\ \mu\text{F}} = 3.3\ \text{k}\Omega$$
$$R_B = \frac{750\ \mu\text{s}}{0.75 \times C} = \frac{750\ \mu\text{s}}{0.75 \times 0.1\ \mu\text{F}} = 10\ \text{k}\Omega$$

Generadores de reloj controlados por cristal

Las frecuencias de salida de las señales de los circuitos generadores de reloj antes descritos dependen de los valores de las resistencias y los capacitares, por lo cual no son en extremo precisos o estables. Aún si se utilizaran resistencias variables, de manera que la frecuencia deseada pudiera ajustarse mediante un ajuste en los valores de la resistencia, se producirían cambios en los valores de R y C debido a los cambios en la temperatura ambiental y al desgaste, con lo cual la frecuencia ajustada variaría. Si es imprescindible tener una frecuencia precisa y estable, podemos utilizar otro método para generar señales de reloj: un **generador de reloj controlado por cristal**. Este tipo de generador emplea un componente de alta estabilidad y precisión, al cual se le conoce como *cristal de cuarzo*. Puede cortarse una pieza de cristal de cuarzo a un tamaño y forma específicos para que vibre (resuene) a una frecuencia precisa que es en extremo estable en relación con los cambios de temperatura y el desgaste; pueden lograrse frecuencias desde 10 kHz hasta 80 MHz. Cuando se coloca un cristal en ciertas configuraciones de circuitos, puede producir oscilaciones a una frecuencia precisa y estable, igual a la frecuencia resonante del cristal. Los osciladores de cristal están disponibles como encapsulados de CI.

Los circuitos generadores de reloj controlados por cristal se utilizan en todos los sistemas basados en microprocesador y en las microcomputadoras, y en cualquier aplicación en la que se utilice una señal de reloj para generar intervalos de sincronización precisos. En los siguientes capítulos veremos su funcionamiento en algunas de las aplicaciones de ejemplo.

PREGUNTAS DE REPASO

1. Determine la frecuencia aproximada de un oscilador disparador de Schmitt que utiliza un 74HC14 con $R = 10$ kΩ y $C = 0.005$ μF.
2. Determine la frecuencia y el ciclo de trabajo aproximados del oscilador 555 para $R_A = R_B = 2.2$ kΩ y $C = 2000$ pF.
3. ¿Cuál es la ventaja de los circuitos generadores de reloj controlados por cristal, en comparación con los circuitos controlados por *RC*?

5-24 DIAGNÓSTICO DE FALLAS EN CIRCUITOS CON FLIP-FLOPS

Los CIs con flip-flops son susceptibles a los mismos tipos de fallas internas y externas que ocurren en los circuitos lógicos combinacionales. Todas las ideas sobre diagnóstico de fallas que vimos en el capítulo 4 pueden aplicarse también a los circuitos que contienen FFs, así como compuertas lógicas.

Debido a su característica de memoria y a su operación sincronizada, los circuitos con FFs están sujetos a varios tipos de fallas y síntomas asociados que no se presentan en los circuitos combinacionales. En especial, los circuitos con FFs son susceptibles a los problemas de sincronización que, por lo general, no son de cuidado en los circuitos combinacionales. A continuación describiremos los tipos más comunes de fallas en los circuitos con FFs.

Entradas abiertas

Las entradas desconectadas o flotantes de cualquier circuito lógico son en especial susceptibles a la atracción de fluctuaciones espurias de voltaje, conocidas como *ruido*. Si el ruido es lo suficientemente grande en amplitud y largo en duración, la salida del circuito lógico puede llegar a cambiar de estado en respuesta a ese ruido. En una compuerta lógica, la salida regresará a su estado original cuando la señal de ruido se detenga. No obstante, en un FF la salida permanecerá en su nuevo estado debido a su característica de memoria. Por ende, el efecto de atraer ruido en cualquier entrada abierta es, por lo general, más crítico para un FF o latch que para una compuerta lógica.

Las entradas más susceptibles en un FF son las que pueden dispararlo a un estado distinto, tales como *CLK*, PRESET y CLEAR. Cada vez que vea que la salida de un FF cambia de estado en forma errónea, deberá considerar la posibilidad de una conexión abierta en una de estas entradas.

EJEMPLO 5-19

La figura 5-57 muestra un registro de desplazamiento de tres bits compuesto de flip-flops TTL. En un principio todos los FFs se encuentran en el estado BAJO, antes de aplicar pulsos de reloj.

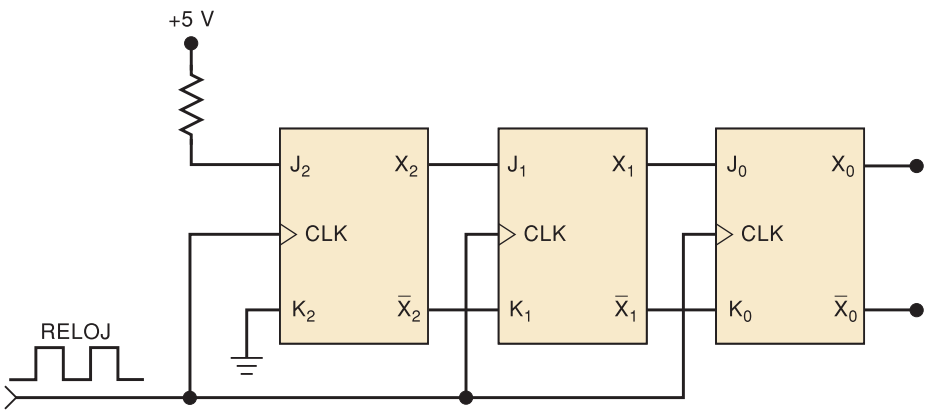


FIGURA 5-57 Ejemplo 5-19.

Número de pulso de reloj	"Esperado" X_2	X_1	X_0	"Real" X_2	X_1	X_0
0	0	0	0	0	0	0
1	1	0	0	1	0	0
2	1	1	0	1	1	0
3	1	1	1	1	1	1
4	1	1	1	1	1	0
5	1	1	1	1	1	1
6	1	1	1	1	1	0
7	1	1	1	1	1	1
8	1	1	1	1	1	0

A medida que se apliquen pulsos de reloj, cada PGT hará que la información se desplace desde cada FF hacia el que se encuentra a su derecha. El diagrama muestra la secuencia "esperada" de estados de los FFs después de cada pulso de reloj. Como $J_2 = 1$ y $K_2 = 0$, el flip-flop X_2 cambiará a ALTO en el pulso de reloj 1 y permanecerá ahí durante todos los siguientes pulsos. Este nivel ALTO se desplazará hacia X_1 y después hacia X_0 en los pulsos 2 y 3, respectivamente. En consecuencia, después del tercer pulso todos los FFs estarán en ALTO y deberán permanecer ahí a medida que se apliquen pulsos en forma continua.

Ahora suponga que la respuesta "real" de los estados de los FFs es como se muestra en el diagrama. Aquí los FFs cambian según lo esperado durante los primeros tres pulsos de reloj. De ahí en adelante el flip-flop X_0, en vez de permanecer en ALTO alterna entre ALTO y BAJO. ¿Cuál de las posibles fallas del circuito pueden producir esta situación?

Solución

En el segundo pulso X_1 cambia a ALTO, con lo cual se debe producir la condición $J_0 = 1, K_0 = 0$ de manera que todos los pulsos de reloj subsiguientes deberán establecer la condición $X_0 = 1$. En vez de ello, vemos que X_0 cambia de estado (conmuta) en todos los pulsos después del segundo. Esta operación de conmutación se produciría si J_0 y K_0 estuvieran ambas en ALTO. La falla más probable es una interrupción en la conexión entre $\overline{X}_1$ y K_0. Recuerde que un dispositivo TTL responde a una entrada abierta como si fuera un nivel ALTO lógico, por lo que un circuito abierto en K_0 es lo mismo que un nivel ALTO.

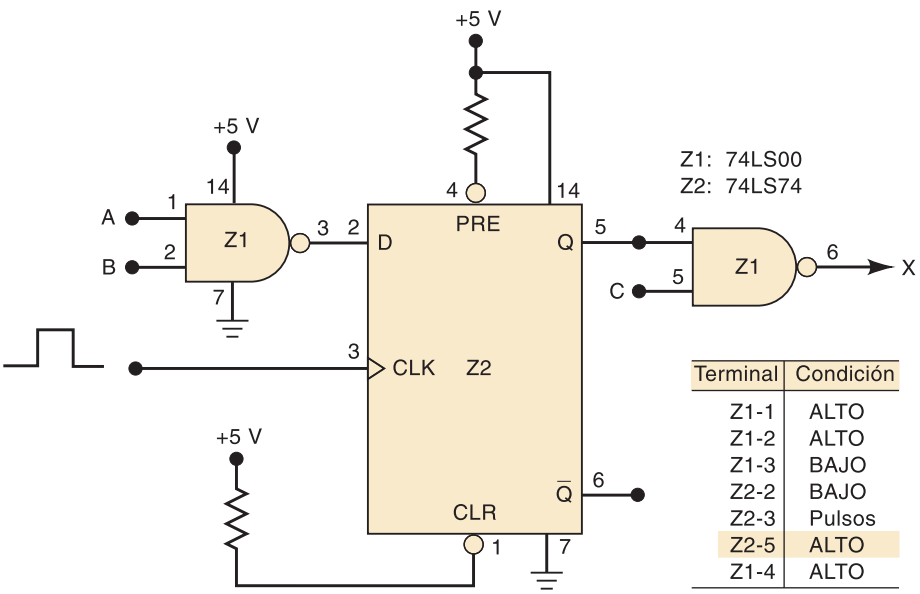
Salidas en corto

El siguiente ejemplo ilustra cómo una falla en un circuito con FF puede provocar un síntoma confuso, por el cual se requiera más tiempo para aislar la falla.

EJEMPLO 5-20

Considere el circuito de la figura 5-58 y examine las indicaciones de la sonda lógica que se muestran en la tabla que lo acompaña. Hay un nivel BAJO en la entrada *D* del FF cuando se aplican pulsos a su entrada *CLK*, pero la salida *Q* no cambia al estado BAJO. El técnico que está probando este circuito considera cada una de las posibles fallas:

1. Que Z2-5 esté en corto interno con V_{CC}.
2. Que Z1-4 esté en corto interno con V_{CC}.



Terminal	Condición
Z1-1	ALTO
Z1-2	ALTO
Z1-3	BAJO
Z2-2	BAJO
Z2-3	Pulsos
Z2-5	ALTO
Z1-4	ALTO

FIGURA 5-58 Ejemplo 5-20.

3. Que Z2-5 o Z1-4 estén en corto externo con V_{CC}.
4. Que Z2-4 esté en corto interno o externo con TIERRA. Con esto se mantendría la entrada *PRE* activa y se ignoraría la entrada *CLK*.
5. Que haya una falla interna en Z2, la cual evite que *Q* responda en forma apropiada a sus entradas.

Después de realizar las comprobaciones necesarias con el óhmetro, el técnico descarta las primeras cuatro posibilidades. También comprueba las terminales V_{CC} y TIERRA de Z2 y encuentra que tienen los voltajes apropiados. Está renuente a desoldar Z2 del circuito hasta que esté seguro de que está defectuoso, por lo que decide analizar la señal del reloj. Utiliza un osciloscopio para revisar su amplitud, frecuencia, anchura de pulso y tiempos de transición. Encuentra que todos están dentro de las especificaciones para el 74LS74. Por último concluye que Z2 está defectuoso.

Quita el chip 74LS74 y lo sustituye con otro. Para su sorpresa, el circuito con el nuevo chip se comporta de igual forma. Después de rascarse la cabeza, decide cambiar el chip de la compuerta NAND aunque no sabe por qué. Como lo esperaba, no ve ningún cambio en la operación del circuito.

Quedando aún más desconcertado, recuerda que su instructor del laboratorio de electrónica enfatizaba el valor de realizar una comprobación visual detallada en el tablero de circuitos, por lo que comienza a examinarlo con cuidado. Mientras lo hace, detecta un puente de soldadura entre las terminales 6 y 7 de Z2. Lo elimina, prueba el circuito y éste funciona de manera correcta. Explique cómo esta falla produjo la operación observada.

Solución

El puente de soldadura estaba poniendo en corto la salida $\overline{Q}$ con TIERRA. Esto significa que $\overline{Q}$ se mantuvo siempre en el nivel BAJO. Recuerde que en todos los latches y FFs las salidas $\overline{Q}$ y Q en su interior están conectadas en forma transversal, de manera que el nivel en una afectará al nivel en la otra. Por ejemplo, de otro vistazo a los circuitos internos para un flip-flop J-K en la figura 5-25. Observe que un nivel BAJO constante en $\overline{Q}$ mantendría un nivel BAJO en una entrada de la compuerta NAND 3, por lo que Q tendría que mantenerse en ALTO, sin importar las condiciones en *J*, *K* y *CLK*.

El técnico aprendió una valiosa lección acerca del diagnóstico de fallas en circuitos con FF. Aprendió que deben comprobarse ambas salidas para ver si hay fallas, aún aquellas que no están conectadas a otros dispositivos.

Desfasamiento de reloj

Uno de los problemas de sincronización más comunes en los circuitos secuenciales es el **desfasamiento de reloj**. Esto ocurre cuando una señal de reloj, debido a los tiempos de propagación, llega a las entradas *CLK* de distintos FFs en distintos tiempos. En muchas situaciones el desfasamiento puede provocar que un FF cambie a un estado incorrecto. Esto se ilustra mejor con un ejemplo.

Consulte la figura 5-59(a), en donde la señal *RELOJ1* está conectada en forma directa al FF Q_1 y en forma indirecta a Q_2 por medio de una compuerta NAND y un INVERSOR. Se supone que ambos FFs deben estar sincronizados por la ocurrencia de una NGT de *RELOJ1*, siempre y cuando *X* esté en ALTO. Si suponemos que al principio $Q_1 = Q_2 = 0$ y $X = 1$, la NGT de *RELOJ1* debería hacer que $Q_1 = 1$ y no tendría efecto sobre Q_2. Las formas de onda en la figura 5-59(b) muestran cómo el desfasamiento del reloj puede producir un disparo incorrecto de Q_2.

Debido a la combinación de los tiempos de propagación de la compuerta NAND y del INVERSOR, las transiciones de la señal *RELOJ2* se retrasan con respecto a *RELOJ1* por una cantidad de tiempo t_1. La NGT de *RELOJ2* llega a la entrada *CLK* de Q_2 un tiempo t_1 después de que la NGT de *RELOJ1* aparece en la entrada *CLK* de Q_1. Este t_1 es el desfasamiento del reloj. La NGT de *RELOJ1* hará que Q_1



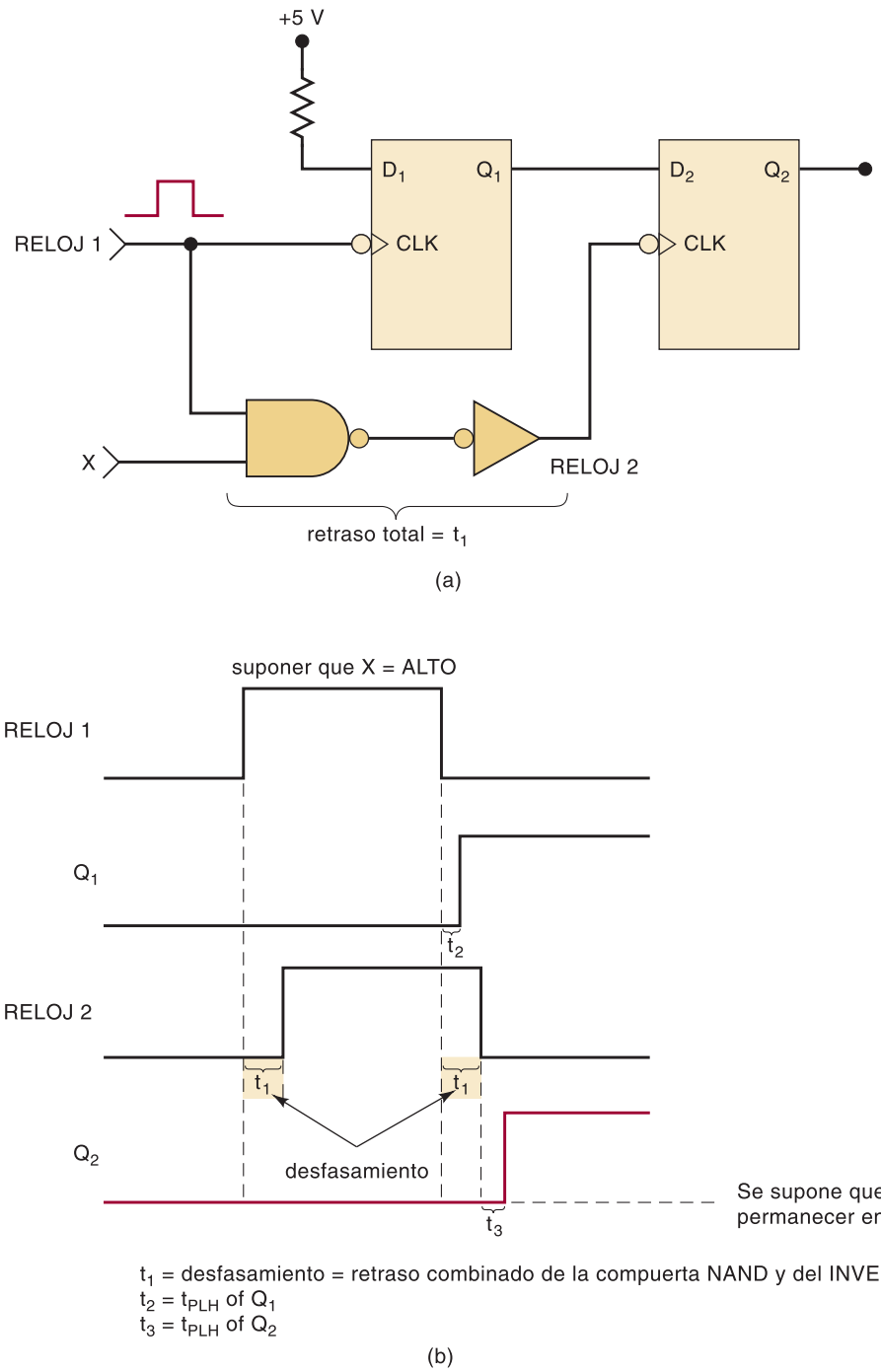


FIGURA 5-59 El desfasamiento del reloj se produce cuando dos flip-flops que se supone deben estar sincronizados en forma simultánea se sincronizan en tiempos ligeramente distintos debido a un retraso en la llegada de la señal de reloj en el segundo flip-flop. (a) Circuitos de compuerta adicionales que pueden provocar un desfasamiento del reloj; (b) sincronización que muestra la llegada tardía de RELOJ 2.

cambie a ALTO después de un tiempo t_2, el cual es igual al retraso de propagación t_{PLH} de Q_1. Si t_2 es menor que el desfasamiento t_1, Q_1 estará en ALTO cuando ocurra la NGT de *RELOJ2* y esto podría provocar que Q_2 se estableciera de forma incorrecta en 1 ($Q_2 = 1$) si se cumple su requerimiento de tiempo de estabilización, t_S.

Por ejemplo, suponga que el desfasamiento del reloj es de 40 ns y que el t_{PLH} de Q_1 es de 25 ns. Por ende, Q_1 cambiará a ALTO 15 ns antes de la NGT de *RELOJ2*. Si el requerimiento de tiempo de estabilización de Q_2 es menor de 15 ns, Q_2 responderá al nivel ALTO en su entrada *D* cuando ocurra la NGT de *RELOJ2*, y Q_2 cambiará a ALTO. Desde luego que ésta no es la respuesta esperada de Q_2. Se supone que debe permanecer en BAJO.

Los efectos del desfasamiento del reloj no siempre son fáciles de detectar, ya que la respuesta del FF afectado puede ser intermitente (algunas veces funciona bien, otras no). Esto se debe a que la situación es dependiente de los tiempos de propagación del circuito y de los parámetros de sincronización del FF, los cuales varían con la temperatura, la longitud de las conexiones, el voltaje de suministro de energía y la carga. Algunas veces con sólo conectar una sonda de osciloscopio a la salida de un FF o de una compuerta se agrega la suficiente capacitancia de carga como para incrementar el retraso de propagación del dispositivo de manera que el circuito funcione en forma correcta; después cuando se quita la sonda, reaparece la operación incorrecta. Éste es el tipo de situación que explica por qué algunos técnicos envejecen en forma prematura.

Los problemas ocasionados por el desfasamiento del reloj pueden eliminarse si se ecualizan los retrasos en las diversas rutas de la señal del reloj, de manera que la transición activa llegue a cada FF casi al mismo tiempo. En el problema 5-52 examinaremos esta situación.

PREGUNTAS DE REPASO

1. ¿Qué es el desfasamiento del reloj? ¿Cómo puede ocasionar un problema?

5-25 CIRCUITOS SECUENCIALES MEDIANTE EL USO DE HDL*

En los capítulos 3 y 4 utilizamos el HDL para programar circuitos lógicos combinacionales simples. En este capítulo hemos estudiado circuitos lógicos tipo latch y circuitos de flip-flop sincronizado por reloj que avanzan en secuencia a través de varios estados, en respuesta a un flanco del reloj. Estos circuitos de latch y secuenciales también pueden implementarse mediante el uso de los PLDs, y también pueden describirse mediante el uso de HDL.

En la sección 5-1 de este capítulo describimos un match de compuerta NAND. Recordará que la característica única de este circuito es el hecho de que sus salidas están conectadas en forma transversal de vuelta con las entradas de sus compuertas. Esto hace que el circuito responda de manera distinta, dependiendo del estado en el que se encuentre su salida. Para describir circuitos que tienen salidas que se *retroalimentan* a la entrada con ecuaciones booleanas o mediante HDL se requiere utilizar las variables de salida en la porción condicional de la descripción. Con las ecuaciones booleanas esto significa incluir los términos de salida en el lado derecho de la ecuación. Al utilizar instrucciones IF/THEN se deben incluir las variables de salida en la cláusula IF. La mayoría de los PLDs tienen la habilidad de retroalimentar la señal de salida a los circuitos de entrada, para poder acomodar la acción del latch.

Al escribir ecuaciones que utilizan retroalimentación, algunos lenguajes tales como el VHDL requieren una designación especial para el puerto de salida. En estos casos, el bit de puerto no es sólo una salida; es una salida con retroalimentación. La diferencia se muestra en la figura 5-60.

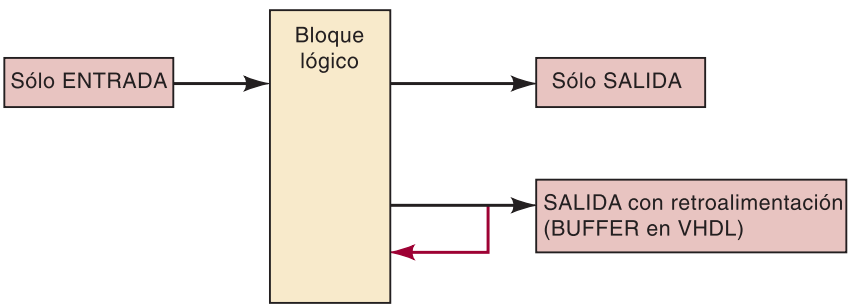


FIGURA 5-60 Tres modos de entrada/salida.

* Como se dijo en el capítulo 3, puede omitir esta sección y todas las secciones que tratan acerca de PLDs y HDLs si lo desea.

En vez de describir la operación de un latch mediante ecuaciones Booleanas, tratemos de pensar en una descripción del comportamiento en cuanto a la operación del latch. Las situaciones que necesitamos abordar son cuando se activa SBAR, cuando se activa RBAR y cuando no se activa ninguna de las dos. Recuerde que el estado inválido ocurre cuando ambas entradas se activan al mismo tiempo. SI podemos describir un circuito que siempre reconoce una de las entradas como la ganadora cuando ambas están activas, podremos evitar los resultados indeseables de tener una condición de entrada inválida. Para describir tal circuito, tenemos que hacernos la pregunta acerca de bajo qué condiciones debe establecerse el latch ($Q = 1$). Es evidente que el latch debe establecerse si la entrada SET está activa, pero ¿qué pasa después de que SET regresa a su nivel inactivo? ¿Cómo sabe el latch que debe permanecer en el estado SET? La descripción necesita utilizar la *condición de la salida en estos momentos* para determinar la condición *futura* de la misma. La siguiente instrucción describe las condiciones que deben hacer que la salida cambie a ALTO en un latch SR:

SI (IF) *ESTABLECER* está activa, ENTONCES (THEN) *Q* debe estar en ALTO.

¿Qué condiciones deben hacer que la salida cambie a BAJO?

SI (IF) *RESET* está activa, ENTONCES (THEN) *Q* debe estar en BAJO.

¿Qué pasa si no se activa ninguna entrada? Entonces la salida deberá permanecer igual y podemos expresar esto como $Q = Q$. Esta expresión proporciona la retroalimentación del estado de salida para combinarla con las condiciones de entrada, con el fin de decidir lo que va a ocurrirle a continuación a la salida.

¿Qué pasa si se activan ambas entradas (es decir, la combinación de entrada inválida)? La estructura de la decisión IF/ELSE que se muestra en forma gráfica en la figura 5-61 se asegura de que el latch nunca trate de responder a ambas entradas. Si SET está activa, sin importar lo que haya en RESET la salida se verá forzada a cambiar a ALTO. De esta forma, la entrada inválida siempre tendrá como valor predeterminado la condición de SET. La cláusula ELSIF se considera sólo cuando SET no está activa. El uso del término de retroalimentación ($Q = Q$) afecta a la operación (acción de retención) sólo cuando ninguna de las entradas están activas.

Cuando diseñamos circuitos secuenciales que alimentan el valor de salida de vuelta a las entradas, existe la posibilidad de crear un sistema inestable. Un cambio en el estado de salida podría alimentarse de vuelta a las entradas, con lo

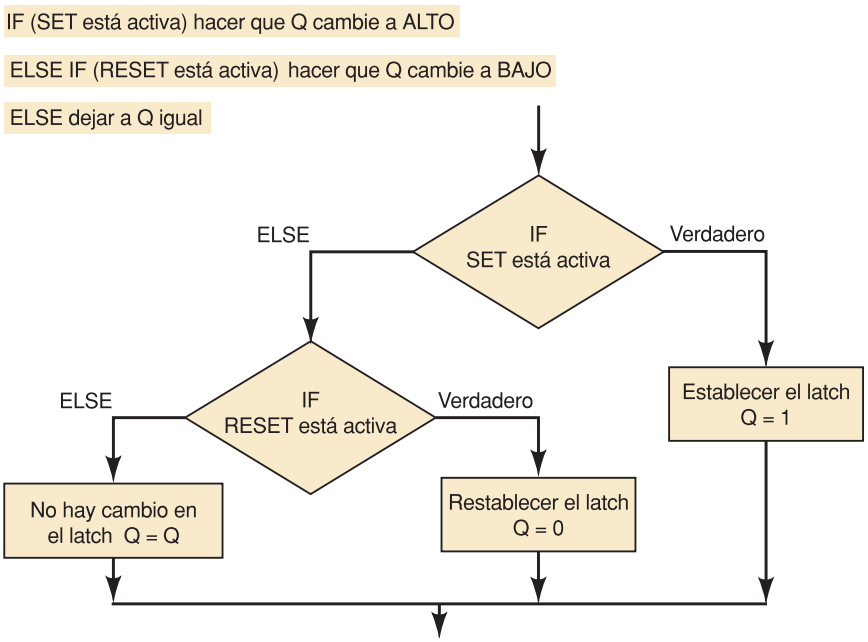


FIGURA 5-61 La lógica de una descripción del comportamiento de un latch SR.

cual se cambiaría de nuevo el estado de la salida, el cual se retroalimentaría a las entradas, con lo que se cambiaría la salida de nuevo. Es obvio que esta oscilación es indeseable, por lo cual es muy importante asegurarnos de que ninguna combinación de entradas y salidas pueda hacer que esto ocurra. Debemos utilizar un proceso cuidadoso de análisis, simulación y prueba para asegurarnos que nuestro circuito sea estable bajo todas las condiciones. Para este circuito es necesario habilitar la síntesis de multinivel para la serie MAX 7000 de componentes antes de realizar la compilación, con lo cual se evita la oscilación (por lo menos en el simulador) cuando se cambia de la entrada "inválida" a la entrada "sin cambios".

EJEMPLO 5-21

Describa un match SR con entradas activas en BAJO identificadas como SBAR, RBAR y una salida llamada *Q*. Debe seguir la tabla de funciones de un latch NAND (vea la figura 5-6) y la combinación de entrada inválida debe producir $Q = 1$.

(a) Use AHDL.
(b) Use VHDL.

AHDL

Solución

(a) La figura 5-62 muestra una posible solución en AHDL. Los puntos importantes a observar son:

1. *Q* se define como una SALIDA, aún y cuando se *retroalimenta* en el circuito. AHDL permite que las salidas se retroalimenten al circuito.
2. La cláusula después del IF determinará el estado de salida que se produzca cuando ambas entradas estén activas (estado inválido). En este código el comando SET es el que predomina.
3. Para evaluar la igualdad se utiliza el doble signo de igual. En otras palabras, SBAR == 0 se evalúa como VERDADERO cuando SBAR está activa (BAJO).

```
SUBDESIGN fig5_62
(
      sbar, rbar              :INPUT;
      q                       :OUPTUT;
)
BEGIN
      IF    sbar == 0   THEN  q = VCC;     -- establecer o comando ilegal
      ELSIF rbar == 0   THEN  q = GND;     -- restablecer
      ELSE                    q = q;       -- retención
      END IF;
END;
```

FIGURA 5-62 Un latch NAND mediante el uso de AHDL.

VHDL

(b) La figura 5-63 muestra una posible solución en VHDL. Los puntos importantes a observar son:

1. *Q* se define como BUFFER en lugar de OUTPUT. Esto le permite retroalimentarse al circuito.
2. La palabra PROCESS describe lo que ocurre cuando los valores en la lista de sensibilidad (SBAR, RBAR) cambian de estado.
3. La cláusula después de IF determinará el estado de salida que se produzca cuando ambas entradas estén activas (estado inválido). En este código el comando SET es el que predomina.

```
-- debe compilarse con Síntesis de Multinivel para MAX 7000 habilitada
ENTITY fig5_63 IS
PORT (sbar, rbar         :IN BIT;
      Q                  :BUFFER BIT);
END fig5_63;
ARCHITECTURE comportamiento OF fig5_63 IS
BEGIN
    PROCESS (sbar, rbar)
      BEGIN
            IF     sbar = '0' THEN     q <= '1'; -- establecer o comando ilegal
            ELSIF  rbar = '0' THEN     q <= '0'; -- restablecer
            ELSE                       q <= q;   -- retención
            END IF;
      END PROCESS;
END comportamiento;
```

FIGURA 5-63 Un latch NAND mediante el uso de VHDL.

El latch *D*

El latch *D* transparente también puede implementarse con facilidad mediante los HDLs. El software de Altera cuenta con una primitiva de biblioteca llamada LATCH. El módulo de AHDL que se muestra a continuación ilustra el uso de esta primitiva LATCH. Todo lo que se necesita es conectar los puertos de habilitación (.ena) y de datos (.d) de la primitiva con las señales del módulo apropiado. El módulo VHDL que se muestra también a continuación es una descripción del comportamiento de la función del latch *D*. También puede usar la primitiva LATCH como un componente en VHDL.

AHDL D latch

```
SUBDESIGN latchd_ahdl
(habilita,    dent          :INPUT;
     q                      :OUTPUT;)

VARIABLE
       q                    :LATCH;
BEGIN
       q.ena = habilita;
       q.d = dent;
END;
```

VHDL D latch

```
ENTITY  latchd_vhdl  IS
PORT (habilita, dent       :IN BIT;
        q                  :OUT BIT);
END  latchd_vhdl;

ARCHITECTURE v OF latchd_vhdl  IS
BEGIN
       PROCESS (habilita, dent)
       BEGIN
          IF  habilita 5 '1'  THEN
               q <= dent;
          END IF;
       END PROCESS;
END v;
```

PREGUNTAS DE REPASO

1. ¿Cuál es la característica distintiva de hardware de los circuitos lógicos con latches?
2. ¿Cuál es la principal característica de los circuitos secuenciales?

5-26 DISPOSITIVOS DISPARADOS POR FLANCO

En secciones anteriores de este capítulo presentamos los dispositivos disparados por flanco, cuyas salidas responden a las entradas cuando la entrada de reloj ve un "flanco", el cual lo único que indica es una transición de ALTO a BAJO o viceversa, y a menudo se le llama **evento**. Si en el código vamos a escribir instrucciones concurrentes, ¿cómo pueden cambiar las salidas sólo cuando una entrada de reloj detecta un evento de flanco? La respuesta a esta pregunta difiere en forma sustancial, dependiendo del HDL que usted utilice. En esta sección queremos concentrarnos en la creación de circuitos lógicos sincronizados por reloj en su forma más simple, mediante el uso de HDL. Utilizaremos flip-flops J-K para correlacionar con muchos de los ejemplos que vimos antes en este capítulo.

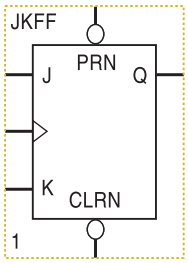


FIGURA 5-64
Primitiva lógica del flip-flop J-K.

El flip-flop J-K es un bloque de construcción estándar de circuitos lógicos sincronizados por reloj (secuenciales), al cual se le conoce como **primitiva lógica**. En su forma más común tiene cinco entradas y una salida, como se muestra en la figura 5-64. Se pueden estandarizar los nombres de las entradas/salidas para poder hacer referencia a las conexiones de este circuito primitivo o fundamental. La operación real del circuito primitivo se define en una biblioteca de componentes, la cual está disponible para el compilador de HDL a medida que éste genera un circuito a partir de nuestra descripción. AHDL utiliza primitivas lógicas para describir la operación de un flip-flop. VHDL ofrece algo similar, pero también permite al diseñador describir la operación del circuito lógico sincronizado por reloj de manera explícita en el código.

AHDL

FLIP-FLOPS EN AHDL

Para utilizar un flip-flop en AHDL hay que declarar un registro (de hecho a un flip-flop se le conoce como registro). Hay varios tipos de primitivas de registros disponibles en AHDL, incluyendo JKFF, DFF, SRFF y latch. Cada tipo distinto de primitiva de registros tiene sus propios nombres oficiales (de acuerdo con el software de Altera) para los puertos de estas primitivas. Para encontrarlos podemos usar el menú HELP (ayuda) en el software de ALTERA y buscar bajo Primitives (primitivas). La tabla 5-3 lista algunos de estos nombres. Los registros que utilizan estas primitivas se declaran en la sección VARIABLE del código. Al registro se le da un nombre de instancia, de la misma forma como hemos nombrado a las variables intermedias o a los nodos ocultos en ejemplos anteriores. No obstante, en vez de declararlo como nodo se declara con base en el tipo de primitiva del registro. Por ejemplo, un flip-flop JK puede declararse como:

```
VARIABLE
       ff1  :JKFF;
```

TABLA 5-3
Identificadores de puertos de primitiva de Altera.

Función de pieza estándar	Nombre de puerto de primitiva
Entrada de reloj	clk
PRESET asíncrono (activo en BAJO)	prn
CLEAR asíncrono (activo en BAJO)	clrn
Entradas J, K, S, R, D	j, k, s, r, d
Entrada HABILITAR disparada por nivel	ena
Salida Q	Q

El nombre de instancia es ff1 (usted puede inventarlo) y el tipo de primitiva de registro es JKFF (es requerido por Altera). Una vez declarado un registro, éste se conecta con la demás lógica en el diseño mediante sus nombres de puertos estándar.

```
1   %     JK circuito flip-flop     %
2   SUBDESIGN fig5_65
3   (
4         jent, kent, clkent, preset, clear    :INPUT
5         qsal                                 :OUTPUT;
6   )
7   VARIABLE
8   ff1         :JKFF;       -- define este flip flop como de tipo JKFF
9   BEGIN
10        ff1.prn = preset; -- éstos son opcionales, su valor predeterminado es vcc
11        ff1.clrn = clear;
12        ff1.j = jent;     -- conecta primitiva a la señal de entrada
13        ff1.k = kent;
14        ff1.clk = clkent;
15        qsal = ff1.q;     -- conecta la terminal de salida a la primitiva
16  END;
```

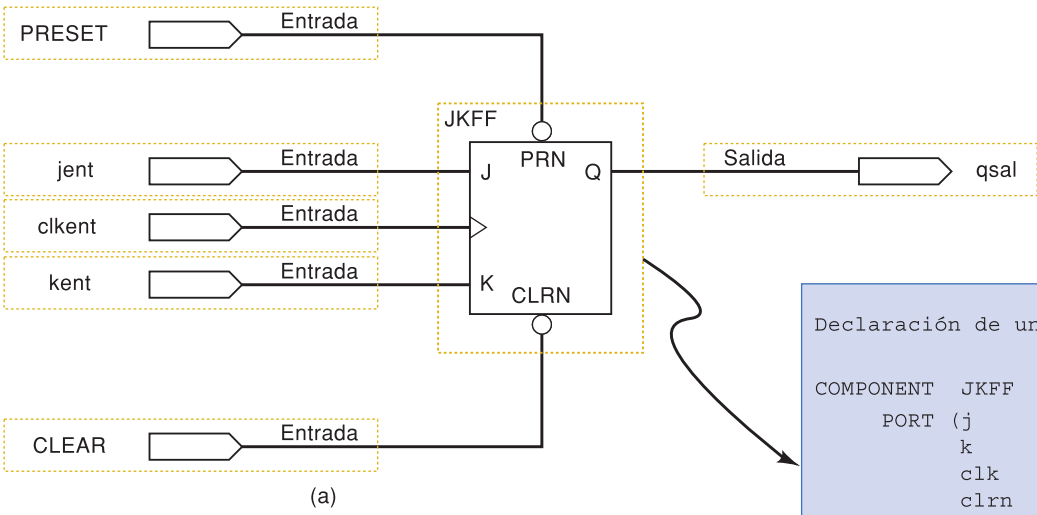
FIGURA 5-65 Flip-flop JK individual mediante el uso de AHDL.

Para hacer referencia a los puertos (o terminales) en el flip-flop se utiliza el nombre de instancia, con una extensión después del punto que designa la entrada o salida específica. La figura 5-65 muestra un ejemplo para un flip-flop JK en AHDL. Observe que hemos creado nuestros propios nombres para las entradas/salidas en este SUBDESIGN, para poder distinguirlos de los nombres de los puertos de las primitivas. El flip-flop individual se declara en la línea 8, como se describió antes. La entrada o puerto J para este dispositivo se etiqueta entonces como *ff1.j*, la entrada K es *ff1.k*, la entrada de reloj es *ff1.clk* y así en lo sucesivo. Cada una de las instrucciones de asignación de puertos dadas realizarán las conexiones de alambrado necesarias para este bloque de diseño. Los puertos *prn* y *clrn* son controles asíncronos activos en BAJO, como los que se encuentran, por lo general, en un flip-flop estándar. De hecho, estos controles asíncronos en una primitiva de FF pueden utilizarse para implementar un latch SR con más eficiencia que el código en la figura 5-62. Los controles *prn* y *clrn* son opcionales en AHDL y su valor predeterminado es una condición deshabilitada (en un 1 lógico) si se omiten de la sección lógica. En otras palabras, si se eliminaran las líneas 10 y 11 los puertos *prn* y *clrn* de ff1 se enlazarían en forma automática a V_{CC}.

COMPONENTES DE LAS BIBLIOTECAS DE VHDL

El software de Altera incluye varias bibliotecas extensas de componentes y primitivas para que las utilicen los diseñadores. En la figura 5-66(a) se muestra la descripción gráfica de un componente JKFF en la biblioteca Altera. Después de colocar el componente en la hoja de trabajo, cada uno de sus puertos se conecta a las entradas y salidas del módulo. Este mismo concepto puede implementarse en VHDL mediante el uso de un componente de biblioteca. Para encontrar las entradas y salidas de estos componentes de biblioteca hay que buscar bajo el menú HELP/Primitives. La figura 5-66(b) muestra la declaración **COMPONENT** en VHDL para una primitiva de flip-flop JK. Los puntos clave a observar son el nombre del componente (JKFF) y los nombres de los puertos. Son los mismos nombres que los que se utilizan en el símbolo gráfico de la figura 5-66(a). Observe además que el *tipo* de cada una de las variables de entrada y salida es STD_LOGIC. Éste es uno de los tipos de datos del estándar IEEE definidos en la biblioteca, que muchos componentes de la misma utilizan.

VHDL



(a)

```
Declaración de un componente en VHDL

COMPONENT  JKFF
     PORT (j     : IN  STD_LOGIC;
           k     : IN  STD_LOGIC;
           clk   : IN  STD_LOGIC;
           clrn  : IN  STD_LOGIC;
           prn   : IN  STD_LOGIC;
           q     : OUT  STD_LOGIC);

END COMPONENT;
```

(b)

FIGURA 5-66 (a) Representación gráfica mediante el uso de un componente. (b) Declaración de un componente en VHDL.

La figura 5-67 utiliza un componente JKFF de la biblioteca en VHDL para crear un circuito equivalente al diseño gráfico de la figura 5-66(a). Las primeras dos líneas indican al compilador que debe usar la biblioteca IEEE para encontrar las definiciones de los tipos de datos std_logic. Las siguientes dos líneas indican al compilador que deberá buscar en la biblioteca Altera cualquier componente de biblioteca estándar que se utilice más adelante en el código. Las entradas y salidas del módulo se declaran como en los ejemplos anteriores, sólo que ahora el tipo es STD_LOGIC en vez de BIT. Esto se debe a que los tipos de los puertos de los módulos deben concordar con los tipos de los puertos de los componentes. Dentro de la sección de arquitectura se proporciona un nombre (ff1) a esta instancia del com-

```
LIBRARY   ieee;
USE         ieee.std_logic_1164.all;       -- define los tipos std_logic
LIBRARY     altera;
USE         altera.maxplus2.all;           -- provee los componentes estándar

ENTITY  fig5_67  IS
PORT(  clkent, jent, kent, preset, clear              :IN  std_logic;
         Qsal                                         :OUT  std_logic);
END  fig5_67;

ARCHITECTURE a  OF  fig5_67  IS
BEGIN
      ff1:  JKFF  PORT MAP  ( clk   =>    clkent,
                              j     =>    jent,
                              k     =>    kent,
                              prn   =>    preset,
                              clrn  =>    clear,
                              q     =>    qsal);
END a;
```

FIGURA 5-67 Un flip-flop JK mediante el uso de VHDL.

ponente JKFF. Las palabras clave **PORT MAP** van seguidas de una lista de todas las conexiones que deben hacerse con los puertos del componente. Observe que los puertos del componente (por ejemplo, clk) se listan a la izquierda del símbolo => y que los objetos a los que se conectan (por ejemplo, clkent) se listan a la derecha.

FLIP-FLOPS EN VHDL

Ahora que hemos visto cómo usar los componentes estándar disponibles en la biblioteca, veamos cómo crear nuestro propio componente que pueda usarse una y otra vez. Para simplificar la comparación describiremos el código VHDL para un flip-flop JK que es idéntico al componente de biblioteca JKFF.

VHDL se creó para ser un lenguaje muy flexible, ya que nos permite definir la operación de dispositivos sincronizados por reloj en forma explícita en el código, sin depender de las primitivas lógicas. La clave para los circuitos secuenciales disparados por flanco en VHDL es PROCESS. Como recordará, esta palabra clave va seguida de una lista de sensibilidad entre paréntesis. Cada vez que una de las variables de la lista de sensibilidad cambia de estado, el código en el bloque del proceso (PROCESS) determina cómo debe responder el circuito. Esto es muy parecido a un flip-flop que no hace nada sino hasta que la entrada de reloj cambie de estado, momento en el cual evalúa sus entradas y actualiza sus salidas. Si el flip-flop necesita responder a entradas distintas de la entrada de reloj (por ejemplo, PRESET y CLEAR), éstas pueden agregarse a la lista de sensibilidad. El código en la figura 5-68 demuestra un flip-flop JK escrito en VHDL.

En la línea 9 de la figura se declara una señal con el nombre de *qestado*. Las señales pueden considerarse como alambres que conectan dos puntos en la descripción del circuito, pero también tienen características de "memoria" implícita. Esto significa que una vez que se asigne un valor a la señal, permanecerá en ese valor hasta que se le asigne un valor distinto en el código. En VHDL se utiliza a menudo la palabra clave VARIABLE para implementar esta característica de "memoria",

```
1  -- Circuito de Flip-flop JK
2  ENTITY jk IS
3  PORT(
4        clk, j, k, prn, clrn :IN BIT;
5        q                    :OUT BIT);
6  END jk ;
7
8  ARCHITECTURE a OF jk IS
9  SIGNAL qestado  :BIT;
10 BEGIN
11    PROCESS(clk, prn, clrn)     -- responde a cualquiera de estas señales
12    BEGIN
13       IF prn = '0' THEN qestado <= '1'; -- PRESET asíncrono
14       ELSIF clrn = '0' THEN qestado <= '0';-- CLEAR asíncrono
15       ELSIF clk = '1' AND clk'EVENT THEN  -- en PGT del flanco del reloj
16          IF j = '1' AND k = '1' THEN qestado <= NOT qestado;
17          ELSIF j = '1' AND k = '0' THEN qestado <= '1';
18          ELSIF j = '0' AND k = '1' THEN qestado <= '0';
19          END IF;
20       END IF;
21    END PROCESS;
22    q <= qestado;          -- actualiza terminal de salida
23 END a;
```

FIGURA 5-68 Flip-flop JK individual mediante el uso de VHDL.

pero las variables deben declararse y utilizarse dentro del mismo bloque de descripción. En este ejemplo, si *qestado* se declarara como VARIABLE, tendría que declararse dentro de PROCESS (después de la línea 11) y debería asignarse a *q* antes del final de PROCESS (línea 21). Nuestro ejemplo utiliza una señal mediante la directiva SIGNAL, la cual puede declararse y usarse a lo largo de toda la descripción de la arquitectura.

Observe que la lista de sensibilidad de PROCESS contiene las señales asíncronas PRESET y CLEAR. El flip-flop debe responder a estas entradas tan pronto como se asignen (BAJO), y estas entradas deben predominar sobre las entradas *J*, *K* y *CLK*. Para lograr esto podemos usar la naturaleza secuencial de las instrucciones IF/ELSE. Primero la directiva PROCESS describirá lo que ocurre sólo cuando una de las tres señales (*clk*, *prn* o *clrn*) cambia de estado. En este ejemplo, la entrada de mayor prioridad es *prn* ya que se evalúa primero en la línea 13. Si se asigna, *qestado* cambiará a ALTO y todas las demás entradas ni siquiera se evaluarán, ya que se encuentran en la ramificación ELSE de la decisión. Si *prn* está en ALTO, *clrn* se evaluará en la línea 14 para ver si está en BAJO. De ser así, el flip-flop se borrará y no se evaluará nada más en el bloque PROCESS. La línea 15 se evaluará sólo si tanto *prn* como *clrn* están en ALTO. El término clk'**EVENT** en la línea 5 se evalúa como VERDADERO sólo si se ha producido una transición en *clk*. Como *clk* = '1' debe ser también VERDADERO, esta condición sólo responde a una transición del reloj en el flanco de subida. Las siguientes tres condiciones de las líneas 16, 17 y 18 se evalúan sólo después de un flanco de subida en *clk* y sirven para actualizar el estado del flip-flop. En otras palabras, están **anidadas** dentro de la instrucción ELSIF de la línea 15. Sólo se evalúan los comandos de las entradas JK para conmutación, establecer y restablecer mediante la instrucción IF/ELSIF de las líneas 16-18. Desde luego que en un JK hay un cuarto comando: retención. La condición ELSE "faltante" se interpretará en VHDL como un dispositivo de memoria implícito que retendrá el estado PRESENTE si ninguna de las condiciones del JK dadas es VERDADERA. Observe que cada estructura IF/ELSIF tiene su propia instrucción END IF. La línea 19 termina la estructura de decisión que se encarga de establecer, borrar o conmutar. La línea 20 termina la estructura IF/ELSIF que decide entre las respuestas de PRESET, CLEAR y flanco del reloj. Tan pronto como termina el bloque PROCESS, el estado del flip-flop se transfiere al puerto de salida *q*.

Sin importar si usted desarrolla su descripción en AHDL o en VHDL, puede verificar que el circuito opere en forma apropiada mediante el uso de un simulador. La parte más importante y retadora de la verificación por medio de un simulador es la creación de un conjunto de condiciones de entrada hipotéticas que demostrarán que el circuito hace todo lo que debe hacer. Hay muchas formas de hacer esto; es decisión del diseñador elegir la mejor. En la figura 5-69 se muestra la simulación que se utiliza para verificar la operación de la primitiva JKFF. La entrada *PRESET* se activa en un principio y después, en *t*1, se activa la entrada *CLEAR*. Estas pruebas aseguran que *PRESET* y *CLEAR* funcionen en forma asíncrona. La entrada *jent* está en ALTO en t2 y *kent* está en ALTO en t3. Entre estos puntos, las entradas *jent* y *kent* están en BAJO. Esta porción de la simulación prueba los modos síncronos de establecer, retención y PRESET. Empezando en t4, se prueba el comando de conmutación

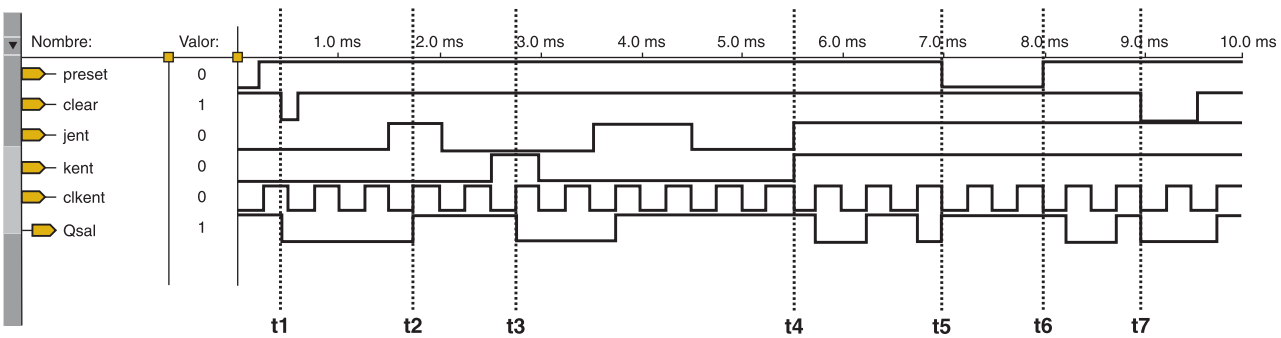


FIGURA 5-69 Simulación del flip-flop JK.

con *jent* = *kent* = 1. Observe que en t5 *PRESET* se asigna (BAJO) para evaluar si *PRESET* predomina sobre el comando de conmutación. Después de t6 la salida empieza a conmutar de nuevo, y en t7 se muestra cómo la entrada *CLEAR* predomina sobre las entradas síncronas. Es muy importante probar todos los modos de operación y la interacción de varios controles cuando se está simulando un circuito.

PREGUNTAS DE REPASO

1. ¿Qué es una primitiva lógica?
2. ¿Qué necesita saber el diseñador para poder utilizar una primitiva lógica?
3. En el sistema de Altera, ¿en dónde puede encontrar información acerca de las primitivas y funciones de biblioteca?
4. ¿Cuál es el elemento clave en VHDL que permite la descripción explícita de los circuitos lógicos sincronizados por reloj?
5. ¿Cuál es la biblioteca que define los tipos de datos std_logic?
6. ¿Cuál es la biblioteca que define las primitivas lógicas y los componentes comunes?

5-27 CIRCUITOS DE HDL CON VARIOS COMPONENTES

Empezamos este capítulo con el estudio de los latches. Éstos se utilizan para crear flip-flops y éstos, a su vez, para crear muchos circuitos, incluyendo los contadores binarios. En la figura 5-70 se muestra una descripción gráfica (diagrama lógico) de un simple contador binario ascendente. En cuanto a su funcionalidad, este circuito es el mismo que el de la figura 5-47, que se dibujó con el LSB a la derecha para facilitar la visualización del valor numérico del conteo binario. Aquí hemos vuelto a dibujar el circuito para mostrar el flujo de las señales en el formato más convencional, con entradas a la izquierda y salidas a la derecha. Observe que estos símbolos lógicos se disparan en los flancos negativos. Además estos flip-flops tampoco tienen las entradas asíncronas *prn* o *clrn*. Nuestro objetivo es describir este circuito contador mediante el uso de HDL; para ello interconectamos tres instancias del mismo componente flip-flop JK.

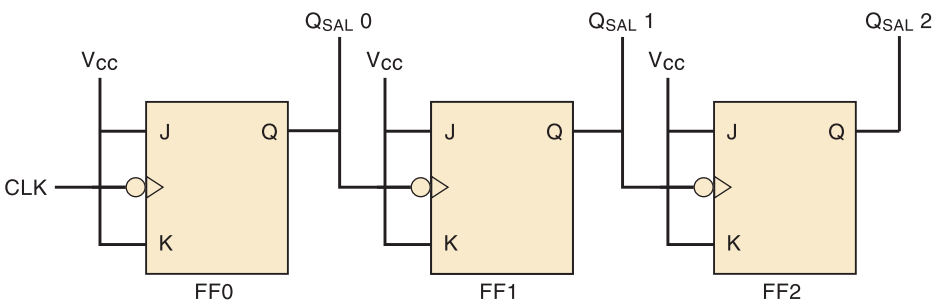


FIGURA 5-70
Contador binario de tres bits.

CONTADOR DE RIZO ASCENDENTE

Una descripción basada en texto de este circuito requiere tres flip-flops del mismo tipo, al igual que la descripción gráfica. Consulte la figura 5-71. En la línea 8 de esta figura se utiliza la notación en arreglo de bits para declarar un registro de tres flip-flops JK. El nombre del registro es *q*, al igual que el nombre del puerto de salida. AHDL puede interpretar que esto indica que la salida de cada flip-flop debe

AHDL

```
1   %  Contador de rizos MOD 8 ascendente %
2   SUBDESIGN fig5_71
3   (
4         reloj                 :INPUT;
5         q[2..0]               :OUTPUT;
6   )
7   VARIABLE
8         q[2..0] :JKFF;        -- define tres FFs JK
9   BEGIN
10                              -- nota: valor determinado de prn, clrn es vcc
11        q[2..0].j = VCC;      -- modo de conmutación J=K=1 para todos los FFs
12        q[2..0].k = VCC;
13        q[0].clk = !reloj;
14        q[1].clk = !q[0].q;
15        q[2].clk = !q[1].q;   -- conecta los relojes en forma de rizo
16  END;
```

FIGURA 5-71 Contador de rizos MOD-8 en AHDL.

conectarse al puerto de salida. Cada bit del arreglo *q* tiene todos los atributos de una primitiva JKFF. AHDL es muy flexible en cuanto al uso de los conjuntos indizados como éste. Como ejemplo del uso de esta notación de conjunto, observe cómo las entradas J y K para todos los flip-flops se enlazan a VCC en las líneas 11 y 12. Si los flip-flops se hubieran llamado A, B y C en lugar de usar un arreglo de bits, sería necesario utilizar asignaciones individuales para cada entrada J y K, con lo cual el código sería más extenso. A continuación se hacen las interconexiones clave entre los flip-flops para crear el contador de rizo ascendente. La señal de reloj se invierte y se asigna a la entrada de reloj de FF0 (línea 13), la salida Q de FF0 se invierte y se asigna a la entrada de reloj de FF1 (línea 14) y así en lo sucesivo, para formar un contador de rizo.

VHDL

CONTADOR DE RIZO ASCENDENTE EN VHDL

En la figura 5-68 describimos el código de VHDL para un JKFF disparado con flanco positivo, con los controles PRESET y CLEAR. El contador de la figura 5-70 se dispara con los flancos negativos y no requiere de los controles asíncronos PRESET o CLEAR. Nuestro objetivo ahora es escribir el código de VHDL para uno de estos flip-flops, representar tres instancias del mismo flip-flop e interconectar los puertos para crear el contador.

Primero vamos a analizar la descripción de VHDL en la figura 5-72, comenzando con la línea 18. Este módulo de VHDL está describiendo la operación de un componente flip-flop JK individual. El nombre del componente es jk_neg (línea 18); tiene las entradas *clk*, *j*, *k* (línea 19) y la salida *q* (línea 20). Una señal llamada *qestado* se utiliza para retener el estado del flip-flop y conectarlo a la salida *q*. En la línea 25 el bloque PROCESS sólo tiene a *clk* en su lista de sensibilidad, por lo que sólo responde a los cambios en *clk* (PGTs y NGTs). La instrucción que hace que este flip-flop se dispare con los flancos negativos se encuentra en la línea 27. Si la instrucción IF (*clk*'EVENT AND *clk* = '0') es verdadera, entonces se acaba de producir un flanco en *clk* y ahora *clk* está en BAJO, lo cual significa que debió haber sido una NGT de *clk*. Las decisiones IF/ELSE que van después implementan los cuatro estados de un flip-flop JK.

```
1   ENTITY fig5_72 IS
2   PORT (   reloj       :IN  BIT;
3            qsal        :BUFFER BIT_VECTOR (2 DOWNTO 0));
4   END fig5_72;
5   ARCHITECTURE contador of fig5_72 IS
6      SIGNAL alto       :BIT;
7      COMPONENT jk_neg
8      PORT ( clk, j, k  :IN  BIT;
9             q          :OUT  BIT);
10     END COMPONENT;
11  BEGIN
12     alto <= '1'        -- conectado a Vcc
13  ff0:  jk_neg    PORT MAP (j => alto, k => alto, clk => reloj, q => qsal(0));
14  ff1:  jk_neg    PORT MAP (j => alto, k => alto, clk => qsal(0), q => qsal(1));
15  ff2:  jk_neg    PORT MAP (j => alto, k => alto, clk => qsal(1), q => qsal(2));
16  END contador;
17
18  ENTITY jk_neg  IS
19  PORT (   clk, j, k   :IN  BIT;
20           q           :OUT  BIT);
21  END  jk_neg;
22  ARCHITECTURE simple OF  jk_neg  IS
23        SIGNAL   qestado      :BIT;
24  BEGIN
25        PROCESS (clk)
26        BEGIN
27      IF (clk'EVENT AND clk = '0') THEN
28         IF j = '1' AND k = '1' THEN qestado <= NOT qestado;   -- conmutación
29         ELSIF j = '1' AND k = '0'  THEN  qestado <= '1';      -- establecer
30         ELSIF j = '0' AND k = '1'  THEN qestado <= '0';        -- restablecer
31         END IF;
32     END IF;
33      END PROCESS;
34      q <= qestado            -- conecta el estado del flip-flop a la salida
35  END simple;;
```

FIGURA 5-72 Contador de rizos MOD-8 en VHDL.

Ahora que sabemos cómo funciona un flip-flop llamado jk_neg, veamos cómo podemos usarlo tres veces en un circuito y enlazar todos los puertos. La línea 1 define la entidad (ENTITY) que constituirá el contador de tres bits. Las líneas 2-3 contienen las definiciones de las entradas y las salidas. Observe que las salidas se encuentran en la forma de un arreglo de tres bits (vector de bits). En la línea 6 la señal (SIGNAL) *alto* puede considerarse como un alambre que se utiliza para conectar los puntos en el circuito con V_{CC}. La línea 7 es muy importante, ya que aquí es donde declaramos que planeamos utilizar un componente en nuestro diseño, cuyo nombre es jk_neg. En este ejemplo el verdadero código está escrito en la parte inferior de la página, pero podría estar en un archivo separado o incluso en una biblioteca. Esta declaración indica al compilador todos los datos importantes sobre el componente y los nombres de sus puertos.

La parte final de la descripción es la sección concurrente de las líneas 12-15. Primero la señal *alto* se conecta a V_{CC} en la línea 12. En las siguientes tres líneas se crean las instancias de los componentes tipo flip-flop. Las tres instancias se llaman ff0, ff1 y ff2. Cada instancia va seguida de una instrucción PORT MAP, en la cual se lista cada puerto del componente y se describe con qué está conectado en el módulo.

No es difícil conectar componentes entre sí mediante el uso de HDL, pero es muy tedioso. Como podrá ver, hasta el archivo para un circuito muy simple puede ser bastante extenso. A este método de describir circuitos se le conoce como **nivel de abstracción estructural**. Requiere que el diseñador tome en cuenta cada una de las terminales de cada componente y que defina señales para cada alambre que se utilice para interconectar los componentes. Por lo general, las personas acostumbradas a utilizar diagramas lógicos para describir circuitos encuentran que es fácil comprender el nivel estructural, pero no es tan fácil de leer de un vistazo, en comparación con el diagrama del circuito lógico equivalente. De hecho, podemos decir que si el nivel estructural de descripción fuera lo único que estuviera disponible, la mayoría de las personas preferiría utilizar descripciones gráficas (esquemáticos) en vez de HDL. La verdadera ventaja del HDL está en el uso de niveles más altos de abstracción y la habilidad de manipular los componentes para que se ajusten a las necesidades exactas del proyecto. En los siguientes capítulos exploraremos el uso de estos métodos, así como las herramientas gráficas para conectar los módulos.

PREGUNTAS DE REPASO

1. ¿Puede el mismo componente utilizarse más de una vez en el mismo circuito?
2. En AHDL, ¿dónde se declaran las instancias múltiples de un componente?
3. ¿Cómo podemos diferenciar entre múltiples instancias de un componente?
4. En AHDL, ¿qué operador se utiliza para "conectar" señales?
5. En VHDL, ¿qué sirve como "alambres" para conectar componentes?
6. En VHDL, ¿qué palabra clave identifica la sección de código en donde se especifican las conexiones para las instancias de los componentes?

RESUMEN

1. Un flip-flop es un circuito lógico con una característica de memoria tal que sus salidas Q y $\overline{Q}$ cambiarán a un nuevo estado en respuesta a un pulso de entrada y permanecerán en ese nuevo estado aún después de que termine el pulso de entrada.
2. Un latch NAND y un latch NOR son simples FFs que responden a los niveles lógicos en sus entradas SET y RESET.
3. Borrar (CLEAR, RESET) un FF significa que su salida termina en el estado $Q = 0$ /$\overline{Q} = 1$. Establecer (SET) un FF significa que termina en el estado $Q = 1/\overline{Q} = 0$.
4. Los FFs sincronizados por reloj tienen una entrada de reloj (*CLK*, *CP*, *CK*) que se dispara por flanco, lo cual significa que dispara el FF en una transición de pendiente positiva (PGT) o en una transición de pendiente negativa (NGT).
5. Los FFs disparados por flanco (sincronizados) pueden dispararse a un nuevo estado mediante el flanco activo de la entrada de reloj, de acuerdo con el estado de las entradas de control síncronas del FF (S, R o J, K o D).
6. La mayoría de los FFs sincronizados por reloj tienen entradas asíncronas que pueden establecer o borrar el FF, sin importar el estado de la entrada de reloj.
7. El latch D es un latch NAND modificado que opera como un flip-flop D, con la excepción de que no se dispara por flanco.
8. Algunos de los principales usos de los FFs son: almacenamiento y transferencia de datos, desplazamiento de datos, conteo y división de frecuencia. Se utilizan en circuitos secuenciales que siguen una secuencia predeterminada de estados.

9. Un monoestable (one-shot, OS) es un circuito lógico que puede dispararse desde su estado inactivo normal ($Q = 0$) a su estado activo ($Q = 1$), en donde permanecerá durante un intervalo de tiempo proporcional a una constante de tiempo *RC*.
10. Los circuitos que tienen una entrada tipo disparador de Schmitt responden en forma confiable a las señales con cambios lentos y producen salidas con flancos limpios y bien definidos.
11. Puede usarse una variedad de circuitos para generar señales de reloj a una frecuencia deseada, incluyendo los osciladores disparadores de Schmitt, un temporizador 555 y un oscilador controlado por cristal.
12. En la contraportada interior encontrará un resumen completo de los diversos tipos de FFs.
13. Los dispositivos lógicos programables pueden programarse para operar como circuitos tipo latch y circuitos secuenciales.
14. Los bloques de construcción fundamentales llamados primitivas lógicas están disponibles en la biblioteca de Altera para ayudar a implementar sistemas más grandes.
15. Los flip-flops sincronizados por reloj están disponibles como primitivas lógicas.
16. Puede escribirse código en VHDL para describir la lógica sincronizada por reloj de manera explícita, sin necesidad de usar primitivas lógicas.
17. VHDL permite utilizar archivos de HDL como componentes en sistemas más grandes. Hay componentes prefabricados en la biblioteca de Altera.
18. Podemos usar HDL para describir componentes interconectados en forma muy parecida a una herramienta gráfica de captura de diagramas esquemáticos.

TÉRMINOS IMPORTANTES

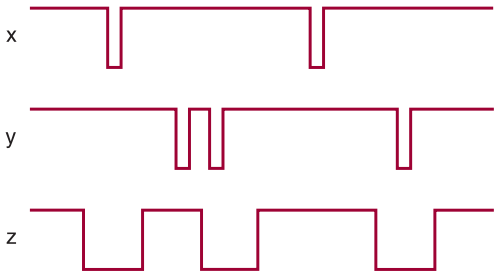
anidadas
bloque de control común
circuito de conducción de pulso
circuito detector de flancos
circuito disparador de Schmitt
circuitos secuenciales
CLEAR (estados/entradas)
COMPONENT
contador binario
desfasamiento de reloj
diagrama de transición de estados
disparo
disparada por flanco
división de frecuencia
entradas asíncronas
entradas de control
entradas de control síncronas
entradas predominantes
estado casi estable
EVENT
flip-flop
flip-flop sincronizado por reloj
flip-flop sincronizado por reloj en D
flip-flop sincronizado por reloj en J-K
flip-flop S-R sincronizado por reloj
frecuencia
generador de reloj controlado por cristal
ignorar entradas
latch *D*
latch de compuerta NAND
latch de compuerta NOR
modo de conmutación
monoestable (one-shot - OS)
multivibrador astable o de funcionamiento libre
nivel estructural de abstracción
número MOD
OS no redisparable
OS redisparable
periodo
PORT MAP
primitiva lógica
pulsos
rebote de contacto
registro de desplazamiento
registros
reloj
retrasos por tiempos de propagación
RESET (estados/entradas)
retraso de propagación
SET (estados/entradas)
tabla de estados
temporizador 555
tiempo de estabilización, t_S
tiempo de retención, t_H
transferencia asíncrona (atascada)
transferencia atascada
transferencia de datos
transferencia de datos en paralelo
transferencia de datos en serie
transferencia síncrona
transición de pendiente negativa (NGT)
transición de pendiente positiva (PGT)

PROBLEMAS

SECCIONES 5-1 A 5-3

B 5-1.* Suponiendo que al principio $Q = 0$, aplique las formas de onda x y y de la figura 5-73 a las entradas SET y RESET de un latch NAND y determine las formas de onda de Q y $\overline{Q}$.

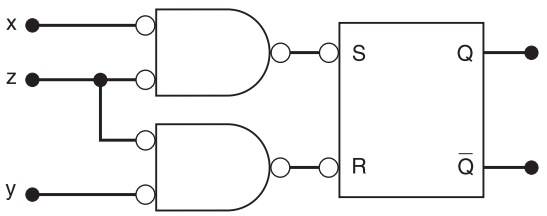
FIGURA 5-73 Problemas 5-1 a 5-3.



B 5-2. Invierta las formas de onda x y y de la figura 5-73, aplíquelas a las entradas SET y RESET de un latch NOR y determine las formas de onda de Q y $\overline{Q}$. Suponga que al principio $Q = 0$.

5-3.* Las formas de onda de la figura 5-73 están conectadas al circuito de la figura 5-74. Suponga que al principio $Q = 0$ y determine la forma de onda de Q.

FIGURA 5-74 Problema 5-3.

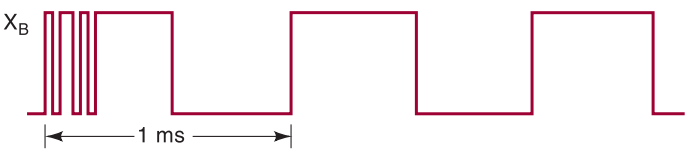


D 5-4. Modifique el circuito de la figura 5-9 para que utilice un latch de compuerta NOR.

D 5-5. Modifique el circuito de la figura 5-12 para que utilice un latch de compuerta NAND.

F 5-6.* Consulte el circuito de la figura 5-13. Para probar la operación del circuito, un técnico observa las salidas con un osciloscopio de almacenamiento mientras que el interruptor se mueve de A a B. Cuando el interruptor se mueve de A a B, aparece la pantalla de X_B en el osciloscopio según se muestra en la figura 5-75. ¿Qué falla del circuito podría producir este resultado? (*Sugerencia:* ¿cuál es la función del latch NAND?)

FIGURA 5-75 Problema 5-6.



SECCIONES 5-4 A 5-6

B 5-7. Cierto FF sincronizado por reloj tiene los valores mínimos $t_S = 20$ ns y $t_H = 5$ ns. ¿Cuánto tiempo deben permanecer estables las entradas de control antes de la transición activa del reloj?

---

* Encontrará las respuestas a los problemas marcados con un asterisco al final del capítulo.

B 5-8. Aplique las formas de onda de *S*, *R* y *CLK* de la figura 5-19 al FF de la figura 5-20 y determine la forma de onda de *Q*.

B 5-9.* Aplique las formas de onda de la figura 5-76 al FF de la figura 5-19 y determine la forma de onda en *Q*. Repita el proceso para el FF de la figura 5-20. Suponga que al principio $Q = 0$.

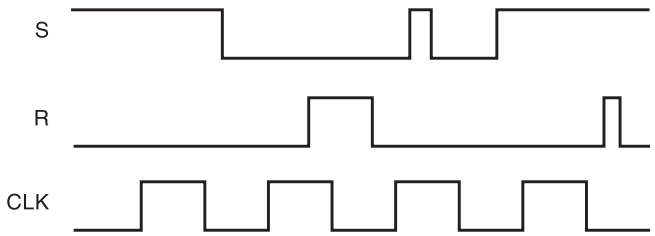


FIGURA 5-76
Problema 5-9.

5-10. Dibuje las siguientes formas de onda de pulsos digitales. Etiquételas como t_r, t_f y t_w, franco anterior y franco posterior.

(a) Un pulso TTL negativo con $t_r = 20$ ns, $t_f = 5$ ns y $t_w = 50$ ns.

(b) Un pulso TTL positivo con $t_r = 5$ ns, $t_f = 1$ ns y $t_w = 25$ ns.

(c) Un pulso positivo con $t_w = 1$ ms, cuyo flanco anterior ocurra cada 5 ms. Proporcione la frecuencia de esta forma de onda.

SECCIÓN 5-7

B 5-11.* Aplique las formas de onda de *J*, *K* y *CLK* de la figura 5-23 al FF de la figura 5-24. Suponga que al principio $Q = 1$ y determine la forma de onda de *Q*.

D 5-12. (a)* Muestre cómo un flip-flop J-K puede operar como un FF *conmutador* (que cambia de estado en cada pulso del reloj). Después aplique una señal de reloj de 10-kHz en su entrada *CLK* y determine la forma de onda en *Q*.

(b) Conecte la salida *Q* de este FF a la entrada *CLK* de un segundo FF J-K que también tiene $J = K = 1$. Determine la frecuencia de la señal en la salida de este FF.

B 5-13. Las formas de onda que se muestran en la figura 5-77 se van a aplicar a dos FFs distintos:

(a) J-K disparado con flanco positivo.

(b) J-K disparado con flanco negativo.

Dibuje la respuesta de la forma de onda de *Q* para cada uno de estos FFs, suponiendo que al principio $Q = 0$. Suponga también que cada FF tiene $t_H = 0$.

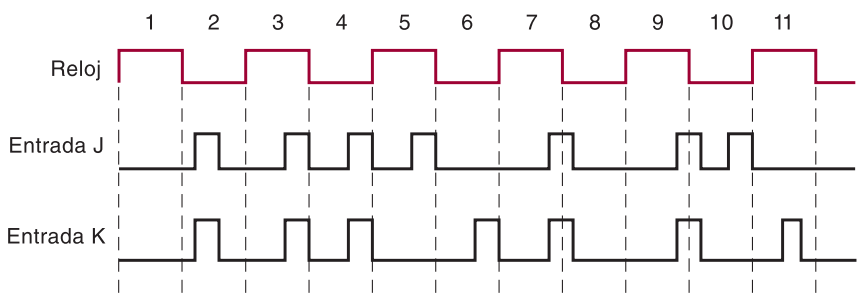


FIGURA 5-77
Problema 5-13.

SECCIÓN 5-8

N 5-14. Algunas veces se utiliza un FF D para *retrasar* una forma de onda binaria, de manera que la información binaria aparezca en la salida durante una cierta cantidad de tiempo después de que aparece en la entrada *D*.

(a)* Determine la forma de onda de *Q* en la figura 5-78 y compárela con la forma de onda de entrada. Observe que está retrasada desde la entrada por un periodo de reloj.

(b) ¿Cómo puede obtenerse un retraso de dos periodos de reloj?

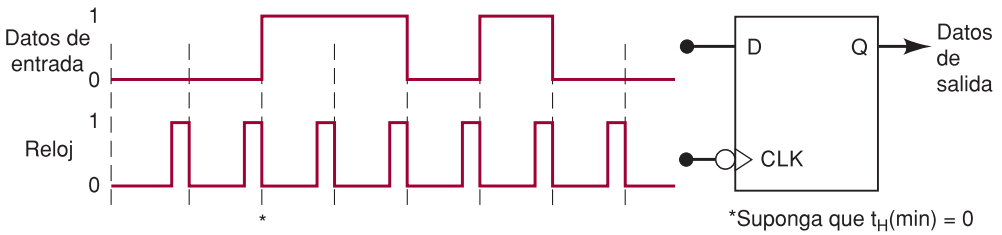


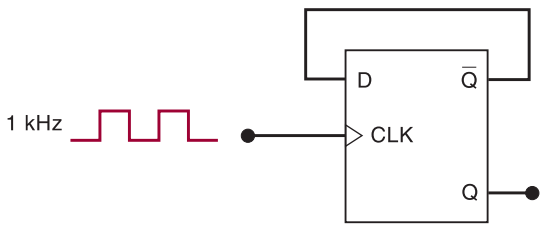
FIGURA 5-78 Problema 5-14.

B 5-15. (a) Aplique las formas de onda de *S* y *CLK* de la figura 5-76 a las entradas *D* y *CLK* de un FF D que se dispara en las PGTs. Después determine la forma de onda en *Q*.

(b) Repita el procedimiento, utilizando la forma de onda de *C* de la figura 5-76 para la entrada *D*.

B 5-16.* Se puede hacer que un flip-flop D disparado por flanco opere en el modo de conmutación si se le conecta como se muestra en la figura 5-79. Suponga que al principio $Q = 0$ y determine la forma de onda de *Q*.

FIGURA 5-79 Flip-flop D conectado para conmutar (problema 5-16).



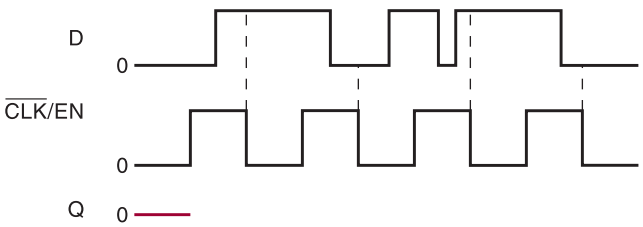
SECCIÓN 5-9

B 5-17. (a) Aplique las formas de onda de *S* y *CLK* de la figura 5-76 a las entradas *D* y *EN* de un latch D en forma respectiva y determine la forma de onda en *Q*.

(b) Repita el proceso utilizando la forma de onda de *C* aplicada a *D*.

5-18. Compare la operación del latch *D* con un flip-flop *D* disparado por flanco negativo, mediante la aplicación de las formas de onda de la figura 5-80 a cada uno de ellos y determine la forma de onda de *Q*.

FIGURA 5-80 Problema 5-18.

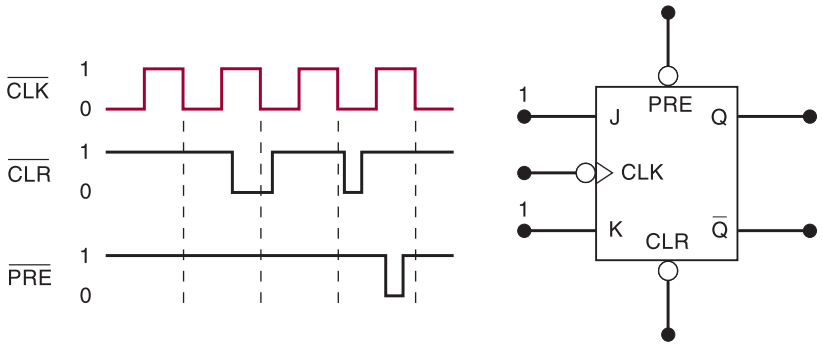


5-19. En el problema 5-16 vimos cómo un flip-flop D disparado por flanco puede operar en el modo de conmutación. Explique por qué la misma idea no funcionará para un latch *D*.

SECCIÓN 5-10

B 5-20. Determine la forma de onda de *Q* para el FF de la figura 5-81. Suponga que al principio $Q = 0$ y recuerde que las entradas asíncronas predominan sobre todas las demás entradas.

FIGURA 5-81
Problema 5-20.



B 5-21.* Aplique las formas de onda de $\overline{CLK}$, $\overline{PRE}$ y $\overline{CLR}$ (figura 5-32) a un flip-flop D disparado por flanco positivo con entradas asíncronas activas en BAJO. Suponga que *D* se mantiene en ALTO y que al principio *Q* está en BAJO. Determine la forma de onda de *Q*.

B 5-22. Aplique las formas de onda de la figura 5-81 a un flip-flop D que se dispara con las NGTs y que tiene entradas asíncronas activas en BAJO. Suponga que *D* se mantiene en BAJO y que al principio *Q* está en ALTO. Dibuje la forma de onda resultante en *Q*.

SECCIÓN 5-12

B 5-23. Use la tabla 5-2 de la sección 5-12 para determinar lo siguiente.

(a)* ¿Cuánto puede tardar la salida *Q* de un 74C74 en cambiar de 0 a 1, en respuesta a una transición activa de *CLK*?

(b)* ¿Cuál FF en la tabla 5-2 requiere que sus entradas de control permanezcan estables durante el mayor tiempo *después* de la transición activa en *CLK*? ¿*Antes* de la transición?

(c) ¿Cuál es el pulso más estrecho que puede aplicarse a la entrada $\overline{PRE}$ de un FF 7474?

B 5-24. Use la tabla 5-2 para determinar lo siguiente:

(a) ¿Cuánto tiempo se requiere para borrar un 74LS112 en forma asíncrona?

(b) ¿Cuánto tiempo se requiere para establecer un 74HC112 en forma asíncrona?

(c) ¿Cuál es el intervalo más corto aceptable entre las transiciones activas de reloj para un 7474?

(d) La entrada D de un 74HC112 cambia a ALTO 15 ns antes del flanco activo del reloj. ¿Se guardarán los datos en forma confiable en el flip-flop?

(e) ¿Cuánto tiempo se requiere (después del flanco del reloj) para almacenar en forma síncrona un 1 en un flip-flop D 7474 borrado?

SECCIONES 5-15 Y 5-16

D 5-25.* Modifique el circuito de la figura 5-40 para que utilice un flip-flop J-K.

D 5-26. En el circuito de la figura 5-82, al principio las entradas *A*, *B* y *C* están en BAJO. Se supone que la salida *Y* debe cambiar a ALTO sólo cuando *A*, *B* y *C* cambien a ALTO en cierta secuencia.

(a) Determine la secuencia que hará que *Y* cambie a ALTO.

(b) Explique por qué se necesita el pulso INICIO.

(c) Modifique este circuito para que utilice FFs D.

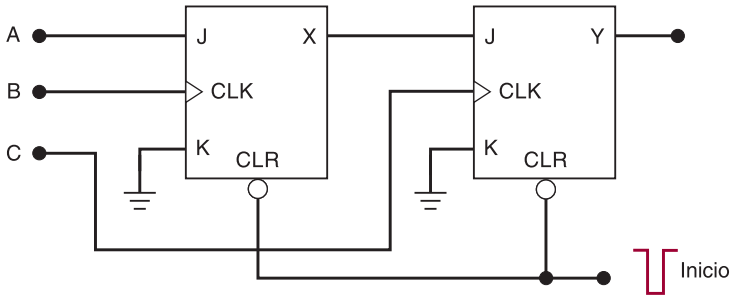


FIGURA 5-82
Problema 5-26.

SECCIONES 5-17 Y 5-18

D 5-27.* (a) Dibuje un diagrama de circuito para la transferencia síncrona en paralelo de un registro de tres bits a otro registro, mediante el uso de flip-flops J-K.

(b) Repita el proceso para una transferencia asíncrona en paralelo.

N, D 5-28. Un registro de desplazamiento *recirculante* es un registro de desplazamiento que mantiene circulando la información binaria a través del registro, a medida que se aplican pulsos de reloj. El registro de desplazamiento de la figura 5-45 puede convertirse en un registro circulante si se conecta X_0 a la línea DATOS ENT. No se utilizan entradas externas. Suponga que este registro circulante empieza con el valor 1011 almacenado en el mismo (es decir, $X_3 = 1$, $X_2 = 0$, $X_1 = 1$ y $X_0 = 1$). Liste la secuencia de estados por los que pasan los FFs del registro a medida que se aplican ocho pulsos de desplazamiento.

D 5-29.* Consulte la figura 5-46, en donde un número de tres bits almacenado en el registro *X* se está desplazando en forma serial hacia el registro *Y*. ¿Cómo puede modificarse el circuito de manera que al final de la operación de transferencia esté el número original almacenado en *X* presente en ambos registros? (*Sugerencia:* vea el problema 5-28.)

SECCIÓN 5-19

B 5-30. Consulte el circuito contador de la figura 5-47 y responda a lo siguiente:

(a)* Si el contador empieza en 000, ¿cuál será la cuenta después de 13 pulsos de reloj? ¿Después de 99 pulsos? ¿Después de 256 pulsos?

(b) Si el contador empieza en 100, ¿cuál será la cuenta después de 13 pulsos? ¿Después de 99 pulsos? ¿Después de 256 pulsos?

(c) Conecte un cuarto FF J-K (X_3) a este contador y dibuje el diagrama de transición de estados para este contador de 4 bits. Si la frecuencia del reloj de entrada es de 80 MHz, ¿cuál será la apariencia de la forma de onda en X_3?

B 5-31. Consulte el contador binario de la figura 5-47. Cámbielo conectando $\overline{X}_0$ a la señal *CLK* del flip-flop X_1, y $\overline{X}_1$ a la señal *CLK* del flip-flop X_2. Empiece con todos los FFs en el estado 1 y dibuje las diversas formas de onda de salida de los FFs (X_0, X_1, X_2) para 16 pulsos de entrada. Después liste la secuencia de estados de los FFs, como se hizo en la figura 5-48. A este contador se le conoce como *contador descendente*. ¿Por qué?

B 5-32. Dibuje el diagrama de transición de estados para este contador descendente y compárelo con el diagrama de la figura 5-49. ¿Cuál es la diferencia?

B 5-33.* (a) ¿Cuántos FFs se requieren para construir un contador binario que cuente desde 0 hasta 1023?

(b) Determine la frecuencia en la salida del último FF de este contador, para una frecuencia de entrada de reloj de 2 MHz.

(c) ¿Cuál es el número MOD del contador?

(d) Si al principio el contador está en cero, ¿qué cuenta tendrá después de 2060 pulsos?

B 5-34. Un contador binario está recibiendo pulsos a través de una señal de reloj de 256-kHz. La frecuencia de salida del último FF es de 2 kHz.

(a) Determine el número MOD.

(b) Determine el intervalo de conteo.

B 5-35. Un circuito fotodetector se está usando para generar un pulso cada vez que un cliente entra en cierto establecimiento. Los pulsos se alimentan a un contador de ocho bits. Este contador se utiliza para contar estos pulsos como un medio para determinar cuántos clientes han entrado en la tienda. Después de cerrar la tienda, el propietario revisa el contador y descubre que muestra un conteo de $00001001_2 = 9_{10}$. Él sabe que esto es incorrecto, ya que había mucho más de nueve personas en su tienda. Suponiendo que el circuito contador esté funcionando en forma apropiada, ¿cuál podría ser el motivo de la discrepancia?

SECCIÓN 5-20

D 5-36.* Modifique el circuito de la figura 5-50 de manera que sólo la presencia del código de dirección 10110110 permita transferir datos al registro *X*.

F 5-37. Suponga que el circuito de la figura 5-50 está funcionando mal, de manera que se transfieran datos a *X* para cualquiera de los códigos de dirección 11111110 o 11111111. ¿Cuáles son algunas de las fallas de un circuito que podrían estar ocasionando esto?

N, D 5-38. Muchos microcontroladores comparten las mismas terminales para enviar de salida la dirección inferior y transferir datos. Para poder mantener la constante de dirección mientras que se transfieren los datos. La información sobre la dirección se almacena en un latch, el cual se habilita mediante la señal de control ALE (habilita latch de dirección), como se muestra en la figura 5-83. Conecte este latch al microcontrolador de tal forma que reciba lo que haya en la dirección inferior y en las líneas de datos mientras que ALE esté en ALTO, y que lo retenga en las líneas de sólo dirección inferior cuando ALE esté en BAJO.

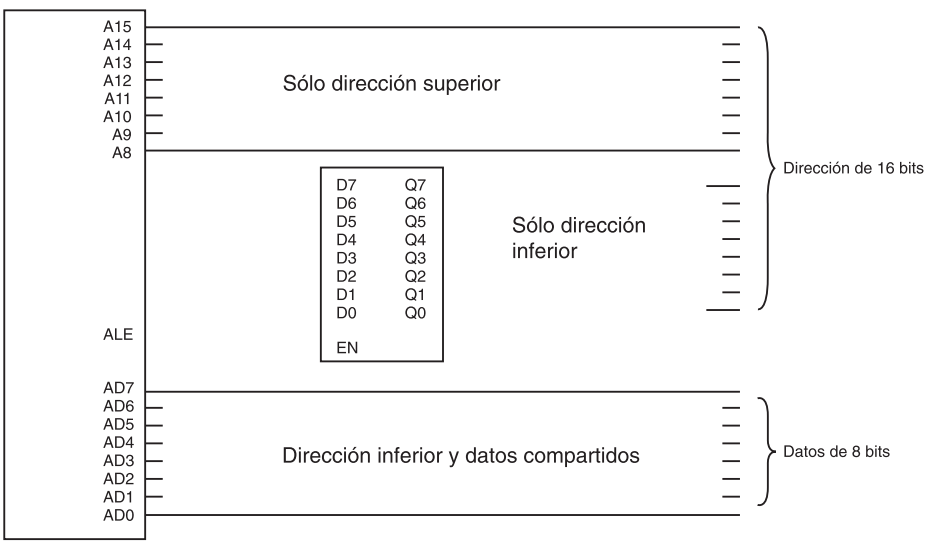


FIGURA 5-83
Problema 5-38.

D 5-39. Modifique el circuito de la figura 5-50, de forma que la MPU tenga ocho líneas de salida de datos conectadas para transferir ocho bits de datos a un registro de ocho bits formado por dos CIs 74HC175 [Figura 5-34(b)]. Muestre todas las conexiones del circuito.

SECCIÓN 5-22

B 5-40. Consulte las formas de onda de la figura 5-53(a). Cambie la duración del pulso del OS a 0.5 ms y determine la salida Q para ambos tipos de OS. Después repita el proceso utilizando una duración de pulso de OS de 1.5 ms.

N 5-41.* La figura 5-84 muestra tres monoestables no redisparables conectados en una cadena de sincronización que produce tres pulsos de salida secuenciales. Observe el "1" enfrente del pulso en cada símbolo de OS para indicar la operación no redisparable. Dibuje un diagrama de tiempo que muestre la relación entre el pulso de entrada y las salidas de los tres OS. Suponga una duración de 10 ms para el pulso de entrada.

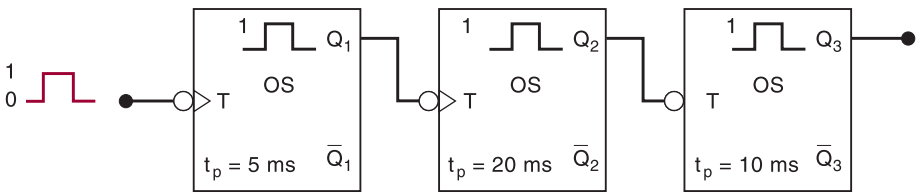


FIGURA 5-84
Problema 5-41.

5-42. Un OS *redisparable* puede utilizarse como detector de frecuencia de pulso que detecte cuando la frecuencia de un pulso de entrada esté por debajo de un valor predeterminado. En la figura 5-85 se muestra un ejemplo simple de esta aplicación. La operación empieza con el cierre momentáneo del interruptor SW1.

(a) Describa cómo responde el circuito a las frecuencias de entrada por encima de 1 kHz.

(b) Describa cómo responde el circuito a las frecuencias de entrada por debajo de 1 kHz.

(c) ¿Cómo modificaría el circuito para detectar cuando la frecuencia de entrada cae por debajo de 50 kHz?

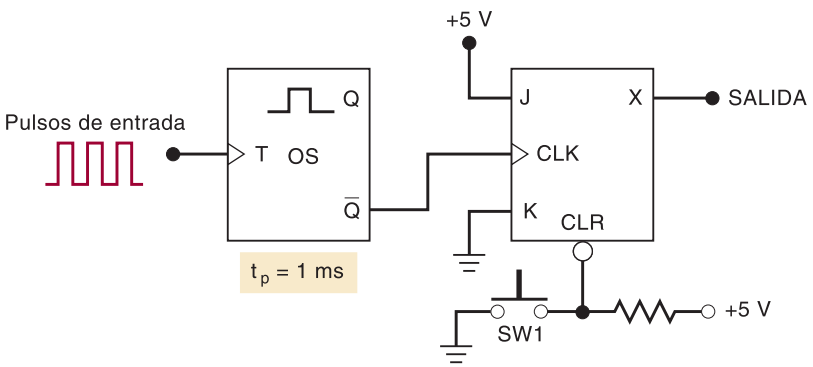


FIGURA 5-85
Problema 5-42.

5-43. Consulte el símbolo lógico para un monoestable no redisparable 74121 en la figura 5-54(a).

(a)* ¿Qué condiciones de entrada se necesitan para que el OS se dispare mediante una señal en la entrada B?

(b) ¿Qué condiciones de entrada se necesitan para que el OS se dispare mediante una señal en la entrada A_1?

A, D 5-44. La anchura del pulso de salida de un OS 74121 se obtiene mediante la fórmula aproximada

$$t_p \approx 0.7\, R_T C_T$$

en donde R_T es la resistencia conectada entre la terminal R_{EXT}/C_{EXT} y V_{CC}, y C_T es la capacitancia conectada entre la terminal C_{EXT} y la terminal R_{EXT}/C_{EXT}. El valor de R_T puede variar entre 2 y 40 kΩ, y C_T puede ser de hasta 1000 μF.

(a) Muestre cómo puede conectarse un 74121 para producir un pulso de pendiente negativa con una duración de 5 ms, cada vez que cualquiera de dos señales lógicas (E o F) produce una NGT. Tanto E como F se encuentran, por lo general, en el estado ALTO.

(b) Modifique el circuito de manera que una señal de entrada de control (G) pueda deshabilitar el pulso de salida del OS, sin importar lo que ocurra en E o en F.

SECCIÓN 5-23

B, N 5-45.* Muestre cómo usar un INVERSOR disparador de Schmitt 74LS14 para producir una onda cuadrada aproximada con una frecuencia de 10 kHz.

B, D 5-46. Diseñe un oscilador 555 de funcionamiento libre para producir una onda cuadrada aproximada a 40 kHz. C deberá mantenerse en 500 pF o más.

D 5-47. Un oscilador 555 puede combinarse con un flip-flop J-K para producir una onda cuadrada perfecta (ciclo de trabajo del 50 por ciento). Modifique el circuito del problema 5-46 para que incluya un flip-flop J-K. La salida final tiene que seguir siendo una onda cuadrada de 40 kHz.

5-48. Diseñe un circuito temporizador 555 que produzca una forma de onda de 5 kHz con un ciclo de trabajo del 10 por ciento. Seleccione un capacitor mayor de 500 pF y resistencias menores de 100 kΩ. Dibuje el diagrama del circuito con los números de las terminales etiquetados.

A, N 5-49. El circuito de la figura 5-86 puede usarse para generar dos señales de reloj que no se traslapen entre sí, a la misma frecuencia. Estas señales de reloj se utilizaban en los primeros sistemas de microprocesador, los cuales requerían cuatro transiciones de reloj distintas para sincronizar sus operaciones.

(a) Dibuje las formas de onda de sincronización CP1 y CP2 si *RELOJ* es una onda cuadrada de 1 MHz. Suponga que t_{PLH} y t_{PHL} son de 20 ns para el FF y de 10 ns para las compuertas AND.

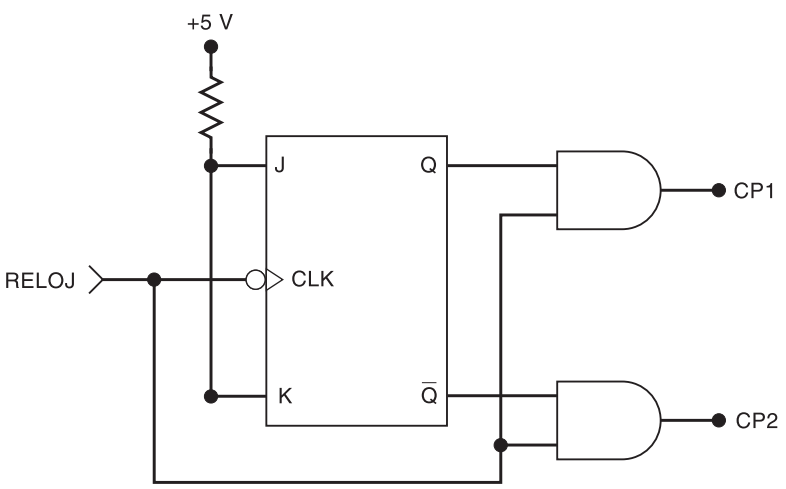


FIGURA 5-86
Problema 5-49.

(b) Este circuito tendría un problema si el FF se cambiara por uno que respondiera a una PGT en *CLK*. Dibuje las formas de onda de CP1 y CP2 para esa situación. Ponga especial atención a las condiciones que pueden producir deformaciones.

SECCIÓN 5-24

F 5-50. Consulte el circuito contador de la figura 5-47. Suponga que todas las entradas asíncronas se conectan a V_{CC}. Al probarlo, las formas de onda del circuito aparecen como se muestra en la figura 5-87. Considere la siguiente lista de posibles fallas. Para cada una, indique "sí" o "no" en cuanto a si pueden producir los resultados observados. Explique cada respuesta.

(a)* La entrada *CLK* de X_2 está abierta.

(b)* Los tiempos de transición de la salida X_1 son demasiado largos, tal vez debido a la carga.

(c) La salida de X_2 está en corto con tierra.

(d) No se cumple con el requerimiento de tiempo de retención de X_2.

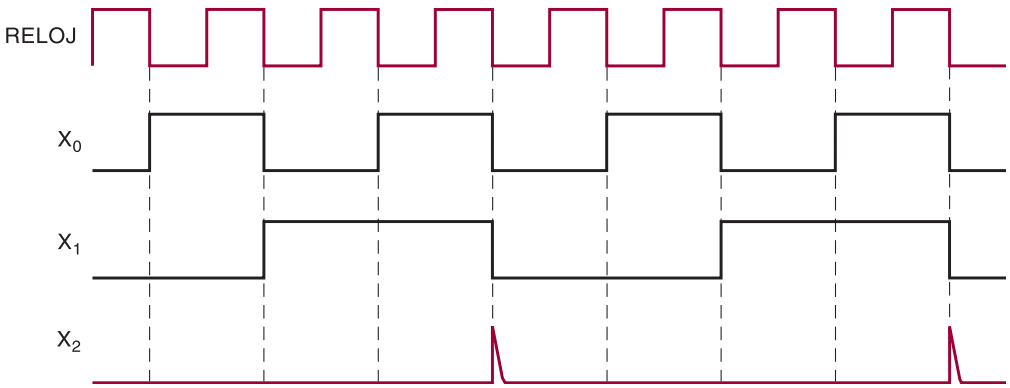


FIGURA 5-87 Problema 5-50.

A, F 5-51. Considere la situación de la figura 5-59 para cada uno de los siguientes conjuntos de valores de sincronización. Para cada uno de ellos indique si el flip-flop Q_2 responderá o no en forma correcta.

(a)* Cada FF: $t_{PLH} = 12$ ns; $t_{PHL} = 8$ ns; $t_S = 5$ ns; $t_H = 0$ ns;
Compuerta NAND: $t_{PLH} = 8$ ns; $t_{PHL} = 6$ ns;
INVERSOR: $t_{PLH} = 7$ ns; $t_{PHL} = 5$ ns;

(b) Cada FF: $t_{PLH} = 10$ ns; $t_{PHL} = 8$ ns; $t_S = 5$ ns; $t_H = 0$ ns;
Compuerta NAND: $t_{PLH} = 12$ ns; $t_{PHL} = 10$ ns;
INVERSOR: $t_{PLH} = 8$ ns; $t_{PHL} = 6$ ns;

D 5-52. Muestre y explique cómo el problema de desfasamiento de reloj de la figura 5-59 puede eliminarse si se insertan dos INVERSORES en forma apropiada.

F 5-53. Consulte el circuito de la figura 5-88. Suponga que los CIs son de la familia lógica TTL. La forma de onda de Q se obtuvo cuando se probó el circuito con las señales de entrada que se muestran, y con el interruptor en la posición "arriba"; no es correcta. Considere la siguiente lista de fallas y para cada una indique "sí" o "no" en cuanto a cuál podría ser la verdadera falla. Explique cada respuesta.

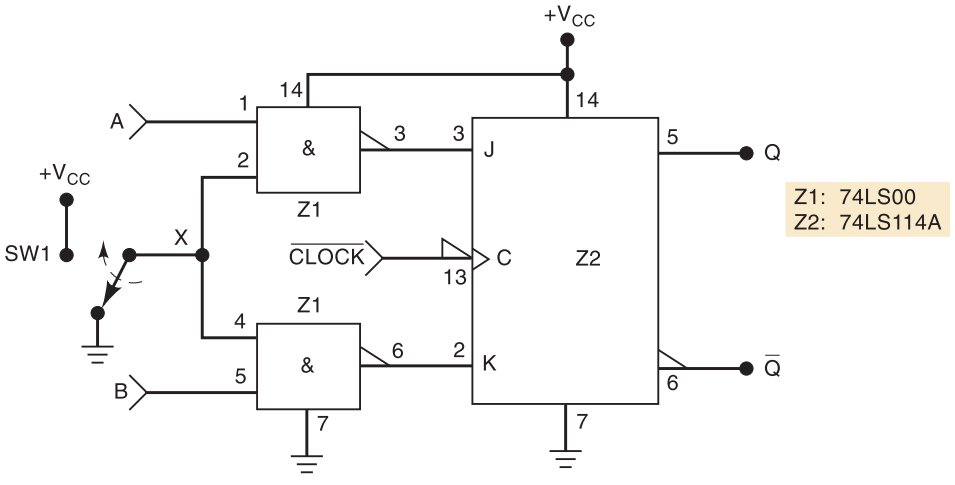
(a)* El punto X siempre está en BAJO debido a un interruptor defectuoso.

(b)* La terminal 1 de Z1 está en corto interno con V_{CC}.

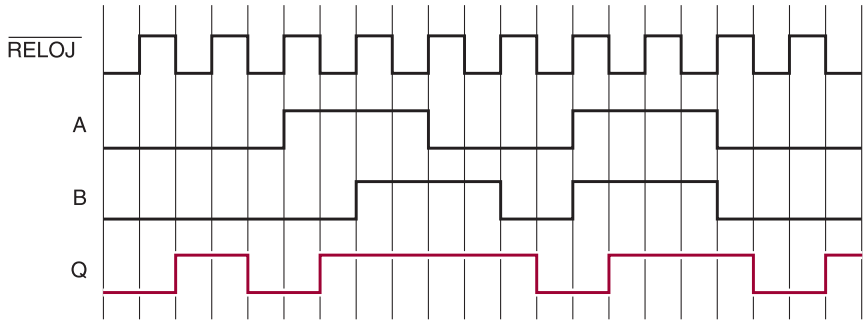
(c) La conexión de Z1-3 a Z2-3 está rota.

(d) Hay un puente de soldadura entre las terminales 6 y 7 de Z1.

FIGURA 5-88
Problema 5-53.



(a)



(b)

A 5-54. El circuito de la figura 5-89 funciona como un candado de combinación secuencial. Para operar el candado, proceda de la siguiente manera:

1. Active por un momento el interruptor RESET.
2. Ajuste los interruptores SWA, SWB y SWC con la primera parte de la combinación. Después conmute por un momento el interruptor INTRODUCIR hacia delante y hacia atrás.
3. Ajuste los interruptores con la segunda parte de la combinación y conmute el interruptor INTRODUCIR de nuevo. Esto deberá producir un nivel ALTO en Q_2 para abrir el candado.

   Si se introduce la combinación incorrecta en cualquiera de los pasos, el operador deberá comenzar la secuencia de nuevo. Analice el circuito y determine la secuencia correcta de combinaciones que abrirán el candado.

A, F 5-55.* Cuando se prueba el candado de combinación de la figura 5-89, se descubre que al introducir la combinación correcta no se abre el candado. Una comprobación con la sonda lógica muestra que al introducir la primera combinación correcta Q_1 se establece en ALTO, pero al introducir la segunda combinación correcta sólo se produce un pulso momentáneo en Q_2. Considere cada una de las siguientes fallas e indique cuál(es) podría(n) producir la operación observada. Explique cada una de las opciones.

(a) Rebote de interruptor en SWA, SWB o SWC.

(b) La entrada *CLR* de Q_2 está abierta.

(c) La conexión de la salida de la compuerta NAND 4 a la entrada de la compuerta NAND 3 está abierta.

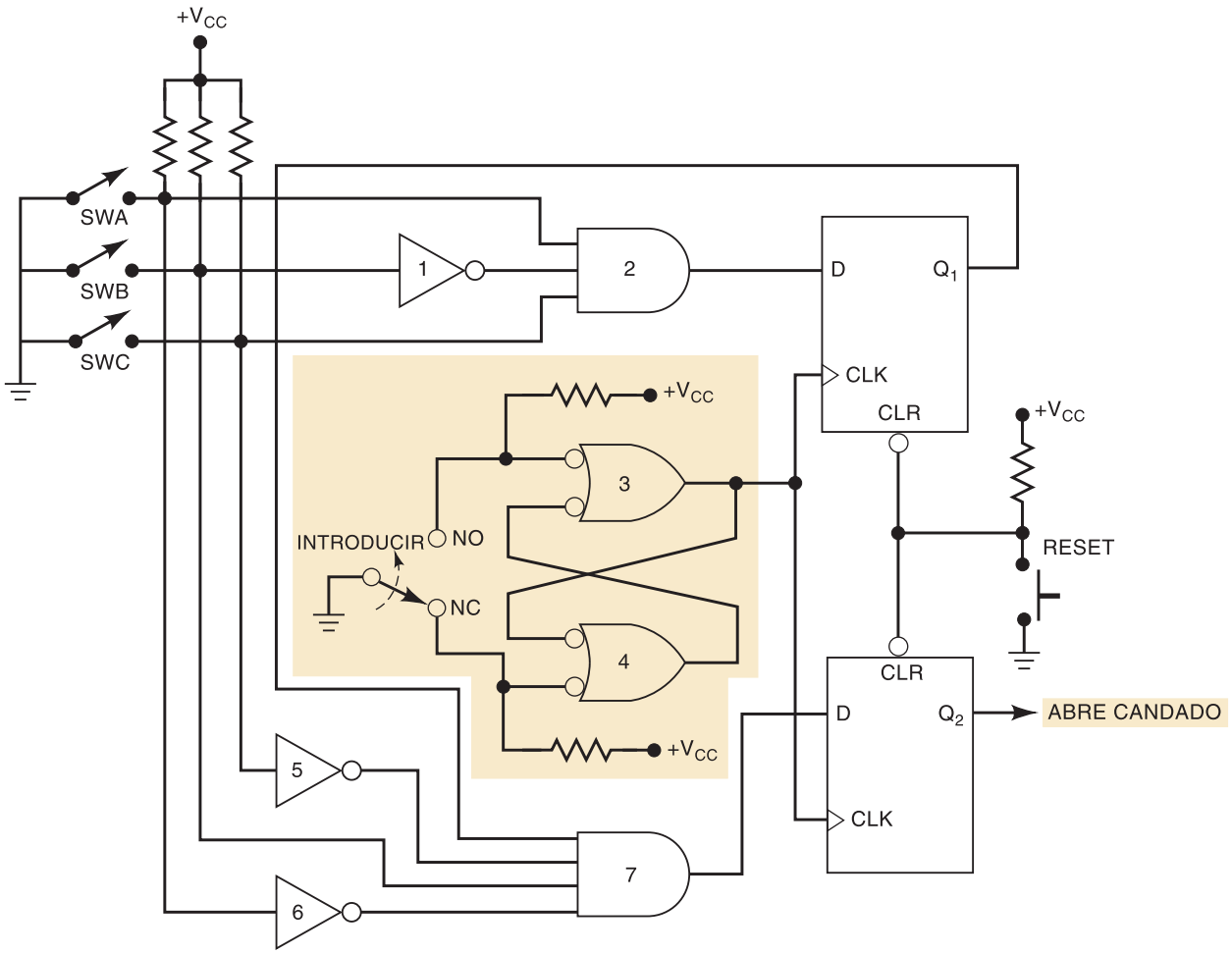


FIGURA 5-89 Problemas 5-54 y 5-55.

PREGUNTAS DE PRÁCTICA

B 5-56. Para cada una de las siguientes aseveraciones, indique qué tipo de FF se está describiendo.

(a)* Tiene una entrada SET y una entrada CLEAR pero no tiene entrada *CLK*.

(b)* Conmutará en cada pulso de *CLK* cuando ambas entradas de control estén en ALTO.

(c)* Tiene una entrada HABILITAR en vez de una entrada *CLK*.

(d)* Se utiliza para transferir datos con facilidad de un registro de FF a otro.

(e) Sólo tiene una entrada de control.

(f) Tiene dos salidas que son complementos una de la otra.

(g) Puede cambiar de estado sólo en la transición activa de CLK.

(h) Se utiliza en los contadores binarios.

B 5-57. Defina los siguientes términos.

(a) Entradas asíncronas.

(b) Disparado por flanco.

(c) Registro de desplazamiento.

(d) División de frecuencia.

(e) Transferencia asíncrona (atascada).
(f) Diagrama de transición de estados.
(g) Transferencia de datos en paralelo.
(h) Transferencia de datos en serie.
(i) Monoestable (one-shot) redisparable.
(j) Entradas tipo disparador de Schmitt.

SECCIÓN 5-25

B 5-58. Simule el diseño de HDL para un latch NAND, el cual se muestra en la figura 5-62 (AHDL) o en la figura 5-63 (VHDL). ¿Qué hace este latch SR si se aplica un comando de entrada "inválido"? Como sabemos que cualquier latch SR puede tener un resultado de salida inusual cuando se le aplica un comando de entrada inválido, debe simular esa condición de entrada junto con los comandos normales establecer, restablecer y retener del latch. Algunos diseños de latches pueden tener una tendencia en la que la salida oscila cuando un comando inválido va seguido de un comando de retención, por lo que debe revisar eso en su simulación.

B, H 5-59.* Escriba el código de HDL para un latch SR con entradas activas en ALTO.

B, H 5-60. Modifique la descripción del latch que se muestra en la figura 5-62 (AHDL) o en la figura 5-63 (VHDL) para hacer que el SR se restablezca si se le aplica una entrada inválida. Simule el diseño.

B, H 5-61.* Agregue salidas invertidas a los diseños en HDL del latch NAND que se muestran en la figura 5-62 o en la figura 5-63. Verifique su correcta operación por medio de la simulación.

B 5-62. Simule el diseño en AHDL o en VHDL para un latch D que aparece en la sección 5-25.

D, H 5-63. Cree un latch transparente de cuatro bits con una entrada *habilitar*, con base en el diseño en AHDL o VHDL para un dispositivo latch D individual que aparece en la sección 5-25. Simule el latch de cuatro bits.

D, H, N 5-64. Un flip-flop de conmutación (T) tiene una sola entrada de control (T). Cuando T = 0, el flip-flop se encuentra en el estado sin cambio, algo similar a un FF J-K con J = K = 0. Cuando T = 1 el flip-flop se encuentra en el modo de conmutación, algo similar a un JKFF con J = K = 1. Escriba el archivo de diseño en:
(a) AHDL.
(b) VHDL.

H 5-65. (a) Escriba el código en AHDL para el registro de desplazamiento que se muestra en la figura 5-45.
(b) Escriba el código en VHDL para el registro de desplazamiento que se muestra en la figura 5-45.

H 5-66. (a)* Escriba el código en AHDL para el registro de desplazamiento que se muestra en la figura 5-46.
(b) Escriba el código en VHDL para el registro de desplazamiento que se muestra en la figura 5-46.

H 5-67. (a) Escriba el código en AHDL para el circuito de FF que se muestra en la figura 5-59.
(b) Escriba el código en VHDL para el circuito de FF que se muestra en la figura 5-59.

H 5-68. (a) Escriba el código en AHDL para implementar todo el circuito de la figura 5-89.
(b) Escriba el código en VHDL para implementar todo el circuito de la figura5-89.

RESPUESTAS A LAS PREGUNTAS DE REPASO DE LAS SECCIONES

SECCIÓN 5-1

1. ALTO; BAJO 2. $Q = 0, \overline{Q} = 1$ 3. Verdadero 4. Aplicar un nivel BAJO momentáneo en la entrada $\overline{\text{SET}}$.

SECCIÓN 5-2

1. BAJO; ALTO 2. $Q = 1$ y $\overline{Q} = 0$ 3. Hacer que CLEAR = 1 4. $\overline{\text{SET}}$ y $\overline{\text{RESET}}$ por lo general, estarían en su estado activo en BAJO.

SECCIÓN 5-4

1. Entradas de control síncronas y entrada de reloj 2. La salida del FF puede cambiar sólo cuando ocurre la transición de reloj apropiada. 3. Falso. 4. El tiempo de estabilización es el intervalo requerido justo antes del flanco activo de la señal *CLK*, durante la cual las entradas de control deben mantenerse estables. El tiempo de retención es el intervalo requerido justo después del flanco activo de *CLK*, durante el cual las entradas de control deben mantenerse estables.

SECCIÓN 5-5

1. ALTO; BAJO; ALTO 2. Porque *CLK** está en ALTO sólo durante unos cuantos nanosegundos.

SECCIÓN 5-6

1. Verdadero 2. No 3. $J = 1, K = 0$

SECCIÓN 5-7

1. Q cambiará a BAJO en el punto a y permanecerá en BAJO. 2. Falso. La entrada D puede cambiar sin afectar a Q, ya que ésta puede cambiar sólo en el flanco activo de *CLK*. 3. Sí, convirtiendo a FFs D (figura 5-25).

SECCIÓN 5-8

1. En un latch D, la salida Q puede cambiar mientras *EN* está en ALTO. En un flip-flop D, la salida puede cambiar sólo en el flanco activo de *CLK*. 2. Falso. 3. Verdadero.

SECCIÓN 5-9

1. Las entradas asíncronas funcionan en forma independiente de la entrada *CLK*.
2. Sí, ya que $\overline{PRE}$ es activa en BAJO. 3. $J = K = 1$, $\overline{PRE} = \overline{CLR} = 1$, y una PGT en *CLK*.

SECCIÓN 5-10

1. El triángulo dentro del rectángulo indica la operación de disparo por flanco; el triángulo recto fuera del rectángulo indica el disparo en una NGT. 2. Se utiliza para indicar la función de las entradas que son comunes para más de un circuito en el chip.

SECCIÓN 5-11

1. t_{PLH} y t_{PHL} 2. Falso; la forma de onda también debe satisfacer los requerimientos de $t_w(L)$ y $t_w(H)$.

SECCIÓN 5-17

1. Falso. 2. Flip-flop D. 3. Seis. 4. Verdadero.

SECCIÓN 5-18

1. Verdadero 2. Menos interconexiones entre registros
3. $X_2X_1X_0 = 111$; $Y_2Y_1Y_0 = 101$ 4. En la transmisión de datos en paralelo.

SECCIÓN 5-19

1. 10 kHz 2. Ocho 3. 256 4. 2 kHz 5. $00001000_2 = 8_{10}$

SECCIÓN 5-21

1. La salida puede contener oscilaciones. 2. Producirá señales de salida más rápidas y limpias, incluso hasta para las señales de entrada con cambios lentos.

SECCIÓN 5-22

1. $Q = 0, \overline{Q} = 1$ 2. Verdadero. 3. Los valores externos de R y C. 4. Para un OS redisparable, cada nuevo pulso de disparo comienza un nuevo intervalo t_P sin importar el estado de la salida Q.

SECCIÓN 5-23

1. 24 kHz 2. 109.3 kHz; 66.7 por ciento. 3. Estabilidad de frecuencia.

SECCIÓN 5-24

1. El desfasamiento de reloj es la llegada de una señal de reloj a las entradas *CLK* de distintos FFs en tiempos distintos. Puede hacer que un FF cambie a un estado incorrecto.

SECCIÓN 5-25

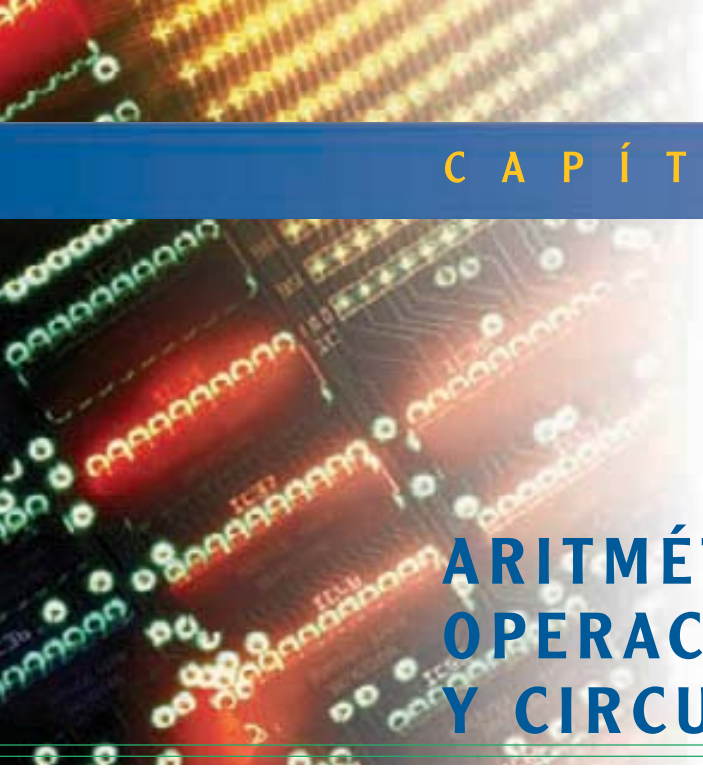
1. Retroalimentación: las salidas se combinan con las entradas para determinar el siguiente estado de las salidas. 2. Progresa a través de una secuencia predeterminada de estados en respuesta a una señal de reloj de entrada.

SECCIÓN 5-26

1. Un bloque de construcción estándar de una biblioteca de componentes que realice cierta función lógica fundamental. 2. Los nombres de cada entrada y salida, y el nombre de la primitiva que reconoce el sistema de desarrollo. 3. Bajo el menú HELP (AYUDA). 4. El bloque PROCESS permite las instrucciones IF secuenciales y el atributo EVENT detecta las transiciones. 5. ieee.std_logic_1164. 6. altera.maxplus2

SECCIÓN 5-27

1. Sí 2. En la sección VARIABLE. 3. A cada una se le asigna un nombre de variable. 4. = 5. Las señales (SIGNAL). 6. PORT MAP



CAPÍTULO 6

ARITMÉTICA DIGITAL: OPERACIONES Y CIRCUITOS

CONTENIDO

6-1 Suma binaria
6-2 Representación de números con signo
6-3 Suma en el sistema de complemento a 2
6-4 Resta en el sistema de complemento a 2
6-5 Multiplicación de números binarios
6-6 División binaria
6-7 Suma BCD
6-8 Aritmética hexadecimal
6-9 Circuitos aritméticos
6-10 Sumador binario en paralelo
6-11 Diseño de un sumador completo
6-12 Sumador completo en paralelo con registros
6-13 Propagación del acarreo
6-14 Sumador en paralelo de circuito integrado
6-15 Sistema de complemento a 2
6-16 Circuitos integrados tipo ALU
6-17 Ejemplo práctico de diagnóstico de fallas
6-18 Uso de las funciones de la biblioteca TTL con Altera
6-19 Operaciones lógicas con arreglos de bits
6-20 Sumadores en HDL
6-21 Expansión de la capacidad de bits de un circuito

OBJETIVOS

Al terminar este capítulo, usted podrá:

- Realizar sumas, restas, multiplicaciones y divisiones binarias con dos números binarios.
- Sumar y restar números hexadecimales.
- Conocer la diferencia entre la suma binaria y la suma con OR.
- Comparar las ventajas y desventajas entre tres sistemas distintos de representación de números binarios con signo.
- Manipular números binarios con signo utilizando el sistema de complemento a 2.
- Comprender el proceso de suma BCD.
- Describir la operación básica de una unidad aritmética/lógica.
- Emplear sumadores completos en el diseño de sumadores binarios en paralelo.
- Citar las ventajas de los sumadores en paralelo con la característica de acarreo anticipado.
- Explicar la operación de un circuito sumador/restador en paralelo.
- Utilizar un circuito integrado ALU para realizar varias operaciones lógicas y aritméticas sobre los datos de entrada.
- Analizar ejemplos prácticos de diagnóstico de fallas para los circuitos sumadores/restadores.
- Usar formas de HDL de piezas TTL estándar de bibliotecas para implementar circuitos más complicados.
- Utilizar la forma de descripción de la ecuación Booleana para realizar operaciones sobre conjuntos completos de bits.
- Aplicar las técnicas de ingeniería de software para expandir la capacidad de una descripción de hardware.

INTRODUCCIÓN

Las computadoras y calculadoras digitales realizan las diversas operaciones aritméticas sobre números que se representan en forma binaria. El tema de la aritmética digital puede ser muy complejo si queremos comprender todos los diversos métodos de cálculo y la teoría detrás de ellos. Por fortuna la mayoría de los técnicos no requiere este nivel de conocimiento, al menos no hasta que se hayan convertido en programadores de computadoras experimentados. Nuestro enfoque en este capítulo será concentrarnos en los principios básicos que son necesarios para comprender la manera en que los equipos digitales (es decir, las computadoras) realizan las operaciones aritméticas básicas.

Primero veremos cómo se realizan las diversas operaciones aritméticas sobre números binarios mediante “lápiz y papel”, y después estudiaremos los circuitos

lógicos reales que llevan a cabo estas operaciones en un sistema digital. Por último veremos cómo describir estos circuitos simples mediante el uso de las técnicas de HDL. También cubriremos varios métodos para expandir la capacidad de estos circuitos. Nos enfocaremos en los fundamentos del HDL, utilizando los circuitos aritméticos como ejemplo. La poderosa capacidad del HDL combinada con el hardware del PLD nos proveerá la base para más estudio, diseño y experimentación con circuitos aritméticos mucho más sofisticados en cursos más avanzados.

6-1 SUMA BINARIA

La suma de dos números binarios se realiza exactamente de la misma forma que la suma de números decimales. De hecho, la suma binaria es más simple debido a que hay menos casos que aprender. Primero revisemos la suma decimal:

```
    3  7  6     LSD
  +4  6  1
  ---------
    8  3  7
```

Primero se opera sobre la posición del dígito menos significativo (LSD), con lo cual se produce una suma de 7. Después se suman los dígitos en la segunda posición para producir una suma de 13, la cual produce un **acarreo** de 1 hacia la tercera posición. En esta posición se produce una suma de 8.

En la suma binaria se siguen los mismos pasos generales. No obstante, sólo pueden ocurrir cuatro casos cuando se suman los dos dígitos binarios (bits) en cualquier posición. Éstos son:

$$0 + 0 = 0$$
$$1 + 0 = 1$$
$$1 + 1 = 10 = 0 + \text{acarreo de 1 hacia la siguiente posición}$$
$$1 + 1 + 1 = 11 = 1 + \text{acarreo de 1 hacia la siguiente posición}$$

El último caso ocurre cuando los dos bits en cierta posición son 1 y hay un acarreo de la posición anterior. He aquí varios ejemplos de la suma de dos números binarios (los equivalentes decimales están entre paréntesis):

011 (3)	1001 (9)	11.011 (3.375)
+ 110 (6)	+ 1111 (15)	+ 10.110 (2.750)
1001 (9)	11000 (24)	110.001 (6.125)

No es necesario considerar la suma de más de dos números binarios a la vez, ya que en todos los sistemas digitales los circuitos que se encargan de realizar la suma sólo pueden manejar dos números al mismo tiempo. Cuando se van a sumar más de dos números, los primeros dos se suman y luego el resultado se suma al tercer número, y así sucesivamente. Ésta no es una desventaja grave, ya que por lo general las computadoras digitales modernas pueden realizar una operación de suma en varios nanosegundos.

La suma es la operación aritmética más importante en los sistemas digitales. Como veremos, las operaciones de resta, multiplicación y división que se realizan

en la mayoría de las computadoras y calculadoras digitales modernas en realidad utilizan sólo la suma como su operación básica.

PREGUNTAS DE REPASO

1. Sume los siguientes pares de números binarios.
   (a) 10110 + 00111
   (b) 011.101 + 010.010
   (c) 10001111 + 00000001

6-2 REPRESENTACIÓN DE NÚMEROS CON SIGNO

En las computadoras digitales, los números binarios se representan mediante un conjunto de dispositivos de almacenamiento binario (por ejemplo, flip-flops). Cada dispositivo representa un bit. Por ejemplo, un registro FF de seis bits puede almacenar números binarios que varían desde 000000 hasta 111111 (de 0 a 63 en decimal). Esto representa la *magnitud* del número. Debido a que la mayoría de las computadoras y calculadoras digitales maneja números tanto negativos como positivos, se requiere de algún medio para representar el *signo* del número (+ o −). Por lo general lo que se hace es agregar otro bit al número; a este bit se le llama **bit de signo**. En general, la convención común es que un 0 en el signo representa a un número positivo y un 1 en el bit de signo representa a un número negativo. Esto se ilustra en la figura 6-1. El registro A contiene los bits 0110100. El 0 en el bit más a la izquierda (A_6) es el bit de signo que representa al signo +. Los otros seis bits son la magnitud del número 110100_2, el cual es equivalente al 52 decimal. Entonces, el número almacenado en el registro A es +52. De manera similar, el número almacenado en el registro B es −52 ya que el bit de signo es 1, que representa al −.

El bit de signo se utiliza para indicar la naturaleza positiva o negativa del número binario almacenado. Los números de la figura 6-1 consisten de un bit de signo y de seis bits de magnitud. Los bits de magnitud son el equivalente binario real del valor decimal que se está representando. A éste se le conoce como **sistema de signo-magnitud** para representar números binarios con signo.

Aunque el sistema de signo-magnitud es simple, las calculadoras y computadoras no lo utilizan de manera usual porque la implementación del circuito es más compleja que en otros sistemas. El sistema más común que se utiliza para representar números binarios con signo es el **sistema de complemento a 2**. Antes de ver cómo se hace esto, primero debemos ver cómo se forma el complemento a 1 y el complemento a 2 de un número binario.

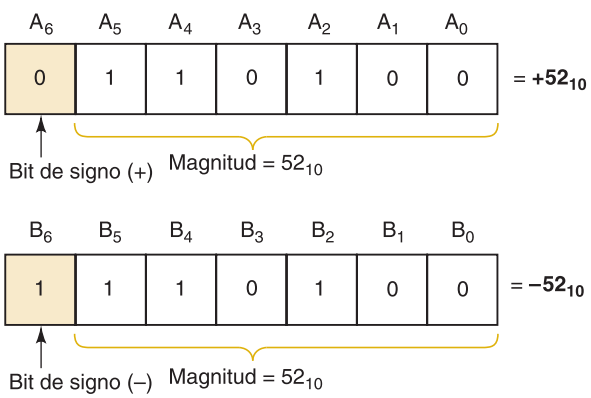


FIGURA 6-1
Representación de números con signo en la forma de signo-magnitud.

Forma de complemento a 1

El complemento a 1 de un número binario se obtiene al cambiar cada 0 por un 1 y cada 1 por un 0. En otras palabras, se cambia cada uno de los bits en el número por su complemento. El proceso se muestra a continuación.

```
1 0 1 1 0 1  número binario original
↓ ↓ ↓ ↓ ↓ ↓
0 1 0 0 1 0  se complementa cada bit para formar el complemento a 1
```

Por lo tanto, decimos que el complemento a 1 de 101101 es 010010.

Forma de complemento a 2

El complemento a 2 de un número binario se forma al tomar el complemento a 1 de ese número y sumarle 1 a la posición del bit menos significativo. El proceso se ilustra a continuación para el número $101101_2 = 45_{10}$.

```
  1 0 1 1 0 1    equivalente binario de 45
  0 1 0 0 1 0    se complementa cada bit para formar el complemento a 1
+         1      se le suma 1 para formar el complemento a 2
  0 1 0 0 1 1    complemento a 2 del número binario original
```

Entonces, decimos que 010011 es la representación en complemento a 2 de 101101.

He aquí otro ejemplo de cómo convertir un número binario a su representación en complemento a 2:

```
  1 0 1 1 0 0    número binario original
  0 1 0 0 1 1    complemento a 1
+         1      se le suma 1
  0 1 0 1 0 0    complemento a 2 del número original
```

Representación de números con signo mediante el complemento a 2

El sistema de complemento a 2 para representar números con signo funciona así:

- Si el número es positivo, la magnitud se representa en su forma binaria real y se coloca el bit de signo 0 enfrente del MSB. Esto se muestra en la figura 6-2 para el número $+45_{10}$.
- Si el número es negativo, la magnitud se representa en su forma de complemento a 2 y se coloca el bit de signo 1 enfrente del MSB. Esto se muestra en la figura 6-2 para el número -45_{10}.



FIGURA 6-2
Representación de números con signo en el sistema de complemento a 2.

El sistema de complemento a 2 se utiliza para representar números con signo ya que, como veremos, nos permite realizar la operación de la resta, a partir de una operación de suma. Esto es importante ya que significa que una computadora digital puede utilizar los mismos circuitos tanto para sumar como para restar, lo cual redunda en un ahorro en el hardware.

EJEMPLO 6-1

Represente cada uno de los siguientes números decimales con signo en forma de un número binario con signo en el sistema de complemento a 2. Use un total de cinco bits, incluyendo el bit de signo.

(a) +13 (b) −9 (c) +3 (d) −2 (e) −8

Solución

(a) El número es positivo, por lo que la magnitud (13) se representará en su forma de magnitud real, es decir: $13 = 1101_2$. Si agregamos el bit de signo 0 tenemos que

$$+13 = 01101$$

bit de signo ↑

(b) El número es negativo, por lo que la magnitud (9) deberá representarse en forma de complemento a 2:

$$\begin{array}{rll} 9_{10} = 1001_2 & & \\ \quad 0110 & & \text{(complemento a 1)} \\ + \quad 1 & & \text{(se suma 1 al LSB)} \\ \hline \quad 0111 & & \text{(complemento a 2)} \end{array}$$

Cuando agregamos el bit de signo 1, el número con signo completo se convierte en

$$-9 = 10111$$

bit de signo ↑

El procedimiento que acabamos de seguir requiere de dos pasos. Primero determinamos el complemento a 2 de la magnitud y después agregamos el bit de signo. Esto puede lograrse en un solo paso si incluimos el bit de signo en el proceso del complemento a 2. Por ejemplo, para encontrar la representación para −9, empezamos con la representación para +9, *incluyendo el bit de signo*, y tomamos el complemento a 2 de ese número para poder obtener la representación para −9.

$$\begin{array}{rll} +9 = 01001 & & \\ \quad 10110 & & \text{(complemento a 1 de cada bit, incluyendo el bit de signo)} \\ + \quad 1 & & \text{(se suma 1 al LSB)} \\ \hline -9 = 10111 & & \text{(representación de } -9 \text{ en complemento a 2)} \end{array}$$

El resultado es, desde luego, el mismo que obtuvimos antes.

(c) El valor decimal 3 puede representarse en binario con sólo usar dos bits. No obstante, la declaración del problema requiere una magnitud de cuatro bits precedida por un bit de signo. Por lo tanto, tenemos:

$$+3_{10} = 0011$$

En muchas situaciones el número de bits se fija con base en el tamaño del registro que almacenará los números binarios, por lo que tal vez debamos agregar 0s para poder llenar el número requerido de posiciones de bits.

$$
\begin{array}{rll}
+2 = & 00010 & \\
 & 11101 & \text{(complemento a 1)} \\
+ & \underline{\quad\quad 1} & \text{(se suma 1)} \\
-2 = & 11110 & \text{(representación de } -2 \text{ en complemento a 2)}
\end{array}
$$

(d) Empezamos por escribir +2 mediante el uso de cinco bits:
(e) Empezamos con +8:

$$
\begin{array}{rll}
+8 = & 01000 & \\
 & 10111 & \text{(se complementa cada bit)} \\
+ & \underline{\quad\quad 1} & \text{(se suma 1)} \\
-8 = & 11000 & \text{(representación de } -8 \text{ en complemento a 2)}
\end{array}
$$

Extensión del signo

En el ejemplo 6-1 tuvimos que utilizar un total de cinco bits para representar los números con signo. El tamaño de un registro (*número de flip-flops*) determina el número de dígitos binarios que se almacenan para cada número. La mayoría de los sistemas digitales hoy en día almacenan números en registros con tamaños en múltiplos pares de cuatro bits. En otras palabras, los registros de almacenamiento están compuestos de 4, 8, 16, 32 o 64 bits. En un sistema que almacena números de ocho bits, siete de ellos representan la magnitud y el MSB representa el signo. Si necesitamos almacenar un número positivo de cinco bits en un registro de ocho bits, sólo basta con agregar ceros a la izquierda. El MSB (bit de signo) sigue siendo 0, lo

$$\underbrace{0000}_{\text{0s a la izquierda que se agregaron}}\ \underbrace{1001}_{\text{valor binario para el 9}}$$

cual indica un valor positivo.

¿Qué ocurre si tratamos de almacenar números negativos de cinco bits en un registro de ocho bits? En la sección anterior vimos que la representación binaria en

$$1\,0111$$

complemento a 2 de cinco bits para el −9 es 10111.
Si anexáramos 0s a la izquierda, éste ya no sería un número negativo en el formato de ocho bits. La manera apropiada de extender un número negativo es anexando 1s a la izquierda. Así, el valor almacenado para el 9 negativo es

$$111\,1\,0111$$

- 0111: magnitud en complemento a 2
- 1: signo en formato de cinco bits
- 111: extensión del signo al formato de ocho bits

Negación

La negación es la operación de convertir un número positivo en su equivalente negativo, o un número negativo en su equivalente positivo. Cuando los números binarios con signo se representan en el sistema de complemento a 2, para realizar la negación sólo basta con realizar la operación de complemento a 2. Para ilustrar esto empecemos con el +9 en su forma binaria de ocho bits. Su representación con signo es 00001001. Si sacamos su complemento a 2 obtendremos 11110111, el cual representa al valor con signo −9. De igual forma podemos empezar con la representación de −9, que es 11110111, y sacar su complemento a 2 para obtener 00001001, el cual representa a +9. A continuación se muestran estos pasos en un diagrama.

Empezamos con	00001001	+9
complemento a 2 (se cambia el signo)	11110111	−9
se niega otra vez	00001001	+9

Por lo tanto, para cambiar el signo de un número binario con signo lo complementamos a 2.

Esta negación cambia el número a su equivalente con signo opuesto. En los pasos (d) y (e) del ejemplo 6-1 utilizamos la negación para convertir los números positivos en sus equivalentes negativos.

EJEMPLO 6-2

Cada uno de los siguientes números es binario con signo de cinco bits en el sistema de complemento a 2. Determine el valor decimal en cada caso:

(a) 01100 (b) 11010 (c) 10001

Solución

(a) El bit de signo es 0, por lo que el número es *positivo* y los otros cuatro bits representan la magnitud real del número. Esto es, $1100_2 = 12_{10}$. Por lo tanto, el número decimal es +12.

(b) El bit de signo de 11010 es un 1, por lo que sabemos que el número es negativo, pero no podemos saber cuál es la magnitud. Podemos encontrarla si cambiamos el signo (complementamos a 2) al número para convertirlo en su equivalente positivo.

11010	(número negativo original)
00101	(complemento a 1)
+ 1	(se suma 1)
110	(+6)

Como el resultado de la negación es 0010 = +6, el número original 11010 debe ser equivalente a −6.

(c) Siga el mismo procedimiento que en (b):

10001	(número negativo original)
01110	(complemento a 1)
+ 1	(se suma 1)
1111	(+15)

Por lo tanto, 10001 = −15.

Caso especial en la representación de complemento a 2

Siempre que un número con signo tiene un 1 en el bit de signo y todos los bits de su magnitud son 0, su equivalente decimal es -2^N, en donde N es el número de bits en la *magnitud*. Por ejemplo,

$$\begin{aligned} 1000 &= -2^3 = -8 \\ 10000 &= -2^4 = -16 \\ 100000 &= -2^5 = -32 \end{aligned}$$

y así en lo sucesivo. Observe que en este caso especial, al sacar el complemento a 2 de estos números se produce el valor con el que comenzamos, ya que estamos en el límite negativo del intervalo de números que pueden representarse mediante esta cantidad de bits. Si extendemos el signo de estos números especiales, el procedimiento normal de negación funciona sin problemas. Por ejemplo, al extender el número 1000 (-8) a 11000 (8 negativo de cinco bits) y sacar su complemento a 2 obtenemos 01000 (8), que es la magnitud del número negativo.

Así, podemos afirmar que el intervalo completo de valores que pueden representarse en el sistema de complemento a 2 con N bits de magnitud es

$$-2^N \text{ to } +(2^N - 1)$$

Hay un total de 2^{N+1} valores distintos, *incluyendo* el cero.

Por ejemplo, la tabla 6-1 lista todos los números con signo que pueden representarse en cuatro bits mediante el sistema de complemento a 2 (observe que hay tres bits de magnitud, por lo que $N = 3$). La secuencia empieza en $-2^N = -2^3 = -8_{10} = 1000_2$ y continúa hasta $+(2^N - 1) = +2^3 - 1 = +7_{10} = 0111_2$ al sumar 0001 en cada paso, como en un contador ascendente.

TABLA 6-1

Valor decimal	Binario con signo mediante el uso del complemento a 2
$+7 = 2^3 - 1$	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
0	0000
−1	1111
−2	1110
−3	1101
−4	1100
−5	1011
−6	1010
−7	1001
$-8 = -2^3$	1000

EJEMPLO 6-3

¿Cuál es el intervalo de valores decimales *sin signo* que pueden representarse en un byte?

Solución

Recuerde que un byte es de ocho bits. Aquí nos interesan los números sin signo, por lo que no hay bit de signo y todos los ocho bits se utilizan para la magnitud. Por lo tanto, los valores variarán desde

$$00000000_2 = 0_{10}$$

hasta

$$11111111_2 = 255_{10}$$

Esto nos da un total de 256 valores distintos, los cuales podríamos haber previsto, ya que $2^8 = 256$.

EJEMPLO 6-4

¿Cuál es el intervalo de valores decimales *con signo* que pueden representarse en un byte?

Solución

Como el MSB se va a utilizar como el bit de signo, se tienen siete bits para la magnitud. El valor negativo más grande es

$$10000000_2 = -2^7 = -128_{10}$$

El valor positivo más grande es

$$01111111_2 = +2^7 - 1 = +127_{10}$$

En consecuencia, el intervalo es de -128 a $+127$; esto nos da un total de 256 valores distintos, incluyendo el cero. De manera alternativa, y como hay siete bits de magnitud ($N = 7$), entonces hay $2^{N+1} = 2^8 = 256$ valores distintos.

EJEMPLO 6-5

Cierta computadora almacena los siguientes dos números con signo en su memoria, utilizando el sistema de complemento a 2:

$$00011111_2 = +31_{10}$$
$$11110100_2 = -12_{10}$$

Mientras ejecuta un programa, a la computadora se le pide que convierta cada número en su signo opuesto; esto es, debe cambiar el $+31$ a -31 y el -12 a $+12$. ¿Cómo hará esto?

Solución

Ésta es la operación de cambio de signo, en la que un número con signo puede cambiar su polaridad con sólo realizar la operación de complemento a 2 sobre el número *completo*, incluyendo el bit de signo. Los circuitos de la computadora tomarán el número con signo de la memoria, encontrarán su complemento a 2 y pondrán el resultado de vuelta en la memoria.

PREGUNTAS DE REPASO

1. Represente cada uno de los siguientes valores como un número con signo de ocho bits en el sistema de complemento a 2.
   (a) +13 (b) − 7 (c) −128
2. Cada uno de los siguientes números es binario con signo en el sistema de complemento a 2. Determine el equivalente decimal para cada uno de ellos.
   (a) 100011 (b) 1000000 (c) 01111110
3. ¿Qué intervalo de valores decimales con signo pueden representarse en 12 bits (incluyendo el bit de signo)?
4. ¿Cuántos bits se requieren para representar valores decimales que varían desde −50 hasta +50?
5. ¿Cuál es el valor decimal negativo más grande que puede representarse mediante un número de dos bytes?
6. Realice la operación de complemento a 2 en cada uno de los siguientes números:
   (a) 1000 (b) 10000000 (c) 1000
7. Defina la operación de negación.

6-3 SUMA EN EL SISTEMA DE COMPLEMENTO A 2

Ahora investigaremos cómo se realizan las operaciones de suma y resta en los equipos digitales que utilizan la representación de complemento a 2 para los números negativos. En los diversos casos a considerar, es importante observar que el bit de signo de cada número se opera sobre la misma forma que los bits de magnitud.

Caso I: dos números positivos. La suma de dos números positivos es simple. Considere la suma de +9 y +4:

```
+9 → 0 1001   (primer sumando)
+4 → 0 0100   (sumando)
     0 1101   (suma = +13)
     ↑
     bits de signo
```

Observe que los bits de signo del **primer sumando** y del **sumando** son ambos 0 y que el bit de signo de la suma es 0, lo cual indica que la suma es positiva. Observe, además, que el primer sumando y el sumando se ajustan para tener el mismo número de bits. Esto debe hacerse *siempre* en el sistema de complemento a 2.

Caso II: número positivo y número negativo más pequeño. Considere la suma de +9 y −4. Recuerde que el −4 estará en su forma de complemento a 2. Por lo tanto, +4 (00100) debe convertirse en −4 (11100).

```
          bits de signo
          ↓
+9 → 0 1001   (primer sumando)
−4 → 1 1100   (sumando)
1  0 0101
↑
└ Este acarreo se descarta; el resultado es 00101 (suma = +5).
```

En este caso el bit de signo del sumando es 1. Observe que los bits de signo también participan en el proceso de la suma. De hecho, se genera un acarreo en la última

posición de la suma. *Este acarreo siempre se descarta*, por lo que la suma final es 00101, que equivale a +5.

Caso III: número positivo y número negativo más grande. Considere la suma de −9 y +4:

```
−9 →    10111
+4 →    00100
        11011      (suma = –5)
        ↑
        └── bit de signo negativo
```

Aquí la suma tiene un bit de signo en 1, lo cual indica un resultado negativo. Como el resultado de la suma es negativo se encuentra en su forma de complemento a 2, por lo que los últimos cuatro bits (1011) en realidad representan el complemento a 2 del resultado. Para encontrar la magnitud real, debemos negar (complementar a 2) el 11011; el resultado es 00101 = +5. De esta forma, 11011 representa a −5.

Caso IV: dos números negativos

```
−9 →    10111
−4 →    11100
      1 10011
      ↑ ↑
      │ └── bit de signo
      └──── Este acarreo se descarta; el resultado es 10011 (suma = –13).
```

Nuevamente el resultado final es negativo y está en forma de complemento a 2 con un bit de signo de 1. Al cambiar de signo (complementar a 2) este resultado se produce 01101 = +13.

Caso V: números iguales y opuestos

```
−9 →    10111
+9 →    01001
      1 00000
      ↑
      └──── Se descarta; el resultado es 0000 (suma = +0).
```

Es obvio que el resultado es +0, como se esperaba.

PREGUNTAS DE REPASO

Suponga que se está trabajando con el sistema de complemento a 2 para ambas preguntas.

1. *Verdadero o falso:* siempre que la suma de dos números binarios con signo tiene un bit de signo de 1, la magnitud de la suma se encuentra en forma de complemento a 2.
2. Sume los siguientes pares de números con signo. Exprese la suma como un número binario con signo y como un número decimal.
   (a) 100111 + 111011 (b) 100111 + 011001

6-4 RESTA EN EL SISTEMA DE COMPLEMENTO A 2

La operación de resta mediante el uso del sistema de complemento a 2 en realidad implica la operación de la suma y no es distinta de los diversos casos para la suma

que vimos en la sección 6-3. Cuando se resta un número binario (el **sustraendo**) de otro número binario (el **minuendo**) se utiliza el siguiente procedimiento:

1. *Negar el sustraendo*. Esto cambiará el sustraendo por su valor equivalente del signo opuesto.
2. *Sumar éste al minuendo*. El resultado de esta suma representará la *diferencia* entre el sustraendo y el minuendo.

Una vez más, como en todas las operaciones aritméticas de complemento a 2 es necesario que ambos números tengan la misma cantidad de bits en su representación.

Consideremos el caso en donde +4 se va a restar de +9.

$$\begin{aligned} \text{minuendo } (+9) \rightarrow &\quad 01001 \\ \text{sustraendo } (+4) \rightarrow &\quad 00100 \end{aligned}$$

Se cambia el signo al sustraendo para producir 11100, el cual representa a −4. Ahora se suma éste al minuendo.

$$\begin{array}{rl} 01001 & (+9) \\ +\ 11100 & (-4) \\ \hline 1\ 00101 & (+5) \end{array}$$

↑ Se descarta, por lo que el resultado es 00101 = +5.

Cuando el sustraendo se cambia por su complemento a 2, en realidad se convierte en −4 por lo que estamos *sumando* −4 y +9, que es lo mismo que restar +4 de +9. Este caso es igual al caso II de la sección 6-3. Entonces, cualquier operación de resta se convierte en una de suma cuando se utiliza el sistema de complemento a 2. Esta característica del sistema de complemento a 2 lo ha convertido en el sistema de más amplio uso de todos los métodos disponibles, ya que permite realizar la suma y la resta mediante el mismo circuito.

He aquí otro ejemplo que muestra la resta de +9 con −4:

$$\begin{array}{rl} 11100 & (-4) \\ -\ 01001 & (+9) \end{array}$$

Se niega el sustraendo (+9) para producir 10111 (−9) y se suma éste al minuendo (−4):

$$\begin{array}{rl} 11100 & (-4) \\ +\ 10111 & (-9) \\ \hline 1\ 10011 & (-13) \end{array}$$

↑ Se descarta

El lector deberá verificar los resultados de utilizar el procedimiento anterior para las siguientes restas: (a) +9 − (− 4); (b) −9 − (+4); (c) −9 − (−4); (d) +4 − (−4). Recuerde que cuando el resultado tiene un bit de signo de 1, es negativo y está en forma de complemento a 2.

Desbordamiento aritmético

En cada uno de los ejemplos anteriores de sumas y restas, los números que se sumaron consistían de un bit de signo y cuatro bits de magnitud. Las respuestas también consistían de un bit de signo y cuatro bits de magnitud. Se descartó cualquier acarreo hacia la sexta posición de bit. En todos los casos considerados, la magnitud de

la respuesta fue lo bastante pequeña como para ajustarla en cuatro bits. Veamos la suma de +9 y +8.

$$\begin{array}{rcc} +9 \rightarrow & 0 & 1001 \\ +8 \rightarrow & 0 & 1000 \\ \hline & 1 & 0001 \end{array}$$

signo incorrecto — magnitud incorrecta

La respuesta tiene un bit de signo negativo, lo cual es obvio que es incorrecto ya que estamos sumando dos números positivos. La respuesta deberá ser +17, pero la magnitud 17 requiere más de cuatro bits y, por lo tanto, se *desborda* hacia la posición del bit de signo. Esta condición de **desbordamiento** puede ocurrir sólo cuando se están sumando dos números positivos o dos números negativos, y siempre produce un resultado incorrecto. Para detectar el desbordamiento podemos comprobar si el bit de signo del resultado es el mismo que los bits de signo de los números que se van a sumar.

Para realizar la resta en el sistema de complemento a 2, se niega el minuendo y se *suma* al sustraendo, por lo que el desbordamiento sólo puede ocurrir cuando el minuendo y el sustraendo tienen signos diferentes. Por ejemplo, si vamos a restar −8 de +9, el −8 se niega para convertirse en +8 y se suma a +9 como se muestra a continuación, y el desbordamiento produce un resultado negativo erróneo ya que la magnitud es demasiado grande.

Una computadora debe tener un circuito especial para detectar cualquier condición de desbordamiento al sumar o restar dos números. Este circuito de detección enviará una señal a la unidad de control de la computadora para indicarle que ha ocurrido un desbordamiento y que el resultado es incorrecto. En uno de los problemas al final del capítulo examinaremos dicho circuito.

Círculos numéricos y aritmética binaria

Para ilustrar el concepto de la aritmética con signo y el desbordamiento vamos a tomar los números de la tabla 6-1 para "doblarlos" en un círculo numérico como el que se muestra en la figura 6-3. Observe que hay dos maneras de ver este círculo. Puede considerarse como un círculo de números sin signo (como se muestra en el anillo exterior) con un valor mínimo de 0 y un valor máximo de 15, o como números complementados a 2 con signo (como se muestra en el anillo interior) con un valor máximo de 7 y un valor mínimo de −8. Para sumar mediante el uso de un círculo numérico, sólo hay que empezar en el valor del primer sumando y avanzar alrededor

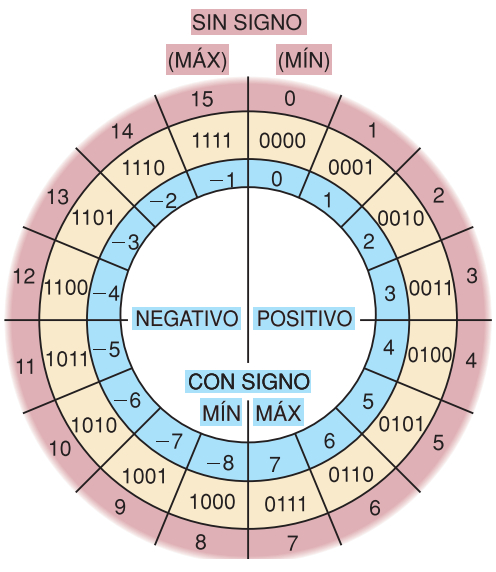


FIGURA 6-3 Un círculo numérico de cuatro bits.

del círculo numérico en sentido de las manecillas del reloj, según el número de espacios en el sumando. Por ejemplo, para sumar 2 + 3 se empieza en 2 (0010) y después se avanza en sentido de las manecillas del reloj tres espacios más para llegar al 5 (0101). El desbordamiento ocurre cuando la suma es demasiado grande como para caber dentro del formato con signo de cuatro bits, lo cual significa que nos hemos excedido del valor máximo de 7. En el círculo numérico esto se indica cuando al sumar dos valores positivos tenemos que cruzar la línea entre 0111 (máximo positivo) y 1000 (máximo negativo).

El círculo numérico puede ilustrar también cómo funciona en realidad la resta con complemento a 2. Por ejemplo, vamos a restar 5 de 3. Desde luego que sabemos que la respuesta es −2, pero vamos a hacerlo con el círculo numérico. Primero empezamos en el número 3 (0011) del círculo numérico. La manera más aparente de restar es avanzar *en sentido contrario a las manecillas del reloj* alrededor del círculo por cinco espacios, para quedar en el número 1110 (−2). La operación menos obvia que ilustra la aritmética con complemento a 2 es sumar −5 al número 3. El cinco negativo (el complemento a 2 de 0101) es 1011, que si se interpreta como número binario sin signo representa el valor 11 (once) en decimal. Empiece en el número 3 (0011) y avance en sentido a favor de las manecillas del reloj alrededor del círculo 11 espacios y se encontrará una vez más en el número 1110 (−2), que es el resultado correcto.

Cualquier operación de resta entre números de cuatro bits con signos opuestos que produzca un resultado mayor de 7 o menor de −8 es un desbordamiento del formato de cuatro bits y produce una respuesta incorrecta. Por ejemplo, 3 menos −6 debe producir la respuesta 9, pero si avanzamos en sentido de las manecillas del reloj seis espacios a partir del 3, quedaremos en el número con signo −7: se ha producido una condición de desbordamiento, la cual nos proporciona una respuesta incorrecta.

PREGUNTAS DE REPASO

1. Realice la resta en los siguientes pares de números con signo; utilice el sistema de complemento a 2. Exprese los resultados como números binarios con signo y como valores decimales.
   (a) 01001 − 11010 (b) 10010 − 10011
2. ¿Cómo puede detectarse el desbordamiento aritmético cuando se están sumando números con signo? ¿Y cuando se están restando?

6-5 MULTIPLICACIÓN DE NÚMEROS BINARIOS

La multiplicación de números binarios se realiza de la misma forma que la multiplicación de números decimales. En realidad el proceso es más simple, ya que los dígitos multiplicadores son 0 o 1, por lo que siempre estamos multiplicando por 0 o 1 y ningún otro dígito. El siguiente ejemplo ilustra esto para los números binarios sin signo:

```
   1001      ← multiplicando = 9₁₀
   1011      ← multiplicador = 11₁₀
  ------
   1001   }
   1001   }  productos parciales
  0000    }
  1001    }
  ------
1100011   }  producto final
```

En este ejemplo el multiplicando y el multiplicador están en su forma binaria natural por lo que no se utilizan bits de signo. Los pasos que se siguen en el proceso son

exactamente los mismos que en la multiplicación decimal. Primero se examina el LSB del multiplicador; en nuestro ejemplo es un 1. Este 1 multiplica el multiplicando para producir 1001, el cual se anota como el primer producto parcial. A continuación se examina el segundo bit del multiplicador. Es un 1, por lo que se escribe 1001 para el segundo producto parcial. Observe que este segundo producto parcial se *desplaza* una posición a la izquierda, en forma relativa al primer producto. El tercer bit del multiplicador es 0, por lo que se escribe 0000 como el tercer producto parcial; de nuevo, se desplaza una posición a la izquierda en forma relativa al producto parcial anterior. El cuarto bit del multiplicador es 1, por lo que el último producto parcial es 1001, desplazado una posición a la izquierda. Después los cuatro productos parciales se suman para producir el producto final.

La mayoría de los equipos digitales puede sumar sólo dos números binarios a la vez. Por esta razón los productos parciales que se forman durante la multiplicación no pueden sumarse todos juntos al mismo tiempo, sino que se suman de dos en dos; es decir, el primero se suma al segundo, después el resultado se suma al tercero y así en lo sucesivo. A continuación ilustraremos este proceso para el ejemplo anterior:

Sumar {	1001	← primer producto parcial
	1001	← segundo producto parcial desplazado a la izquierda
Sumar {	11011	← suma de los primeros dos productos parciales
	0000	← tercer producto parcial desplazado a la izquierda
Sumar {	011011	← suma de los primeros tres productos parciales
	1001	← cuarto producto parcial desplazado a la izquierda
	1100011	← suma de los cuatro productos parciales, que equivale al producto total final

Multiplicación en el sistema de complemento a 2

En las computadoras que utilizan la representación en complemento a 2, la multiplicación se lleva a cabo en la manera antes descrita, siempre y cuando tanto el multiplicando como el multiplicador se coloquen en forma binaria natural. Si los dos números a multiplicar son positivos, ya se encuentran en la forma binaria natural y se multiplican como están. Desde luego que el producto resultante es positivo y se le asigna un bit de signo de 0. Cuando los dos números sean negativos, deberán estar en forma de complemento a 2. Se saca el complemento a 2 de cada uno para convertirlo en un número positivo y después se multiplican esos dos números. El producto se mantiene como un número positivo y recibe un bit de signo de 0.

Cuando uno de los números es positivo y el otro negativo, primero se convierte el número negativo en una magnitud positiva, sacando su complemento a 2. El producto se encontrará en forma de magnitud real. No obstante, el producto debe ser negativo ya que los números originales son de signos opuestos. Como consecuencia, el producto se cambia a su forma de complemento a 2 y se le asigna un bit de signo de 1.

PREGUNTAS DE REPASO

1. Multiplique los números sin signo 0111 y 1110.

6-6 DIVISIÓN BINARIA

El proceso para dividir un número binario (el *dividendo*) entre otro (el *divisor*) es el mismo que el que se utiliza para los números decimales, al cual, por lo general, se le conoce como “división larga”. El proceso actual es más simple en binario, ya que cuando estamos comprobando cuántas veces “cabe” el divisor en el dividendo sólo

hay dos posibilidades: 0 o 1. Para ilustrar esto, considere los siguientes ejemplos simples de división:

```
   0011          0010.1
11/1001       100/1010.0
   011          100
   0011          100
    11           100
     0             0
```

En el primer ejemplo tenemos el número 1001_2 dividido entre 11_2, que en decimal equivale a $9 \div 3$. El cociente resultante es $0011_2 = 3_{10}$. En el segundo ejemplo, el número 1010_2 se divide entre 100_2, o $10 \div 4$ en decimal. El resultado es $0010.1_2 = 2.5_{10}$.

En la mayoría de los equipos digitales modernos, por lo general, las restas que son parte de la operación de división se llevan a cabo mediante el uso de la resta con complemento a 2; es decir, se saca el complemento a 2 del sustraendo y después se suma.

La división de números con signo se maneja de la misma forma que la multiplicación. Los números negativos se convierten en positivos mediante su negación y después se lleva a cabo la división. Si el dividendo y el divisor tienen signos opuestos, el cociente resultante se cambia a número negativo, para lo cual se saca su complemento a 2 y se le asigna un bit de signo de 1. Si el dividendo y el divisor son del mismo signo, el cociente se deja como número positivo y se le asigna un bit de signo de 0.

6-7 SUMA BCD

En el capítulo 2 mencionamos que muchas computadoras y calculadoras utilizan el código BCD para representar los números decimales. Recuerde que este código toma *cada* dígito decimal y lo representa mediante un código de cuatro bits que varía desde 0000 hasta 1001. La suma de números decimales que están en la forma BCD puede comprenderse mejor si se consideran los dos casos que pueden ocurrir cuando se suman dos dígitos decimales.

La suma es igual a 9 o menor

Considere la suma de 5 y 4 mediante BCD para representar cada dígito:

```
  5      0101   ← BCD para el 5
+4     + 0100   ← BCD para el 4
  9      1001   ← BCD para el 9
```

La adición se lleva a cabo como en la suma binaria normal, lo que nos da un resultado de 1001, que es el código BCD para el 9. Veamos otro ejemplo en el que se suma el 45 al 33:

```
 45      0100 0101   ← BCD para el 45
+33    + 0011 0011   ← BCD para el 33
 78      0111 1000   ← BCD para el 78
```

En este ejemplo los códigos de cuatro bits para el 5 y el 3 se suman en binario para producir el número 1000, que es el BCD para el 8. De manera similar, al sumar las posiciones del segundo dígito decimal se produce 0111, que es el BCD para el 7. El total es 01111000, que es el código BCD para el 78.

En los ejemplos anteriores ninguna de las sumas de los pares de dígitos decimales excede 9; *no se produjeron acarreos decimales*. Para estos casos el proceso de suma con BCD es directo y es en realidad el mismo que la suma binaria.

La suma es mayor de 9

Considere la suma de 6 y 7 en BCD:

```
 6      0110   ← BCD para el 6
+7    + 0111   ← BCD para el 7
+13    1101   ← Código de grupo inválido para el BCD
```

La suma 1101 no existe en el código BCD; éste es uno de los seis grupos de código de cuatro bits prohibidos o inválidos. Se produjo este resultado debido a que la suma de los dos dígitos excede 9. Cada vez que esto ocurra, la suma deberá corregirse sumando seis (0110) para tomar en cuenta la omisión de los seis códigos no válidos:

```
          0110   ← BCD para el 6
        + 0111   ← BCD para el 7
          1101   ← suma inválida
          0110   ← se suman 6 para corregir
        ------
  0001    0011   ← BCD para el 13
    1       3
```

Como se muestra, se suma el 0110 al resultado con código inválido en BCD y se produce el resultado correcto en BCD. Observe que con la suma de 0110 se produce un acarreo en la posición del segundo decimal. Esta suma debe llevarse a cabo siempre que la suma de los dos dígitos decimales sea mayor de 9.

Como otro ejemplo, veamos la suma de 47 y 35 en BCD:

```
 47     0100   0111   ← BCD para el 47
+35   + 0011   0101   ← BCD para el 35
 82     0111   1100   ← suma inválida en el primer dígito
          1←   0110   ← se suman 6 para corregir
       1000   0010   ← BCD para el 97
        8      2
```

La suma de los códigos de cuatro bits para el 7 y el 5 produce un resultado inválido, el cual se corrige al sumar 0110. Observe que esto genera un acarreo de 1, el cual se agrega a la suma BCD de los dígitos de la segunda posición.

Considere la suma de 59 y 38 en BCD:

```
                ↓
                1
 59     0101   1001   ← BCD para el 59
+38   + 0011   1000   ← BCD para el 38
 97     1001   0001   ← realiza la suma
                0110   ← se suman 6 para corregir
              ------
       1001   0111     BCD para el 97
        9      7
```

Aquí la suma de los dígitos menos significativos (LSDs) produce una suma de 17 = 10001. Esto genera un acarreo hacia la posición del siguiente dígito; este acarreo se va a sumar a los códigos para el 5 y el 3. Como 17 > 9, debe agregarse un factor

de corrección de 6 a la suma de los LSDs. Al sumar esta corrección no se genera un acarreo, porque éste ya se hizo en la suma original.

Para resumir el procedimiento de suma BCD:

1. Mediante el uso de la suma binaria ordinaria se suman los grupos de código BCD para la posición de cada dígito.
2. Para las posiciones en las que la suma sea 9 o menos, no se necesita corrección. La suma se encuentra en el formato BCD apropiado.
3. Cuando la suma de dos dígitos es mayor de 9, debe agregarse una corrección de 0110 a esa suma para obtener el resultado BCD apropiado. Este caso siempre produce un acarreo hacia la posición del siguiente dígito, ya sea de la suma original (paso 1) o de la suma de corrección.

Es evidente que el procedimiento para la suma en BCD es más complicado que la suma binaria directa. Esto también es cierto en las demás operaciones aritméticas en BCD. Deberá realizar la suma de 275 + 641. Después compruebe el procedimiento correcto como sigue.

275	0010	0111	0101	← BCD para el 275
+641	+ 0110	0100	0001	← BCD para el 641
916	1000	1011	0110	← Se realiza la suma
	+	0110		← Se suman 6 para corregir el segundo dígito
	1001	0001	0110	← BCD para el 916

Resta BCD

El proceso de restar números en BCD es mucho más difícil que la suma, ya que implica un procedimiento de complementar y después sumar, similar al del método de complemento a 2, por lo que no lo cubriremos en este libro.

PREGUNTAS DE REPASO

1. ¿Cómo podemos saber cuando se necesita una corrección en la suma BCD?
2. Represente los números 135_{10} y 265_{10} en BCD y después realice la suma BCD. Compruebe su trabajo convirtiendo el resultado de vuelta a decimal.

6-8 ARITMÉTICA HEXADECIMAL

Los números hexadecimales se utilizan de manera extensa en la programación de computadoras en lenguaje máquina y en conjunto con las memorias de computadora (es decir, como direcciones). Cuando trabaje en estas áreas se encontrará con situaciones en la que deban sumarse o restarse números hexadecimales.

Suma hexadecimal

La suma de números hexadecimales se realiza en forma muy parecida a la suma decimal, siempre y cuando se tenga en cuenta que el dígito hexadecimal más grande es F en vez de 9. Se sugiere el siguiente procedimiento:

1. Se suman los dos dígitos hexadecimales en decimal, insertando mentalmente el equivalente decimal para los dígitos mayores de 9.

2. Si la suma es de 15 o menor, puede expresarse en forma directa como un dígito hexadecimal.
3. Si la suma es mayor o igual a 16, se restan 16 y se acarrea un 1 hacia la posición del siguiente dígito.

Los siguientes ejemplos ilustrarán este procedimiento.

EJEMPLO 6-6

Sume los números hexadecimales 58 y 24.

Solución

$$\begin{array}{r} 58 \\ +\ 24 \\ \hline 7C \end{array}$$

Al sumar los LSDs (8 y 4) se obtiene un 12, que viene siendo C en hexadecimal. No hay acarreo hacia la posición del siguiente dígito. Al sumar 5 y 2 se produce un 7.

EJEMPLO 6-7

Sume los números hexadecimales 58 y 4B.

Solución

$$\begin{array}{r} 58 \\ +\ 4B \\ \hline A3 \end{array}$$

Se empieza por sumar 8 y B, sustituyendo mentalmente el 11 decimal por B. Esto produce una suma de 19. Como 19 es mayor que 16, se restan 16 y se obtiene un 3; se anota el 3 y se acarrea un 1 hacia la siguiente posición. Este acarreo se suma al 5 y al 4 para producir una suma de 10_{10}, que después se convierte en la A hexadecimal.

EJEMPLO 6-8

Sume 3AF y 23C.

Solución

$$\begin{array}{r} 3AF \\ +\ 23C \\ \hline 5EB \end{array}$$

La suma de F y C se considera como $15 + 12 = 27_{10}$. Como es mayor que 16, se restan 16 para obtener 11_{10}, que es B en hexadecimal, y se acarrea un 1 hacia la segunda posición. Se suma este acarreo con A y con 3 para obtener E. No hay acarreo en la posición MSD.

Resta hexadecimal

Recuerde que los números hexadecimales sólo son una manera eficiente de representar números binarios. De esta forma, podemos restarlos utilizando el mismo método que para los números binarios. Se saca el complemento a 2 del sustraendo hexadecimal y después se *suma* al minuendo; cualquier acarreo fuera de la posición MSD se descarta.

¿Cómo encontramos el complemento a 2 de un número hexadecimal? Una manera es convertirlo en binario, sacar el complemento a 2 del equivalente binario y después convertirlo de vuelta en hexadecimal. Este proceso se ilustra a continuación.

```
          73A                ← número hexadecimal
0111      0011      1010     ← se convierte en binario
1000      1100      0110     ← se saca el complemento a 2
          8C6                ← se convierte de vuelta en hexadecimal
```

Hay un procedimiento más rápido: se resta *cada* dígito hexadecimal de F; después se le suma 1. Probemos esto para el mismo número hexadecimal del ejemplo anterior.

```
  F    F    F
 −7   −3   −A     ← se resta cada dígito de F
  8    C    5
            +1    ← se le suma 1
  8    C    6     ← equivalente hexadecimal del complemento a 2
```

Pruebe cualquiera de los procedimientos anteriores con el número hexadecimal E63. El resultado correcto para el complemento a 2 es 19D.

SUGERENCIA PARA EL USO DE CALCULADORA:

En una calculadora hexadecimal usted puede restar los dígitos hexadecimales a una cadena de Fs y después sumarle uno como lo acabamos de demostrar, o puede sumar 1 a la cadena de todas las Fs y después restar. Por ejemplo, al sumar 1 a FFF_{16} se obtiene 1000_{16}. En la calculadora hexadecimal escriba:

1000 − 73A = La respuesta es 8C6

EJEMPLO 6-9

Reste $3A5_{16}$ de 592_{16}.

Solución

Primero se convierte el sustraendo (3A5) en su forma de complemento a 2, utilizando cualquiera de los métodos antes presentados. El resultado es C5B. Después se suma este resultado al minuendo (592):

```
  592
+ C5B
 11ED
 ↑
 └── Se descarta el acarreo.
```

Se ignora el acarreo de la suma del MSD; el resultado es 1ED. Para demostrar que esto es correcto sumamos 1ED con 3A5 y comprobamos que sea igual a 592_{10}.

Representación hexadecimal de números con signo

Los datos que se almacenan en la memoria funcional interna de una microcomputadora, o en un disco duro o CD-ROM, por lo general, se almacenan en bytes (grupos

TABLA 6-2

Dirección hexadecimal	Datos binarios almacenados	Valor hexadecimal	Valor decimal
4000	00111010	3A	+58
4001	11100101	E5	−29
4002	01010111	57	+87
4003	10000000	80	−128

de ocho bits). Los bytes de datos que se almacenan en una posición de memoria específica se expresan a menudo en hexadecimal, ya que es más eficiente y menos propenso a errores que expresarlo en binario. Cuando los datos consisten de números *con signo*, es útil poder reconocer si un valor hexadecimal representa a un número positivo o negativo. Por ejemplo, la tabla 6-2 lista los datos almacenados en un pequeño segmento de memoria que empieza en la dirección 4000.

Cada posición de memoria almacena un solo byte (ocho bits), que es el equivalente binario de un número decimal con signo. La tabla también muestra el equivalente hexadecimal de cada byte. Para un valor de datos negativo, el bit de signo (MSB) del número binario será 1; esto siempre hará que el MSD del número hexadecimal sea 8 o mayor. Cuando el valor de los datos sea positivo, el bit de signo será un 0 y el MSD del número hexadecimal será de 7 o menor. Lo mismo se aplica sin importar cuántos dígitos tenga el número hexadecimal. *Cuando el MSD es de 8 o mayor, el número que se está representando es negativo; cuando el MSD es de 7 o menor, el número es positivo.*

PREGUNTAS DE REPASO

1. Sume 67F + 2A4.
2. Reste 67F – 2A4.
3. ¿Cuál de los siguientes números hexadecimales representa valores positivos: 2F, 77EC, C000, 6D, FFFF?

6-9 CIRCUITOS ARITMÉTICOS

Una función esencial de la mayoría de las computadoras y calculadoras es el desempeño de las operaciones aritméticas. Estas operaciones se realizan en la unidad aritmética/lógica de una computadora, en donde se combinan compuertas lógicas y flip-flops para poder sumar, restar, multiplicar y dividir números binarios. Estos circuitos realizan operaciones aritméticas a velocidades que no son posibles para los humanos. Por lo general, una operación de suma tarda menos de 100 ns.

Ahora estudiaremos algunos de los circuitos aritméticos básicos que se utilizan para realizar las operaciones aritméticas que vimos antes. En algunos casos veremos el proceso de diseño en sí, aún y cuando los circuitos no estén disponibles a la venta en forma de circuitos integrados, para que obtenga más práctica en relación con el uso de las técnicas que vimos en el capítulo 4.

Unidad aritmética/lógica

Todas las operaciones aritméticas se llevan a cabo en la **unidad aritmética/lógica (ALU)** de una computadora. La figura 6-4 es un diagrama de bloques en el que se muestran los elementos principales que se incluyen en una ALU común. El propósito principal de la ALU es recibir datos binarios que se almacenan en la memoria

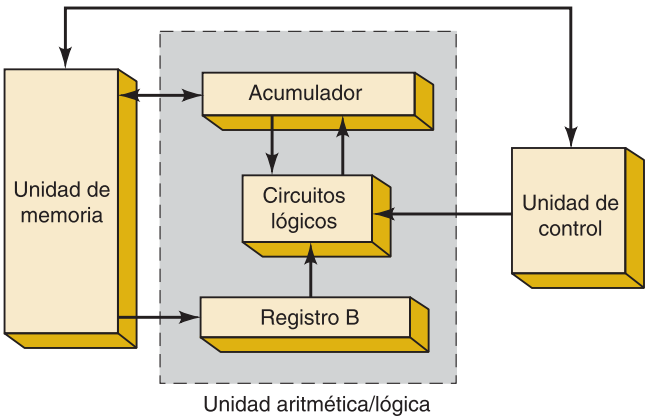


FIGURA 6-4 Partes funcionales de una ALU.

y ejecutar operaciones aritméticas y lógicas sobre estos datos, de acuerdo con las instrucciones provenientes de la unidad de control.

La unidad aritmética/lógica contiene al menos dos registros de flip-flop: el *registro B y el* ***registro acumulador***. También contiene lógica combinacional, la cual realiza las operaciones aritméticas y lógicas sobre los números binarios que se almacenan en el registro *B* y en el acumulador. Una secuencia común de operaciones puede ocurrir de la siguiente manera:

1. La unidad de control recibe una instrucción (de la unidad de memoria) en la que se especifica que un número almacenado en cierta posición de memoria (dirección) se debe sumar al número que se encuentra almacenado en el registro acumulador.
2. El número que se va a sumar se transfiere de la memoria al registro *B*.
3. El número en el registro *B* y el número en el registro acumulador se suman en los circuitos lógicos (cuando lo ordena la unidad de control). La suma resultante se envía al acumulador para que sea almacenada.
4. El nuevo número en el acumulador puede permanecer ahí para que se le pueda sumar otro número, o si se terminó ese proceso aritmético, puede transferirse de vuelta a la memoria para su almacenamiento.

Estos pasos deben dejar en claro de dónde proviene el nombre del registro acumulador. Este registro "acumula" las sumas que ocurren cuando se realizan sumas sucesivas entre nuevos números que se adquieren de memoria y la suma acumulada anterior. De hecho, para cualquier problema aritmético que contenga varios pasos, el acumulador, por lo general, contiene los resultados de los pasos intermedios a medida que se van completando, junto con el resultado final cuando se termina el problema.

6-10 SUMADOR BINARIO EN PARALELO

Las computadoras y calculadoras realizan la operación de la suma sobre dos número binarios a la vez, en donde cada número binario puede tener varios dígitos binarios. La figura 6-5 ilustra el proceso de la suma de dos números de cinco bits. El **primer sumando** se almacena en el registro acumulador; es decir, el acumulador contiene FFs que almacenan los valores 10101 en FFs sucesivos. De manera similar, el **sumando** (el número que se va a sumar al primer sumando) se almacena en el registro *B* (en este caso, 00111).

El proceso de adición empieza por sumar los bits menos significativos (LSBs) del primer sumando y del sumando. Así, 1 + 1 = 10, lo cual significa que la *suma* de esa posición es 0, con un *acarreo* de 1.

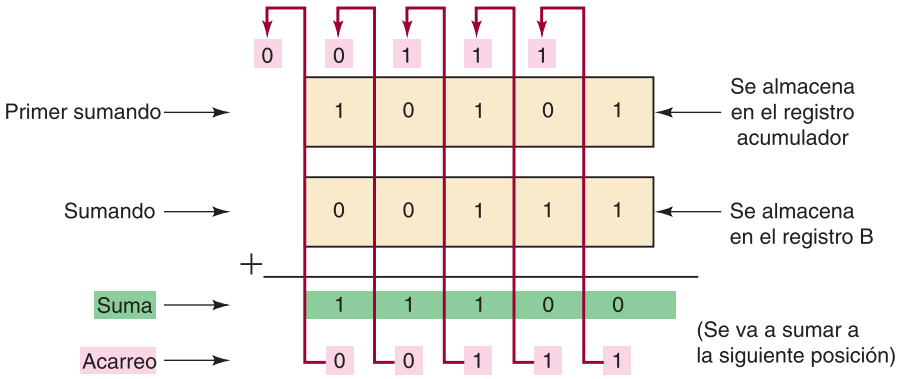


FIGURA 6-5 Proceso común de suma binaria.

Este acarreo debe sumarse a la siguiente posición, junto con los bits del primer sumando y del sumando en esa posición. Así, en la segunda posición 1 + 0 + 1 = 10, lo cual produce de nuevo una suma de 0 y un acarreo de 1. Este acarreo se suma a la siguiente posición junto con los bits del primer sumando y del sumando en esa posición, y así en lo sucesivo para el resto de las posiciones, como se muestra en la figura 6-5.

En cada uno de los pasos de este proceso estamos realizando la suma de tres bits: el bit del primer sumando, el bit del sumando y un bit de acarreo proveniente de la posición anterior. El resultado de la suma de estos tres bits produce dos bits: un bit de *suma* y un bit de *acarreo* que se va a sumar a la siguiente posición. Debe quedar claro que se sigue el mismo proceso para la posición de cada bit. Por lo tanto, si podemos diseñar un circuito lógico que pueda duplicar este proceso entonces todo lo que tenemos que hacer es utilizar el mismo circuito para cada una de las posiciones de los bits. Esto se ilustra en la figura 6-6.

En este diagrama, las variables A_4, A_3, A_2, A_1 y A_0 representan los bits del primer sumando que se almacenan en el acumulador (al cual se le conoce también como registro A). Las variables B_4, B_3, B_2, B_1 y B_0 representan los bits del sumando almacenados en el registro B. Las variables C_4, C_3, C_2, C_1 y C_0 representan los bits de acarreo que van en las posiciones correspondientes. Las variables S_4, S_3, S_2, S_1, S_0 son los bits de salida de la suma para cada posición. Los bits correspondientes del primer sumando y del sumando se alimentan a un circuito lógico conocido como **sumador completo (FA)**, junto con un bit de acarreo proveniente de la posición anterior. Por ejemplo, los bits A_1 y B_1 se alimentan al sumador completo 1 junto con C_1, el

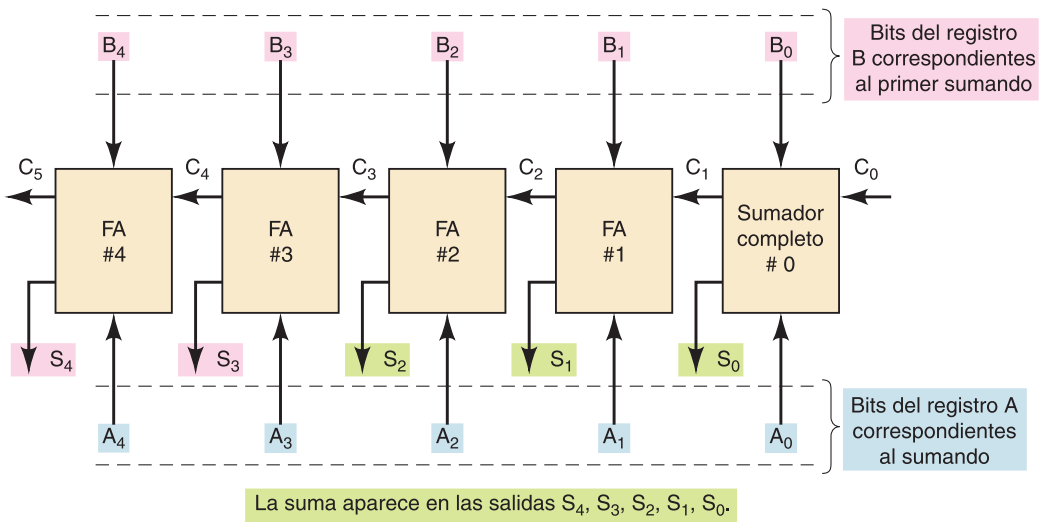


FIGURA 6-6 Diagrama de bloques de un circuito sumador en paralelo de cinco bits, mediante el uso de sumadores completos.

bit de acarreo producido por la suma de los bits A_0 y B_0. Los bits A_0 y B_0 se alimentan al sumador completo 0 junto con C_0. A_0 y B_0 son los LSBs del primer sumando y del sumando, por lo cual parece ser que C_0 siempre tendría que ser 0, ya que no puede haber acarreo hacia esa posición. No obstante, más adelante veremos situaciones en las que C_0 también puede ser 1.

El circuito sumador completo en paralelo que se utiliza en cada posición tiene tres entradas: un bit A, un bit B y un bit C. También produce dos salidas: un bit de suma y un bit de acarreo. Por ejemplo, el sumador completo 0 tiene las entradas A_0, B_0 y C_0, y produce las salidas S_0 y C_1. El sumador completo 1 tiene las entradas A_1, B_1 y C_1, y las salidas S_1 y C_2, y así sucesivamente. Este arreglo se repite para todas las posiciones que tengan el primer sumando y el sumando. Aunque esta ilustración es para números de cinco bits, en las computadoras modernas por lo general la cantidad varía desde 8 hasta 64 bits.

Al arreglo de la figura 6-6 se le conoce como **sumador en paralelo**, ya que todos los bits del primer sumando y del sumando están presentes y se alimentan en los circuitos del sumador *en forma simultánea*. Esto significa que las sumas en cada posición se están llevando a cabo al mismo tiempo. Esto es diferente a la manera en que sumamos en papel, en donde se toma una posición a la vez, empezando con el LSB. Es evidente que la suma en paralelo es en extremo rápida. Hablaremos más sobre esto después.

PREGUNTAS DE REPASO

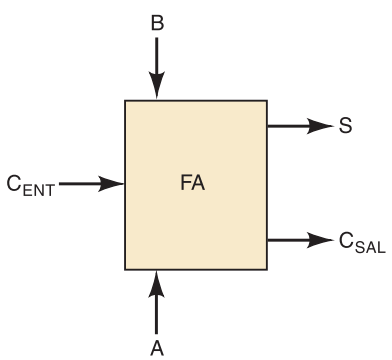
1. ¿Cuántas entradas tiene un sumador completo? ¿Cuántas salidas lo tienen?
2. Suponga los siguientes niveles de entrada en la figura 6-6: $A_4A_3A_2A_1A_0 = 01001$; $B_4B_3B_2B_1B_0 = 00111$; $C_0 = 0$.
   (a) ¿Cuáles son los niveles lógicos en las salidas del FA # 2?
   (b) ¿Cuál es el nivel lógico en la salida C_5?

6-11 DISEÑO DE UN SUMADOR COMPLETO

Ahora que conocemos la función del sumador completo, podemos diseñar un circuito lógico que desarrolle esta función. Primero debemos construir una tabla de verdad que muestre los valores de las diversas entradas y salidas para todos los posibles casos. La figura 6-7 muestra la tabla de verdad con tres entradas (A, B y C_{ENT}) y dos salidas (S y C_{SAL}). Hay ocho posibles casos para las tres entradas; en cada caso

FIGURA 6-7 Tabla de verdad para un circuito sumador completo.

Bit de entrada del primer sumando	Bit de entrada del sumando	Bit de entrada del acarreo	Bit de salida de la suma	Bit de salida del acarreo
A	B	C_{ENT}	S	C_{SAL}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



se listan los valores de salida deseados. Por ejemplo, considere el caso $A = 1, B = 0$ y $C_{ENT} = 1$. El sumador completo (FA) debe sumar estos bits para producir una suma (S) de 0 y un acarreo (C_{SAL}) de 1. El lector deberá comprobar los demás casos para asegurarse de comprenderlos.

Como hay dos salidas, diseñaremos los circuitos para cada salida de manera individual, empezando con la salida S. La tabla de verdad muestra que existen cuatro casos en los que S debe ser 1. Si utilizamos el método de suma de productos podemos escribir la expresión para S como

$$S = \overline{A}\,\overline{B}C_{ENT} + \overline{A}B\overline{C}_{ENT} + A\overline{B}\,\overline{C}_{ENT} + ABC_{ENT} \tag{6-1}$$

Ahora podemos tratar de simplificar esta expresión por medio de la factorización. Desafortunadamente ninguno de los términos en la expresión tiene dos variables en común con alguno de los otros términos. No obstante, podemos factorizar A de los primeros dos términos, y $\overline{A}$ de los dos últimos:

$$S = \overline{A}(\overline{B}C_{ENT} + B\overline{C}_{ENT}) + A(\overline{B}\,\overline{C}_{ENT} + BC_{ENT})$$

El primer término entre paréntesis debe reconocerse como la combinación del OR exclusivo entre B y C_{ENT}, la cual puede escribirse como $B \oplus C_{ENT}$. El segundo término entre paréntesis debe reconocerse como el NOR exclusivo entre B y C_{ENT}, que puede escribirse como $\overline{B \oplus C_{ENT}}$. Así, la expresión para S se convierte en

$$S = \overline{A}(B \oplus C_{ENT}) + A(\overline{B \oplus C_{ENT}})$$

Si dejamos que $X = B \oplus C_{ENT}$, podemos escribir esto como

$$S = \overline{A} \cdot X + A \cdot \overline{X} = A \oplus X$$

que es tan sólo el OR exclusivo entre A y X. Si sustituimos la expresión para X tenemos que

$$S = A \oplus [B \oplus C_{ENT}] \tag{6-2}$$

Considere ahora la salida C_{SAL} en la tabla de verdad de la figura 6-7. Podemos escribir la expresión de suma de productos para C_{SAL} de la siguiente manera:

$$C_{SAL} = \overline{A}BC_{ENT} + A\overline{B}C_{ENT} + AB\overline{C}_{ENT} + ABC_{ENT}.$$

Podemos simplificar esta expresión a través de la factorización. Emplearemos el truco que presentamos en el capítulo 4, en el cual utilizaremos el término ABC_{ENT} *tres* veces debido a que tiene factores en común con cada uno de los otros términos. Así:

$$\begin{aligned} C_{SAL} &= BC_{ENT}(\overline{A} + A) + AC_{ENT}(\overline{B} + B) + AB(\overline{C}_{ENT} + C_{ENT}) \\ &= BC_{ENT} + AC_{ENT} + AB \end{aligned} \tag{6-3}$$

Esta expresión ya no puede simplificarse más.

Las expresiones (6-2) y (6-3) pueden implementarse como se muestra en la figura 6-8. Pueden utilizarse otras implementaciones para producir las mismas expresiones para S y C_{SAL}, ninguna de las cuales tiene una ventaja específica sobre las que acabamos de mostrar. El circuito completo con las entradas A, B, C_{ENT} y las salidas S y C_{SAL} representa el sumador completo. Cada una de las FAs en la figura 6-6 contiene estos mismos circuitos (o su equivalente).

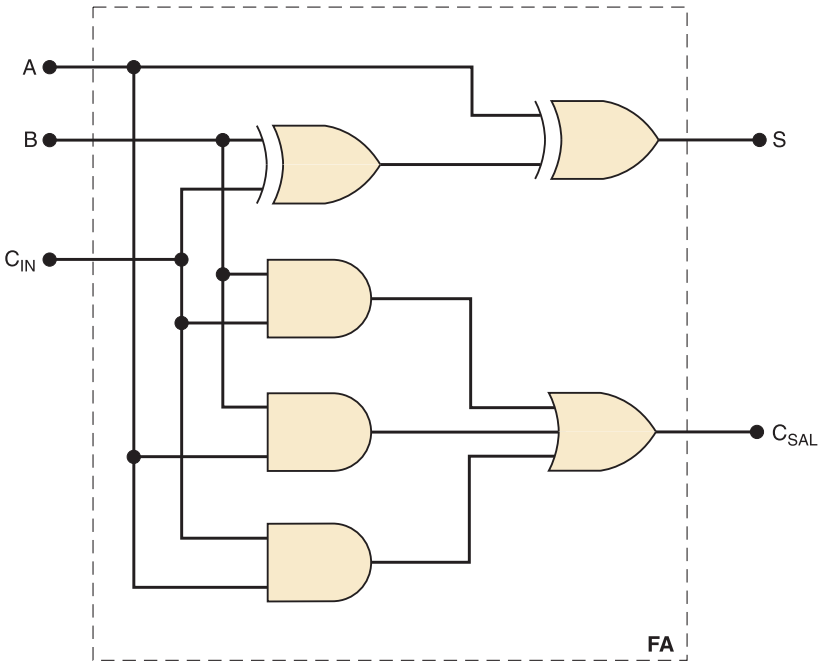


FIGURA 6-8 Circuitos completos para un sumador completo.

Simplificación mediante el mapa K

Ya simplificamos las expresiones para S y C_{SAL} mediante el uso de los métodos algebraicos. También puede usarse el método del mapa K. La figura 6-9(a) muestra el mapa K para la salida S. Este mapa no tiene 1s adyacentes por lo que no hay pares o cuádruples que agregar. En consecuencia no se puede simplificar la expresión para S mediante el mapa K. Esto nos muestra una limitación del método del mapa K en comparación con el método algebraico. Pudimos simplificar la expresión para S por medio de la factorización y del uso de las operaciones XOR y XNOR.

El mapa K para la salida C_{SAL} se muestra en la figura 6-9(b). Los tres pares que se agrupan producirán la misma expresión que se obtuvo mediante el método algebraico.

Medio sumador

El FA opera sobre tres entradas para producir una salida de suma y de acarreo. En algunos casos se requiere de un circuito que sume sólo dos bits de entrada sin aca-

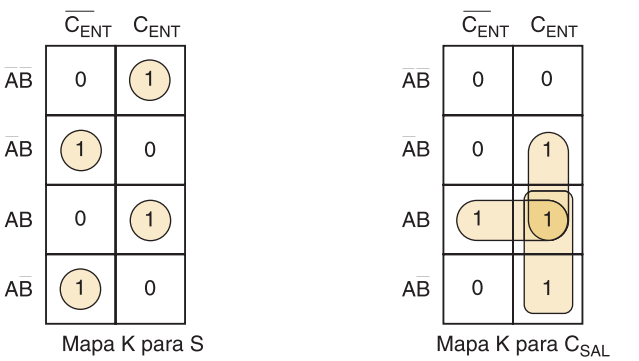


FIGURA 6-9 Mapas K para las salidas del sumador completo.

$S = \overline{A}\overline{B}C_{ENT} + \overline{A}B\overline{C}_{ENT} + AB C_{ENT} + A\overline{B}\overline{C}_{ENT}$

(a)

$C_{SAL} = BC_{ENT} + AC_{IN} + AB$

(b)

rreo para producir una salida de suma y de acarreo. Un ejemplo sería la suma de la posición LSB de dos números binarios en donde no haya un acarreo que sumar. Puede diseñarse un circuito lógico especial para tomar *dos* bits de entrada (A y B) y producir las salidas de suma (S) y acarreo (C_{SAL}). A este circuito se le conoce como **medio sumador (HA)**. Su operación es similar a la de un FA, excepto que opera sólo con dos bits. Dejaremos el diseño del HA como un ejercicio al final del capítulo.

6-12 SUMADOR COMPLETO EN PARALELO CON REGISTROS

En una computadora, los números que se van a sumar se almacenan en registros de FF. La figura 6-10 muestra el diagrama de un sumador en paralelo de cuatro bits, incluyendo los registros de almacenamiento. Los bits del primer sumando ($A_3 - A_0$) se almacenan en el acumulador (registro A); los bits del sumando ($B_3 - B_0$) se almacenan en el registro B. Cada uno de estos registros está compuesto de flip-flops D para facilitar la transferencia de los datos.

El contenido del registro A (es decir, el número binario almacenado en $A_3 - A_0$) se suma al contenido del registro B a través de los cuatro FAs y la suma se produce en las salidas $S_3 - S_0$. C_4 es el acarreo del cuarto FA y puede usarse como la entrada de acarreo para un quinto FA, o como un bit de *desbordamiento* para indicar que la suma se excedió de 1111.

Observe que las salidas de la suma están conectadas a las entradas D del registro A. Esto permitirá que la suma se transfiera en paralelo hacia el registro A en la transición de tendencia positiva (PGT) del pulso TRANSFERIR. De esta forma, la suma puede almacenarse en el registro A.

Observe además que las entradas D del registro B vienen de la memoria de la computadora, por lo que los números binarios de la memoria se transferirán en paralelo hacia el registro B en la PGT del pulso CARGAR. En la mayoría de las computadoras también se tiene una provisión para transferir en paralelo números binarios de la memoria hacia el acumulador (registro A). Por cuestión de simplicidad no se muestran en este diagrama los circuitos necesarios para realizar esta transferencia; veremos esto en un ejercicio al final del capítulo.

Por último, observe que las salidas del registro A están disponibles para realizar transferencias hacia otras ubicaciones tales como otro registro de la computadora o la memoria. Esto hará que el circuito sumador esté disponible para un nuevo conjunto de números.

Notación de registro

Antes de ver todo el proceso completo de cómo este circuito suma dos números binarios será de utilidad introducir cierta notación que facilita la descripción del contenido de un registro y las operaciones de transferencia de datos.

Cuando queramos proporcionar los niveles que están presentes en cada FF de un registro, o en cada salida de un grupo de salidas, utilizaremos corchetes, como se muestra a continuación:

$$[A] = 1011$$

Esto es lo mismo que decir que $A_3 = 1, A_2 = 0, A_1 = 1, A_0 = 1$. En otras palabras, piense en que $[A]$ representa "el contenido del registro A".

Cuando queramos indicar la transferencia de datos hacia o desde un registro utilizaremos una flecha, como se muestra a continuación:

$$[B] \rightarrow [A]$$

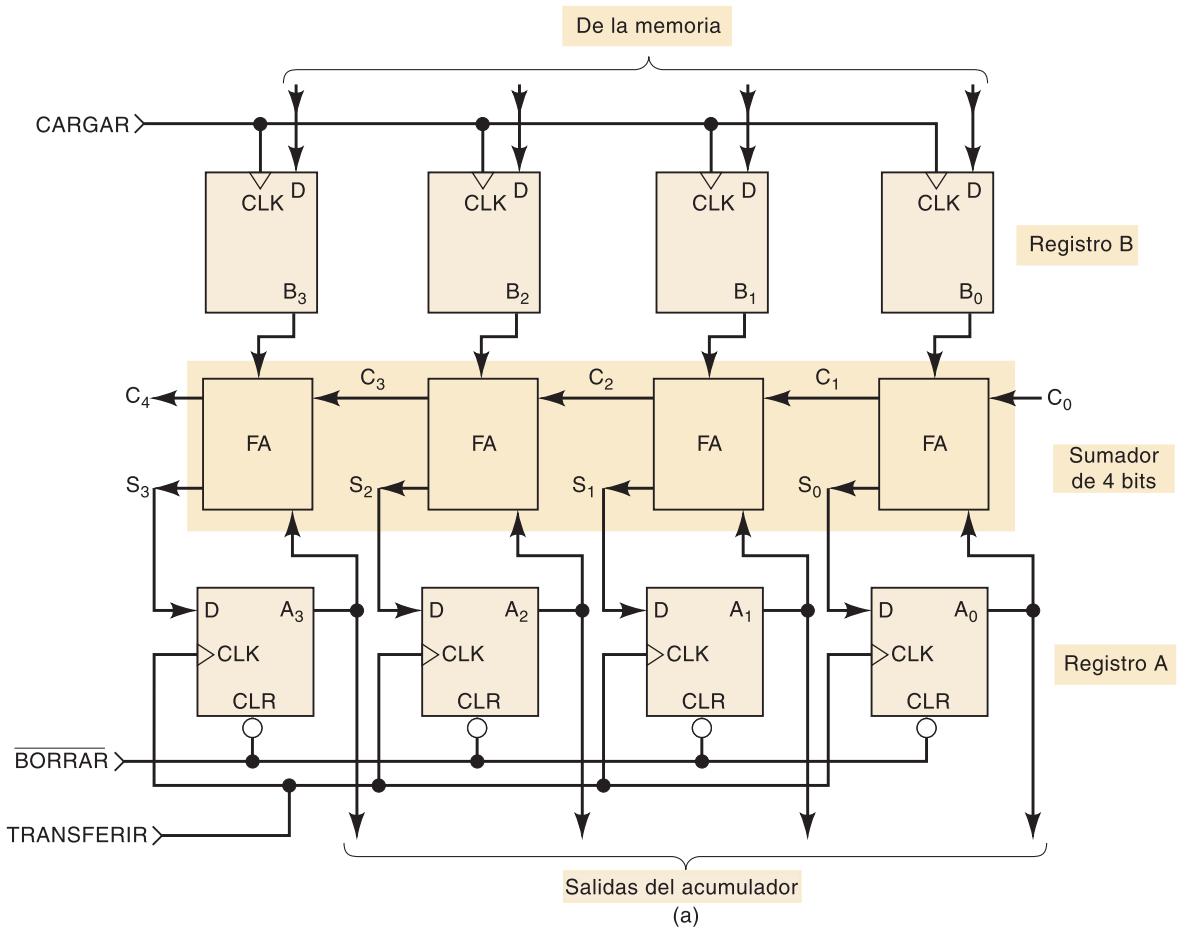


FIGURA 6-10 (a) Sumador completo en paralelo de cuatro bits con registros; (b) señales utilizadas para sumar números binarios de la memoria y almacenar su suma en el acumulador.

Esto significa que el contenido del registro *B* se ha transferido al registro *A*. El contenido anterior del registro *A* se perderá como resultado de esta operación y el registro *B* quedará sin cambio. Este tipo de notación es muy común, en especial en los libros de datos que describen las operaciones de microprocesadores y microcontroladores. En muchas formas es muy similar a la notación que se utiliza para hacer referencia a los objetos de datos de un arreglo de bits mediante el uso de lenguajes de descripción de hardware.

Secuencia de operaciones

Ahora describiremos el proceso mediante el cual el circuito de la figura 6-10 sumará los números binarios 1001 y 0101. Suponga que $C_0 = 0$; es decir, que no hay acarreo hacia la posición del LSB.

1. $[A] = 0000$. Se aplica un pulso $\overline{\text{BORRAR}}$ a las entradas asíncronas ($\overline{CLR}$) de cada FF en el registro A. Esto ocurre en el tiempo t_1.
2. $[M] \rightarrow [B]$. Este primer número binario se transfiere desde la memoria (M) hasta el registro B. En este caso se carga el número binario 1001 en el registro B durante la PGT del pulso CARGAR en t_2.
3. $[S]^* \rightarrow [A]$. Cuando $[B] = 1001$ y $[A] = 0000$, los sumadores completos producen una suma de 1001; esto es, $[S] = 1001$. Estas salidas de suma se transfieren hacia el registro A en la PGT del pulso TRANSFERIR en t_3. Esto hace que $[A] = 1001$.
4. $[M] \rightarrow [B]$. El segundo número binario 0101 se transfiere desde la memoria hacia el registro B en la PGT del segundo pulso CARGAR en t_4. Esto hace que $[B] = 0101$.
5. $[S] \rightarrow [A]$. Cuando $[B] = 0101$ y $[A] = 1001$, los FAs producen $[S] = 1110$. Estas salidas de suma se transfieren hacia el registro A cuando ocurre el segundo pulso TRANSFERIR en t_5. Por lo tanto, $[A] = 1110$.
6. En este punto, la suma de los dos números binarios está presente en el acumulador. Enla mayoría de las computadoras, por lo general, el contenido del acumulador $[A]$ se transferirá a la memoria de la computadora, de manera que el circuito sumador pueda utilizarse para un nuevo conjunto de números. Los circuitos que realizan esta transferencia $[A] \rightarrow [M]$ no se muestran en la figura 6-10.

PREGUNTAS DE REPASO

1. Suponga que se van a tomar cuatro números distintos de cuatro bits de la memoria y que se van a sumar mediante el circuito de la figura 6-10. ¿Cuántos pulsos BORRAR se necesitarán? ¿Cuántos pulsos TRANSFERIR? ¿Cuántos pulsos CARGAR?
2. Determine el contenido del registro A después de la siguiente secuencia de operaciones: $[A] = 0000$, $[0110] \rightarrow [B]$, $[S] \rightarrow [A]$, $[1110] \rightarrow [B]$, $[S] \rightarrow [A]$.

6-13 PROPAGACIÓN DEL ACARREO

El sumador en paralelo de la figura 6-10 realiza las sumas a una velocidad relativamente alta, debido a que suma los bits de cada posición al mismo tiempo. No obstante, su velocidad se ve limitada por un efecto llamado **propagación del acarreo** o **rizo del acarreo**, el cual puede explicarse mejor si consideramos la siguiente suma:

$$\begin{array}{rr} & 0111 \\ + & 0001 \\ \hline & 1000 \end{array}$$

La suma de la posición del LSB produce un acarreo hacia la segunda posición. Cuando este acarreo se suma a los bits de la segunda posición, produce un acarreo

* Aún y cuando S no es un registro, utilizaremos $[S]$ para representar el grupo de salidas S.

hacia la tercera posición. Cuando este último acarreo se suma a los bits de la tercera posición, produce un acarreo hacia la última posición. El punto clave a observar en este ejemplo es que el bit de suma que se genera en la *última* posición (MSB) depende del acarreo generado por la suma en la *primera* posición (LSB).

Si analizamos esto desde el punto de vista del circuito de la figura 6-10, la salida S_3 del último sumador completo depende de la salida C_1 del primer sumador completo. Pero la señal C_1 debe pasar a través de tres FAs para poder producir S_3. Lo que esto significa es que la salida S_3 no llegará a su valor correcto sino hasta que C_1 se haya propagado a través de los FAs intermedios. Esto representa un retraso de tiempo que depende del retraso de propagación que se produce en cada FA. Por ejemplo, si cada FA tiene un retraso de propagación de 40 ns, entonces S_3 no llegará a su nivel correcto sino hasta 120 ns después de que se genere C_1. Esto significa que el pulso del comando de suma no puede aplicarse sino hasta 160 ns después de que los números del primer sumando y del sumando estén presentes en los registros de FF (los 40 ns adicionales se deben al retraso del sumador completo del LSB, el cual genera a C_1).

Es obvio que la situación se empeora mucho más si extendemos el circuito sumador para que sume un mayor número de bits. Si el sumador manejara números de 32 bits, el retraso de propagación del acarreo podría ser de 1280 ns = 1.28 μs después de que los números hayan estado presentes en los registros.

Esta magnitud de retraso es algo que no puede permitirse en las computadoras de alta velocidad. Por fortuna los diseñadores lógicos han ideado varios esquemas ingeniosos para reducir este retraso. Uno de los esquemas, conocido como **acarreo anticipado**, utiliza compuertas lógicas para analizar los bits de menor orden del primer sumando y del sumando, para ver si se va a generar un acarreo de mayor orden. Por ejemplo, es posible construir un circuito lógico con B_2, B_1, B_0, A_2, A_1 y A_0 como entradas y C_3 como salida. Este circuito lógico tendría un retraso más corto del que se obtiene mediante la propagación del acarreo a través de los FAs. Este esquema requiere de una extensa cantidad de circuitos adicionales, pero es necesario para producir sumadores de alta velocidad. Los circuitos extras no son una consideración importante gracias al uso actual de los circuitos integrados. Muchos sumadores de alta velocidad disponibles en forma de circuitos integrados utilizan el acarreo adelantado o una técnica similar para reducir los retrasos de propagación en general.

6-14 SUMADOR EN PARALELO DE CIRCUITO INTEGRADO

Hay varios sumadores en paralelo disponibles como CIs. El más común es un CI de sumador en paralelo de cuatro bits, el cual contiene cuatro FAs interconectados y los circuitos de acarreo adelantado necesarios para la operación de alta velocidad. Los chips 7483A, 74LS83A, 74LS283 y 74HC283 son todos sumadores en paralelo de cuatro bits.

La figura 6-11(a) muestra el símbolo funcional para el sumador en paralelo de cuatro bits 74HC283 (y sus equivalentes). Las entradas para este CI son dos números de cuatro bits ($A_3A_2A_1A_0$ y $B_3B_2B_1B_0$) y el acarreo C_0 hacia la posición del LSB. Las salidas son los bits de suma y el acarreo C_4 que proviene de la posición del MSB. Los bits de suma se etiquetan como $\Sigma_3\Sigma_2\Sigma_1\Sigma_0$, en donde Σ es la letra mayúscula griega *sigma*. La etiqueta Σ es sólo una alternativa común para la etiqueta S, para un bit de suma.

Sumadores en paralelo en cascada

Podemos conectar dos o más sumadores de CI juntos (en cascada) para lograr la suma de números binarios más grandes. La figura 6-11(b) muestra dos sumadores 74HC283 conectados para sumar dos números de 8 bits: $A_7A_6A_5A_4A_3A_2A_1A_0$ y $B_7B_6B_5B_4B_3B_2B_1B_0$. El sumador de la derecha suma los bits de menor orden de los números. El sumador de la izquierda suma los bits de mayor orden *más* el acarreo C_4 que proviene del sumador de menor orden. Las ocho salidas de suma son la suma resultante de los dos números de 8 bits. C_8 es el acarreo que proviene de la posición

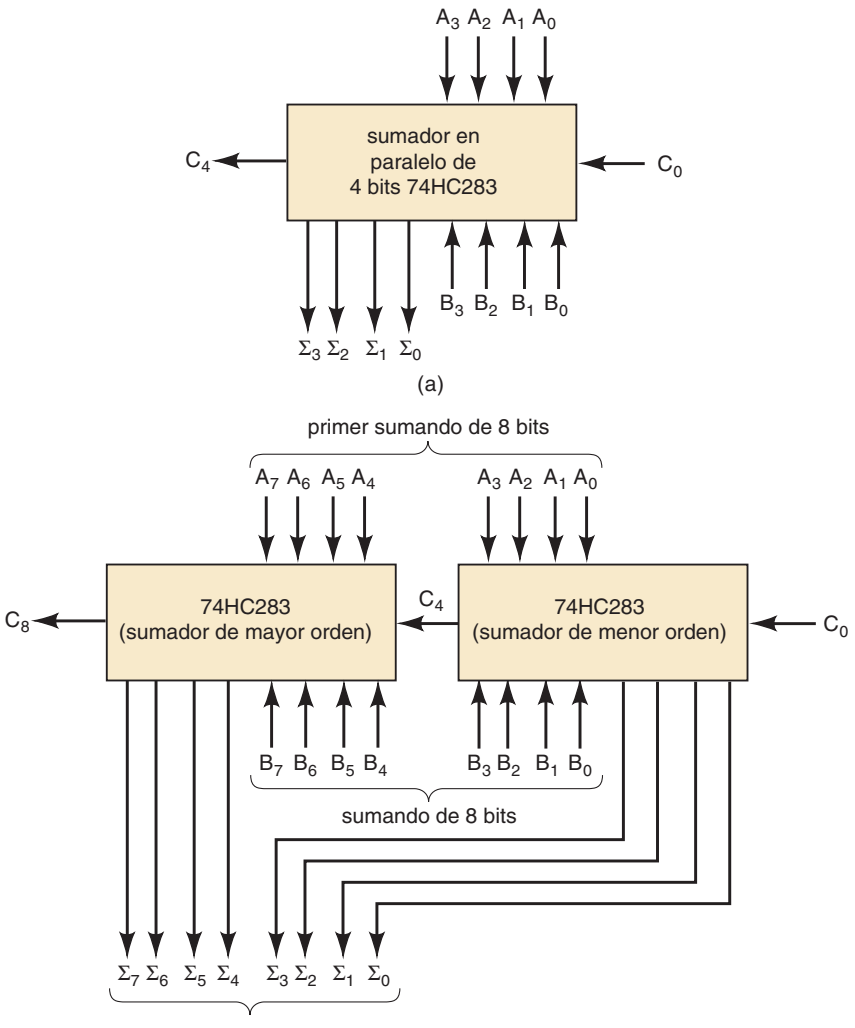


FIGURA 6-11
(a) Símbolo de bloque para el sumador en paralelo de cuatro bits 74HC283; (b) conexión en cascada de dos CIs 74HC283.

del MSB. Puede usarse como la entrada de acarreo para una tercera etapa de sumador, si se van a sumar números binarios más grandes.

La característica de acarreo adelantado del 74HC283 agiliza la operación de este sumador de dos etapas, ya que el nivel lógico en C_4 (el acarreo de la etapa de menor orden) se genera con más rapidez que si no hubiera circuitos de acarreo adelantado en el chip 74HC283. Esto permite que la etapa de mayor orden produzca sus salidas de suma con más rapidez.

EJEMPLO 6-10

Determine los niveles lógicos en las entradas y las salidas del sumador de ocho bits en la figura 6-11(b), cuando se suma el 72_{10} con el 137_{10}.

Solución

Primero se convierte cada número en binario de ocho bits:

$$137 = 10001001$$
$$72 = 01001000$$

Estos dos valores binarios se aplicarán a las entradas A y B; esto es, las entradas de A serán 10001001 de izquierda a derecha, y las entradas de B serán 01001000 de izquierda a derecha. El sumador producirá la suma binaria de los dos números:

$$\begin{aligned}[A] &= 10001001\\ [B] &= \underline{01001000}\\ [\Sigma] &= 11010001\end{aligned}$$

Las salidas de la suma indicarán 11010001 de izquierda a derecha. Como no hay desbordamiento hacia el bit C_8, quedará como un 0.

PREGUNTAS DE REPASO

1. ¿Cuántos chips 74HC283 se necesitan para sumar dos números de 20 bits?
2. Si un 74HC283 tiene un retraso de propagación máximo de 30 ns de C_0 a C_4, ¿cuál será el retraso de propagación total de un sumador de 32 bits construido a partir de chips 74HC283?
3. ¿Cuál será el nivel lógico en C_4, en el ejemplo 6-10?

6-15 SISTEMA DE COMPLEMENTO A 2

La mayoría de las computadoras modernas utiliza el sistema de complemento a 2 para representar números negativos y realizar restas. Si utilizamos la forma de complemento a 2 para representar los números negativos sólo será necesario la operación de suma para realizar las operaciones tanto de suma como de resta de números con signo.

Suma

Los números positivos y negativos (incluyendo los bits de signo) pueden sumarse en el circuito sumador en paralelo básico cuando los números negativos se encuentran en forma de complemento a 2. Esto se ilustra en la figura 6-12 para la suma de -3 y $+6$. El -3 se representa en su forma de complemento a 2 como 1101, en donde el primer 1 es el bit de signo; el $+6$ se representa como 0110, en donde el primer cero es el bit de signo. Estos números se almacenan en sus correspondientes registros. El sumador en paralelo de cuatro bits produce salidas de suma de 0011, lo cual representa un $+3$. La salida C_4 es 1, pero recuerde que en el método de complemento a 2 se descarta.

Resta

Cuando se utiliza el sistema de complemento a 2, el número a restar (el sustraendo) se cambia por su complemento a 2 y después se *suma* al minuendo (el número al que se le está restando el sustraendo). Por ejemplo, podemos suponer que el minuendo ya está almacenado en el acumulador (registro A). Después el sustraendo se coloca en el registro B (en una computadora se transferiría a este registro desde la memoria) y se cambia por su forma de complemento a 2 antes de sumarlo al número que está en el registro A. Las salidas de suma del circuito sumador ahora representan la *diferencia* entre el minuendo y el sustraendo.

El circuito sumador en paralelo del cual hemos estado hablando puede adaptarse para realizar la resta que se describe a continuación si proporcionamos un medio para sacar el complemento a 2 del número en el registro B. Para obtener el

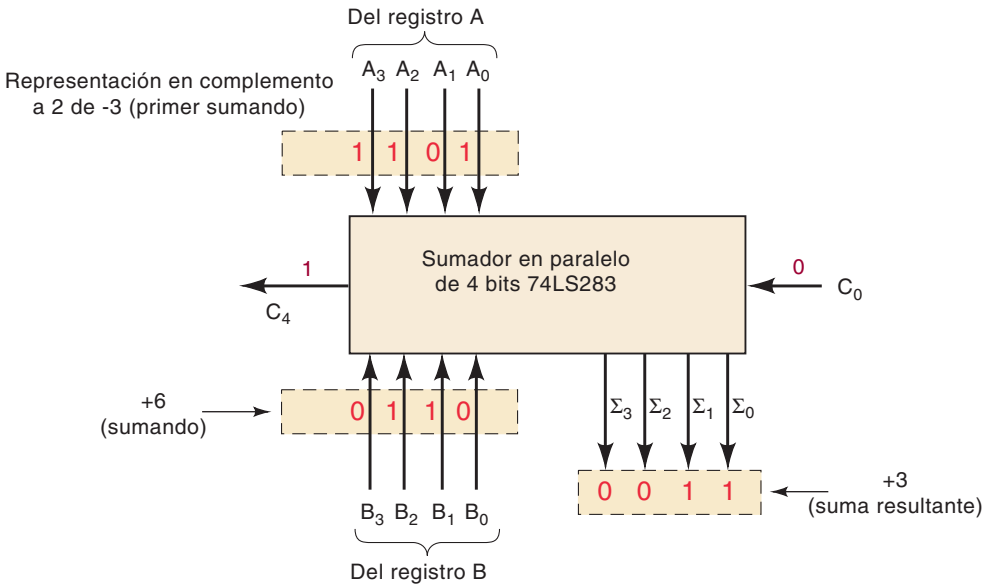


FIGURA 6-12 Sumador en paralelo que se utiliza para sumar y restar números en el sistema de complemento a 2.

complemento a 2 de un número binario se complementa (invierte) cada bit y después se suma 1 al LSB. La figura 6-13 muestra cómo lograrlo. Se utilizan las salidas *invertidas* del registro B en vez de las salidas normales; esto es, $\overline{B}_0$, $\overline{B}_1$, $\overline{B}_2$ y $\overline{B}_3$ se alimentan a las entradas del sumador (recuerde que B_3 es el bit de signo). Con esta operación se complementa cada uno de los bits del número en B. Además C_0 se hace 1 lógico, por lo que se suma un 1 adicional al LSB del sumador; esto logra el mismo efecto que si se suma 1 al LSB del registro B para formar el complemento a 2.

Las salidas Σ_3 a Σ_0 representan los resultados de la operación de resta. Desde luego que Σ_3 es el bit de signo del resultado e indica si éste es + o −. La salida de acarreo C_4 se descarta de nuevo.

Para aclarar mejor esta operación, estudie los siguientes pasos para restar +6 de +4:

1. El +4 se almacena en el registro A como 0100.
2. El +6 se almacena en el registro B como 0110.
3. Las salidas invertidas de los FFs del registro B (1001) se alimentan al sumador.

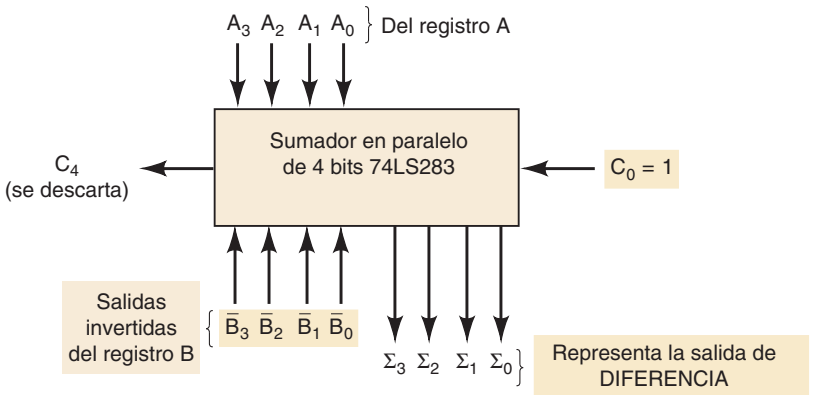


FIGURA 6-13 Sumador en paralelo que se utiliza para restar $(A - B)$ mediante el sistema de complemento a 2. Los bits del sustraendo (B) están invertidos, y $C_0 = 1$ para producir el complemento a 2.

4. Los circuitos del sumador en paralelo suman $[A] = 0100$ con $[\overline{B}] = 1001$, junto con un acarreo $C_0 = 1$ hacia el LSB. La operación se muestra a continuación:

$$\begin{array}{rl} 1 & \leftarrow C_0 \\ 0100 & \leftarrow [A] \\ +\ \underline{1001} & \leftarrow [\,\overline{B}\,] \\ 1110 & \leftarrow [\Sigma] = [A] - [B] \end{array}$$

El resultado en las salidas de suma es 1110. En realidad esto representa el resultado de la operación de *resta*, la *diferencia* entre el número en el registro A y el número en el registro B; esto es, $[A] - [B]$. Como el bit de signo = 1, es un resultado negativo y se encuentra en la forma de complemento a 2. Para verificar que 1110 representa el -2_{10} sacamos su complemento a 2 y obtenemos $+2_{10}$:

$$\begin{array}{rl} & 1110 \\ & 0001 \\ + & \underline{\quad 1} \\ & 0010 = +2_{10} \end{array}$$

Combinación de suma y resta

Ahora debe quedar claro que el circuito sumador en paralelo básico puede usarse para sumar o restar, dependiendo de si el número B se deja sin cambio o se convierte en su complemento a 2. En la figura 6-14 se muestra un circuito completo que puede realizar *tanto* sumas como restas en el sistema de complemento a 2.

Este circuito **sumador/restador** se controla mediante las dos señales de control SUMA y RESTA. Cuando el nivel de SUMA es ALTO, el circuito suma los números almacenados en los registros A y B. Cuando el nivel RESTA es ALTO, el circuito resta el número en el registro B del número en el registro A. La operación se describe de la siguiente manera:

1. Suponga que SUMA = 1 y que RESTA = 0. La condición RESTA = 0 *deshabilita* (inhibe) las compuertas AND 2, 4, 6 y 8, con lo que sus salidas se mantienen en 0. La condición SUMA = 1 *habilita* las compuertas AND 1, 3, 5 y 7, con lo cual se permite que sus salidas pasen los niveles B_0, B_1, B_2 y B_3 en forma respectiva.
2. Los niveles B_0 a B_3 pasan a través de las compuertas OR en el sumador en paralelo de cuatro bits, para que se sumen a los bits $A_0 - A_3$. La *suma* aparece en las salidas Σ_0 a Σ_3.
3. Observe que RESTA = 0 produce un acarreo de $C_0 = 0$ en el sumador.
4. Ahora suponga que SUMA = 0 y RESTA = 1. La condición SUMA = 0 inhibe las compuertas AND 1, 3, 5 y 7. La condición RESTA = 1 habilita las compuertas 2, 4, 6 y 8, por lo que sus salidas pasan los niveles $\overline{B}_0$, $\overline{B}_1$, $\overline{B}_2$ y $\overline{B}_3$ respectivamente.
5. Los niveles $\overline{B}_0$ a $\overline{B}_3$ pasan a través de las compuertas OR hacia el sumador, para agregarse a los bits $A_0 - A_3$. Observe también que C_0 ahora es 1. Por lo tanto, el número en el registro B se ha convertido en su complemento a 2.
6. La *diferencia* aparece en las salidas Σ_0 a Σ_3.

Los circuitos como el sumador/restador de la figura 6-14 se utilizan en computadoras debido a que proporcionan un medio relativamente sencillo para sumar y restar números binarios con signo. En la mayoría de las computadoras, las salidas presentes en las líneas de salida Σ por lo general se transfieren hacia el registro A (acumulador), por lo que los resultados de la suma o la resta siempre terminan almacenados en el registro A. Esto se logra aplicando un pulso TRANSFERIR a las entradas *CLK* del registro A.

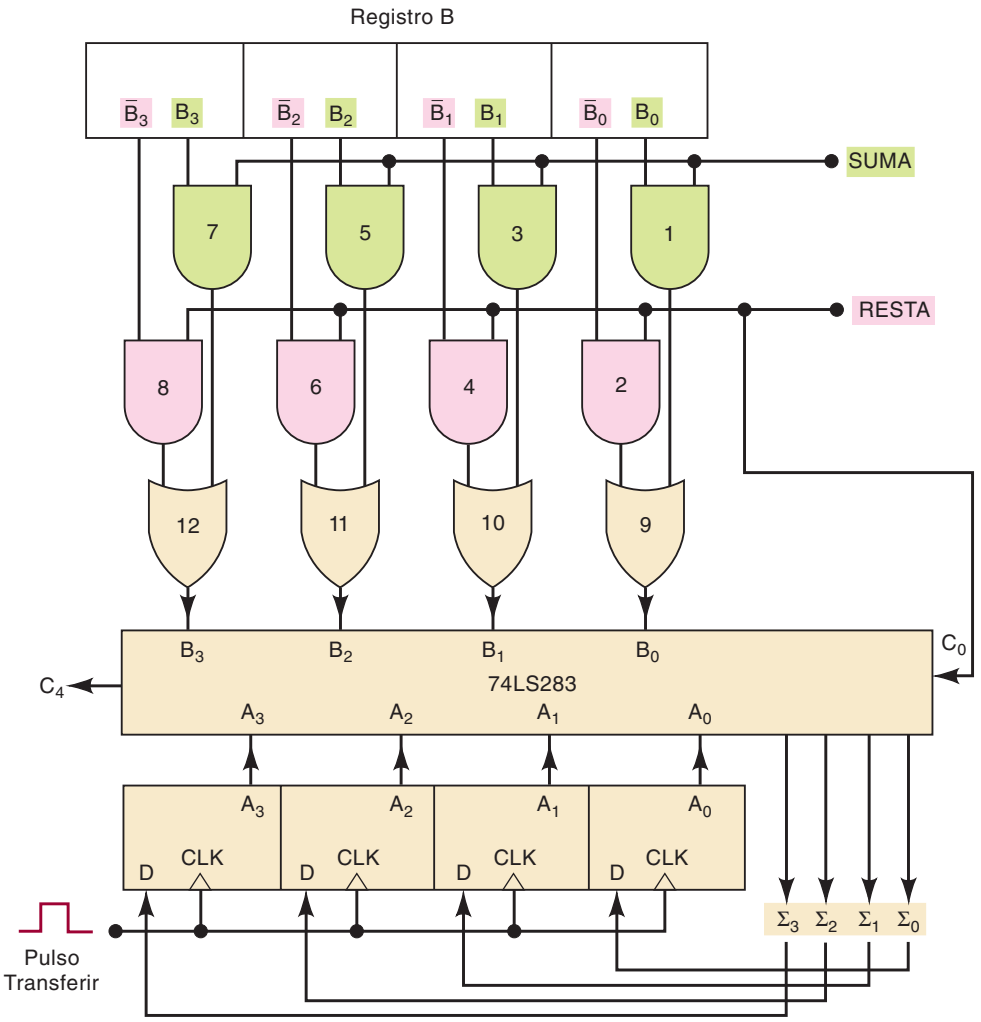


FIGURA 6-14
Sumador/restador en paralelo que utiliza el sistema de complemento a 2.

PREGUNTAS DE REPASO

1. ¿Por qué C_0 tiene que ser 1 para poder utilizar el circuito sumador de la figura 6-13 como restador?
2. Suponga que $[A] = 0011$ y $[B] = 0010$ en la figura 6-14. Si SUMA = 1 y RESTA = 0, determine los niveles lógicos en las salidas de la compuerta OR.
3. Repita la pregunta 2 para la condición SUMA = 0, RESTA = 1.
4. *Verdadero o falso:* cuando el circuito sumador/restador se utiliza para restar, el complemento a 2 del sustraendo aparece en la entrada del sumador.

6-16 CIRCUITOS INTEGRADOS TIPO ALU

Existen varios circuitos integrados a los que se les conoce como unidades aritmético/lógicas (ALUs), aún cuando no tienen todas las herramientas de una unidad aritmética/lógica de computadora. Estos chips ALU son capaces de realizar varias operaciones aritméticas y lógicas con entradas de datos binarios. La operación específica que debe realizar un CI tipo ALU se determina con base en un código binario específico que se aplica en sus entradas de selección de función. Algunos de los CIs tipo ALU son bastante complejos y se requeriría una gran cantidad de tiempo y espacio para explicar e ilustrar su operación. En esta sección utilizaremos un chip tipo ALU

relativamente simple pero útil, para demostrar los conceptos básicos detrás de todos los chips tipo ALU. Las ideas que presentamos aquí podrán extenderse después a los dispositivos más complejos.

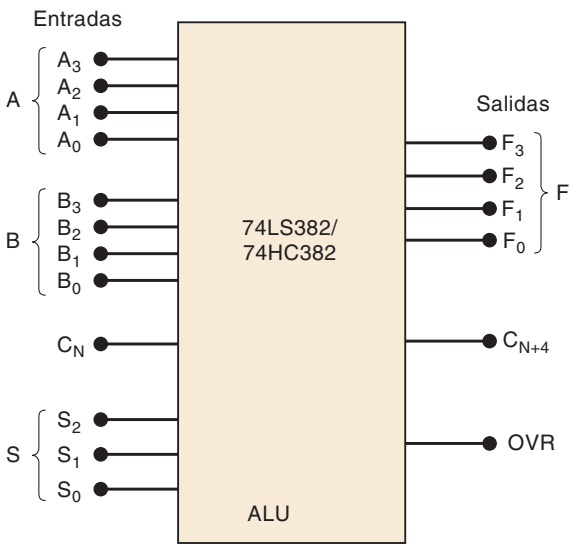
El chip ALU 74LS382/HC382

La figura 6-15(a) muestra el símbolo de bloque para una ALU que está disponible como un 74LS382 (TTL) y como un 74HC382 (CMOS). Este CI de 20 terminales opera sobre dos números de entrada de cuatro bits ($A_3A_2A_1A_0$ y $B_3B_2B_1B_0$) para producir un resultado de salida de cuatro bits ($F_3F_2F_1F_0$). Esta ALU puede realizar *ocho* operaciones distintas. En cualquier momento, la operación que se vaya a realizar dependerá del código de entrada que se aplique a las entradas de selección de función $S_2S_1S_0$. La tabla en la figura 6-15(b) muestra las ocho operaciones disponibles. Enseguida describiremos cada una de estas operaciones.

OPERACIÓN BORRAR Cuando $S_2S_1S_0 = 000$, la ALU *borrará* todos los bits de la salida F, de manera que $F_3F_2F_1F_0 = 0000$.

OPERACIÓN DE SUMA Cuando $S_2S_1S_0 = 011$, la ALU sumará $A_3A_2A_1A_0$ y $B_3B_2B_1B_0$ para producir su suma en $F_3F_2F_1F_0$. Para esta operación, C_N es el acarreo hacia la posición del LSB y debe hacerse 0. C_{N+4} es la salida de acarreo que proviene de la posición del MSB. *OVR* es la salida indicadora de desbordamiento; detecta un desbordamiento cuando se utilizan números con signo. *OVR* será 1 cuando una operación de suma o de resta produzca un resultado demasiado grande como para caber en cuatro bits (incluyendo el bit de signo).

OPERACIONES DE RESTA Cuando $S_2S_1S_0 = 001$, la ALU restará el número en la entrada A del número en la entrada B. Cuando $S_2S_1S_0 = 010$, la ALU restará B de A. En cualquier caso, la diferencia aparecerá en $F_3F_2F_1F_0$. Observe que las operaciones de resta requieren que la entrada C_N sea 1.



A = número de entrada de 4 bits
B = número de entrada de 4 bits
C_N = acarreo hacia la posición del LSB
S = entradas de selección de operación de 3 bits

F = número de salida de 4 bits
C_{N+4} = acarreo que proviene de la posición del MSB
OVR = indicador de desbordamiento

(a)

Tabla de funciones

S_2	S_1	S_0	Operación	Comentarios
0	0	0	BORRAR	$F_3F_2F_1F_0 = 0000$
0	0	1	B menos A	Necesita $C_N = 1$
0	1	0	A menos B	Necesita $C_N = 1$
0	1	1	A más B	Necesita que $C_N = 0$
1	0	0	A ⊕ B	OR exclusivo
1	0	1	A + B	OR
1	1	0	AB	AND
1	1	1	PRESET	$F_3F_2F_1F_0 = 1111$

Observaciones: Las entradas S seleccionan la operación
OVR = 1 para el desbordamiento en números con signo

(b)

FIGURA 6-15 (a) Símbolo de bloque para el chip ALU 74LS382/HC382; (b) tabla de funciones que muestra cómo las entradas de selección (S) determinan la operación que se va a realizar sobre las entradas A y B.

OPERACIÓN XOR Cuando $S_2S_1S_0 = 100$, la ALU realizará una operación XOR bit por bit sobre las entradas A y B. Esto se ilustra a continuación para $A_3A_2A_1A_0 = 0110$ y $B_3B_2B_1B_0 = 1100$.

$$\begin{aligned} A_3 \oplus B_3 &= 0 \oplus 1 = 1 = F_3 \\ A_2 \oplus B_2 &= 1 \oplus 1 = 0 = F_2 \\ A_1 \oplus B_1 &= 1 \oplus 0 = 1 = F_1 \\ A_0 \oplus B_0 &= 0 \oplus 0 = 0 = F_0 \end{aligned}$$

El resultado es $F_3F_2F_1F_0 = 1010$.

OPERACIÓN OR Cuando $S_2S_1S_0 = 101$, la ALU realizará una operación OR bit por bit sobre las entradas A y B. Por ejemplo, para $A_3A_2A_1A_0 = 0110$ y $B_3B_2B_1B_0 = 1100$ la ALU generará un resultado de $F_3F_2F_1F_0 = 1110$.

OPERACIÓN AND Cuando $S_2S_1S_0 = 110$, la ALU realizará una operación AND bit por bit sobre las entradas A y B. Por ejemplo, para $A_3A_2A_1A_0 = 0110$ y $B_3B_2B_1B_0 = 1100$ la ALU generará un resultado de $F_3F_2F_1F_0 = 1100$.

OPERACIONES PRESET Cuando $S_2S_1S_0 = 111$, la ALU *establecerá* todos los bits de la salida, de manera que $F_3F_2F_1F_0 = 1111$.

EJEMPLO 6-11

(a) Determine las salidas del 74HC382 para las siguientes entradas: $S_2S_1S_0 = 010$, $A_3A_2A_1A_0 = 0100$, $B_3B_2B_1B_0 = 0001$ y $C_N = 1$.

(b) Cambie el código de selección a 011 y repita.

Solución

(a) En la tabla de funciones de la figura 6-15(b) podemos ver que 010 selecciona la operación $(A - B)$. La ALU realizará la resta en complemento a 2; para ello complementará B y sumará el resultado a A y C_N. Observe que se necesita la condición $C_N = 1$ para completar el complemento a 2 de B en forma efectiva.

$$\begin{array}{rll} & 1 & \leftarrow C_N \\ & 0100 & \leftarrow A \\ + & \underline{1110} & \leftarrow \overline{B} \\ & 10011 & \\ C_{N+4} & \uparrow \quad \uparrow & F_3\ F_2\ F_1\ F_0 \end{array}$$

Como siempre pasa en la resta en complemento a 2, el ACARREO del MSB se descarta. El resultado correcto de la operación $(A - B)$ aparece en las salidas F.

La salida *OVR* se determina considerando que los números de entrada son con signo. Por lo tanto, tenemos que $A_3A_2A_1A_0 = 0100 = +4_{10}$ y $B_3B_2B_1B_0 = 0001 = +1_{10}$. El resultado de la operación de resta es $F_3F_2F_1F_0 = 0011 = +3_{10}$, el cual es correcto. Por lo tanto, no se ha producido ningún desbordamiento y $OVR = 0$. Si el resultado hubiera sido negativo, habría estado en forma de complemento a 2.

(b) Un código de selección de 011 producirá la suma de las entradas A y B. No obstante, como $C_N = 1$, habrá un acarreo de 1 que se sumará a la posición del LSB. Esto producirá un resultado de $F_3F_2F_1F_0 = 0110$, el cual es 1 mayor que $(A + B)$. Las salidas C_{N+4} y OVR serán 0. Para que aparezca la suma correcta en F, la entrada C_N deberá ser 0.

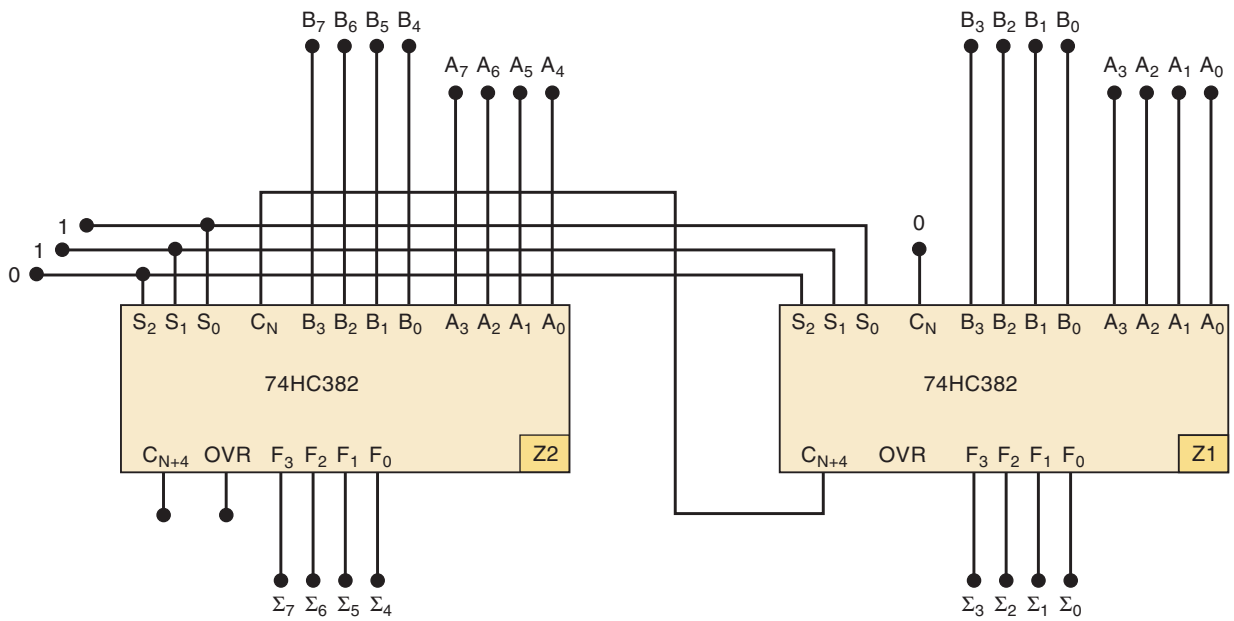
Expansión de la ALU

Un chip 74LS382 o 74HC382 individual opera sobre números de cuatro bits. Pueden conectarse dos o más de estos chips en conjunto para operar sobre números más grandes. La figura 6-16 muestra cómo se combinan dos ALUs de cuatro bits para sumar dos números de ocho bits ($B_7B_6B_5B_4B_3B_2B_1B_0$ y $A_7A_6A_5A_4A_3A_2A_1A_0$) y producir la suma de salida $\Sigma_7\Sigma_6\Sigma_5\Sigma_4\Sigma_3\Sigma_2\Sigma_1\Sigma_0$. Estudie el diagrama del circuito y observe los siguientes puntos:

1. El chip Z1 opera con los cuatro bits de mayor orden de los dos números de entrada. El chip Z2 opera con los cuatro bits de mayor orden.
2. La suma aparece en las salidas *F* de Z1 y Z2. Los bits de menor orden aparecen en Z1 y los bits de mayor orden aparecen en Z2.
3. La entrada C_N de Z1 es el acarreo hacia la posición del LSB. Para la suma se hace 0.
4. La salida de acarreo [C_{N+4}] de Z1 se conecta a la entrada de acarreo [C_N] de Z2.
5. La salida *OVR* de Z2 es el indicador de desbordamiento cuando se utilizan números de ocho bits con signo.
6. Las entradas de selección correspondientes de los dos chips se conectan entre sí, de manera que Z1 y Z2 siempre estén realizando la misma operación. Para la suma, las entradas de selección se muestran como 011.

EJEMPLO 6-12

¿Cómo tendría que cambiarse el arreglo de la figura 6-16 para realizar la resta $(B - A)$?



Observaciones:
Z1 suma los bits de menor orden.
Z2 suma los bits de mayor orden.
Σ_7–Σ_0= suma de 8 bits.
OVR de Z2 es el indicador de desbordamiento de 8 bits.

FIGURA 6-16 Dos chips ALU 74HC382 conectados como un sumador de ocho bits.

Solución

El código de entrada de selección [consulte la tabla en la figura 6-15(b)] debe cambiarse a 001 y la entrada C_N de Z1 debe hacerse 1.

Otras ALUs

El chip 74LS181/HC181 es otra ALU de cuatro bits. Tiene cuatro entradas de selección que pueden elegir cualquiera de 16 operaciones distintas. También tiene un bit de entrada de modo que puede alternar entre las operaciones lógicas y las aritméticas (sumar y restar). Esta ALU tiene una salida $A = B$ que se utiliza para comparar las magnitudes de las entradas A y B. Cuando los dos números de entrada sean exactamente iguales, la salida $A = B$ será 1; en caso contrario será 0.

El chip 74LS881/HC881 es similar al chip 181, sólo que tiene la capacidad de realizar algunas operaciones lógicas adicionales.

PREGUNTAS DE REPASO

1. Aplique las siguientes entradas a la ALU de la figura 6-15 y determine las salidas: $S_2S_1S_0 = 001$, $A_3A_2A_1A_0 = 1100$, $B_3B_2B_1B_0 = 1001$, $C_N = 1$.
2. Cambie el código de selección a 011 y C_N a 0, y repita la pregunta de repaso 1.
3. Cambie el código de selección a 110 y repita la pregunta 1.
4. Aplique las siguientes entradas al circuito de la figura 6-16 y determine las salidas: $B = 01010011$, $A = 00011000$.
5. Cambie el código de selección a 111 y repita la pregunta 4.
6. ¿Cuántos chips 74HC382 se necesitan para sumar dos números de 32 bits?

6-17 EJEMPLO PRÁCTICO DE DIAGNÓSTICO DE FALLAS

Un técnico está probando el sumador/restador que aparece en la figura 6-17 y registra los siguientes resultados de la prueba para los diversos modos de operación:

Modo 1: SUMA = 0, RESTA = 0. Las salidas de la suma siempre son iguales al número en el registro *A más uno*. Por ejemplo, cuando [A] = 0110 la suma es [Σ] = 0111. Esto es incorrecto, ya que las salidas de la compuerta OR y C_0 deben ser 0 en este modo para producir [Σ] = [A].

Modo 2: SUMA = 1, RESTA = 0. La suma siempre es 1 más de lo que debe ser. Por ejemplo, cuando [A] = 0010 y [B] = 0100 la salida de la suma es 0111 en vez de 0110.

Modo 3: SUMA = 0, RESTA = 1. Las salidas Σ siempre son iguales a [A] − [B], como es de esperarse.

Cuando el técnico examina estos resultados, ve que las salidas de la suma exceden los resultados esperados por 1 para los primeros dos modos de operación. Al principio sospecha que hay una posible falla en una de las entradas del LSB para el sumador, pero lo descarta debido a que una falla así también afectaría a la operación de resta, la cual está funcionando en forma correcta. Más adelante se da cuenta que hay otra falla que podría agregar un 1 a los resultados para los primeros dos modos sin provocar un error en el modo de resta.

Recuerde que C_0 se hace 1 en el modo de resta como parte de la operación de complemento a 2 sobre [B]. Para los demás modos, C_0 debe ser 0. El técnico comprueba la conexión entre la señal *RESTA* y la entrada C_0 que va al sumador, y

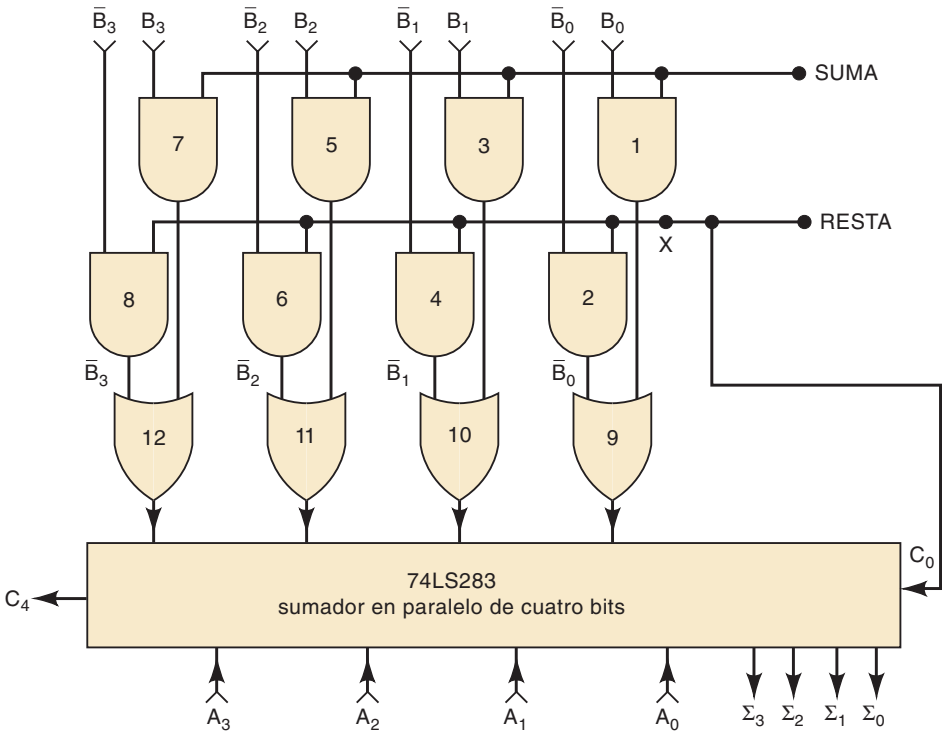


FIGURA 6-17 Circuito sumador/restador en paralelo.

descubre que está abierta debido a una mala conexión de soldadura. Esta conexión abierta explica los resultados observados, ya que el sumador TTL responde como si C_0 fuera un 1 lógico constante, lo cual provoca que se agregue un 1 adicional al resultado en los modos 1 y 2. La conexión abierta no tendría efecto en el modo 3, ya que se supone que C_0 debe ser 1 de todas formas.

EJEMPLO 6-13

Considere de nuevo el circuito sumador/restador. Suponga que en la figura 6-17 hay una interrupción en la ruta de conexión entre la entrada RESTA y las compuertas AND en el punto *X*. Describa los efectos de esta conexión abierta sobre la operación del circuito para cada modo.

Solución

Primero hay que tener en cuenta que esta falla producirá un 1 lógico en la entrada afectada de las compuertas AND 2, 4, 6 y 8, con lo cual se habilitará en forma permanente cada una de estas compuertas para pasar su entrada $\bar{B}$ a la siguiente compuerta OR, como se muestra.

Modo 1: SUMA = 0, RESTA = 0. La falla provocará que el circuito realice una resta… casi. El complemento a 1 de [*B*] llegará a las salidas de la compuerta OR y se aplicará al sumador junto con [*A*]. Con $C_0 = 0$ no se completará el complemento a 2 de [*B*]; le faltará 1. Por lo tanto, el sumador producirá [*A*] – [*B*] – 1. Para ilustrar esto, probemos con [*A*] = +6 = 0110 y [*B*] = +3 = 0011. El sumador realizará la suma de la siguiente manera:

$$\begin{aligned} \text{complemento a 1 de } [B] &= \ 1100 \\ [A] &= \ 0110 \\ \text{resultado} &= \ 10010 \\ &\quad\ \uparrow \text{— Se descarta el acarreo.} \end{aligned}$$

El resultado es 0010 = +2 en vez de 0011 = +3, como debería ser para la resta normal.

Modo 2: SUMA = 1, RESTA = 0. Cuando SUMA = 1, las compuertas AND 1, 3, 5 y 7 pasarán las entradas B a la siguiente compuerta OR. Así, cada compuerta OR tendrá $\overline{B}$ y B en sus entradas, con lo cual se producirá una salida de 1. Por ejemplo, las entradas para la compuerta OR 9 serán $\overline{B}_0$ que proviene de la compuerta AND 2 (debido a la falla) y B_0 que proviene de la compuerta AND 1 (debido a que SUMA = 1). En consecuencia, la compuerta OR 9 producirá una salida de $\overline{B}_0 + B_0$, que siempre será un 1 lógico.

El sumador sumará el 111 que proviene de las compuertas OR con $[A]$ para producir una suma que será 1 menos que $[A]$. ¿Por qué? Porque $1111_2 = -1_{10}$.

Modo 3: SUMA = 0, RESTA = 1. Este modo funcionará de manera correcta, ya que se supone que RESTA = 1 habilita de todas formas las compuertas AND 2, 4, 6 y 8.

6-18 USO DE LAS FUNCIONES DE LA BIBLIOTECA TTL CON ALTERA

Los CIs sumador y ALU que vimos en este capítulo son sólo unos cuantos de los chips MSI que han servido como bloques fundamentales de los sistemas digitales durante décadas. Siempre que una tecnología tiene un tiempo de vida tan largo y útil, deja un impacto duradero en el campo y en las personas que la utilizan. Es evidente que los circuitos integrados TTL caen en esta categoría y continúan en varias formas hoy en día. Los ingenieros y técnicos experimentados están familiarizados con las piezas estándar. Los diseños existentes pueden refabricarse y actualizarse utilizando los mismos circuitos básicos si pueden implementarse en un PLD de VLSI. Las hojas técnicas para estos dispositivos ya están disponibles y el estudiar estas viejas piezas TTL es una excelente manera de aprender los fundamentos de cualquier sistema digital.

Por todas esas razones, el sistema de desarrollo de Altera ofrece lo que ellos denominan como macrofunciones al estilo antiguo. Una **macrofunción** es una descripción autocontenida de un circuito lógico con todas sus entradas, salidas y características operacionales definidas. En otras palabras, se han tomado la molestia de escribir el código necesario para hacer que un PLD emule la operación de muchos dispositivos MSI TTL convencionales. Todo lo que el diseñador necesita saber es cómo conectarlo al resto del sistema. En esta sección nos extenderemos en los conceptos de las primitivas lógicas y las bibliotecas que presentamos en el capítulo 5, para ver cómo podemos usar piezas MSI estándar en nuestros diseños.

La unidad aritmética lógica (ALU) 74382 es un CI bastante sofisticado. La tarea de describir su operación mediante el uso de código de HDL es un buen reto, pero ciertamente está dentro de nuestro alcance. Consulte de nuevo los ejemplos de este CI y su operación, que cubrimos en la sección 6-16. Específicamente analice la figura 6-16, que muestra cómo conectar en cascada dos chips ALU de cuatro bits para formar una ALU de ocho bits que podría servir como el corazón de la unidad central de procesamiento (CPU) de un microcontrolador. La figura 6-18 muestra el método gráfico para describir el circuito de ocho bits mediante el archivo de descripción gráfica de Altera y los bloques de macrofunciones de su biblioteca de componentes. Los símbolos para el 74382 sólo se eligen de la lista en la biblioteca de macrofunciones y se colocan en la pantalla. El proceso de conectar con alambres estos chips es simple e intuitivo.

Es posible conectar piezas MSI de la biblioteca estándar utilizando sólo HDL. Así como demostramos la conexión de primitivas de flip-flops para formar circuitos más complejos, podemos conectar CIs tales como el 74382 con otras piezas. Los nombres de todos los puertos de entrada y salida de estas piezas estándar se definen en

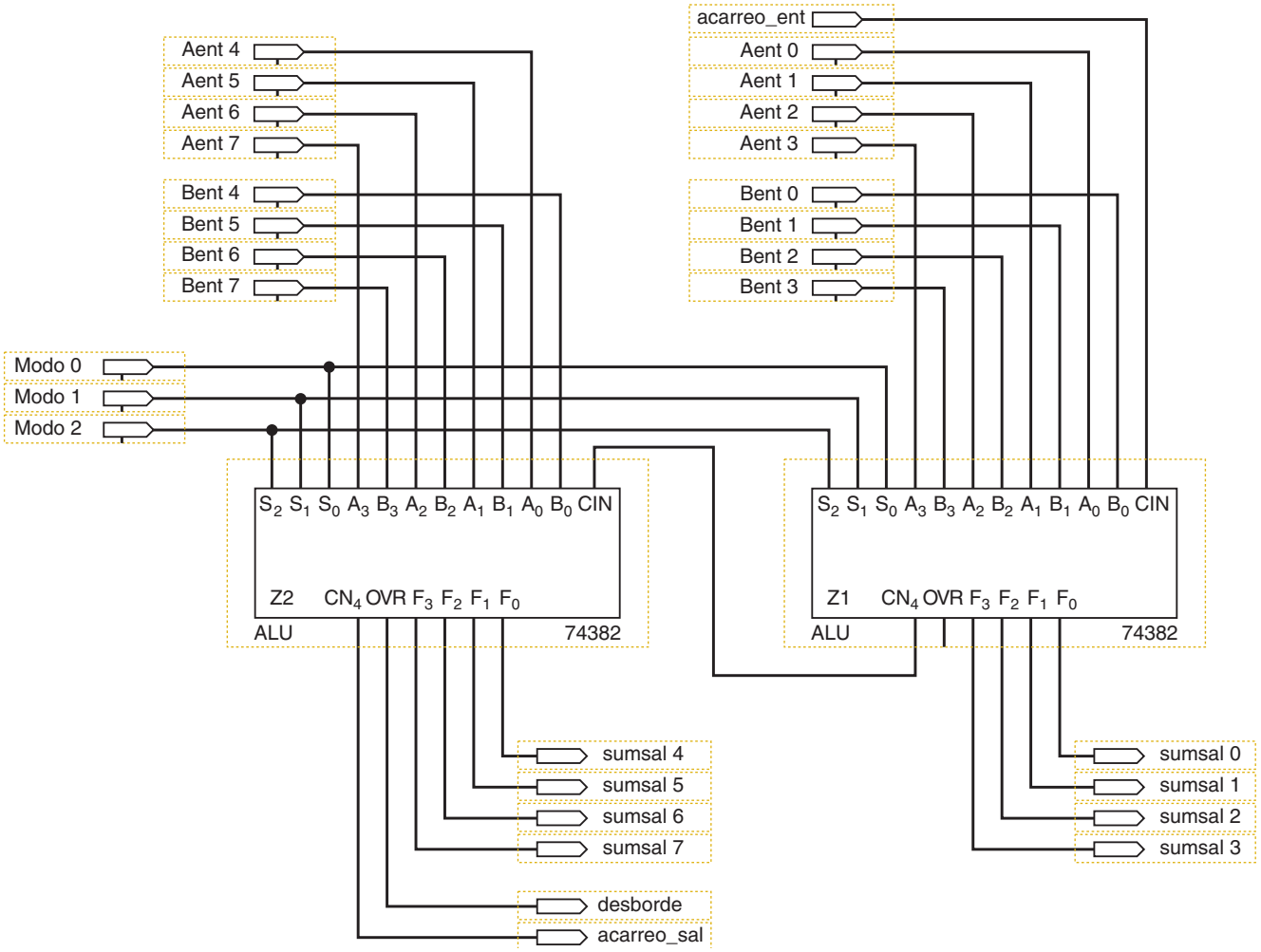


FIGURA 6-18 Un archivo de descripción gráfica de Altera para una ALU de ocho bits.

un **prototipo de función**, el cual podemos encontrar en el menú de ayuda. El prototipo de función que se da para un 74382 es

Prototipo de función en AHDL (el nombre de los puertos y el orden también se aplican al HDL Verilog):

```
FUNCTION 74382 (s[2..0], a[3..0], b[3..0], cin)
RETURNS (ovr, cn4, f[3..0]);
```

PREGUNTAS DE REPASO

1. ¿En dónde podemos encontrar información sobre el uso de un sumador completo 74283 en nuestro diseño de HDL?
2. ¿Qué es una macrofunción?

6-19 OPERACIONES LÓGICAS CON ARREGLOS DE BITS

En la sección anterior examinamos el uso de las macrofunciones para construir sistemas a partir de piezas estándar. Ahora necesitamos practicar la escritura de

código de HDL en lugar de usar una macrofunción para crear un sumador similar al de la figura 6-6. En esta sección extenderemos nuestra comprensión de las técnicas de HDL en dos áreas principales: especificar grupos de bits en un arreglo y utilizar operaciones lógicas para combinar arreglos de bits mediante el uso de expresiones booleanas.

En la sección 6-12 analizamos el concepto de registro, la cual facilita mucho la descripción del contenido de los registros y las señales que consisten de varios bits. Los HDLs utilizan arreglos de bits en una notación similar para describir las señales, como vimos en el capítulo 4. Por ejemplo, en AHDL la señal de cuatro bits llamada *d* se define como:

```
VARIABLE d[3..0] :NODE.
```

En VHDL, el mismo formato de datos se expresa como:

```
SIGNAL d :BIT VECTOR (3 DOWNTO 0).
```

Cada bit en estos tipos de datos se designa en base a un número de elemento. En este ejemplo de un arreglo de bits llamado *d*, se puede hacer referencia a los bits como d3, d2, d1, d0. También se pueden agrupar bits en **conjuntos**. Por ejemplo, si queremos hacer referencia a los tres bits más significativos de *d* como un conjunto, podemos usar la expresión d[3..1] en AHDL y la expresión d (3 DOWNTO 1) en VHDL. Una vez que se asigna un valor al arreglo y que se identifica el conjunto deseado de bits, pueden realizarse operaciones lógicas sobre todo el conjunto completo de bits. Dos conjuntos pueden combinarse en una expresión lógica siempre y cuando éstos sean del mismo tamaño (el mismo número de bits), de igual forma como se combinan las variables individuales en una ecuación booleana. Cada uno de los pares de bits correspondiente en los dos conjuntos se combina como se declara en la ecuación lógica. Esto permite que una ecuación describa la operación lógica que se va a realizar en cada bit de un conjunto.

EJEMPLO 6-14

Suponga que D_3, D_2, D_1, D_0 tiene el valor 101 y que G_3, G_2, G_1, G_0 tiene el valor 1100. Vamos a definir Dnum = $[D_3, D_2, D_1, D_0]$ y Gnum = $[G_3, G_2, G_1, G_0]$. Definamos también Y = $[Y_3, Y_2, Y_1, Y_0]$, en donde Y se relaciona con Dnum y Gnum de la siguiente forma:

Y = Dnum • Gnum;

¿Cuál es el valor de Y después de esta operación?

D_3, D_2, D_1, D_0	1 0 1 1	
↕ ↕ ↕ ↕	↕ ↕ ↕ ↕	Se aplica un AND a cada posición de bit
G_3, G_2, G_1, G_0	1 1 0 0	
Y_3, Y_2, Y_1, Y_0	1 0 0 0	

Solución

Así, Y es un conjunto de cuatro bits con el valor 1000.

EJEMPLO 6-15

Para los valores de los registros descritos en el ejemplo 6-14, declare cada *d*, *g* y *y*. Después escriba una expresión en la que utilice su HDL favorito, que realice la operación AND sobre todos los bits.

Solución

```
SUBDESIGN and nivelbits
(    d[3..0], g[3..0]      :INPUT;
     y[3..0]               :OUTPUT;)
BEGIN
     y[] = d[] & g[];
END;
```

```
ENTITY and nivelbits IS
PORT(d, g          :IN BIT VECTOR (3 DOWNTO 0);
     y             :OUT BIT VECTOR (3 DOWNTO 0));
END and nivelbits;
ARCHITECTURE a OF and nivelbits IS
BEGIN
     y <= d AND g;
END a;
```

6-20 SUMADORES EN HDL

En esta sección veremos cómo crear un circuito sumador en paralelo que pueda utilizarse para sumar arreglos de bits mediante la ecuación lógica para un sumador completo de un solo bit. La figura 6-19 muestra el diagrama de bloques básico con las señales etiquetadas para crear un sumador de cuatro bits. Observe que cada uno de los bits del primer sumando [*A*], del sumando [*B*], del acarreo [*C*] y de la suma [*S*] son variables tipo arreglo de bits y tienen números de índice asociados. La ecuación para la suma si se utiliza la notación de registro es:

$$[S] = [A] \oplus [B] \oplus [C];$$

Observe que las señales de acarreo entre las etapas no son entradas o salidas de este circuito en general, sino variables intermedias. Debemos desarrollar una estrategia para etiquetar los bits de acarreo, de manera que puedan utilizarse en un arreglo. Hemos optado por dejar que cada bit de acarreo sirva como entrada para su correspondiente etapa de sumador, como se muestra en la figura 6-19. Por ejemplo, C0 es una entrada para la etapa del bit 0, C1 es la entrada de acarreo para la etapa del bit 1, y así en lo sucesivo. Los bits del arreglo de acarreos pueden considerarse como los "alambres" que conectan a los sumadores. Este arreglo debe tener un acarreo como entrada para cada etapa y también un acarreo como salida para la etapa del bit más significativo. En este ejemplo habrá cinco bits, etiquetados de C4 a C0, en el arreglo de acarreos. El conjunto de datos que representa las salidas de acarreo sería de C4 a C1, y el conjunto de datos que representa las entradas de acarreo sería de C3 a C0.

$$Cent = C[3..0]$$
$$Csal = C[4..1]$$

Los HDLs nos permiten especificar cuáles conjuntos de bits que salgan de todo el arreglo queremos utilizar en una ecuación. Para asegurar que todas las variables que se están combinando en una ecuación lógica contengan el mismo número de bits, podemos empezar con la ecuación general para el acarreo de salida de un sumador de un bit, como se muestra a continuación:

$$Csal = AB + ACent + BCent$$

PRIMER SUMANDO	A_3	A_2	A_1	A_0	A
SUMANDO	B_3	B_2	B_1	B_0	B
ACARREOent	C_3	C_2	C_1	C_0	Cent
SUMA	S_3	S_2	S_1	S_0	S

Genera la suma
S=A ⊕ [B ⊕ Cent]

PRIMER SUMANDO	A_3	A_2	A_1	A_0	A
SUMANDO	B_3	B_2	B_1	B_0	B
ACARREOent	C_3	C_2	C_1	C_0	Cent
ACARREOsal	C_4	C_3	C_2	C_1	Csal

Genera los bits de acarreo
Csal = A•B + A•Cent + B•Cent

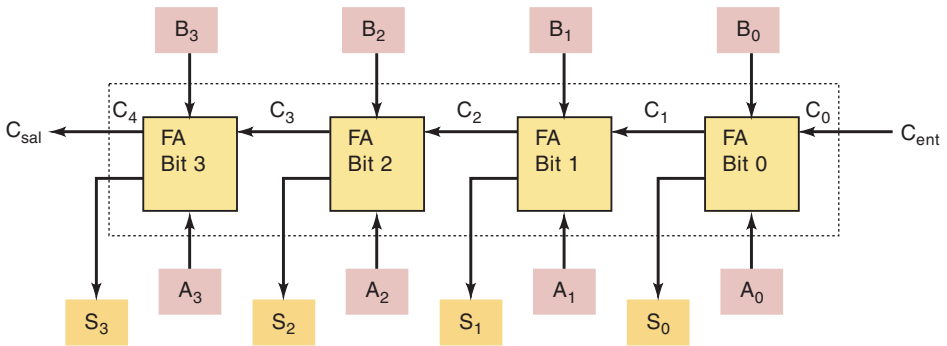


FIGURA 6-19 Sumador en paralelo de cuatro bits.

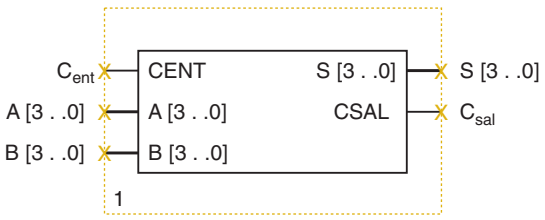
Si sustituimos nuestra definición anterior de Cent y Csal, tendremos la siguiente ecuación para el acarreo de salida del sumador de cuatro bits:

$$C[4..1] = A[3..0] \,\&\, B[3..0] + A[3..0] \,\&\, C[3..0] + B[3..0] \,\&\, C[3..0]$$

El símbolo gráfico para este dispositivo se muestra en la figura 6-20. Observe que no muestra el arreglo de acarreos. Es una variable o señal *dentro* del bloque. Altera permite que cualquier bloque SUBDESIGN en AHDL o ENTITY en VHDL (incluso los que usted cree) se represente mediante un símbolo de diagrama de bloques gráfico tal como éste. Todo esto forma parte del esquema de diseño jerárquico del sistema de desarrollo de Altera (descrito en el capítulo 4).

Para resumir esta información, la figura 6-19 muestra el "interior" del diagrama de bloques en la figura 6-20 y sintetiza la operación descrita por las dos ecuaciones. Ahora veamos los archivos basados en texto que pueden usarse para generar un símbolo de bloque como el de la figura 6-20, mediante el uso de AHDL y de VHDL.

FIGURA 6-20 Símbolo de bloque generado por MAX+PLUS de Altera.



AHDL

SUMADOR DE CUATRO BITS EN AHDL

En las líneas 14 y 15 del código de AHDL de la figura 6-21, observe la sintaxis para hacer referencia a los arreglos de bits en su totalidad. Se da el nombre, seguido de [ ]. Si no se designan bits dentro de los corchetes, significa que todos los bits que se declararon se incluyen en las operaciones. Las líneas 14 y 15 describen por completo los cuatro circuitos sumadores y calculan la suma. Para poder elegir un conjunto específico de elementos del arreglo (es decir, un subconjunto del arreglo), el nombre va seguido del intervalo de números de los elementos entre corchetes. Por ejemplo, la ecuación de acarreo (línea 15) en sintaxis de AHDL es:

c[4..1] = a[] & b[] # a[] & c[3..0] # b[] & c[3..0];

Observe que sólo se están asignando cuatro bits del arreglo de acarreos *c[ ]*, incluso aunque el arreglo sea de cinco bits. De esta forma, el acarreo que sale de cada sumador de bit individual se asigna como el acarreo que entra a la siguiente etapa.

```
1   SUBDESIGN  fig6_21
2   (
3          cent          :INPUT;       --acarreo de entrada
4          a[3..0]       :INPUT;       -- primer sumando
5          b[3..0]       :INPUT;       -- sumando
6          s[3..0]       :OUTPUT;      -- suma
7          csal          :OUTPUT;      -- acarreo de salida
8    )
9   VARIABLE
10         c[4..0]              :NODE; -- el arreglo de acarreos es de 5 bits
11
12  BEGIN
13         c[0] = cent;
14         s[ ] = a[ ]  $  b[ ]  $  c[3..0]; -- genera la suma
15         c[4..1] = (a[ ] & b[ ] # (a[ ] & c[3..0]) # (b[ ] & c[3..0]);
16         csal = c[4];                        -- carry out
17  END;
```

FIGURA 6-21 Sumador en AHDL.

VHDL

SUMADOR DE CUATRO BITS EN VHDL

En el código de VHDL de la figura 6-22, observe la sintaxis para hacer referencia a los arreglos de bits en su totalidad. Sólo se utiliza el nombre sin designaciones de bits. Las líneas 15 y 16 describen por completo los cuatro circuitos sumadores que calcularán la suma. Para poder elegir un conjunto específico de elementos del arreglo (es decir, un subconjunto del arreglo), el nombre va seguido del intervalo de números de los elementos entre paréntesis. La ecuación del acarreo (línea 16) en sintaxis de VHDL es:

c(4 DOWNTO 1) <= (a AND b) OR (a AND c(3 DOWNTO 0)) OR (b AND c(3 DOWNTO 0));

Observe que sólo se están asignando cuatro bits del arreglo de acarreos *c*, incluso aunque el arreglo sea de cinco bits. De esta forma, el acarreo que sale de cada sumador de un solo bit se asigna como el acarreo que entra a la siguiente etapa.

```
1   ENTITY  fig6_22  IS
2   PORT(
3         cent   :IN BIT;
4         a      :IN BIT_VECTOR(3 DOWNTO 0);
5         b      :IN BIT_VECTOR(3 DOWNTO 0);
6         s      :OUT BIT_VECTOR (3 DOWNTO 0);
7         csal   :OUT BIT);
8   END  fig6_22;
9
10  ARCHITECTURE  a  OF  fig6_22  IS
11  SIGNAL c :BIT_VECTOR (4 DOWNTO 0); -- el acarreo requiere un arreglo de 5 bits
12
13  BEGIN
14        c(0) <= cent;        -- Lee el acarreo que va al arreglo de bits
15        s <= a XOR b XOR c(3 DOWNTO 0);   -- Genera los bits de suma
16        c(4 DOWNTO 0) <=     (a  AND  b)
17                       OR    (a AND c(3 DOWNTO 0))
18                       OR    (b AND c(3 DOWNTO 0));
19        csal  <=  c(4);      -- sacar el acarreo del MSB.
20  END
```

FIGURA 6-22 Sumador en VHDL.

PREGUNTAS DE REPASO

1. Si [*A*] = 1001 y [*B*] = 0011, ¿cuál es el valor de (a) [*A*] · [*B*]? (b) ¿de [*A*] + [*B*]? (Observe que · significa AND; + significa OR.)
2. Si *A*[7..0] = 1010 1100, ¿cuál es el valor de (a) *A*[7..4]? ¿de (b) *A*[5..2]?
3. En AHDL se declara el siguiente objeto: conmuta[7..0]:INPUT. Proporcione una expresión para los cuatro bits menos significativos, utilizando sintaxis de AHDL.
4. En VHDL se declara el siguiente objeto: conmuta :IN BIT_VECTOR (7 DOWNTO 0). Proporcione una expresión para los cuatro bits menos significativos, utilizando sintaxis de VHDL.
5. ¿Cuál sería el resultado de aplicar la operación OR a los dos registros del ejemplo 6-14?
6. Escriba una instrucción en HDL que aplique una operación OR a los dos objetos *d* y *g*. Utilice su HDL favorito.
7. Escriba una instrucción en HDL que aplique una operación XOR a los dos bits más significativos de *d*, con los dos bits menos significativos de *g* y que coloque el resultado en los dos bits medios de *x*.

6-21 EXPANSIÓN DE LA CAPACIDAD DE BITS DE UN CIRCUITO

Una de las formas en que hemos aprendido a aumentar la capacidad de un circuito es conectar en cascada las etapas, como lo hicimos con el chip ALU 74382 en la sección anterior. Podemos hacer esto mediante el método del archivo de diseño gráfico de Altera (como en la figura 6-18) o mediante el método del HDL estructural basado en texto. Con cualquiera de estos métodos necesitamos especificar todas las entradas, salidas e interconexiones entre los bloques. En el caso de este circuito sumador

sería más sencillo empezar con el archivo HDL para un sumador de cuatro bits y sólo incrementar el tamaño de cada una de las variables de los operandos en la ecuación. Por ejemplo, si quisiéramos un sumador de ocho bits sólo tendríamos que expandir *a*, *b* y *s* a ocho bits. El código quedaría casi igual al del sumador de cuatro bits antes mostrado. Esto es sólo un vistazo de algunas de las mejoras en eficiencia que ofrece el HDL. No obstante, la forma en que está escrito el código, los índices de cada señal y cada una de las especificaciones de los *arreglos de bits* en la ecuación tendrían que redefinirse también. En otras palabras, el diseñador tendría que examinar el código con cuidado y cambiar todos los 3s por 7s, todos los 4s por 8s, y así en lo sucesivo.

Uno de los principios importantes en la ingeniería de software es la representación simbólica de las **constantes** que se utilizan en el código. Las constantes son sólo números fijos representados por un nombre (símbolo). Si podemos definir un símbolo (es decir, idear un nombre) en la parte superior del código fuente que se asigne al valor para el número total de bits, y después utilizar este símbolo (nombre) en el código, es mucho más sencillo modificar el circuito. Sólo hay que cambiar una línea de código para expandir la capacidad del circuito. Los ejemplos que se muestran a continuación agregan esta característica al código y también lo actualizan para implementar el circuito sumador/restador como el de la figura 6-14. Hay que recalcar que al expandir la capacidad de un circuito sumador tal como éste también reducirá la velocidad del circuito, debido a la propagación del acarreo (que describimos en la sección 6-13). Para poder mantener estos ejemplos simples, no hemos agregado ningún tipo de lógica para generar un acarreo adelantado.

AHDL

SUMADOR/RESTADOR EN AHDL

En AHDL es muy fácil usar constantes, como se muestra en las líneas 1 y 2 de la figura 6-23. La palabra clave CONSTANT va seguida del nombre simbólico y del

```
1   CONSTANT numero_de_bits = 8;      -- establece el número total de bits
2   CONSTANT n = numero_de_bits - 1;  -- n es el índice de bit más alto
3
4   SUBDESIGN fig6_23
5   (
6     suma        :INPUT;       -- control de suma
7     resta       :INPUT;       -- control de resta y acarreo entrante LSB
8     a[n..0]     :INPUT;       -- bits del primer sumando
9     bent[n..0]  :INPUT;       -- bits del sumando
10    s[n..0]     :OUTPUT;      -- bits de suma
11    acarrsal    :OUTPUT;      -- acarreo de SALIDA del MSB
12  )
13  VARIABLE
14    c[n+1..0]   :NODE;        -- vector de acarreo intermedio
15    b[n..0]     :NODE;        -- vector de operando intermedio
16  BEGIN
17    b[ ] = bent[ ] & suma # NOT bent[ ] & resta;
18    c[0] = resta;                   -- Lee acarreo y lo guarda en variable de grupo
19    s[ ] = a[ ] $ b[ ] $ c[n..0];   -- Genera las sumas
20    c[n+1..1] = a[ ] & b[ ] # a[ ] & c[n..0] # (b[ ] & c[n..0]);
21    acarrsal = c[n+1];              -- saca el acarreo del MSB.
22  END;
```

FIGURA 6-23 Descripción de un sumador/restador de *n* bits en AHDL.

valor que se le va a asignar. Observe que podemos permitir que el compilador realice algunos cálculos matemáticos simples para establecer un valor para una constante con base en otra. También podemos usar esta característica al referirnos a la constante en el código, como se muestra en las líneas 14, 20 y 21. Por ejemplo, podemos hacer referencia a *c[7]* como *c[n]* y a *c[8]* como *c[n+1]*. Para expandir el tamaño de este sumador/restador sólo hay que cambiar el valor de *numero_de_bits* por el número de bits deseado y después volver a compilar el programa.

Como mencionamos antes, actualizamos este código del ejemplo anterior para convertirlo en un sumador/restador como el de la figura 6-14. También hemos incluido las entradas *suma* y *resta* en las líneas 6 y 7, y una nueva variable intermedia llamada *b[ ]* en la línea 15. La primera instrucción concurrente en la línea 17 describe toda la lógica SOP que controla las entradas *b* para el sumador en la figura 6-14. Primero describe una operación AND lógica entre cada bit de *bent[ ]* y el nivel lógico en *suma*. A este resultado se le aplica un OR (bit por bit) con el resultado de aplicar un AND al complemento de cada bit de *bent[ ]* con *resta*. En otras palabras, crea la siguiente función booleana para cada bit: $b = bent \cdot suma + \overline{bent} \cdot resta$. La señal *b[ ]* se utiliza entonces en las ecuaciones del sumador en vez de *bent[ ]*, como en los ejemplos anteriores. Observe en la línea 18 que también se usa *resta* para conectar el LSB del arreglo de acarreos (acarreo hacia el bit 0) con el valor en *resta*, que necesita ser 0 cuando se suma y 1 cuando se resta.

SUMADOR/RESTADOR EN VHDL

En VHDL, utilizar constantes es algo un poco más complicado. Las constantes deben incluirse en un paquete dentro del bloque **PACKAGE**, como se muestra en la figura 6-24, líneas 1 a 4. Los paquetes también se utilizan para contener definiciones de componentes y demás información que debe estar disponible para todas las entidades en el archivo de diseño. Observe en la línea 6 que la palabra clave USE indica al compilador que debe usar las definiciones en este paquete para todo el archivo de diseño. Dentro del paquete, la palabra clave CONSTANT va seguida del nombre simbólico, su tipo y el valor que se le va a signar por medio del operador :=. Observe en la línea 3 que podemos permitir que el compilador realice algunos cálculos matemáticos simples para establecer un valor de una constante con base en otra. También podemos usar esta característica al referirnos a las constantes en el código, como se muestra en las líneas 34 y 37. Por ejemplo, podemos referirnos a *c(7)* como *c(n)* y a *c(8)* como *c(n+1)*. Para expandir el tamaño de este sumador/restador sólo necesitamos cambiar el valor de *numero_de_bits* por el número de bits deseado y después volvemos a compilar el programa.

Como mencionamos antes, hemos actualizado el código del ejemplo anterior para convertirlo en un sumador/restador como el de la figura 6-14. Se han incluido las entradas *suma* y *resta* en las líneas 10 y 11, junto con una nueva señal llamada *b* en la línea 20, *bnot* en la línea 21 y *modo* en la línea 22. La primera instrucción concurrente en la línea 24 sirve para crear el complemento a 1 de *bent*. Los circuitos SOP en la figura 6-14 que controlan las entradas *b* para el sumador seleccionan las entradas *bent* si *suma* = 1, o el complemento a 1 (*bnot*) si *resta* = 1. Ésta es una excelente aplicación de la asignación de señal selecta de VHDL, como se muestra en las líneas 27 a 30. Cuando *suma* es 1, *bent* se canaliza hacia *b*. Cuando *resta* es 1, *bnot* se canaliza a *b*. La señal *b* se utiliza entonces en las ecuaciones de los sumadores en vez de *bent*, como vimos en los ejemplos anteriores de los sumadores. Observe en la línea 32 que también se utiliza *resta* para conectar el LSB del arreglo de acarreos (el acarreo hacia el bit 0) con el valor en *resta*, el cual necesita ser 0 cuando se suma y 1 cuando se resta.

```
1   PACKAGE const IS
2      CONSTANT numero_de_bits :INTEGER:=8;  -- establece el número total de bits
3      CONSTANT n :INTEGET:= numero_de_bits - 1;  -- número de índice del MSB
4   END const;
5
6   USE  work.const.all;
7
8   ENTITY  fig6_24  IS
9   PORT(
10    suma        :IN BIT; -- control de suma
11    resta       :IN BIT; -- control de resta y acarreo del LSB
12    a           :IN BIT_VECTOR(n DOWNTO 0);
13    bent        :IN BIT_VECTOR(n DOWNTO 0);
14    s           :OUT BIT_VECTOR(n DOWNTO 0);
15    acarrsal    :OUT BIT);
16  END  fig6_24;
17
18  ARCHITECTURE  a  OF  fig6_24  IS
19  SIGNAL  c     :BIT_VECTOR (n+1 DOWNTO 0);  -- define acarreos intermedios
20  SIGNAL  b     :BIT_VECTOR (n DOWNTO 0);    -- define operando intermedio
21  SIGNAL  bnot  :BIT_VECTOR (n DOWNTO 0);
22  SIGNAL  modo  :BIT_VECTOR (1 DOWNTO 0);
23  BEGIN
24    bnot    <=  NOT  bent;
25    modo  <=  suma & resta;
26
27    WITH modo SELECT
28       b <= bent      WHEN "10",   -- suma
29            bnot      WHEN "01",   -- resta
30            "0000"    WHEN  OTHERS;
31
32    c(0)  <=  resta;                      -- lee el acarr_ent al arreglo de bits
33    s  <=  a  XOR  b  XOR  x(n DOWNO 0);  -- genera los bits de suma
34    c(n+1 DOWNTO 1) <= (a AND b) OR
35                       (a AND c(n DOWNTO 0)) OR
36                       (b AND c(n DOWNTO 0)); -- genera acarreos
37    acarrsal  <=  c(n+1);          -- saca el acarreo del MSB.
38
39  END a;
```

FIGURA 6-24 Descripción de un sumador/restador de *n* bits en VHDL.

Instrucción GENERATE de VHDL

Otra manera de crear circuitos que procesen más bits es mediante la instrucción **GENERATE** de VHDL. Es una forma muy concisa de indicar al compilador que debe replicar varios componentes que se conectarán en cascada. Como hemos mostrado, existen muchas otras formas de lograr lo mismo y si la naturaleza abstracta de este método le parece difícil, puede usar otro método. Aquí ofrecemos la instrucción GENERATE por cuestión de cubrir todos los métodos. Los circuitos sumadores que hemos estado viendo son cadenas en cascada de módulos de sumadores completos de un solo bit. El código de VHDL para un módulo de sumador completo de un solo bit se muestra en la figura 6-25. Se deben conectar múltiples instancias de este

FIGURA 6-25 Sumador completo de un solo bit en VHDL.

```
1   ENTITY suma1 IS
2   PORT (
3           cent    :IN BIT;
4           a       :IN BIT;
5           b       :IN BIT;
6           s       :OUT BIT;
7           csal    :OUT BIT;
8   END suma1;
9
10  ARCHITECTURE a OF suma1 IS
11  BEGIN
12
13          s       <= a XOR b XOR cent;
14          csal    <= (a AND b) OR (a AND cent) OR (b AND cent);
15  END a;
```

módulo entre sí para formar un circuito sumador de *n* bits. Desde luego que podemos hacerlo mediante el uso de las mismas técnicas de componentes que vimos antes, pero obtendríamos un código muy extenso.

Para que el código sea más conciso y fácil de modificar se necesita una estrategia para la forma en que se van a etiquetar las entradas y salidas para cada módulo. Como mencionamos antes, el sumador del bit 0 tiene entradas con un índice de 0 (por ejemplo, *a0*, *b0*, *c0*, *s0*). El acarreo que sale del bit 0 está etiquetado como *c1* y se convierte en la entrada de acarreo para el módulo del sumador del bit 1. Cada vez que instanciamos otro componente para el siguiente bit del sumador multibits, el número de índice de todas las conexiones avanza en 1 (*a1*, *b1*, *c1*, *s1*). La instrucción GENERATE nos permite repetir un instanciamiento de un componente *n* veces, pues se incrementa el número de índice en 1 para cada instanciamiento hasta *n*. En la línea 27 de la figura 6-26 se utiliza la palabra clave GENERATE en un **ciclo iterativo (ciclo FOR)**, lo cual significa que un conjunto de acciones descriptivas (PORT MAP) se repetirá cierto número de veces. La variable *i* representa un número de índice que empieza en 0 (para la primera iteración) y termina en *n* (la última iteración). Las ventajas de este método son la compactación del código y la facilidad con la que se puede expandir el número de bits. El código en la figura 6-26 muestra cómo usar un sumador de un solo bit (figura 6-25) como un componente para generar un circuito sumador de ocho bits. Recuerde que el archivo para el sumador de un solo bit (suma1.vhd en la figura 6-25) debe guardarse en la misma carpeta que el archivo de diseño que lo utiliza, para generar múltiples instancias del sumador (fig6_26.vhd).

El componente del sumador de un solo bit se define en las líneas 17 a 23 de la figura 6-26. Para la primera iteración de las líneas 29 y 30 el valor de *i* es 0, con lo cual se crea una etapa de sumador para el bit 0. La segunda iteración *i* se incrementa a 1 para formar la etapa de sumador 1. Esto continúa hasta que *i* sea igual a *n*, con lo que se genera cada una de las etapas del sumador de $(n + 1)$ bits. Observe que VHDL nos permite la opción de colocar etiquetas al principio de una línea de código para ayudarnos a describir su propósito. Por ejemplo, en la línea 27 se utiliza la etiqueta *repetir* y en la línea 29 se utiliza la etiqueta *casc*. Las etiquetas son opcionales pero siempre deben terminar con el signo de dos puntos.

Bibliotecas de módulos parametrizados

Es evidente que el uso de las técnicas de HDL facilita la modificación de la capacidad de bits de un circuito genérico. En este capítulo podemos ver que es fácil cambiar de un sumador de cuatro bits a un sumador de ocho, 12 o 16 bits. Cuando Altera

```
PACKAGE const IS
   CONSTANT numero_de_bits :INTEGER:=8;  -- especifica el número de bits
   CONSTANT n :INTEGER:=numero_de_bits - 1;  -- n es el número del bit MSB
END const;
USE work.const.all;
ENTITY fig6_26 IS
PORT(
   acarrent    :IN bit;
   aent        :IN BIT_VECTOR (n DOWNTO 0);
   bent        :IN BIT_VECTOR (n DOWNTO 0);
   ssal        :OUT BIT_VECTOR (n DOWNTO 0);
   acarrsal    :OUT bit);
END fig6_26;

ARCHITECTURE a OF fig6_26 IS

COMPONENT suma1             -- declara un sumador complete de un solo bit
   PORT  (
               cin    :IN BIT;
               a, b   :IN BIT;
               s      :OUT BIT;
               csal   :OUT BIT);
END COMPONENT;
SIGNAL c :BIT_VECTOR (n+1 DOWNTO 0); -- declara arreglo de bits para acarreos
BEGIN
   c(0) <= acarrent;                -- coloca el LSB en el arreglo
                                       (acarreo de entrada)
   repetir:FOR i IN 0 TO n GENERATE  -- instancia n+1 sumadores
                                     -- los conecta en cascada
   casc:suma1 PORT MAP  (cent => c(i), a=> aent(i), b => bent(i),
                         s => ssal(i), csal => c(i+1));
   END GENERATE;
   acarrsal   <= c(n+1);            -- saca el acarreo de la etapa del n-esimo bit
END a;
```

FIGURA 6-26 Uso de la instrucción GENERATE de VHDL.

creó su biblioteca de funciones útiles, también aprovechó estas técnicas y creó lo que se conoce como **megafunciones**, que incluyen una **biblioteca de módulos parametrizados (LPMs)**. Estas funciones no tratan de imitar un CI estándar específico como las macrofunciones al estilo antiguo, sino que ofrecen una solución genérica para los diversos tipos de circuitos lógicos que son útiles en los sistemas digitales. Algunos ejemplos de estos circuitos genéricos que hemos visto hasta ahora son las compuertas lógicas (AND, OR, XOR), los latches, los contadores, los registros de desplazamiento y los sumadores. El término *parametrizados* significa que cuando se instancia una función de la biblioteca, también se especifican ciertos parámetros para definir ciertos atributos (por ejemplo, la capacidad de bits) para el circuito que se está describiendo. Puede consultar los diversos LPMs que hay disponibles a través del menú HELP (AYUDA) bajo megafunctions/LPM (megafunciones/LPM). Esta documentación describe los parámetros que puede especificar el usuario, así como las características de los puertos del dispositivo.

PREGUNTAS DE REPASO

1. ¿Qué palabra clave se utiliza para asignar un nombre simbólico a un número fijo?
2. ¿En dónde se definen las constantes en AHDL? ¿En dónde se definen en VHDL?
3. ¿Por qué son útiles las constantes?
4. Si la constante val_max tiene un valor de 127, ¿cómo interpretará un compilador la expresión val_max -5?
5. ¿Para qué se utiliza la instrucción GENERATE en VHDL?

RESUMEN

1. Para representar números con signo en binario se adjunta un bit de signo como el MSB. Un signo + es un 0 y un signo − es un 1.
2. Para obtener el complemento a 2 de un número binario se complementa cada bit y después se suma 1 al resultado.
3. En el método de complemento a 2 para representar números binarios con signo, los números positivos se representan mediante un bit de signo de 0, seguido de la magnitud en su forma binaria natural. Los números negativos se representan mediante un bit de signo de 1, seguido de la magnitud en forma de complemento a 2.
4. Para cambiar el signo a un número binario con signo, se obtiene el complemento a 2 del número, incluyendo el bit de signo.
5. Para restar números binarios con signo se cambia el signo al el sustraendo (complementando a 2) y se suma al minuendo.
6. En la suma BCD se necesita un paso de corrección especial cada vez que la suma de la posición de un dígito excede de 9 (1001).
7. Cuando se representan números binarios con signo en hexadecimal, el MSD del número hexadecimal será 8 o mayor cuando el número sea negativo; será 7 o menor cuando el número sea positivo.
8. La unidad aritmética/lógica (ALU) de una computadora contiene los circuitos necesarios para realizar operaciones aritméticas y lógicas sobre números binarios almacenados en la memoria.
9. El acumulador es un registro en la ALU. Almacena uno de los números sobre los que se opera y también es el lugar en donde se almacena el resultado de la operación en la ALU.
10. Un sumador completo realiza la suma de dos bits más una entrada de acarreo. Un sumador binario en paralelo está formado por sumadores completos conectados en cascada.
11. El problema de los retrasos excesivos producidos por la propagación del acarreo puede reducirse mediante un circuito lógico de acarreo adelantado.
12. Los sumadores de CI tales como el 74LS83/HC83 y el 74LS283/HC283 pueden utilizarse para construir sumadores y restadores en paralelo de alta velocidad.
13. Un circuito sumador de BCD requiere circuitos de corrección especiales.
14. Hay ALUs de circuito integrado disponibles, las cuales pueden combinarse para realizar una amplia gama de operaciones aritméticas y lógicas sobre dos números de entrada.
15. Hay funciones prefabricadas disponibles en las bibliotecas de Altera.
16. Estas piezas de biblioteca y los circuitos de HDL que usted cree pueden interconectarse mediante las técnicas de HDL gráfica o estructural.

17. Pueden realizarse operaciones lógicas sobre todos los bits en un conjunto mediante el uso de ecuaciones booleanas.
18. El hábito de practicar buenas técnicas de ingeniería de software, en específico el uso de símbolos para representar constantes, nos permite modificar y expandir con facilidad el código en cuanto a la capacidad de bits de circuitos tales como los sumadores completos.
19. Las bibliotecas de módulos parametrizados (LPMs) ofrecen una solución flexible y que puede modificarse o expandirse con facilidad para muchos tipos de circuitos digitales.

TÉRMINOS IMPORTANTES

acarreo
acarreo adelantado
biblioteca de funciones parametrizadas (LPMs)
bit de signo
ciclo FOR
ciclo iterativo
conjuntos
constantes
desbordamiento
GENERATE
macrofunción
megafunciones
minuendo
negación
PACKAGE
primer sumando
propagación de acarreo
prototipo de función
registro acumulador
sistema de complemento a 2
sistema de signo-magnitud
sumador completo (FA)
sumador/restador
sumando
sustraendo
unidad aritmética/lógica (ALU)

PROBLEMAS

SECCIÓN 6-1

B 6-1. Sume lo siguiente en binario. Compruebe sus resultados realizando la suma en decimal.

(a)* 1010 + 1011
(b)* 1111 + 0011
(c)* 1011.1101 + 11.1
(d) 0.1011 + 0.1111
(e) 10011011 + 10011101
(f) 1010.01 + 10.111

SECCIÓN 6-2

B 6-2. Represente cada uno de los siguientes números decimales con signo en el sistema de complemento a 2. Use un total de ocho bits, incluyendo el bit de signo.

(a)* +32 (b)* −14 (c)* +63 (d)* −104
(e)* +127 (f)* −127 (g)* +89 (h)* −55
(i) −1 (j) −128 (k) +169 (l) 0
(m) +84 (n) +3 (o) −3 (p) −190

B 6-3. Cada uno de los siguientes números representa un número decimal con signo en el sistema de complemento a 2. Determine el valor decimal en cada caso. (*Sugerencia:* use la negación para convertir los números negativos en positivos.)

(a)* 01101
(b)* 11101
(c)* 01111011
(d)* 10011001
(e)* 01111111
(f) 10000000
(g) 11111111
(h) 10000001
(i) 01100011
(j) 11011001

* Encontrará las respuestas a los problemas marcados con asterisco al final del libro.

6-4. (a) ¿Cuál es el intervalo de valores decimales con signo que puede representarse utilizando 12 bits, incluyendo el bit de signo?

(b) ¿Cuántos bits se requieren para representar números decimales desde −32,768 hasta +32,767?

6-5.* Liste en orden todos los números con signo que puedan representarse en cinco bits, usando el sistema de complemento a 2.

6-6. Represente cada uno de los siguientes valores decimales como un valor binario con signo de ocho bits. Después niegue cada uno de ellos.

(a)* +73 (b) *−12 (c) +15 (d) −1 (e) −128 (f) +127

6-7. (a)* ¿Cuál es el intervalo de valores decimales sin signo que puede representarse en 10 bits? ¿Cuál es el intervalo de valores decimales con signo si utilizamos el mismo número de bits?

(b) Repita ambos problemas usando ocho bits.

SECCIONES 6-3 Y 6-4

6-8. La razón por la que el método de signo-magnitud para representar números con signo no se utiliza en la mayoría de las computadoras puede ilustrarse mediante lo siguiente:

(a) Represente +12 en ocho bits, utilizando la forma de signo-magnitud.

(b) Represente −12 en ocho bits, utilizando la forma de signo-magnitud.

(c) Sume los dos números binarios y observe que la suma no se parece en nada a cero.

6-9. Realice las siguientes operaciones en el sistema de complemento a 2. Use ocho bits (incluyendo el bit de signo) para cada número. Compruebe sus resultados convirtiendo el resultado binario de vuelta en decimal.

(a)* Sume +9 y +6.
(b)* Sume −14 y −17.
(c)* Sume +19 y −24.
(d)* Sume −48 de −80.
(e)* Reste +16 de +17.
(f) Reste +21 de −13.
(g) Reste +47 de +47.
(h) Reste −36 de −15.
(i) Sume +17 y −17.
(j) Reste −17 de −17.

6-10. Repita el problema 6-9 para los siguientes casos y demuestre que hay desbordamiento en cada uno de ellos.

(a) Sume +37 y +95.
(b) Reste +37 de −95.
(c) Sume −37 y −95.
(d) Reste −37 de +95.

SECCIONES 6-5 Y 6-6

B 6-11. Multiplique los siguientes pares de números binarios y compruebe sus resultados realizando la multiplicación en decimal.

(a)* 111 × 101
(b)* 1011 × 1011
(c) 101.101 × 110.010
(d) .1101 × .1011

B 6-12. Realice las siguientes divisiones. Compruebe sus resultados realizando la división en decimal.

(a)* 1100 ÷ 100
(b)* 111111 ÷ 1001
(c) 10111 ÷ 100
(d) 10110.1101 ÷ 1.1

SECCIONES 6-7 Y 6-8

B 6-13. Sume los siguientes números decimales después de convertir cada uno de ellos en su código BCD.

(a)* 74 + 23 (d) 385 + 118
(b)* 58 + 37 (e) 998+003
(c)* 147 + 380 (f) 623 + 599

B 6-14. Encuentre la suma de cada uno de los siguientes pares de números hexadecimales.

(a)* 3E91 + 2F93 (d) 2FFE + 0002
(b)* 91B + 6F2 (e) FFF + 0FF
(c)* ABC + DEF (f) D191 + AAAB

B 6-15. Realice las siguientes restas con los pares de números hexadecimales:

(a)* 3E91 – 2F93 (d) 0200 – 0003
(b)* 91B – 6F2 (e) F000 – EFFF
(c)* 0300 – 005A (f) 2F00 – 4000

6-16. El manual del propietario para una pequeña microcomputadora establece que la computadora cuenta con ubicaciones de memoria utilizables en las siguientes direcciones hexadecimales: 0200 a 03FF y 4000 a 7FD0. ¿Cuál es el número total de ubicaciones de memoria disponibles?

6-17. (a)* Cierta ubicación de memoria de ocho bits almacena los datos hexadecimales 77. Si éste representa a un número *sin signo*, ¿cuál es su valor decimal?

(b)* Si éste representa a un número *con signo*, ¿cuál es su valor decimal?

(c) Repita los incisos (a) y (b) si el valor de los datos es E5.

SECCIÓN 6-11

6-18. Convierta todas las compuertas en el circuito FA de la figura 6-8 a compuertas NAND.

6-19.* Escriba la tabla de funciones para un medio sumador (entradas *A* y *B*; salidas SUMA y ACARREO). A partir de la tabla de funciones, diseñe un circuito lógico que actúe como medio sumador.

6-20. Un sumador completo puede implementarse en muchas formas distintas. La figura 6-27 muestra cómo puede construirse uno a partir de dos medios sumadores. Construya una tabla de funciones para este arreglo y verifique que opere como un FA.

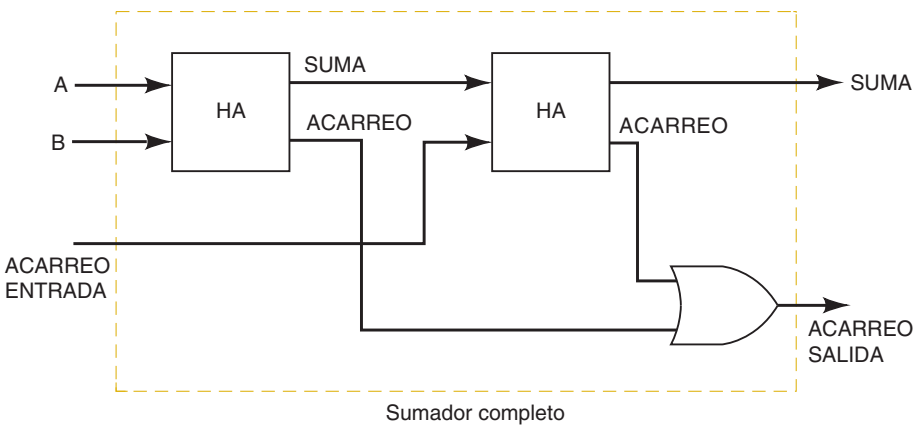


FIGURA 6-27
Problema 6-20.

SECCIÓN 6-12

6-21.* Consulte la figura 6-10. Determine el contenido del registro A después de la siguiente secuencia de operaciones: $[A] = 0000$, $[0100] \rightarrow [B]$, $[S] \rightarrow [A]$, $[1011] \rightarrow [B]$, $[S] \rightarrow [A]$.

6-22. Consulte la figura 6-10. Suponga que cada FF tiene $t_{PLH} = t_{PHL} = 30$ ns y un tiempo de estabilización de 10 ns, y que cada FA tiene un retraso de propagación de 40 ns. ¿Cuál es el mínimo tiempo permitido entre la PGT del pulso CARGAR y la PGT del pulso TRANSFERIR para una operación apropiada?

D 6-23. En los circuitos del sumador y del restador que vimos en este capítulo, no consideramos la posibilidad de *desbordamiento*. El desbordamiento ocure cuando los dos números que se van a sumar o a restar producen un resultado que contiene más bits que la capacidad del acumulador. Por ejemplo, si se utilizan registros de cuatro bits (incluyendo un bit de signo) se pueden almacenar números que varíen desde +7 hasta −8 (en complemento a 2). Por lo tanto, si el resultado de una suma o resta se excede de +7 o de −8, se produce un desbordamiento. Cuando esto ocurre los resultados son inútiles, ya que no pueden almacenarse en forma correcta en el registro acumulador. Para ilustrar esto, sume +5 (0101) y +4 (0100), que nos da el resultado de 1001. Este 1001 se interpretaría en forma incorrecta como un número negativo, ya que hay un 1 en la posición del bit de signo.

En las computadoras y calculadoras, por lo general, hay circuitos que se utilizan para detectar una condición de desbordamiento. Hay varias formas de hacer esto. Uno de los métodos que pueden utilizarse para el sumador que opera en el sistema de complemento a 2 funciona de la siguiente manera:

1. Se examinan los bits de signo de los dos números que se van a sumar.
2. Se examina el bit de signo del resultado.
3. Se produce un desbordamiento cuando los números que se van a sumar *son ambos positivos* y el bit de signo del resultado es 1, *o* cuando los números son *ambos negativos* y el bit de signo del resultado es 0.

Para verificar este método, probemos varios ejemplos. Los lectores deberán probar los siguientes casos para aclarar sus propias dudas: (1) 5 + 4; (2) −4 + (-6); (3) 3 + 2. Los casos 1 y 2 producirán un desbordamiento y el caso 3 no. De esta forma, mediante el análisis de los bits de signo podemos diseñar un circuito lógico que produzca una salida de 1 cada vez que ocurra la condición de desbordamiento. Diseñe este circuito de desbordamiento para el sumador de la figura 6-10.

A, D 6-24. Agregue los circuitos lógicos necesarios a la figura 6-10 para dar cabida a la transferencia de datos desde memoria hacia el registro A. Los valores de datos de la memoria deben entrar al registro A por medio de sus entradas D en la PGT del *primer* pulso en TRANSFERIR; los datos de las salidas de suma de los FAs se cargarán en A cuando ocurra la PGT del *segundo* pulso en TRANSFERIR. En otras palabras, se requiere un pulso en CARGAR seguido de dos pulsos en TRANSFERIR para realizar la secuencia completa de cargar el registro B desde memoria, cargar el registro A desde memoria y después transferir su suma hacia el registro A. (*Sugerencia:* utilice un flip-flop X para controlar cuál origen de datos se debe cargar en las entradas D del acumulador.)

SECCIÓN 6-13

A, D 6-25.* Diseñe un circuito de acarreo adelantado para el sumador de la figura 6-10, que genere el acarreo C_3 que se va a alimentar al FA de la posición del MSB, con base en los valores de A_0, B_0, C_0, A_1, B_1, A_2 y B_2. En otras palabras, derive una expresión para C_3 en términos de A_0, B_0, C_0, A_1, B_1, A_2 y B_2. (*Sugerencia:* comience por escribir la expresión para C_1 en términos de A_0,

B_0 y C_0. Después escriba la expresión para C_2 en términos de A_1, B_1 y C_1. Sustituya la expresión para C_1 en la expresión para C_2. Después escriba la expresión para C_3 en términos de A_2, B_2 y C_2. Sustituya la expresión para C_2 en la expresión para C_3. Simplifique la expresión final para C_3 y colóquela en forma de suma de productos. Implemente el circuito.)

SECCIÓN 6-14

6-26. Muestre los niveles lógicos en cada entrada y salida de la figura 6-11(a) al sumar 354_8 con 103_8.

SECCIÓN 6-15

6-27. Para el circuito de la figura 6-14, determine las salidas de suma para los siguientes casos:

(a)* Registro $A = 0101$ (+5), registro $B = 1110$ (−2); RESTA = 1, SUMA = 0

(b) Registro $A = 1100$ (−4), registro $B = 1110$ (−2); RESTA = 0, SUMA = 1

(c) Repita (b) con SUMA = RESTA = 0.

6-28. Para el circuito de la figura 6-14, determine las salidas de suma para los siguientes casos:

(a) Registro A = 1101 (−3), registro B = 0011 (+3), RESTA = 1, SUMA = 0.

(b) Registro A = 1100 (−4), registro B = 0010 (+2), RESTA = 0, SUMA = 1.

(c) Registro A = 1011 (−5), registro B = 0100 (+4), RESTA = 1, SUMA = 0.

6-29. Para cada uno de los cálculos del problema 6-27, determine si se ha producido un desbordamiento.

6-30. Para cada uno de los cálculos del problema 6-28, determine si se ha producido un desbordamiento.

D 6-31. Muestre cómo pueden implementarse las compuertas de la figura 6-14 utilizando tres chips 74HC00.

D 6-32.* Modifique el circuito de la figura 6-14, de manera que se utilice una sola entrada de control X en vez de SUMA y RESTA. El circuito deberá funcionar como un sumador cuando $X = 0$ y como restador cuando $X = 1$. Después simplifique cada conjunto de compuertas. (*Sugerencia:* observe cómo funciona cada conjunto de compuertas como un inversor controlado.)

SECCIÓN 6-16

B 6-33. Determine las salidas F, C_{N+4} y *OVR* para cada uno de los siguientes conjuntos de entradas que se aplican a un 74LS382.

(a)* $[S] = 011$, $[A] = 0110$, $[B] = 0011$, $C_N = 0$

(b) $[S] = 001$, $[A] = 0110$, $[B] = 0011$, $C_N = 1$

(c) $[S] = 010$, $[A] = 0110$, $[B] = 0011$, $C_N = 1$

D 6-34. Muestre cómo puede usarse el 74HC382 para producir $[F] = [A]$. (*Sugerencia:* recuerde esa propiedad especial de una compuerta XOR.)

6-35. Determine las salidas Σ en la figura 6-16 para los siguientes conjuntos de entradas.

(a)* $[S] = 110$, $[A] = 10101100$, $[B] = 00001111$

(b) $[S] = 100$, $[A] = 11101110$, $[B] = 00110010$

A, D 6-36. Agregue la lógica necesaria a la figura 6-16 para producir una salida individual en ALTO cada vez que el número binario en *A* sea exactamente el mismo que el número binario en *B*. Aplique el código de entrada de selección apropiado (pueden usarse tres códigos).

SECCIÓN 6-17

F 6-37. Considere el circuito de la figura 6-10. Suponga que la salida A_2 se queda atascada en BAJO. Siga la secuencia de operaciones para sumar dos números y determine los resultados que aparecerán en el registro *A* después del segundo pulso en TRANSFERIR para cada uno de los siguientes casos. Observe que los números se proporcionan en decimal y que el primer número es el que se carga en *B* mediante el primer pulso en CARGAR.

(a)* 2 + 3

(b)* 3 + 7

F (c) 7 + 3

(d) 8 + 3

(e) 9 + 3

F 6-38. Un técnico implementa en un tablero experimental el sumador/restador de la figura 6-14. Durante la prueba descubre que cada vez que se realiza una suma, el resultado es 1 más de lo que se espera y cuando se realiza una resta, el resultado es 1 menos más de lo esperado. ¿Cuál es el probable error que cometió el técnico al conectar este circuito?

6-39.* Describa los síntomas que se producirían en los siguientes puntos en el circuito de la figura 6-14, si las líneas SUMA y RESTA se pusieran en corto.

(a) Las entradas B[3..0] del CI 74LS283

(b) La entrada C_0 del CI 74LS283

(c) Las salidas SUMA (Σ) [3..0]

(d) C_4

SECCIÓN 6-19

Los problemas del 6-40 al 6-45 tratan con los mismos dos arreglos *a* y *b*, los cuales supondremos que se han definido en un archivo fuente de HDL y que tienen los siguientes valores: [a] = [1001011], [b] = [001010]. El arreglo de salida [z] también es un arreglo de ocho bits. Resuelva los problemas del 6-40 al 6-45 con base en esta información. (Suponga que los bits indefinidos en *z* son 0.)

B, H 6-40. Declare estos objetos de datos utilizando su sintaxis favorita de HDL.

B, H 6-41. Proporcione el valor de *z* para cada expresión (se proporcionan expresiones idénticas para AHDL y VHDL):

(a)* z[ ] = a[ ] & b[ ]; z <= a AND b;

(b)* z[ ] = a[ ] # b[ ]; z <= a OR b;

(c) z[ ] = a[ ] $!b[ ]; z <= a XOR NOT b;

(d) z[7..4] = a[3..0] & b[3..0]; z(7 DOWNTO 4) <= a(3 DOWNTO 0) AND b(3 DOWNTO 0);

(e) z[7..1] = a[6..0]; z[0] = GND; z(7 DOWNTO 1) <= a(6 DOWNTO 0); z(0) <= '0';

6-42. ¿Cuál es el valor de cada una de las siguientes variables?

(a) a[3..0] a(3 DOWNTO 0)

(b) b[0] b(0)

(c) a[7] b(7)

B, H 6-43. ¿Cuál es el valor de cada una de las siguientes variables?

(a)* a[5]	a(5)
(b)* b[2]	b(2)
(c)* b[7..1]	b(7 DOWNTO 1)

H 6-44.* Escriba una o más instrucciones en HDL que desplace todos los bits en [a], una posición a la derecha. EL LSB deberá moverse a la posición del MSB. Los datos que se roten deberán terminar en z[ ].

6-45. Escriba una o más instrucciones en HDL que tomen el nibble superior de b y lo coloquen en el nibble inferior de z. El nibble superior de z deberá ser cero.

D, H 6-46. Consulte el problema 6-23. Modifique el código de la figura 6-21 o de la figura 6-22 para agregar una salida de desbordamiento.

D, H, N 6-47.* Otra forma de detectar el desbordamiento con el complemento a 2 es aplicar un XOR al acarreo que va hacia el MSB, con el acarreo que sale del MSB en un sumador/restador. Utilice los mismos números que se dan en el problema 6-23 para verificar esto. Modifique la figura 6-21 o la figura 6-22 para detectar un desbordamiento mediante este método.

D, H 6-48.* Modifique la figura 6-21 o la figura 6-22 para implementar la figura 6-10.

SECCIÓN 6-20

B, H 6-49. Modifique la figura 6-21 o la figura 6-22 para convertirla en un sumador de 12 bits sin usar constantes.

B, H 6-50. Modifique la figura 6-21 o la figura 6-22 para convertirla en un módulo sumador versátil de *n* bits, con una constante que defina el número de bits.

D, A, H 6-51. Escriba un archivo de HDL para crear el equivalente de una ALU 74382 sin usar una macrofunción integrada.

PREGUNTA DE PRÁCTICA

6-52. Defina cada uno de los siguientes términos.

(a) Sumador completo	(f) Acumulador
(b) Complemento a 2	(g) Sumador en paralelo
(c) Unidad aritmética/lógica	(h) Acarreo adelantado
(d) Bit de signo	(i) Negación
(e) Desbordamiento	(j) Registro *B*

APLICACIONES DE MICROCOMPUTADORA

A, D 6-53.* En una ALU de un microprocesador, por lo general (pero no siempre), los resultados de todas las operaciones aritméticas se transfieren al registro acumulador, como en las figuras 6-10, 6-14 y 6-15. En la mayoría de las ALUs de microprocesador también se utiliza el resultado de cada operación aritmética para controlar los estados de varios flip-flops especiales conocidos como *banderas*. El microprocesador utiliza estas banderas cuando realiza decisiones durante la ejecución de ciertos tipos de instrucciones. Las tres banderas más comunes son:

S (bandera de signo). Este FF siempre se encuentra en el mismo estado que el signo del último resultado de la ALU.

Z(bandera de cero). Esta bandera se establece en 1 cada vez que el resultado de una operación de la ALU es igual a 0. En caso contrario se borra para quedar en 0.

C (bandera de acarreo). Este FF siempre se encuentra en el mismo estado que el acarreo proveniente del MSB de la ALU.

Diseñe el circuito lógico para implementar estas banderas, utilizando el sumador/restador de la figura 6-14 como la ALU. Las salidas de suma y la salida C_4 deben usarse para controlar el estado al cual cambiará cada bandera al momento en que ocurra el pulso en TRANSFERIR. Por ejemplo, si la suma es igual a 0 (es decir, 0000) deberá establecerse la bandera Z mediante la PGT de TRANSFERIR; en caso contrario deberá borrarse.

6-54.* Cuando se trabaja con las microcomputadoras, a menudo es necesario mover números binarios desde un registro de ocho bits hacia un registro de 16 bits. Considere los números 01001001 y 10101110, que representan +73 y −82 en forma respectiva, en el sistema de complemento a 2. Determine las representaciones de 16 bits para estos números decimales.

6-55. Compare las representaciones de ocho y 16 bits para el +73 del problema 6-53. Después compare las dos representaciones para el −82. Hay una regla general que puede usarse para convertir con facilidad de la representación de ocho bits a la de 16 bits. ¿Puede usted ver cuál es? Tiene algo que ver con el bit de signo del número de ocho bits.

RESPUESTAS A LAS PREGUNTAS DE REPASO DE LAS SECCIONES

SECCIÓN 6-1

1. (a) 11101 (b) 101.111 (c) 10010000

SECCIÓN 6-2

1. (a) 00001101 (b) 11111001 (c) 10000000 2. (a) −29 (b) −64 (c) +126 3. −2048 a +2047 4. Siete 5. −32768 6. (a) 10000 (b) 10000000 (c) 1000 7. Consulte el texto.

SECCIÓN 6-3

1. Verdadero 2. (a) $100010_2 = -30_{10}$ (b) $000000_2 = 0_{10}$

SECCIÓN 6-4

1. (a) $01111_2 = +15_{10}$ (b) $11111_2 = -{-}1_{10}$ 2. Mediante la comparación del bit de signo de la suma con los bits de signo de los números que se van a sumar.

SECCIÓN 6-5

1. 1100010

SECCIÓN 6-7

1. Cuando la suma de al menos una posición de dígito decimal es mayor que 1001 (9).
2. El factor de corrección se suma tanto a los dígitos de las unidades como a los dígitos de las décadas.

SECCIÓN 6-8

1. 923 2. 3DB 3. 2F, 77EC, 6D

SECCIÓN 6-10

1. Tres; dos 2. (a) $S_2 = 0, C_3 = 1$ (b) $C_5 = 0$

SECCIÓN 6-12

1. Uno; cuatro; cuatro 2. 0100

SECCIÓN 6-14

1. Cinco chips 2. 240 ns 3. 1

SECCIÓN 6-15

1. Para agregar el 1 que se necesita para completar la representación en complemento a 2 del número en el registro *B* 2. 0010 3. 1101 4. Falso; ahí aparece el complemento a 1.

SECCIÓN 6-16

1. $F = 1011$; $OVR = 0$; $C_{N+4} = 0$ 2. $F = 0111$; $OVR = 1$; $C_{N+4} = 1$
3. $F = 1000$; $\Sigma = 01101011$; $C_{N+4} = OVR = 0$; 5. $\Sigma = 11111111$ 6. Ocho

SECCIÓN 6-18

1. Vea el menú HELP de MAX+PLUS, bajo old-style macrofunctions/adders
2. Una descripción de un CI estándar que puede usarse de la biblioteca.

SECCIÓN 6-19

1. (a) 0001(b) 1011 2 . (a) 1010 (b) 1011 3. conmuta[3..0]
4. conmuta(3 DOWNTO 0) 5. [X] = [1,1,1,1] 6. AHDL:xx[ ] = d[ ] # g[ ]; VHDL: x <= d OR g; 7. AHDL:xx[2..1] = d[3..2] $ g[1..0]; VHDL: x(2 DOWNTO 1) <= d(3 DOWNTO 2) XOR g(1 DOWNTO 0);

SECCIÓN 6-21

1. CONSTANT. 2. En AHDL, cerca de la parte superior del archivo fuente. En VHDL, en un bloque PACKAGE cerca de la parte superior del archivo fuente. 3. Permiten los cambios globales del valor de un símbolo que se utiliza en todo el código.
4. val_max −5 representa el número 122. 5. GENERATE se utiliza con una instrucción FOR iterativa para instanciar módulos de código duplicado que puedan conectarse entre sí o en cascada.



CAPÍTULO 7

CONTADORES Y REGISTROS

CONTENIDO

Parte 1

7-1 Contadores asíncronos (de rizo)

7-2 Retraso de propagación en contadores de rizo

7-3 Contadores síncronos (en paralelo)

7-4 Contadores con números MOD $< 2^N$

7-5 Contadores síncronos descendentes y ascendentes/descendentes

7-6 Contadores preajustables

7-7 Contadores síncronos de CI

7-8 Decodificación de un contador

7-9 Análisis de contadores síncronos

7-10 Diseño de un contador síncrono

7-11 Contadores básicos mediante el uso de HDL

7-12 Contadores con todas las características en HDL

7-13 Cómo alambrar módulos de HDL en conjunto

7-14 Máquina de estados

Parte 2

7-15 Registros de circuito integrado

7-16 Entrada en paralelo/salida en paralelo: 74ALS174/74HC174

7-17 Entrada en serie/salida en serie: 74ALS166/74HC166

7-18 Entrada en paralelo/salida en serie: 74ALS165/74HC165

7-19 Entrada en serie/salida en paralelo: 74ALS164/74HC164

7-20 Contadores de registro de desplazamiento

7-21 Diagnóstico de fallas

7-22 Registros en HDL

7-23 Contadores de anillo en HDL

7-24 Monoestables en HDL

OBJETIVOS

Al terminar este capítulo, usted podrá:

- Comprender la operación y las características de los contadores síncronos y asíncronos.
- Construir contadores con números MOD menores de 2^N.
- Construir contadores tanto ascendentes como descendentes.
- Conectar contadores con múltiples etapas.
- Analizar y evaluar varios tipos de contadores.
- Diseñar contadores síncronos de secuencia arbitraria.
- Comprender varios tipos de esquemas utilizados para decodificar distintos tipos de contadores.
- Describir circuitos de contadores mediante el uso de distintos niveles de abstracción en HDL.
- Comparar las principales diferencias entre los contadores de anillo y Johnson.
- Reconocer y comprender la operación de varios tipos de registros de CI.
- Describir los registros de desplazamiento y los contadores de registro de desplazamiento mediante el uso de HDL.
- Aplicar las técnicas existentes de diagnóstico de fallas que se utilizan en los sistemas lógicos combinacionales para diagnosticar fallas en sistemas lógicos secuenciales.

INTRODUCCIÓN

En el capítulo 5 vimos cómo podían conectarse los flip-flops para funcionar como contadores y registros. En ese momento sólo estudiamos los circuitos básicos de contadores y registros. Los sistemas digitales emplean muchas variaciones de estos circuitos básicos, la mayoría en forma de circuitos integrados. En este capítulo veremos cómo pueden combinarse los FFs y las compuertas lógicas para producir diferentes tipos de contadores y registros.

Debido al extenso número de temas en este capítulo, lo hemos dividido en dos partes. En la **PARTE 1** cubriremos los principios de la operación de los contadores, los diversos arreglos de circuitos de contadores y los contadores de CI representativos. En la **PARTE 2** presentaremos varios tipos de registros de CI, contadores de registro de desplazamiento y el diagnóstico de fallas. Cada parte incluye una sección que contiene descripciones en HDL de los contadores y registros.

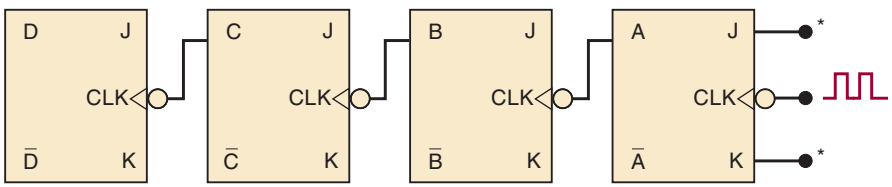
A medida que vaya avanzando en el capítulo tendrá que recurrir con frecuencia al material que cubrimos en los capítulos anteriores. Regrese y repase el material que aprendió con anterioridad cada vez que lo necesite.

PARTE 1

7-1 CONTADORES ASÍNCRONOS (DE RIZO)

La figura 7-1 muestra un circuito contador binario de cuatro bits tal como el que vimos en el capítulo 5. Recuerde los siguientes puntos con respecto a su operación:

1. Los pulsos de reloj se aplican sólo a la entrada *CLK* del flip-flop *A*. En consecuencia, el flip-flop *A* conmutará (cambiará a su estado opuesto) cada vez que los pulsos de reloj hagan una transición negativa (de ALTO a BAJO). Observe que $J = K = 1$ para todos los FFs.
2. La salida normal del flip-flop *A* actúa como la entrada *CLK* para el flip-flop *B*, por lo que el flip-flop *B* conmutará cada vez que la salida de *A* cambie de 1 a 0. De manera similar, el flip-flop *C* conmutará cuando *B* cambie de 1 a 0 y el flip-flop *D* conmutará cuando *C* cambie de 1 a 0.
3. Las salidas de los FFs *D*, *C*, *B* y *A* representan un número binario de cuatro bits, en donde *D* es el MSB. Vamos a suponer que todos los FFs se han borrado para quedar en el estado 0 (no se muestran las entradas de LIMPIA). Las formas de onda en la figura 7-1 muestran que se sigue una secuencia de conteo binario de 0000 a 1111 a medida que se aplican pulsos de reloj en forma continua.
4. Después de que ocurre la NGT (transición negativa) del quinceavo pulso de reloj, los FFs del contador se encuentran en la condición 1111. En la diecisei-



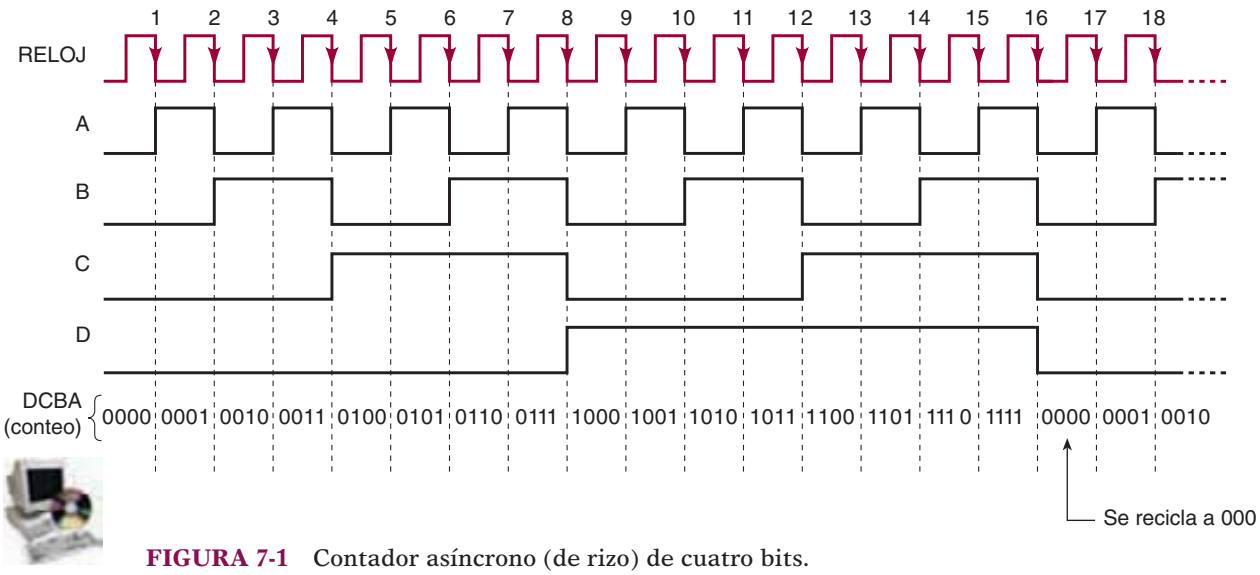


FIGURA 7-1 Contador asíncrono (de rizo) de cuatro bits.

sava NGT el flip-flop *A* cambia de 1 a 0, lo cual hace que el flip-flop *B* cambie de 1 a 0 y así en lo sucesivo, hasta que el contador queda en el estado 0000. En otras palabras, el contador ha pasado por un ciclo completo (de 0000 a 1111) y ha *reciclado* de vuelta a 0000. A partir de este punto empezará un nuevo ciclo de conteo, a medida que se apliquen los subsiguientes pulsos de reloj.

En este contador, cada salida de los FFs excita la entrada *CLK* del siguiente FF. A este tipo de arreglo de contador se le conoce como **contador asíncrono**, ya que los FFs no cambian de estado en sincronía exacta con los pulsos de reloj que se aplican; sólo el flip-flop *A* responde a los pulsos de reloj. El FF *B* debe esperar a que el FF *A* cambie de estado para poder conmutar; el FF *C* debe esperar al FF *B*, y así en lo sucesivo. Por ende, hay un retraso entre las respuestas de los FFs sucesivos. Por lo general este retraso es de 5 a 20 ns por cada FF. Como veremos, en algunos casos este retraso puede ser problemático. A este tipo de contador también se le conoce comúnmente como **contador de rizo** debido a la forma en que responden los FFs, uno después de otro, en un tipo de efecto de rizo. Utilizaremos los términos *contador asíncrono* y *contador de rizo* con el mismo fin, de manera indistinta.

Flujo de la señal

Es una convención en los diagramas esquemáticos dibujar los circuitos (siempre que sea posible) de manera que el flujo de la señal sea de izquierda a derecha, con las entradas a la izquierda y las salidas a la derecha. En este capítulo quebrantaremos con frecuencia esta convención, en especial en los diagramas que muestren contadores. Por ejemplo, en la figura 7-1 las entradas *CLK* de cada FF están a la derecha, las salidas están a la izquierda y la señal de reloj de entrada se muestra saliendo de la derecha. Utilizaremos este arreglo debido a que facilita la comprensión y el seguimiento de la operación del contador (pues el orden de los FFs es el mismo que el orden de los bits en el número binario que representa el contador). En otras palabras, el FF *A* (que es el LSB) es el FF más a la derecha y el FF *D* (que es el MSB) es el FF más a la izquierda. Si nos apegáramos al flujo de señal convencional de izquierda a derecha tendríamos que colocar el FF *A* a la izquierda y el FF *D* a la derecha, lo opuesto a sus posiciones en el número binario que representa al contador. En algunos de los diagramas de contadores que veremos más adelante en este capítulo emplearemos el flujo de señal convencional de izquierda a derecha, para que usted pueda acostumbrarse a verlo.

EJEMPLO 7-1

El contador de la figura 7-1 empieza en el estado 0000 y después se aplican pulsos de reloj. Cierto tiempo después se quitan los pulsos de reloj y los FFs del contador indican 0011. ¿Cuántos pulsos de reloj han transcurrido?

Solución

La respuesta aparente parece ser 3, ya que 0011 es el equivalente binario de 3. Sin embargo, con la información que se proporciona no hay manera de saber si el contador ha reciclado o no. Esto significa que podrían haber transcurrido 19 pulsos de reloj; los primeros 16 regresan el contador de vuelta a 0000 y los últimos 3 lo llevan hasta 0011. O podrían haber transcurrido 35 pulsos (dos ciclos completos y después tres conteos más), o 51 pulsos, etcétera.

Número MOD

El contador de la figura 7-1 tiene 16 estados diferentes (de 0000 a 1111). Por ende, es un *contador de rizo MOD 16*. Recuerde que, por lo general, el **número MOD** es igual al número de estados por los que pasa el contador en cada ciclo completo,

antes de que recicle de vuelta a su estado inicial. El número MOD puede aumentarse con sólo agregar más FFs al contador. Esto es,

$$\text{Número MOD} = 2^N \tag{7-1}$$

en donde N es el número de FFs conectados en el arreglo de la figura 7-1.

EJEMPLO 7-2

Se necesita un contador que cuente el número de elementos que pasan a través de una banda transportadora. Se utiliza una combinación de fotocelda y fuente de luz para generar un pulso individual cada vez que un elemento cruza su camino. El contador debe ser capaz de contar hasta mil elementos. ¿Cuántos FFs se requieren?

Solución

Es muy sencillo determinar qué valor de N se necesita para que $2^N \geq 1000$. Como $2^9 = 512$, 9 FFs no serán suficientes. $2^{10} = 1024$, por lo que 10 FFs producirían un contador que podría contar hasta $1111111111_2 = 1023_{10}$. Por lo tanto, utilizaríamos 10 FF; podríamos usar más de 10 pero sería un desperdicio, ya que cualquier FF más allá del diez no se necesitaría.

División de frecuencia

En el capítulo 5 vimos que en el contador básico cada FF proporciona una forma de onda de salida que es la *mitad* exacta de la frecuencia de la forma de onda en su entrada *CLK*. Para ilustrar esto, suponga que la señal de reloj en la figura 7.1 es de 16 kHz. La figura 7.2 muestra las formas de onda de salida. La forma de onda en la salida *A* es una *onda cuadrada* de 8 kHz, en la salida *B* es de 4 kHz, en la salida *C* es de 2 kHz y en la salida *D* es de 1 kHz. Observe que la salida del flip-flop *D* tiene una frecuencia igual a la original del reloj, dividida entre 16. En general,

> **En cualquier contador, la señal en la salida del último FF (es decir, el MSB) tendrá una frecuencia igual a la frecuencia del reloj de entrada dividida entre el número MOD del contador.**

Por ejemplo, en un contador MOD-16 la salida del último FF tendrá una frecuencia igual a 1/16 de la frecuencia del reloj de entrada. Por ende, también se le puede llamar *contador de división entre 16*. De igual forma, un contador MOD 8 tiene una frecuencia de salida igual a 1/8 de la frecuencia de entrada; es un *contador de división entre 8*.

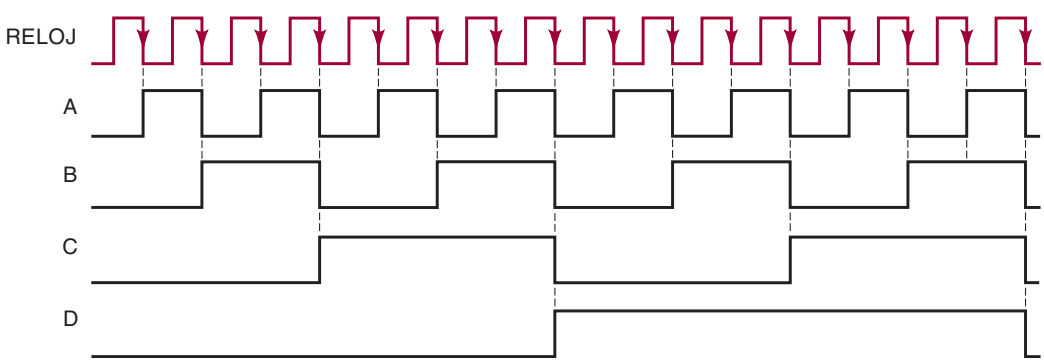


FIGURA 7-2 Formas de onda del contador que muestran la división entre 2 para cada FF.

EJEMPLO 7-3

El primer paso relacionado con la construcción de un reloj digital es tomar la señal de 60 Hz y alimentarla a un circuito de conformación de pulsos tipo monoestable de Schmitt* para producir una onda cuadrada, como se ilustra en la figura 7-3. La onda cuadrada de 60 Hz se aplica después a un contador MOD-60, el cual se utiliza para dividir la frecuencia de 60 Hz entre un valor exacto de 60 para producir una forma de onda de 1 Hz. Esta forma de onda de 1 Hz se alimenta a una serie de contadores, los cuales a su vez se encargan de contar en segundos, minutos, horas, etcétera. ¿Cuántos FFs se requieren para el contador MOD-60?

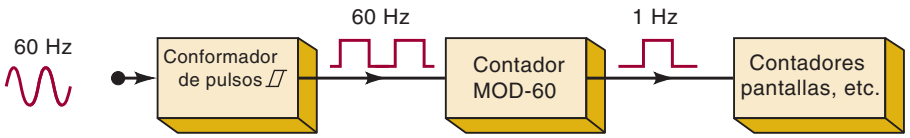


FIGURA 7-3 Ejemplo 7-3.

Solución

No hay una potencia entera de 2 que sea igual a 60. La más cercana es $2^6 = 64$. Por lo tanto, un contador que utilice seis FFs actuará como un contador MOD-64. Es obvio que esto no cumple con el requerimiento. Parece que no hay solución si se utiliza un contador del tipo que se muestra en la figura 7-1. En parte esto es verdad; en la sección 7-4 veremos cómo modificar los contadores binarios básicos para que pueda obtenerse casi *cualquier* número MOD y así no estaremos limitados a los valores de 2^N.

PREGUNTAS DE REPASO

1. *Verdadero o falso:* en un contador asíncrono todos los FFs cambian de estado al mismo tiempo.
2. Suponga que el contador en la figura 7-1 tiene el conteo 0101. ¿Cuál será el conteo después de 27 pulsos de reloj?
3. ¿Cuál sería el número MOD del contador si se agregaran tres FFs más?

7-2 RETRASO DE PROPAGACIÓN EN CONTADORES DE RIZO

Los contadores de rizo son el tipo más simple de contadores binarios, ya que requieren la menor cantidad de componentes para producir una operación de conteo dada. No obstante tienen una gran desventaja debido a su principio básico de operación: cada FF se dispara mediante la transición en la salida del FF anterior. Debido al tiempo de retraso de propagación inherente (t_{pd}) de cada FF, esto significa que el segundo FF no responderá sino hasta un tiempo t_{pd} después de que el primer FF reciba una transición activa de reloj; el tercer FF no responderá sino hasta un tiempo igual a $2 \times t_{pd}$ después de esa transición del reloj; y así en lo sucesivo. En otras palabras, los retrasos de propagación de los FFs se acumulan de manera que el N-ésimo FF no puede cambiar de estado sino hasta un tiempo igual a $N \times t_{pd}$ después de que ocurre la transición del reloj. Esto se ilustra en la figura 7-4, en donde se muestran las formas de onda para un contador de rizo de tres bits.

El primer conjunto de formas de onda en la figura 7-4(a) muestra una situación en la que se produce un pulso de entrada cada 1000 ns (el periodo del reloj T = 1000 ns) y se supone que cada FF tiene un retraso de propagación de 50 ns ($t_{pd} = 50$

* Consulte la sección 5-21.

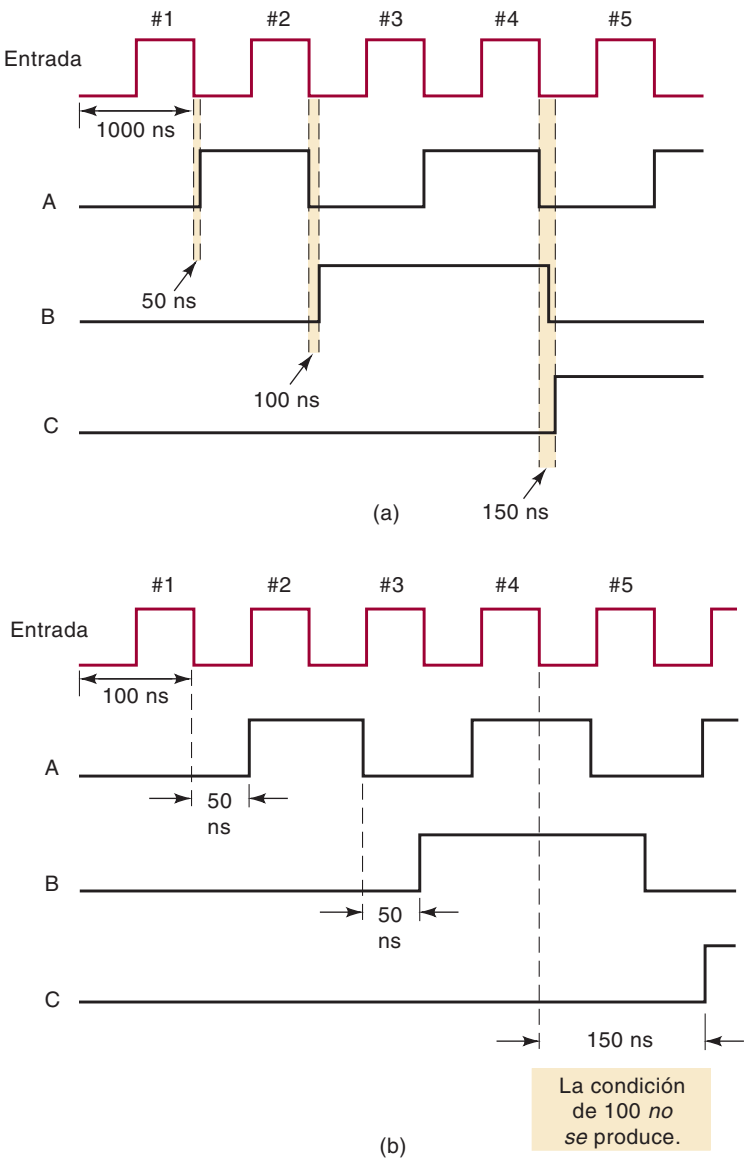


FIGURA 7-4 Formas de onda de un contador de rizo de tres bits, en donde se ilustran los efectos de los retrasos de propagación de los FFs para distintas frecuencias del pulso de entrada.

ns). Observe que la salida del flip-flop *A* conmuta 50 ns después de la NGT de cada pulso de entrada. De manera similar, *B* conmuta 50 ns después de que *A* cambia de 1 a 0 y *C* conmuta 50 ns después de que *B* cambia de 1 a 0. Como resultado, cuando ocurre la cuarta NGT de entrada la salida *C* cambia a ALTO después de un retraso de 150 ns. En esta situación el contador opera de manera apropiada en el sentido en que en cierto momento los FFs llegan a sus estados correctos para representar el conteo binario. No obstante, la situación se empeora si los pulsos de entrada se aplican a una frecuencia mucho mayor.

Las formas de onda en la figura 7-4(b) muestran lo que ocurre si los pulsos de entrada se producen una vez cada 100 ns. De nuevo, cada salida de FF responde 50 ns después de la transición de 1 a 0 en su entrada *CLK* (observe el cambio en la escala de tiempo relativa). Es de particular interés la situación después del flanco de caída del *cuarto* pulso de entrada, en donde la salida *C* no cambia a ALTO sino hasta 150 ns después, que viene siendo el mismo tiempo que transcurre hasta que la salida *A* cambia a ALTO en respuesta al *quinto* pulso de entrada. En otras palabras, la condición $C = 1, B = A = 0$ (conteo de 100) nunca aparece, ya que la frecuencia de entrada es demasiado alta. Esto podría provocar un grave problema si esta condición fuera a utilizarse para controlar alguna otra operación en un sistema digital.

Se pueden evitar problemas como éste si el periodo entre los pulsos de entrada se alarga más que el retraso de propagación total del contador. Esto es, para que el contador opere en forma correcta necesitamos que

$$T_{reloj} \geq N \times t_{pd} \tag{7-2}$$

en donde N = el número de FFs. Dicho en términos de la frecuencia del reloj de entrada, la frecuencia máxima que puede utilizarse se da mediante la siguiente ecuación:

$$f_{máx} = \frac{1}{N \times t_{pd}} \tag{7-3}$$

Suponga, por ejemplo, que se construye un contador de rizo de cuatro bits utilizando el flip-flop J-K 74LS112. La tabla 5-2 muestra que el 74LS112 tiene los retrasos de propagación $t_{PLH} = 16$ ns y $t_{PHL} = 24$ ns de *CLK* a *Q*. Para calcular $f_{máx}$ supondremos el "peor caso"; es decir, utilizaremos $t_{pd} = t_{PLH} = 24$ ns, para que

$$f_{máx} = \frac{1}{4 \times 24\text{ ns}} = 10.4\text{ MHz}$$

Es evidente que a medida que aumenta el número de FFs en el contador, se incrementa el tiempo de propagación total y se decrementa $f_{máx}$. Por ejemplo, un contador de rizo que utiliza seis FFs 74LS112 tendrá

$$f_{máx} = \frac{1}{6 \times 24\text{ ns}} = 6.9\text{ MHz}$$

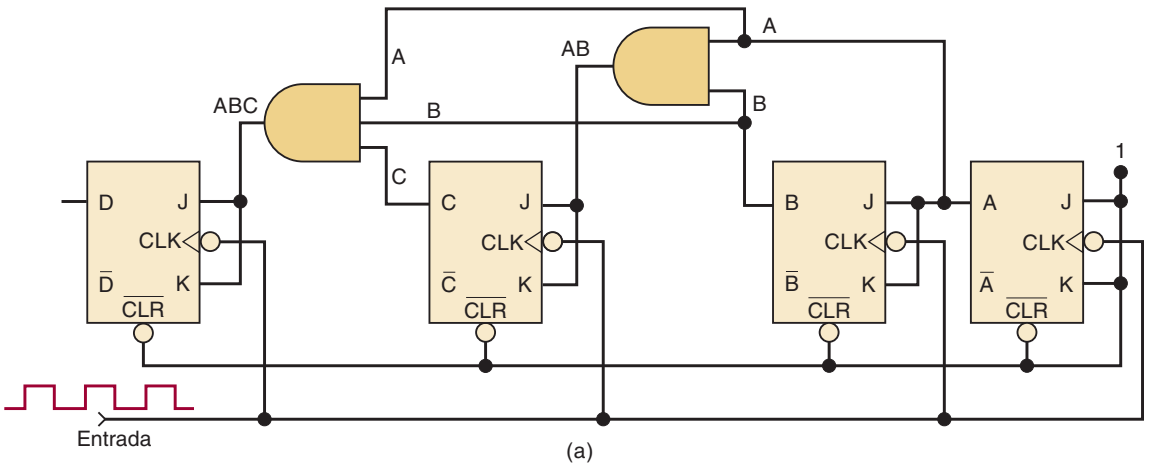
Por lo tanto, los contadores asíncronos no son útiles a frecuencias muy altas, en especial para los contadores con grandes números de bits. Otro problema que ocasionan los retrasos de propagación en los contadores asíncronos ocurre cuando tratamos de detectar por medio electrónico (*decodificar*) los estados de salida del contador. Si analiza con detalle la figura 7-4(a), durante un breve tiempo (50 ns en nuestro ejemplo) justo después del estado 011 verá que el estado 010 ocurre antes que el estado 100. Es obvio que ésta no es la secuencia correcta de conteo binario, y aunque el ojo humano es demasiado lento como para ver este estado temporal, nuestros circuitos digitales serán lo bastante rápidos como para detectarlo. Estos patrones de conteo erróneos pueden generar lo que se conoce como **deformaciones** en las señales producidas por sistemas digitales que utilizan contadores asíncronos. A pesar de su simpleza, estos problemas limitan la utilidad de los contadores asíncronos en aplicaciones digitales.

PREGUNTAS DE REPASO

1. Explique por qué la limitación de frecuencia máxima de un contador de rizo disminuye a medida que se agregan más FFs al contador.
2. Cierto flip-flop J-K tiene $t_{pd} = 12$ ns. ¿Cuál es el contador MOD más grande que puede construirse a partir de estos FFs y que pueda operar hasta 10 MHz?

7-3 CONTADORES SÍNCRONOS (EN PARALELO)

Los problemas que se encuentran en los contadores de rizo son ocasionados por los retrasos de propagación acumulados en los FF; dicho de otra forma, los FFs no cambian de estado todos al mismo tiempo, en sincronía con los pulsos de entrada. Estas limitaciones pueden resolverse con el uso de los **contadores en paralelo** o **síncronos**,



(a)

Conteo	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
0	0	0	0	0
.	.	.	.	.
.	.	.	.	.
.	.	etc.	.	.

(b)



FIGURA 7-5 Contador MOD-16 síncrono. Cada FF está sincronizado por la NGT de la señal de entrada de reloj, de manera que todas las transiciones de los FFs ocurran al mismo tiempo.

en los cuales todos los FFs se disparan al mismo tiempo (en paralelo) mediante los pulsos de entrada de reloj.

Debido a que los pulsos de entrada se aplican a todos los FFs, debe utilizarse algún medio para controlar cuándo debe conmutar un FF y cuándo no debe ser afectado por un pulso de reloj. Esto se logra mediante el uso de las entradas *J* y *K*, y se ilustra en la figura 7-5 para un contador síncrono MOD-16 de cuatro bits.

Si comparamos el arreglo del circuito para este contador síncrono con su contraparte asíncrona de la figura 7-1, podremos ver las siguientes diferencias notables:

- Las entradas *CLK* de todos los FFs están conectadas en conjunto, de manera que la señal de reloj de entrada se aplique a cada FF al mismo tiempo.
- Sólo el flip-flop *A*, el LSB, tiene sus entradas *J* y *K* de manera permanente en el nivel ALTO. Las entradas *J*, *K* de los demás FFs se excitan mediante cierta combinación de salidas de los FFs.
- El contador síncrono requiere más circuitos que el contador asíncrono.

Operación del circuito

Para que este circuito pueda contar en forma apropiada, en una NGT dada del reloj sólo los FFs que se supone deben conmutar en esa NGT tienen la condición $J = K = 1$ cuando ocurre esa NGT. Analicemos la secuencia de conteo en la figura 7-5(b) para ver lo que esto significa para cada FF.

La secuencia de conteo muestra que el flip-flop A debe cambiar de estado en cada NGT. Por esta razón sus entradas J y K se mantienen en ALTO de manera permanente, para que conmute en cada NGT de la entrada de reloj.

La secuencia de conteo muestra que el flip-flop B debe cambiar de estado en cada NGT que se produzca mientras que $A = 1$. Por ejemplo, cuando el conteo es 0001 la siguiente NGT debe conmutar a B al estado 1; cuando el conteo es 0011, la siguiente NGT debe conmutar a B al estado 0; y así en lo sucesivo. Para lograr esta operación se conecta la salida de A con las entradas J y K del flip-flop B, de manera que $J = K = 1$ sólo cuando $A = 1$.

La secuencia de conteo muestra que el flip-flop C debe cambiar de estado en cada NGT que se produzca mientras que $A = B = 1$. Por ejemplo, cuando el conteo es 0011 la siguiente NGT debe conmutar a C al estado 1; cuando el conteo es 0111 la siguiente NGT debe conmutar a C al estado 0; y así en lo sucesivo. Al conectar la señal lógica AB a las entradas J y K del FF C, este FF conmutará sólo cuando $A = B = 1$.

De manera similar, podemos ver que el flip-flop D debe conmutar en cada NGT que se produzca mientras que $A = B = C = 1$. Cuando el conteo es 0111, la siguiente NGT debe conmutar a D al estado 1; cuando el conteo es 1111 la siguiente NGT debe conmutar a D al estado 0. Al conectar la señal lógica ABC a las entradas J y K del FF D, este FF conmutará sólo cuando $A = B = C = 1$.

Por lo tanto, podemos establecer el principio básico para construir un contador de la siguiente manera:

> **Cada FF deberá tener sus entradas *J* y *K* conectadas de manera que estén en ALTO sólo cuando las salidas de todos los FFs de menor orden se encuentren en el estado ALTO.**

Ventaja de los contadores síncronos en comparación con los contadores asíncronos

En un contador en paralelo todos los FFs cambiarán de estado al mismo tiempo; esto es, todos están sincronizados con las NGTs de los pulsos de entrada del reloj. En consecuencia y a diferencia de los contadores asíncronos, los retrasos de propagación de los FFs no se acumulan para producir el retraso total, sino que el tiempo de respuesta total de un contador síncrono como el de la figura 7-5 es el tiempo que requiere *un* FF para conmutar, más el tiempo para que los nuevos niveles lógicos se propaguen a través de *una sola* compuerta AND para llegar a las entradas J, K. Por lo tanto, para un contador síncrono,

$$\text{retraso total} = t_{pd} \text{ del FF} + t_{pd} \text{ de la compuerta AND}$$

Este retraso total es el mismo, sin importar cuántos FFs haya en el contador y, por lo general, será mucho menor que en un contador asíncrono con el mismo número de FFs. Por ende, un contador síncrono puede operar a una frecuencia de entrada mucho más alta. Desde luego que los circuitos del contador síncrono son más complejos que los del contador asíncrono.

CIs reales

Existen muchos contadores de CI síncronos tanto en la familia lógica TTL como en la CMOS. Algunos de los dispositivos que se utilizan con más frecuencia son:

- 74ALS160/162, 74HC160/162: contadores síncronos de décadas.
- 74ALS161/163, 74HC161/163: contadores MOD-16 síncronos.

EJEMPLO 7-4

(a) Determine el valor de $f_{máx}$ para el contador de la figura 7-5(a) si el t_{pd} para cada FF es de 50 ns y el t_{pd} para cada compuerta AND es de 20 ns. Compare este valor con $f_{máx}$ para un contador de rizo MOD-16.

(b) ¿Qué debe hacerse para convertir este contador en MOD-32?

(c) Determine el valor de $f_{máx}$ para el contador MOD-32 en paralelo.

Solución

(a) El retraso total que debe permitirse entre los pulsos de reloj de entrada es igual al t_{pd} del FF + el t_{pd} de la compuerta AND. Por ende, $T_{reloj} \geq 50 + 20 = 70$ ns, y en consecuencia el contador en paralelo tiene

$$f_{máx} = \frac{1}{70 \text{ ns}} = 14.3 \text{ MHz} \quad \text{contador en paralelo}$$

Un contador de rizo MOD-16 utiliza cuatro FFs con $t_{pd} = 50$ ns. Por lo tanto, la $f_{máx}$ para el contador de rizo es

$$f_{máx} = \frac{1}{4 \times 50 \text{ ns}} = 5 \text{ MHz} \quad \text{contador de rizo}$$

(b) Debe agregarse un quinto FF, ya que $2^5 = 32$. La entrada *CLK* de este FF también está enlazada a los pulsos de entrada. Sus entradas *J* y *K* se alimentan mediante la salida de una compuerta AND de cuatro entradas *A*, *B*, *C* y *D*.

(c) La $f_{máx}$ se determina igual que en (a), sin importar el número de FFs en el contador en paralelo. Por lo tanto, $f_{máx}$ sigue siendo 14.3 MHz.

PREGUNTAS DE REPASO

1. ¿Cuál es la ventaja de un contador síncrono, en comparación con un contador asíncrono? ¿Cuál es la desventaja?
2. ¿Cuántos dispositivos lógicos se requieren para un contador MOD-64 en paralelo?
3. ¿Qué señal lógica excita las entradas *J*, *K* del flip-flop del MSB para el contador de la pregunta 2?

7-4 CONTADORES CON NÚMEROS MOD < 2^N

El contador síncrono básico de la figura 7-5 está limitado a números MOD que sean iguales a 2^N, en donde *N* es el número de FFs. En realidad este valor es el número MOD máximo que puede obtenerse mediante el uso de *N* flip-flops. El contador básico puede modificarse para producir números MOD menores que 2^N si se permite al contador *omitir estados* que, por lo general, forman parte de la secuencia de conteo. Uno de los métodos más comunes para esto se ilustra en la figura 7-6, en donde se muestra un contador de tres bits. Si descartamos la compuerta NAND por un momento, podremos ver que el contador es del tipo binario MOD-8, el cual contará en secuencia desde 0000 hasta 1111. No obstante, la presencia de la compuerta NAND alterará esta secuencia de la siguiente manera:

1. La salida NAND se conecta a las entradas LIMPIAR asíncronas de cada FF. Mientras que la salida de la compuerta NAND esté en ALTO no tendrá efecto sobre el contador. Pero cuando cambie a BAJO borrará todos los FFs, lo cual provocará que el contador cambie de inmediato al estado 000.

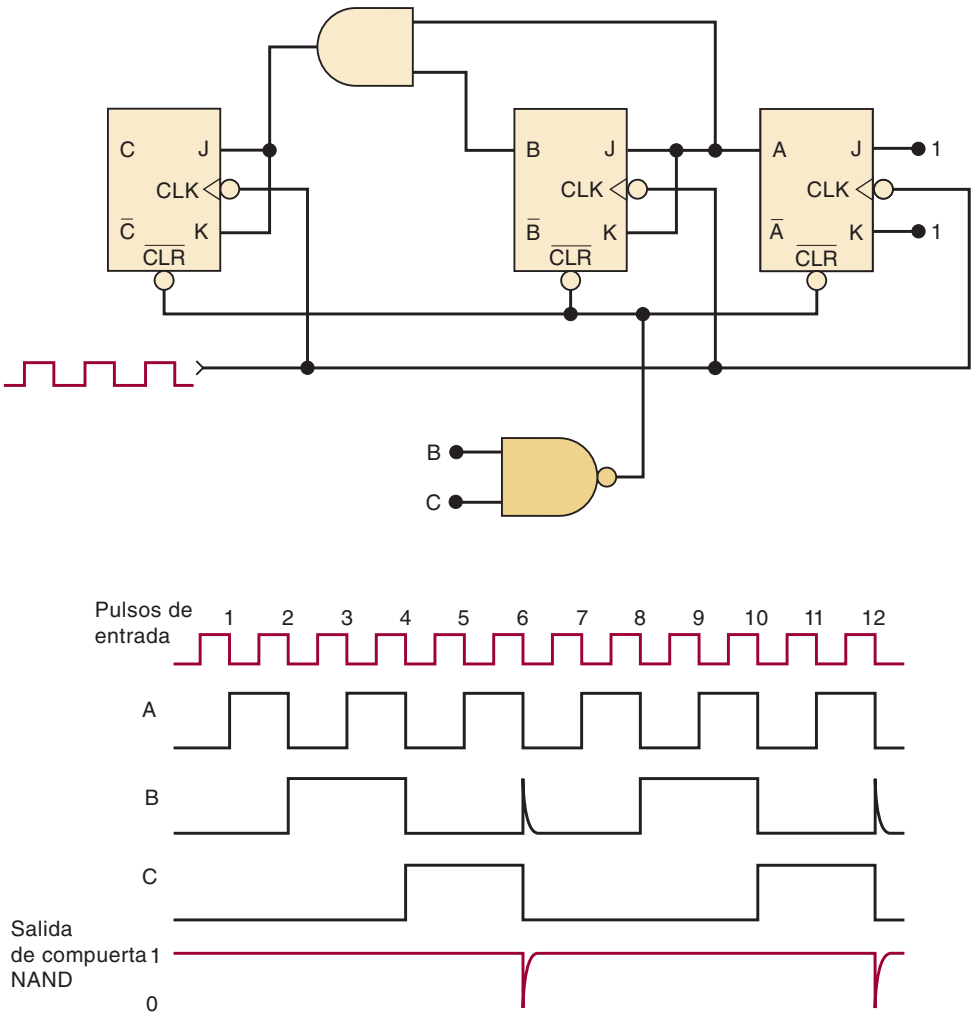


FIGURA 7-6 Contador MOD-6 que se produce al borrar un contador MOD-8 cuando llega a un conteo de seis (110).

2. Las entradas de la compuerta NAND son las salidas de los flip-flops *B* y *C*, por lo que la salida de la compuerta NAND cambiará a BAJO siempre que $B = C = 1$. Esta condición se producirá cuando el contador cambie del estado 101 al 110 en la NGT del pulso de entrada 6. El nivel BAJO en la salida de la compuerta NAND borrará de inmediato (por lo general, en unos cuantos nanosegundos) el contador para que quede en el estado 000. Una vez que se hayan borrado los FFs, la salida de la compuerta NAND cambiará de vuelta al nivel ALTO, ya que la condición $B = C = 1$ dejó de existir.
3. Por lo tanto, la secuencia de conteo es

CBA

000 ←
001
010
011
100
101
110 → (estado temporal necesario para borrar el contador)

Aunque el contador cambia al estado 110, permanece ahí sólo durante unos cuantos nanosegundos antes de reciclarse a 000. Por lo tanto, en esencia podemos decir que este contador cuenta desde 000 (cero) hasta 101 (cinco) y después se recicla a 000. Omite los estados 110 y 111, de manera que sólo pasa por seis estados distintos; por lo tanto es un contador MOD-6.

Observe que la forma de onda en la salida de *B* contiene un *pico* o *deformación* producida por la ocurrencia momentánea del estado 110 antes de que el FF se borre. Esta deformación es muy estrecha y no producirá ninguna indicación visual en LEDs indicadores o en pantallas numéricas. No obstante, podría provocar un problema si la salida *B* se utilizara para excitar otros circuitos fuera del contador. Hay que observar también que la salida *C* tiene una frecuencia igual a un sexto de la frecuencia de entrada; en otras palabras, este contador MOD-6 ha dividido la frecuencia de entrada entre *seis*. La forma de onda en *C no es* una onda cuadrada simétrica (ciclo de trabajo del 50 por ciento) debido a que está en ALTO sólo durante dos ciclos de reloj, mientras que está en BAJO durante cuatro ciclos.

Diagrama de transición de estados

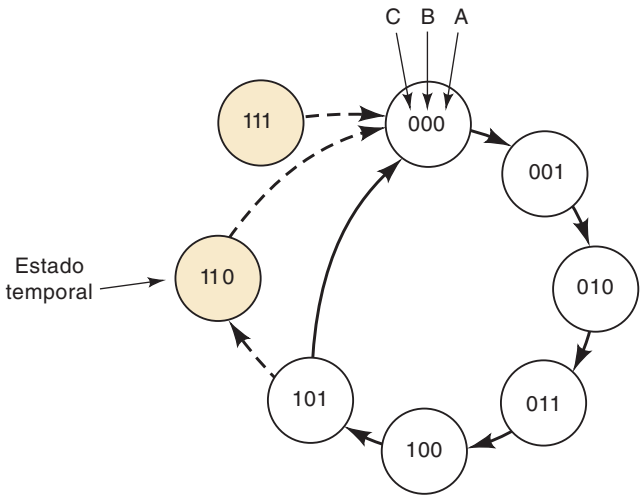
La figura 7-7(a) es el diagrama de transición de estados para el contador MOD-6 de la figura 7-6. En este diagrama se muestra cómo los FFs *C*, *B* y *A* cambian de estado a medida que se aplican pulsos en la entrada *CLK* del flip-flop *A*. Recuerde que cada círculo representa uno de los posibles estados del contador y que las flechas indican cómo un estado cambia a otro en respuesta a un pulso de reloj de entrada.

Si suponemos un conteo inicial de 000, el diagrama nos muestra que los estados del contador cambian, por lo general, en forma ascendente hasta llegar al conteo de 101. Cuando se produce el siguiente pulso de reloj, el contador cambia de manera temporal al conteo 110 antes de pasar al conteo 000 estable. Las líneas punteadas indican la naturaleza temporal del estado 110. Como dijimos antes, la duración de este estado temporal es tan breve que para la mayoría de los fines podemos considerar que el contador cambia en forma directa de 101 a 000 (flecha sólida).

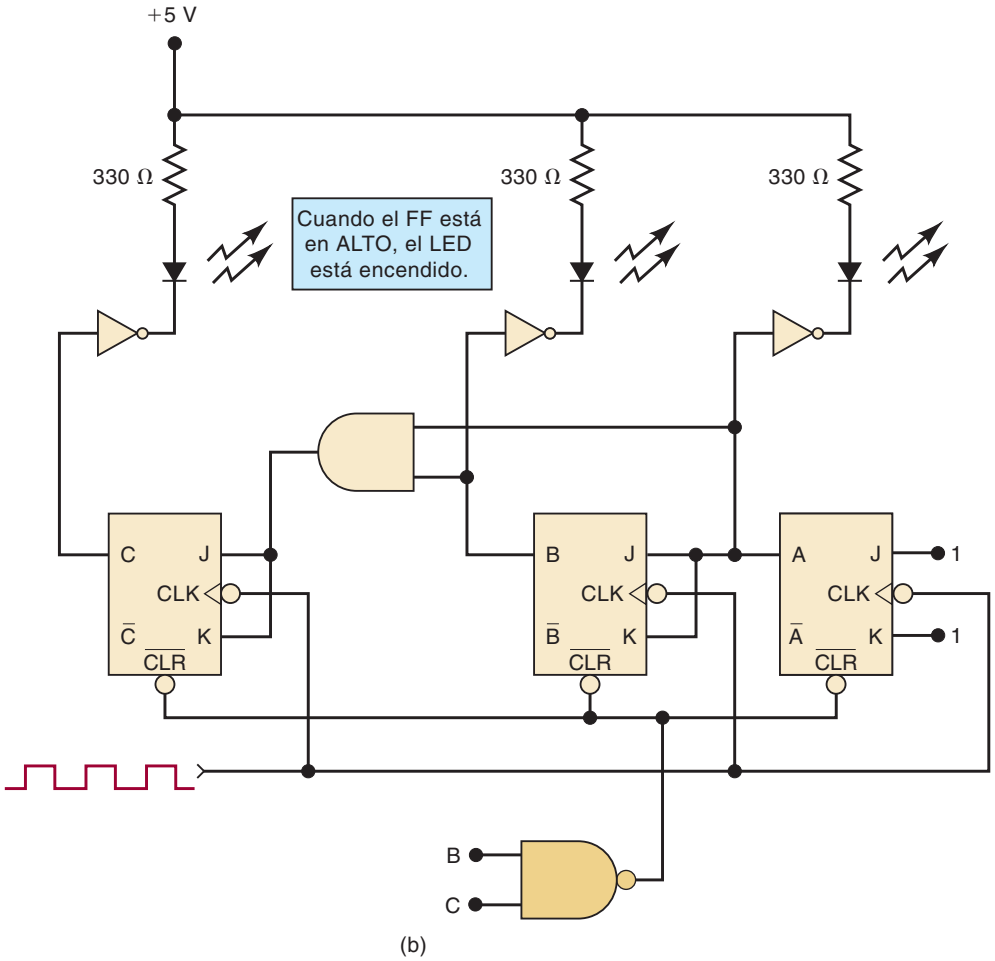
Observe que no hay una flecha que vaya al estado 111, ya que el contador nunca podrá avanzar hasta ese estado. No obstante, puede producirse el estado 111 al momento de encender el circuito, cuando los FFs prenden en estados aleatorios. Si esto ocurre, la condición 111 producirá un nivel BAJO en la salida de la compuerta NAND y de inmediato se borrará el contador para quedar en 000. Por ende, el estado 111 también es una condición temporal que termina en 000.

Visualización de los estados de un contador

Algunas veces durante la operación normal, y con mucha frecuencia durante la prueba, es necesario tener una pantalla visual de la manera en que el contador cambia de estados en respuesta a los pulsos de entrada. Más adelante veremos con detalle varias formas de hacer esto. Por ahora, la figura 7-7(b) muestra uno de los métodos más simples, en el que se utilizan LEDs indicadores individuales para cada salida de FF. Cada salida se conecta a un INVERSOR cuya salida proporciona la ruta de la corriente para el LED. Por ejemplo, cuando la salida *A* está en ALTO, la salida del INVERSOR cambia a BAJO y el LED se enciende. Un LED encendido indica que $A = 1$. Cuando la salida *A* está en BAJO, la salida del INVERSOR está en ALTO y el LED se apaga. Cuando pasa esto, nos indica que $A = 0$.



(a)



(b)

FIGURA 7-7 (a) Diagrama de transición de estados para el contador MOD-6 de la figura 7-6. (b) A menudo se utilizan LEDs para visualizar los estados de un contador.

EJEMPLO 7-6

(a) ¿Cuál será el estado de los LEDs cuando el contador tenga el conteo de cinco?

(b) ¿Qué mostrarán los LEDs cuando el contador esté sincronizado por una entrada de 1 kHz?

(c) ¿Será visible el estado 110 en los LEDs?

Solución

(a) Como $5_{10} = 101_2$, los LEDs 2^0 y 2^2 estarán encendidos y el LED 2^1 estará apagado.

(b) A 1 kHz, los LEDs estarán cambiando entre encendido y apagado tan rápido que parecerá (al ojo humano) que están encendidos todo el tiempo, con un brillo aproximado a la mitad del normal.

(c) No; el estado 110 persistirá durante sólo unos cuantos nanosegundos, hasta que el contador se recicle a 000.

Cambiar el número MOD

El contador de las figuras 7-6 y 7-7 es del tipo MOD-6 debido a la elección de entradas que van a la compuerta NAND. Si se cambian estas entradas, podrá obtenerse cualquier número MOD que se desee. Por ejemplo, si utilizamos una compuerta NAND de tres entradas *A*, *B* y *C*, el contador funcionaría en forma normal hasta que llegara a la condición 111, punto en el cual se restablecería de inmediato al estado 000. Si ignoramos la excursión temporal al estado 111, el contador iría de 000 a 110 y después se reciclaría de nuevo a 000, lo cual nos lleva a deducir que es un contador MOD-7 (7 estados).

EJEMPLO 7-6

Determine el número MOD del contador en la figura 7-8(a). Determine además la frecuencia en la salida *D*.

Solución

Éste es un contador de cuatro bits, que, por lo general, contaría desde 0000 hasta 1111. Las entradas NAND son *D*, *C* y *B*, lo cual significa que el contador se reciclará de inmediato a 0000 cuando se llegue al conteo de 1110 (14 decimal). En consecuencia, el contador en realidad tiene 14 estados estables (de 0000 a 1101) y es, por lo tanto, del tipo *MOD-14*. Como la frecuencia de entrada es de 30 kHz, la frecuencia en la salida *D* será

$$\frac{30\text{ kHz}}{14} = 2.14\text{ kHz}$$

Procedimiento general

Para construir un contador que empiece desde 0 y que tenga un número MOD *X*:

1. Busque el número más pequeño de FFs tal que $2^N \geq X$, y conéctelos para formar un contador. Si $2^N = X$, no realice los pasos 2 y 3.
2. Conecte una compuerta NAND a las entradas LIMPIAR asíncronas de todos los FFs.
3. Determine cuáles FFs estarán en el estado ALTO cuando el conteo sea igual a *X*; después conecte las salidas normales de estos FFs a las entradas de la compuerta NAND.

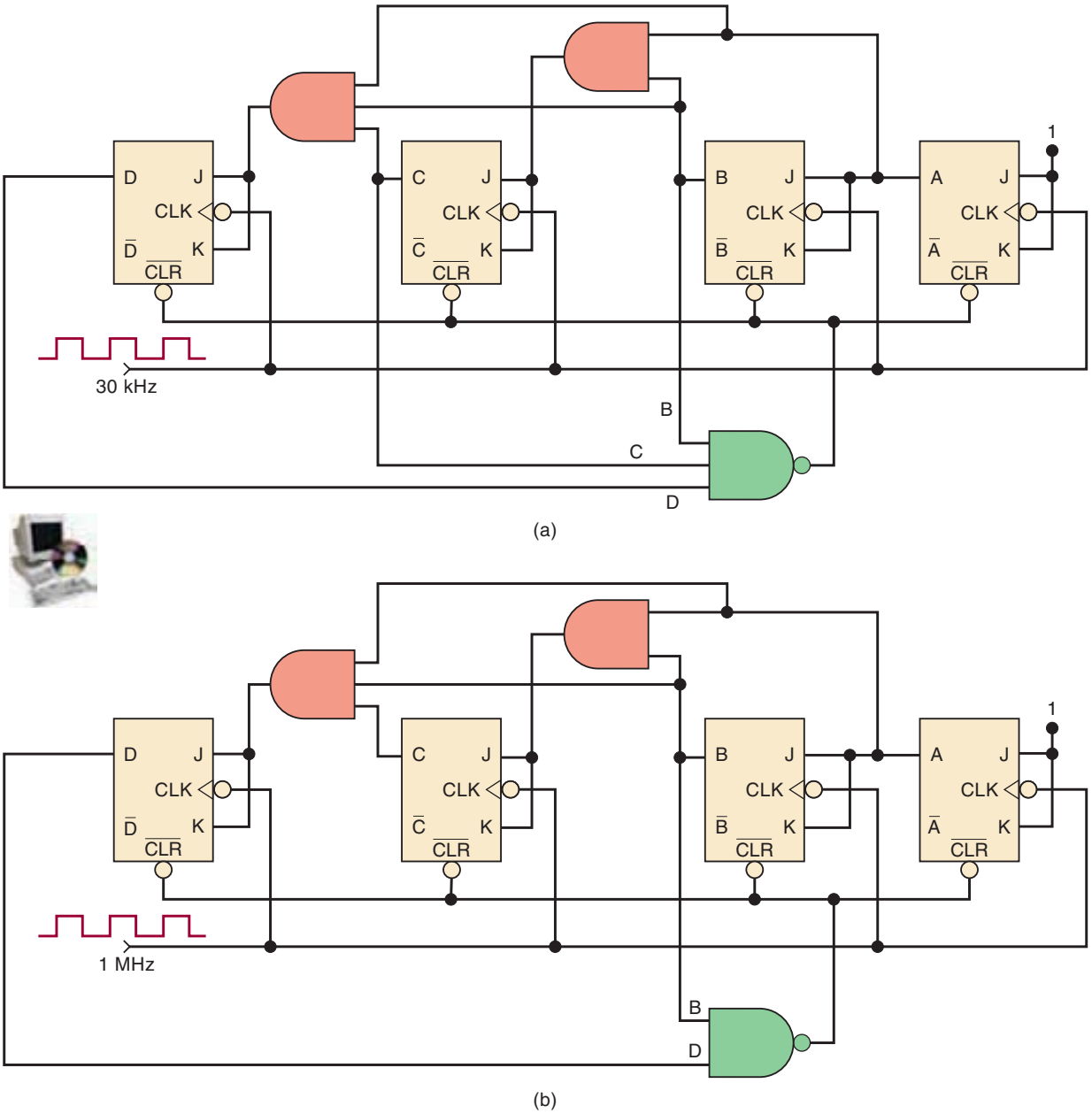


FIGURA 7-8 (a) Contador de rizo MOD-14; (b) Contador de rizo MOD-10 (de décadas).

EJEMPLO 7-7

Construya un contador MOD-10 que vaya desde 0000 (cero) hasta 1001 (9 decimal).

Solución

$2^3 = 8$ y $2^4 = 16$; entonces se requieren cuatro FFs. Como el contador debe tener una operación estable hasta el conteo de 1001, debe restablecerse a cero cuando llegue al conteo de 1010. Por lo tanto, las salidas de los FFs *D* y *B* deberán conectarse como entradas para la compuerta NAND. La figura 7-8(b) muestra el arreglo.

Contadores de décadas/contadores BCD

Al contador MOD-10 del ejemplo 7-7 también se le conoce como **contador de décadas**. De hecho, un contador de décadas es cualquiera que tenga 10 estados distintos,

sin importar la secuencia. A un contador de décadas como el de la figura 7-8(b), que cuenta en secuencia desde 0000 (cero) hasta 1001 (9 decimal), también se le conoce comúnmente como **contador BCD** ya que sólo utiliza los 10 grupos de código BCD 0000, 0001, . . . , 1000, y 1001. Para reiterar, cualquier contador MOD-10 es de décadas y cualquier contador de décadas que cuente en binario desde 0000 hasta 1001 es del tipo BCD.

Los contadores de décadas (en especial el tipo BCD) tienen un amplio uso en aplicaciones en las que se van a contar pulsos o eventos y los resultados se mostrarán en algún tipo de indicador numérico decimal. Más adelante examinaremos esto con mayor detalle. Un contador de décadas también se utiliza a menudo para dividir la frecuencia de un pulso *exactamente* entre 10. Los pulsos de entrada se aplican a las entradas de reloj en paralelo, y los pulsos de salida se toman de la salida del flip-flop *D*, que tiene una frecuencia igual a un décimo de la frecuencia de la señal de entrada.

EJEMPLO 7-8

En el ejemplo 7-3 se necesitaba un contador MOD-60 para dividir la frecuencia de línea de 60 Hz hasta 1 Hz. Construya un contador MOD-60 apropiado.

Solución

$2^5 = 32$ y $2^6 = 64$, por lo que necesitamos seis FFs, como muestra la figura 7-9. El contador deberá borrarse cuando llegue a 60 (111100). Entonces, las salidas de los flip-flops Q_5, Q_4, Q_3 y Q_2 deberán conectarse a la compuerta NAND. La salida del flip-flop Q_5 tendrá una frecuencia de 1 Hz.

PREGUNTAS DE REPASO

1. ¿Cuáles salidas de los FFs deberán conectarse a la compuerta NAND que se encarga de borrar los FFs, para formar un contador MOD-13?
2. *Verdadero o falso:* todos los contadores BCD son de décadas.
3. ¿Cuál es la frecuencia de salida de un contador de décadas que se sincroniza a partir de una señal de 50 kHz?

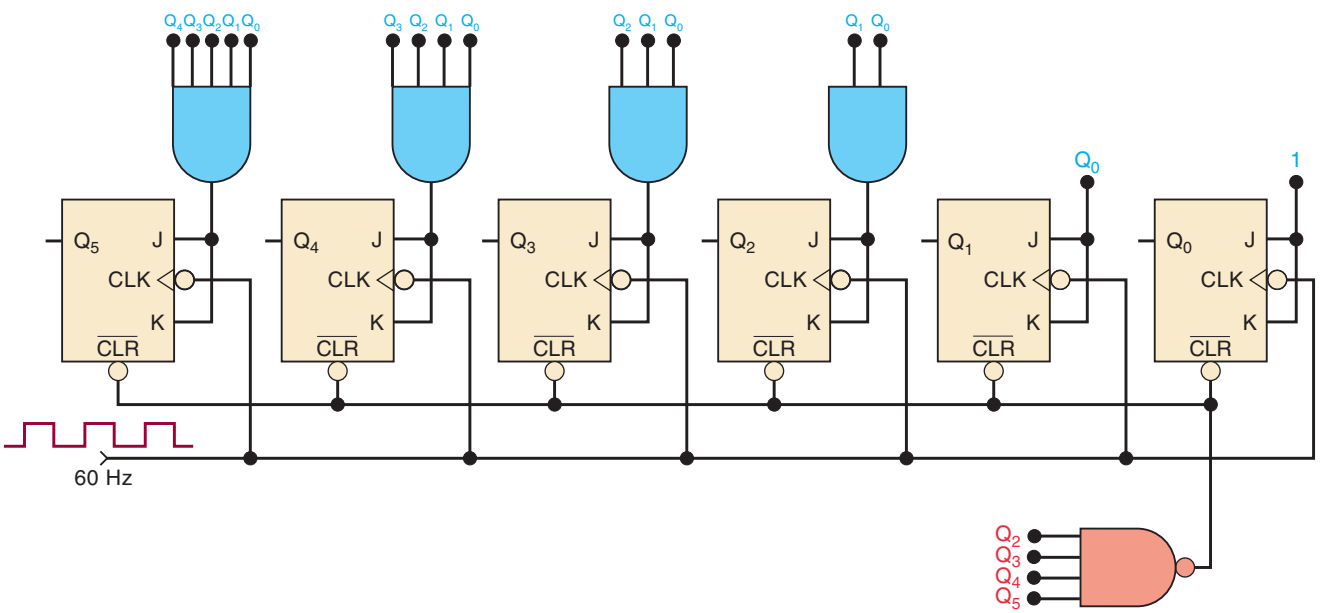
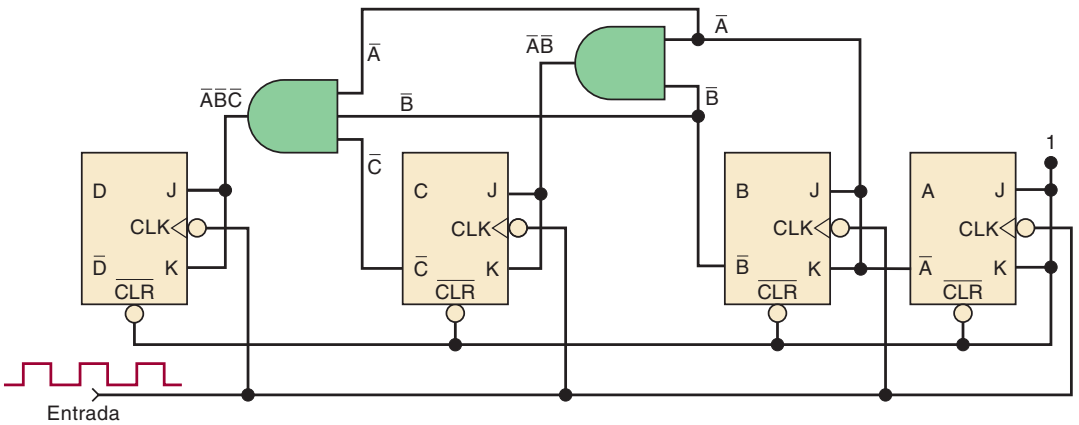


FIGURA 7-9 Contador MOD-60.

7-5 CONTADORES SÍNCRONOS DESCENDENTES Y ASCENDENTES/DESCENDENTES

En la sección 7-3 vimos que al utilizar la salida de los FFs de menor orden para controlar la conmutación de cada FF se crea un **contador ascendente** síncrono. Un **contador descendente** síncrono se construye en forma similar, sólo que utilizamos las salidas de los FF invertidas para controlar las entradas *J*, *K* de mayor orden. Si comparamos el contador descendente síncrono MOD-16 de la figura 7-10 con el contador ascendente de la figura 7-5 veremos que sólo necesitamos sustituir la correspondiente salida invertida del FF en vez de las salidas *A*, *B* y *C*. Para una secuencia de conteo descendente, el FF (A) del LSB aún necesita conmutar con cada NGT de la señal de entrada del reloj. El flip-flop *B* debe cambiar de estado en la siguiente NGT del reloj cuando $A = 0$ ($\overline{A} = 1$). El flip-flop *C* cambia de estado cuando $A = B = 0$ ($\overline{A} \cdot \overline{B} = 1$) y el flip-flop *D* cambia de estado cuando $A = B = C = 0$ ($\overline{A} \cdot \overline{B} \cdot \overline{C} = 1$). Esta configuración del circuito producirá la secuencia de conteo: 15, 14, 13, 12, …, 3, 2, 1, 0, 15, 14…, como se muestra en el diagrama de sincronización.

La figura 7-11(a) muestra cómo formar un **contador ascendente/descendente** en paralelo. La entrada de control Arriba/Abajo controla si se alimentan las salidas normales o las salidas invertidas del FF a las entradas *J* y *K* de los FFs sucesivos. Cuando Arriba/Abajo se mantiene en ALTO se habilitan las compuertas AND 1 y 2 mientras que se deshabilitan las compuertas AND 3 y 4 (observe el inversor). Esto permite que las salidas *A* y *B* pasen a través de las compuertas 1 y 2 a las entradas *J* y *K* de los FFs *B* y *C*. Cuando Arriba/Abajo se mantiene en BAJO, las compuertas AND 1 y 2 se deshabilitan mientras que las compuertas AND 3 y 4 se habilitan. Esto permite que las salidas *A* y *B* invertidas pasen a través de las compuertas 3 y 4 a las entradas *J* y *K* de los FFs *B* y *C*. Las formas de onda en la figura 7-11(b) ilustran la operación. Observe que para los primeros cinco pulsos de reloj, Arriba/Abajo = 1 y el contador cuenta hacia arriba; para los últimos cinco pulsos Arriba/Abajo = 0 y el contador cuenta hacia abajo.



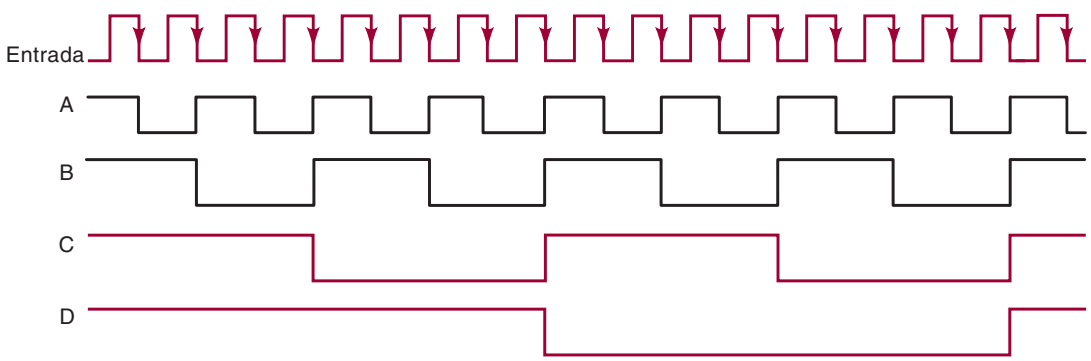


FIGURA 7-10 El contador descendente MOD-16 síncrono y las formas de onda de salida.

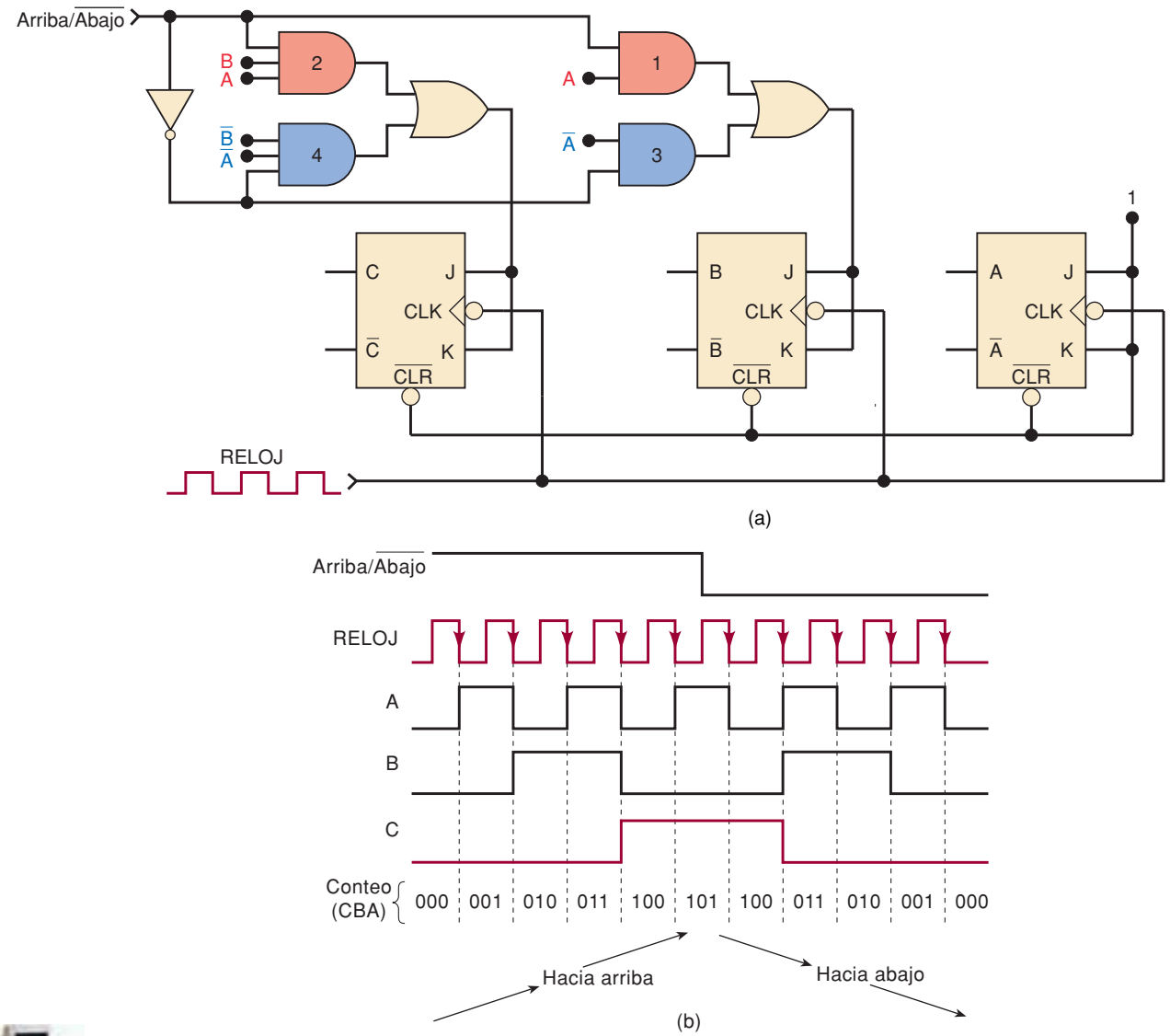


FIGURA 7-11 (a) Contador ascendente/descendente MOD-8 síncrono. (b) El contador cuenta hacia arriba cuando la entrada de control Arriba/Abajo = 1; cuenta hacia abajo cuando la entrada de control Arriba/Abajo = 0.

La nomenclatura utilizada para la señal de control (Arriba/Abajo) se eligió de manera que nos haga ver claro la forma en que afecta al contador. La operación de conteo ascendente es activa en ALTO; la operación de conteo descendente es activa en BAJO.

EJEMPLO 7-9

¿Qué problemas podrían ocasionarse si la señal Arriba/Abajo cambiara de niveles en la NGT del reloj?

Solución

Los FFs podrían operar en forma impredecible, ya que en algunos de ellos sus entradas *J* y *K* estarían cambiando casi al mismo tiempo que se produjera una NGT en su entrada *CLK*. No obstante, los efectos del cambio en la señal de control deben propagarse a través de dos compuertas antes de llegar a las entradas *J*, *K*, por lo que es más probable que los FFs respondan en forma predecible a los niveles que hay en *J*, *K* antes de la NGT de *CLK*.

PREGUNTAS DE REPASO

1. ¿Cuál es la diferencia entre la secuencia de conteo de un contador ascendente y de un contador descendente?
2. ¿Qué cambios en el circuito convertirán a un contador binario síncrono ascendente en un contador binario descendente?

7-6 CONTADORES PREAJUSTABLES

Muchos contadores síncronos (en paralelo) que están disponibles como CIs están diseñados para ser **preajustables**; en otras palabras, pueden preajustarse a cualquier conteo inicial deseado, ya sea en forma asíncrona (independiente de la señal del reloj) o síncrona (en la transición activa de la señal del reloj). A esta operación de preajuste se le conoce también como **carga en paralelo** del contador.

La figura 7-12 muestra el circuito lógico para un contador ascendente en paralelo preajustable de 3 bits. Las entradas *J*, *K* y *CLK* se conectan para que opere como un contador ascendente en paralelo. Las entradas asíncronas PRESTABLECER y LIMPIAR se conectan para realizar un ajuste asíncrono. El contador se carga con cualquier conteo deseado en cualquier momento mediante el siguiente procedimiento:

1. Se aplica el conteo deseado a las entradas de datos en paralelo P_2, P_1 y P_0.
2. Se aplica un pulso BAJO a la entrada de CARGA EN PARALELO ($\overline{PL}$).

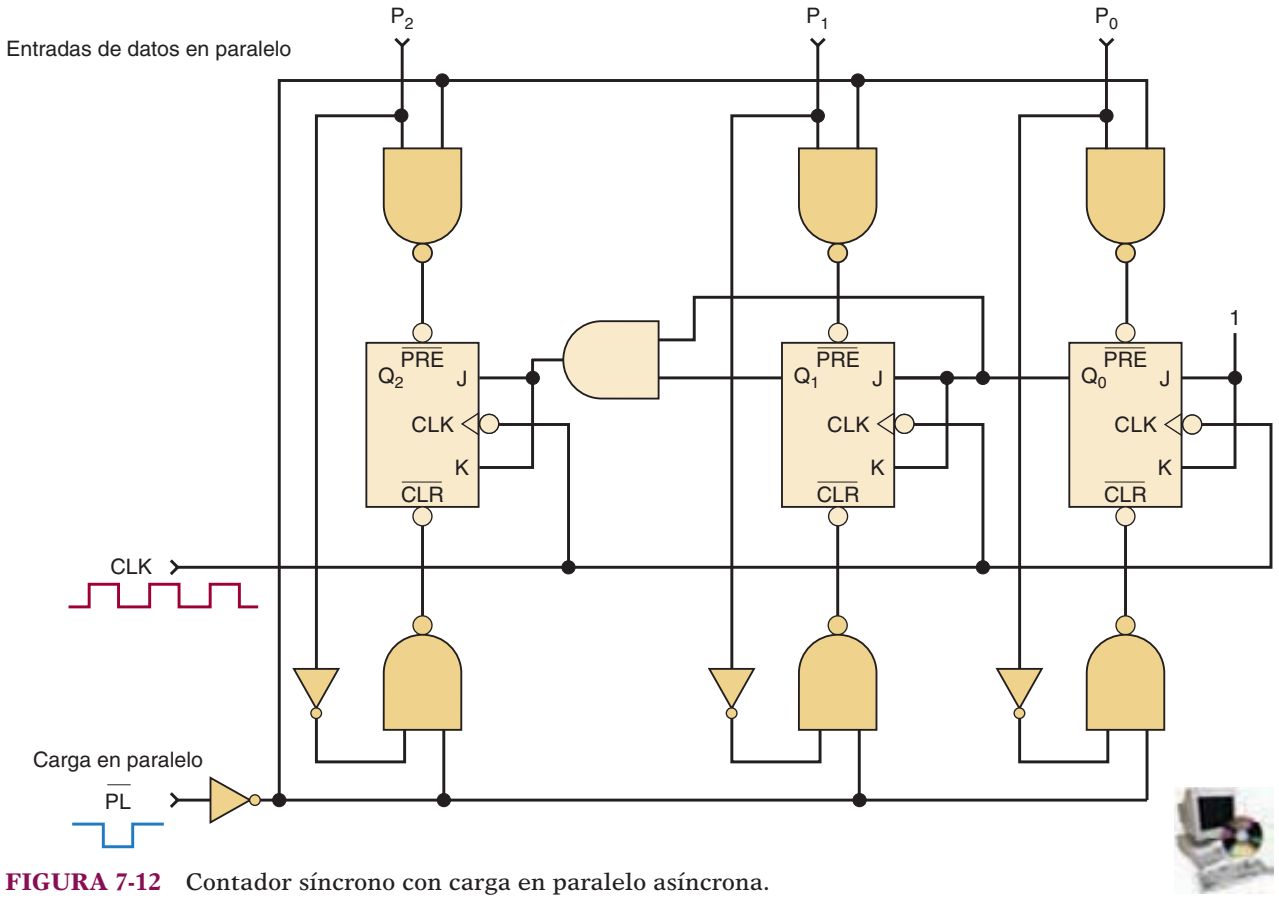


FIGURA 7-12 Contador síncrono con carga en paralelo asíncrona.

Este procedimiento realizará una transferencia asíncrona de los niveles P_2, P_1 y P_0 hacia los flip-flops Q_2, Q_1 y Q_0, respectivamente (sección 5-17). Esta *transferencia forzada* ocurre de manera independiente de las entradas *J*, *K* y *CLK*. El efecto de la entrada *CLK* quedará deshabilitado mientras $\overline{PL}$ se encuentre en su estado activo en BAJO, ya que cada FF tendrá activada una de sus entradas asíncronas mientras $\overline{PL} = 0$. Una vez que $\overline{PL}$ regrese a nivel ALTO, los FFs podrán responder con sus entradas *CLK* y resumir la operación de conteo ascendente, empezando en la cuenta que se cargó en el contador.

Por ejemplo, digamos que $P_2 = 1$, $P_1 = 0$ y $P_0 = 1$. Mientras $\overline{PL}$ esté en ALTO, estas entradas de datos en paralelo no tendrán efecto. Si hay pulsos de reloj presentes, el contador realizará la operación normal de conteo ascendente. Ahora digamos que $\overline{PL}$ cambia a BAJO cuando el contador está en el conteo 010 (es decir, $Q_2 = 0$, $Q_1 = 1$ y $Q_0 = 0$). Este nivel BAJO en $\overline{PL}$ producirá niveles BAJO en la entrada *CLR* de Q_1 y en las entradas *PRE* de Q_2 y Q_0, de manera que el contador cambiará al conteo 101 *sin importar lo que esté ocurriendo en la entrada CLK*. El conteo se mantendrá en 101 hasta que $\overline{PL}$ se desactive (regrese a ALTO); en ese momento el contador seguirá contando hacia arriba con cada pulso de reloj, empezando desde el conteo de 101.

Este preajuste asíncrono lo utilizan varios contadores de CI, tales como los chips TTL 74ALS190, 74ALS191, 74ALS192 y 74ALS193, y sus equivalentes en CMOS, 74HC190, 74HC191, 74HC192 y 74HC193.

Preajuste síncrono

Muchos contadores en paralelo de CI utilizan el *preajuste síncrono*, en el cual el contador se prestablece en la transición activa de la misma señal de reloj que se utiliza para el conteo. El nivel lógico en la entrada de control de carga en paralelo determina si el contador se va a prestablecer con los datos de entrada aplicados en la siguiente transición activa del reloj.

Algunos ejemplos de contadores de CI que utilizan el preajuste síncrono son: De la familia TTL 74ALS160, 74ALS161, 74ALS162 y 74ALS163, y sus equivalentes en la familia CMOS 74HC160, 74HC161, 74HC162 y 74HC163.

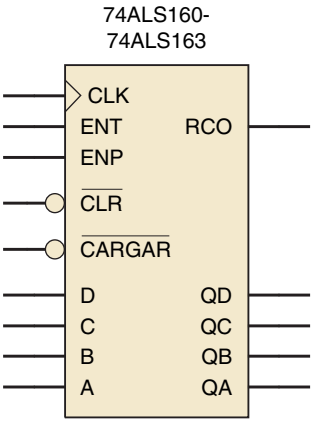
PREGUNTAS DE REPASO

1. ¿Qué queremos decir cuando expresamos que un contador es preajustable?
2. Describa la diferencia entre preajuste asíncrono y síncrono.

7-7 CONTADORES SÍNCRONOS DE CI

La serie 74ALS160-163/74HC160-163

La figura 7-13 muestra el símbolo lógico, el módulo y la tabla de funciones para la serie 74ALS160 a 74ALS163 de contadores de CI (y sus contrapartes equivalentes en CMOS, 74HC160 a 74HC163). Estos contadores reciclables de cuatro bits tienen sus salidas etiquetadas como QD, QC, QB, QA, en donde QA es el LSB y QD es el MSB. Están sincronizados mediante una señal PGT que se aplica a CLK. Cada uno de estos circuitos integrados tiene una combinación distinta de dos características diferentes. Como puede ver en la figura 7-13(b), dos de los contadores son MOD-10 (74ALS160 y 74ALS162), mientras que los otros dos son contadores binarios MOD-16 (74ALS161 y 74ALS163). La otra variación para estas piezas está en la operación de la función de limpiar [como se resalta en la figura 7-13(c)]. El 74ALS160 y el 74ALS161 tienen, cada uno, una entrada para limpiar asíncrona. Esto significa que tan pronto como $\overline{CLR}$ cambie a BAJO ($\overline{CLR}$ es activa en BAJO en los circuitos mencionados), la salida del contador se restablecerá a 0000. Por otro lado, los contadores de CI 74ALS162 y 74ALS163 se borran en forma síncrona. Para ello la entrada $\overline{CLR}$ debe estar en BAJO y debe aplicarse una señal PGT a la entrada de reloj. La entrada limpiar tiene prioridad sobre todas las demás funciones en esta serie de contadores de CI. Limpiar predominará sobre todas las demás entradas de control, como se indica mediante las X en la tabla de funciones de la figura 7-13(c).



(a)

Número de pieza	Módulo
74ALS160	10
74ALS161	16
74ALS162	10
74ALS163	16

(b)

FIGURA 7-13
Contadores síncronos de la serie 74ALS160-74ALS163: (a) símbolo lógico, (b) módulos; (c) tabla de funciones.

Tabla de funciones de los CIs 74ALS160-74ALS163

$\overline{CLR}$	$\overline{CARGAR}$	ENP	ENT	CLK	Función	Números de parte
L	X	X	X	X	Limpiar asíncrona	74ALS160 & 74ALS161
L	X	X	X	↑	Limpiar síncrona	74ALS162 & 74ALS163
H	L	X	X	↑	Cargar síncrona	Todos
H	H	H	H	↑	Conteo ascendente	Todos
H	H	L	X	X	Sin cambios	Todos
H	H	X	L	X	Sin cambios	Todos

(c)

La función de segunda prioridad disponible en esta serie de contadores de CI es la carga en paralelo de datos en los flip-flops del contador. Para prestablecer un valor de datos, la entrada limpiar se pone inactiva (ALTO), se aplica el valor de cuatro bits deseado a las terminales de entrada de datos D, C, B, A, (A es el LSB y D es el MSB), se aplica un nivel BAJO a la entrada $\overline{CARGAR}$ y después se pone la señal de reloj al circuito por medio de una PGT. Por lo tanto, la función de carga es síncrona y tiene prioridad sobre el conteo, por lo que no importa qué niveles lógicos se apliquen a ENT o ENP. Para contar a partir del estado preestablecido será necesario deshabilitar la carga (con un nivel ALTO) y habilitar la función de conteo. Si la función de carga está inactiva, no importa lo que se aplique a las terminales de entrada de datos.

Para habilitar el conteo, la función de menor prioridad, las entradas de control $\overline{CLR}$ y $\overline{CARGAR}$ deben estar inactivas. Además hay dos controles de habilitación de conteo activos en ALTO, ENT y ENP. En esencia, se aplica una operación AND a ENT y ENP para controlar la función de conteo. Si cualquiera o ambos controles de **habilitación de conteo** están inactivos (BAJO), el contador mantendrá el estado actual. Por lo tanto, para incrementar el conteo en cada PGT de CLK las cuatro entradas de control deben estar en ALTO. Al contar, los contadores de décadas (74ALS160 y 74ALS162) se reciclarán de manera automática a 000 después del estado 1001 (9) y los contadores binarios (74ALS161 y 74ALS163) se reciclarán de manera automática después de 1111 (15).

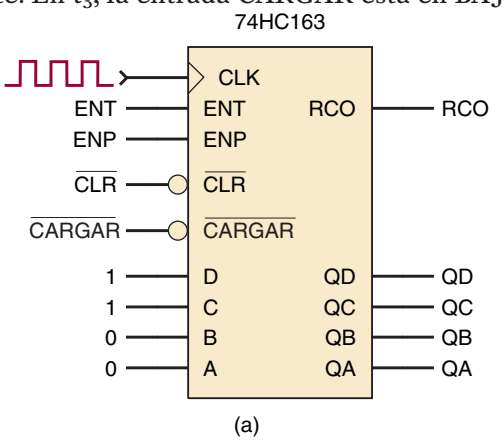
Esta serie de contadores de CI tiene una terminal más de salida: RCO. La función de esta salida activa en ALTO es detectar (*decodificar*) el último estado (o estado terminal) del contador. El estado terminal para un contador de décadas es 1001 (9), mientras que el estado terminal para un contador MOD-16 es 1111 (15). La entrada de habilitación de conteo primaria ENT también controla la operación de RCO. ENT debe estar en ALTO para que el contador indique con la salida RCO que ha llegado a su estado terminal. Esta característica es muy útil para conectar dos o más circuitos integrados tipo contador entre sí en un arreglo de múltiples etapas para crear contadores más grandes.

EJEMPLO 7-10

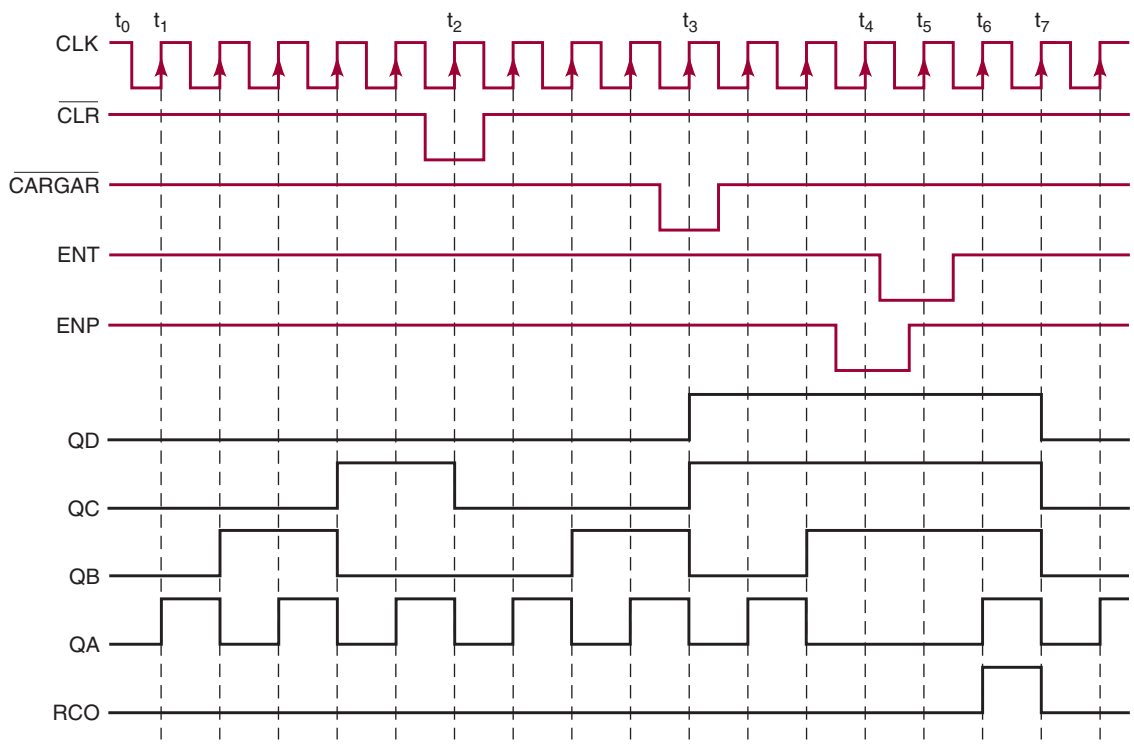
Consulte la figura 7-14, en donde un 74HC163 tiene las señales de entrada que se proporcionan en el diagrama de sincronización que se aplica. Las entradas de datos en paralelo están conectadas en forma permanente como 1100. Suponga que al principio el contador se encuentra en el estado 0000 y determine las formas de onda de salida de éste.

Solución

Al principio (en t_0) todos los FFs del contador están en BAJO. Como éste no es el estado terminal para el contador, la salida RCO también estará en BAJO. La primera PGT en la entrada $\overline{\text{CLK}}$ ocurre en t_1 y, como todas las entradas de control están en ALTO, el contador se incrementará a 0001. El contador seguirá contando en forma ascendente con cada PGT hasta t_2. En este tiempo, la entrada $\overline{\text{CLR}}$ está en BAJO. Debido a esto, el contador se restablecerá en forma síncrona al conteo 0000 en t_2. Después de t_2, la entrada CLR se vuelve inactiva (ALTO), por lo que el contador empezará a contar de nuevo hacia arriba, partiendo desde 0000, con cada PGT subsiguiente. En t_3, la entrada $\overline{\text{CARGAR}}$ está en BAJO. Debido a esto, en t_3 se



(a)



(b)

FIGURA 7-14
Ejemplo 7-10.

cargará en forma síncrona el valor de datos 1100 (12) aplicado al contador. Después de t_3 la entrada $\overline{\text{CARGAR}}$ se vuelve inactiva (ALTO), por lo que el contador seguirá contando hacia arriba, partiendo desde 1100 con cada PGT subsiguiente hasta t_4. La salida del contador no cambia en t_4 ni en t_5, ya que ENP o ENT (las entradas de habilitación de conteo) están en BAJO. Esto mantiene el conteo en 1110 (14). En t_6 el contador se habilita de nuevo y cuenta hasta 1111 (15), su estado terminal. Como resultado, ahora la salida RCO cambia a ALTO. En t_7 otra PGT en CLK hará que el contador se recicle a 0000 y RCO regresa al nivel BAJO.

EJEMPLO 7-11

Consulte la figura 7-15, ahí se muestra un 74HC160 al que se le aplican las señales de entrada indicadas en el diagrama de tiempo. Las entradas de datos en paralelo se conectan de manera permanente como 0111. Suponga que al principio el conta-

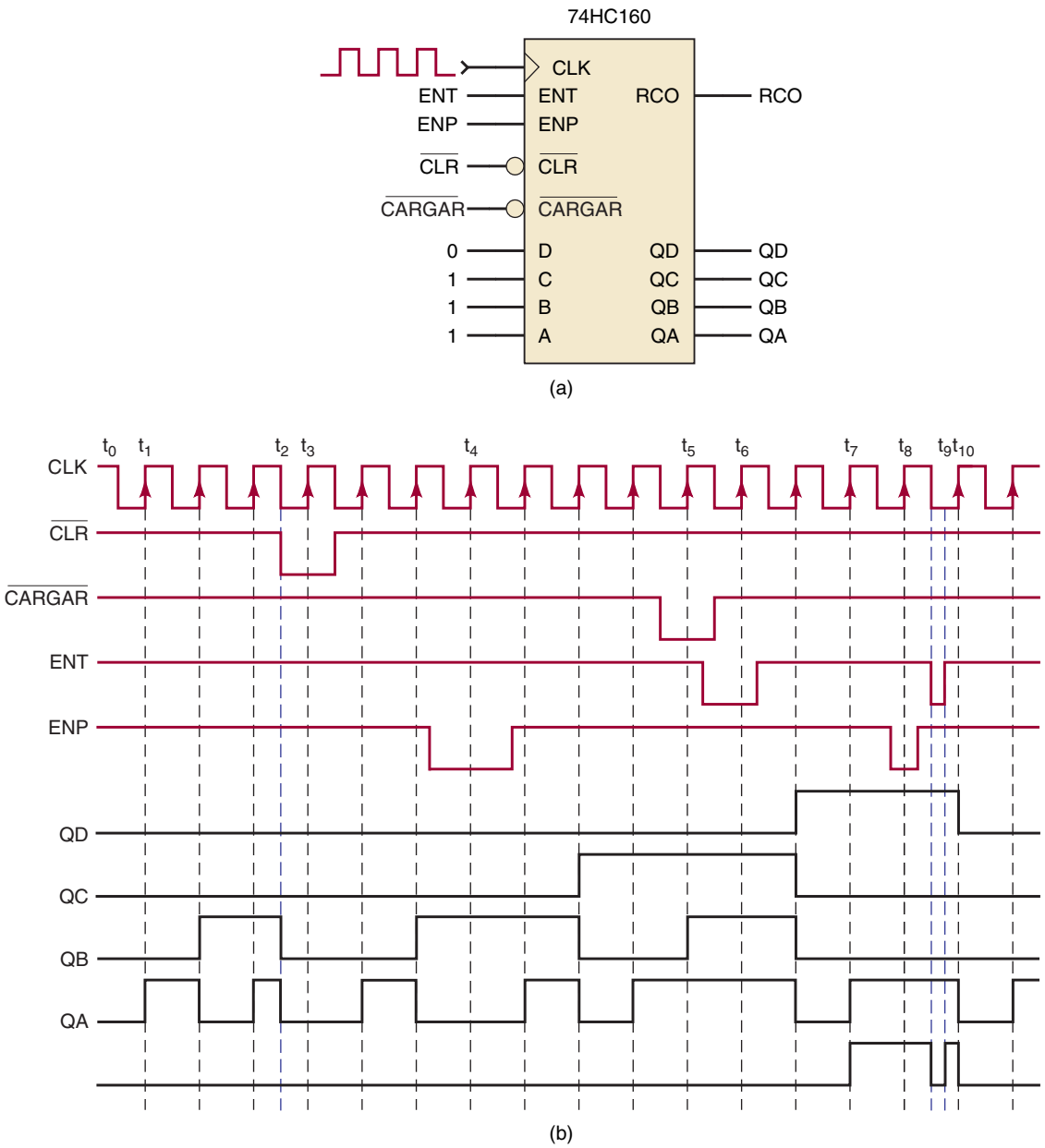


FIGURA 7-15 Ejemplo 7-11.

dor se encuentra en el estado 0000 y determine las formas de onda de salida del contador.

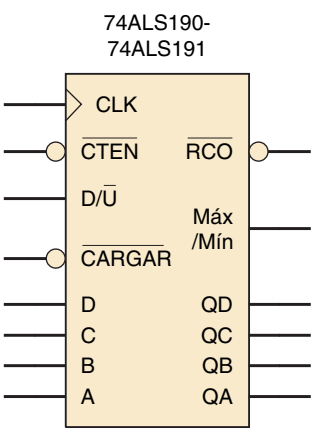
Solución

Al principio (en t_0) los FFs del contador están todos en BAJO. Como éste no es el estado terminal para el contador BCD, la salida RCO también estará en BAJO. La primera PGT en la entrada CLK ocurre en t_1, y como todas las entradas de control están en ALTO, el contador se incrementará a 0001. El contador seguirá contando en forma ascendente con cada PGT hasta t_2. La entrada $\overline{\text{CLR}}$ asíncrona cambia a BAJO en t_2, con lo cual el contador se restablecerá de inmediato a 0000 en ese punto. En t_3 la entrada $\overline{\text{CLR}}$ sigue activa (BAJO), por lo que se ignorará la PGT de la entrada CLK y el contador permanecerá en 0000. Más adelante la entrada $\overline{\text{CLR}}$ se vuelve inactiva de nuevo y el contador seguirá contando hasta 0001 y después hasta 0010. En t_4 la señal de habilitación de conteo ENP está en BAJO, por lo que el conteo se mantiene en 0010. En las PGTs subsiguientes de la entrada CLK, el contador se habilita y cuenta hacia arriba hasta t_5. La entrada $\overline{\text{CARGAR}}$ está en bajo para t_5. Esto cargará en forma síncrona el valor de datos aplicado 0111 (7) en el contador, en t_5. En t_6 la señal de habilitación de conteo ENT está en BAJO, por lo que el conteo se mantiene en 0111. Para las dos PGTs subsiguientes después de t_6, el contador seguirá contando hacia arriba ya que está rehabilitado. En t_7, el contador BCD llega a su estado terminal 1001 (9) y la salida RCO cambia a ALTO. En t_8, ENP está en BAJO y el contador se detiene (se queda en 1001). En t_9, mientras ENT esté en BAJO la salida RCO se deshabilitará, de forma que regrese a BAJO aún y cuando el contador siga en su estado terminal (1001). Recuerde que sólo ENT controla la salida RCO. Cuando ENT regresa a ALTO durante el estado terminal del contador, RCO cambia a ALTO de nuevo. En t_{10} el contador se habilita y se recicla a 0000, después cuenta hasta 0001 en la última PGT.

La serie 74ALS190-191/74HC190-191

La figura 7-16 muestra el símbolo lógico, el módulo y la tabla de funciones para las series de circuitos integrados contadores 74ALS190 y 74ALS191 (y sus equivalentes en CMOS, 74HC190 y 74HC191). Estos contadores reciclables de cuatro bits tienen salidas etiquetadas como QD, QC, QB, QA, en donde QA es el LSB y QD es el MSB. Se sincronizan mediante una señal PGT que se aplica en CLK. La única diferencia entre dichos circuitos es el módulo del contador. El 74ALS190 es un contador MOD-10 y el 74ALS191 es un contador binario MOD-16. Ambos circuitos son contadores ascendentes/descendentes y tienen una entrada de carga asíncrona, activa en BAJO. Esto significa que, tan pronto como $\overline{\text{CARGAR}}$ cambie a BAJO, el contador

FIGURA 7-16
Contadores síncronos de la serie 74ALS190-74ALS191: (a) símbolo lógico; (b) módulo; (c) tabla de funciones.



(a)

Número de parte	Módulo
74ALS190	10
74ALS191	16

(b)

Tabla de funciones de los circuitos 74ALS190-74ALS191

$\overline{\text{CARGAR}}$	$\overline{\text{CTEN}}$	D/$\overline{\text{U}}$	CLK	Función
L	X	X	X	Carga asíncrona
H	L	L	↑	Conteo ascendente
H	L	H	↑	Conteo descendente
H	H	X	X	Sin cambio

(c)

se prestablecerá (en paralelo) con los datos disponibles en las terminales de entrada D, C, B, A (A es el LSB y D es el MSB). Si la función de carga está inactiva, no importa lo que se aplique a las terminales de datos de entrada. La entrada de carga tiene prioridad sobre la función de conteo.

Para contar, la entrada de control $\overline{\text{CARGAR}}$ debe estar inactiva (ALTO) y la entrada de control de habilitación de conteo ($\overline{\text{CTEN}}$) debe estar en BAJO. La dirección del conteo se controla mediante la entrada de control D/$\overline{\text{U}}$. Si D/$\overline{\text{U}}$ está en BAJO, el conteo se incrementa con cada PGT de CLK, mientras que un nivel ALTO en D/$\overline{\text{U}}$ disminuirá el conteo. Ambos contadores se reciclan de manera automática en cualquier dirección de conteo. El contador de décadas se recicla a 0000 después del estado 1001 (9) cuando cuenta hacia arriba, o a 1001 después del estado 0000 cuando cuenta hacia abajo. El contador binario se recicla a 0000 después de 1111 (15) cuando cuenta hacia arriba, o a 1111 después del estado 0000 cuando cuenta hacia abajo.

Estos circuitos tipo contador tienen dos terminales más de salida, MÁX/MÍN y RCO. La primera es una salida activa en ALTO que detecta (decodifica) el estado terminal del contador. Como son contadores ascendentes/descendentes, el estado terminal depende de la dirección del conteo. El estado terminal (MÍN) para cualquiera de los contadores cuando el conteo es descendente es 0000 (0). No obstante, cuando el conteo es ascendente el estado terminal (MÁX) para un contador de décadas es 1001 (9), mientras que el estado terminal para un contador MOD-16 es 1111 (15). Observe que MÁX/MÍN sólo detecta un estado en la secuencia de conteo; sólo depende de si está contando hacia arriba o hacia abajo. La salida $\overline{\text{RCO}}$ activa en BAJO también detecta el estado terminal apropiado para el contador, pero es más complicada. En primer lugar, sólo se habilita cuando $\overline{\text{CTEN}}$ está en BAJO. Además, $\overline{\text{RCO}}$ sólo estará en BAJO mientras que la entrada CLK se encuentre también en BAJO. Así que, en esencia, $\overline{\text{RCO}}$ imitará la forma de onda de CLK sólo durante el estado terminal mientras el contador esté habilitado.

EJEMPLO 7-12

Consulte la figura 7-17, en donde se muestra un circuito 74HC190 al que se le aplican las señales de entrada mostradas en el diagrama de tiempo. Las entradas de datos en paralelo se conectan de manera permanente como 0111. Suponga que al principio el contador se encuentra en el estado 0000 y determine las formas de onda de salida del contador.

Solución

Al principio (en t_0) los FFs del contador están todos en BAJO. Como el contador está habilitado ($\overline{\text{CTEN}} = 0$) y la entrada de control de dirección de conteo D/$\overline{\text{U}} = 0$, el contador BCD empezará a contar en forma ascendente en la primera señal PGT que se aplique a CLK en t_1, y continuará contando hacia arriba con cada PGT hasta t_2, en donde el conteo ha llegado a 0101. La entrada $\overline{\text{CARGAR}}$ asíncrona cambia a BAJO en t_2 y cargará de inmediato el valor 0111 en el contador en ese punto. En t_3 la entrada $\overline{\text{CARGAR}}$ sigue activa (BAJO), por lo que se ignorará la señal PGT en la entrada CLK y el contador permanecerá en 0111. Más adelante la entrada $\overline{\text{CARGAR}}$ cambia a ALTO de nuevo y el contador avanzará el conteo ascendente hasta 1000 en la siguiente PGT. En t_4 el contador se incrementa a 1001, que es el estado terminal para un contador BCD ascendente y en consecuencia la salida MÁX/MÍN cambiará a ALTO. Durante t_5 el contador se encuentra en su estado terminal y la entrada CLK está en BAJO, por lo que $\overline{\text{RCO}}$ cambia a BAJO. Para las PGTs subsiguientes de la entrada CLK, el contador se recicla a 0000 y continúa contando hacia arriba hasta t_6. Justo antes de t_6, el control D/$\overline{\text{U}}$ cambia al nivel ALTO. Esto hará que el contador cuente hacia abajo en t_6 y de nuevo en t_7, en donde quedará en el estado 0000, que ahora es el estado terminal ya que está contando hacia abajo, y MÁX/MÍN tendrá un nivel ALTO. Durante t_8, cuando la entrada CLK cambia a BAJO, la salida $\overline{\text{RCO}}$ estará de nuevo en BAJO. En t_9 el contador se deshabilita con $\overline{\text{CTEN}} = 1$ y mantendrá el conteo en 1001. Para los siguientes pulsos en CLK, el contador seguirá contando hacia abajo.

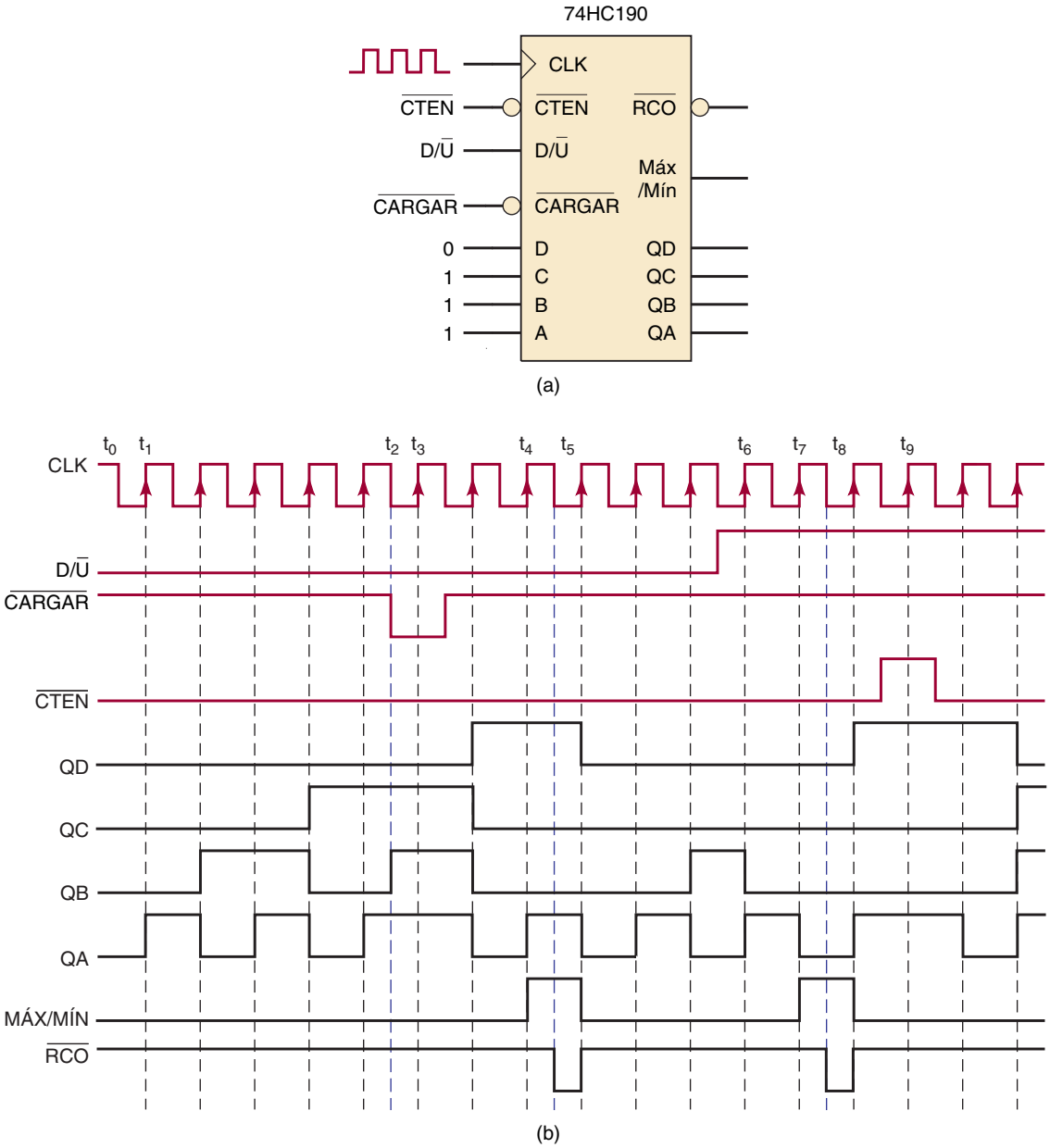
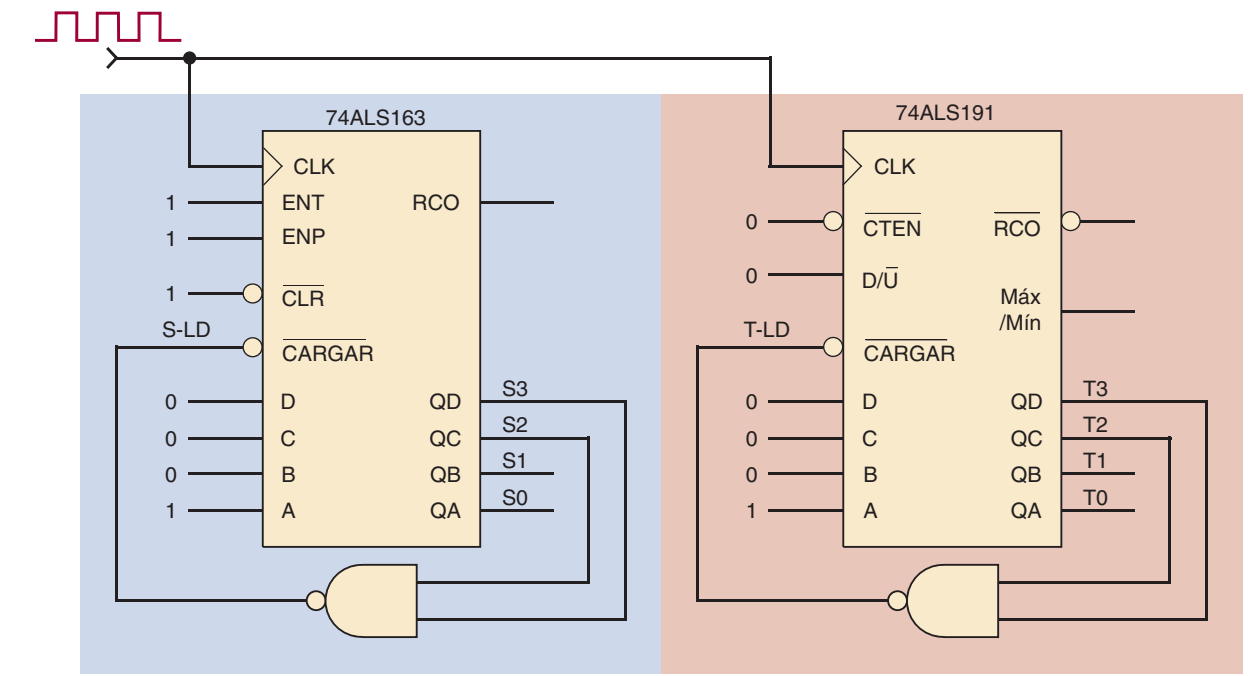


FIGURA 7-17 Ejemplo 7-12.

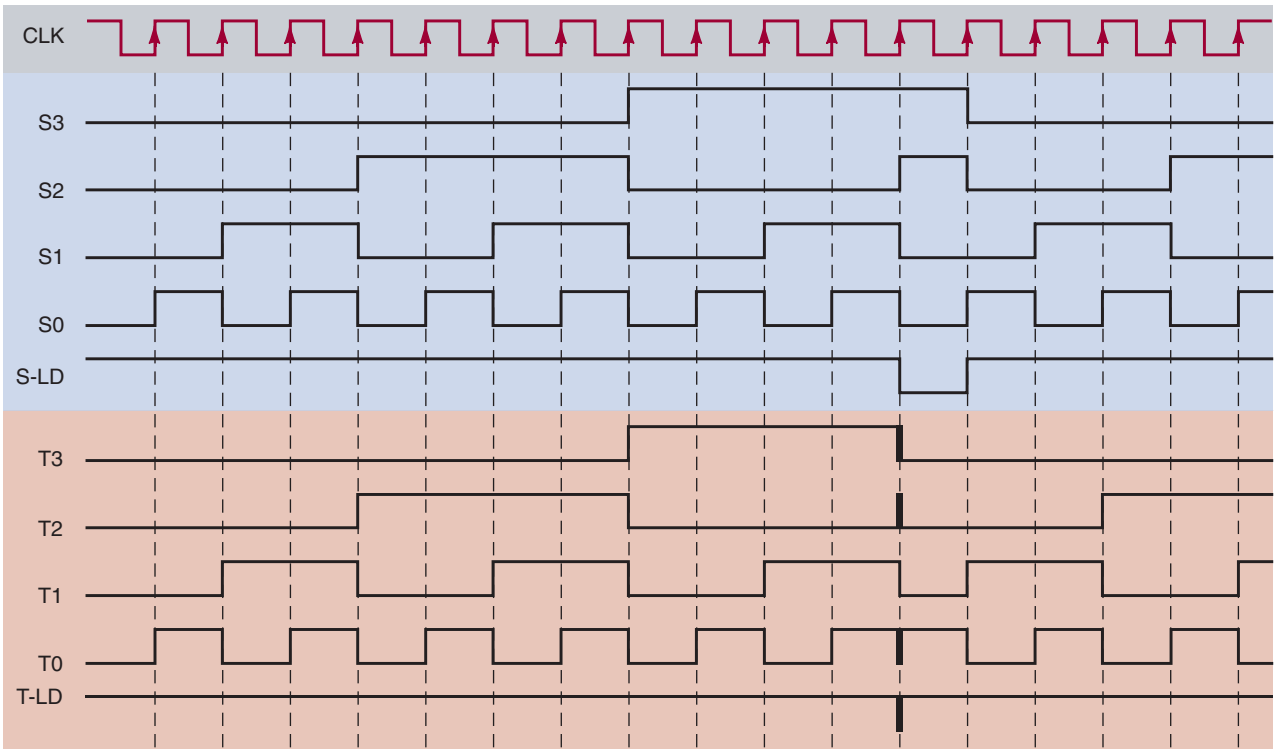
EJEMPLO 7-13

Compare la operación de dos contadores: uno con carga síncrona y el otro con carga asíncrona. Consulte la figura 7-18(a), en donde un 74ALS163 y un 74ALS191 se han conectado en forma similar para contar hacia arriba en binario. Ambos circuitos se controlan mediante la misma señal de reloj y sus salidas QD y QC están conectadas a una compuerta NAND para controlar la entrada de control CARGAR respectiva. Suponga que al principio ambos contadores se encuentran en el estado 0000.

(a) Determine la forma de onda de salida para cada contador.
(b) ¿Cuál es la secuencia de conteo de reciclaje y el módulo para cada contador?
(c) ¿Por qué tienen distintas secuencias de conteo?



(a)



(b)

FIGURA 7-18 Ejemplo 7-13.

Solución

(a) Comenzando en el estado 0000, cada contador contará hacia arriba hasta llegar al estado 1100 (12), como muestra la figura 7-18(b). La salida de cada compuerta NAND aplicará un nivel BAJO a la entrada $\overline{\text{CARGAR}}$ respectiva en ese tiempo. El 74ALS163 tiene una entrada $\overline{\text{CARGAR}}$ síncrona, por lo que esperará

hasta la siguiente PGT en CLK para cargar los datos de entrada 0001 en el contador. El 74ALS191 tiene una entrada $\overline{\text{CARGAR}}$ asíncrona, por lo que cargará de inmediato los datos de entrada 0001 en el contador. Esto convertirá al estado 1100 en temporal o transiente para el 74ALS191. El estado transiente producirá algunos picos o deformaciones en algunas de las salidas del contador, debido a la rápida acción de alternar hacia atrás y hacia delante.

(b) El circuito 74ALS163 tiene una secuencia de conteo de reciclaje de 0001 hasta 1100 y es un contador MOD-12. El circuito 74ALS191 tiene una secuencia de conteo de reciclaje de 0001 hasta 1011 y es un contador MOD-11. No se incluyen los estados transientes al determinar el módulo para un contador.

(c) Los circuitos tipo contador tienen distintas secuencias de conteo debido a que uno de ellos tiene una carga síncrona y el otro tiene una carga asíncrona.

Arreglo de múltiples etapas

Muchos contadores de CI estándar se han diseñado para facilitar la conexión de varios circuitos integrados en conjunto para crear circuitos con un rango de conteo más alto. Todos los circuitos tipo contador que se presentan en esta sección pueden conectarse en un arreglo de **múltiples etapas** o **en cascada**. En la figura 7-19, dos circuitos 74ALS163 se conectan en un arreglo de contadores de dos etapas que produce una secuencia binaria con reciclaje de 0 a 255, para un módulo máximo de 256. Al aplicar un nivel BAJO en la entrada $\overline{\text{CLR}}$ se borrarán en forma síncrona ambas etapas de contador, y al aplicar un nivel BAJO a $\overline{\text{LD}}$ el contador de ocho bits se prestablecerá en forma síncrona al valor binario en las entradas D7, D6, D5, D4, D3, D2, D1, D0 (D0 = LSB). El bloque a la izquierda (etapa 1) es la etapa de menor orden y proporciona las salidas menos significativas del contador: Q3, Q2, Q1, Q0 (en donde Q0 = LSB). La etapa 2 a la derecha proporciona las salidas más significativas del contador: Q7, Q6, Q5, Q4 (en donde Q7 = MSB).

EN, la entrada de habilitación para el contador de ocho bits, se conecta a la entrada ENT en la etapa 1. Observe que debemos usar la entrada ENT y no ENP, ya que sólo ENT controla la salida RCO. El uso de ENT y RCO facilita en forma considerable la conexión en cascada. Ambos bloques del contador se coordinan en forma síncrona, pero el bloque a la derecha (etapa 2) está deshabilitado hasta que el nibble de salida menos significativo haya llegado a su estado terminal, el cual se indica mediante la salida TC1. Cuando Q3, Q2, Q1, Q0 llegan a 1111 y si EN está en ALTO, entonces TC1 producirá un nivel ALTO. Esto permitirá que ambas etapas del contador cuenten uno hacia arriba con la siguiente PGT en el reloj. La etapa 1 se

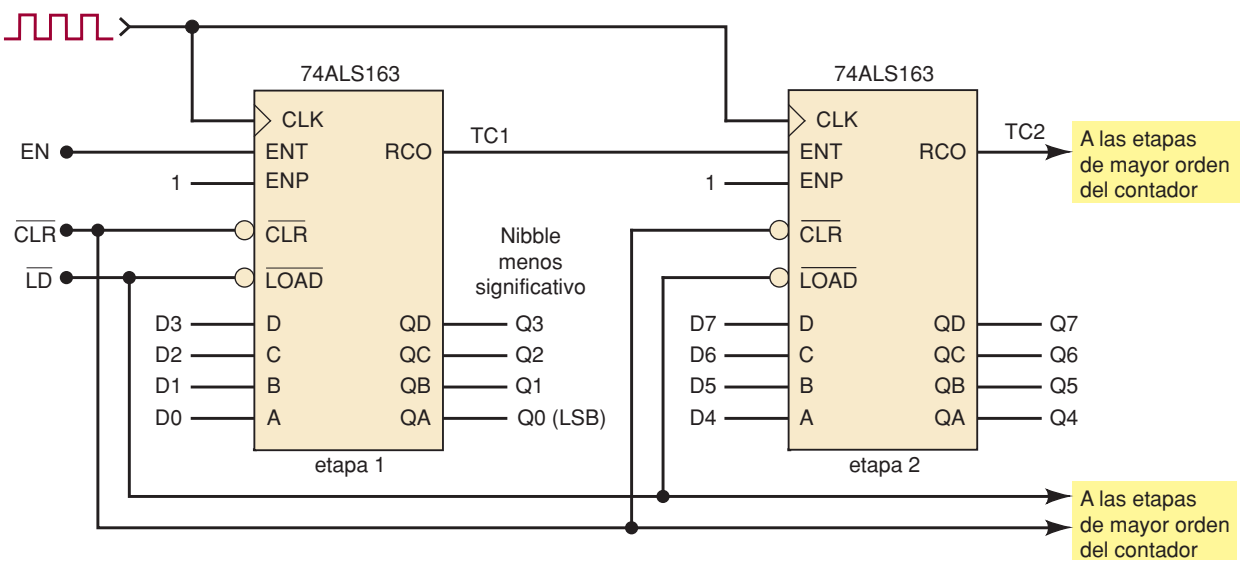


FIGURA 7-19 Dos chips 74ALS163 conectados en un arreglo de dos etapas, para extender el rango máximo de conteo.

reciclará de vuelta a 0000 y la etapa 2 se incrementará a partir de su estado de salida anterior. TC1 regresará a un nivel BAJO ya que la etapa 1 no se encuentra más en su estado terminal. Con los subsiguientes ciclos de reloj, la etapa 1 continuará contando hacia arriba si EN = 1, hasta que llegue de nuevo a 1111 y se repite el proceso. Cuando el contador de ocho bits llega a 11111111, se reciclará de vuelta a 00000000 en el siguiente pulso de reloj.

De la misma forma, pueden conectarse en cascada circuitos contador 74ALS163 adicionales. TC2 se conectaría al control ENT en el siguiente circuito, y así en lo sucesivo. TC2 estará en ALTO cuando Q7, Q5, Q4 sea igual a 1111 y TC1 esté en ALTO, lo cual a su vez significa que Q3, Q2, Q1, Q0 son igual a 1111 y EN está en ALTO. Esta técnica en cascada funciona para todos los circuitos (familia TTL o CMOS) en esta serie, incluso para los contadores BCD. La serie 74ALS190-191 (o 74HC190-191) también puede conectarse en cascada de manera similar si se utilizan las terminales $\overline{\text{CTEN}}$ y $\overline{\text{RCO}}$ activas en BAJO. Un contador de múltiples etapas que utilice circuitos 74ALS190-191 conectado de esta forma podrá contar hacia arriba o hacia abajo.

PREGUNTAS DE REPASO

1. Describa la función de las entradas $\overline{\text{CARGAR}}$ y D, C, B, A.
2. Describa la función de la entrada $\overline{\text{CLR}}$.
3. *Verdadero o falso:* el 74HC161 no puede prestablecerse mientras $\overline{\text{CLR}}$ está activa.
4. ¿Qué niveles lógicos deben estar presentes en las entradas de control para que el 74ALS162 pueda contar pulsos que aparezcan en CLK?
5. ¿Qué niveles lógicos deben estar presentes en las entradas de control para que el 74HC190 pueda contar hacia abajo con pulsos que aparezcan en CLK?
6. ¿Cuál sería el rango máximo de conteo para un contador de cuatro etapas formado por CIs 74HC163? ¿Cuál es el rango máximo de conteo para CIs 74ALS190?

7-8 DECODIFICACIÓN DE UN CONTADOR

Los contadores digitales se utilizan con frecuencia en aplicaciones en las que el conteo representado por los estados de los FFs debe determinarse o visualizarse de alguna forma. Uno de los medios más simples para visualizar el contenido de un contador implica sólo conectar la salida de cada FF a un pequeño LED indicador [vea la figura 7-7(b)]. De esta manera, los estados de los FFs se representan en forma visual mediante los LEDs (encendido = 1, apagado = 0) el conteo se puede determinar mentalmente, **decodificando** los estados binarios de los LEDs. Por ejemplo, suponga que se utiliza este método para un contador BCD y que los estados de los LEDs son apagado-encendido-encendido-apagado, respectivamente. Esto representaría el valor 0110, el cual decodificaríamos mentalmente como el 6 decimal. Otras combinaciones de los estados de los LEDs representarían los demás posibles conteos.

El método del LED indicador se vuelve inconveniente a medida que aumenta el tamaño (número de bits) del contador, ya que es demasiado difícil decodificar mentalmente los resultados visualizados. Por esta razón es preferible desarrollar un medio para decodificar por medios *electrónicos* el contenido de un contador y visualizar los resultados en un formato que se reconozca de inmediato y no requiera operaciones mentales.

Una razón aún más importante para la decodificación electrónica de un contador se debe a las diversas aplicaciones en las que se utilizan los contadores para controlar la sincronización o la secuencia de las operaciones en forma *automática*, sin intervención humana. Por ejemplo, la operación de cierto sistema tal vez tenga que iniciarse cuando un contador llegue al estado 101100 (conteo de 44_{10}). Puede utilizarse un circuito lógico para decodificar o detectar el momento en el que se presente este conteo específico y después iniciar la operación. En un sistema digi-

tal, muchas operaciones tienen que controlarse de esta forma. Es evidente que no sería conveniente la intervención humana en este proceso, excepto en sistemas en extremo lentos.

Decodificación activa en ALTO

Un contador MOD-X tiene X estados distintos; cada estado es un patrón específico de 0s y 1s que se almacenan en los FFs del contador. Una red decodificadora es un circuito lógico que genera X salidas distintas, cada una detecta (decodifica) la presencia de un estado específico del contador. Las salidas del decodificador pueden diseñarse para producir ya sea un nivel ALTO o BAJO cuando ocurre la detección. Un decodificador activo en ALTO produce salidas en ALTO para indicar la detección. La figura 7-20 muestra la lógica de decodificación activa en ALTO completa

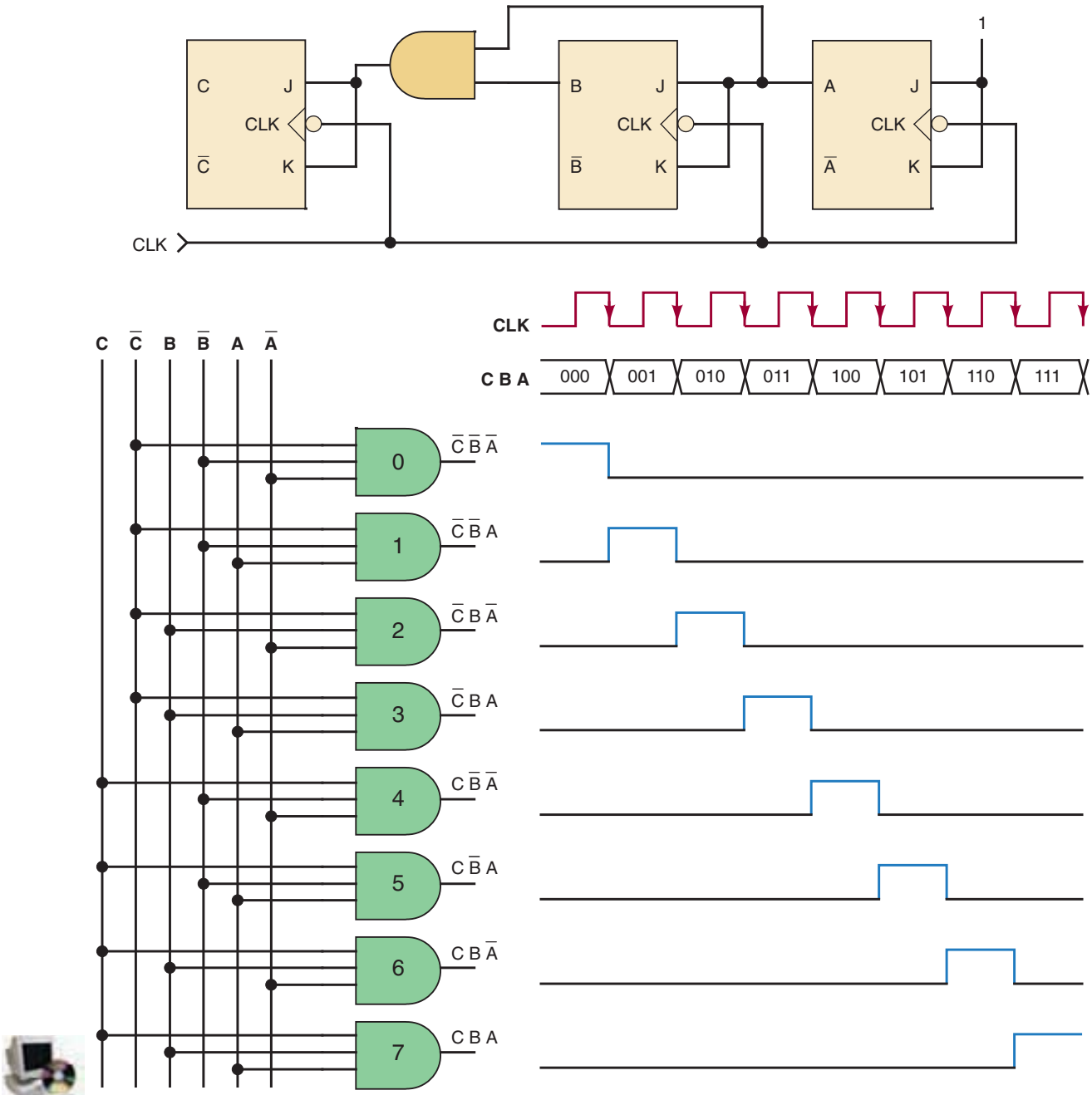


FIGURA 7-20 Uso de compuertas AND para decodificar un contador MOD-8.

para un contador MOD-8. El decodificador consiste de ocho compuertas AND de tres entradas; cada una estas compuertas produce una salida en ALTO para un estado específico del contador.

Por ejemplo, la compuerta AND 0 tiene como entradas las salidas $\overline{C}$, $\overline{B}$ y $\overline{A}$ de los FFs. Por ende, su salida estará en BAJO en todo momento, *excepto* cuando $A = B = C = 0$; esto es, en el conteo de 000 (cero). De manera similar, la compuerta AND 5 tiene como entradas las salidas C, $\overline{B}$ y A de los FFs, por lo que su salida cambiará a ALTO sólo cuando $C = 1$, $B = 0$ y $A = 1$; esto es, en el conteo de 101 (5 decimal). El resto de las compuertas AND funciona de la misma forma para los demás posibles conteos. En cualquier momento dado, sólo la salida de una compuerta AND está en ALTO: la que está decodificando para el conteo específico presente en el contador. Las formas de onda de la figura 7-20 muestran esto con claridad.

Las salidas de las ocho compuertas AND pueden utilizarse para controlar ocho LEDs, los cuales representan los números decimales del 0 al 7. Sólo un LED estará encendido en cualquier momento dado, indicando el conteo apropiado.

El decodificador de compuerta AND puede extenderse para funcionar con contadores con cualquier número de estados. Esto se ilustra mediante el siguiente ejemplo.

EJEMPLO 7-14

¿Cuántas compuertas AND se requieren para decodificar por completo todos los estados de un contador binario MOD-32? ¿Cuáles son las entradas para la compuerta que decodifica el conteo de 21?

Solución

Un contador MOD-32 tiene 32 posibles estados. Se necesita una compuerta AND para decodificar para cada estado; por lo tanto, el decodificador requiere 32 compuertas AND. Como $32 = 2^5$, el contador contiene cinco FFs. Por ende, cada compuerta tendrá cinco entradas, una de cada FF. Para decodificar el conteo de 21 (esto es, 10101_2) se requieren las entradas de E, D, C, B y A en la compuerta AND, en donde E es el flip-flop del MSB.

Decodificación activa en BAJO

Si se utilizan compuertas NAND en vez de compuertas AND, las salidas del decodificador producen una señal que está, por lo general, en nivel ALTO, la cual cambia a BAJO sólo cuando ocurre el número que se está decodificando. Se utilizan ambos tipos de decodificadores, dependiendo del tipo de circuitos que se van a controlar mediante las salidas del decodificador.

EJEMPLO 7-15

La figura 7-21 muestra una situación común en la que se utiliza un contador para generar una forma de onda de control, la cual podría usarse para controlar dispositivos tales como un motor, una válvula solenoide o un calentador. El contador MOD-16 cicla y recicla a través de su secuencia de conteo. Cada vez que llega al conteo de 8 (1000), la compuerta NAND superior produce una salida en BAJO que establece el flip-flop X para que quede en el estado 1. El flip-flop X permanece en ALTO hasta que el contador llega al conteo de 14 (1110), tiempo en el cual la compuerta NAND inferior la decodifica y produce una salida en BAJO para borrar X y que quede en el estado 0. En consecuencia, la salida X está en ALTO entre los conteos de 8 y 14 para cada ciclo del contador.

Decodificación de contadores BCD

Un contador BCD tiene 10 estados que pueden decodificarse mediante el uso de las técnicas antes descritas. Los decodificadores BCD proporcionan 10 salidas que corresponden a los dígitos decimales del 0 al 9 y se representan mediante los esta-

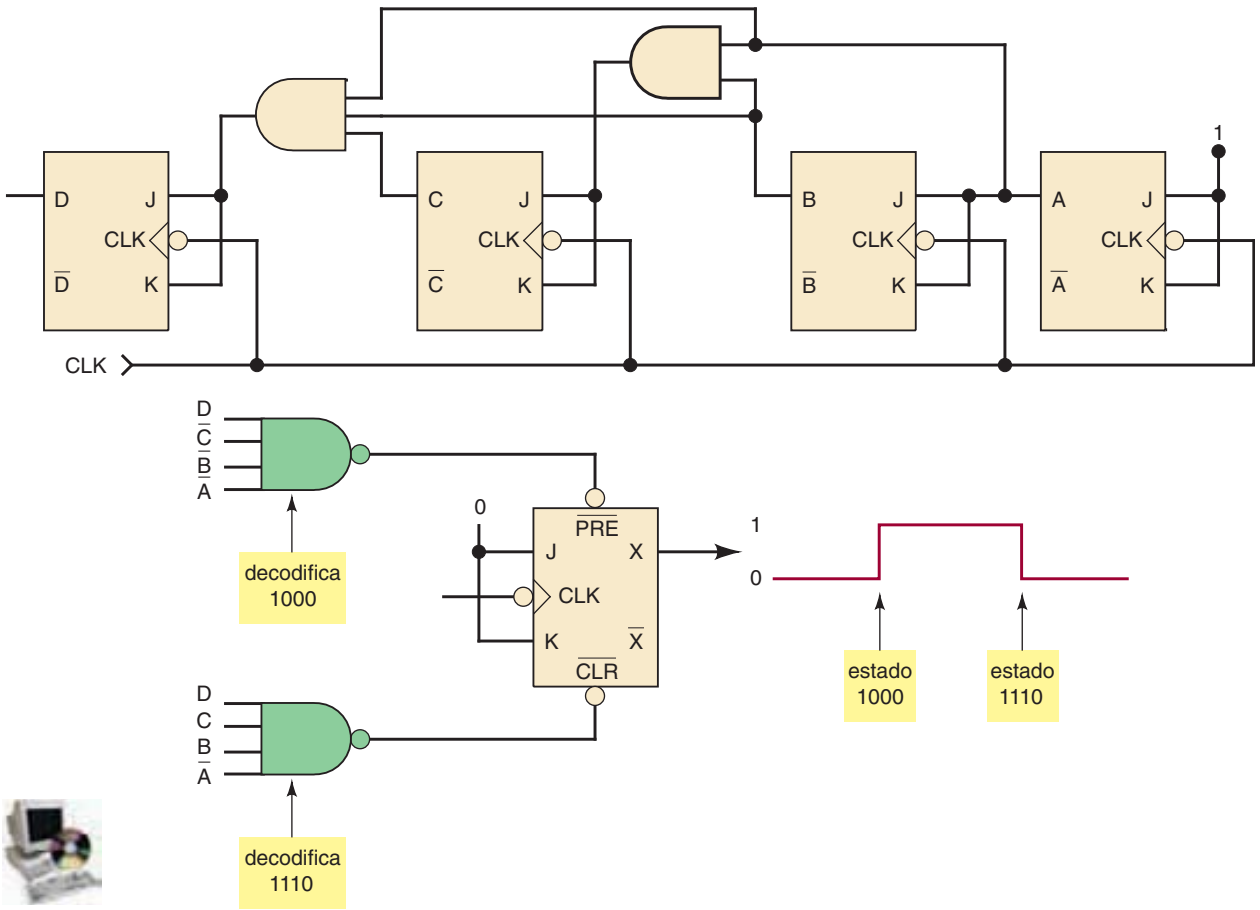


FIGURA 7-21 Ejemplo 7-15.

dos de los FFs del contador. Estas 10 salidas pueden utilizarse para controlar 10 LEDs indicadores individuales para un despliegue visual. Es más común utilizar un solo dispositivo de visualización para mostrar los números decimales del 0 al 9 en lugar de 10 LEDs separados. Una clase de visualizadores decimales contiene siete pequeños segmentos formados de un material (por lo general, LEDs o pantallas de cristal líquido) que emiten luz o reflejan la luz ambiental. Las salidas del decodificador BCD controlan qué segmentos se iluminan para producir un patrón que represente uno de los dígitos decimales.

En el capítulo 9 veremos con más detalle todos estos tipos de decodificadores y pantallas. No obstante y como los contadores BCD y sus decodificadores y exhibidores asociados son muy comunes, utilizaremos la unidad decodificadora/pantalla (vea la figura 7-22) para representar los circuitos completos que se utilizan para mostrar en forma visual el contenido de un contador BCD como un dígito decimal.

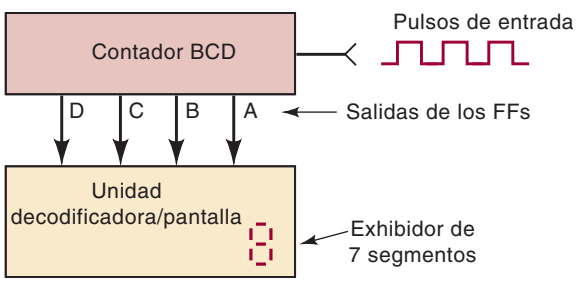


FIGURA 7-22 Por lo general, los contadores BCD muestran su conteo en un solo dispositivo de visualización.

PREGUNTAS DE REPASO

1. ¿Cuántas compuertas se necesitan para decodificar un contador de seis bits por completo?
2. Describa la compuerta decodificadora necesaria para producir una salida en BAJO cuando un contador MOD-64 se encuentra en el conteo de 23.

7-9 ANÁLISIS DE LOS CONTADORES SÍNCRONOS

Los circuitos tipo contador síncrono pueden diseñarse en forma personalizada para generar cualquier secuencia de conteo deseada. Podemos usar sólo las entradas síncronas que se apliquen a los flip-flops individuales para producir la secuencia del contador. Si no utilizamos controles asíncronos en los FFs (tales como los controles para limpiar) para cambiar la secuencia del contador, nunca tendremos que enfrentar los estados transitorios y las posibles deformaciones en las formas de onda de la salida. En la siguiente sección investigaremos el proceso de diseñar contadores completamente síncronos. Primero veamos cómo analizar el diseño de un contador de este tipo, prediciendo las entradas de control de los FFs para cada estado del contador. Una **tabla de estado PRESENTE / estado SIGUIENTE** es una herramienta muy útil en este proceso de análisis. El primer paso es escribir la expresión lógica para cada una de las entradas de control de los FFs. Después se asume un estado PRESENTE para el contador y se aplica esa combinación de bits a las expresiones lógicas de control. Las salidas de las expresiones de control nos permitirán predecir los comandos para cada FF y el estado SIGUIENTE resultante para el contador después de la sincronización por el reloj. Se repite el proceso de análisis hasta determinar toda la secuencia completa de conteo.

La figura 7-23 es un contador síncrono que tiene entradas J y K un poco distintas a las que vimos en la sección 7-3 para un contador ascendente binario común. Estos pequeños cambios en los circuitos de control harán que el contador produzca una secuencia de conteo distinta. Las expresiones de las entradas de control para este contador son:

$$J_C = A \cdot B$$
$$K_C = C$$
$$J_B = K_B = A$$
$$J_A = K_A = \overline{C}$$

Vamos a suponer que el estado PRESENTE para el contador es CBA = 000. Si aplicamos esta combinación a las expresiones de control antes mencionadas se producirá la condición J_CK_C = 0 0, J_BK_B = 0 0 y J_AK_A = 1 1. Estas entradas de control indicarán a los FFs C y B que retengan su estado y al FF A que conmute en la siguiente NGT de CLK. El estado SIGUIENTE previsto es 001 para CBA. Hemos

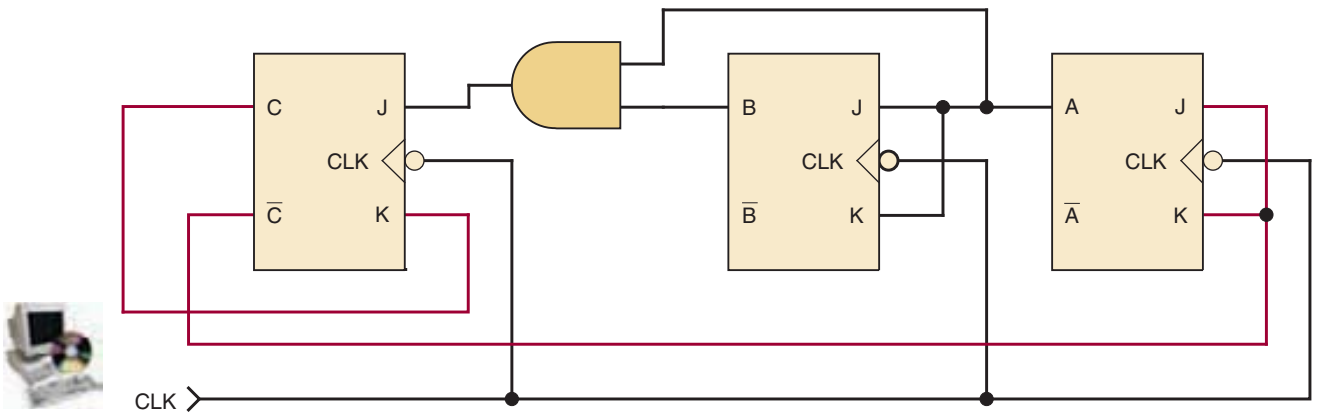


FIGURA 7-23 Contador síncrono con distintas entradas de control.

TABLA 7-1

Estado PRESENTE			Entradas de control						Estado SIGUIENTE		
C	B	A	J_C	K_C	J_B	K_B	J_A	K_A	C	B	A
0	0	0	0	0	0	0	1	1	0	0	1
0	0	1	0	0	1	1	1	1	0	1	0
0	1	0	0	0	0	0	1	1	0	1	1
0	1	1	1	0	1	1	1	1	1	0	0
1	0	0	0	1	0	0	0	0	0	0	0
1	0	1	0	1	1	1	0	0	0	1	1
1	1	0	0	1	0	0	0	0	0	1	0
1	1	1	1	1	1	1	0	0	0	0	1

introducido esta información en la primera línea de la tabla de estado PRESENTE / estado SIGUIENTE que se muestra en la tabla 7-1. A continuación podemos utilizar el estado 001 como nuestro estado PRESENTE. Si analizamos las expresiones de control con esta nueva combinación se producirá ahora la condición $J_CK_C = 0\ 0, J_BK_B = 1\ 1$ y $J_AK_A = 1\ 1$, lo cual nos da un comando de retención para el FF *C* y comandos de conmutación para los FFs B y A. Esto producirá el estado SIGUIENTE de 010 para CBA, que hemos listado en la segunda línea de la tabla 7-1. Si continuamos con este proceso se producirá una secuencia de conteo reciclable de 000, 001, 010, 011, 100, 000. Esta sería una secuencia de conteo MOD-5. Podemos predecir los estados SIGUIENTE para las tres posibles combinaciones de estado restantes de la misma forma. Al hacerlo podremos determinar si el diseño del contador es *autocorregible*. Un **contador autocorregible** es aquel en el que, por lo general, los estados sin utilizar regresarán de alguna forma a la secuencia de conteo normal. Si cualquiera de estos estados no utilizados no puede regresar a la secuencia normal, se dice que el contador no es autocorregible. Hemos introducido en la tabla 7-1 nuestras predicciones del estado SIGUIENTE para todos los posibles estados. La información resaltada indica que el diseño de este contador es autocorregible. La figura 7-24 muestra completos el diagrama de transición de estados y el diagrama de tiempos para este contador.

De igual forma podemos analizar la operación de los circuitos tipo contador que utilizan flip-flops *D* para almacenar el estado presente del contador. Por lo

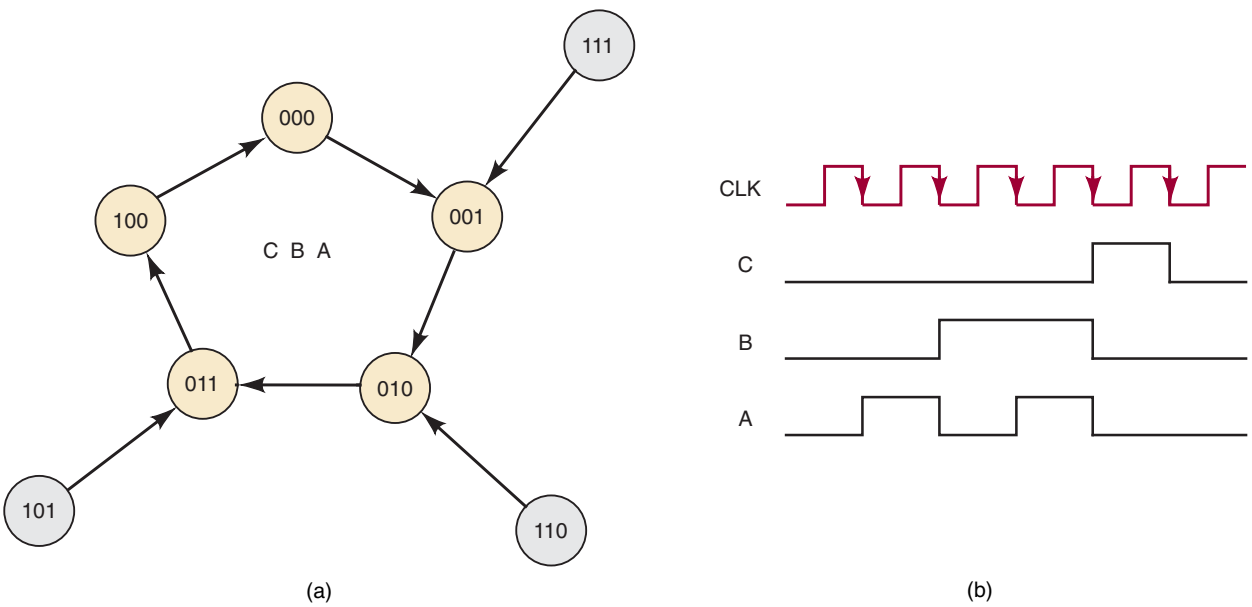


FIGURA 7-24 (a) Diagrama de transición de estados y (b) diagrama de sincronización para el contador asíncrono de la figura 7-23.

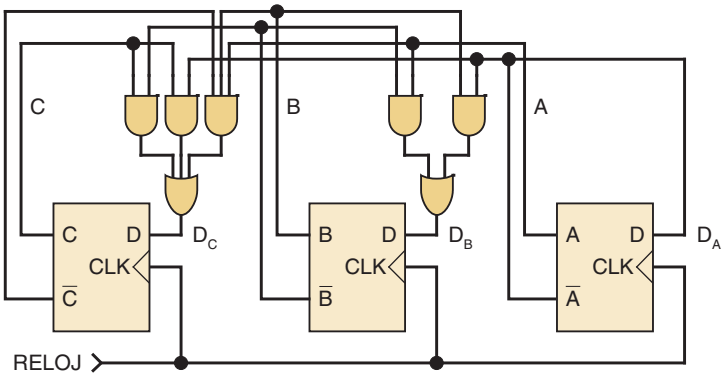


FIGURA 7-25
Contador síncrono que utiliza flip-flops D.



general, los circuitos de control para un flip-flop tipo *D* serán más complejos que para un contador tipo JK equivalente que produzca la misma secuencia de conteo, pero también tendremos que controlar la mitad del número de entradas síncronas. La mayoría de los PLDs utilizan flip-flops *D* para sus elementos de memoria, por lo que el análisis de este tipo de circuito contador nos proporcionará algo de detalle sobre la forma en que se programan los contadores dentro de un PLD.

La figura 7-25 muestra un contador síncrono diseñado con flip-flops *D*. El primer paso es escribir las expresiones lógicas para las entradas *D*:

$$D_C = C\overline{B} + C\overline{A} + \overline{C}BA$$
$$D_B = \overline{B}A + B\overline{A}$$
$$D_A = \overline{A}$$

Después determinaremos la tabla del estado PRESENTE / estado SIGUIENTE para el circuito contador; para ello debemos suponer un estado y aplicar ese conjunto de valores de bits a las expresiones de entrada antes descritas. Si elegimos CBA = 000 para el estado inicial del contador obtendremos como resultado $D_C = 0$, $D_B = 0$ y $D_A = 1$. Con una PGT en RELOJ los flip-flops "cargarán" el valor 001, que se convertirá en el estado SIGUIENTE del contador. Si utilizamos 001 como el estado PRESENTE se producirán entradas de $D_C = 0$, $D_B = 1$ y $D_A = 0$, de manera que 010 será el estado SIGUIENTE, y así en lo sucesivo. La tabla del estado PRESENTE / estado SIGUIENTE que se muestra en la tabla 7-2 indica que este circuito es un contador binario MOD-8 reciclable. Si aplicamos un poco de álgebra booleana a las expresiones de entrada, podremos ver que en realidad hay un patrón de circuito bastante simple para crear contadores binarios a partir de flip-flops *D*:

$$D_C = C\overline{B} + C\overline{A} + \overline{C}BA = C(\overline{B} + \overline{A}) + \overline{C}BA$$
$$= C\overline{BA} + \overline{C}(BA) = C \oplus (AB)$$
$$D_B = \overline{B}A + B\overline{A} = B \oplus A$$
$$D_A = \overline{A}$$

TABLA 7-2

Estado PRESENTE			Entradas de control			Estado SIGUIENTE		
C	*B*	*A*	D_C	D_B	D_A	*C*	*B*	*A*
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0

Es importante observar que los recursos de compuertas para la mayoría de los PLDs en realidad consisten de conjuntos de arreglos de circuitos AND-OR y que la expresión lógica SOP (Suma de Productos) describe con más precisión la implementación interna del circuito. No obstante, podemos ver que las expresiones se han simplificado de manera considerable mediante el uso de la función XOR. Esto nos lleva a predecir de manera correcta que para crear un contador binario MOD-16 con flip-flops *D* necesitaríamos un cuarto FF con:

$$D_D = D \oplus (A\,B\,C)$$

PREGUNTAS DE REPASO

1. ¿Por qué es conveniente evitar tener controles asíncronos en los contadores?
2. ¿Qué herramienta es útil para el análisis de los contadores síncronos?
3. ¿Qué es lo que determina la secuencia de conteo para un circuito tipo contador?
4. ¿Qué característica del contador se describe diciendo que es autocorregible?

7-10 DISEÑO DE UN CONTADOR SÍNCRONO*

Hay muchos arreglos de contador en forma de CIs: asíncronos, síncronos y combinación de asíncrono/síncrono. La mayoría cuenta con una secuencia binaria normal o en BCD, aunque sus secuencias de conteo pueden alterarse en cierta forma mediante los métodos para borrar o cargar que demostramos para las series de CIs 74ALS160-163 y 74ALS190-191. No obstante, hay situaciones en las que se requiere un contador personalizado que siga una secuencia que no sea un patrón de conteo binario regular, por ejemplo: 000, 010, 101, 001, 110, 000 ….

Existen varios métodos para diseñar contadores que siguen secuencias arbitrarias. Vamos a presentar los detalles para un método común que utiliza flip-flops J-K en una configuración de contador síncrono. El mismo método puede utilizarse en diseños con flip-flops D. La técnica es uno de varios procedimientos de diseño que forman parte de un área del diseño de circuitos digitales, conocida como **diseño de circuitos secuenciales**, que, por lo general, es parte de un curso avanzado.

Idea básica

En los contadores síncronos todos los FFs se sincronizan al mismo tiempo. Antes de cada pulso de reloj, las entradas *J* y *K* de cada FF en el contador deben estar en el nivel correcto para asegurar que el FF cambie al estado correcto. Por ejemplo, considere la situación en la que el estado 101 para el contador CBA debe ir seguido del estado 011. Cuando ocurre el siguiente pulso de reloj, las entradas *J* y *K* de los FFs deben estar en los niveles correctos que hagan que el flip-flop *C* cambie de 1 a 0, que el flip-flop *B* cambie de 0 a 1 y el flip-flop *A* de 1 a 1 (es decir, no cambie).

En consecuencia, el proceso de diseñar un contador síncrono se convierte en el proceso de diseñar los circuitos lógicos que *decodifiquen* los diversos estados del contador para suministrar los niveles lógicos apropiados a cada entrada J *y* K en el tiempo correcto. Las entradas para estos circuitos decodificadores provendrán de las salidas de uno o más FFs. Para ilustrar esto, en el contador síncrono de la figura 7-5 la compuerta AND que alimenta las entradas *J* y *K* del flip-flop *C* decodifica los estados de los flip-flops *A* y *B*. De igual forma, la compuerta AND que alimenta las entradas *J* y *K* del flip-flop *D* decodifica los estados de *A*, *B* y *C*.

* Puede omitirse este tema sin afectar la continuidad del resto del libro.

Tabla de excitación J-K

Antes de comenzar el proceso de diseñar los circuitos decodificadores para cada entrada J y K, primero debemos repasar la operación del flip-flop *J-K* utilizando un enfoque distinto, conocido como *tabla de excitación* (tabla 7-3). La columna más a la izquierda en esta tabla lista cada posible transición de salida de un FF. Las columnas segunda y tercera listan el estado PRESENTE del FF (simbolizado como Q_n) y el estado SIGUIENTE (simbolizado como Q_{n+1}) para cada transición. Las últimas dos columnas listan los niveles de J y K requeridos para producir cada transición. Ahora examinemos cada caso.

TRANSICIÓN 0 → 0 El estado PRESENTE del FF es 0 y deberá permanecer en 0 cuando se aplique un pulso de reloj. De lo que conocemos acerca de la forma en que funciona un flip-flop J-K, esto puede ocurrir cuando $J = K = 0$ (condición sin cambio) o cuando $J = 0$ y $K = 1$ (condición de limpiar). Por ende, J debe estar en 0 pero K puede estar en cualquier nivel. La tabla indica esto con un "0" bajo J y una "X" bajo K. Recuerde que "X" representa la condición de "no importa".

TRANSICIÓN 0 → 1 El estado PRESENTE es 0 y va a cambiar a 1, lo cual puede ocurrir cuando $J = 1$ y $K = 0$ (condición de establecer) o cuando $J = K = 1$ (condición de conmutación). En consecuencia, J debe ser 1 pero K puede estar en cualquier nivel para que ocurra la transición.

TRANSICIÓN 1 → 0 El estado PRESENTE es 1 y va a cambiar a 0, lo cual puede ocurrir cuando $J = 0$ y $K = 1$, o cuando $J = K = 1$. Por lo tanto, K debe ser 1 pero J puede estar en cualquier nivel.

TRANSICIÓN 1 → 1 El estado PRESENTE es 1 y va a permanecer en 1, lo cual puede ocurrir cuando $J = K = 0$ o cuando $J = 1$ y $K = 0$. Por ende, K debe ser 0 mientras que J puede estar en cualquier nivel.

El uso de esta **tabla de excitación J-K** (tabla 7-3) es una parte imprescindible del procedimiento de diseño de contadores síncronos.

TABLA 7-3 Tabla de excitación de un flip-flop J-K.

Transición en la salida del FF	Estado PRESENTE Q_n	Estado SIGUIENTE Q_{n+1}	J	K
0 → 0	0	0	0	*x*
0 → 1	0	1	1	*x*
1 → 0	1	0	*x*	1
1 → 1	1	1	*x*	0

Procedimiento de diseño

Ahora veremos todo un procedimiento de diseño de un contador síncrono completo. Aunque lo haremos para una secuencia de conteo específica, pueden seguirse los mismos pasos para cualquier secuencia deseada.

TABLA 7-4

C	*B*	*A*
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
0	0	0
0	0	1
etc.		

Paso 1. Determine el número deseado de bits (FFs) y la secuencia de conteo deseada.

Para nuestro ejemplo, diseñaremos un contador de tres bits que pasa por la secuencia mostrada en la tabla 7-4. Observe que esta secuencia no incluye los estados 101, 110 y 111. Nos referiremos a éstos como *estados indeseables*.

Paso 2. Dibuje el diagrama de transición de estados que muestre *todos* los estados posibles, incluyendo aquellos que no forman parte de la secuencia de conteo deseada.

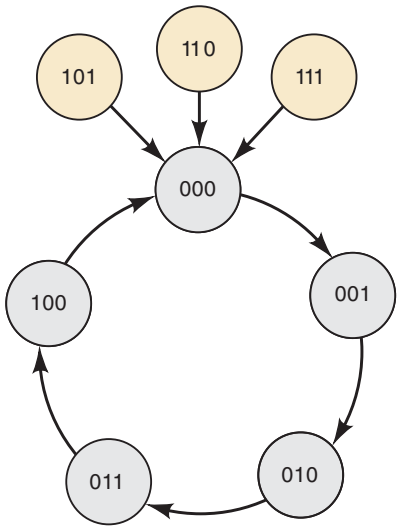


FIGURA 7-26 Diagrama de transición de estados para el ejemplo de diseño de un contador síncrono.

Para nuestro ejemplo el diagrama de transición de estados aparece como se muestra en la figura 7-26. Los estados 000 a 100 se conectan en la secuencia esperada. También hemos incluido un estado SIGUIENTE definido para cada uno de los estados indeseables. Esto se hizo en caso de que el contador llegue por accidente a uno de estos estados al momento del encendido o debido al ruido. El diseñador del circuito puede optar por hacer que cada uno de estos estados indeseables vaya a cualquier estado al momento en el que se aplique el siguiente pulso de reloj. De manera alternativa, el diseñador puede optar por no definir la acción del contador para los estados indeseables. En otras palabras, podríamos no preocuparnos por el estado SIGUIENTE para cualquier estado indeseable. Por lo general, el uso del método de diseño "no importa" antes mencionado producirá un diseño más simple, pero puede ser un problema potencial en la aplicación en donde se vaya a utilizar este contador. Para nuestro ejemplo de diseño optaremos por hacer que todos los estados indeseables se vayan al estado 000. Esto hará que nuestro diseño sea autocorregible pero algo distinto del ejemplo del contador MOD-5 que analizamos en la sección 7-9.

Paso 3. Utilice el diagrama de transición de estados para preparar una tabla que liste *todos* los estados PRESENTES y sus estados SIGUIENTES.

Para nuestro ejemplo, la tabla 7-5 muestra la información. La porción de la izquierda lista *todos* los posibles estados, aún aquellos que no forman parte de la secuencia. Vamos a etiquetarlos como los estados PRESENTES. La porción de la derecha lista el estado SIGUIENTE para cada estado PRESENTE. Éstos se obtienen del diagrama de transición de estados de la figura 7-26. Por ejemplo, la línea 1 muestra que el estado

TABLA 7-5

	Estado PRESENTE			Estado SIGUIENTE		
	C	*B*	*A*	*C*	*B*	*A*
Línea 1	0	0	0	0	0	1
2	0	0	1	0	1	0
3	0	1	0	0	1	1
4	0	1	1	1	0	0
5	1	0	0	0	0	0
6	1	0	1	0	0	0
7	1	1	0	0	0	0
8	1	1	1	0	0	0

PRESENTE 000 tiene el estado SIGUIENTE 001, y la línea 5 muestra que el estado PRESENTE 100 tiene el estado SIGUIENTE 000. Las líneas 6, 7 y 8 muestran que los estados PRESENTES indeseables 101, 110 y 111 tienen todos el estado SIGUIENTE 000.

Paso 4. Añada una columna a esta tabla para cada entrada J y K. Para cada estado PRESENTE, indique los niveles requeridos en cada entrada J y K para poder producir la transición al estado SIGUIENTE.

Nuestro ejemplo de diseño utiliza tres FFs (C, B y A) y cada uno de ellos tiene una entrada J y K. Por lo tanto, debemos agregar seis nuevas columnas como se muestra en la tabla 7-6. A esta tabla completa se le conoce como la **tabla de excitación del circuito**. Las seis nuevas columnas son las entradas J y K de cada FF. Las entradas bajo cada J y K se obtienen de la tabla 7-3, la tabla de excitación del flip-flop J-K que desarrollamos antes. Demostraremos esto para varios de los casos y usted podrá verificar el resto.

Veamos la línea 1 en la tabla 7-6. El estado PRESENTE 000 va a cambiar al estado SIGUIENTE 001 cuando ocurra un pulso de reloj. Para esta transición de estado, el flip-flop C cambia de 0 a 0. De la tabla de excitación J-K podemos ver que J_C debe estar en 0 y K_C en "X" para que ocurra esta transición. El flip-flop B también cambia de 0 a 0, por lo que $J_B = 0$ y $K_B = X$. El flip-flop A cambia de 0 a 1. También de la tabla 7-3 podemos ver que $J_A = 1$ y $K_A = x$ para esta transición.

En la línea 4 de la tabla 7-6, el estado PRESENTE 011 tiene un estado SIGUIENTE de 100. Para esta transición de estado, el flip-flop C cambia de 0 a 1, para lo cual requiere que $J_C = 1$ y $K_C = x$. Los flip-flops A y B cambian de 1 a 0. La tabla de excitación J-K indica que estos dos FFs necesitan que $J = X$ y $K = 1$ para que esto ocurra.

Los niveles requeridos de J y K para las demás líneas de la tabla 7-6 pueden determinarse de la misma forma.

Paso 5. Diseñe los circuitos lógicos necesarios para generar los niveles requeridos en cada entrada J y K.

La tabla 7-6 (la tabla de excitación del circuito) lista seis entradas J, K: J_C, K_C, J_B, K_B, J_A y K_A. Debemos considerar cada una de éstas como una salida de su propio circuito lógico con entradas de los flip-flops C, B y A. Después debemos diseñar el circuito para cada una. Vamos a diseñar el circuito para J_A.

Para ello, necesitamos analizar los estados PRESENTES de C, B y A, y los niveles deseados en J_A para cada caso. Hemos extraído esta información de la tabla 7-6 y la presentamos en la figura 7-27(a). La tabla de verdad muestra los niveles deseados en J_A para cada estado PRESENTE. Desde luego que para algunos de los casos, J_A es

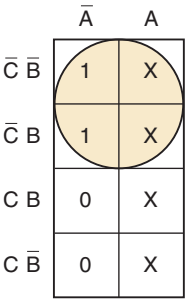
TABLA 7-6
Tabla de excitación del circuito.

	Estado PRESENTE			Estado SIGUIENTE								
	C	B	A	C	B	A	J_C	K_C	J_B	K_B	J_A	K_A
Línea 1	0	0	0	0	0	1	0	x	0	x	1	x
2	0	0	1	0	1	0	0	x	1	x	x	1
3	0	1	0	0	1	1	0	x	x	0	1	x
4	0	1	1	1	0	0	1	x	x	1	x	1
5	1	0	0	0	0	0	x	1	0	x	0	x
6	1	0	1	0	0	0	x	1	0	x	x	1
7	1	1	0	0	0	0	x	1	x	1	0	x
8	1	1	1	0	0	0	x	1	x	1	x	1

FIGURA 7-27 (a) Porción de la tabla de excitación del circuito que muestra a J_A para cada estado PRESENTE; (b) mapa K utilizado para obtener la expresión simplificada para J_A.

PRESENTE			
C	B	A	J_A
0	0	0	1
0	0	1	x
0	1	0	1
0	1	1	x
1	0	0	0
1	0	1	x
1	1	0	0
1	1	1	x

(a)



$J_A = \bar{C}$

(b)

una condición de "no importa". Para desarrollar el circuito lógico para J_A primero debemos determinar su expresión en términos de C, B y A. Para ello vamos a transferir la información de la tabla de verdad a un mapa de Karnaugh de tres variables y realizaremos la simplificación del mapa K, como se muestra en la figura 7-27(b).

Sólo hay dos 1s en este mapa K, y éstos pueden agruparse para obtener el término $\bar{A}\,\bar{C}$ pero si utilizamos las condiciones de "no importa" en $A\,\bar{B}\,\bar{C}$ y en $AB\bar{C}$ como 1s, podremos agrupar un cuádruple para obtener el término C más simple. Por ende, la expresión final es

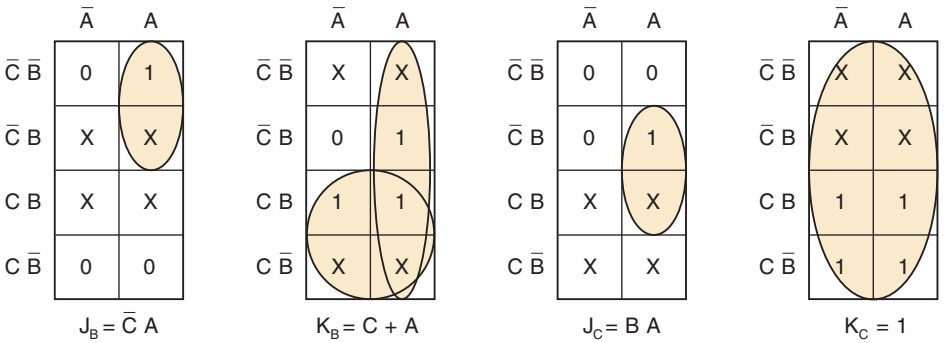
$$J_A = \bar{C}$$

Ahora vamos a considerar a K_A. Podemos seguir los mismos pasos que para J_A. No obstante, un análisis de las entradas bajo K_A en la tabla de excitación del circuito nos muestra sólo valores de 1 y "no importa". Si cambiamos todos los valores "no importa" por 1s, entonces K_A siempre será 1. Por lo tanto, la expresión final es

$$K_A = 1$$

De manera similar, podemos derivar las expresiones para J_C, K_C, J_B y K_B. Los mapas K para estas expresiones se proporcionan en la figura 7-28. Tal vez usted quiera confirmar que estén correctos, comparándolos con la tabla de excitación del circuito.

FIGURA 7-28
(a) Mapas K para los circuitos lógicos J_B y K_B; (b) mapas K para los circuitos lógicos J_C y K_C.



(a) (b)

Paso 6. Implemente las expresiones finales.

Los circuitos lógicos para cada entrada J y K se implementan a partir de las expresiones que se obtienen del mapa K. El diseño completo del contador síncrono se implementa en la figura 7-29. Observe que todos los FFs están sincronizados en paralelo. Tal vez quiera verificar que la lógica para las entradas J y K concuerde con las figuras 7-27 y 7-28.

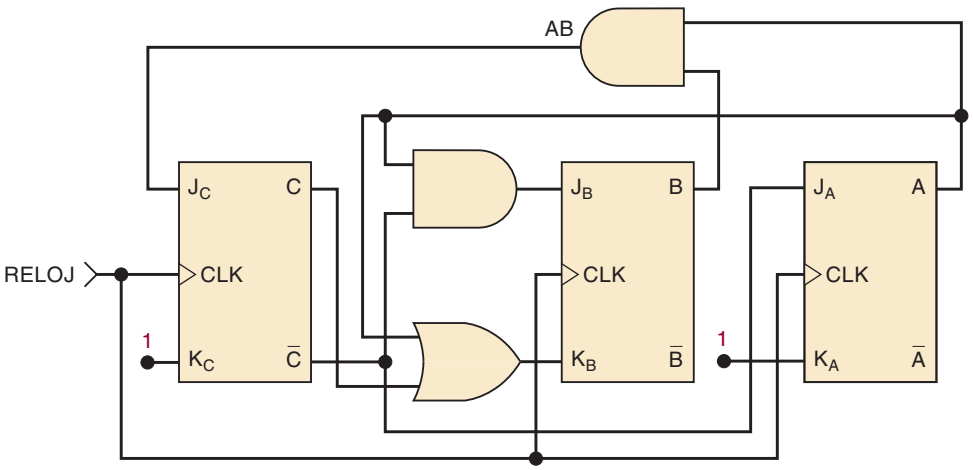
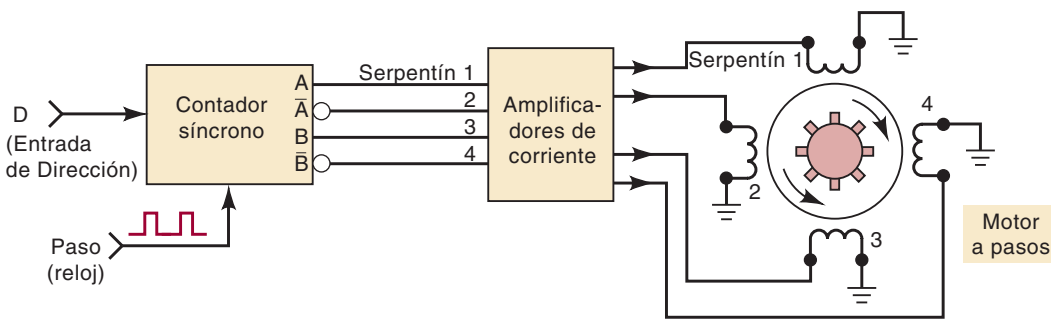


FIGURA 7-29
Implementación final del ejemplo del diseño del contador síncrono.

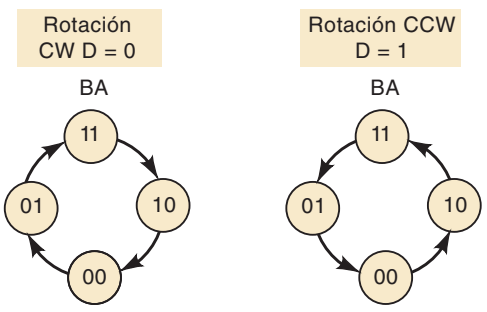
Control de un Motor a Pasos

Ahora aplicaremos este procedimiento de diseño a una situación práctica: el control de un *motor a pasos*. Este tipo de motor gira en pasos, por lo general, de 15° por paso, en vez de girar en un movimiento continuo. Los serpentines o devanados magnéticos dentro del motor deben energizarse y desenergizarse en una secuencia específica para poder producir esta acción de movimiento paso a paso. Por lo general, se utilizan señales digitales para controlar la corriente en cada uno de los serpentines del motor. Los motores a pasos se utilizan mucho en situaciones en las que se requiere un control preciso de posición, como en el posicionamiento de las cabezas de lectura/escritura en los discos magnéticos, en el control de las cabezas de impresión en impresoras, y en los robots.

La figura 7-30(a) es un diagrama de un motor a pasos ordinario con cuatro serpentines. Para que el motor gire en forma apropiada, los serpentines 1 y 2 deben



(a)



(b)

FIGURA 7-30 (a) Un contador síncrono suministra las salidas secuenciales apropiadas para controlar un motor a pasos; (b) diagramas de transición de estados para ambos estados de la entrada de Dirección, *D*.

estar siempre en estados opuestos; esto es, cuando el serpentín 1 está energizado, el serpentín 2 no lo está y viceversa. De igual forma, los serpentines 3 y 4 deben estar siempre en estados opuestos. Las salidas de un contador síncrono de dos bits se utilizan para controlar la corriente en los cuatro serpentines; A y $\overline{A}$ controlan los serpentines 1 y 2; B y $\overline{B}$ controlan los serpentines 3 y 4. Los amplificadores de corriente son necesarios ya que las salidas de los FFs no pueden suministrar la cantidad de corriente que requieren los serpentines.

Como este motor a pasos puede girar ya sea en sentido a favor de las manecillas del reloj (CW) o en contra de las manecillas del reloj (CCW), tenemos una entrada de Dirección (D) que se utiliza para controlar la dirección de rotación. Los diagramas de estado en la figura 7-30(b) muestran los dos casos. Para que ocurra la rotación CW debemos tener $D = 0$, y el estado del contador BA debe seguir la secuencia 11, 10, 00, 01, 11, 10,..., y así en lo sucesivo, a medida que se sincroniza mediante la señal de entrada Paso. Para la rotación CCW, $D = 1$ y el contador debe seguir la secuencia 11, 01, 00, 10, 11, 01,...., y así en lo sucesivo.

TABLA 7-7

Estado PRESENTE			Estado SIGUIENTE		Entradas de control			
D	B	A	B	A	J_B	K_B	J_A	K_A
0	0	0	0	1	0	*x*	1	*x*
0	0	1	1	1	1	*x*	*x*	0
0	1	0	0	0	*x*	1	0	*x*
0	1	1	1	0	*x*	0	*x*	1
1	0	0	1	0	1	*x*	0	*x*
1	0	1	0	0	0	*x*	*x*	1
1	1	0	1	1	*x*	0	1	*x*
1	1	1	0	1	*x*	1	*x*	0

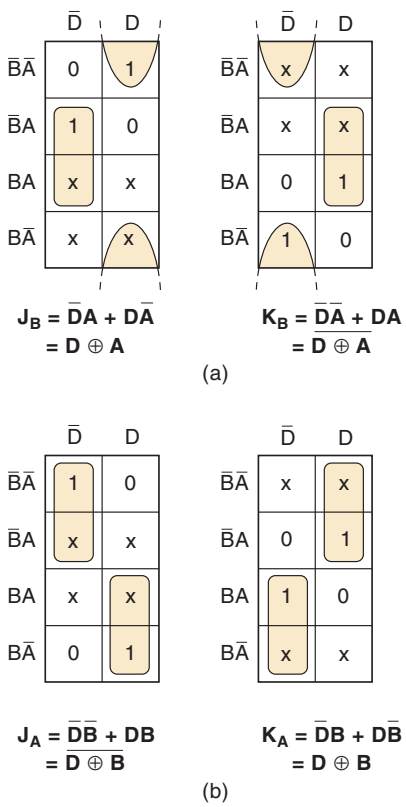


FIGURA 7-31 (a) Mapas K para J_B y K_B; (b) mapas K para J_A y K_A.

Ahora estamos listos para seguir los seis pasos del procedimiento de diseño de contadores síncronos. Ya hemos realizado los pasos 1 y 2, por lo que podemos proceder con los pasos 3 y 4. La tabla 7-7 muestra cada uno de los posibles estados PRESENTEs de *D*, *B* y *A*, y el estado SIGUIENTE deseado, junto con los niveles en cada entrada *J* y *K* necesarios para lograr las transiciones. En todos los casos, la entrada de Dirección *D* no cambia al pasar del estado PRESENTE al estado SIGUIENTE ya que es una entrada independiente que se mantiene en ALTO o en BAJO, a medida que el contador avanza a través de su secuencia.

El paso 5 del proceso de diseño se presenta en la figura 7-31, en donde la información de la tabla 7-7 se ha transferido a los mapas K que muestran cómo cada señal *J* y *K* está relacionada con los PRESENTES estados de *D*, *B* y *A*. Usando el agrupamiento apropiado se obtienen las expresiones lógicas simplificadas para cada señal *J* y *K*.

El paso final se muestra en la figura 7-32, en donde el contador síncrono de dos bits se implementa mediante el uso de las expresiones *J*, *K* que se obtuvieron de los mapas K.

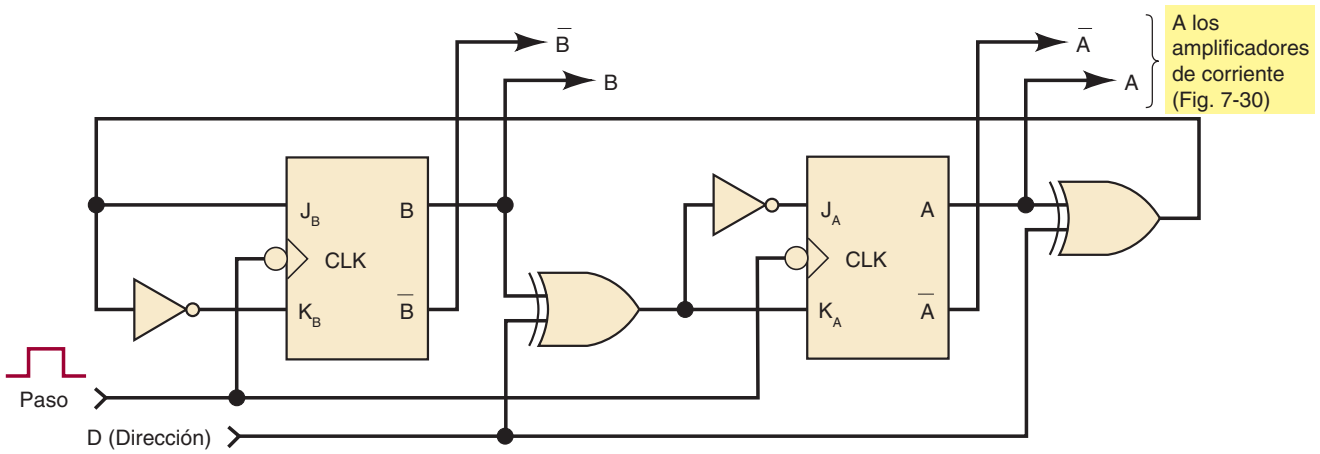


FIGURA 7-32 Contador síncrono implementado a partir de las ecuaciones *J*, *K*.

Diseño de un contador síncrono con un FF D

Hemos proporcionado un procedimiento detallado para diseñar contadores síncronos mediante flip-flops K. Siempre se han utilizado los flip-flops J-K para implementar contadores debido a que los circuitos lógicos necesarios para las entradas *J* y *K* son, por lo general, más simples que los circuitos lógicos necesario para controlar un contador síncrono equivalente mediante flip-flops *D*. Al diseñar contadores que se van a implementar en PLDs, en donde, por lo general, hay una cantidad abundante de compuertas disponibles, es conveniente usar flip-flops D en vez de J-K. Veamos ahora el diseño de contadores síncronos mediante el uso de FFs D.

Este método de diseño de circuitos contadores es aún más sencillo que utilizar flip-flops J-K. Para demostrar esto vamos a diseñar un circuito con FFs D que produzca la misma secuencia de conteo que se muestra en la figura 7-26. Los primeros tres pasos para el diseño de contadores D síncronos son idénticos a la técnica con flip-flops J-K. El paso 4 para el diseño con FFs D es trivial, ya que las entradas D necesarias son las mismas que el estado SIGUIENTE deseado, como puede verse en la tabla 7-8. El paso 5 es generar las expresiones lógicas a partir de la tabla de estado PRESENTE / estado SIGUIENTE para las entradas D. La figura 7-33 muestra

TABLA 7-8

Estado PRESENTE			Estado SIGUIENTE			Entradas de control		
C	B	A	C	B	A	D_C	D_B	D_A
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0

los mapas K y las expresiones simplificadas. Por último, para el paso 6 puede implementarse el contador con el circuito que se muestra en la figura 7-34.

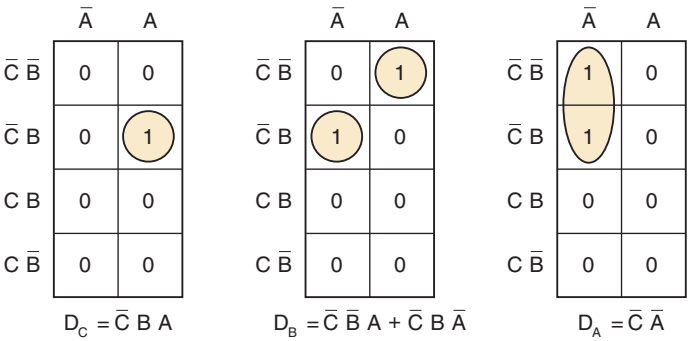


FIGURA 7-33 Los mapas K y las expresiones lógicas simplificadas para el diseño de contadores MOD-5 con flip-flops.

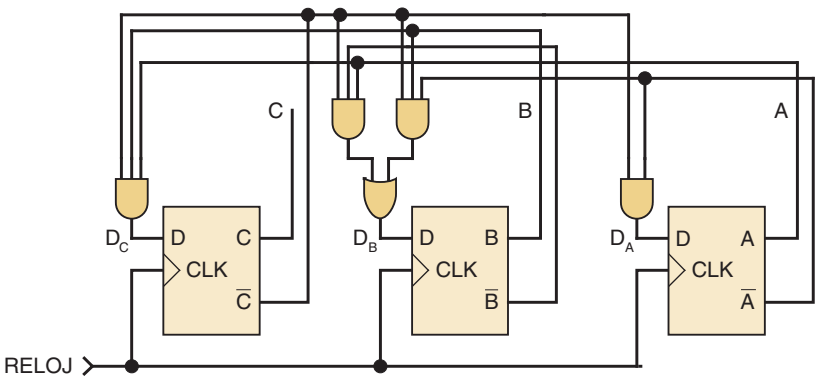


FIGURA 7-34 Implementación del circuito del diseño de un contador MOD-5 con flip-flops D.

PREGUNTAS DE REPASO

1. Liste los seis pasos en el procedimiento para diseñar un contador síncrono.
2. ¿Qué información contiene una tabla de estado PRESENTE / estado SIGUIENTE?
3. ¿Qué información contiene la tabla de excitación del circuito?
4. *Verdadero o falso:* el procedimiento de diseño de un contador síncrono puede utilizarse para la siguiente secuencia: 0010, 0011, 0100, 0111, 1010, 1110, 1111 y se repite.

7-11 CONTADORES BÁSICOS MEDIANTE EL USO DE HDL

En el capítulo 5 estudiamos los flip-flops y los métodos que se utilizan con los HDLs para representar los circuitos con flip-flops. La última sección en el capítulo 5 ilustró cómo conectar componentes de FF de una forma muy parecida a como se conectan los circuitos integrados entre sí. Al conectar la salida *Q* de un FF con la entrada de reloj del siguiente FF descubrimos que podía crearse un circuito. Al proceso de utilizar un HDL para describir las conexiones de los componentes se le conoce como el nivel estructural de abstracción. Es obvio que sería muy tedioso construir un circuito complicado mediante el uso de los métodos estructurales; además sería muy difícil de leer y de interpretar. En esta sección ampliaremos nuestro uso de HDL para describir circuitos mediante el uso de métodos que se consideran niveles más altos de abstracción. Este término suena intimidante, pero sólo significa que hay muchas formas más concisas y sensibles de describir lo que queremos que haga un contador, sin necesidad de preocuparnos por los detalles acerca de cómo cablear circuitos de flip-flops para hacerlo.

Aún es vital que comprendamos los principios fundamentales de la operación de los flip-flops, en comparación con las compuertas lógicas combinacionales. Como podrá recordar, los flip-flops tienen las siguientes características únicas. Por lo general, la salida se actualiza de acuerdo con la condición de las entradas de control síncronas cuando se produce el *flanco activo* del reloj, lo cual significa que hay un estado lógico en la salida *Q* antes del flanco del reloj (estado PRESENTE) y es posible que haya un estado distinto en la salida *Q* después del flanco del reloj (estado SIGUIENTE). Un flip-flop "recuerda" o retiene su estado entre los pulsos de reloj, sin importar los cambios en las entradas de control síncronas (por ejemplo, *J* y *K*).

Los circuitos tipo contador que utilizan HDL se basan en esta comprensión básica de un circuito que pasa a través de una secuencia de estados en respuesta al evento de un flanco de reloj. Los contadores de rizo proporcionan un circuito fácil de analizar y de comprender. También son mucho menos complicados de construir mediante el uso de flip-flops y compuertas lógicas, en comparación con sus contrapartes síncronos. El problema con los contadores de rizo es la combinación del retraso de tiempo y los estados temporales espurios que ocurren cuando el contador cambia de estado. Cuando avancemos al siguiente nivel de abstracción y planeemos usar PLDs para implementar nuestro diseño, ya no nos enfocaremos en las cuestiones relacionadas con el cableado, sino en describir la operación del circuito en forma concisa. En consecuencia, los métodos que utilizamos para describir circuitos tipo contador mediante el uso de HDL utilizan en su mayor parte técnicas síncronas, en las que todos los flip-flops se actualizan simultáneamente, en respuesta al mismo evento de reloj. Todos los bits en una secuencia de conteo pasan de su estado PRESENTE a su estado SIGUIENTE prescrito simultáneamente, previniendo, por lo tanto, cualquier estado espurio inmediato.

Métodos de descripción de transición de estados

El siguiente método para describir circuitos que necesitamos examinar utiliza tablas. Este método no se relaciona con la conexión de los puertos de los componentes, sino con la asignación de valores a objetos tales como puertos, señales y variables. En otras palabras, describe la manera en que los datos de salida se relacionan con los datos de entrada en el circuito. Ya hemos utilizado este método en varios de los circuitos introductorios en los capítulos 3 y 4, en forma de tablas de verdad. Con los circuitos tipo contador secuencial el equivalente de la tabla de verdad es la tabla de estado PRESENTE / estado SIGUIENTE, como vimos en la sección anterior. En esencia, podemos utilizar el HDL para describir la tabla de estado PRESENTE / estado SIGUIENTE y evitar así los tediosos detalles de generar las ecuaciones booleanas, como lo hicimos en la sección 7-10 para diseñar usando dispositivos lógicos estándar.

DESCRIPCIONES DE ESTADO EN AHDL

Como ejemplo de un circuito contador simple, implementaremos el contador MOD-5 de la figura 7-26 en AHDL. Las entradas y las salidas se definen en la sección SUBDESIGN de la figura 7-35, como siempre. En la sección VARIABLE de la línea 7 hemos declarado (o instanciado) un arreglo de tres bits de primitivas DFF, las cuales reciben el nombre de instancia *conteo[ ]*. Este arreglo se tratará como un registro de tres bits en el diseño; en esencia, definiremos qué valor debe almacenarse para cada estado SIGUIENTE. Como éste es un contador síncrono, necesitamos enlazar todas las entradas *clk* de cada DFF con la entrada *reloj* de SUBDESIGN. Esto se logra en AHDL mediante la siguiente instrucción en la sección lógica:

```
conteo[ ].clk = reloj;
```

Las primitivas de flip-flop que se proporcionan en AHDL tienen entradas y salidas estándar, a las cuales se les conoce como "puertos". Estos puertos se etiquetan con base en un nombre estándar de puerto que se adjunta al nombre de instancia de los flip-flops. Como puede ver en la tabla 5-3, el nombre del puerto de reloj es *.clk*, una entrada *D* se llama *.d* y la salida del FF tiene el nombre *.q*. Para implementar la tabla de estado PRESENTE / estado SIGUIENTE se utiliza una instrucción CASE. Para cada uno de los posibles valores del registro *conteo[ ]* determinamos el valor que debe colocarse en las entradas *D* de los flip-flops, el cual determinará el estado SIGUIENTE del contador. La instrucción en la línea 21 asigna el valor en *conteo[ ]* a las terminales de salida. Sin esta línea el contador quedaría "oculto" en el bloque SUBDESIGN y no sería visible para el mundo exterior.

En la figura 7-36 se muestra una solución de diseño alternativa. Hay dos modificaciones de la figura 7-35. La primera se ve en la línea 7, en donde el nombre del arreglo para los flip-flops D es ahora el mismo que el puerto de salida para el bloque SUBDESIGN.

```
1   SUBDESIGN fig7 35
2   (
3         reloj         :INPUT;
4         q[2..0]       :OUTPUT;
5   )
6   VARIABLE
7         conteo[2..0]  :DFF;       crea un registro de 3 bits
8   BEGIN
9         conteo[].clk = reloj;     conecta todos los relojes en paralelo
10
11              CASE conteo[] IS
12                    Presente           Siguiente
13
14                     WHEN  0      =>    conteo[].d = 1;
15                     WHEN  1      =>    conteo[].d = 2;
16                     WHEN  2      =>    conteo[].d = 3;
17                     WHEN  3      =>    conteo[].d = 4;
18                     WHEN  4      =>    conteo[].d = 0;
19                     WHEN OTHERS  =>    conteo[].d = 0;
20               END CASE;
21         q[] = conteo[];           asigna el registro a las terminales de salida
22  END;
```

FIGURA 7-35 Contador MOD-5 en AHDL.

FIGURA 7-36 Otra versión del contador MOD-5 descrito en la figura 7-26.

```
1   SUBDESIGN fig7 36
2   (
3       reloj      :INPUT;
4       q[2..0]    :OUTPUT;
5   )
6   VARIABLE
7      q[2..0]     :DFF;       crea un registro de 3 bits
8   BEGIN
9      q[].clk = reloj;        conecta todos los relojes en paralelo
10     TABLE
11        q[].q =>     q[].d;
12        0     =>     1;
13        1     =>     2;
14        2     =>     3;
15        3     =>     4;
16        4     =>     0;
17        5     =>     0;
18        6     =>     0;
19        7     =>     0;
20     END TABLE;
21  END;
```

Esto conectará automáticamente las salidas del flip-flop a las salidas de SUBDESIGN y eliminará la necesidad de incluir una instrucción de asignación como la línea 21 en la primera solución. La segunda modificación es el uso de una instrucción TABLE de AHDL en vez de la instrucción CASE utilizada en la figura 7-35. En la línea 11, el puerto *.q* en el arreglo DFF *q[ ]* representa el lado del estado PRESENTE de la tabla, mientras que el puerto *.d* para *q[ ]* representa el estado SIGUIENTE que se introducirá en el conjunto de entradas D del arreglo, cuando se aplique una PGT a *reloj*.

DESCRIPCIONES DE ESTADO EN VHDL

Como ejemplo de un circuito de contador simple, implementaremos el contador MOD-5 de la figura 7-26 en VHDL. Nuestro fin en este ejemplo es demostrar un contador mediante una estructura de control similar a una tabla de estado PRESENTE / estado SIGUIENTE. En VHDL se deben realizar dos tareas clave: detectar el flanco de reloj deseado y asignar el estado SIGUIENTE apropiado al contador. En capítulos anteriores sobre flip-flops vimos que puede usarse un bloque PROCESS para responder a una transición de una señal de entrada. Además aprendimos que una instrucción CASE puede evaluar una expresión y, para cualquier valor de entrada válido, asignar un valor correspondiente a otra señal. El código de la figura 7-37 utiliza una instrucción PROCESS y una instrucción CASE para implementar este contador. Las entradas y salidas están definidas en la declaración ENTITY, como en ejemplos pasados.

Cuando se utiliza VHDL para describir un contador, debemos encontrar una manera de "almacenar" el estado del contador entre los pulsos de reloj (es decir, la acción de un flip-flop). Esto se hace en una de dos formas: mediante señales (instrucción SIGNAL) o mediante variables (instrucción VARIABLE). Hemos usado señales SIGNAL con frecuencia en ejemplos anteriores que operaban en forma concurrente. Una señal en VHDL retiene el último valor que se le asignó, en forma muy parecida a un flip-flop. En consecuencia, podemos usar una señal como objeto de datos para representar el valor del contador. Después, esta señal puede usarse para

```
1   ENTITY fig7 37 IS
2   PORT  (
3            reloj  :IN BIT;
4            q      :OUT BIT VECTOR(2 DOWNTO 0)
5         );
6   END fig7 37 ;
7
8   ARCHITECTURE a OF fig7 37    IS
9   BEGIN
10     PROCESS (reloj)                                  responde a la entrada clk
11     VARIABLE conteo: BIT VECTOR(2 DOWNTO 0);        crea un registro de 3 bits
12     BEGIN
13        IF (reloj = '1' AND reloj'EVENT) THEN         disparo en flanco de subida
14           CASE conteo IS
15              Presente              Siguiente
16
17              WHEN "000"    =>    conteo := "001";
18              WHEN "001"    =>    conteo := "010";
19              WHEN "010"    =>    conteo := "011";
20              WHEN "011"    =>    conteo := "100";
21              WHEN "100"    =>    conteo := "000";
22              WHEN OTHERS   =>    conteo := "000";
23           END CASE;
24        END IF;
25        q <= conteo;              asigna el registro a las terminales de salida
26     END PROCESS;
27  END a;
```

FIGURA 7-37 Contador MOD-5 en VHDL.

conectar el valor del contador con cualquier otro elemento en la descripción de la arquitectura.

En este diseño hemos optado por usar una **VARIABLE** en lugar de una señal (SIGNAL) como el objeto de datos que almacena el valor del contador. Las VARIABLEs no son idénticas a las señales porque no se utilizan para conectar diversas partes del diseño. En vez de ello se utilizan como una ubicación local para "almacenar" un valor. Las variables se consideran como objetos de datos locales, ya que se reconocen sólo dentro del bloque PROCESS en el que se declaran. En la línea 11 de la figura 7-37 se declara la variable llamada *conteo* dentro del bloque PROCESO, antes de la instrucción BEGIN. Su tipo es el mismo que el del puerto de salida *q*. La palabra clave PROCESS en la línea 10 va seguida de la lista de sensibilidad que contiene la señal de entrada *reloj*. Cada vez que *reloj* cambia de estado se invoca el bloque PROCESS y se evalúan las instrucciones dentro de PROCESS para producir un resultado. Un atributo 'EVENT se evaluará como VERDADERO si la señal que va antes de él acaba de cambiar de estado. La línea 13 establece que si *reloj* acaba de cambiar de estado y justo ahora es '1', entonces sabemos que fue un flanco de subida. Para implementar la tabla de estado PRESENTE/estado SIGUIENTE se utiliza una instrucción CASE. Para cada uno de los posibles valores de la variable *conteo* determinamos el estado SIGUIENTE del contador. Observe que se utiliza el operador "=" para asignar un valor a una variable. La línea 25 asigna el valor almacenado en *conteo* a las terminales de salida. Como *conteo* es una variable local, esta asignación debe realizarse antes de la instrucción END PROCESS en la línea 26.

Descripción del comportamiento

El **nivel de abstracción de comportamiento** es una manera de describir un circuito mediante la descripción de su comportamiento en términos muy similares a la manera en que podría describirse su operación en español. Piense acerca de la forma en que podría describirse la operación de un circuito contador por alguien que no conozca nada acerca de los flip-flops o las compuertas lógicas. Tal vez la descripción de esa persona sería algo así como: "Cuando la entrada del contador cambie de BAJO a ALTO, el número en la salida cuenta en forma ascendente por 1". Este nivel de descripción se relaciona más con las relaciones de causa y efecto que con la ruta del flujo de datos o los detalles del cableado. Sin embargo, en realidad no podemos tan solo utilizar cualquier descripción en español para describir el comportamiento del circuito. Debe utilizarse la sintaxis apropiada dentro de las restricciones del HDL.

AHDL

En AHDL, el primer paso importante en este método de descripción es declarar las terminales de salida del contador en forma apropiada. Deben declararse como un arreglo de bits, en donde los índices se decrementen de izquierda a derecha y 0 sea el índice menos significativo en el arreglo, algo opuesto a los bits individuales llamados a, b, c, d, y así en lo sucesivo. De esta forma, el valor numérico asociado con el nombre del arreglo de bits se interpreta como un número binario con base en el cual pueden realizarse ciertas operaciones aritméticas. Por ejemplo, el arreglo de bits *conteo* que se muestra en la figura 7-38 podría contener los bits 1001, como se muestra. El compilador de AHDL interpreta este patrón de bits como si tuviera el valor de 9 en decimal.

Para poder crear nuestro contador MOD-5 en AHDL, necesitaremos un registro de tres bits que almacene el estado actual del contador. Este arreglo de tres bits, al cual llamaremos *conteo*, se declara mediante el uso de flip-flops D en la línea 7 de la figura 7-39. Si recuerda de la figura 7-36, podemos nombrar el arreglo DFF igual que el puerto de salida *q[2..0]* y en consecuencia eliminar la línea 15, pero también tendríamos que cambiar *conteo[ ]* por *q[ ]* en cualquier parte en donde aparezca dentro de la sección lógica. En otras palabras, la instrucción de la línea 7 puede cambiarse por

```
q[2..0]    :DFF;.
```

Si hiciéramos esto, todas las referencias a *conteo* a partir de ese punto se cambiarían por *q*. Esto puede hacer que el código sea más corto, pero no demuestra los conceptos universales de HDL con tanta claridad. En AHDL todos los relojes pueden especificarse como si estuvieran enlazados entre sí y conectados a una fuente de reloj común, mediante el uso de la instrucción de la línea 10, *conteo[ ].clk = reloj*. En este ejemplo, *conteo[ ].clk* hace referencia a la entrada de reloj de cada flip-flop en el arreglo llamado *conteo*.

La descripción del comportamiento de este contador es muy simple. El estado actual del contador se evalúa (*conteo[ ].q*) en la línea 11, y si es menor que el valor de conteo deseado más alto, utiliza la descripción *conteo[ ].d = conteo.q + 1* (línea 12). Esto significa que el estado actual de las entradas *D* debe ser igual a un valor que sea un conteo mayor que el estado actual de las salidas *Q*. Cuando el estado actual del contador ha llegado al estado deseado más alto (o mayor), la prueba de la instrucción *IF* será falsa y se producirá un valor de cero en la entrada del estado

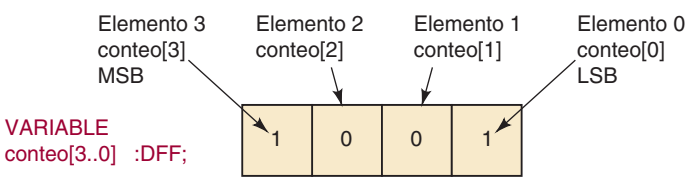


FIGURA 7-38 Los elementos de un registro D, en el que se almacena el número 9.

```
1   SUBDESIGN fig7 39
2   (
3      reloj    :INPUT;
4      q[2..0]  :OUTPUT;     declara arreglo de 3 bits de los bits de salida
5   )
6   VARIABLE
7      conteo[2..0]  :DFF;    declara un registro de flip flops D.
8
9   BEGIN
10     conteo[].clk = reloj;     conecta todos los relojes a la fuente síncrona
11     IF conteo[].q < 4 THEN      nota; conteo[ ] es lo mismo que conteo[ ].q
12        conteo[].d = conteo[].q + 1;    incrementa en uno el valor actual
13     ELSE conteo[].d = 0;        se recicla a cero: fuerza los estados no usados a 0
14     END IF;
15     q[] = conteo[];              transfiere el contenido del registro a las salidas
16  END;
```

FIGURA 7-39 Descripción del comportamiento de un contador en AHDL

SIGUIENTE (línea 13), con lo cual se reciclará el contador. La última instrucción en la línea 15 sólo conecta el valor del contador con las terminales de salida del dispositivo.

VHDL

VHDL

En VHDL el primer paso importante en este método de descripción es declarar en forma apropiada el puerto de salida del contador, como se muestra en la figura 7-40. El tipo de datos del puerto de salida (línea 3) debe concordar con el tipo de la

```
1   ENTITY fig7 40 IS
2   PORT( reloj   :IN BIT;
3         q  :OUT INTEGER RANGE 0 TO 7  );
4   END  fig7 40;
5
6   ARCHITECTURE a OF fig7 40 IS
7   BEGIN
8      PROCESS (reloj)
9      VARIABLE conteo: INTEGER RANGE 0 to 7;     define una VARIABLE numérica
10        BEGIN
11          IF (reloj = '1' AND reloj'EVENT) THEN      flanco ascendente?
12             IF conteo < 4 THEN            menor que máx?
13                conteo := conteo + 1;      incrementa el valor
14             ELSE                          debe estar en máx o mayor
15                conteo := 0;               se recicla a cero
16             END IF;
17          END IF;
18     q <= conteo;                transfiere el contenido del registro a las salidas
19     END PROCESS;
20  END a;
```

FIGURA 7-40 Descripción del comportamiento de un contador en VHDL.

variable del contador (línea 9) y debe ser un tipo que permita operaciones aritméticas. Recuerde que VHDL trata a los objetos BIT_VECTOR como una cadena de bits, no como una cantidad numérica binaria. Para poder reconocer la señal como una cantidad numérica, el objeto de datos debe escribirse como tipo INTEGER. El compilador busca en la cláusula RANGE 0 TO 7 de la línea 3 y sabe que el contador necesita tres bits. Se necesita una declaración similar para la variable del registro en la línea 9, para que cuente hacia arriba. A ésta se le llamará *conteo*. La primera instrucción después de BEGIN en el bloque PROCESS responde al flanco de subida del reloj, como en los ejemplos anteriores. Después utiliza métodos de descripción del comportamiento para definir la respuesta del contador al flanco del reloj. Si el contador no ha llegado a su máximo (línea 12) entonces deberá incrementarse (línea 13). En caso contrario (línea 14) deberá reciclar el contador a cero (línea 15). La última instrucción en la línea 18 sólo conecta el valor del contador con las terminales de salida del dispositivo.

Simulación de contadores básicos

La simulación de cualquiera de nuestros diseños de contadores MOD-5 es bastante simple. Los contadores sólo tienen un bit de entrada (*reloj*) y tres bits de salida (*q2 q1 q0*) para mostrar en la simulación. La frecuencia de reloj no se ha especificado, por lo que podemos utilizar cualquier frecuencia que queramos para una simulación funcional; aunque tal vez deberíamos evitar un reloj de alta frecuencia, a menos que se desee investigar los efectos de los retrasos de propagación. La única decisión que debemos hacer es cuántos pulsos aplicar. Como el contador es MOD-5, debemos aplicar por lo menos cinco pulsos de reloj para verificar que el diseño de HDL tenga la secuencia de conteo correcta y que se recicle. Empezaremos la simulación con el estado inicial 000, ya que los PLDs de Altera tienen una característica de reinicio de encendido integrada. No podremos probar cualquiera de los estados no usados, ya que los diseños de HDL no proporcionan la manera de prestablecer el contador a cualquiera de los estados no usados. En la figura 7-41 se muestran los resultados que obtuvimos con nuestra simulación para el diseño en HDL de un contador MOD-5.

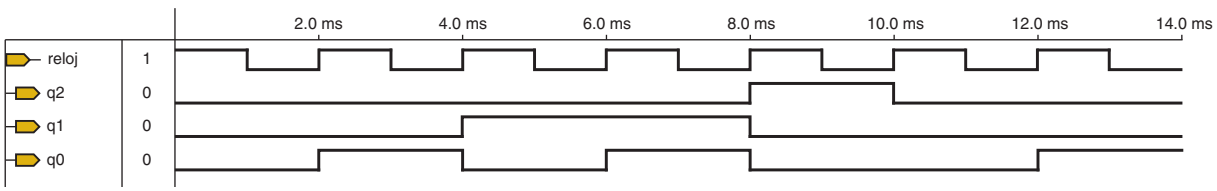


FIGURA 7-41 Resultados de la simulación para el diseño en HDL de un contador MOD-5.

PREGUNTAS DE REPASO

1. ¿Qué tipo de tabla se utiliza para describir la operación de un contador?
2. Al diseñar un contador con flip-flops D, ¿qué se aplica a las entradas D para poder excitarlo hacia el estado SIGUIENTE en el siguiente flanco activo del reloj?
3. ¿Cómo escribiría la descripción en HDL para disparar un dispositivo de almacenamiento (flip-flop) en un flanco de caída, en vez de un flanco de subida del reloj?
4. ¿Qué método describe la operación del circuito mediante el uso de relaciones tipo causa y efecto?

7-12 CONTADORES CON TODAS LAS CARACTERÍSTICAS EN HDL

Los ejemplos que hemos elegido hasta ahora han sido de contadores muy básicos. Todo lo que hacen es contar hasta cuatro y luego se regresan a cero. Los contadores de CI estándar que hemos examinado tienen muchas otras características que los hacen muy útiles para numerosas aplicaciones digitales. Por ejemplo, considere los contadores de CI 74161 y 74191 que vimos en la sección 7-7. Estos dispositivos tienen combinaciones de varias características, incluyendo la habilitación del conteo, el conteo ascendente/descendente, la carga en paralelo (se prestablece a cualquier conteo) y el borrado. Además, estos contadores se han diseñado para conectarse en cascada con facilidad y en forma síncrona para crear contadores más grandes. En esta sección exploraremos las técnicas que nos permitirán incluir esas características en un contador de HDL. Vamos a crear un contador que combinará más características de las que se encuentran ya sea en el 74161 o en el 74191. Utilizaremos este ejemplo para demostrar los métodos de diseño de un contador con capacidades que se adapten en forma específica a nuestras necesidades. Cuando utilizamos HDLs para crear diseños digitales, no estamos limitados a las características que suelen incluirse con cierto CI.

Vamos a revisar las especificaciones para nuestro ejemplo de un contador más complejo. El contador binario MOD-16 reciclable debe cambiar de estado en el flanco de subida de la entrada de reloj cuando el contador esté habilitado en un nivel ALTO. Una entrada de control de dirección hará que el contador cuente en forma ascendente cuando esté en BAJO, o que cuente en forma descendente cuando esté en ALTO. El contador tendrá una señal de limpiar asíncrona, activa en ALTO, para restablecer el contador de inmediato cuando la entrada de control se active. El contador puede cargarse en forma síncrona con un número en las terminales de entrada de datos cuando el control de carga está en ALTO. La prioridad de las funciones de control de entrada, de mayor a menor, serán borrar, cargar y contar. Por último, el contador también incluirá una salida activa en ALTO que detecte su estado terminal cuando esté habilitada la función de conteo. Recuerde que el estado terminal dependerá de la dirección de conteo. Como veremos, la correcta operación de estas características se determina en base a la forma en que escribimos el código de HDL, por lo que tendremos que poner mucha atención a los detalles.

AHDL

CONTADOR CON TODAS LAS CARACTERÍSTICAS EN AHDL

El código de la figura 7-42 implementa todas las características de las que hemos hablado. Es un contador de cuatro bits, pero puede expandirse en tamaño con facilidad. Analice las entradas y salidas de las líneas 3 y 4 para asegurarse de comprender lo que se supone debe hacer cada una. Si no lo entiende, vuelva a leer los párrafos anteriores de esta sección. La línea 7 define un registro de cuatro bits con flip-flops D, el cual servirá como contador. Debemos observar de nuevo aquí que este registro podría haberse llamado de igual forma que la variable de entrada (*q*). El código está escrito con distintos nombres para diferenciar los puertos (entradas y salidas) del circuito y los dispositivos que operan dentro del mismo. La entrada de reloj está conectada a todas las entradas *clk* de todos los flip-flops D en la línea 10. Todas las entradas borrar activas en BAJO (*clrn*) que van a la primitiva DFF están conectadas al complemento de la señal de entrada *borrar* en la línea 11. Esto hace que el flip-flop se borre de inmediato cuando la entrada *borrar* cambia a ALTO, debido a que las entradas *prn* y *clrn* de la primitiva DFF no dependen del reloj (es decir, son asíncronas).

Para que la función de carga pueda funcionar en forma síncrona deben controlarse las entradas *D* de los flip-flops, de manera que los datos de entrada (*dent*) estén presentes en las entradas *D* cuando la línea de carga esté en ALTO. De esta manera, cuando llegue el siguiente flanco activo del reloj los datos se cargarán en el contador. Esta acción debe ocurrir sin importar que el contador esté habilitado o no. En consecuencia, la primera decisión condicional (IF) en la línea 12 evalúa la entrada de carga. Recuerde del capítulo 4 que la estructura de decisión IF/ELSE da

```
1   SUBDESIGN fig7 42
2   (
3      reloj, borrar, cargar, habilcnt, abajo, dent[3..0]      :INPUT;
4      q[3..0], term ct :OUTPUT;     declara arreglo de 4 bits de salida
5   )
6   VARIABLE
7      conteo[3..0]    :DFF;         declara un registro de flip flops D
8
9   BEGIN
10     conteo[].clk = reloj;         conecta todos los relojes a la fuente de sincronía
11     conteo[].clrn= !borrar;       conecta para borrar asíncrona activa en ALTO
12     IF cargar THEN conteo[].d = dent[];    carga síncrona
13        ELSIF !habilcnt THEN conteo[].d = conteo[].q;    retiene el conteo
14        ELSIF !abajo THEN conteo[].d = conteo[].q + 1;    incremento
15        ELSE conteo[].d = conteo[].q   1;                 decremento
16     END IF;
17     IF ((conteo[].q == 0) & abajo # (conteo[].q == 15) & !abajo)& habilcnt
18     THEN     ct term = VCC;       señal de salida en cascada síncrona
19     ELSE ct term = GND;
20     END IF;
21     q[] = conteo[];               transfiere el contenido del registro a las salidas
22  END;
```

FIGURA 7-42 Contador con todas las características en AHDL.

precedencia a la primera condición que se encuentra verdadera ya que, una vez que encuentra una condición verdadera, no sigue evaluando las condiciones en las cláusulas ELSE subsiguientes. En este caso significa que si se activa la línea de carga, no importa si se habilita el conteo o si está tratando de contar hacia arriba o hacia abajo, sino que realizará una carga en paralelo en el siguiente flanco del reloj.

Suponiendo que la línea de carga no esté activa, se evalúa la cláusula ELSIF en la línea 13 para ver si el conteo está deshabilitado. En AHDL es muy importante tener en cuenta que la salida *Q* debe retroalimentarse a la entrada *D* para que en el siguiente flanco del reloj el registro retenga su valor anterior. Si no se inserta esta cláusula las entradas *D* quedarán en cero, con lo cual se restablecerá el contador. Si el contador está habilitado se evalúa la cláusula ELSIF de la línea 14 y *conteo* se incrementa (línea 14) o se decrementa (línea 15). Para resumir estas decisiones, primero se decide si es tiempo de cargar, después se decide si el conteo debe retenerse o cambiar, después se decide si se va a contar hacia arriba o hacia abajo.

La siguiente función descrita es la detección (o decodificación) del conteo terminal. Las líneas 17-20 deciden si se ha llegado al conteo terminal mientras se cuenta hacia arriba o hacia abajo. El operador de doble signo de igual (= =) es el símbolo que prueba la igualdad entre las expresiones de cada lado del operador. El estado del contador que sea el estado terminal depende de la dirección de conteo. Esto se determina mediante la aplicación de una operación AND entre la detección apropiada del estado terminal de 0 o 15 y la expresión correcta, *abajo* o *!abajo*. *ct_term* producirá como salida un nivel ALTO si se ha llegado al estado correcto, en caso contrario será BAJO. La línea 21 conectará la salida para *conteo* con las terminales de salida para el bloque SUBDESIGN.

Uno de los conceptos clave acerca del uso de HDLs es que, por lo general, es muy sencillo expandir el tamaño de un módulo lógico. Veamos ahora los cambios necesarios a este diseño de AHDL para incrementar el módulo del contador binario a 256. Como $2^8 = 256$, necesitaremos incrementar el número de bits a ocho. Sólo

se requieren cuatro modificaciones a la figura 7-42 para hacer este cambio en el módulo del contador:

# de línea	*Modificación*
3	dent[~~3~~ 7..0]
4	q[~~3~~ 7..0]
7	conteo[~~3~~ 7..0]
17	(conteo[ ].q == ~~15~~ 255)

VHDL

CONTADOR CON TODAS LAS CARACTERÍSTICAS EN VHDL

El código de la figura 7-43 implementa todas las características de las que hemos hablado. Es un contador de cuatro bits, pero puede expandirse en tamaño con facilidad. Analice las entradas y salidas de las líneas 2 a la 5 para asegurarse de comprender lo que se supone debe hacer cada una. Si no lo entiende, vuelva a leer los párrafos anteriores de esta sección. La instrucción PROCESS en la línea 10 es la clave para todos los circuitos sincronizados por reloj que se describen en VHDL,

```
1   ENTITY fig7 43 IS
2   PORT( reloj, borrar, cargar, habilcnt, descendente   :IN BIT;
3         dent          :IN INTEGER RANGE 0 TO 15;
4         q            :OUT INTEGER RANGE 0 TO 15;
5         ct term      :OUT BIT);
6   END fig7 43;
7
8   ARCHITECTURE a OF fig7 43 IS
9      BEGIN
10        PROCESS ( reloj, borrar, descendente)
11        VARIABLE conteo :INTEGER RANGE 0 to 15;      define una señal numérica
12           BEGIN
13              IF borrar = '1' THEN conteo := 0;       borrar asíncrona
14              ELSIF (reloj = '1' AND reloj'EVENT)  THEN     flanco ascendente?
15                 IF cargar = '1' THEN conteo := din;       carga en paralelo
16                 ELSIF habilcnt = '1' THEN              habilitado?
17                    IF descendente = '0' THEN conteo := conteo + 1;     incremento
18                    ELSE           conteo := conteo   1;     decremento
19                    END IF;
20                 END IF;
21              END IF;
22              IF (((conteo = 0) AND (descendente = '1')) OR
23                 ((conteo = 15) AND (descendente = '0'))) AND habilcnt = '1'
24                 THEN  ct term <= '1';
25              ELSE     ct term <= '0';
26              END IF;
27              q <= conteo;       transfiere el contenido del registro a las salidas
28           END PROCESS;
29        END a;
```

FIGURA 7-43 Contador con todas las características en VHDL.

pero también juega un papel importante en cuanto a determinar si el circuito responde en forma síncrona o asíncrona a sus entradas. Queremos que este circuito responda de inmediato a las transiciones en las entradas *reloj*, *borrar* y *abajo*. Con estas señales en la lista de sensibilidad nos aseguramos que se evalúe el código dentro del bloque PROCESS tan pronto como cualquiera de estas entradas cambie de estado. La variable *conteo* se define en la línea 11 como tipo INTEGER, de forma que pueda incrementarse y decrementarse con facilidad. Las variables se declaran dentro del bloque PROCESS y pueden utilizarse sólo dentro de este bloque.

La entrada *clear* recibe precedencia al evaluarla con la primera instrucción IF en la línea 13. Recuerde del capítulo 4 que la estructura de decisión IF/ELSE da precedencia a la primera condición que se encuentre verdadera, ya que no sigue evaluando las condiciones en las cláusulas ELSE subsiguientes. En este caso, si la señal *borrar* está activa las demás condiciones no importarán. La salida será cero. Para que la función *cargar* pueda operar en forma síncrona, debe evaluarse después de detectar el flanco del reloj. Éste se detecta en la línea 14 y el circuito comprueba de inmediato si *cargar* está activa. De ser así, el *conteo* se carga de *dent*, sin importar que esté habilitado el conteo o no. En consecuencia, la decisión condicional (IF) de la línea 15 evalúa la entrada *cargar*; sólo si está inactiva se evalúa la línea 16 para ver si el contador está habilitado. Si está habilitado, el *conteo* se incrementará o decrementará (líneas 17 y 18, respectivamente).

Lo siguiente es detectar el conteo terminal. Las líneas 22 a 25 deciden si se ha llegado al conteo terminal máximo o mínimo y excitan la salida para que cambie al nivel apropiado. Aquí, la estructura de toma de decisiones es muy importante ya que queremos evaluar esta situación, sin importar que el proceso de toma de decisiones haya sido invocado por *reloj*, *borrar* o *abajo*. Observe que esta decisión no es otra ramificación ELSE de las decisiones IF anteriores, sino que se evalúa para cada señal en la lista de sensibilidad *después* de que se produce el borrado o el conteo. Una vez que se realicen todas estas decisiones, *conteo* deberá tener el valor correcto en el registro y la línea 27 conectará el registro con las terminales de salida.

Uno de los conceptos clave acerca del uso de HDLs es que, por lo general, es muy sencillo expandir el tamaño de un módulo lógico. Veamos ahora los cambios necesarios a este diseño de VHDL para incrementar el módulo del contador binario a 256. Sólo se requieren cuatro modificaciones a la figura 7-42 para hacer este cambio en el módulo del contador:

# de línea	***Modificación***
3	RANGE 0 TO ~~15~~255
4	RANGE 0 TO ~~15~~255
11	RANGE 0 TO ~~15~~255
23	(conteo = ~~15~~255)

Simulación de un contador con todas las características

Para simular nuestro diseño del contador con todas las características requeriremos de cierta planeación para generar formas de onda de entrada apropiadas. Aunque tal vez no sea necesario simular a detalle todas las combinaciones de entrada imaginables, sí necesitamos probar la suficiente cantidad de condiciones de entrada posibles para convencernos de que funciona en forma apropiada. Esto es exactamente lo que debemos hacer también para probar nuestro diseño de prototipo en el laboratorio. El contador tiene cinco señales de entrada distintas (*reloj*, *borrar*, *cargar*, *habilcnt* y *dent*) y dos señales de salida distintas (*q* y *ct_term*) que debemos visualizar en nuestra simulación. Una de las señales de entrada y una de las señales de salida son en realidad de cuatro bits. Elegiremos una frecuencia de reloj conveniente, ya que no se ha especificado ninguna para nuestra simulación funcional del contador. Necesitaremos proveer los suficientes pulsos de reloj como para poder analizar varias condiciones de operación. La simulación deberá evaluar las funciones de habilitar y deshabilitar el contador, contar hacia arriba y hacia abajo, borrar

el contador, cargar un valor en el contador y contar a partir de ese valor, y detectar el estado de conteo terminal.

Hay ciertas cuestiones de simulación en general que debemos considerar al crear nuestras formas de onda de entrada. Como los PLDs de destino se restablecen al momento de encenderse, nuestra simulación empezará con el estado de salida inicial en 0000. Por lo tanto, sería mejor esperar hasta que el contador haya llegado a otro estado antes de aplicar una entrada para borrarlo, de manera que podamos ver un cambio en la salida. De igual forma, cargar el mismo valor que el estado SIGUIENTE del contador no nos convencerá en realidad de que la función *cargar* está operando en forma correcta. Si se cambian las señales de control de entrada al mismo tiempo que se produzca el flanco de reloj se pueden provocar algunos problemas con el tiempo de establecimiento y se obtendrían resultados cuestionables. Los controles asíncronos deben aplicarse en un tiempo distinto al del flanco de reloj apropiado para demostrar con claridad que la acción resultante del circuito es inmediata y no depende del reloj. En general, debemos usar el sentido común para crear nuestras formas de onda de entrada y considerar qué es lo que estamos tratando de verificar con la simulación. Ésta será valiosa en el proceso de diseño sólo si aplicamos las condiciones de entrada apropiadas y evaluamos los resultados de manera crítica.

En la figura 7-44 se muestran algunos resultados de la simulación para el contador con todas las características. La entrada *dent* de cuatro bits y la salida *q* de cuatro bits se muestran en hexadecimal. Al principio el contador está habilitado (*habilcnt* = 1) para contar hacia arriba (*abajo* = 0), y podemos ver que la salida se incrementa en 0, 1, 2, 3, 4, 5. En t_1 el contador responde en forma síncrona (es decir, en la PGT de *reloj*) al nivel ALTO que se aplica en la entrada *cargar*. El contador se prestablece con el valor de 8 de los datos en paralelo (*dent*). Esto también muestra que la carga tiene prioridad sobre el conteo, ya que ambas entradas están activas al mismo tiempo. Después de t_1, la entrada *cargar* está en BAJO otra vez y el contador sigue contando en forma ascendente a partir de 8. Una entrada en BAJO para *habilcnt* hace que el contador se detenga en el estado 9 durante un ciclo extra de reloj. El conteo se continúa cuando *habilcnt* está en ALTO otra vez hasta t_2, cuando el contador se borra en forma asíncrona. Observe el tiempo reducido para el estado de salida A debido a que el contador se borra de inmediato. En realidad tendríamos que hacer un acercamiento para ver que se muestra el estado A. También podemos ver que la función *borrar* tiene la prioridad más alta cuando los tres controles (*borrar*, *cargar* y *habilcnt*) están al mismo tiempo en ALTO. La secuencia de conteo ascendente continúa y se recicla a 0 después del estado F, para verificar que el contador sea un contador binario MOD-16. En t_3 el contador llega a su estado terminal F cuando cuenta hacia arriba, y *ct_term* produce un nivel ALTO. En t_4 el contador empieza a contar hacia abajo, ya que *abajo* se cambió a un nivel ALTO. De nuevo, *ct_term* produce un nivel ALTO ya que el contador se encuentra ahora en el estado 0, que es el estado terminal cuando se cuenta hacia abajo. Observe que a través de la acción de *ct_term*, el estado terminal para el contador depende de su dirección de conteo, la cual se controla mediante la entrada *abajo*. El conteo se retiene en el

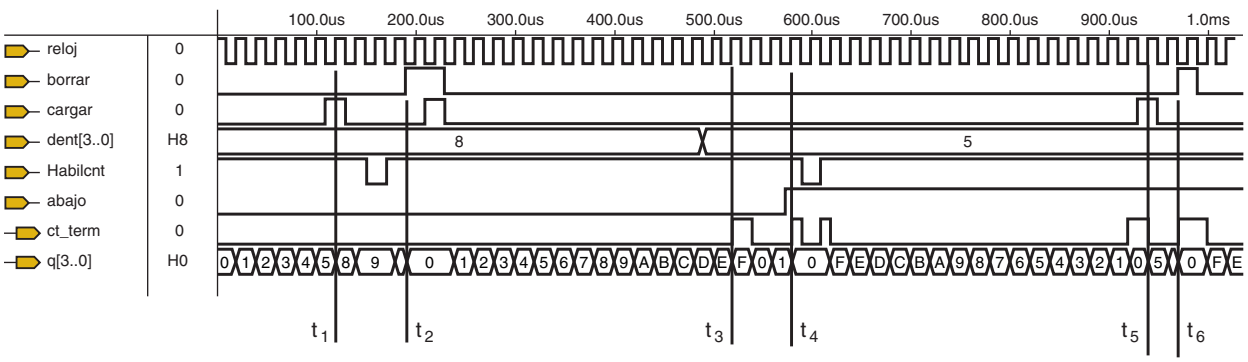


FIGURA 7-44 Resultados de la simulación para el diseño en HDL de un contador con todas las características.

estado 0 durante un periodo de reloj adicional cuando *habilcnt* cambia a BAJO. La salida *ct_term* también se deshabilita mientras que *habilcnt* = 0. La secuencia de conteo descendente continúa en forma correcta cuando *habilcnt* cambia de nuevo a ALTO. En t_5 el contador carga en forma síncrona el valor de datos en paralelo 5. En t_6 el contador se borra en forma asíncrona. De nuevo se verifica que la función de cargar o borrar tenga prioridad sobre la función de conteo descendente en t_5 y en t_6. ¿Verificamos que nuestro diseño opere en forma correcta, en comparación con las especificaciones? Hicimos un muy buen trabajo, pero hay un par de condiciones de prueba que también podrían agregarse para hacer la simulación más completa. ¿Se borró o se cargó el contador cuando *habilcnt* estaba en BAJO? Parece ser que no verificamos esos casos. Como puede ver, los diseños complejos pueden requerir mucho razonamiento para verificar su operación de una manera adecuada mediante la simulación o la prueba en el laboratorio. ¿Puede pensar en alguna otra prueba que deberíamos realizar?

PREGUNTAS DE REPASO

1. ¿Cuál es la diferencia entre la función borrar asíncrona y la función cargar síncrona?
2. ¿Cómo se crea una función borrar asíncrona en un HDL?
3. ¿Cómo se crea la prioridad de las funciones en una descripción en HDL de un contador?

7-13 CÓMO ALAMBRAR MÓDULOS DE HDL EN CONJUNTO

En las dos secciones anteriores vimos cómo implementar características comunes de un contador mediante HDL. También deberíamos investigar cómo podemos conectar estos circuitos tipo contador a otros módulos digitales para crear sistemas más grandes. El diseño de sistemas digitales extensos se facilita mucho si el sistema se subdivide en módulos más pequeños y manejables que después se interconectan. Ésta es la esencia del concepto del **diseño jerárquico**, del cual veremos sus beneficios mediante proyectos de ejemplo en el capítulo 10. Ahora veamos las técnicas básicas para alambrar módulos entre sí.

DECODIFICACIÓN DEL CONTADOR MOD-5 EN AHDL

En la sección 7-8 dimos un breve vistazo a la idea de decodificar un contador. Como recordará, un circuito decodificador detecta un estado de conteo mediante el patrón de bits único para ese estado. Ahora veamos cómo conectar un circuito decodificador al diseño del contador MOD-5 de la figura 7-35 (o 7-36). Cambiaremos el nombre del contador por SUBDESIGN mod5 para ser un poco más descriptivos en el diagrama de bloques para el circuito general que dibujaremos más adelante. Como el contador no produce los ocho estados posibles para un contador de tres bits, nuestro diseño del decodificador que se muestra en la figura 7-45 sólo decodificará los estados que se utilicen, de 000 hasta 100. Los tres bits de entrada (*c* = MSB) que se declaran en la línea 3 se conectarán más adelante a las salidas del contador MOD5. En la línea 4, las cinco salidas para el decodificador se llaman *estado0* a *estado4*. Una instrucción CASE (líneas 7 a 14) describe el comportamiento del decodificador mediante la comprobación de la combinación de entradas *c b a* para determinar cuál de las salidas del decodificador debe estar en ALTO. Cuando la entrada *c b a* sea 000 sólo la salida *estado0* estará en ALTO, o cuando *c b a* sea 001 sólo la salida *estado1* estará en ALTO, y así en lo sucesivo. Cualquier valor de entrada mayor de 100, que está cubierto por la instrucción OTHERS y en realidad no debe ocurrir en esta aplicación, producirá niveles BAJO en todas las salidas.

AHDL

```
1   SUBDESIGN decodifica5
2   (
3        c, b, a              :INPUT;
4        estado[0..4]         :OUTPUT;
5        )
6        BEGIN
7        CASE (c, b, a)  IS        -- decodifica el valor binario
8             WHEN B="000"    =>   estado[ ] = B"10000";
9             WHEN B="001"    =>   estado[ ] = B"01000";
10            WHEN B="010"    =>   estado[ ] = B"00100";
11            WHEN B= "011"   =>   estado[ ] = B"00010";
12            WHEN B= "100"   =>   estado[ ] = B"00001";
13            WHEN OTHERS     =>   estado[ ] = B"00000";
14       END CASE;
15  END;
```

FIGURA 7-45 Módulo decodificador del contador MOD-5 en AHDL.

Indicaremos al software de Altera que cree símbolos para nuestros dos archivos de diseño, mod5 y decodifica5. Esto nos permitirá dibujar un diagrama de bloques (vea la figura 7-46) para nuestro circuito completo, el cual consistirá de estos dos módulos, los puertos de entrada y salida, y el cableado entre ellos. Cada símbolo se etiqueta con su respectivo nombre mod5 o decodifica5 de SUBDESIGN. Observe que parte del cableado se dibuja con líneas más gruesas. Con el fin de representar un bus, el cual es una colección de líneas de señal. Las líneas más delgadas son señales individuales. En los símbolos creados por Altera se dibujarán de manera automática puertos para indicar si representan señales individuales o buses. Esto se determinará mediante las declaraciones de las señales en la sección SUBDESIGN. Los puertos con nombres de grupo se dibujarán como buses. Como el puerto de salida del contador es un bus pero los puertos de entrada del decodificador son señales individuales, será necesario dividir el bus en líneas de señal individuales para cablear los dos módulos entre sí. Cada vez que se divide un bus, hay que etiquetar tanto el nombre de la señal de grupo del bus como las señales individuales que se vayan a utilizar. Nuestro diagrama de bloques tiene un bus etiquetado como *q[2..0]* y las correspondientes señales individuales *q2*, *q1* y *q0*. Los resultados de la simulación para este circuito contador y decodificador se muestran en la figura 7-47.

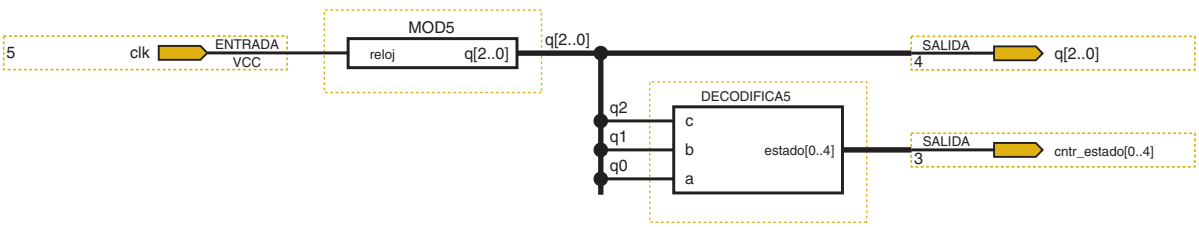


FIGURA 7-46 Diseño de diagrama de bloques para el circuito contador y decodificador MOD-5.

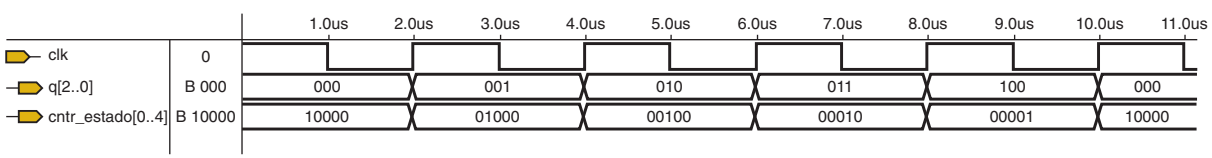


FIGURA 7-47 Simulación del circuito contador y decodificador MOD-5.

DECODIFICACIÓN DEL CONTADOR MOD-5 EN VHDL

En la sección 7-8 dimos un breve vistazo a la idea de decodificar un contador. Como podrá recordar, un circuito decodificador detecta un estado de conteo mediante el patrón de bits único para ese estado. Ahora veamos cómo conectar un circuito decodificador al diseño del contador MOD-5 de la figura 7-37. Cambiaremos el nombre del contador por ENTITY mod5 para facilitar la identificación del módulo en nuestro circuito en general. Como el contador no produce los ocho estados posibles para un contador de tres bits, nuestro diseño del decodificador que se muestra en la figura 7-48 sólo decodificará los estados que se utilicen, de 000 hasta 100. Los tres bits de entrada (*c* = MSB) que se declaran en la línea 3 se conectarán más adelante a las salidas del contador MOD5. En la línea 4, las cinco salidas para el decodificador se llaman *estado*, un vector de bits. En la línea 9 se declara una señal tipo vector de bits interna llamada *entrada*. Después la línea 11 combina los tres bis de entrada (*c b a*) para formar un vector de bits llamado *entrada*, el cual puede entonces evaluarse mediante la instrucción CASE en las líneas 14-21. Si cualquiera de los bits de entrada cambia de nivel lógico, se invocará el bloque PROCESS para determinar la salida resultante. Para describir el comportamiento del decodificador, la instrucción CASE comprueba la combinación de *entrada* (que representa *c b a*) para determinar cuál de las salidas del decodificador debe estar en ALTO. Cuando la *entrada* sea 000 sólo la salida *estado(0)* estará en ALTO; cuando *entrada* sea 001 sólo la salida *estado(1)* estará en ALTO; y así en lo sucesivo. Cualquier valor de entrada mayor de 100, que está cubierto por la instrucción OTHERS y en realidad no debe ocurrir en esta aplicación, producirá niveles BAJO en todas las salidas.

Como estamos usando el software de desarrollo de PLDs de Altera, podemos conectar los dos módulos en forma gráfica. Para ello necesitará instruir al software para que cree símbolos para nuestros dos archivos de diseño, mod5 y decodifica5. Esto nos permitirá dibujar un diagrama de bloques (vea la figura 7-46) para nuestro circuito completo, el cual consistirá de dos módulos, puertos de entrada y salida y el cableado entre ellos. Observe que parte del cableado se dibuja con líneas más gruesas. Para representar un bus, el cual es una colección de líneas de señal. Las líneas más delgadas son señales individuales. En los símbolos creados por Altera se dibujarán automáticamente puertos para indicar si representan señales individuales o buses. Esto se determinará mediante las declaraciones del tipo de datos

```
1   ENTITY  decodifica5  IS
2   PORT (
3             c, b, a         :IN BIT;
4             estado     :OUT BIT_VECTOR (0 TO 4)
5    );
6    END decodifica5;
7
8    ARCHITECTURE  a  OF  decodifica5  IS
9   SIGNAL     entrada    : BIT_VECTOR (2 DOWNTO 0);
10   BEGIN
11       entrada <= (c & b & a);        -- combina las entradas en un vector de bits
12       PROCESS (c, b, a)
13       BEGIN
14            CASE entrada IS
15                 WHEN "000"          =>        estado <= "10000";
16                 WHEN "001"          =>        estado <= "01000";
17                 WHEN "010"          =>        estado <= "00100";
18                 WHEN "011"          =>        estado <= "00010";
19                 WHEN "100"          =>        estado <= "00001";
20                 WHEN OTHERS     =>         estado <= "00000";
21            END CASE;
22       END PROCESS;
23  END a;
```

FIGURA 7-48 Módulo decodificador del contador MOD-5 en VHDL.

para cada puerto del bloque ENTITY. Los puertos BIT_VECTOR se dibujarán como buses y los puertos tipo BIT se dibujarán como líneas de señal individuales. Como el puerto de salida del contador es un bus pero los puertos de entrada del decodificador son señales individuales, será necesario dividir el bus en líneas de señal individuales para cablear los dos módulos entre sí. Cada vez que se divide un bus se debe etiquetar tanto el nombre de la señal de grupo del bus como las señales individuales que se vayan a utilizar. Nuestro diagrama de bloques tiene un bus etiquetado como *q[2..0]* y las señales individuales correspondientes *q2*, *q1* y *q0*. Los resultados de la simulación para este circuito contador y decodificador se muestran en la figura 7-47.

La técnica de VHDL estándar (y una alternativa con el software de Altera) para conectar módulos de diseño es utilizar VHDL para describir las conexiones entre los módulos en un archivo de texto. Se crean instancias de los módulos deseados en un archivo de diseño de mayor nivel mediante el uso de componentes (COMPONENT), en los cuales se declaran los puertos (PORT) del módulo. Las conexiones de cableado para cada instancia en donde se utiliza el módulo se listan en una instrucción PORT MAP. En la figura 7-49 se muestra un archivo de VHDL que conecta los módulos mod5 y decodifica5 entre sí. Aún y cuando *q* es un puerto de salida para nuestro archivo de diseño de nivel superior, se escribe como BUFFER en la línea 4 debido al hecho de que es necesario "leer" el arreglo de vectores de bit para una entrada al componente (COMPONENT) *decodifica5* en su mapa de puertos (PORT MAP, línea 25). VHDL no permite utilizar puertos de salida como entradas. La declaración del tipo de datos BUFFER proporciona un puerto que puede usarse tanto para entrada como para salida. El módulo mod5 se declara en las líneas 10 a 15 y el módulo decodifica5 se declara en las líneas 16 a 21. Las descripciones ENTITY/ARCHITECTURE de mod5 y decodifica5 pueden incluirse dentro del archivo de diseño de nivel superior, o en su defecto pueden guardarse en la misma carpeta que el archivo de nivel superior, como lo hicimos aquí. La instrucción PORT MAP para cada instancia de los módulos se lista en las líneas 23 a 25. La palabra a la izquierda de los dos puntos es una etiqueta única para cada instancia y el nombre del módulo va a la derecha, después las palabras clave PORT MAP y por último, entre paréntesis, van las asociaciones con nombre entre las señales y los puertos del diseño. El

```
 1   ENTITY  mod5decodifica1  IS
 2   PORT (
 3       clk                      :IN BIT;
 4       q                        :BUFFER BIT_VECTOR (2 DOWNTO 0);
 5       cntr_estado              :OUT BIT_VECTOR (0 TO 4)
 6       );
 7   END  mod5decodifica1
 8
 9  ARCHITECTURE  nivelsuperior OF mod5decodifica1  IS
10  COMPONENT mod5
11            PORT (
12            reloj               :IN BIT;
13            q                   :OUT BIT_VECTOR (2 DOWNTO 0)
14            );
15       END COMPONENT;
16       COMPONENT decodifica5
17       PORT (
18            c, b, a             :IN BIT;
19            estado              :OUT BIT_VECTOR (0 TO 4)
20            );
21  END COMPONENT;
22   BEGIN
23   contador:          mod5      PORT MAP (reloj => clk, q => q);
24  decodificador: decodifica5    PORT MAP
25            (c => q(2), b => q(1), a => q(0), estado => cntr_estado);
26   END  nivelsuperior;
```

FIGURA 7-49 Archivo en VHDL de mayor nivel para conectar mod5 y decodifica5 entre sí.

operador => indica qué puertos del módulo (a la izquierda) están conectadas con cuáles señales del sistema de mayor nivel (a la derecha). Este circuito produce los resultados de la simulación que se muestran en la figura 7-47.

Contador BCD MOD-100

Deseamos diseñar un contador BCD MOD-100 reciclable que tenga una entrada *borrar* asíncrona. La manera más sencilla de hacerlo es crear un módulo de contador BCD MOD-10 y conectar en cascada de manera síncrona dos de estos módulos entre sí, en un archivo de diseño de mayor nivel. Las entradas de reloj para los dos módulos MOD-10 estarán conectadas al reloj del sistema para lograr la conexión en cascada síncrona de los dos módulos de contador. Recuerde que hay beneficios considerables en cuanto al uso del diseño de contadores síncronos en vez de las técnicas asíncronas de sincronización por reloj. Además, si no empleáramos la sincronización por reloj en forma síncrona, la señal borrar síncrona no funcionaría en forma apropiada. Aún y cuando las especificaciones de diseño no requieren una señal de habilitación de conteo o la detección del conteo terminal para el contador MOD-100, será necesario incluir estas características en nuestro diseño. Para poder conectar en cascada en forma síncrona dos contadores, se necesitarán las características de habilitación y decodificación. La entrada de habilitación de conteo hace que el contador ignore los flancos del reloj hasta que esté habilitado. La salida de conteo terminal indica que la secuencia de conteo ha llegado a su límite y volverá a empezar en el siguiente pulso de reloj. Para conectar en cascada las etapas de contadores en forma síncrona, la salida de conteo terminal se conecta a la entrada de habilitación de la siguiente etapa de mayor orden. Al utilizar la entrada de habilitación de conteo para controlar también la decodificación del conteo terminal, podemos usar nuestro módulo MOD-10 para crear contadores BCD aún más grandes.

CONEXIÓN EN CASCADA DE CONTADORES BCD EN AHDL

El bloque SUBDESIGN de nuestro contador BCD MOD-10 se muestra en la figura 7-50. El estado terminal para un contador BCD es 9. Las líneas 10 a 13 detectarán

```
1  SUBDESIGN  mod10
2  (
3       reloj, habilita, borrar            :INPUT;
4       contador[3..0], tc                 :OUTPUT;
5   )
6  VARIABLE
7       contador[3..0]                     :DFF;
8   BEGIN
9       contador[ ].clk = reloj;
10      IF contador[ ].q == 9 & habilita   == VCC THEN
11                tc = VCC;                -- detecta el conteo terminal
12      ELSE      tc = GND;
13      END IF;
14      IF borrar THEN
15           contador[ ].d = B"0000";    -- borrar síncrona
16      ELSIF  enable  THEN              -- borrar tiene prioridad
17           IF contador[ ].q == 9 THEN -- comprueba el último estado
18                contador[ ].d = B"0000";
19           ELSE
20                contador[ ].d = contador[ ].q + 1; -- incremento
21           END IF;
22      ELSE                             -- retiene el conteo cuando está deshabilitado
23           contador[ ].d = contador[ ].q;
24      END IF;
25 END;
```

FIGURA 7-50 Contador BCD MOD-10 en AHDL.

AHDL

este estado terminal sólo cuando el contador esté habilitado con un nivel ALTO. Si se aplica un AND al control *habilita* en la función de decodificación se permitirá que dos o más módulos de contador se conecten en cascada de manera síncrona si es necesario, con lo cual nuestro diseño mod10 será más versátil. La función *borrar* operará en forma síncrona en AHDL si se incluye en la instrucción IF, como se muestra en las líneas 14 a 15. Si *borrar* está inactiva, el contador comprobará mediante una instrucción IF anidada en las líneas 17 a 21 si se ha llegado al último estado 9. Después del estado 9 el contador se recicla en forma síncrona a 0. En caso contrario, el contador se incrementará. Si el contador se deshabilita, las líneas 22 a 23 retendrán el valor de conteo actual mediante el proceso de alimentar la salida actual de regreso a la entrada del contador. Esta acción de retención será necesaria en el contador MOD-100 en cascada para que el dígito de los 10s retenga su estado actual mientras que el dígito de los 1s progrese a través de su secuencia de conteo. Una estrategia de diseño apropiada para nosotros sería simular este módulo para determinar si funciona en forma correcta, antes de usarlo en una aplicación de circuito más compleja. De los resultados de la simulación para mod10, que se proporcionan en la figura 7-51, podemos ver que la secuencia de conteo es correcta, la entrada *borrar* es síncrona y tiene prioridad, y *habilita* controla tanto la función de conteo como la salida de decodificación *tc*.

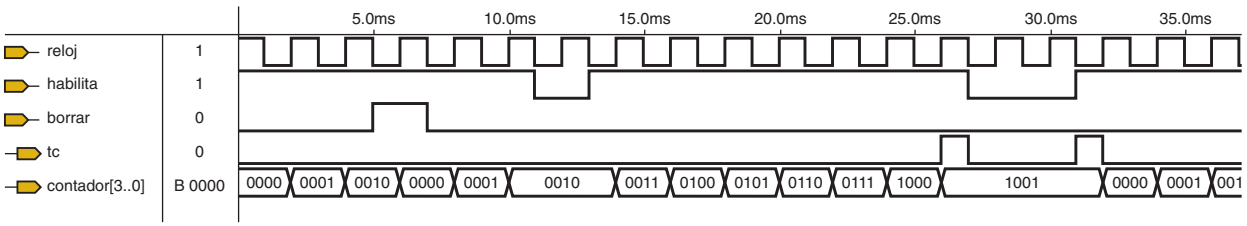


FIGURA 7-51 Resultados de la simulación del contador MOD-10.

Después de crear un símbolo predeterminado para nuestro módulo de contador mod10, podemos ahora dibujar el diagrama de bloques para la aplicación del contador BCD MOD-100. También se han agregado los puertos de entrada, los puertos de salida y el cableado para crear el diseño de la figura 7-52. Observe que las salidas del contador que representan los dígitos de los 1s y los 10s se dibujan como buses. Los módulos mod10 están sincronizados por reloj en forma síncrona. Están conectados en cascada mediante el uso de la salida de conteo terminal del dígito de los 1s para controlar la entrada de habilitación en el dígito de los 10s. El puerto de entrada *en* controla la habilitación/deshabilitación de todo el circuito completo del contador MOD-100. El diseño del contador BCD puede expandirse con facilidad mediante una etapa mod10 adicional, conectando la salida *tc* con la siguiente entrada *habilita* para cada dígito que se necesite. En la figura 7-53 puede verse una muestra de los resultados de la simulación. Ésta nos muestra que el contador MOD-100 tiene una secuencia de conteo BCD correcta y que puede borrarse en forma síncrona.

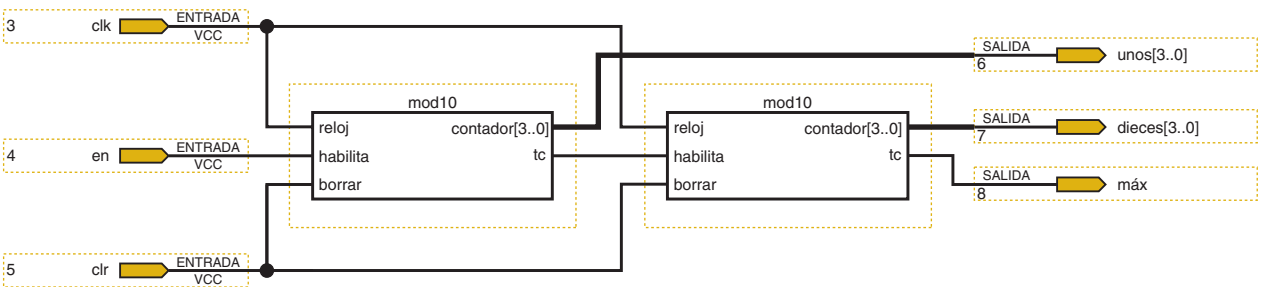


FIGURA 7-52 Diseño de diagrama de bloques para un contador BCD MOD-100.

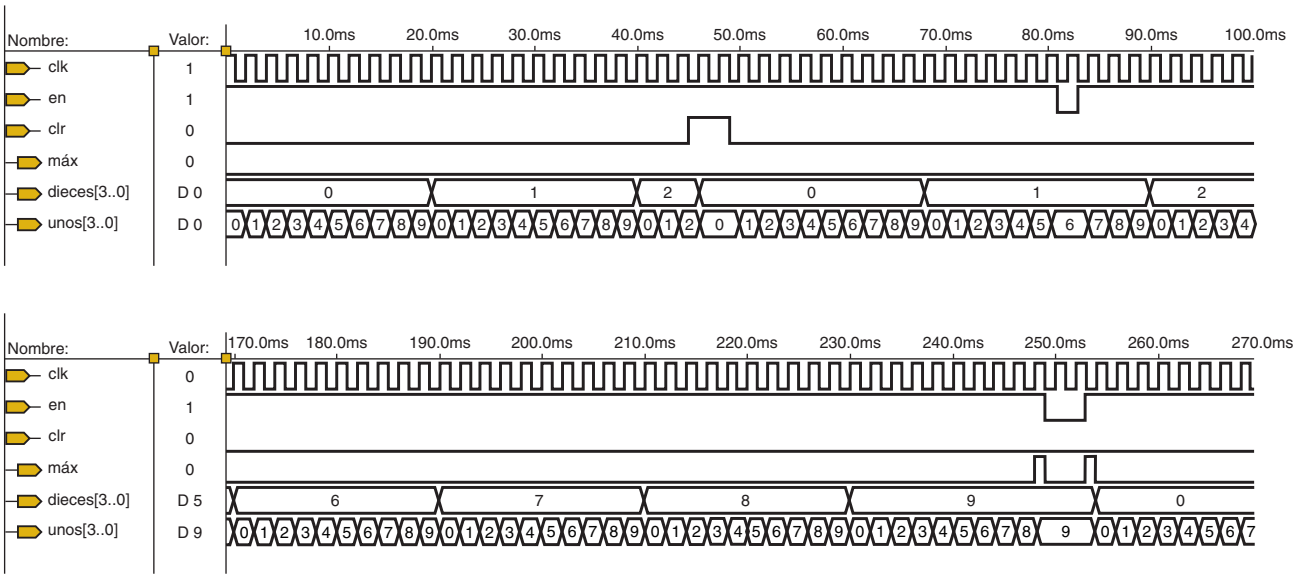


FIGURA 7-53 Resultados de la simulación para el diseño del contador BCD MOD-100.

CONEXIÓN EN CASCADA DE CONTADORES BCD EN AHDL

Los bloques ENTITY y ARCHITECTURE para nuestro contador BCD MOD-10 se muestran en las líneas 26 a 51 de la figura 7-54. El estado terminal para un contador BCD es 9. Las líneas 38 a 40 detectarán este estado terminal sólo cuando el contador se habilite con un nivel ALTO. Si se aplica un AND al control *habilita* en la función de decodificación se permitirá que se conecten en cascada más de dos módulos de contador en forma síncrona si es necesario; además, nuestro diseño mod10 será más versátil. La función *borrar* será síncrona en VHDL si se coloca en la instrucción IF anidada (línea 42) una vez que se detecte el flanco de reloj en la línea 41. Si *borrar* está inactiva, a continuación revisamos si el contador está habilitado (línea 43). Si *habilita* está en ALTO, el contador comprueba mediante otra instrucción IF anidada en las líneas 44 a 46 para ver si se ha llegado al último estado 9. Después del estado 9 el contador se recicla en forma síncrona a 0. En caso contrario se incrementará el conteo. Si el contador está deshabilitado, VHDL retendrá en forma automática el valor de conteo actual. Esta acción de retención será necesaria en el contador MOD-100 en cascada para que el dígito de los 10s retenga su estado actual mientras que el dígito de los 1s progresa a través de su secuencia de conteo. Una estrategia de diseño apropiada para nosotros sería que simuláramos este módulo como un bloque ENTITY separado para determinar si funciona en forma correcta, antes de usarlo en una aplicación de circuito más compleja. Los resultados de la simulación para el bloque ENTITY mod10, que se muestran en la figura 7-51, muestran que la secuencia de conteo es correcta, que la entrada *borrar* es síncrona y tiene prioridad, y que *habilita* controla tanto la función de conteo como la entrada de decodificación.

Tenemos dos opciones para implementar el contador MOD-10. Una técnica es representar el diseño en forma gráfica en un diagrama de bloques, como puede verse en la figura 7-52. También hemos agregado los módulos del contador mod10, los puertos de entrada, de salida y el cableado para crear el contador MOD-100. Observe que las salidas del contador que representan los dígitos de los 1s y los 10s se dibujan como buses. Los módulos mod10 se sincronizan por reloj en forma síncrona. Están conectados en cascada mediante el uso de la salida de conteo terminal del dígito de los 1s para controlar la entrada de habilitación en el dígito de los 10s. El puerto de entrada *en* controla la habilitación/deshabilitación de todo el circuito completo del contador MOD-100. El diseño del contador BCD puede expandirse con facilidad mediante una etapa mod10 adicional, conectando la salida *tc* con la

```
 1   ENTITY mod100 IS
 2   PORT (
 3               clk, en, clr                    :IN BIT;
 4               unos                            :OUT INTEGER RANGE 0 TO 15;
 5               dieces                          :OUT INTEGER RANGE 0 TO 15;
 6               max                             :OUT BIT
 7    );
 8   END mod100;
 9   ARCHITECTURE nivelsuperior OF mod100 IS
10   COMPONENT mod10
11               PORT (
12               reloj, habilita, borrar         :IN BIT;
13               q                               :OUT INTEGER RANGE 0 TO 15;
14               tc                              :OUT BIT;
15               );
16   END COMPONENT;
17    SIGNAL rco                                 :BIT;
18    BEGIN
19    digito1:   mod10  PORT MAP (reloj => clk, habilita => en,
20                                borrar => clr, q => unos, tc => rco);
21   digito2:    mod10  PORT MAP (reloj => clk, habilita => rco,
22                                borrar => clr, q => dieces, tc => max);
23   END nivelsuperior;
24
25
26   ENTITY mod10 IS
27   PORT (
28               reloj, habilita, borrar         :IN BIT;
29               q                               :OUT INTEGER RANGE 0 TO 15;
30               tc                              :OUT BIT
31   );
32    END mod10;
33   ARCHITECTURE bloqinf OF mod10 IS
34   BEGIN
35               PROCESS (reloj, habilita)
36                     VARIABLE  contador        :INTEGER RANGE 0 TO 15;
37               BEGIN
38                     IF ((contador = 9) AND (habilita = '1')) THEN tc <= '1';
39                     ELSE tc <= '0';
40                     END IF;
41                     IF (reloj'EVENTO AND reloj = '1')  THEN
42                           IF (borrar = '1') THEN contador := 0;
43                           ELSIF (habilita = '1') THEN
44                                 IF (contador = 9) THEN contador := 0;
45                                 ELSE        contador := contador + 1;
46                                 END IF;
47                           END IF;
48                     END IF;
49                     q <= contador;
50               END PROCESS;
51   END bloqinf;
```

FIGURA 7-54 Contador BCD MOD-100 en VHDL.

siguiente entrada *habilita* para cada dígito que se necesite. En la figura 7-53 puede verse una muestra de los resultados de la simulación.

La simulación muestra que el contador MOD-100 tiene una secuencia de conteo BCD correcta y puede borrarse en forma síncrona.

La segunda técnica para crear el contador MOD-100 es hacer las conexiones necesarias entre los módulos de diseño, describiendo la estructura del circuito con VHDL. La figura 7-54 muestra el listado para el archivo de diseño de este sistema. La descripción ENTITY/ARCHITECTURE para el subbloque mod10 está conteni-

da dentro del archivo de diseño mod100 general (pero podría estar en un archivo separado dentro de la carpeta de este proyecto). El archivo de diseño mod100 sería el nivel superior para el diseño jerárquico de este sistema. Contiene subbloques de nivel inferior, los cuales son en realidad dos copias del contador mod10 de nivel inferior. El bloque COMPONENT de mod10 se declara en este archivo de diseño de nivel superior (líneas 10 a 16). Las conexiones del cableado para cada instancia en la que se utiliza el módulo se listan en una instrucción PORT MAP. Como necesitamos dos instancias de mod10, hay una instrucción PORT MAP para cada instancia (líneas 19 a 20 y 21 a 22). Cada instancia debe tener una etiqueta única (*digito1* o *digito2*) para diferenciarlos una de la otra. Las instrucciones PORT MAP contienen asociaciones con nombre entre los puertos del módulo de nivel inferior, las cuales se dan a la izquierda, y las señales de nivel superior a las cuales están conectadas, se dan ala derecha. Este circuito produce los mismos resultados de simulación que se muestran en la figura 7-53.

PREGUNTAS DE REPASO

1. Describa cómo conectar módulos de HDL entre sí para crear un sistema digital.
2. ¿Qué es un bus y cómo se representa en un archivo de diseño de diagrama de bloques gráfico en Altera?
3. ¿Qué características de contador deben incluirse para conectar en cascada módulos de contador entre sí, en forma síncrona?

7-14 MÁQUINA DE ESTADOS

El término **máquina de estados** se refiere a un circuito que progresa en secuencia a través de un conjunto de estados predeterminados, controlado por un reloj y por otras señales de entrada. Por lo tanto, los circuitos tipo contador que hemos estudiado hasta ahora en el capítulo 7 son máquinas de estado. En general, utilizamos el término *contador* para los circuitos secuenciales que tienen una secuencia de conteo numérica regular. Pueden contar hacia arriba o hacia abajo, pueden tener módulos completos de 2^N o pueden tener un módulo $<2^N$, o pueden reciclarse o detenerse en forma automática en cierto estado predeterminado. Como su nombre lo indica, un contador se utiliza para contar cosas. Las cosas que se cuentan en realidad se conocen como pulsos de reloj, pero éstos pueden representar muchos tipos de eventos. Los pulsos pueden ser los ciclos de una señal para la división de frecuencia o pueden ser segundos, minutos y horas de un día para un reloj digital. Pueden indicar que un elemento se ha desplazado por el transportador en una fábrica o que un automóvil ha pasado por cierto punto en la autopista.

El término *máquina de estados* se utiliza con más frecuencia para describir otros tipos de circuitos secuenciales. Pueden tener un patrón de conteo irregular tal como nuestro circuito de control de motores a pasos de la sección 7-10. El objetivo para ese diseño fue controlar un motor a pasos, de manera que pudiera girar en pasos angulares precisos. El circuito de control tenía que producir la secuencia específica requerida de estados para ese movimiento, en lugar de contar en forma numérica. También existen muchas aplicaciones en las que no nos importa el valor binario específico para cada estado, ya que utilizaremos la lógica de decodificación apropiada para identificar estados específicos de interés y para generar las señales de salida deseadas. La distinción general entre los dos términos es que un *contador* se utiliza, por lo general, para contar eventos, mientras que una *máquina de estados* se utiliza por lo común para controlar eventos. El término descriptivo correcto depende de la manera en que deseamos utilizar el circuito secuencial.

El diagrama de bloques que se muestra en la figura 7-55 puede representar una máquina de estados o un contador. En la sección 7-10 vimos que el proceso de diseño clásico de un circuito secuencial era averiguar cuántos flip-flops se requerían y después determinar el circuito combinacional necesario para producir la secuencia

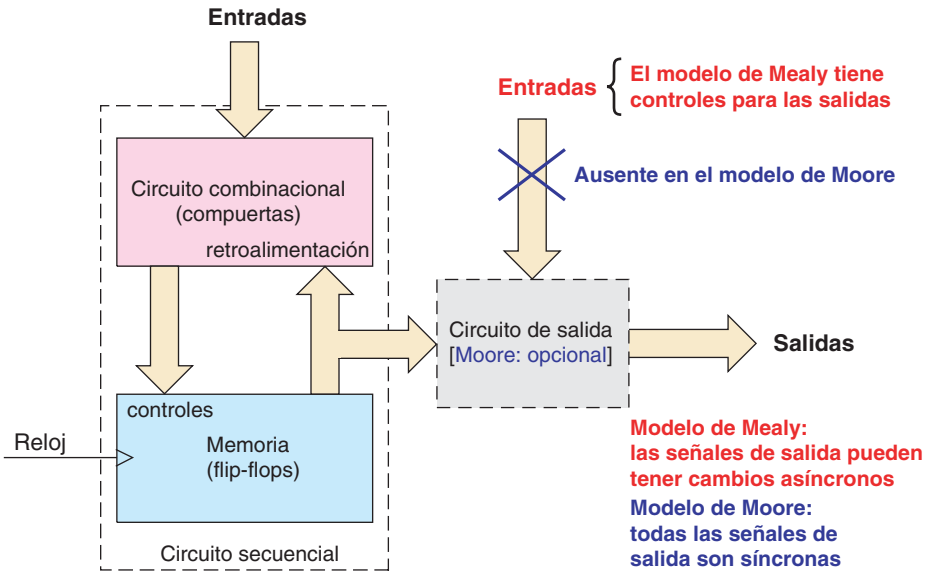


FIGURA 7-55
Diagrama de bloques para contadores y máquinas de estados.

deseada. La salida producida por un contador o una máquina de estados puede venir en forma directa de las salidas de los flip-flops o tal vez se requiera de algún circuito de compuertas, como se indica en el diagrama de bloques. Las dos variaciones se describen ya sea como un **modelo de Mealy** para un circuito secuencial o como un **modelo de Moore**. En el modelo de Mealy las señales de salida también se controlan mediante señales de entrada adicionales, mientras que el modelo de Moore no tiene controles externos para las señales de salida generadas. La salida de Moore es función sólo del estado actual del flip-flop. Un ejemplo de diseño tipo Moore sería el circuito MOD-5 decodificado de la sección 7-13. Por otro lado, el diseño del contador BCD en la misma sección sería un diseño tipo Mealy, debido a la entrada externa (*habilita*) que controla la salida de decodificación del estado terminal (*tc*). Una consecuencia considerable de esta sutil variación del diseño es que las salidas de un circuito tipo Moore serán por completo síncronas con el reloj del circuito, mientras que las salidas producidas por un circuito tipo Mealy pueden cambiar en forma asíncrona. En nuestro diseño MOD-10, la entrada de habilitación no está sincronizada con el reloj del sistema.

Desde luego que los HDLs pueden hacer que las máquinas de estado sean fáciles e intuitivas de describir. Como un ejemplo simplificado en extremo con el que todos pueden relacionarse, la siguiente descripción de hardware trata con cuatro estados, a través de los cuales podría progresar una máquina lavadora ordinaria. Aunque una máquina lavadora real es más compleja que este ejemplo, servirá para demostrar las técnicas. Esta máquina lavadora está inactiva hasta que se oprime el botón de arranque, después se llena con agua hasta que la tina está llena y luego opera el agitador hasta que un temporizador expira; por último gira la tina hasta que el agua se avienta hacia fuera y después de eso regresa a la inactividad. El punto de este ejemplo se enfoca en el uso de un conjunto de estados con nombre para los cuales no hay valores binarios definidos. El nombre de la variable contador es *lavar*, la cual puede estar en cualquiera de los siguientes estados con nombre: *inactiva*, *llenar*, *agitar* o *girar*.

AHDL

MÁQUINA DE ESTADOS SIMPLE EN AHDL

El código de AHDL en la figura 7-56 muestra la sintaxis para declarar un contador con estados con nombre en las líneas 6 y 7. El nombre de este contador es *ciclo*. La palabra clave **MACHINE** se utiliza en AHDL para definir a *ciclo* como una máquina de estado. El número de bits necesarios para que este contador produzca los estados con nombre se determinará mediante el compilador. Observe que en la línea 7

FIGURA 7-56 Ejemplo de una máquina de estados mediante el uso de AHDL.

```
1   SUBDESIGN fig7 56
2   (  reloj, arranque, llena, tiempoagotado, secar  :INPUT;
3      valvula agua, modo ag, modo gi                :OUTPUT;
4   )
5   VARIABLE
6   ciclo:  MACHINE
7           WITH STATES (inactiva, llenar, agitar, girar);
8   BEGIN
9   ciclo.clk = reloj;
10
11     CASE ciclo IS
12        WHEN inactiva =>IF arranque THEN ciclo = llenar;
13                        ELSE     ciclo = inactiva;
14                        END IF;
15        WHEN llenar =>IF llena THEN ciclo = agitar;
16                        ELSE     ciclo = llenar;
17                        END IF;
18        WHEN agitar => IF tiempoagotado THEN ciclo = girar;
19                        ELSE     ciclo = agitar;
20                        END IF;
21        WHEN girar => IF secar THEN ciclo = inactiva;
22                        ELSE     ciclo = girar;
23                        END IF;
24        WHEN OTHERS => ciclo = inactiva;
25     END CASE;
26
27     TABLE
28        ciclo     => valvula agua,   modo ag, modo gi;
29        inactiva  => GND,            GND,     GND;
30        llenar    => VCC,            GND,     GND;
31        agitar    => GND,            VCC,     GND;
32        girar     => GND,            GND,     VCC;
33     END TABLE;
34  END;
```

los estados tienen nombre, pero el valor binario para cada estado también se deja para que el compilador lo determine. El diseñador no tiene que preocuparse por este nivel de detalle. La estructura CASE en las líneas 11 a 25 y la lógica de decodificación que controla las salidas (líneas 27 a 33) hace referencia a los estados por su nombre. Esto hace que la descripción sea fácil de leer y permite al compilador más libertad para minimizar los circuitos. Si el diseño requiere que la máquina de estado también se conecte a un puerto de salida, entonces la línea 6 se puede cambiar por:

```
ciclo: MACHINE OF BITS (st [1..0])
```

y el puerto de salida *st[1..0]* puede agregarse a la sección SUBDESIGN. Una segunda opción de máquina de estado disponible es la habilidad para que el diseñador defina un valor binario para cada estado. En este ejemplo, esto puede lograrse si se cambia la línea 7 por:

```
WITH STATES (inactiva = B"00", llenar = B"01", agitar = B"11", girar
= B"10");
```

MÁQUINA DE ESTADOS SIMPLE EN VHDL

El código de VHDL en la figura 7-57 muestra la sintaxis para declarar un contador con estados con nombre. En la línea 6 se declara un objeto llamado *maquina_estados*. Observe la palabra clave TYPE. A esto se le conoce como un **tipo enumerado** en VHDL, en el cual el diseñador lista por nombres simbólicos todos los posibles valores que una señal, variable o puerto que se declara de ese tipo tiene permitido tener. Observe también que en la línea 6 los estados tienen nombre, pero el valor binario para cada estado se deja para que el compilador lo determine. El diseñador no necesita preocuparse por este nivel de detalle. La estructura CASE en las líneas 12 a 29 y la lógica de decodificación que controla las salidas (líneas 31 a 36) hacen referencia a los estados por su nombre. Esto hace que la descripción sea fácil de leer y permite al compilador más libertad para minimizar los circuitos.

Al utilizar el simulador para verificar nuestros diseños de HDL se producen los resultados que muestra la figura 7-58. El simulador de Altera permite también simular nodos intermedios en nuestros módulos de diseño. La máquina de estado "oculta" llamada *ciclo* se ha incluido en la simulación para poder confirmar que opere en forma correcta. Observe que los resultados para *ciclo* se proporcionan dos veces, ya que se mostrará en forma distinta para los dos HDLS. En realidad el

```
 1  ENTITY  fig7_57  IS
 2  PORT( reloj, arranque, llena, tiempoagotado, secar          :IN BIT;
 3        valvula_agua, modo_ag, modo_gi                       :OUT BIT);
 4  END  fig7_57;
 5  ARCHITECTURE  vhdl  OF  fig7_57 IS
 6  TYPE  maquina_estados  IS  (inactiva, llenar, agitar, girar);
 7  BEGIN
 8        PROCESS (reloj)
 9        VARIABLE    ciclo             :maquina_estado;
10        BEGIN
11        IF (reloj'EVENT  AND  reloj = '1')    THEN
12          CASE ciclo IS
13              WHEN  inactiva  =>
14                    IF arranque = '1' THEN        ciclo := llenar;
15                    ELSE                          ciclo := inactiva;
16                    END IF;
17              WHEN llenar  =>
18                    IF llena = '1'  THEN          ciclo := agitar;
19                    ELSE                          ciclo := llenar;
20                    END IF;
21              WHEN agitar  =>
22                    IF tiempoagotado = '1' THEN   ciclo := girar;
23                    ELSE                          ciclo := agitar;
24                    END IF;
25              WHEN girar =>
26                    IF secar = '1' THEN           ciclo := inactiva;
27                    ELSE                          ciclo := girar;
28                    END IF;
29          END CASE;
30        END IF;
31        CASE ciclo IS
32    WHEN inactiva    => valvula_agua  <= '0';  modo_ag <= '0';  modo_gi <= '0';
33          WHEN llenar=> valvula_agua  <= '1';  modo_ag <= '0';  modo_gi <= '0';
34          WHEN agitar=> valvula_agua  <= '0';  modo_ag <= '1';  modo_gi <= '0';
35          WHEN girar => valvula_agua  <= '0';  modo_ag <= '0';  modo_gi <= '1';
36        END CASE;
37        END PROCESS;
38  END vhdl;
```

FIGURA 7-57 Ejemplo de máquina de estados mediante el uso de VHDL.

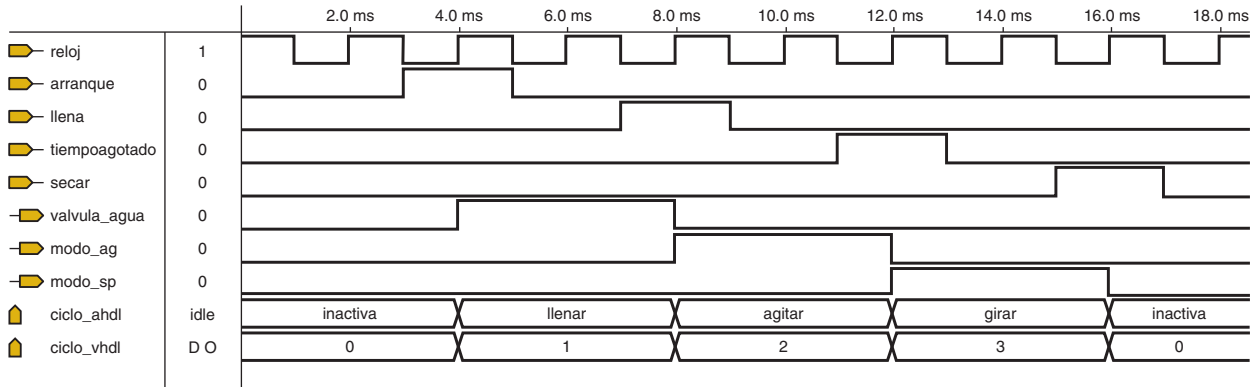


FIGURA 7-58 Simulación del ejemplo de diseño en HDL de la máquina lavadora para una máquina de estados.

simulador no puede mostrar las simulaciones para AHDL y VHDL al mismo tiempo. La información del segundo nodo oculto tan sólo se copió y se pegó para obtener aquí una figura compuesta. En AHDL se muestran los nombres de la máquina de estados, mientras que en VHDL se muestran en su lugar los valores asignados por el compilador para los nombres de los estados enumerados.

Máquina de estados del controlador de un semáforo

Vamos a estudiar el diseño de una máquina de estados de un circuito que es más complicado: un controlador de un semáforo. En la figura 7-59 se muestra el diagrama de bloques. Nuestro controlador está diseñado para controlar el flujo de tráfico en la intersección de un camino principal con un camino lateral menos fluido. El tráfico fluirá sin interrupción en el camino principal con una luz verde, hasta que se detecte un automóvil en el camino lateral (lo cual se indica mediante la entrada etiquetada como *auto*). Después de un lapso que se establece mediante la entrada binaria de cinco bits etiquetada como *tprinverde*, la luz del camino principal cambiará a ámbar. El *tprinverde* asegura que el camino principal reciba una luz verde por al menos ese lapso en

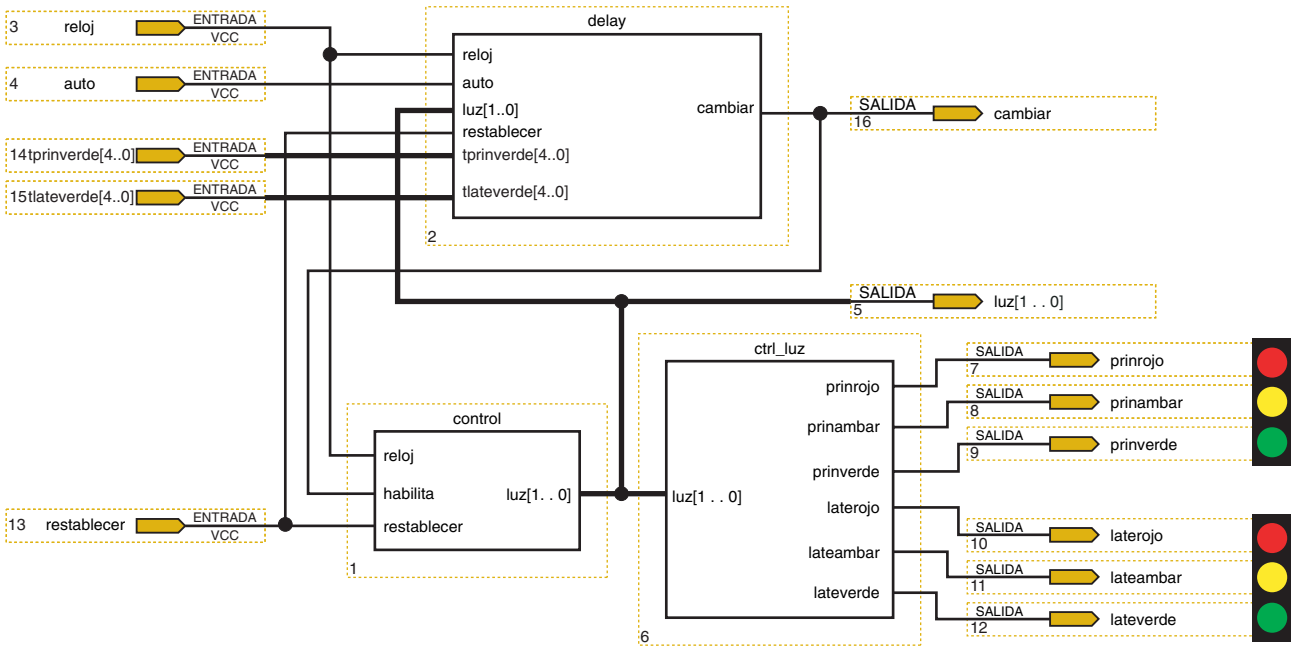


FIGURA 7-59 Controlador de semáforo.

cada ciclo de las luces. La luz ámbar durará una constante de tiempo, el cual se establece en el diseño de HDL y después realizará la transición a rojo. Cuando la luz del camino principal cambie a rojo, la luz del camino lateral cambiará a verde. La luz del camino lateral estará en verde durante un tiempo que se establece mediante la entrada binaria de cinco bits llamada *tlateverde*. De nuevo, la luz ámbar durará la misma constante de tiempo y después el camino lateral regresará a una luz roja y el camino principal tendrá luz verde de nuevo. El módulo generador de lapsos controlará los periodos de tiempo para cada una de las luces. Los lapsos actuales serán iguales al periodo del reloj del sistema multiplicado por el factor de lapso. El módulo de control determina el estado del controlador de tráfico. Hay cuatro combinaciones de luces: principal-verde/lateral-rojo, principal-ámbar/lateral-rojo, principal-rojo/lateral-verde y principal-rojo/lateral-ámbar; por lo tanto, el control requerirá cuatro estados. Los estados del semáforo se traducen en los patrones de encendido-apagado apropiados para cada uno de los seis pares de luces, mediante el módulo ctrl_luz. Las salidas etiquetadas como *cambiar* y *luz* se proporcionan para fines de diagnóstico. La pata de *Restablecer* (*Reset*) se utiliza para inicializar cada uno de los dos circuitos secuenciales.

AHDL

CONTROLADOR DE SEMÁFORO EN AHDL

Los tres módulos de diseño para nuestro controlador de semáforo en AHDL se muestran juntos en la figura 7-60. En realidad son tres archivos de diseño separados que se interconectan con el diseño del diagrama de bloques que se muestra en la figura 7-59. El módulo generador de lapsos (líneas1 a 23) es en esencia un contador descendente oculto (línea 20) llamado *mach*, el cual espera en cero cuando el camino principal tiene luz verde (*luz* = 0), hasta que se dispara mediante el sensor de autos (línea 13) para cargar el factor de lapso o retraso *tprinverde* – 1 en la línea 14. Como el contador se decrementa hasta cero, se resta uno a cada retraso para hacer que el módulo del contador de retraso sea igual al valor del retraso. Por ejemplo, si deseamos tener un factor de retraso de 25, el contador debe contar en forma descendente desde 24 hasta 0. La longitud actual de tiempo representada por los factores de retraso depende de la frecuencia del reloj. Con una frecuencia de reloj de 1-Hz el periodo sería de 1s, y los factores de retraso estarían en segundos. La línea 22 define una señal de salida llamada *cambio*, la cual detecta cuando *mach* es igual a uno. *Cambio* estará en ALTO para indicar que la condición de prueba es verdadera, lo cual a su vez habilitará la máquina de estados en el módulo de control para que avance a su siguiente estado (*luz* = 1) cuando se sincronice con el reloj, para indicar una luz ámbar en el camino principal. A medida que el contador de retraso *mach* cuente hacia abajo y llegue a cero, la instrucción CASE determinará que *luz* tiene un nuevo valor y se cargará el factor de retraso de tiempo fijo de 5 para una luz ámbar (en realidad se carga uno menos que 5, como dijimos antes) en *mach* (línea 16), en el siguiente pulso de reloj. El conteo descendente continuará a partir de este nuevo tiempo de retraso y *cambio* habilitará de nuevo el módulo de control para que avance a su siguiente estado (*luz* = 2) cuando *mach* sea igual a 1; el resultado será una luz verde para el camino lateral. Cuando *mach* llegue de nuevo a cero se cargará en el contador descendente (línea 17) el retraso de tiempo (*tlateverde – 1*) para una luz verde en el camino lateral. Cuando *cambio* se vuelva a activar, *luz* avanzará al estado 3 para una luz ámbar en el camino lateral. *Mach* se reciclará al valor 4 (5-1) en la línea 18 para el retraso de tiempo fijo para una luz ámbar. Cuando *cambio* se active esta vez, el módulo de control regresará al estado *luz* = 0 (luz verde en el camino principal). Cuando *mach* se decremente a su estado terminal (0) esta vez, las líneas 13 a 15 determinarán mediante el estado de la entrada del sensor *auto* si debe esperar a otro auto o si debe cargar el factor de retraso para una luz verde en el camino principal (*tprinverde* – 1), para empezar el ciclo de nuevo. El camino principal recibirá una luz verde durante al menos esta longitud de tiempo, aunque haya un flujo continuo de automóviles en el camino lateral. Es obvio que podríamos hacer mejoras a este diseño, pero sin duda eso lo complicaría aún más.

```
SUBDESIGN  retraso
( reloj, auto, luz[1..0], restablecer          :INPUT;
  tprinverde[4..0], tlateverde[4..0]           :INPUT;
  cambio                                       :OUTPUT; )
VARIABLE
  mach[4..0]                                   :DFF;
BEGIN
  mach[ ].clk = reloj;                              -- con un reloj de 1 Hz, tiempos en segundos
  mach[ ].crln = restablecer;
  IF  mach[ ] == 0  THEN
    CASE luz[ ]  IS                                 -- comprueba el estado del controlador de luz
       WHEN 0 =>
         IF  !auto THEN  mach[ ].d = 0;             -- espera auto en camino lateral
         ELSE mach[ ].d = tprinverde[ ] - 1;        -- establece tiempo para luz verde principal
         END IF;
           WHEN 1 => mach[ ].d = 5 - 1;             -- establece tiempo para luz ambar principal
           WHEN 2 => mach[ ].d = tlateverde[ ] - 1; -- establece tiempo para luz verde lateral
           WHEN 3 => mach[ ].d = 5 - 1;             -- establece tiempo para luz ambar lateral
    END CASE;
   ELSE mach[ ].d = mach[ ].q - 1;                  -- decrementa contador temporizador
   END IF;
   cambio = mach[ ] == 1;                           -- cambia luces en módulo de control
END;
--------------------------------------------------------------------------------------------------
SUBDESIGN  control
( reloj, habilita, restablecer  :INPUT;
    luz[1..0]                   :OUTPUT;   )
VARIABLE
    semaforo: MACHINE OF BITS  (luz[1..0]) -- necesita 4 estados para combinaciones de luces
              WITH STATES (pvrd = B"00", pamb = B"01", lvrd = B"10", lamb = B"11");
BEGIN
    semaforo.clk = reloj;
    semaforo.restablecer = !restablecer; -- las máquinas tienen señal restablecer
                                            asíncrona, activa en alto
    CASE semaforo IS     -- espera a que señal de habilitación cambie estados del semáforo
      WHEN pvrd  =>      IF habilita THEN semaforo = pamb;  ELSE semaforo = pvrd; END IF;
      WHEN pamb  =>      IF habilita THEN semaforo = lvrd;  ELSE semaforo = pamb; END IF;
      WHEN lvrd  =>      IF habilita THEN semaforo = lamb;  ELSE semaforo = lvrd; END IF;
      WHEN lamb  =>      IF habilita THEN semaforo = pvrd;  ELSE semaforo = lamb; END IF;
    END CASE;
END;
--------------------------------------------------------------------------------------------------
SUBDESIGN ctrl_luz
( luz[1..0]                                     :INPUT;
  prinrojo, prinambar, prinverde                :OUTPUT;
  laterojo, lateambar, lateverde                :OUTPUT;     )
BEGIN
  CASE luz[ ]           IS            -- determina cuales luces se van a encender
       WHEN B"00"       =>     prinverde = VCC;  prinambar = GND;  prinrojo = GND;
                               lateverde = GND;  lateambar = GND;  laterojo = VCC;
       WHEN B"01"       =>     prinverde = GND;  prinambar = VCC;  prinrojo = GND;
                               lateverde = GND;  lateambar = GND;  laterojo = VCC;
     WHEN B"10" =>     prinverde = GND;  prinambar = GND;  prinrojo = VCC;
         lateverde = VCC;      lateambar = GND;  laterojo = GND;
       WHEN B"11"       =>     prinverde = GND;  prinambar = GND;  prinrojo = VCC;
                               lateverde = GND;  lateambar = VCC;  laterojo = GND;
         END CASE;
END;
```

FIGURA 7-60 Archivos de diseño de AHDL para el controlador de semáforo.

El módulo de control (líneas 25 a 40) contiene una máquina de estados llamada *semaforo*, la cual avanzará en secuencia a través de los cuatro estados para las combinaciones de semáforos. Los bits para la máquina de estados tienen nombre y están conectados como un puerto de salida para este módulo (líneas 27 y 29). Los cuatro estados para *semaforo* se llaman *pvrd*, *pamb, svrd* y *samb* en la línea 30. Cada estado representa cuál de los caminos (principal o lateral) va a recibir una luz verde o ámbar. El otro camino tendrá una luz roja. También se han especificado los valores para cada estado del módulo de control en la línea 30, de manera que podamos identificarlos como entradas para los otros dos módulos, *retraso* y *ctrl_luz*. La entrada *habilita* está conectada a la señal de salida *cambio*, producida por el módulo de retraso. Cuando se habilite, la máquina de estados *semaforo* avanzará al siguiente estado cuando se aplique un pulso de reloj, como se describe mediante la instrucción CASE y las instrucciones IF anidadas en las líneas 34 a 39. En caso contrario, *semaforo* permanecerá en el estado actual.

El módulo *ctrl_luz* (líneas 42 a 57) tiene como entrada *luz[1..0]*, la cual representa el estado de la máquina de estados *semaforo* del módulo de control, y producirá como salida las señales que enciendan las combinaciones apropiadas de luces verde, ámbar y roja para los caminos principal y lateral. En realidad, cada una de las salidas del módulo *ctrl_luz* estará conectada a circuitos controladores de lámparas para controlar los voltajes y las corrientes con valores más altos, necesarios para los focos reales en un semáforo. La instrucción CASE en las líneas 47 a 55 determina la combinación de luces del (camino principal, camino lateral) que se debe encender para cada estado del *semaforo*. La función del módulo *ctrl_luz* es muy parecida a la de un decodificador. En esencia, decodifica la combinación de cada estado de *luz* para encender una luz verde o ámbar para un camino, y una luz roja para el otro. Para cada estado de entrada se produce una combinación de salida única.

VHDL

CONTROLADOR DE SEMÁFORO EN VHDL

En la figura 7-61 se muestra el diseño de VHDL para el controlador de semáforo. El nivel superior del diseño se describe en forma estructural, en las líneas 1 a 34. Hay que declarar tres módulos COMPONENT (líneas 10 a 24). En las líneas 26 a 33 aparecen las instrucciones PORT MAP, las cuales proporcionan las interconexiones de cableado entre cada módulo y el diseño de nivel superior.

En esencia, el módulo de retraso (líneas 36 a 66) es un contador descendente oculto (línea 59), el cual se crea con la variable entera *mach* que espera en cero cuando el camino principal tiene luz verde (*luz* = "00"), hasta que se dispara mediante el sensor de autos (línea 52) para cargar el factor de retraso *tprinverde* − 1 en la línea 53. Como el contador se decrementa hasta llegar a cero, se resta uno a cada factor de retraso para hacer que el módulo del contador de retraso sea igual al valor del factor de retraso. Por ejemplo, si deseamos tener un factor de retraso de 25, el contador debe contar en forma descendente desde 24 hasta 0. La longitud actual de tiempo representada por los factores de retraso depende de la frecuencia del reloj. Con una frecuencia de reloj de 1-Hz el periodo sería de 1 s, y los factores de retraso estarían en segundos. Las líneas 62-64 definen una señal de salida llamada *cambio*, la cual detecta cuando *mach* es igual a uno. *Cambio* estará en ALTO para indicar que la condición de prueba es verdadera, lo cual a su vez habilitará la máquina de estado en el módulo de control para que avance a su siguiente estado (*luz* = "01") cuando se sincronice con el reloj, para indicar una luz ámbar en el camino principal. Ahora cuando *mach* llegue a cero, la instrucción CASE determinará que *luz* tiene un nuevo valor y se cargará el factor de retraso de tiempo fijo de 5 para una luz ámbar (en realidad se carga uno menos, como dijimos antes) en *mach* (línea 55), en el siguiente pulso de reloj. El conteo descendente continuará a partir de este nuevo tiempo de retraso y *cambio* habilitará de nuevo el módulo de control para que avance a su siguiente estado (*luz* = "10"), con lo cual se obtendrá una luz verde para el camino lateral. Cuando *mach* llegue de nuevo a cero se cargará

```
ENTITY  trafico IS
PORT  (   reloj, auto, restablecer            :IN BIT;
          tprinverde, rlateverde              :IN INTEGER RANGE 0 TO 31;
          luz                                 :BUFFER INTEGER RANGE 0 TO 3;
          cambio                              :BUFFER BIT;
          prinrojo, prinambar, prinverde      :OUT BIT;
          latrojo, latambar, latverde         :OUT BIT);
END trafico;
ARCHITECTURE  nivelsuperior OF trafico IS
COMPONENT retraso
  PORT ( reloj, auto, restablecer             :IN BIT;
          luz                                 :IN INTEGER RANGE 0 TO 3;
          tprinverde, tlateverde              :IN INTEGER RANGE 0 TO 31;
          cambio                              :OUT BIT);
END COMPONENT;
COMPONENT  control
          PORT ( reloj, habilita, restablecer :IN BIT;
          luz                                 :OUT INTEGER RANGE 0 TO 3);
END COMPONENT;
COMPONENT  ctrl_luz
  PORT ( luz                                  :IN INTEGER RANGE 0 TO 3;
          prinrojo, prinambar, prinverde      :OUT BIT;
          latrojo, latambar, latverde         :OUT BIT);
END COMPONENT;
BEGIN
modulo1:  retraso PORT MAP ( reloj => reloj, auto => auto, restablecer => restablecer,
                  luz => luz, trpinverde => tprinverde, tlateverde => tlateverde,
                  cambio => cambio);
modulo2:  control PORT MAP (reloj => reloj, habilita => cambio, restablecer => restablecer,
                  luz => luz);
modulo3:  ctrl_luz PORT MAP (luz => luz, prinrojo => prinrojo, prinambar => prinambar;
                  prinverde => prinverde, latrojo => latrojo, latambar => latambar,
                  latverde => latverde);
END nivelsuperior;
-------------------------------------------------------------------------------------------
ENTITY  retraso  IS
PORT    ( reloj, auto, restablecer            :IN BIT;
          luz                                 :IN BIT_VECTOR (1 DOWNTO 0);
          tprinverde, tlateverde              :IN INTEGER RANGE 0 TO 31;
          cambio                              :OUT BIT);
END retraso;
ARCHITECTURE  tiempo  OF  retraso  IS
BEGIN
  PROCESS  (reloj, restablecer)
  VARIABLE  mach                              :INTEGER RANGE 0 TO 31;
  BEGIN
  IF restablecer = '0'  THEN  mach := 0;
  ELSIF (reloj = '1' AND reloj'EVENT) THEN  -- con reloj de 1 Hz, tiempo en segundos
      IF mach = 0  THEN
         CASE luz  IS
            WHEN "00"
              IF auto = '0' THEN  mach := 0;     -- espera auto en camino lateral
              ELSE  mach:=  tprinverde - 1;      -- establece tiempo para luz verde principal
              END IF;
           WHEN "01" => mach := 5 - 1;           -- establece tiempo para luz ambar principal
           WHEN "10" => mach := tlateverde - 1; -- establece tiempo para luz verde lateral
           WHEN "11" => mach :=  5 - 1:          -- establece tiempo para luz ambar lateral
         END CASE;
     ELSE  mach :=  mach - 1;                   -- decrementa contador temporizador
     END IF;
END IF;
```

FIGURA 7-61 Diseño de VHDL para el controlador de semáforo.

```
62      IF  mach = 1 THEN cambio <= '1';                    -- cambia luces en control
63      ELSE  cambio <= '0';
64      END IF;
65      END PROCESS;
66  END tiempo;
67  ---------------------------------------------------------------------------------
68  ENTITY  control  IS
69  PORT (              reloj, habilita, restablecer  :IN BIT;
70                 luz                                :OUT BIT_VECTOR (1 DOWNTO 0) );
71  END control;
72  ARCHITECTURE  a  OF  control IS
73  TYPE enumerado IS (pvrd, pamb, lvrd, lamb); -- necesita 4 estados para las
                                                    combinaciones de luces
74  BEGIN
75      PROCESS (reloj, restablecer)
76      VARIABLE  luces:  enumerado;
77      BEGIN
78          IF  restablecer = '0'  THEN  luces := pvrd;
79          ELSIF (reloj = '1' AND reloj'EVENT)  THEN
80             IF habilita = '1'  THEN -- espera a habilita para cambiar estados de luces
81                 CASE  luces  IS
82                        WHEN  pvrd    =>     luces := pamb;
83                        WHEN  pamb    =>     luces := lvrd;
84                        WHEN lvrd     =>     luces := lamb;
85                        WHEN lamb     =>     luces := pvrd;
86                              END CASE;
87                        END IF;
88                 END IF;
89                 CASE  luces  IS            -- patrones para los estados de las luces
90                        WHEN  pvrd=> luz <= "00";
91                        WHEN  pamb=> luz <= "01;
92                        WHEN  lvrd=> luz <= "10"
93                        WHEN  lamb=> luz <= "11";
94                 END CASE;
95          END PROCESS;
96  END  a;
97  ---------------------------------------------------------------------------------
98  ENTITY  ctrl_luz  IS
99  PORT      ( luz                                  :IN BIT_VECTOR (1 DOWNTO 0);
100             prinrojo, prinambar, prinverde       :OUT BIT;
101             latrojo, latambar, latverde          :OUT BIT);
102 END  ctrl_luz;
103 ARCHITECTURE  patrones  OF  ctrl_luz  IS
104 BEGIN
105        PROCESS  (luz)
106        BEGIN
107        CASE  luz IS --   el estado de control determina cuales luces encender/apagar
108              WHEN "00"    =>  prinverde <= '1';  prinambar <= '0';   prinrojo <= '0';
109                               latverde  <= '0';  latambar  <= '0';   latrojo  <= '1';
110              WHEN "01"    =>  prinverde <= '0';  prinambar <= '1';   prinrojo <= '0';
111                               latverde  <= '0';  latambar  <= '0';   latrojo  <= '1';
112              WHEN "10"    =>  prinverde <= '0';  prinambar <= '0';   prinrojo <= '1';
113                               latverde  <= '1';  latambar  <= '0';   latrojo  <= '0';
114              WHEN "1"     =>  prinverde <= '0';  prinambar <= '0';   prinrojo <= '1';
115                               latverde  <= '0';  latambar  <= '1';   latrojo  <= '0';
116                    END CASE;
117        END PROCESS;
118 END  patrones;
```

FIGURA 7-61 *Continuación*

en el contador descendente (línea 56) el retraso de tiempo (*tlateverde – 1*) para una luz verde en el camino lateral. Cuando *cambio* se vuelva a activar, *luz* avanzará al estado "11" para una luz ámbar en el camino lateral. *Mach* se reciclará al valor 4 (5-1) en la línea 57 para el retraso de tiempo fijo para una luz ámbar. Cuando *cambio* se active esta vez, el módulo de control regresará a *luz* = "00" (luz verde en el camino principal). Cuando *mach* se decremente a su estado terminal (0) esta vez, las líneas 52 a 54 determinarán mediante el estado de la entrada del sensor *auto* si debe esperar a otro auto o si debe cargar el factor de retraso para una luz verde en el camino principal (*tprinverde* – 1), para empezar el ciclo de nuevo. El camino principal recibirá una luz verde durante al menos esta longitud de tiempo, aún si hay un flujo continuo de automóviles en el camino lateral. Es obvio que podríamos hacer mejoras a este diseño, pero sin duda eso lo complicaría aún más.

El módulo de control (líneas 68 a 96) contiene una máquina de estado llamada *luces*, la cual avanzará en secuencia a través de cuatro estados enumerados para las combinaciones de luces del semáforo. Los cuatro estados enumerados para *luces* son *pvrd*, *pamb*, *lvrd* y *lamb* (líneas 73 y 76). Cada estado representa el camino (principal o lateral) que va a recibir una luz verde o ámbar. El otro camino tendrá una luz roja. La entrada *habilita* se conecta a la señal de salida *cambio* que produce el módulo de retraso. Cuando se habilite, la máquina de estado *luces* avanzará al siguiente estado cuando se sincronice por el reloj, como se describe en las instrucciones IF anidada y CASE de las líneas 79 a 88. En caso contrario, *luces* permanecerá en el estado actual. Los patrones de bits para el puerto de salida *luz* se han especificado para cada estado de *luces* mediante la instrucción CASE en las líneas 89-94, para que podamos identificarlas como entradas para los otros dos módulos: *retraso* y *ctrl_luz*.

El módulo *ctrl_luz* (líneas 98 a 118) tiene como entrada a *luz*, la cual representa el estado de la máquina de estados *luces* del módulo de control, y producirá como salida las señales que enciendan las combinaciones apropiadas de luces verde, ámbar y roja para los caminos principal y lateral. Cada salida del módulo *ctrl_luz* estará conectada a circuitos controladores de lámparas para suministrar los voltajes y corrientes necesarios para los focos en un semáforo real. La instrucción CASE en las líneas 107 a 116, la cual se invoca mediante el bloque PROCESS cuando cambia la entrada *luz*, determina cuál combinación de luces del camino principal/camino lateral deben encenderse para cada estado de *luces*. La función del módulo *ctrl_luz* es muy parecida a la de un decodificador. En esencia, decodifica cada combinación de estado de *luz* para encender una luz verde o ámbar para un camino y una luz roja para el otro. Para cada estado de entrada se produce una combinación de salida única.

En estos momentos tal vez usted se esté preguntando: ¿Por qué hay tantas formas de describir los circuitos lógicos? Si una es más sencilla que las demás, ¿por qué no sólo estudiar esa? Sin duda la respuesta es que cada nivel de abstracción ofrece ciertas ventajas en comparación con los demás, en ciertos casos. El método estructural proporciona el control más completo sobre las interconexiones. El uso de ecuaciones booleanas, tablas de verdad y tablas de estado PRESENTE / estado SIGUIENTE nos permite describir la manera en que fluyen los datos a través del circuito, mediante el uso de HDL. Por último, el método de comportamiento permite una descripción más abstracta de la operación del circuito, en términos de causa y efecto. En la práctica, cada archivo fuente puede tener porciones que se clasifiquen bajo cada uno de los niveles de abstracción. La acción de seleccionar el nivel correcto al escribir código no es una cuestión de bueno o malo, sino de estilo y preferencia.

También hay varias formas de lidiar con cualquier tarea, desde el punto de vista relacionado con la elección de las estructuras de control. ¿Debemos usar asignaciones de señales selectas o ecuaciones booleanas, IF/ELSE o CASE, procesos secuenciales o instrucciones concurrentes, macrofunciones o megafunciones? ¿O deberíamos escribir nuestro propio código? Las respuestas a esas preguntas definirán en última instancia la estrategia personal que usted utilice para resolver el problema. Sus preferencias y las ventajas que encuentre al utilizar un método en comparación con otro se establecerán con la práctica y la experiencia.

PREGUNTAS DE REPASO

1. ¿Cuál es la diferencia fundamental entre un contador y una máquina de estados?
2. ¿Cuál es la diferencia entre describir un contador y describir una máquina de estados en un HDL?
3. Si los estados binarios actuales para una máquina de estados no se definen en el código de HDL, ¿cómo se asignan?
4. ¿Cuál es la ventaja de utilizar la descripción de una máquina de estados?

RESUMEN PARTE 1

1. En los contadores asíncronos (de rizo) la señal de reloj se aplica al FF del LSB y todos los demás FFs se sincronizan mediante la salida del FF anterior.
2. El número MOD de un contador es el número de estados estables en su ciclo de conteo; también es la proporción máxima de división de frecuencia.
3. El número MOD normal (máximo) de un contador es 2^N. Una manera de modificar el número MOD de un contador es agregar circuitos que hagan que se recicle antes de llegar a su último conteo normal.
4. Los contadores pueden conectarse en cascada (encadenarse) para producir rangos de conteo mayores y proporciones de división de frecuencia más altas.
5. En un contador síncrono (en paralelo), todos los FFs se sincronizan en forma simultánea a partir de la señal de reloj de entrada.
6. La frecuencia de reloj máxima para un contador asíncrono ($f_{máx}$) se decrementa a medida que se incrementa el número de bits. En un contador síncrono $f_{máx}$ permanece igual, sin importar el número de bits.
7. Un contador de décadas es cualquier contador MOD-10. Un contador BCD es un contador de décadas que avanza en secuencia a través de los 10 códigos BCD (0-9).
8. Un contador preajustable puede cargarse con cualquier conteo inicial que se desee.
9. Un contador ascendente/descendente puede controlarse para contar hacia arriba o hacia abajo.
10. Las compuertas lógicas pueden utilizarse para decodificar (detectar) uno o todos los estados de un contador.
11. La secuencia de conteo para un contador síncrono puede determinarse con facilidad mediante el uso de una tabla de estado PRESENTE / estado SIGUIENTE, en la cual se listen todos los posibles estados, la información de control de entrada de los flip-flops y los estados SIGUIENTE resultantes.
12. Para implementar los contadores síncronos con secuencias de conteo arbitrarias se sigue un procedimiento de diseño estándar.
13. Los contadores pueden describirse de muchas formas distintas mediante el uso de HDL, incluyendo descripciones de cableado estructurales, tablas de estado PRESENTE estado SIGUIENTE y descripciones de comportamiento.
14. Todas las características disponibles en los diversos CIs estándar tipo contador de CI, tales como la carga o el borrado asíncronos o síncronos, la habilitación del conteo y la decodificación del conteo terminal, pueden describirse mediante HDL. Los contadores en HDL pueden modificarse con facilidad para números MOD más altos o cambios en los niveles activos para los controles.
15. Los sistemas digitales pueden subdividirse en módulos o bloques más pequeños, los cuales pueden interconectarse como un diseño jerárquico.
16. Las máquinas de estados pueden representarse en HDL mediante el uso de nombres descriptivos para cada estado, en vez de especificar una secuencia numérica de estados.

TÉRMINOS IMPORTANTES PARTE 1

carga en paralelo
conexión en cascada
contador ascendente
contador asíncrono (de rizo)
contador autocorregible
contador BCD
contador de décadas
contador descendente
contadores ascendentes/descendentes
contadores de múltiples etapas
contadores preajustables
contadores síncronos (en paralelo)
decodificación
deformaciones
diseño de circuitos secuenciales
diseño jerárquico
habilitación de conteo
MACHINE
máquina de estados
modelo de Mealy
modelo de Moore
nivel de abstracción de comportamiento
número MOD
tabla de estado PRESENTE, estado SIGUIENTE
tabla de excitación de circuitos
tabla de excitación J-K
tipo enumerado
VARIABLE

PARTE 2

7-15 REGISTROS DE CIRCUITO INTEGRADO

Los diversos tipos de registros pueden clasificarse de acuerdo con la forma en que pueden introducirse datos en el registro para su almacenamiento y la forma en que se envían los datos de salida desde el registro. A continuación se muestra una lista de las diversas clasificaciones.

1. Entrada en paralelo/Salida en paralelo (PIPO).
2. Entrada en serie/Salida en serie (SISO).
3. Entrada en paralelo/Salida en serie (PISO).
4. Entrada en serie/Salida en paralelo (SIPO).

Cada uno de estos tipos y algunas variaciones de los mismos están disponibles en forma de CI para que un diseñador de lógica digital pueda encontrar con exactitud lo que se requiere para cierta aplicación. En las siguientes secciones examinaremos una forma de CI representativa de cada una de las categorías antes mencionadas.

7-16 ENTRADA EN PARALELO/SALIDA EN PARALELO: 74ALS174/74HC174

El registro tipo **entrada en paralelo/salida en paralelo** es un grupo de flip-flops que puede almacenar varios bits al mismo tiempo; en este tipo de registros todos los bits del valor binario almacenado están disponibles de manera directa. La figura 7-62(a) muestra el diagrama lógico para el 74ALS174 (también el 74HC174), un registro de seis bits que tiene las entradas en paralelo D_5 a D_0 y las salidas en paralelo Q_5 a Q_0. Los datos en paralelo se cargan en el registro, en la PGT de la entrada de reloj *CP*. Puede utilizarse una entrada maestra *MR* (Master Reset) para restablecer a 0 en forma asíncrona todos los FFs del registro. La figura 7-62(b) muestra el símbolo lógico para el 74ALS174. Este símbolo se utiliza en los diagramas de circuitos para representar los circuitos de la figura 7-62(a).

Por lo general, el 74ALS174 se utiliza para la transferencia síncrona de datos en paralelo, en donde los niveles lógicos presentes en las entradas *D* se transfieren a las salidas *Q* correspondientes cuando se produce una PGT en la entrada de reloj *CP*. No obstante, este CI puede cablearse para la transferencia de datos en serie, como se mostrará en los siguientes ejemplos.

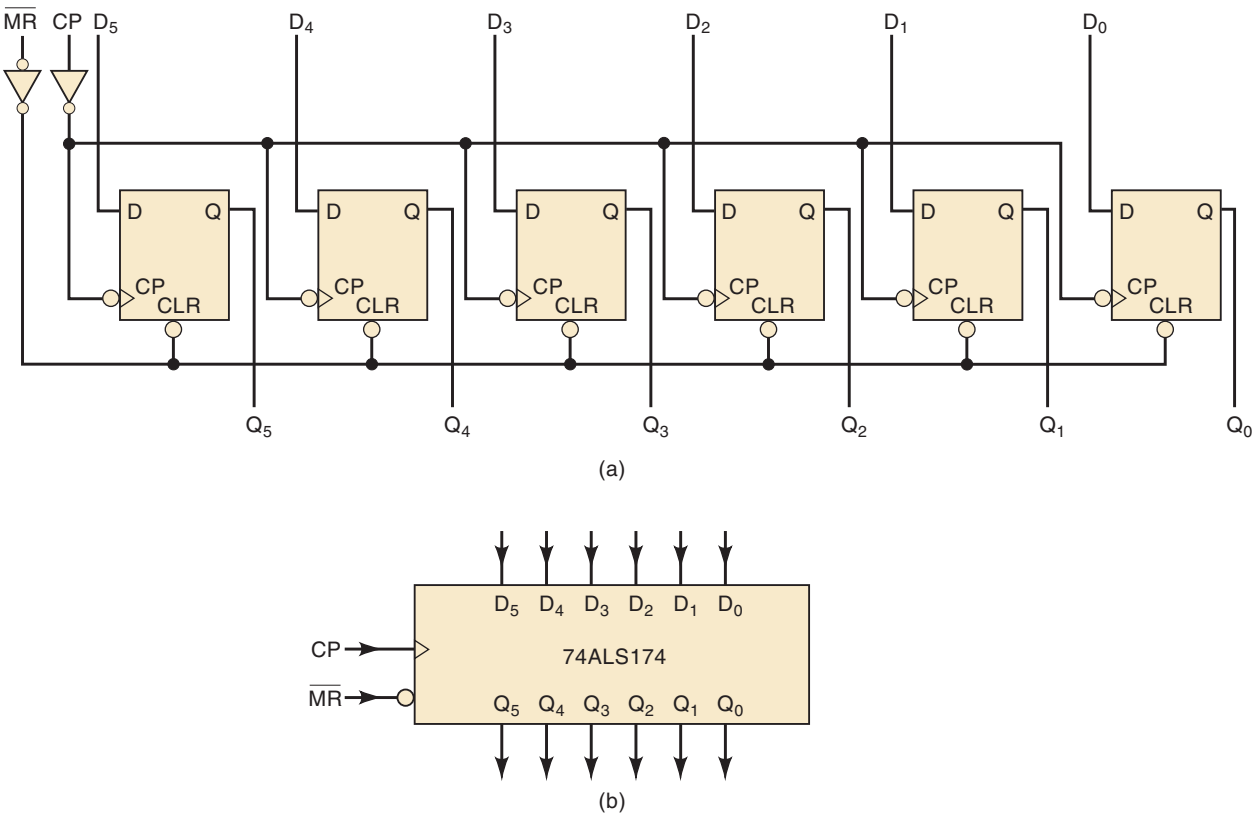


FIGURA 7-62 (a) Diagrama del circuito del 74ALS174; (b) símbolo lógico.

EJEMPLO 7-16

Muestre cómo conectar el 74ALS174 de manera que opere como un registro de desplazamiento en serie, en donde los datos se desplacen en cada PGT de *CP* como se indica a continuación: Entrada en serie → $Q_5 \rightarrow Q_4 \rightarrow Q_3 \rightarrow Q_2 \rightarrow Q_1 \rightarrow Q_0$. En otras palabras, los datos en serie entrarán en D_5 y saldrán en Q_0.

Solución

Si analizamos la figura 7-62(a) podremos ver que para conectar los seis FFs como un registro de desplazamiento en serie, tenemos que conectar la salida Q de uno a la entrada D del siguiente, de manera que los datos se transfieran de la forma requerida. La figura 7-63 muestra cómo se realiza esto. Observe que los datos se desplazan de izquierda a derecha; los datos de entrada se aplican en D_5 y los datos de salida aparecen en Q_0.

EJEMPLO 7-17

¿Cómo conectaría dos CIs 74ALS174 para operar como un registro de desplazamiento de 12 bits?

Solución

Se conecta un segundo CI 74ALS174 como registro de desplazamiento y se conecta Q_0 del primer CI a D_5 del segundo CI. Se conectan las entradas *CP* de ambos CIs, de manera que se sincronicen con la misma señal. Además se conectan las entradas MR juntas si se utiliza la señal de restablecer asíncrona (RESET).

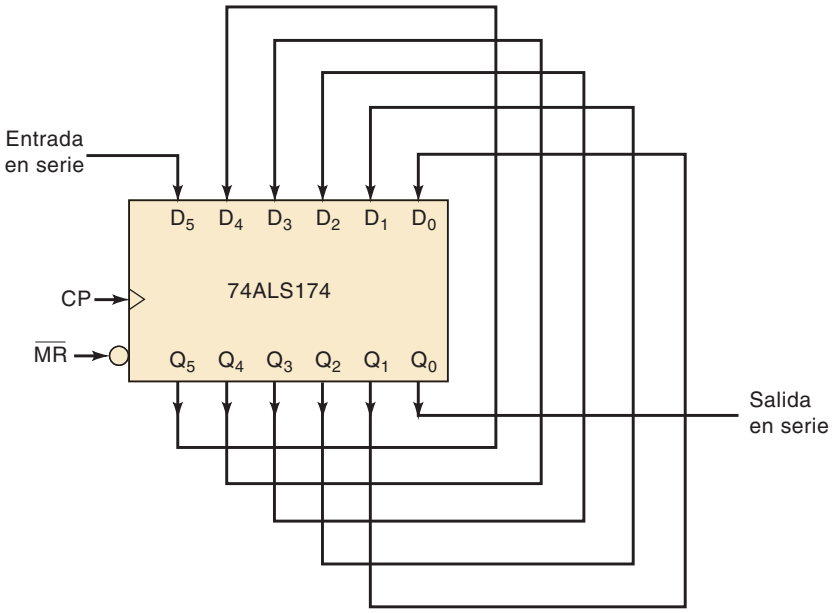


FIGURA 7-63
Ejemplo 7-16: el 74ALS174 cableado como un registro de desplazamiento.

7-17 ENTRADA EN SERIE/SALIDA EN SERIE: 74ALS166/74HC166

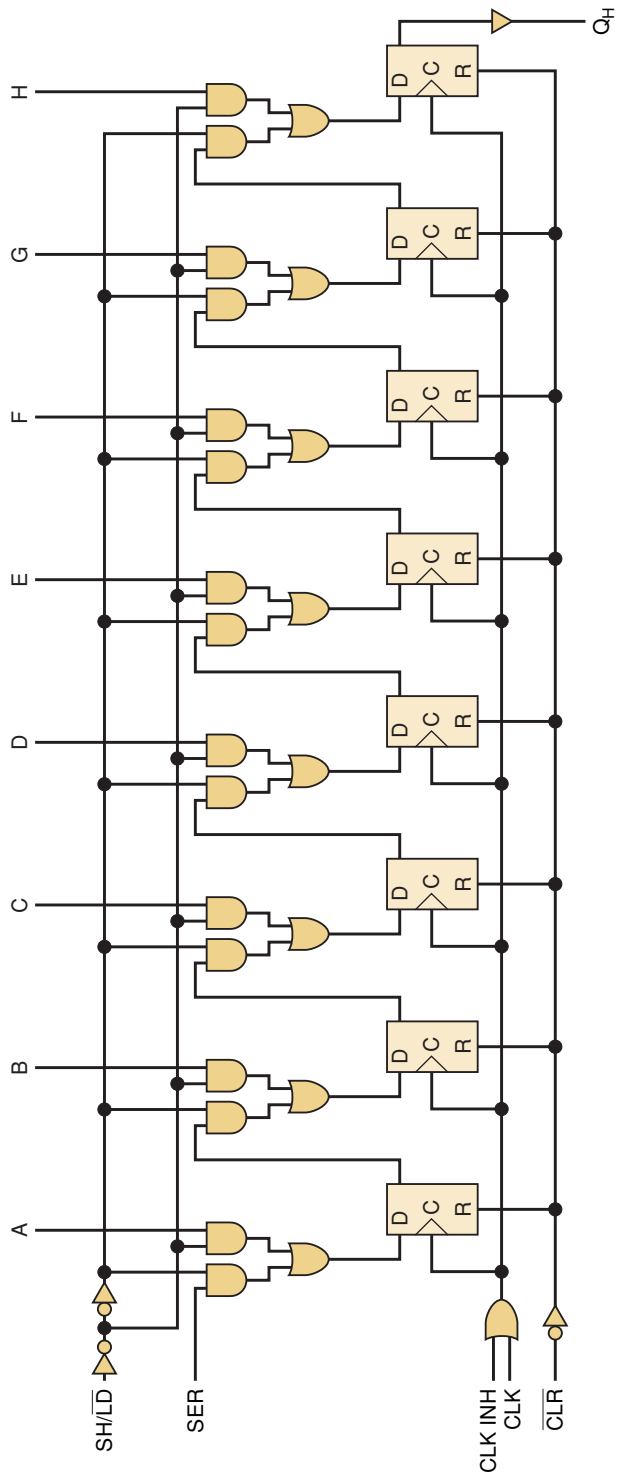
Un registro de desplazamiento del tipo **entrada en serie/salida en serie** cargará los datos un bit a la vez. Los datos se desplazarán bit por bit con cada pulso de reloj, a través del conjunto de flip-flops y hacia el otro extremo del registro. Con los pulsos de reloj continuos, los datos saldrán del registro uno por uno en el mismo orden en el que se cargaron. El 74HC166 (y también el 74ALS166) puede utilizarse como un registro de entrada en serie/salida en serie. La figura 7-64 muestra el diagrama lógico y el símbolo esquemático para el 74HC166. Es un registro de desplazamiento de ocho bits, del cual sólo se puede tener acceso al FF QH. Los datos en serie se introducen en *SER* y se almacenan en el FF QA. La salida en serie se obtiene en el otro extremo del registro de desplazamiento, en Q_H. Como podemos ver en la tabla de funciones de la figura 7-64(c) a este registro de desplazamiento también se le pueden cargar datos en paralelo. Si SH/LD = 1 la función del registro será desplazamiento en serie, mientras que un nivel BAJO indicará datos de carga en paralelo a través de las entradas *A* – *H*. Las funciones síncronas de desplazamiento en serie y carga en paralelo pueden inhibirse (deshabilitarse) mediante la aplicación de un nivel ALTO a la entrada de control *CLK INH*. EL registro también tiene una entrada borrar asíncrona (*CLR*) activa en BAJO.

EJEMPLO 7-18

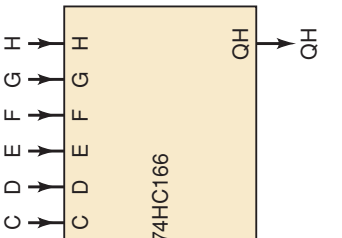
A menudo un registro de desplazamiento se utiliza como una forma de retrasar una señal digital, con base en cierto número de ciclos de reloj. La señal digital se aplica a la entrada en serie del registro de desplazamiento y se desplaza a través de éste mediante pulsos sucesivos de reloj hasta que llega al final del registro de desplazamiento, en donde aparece como la señal de salida. Este método para retrasar el efecto de una señal digital es común en el campo de las comunicaciones digitales. Por ejemplo, la señal digital podría ser la versión digitalizada de una señal de audio que debe retrasarse antes de transmitirla. Las formas de onda de entrada que se muestran en la figura 7-65 se aplican a un 74HC166. Determine la forma de onda de salida resultante.

Solución

QH empieza en nivel BAJO, ya que al principio todos los flip-flops se borran mediante el nivel BAJO que se aplica a la entrada *CLR* asíncrona, en el comienzo del diagrama de sincronización.



(a)



(b)

ENTRADAS							SALIDAS		
					EN PARALELO		INTERNAS		
$\overline{CLR}$	SH/$\overline{LD}$	CLK INH	CLK	SER	A . . . H		Q_A	Q_B	Q_H
L	X	X	X	X	X		L	L	L
H	X	L	L	X	X		Q_{A0}	Q_{B0}	Q_{H0}
H	L	L	↑	X	a . . . h		a	b	h
H	H	L	↑	H	X		H	Q_{An}	Q_{Gn}
H	H	L	↑	L	X		L	Q_{An}	Q_{Gn}
H	X	H	↑	X	X		Q_{A0}	Q_{B0}	Q_{H0}

(c)

FIGURA 7-64 (a) Diagrama del circuito del 74HC166; (b) símbolo lógico; (c) tabla de funciones.

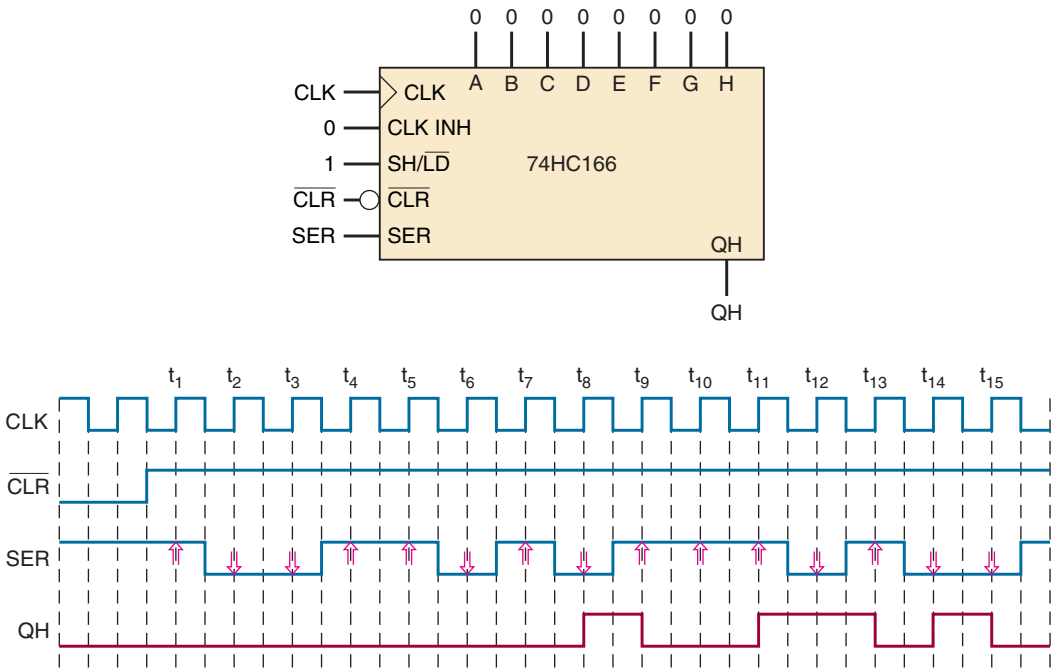


FIGURA 7-65 Ejemplo 7-18.

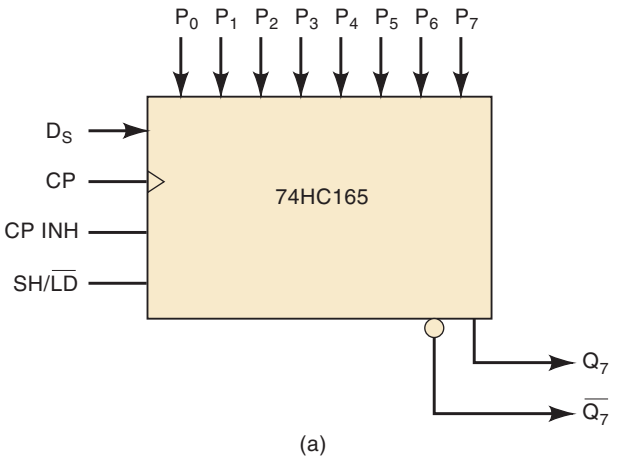
En t_1 el registro de desplazamiento recibirá como entrada el bit que se está aplicando a *SER* en esos momentos. Éste se almacenará en QA. En t_2 el primer bit se desplazará a QB y se almacenará el segundo bit de *SER* en *QA*. En t_3 el primer bit se desplazará ahora a *QC* y el tercer bit de *SER* se almacenará en QA. Por último, el primer bit de entrada de datos aparecerá en la salida *QH*, en t_8. Cada bit de entrada sucesivo en *SER* llegará a QH, retrasado por ocho ciclos de reloj.

7-18 ENTRADA EN PARALELO/SALIDA EN SERIE: 74ALS165/74HC165

La figura 7-66(a) muestra el símbolo lógico para el 74HC165. Este CI es un registro de ocho bits, tipo **entrada en paralelo/salida en serie**. En realidad tiene una entrada de datos en serie a través de D_S y una entrada de datos síncrona en paralelo a través de $P_0 - P_7$. El registro contiene ocho FFS (Q_0 a Q_7) conectados en forma interna como un registro de desplazamiento, pero las únicas salidas accesibles de los FFs son Q_7 y $\overline{Q}_7$. *CP* es la entrada de reloj que se utiliza para la operación de desplazamiento. La entrada de inhibición de reloj *CP INH* se utiliza para inhibir el efecto de la entrada *CP*. La entrada de desplazamiento/carga, *SH/LD*, controla qué operación se está llevando a cabo: desplazamiento o carga en paralelo. La tabla de funciones en la figura 7-66(b) muestra cómo las diversas combinaciones de entrada determinan cuál operación se está realizando, en caso de haberla. La carga en paralelo es asíncrona y el desplazamiento en serie es síncrono. La función de desplazamiento en serie siempre será síncrona, ya que el reloj necesita asegurar que los datos de entrada se desplacen sólo un bit a la vez, con cada flanco de reloj apropiado.

EJEMPLO 7-19

Examine la tabla de funciones del 74HC165 y determine: (a) las condiciones necesarias para cargar el registro con datos en paralelo; (b) las condiciones necesarias para la operación de desplazamiento.



(a)

Tabla de funciones

Entradas			Operación
SH/$\overline{LD}$	CP	CP INH	
L	X	X	Carga en paralelo
H	H	X	Sin cambio
H	X	H	Sin cambio
H	↓	L	Desplazamiento
H	L	↓	Desplazamiento

H = nivel alto
L = nivel bajo
X = no importa
↓ = PGT

(b)

FIGURA 7-66
(a) Símbolo lógico para el registro 74HC165 tipo entrada en paralelo/salida en serie; (b) tabla de funciones.

Solución

(a) La primera entrada en la tabla muestra que la entrada *SH/LD* tiene que estar en BAJO para la operación de carga en paralelo. Cuando esta entrada está en BAJO, los datos presentes en las entradas *P* se cargan en forma *asíncrona* en los FFs del registro, sin importar las entradas *CP* y *CP INH*. Desde luego que sólo las salidas del último FF están disponibles desde el exterior.

(b) La operación de desplazamiento no puede llevarse a cabo sino hasta que la entrada *SH/LD* esté en ALTO y se produzca una PGT en *CP* mientras que *CP INH* esté en BAJO [vea la cuarta entrada de la tabla en la figura 7-66(b)]. Un nivel ALTO en *CP INH* inhibirá el efecto de cualquier pulso de reloj. Observe que pueden invertirse los papeles de las entradas *CP* y *CP INH* según lo indica la última entrada en la tabla, ya que dentro del CI se aplica una operación OR entre estas dos señales.

EJEMPLO 7-20

Determine la señal de entrada en Q_7 si conectamos un 74HC165 con $D_S = 0$ y CPINH = 0; después aplique las formas de onda de entrada que se muestran en la figura 7-67. $P_0 - P_7$ representan los datos paralelos en $P_0P_1P_2P_3P_4P_5P_6P_7$.

Solución

Hemos dibujado el diagrama de sincronización para los ocho FFs, de manera que se pueda rastrear su contenido en el tiempo, aún y cuando sólo Q_7 estará accesible. La carga en paralelo es síncrona y ocurrirá tan pronto como SH/LD cambie a BAJO. Después de que SH/LD regrese al nivel ALTO, los datos almacenados en el registro se desplazarán un FF a la derecha (hacia Q_7) con cada PGT en CP.

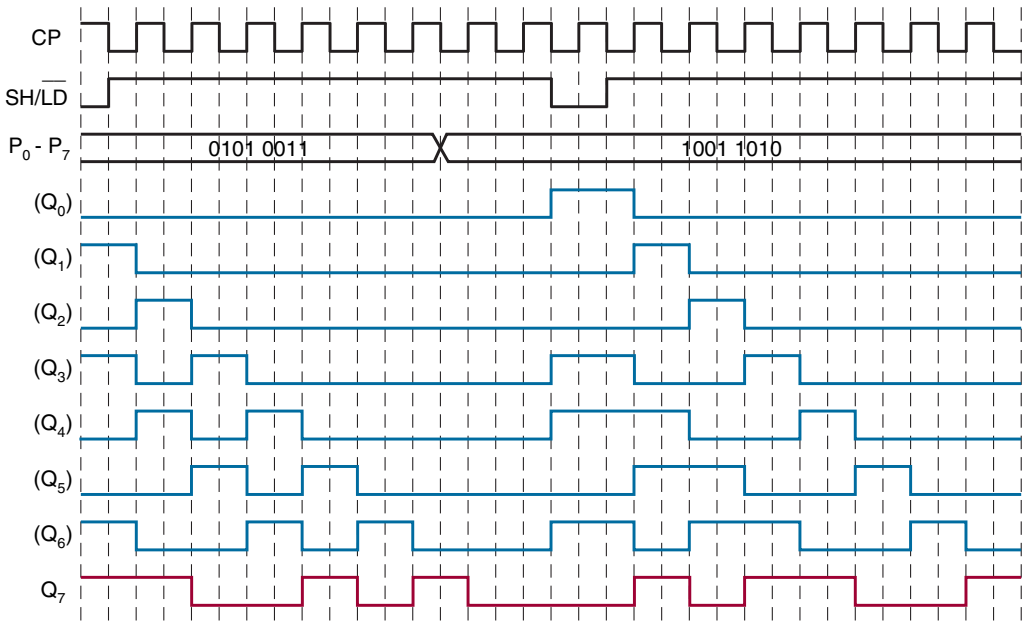


FIGURA 7-67 Ejemplo 7-20.

7-19 ENTRADA EN SERIE/SALIDA EN PARALELO: 74ALS164/74HC164

La figura 7-68(a) muestra el diagrama lógico para el 74ALS164. Es un registro de desplazamiento de ocho bits tipo **entrada en serie/salida en paralelo**, en el cual la salida de cada FF es accesible desde el exterior. En vez de una sola entrada en serie, una compuerta AND combina las entradas *A* y *B* para producir la entrada serial para el flip-flop Q_0.

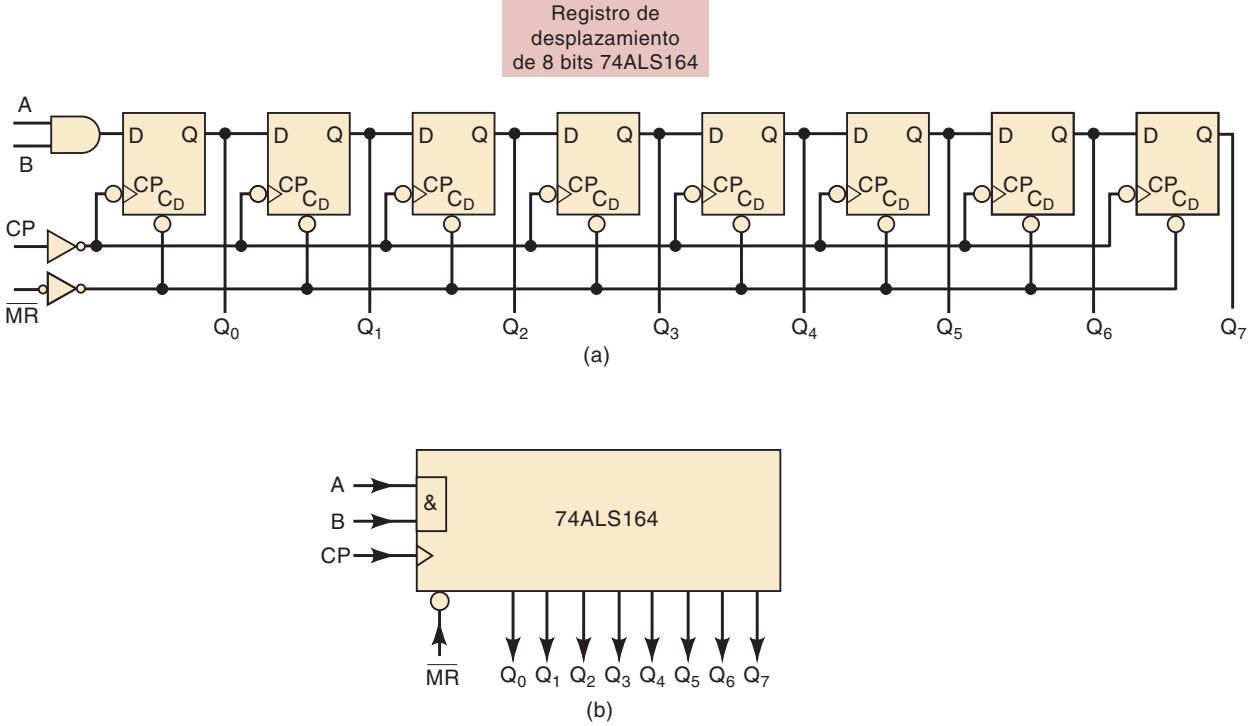


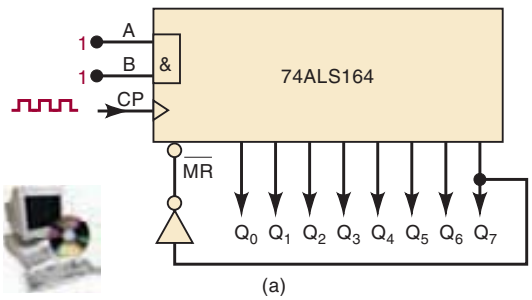
FIGURA 7-68 (a) Diagrama lógico para el 74ALS164; (b) símbolo lógico.

La operación de desplazamiento ocurre en las PGTs de la entrada de reloj *CP*. La entrada $\overline{MR}$ proporciona el restablecimiento asíncrono de todos los FFs, para que queden en nivel BAJO.

La figura 7-68(b) muestra el símbolo lógico para el 74ALS164. Observe que se utiliza el símbolo & dentro del bloque para indicar que se aplica una operación AND entre las entradas *A* y *B* dentro del CI, y que el resultado se aplica a la entrada *D* de Q_0.

EJEMPLO 7-21

Suponga que el contenido inicial del registro 74ALS164 en la figura 7-69(a) es 00000000. Determine la secuencia de estados a medida que se aplican pulsos de reloj.



Número de pulso de entrada	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7
0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0
3	1	1	1	0	0	0	0	0
4	1	1	1	1	0	0	0	0
5	1	1	1	1	1	0	0	0
6	1	1	1	1	1	1	0	0
7	1	1	1	1	1	1	1	0
8	1	1	1	1	1	1	1	1

Se recicla

Estado temporal

(b)

FIGURA 7-69 Ejemplo 7-21.

Solución

La figura 7-69(b) muestra la secuencia correcta. Cuando $A = B = 1$ la entrada en serie es 1, por lo que se desplazarán 1s en el registro durante cada PGT de *CP*. Debido a que al principio Q_7 está en 0, la entrada $\overline{MR}$ está inactiva.

En el octavo pulso, el registro trata de cambiar al estado 11111111 cuando el 1 de Q_6 se desplaza hacia Q_7. Este estado ocurre sólo en forma momentánea debido a que $Q_7 = 1$ produce un nivel BAJO en $\overline{MR}$, con lo cual se restablece de inmediato el registro de vuelta a 00000000. Después se repite la secuencia en los siguientes ocho pulsos de reloj.

A continuación se muestra una lista de algunos otros CIs de registro que son variaciones de los que ya presentamos antes:

- 74194/ALS194/HC194. Este CI es un *registro de desplazamiento universal bidireccional de cuatro bits*, el cual puede realizar las operaciones de desplazamiento a la izquierda, desplazamiento a la derecha, entrada en paralelo y salida en paralelo. Estas operaciones se seleccionan mediante un código de selección de modo de dos bits, el cual se aplica en forma de entradas al dispositivo. (El problema 7-71 le dará la oportunidad de averiguar más acerca de este versátil CI).
- 74373/ALS373/HC373/HCT373. Este CI es un registro de entrada en paralelo/salida en paralelo de ocho bits (octal), el cual contiene ocho circuitos de retención (*latches*) tipo *D* con salidas *de tercer estado* (triestado). Una salida de tercer estado es un tipo especial de salida que permite enlazar salidas de dispositivos en forma segura. En el siguiente capítulo veremos las características de los dispositivos de tercer estado, tales como el 74373.
- 74374/ALS374/HC374/HCT374. Este CI es un registro de entrada en paralelo/salida en paralelo de ocho bits (octal), el cual contiene ocho flip-flops D disparados por flanco, con salidas triestado.

Los registros de CI que presentamos aquí son representativos de los diversos tipos que están disponibles en forma comercial. Aunque hay muchas variaciones de estos registros básicos, en este momento usted podrá entender con facilidad la mayoría de ellos, mediante la consulta de las hojas técnicas proporcionadas por los fabricantes.

En los problemas al final del capítulo y en el material que cubriremos en capítulos subsiguientes presentaremos varias aplicaciones de registros.

PREGUNTAS DE REPASO

1. ¿En qué tipo de registro se puede cargar un número binario completo y después desplazarlo a la salida un bit a la vez?
2. *Verdadero o falso:* un registro de entrada en serie/salida en paralelo puede mostrar todos sus bits al mismo tiempo.
3. ¿En qué tipo de registro se pueden introducir datos sólo un bit a la vez y en el que todos los bits de datos pueden estar disponibles como salida?
4. ¿En qué tipo de registro podemos almacenar datos un bit a la vez y tener acceso a sólo un bit de salida a la vez?
5. ¿Qué diferencia hay entre la entrada de datos en paralelo del 74165 y del 74174?
6. ¿Cómo funciona la entrada *CP INH* del 74ALS165?

7-20 CONTADORES DE REGISTRO DE DESPLAZAMIENTO

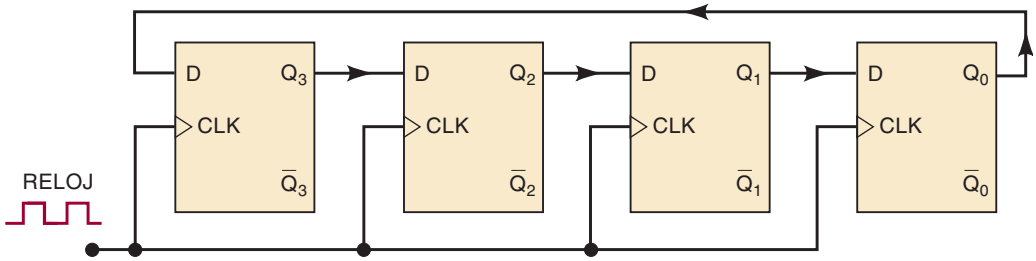
En la sección 5-18 vimos cómo conectar FFs en un arreglo de registro de desplazamiento para transferir datos de izquierda a derecha o viceversa, un bit a la vez (en serie). Los contadores de registro de desplazamiento utilizan *retroalimentación*, lo cual significa que la salida del último FF en el registro se conecta de regreso al primer FF.

Contador de anillo

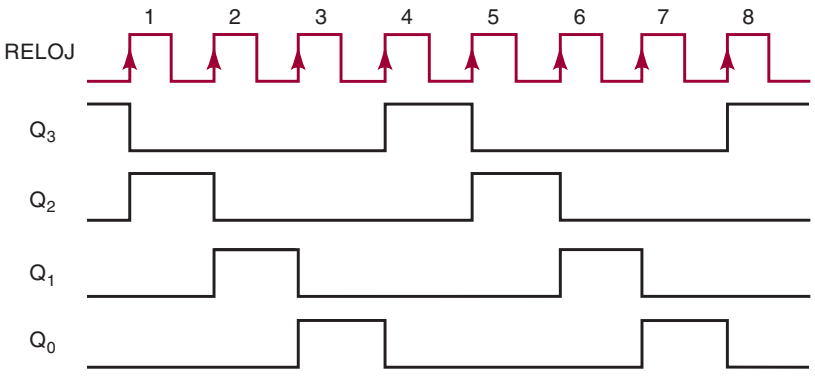
El contador de registro de desplazamiento más sencillo es un **registro de desplazamiento circular**, éste es conectado de tal forma que el último FF desplaza su valor hacia el primer FF. La figura 7-70 muestra este arreglo, en el cual se utilizan FFs tipo D (también pueden utilizarse flip-flops J-K). Los FFs se conectan de tal forma que la información se desplaza de izquierda a derecha y de regreso desde Q_0 hasta Q_3. En la mayoría de los casos sólo hay un 1 en el registro, el cual se pone a circular alrededor del registro mientras se aplican pulsos de reloj. Por esta razón se le conoce como **registro de anillo**.

Las formas de onda, la tabla de secuencias y el diagrama de estados de la figura 7-70 muestran los diversos estados de los FFs a medida que se aplican pulsos; aquí se supone un estado inicial de $Q_3 = 1$ y $Q_2 = Q_1 = Q_0 = 0$. Después del primer pulso el 1 se desplaza de Q_3 a Q_2, de forma tal que el contador se encuentra en el estado 0100. El segundo pulso produce el estado 0010 y el tercer pulso produce el estado 0001. En el *cuarto* pulso de reloj, el 1 de Q_0 se transfiere a Q_3 para producir el estado 1000 que es, desde luego, el estado inicial. Los siguientes pulsos hacen que se repita la secuencia.

Este contador funciona como un contador MOD-4 debido a que tiene *cuatro* estados distintos antes de que se repita la secuencia. Aunque este circuito no progresa por la secuencia de conteo binaria normal, de todas formas se considera un contador ya que cada conteo corresponde a un conjunto único de estados de los FFs. Observe que cada una de las formas de onda de salida de los FFs tiene una frecuencia igual a un cuarto de la frecuencia de reloj, debido a que éste es un contador de anillo MOD-4.



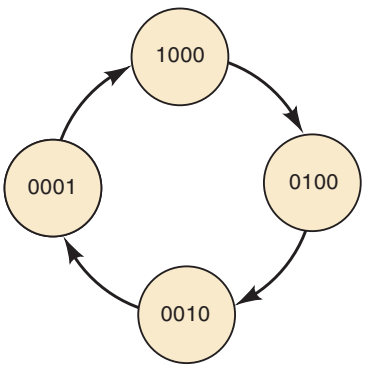
(a)



(b)

Q_3	Q_2	Q_1	Q_0	Pulso de RELOJ
1	0	0	0	0
0	1	0	0	1
0	0	1	0	2
0	0	0	1	3
1	0	0	0	4
0	1	0	0	5
0	0	1	0	6
0	0	0	1	7
.	.	.	.	.
.	.	.	.	.

(c)



(d)



FIGURA 7-70 (a) Contador de anillo de cuatro bits; (b) formas de onda; (c) tabla de secuencia; (d) diagrama de estado.

Pueden construirse contadores de anillo para cualquier número MOD que se desee; un contador de anillo MOD-*N* utiliza *N* flip-flops conectados en el arreglo que aparece en la figura 7-70.

En general, un contador de anillo requiere más FFs que un contador binario para el mismo número MOD; por ejemplo, un contador de anillo MOD-8 requiere ocho FFs, mientras que un contador binario MOD-8 sólo requiere tres.

A pesar del hecho de que es menos eficiente en el uso de FFs, un contador de anillo sigue siendo útil debido a que puede decodificarse sin el uso de compuertas de decodificación. La señal de decodificación para cada estado se obtiene en la salida de su FF correspondiente. Compare las formas de onda de los FFs del contador de anillo con las formas de onda de decodificación en la figura 7-20. En algunos casos, un contador de anillo podría ser mejor elección que un contador binario con sus compuertas de decodificación asociadas. En especial esto es útil en aplicaciones en donde se utiliza el contador para controlar la secuencia de operaciones de un sistema.

Inicio de un contador de anillo

Para operar en forma apropiada, un contador de anillo debe comenzar con sólo un FF en el estado 1 y todos los demás en el estado 0. Como los estados iniciales de los FFs serán impredecibles al momento de encenderlos, el contador debe prestablecerse en el estado inicial requerido antes de aplicar pulsos de reloj. Una manera de hacer esto es aplicar un pulso momentáneo a la entrada $\overline{PRE}$ asíncrona de uno de los FFs (por ejemplo, Q_3 en la figura 7-70) y a la entrada $\overline{CLR}$ de todos los demás FFs. La figura 7-71 muestra otro método. Al momento del encendido, el capacitor se cargará lentamente hacia $+V_{CC}$. La salida del INVERSOR disparador de Schmitt 1 permanecerá en ALTO y la salida del INVERSOR 2 permanecerá en BAJO hasta que el voltaje del capacitor se exceda del voltaje de umbral con tendencia positiva (V_{T+}) de la entrada del INVERSOR 1 (un valor aproximado de 1.7 V). Con esto se retendrá en el estado BAJO las entradas $\overline{PRE}$ de Q_3 y $\overline{CLR}$ de Q_2, Q_1 y Q_0 el tiempo suficiente durante el encendido como para asegurar que el contador empiece en 1000.

Contador Johnson

El contador de anillo básico puede modificarse un poco para producir otro tipo de contador de registro de desplazamiento, el cual tendrá propiedades un poco distintas. El **contador Johnson** o de **anillo torcido** se construye en forma idéntica a un contador de anillo normal, sólo que la salida *invertida* del último FF se conecta a la entrada del primer FF. En la figura 7-72 se muestra un contador Johnson de tres bits. Observe que la salida $\overline{Q}_0$ se conecta de regreso a la entrada D de Q_2, lo cual significa que el *inverso* del nivel almacenado en Q_0 se transferirá a Q_2 en el pulso de reloj.

Es fácil analizar la operación del contador Johnson si tenemos en cuenta que en cada transición positiva del pulso de reloj el nivel en Q_2 se desplaza hacia Q_1, el nivel en Q_1 se desplaza hacia Q_0 y el *inverso* del nivel en Q_0 se desplaza hacia Q_2. Si utilizamos estas ideas y suponemos que todos los FFs están al principio en 0, se pueden generar las formas de onda, la tabla de secuencias y el diagrama de estados de la figura 7-72.

El análisis de las formas de onda y la tabla de secuencias nos revela los siguientes puntos importantes:

1. Este contador tiene seis estados distintos (000, 100, 110, 111, 011 y 001) antes de repetir la secuencia. Por ende, es un contador Johnson MOD-6. Observe que no cuenta con una secuencia binaria normal.
2. La forma de onda de cada FF es una onda cuadrada (ciclo de trabajo del 50 por ciento) que tiene una frecuencia igual a un sexto de la frecuencia del reloj. Además, la forma de onda de cada FF se desplaza por un periodo de reloj, con respecto a las demás.

El número MOD de un contador Johnson siempre será igual al *doble* del número de FFs. Por ejemplo, si conectamos cinco FFs en el arreglo de la figura 7-72 el

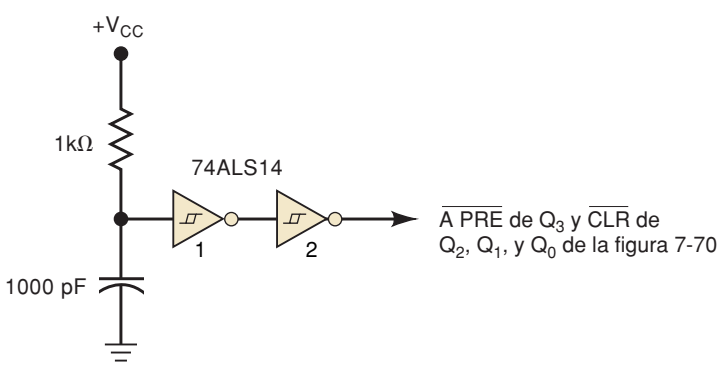
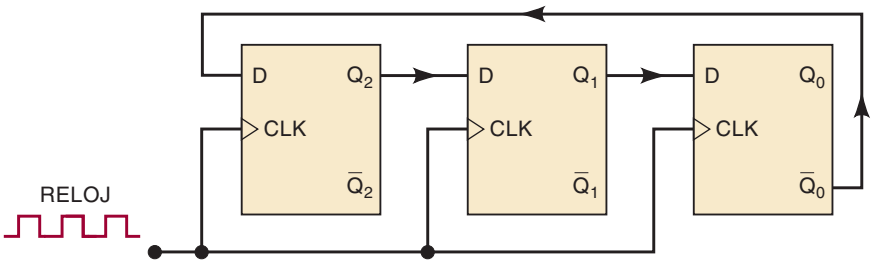


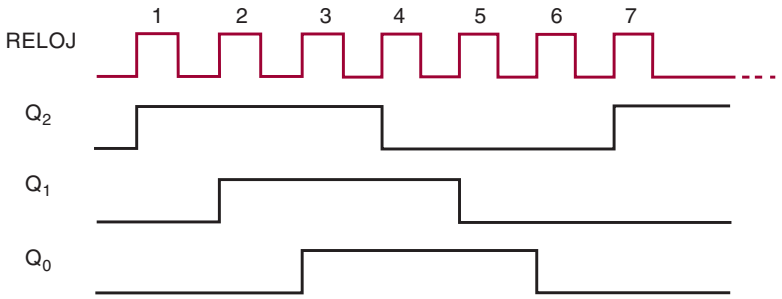
FIGURA 7-71 Circuito para asegurar que el contador de anillo de la figura 7-70 inicie en el estado 1000 al momento del encendido.

FIGURA 7-72
(a) Contador Johnson MOD-6; (b) forma de onda; (c) tabla de secuencias; (d) diagrama de estados.





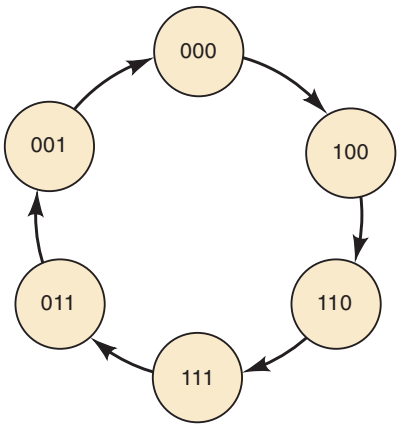
(a)



(b)

Q_2	Q_1	Q_0	Pulso de RELOJ
0	0	0	0
1	0	0	1
1	1	0	2
1	1	1	3
0	1	1	4
0	0	1	5
0	0	0	6
1	0	0	7
1	1	0	8
.	.	.	.
.	.	.	.
.	.	.	.

(c)

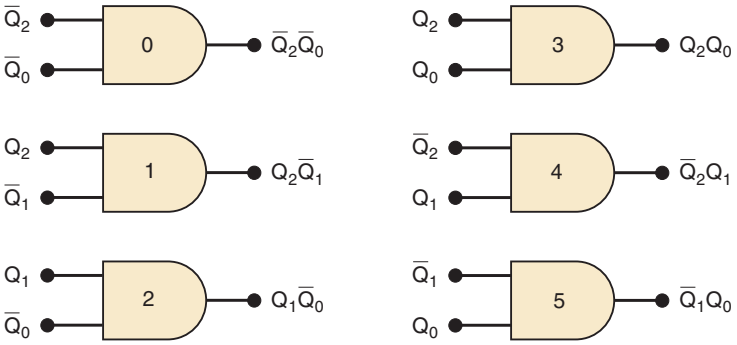


(d)

resultado será un contador Johnson MOD-10, en donde la forma de onda de salida de cada FF es una onda cuadrada a un décimo de la frecuencia del reloj. Por ende, es posible construir un contador MOD-*N* (en donde *N* es un número par) mediante la conexión de *N/2* flip-flops en un arreglo de contadores Johnson.

Decodificación de un contador Johnson

Para un número MOD dado, un contador Johnson sólo requiere la mitad del número de FFs que requiere un contador de anillo. No obstante, un contador Johnson requiere compuertas de decodificación, mientras que un contador de anillo no. Al igual que el contador binario, el de Johnson utiliza una compuerta lógica para decodificar en cada conteo, pero cada compuerta sólo requiere dos entradas, sin importar el número de FFs en el contador. La figura 7-73 muestra las compuertas de decodificación para los seis estados del contador Johnson de la figura 7-72.



Q_2	Q_1	Q_0	Compuerta activa
0	0	0	0
1	0	0	1
1	1	0	2
1	1	1	3
0	1	1	4
0	0	1	5

FIGURA 7-73 Lógica de decodificación para un contador Johnson MOD-6.

Observe que cada compuerta de decodificación sólo tiene dos entradas aún y cuando hay tres FFs en el contador, porque para cada conteo dos de los tres FFs se encuentran en una combinación única de estados. Por ejemplo, la combinación $Q_2 = Q_0 = 0$ sólo ocurre una vez en la secuencia de conteo, en la cuenta de 0. Por lo tanto, puede utilizarse la compuerta AND 0 con las entradas $\overline{Q}_2$ y Q_0 para decodificar en este conteo. Esta misma característica la comparten todos los demás estados en la secuencia, como podrá verificar el lector. De hecho, para un contador Johnson de *cualquier tamaño* las compuertas de decodificación sólo tendrán dos entradas.

Los contadores Johnson representan un punto intermedio entre los contadores de anillo y los contadores binarios. Un contador Johnson requiere menos FFs que un contador de anillo, pero, por lo general, requiere más que un contador binario; tiene más circuitos de decodificación que un contador de anillo pero menos que un contador binario. Por ende, algunas veces representa una buena opción para ciertas aplicaciones.

Contadores de registro de desplazamiento de CI

Hay muy pocos contadores de anillo o Johnson disponibles como CIs, debido a que es muy fácil tomar un CI de registro de desplazamiento y cablearlo como un contador de anillo o como un Johnson. Algunos de los CIs de contador Johnson CMOS son el 74HC4017 y el 74HC4022, éstos incluyen circuitos de decodificación completos.

PREGUNTAS DE REPASO

1. ¿Cuál contador de registro de desplazamiento requiere la mayor cantidad de FFs para un número MOD dado?
2. ¿Cuál contador de registro de desplazamiento requiere la mayor cantidad de circuitos de decodificación?
3. ¿Cómo puede convertirse un contador de anillo en un contador Johnson?
4. *Verdadero o falso:*
   (a) Las salidas de un contador de anillo son siempre ondas cuadradas.
   (b) Los circuitos de decodificación para un contador Johnson son más simples que los de un contador binario.
   (c) Los contadores de anillo y Johnson son contadores síncronos.
5. ¿Cuántos FFs se necesitan en un contador de anillo MOD-16? ¿Cuántos se necesitan en un contador Johnson MOD-16?

7-21 DIAGNÓSTICO DE FALLAS

Los flip-flops, los contadores y los registros son los principales componentes en los **sistemas lógicos secuenciales**. Debido a sus dispositivos de almacenamiento, un sistema lógico secuencial tiene la característica de que sus salidas y la secuencia de las operaciones dependen tanto de las entradas presentes como de las que se produjeron antes. Aún y cuando los sistemas lógicos secuenciales son, por lo general, más complejos que los combinacionales, los procedimientos esenciales para el diagnóstico de fallas se aplican con la misma eficacia en ambos tipos de sistemas. Los sistemas secuenciales sufren de los mismos tipos de fallas (circuitos abiertos, cortos, fallas internas en los CIs, etcétera) que los combinacionales.

Muchos de los mismos pasos que se utilizan para aislar fallas en un sistema combinacional pueden aplicarse a los sistemas secuenciales. Una de las técnicas de diagnóstico de fallas más efectiva comienza desde que el técnico de diagnóstico de fallas observa la operación del sistema y, mediante el razonamiento analítico, determina las posibles causas de la falla en el sistema. Después el técnico utiliza los instrumentos de prueba disponibles para aislar la falla exacta. Los siguientes ejemplos demostrarán los tipos de razonamiento analítico que deben ser el paso inicial al diagnosticar fallas en los sistemas secuenciales. Después de estudiar estos ejemplos, usted deberá estar preparado para enfrentar los problemas de diagnóstico de fallas al final del capítulo.

EJEMPLO 7-22

La figura 7-74(a) muestra un 74ALS161 cableado como un contador MOD-12, pero produce la secuencia de conteo que se muestra en la figura 7-74(b). Determine la causa del comportamiento incorrecto del circuito.

Solución

Las salidas QB y QA parecen operar en forma correcta, pero QC y QD permanecen en BAJO. Nuestra primera opción para la falla es que QC está en corto con tierra, pero al comprobar con un óhmetro no se confirma esto. El 74ALS161 podría tener una falla interna que evite que cuente más allá de 0011. Tratamos de extraer el chip NAND 7400 de su zócalo y ponemos en corto la terminal CLR con un nivel ALTO.

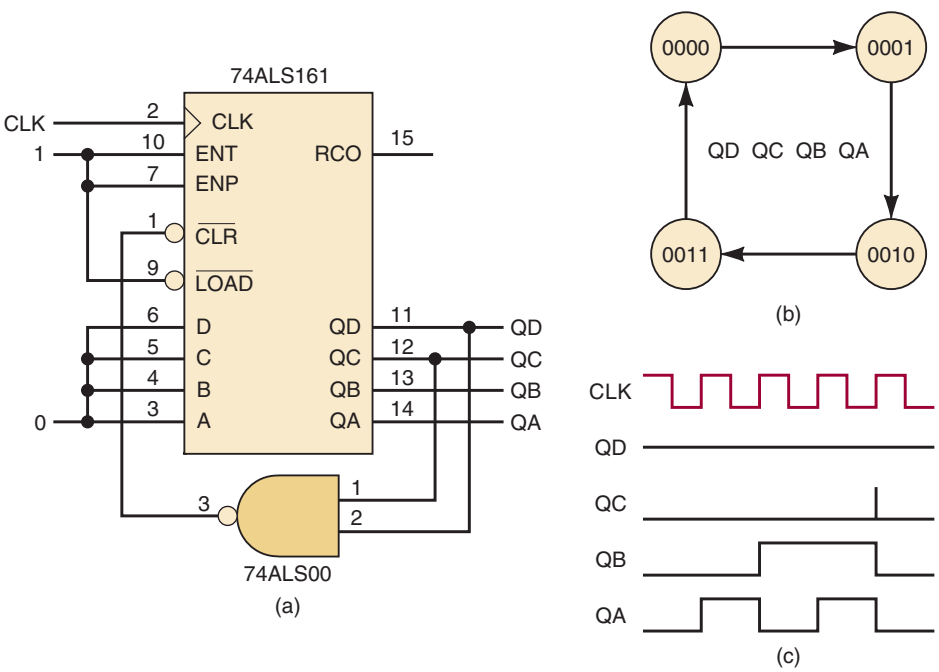


FIGURA 7-74 Ejemplo 7-22.

Ahora el contador cuenta en una secuencia MOD-16 regular, por lo que al menos las salidas del contador parecen estar bien. A continuación decidimos analizar la terminal CLR volviendo a conectar la compuerta NAND. Al utilizar una sonda lógica con su "captura de pulsos" encendida, descubrimos que la terminal CLR está recibiendo pulsos. Al conectar un osciloscopio a las salidas vemos que el contador produce las formas de onda que se muestran en la figura 7-74(c). Se observa una deformación en QC cuando el contador debería pasar al estado 0100. Eso indica que 0100 es un estado transitorio, cuando en realidad el estado transitorio debería ser 1100. Ahora sospechamos de la conexión entre QD y la compuerta NAND, por lo que utilizamos la sonda lógica para revisar la terminal 2. En esta terminal no hay ninguna señal lógica, lo cual nos lleva a la conclusión de que la falla es debido a un circuito abierto entre la salida QD y la terminal 2 en la compuerta NAND. La entrada de la compuerta NAND está flotando en ALTO, lo cual provoca que el circuito detecte el estado 0100 en vez del 1100, como debería ser.

EJEMPLO 7-23

Un técnico recibe una "orden de reparación" para un tablero de circuitos, en la cual se indica que el divisor de frecuencia variable opera "algunas veces". Suena como uno de los temidos problemas en los que la falla es intermitente; a menudo uno de lo problemas más difíciles de resolver. Su primer pensamiento es regresar el tablero con la nota "¡Utilícelo sólo cuando opere en forma correcta!", pero decide investigar más, ya que se siente capaz de enfrentar un buen reto. En la figura 7-75 se muestra el diagrama esquemático para el bloque del circuito. El factor de división deseado se aplica a la entrada *f[7..0]* en binario. El contador de ocho bits cuenta en

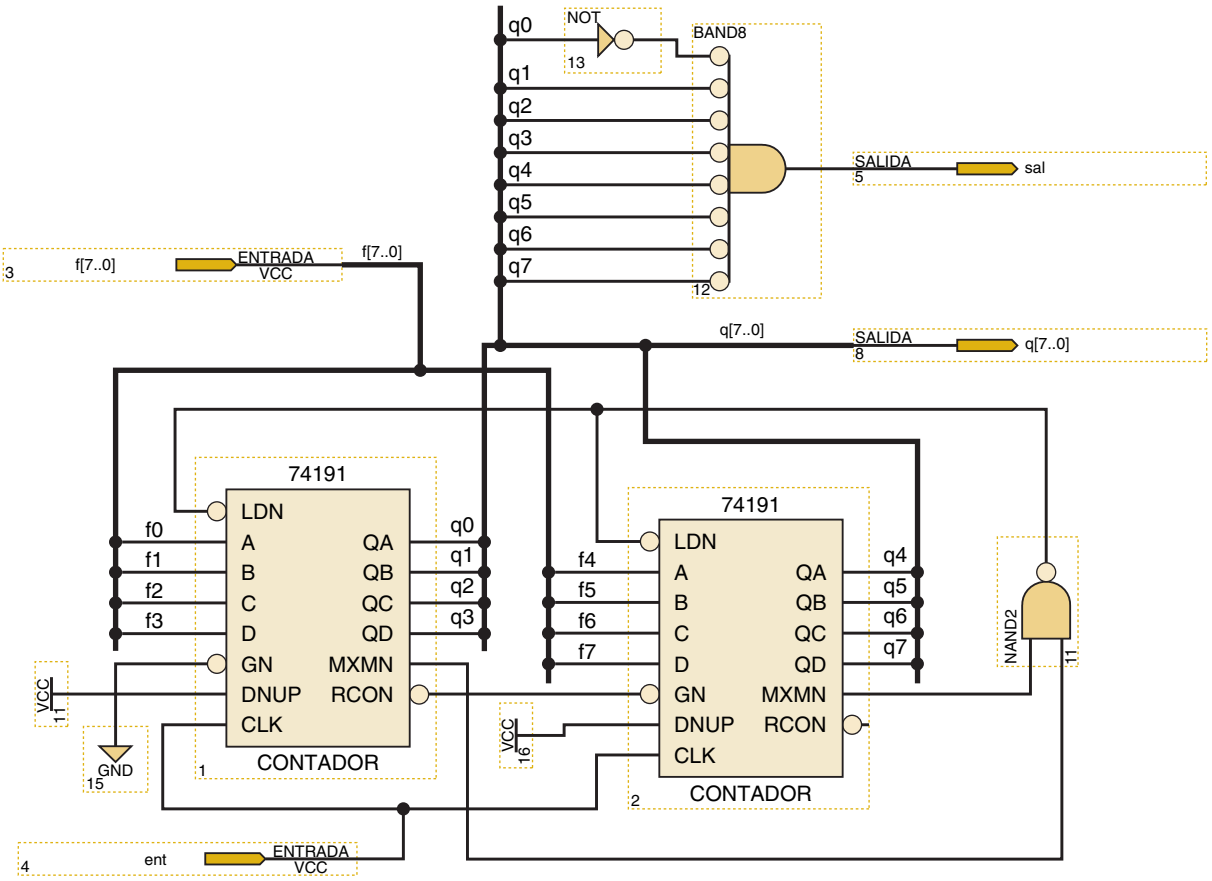


FIGURA 7-75 Ejemplo 7-23.

forma descendente a partir de este número hasta que llega a cero y después realiza una carga asíncrona de *f[ ]* otra vez, convirtiendo al cero en un estado transitorio. El módulo resultante será igual al valor en *f[ ]*. Para obtener la señal de frecuencia de salida se decodifica el estado 00000001, con lo cual la frecuencia de *sal* se hace igual a la frecuencia de *ent*, dividida entre el valor binario *f[ ]*. En la aplicación, la frecuencia de *ent* es 100 kHz. Si se cambia *f[ ]* se producirá una nueva frecuencia de salida.

Solución

El técnico decide que necesita obtener algunos resultados de prueba para analizarlos. Elije algunos factores de división sencillos para aplicarlos a *f* y registra los resultados que se muestran en la tabla 7-9.

TABLA 7-9

f[ ] (decimal)	f[ ] (binario)	f_{sal} medida	¿Correcto?
255	11111111	398.4 Hz	
240	11110000	416.7 Hz	✓
200	11001000	500.0 Hz	✓
100	01100100	1041.7 Hz	
50	00110010	2000.0 Hz	✓
25	00011001	4000.0 Hz	✓
15	00001111	9090.9 Hz	

El técnico observa que el circuito produce resultados correctos en algunos casos de prueba, pero produce resultados incorrectos en otros. El problema no parece ser intermitente después de todo, sino que parece depender del valor para *f*. El técnico decide calcular la relación entre las frecuencias de entrada y de salida para las tres pruebas que fallaron y obtiene lo siguiente:

```
100 kHz/398.4 Hz = 251
100 kHz/1041.7 Hz = 96
100 kHz/9090.9 Hz = 11
```

Cada falla parece un factor de división que es cuatro unidades menor que el valor que se aplicó a la entrada. Después de analizar otra vez la representación binaria para *f*, el técnico observa que todas las fallas ocurrieron cuando *f2* = 1. Sin duda, el peso para ese bit es cuatro. ¡Eureka! Ese bit no parece estar funcionando; es hora de hacer una prueba con una sonda lógica en la terminal *f2*. Asimismo, la sonda lógica indica que la terminal está en BAJO sin importar el valor para *f2*.

7-22 REGISTROS EN HDL

En las secciones 7-15 a 7-19 describimos con detalle las diversas opciones de la transferencia de datos en serie y en paralelo dentro de los registros; además describimos algunos ejemplos de CIs que realizan estas operaciones. La belleza de utilizar HDL para describir un registro está en el hecho de que a un circuito se le puede dar cualquiera de las opciones y se le pueden asignar tantos bits como sean necesarios con sólo cambiar unas cuantas palabras.

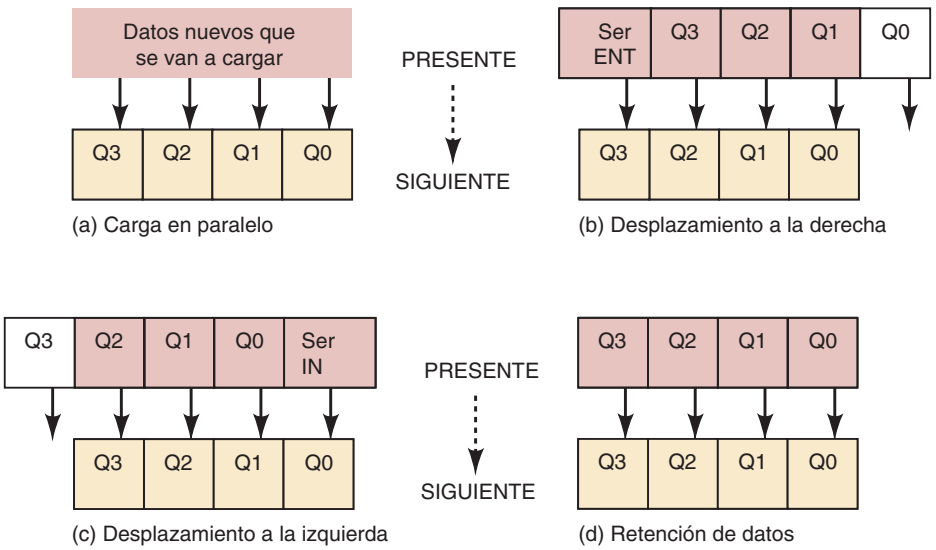


FIGURA 7-76
Transferencias de datos que se realizan en registros de desplazamiento: (a) carga en paralelo; (b) desplazamiento a la derecha; (c) desplazamiento a la izquierda; (d) retención de datos.

Las técnicas de HDL utilizan arreglos de bits para describir los datos de un registro y transferir esos datos en formato en paralelo o en serie. Para comprender cómo se desplazan los datos en HDL considere los diagramas de la figura 7-76, en donde se muestran cuatro flip-flops que realizan operaciones de transferencia de carga en paralelo, desplazamiento a la derecha, desplazamiento a la izquierda y retención de datos. Para todos estos diagramas los bits se transfieren en forma síncrona, lo cual significa que todos se desplazan en forma simultánea en un solo flanco de reloj. En la figura 7-76(a) los datos que se van a cargar en paralelo en el registro se presentan en las entradas *D*, y en el siguiente pulso de reloj se transferirán a las salidas *q*. Desplazar los datos a la derecha significa que cada bit se transfiere hacia la posición del bit que está a su derecha inmediata, mientras que se transfiere un nuevo bit de entrada en el extremo izquierda y se pierde el último bit en el extremo derecho. Esta situación se ilustra en la figura 7-76(b). Observe que el conjunto de datos que queremos en el estado SIGUIENTE está compuesto de la nueva entrada en serie y tres de los cuatro bits en el arreglo del estado PRESENTE. Estos datos sólo necesitan desplazarse y sobrescribir los cuatro bits de datos del registro. La misma operación se produce en la figura 7-76(c), pero los datos se desplazan a la izquierda. La clave para desplazar el contenido del registro a la derecha o a la izquierda es agrupar los tres bits de datos apropiados del estado PRESENTE en orden correcto con el bit de entrada en serie, de manera que estos cuatro bits puedan cargarse en paralelo en el registro. Puede utilizarse la **concatenación** (agrupamiento en una secuencia específica) del conjunto deseado de bits de datos para describir el movimiento de datos necesario para el desplazamiento en serie en cualquier dirección. A la última posibilidad se le conoce como el modo de retención de datos y se muestra en la figura 7-76(d). Ésta puede parecer innecesaria, ya que los registros (flip-flops) retienen los datos por su propia naturaleza. No obstante, debemos tener en cuenta lo que debe hacerse a un registro para que pueda retener su valor cuando recibe pulsos de reloj. Las salidas *Q* deben enlazarse con las entradas *D* para cada flip-flop, de manera que los datos anteriores se recarguen en cada pulso de reloj. Ahora analicemos algunos ejemplos de circuitos de registro de desplazamiento en HDL.

REGISTRO SISO EN AHDL

La figura 7-77 lista un registro de cuatro bits, tipo entrada en serie/salida en serie (SISO), en AHDL. En la línea 7 se crea una instancia de un arreglo de cuatro flip-flops D y se obtiene la salida en serie a partir del último FF *q0* (línea 10). Si el control *desplaza* está en ALTO, *serial_ent* se desplazará hacia el registro y los demás

AHDL

```
1   SUBDESIGN  fig7_77
2   (
3           clk, desplaza, serial_ent       :INPUT;
4           serial_sal                      :OUTPUT;
5   )
6   VARIABLE
7           q[3..0]                         :DFF;
8   BEGIN
9           q[ ].clk = clk;                         -- envía último bit del registro como salida
10          serial_sal = q0.q;
11          IF (desplaza == VCC) THEN
12            q[3..0].d = (serial_ent, q[3..1].q); -- concatena para el desplazamiento
13          ELSE
14            q[3..0].d = (q[3..0].q);              -- retiene los datos
15          END IF;
16  END;
```

FIGURA 7-77 Registro de entrada en serie/salida en serie mediante el uso de AHDL.

bits se moverán a la derecha (líneas 11 a 15). Al concatenar *serial_ent* con los bits de salida de los FFs *q3*, *q2* y *q1* en ese orden se crea el patrón de bits apropiado de entrada de datos de desplazamiento a la derecha (línea 12). Si el control *desplaza* está en BAJO, el registro retendrá los datos actuales (línea 14). En la figura 7-78 se muestran los resultados de la simulación.

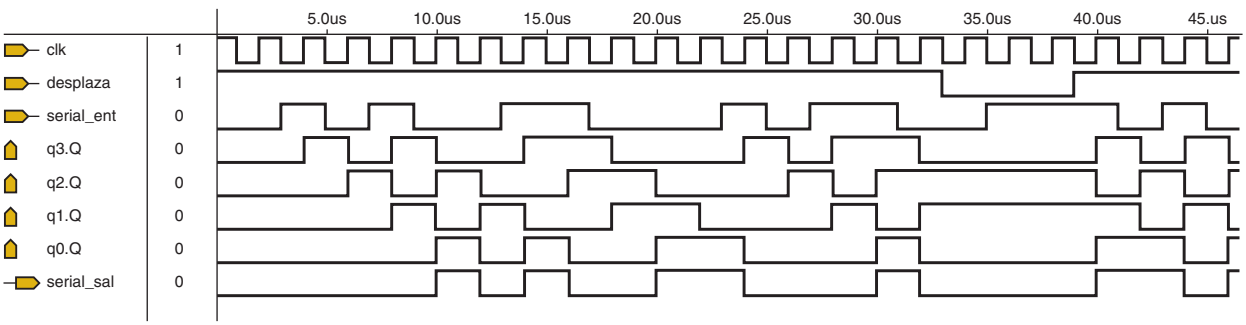


FIGURA 7-78 Simulación de un registro SISO.

VHDL

REGISTRO SISO EN VHDL

La figura 7-79 lista un registro tipo entrada en serie/salida en serie (SISO) de cuatro bits en VHDL. En la línea 8 se crea un registro con la declaración de la variable *q* y se obtiene la salida en serie mediante el último bit del registro o *q(0)* (línea 10). Si el control *desplaza* está en ALTO, *serial_ent* se desplazará hacia el registro y los otros bits se moverán a la derecha (líneas 12 a 14). Al concatenar *serial_ent* y los bits del registro *q(3)*, *q(2)* y *q(1)* en ese orden se crea el patrón apropiado de bits de entrada de datos de desplazamiento a la derecha (línea 13). Si el control de desplazamiento está en BAJO, VHDL supondrá que la variable se queda igual y, por lo tanto, retendrá los datos actuales. La figura 7-78 muestra los resultados de la simulación.

```
1  ENTITY  fig7_79   IS
2  PORT ( clk, desplaza, serial_ent           :IN BIT;
3               serial_sal                    :OUT BIT    );
4  END  fig7_79;
5  ARCHITECTURE  vhdl  OF  fig7_79  IS
6  BEGIN
7  PROCESS  (clk)
8         VARIABLE    q                :BIT_VECTOR (3 DOWNTO 0);
9         BEGIN
10        serial_sal <= q(0);          -- envía último bit del registro como salida
11        IF (clk'EVENT AND clk = '1')  THEN
12           IF (desplaza = '1')  THEN
13              q := (serial_ent & q(3 DOWNTO 1)); -- concatena para el desplazamiento
14         END IF;                             -- en caso contrario, retiene los datos
15        END IF;
16 END PROCESS;
17 END vhdl;
```

FIGURA 7-79 Registro de entrada en serie/salida en serie mediante el uso de VHDL.

REGISTRO PISO EN AHDL

La figura 7-80 lista un registro tipo entrada en paralelo/salida en serie (PISO) de cuatro bits. En la línea 8 se crea el registro llamado *q* mediante el uso de cuatro FFs D y en la línea 11 se describe la salida en serie de *q0*. El registro cuenta con controles separados de carga (*carga*) en paralelo y desplazamiento (*desplaza*) en serie. En las líneas 12 a 15 se definen las funciones del registro. Si *carga* está en ALTO, se cargará en forma síncrona la entrada externa *datos[3..0]*. *Carga* tiene prioridad y debe estar en BAJO para desplazar en serie el contenido del registro en cada PGT de *clk*, cuando *desplaza* está en ALTO. En la línea 13 se crea el patrón para desplazar datos a la derecha mediante la concatenación. Observe que un nivel BAJO constante será la entrada de datos en serie para una operación de desplazamiento. Si ni *carga* ni *desplaza* están en ALTO, el registro retendrá el valor de datos actual (línea 14). La figura 7-81 muestra los resultados de la simulación.

```
1  SUBDESIGN  fig7_80
2  (
3         clk, desplaza, carga     :INPUT;
4         datos[3..0]              :INPUT;
5         serial_sal               :OUTPUT;
6   )
7  VARIABLE
8         q[3..0]                  :DFF;
9  BEGIN
10        q[ ].clk = clk;
11        serial_sal = q0.q;             -- envía último bit del registro como salida
12        IF (carga ==VCC)  THEN  q[3..0].d = datos[3..0]; -- carga en paralelo
13        ELSIF (desplaza == VCC) THEN q[3..0].d = (GND, q[3..1].q); -- desplaza
14        ELSE  q[3..0].d = q[3..0].q;                               -- retiene
15        END IF;
16 END;
```

FIGURA 7-80 Registro de entrada en paralelo/salida en serie mediante el uso de AHDL.

AHDL

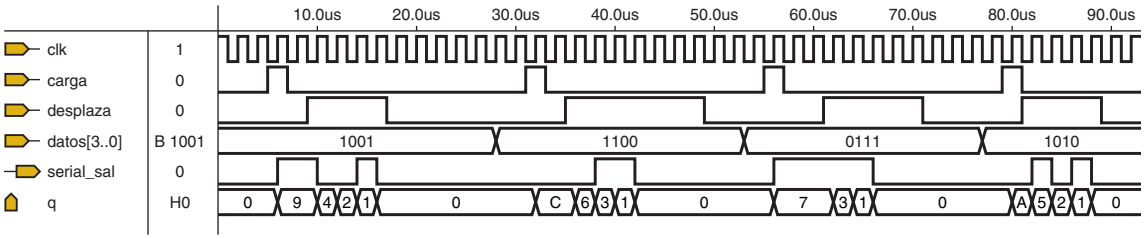


FIGURA 7-81 Simulación de un registro PISO.

VHDL

REGISTRO PISO EN VHDL

La figura 7-82 lista un registro tipo entrada en paralelo/salida en serie (PISO) de cuatro bits. En la línea 11 se crea el registro con la declaración de la variable *q* y en la línea 13 se describe la salida en serie de *q(0)*. El registro tiene controles separados de carga (*carga*) y desplazamiento en serie (*desplaza*). En las líneas 14 a 18 se definen las funciones del registro. Si *carga* está en ALTO, la entrada externa *datos* se cargará en forma síncrona. *Carga* tiene prioridad y debe estar en BAJO para desplazar en serie el contenido del registro en cada PGT de *clk* cuando *cambia* está en ALTO. En la línea 16 se crea el patrón para desplazar datos a la derecha mediante la concatenación. Observe que la entrada de datos en serie será un nivel BAJO constante para una operación de desplazamiento. Si ni *carga* ni *desplaza* están en ALTO, el registro retendrá el valor actual de datos mediante la operación implícita de VHDL. La figura 7-81 muestra los resultados de la simulación.

```
1  ENTITY  fig7_82  IS
2  PORT  (
3       clk, desplaza, carga          :IN BIT;
4       datos                         :IN BIT_VECTOR (3 DOWNTO 0);
5       serial_sal                    :OUT BIT
6  );
7  END  fig7_82;
8  ARCHITECTURE  vhdl  OF  fig7_82  IS
9  BEGIN
10 PROCESS  (clk)
11      VARIABLE  q          :BIT_VECTOR (3 DOWNTO 0);
12      BEGIN
13      serial_sal  <=  q(0);              -- envía último bit del registro como salida
14      IF  (clk'EVENT AND clk = '1')  THEN
15           IF  (carga = '1') THEN q := datos; -- carga en paralelo
16           ELSIF  (desplaza = '1') THEN  q:= ('0' & q[3 DOWNTO 1)); -- desplaza
17           ENDIF;                              -- en caso contrario retiene
18      END IF;
19 END PROCESS;
20 END vhdl;
```

FIGURA 7-82 Registro de entrada en paralelo/salida en serie mediante el uso de VHDL.

EJEMPLO 7-24

Suponga que deseamos diseñar un registro de desplazamiento de cuatro bits universal mediante HDL, el cual tenga cuatro modos síncronos de operación: Retener datos, Desplazar a la izquierda, Desplazar a la derecha y Carga en paralelo. Dos bits de entrada seleccionarán la operación que deba realizarse en cada flanco de subida del reloj. Para implementar un registro de desplazamiento podemos utilizar código

estructural para describir una cadena de flip-flops. Para hacer que el registro de desplazamiento sea eficiente y nos permita desplazar a la izquierda o a la derecha, o realizar la carga en paralelo, este archivo se haría bastante largo y por ende difícil de leer y comprender utilizando los métodos estructurales. Un enfoque mucho más conveniente es utilizar los métodos más abstractos e intuitivos disponibles en HDL para describir el circuito en forma concisa. Para ello debemos desarrollar una estrategia para crear la acción de desplazamiento. El concepto es muy similar al que se presenta en el ejemplo 7-16, en donde se cableó un chip de registro de flip-flop D (74174) para formar un registro de desplazamiento. En vez de pensar en el registro de desplazamiento como una cadena en serie de flip-flops, debemos considerarlo como un registro en paralelo cuyo contenido se va a transferir en paralelo a un conjunto de bits que está desplazado por una posición de un bit. La figura 7-76 demuestra el concepto de cada transferencia necesaria en este diseño.

Solución

Un primer paso muy razonable es definir una entrada de dos bits llamada *modo*, con la cual podemos especificar el modo 0, 1, 2 o 3. El siguiente reto es decidir cómo elegir entre las cuatro operaciones mediante el uso de HDL. Aquí pueden funcionar varios métodos. Se eligió la estructura CASE debido a que nos permite elegir un conjunto distinto de instrucciones en HDL para cada uno de todos los posibles valores de modo. No hay prioridad asociada con la comprobación de la configuración de los modos existentes o de los intervalos traslapados de números de modo, por lo que no necesitamos las ventajas de la instrucción IF/ELSE. En las figuras 7-83 y 7-84 se muestran las soluciones en HDL. En cada una de las soluciones se definen las mismas entradas y salidas: un reloj, cuatro bits de datos de carga en paralelo, un bit individual para la entrada en serie al registro, dos bits para la selección del modo y cuatro bits de salida.

```
1   SUBDESIGN fig7 83
2   (
3      reloj         :INPUT;
4      dent[3..0]    :INPUT;    datos en paralelo de entrada
5      ser ent       :INPUT;    datos en serie de entrada, de izquierda o derecha
6      modo [1..0]   :INPUT;    Selección de MODO: 0 = retención,
                                 1 = derecha, 2=izquierda, 3=carga
7      q[3..0]       :OUTPUT;
8   )
9   VARIABLE
10     ff[3..0] :DFF;           define el conjunto de registros
11  BEGIN
12     ff[].clk = reloj;        reloj síncrono
13     CASE modo[] IS
14        WHEN 0  => ff[].d     = ff[].q;          retiene desplazamiento
15        WHEN 1  => ff[2..0].d = ff[3..1].q);     desplazamiento a la derecha
16                   ff[3].d    = ser ent;         nuevos datos provenientes
                                                    de la izquierda
17        WHEN 2  => ff[3..1].d = ff[2..0].q;      desplazamiento a la izquierda
18                   ff[0].d    = ser ent;         nuevos datos provenientes
                                                    de la derecha
19        WHEN 3  => ff[].d     = dent[];          carga en paralelo
20     END CASE;
21     q[] = ff[];                 actualiza las salidas
22  END;
```

FIGURA 7-83 Registro de desplazamiento universal en AHDL.

```
1   ENTITY  fig7_84  IS
2   PORT  (
3        reloj       :IN BIT;
4        dent        :IN BIT_VECTOR (3 DOWNTO 0);      -- datos en paralelo de entrada
5        ser_ent     :IN BIT;                          -- datos en serie en Izq o Der
6        modo        :IN INTEGER RANGE 0 TO 3;         -- 0=retención  1=der  2=izq  3=carga
7        q           :OUT BIT_VECTOR (3 DOWNTO 0));
8   END  fig7_84;
9   ARCHITECTURE a  OF  fig7_84  IS
10   BEGIN
11       PROCESS (reloj)                               -- responde al reloj
12       VARIABLE     ff  :BIT_VECTOR  (3 DOWNTO 0);
13       BEGIN
14         IF (reloj'EVENT  AND  reloj = '1') THEN
15            CASE  modo  IS
16                 WHEN 0 =>  ff  :=  ff;                          -- retiene los datos
17                 WHEN 1 =>  ff(2 DOWNTO 0)   :=  ff(3 DOWNTO 1); -- desplaza a la derecha
18                            ff(3) := ser_ent;
19                 WHEN 2 =>  ff(3 DOWNTO 1)   :=  ff(2 DOWNTO 0); -- desplaza a la izquierda
20                            f(0) := ser_ent;
21                 WHEN 3 =>  ff := dent;                          -- carga en paralelo
22            END CASE;
23        END IF;
24       q <= ff;                                -- actualiza las salidas
25       END PROCESS;
26  END a;
```

FIGURA 7-84 Registro de desplazamiento universal mediante el uso de VHDL.

AHDL

SOLUCIÓN EN AHDL

La solución en AHDL de la figura 7-83 utiliza un registro de flip-flops D, los cuales se declaran mediante el nombre *ff* en la línea 10 para representar el estado actual del registro. Debido a que todos los flip-flops necesitan recibir pulsos de reloj al mismo tiempo (en forma síncrona), todas las entradas de reloj se asignan a *reloj* en la línea 12. La instrucción CASE selecciona una configuración de transferencia distinta para cada valor de las entradas *modo*. El modo 0 (retener datos) utiliza una transferencia en paralelo directa del estado actual a las mismas posiciones de bit en las entradas *D*, para producir el estado SIGUIENTE idéntico. El modo 1 (desplazar a la derecha), que se describe en las líneas 15 y 16, transfiere los bits 3, 2 y 1 a las posiciones de bit 2, 1 y 0, en forma respectiva, y carga el bit 3 de la entrada serial. El modo 2 (desplazar a la izquierda) realiza una operación similar en el sentido opuesto (vea las líneas 17 y 18). El modo 3 (carga en paralelo) transfiere el valor en las entradas de datos en paralelo para que se conviertan en el estado SIGUIENTE del registro. El código crea el circuito que selecciona una de estas operaciones lógicas en el registro actual y los datos apropiados se transfieren a las terminales de salida en el siguiente pulso de reloj. Este código se puede reducir si combinamos las líneas 15 y 16 en una sola instrucción que concatene *ser_ent* con los tres bits de datos y los agrupe como un conjunto de cuatro bits. La instrucción que puede sustituir las líneas 15 y 16 es:

```
WHEN 1   =>   ff[ ].d   = (ser ent, ff[3..1].q);
```

Las líneas 17 y 18 también pueden sustituirse por:

```
WHEN 2   =>   ff[ ].d = (ff[2..0].q, ser ent);
```

SOLUCIÓN EN VHDL

La solución en VHDL de la figura 7-84 define una variable interna por el nombre *ff* en la línea 12, la cual representa el estado actual del registro. Debido a que todas las operaciones de transferencia necesitan llevarse a cabo en respuesta a un flanco de subida del reloj, se utiliza un bloque PROCESS en el cual se especifica *reloj* en la lista de sensibilidad. La instrucción CASE selecciona una configuración de transferencia distinta para cada valor de las entradas *modo*. El modo 0 (retener datos) utiliza una transferencia en paralelo directa del estado actual a las mismas posiciones de bit para producir el estado SIGUIENTE idéntico. El modo 1 (desplazar a la derecha) transfiere los bits 3, 2 y 1 a las posiciones de bit 2, 1 y 0, en forma respectiva (línea 17) y carga el bit 3 de la entrada en serie (línea 18). El modo 2 (desplazar a la izquierda) realiza una operación similar en el sentido opuesto. El modo 3 (carga en paralelo) transfiere el valor en las entradas de datos en paralelo al estado SIGUIENTE del registro. Después de elegir una de estas operaciones en el registro actual, los datos se transfieren a las terminales de salida en la línea 24. Este código puede reducirse si combinamos las líneas 17 y 18 en una sola instrucción que concatene *ser_ent* con los tres bits de datos y los agrupe como un conjunto de cuatro bits. La instrucción que puede sustituir las líneas 17 y 18 es:

```
WHEN  1  =>  ff  :=  ser ent & ff(3 DOWNTO 1);
```

Las líneas 19 y 20 también pueden sustituirse por:

```
WHEN 2  => ff  :=  ff(2 DOWNTO 0)  & ser ent;
```

PREGUNTAS DE REPASO

1. Escriba una expresión en HDL para implementar un desplazamiento a la izquierda de un arreglo de ocho bits llamado *reg[7..0]* con la entrada en serie *dat*.
2. ¿Por qué es necesario recargar los datos actuales durante el modo de retención de datos en un registro de desplazamiento?

7-23 CONTADORES DE ANILLO EN HDL

En la sección 7-20 utilizamos un registro de desplazamiento para crear un contador que circula un solo nivel lógico activo a través de todos sus flip-flops. A este tipo de registro se le conoce como contador de anillo. Una característica de los contadores de anillo es que el módulo es igual al número de flip-flops en el registro y, por lo tanto, siempre hay muchos estados sin usar e inválidos. Ya hemos visto las formas de describir a los contadores mediante el uso de la instrucción CASE para especificar las transiciones del estado PRESENTE y del estado SIGUIENTE. En esos ejemplos nos hicimos cargo de los estados inválidos al incluirlos bajo "otros" (OTHERS). Este método también funciona para los contadores de anillo. No obstante, en esta sección veremos una manera más intuitiva para describir los contadores de desplazamiento.

Estos métodos utilizan las mismas técnicas descritas en la sección 7-22 para que el registro se desplace una posición en cada pulso de reloj. La principal característica de este código es el método para completar el "anillo" al dividir la línea *ser_ent* del registro de desplazamiento. Con un poco de planeación también deberíamos ser capaces de asegurar que el contador llegue en algún momento a la secuencia deseada, sin importar en qué estado se encuentre al principio. Para este ejemplo recreamos la operación del contador de anillo cuyo diagrama de estado se muestra en la figura 7-70(d). Para poder hacer que este contador inicie en forma automática sin utilizar entradas asíncronas, controlamos la línea *ser_ent* del registro de desplaza-

miento mediante el uso de una instrucción IF/ELSE. Cada vez que detectamos que los tres bits superiores están en BAJO, suponemos que el bit de menor orden está en ALTO y en el siguiente ciclo de reloj queremos desplazar un nivel ALTO hacia *ser_ent*. Para todos los demás estados (válidos e inválidos) desplazamos un nivel BAJO. Sin importar el estado con el que se inicialice el contador, en un momento dado se llenará con ceros; en ese momento nuestra lógica desplazará un nivel ALTO para comenzar la secuencia de anillo.

AHDL

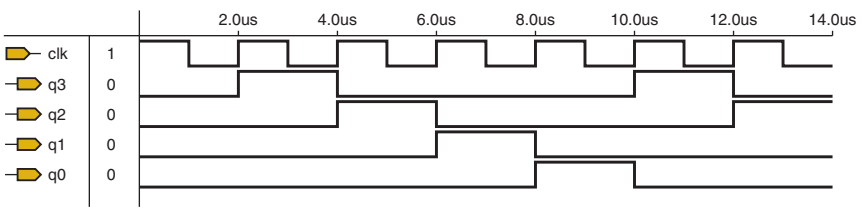
CONTADOR DE ANILLO EN AHDL

Para estos momentos, el código de AHDL que se muestra en la figura 7-85 le deberá ser familiar. Las líneas 11 y 12 controlan la entrada en serie mediante el uso de la estrategia que acabamos de describir. Observe el uso del operador de doble signo de igualdad (==) en la línea 11. Este operador evalúa si las expresiones en cada lado son iguales o no. Recuerde que el operador de signo de igualdad (=) asigna (es decir, conecta) un objeto a otro. La línea 14 implementa la acción de desplazamiento a la derecha que describimos en la sección anterior. La figura 7-86 muestra los resultados de la simulación.

```
1   SUBDESIGN fig7 85
2   (
3      clk          :INPUT;
4      q[3..0]      :OUTPUT;
5   )
6   VARIABLE
7      ff[3..0]     :DFF;
8      ser ent      :NODE;
9   BEGIN
10    ff[].clk = clk;
11    IF ff[3..1].q == B"000" THEN ser ent = VCC;    inicio automático
12    ELSE ser ent = GND;
13    END IF;
14    ff[3..0].d = (ser ent, ff[3..1].q);         desplaza a la derecha
15    q[] = ff[];
16  END;
```

FIGURA 7-85 Contador de anillo de cuatro bits en AHDL.

FIGURA 7-86
Simulación del contador de anillo en HDL.



VHDL

CONTADOR DE ANILLO EN VHDL

Para estos momentos, el código de VHDL que se muestra en la figura 7-87 deberá serle familiar. Las líneas 12 y 13 controlan la entrada en serie mediante el uso de la estrategia que acabamos de describir. La línea 16 implementa la acción de desplazamiento a la derecha que describimos en la sección anterior. Los resultados de la simulación se muestran en la figura 7-86.

```
 1   ENTITY  fig7_87  IS
 2  PORT (    clk            :IN BIT;
 3            q              :OUT BIT_VECTOR (3 DOWNTO 0));
 4   END  fig7_87;
 5
 6   ARCHITECTURE  vhdl  OF  fig7_87 IS
 7  SIGNAL  ser_ent          :BIT;
 8  BEGIN
 9  PROCESS (clk)
10       VARIABLE  ff        :BIT_VECTOR (3 DOWNTO 0);
11       BEGIN
12            IF (ff (3 DOWNTO 1) = "000") THEN ser_ent <= '1'; -- inicio automático
13            ELSE  ser_ent  <= '0';
14            END IF;
15            IF (clk'EVENT  AND  clk = '1') THEN
16                 ff(3 DOWNTO 0) := (ser_ent & ff(3 DOWNTO 1)); -- desplaza a la derecha
17            END IF;
18       q  <=  ff;
19   END PROCESS;
20   END vhdl;
```

FIGURA 7-87 Contador de anillo de cuatro bits en VHDL.

PREGUNTAS DE REPASO

1. ¿Qué significa que un contador de anillo inicie en forma automática?
2. ¿Qué líneas de la figura 7-85 aseguran que el contador de anillo inicie en forma automática?
3. ¿Qué líneas de la figura 7-87 aseguran que el contador de anillo inicie en forma automática?

7-24 MONOESTABLES EN HDL

Otro circuito importante que hemos estudiado es el circuito monoestable (*one-shot*). Podemos aplicar el concepto de un contador para implementar un **monoestable digital** mediante el uso de HDL. Si recuerda del capítulo 5, los monoestables son dispositivos que producen un pulso de una anchura predefinida cada vez que se activa la entrada de disparo. Un monoestable *no redisparable* ignora la señal de disparo siempre y cuando la salida de pulso siga estando activa. Un monoestable *redisparable* inicia un pulso en respuesta a una señal de disparo y limpia el temporizador de pulso interno cada vez que ocurre un flanco de disparo subsiguiente, antes de completar el pulso. El primer ejemplo que investigaremos es un monoestable digital no redisparable, que se dispara por un nivel ALTO. Los monoestables que estudiamos en el capítulo 5 utilizaban una resistencia y un capacitor como el mecanismo de sincronización de pulso interno. Para poder crear un monoestable mediante las técnicas de HDL, utilizamos un contador de cuatro bits para determinar la anchura del pulso. Las entradas son una señal de reloj, disparo, borrar y el valor de la anchura del pulso. La única salida es la de pulso, *Q*. La idea es bastante simple. Cada vez que se detecte un disparo hay que hacer que el pulso cambie a ALTO y se cargue un contador descendente con un número proveniente de las entradas de anchura de pulso. Entre mayor sea el número, más tiempo tardará el conteo hasta cero. La ventaja de este circuito es que la anchura del pulso puede ajustarse con facilidad si se cambia el valor que se cargó en el contador. A medida que vaya leyendo las siguientes secciones, considere la siguiente pregunta: "¿Qué hace que este circuito sea no redisparable y qué hace que se dispare por nivel?"

AHDL

MONOESTABLES EN AHDL

La figura 7-88 muestra la descripción en AHDL de un monoestable no redisparable, sensible al nivel. En la línea 8 se crea un registro de cuatro flip-flops y sirve como el contador para realizar el conteo descendente durante el pulso. La señal *reloj* se conecta en paralelo a todos los flip-flops en la línea 10. La función de borrado (reset) se implementa mediante la conexión de la línea de control *reset* en forma directa a la entrada *clrn* asíncrona de cada flip-flop en la línea 11. Después de estas asignaciones la primera condición que se prueba es el disparo. Si se activa (ALTO) en cualquier momento mientras que el valor del conteo sea 0 (es decir, que termine el pulso anterior), entonces se cargará el valor de retraso en el contador. En la línea 14 se prueba para ver si el pulso ya terminó; para ello se comprueba si el contador llegó hasta cero o no. Si llegó a cero, entonces el contador no debe reciclarse sino permanecer en cero. Si el conteo no está en cero, entonces debe estar contando, por lo que la línea 15 prepara los flip-flops para que se decrementen en el siguiente pulso de reloj. Por último, la línea 17 genera el pulso de salida. Esta expresión booleana puede definirse de la siguiente manera: "Hacer que el pulso (*Q*) sea ALTO cuando el *conteo* sea cualquier número distinto de cero."

FIGURA 7-88
Monoestable no redisparable en AHDL.

```
1   SUBDESIGN fig7 88
2   (
3      reloj, disparo, reset     : INPUT;
4      retraso[3..0]             : INPUT;
5      q                         : OUTPUT;
6   )
7   VARIABLE
8      conteo[3..0]    : DFF;
9   BEGIN
10     conteo[].clk = reloj;
11     conteo[].clrn = reset;
12     IF disparo & conteo[].q == b"0000" THEN
13           conteo[].d = retraso[];
14     ELSIF conteo[].q == B"0000" THEN conteo[].d = B"0000";
15     ELSE conteo[].d = conteo[].q   1;
16     END IF;
17     q = conteo[].q != B"0000";     crea el pulso de salida
18  END;
```

VHDL

MONOESTABLES SIMPLES EN VHDL

La figura 7-89 muestra la descripción en VHDL de un circuito digital monoestable no redisparable, sensible al nivel. Las entradas y las salidas se muestran en las líneas 3 a 5, como se describió antes. En la descripción de la arquitectura se utiliza un bloque PROCESS (línea 11) para responder a cualquiera de las dos entradas: reloj o *reset*. Dentro de este bloque PROCESS se utiliza una variable para representar el valor en el contador. La entrada que debería tener precedencia sobre las demás es la señal *reset*. Ésta se evalúa primero (línea 14) y si está activa, *conteo* se borra de inmediato. Si *reset* no está activa se evalúa la línea 15 y se busca un flanco de subida en *reloj*. La línea 16 comprueba el disparo. Si se activa en cualquier momento mientras que el valor del conteo sea 0 (es decir, si terminó el pulso anterior), entonces se carga el valor de la anchura en el contador. En la línea 18 se evalúa si el pulso terminó mediante una comprobación para ver si el contador llegó a cero. Si es así, entonces el contador no deberá reciclarse sino permanecer en cero.

```
 1   ENTITY fig7 89 IS
 2   PORT (
 3         reloj, disparo, reset    :IN BIT;
 4         retraso                  :IN INTEGER RANGE 0 TO 15;
 5         q                        :OUT BIT
 6         );
 7   END fig 7 89;
 8
 9   ARCHITECTURE vhdl OF fig7 89 IS
10   BEGIN
11      PROCESS (reloj, reset)
12      VARIABLE conteo       : INTEGER RANGE 0 TO 15;
13      BEGIN
14         IF restablecer = '0' THEN conteo := 0;
15         ELSIF (reloj EVENT AND reloj = '1' ) THEN
16            IF disparo = '1' AND conteo = 0 THEN
17               conteo := retraso;                          carga el contador
18            ELSIF conteo = 0 THEN conteo := 0;
19            ELSE conteo := conteo   1;
20            END IF;
21         END IF;
22         IF conteo /= 0 THEN q <= '1';
23         ELSE q <= '0';
24         END IF;
25      END PROCESS;
26   END vhdl;
```

FIGURA 7-89 Monoestable no redisparable en VHDL.

Si *conteo* no está en cero significa que debe estar contando, por lo que la línea 19 prepara los flip-flops para que se decrementen en el siguiente pulso de reloj. Por último, las líneas 22 y 23 generan el pulso de salida. Esta expresión booleana puede definirse de la siguiente manera: "Hacer que el pulso (q) sea ALTO cuando el conteo sea cualquier número distinto de cero".

Ahora que revisamos el código que describe a este monoestable, vamos a evaluar su rendimiento. La conversión de un circuito analógico tradicional en uno digital, por lo general, ofrece ciertas ventajas y ciertas desventajas. En un CI monoestable estándar, el pulso de salida empieza justo después del disparo. Para el monoestable digital que describimos aquí, el pulso de salida comienza en el siguiente flanco de reloj y dura todo el tiempo que el contador sea mayor que cero. Esta situación se muestra en la figura 7-90 durante los primeros ms de la simulación. Observe que el disparo cambia a nivel alto casi 0.5 ms después de que responde la salida *q*. Si ocurre otro evento de disparo mientras el contador está contando hacia abajo (como el que ocurre justo antes de los 3 ms), se ignora. Ésta es la característica no redisparable.

Otro punto a recalcar para este monoestable digital es que el pulso de disparo debe ser lo bastante largo como para verse como un nivel ALTO en el flanco de subida del reloj. Casi en la marca de los 4.5 ms se produce un pulso en la entrada de disparo, pero cambia a BAJO antes del flanco de subida del reloj. Este circuito *no* responde a este evento de entrada. Justo después de los 5 ms la entrada de disparo cambia a ALTO y permanece ahí. El pulso dura exactamente 6 ms, pero debido a que la entrada de disparo permanece en ALTO, responde con otro pulso de salida un ciclo de reloj después. La razón de esta situación es que este circuito se dispara por nivel en vez de dispararse por flanco, como la mayoría de los CIs monoestables convencionales.

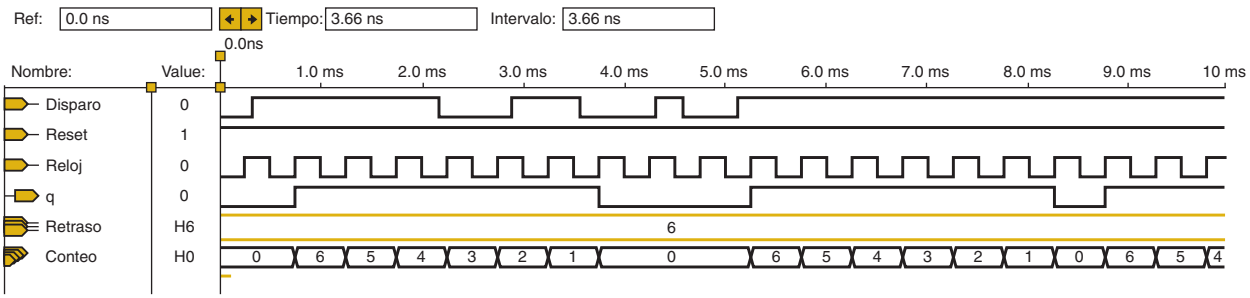


FIGURA 7-90 Simulación de los monoestables no redisparables.

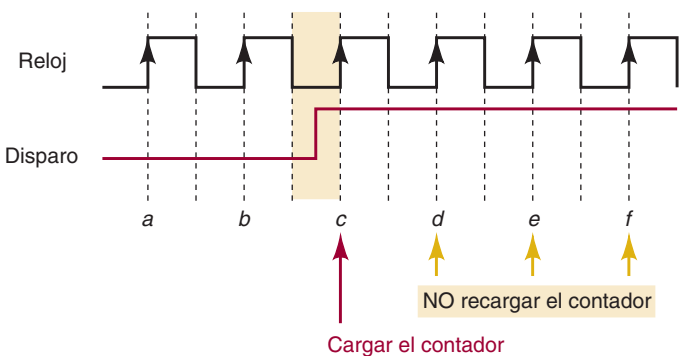
Monoestables redisparables activados por flanco en HDL

Muchas aplicaciones de los monoestables requieren que el circuito responda a un flanco en vez de a un nivel. ¿Cómo podemos utilizar código de HDL para hacer que el circuito responda una vez a cada transición positiva en su entrada de disparo? La técnica que describiremos aquí se conoce como *captura de flancos* y ha sido una herramienta muy útil en la programación de microcontroladores durante años. Como veremos, también es muy útil para describir el disparo por flanco para un circuito digital mediante el uso de HDL. En esta sección ilustraremos un ejemplo de un monoestable redisparable y también demostraremos la captura de flancos, que puede ser útil en muchas otras situaciones.

La operación general de este monoestable redisparable requiere que responda a un flanco de subida de la entrada de disparo. Tan pronto como se detecta el flanco, debe empezar a sincronizar el pulso. En el monoestable digital esto significa que debe cargar el contador lo más pronto posible después del flanco de disparo y debe empezar a contar en forma descendente hasta cero. Si ocurre otro evento de disparo (flanco de subida) antes de que termine el pulso, el contador se recarga de inmediato y la sincronización del pulso empieza de nuevo a partir del principio, con lo cual se sostiene el pulso. Si se activa la señal de reset en cualquier punto se forzará al contador a regresar a cero y se terminará el pulso. La anchura mínima del pulso de salida es tan sólo el número que se aplica a la entrada, multiplicado por el periodo de reloj.

La figura 7-91 demuestra la estrategia de captura de flancos para un monoestable. En cada flanco activo del reloj hay dos partes importantes de información. La primera es el estado de la entrada *disparo ahora* y la segunda es el estado de la entrada *disparo* cuando ocurrió el último flanco activo de reloj. Empiece con el punto *a* en el diagrama de la figura 7-91 y determine estos dos valores, después avance al punto *b* y así en lo sucesivo. Cuando complete esta tarea llegará a la conclusión de que, en el punto *c* se ha obtenido un resultado único. El *disparo* está en ALTO ahora pero estaba en BAJO durante el último flanco activo del reloj. Éste es el punto en el que detectamos el evento de *disparo* por flanco.

FIGURA 7-91
Detección de flancos.



Para poder saber cuál era el *disparo* en el último flanco activo del reloj, el sistema debe recordar el último valor que *tenía el disparo en ese punto*. Para ello hay que almacenar el valor del bit de disparo en un flip-flop. Recuerde que vimos un concepto similar en el capítulo 5, cuando hablamos sobre el uso de un flip-flop para detectar una secuencia. El código para un monoestable se escribe de forma que el contador se cargue sólo después de detectar un flanco de subida en la entrada *disparo*.

MONOESTABLE REDISPARABLE EN AHDL, DISPARADO POR FLANCO

Las primeras cinco líneas de la figura 7-92 son idénticas al ejemplo anterior del monoestable no redisparable. En AHDL, la única forma de recordar un valor que se obtuvo en el pasado es almacenarlo en un flip-flop. En esta sección utilizaremos un flip-flop llamado *disp_era* (línea 9) para almacenar el valor que había en el disparo, durante el último flanco activo del reloj. Este flip-flop se conecta de manera que el disparo esté en su entrada *D* (línea 14) y el reloj esté conectado a su entrada *clk* (línea 13). La salida *Q* de *disp_era* recuerda el valor de *disparo* justo hasta el siguiente flanco de reloj. En este punto utilizamos la línea 16 para evaluar si se ha producido un flanco de disparo. Si *disparo* está en ALTO (ahora) pero estaba en BAJO (último ciclo de reloj), es tiempo de cargar el contador (línea 17). La línea 18 asegura que una vez que el *conteo* llegue a cero, permanecerá en cero hasta que se produzca un nuevo disparo. Si las decisiones permiten evaluar la línea 19, significa que hay un valor cargado en el contador y no es cero, por lo que necesita decrementarse. Por último, el pulso de salida se cambia a ALTO cada vez que hay un valor distinto de 000 en el contador, como vimos antes.

FIGURA 7-92
Descripción de un circuito monoestable redisparable en AHDL, con disparo por flanco.

```
1   SUBDESIGN fig7 92
2   (
3      reloj, disparo, restablecer  : INPUT;
4      delay[3..0]                  : INPUT;
5      q                            : OUTPUT;
6   )
7   VARIABLE
8              conteo[3..0]         : DFF;
9              disp era             : DFF;
10  BEGIN
11     conteo[].clk = reloj;
12     conteo[].clrn = restablecer;
13     disp era.clk = reloj;
14     disp era.d = disparo;
15
16     IF disparo & !disp era.q THEN
17           conteo[].d = retraso[];
18     ELSIF conteo[].q == B"0000" THEN conteo[].d = B"0000";
19     ELSE conteo[].d = conteo[].q   1;
20     END IF;
21     q = conteo[].q != B"0000";
22  END;
```

DESCRIPCIÓN VHDL DE MONOESTABLE REDISPARABLE DISPARADO POR FLANCO

La descripción ENTITY en la figura 7-93 es idéntica al ejemplo anterior del monoestable no redisparable. De hecho, las únicas diferencias entre este ejemplo y el que se muestra en la figura 7-89 están relacionadas con la lógica del proceso de decisión. Cuando queremos recordar un valor en VHDL, debemos almacenarlo en una VARIABLE. Recuerde que podemos considerar un bloque PROCESS como una descripción de lo que ocurre cada vez que una señal en la lista de sensibilidad cambia de estado. Una VARIABLE retiene el último valor que se le asigna entre los tiempos en los que se invoca el proceso. En este sentido actúa como un flip-flop. Para el monoestable necesitamos almacenar un valor que nos indique en qué estado se encontraba el disparo en el último flanco activo del reloj. La línea 11 declara un bit variable para servir a este fin. La primera decisión (línea 13) es la de predominancia que comprueba y responde a la entrada *reset*. Observe que éste es un control asíncrono, ya que se evalúa antes de que se detecte el flanco del reloj en la línea 14. Esta línea determina que se ha producido un flanco de subida del reloj, y después se evalúa la lógica principal de este proceso entre las líneas 15 y 20.

Cuando se produce un flanco de reloj, existe una de tres condiciones:

1. Se ha producido un flanco de disparo y debemos cargar el contador.
2. El contador está en cero y necesitamos mantenerlo en cero.
3. El contador no está en cero y necesitamos contar uno hacia abajo.

```
1   ENTITY fig7 93 IS
2   PORT (   reloj, disparo, reset    : IN BIT;
3         retraso                     : IN INTEGER RANGE 0 TO 15;
4         q                           : OUT BIT);
5   END fig7 93;
6
7   ARCHITECTURE vhdl OF fig7 93 IS
8   BEGIN
9      PROCESS (reloj, reset)
10     VARIABLE conteo      : INTEGER RANGE 0 TO 15;
11     VARIABLE disp era    : BIT;
12     BEGIN
13        IF reset = '0' THEN conteo := 0;
14        ELSIF (reloj'EVENT AND reloj = '1' ) THEN
15           IF disparo = '1' AND disp era = '0' THEN
16              conteo := retraso;                carga el contador
17              disp era := '1';                  "recuerda" el flanco detectado
18           ELSIF conteo = 0 THEN conteo := 0;    retiene en 0
19           ELSE conteo := conteo   1;           decrementa
20           END IF;
21           IF disparo = '0' THEN disp era := '0';
22           END IF;
23        END IF;
24        IF conteo /= 0  THEN q <= '1';
25        ELSE q <= '0';
26        END IF;
27     END PROCESS;
28  END vhdl;
```

FIGURA 7-93 Monoestable redisparable en VHDL, con disparo por flanco.

Recuerde que es muy importante considerar el orden en el que se hacen las preguntas y las asignaciones en las instrucciones PROCESS de VHDL, ya que la *secuencia* afecta la operación del circuito que estamos describiendo. El código que actualiza la variable *disp_era* debe ocurrir después de la evaluación de su condición anterior. Por esta razón, las condiciones necesarias para detectar un flanco de subida en *disparo* se evalúan en la línea 15. Si se produjo un flanco, entonces se carga el contador (línea 16) y la variable se actualiza (línea 17) para recordar esto para la próxima vez. Si no se produjo un flanco de disparo, el código se retiene en cero (línea 18) o cuenta hacia abajo (línea 19). La línea 21 se asegura de que tan pronto como la entrada de disparo cambie a BAJO, la variable *disp_era* recuerde esto mediante la operación reset. Por último, las líneas 24 a 25 se utilizan para crear el pulso de salida durante el tiempo en el que el contador no sea cero.

Las dos mejoras que se hicieron en este monoestable, en comparación con el otro ejemplo, son el disparo por flanco y la característica de ser redisparable. La figura 7-94 evalúa las nuevas características de rendimiento. Observe en el primer ms del diagrama de sincronización que se detecta un flanco de disparo, pero la respuesta no es inmediata. El pulso de salida cambia a nivel alto en el siguiente flanco del reloj. Ésta es una desventaja para el monoestable digital. La característica de redisparable se demuestra casi en la marca de los 2 ms. Observe que *disparo* cambia a nivel alto y en el siguiente flanco de reloj el *conteo* empieza de nuevo en 5, con lo que se sostiene el pulso de salida. Observe además que aún después de que se termina el pulso de salida *q* y el *disparo* se encuentra todavía en ALTO, el circuito monoestable no dispara otro pulso ya que no se dispara por nivel, sino por flanco. En la marca de los 6 ms se produce un pulso de disparo corto pero se ignora debido a que no permanece en ALTO sino hasta el siguiente ciclo de reloj. Por otro lado, un pulso de disparo aún más corto que se produce justo después de la marca de los 7 ms dispara el monoestable, ya que está presente durante el flanco de subida del reloj. El pulso de salida resultante dura exactamente cinco ciclos de reloj, ya que no se producen otros disparos durante este periodo.

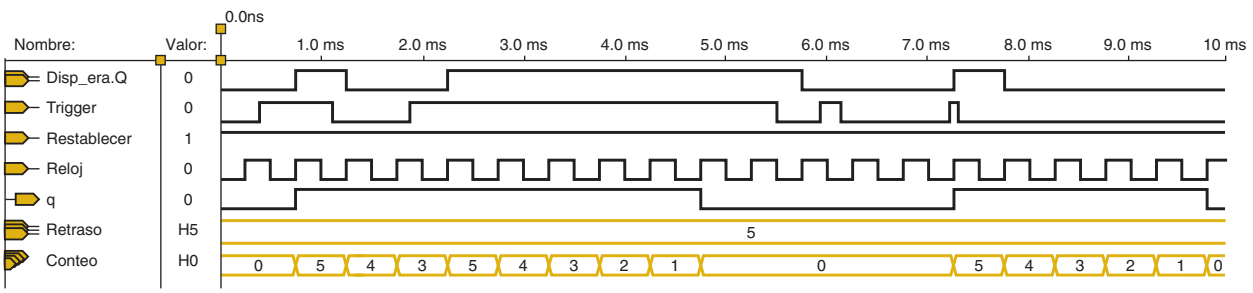


FIGURA 7-94 Simulación del monoestable redisparable, con disparo por flanco.

Para minimizar los efectos de la respuesta retrasada a los flancos de disparo y la posibilidad de dejar pasar los flancos de disparo que sean demasiado cortos, podemos mejorar este circuito de una forma muy sencilla. La frecuencia de reloj y el número de bits que se utilizan para cargar el valor de retraso pueden incrementarse para proporcionar el mismo rango de anchuras de pulso (con un control más preciso) al tiempo que se reduce la anchura mínima del pulso de disparo. Para poder curar este problema por completo, el monoestable debe responder en forma asíncrona a la entrada de disparo. Esto es posible tanto en AHDL como en VHDL, pero siempre se producirá un pulso que fluctúe en anchura de hasta un periodo de reloj.

PREGUNTAS DE REPASO

1. ¿Qué señal de entrada de control tiene la prioridad más alta para cada una de las descripciones de un circuito monoestable?
2. Nombre dos factores que determinen cuánto tiempo durará un pulso de un circuito monoestable digital.
3. Para los circuitos monoestables que se muestran en esta sección, ¿los contadores se cargan en forma síncrona o asíncrona?
4. ¿Cuál es la ventaja de cargar un contador en forma síncrona?
5. ¿Cuál es la ventaja de cargar el contador en forma asíncrona?
6. ¿Cuáles son las dos piezas de información necesarias para detectar un flanco?

RESUMEN PARTE 2

1. Hay numerosos registros de CI disponibles y pueden clasificarse dependiendo de si sus entradas son en paralelo (todos los bits se introducen en forma simultánea), en serie (un bit a la vez) o ambas. De igual forma, los registros pueden tener salidas en paralelo (todos los bits disponibles al mismo tiempo) o en serie (un bit disponible a la vez).
2. Un sistema lógico secuencial utiliza FFs, contadores y registros, además de compuertas lógicas. Sus salidas y la secuencia de las operaciones dependen de las entradas presentes y pasadas.
3. El diagnóstico de fallas en un sistema lógico secuencial comienza con la observación de la operación del sistema, seguida de un razonamiento analítico para determinar las posibles causas de cualquier falla y por último se realizan mediciones de prueba para aislar la verdadera falla.
4. Un contador de anillo es en realidad un registro de desplazamiento de N bits, que recircula un solo 1 en forma continua, actuando en consecuencia como un contador MOD-N. Un contador Johnson es un contador de anillo modificado que opera como un contador MOD-$2N$.
5. Para implementar los registros de desplazamiento con HDL se escriben descripciones personalizadas de su operación.
6. Es muy importante comprender el funcionamiento de los arreglos de bits/vectores de bits y su notación para describir las operaciones de los registros de desplazamiento.
7. Los contadores de registro de desplazamiento tales como los contadores Johnson y de anillo pueden implementarse con facilidad en HDL. Es fácil escribir las características de decodificación e inicio automático en la descripción.
8. Los monoestables digitales se implementan con un contador cargado con un valor de retraso cuando se detecta la entrada de disparo y cuenta en forma descendente hasta cero. Durante el tiempo del conteo descendente, el pulso de salida se mantiene en ALTO.
9. Mediante la colocación estratégica de las instrucciones de descripción de hardware, se puede hacer que los circuitos monoestables en HDL se disparen por flanco o por nivel, y que sean redisparables o no redisparables, ya que producen un pulso de salida que responde en forma síncrona o asíncrona al disparo.

TÉRMINOS IMPORTANTES PARTE 2

concatenación
contador de anillo
contador Johnson (contador de anillo torcido)
entrada en paralelo/salida en paralelo
entrada en paralelo/salida en serie
entrada en serie/salida en paralelo
entrada en serie/salida en serie
monoestable digital
registro de desplazamiento circulante
sistema lógico secuencial

PROBLEMAS

PARTE 1

SECCIÓN 7-1

B 7-1.* Agregue otro flip-flop llamado *E* al contador de la figura 7-1. La señal de reloj es una onda cuadrada de 8-MHz.

(a) ¿Cuál será la frecuencia en la salida *E*? ¿Cuál será el ciclo de trabajo de esta señal?

(b) Repita el inciso (a) si la señal de reloj tiene un ciclo de trabajo del 20 por ciento.

(c) ¿Cuál será la frecuencia en la salida *C*?

(d) ¿Cuál es el número MOD de este contador?

B 7-2. Dibuje un contador binario que convierta una señal de pulso de 64-kHz en una onda cuadrada de 1-kHz.

B 7-3.* Suponga que un contador binario de cinco bits comienza en el estado 00000. ¿Cuál será el conteo después de 144 pulsos de entrada?

B 7-4. A un contador de rizo de 10 bits se le aplica una señal de reloj de 256-kHz.

(a) ¿Cuál es el número MOD de este contador?

(b) ¿Cuál será la frecuencia en la salida del MSB?

(c) ¿Cuál será el ciclo de trabajo de la señal del MSB?

(d) Suponga que el contador empieza en cero. ¿Cuál será el conteo en hexadecimal después de 1000 pulsos de entrada?

SECCIÓN 7-2

7-5.* Un contador de rizo de cuatro bits se controla mediante una señal de reloj de 20-MHz. Dibuje las formas de onda en la salida de cada FF si cada uno tiene un t_{pd} = 20 ns. Determine cuáles estados del contador (si es que los hay) no se producirán debido a los retrasos de propagación.

7-6. (a) ¿Cuál es la máxima frecuencia de reloj que puede utilizarse con el contador del problema 7-5?

(b) ¿Cuál sería $f_{máx}$ si el contador se expandiera a seis bits?

SECCIONES 7-3 Y 7-4

B 7-7.* (a) Dibuje el diagrama del circuito para un contador síncrono MOD-32.

(b) Determine $f_{máx}$ para este contador si cada FF tiene un t_{pd} = 20 ns y cada compuerta tiene un t_{pd} = 10 ns.

B 7-8. (a) Dibuje el diagrama del circuito para un contador síncrono MOD-64.

(b) Determine $f_{máx}$ para este contador si cada FF tiene un t_{pd} = 20 ns y cada compuerta tiene un t_{pd} = 10 ns.

B 7-9.* Dibuje las formas de onda para todos los FFs en el contador de décadas de la figura 7-8(b), en respuesta a una frecuencia de reloj de 1-kHz. Muestre cualquier deformación que pudiera aparecer en cualquiera de las salidas de los FFs. Determine la frecuencia en la salida *D*.

B 7-10. Repita el problema 7-9 para el contador de la figura 7-8(a).

B 7-11.* Cambie las entradas para la compuerta NAND de la figura 7-9, de manera que el contador divida la frecuencia de entrada entre 50.

B 7-12. Dibuje un contador síncrono que produzca como salida una señal de 10-kHz cuando se aplica un reloj de 1-MHz.

---

* Encontrará las respuestas a los problemas marcados con asterisco al final del libro.

SECCIONES 7-5 Y 7-6

B 7-13.* Dibuje un contador descendente síncrono, MOD-32.

B 7-14. Dibuje un contador ascendente/descendente síncrono, MOD-16. La dirección de conteo se controla mediante *dir* (*dir* = 0 para contar hacia arriba).

A, F 7-15.* Determine la secuencia de conteo del contador ascendente/descendente en la figura 7-11, si la salida INVERSOR estuviera forzada en ALTO. Suponga que el contador empieza en 000.

7-16. Complete el diagrama de sincronización de la figura 7-95 para el contador preajustable de la figura 7-12. Observe que la condición inicial para el contador se proporciona en el diagrama de sincronización.

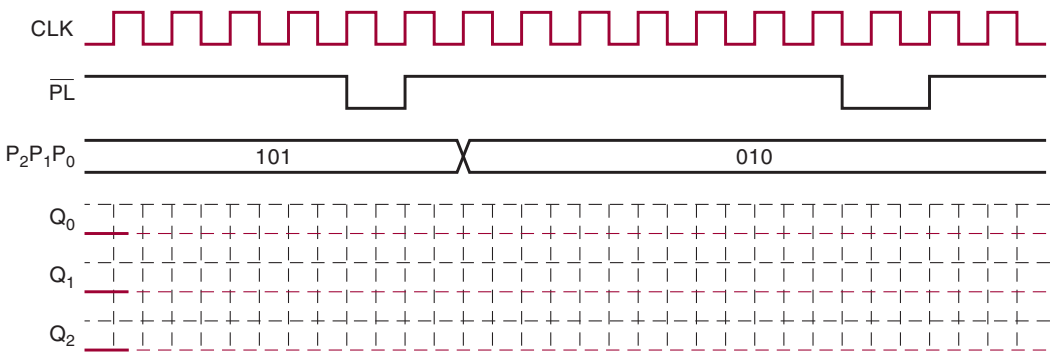


FIGURA 7-95 Diagrama de sincronización del problema 7-16.

SECCIÓN 7.7

7-17.* Complete el diagrama de sincronización de la figura 7-96 para un 74ALS161, al cual se le aplican las formas de onda de entrada que se indican. Suponga que el estado inicial es 0000.

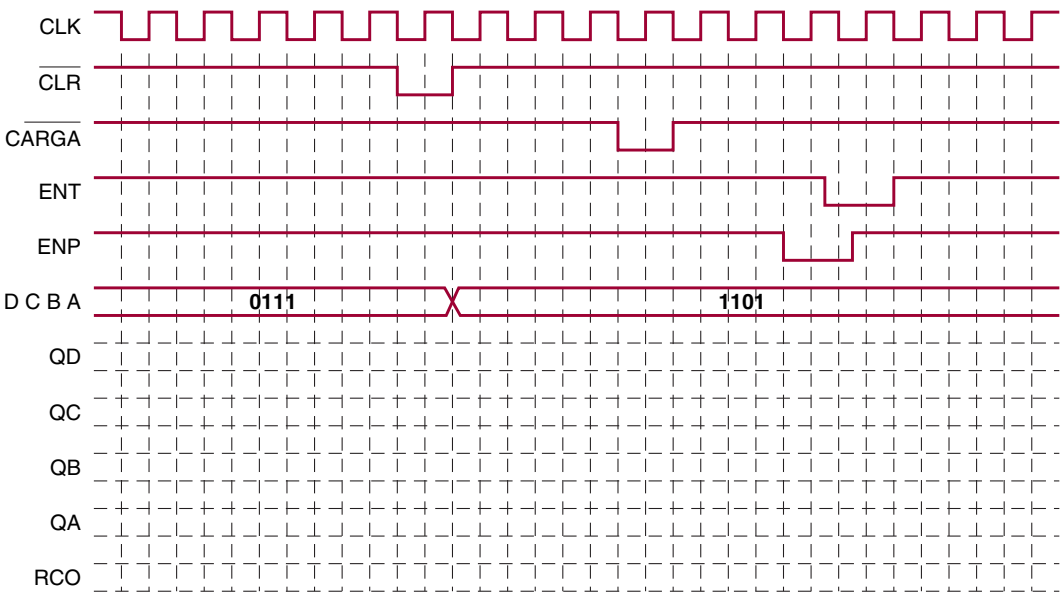


FIGURA 7-96 Diagrama de sincronización del problema 7-17.

7-18. Complete el diagrama de sincronización de la figura 7-97 para el 74ALS162, al cual se le aplican las formas de onda de entrada que se indican. Suponga que el estado inicial es 0000.

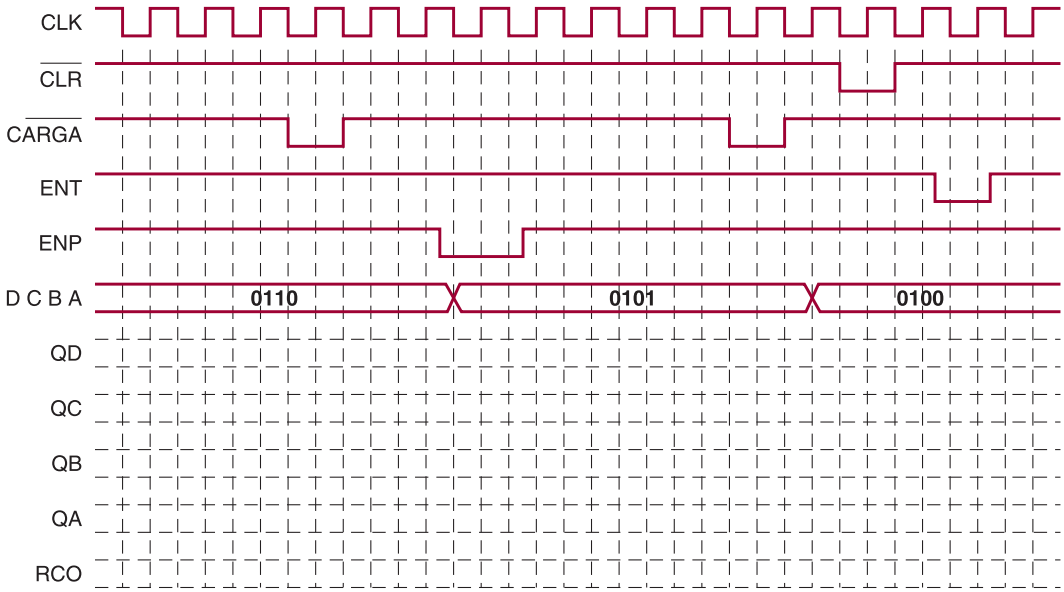


FIGURA 7-97 Diagrama de sincronización del problema 7-18.

7-19.* Complete el diagrama de sincronización de la figura 7-98 para un 74ALS190, al cual se le aplican las formas de onda de entrada que se indican. La entrada *DCBA* es 0101.

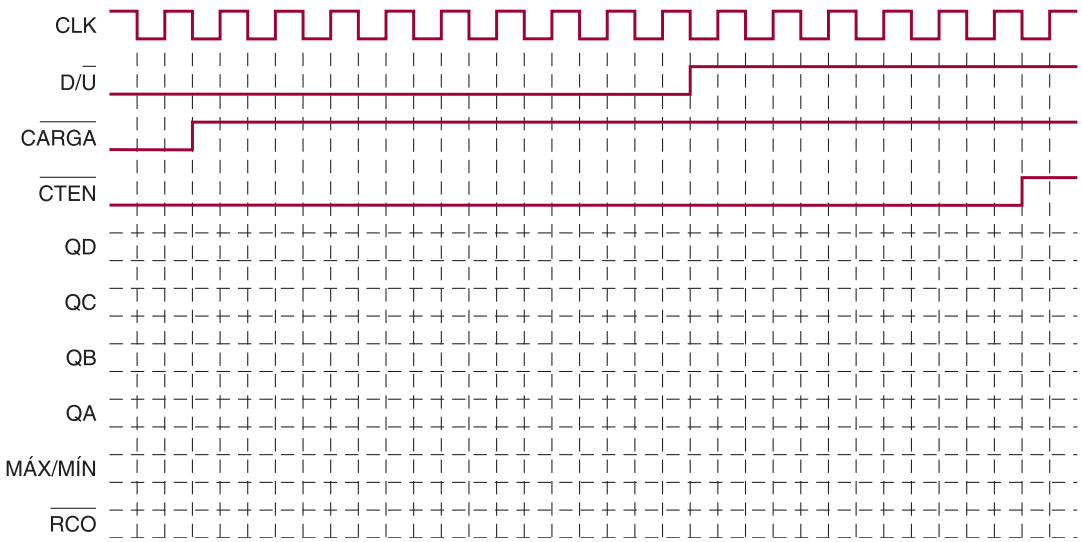


FIGURA 7-98 Diagrama de sincronización de los problemas 7-19 y 7-20.

7-20. Repita el problema 7-19 para un 74ALS191 y una entrada *DCBA* de 1100.

B 7-21.* Consulte el circuito del contador del CI en la figura 7-99(a):

(a) Dibuje el diagrama de transición de estados para las salidas *QD QC QB QA* del contador.

(b) Determine el módulo del contador.

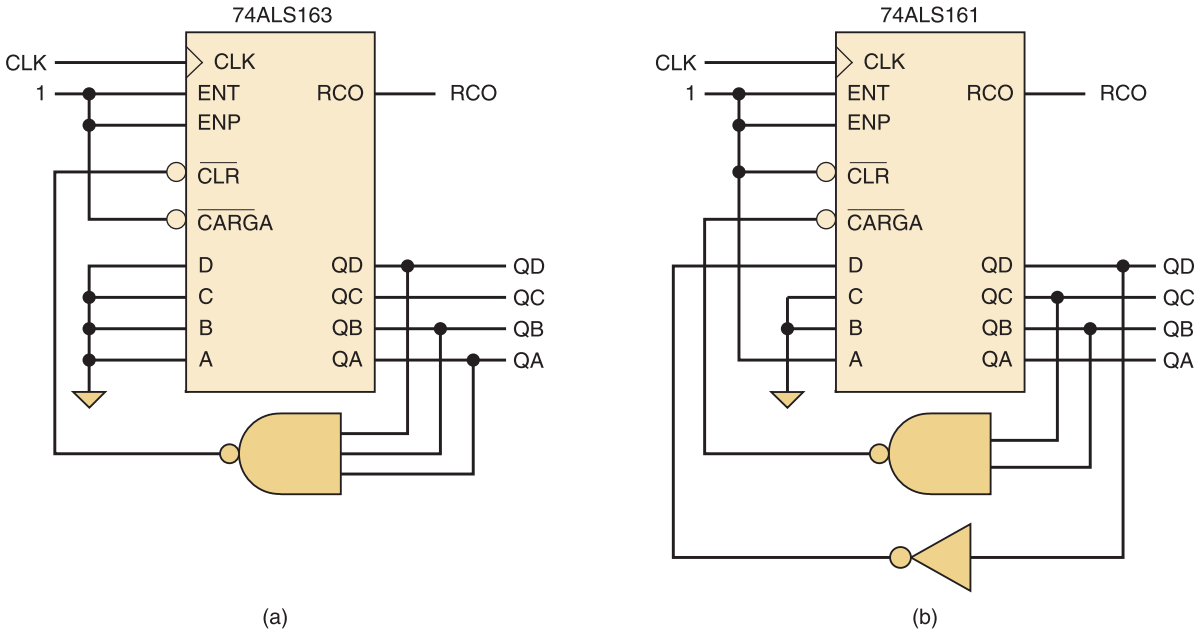


FIGURA 7-99 Problemas 7-21 y 7-22.

(c) ¿Cuál es la relación entre la frecuencia de salida del MSB con la frecuencia de entrada de *CLK*?

(d) ¿Cuál es el ciclo de trabajo de la forma de onda de salida del MSB?

B 7-22. Repita el problema 7-21 para el circuito contador de CI en la figura 7-99(b).

B 7-23.* Consulte el circuito contador de CI en la figura 7-100(a).

(a) Dibuje el diagrama de sincronización para las salidas *QD QC QB QA*.

(b) ¿Cuál es el módulo del contador?

(c) ¿Cuál es la secuencia de conteo? ¿Cuenta hacia arriba o hacia abajo?

(d) ¿Podemos producir el mismo módulo con un 74HC190? ¿Podemos producir la misma secuencia de conteo con un 74HC190?

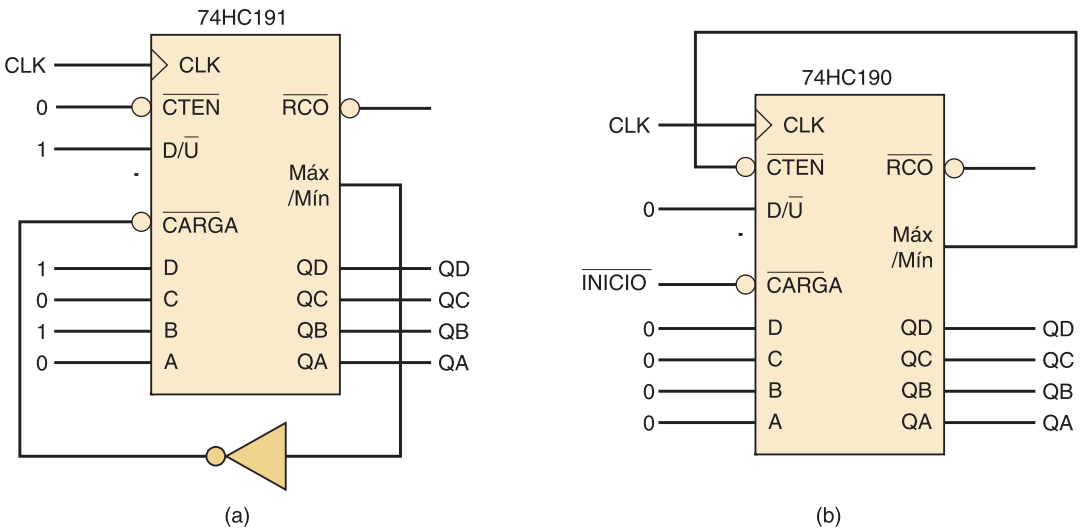


FIGURA 7-100 Problemas 7-23 y 7-24.

7-24. Consulte el circuito contador de CI en la figura 7-100(b):
(a) Describa la salida del contador en *QD QC QB QA* si $\overline{INICIO}$ está en BAJO.
(b) Describa la salida del contador en *QD QC QB QA* si se aplica un pulso momentáneo a $\overline{INICIO}$ para que cambie a BAJO y después regresa al nivel ALTO.
(c) ¿Cuál es el módulo del contador? ¿Es un contador reciclable?

D 7-25.* Dibuje un diagrama esquemático para crear un contador MOD-6 reciclable que utilice:
(a) el control borrar en un 74ALS160.
(b) el control borrar en un 74ALS162.

D 7-26. Dibuje un diagrama esquemático para crear un contador MOD-6 reciclable que produzca la secuencia de conteo:
(a) 1, 2, 3, 4, 5, 6 y se repita con un 74ALS162.
(b) 5, 4, 3, 2, 1, 0 y se repita con un 74ALS190.
(c) 6, 5, 4, 3, 2, 1 y se repita con un 74ALS190.

D 7-27.* Diseñe un contador binario MOD-100; para ello utilice dos CIs 74HC161 o dos CIs 74HC163 y todas las compuertas necesarias. Los CIs contadores deberán conectarse en cascada en forma síncrona para producir la secuencia de conteo binaria del 0 al 99. El contador MOD-100 deberá tener dos entradas de control, una de habilitación de conteo activa en BAJO ($\overline{EN}$) y una borrar asíncrona activa en BAJO ($\overline{CLR}$). Etiquete a las salidas del contador como *Q0, Q1, Q2,* etc., con *Q0* = LSB. ¿Cuál salida es el MSB?

D 7-28. Diseñe un contador BCD MOD-100; para ello utilice dos CIs 74HC160 o dos CIs 74HC162 y todas las compuertas necesarias. Los CIs contadores deberán conectarse en cascada en forma síncrona para producir la secuencia de conteo BCD del 0 al 99. El contador MOD-100 deberá tener dos entradas de control, una de habilitación de conteo activa en ALTO (*EN*) y una de carga asíncrona activa en ALTO (*LD*). Etiquete a las salidas del contador como *Q0, Q1, Q2,* etc., con *Q0* = LSB. ¿Cuál conjunto de salidas representa el dígito de los 10s?

B 7-29.* Con una entrada de reloj de 6-MHz para un 74ALS163 que tenga las cuatro entradas de control en ALTO, determine la frecuencia de salida y el ciclo de trabajo para cada una de las *cinco* salidas (incluyendo RCO).

B 7-30. Con una entrada de reloj de 6-MHz para un 74ALS162 que tenga las cuatro entradas de control en ALTO, determine la frecuencia de salida y el ciclo de trabajo para cada una de las siguientes salidas: QA, QC, QD, RCO. ¿Qué hay de inusual acerca del patrón de la forma de onda que se produciría mediante la salida QB? Esta característica del patrón produce un ciclo de trabajo indefinido.

B 7-31.* La frecuencia de f_{ent} es de 6 MHz en la figura 7-101. Los dos CIs contadores se han conectado en cascada en forma asíncrona, de manera que la frecuencia de salida producida por el contador U1 es la frecuencia de entrada para el contador U2. Determine la frecuencia de salida para f_{sal1} y f_{sal2}.

B 7-32. La frecuencia de f_{ent} es de 1.5 MHz en la figura 7-102. Los dos CIs contadores se han conectado en cascada en forma asíncrona, de manera que la frecuencia de salida producida por el contador U1 es la frecuencia de entrada para el contador U2. Determine la frecuencia de salida para f_{sal1} y f_{sal2}.

D 7-33.* Diseñe un circuito divisor de frecuencia que produzca las siguientes tres frecuencias de señal de salida: 1.5 MHz, 150 kHz y 100 kHz. Utilice los chips tipo contador 74HC162 y 74HC163, junto con todas las compuertas necesarias. La frecuencia de entrada es de 12 MHz.

D 7-34. Diseñe un circuito divisor de frecuencia que produzca las siguientes tres frecuencias de señal de salida: 1 MHz, 800 kHz y 100 kHz. Utilice los CIs tipo contador 74HC160 y 74HC161, junto con todas las compuertas necesarias. La frecuencia de entrada es de 12 MHz.

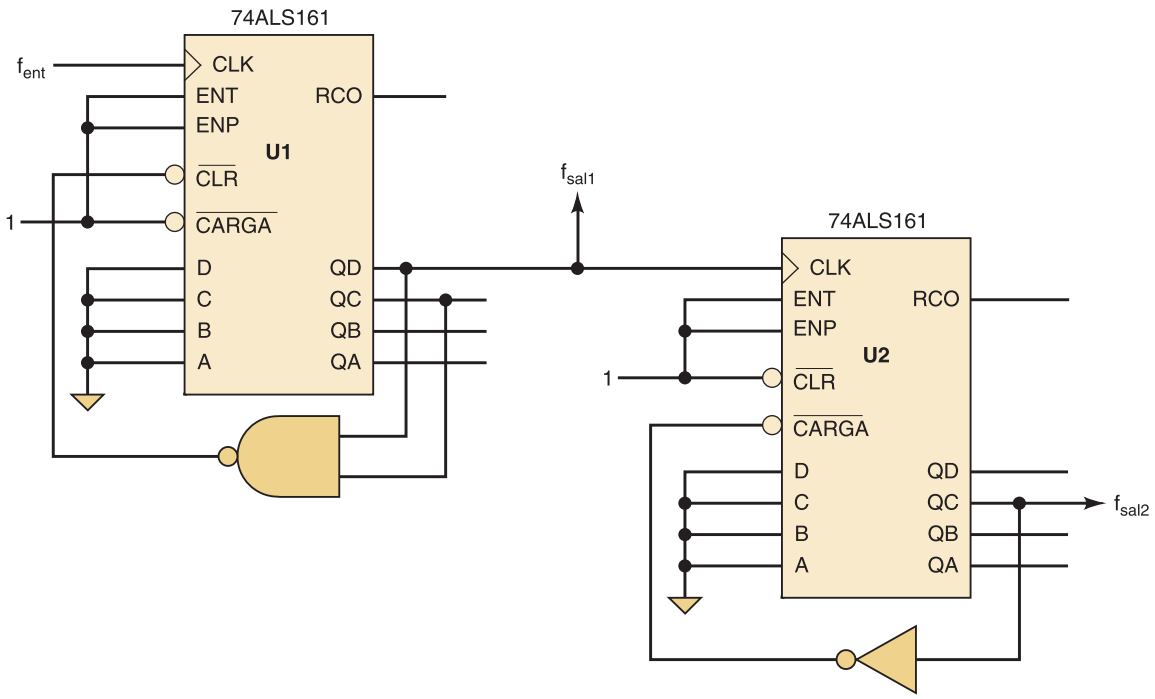


FIGURA 7-101 Problema 7-31.

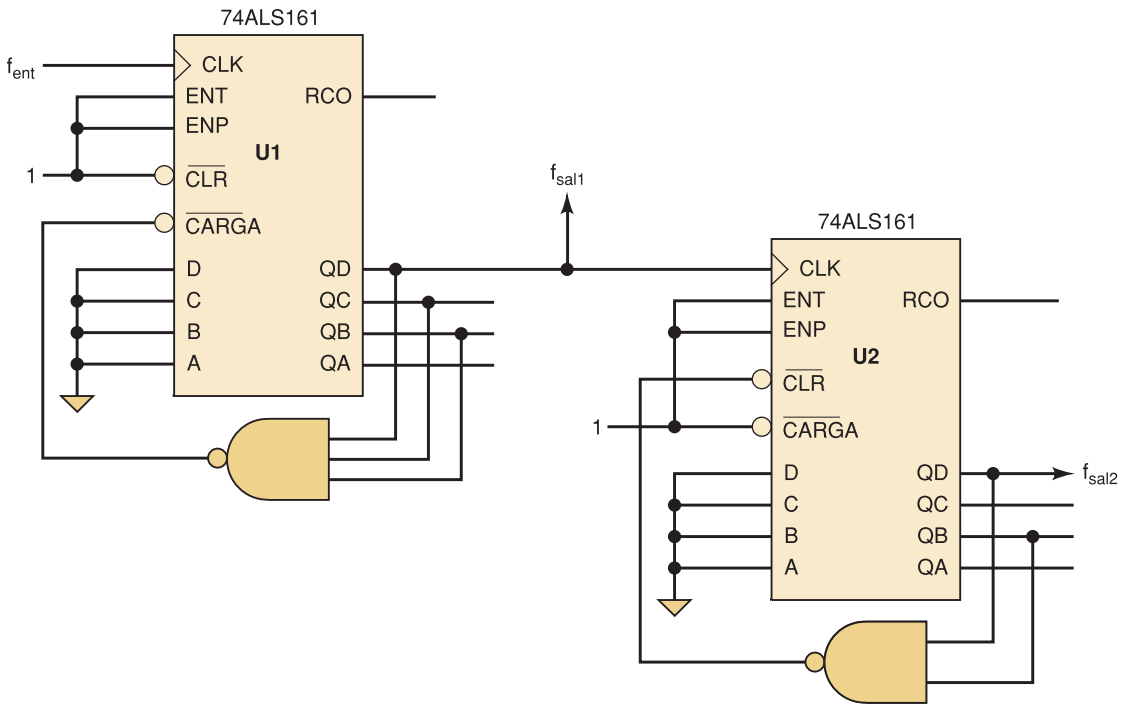


FIGURA 7-102 Problema 7-32.

SECCIÓN 7-8

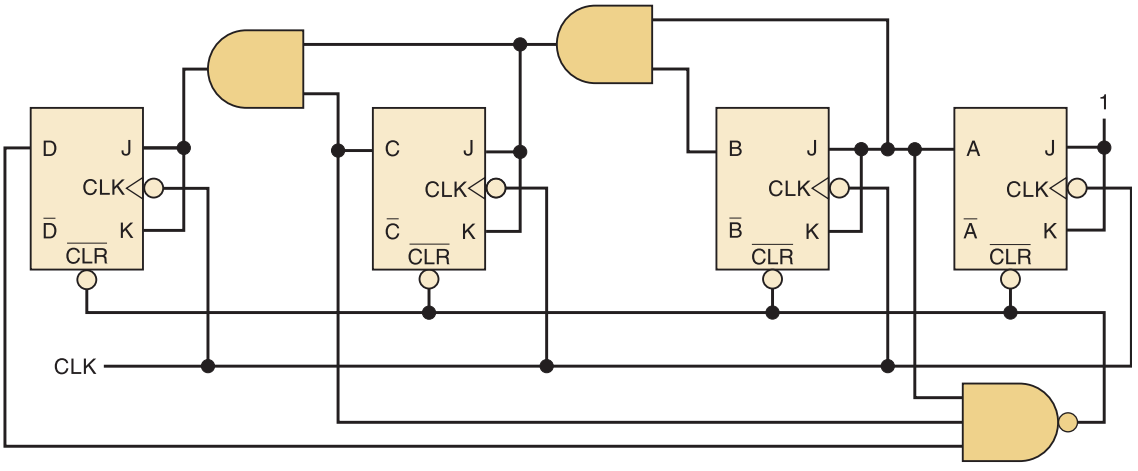
B 7-35.* Dibuje las compuertas necesarias para decodificar todos los estados de un contador MOD-16 que utilice salidas activas en BAJO.

B 7-36. Dibuje las compuertas AND necesarias para decodificar los 10 estados del contador BCD de la figura 7-8(b).

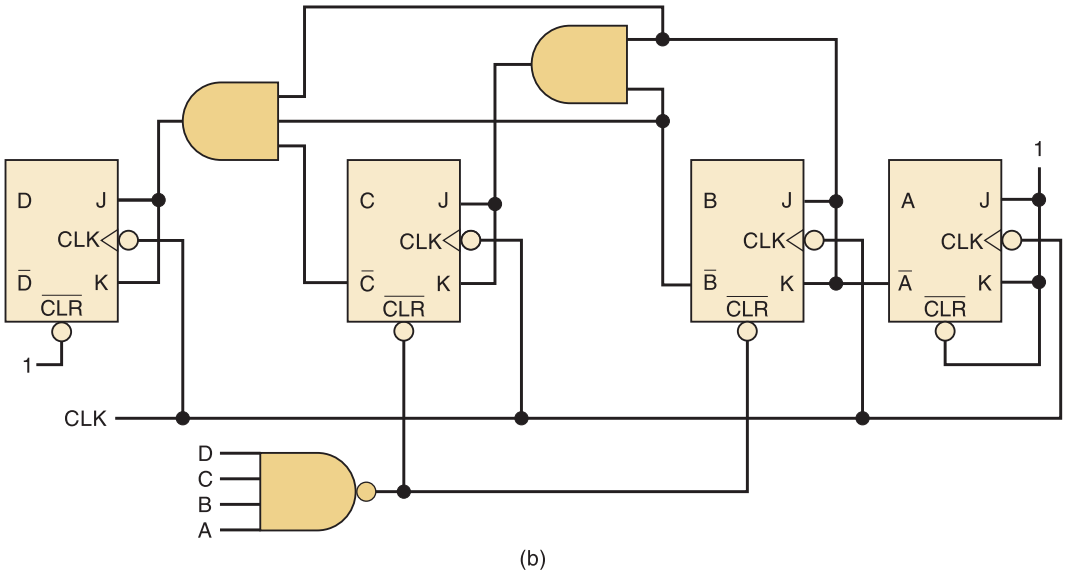
SECCIÓN 7-9

A 7-37.* Analice el contador asíncrono en la figura 7-103(a). Dibuje su diagrama de sincronización y determine el módulo del contador.

A 7-38. Repita el problema 7-37 para la figura 7-103(b).



(a)



(b)

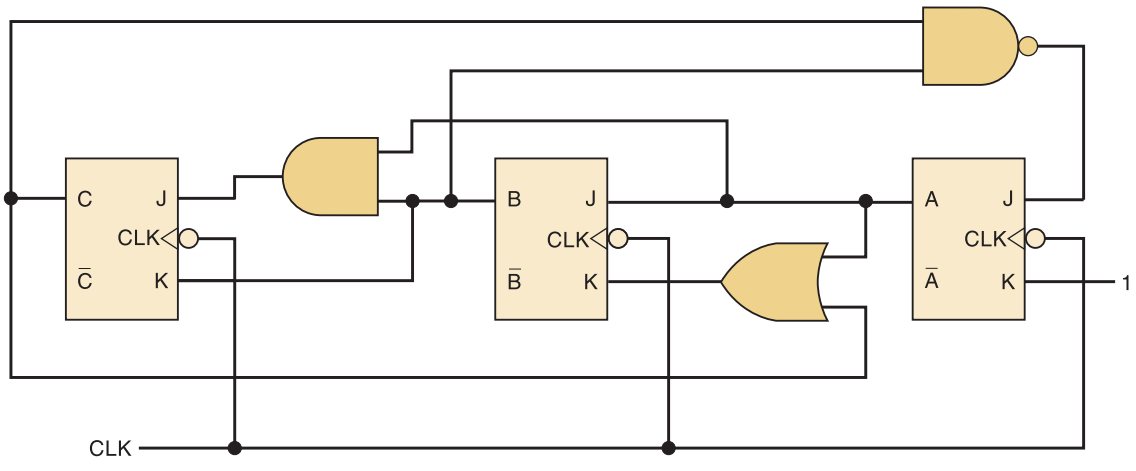
FIGURA 7-103 Problemas 7-37 y 7-38.

A 7-39.* Analice el contador síncrono de la figura 7-104(a). Dibuje su diagrama de sincronización y determine el módulo del contador.

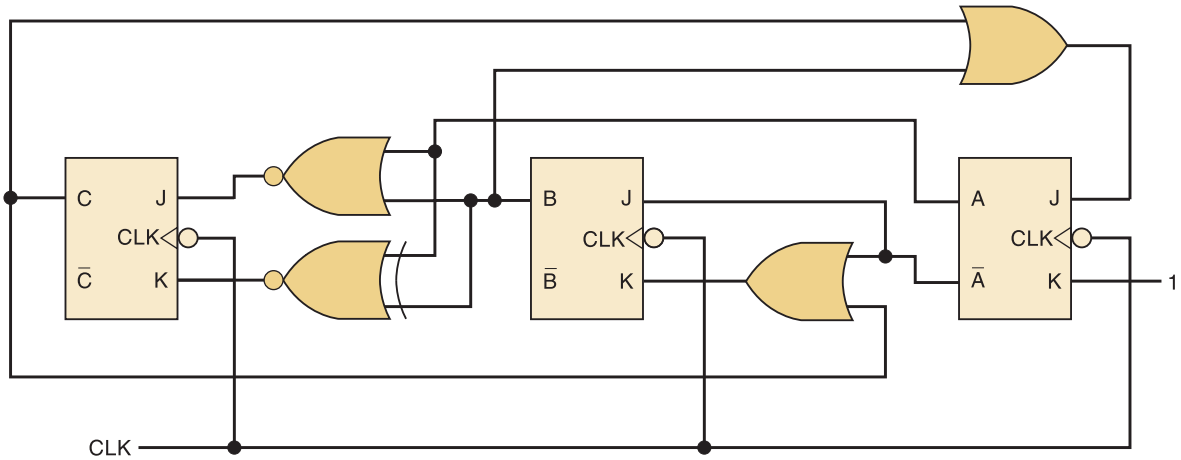
A 7-40. Repita el problema 7-39 para la figura 7-104(b).

A 7-41.* Analice el contador síncrono de la figura 7-105(a). F es una entrada de control. Dibuje su diagrama de transición de estados y determine el módulo del contador.

A 7-42. Analice el contador síncrono de la figura 7-105(b). Dibuje su diagrama de transición de estados completo y determine el módulo del contador. ¿Es el contador autocorregible?



(a)



(b)

FIGURA 7-104 Problemas 7-39 y 7-40.

SECCIÓN 7-10

D 7-43.* Diseñe un contador síncrono; para ello utilice FFs J-K. La secuencia del contador deberá ser: 000, 010, 101, 110 y se debe repetir. Los estados indeseables (no usados) 001, 011, 100 y 111 deberán siempre cambiar a 000 en el siguiente pulso de reloj.

(b) Rediseñe el contador de la parte (a) de manera que no tenga ningún requerimiento sobre los estados no utilizados; esto es, que sus SIGUIENTES estados puedan ser "no importa". Compare este diseño con el del inciso (a).

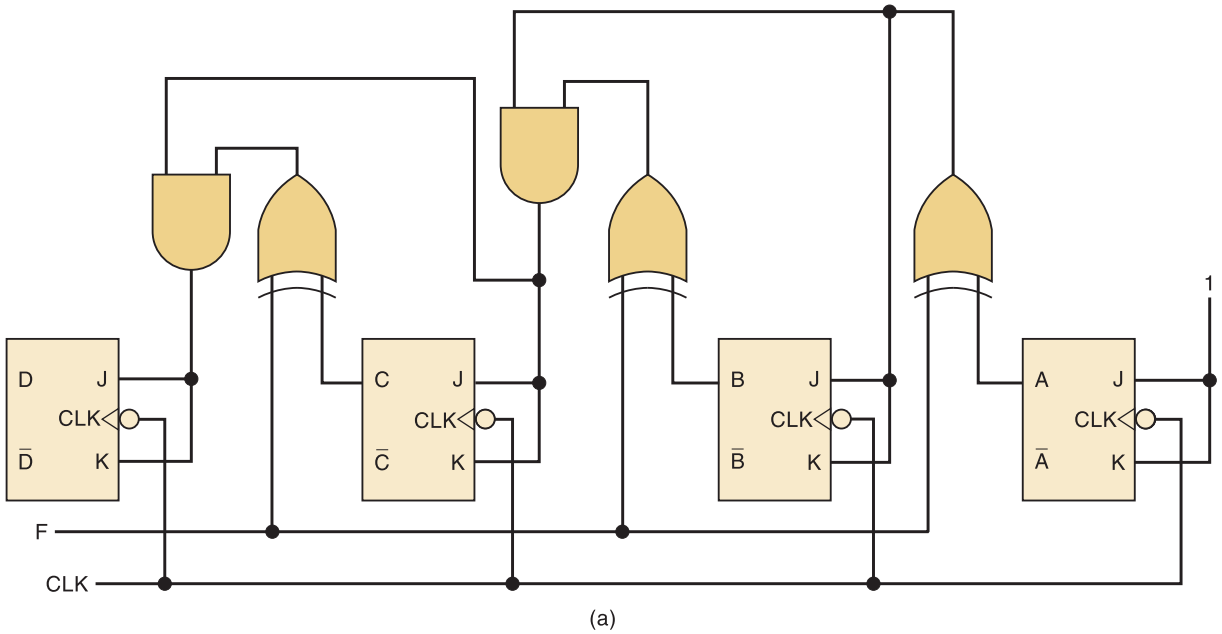
D 7-44. Diseñe un contador descendente MOD-5, síncrono y reciclable, que produzca la secuencia 100, 011, 010, 001, 000 y se repita. Utilice flip-flops J-K.

(a) Haga que los estados no usados cambien a 000 en el siguiente pulso de reloj.

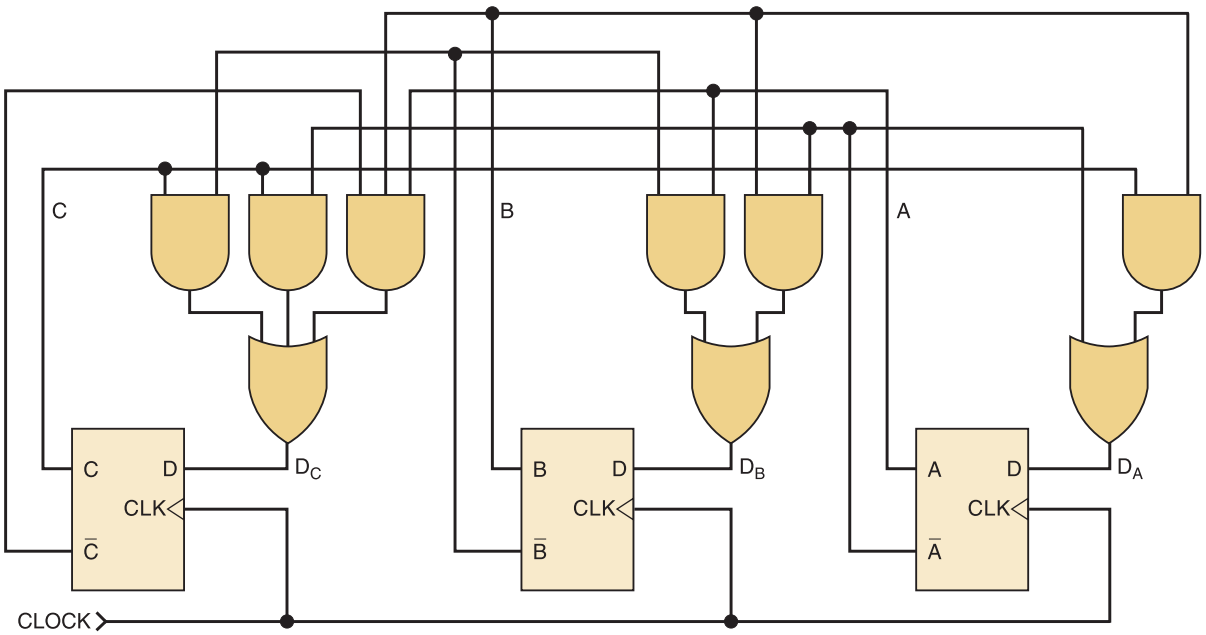
(b) Utilice SIGUIENTES estados tipo "no importa" para los estados no usados. ¿Es este diseño autocorregible?

D 7-45.* Diseñe un contador descendente BCD, síncrono y reciclable; para ello utilice FFs J-K con los estados SIGUIENTES tipo "no importa".

D 7-46. Diseñe un contador ascendente/descendente MOD-7, síncrono y reciclable; para ello utilice FFs J-K. Use los estados del 000 al 110 en el contador. Controle la dirección de conteo mediante la entrada D ($D = 0$ para contar hacia arriba y $D = 1$ para contar hacia abajo.



(a)



(b)

FIGURA 7-105 Problemas 7-41 y 7-42.

D 7-47.* Diseñe un contador descendente binario MOD-8, síncrono y reciclable; para ello utilice flip-flops D.

D 7-48. Diseñe un contador MOD-12, síncrono y reciclable; para ello utilice FFs D. Use los estados 0000 al 1011 en el contador.

SECCIONES 7-11 Y 7-12

H, D 7-49.* Diseñe un contador ascendente MOD-13 reciclable; para ello utilice un HDL. La secuencia de conteo deberá ser del 0000 al 1100. Simule el contador.

H, D 7-50. Diseñe un contador descendente MOD-25 reciclable; para ello utilice un HDL. La secuencia de conteo deberá ser del 11000 al 00000. Simule el contador.

H, D 7-51.* Diseñe un contador de código Gray MOD-16 reciclable; para ello utilice un HDL. El contador deberá tener una señal de habilitación activa en ALTO (*cnt*). Simule el contador.

H, D 7-52. Diseñe un controlador de medio paso bidireccional para un motor a pasos; para ello utilice un HDL. La entrada de control de dirección (*dir*) producirá un patrón en sentido a favor de las manecillas del reloj (CW) cuando esté en ALTO, o un patrón en sentido en contra de las manecillas del reloj cuando esté en BAJO. En la figura 7-106 se muestra la secuencia. Simule el circuito secuencial.

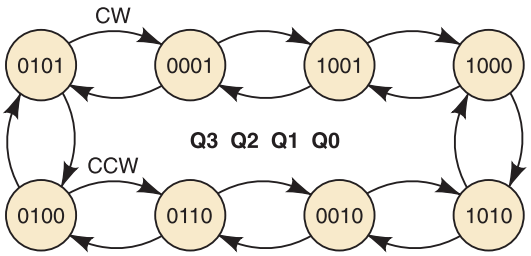


FIGURA 7-106 Problema 7-52.

H, D 7-53.* Diseñe un circuito divisor de frecuencia para producir como salida una señal de 100-kHz; para ello utilice un HDL. La frecuencia de entrada es de 5 MHz. Simule el contador.

H, D 7-54. Diseñe un circuito divisor de frecuencia que produzca como salida una de dos señales de frecuencia especificadas; para ello utilice un HDL. La frecuencia de salida se selecciona mediante la entrada de control f_{selec}. El divisor producirá como salida una frecuencia de 5 kHz cuando $f_{selec} = 0$ o de 12 kHz cuando $f_{selec} = 1$. La frecuencia de entrada es de 60 kHz. Simule el contador.

H, B 7-55.* Expanda el contador en HDL con todas sus características de la sección 7-12 para que sea un contador MOD-256. Simule el contador.

H, B 7-56. Expanda el contador en HDL con todas sus características de la sección 7-12 para que sea un contador MOD-1024. Simule el contador.

H, B 7-57.* Diseñe un contador descendente MOD-16 reciclable; para ello utilice un HDL. El contador deberá tener los siguientes controles (de menor a mayor prioridad): habilitación de conteo activa en BAJO ($\overline{en}$), borrar síncrona activa en ALTO (*clr*) y carga síncrona activa en BAJO ($\overline{ld}$). Decodifique el conteo terminal cuando se habilite mediante $\overline{en}$. Simule el contador.

H, D 7-58. Diseñe un contador ascendente/descendente MOD-10 reciclable; para ello utilice un HDL. El contador deberá contar hacia arriba cuando *arriba* = 1 y deberá contar hacia abajo cuando *arriba* = 0. También deberá tener los siguientes controles (de menor a mayor prioridad): habilitación de conteo activa en ALTO (*habilita*), carga síncrona activa en ALTO (*carga*) y borrar asíncrona activa en BAJO (*borrar*). Decodifique el conteo terminal cuando se habilite mediante *habilita*. Simule el contador.

SECCIÓN 7-13

H 7-59.* Cree un contador BCD MOD-1000; para ello conecte en cascada tres de los módulos de contador BCD en HDL (descritos en la sección 7-13). Simule el contador.

H 7-60. Cree un contador binario MOD-256; para ello conecte en cascada dos de los módulos de contador en HDL MOD-16, con todas sus características (descritas en la sección 7-12). Simule el contador.

H, D 7-61.* Diseñe un contador BCD MOD-50 síncrono; para ello conecte en cascada los diseños en HDL para un contador MOD-10 y otro contador MOD-5. El

contador MOD-50 deberá tener una señal de habilitación de conteo activa en ALTO (*habilita*) y una señal borrar síncrona activa en BAJO ($\overline{clrn}$). Asegúrese de incluir la detección del conteo terminal para el dígito de los 1s, para conectarlo en cascada con el dígito de los 10s. Simule el contador.

H, D 7-62. Diseñe un contador descendente BCD MOD-100 síncrono; para ello conecte en cascada dos módulos de contador descendente MOD-10 en HDL. El contador MOD-100 deberá tener una señal de carga en paralelo síncrona (*carga*). Simule el contador.

SECCIÓN 7-14

H 7-63.* Modifique la descripción de HDL de la figura 7-56 o de la figura 7-57 para agregar una secuencia de enjuagado después de lavar la ropa. La nueva secuencia de la máquina de estado deberá ser *inactiva* → *lavar_llena* → *lavar_agitar* → *lavar_girar* → *enjuagar_llena* → *enjuagar_agitar* → *enjuagar_girar* → *inactiva*. Utilice agua caliente para lavar y agua fría para enjuagar (agregue bits de salida para controlar dos válvulas de agua). Simule el diseño en HDL modificado.

H 7-64. Simule el diseño del controlador de semáforos en HDL que presentamos en la sección 7-14.

PARTE 2

SECCIONES 7-15 A 7-19

B 7-65.* Un conjunto de registros 74ALS174 se conecta como se muestra en la figura 7-107. ¿Qué tipo de transferencia de datos se realiza con cada registro? Determine la salida de cada registro cuando se aplica un pulso momentáneo en $\overline{MR}$ para que cambie a BAJO y después de cada uno de los pulsos de reloj indicados (CP#) en la tabla 7-10. ¿Cuántos pulsos de reloj deben aplicarse antes de que los datos que se introducen en *I5-I0* estén disponibles en *Z5-Z0*?

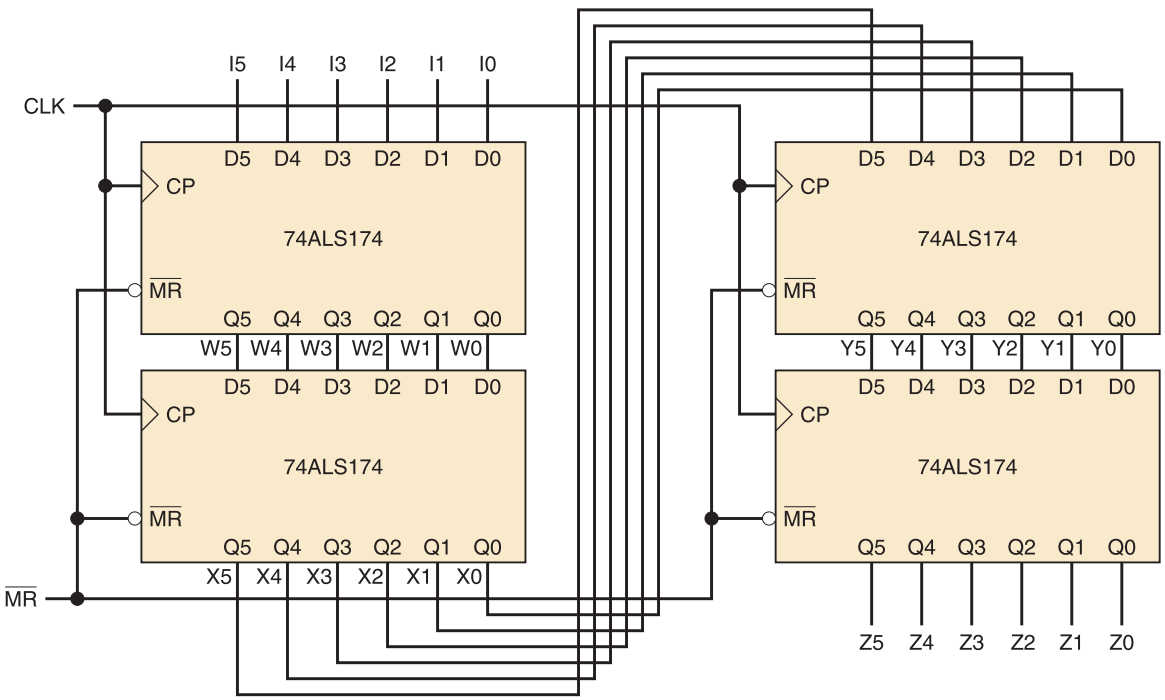


FIGURA 7-107 Problema 7-65.

TABLA 7-10

↑CLK	$\overline{MR}$	I5–I10	W5–W0	X5–X0	Y5–Y0	Z5–Z0
X	0	101010				
CP1	1	101010				
CP2	1	010101				
CP3	1	000111				
CP4	1	111000				
CP5	1	011011				
CP6	1	001101				
CP7	1	000000				
CP8	1	000000				

B 7-66. Complete el diagrama de sincronización de la figura 7-108 para un 74HC174. ¿Cómo muestra el diagrama de sincronización que la señal restablecer maestra es asíncrona?

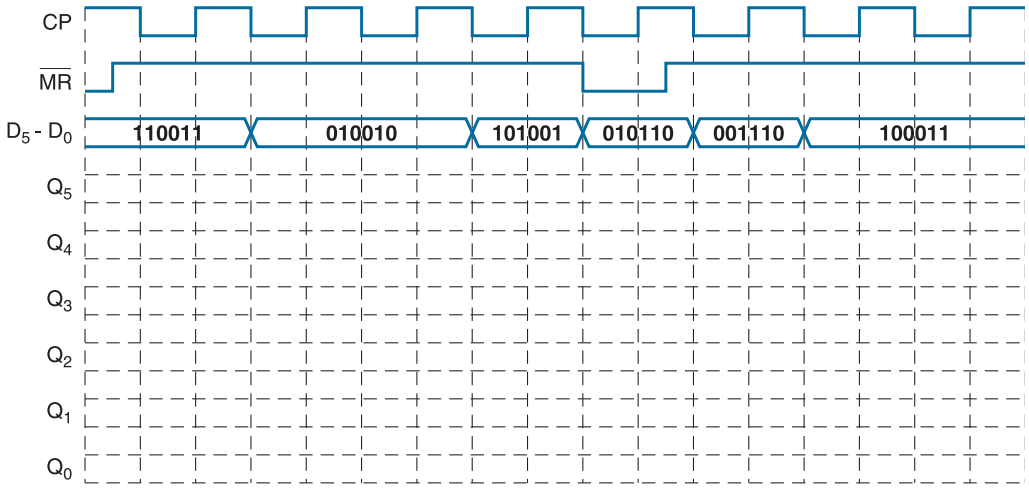


FIGURA 7-108 Problema 7-66.

B 7-67.* ¿Cuántos pulsos de reloj se necesitarán para cargar por completo ocho bits de datos en serie en un 74ALS166? ¿Cómo se relaciona esto con el número de flip-flops que contiene el registro?

B 7-68. Repita el ejemplo 7-18 para las formas de onda de entrada que se proporcionan en la figura 7-109.

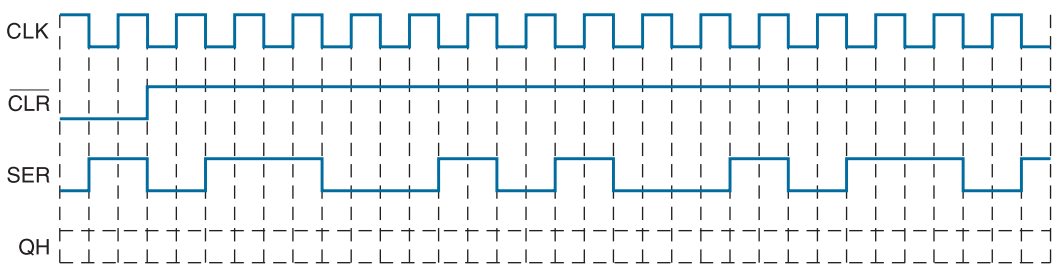


FIGURA 7-109 Problema 7-68.

7-69.* Repita el ejemplo 7-20 con $D_S = 1$ y las formas de onda de entrada que se proporcionan en la figura 7-110.

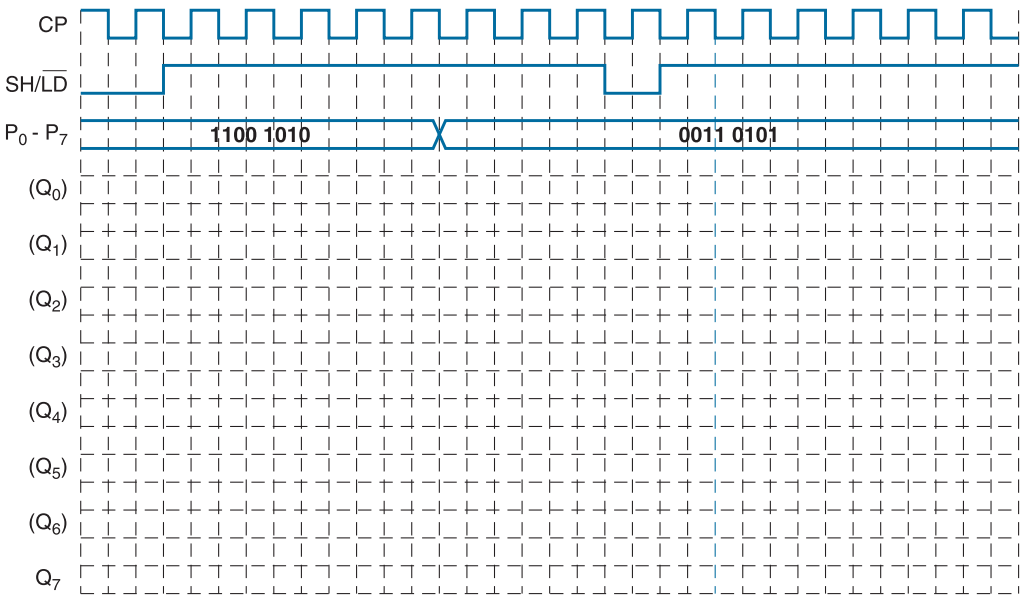
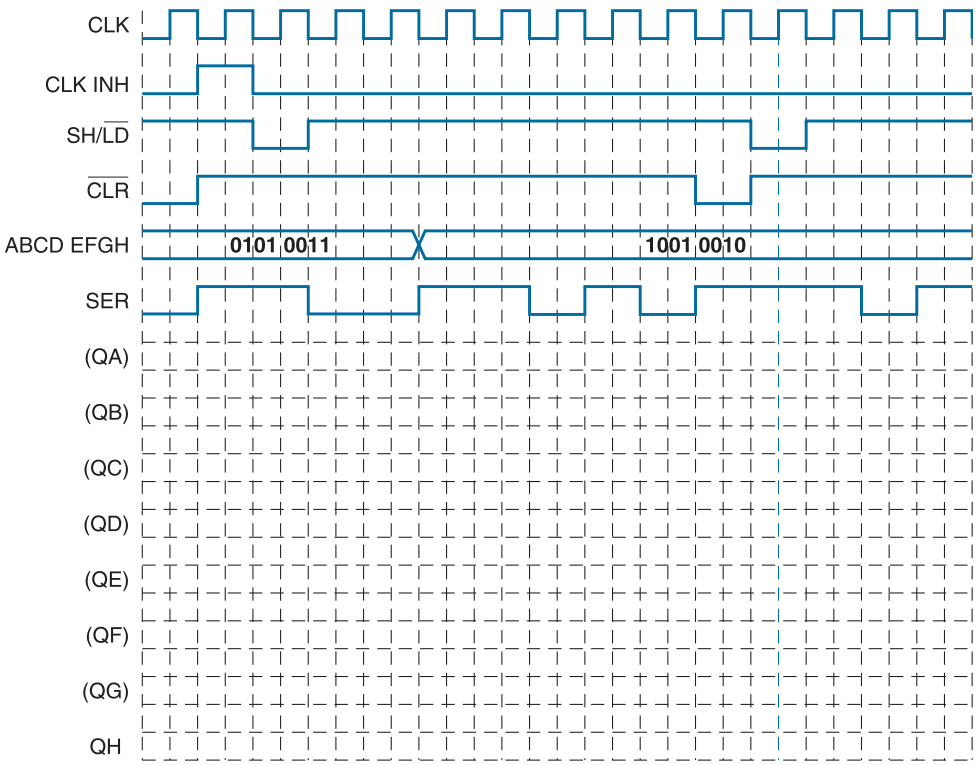


FIGURA 7-110 Problema 7-69.

7-70. Aplique las formas de onda de entrada que se proporcionan en la figura 7-111 a un 74ALS166 y determine la salida que se produce.

FIGURA 7-111
Problema 7-70.



B 7-71.* Al examinar el diagrama esquemático para una pieza de equipo, es común que un técnico o un ingeniero se topen con un CI desconocido. En tales casos a menudo es necesario consultar la hoja técnica del fabricante para ver las especificaciones del dispositivo. Investigue la hoja técnica para el registro de desplazamiento universal bidireccional 74AS194, para responder a las siguientes preguntas:

(a) ¿Es la entrada $\overline{CLR}$ asíncrona o síncrona?

(b) *Verdadero o falso*: cuando *CLK* está en BAJO, las entradas S_0 y S_1 no tienen efecto sobre el registro.

(c) Suponga las siguientes condiciones:

$$Q_A\ Q_B\ Q_C\ Q_D = 1\ 0\ 1\ 1$$
$$A\ B\ C\ D = 0\ 1\ 1\ 0$$
$$\overline{CLR} = 1$$
$$SR\ SER = 0$$
$$SL\ SER = 1$$

Si $S_0 = 0$ y $S_1 = 1$, determine las salidas del registro después de un pulso en *CLK*. Después de dos pulsos en *CLK*. Después de tres. Después de cuatro.

(d) Utilice las mismas condiciones excepto $S_0 = 1$ y $S_1 = 0$, y repita la parte (c).

(e) Repita la parte (c) con $S_0 = 1$ y $S_1 = 1$.

(f) Repita la parte (c) con $S_0 = 0$ y $S_1 = 0$.

(g) Use las mismas condiciones que en la parte (c), sólo que esta vez suponga que Q_A está conectada a *SL SER*. ¿Cuáles serán las salidas del registro después de cuatro pulsos en *CLK*?

A 7-72. Consulte la figura 7-112 para responder a las siguientes preguntas:

(a) ¿Qué función del registro (carga o desplazamiento) se realizará en el siguiente ciclo de reloj si ent = 1 y sal = 0? ¿Qué valor de datos se introducirá cuando se aplique un pulso de reloj?

(b) ¿Qué función del registro (carga o desplazamiento) se realizará en el siguiente ciclo de reloj si ent = 0 y sal = 1? ¿Qué valor de datos se introducirá cuando se aplique un pulso de reloj?

(c) ¿Qué función del registro (carga o desplazamiento) se realizará en el siguiente ciclo de reloj si ent = 0 y sal = 0? ¿Qué valor de datos se introducirá cuando se aplique un pulso de reloj?

(d) ¿Qué función del registro (carga o desplazamiento) se realizará en el siguiente ciclo de reloj si ent = 1 y sal = 1? ¿Qué valor de datos se introducirá cuando se aplique un pulso de reloj?

(e) ¿Qué condición de entrada hará que en algún momento dado (después de varios pulsos de reloj) la salida cambie de estado?

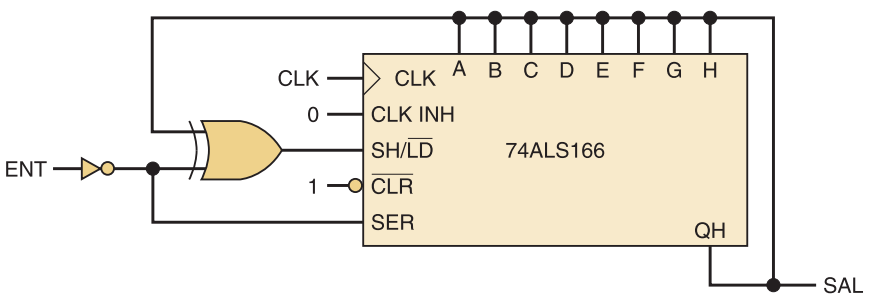


FIGURA 7-112
Problema 7-72.

(f) Para cambiar el nivel lógico de salida se requiere que la nueva condición de entrada dure por lo menos ¿cuántos pulsos de reloj?

(g) Si la señal de entrada cambia de nivel y después regresa a su nivel lógico antes del número de pulsos de reloj especificados en la parte (f), ¿qué ocurre con la señal de salida?

(h) Explique por qué este circuito puede usarse para eliminar los rebotes en los interruptores.

SECCIÓN 7-20

B 7-73.* Dibuje el diagrama para un contador de anillo MOD-5; para ello utilice flip-flops J-K. Asegúrese de que el contador inicie la secuencia de conteo apropiada cuando se encienda.

7-74. Agregue un flip-flop más para convertir el contador de anillo MOD-5 del problema 7-73 en un contador MOD-10. Determine la secuencia de estados para este contador. Éste es un ejemplo de un contador de décadas que no es un contador BCD. Dibuje el circuito de decodificación para este contador.

B 7-75.* Dibuje el diagrama para un contador Jonson MOD-10 mediante el uso de un 74HC164. Asegúrese de que el contador inicie la secuencia de conteo apropiada cuando se encienda. Determine la secuencia de conteo para este contador y dibuje el circuito de decodificación necesario para decodificar cada uno de los 10 estados. Éste es otro ejemplo de un contador de décadas que no es un contador BCD.

7-76. La entrada de reloj para el contador Johnson del problema 7-75 es de 10 Hz. ¿Cuál es la frecuencia y el ciclo de trabajo para cada una de las salidas del contador?

SECCIÓN 7-21

F 7-77.* El contador MOD-10 de la figura 7-8(b) produce la secuencia de conteo 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111 y se repite. Identifique algunas posibles condiciones de falla que podrían producir este resultado.

F 7-78. El contador MOD-10 de la figura 7-8(b) produce la secuencia de conteo 0000, 0101, 0010, 0111, 1000, 1101, 1010, 1111 y se repite. Identifique algunas posibles condiciones de falla que podrían producir este resultado.

SECCIONES 7-22 Y 7-23

H 7-79.* Cree un registro de desplazamiento SISO de ocho bits; para ello utilice HDL. La entrada en serie se llama *ser* y la salida en serie se llama *qsal*. Una señal de habilitación activa en BAJO ($\overline{en}$) controla el registro de desplazamiento. Simule el diseño.

H 7-80. Cree un registro de desplazamiento PIPO de ocho bits; para ello utilice HDL. Los datos de entrada están en *d[7..0]* y las salidas en *q[7..0]*. Una señal de habilitación activa en ALTO (*ld*) controla el registro de desplazamiento. Simule el diseño.

H 7-81.* Cree un registro de desplazamiento PISO de ocho bits; para ello utilice HDL. Los datos de entrada están en *d[7..0]* y la salida es *q0*. La función del registro de desplazamiento se controla mediante *sh*_ld (*sh*_ld = 0 para la carga síncrona en paralelo y *sh*_ld = 1 para el desplazamiento en serie). El registro también debe tener una señal borrar asíncrona activa en BAJO ($\overline{clrn}$). Simule el diseño.

H 7-82. Cree un registro de desplazamiento SIPO de ocho bits; para ello utilice HDL. Los datos de entrada están en *ser_ent* y las salidas en *q[7..0]*. La función del registro de desplazamiento se habilita mediante una señal de control activa en ALTO llamada *desplaza*. El registro de desplazamiento también tiene una señal borrar síncrona activa en ALTO (*borrar*), de mayor prioridad. El registro de desplazamiento también tiene una señal borrar síncrona activa en ALTO (*borrar*) de mayor prioridad. Simule el diseño.

H 7-83.* Simule el diseño del registro de desplazamiento universal del ejemplo 7-24.

H 7-84. Cree un registro de desplazamiento universal de ocho bits; para ello conecte en cascada dos de los módulos del ejemplo 7-24. Simule el diseño.

H, D 7-85.* Diseñe un contador Johnson MOD-10 de inicio automático con una señal restablecer asíncrona (*reset*), activa en ALTO; para ello utilice un HDL. Simule el diseño.

H, D 7-86. Algunas veces una aplicación digital puede necesitar un contador de anillo que recircule un solo cero en vez de un solo uno. Así, el contador de anillo tendría una salida activa en BAJO, en vez de una salida activa en ALTO. Diseñe un contador de anillo MOD-8 de inicio automático con una salida activa en BAJO; para ello utilice un HDL. El contador de anillo también deberá tener una señal de control *retener* activa en ALTO para deshabilitar el conteo. Simule el diseño.

SECCIÓN 7-24

H 7-87.* Utilice el simulador de Altera para probar el ejemplo de diseño del circuito monoestable no redisparable sensible al nivel de la figura 7-88 (AHDL) o 7-89 (VHDL). Utilice un reloj de 1-kHz y cree un pulso de salida de 10 ms para la simulación. Verifique que:

(a) Se cree la anchura de pulso correcta cuando se dispare.

(b) La salida pueda terminarse antes de tiempo con la entrada reset.

(c) El diseño del circuito monoestable sea no redisparable y que no pueda dispararse otra vez sino hasta que se haya agotado el tiempo.

(d) La señal de disparo debe durar lo suficiente como para que el reloj la atrape.

(e) La anchura de pulso pueda cambiarse a un valor distinto.

H 7-88. Modifique el ejemplo de diseño del circuito monoestable no redisparable sensible al nivel de la figura 7-88 (AHDL) o de la figura 7-89 (VHDL), de manera que el circuito monoestable sea redisparable pero continúe siendo sensible al nivel. Simule el diseño.

PREGUNTA DE PRÁCTICA

B 7-89.* Para cada una de las siguientes aseveraciones, indique el (los) tipo(s) de contador(es) que se describen.

(a) Se aplica un pulso de reloj a cada FF al mismo tiempo.

(b) Cada FF divide la frecuencia en su entrada CLK entre 2.

(c) La secuencia de conteo es 11, 10, 101, 100, 011, 010, 001, 000.

(d) El contador tiene 10 estados distintos.

(e) El retraso total de conmutación es la suma de los retrasos individuales de cada FF.

(f) Este contador no requiere lógica de decodificación.

(g) El número MOD siempre es el doble del número de FFs.

(h) Este contador divide la frecuencia de entrada entre su número MOD.

(i) Este contador puede comenzar su secuencia de conteo en cualquier estado inicial deseado.

(j) Este contador puede contar en cualquier dirección.

(k) Este contador puede sufrir de deformaciones en la decodificación, debido a sus retrasos de propagación.

(l) Este contador sólo cuenta del 0 al 9.

(m) Este contador puede diseñarse para que cuente a través de secuencias arbitrarias mediante la determinación del circuito lógico necesario en las entradas de control síncronas de cada flip-flop.

RESPUESTAS A LAS PREGUNTAS DE REPASO DE LAS SECCIONES

PARTE 1

SECCIÓN 7-1

1. Falso 2. 0000 3. 128

SECCIÓN 7-2

1. Cada FF agrega su retraso de propagación al retraso total del contador, en respuesta a un pulso de reloj. 2. MOD-256

SECCIÓN 7-3

1. Puede operar a frecuencias de reloj más altas y tiene un circuito más complejo.
2. Seis FFs y cuatro compuertas AND 3. ABCDE

SECCIÓN 7-4

1. D, C y A 2. Verdadero, ya que un contador BCD tiene 10 estados distintos.
3. 5 kHz

SECCIÓN 7-5

1. En un contador ascendente, el conteo se incrementa en 1 con cada pulso de reloj; en un contador descendente, el conteo se decrementa en 1 con cada pulso. 2. Hay que cambiar las conexiones a las salidas invertidas respectivas, en vez de las *Q*s.

SECCIÓN 7-6

1. Puede prestablecerse a cualquier conteo inicial deseado. 2. El preajuste asíncrono es independiente de la entrada de reloj, mientras que el preajuste síncrono ocurre en el flanco activo de la señal de reloj.

SECCIÓN 7-7

1. $\overline{\text{CARGA}}$ es el control que habilita la carga en paralelo de las entradas de datos D C B A (A = LSB). 2. $\overline{\text{CLR}}$ es el control que habilita el restablecimiento del contador a 0000. 3. Verdadero 4. Todas las entradas de control ($\overline{\text{CLR}}$, $\overline{\text{CARGA}}$, ENT y ENP) en el 74162 deben estar en ALTO. 5. $\overline{\text{CARGA}} = 1$, $\overline{\text{CTEN}} = 0$ y $D/U = 1$ para contar hacia abajo. 6. 74HC163: de 0 a 65,535; 74ALS190: de 0 a 9999 o de 9999 a 0.

SECCIÓN 7-8

1. Sesenta y cuatro 2. Una compuerta NAND de seis entradas: A, B, C, D, E y F.

SECCIÓN 7-9

1. No tendríamos que tratar con los estados transitorios y las posibles deformaciones en las formas de onda de salida. 2. Tabla de estado PRESENTE y estado SIGUIENTE.
3. Las compuertas controlan la secuencia de conteo. 4. Todos los estados sin usar conducen de regreso a la secuencia de conteo del contador.

SECCIÓN 7-10

1. Consulte el texto. 2. Asocia todos los posibles estados PRESENTES con su estado SIGUIENTE deseado. 3. Muestra los niveles necesarios en la entrada asíncrona de cada flip-flop para producir las transiciones de estado del contador. 4. Verdadero

SECCIÓN 7-11

1. Tablas de estado PRESENTE / estado SIGUIENTE. 2. El estado SIGUIENTE deseado. 3. AHDL:

```
ff[ ].clk = !reloj
```

VHDL:

```
IF (reloj = ?0? AND reloj?EVENT) THEN
```

4. Descripción del comportamiento.

SECCIÓN 7-12

1. La señal borrar asíncrona hace que el contador se borre de inmediato. La carga asíncrona se produce en el siguiente flanco activo del reloj. 2. AHDL: utilice el puerto .clrn en los FFs; VHDL: defina la función borrar antes de comprobar el flanco del reloj.
3. Por el orden de evaluación en una instrucción IF.

SECCIÓN 7-13

1. Ambos HDLs pueden utilizar un diagrama de bloques para conectar los módulos; VHDL también puede usar un archivo de texto que describa las conexiones entre los componentes. 2. Un bus es una colección de líneas de señal; se representa en forma gráfica mediante una línea gruesa. 3. Habilitación de conteo y decodificación del conteo terminal.

SECCIÓN 7-14

1. Un contador se utiliza, por lo general, para contar eventos, mientras que una máquina de estados se utiliza, por lo general, para controlar eventos. 2. Una máquina de estados puede describirse mediante el uso de símbolos para describir sus estados, en vez de los estados binarios actuales. 3. El compilador asigna los valores óptimos para minimizar el circuito. 4. La descripción es mucho más fácil de escribir y de comprender.

PARTE 2

SECCIÓN 7-19

1. Entrada en paralelo/salida en serie. 2. Verdadero 3. Entrada en serie/salida en paralelo. 4. Entrada en serie/salida en serie. 5. El 74165 utiliza transferencia de datos asíncrona en paralelo; el 74174 utiliza transferencia de datos síncrona en paralelo. 6. Un nivel ALTO evita el desplazamiento en las entradas CP.

SECCIÓN 7-20

1. Contador de anillo. 2. Contador Johnson. 3. La salida invertida del último FF se conecta a la entrada del primer FF. 4. (a) Falso (b) Verdadero (c) Verdadero
5. Dieciséis; ocho

SECCIÓN 7-22

1. AHDL:

```
reg[ ].d = (reg[6..0], dat)
```

VHDL:

```
reg := reg (6 DOWNTO 0) & dat
```

2. Porque el registro puede seguir recibiendo flancos de reloj durante la retención.

SECCIÓN 7-23

1. Puede comenzar en cualquier estado, pero en cierto momento llegará a la secuencia de anillo deseada. 2. Las líneas 1 y 12 3. Las líneas 12 y 13

SECCIÓN 7-24

1. La entrada restablecer. 2. La frecuencia de reloj y el valor de retraso que se carga en el contador. 3. En forma síncrona. 4. La anchura del pulso de salida es muy consistente. 5. El pulso de salida responde al flanco de disparo de inmediato.
6. El estado del disparo en el flanco de reloj actual y su estado en el flanco anterior.

CAPÍTULO 8

FAMILIAS LÓGICAS DE CIRCUITOS INTEGRADOS

CONTENIDO

8-1 Terminología de CIs digitales
8-2 La familia lógica TTL
8-3 Hojas técnicas TTL
8-4 Características de las series TTL
8-5 Capacidad de carga de la familia TTL
8-6 Otras características de la familia TTL
8-7 Tecnología MOS
8-8 Lógica de MOS complementario
8-9 Características de las series CMOS
8-10 Tecnología de bajo voltaje
8-11 Salidas de colector abierto/ drenador abierto
8-12 Salidas lógicas triestado (tres estados)
8-13 Lógica de interfase de bus de alta velocidad
8-14 La familia ECL de CIs digitales
8-15 Compuerta de transmisión CMOS (interruptor bilateral)
8-16 Interfase de CIs
8-17 Interfase de voltaje mixto
8-18 Comparadores de voltaje analógico
8-19 Diagnóstico de fallas

OBJETIVOS

Al terminar este capítulo, usted podrá:

- Leer y comprender la terminología de CIs digitales, según lo especificado en las hojas técnicas de los fabricantes.
- Comparar las características de la familia TTL estándar y de las diversas series TTL.
- Determinar la capacidad de carga para un dispositivo lógico específico.
- Utilizar dispositivos lógicos con salidas de colector abierto.
- Analizar circuitos que contengan dispositivos triestado.
- Comparar las características de las diversas series CMOS.
- Analizar circuitos que utilicen un interruptor bilateral de CMOS para permitir que un sistema digital controle señales analógicas.
- Describir las principales características y diferencias entre las familias lógicas TTL, ECL, MOS y CMOS.
- Citar e implementar las diversas consideraciones requeridas cuando se integran circuitos digitales de distintas familias lógicas.
- Utilizar comparadores de voltaje para permitir que un sistema digital se controle mediante señales analógicas.
- Utilizar un pulsador lógico y una sonda lógica como herramientas de diagnóstico de fallas en circuitos digitales.

INTRODUCCIÓN

Como se describió en el capítulo 4, la tecnología de CIs digitales ha progresado con rapidez, desde la integración a pequeña escala (SSI) con menos de 12 compuertas por chip; después la integración a mediana escala (MSI), en la que se pueden tener de 12 a 99 compuertas equivalentes por chip; más tarde la integración a gran escala y a muy gran escala (LSI y VLSI, respectivamente), en las que puede haber decenas de miles de compuertas por chip; y, por último, la integración a ultragran escala (ULSI), en la que puede haber más de 100,000 compuertas por chip, y la integración a giga escala (GSI), con 1 millón o más de compuertas.

La mayoría de las razones por las que los sistemas digitales modernos utilizan circuitos digitales son obvias. Los CI encapsulan mucho más circuitos en un encapsulado pequeño, por lo que el tamaño total de casi cualquier sistema digital se reduce. El costo se reduce en forma sustancial debido al ahorro que representa la producción en masa de grandes volúmenes de dispositivos similares. Algunas de las otras ventajas no son tan evidentes.

Los CIs han hecho que los sistemas digitales sean más confiables al reducir el número de interconexiones externas de un dispositivo a otro. Antes de que tuviéramos los CIs, todas las conexiones en los circuitos eran de un componente discreto (transistor, diodo, resistencia, etc.) a otro. Ahora la mayoría de las conexiones son internas en los CIs, en donde están protegidas de malas

soldaduras, quebraduras o cortos en las rutas de conexión en un circuito impreso (PCB), y demás problemas físicos. Los CIs también han reducido en forma dramática la cantidad de energía eléctrica necesaria para realizar una función dada, debido a que sus circuitos en miniatura, por lo general, requieren menos energía que sus contrapartes discretas. Además de los ahorros en costo por la fuente de energía, esta reducción en la energía también significa que un sistema prácticamente no requiere enfriamiento.

Hay algunas cosas que los CIs no pueden hacer: no pueden manejar corrientes o voltajes muy grandes, ya que el calor generado en espacios tan pequeños provocaría que la temperatura aumentara más allá de los límites aceptables. Además, los CIs no pueden implementar con facilidad ciertos dispositivos eléctricos tales como los inductores, los transformadores y los capacitores grandes. Por estas razones, el principal uso de los CIs es para realizar operaciones de circuitos de baja energía, a lo cual se le conoce, por lo general, como *procesamiento de información*. Las operaciones que requieren altos niveles de energía o dispositivos que no pueden integrarse se siguen manejando mediante componentes discretos.

Con el amplio uso de los CIs viene también la necesidad de conocer y comprender las características eléctricas de las familias lógicas de los CIs más comunes. Recuerde que las diversas familias lógicas difieren en cuanto a los componentes principales que utilizan en sus circuitos. TTL y ECL utilizan transistores *bipolares* como su elemento principal en el circuito; PMOS, NMOS y CMOS utilizan transistores *MOS*FET unipolares como su componente principal. En este capítulo presentaremos las características importantes de cada una de estas familias de CIs y sus subfamilias. El punto más importante es comprender la naturaleza de los circuitos de entrada y los circuitos de salida para cada familia lógica. Una vez que comprenda esto, estará mucho mejor preparado para realizar el análisis, diagnóstico de fallas y cierto diseño de los circuitos digitales que contengan cualquier combinación de familias de CIs. Estudiaremos el funcionamiento interno de los dispositivos en cada familia con el circuito más simple que transmita las características imprescindibles de todos los miembros de la familia.

8-1 TERMINOLOGÍA DE CIS DIGITALES

Aunque hay muchos fabricantes de CIs digitales, la mayor parte de la nomenclatura y la terminología están estandarizadas. A continuación definiremos y hablaremos sobre los términos más útiles.

Parámetros de voltaje y corriente (vea la figura 8-1)

- **V_{IH}(mín): voltaje de entrada en nivel alto.** El mínimo nivel de voltaje requerido para un 1 lógico en una *entrada*. El circuito lógico no aceptará como nivel ALTO cualquier voltaje que se encuentre por debajo de este nivel.
- **V_{IL}(máx): voltaje de entrada en nivel bajo.** El máximo nivel de voltaje requerido para un 0 lógico en una *entrada*. El circuito lógico no aceptará como nivel BAJO cualquier voltaje que se encuentre por encima de este nivel.
- **V_{OH}(mín): voltaje de salida en nivel alto.** El mínimo nivel de voltaje en la *salida* de un circuito lógico en el estado 1, bajo condiciones de carga específicas.
- **V_{OL}(máx): voltaje de salida en nivel bajo.** El máximo nivel de voltaje en la *salida* de un circuito lógico en el estado 0, bajo condiciones de carga específicas.
- **I_{IH}: corriente de entrada en nivel alto.** La corriente que fluye hacia una entrada cuando se le aplica a ésta un voltaje específico de alto nivel.
- **I_{IL}: corriente de entrada en nivel bajo.** La corriente que fluye hacia una entrada cuando se le aplica a ésta un voltaje específico de bajo nivel.
- **I_{OH}: corriente de salida en nivel alto.** La corriente que fluye desde una salida en el estado 1 lógico, bajo condiciones de carga específicas.

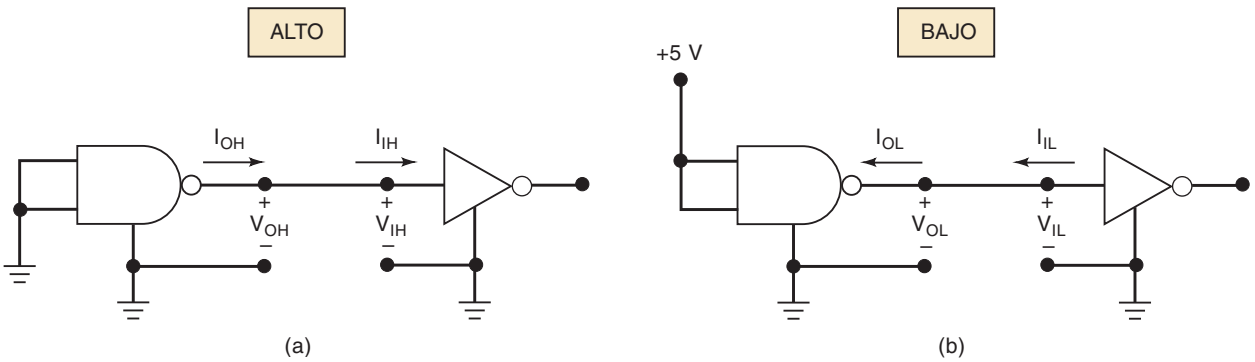


FIGURA 8-1 Corrientes y voltajes en los dos estados lógicos.

- **I_{OL}: corriente de salida en nivel bajo.** La corriente que fluye desde una salida en el estado 0 lógico, bajo condiciones de carga específicas.

Nota: la dirección real de la corriente puede ser opuesta a las que se muestran en la figura 8-1, dependiendo de la familia lógica. Todas las descripciones del flujo de corriente en este texto se refieren al flujo de corriente convencional (del potencial mayor al potencial menor). Para concordar con las convenciones de la mayoría de las hojas de datos, la corriente que fluye hacia dentro un nodo o dispositivo se considera positiva y la corriente que fluye hacia fuera de un nodo o dispositivo se considera negativa.

Capacidad de carga

En general, se requiere que la salida de un circuito lógico controle varias entradas lógicas. Algunas veces todos los CIs en el sistema digital son de la misma familia lógica, pero muchos sistemas tienen una mezcla de varias familias lógicas. La **capacidad de carga** se define como el número *máximo* de entradas lógicas que puede controlar una salida de manera confiable. Por ejemplo, si se especifica que una compuerta lógica tiene una capacidad de carga de 10, significa que puede manejar 10 entradas lógicas. Si se excede este número, no se pueden garantizar los voltajes de los niveles lógicos de salida. Es obvio que la capacidad de carga depende de la naturaleza de los dispositivos de entrada que se conectan a una salida. A menos que se especifique una familia lógica distinta como el dispositivo de carga, se asume que la capacidad de carga se refiere a la acción de cargar dispositivos de la misma familia que la salida reforzadora.

Tiempos de propagación

Una señal lógica siempre experimenta un retraso al pasar a través de un circuito. Los dos tiempos de propagación se definen de la siguiente manera:

- t_{PLH}. El tiempo que tarda la compuerta en cambiar del estado de 0 lógico al de 1 lógico (de BAJO a ALTO).
- t_{PHL}. El tiempo que tarda la compuerta en cambiar del estado de 1 lógico al de 0 lógico (de ALTO a BAJO).

La figura 8-2 ilustra estos tiempos de propagación para un INVERSOR. Observe que t_{PHL} es el retaso en la respuesta de la salida, al cambiar de ALTO a BAJO. Se mide entre los 50 puntos porcentuales en las transiciones de entrada y de salida. El valor t_{PLH} es el retraso en la respuesta de la salida, al cambiar de BAJO a ALTO.

En algunos circuitos lógicos t_{PHL} y t_{PLH} no son el mismo valor, y ambos variarán dependiendo de las condiciones de carga capacitiva. Los valores de los tiempos de propagación se utilizan como una medición de la velocidad relativa de los circuitos lógicos. Por ejemplo, un circuito lógico con valores de 10 ns es mucho más rápido que un circuito lógico con valores de 20 ns, bajo condiciones de carga específicas.

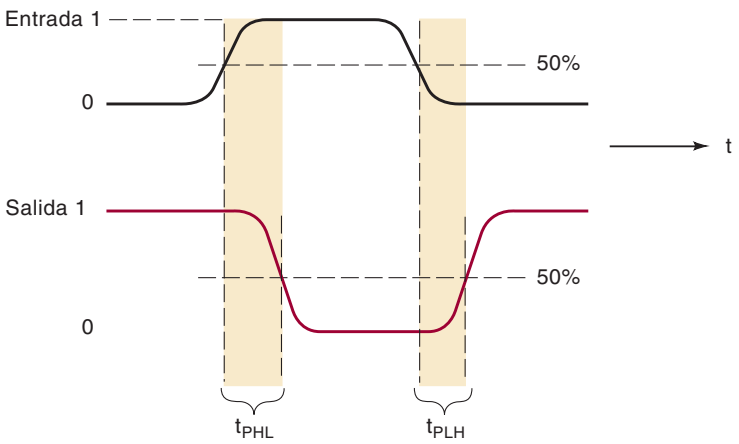


FIGURA 8-2 Tiempos de propagación.

Requerimientos de energía

Todo CI requiere una cierta cantidad de energía eléctrica para operar. Esta energía se suministra mediante uno o más voltajes de fuente de energía conectados a la(s) terminal(es) de energía en el chip, las cuales se etiquetan como V_{CC} (para TTL) o V_{DD} (para los dispositivos MOS).

La cantidad de energía que requiere un CI se determina con base en la corriente I_{CC} (o I_{DD}) que consume de la fuente V_{CC} (o V_{DD}), y la energía real es el producto $I_{CC} \times V_{CC}$. Para muchos CIs, la corriente que se consume de la fuente varía con base en los estados lógicos de los circuitos en el chip. Por ejemplo, la figura 8-3(a) muestra un chip NAND en el cual *todas* las *salidas* de las compuertas están en ALTO. El drenado de corriente en la fuente V_{CC} para este caso se llama I_{CCH}. De igual forma, la figura 8-3(b) muestra la corriente cuando *todas* las *salidas* de las compuertas están en BAJO. Esta corriente se llama I_{CCL}. Los valores siempre se miden con las salidas en circuito abierto (sin carga), ya que el tamaño de la carga también tendrá un efecto sobre I_{CCH}.

En algunos circuitos lógicos I_{CCH} e I_{CCL} tendrán valores distintos. Para estos dispositivos, la corriente promedio se calcula con base en la suposición de que las salidas de las compuertas están en BAJO la mitad del tiempo, y en ALTO la otra mitad del tiempo.

$$I_{CC}\,(\text{prom}) = \frac{I_{CCH} + I_{CCL}}{2}$$

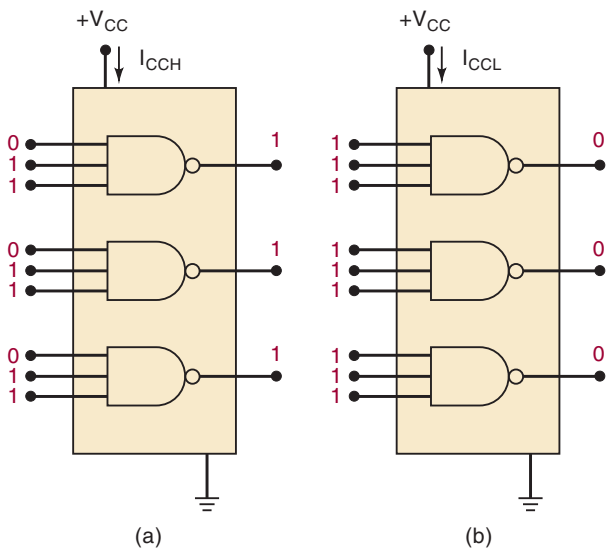


FIGURA 8-3 I_{CCH} e I_{CCL}.

Esta ecuación puede reescribirse para calcular la energía promedio que se disipa:

$$P_D(\text{prom}) = I_{CC}(\text{prom} \times V_{CC})$$

Inmunidad al ruido

Los campos eléctricos y magnéticos dispersos pueden inducir voltajes en los conductores que conectan los circuitos lógicos. Estas señales espurias indeseables se conocen como *ruido* y algunas veces pueden hacer que el voltaje en la entrada de un circuito lógico caiga por debajo de V_{IH}(mín) o se eleve por encima de V_{IL}(máx), lo cual podría producir una operación impredecible. La **inmunidad al ruido** de un circuito lógico se refiere a la habilidad de ese circuito para tolerar el ruido sin producir cambios espurios en el voltaje de salida. A una medida cuantitativa de la inmunidad al ruido se le conoce como **margen de ruido**, el cual se ilustra en la figura 8-4.

La figura 8-4(a) es un diagrama que muestra el intervalo de voltajes que pueden aparecer en la salida de un circuito lógico. Cualquier voltaje mayor que V_{OH}(mín) se considera como un 1 lógico, y cualquier voltaje menor que V_{0L}(máx) se considera como un 0 lógico. Los voltajes en el intervalo indeterminado no deben aparecer en la salida de un circuito lógico, bajo condiciones normales. La figura 8-4(b) muestra los requerimientos de voltaje en la entrada de un circuito lógico. Este circuito responde a cualquier entrada mayor que V_{IH}(mín) como un 1 lógico, y responde a voltajes menores que V_{IL}(máx) como un 0 lógico. Los voltajes en el intervalo indeterminado producen una respuesta impredecible y no deben usarse.

El *margen de ruido de estado alto* V_{NH} se define como

$$V_{NH} = V_{OH}(\text{mín}) - V_{IH}(\text{mín}) \tag{8-1}$$

y se ilustra en la figura 8-4. V_{NH} es la diferencia entre la menor salida en ALTO posible y el mínimo voltaje de entrada requerido para un nivel ALTO. Cuando una salida lógica en ALTO controla la entrada de un circuito lógico, cualquier pico de voltaje negativo mayor que V_{NH} y que aparezca en la línea de señal, podrá hacer que el voltaje caiga en el intervalo indeterminado, en donde puede producirse una operación impredecible.

El *margen de ruido de estado bajo* V_{NL} se define como

$$V_{NL} = V_{IL}(\text{máx}) - V_{OL}(\text{máx}) \tag{8-2}$$

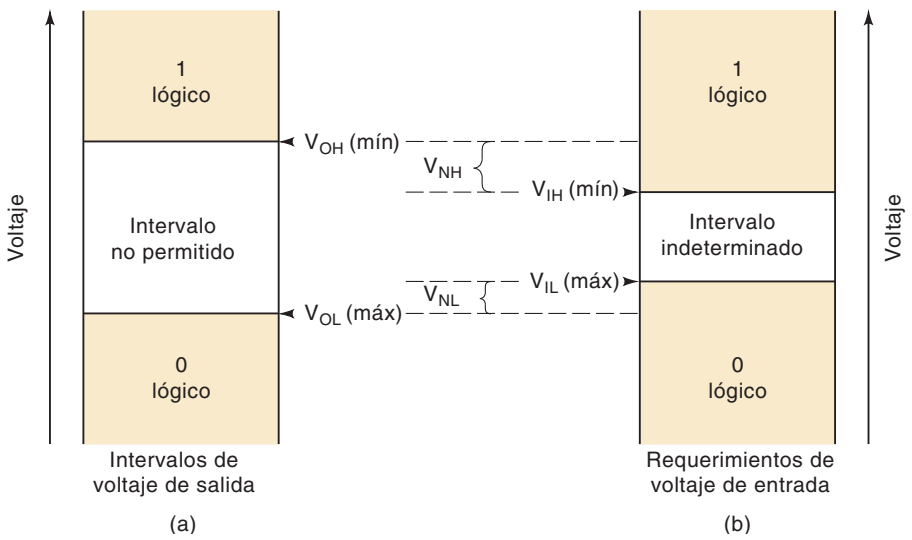


FIGURA 8-4 Márgenes de ruido de corriente directa.

y es la diferencia entre la mayor salida en BAJO posible y el máximo voltaje de entrada requerido para un nivel BAJO. Cuando una salida lógica en BAJO controla una entrada lógica, cualquier pico de ruido positivo mayor que V_{NL} puede hacer que el voltaje se eleve hacia el intervalo indeterminado.

EJEMPLO 8-1

Las especificaciones de voltaje de entrada/salida para la familia TTL estándar se muestran en la tabla 8-1. Utilice estos valores para determinar lo siguiente.

(a) El pico de ruido de máxima amplitud que puede tolerarse cuando una salida en ALTO maneja una entrada.

(b) El pico de ruido de máxima amplitud que puede tolerarse cuando una salida en BAJO maneja una entrada.

TABLA 8-1

Parámetro	Mín (V)	Típico (V)	Máx (V)
V_{OH}	2.4	3.4	
V_{OL}		0.2	0.4
V_{IH}	2.0*		
V_{IL}			0.8*

*Por lo general, sólo se proporcionan los valores de V_{IH} mínimo y V_{IL} máximo.

Solución

(a) Cuando una salida está en ALTO, puede ser tan baja como $V_{OH}(\text{mín}) = 2.4\,\text{V}$. El mínimo voltaje al que puede responder una entrada como ALTO es $V_{IH}(\text{mín}) = 2.0\,\text{V}$. Un pico de ruido negativo puede llevar al voltaje actual por debajo de 2.0 V si su amplitud es mayor que

$$V_{NH} = V_{OH}(\text{mín}) - V_{IH}(\text{mín})$$
$$= 2.4\,\text{V} - 2.0\,\text{V} = 0.4\,\text{V}$$

(b) Cuando una salida está en BAJO, puede ser tan alta como $V_{OL}(\text{máx}) = 0.4\,\text{V}$. El máximo voltaje al que puede responder una entrada como BAJO es $V_{IL}(\text{máx}) = 0.8\,\text{V}$. Un pico de ruido positivo puede llevar al voltaje actual por encima del nivel de 0.8 V, si su amplitud es mayor que

$$V_{NL} = V_{IL}(\text{máx}) - V_{OL}(\text{máx})$$
$$= 0.8\,\text{V} - 0.4\,\text{V} = 0.4\,\text{V}$$

Niveles de voltaje inválidos

Para una operación apropiada, los niveles de voltaje de entrada para un circuito lógico deben mantenerse fuera del intervalo indeterminado que se muestra en la figura 8-4(b); esto es, deben ser menores que $V_{IL}(\text{máx})$ o mayores que $V_{IH}(\text{mín})$. Para las especificaciones de TTL estándar que se proporcionan en el ejemplo 8-1, esto significa que el voltaje de entrada debe ser menor que 0.8 V o mayor que 2.0 V. Un voltaje de entrada entre 0.8 V y 2.0 V se considera un voltaje *inválido* que producirá una respuesta de salida impredecible, por lo cual debe evitarse. En la operación normal, un voltaje de entrada lógico no caerá dentro de la región inválida debido a que proviene de una salida lógica que se encuentra dentro de las especificaciones establecidas. No obstante, cuando esta salida lógica está fallando o se está sobrecargando (es decir, se está excediendo su capacidad de carga), entonces su voltaje puede estar en la región inválida. Los niveles de voltaje inválidos en un circuito digital también pueden producirse debido a voltajes de la fuente de energía que se encuentran fuera del intervalo aceptable. Es importante conocer los intervalos de voltaje válidos para la familia lógica que se está utilizando, de manera que puedan reconocerse las condiciones inválidas cuando se hagan pruebas o se diagnostiquen fallas.

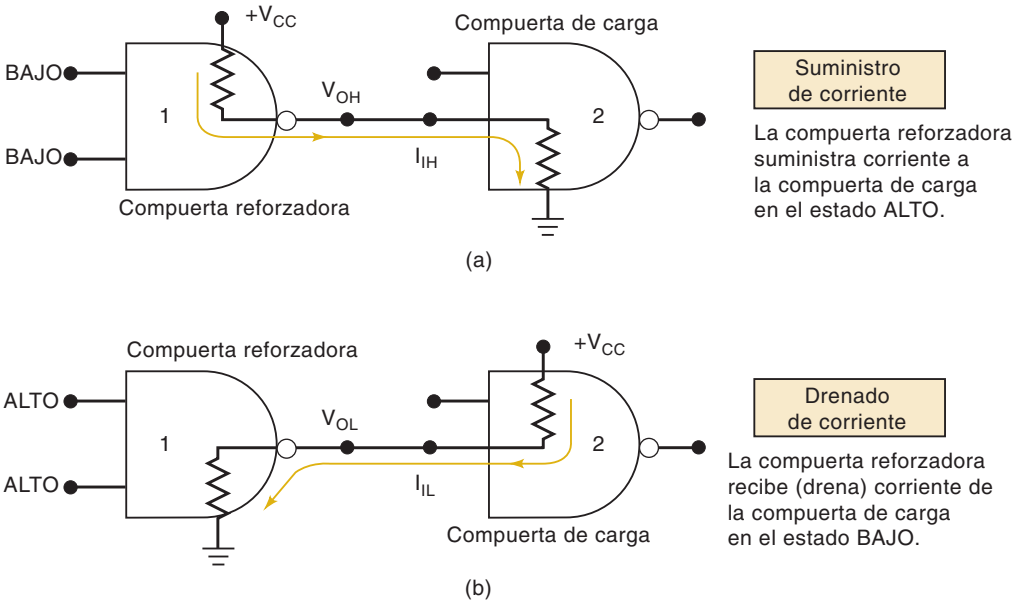


FIGURA 8-5
Comparación de las acciones de suministro de corriente y drenado de corriente.

Acción de suministro de corriente y drenado de corriente

Las familias lógicas pueden describirse de acuerdo con la forma en que fluye la corriente entre la salida de un circuito lógico y la entrada de otro. La figura 8-5(a) ilustra la acción de **suministro de corriente**. Cuando la salida de la compuerta 1 se encuentra en el estado ALTO, suministra una corriente I_{IH} a la entrada de la compuerta 2, la cual actúa en esencia como una resistencia a tierra. Por ende, la salida de la compuerta 1 actúa como un *suministro* de corriente para la entrada de la compuerta 1. Podemos considerar esto como si fuera una llave que actúa como un *suministro* de agua.

La acción de **drenado de corriente** se ilustra en la figura 8-5(b). Aquí el circuito de entrada de la compuerta 2 se representa como una resistencia enlazada a $+V_{CC}$, la terminal positiva de una fuente de energía. Cuando la salida de la compuerta 1 cambie a su estado BAJO, la corriente fluirá en la dirección que se muestra desde el circuito de entrada de la compuerta 2, de vuelta a través de la resistencia de la compuerta 1 hasta tierra. En otras palabras, en el estado BAJO la salida del circuito que maneja la entrada de la compuerta 2 debe poder *drenar* una corriente I_{IL} que proviene de esa entrada. Podemos considerar esto como un *lavabo*, dentro del cual fluye el agua.

La distinción entre suministro de corriente y drenado de corriente es importante, y se volverá más aparente a medida que examinemos las diversas familias lógicas.

Encapsulados de CI

Los desarrollos y avances en los circuitos integrados continúan a un ritmo acelerado. Lo mismo se aplica a los encapsulados de CI. Existen diversos tipos de encapsulados, los cuales difieren en cuanto al tamaño físico, a las condiciones ambientales y de consumo de energía bajo las cuales el dispositivo puede operar en forma confiable, y a la manera en que el encapsulado de CI se monta en el circuito impreso. La figura 8-6 muestra cinco encapsulados de CI representativos.

El paquete en la figura 8-6(a) es el **DIP** (encapsulado dual en línea), el cual ha estado en operación durante un largo tiempo. Sus terminales (o puntas) caen sobre los dos lados extensos del encapsulado rectangular. El dispositivo que se muestra es un DIP de 24 terminales. Observe la presencia de la muesca en un extremo, la cual se utiliza para localizar la terminal 1. Algunos DIPs utilizan un pequeño punto en la superficie superior del encapsulado para localizar la terminal 1. Las terminales se extienden en forma recta hacia fuera del paquete DIP, de manera que el CI pueda insertarse en un zócalo para CI o en agujeros perforados a través de un circuito

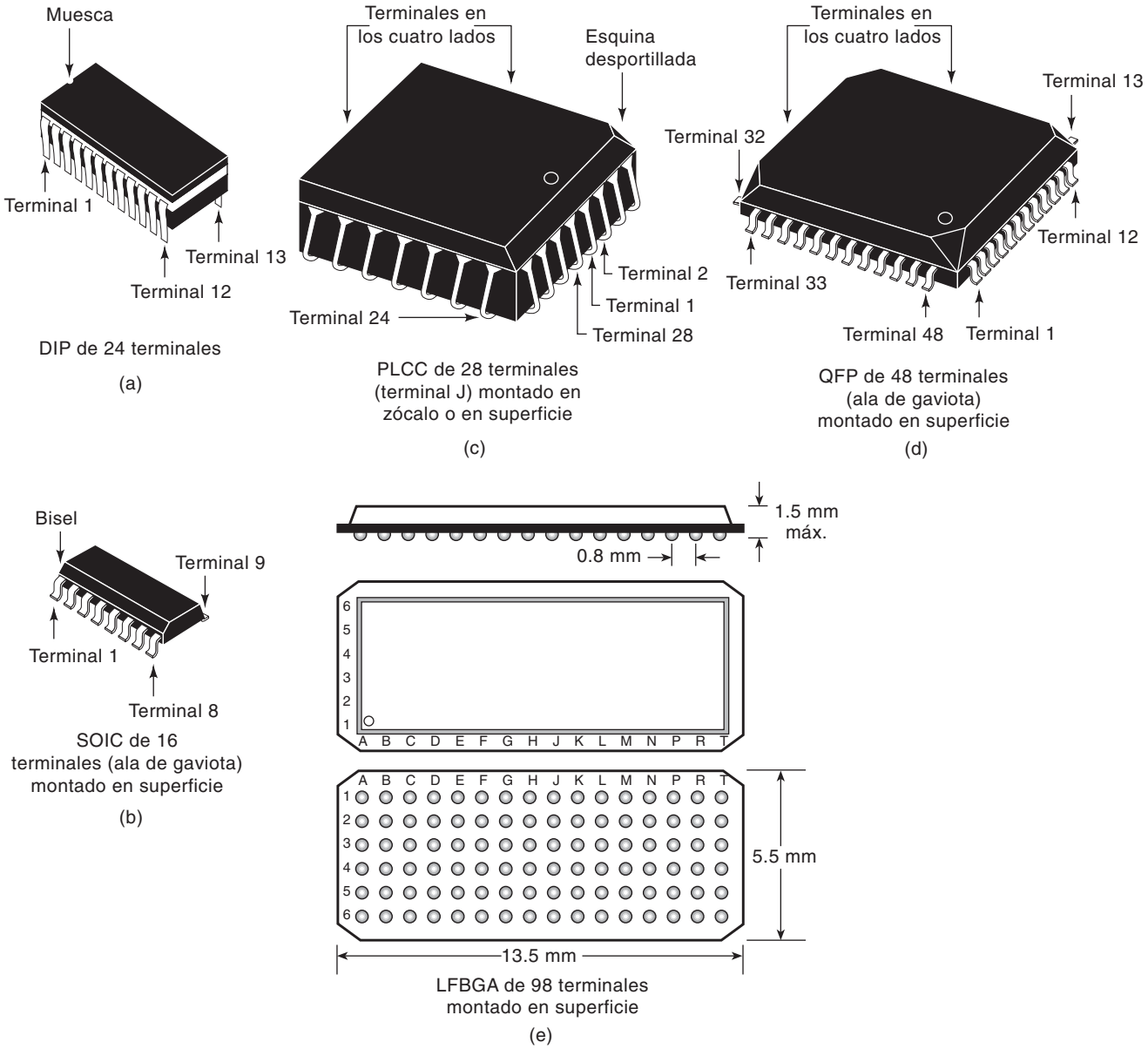


FIGURA 8-6 Encapsulados comunes de CI. (Cortesía de Texas Instruments.)

impreso. El espaciamiento entre terminales (**distancia entre terminales**) es por lo general de 100 mils (un mils es una milésima de pulgada). Los encapsulados DIP siguen siendo los más populares para prototipos, tableros de experimentación (breadboards) y experimentos educativos.

Casi todos las nuevas tarjetas de circuitos que se producen mediante el uso de equipos de fabricación automatizados han dejado de utilizar encapsulados DIP, cuyas terminales se insertan en agujeros en el tablero. Los nuevos métodos de fabricación utilizan la **tecnología de montaje superficial**, la cual coloca un CI en almohadillas conductoras en la superficie del tablero. Se mantienen fijos mediante una pasta de soldadura y todo el tablero se calienta para crear una conexión soldada. La precisión de la máquina de colocación permite un espaciamiento de las terminales muy estrecho. Las terminales en estos encapsulados de montaje superficial se doblan hacia afuera de la cubierta de plástico, con lo cual se proporciona un área superficial adecuada para la unión de soldadura. La forma de estas terminales le ha hecho ganar a este encapsulado el sobrenombre de encapsulado tipo “ala de gaviota”. Se dispone de muchos encapsulados distintos para los dispositivos de mon-

TABLA 8-2 Encapsulados de CI.

Abreviatura	Nombre del encapsulado	Altura	Distancia entre terminales
DIP	Encapsulado dual en línea	200 mils (5.1 mm)	100 mils (2.54 mm)
SOIC	Circuito integrado de pequeño contorno	2.65 mm	50 mils (1.27 mm)
SSOP	Encapsulado reducido de contorno pequeño	2.0 mm	0.65 mm
TSSOP	Encapsulado reducido delgado de contorno pequeño	1.1 mm	0.65 mm
TVSOP	Encapsulado delgado de contorno muy pequeño	1.2 mm	0.4 mm
PLCC	Portador de chip de plástico con terminales	4.5 mm	1.27 mm
QFP	Encapsulado plano de cuatro lados	4.5 mm	0.635 mm
TQFP	Encapsulado plano delgado de cuatro lados	1.6 mm	0.5 mm
LFBGA	Matriz de rejillas de bolas finas de bajo perfil	1.5 mm	0.8 mm

taje superficial. Algunos de los encapsulados más comunes que se utilizan para los CIs lógicos se muestran en la figura 8-6. La tabla 8-2 muestra la definición de cada abreviación, junto con sus dimensiones.

La necesidad de más y más conexiones para un CI complejo ha producido otro encapsulado muy popular, el cual tiene terminales en los cuatro lados del chip. El PLCC tiene terminales en forma de J que se acuñan por debajo del CI, como muestra la figura 8-6(c). Estos dispositivos pueden montarse en la superficie de un circuito impreso, pero también pueden colocarse en un zócalo especial para PLCC. Este zócalo se utiliza con frecuencia para componentes con los que existe la posibilidad de que necesiten reemplazarse para reparación o actualización, como los dispositivos lógicos programables o las unidades centrales de procesamiento en las computadoras. Los encapsulados QFP y TQFP tienen terminales en los cuatro lados en un encapsulado de montaje superficial tipo ala de gaviota, como se muestra en la figura 8-6(d). La matriz de rejillas de bolas (BGA) que se muestra en la figura 8-6(e) es un encapsulado de montaje superficial que ofrece aún más densidad. La matriz de rejillas de terminales (PGA) es un encapsulado similar que se utiliza cuando los componentes deben estar en un zócalo para que puedan extraerse con facilidad. El encapsulado PGA tiene una terminal larga en vez de una bola de contacto (BGA) en cada posición de la rejilla.

La proliferación de equipos portátiles y pequeños para el consumidor, tales como las cámaras de video, los teléfonos celulares, las computadoras (PDAs), los sistemas de audio portátiles y demás dispositivos, ha creado la necesidad de colocar circuitos lógicos en encapsulados muy pequeños. Ahora las compuertas lógicas están disponibles en encapsulados individuales de montaje superficial que contienen una, dos o tres compuertas (1G, 2G y 3G, en forma respectiva). Estos dispositivos pueden tener hasta cinco o seis terminales (energía, tierra, de dos a tres entradas y una salida) y pueden ocupar menos espacio que una sola letra de esta página.

PREGUNTAS DE REPASO

1. Defina cada uno de los siguientes términos: V_{OH}, V_{IL}, I_{OL}, I_{IH}, t_{PLH}, t_{PHL}, I_{CCL}, I_{CCH}.
2. *Verdadero o falso:* si un circuito lógico tiene una capacidad de carga de 5, el circuito tiene cinco salidas.
3. *Verdadero o falso:* el margen de ruido en estado ALTO es la diferencia entre V_{IH}(mín) y V_{CC}.
4. Describa la diferencia entre drenado de corriente y suministro de corriente.
5. ¿Cuál encapsulado de CI puede insertarse en zócalos?
6. ¿Cuál encapsulado tiene sus terminales dobladas debajo del CI?
7. ¿Qué diferencia hay entre los encapsulados de montaje superficial y los DIPs?
8. ¿Funcionará un dispositivo TTL estándar con un nivel de entrada de 1.7 V?

8-2 LA FAMILIA LÓGICA TTL

Al momento de escribir este libro, todavía podían obtenerse muchos CIs de pequeña a mediana escala (SSI y MSI) en las series de la tecnología **TTL** estándar que han estado disponibles durante más de 30 años. Estas series originales de dispositivos y sus descendientes en la familia TTL han tenido una tremenda influencia sobre las características de todos los dispositivos lógicos en la actualidad. Los dispositivos TTL aún se utilizan como lógica de "pegamento" que conecta los dispositivos más complejos en los sistemas digitales. También se utilizan como circuitos de interfase para dispositivos que requieren una operación con alta corriente. A pesar de que la familia TTL bipolar ya prácticamente no es utilizada, comenzaremos nuestra discusión sobre los CIs lógicos con los dispositivos que dieron forma a la tecnología digital.

El circuito lógico TTL básico es la compuerta NAND, la cual se muestra en la figura 8-7(a). Aunque la familia TTL estándar está casi obsoleta, podemos aprender mucho acerca de los miembros más actuales de la familia si estudiamos los circuitos originales en su forma más simple. Las características de las entradas TTL provienen de la configuración de múltiple emisor (unión de diodos) del transistor Q_1. La polarización directa de cualquiera (o ambas) de estas uniones de diodos encenderá a Q_1. Sólo cuando todas las uniones se polaricen en forma inversa se apagará el transistor. Este transistor con entrada de *múltiple emisor* puede tener hasta ocho emisores para una compuerta NAND de ocho entradas.

Observe además que en el lado de salida del circuito, los transistores Q_3 y Q_4 están en un arreglo tipo **tótem**. El tótem está compuesto de dos interruptores de transistor, Q_3 y Q_4. El trabajo de Q_3 es conectar V_{CC} a la salida para crear un nivel ALTO lógico. El trabajo de Q_4 es conectar la salida a tierra para crear un nivel BAJO lógico. Como veremos pronto, en la operación normal Q_3 o Q_4 estarán conduciendo, dependiendo del estado lógico de la salida.

Operación del circuito: estado BAJO

Aunque el circuito se ve en extremo complejo, podemos simplificar su análisis de cierta forma si utilizamos el equivalente en diodos del transistor de múltiple emisor Q_1, como se muestra en la figura 8-7(b). Los diodos D_2 y D_3 representan las dos uniones E-B de Q_1, y D_4 es la unión colector-base (C-B). En el siguiente análisis utilizaremos esta representación para Q_4.

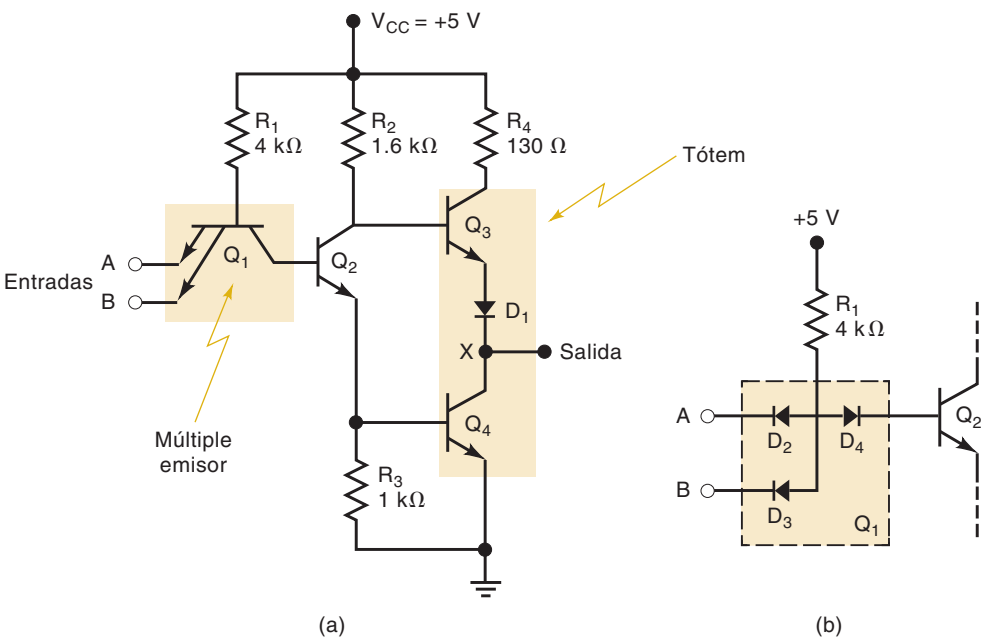
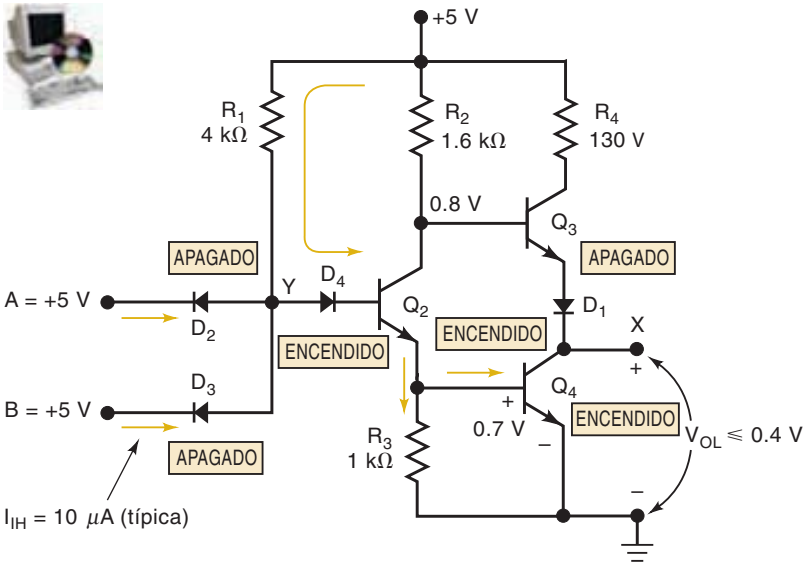


FIGURA 8-7
(a) Compuerta NAND TTL básica; (b) equivalente en diodos para Q_1.

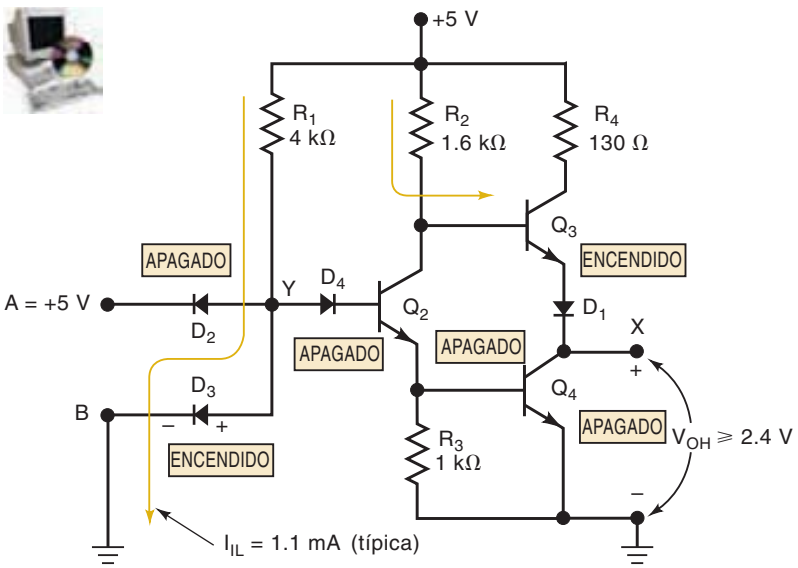
Primero consideremos el caso en el que la salida está en BAJO. La figura 8-8(a) muestra esta situación con ambas entradas A y B en +5 V. Los +5 V en los cátodos de D_2 y D_3 apagarán estos diodos y casi no conducirán corriente. La fuente de +5 V empujará la corriente a través de R_1 y D_4 hacia la base de Q_2, el cual se encenderá. La corriente del emisor de Q_2 fluirá hacia la base de Q_4 y lo encenderá. Al mismo tiempo, el flujo de la corriente de colector de Q_2 producirá una caída de voltaje a través de R_2, con lo que se reducirá el voltaje de colector de Q_2 a un valor bajo que no será suficiente para encender a Q_3.

El voltaje en el colector de Q_2 se muestra como un valor aproximado a 0.8 V. Esto se debe a que el emisor de Q_2 está a 0.7 V con respecto a tierra debido al voltaje positivo en E-B de Q_4, y el colector de Q_2 está a 0.1 V con respecto a su emisor, debido a



Condiciones de entrada	Condiciones de salida
A y B están ambas en ALTO (≥ 2 V)	Q_3 APAGADO
Las corrientes de entrada son muy bajas I_{IH} = 10 μA	Q_4 está ENCENDIDO, por lo que V_X está en BAJO (≤ 0.4 V)

(a) Salida en BAJO



Condiciones de entrada	Condiciones de salida
A o B o ambas están en BAJO (≤ 0.8 V)	Q_4 APAGADO
La corriente fluye de vuelta a tierra a través de la terminal de entrada en BAJO. I_{IL} = 1.1 mA	Q_3 actúa como seguidor de emisor y VOH ≥ 2.4 V, por lo general 3.5 V

(b) Salida en ALTO

FIGURA 8-8 Compuerta NAND TTL en sus dos estados de salida.

V_{CE}(sat). Estos 0.8 V en la base de Q_3 no son suficientes para la polarización directa de la unión E-B de Q_3 y el diodo D_1. De hecho, D_1 se necesita para mantener a Q_3 apagado en esta situación.

Con Q_4 encendido, la terminal de salida X estará a un voltaje muy bajo debido a que la resistencia en el estado ENCENDIDO de Q_4 será baja (de 1 a 25 Ω). En realidad, el voltaje de salida V_{OL} dependerá de cuánta corriente de colector conduzca Q_4. Con Q_3 apagado no hay corriente que provenga de la terminal de +5 V a través de R_4. Como veremos, la corriente de colector de Q_4 llegará de las entradas TTL a las que está conectada la terminal X.

Es importante observar que las entradas en ALTO en A y B tendrán que suministrar sólo una corriente de fuga muy pequeña en el diodo. Por lo general, esta corriente I_{IH} es tan sólo de un valor aproximado a los 10 μA, a temperatura ambiente.

Operación del circuito: estado ALTO

La figura 8-8(b) muestra la situación en la que la salida del circuito está en ALTO. Esta situación puede producirse si se conecta una de las entradas (o ambas) a un nivel BAJO. Aquí, la entrada B está conectada a tierra. Esto producirá la polarización directa de D_3, de forma que la corriente fluirá de la terminal de alimentación de +5 V a través de R_1 y D_3, y de la terminal B a tierra. El voltaje positivo a través de D_3 retendrá el punto Y a un valor aproximado a los 0.7 V. Este voltaje no es suficiente para polarizar en forma directa a D_4 y la unión E-B de Q_2 lo necesario como para la conducción.

Con Q_2 apagado no hay corriente base para Q_4, por lo cual se apaga. Como no hay corriente de colector en Q_2, el voltaje en la base de Q_3 será lo suficientemente grande como para polarizar en forma directa a Q_3 y a D_1, de manera que Q_3 conducirá. En realidad, Q_3 actúa como un seguidor de emisor, ya que en esencia la terminal de salida X se encuentra en su emisor. Sin carga conectada del punto X a tierra, V_{OH} tendrá un valor aproximado entre 3.4 y 3.8 V, ya que se restan dos caídas de 0.7 V en los diodos (E-B de Q_3 y D_1) de los 5 V que se aplican a la base de Q_3. Este voltaje disminuirá bajo una carga, ya que ésta drenará la corriente del emisor de Q_3, el cual drena la corriente de la base a través de R_2 y, por lo tanto, se incrementa la caída de voltaje a través de R_2.

Es importante observar que hay una corriente sustancial que fluye de vuelta a través de la terminal de entrada B a tierra, cuando B se mantiene en BAJO. Esta corriente I_{IL} se determina con base en el valor de la resistencia R_1, la cual variará de serie a serie. Para la familia TTL estándar, es de un valor aproximado a 1.1 mA. La entrada B en BAJO actúa como un drenador a tierra para esta corriente.

Acción de drenado de corriente

Una salida TTL actúa como un drenador de corriente en el estado BAJO, ya que *recibe* corriente de la entrada de la compuerta que está controlando. La figura 8-9 muestra una compuerta TTL que maneja la entrada de otra compuerta (la carga) para ambos estados de voltaje de salida. En la situación del estado de salida en BAJO que se describe en la figura 8-9(a), el transistor Q_4 de la compuerta reforzadora está encendido y en esencia crea un "corto" entre X y tierra. Este BAJO voltaje en X polariza en forma directa la unión emisor-base de Q_1 y la corriente fluye, como se muestra, de vuelta a través de Q_4. Por ende, Q_4 está realizando una acción de drenado de corriente, la cual deriva su corriente de la de entrada (I_{IL}) de la compuerta de carga. Con frecuencia nos referiremos a Q_4 como el **transistor de drenado de corriente**, o como el **transistor de puesta en bajo**, ya que reduce el voltaje de salida hasta su estado BAJO.

Acción de suministro de corriente

En el estado ALTO, una salida TTL actúa como un suministro de corriente. Esto se muestra en la figura 8-9(b), en donde el transistor Q_3 suministra la corriente de entrada I_{IH} que requiere el transistor Q_1 de la compuerta de carga. Como se

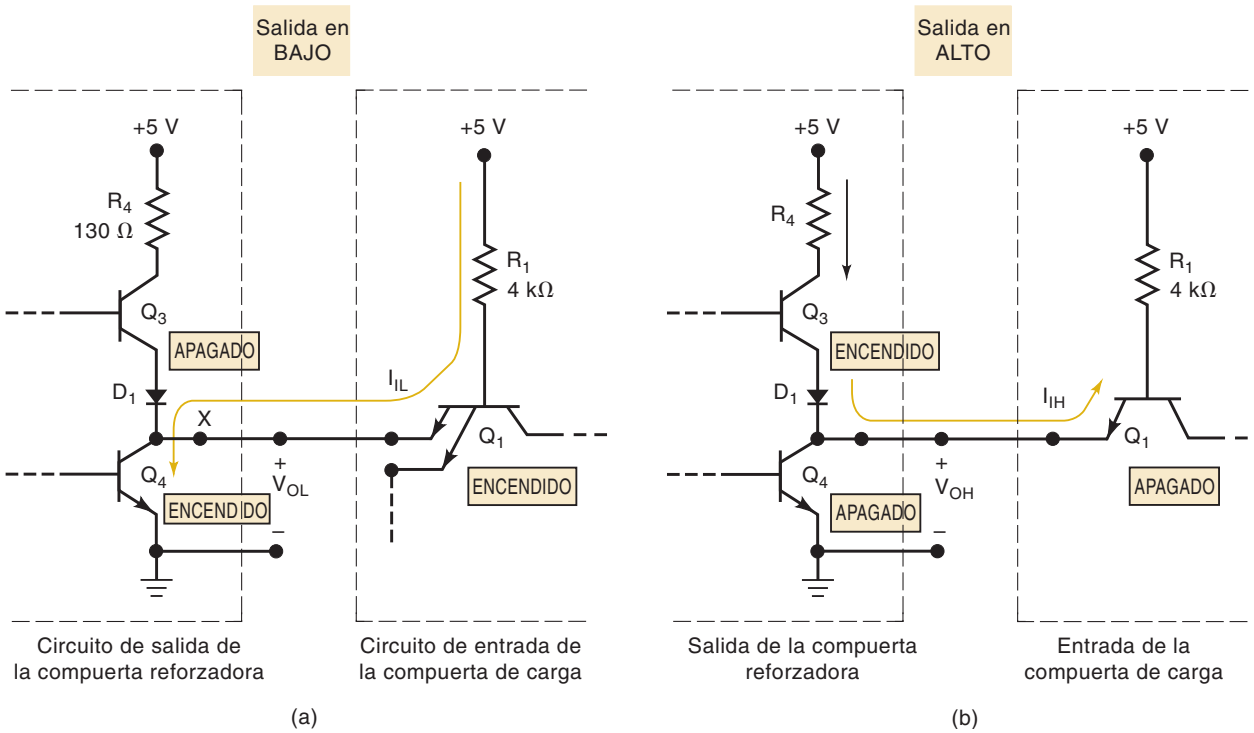


FIGURA 8-9 (a) Cuando la salida TTL se encuentra en el estado BAJO, Q_4 actúa como drenador de corriente y deriva su corriente de la carga. (b) En el estado de salida ALTO, Q_3 actúa como suministro de corriente para la compuerta de carga.

dijo antes, ésta es una pequeña corriente de fuga con polarización inversa (por lo general 10 μA). A menudo nos referiremos a Q_3 como el **transistor de suministro de corriente** o **transistor de puesta en alto**. En algunas de las series TTL más modernas, el circuito de puesta en alto está compuesto de dos transistores, en vez de un transistor y un diodo.

Circuito de salida en forma de tótem

Debemos mencionar varios puntos en relación con el arreglo en forma de tótem del circuito de salida TTL, como se muestra en la figura 8-9, ya que no está claro por qué se utiliza. Puede lograrse la misma lógica si se eliminan Q_3 y D_1, y se conecta la parte inferior de R_4 al colector de Q_4. Pero este arreglo significaría que Q_4 tendría que conducir una corriente muy pesada en su estado de saturación (5 V/130 Ω ≈ 40 mA). Con Q_3 en el circuito no habrá corriente a través de R_4 en el estado de salida BAJO. Esto es importante ya que mantiene la disipación de energía del circuito en un nivel bajo.

En el estado de salida ALTO ocurre otra ventaja de este arreglo. Aquí, Q_3 actúa como un seguidor de emisor con su impedancia de salida baja asociada (por lo general de 10 Ω). Esta impedancia de salida baja proporciona una constante de tiempo corta para cargar cualquier carga capacitiva en la salida. Esta acción, que por lo general se le conoce como *ascenso (pull-up) activo*, proporciona formas de onda con un tiempo de elevación muy rápido en las salidas TTL.

Una desventaja del arreglo de salida en forma de tótem se produce durante la transición de BAJO a ALTO. Desafortunadamente, Q_4 se apaga con más lentitud que la velocidad con que Q_3 se enciende, por lo que hay un periodo de unos cuantos nanosegundos, durante los cuales ambos transistores están conduciendo y se drena una corriente relativamente grande (de 30 a 40 mA) de la fuente de 5 V. Esto puede presentar un problema que examinaremos después.



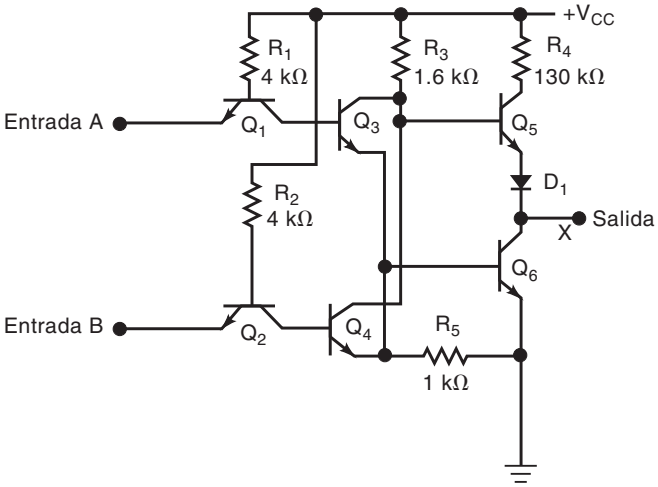


FIGURA 8-10 Circuito TTL de compuerta NOR.

Compuerta NOR TTL

La figura 8-10 muestra el circuito interno para una compuerta NOR TTL. No realizaremos un análisis detallado de este circuito, pero es importante observar cómo se compara con el circuito NAND de la figura 8-8. En el lado de entrada podemos ver que el circuito NOR *no utiliza un* transistor con *múltiple emisor*; en vez de ello, cada entrada se aplica al emisor de un transistor separado. En el lado de salida, el circuito NOR utiliza el mismo arreglo en forma de tótem que el circuito NAND.

Resumen

Todos los circuitos TTL tienen una estructura similar. Las compuertas NAND y AND utilizan entradas de transistor con múltiple emisor o uniones con varios diodos; las compuertas NOR y OR utilizan transistores de entrada separados. En cualquier caso, la entrada será el cátodo (región N) de una unión P-N, por lo que un voltaje de entrada en ALTO desconectará la unión y sólo fluirá una pequeña corriente de fuga (I_{IH}). En contraste, un voltaje de entrada en BAJO enciende la unión y en consecuencia, fluirá una corriente relativamente grande (I_{IL}) a través de la fuente de la señal. La mayoría de los circuitos TTL tendrán algún tipo de configuración de salida en forma de tótem. Hay algunas excepciones que veremos más adelante.

PREGUNTAS DE REPASO

1. *Verdadero o falso:* una salida TTL actúa como un drenador de corriente en el estado BAJO.
2. ¿En qué estado de entrada TTL fluye la mayor cantidad de corriente de entrada?
3. Indique las ventajas y desventajas de una salida en forma de tótem.
4. ¿Cuál transistor TTL es el transistor de puesta en alto en el circuito NAND?
5. ¿Cuál transistor TTL es el transistor de puesta en bajo en el circuito NOR?
6. ¿Qué diferencia hay entre los circuitos TTL NOR y NAND?

8-3 HOJAS TÉCNICAS TTL

En 1964, Texas Instruments Corporation introdujo la primera línea de CIs TTL estándar. La serie conocida como 54/74 ha sido una de las familias lógicas de CIs más utilizadas. Nosotros nos referiremos a ella simplemente como la serie 74, ya que la principal diferencia entre las versiones 54 y 74 es que los dispositivos en la serie 54

pueden operar sobre un intervalo más amplio de temperaturas y voltajes de alimentación. Muchos fabricantes de semiconductores aún producen CIs TTL. Por fortuna todos utilizan el mismo sistema de numeración, por lo que el número CI básico es el mismo de en cualquier fabricante. No obstante, cada fabricante, por lo general, adjunta su propio prefijo especial al número del CI. Por ejemplo, Texas Instruments utiliza el prefijo SN, Nacional Semiconductor utiliza DM y Signetics utiliza la S. Así, dependiendo del fabricante, usted podrá ver un chip de compuerta NOR cuádruple etiquetado como DM7402, SN7402, S7402, o con cualquier otra designación similar. La parte importante es el número 7402, que es el mismo para todos los fabricantes.

Como vimos en el capítulo 4, hay varias series en la familia TTL de dispositivos lógicos (74, 74LS, 74S, etc.). Los fabricantes ya no recomiendan el uso de la serie estándar original y sus descendientes inmediatos (74, 74LS, 74S) en los nuevos diseños. A pesar de todo hay una demanda en el mercado suficiente como para mantenerlos en producción. Es vital tener una comprensión de las características que definen las capacidades y limitaciones de cualquier dispositivo lógico. En esta sección definiremos esas características mediante el uso de la serie Schottky (ALS) de baja potencia y le ayudaremos a interpretar una hoja técnica ordinaria. Más adelante introduciremos las demás series TTL y compararemos sus características.

Para encontrar toda la información que necesitemos sobre un CI podemos consultar las hojas técnicas publicadas por el fabricante para esa familia de CIs específica. Podemos obtener estas hojas técnicas de los hojas de datos, CD-ROMs o del sitio Web del fabricante de CIs. La figura 8-11 es la hoja técnica del fabricante para el CI de compuerta NAND 74ALS00, en la cual se muestran las condiciones de operación recomendadas, las características eléctricas y las de conmutación. En esta hoja técnica encontrará la mayoría de los parámetros que veremos en los siguientes párrafos. A medida que veamos cada una de los parámetros, será conveniente que consulte esta hoja técnica para que vea de dónde provino la información.

Voltaje de alimentación e intervalo de temperatura

Tanto la serie 74ALS como la serie 54ALS utilizan un voltaje de alimentación nominal (V_{CC}) de 5 V, pero pueden tolerar una variación en ésta de 4.5 a 5.5 V. La serie 74ALS está diseñada para operar en forma apropiada en temperaturas ambientales que varían de 0 a 70°C, mientras que la serie 54ALS puede manejar temperaturas de −5 a +125°C. Debido a su mayor tolerancia a las variaciones en voltaje y temperatura, la serie 54ALS es más costosa; se emplea sólo en aplicaciones en las que debe mantenerse una operación confiable sobre una extensa variedad de condiciones. Algunos ejemplos son las aplicaciones militares y espaciales.

Niveles de voltaje

En la hoja técnica de la figura 8-11 encontrará los niveles de voltaje lógicos de entrada y salida para la serie 74ALS. La tabla 8-3 los presenta en forma sintetizada. Los valores mínimo y máximo que se muestran son para las condiciones de la fuente de energía, temperatura y condiciones de carga para el peor de los casos. Una inspección de la tabla nos revela un voltaje de salida en 0 lógico máximo garantizado V_{OL} = 0.5 V, el cual es 300 mV menor que el voltaje en 0 lógico necesario en la entrada V_{IL} = 0.8 V. Esto significa que el margen de ruido de corriente directa en estado BAJO garantizado es de 300 mV. Es decir,

$$V_{NL} = V_{IL}(\text{máx}) - V_{OL}(\text{máx}) = 0.8\,\text{V} - 0.5\,\text{V} = 0.3\,\text{V} = 300\,\text{mV}$$

De manera similar, se garantiza un valor mínimo de 2.5 V en la salida en 1 lógico V_{OH}, la cual es 500 mV mayor que el voltaje en 1 lógico necesario en la entrada, V_{IH} = 2.0 V. Por lo tanto, el margen de ruido de corriente directa en estado ALTO es de 500 mV.

$$V_{NH} = V_{OH}(\text{mín}) - V_{IH}(\text{mín}) = 2.5\,\text{V} - 2.0\,\text{V} = 0.5\,\text{V} = 500\,\text{mV}$$

Por lo tanto, el margen de ruido de corriente directa *garantizado para el peor de los casos* para la serie 74ALS es de 300 mV.

condiciones de operación recomendadas

		SN54ALS00A			SN74ALS00A			UNIDAD
		MÍN	NOM	MÁX	MÍN	NOM	MÁX	
V_{CC}	Voltaje de alimentación	4.5	5	5.5	4.5	5	5.5	V
V_{IH}	Voltaje de entrada de nivel alto	2			2			V
V_{IL}	Voltaje de entrada de nivel bajo			0.8‡			0.8	V
				0.7§				
I_{OH}	Corriente de salida de nivel alto			−0.4			−0.4	mA
I_{OL}	Corriente de salida de nivel bajo			4			8	mA
T_A	Temperatura de operación al aire libre	−55		125	0		70	°C

‡ Se aplica sobre el intervalo de temperatura de −55°C a 70°C.
§ Se aplica sobre el intervalo de temperatura de 70°C a 125°C.

características eléctricas sobre el intervalo de temperatura de operación al aire libre, a menos que se indique lo contrario

PARÁMETRO	CONDICIONES DE PRUEBA		SN54ALS00A			SN74ALS00A			UNIDAD
			MÍN	TIP†	MÁX	MÍN	TIP†	MÁX	
V_{IK}	V_{CC} = 4.5 V,	I_I = −18 mA			−1.2			−1.5	V
V_{OH}	V_{CC} = 4.5 V to 5.5 V,	I_{OH} = −0.4 mA	V_{CC} −2			V_{CC} −2			V
V_{OL}	V_{CC} = 4.5 V	I_{OL} = 4 mA		0.25	0.4		0.25	0.4	V
		I_{OL} = 8 mA					0.35	0.5	
I_I	V_{CC} = 5.5 V,	V_I = 7 V			0.1			0.1	mA
I_{IH}	V_{CC} = 5.5 V,	V_I = 2.7 V			20			20	µA
I_{IL}	V_{CC} = 5.5 V,	V_I = 0.4 V			−0.1			−0.1	mA
I_O‡	V_{CC} = 5.5 V,	V_O = 2.25 V	−20		−112	−30		−112	mA
I_{CCH}	V_{CC} = 5.5 V,	V_I = 0		0.5	0.85		0.5	0.85	mA
I_{CCL}	V_{CC} = 5.5 V,	V_I = 4.5 V		1.5	3		1.5	3	mA

† Todos los valores típicos están a V_{CC} = 5V, T_A = 25°C
‡ Se han elegido las condiciones de salida de forma que se produzca una corriente que se aproxime muy de cerca a la mitad de la corriente de salida de corto circuito IOS.

características de conmutación (vea la figura 1)

PARÁMETRO	DESDE (ENTRADA)	HACIA (SALIDA)	V_{CC} = 4.5 V a 5.5 V, C_L = 50 pF, R_L = 500 Ω, T_A = MÍN a MÁX §				UNIDAD
			SN54ALS00A		SN74ALS00A		
			MÍN	MÁX	MÍN	MÁX	
t_{PLH}	A o B	Y	3	15	3	11	ns
t_{PHL}			2	9	2	8	

§ Para las condiciones que se muestran como MÍN o MÁX, utilice el valor apropiado que se especifica bajo las condiciones de operación recomendada.

FIGURA 8-11 Hoja técnica para el CI de compuerta NAND 74ALS00. (Cortesía de Texas Instruments.)

TABLA 8-3 Niveles de voltaje de la serie 74ALS.

	Mínimo	Típico	Máximo
V_{OL}	—	0.35	0.5
V_{OH}	2.5	3.4	—
V_{IL}	—	—	0.8
V_{IH}	2.0	—	—

Clasificaciones máximas de voltaje

Los valores de voltaje en la tabla 8-3 *no incluyen* las clasificaciones máximas absolutas más allá de las cuales puede atrofiarse la vida útil del CI. Por lo general, las condiciones de operación máximas absolutas se proporcionan en la parte superior de una hoja técnica (que no se muestra en la figura 8-11). Los voltajes que se aplican a cualquier entrada del CI de esta serie nunca deben exceder de +7.0 V. Un voltaje

mayor de +7.0 V que se aplique a un emisor de entrada puede provocar un rompimiento inverso de la unión E-B de Q_1.

También hay un límite en el voltaje máximo *negativo* que puede aplicarse a una entrada TTL. Este límite de −0.5 V se produce debido al hecho de que la mayoría de los circuitos TTL emplean diodos protectores en paralelo en cada entrada. Omitimos estos diodos a propósito en nuestro análisis anterior, ya que no entran en la operación normal del circuito. Se conectan de cada entrada a tierra para limitar las excursiones de voltaje de entrada negativo que ocurren con frecuencia cuando las señales lógicas tienen sobretiros excesivos. Con estos diodos no debemos aplicar más de −0.5 V a una entrada, ya que los diodos protectores comenzarían a conducir y drenarían una cantidad considerable de corriente; tal vez el diodo podría quedar en corto y se produciría una entrada dañada en forma permanente.

Disipación de potencia

Una compuerta NAND TTL ALS disipa una potencia promedio de 2.4 mW. Esto se debe a que I_{CCH} = 0.85 mA e I_{CCL} = 3 mA, lo cual produce I_{CC}(prom) = 1.93 A y P_D(prom) = 1.93 mA × 5 V = 9.65 mW. Estos 9.65 mW son la potencia total requerida por las cuatro compuertas en el chip. Por ende, una compuerta NAND requiere una potencia promedio de 2.4 mW.

Tiempos de propagación

La hoja de datos proporciona los tiempos de propagación mínimo y máximo. Suponiendo que el valor típico es una cantidad intermedia, tenemos que t_{PLH} = 7 ns y t_{PHL} = 5 ns. El retraso de propagación *promedio* total t_{pd}(prom) = 6 ns.

EJEMPLO 8-2

Consulte la hoja de datos para el CI NAND cuádruple de dos entradas 74ALS00 de la figura 8-11. Determine la *máxima* disipación de potencia promedio y el *máximo* retraso de propagación promedio de *una sola* compuerta.

Solución

En las características eléctricas busque los valores *máximos* de I_{CCH} e I_{CCL}. Los valores son 0.85 mA y 3 mA, respectivamente. Por lo tanto, la I_{CC} promedio es de 1.9 mA. La potencia promedio se obtiene al multiplicar por V_{CC}. La hoja técnica indica que estos valores de I_{CC} se obtuvieron cuando V_{CC} se encontraba en su valor máximo (5.5 V para la serie 74ALS). Por ende, tenemos que

$$P_D(\text{prom}) = 1.9 \text{ mA} \times 5.5 \text{ V} = 10.45 \text{ mW}$$

como la potencia que disipa *todo* el CI. Podemos determinar la disipación de potencia de una compuerta NAND si dividimos esta cantidad entre 4:

$$P_D(\text{prom}) = 2.6 \text{ mW por compuerta}$$

Como este parámetro se calculó utilizando los valores máximos de corriente y de voltaje, es la potencia promedio máxima que disipará una compuerta NAND 74ALS00 bajo las peores condiciones. A menudo los diseñadores utilizan estos valores para asegurar que sus circuitos funcionen bajo todas las condiciones.

Los tiempos de propagación máximos para una compuerta NAND 74ALS00 se listan como

$$t_{PLH} = 11 \text{ ns} \qquad t_{PHL} = 8 \text{ ns}$$

así, el tiempo de propagación promedio máximo es

$$t_{pd}(\text{prom}) = \frac{11 + 8}{2} = 9.5\,\text{ns}$$

De nuevo, éste es el máximo tiempo de propagación promedio posible para el peor de los casos.

8-4 CARACTERÍSTICAS DE LAS SERIES TTL

La serie estándar 74 de TTL ha evolucionado a varias series más. Todas ellas ofrecen una amplia variedad de compuertas y flip-flops en la línea de integración a pequeña escala (SSI), y contadores, registros, multiplexores, decodificadores/codificadores y demás funciones lógicas en su línea de integración a mediana escala (MSI). Las siguientes series TTL (a las que con frecuencia se les llama "subfamilias") proporcionan un amplio intervalo de velocidad y potencia.

TTL estándar, serie 74

En la sección 8-2 hablamos sobre la serie 74 estándar original de la lógica TTL. Estos dispositivos siguen disponibles, pero en la mayoría de los casos no son una opción razonable para los nuevos diseños, ya que ahora están disponibles otros dispositivos con un rendimiento mucho mayor y a un menor costo.

TTL Schottky, serie 74S

La serie 7400 opera mediante el uso de la conmutación saturada, en la cual muchos de los transistores al conducir estarán en la condición saturada. Esta operación produce un retraso en el tiempo de almacenamiento t_S cuando los transistores cambian de ENCENDIDO a APAGADO, y limita la velocidad de conmutación del circuito.

La serie 74S reduce este retraso en el tiempo de almacenamiento al no permitir que el transistor caiga tanto en saturación. Para ello utiliza un diodo de barrera Schottky (SBD), el cual se conecta entre la base y el colector de cada transistor, como se muestra en la figura 8-12(a). El SBD tiene un voltaje positivo de sólo 0.25 V. Por ende, cuando la unión C-B se polariza en forma directa al comienzo de la saturación, el SBD conducirá y desviará parte de la corriente de entrada de la base. Esto reduce el exceso de corriente base y disminuye el retraso de tiempo de almacenamiento al momento en que el transistor se apague.

Como se muestra en la figura 8-12(a), la combinación de transistor/SBD recibe un símbolo especial. Este símbolo se utiliza para todos los transistores en el diagrama del circuito para la compuerta NAND 74S00 que se muestra en la figura 8-12(b). Esta compuerta NAND 74S00 tiene un retraso de propagación promedio de sólo 3 ns, lo cual es seis veces más rápido que el 7400. Observe la presencia de los diodos limitadores D_1 y D_2 para limitar los voltajes de entrada negativos.

Los circuitos en la serie 74S también utilizan valores de resistencias menores para ayudar a mejorar los tiempos de conmutación. Esto incrementa la disipación de potencia promedio del circuito a un valor aproximado de 20 mW, casi dos veces más que la serie 74. Los circuitos 74S también utilizan un par Darlington (Q_3 y Q_4) para proveer un tiempo de elevación de salida más corto cuando cambian de ENCENDIDO a APAGADO.

TTL Schottky de baja potencia, serie 74LS (LS-TTL)

La serie 74LS es una versión de menor velocidad y potencia que la serie 74S. Utiliza el transistor Schottky-clamped, pero con valores de resistencia más grandes que la serie 74S. Estos valores de resistencia mayores reducen el requerimiento de energía del circuito, pero a expensas de un incremento en los tiempos de conmutación. Por

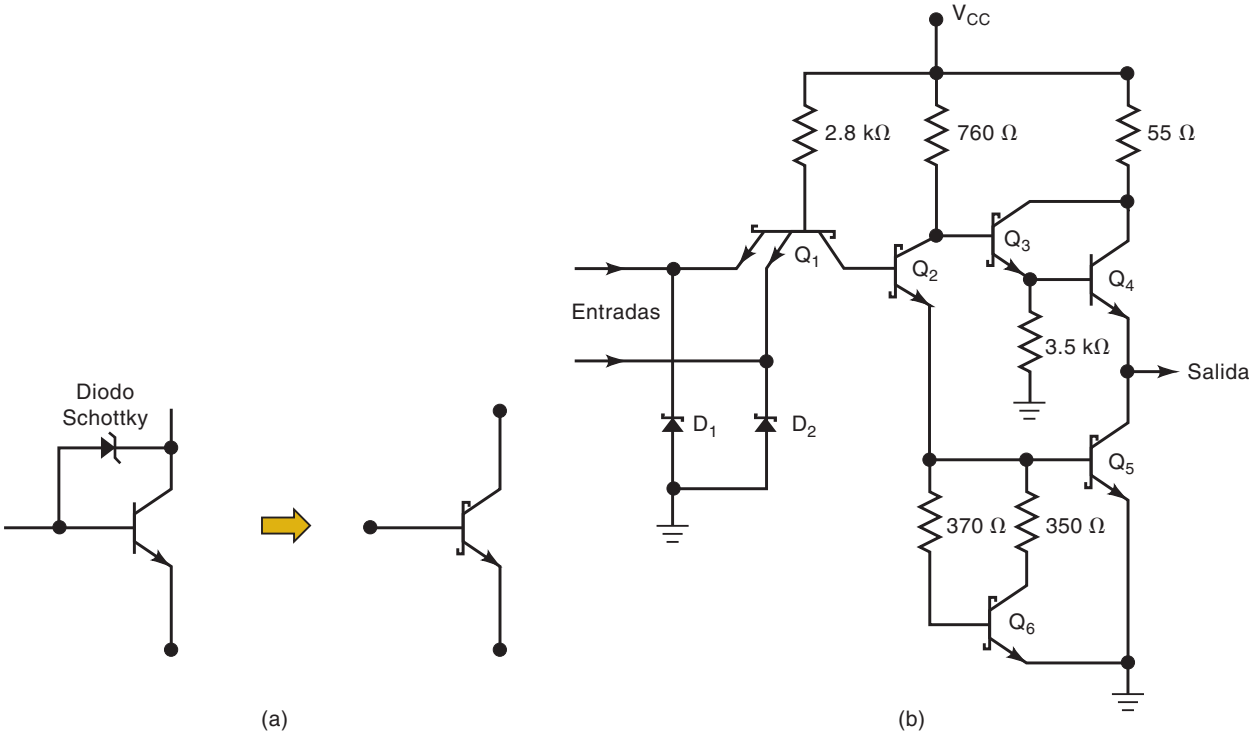


FIGURA 8-12 (a) Transistor Schottky-clamped; (b) compuerta NAND básica en la serie S-TTL.

lo general, una compuerta NAND en la serie 74LS tendrá un retraso de propagación promedio de 9.5 ns y una disipación de potencia promedio de 2 mW.

TTL Schottky avanzado, serie 74AS (AS-TTL)

Las innovaciones en el diseño de circuitos integrados condujeron al desarrollo de dos series TTL mejoradas: Schottky avanzado (74AS) y Schottky avanzado de baja potencia (74ALS). La serie 74AS proporciona una mejora considerable en velocidad, en comparación con la serie 74S, con un requerimiento mucho menor de energía. La tabla 8-4 muestra la comparación para una compuerta NAND en cada serie. Esta comparación muestra con claridad la ventaja de la serie 74AS: la serie TTL es más rápida, y su disipación de potencia es mucho menor que la de la serie 74S. La serie 74AS tiene otras mejoras, incluyendo requerimientos menores de corriente de entrada (I_{IL}, I_{IH}), lo cual produce una capacidad de carga mayor que el de la serie 74S.

TTL Schottky avanzado de baja potencia, serie 74ALS

Ésta es una versión mejorada de la serie 74LS, tanto en velocidad como en disipación de potencia, según los números mostrados en la tabla 8-5. La serie 74ALS tiene la menor disipación de potencia de las compuertas de todas las series TTL.

TABLA 8-4

	74S	74AS
Tiempo de propagación	3 ns	1.7 ns
Disipación de potencia	20 mW	8 mW

TABLA 8-5

	74LS	74ALS
Tiempo de propagación	9.5 ns	4 ns
Disipación de potencia	2 mW	1.2 mW

74F: TTL rápida

Esta serie utiliza una nueva técnica de fabricación de circuitos integrados para reducir las capacitancias entre dispositivos y por ende se logra reducir los tiempos de propagación. Una compuerta NAND ordinaria tiene un tiempo de propagación promedio de 3 ns y un consumo de energía de 6 mW. Los CIs en esta serie están designados con la letra F en su número de pieza. Por ejemplo, el 74F04 es un chip inversor hexadecimal.

Comparación de las características de las series TTL

La tabla 8-6 proporciona los valores típicos para algunas de las características más importantes de cada una de las series TTL. Todas las clasificaciones de rendimiento, excepto la clasificación de frecuencia máxima de reloj, son para una compuerta NAND en cada serie. La frecuencia de reloj máxima se especifica como la máxima frecuencia que puede utilizarse para conmutar un flip-flop J-K. Esto nos proporciona una medida útil del intervalo de frecuencia a través del cual puede operarse cada una de las series de CIs.

TABLA 8-6 Características típicas de las series TTL.

	74	74S	74LS	74AS	74ALS	74F
Clasificaciones de rendimiento						
Tiempo de propagación (ns)	9	3	9.5	1.7	4	3
Disipación de potencia (mW)	10	20	2	8	1.2	6
Frecuencia máxima de reloj (MHz)	35	125	45	200	70	100
Capacidad de carga (misma serie)	10	20	20	40	20	33
Parámetros de voltaje						
V_{OH}(mín)	2.4	2.7	2.7	2.5	2.5	2.5
V_{OL}(máx)	0.4	0.5	0.5	0.5	0.5	0.5
V_{IH}(mín)	2.0	2.0	2.0	2.0	2.0	2.0
V_{IL}(máx)	0.8	0.8	0.8	0.8	0.8	0.8

EJEMPLO 8-3

Utilice la tabla 8-6 para calcular los márgenes de ruido de corriente directa para un CI 74LS ordinario. ¿Cómo se compara esto con los márgenes de ruido de la familia TTL estándar?

Solución

74LS

$$V_{NH} = V_{OH}(\text{mín}) - V_{IH}(\text{mín}) = 2.7\text{ V} - 2.0\text{ V} = 0.7\text{ V}$$

$$V_{NL} = V_{IL}(\text{máx}) - V_{OL}(\text{máx}) = 0.8\text{ V} - 0.5\text{ V} = 0.3\text{ V}$$

74

$$V_{NH} = 2.4\text{ V} - 2.0\text{ V} = 0.4\text{ V}$$

$$V_{NL} = 0.8\text{ V} - 0.4\text{ V} = 0.4\text{ V}$$

EJEMPLO 8-4

¿Cuál de las series TTL puede manejar la mayor cantidad de entradas de dispositivos de la misma serie?

Solución

La serie 74AS tiene la capacidad de carga más alto (40), lo cual significa que una compuerta NAND 74AS00 puede manejar 40 entradas de otros dispositivos 74AS. SI

queremos determinar el número de entradas que puede soportar una salida de una serie TTL *distinta*, tendremos que conocer las corrientes de entrada y de salida de las dos series. En la siguiente sección trataremos este tema.

PREGUNTAS DE REPASO

1. (a) ¿Cuál de las series TTL es la mejor en altas frecuencias?
   (b) ¿Cuál de las series TTL tiene el mayor margen de ruido en estado ALTO?
   (c) ¿Cuál de las series se ha vuelto en esencia obsoleta en los nuevos diseños?
   (d) ¿Cuál de las series utiliza un diodo especial para reducir el tiempo de conmutación?
   (e) ¿Cuál de las series sería la más adecuada para un circuito operado por baterías que funcione a 10 MHz?
2. Suponiendo que cada chip tiene el mismo costo, ¿por qué preferiría usted usar un contador 74ALS193 en vez de un 74LS193 o un 74AS193 en un circuito que opere con un reloj de 40-MHz?
3. Identifique los transistores de puesta en alto y de puesta en bajo para el circuito 74S en la figura 8-12.

8-5 CAPACIDAD DE CARGA DE LA FAMILIA TTL

Es importante comprender qué es lo que determina la capacidad de carga o la capacidad de control de carga de la salida de un CI. La figura 8-13(a) muestra una salida TTL estándar en el estado BAJO, conectada para manejar varias entradas TTL estándar. El transistor Q_4 está encendido y actúa como drenador de corriente para una cantidad de corriente I_{OL} que es la suma de las corrientes I_{IL} de cada entrada. En su estado ENCENDIDO, la resistencia de colector-emisor de Q_4 es muy pequeña pero no es cero, por lo que la corriente I_{OL} producirá una caída de voltaje V_{OL}. Este voltaje no debe exceder el límite V_{OL}(máx) del CI, el cual limita el valor máximo de I_{OL} y en consecuencia el número de cargas que pueden manejarse.

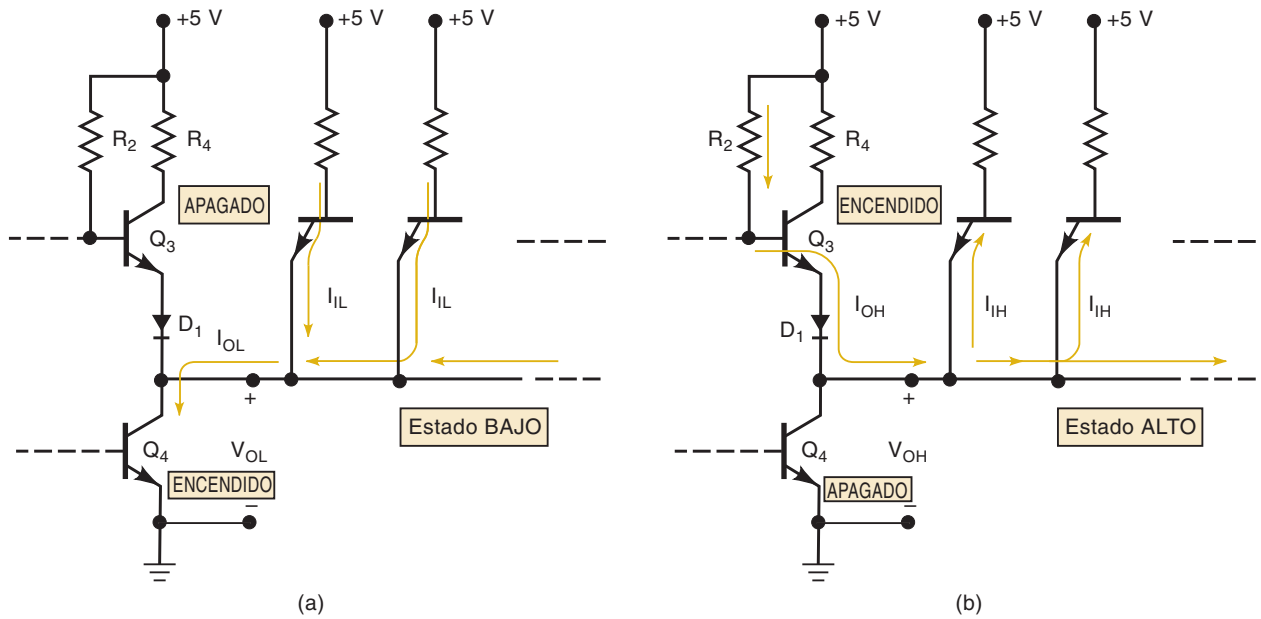


FIGURA 8-13 Las corrientes cuando una salida TTL maneja varias entradas.

Para ilustrarlo, suponga que los CIs son de la serie 74 y que cada I_{IL} es de 1.6 mA. De la tabla 8-6 podemos ver que la serie 74 tiene V_{OL}(máx) = 0.4 V y V_{IL}(máx) = 0.8 V. Supongamos además que Q_4 puede drenar hasta 16 mA antes de que su voltaje de salida llegue a V_{OL}(máx) = 0.4 V. Esto significa que puede drenar la corriente de hasta 16 mA/1.6mA = 10 cargas. Si se conecta a más de 10 cargas, su I_{OL} se incrementará y hará que V_{OL} se incremente por encima de 0.4 V. Por lo general, esto es indeseable, ya que reduce el margen de ruido en las entradas del CI [recuerde que $V_{NL} = V_{IL}$(máx) − V_{OL}(máx)]. De hecho, si V_{OL} se eleva por encima de V_{IL}(máx) = 0.8 V, estará en el intervalo indeterminado.

Una situación similar se presenta en el estado ALTO que se describe en la figura 8-13(b). Aquí Q_3 actúa como un seguidor de emisor que está suministrando una corriente total I_{OH}, que es la suma de las corrientes I_{IH} de las distintas entradas TTL. Si se están manejando demasiadas cargas, esta corriente I_{OH} aumentará lo suficiente como para hacer que las caídas de voltaje a través de R_2, la unión emisor-base de Q_3 y D_1 lleven a V_{OH} por debajo de V_{OH}(mín). Esto también es indeseable, ya que reduce el margen de ruido en estado ALTO y podría incluso ocasionar que V_{OH} caiga en el intervalo indeterminado.

Lo que todo esto significa es que una salida TTL tiene un límite I_{OL}(máx) en relación con la cantidad de corriente que puede drenar en el estado BAJO. También tiene un límite I_{OH}(máx) en relación con la cantidad de corriente que puede suministrar en el estado ALTO. Estos límites de corriente de salida no deben excederse si los niveles de voltaje de salida se van a mantener dentro de sus intervalos especificados.

Determinación de la capacidad de carga

Para determinar cuántas entradas distintas puede controlar la salida de un CI, necesitamos conocer la capacidad de control de corriente de la salida [es decir, I_{OL}(máx) e I_{OH}(máx)] y los requerimientos de corriente de cada entrada (es decir, I_{IL} e I_{IH}). Esta información siempre se presenta de alguna forma en la hoja técnica del fabricante de CIs. Los siguientes ejemplos ilustrarán un tipo de situación.

EJEMPLO 8-5

¿Cuántas entradas de una compuerta NAND 74ALS00 pueden manejarse mediante la salida de una compuerta NAND 74ALS00?

Solución

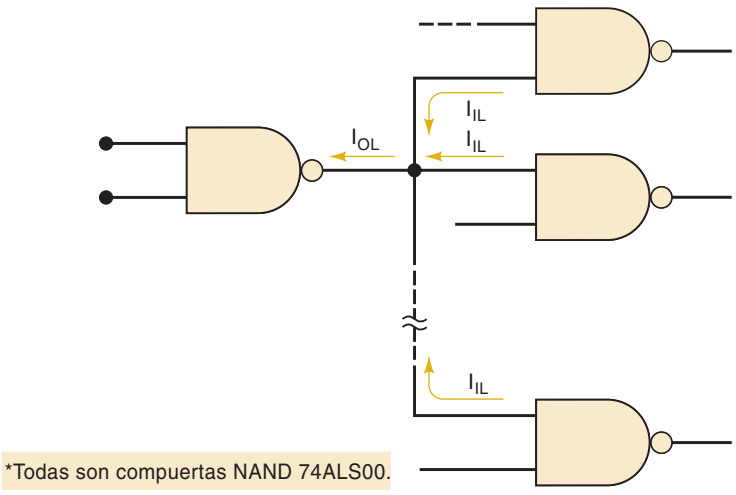
Consideraremos primero el estado BAJO, como se describe en la figura 8-14. Consulte la hoja técnica del 74ALS00 en la figura 8-11 y busque

$$I_{OL}(\text{máx}) = 8 \text{ mA}$$
$$I_{IL}(\text{máx}) = 0.1 \text{ mA}$$

Esto indica que una salida 74ALS00 puede drenar un máximo de 8 mA y que cada entrada del 74ALS00 suministrará un máximo de 0.1 mA de regreso a través de la salida de la compuerta. Por ende, el número de entradas que se pueden manejar en el estado BAJO se obtiene de la siguiente manera:

$$\text{capacidad de carga (BAJO)} = \frac{I_{OL}(\text{máx})}{I_{IL}(\text{máx})} = \frac{8 \text{ mA}}{0.1 \text{ mA}} = 80$$

FIGURA 8-14 Ejemplo 8-5.



(*Nota:* la entrada para I_{IL} es -0.1 mA. El signo negativo se utiliza para indicar que esta corriente fluye *hacia fuera* de la terminal de entrada; en este caso podemos ignorar este signo). El estado ALTO se analiza de la misma forma. Consulte la hoja técnica para buscar valores para I_{OH} e I_{IH}; ignore los signos negativos.

$$I_{OH}(\text{máx}) = 0.4\text{ mA} = 400\ \mu\text{A}$$
$$I_{IH}(\text{máx}) = 20\ \mu\text{A}$$

Por lo tanto, el número de entradas que pueden manejarse en el estado ALTO es

$$\text{capacidad de carga (ALTO)} = \frac{I_{OH}(\text{máx})}{I_{IH}(\text{máx})} = \frac{400\ \mu\text{A}}{20\ \mu\text{A}} = 20$$

Si la capacidad de carga (BAJO) y la capacidad de carga (ALTO) no son iguales, como algunas veces pasa, se elije la menor de las dos. Por ende, a la compuerta NAND 74ALS00 se le pueden conectar hasta 20 compuertas NAND 74ALS00 en su salida.

EJEMPLO 8-6

Consulte la hoja técnica en el CD ROM de TI (o en la tabla 8-7) y determine cuántas compuertas NAND 74AS20 pueden controlarse mediante la salida de otro 74AS20.

Solución

La hoja técnica del 74AS20 nos proporciona los siguientes valores:

$$I_{OH}(\text{máx}) = 2\text{ mA}$$
$$I_{OL}(\text{máx}) = 20\text{ mA}$$
$$I_{IH}(\text{máx}) = 20\ \mu\text{A}$$
$$I_{IL}(\text{máx}) = 0.5\text{ mA}$$

Si consideramos primero el estado ALTO, tenemos que

$$\text{capacidad de carga (ALTO)} = \frac{2\text{ mA}}{20\ \mu\text{A}} = 100$$

Para el estado BAJO tenemos que

$$\text{capacidad de carga (BAJO)} = \frac{20\text{ mA}}{0.5\text{ mA}} = 40$$

En este caso se elije la capacidad de carga total de 40, ya que es el menor de los dos valores. Por ende, un 74AS20 puede manejar otras 40 entradas 74AS20 más.

En equipos antiguos podrá observar que con frecuencia la mayoría de los CIs lógicos se eligieron de la misma familia lógica. En los sistemas digitales actuales es mucho más probable que haya una combinación de varias familias lógicas. En consecuencia, los cálculos de las cargas y las capacidades de carga no son tan simples como una vez lo fueron. Un buen método para determinar la carga de cualquier salida digital es el siguiente:

Paso 1. Sume los valores de I_{IH} de todas las entradas conectadas a una salida. Esta suma debe ser menor que la especificación de I_{OH} de la salida.

Paso 2. Sume los valores de I_{IL} de todas las entradas conectadas a una salida. Esta suma debe ser menor que la especificación de I_{OL} de la salida.

La tabla 8-7 muestra las especificaciones limitantes para las corrientes de entrada y salida en compuertas lógicas simples de las diversas familias TTL. Observe que algunos de los valores de corriente se proporcionan como números negativos. Esta convención se utiliza para mostrar la dirección del flujo de la corriente. Los valores positivos indican que la corriente fluye hacia el nodo especificado, sea una entrada o una salida. Los valores negativos indican que la corriente fluye hacia fuera del nodo especificado. En consecuencia, todos los valores de I_{OH} son negativos ya que la corriente fluye hacia la salida (corriente de suministro) y todos los valores de I_{OL} son positivos ya que la corriente de carga fluye hacia la terminal de salida en su camino a tierra (corriente de drenado). De igual forma, I_{IH} es positiva mientras que I_{IL} es negativa. Cuando calcule la carga y la capacidad de carga como se describió antes, deberá ignorar estos signos.

TABLA 8-7
Clasificaciones de corriente de las compuertas lógicas de la serie TTL.*

	Salidas		Entradas	
Series TTL	I_{OH}	I_{OL}	I_{IH}	I_{IL}
74	−0.4 mA	16 mA	40 μA	−1.6 mA
74S	−1 mA	20 mA	50 μA	−2 mA
74LS	−0.4 mA	8 mA	20 μA	−0.4 mA
74AS	−2 mA	20 mA	20 μA	−0.5 mA
74ALS	−0.4 mA	8 mA	20 μA	−0.1 mA
74F	−1 mA	20 mA	20 μA	−0.6 mA

*Algunos dispositivos pueden tener distintas clasificaciones de corriente de entrada o de salida. Consulte siempre la hoja técnica.

EJEMPLO 8-7

La salida de una compuerta NAND 74ALS00 maneja tres entradas de compuertas 74S y una entrada 7406. Determine si hay un problema de carga.

Solución

1. Sume todos los valores de I_{IH}:

$$\text{Total} = \underset{3 \cdot (I_{IH} \text{ para el 74S})}{3 \cdot (50\ \mu A)} + \underset{1 \cdot (I_{IH} \text{ para el 74})}{1 \cdot (40\ \mu A)} = 190\ \mu A$$

   El valor de I_{OH} para la salida del 74ALS es de 400 μA (máx), que es mayor que la suma de todas las cargas (190 μA). Esto no es problema cuando la salida está en ALTO.

2. Sume todos los valores de I_L:

$$\text{Total} = \underset{3 \cdot (I_L \text{ para 74S})}{3 \cdot (2\ mA)} + \underset{1 \cdot (I_L \text{ para 74})}{1 \cdot (1.6\ mA)} = 7.6\ mA$$

   El valor de I_{OH} para la salida del 74ALS es de 8 mA (máx), lo cual es mayor que la suma de las cargas (7.6 mA). Esto no representa un problema cuando la salida está en BAJO.

EJEMPLO 8-8

La salida de la compuerta NAND 74ALS00 del ejemplo 8-7 necesita utilizarse para manejar algunas entradas 74ALS además de las entradas de carga descritas en el ejemplo 8-7. ¿Cuántas entradas 74ALS adicionales podría manejar la salida sin sobrecargarse?

Solución

De los cálculos del ejemplo 8-7, sólo en el estado BAJO estamos cerca de que se sobrecargue la salida. Una entrada 74ALS tiene una I_{IL} de 0.1 mA. La corriente de drenado máxima (I_{OL}) es de 8 mA y la corriente de carga es de 7.6 mA (según los cálculos del ejemplo 8-7). Para encontrar la corriente adicional que puede drenar la salida hacemos lo siguiente:

$$\begin{aligned}\text{Corriente adicional} &= I_{OL}\text{máx 2 suma de las cargas}(I_{IL}) \\ &= 8\ mA\ 2\ 7.6\ mA = 0.4\ mA\end{aligned}$$

Esta salida puede manejar hasta cuatro entradas 74ALS más, que tengan una I_{IL} de 0.1 mA.

EJEMPLO 8-9

La salida de un inversor 74AS04 proporciona la señal CLEAR a un registro en paralelo, el cual está formado por flip-flops D 74AS74. ¿Cuál es el máximo número de entradas *CLR* de FF que puede maneja r esta compuerta?

Solución

Las especificaciones para las entradas de los flip-flops no son siempre las mismas que para la entrada de una compuerta lógica en la misma familia. Consulte la hoja técnica del 74AS74 en el CD ROM de TI. Las entradas de reloj y *D* son similares a las entradas de las compuertas en la tabla 8-7. No obstante, las entradas *PRE* y *CLR* tienen especificaciones de $I_{IH} = 40\ \mu A$ e $I_{IL} = 1.8$ mA. El 74AS04 tiene especificaciones de $I_{OH} = 2$ mA e $I_{OL} = 20$ mA.

$$\text{Número máximo de entradas (ALTO)} = 2\ mA/40\ \mu A = 50.$$
$$\text{Número máximo de entradas (BAJO)} = 20\ mA/1.8\ mA = 11.11.$$

Debemos limitar la capacidad de carga a 11 entradas *CLR*.

PREGUNTAS DE REPASO

1. ¿Qué factores determinan la clasificación de I_{OL}(máx) de un dispositivo?
2. ¿Cuántas entradas 7407 puede manejar un chip 74AS?
3. ¿Qué puede ocurrir si una salida TTL se conecta a más entradas de compuerta de las que está clasificada para manejar?
4. ¿Cuántas entradas *CP* 74112 pueden manejarse mediante una salida 74LS04? ¿Mediante una salida 74F00?

8-6 OTRAS CARACTERÍSTICAS DE LA FAMILIA TTL

Debemos conocer varias características más de la lógica TTL si queremos utilizar la familia TTL con inteligencia en una aplicación de un sistema digital.

Entradas desconectadas (flotantes)

Cualquier entrada de un circuito TTL que se deje desconectada (abierta) actúa en forma idéntica a un 1 lógico que se aplique a esa entrada, ya que en cualquier caso la unión base-emisor o diodo en la entrada no se polarizará en forma directa. Esto significa que en *cualquier* CI TTL, *todas* las entradas son 1s si no se conectan con alguna señal lógica o a tierra. Cuando una entrada se deja desconectada, se dice que está **flotando**.

Entradas sin utilizar

Con frecuencia, no todas las entradas en un CI TTL se utilizan en una aplicación específica. Un ejemplo común es cuando no todas las entradas de una compuerta lógica se necesitan para la función lógica requerida. Por ejemplo, suponga que necesitamos la operación lógica $\overline{AB}$ y que estamos usando un chip con una compuerta NAND de tres entradas. La figura 8-15 muestra las posibles formas de lograr esto.

En la figura 8-15(a) la entrada sin utilizar se deja desconectada, lo cual significa que actúa como un 1 lógico. Por lo tanto, la salida de la compuerta NAND es $x = \overline{A \cdot B \cdot 1} = \overline{A \cdot B}$, que es el resultado deseado. Aunque la lógica es correcta, no es muy conveniente dejar una entrada desconectada ya que actuará como una antena, con la probabilidad de que recoja señales radiadas dispersas, las cuales podrían provocar que la compuerta opere en forma inapropiada. La figura 8-15(b) muestra una mejor técnica. Aquí la entrada sin utilizar se conecta a +5 V por medio de una resistencia de 1-kΩ, por lo que el nivel lógico es un 1. La resistencia de 1-kΩ se utiliza sólo por protección contra la corriente de las uniones emisor-base de las entradas de la compuerta, en el caso de que haya picos en la línea de voltaje de alimentación. Esta misma técnica puede utilizarse para las compuertas AND, ya que un 1 en una entrada sin usar no afectará a la salida. Puede haber hasta 30 entradas sin utilizar que compartan la misma resistencia de 1-kΩ conectada a V_{CC}.

La figura 8-15(c) muestra una tercera posibilidad, en la que la entrada sin usar se conecta a una entrada que sí se utiliza. Esto es satisfactorio siempre y cuando el circuito que maneja la entrada *B* no se vaya a exceder de su capacidad de carga.

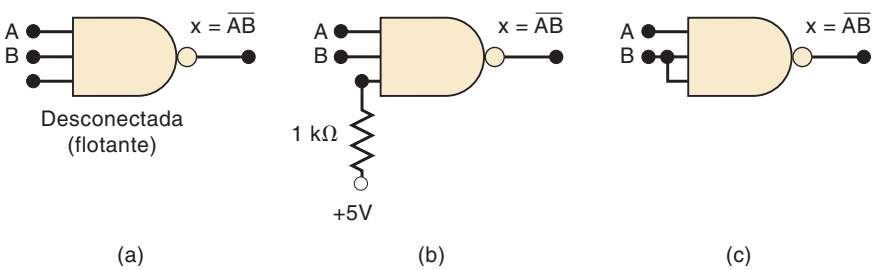


FIGURA 8-15 Tres formas de manejar las entradas lógicas sin utilizar.

Esta técnica puede utilizarse para *cualquier* tipo de compuerta. En las compuertas OR y NOR, las entradas sin usar no pueden dejarse desconectadas o conectadas a +5 V, ya que esto produciría un nivel lógico de salida constante (1 para OR, 0 para NOR) sin importar las demás entradas. Para estas compuertas las entradas sin utilizar deben conectarse a tierra (0 V) para un 0 lógico, o conectarse a una entrada que sí se utilice, como en la figura 8-15(c).

Entradas conectadas entre sí

Cuando dos (o más) entradas TTL en la misma compuerta se conectan entre sí para formar una entrada común, como en la figura 8-15(c), la entrada común, por lo general, representará una carga que es la suma de la clasificación de corriente de carga de cada entrada individual. La única excepción es para las compuertas NAND y AND. En estas compuertas, la carga de entrada en estado BAJO *será la misma que una entrada individual* sin importar cuántas entradas se conecten entre sí.

Para ilustrar esto, suponga que cada entrada de la compuerta NAND de tres entradas en la figura 8-15(c) tiene una clasificación de 0.5 mA para I_{IL} y de 20 μA para I_{IH}. Por lo tanto, la entrada común *B* representará una carga de entrada de 40 μA en el estado ALTO, pero sólo 0.5 mA en el estado BAJO. Lo mismo se aplicaría si fuera una compuerta AND. Si fuera una compuerta OR o NOR, la entrada común *B* presentaría una carga de entrada de 40 μA en el estado ALTO y de 1 mA en el estado BAJO.

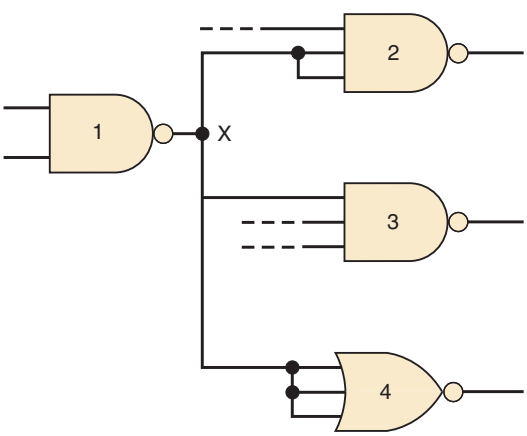
Podemos encontrar la razón de esta característica si analizamos de nuevo el diagrama del circuito de la compuerta NAND TTL en la figura 8-8(b). La corriente I_{IL} está limitada por la resistencia R_1. Aún si las entradas *A* y *B* estuvieran conectadas entre sí y aterrizadas, esta corriente no cambiaría; tan sólo se dividiría y fluiría a través de las rutas paralelas proporcionadas por los diodos D_2 y D_3. La situación es distinta para las compuertas OR y NOR, ya que no utilizan transistores con múltiples emisores, sino que tienen un transistor de entrada separado para cada entrada, como vimos en la figura 8-10.

EJEMPLO 8-10

Determine la carga que maneja la salida *X* en la figura 8-16. Suponga que cada compuerta es un dispositivo de la serie 74LS con I_{IH} = 20 μA e I_{IL} = 0.4 mA.

Solución

La carga en la salida de la compuerta 1 es equivalente a seis cargas de entrada 74LS en el estado ALTO, pero sólo cinco cargas de entrada 74LS en el estado BAJO, ya que la compuerta NAND representa sólo una carga de entrada individual en el estado BAJO.



Carga en la salida de la compuerta 1			
ALTO		BAJO	
Corriente de carga	Compuerta	Corriente de carga	Compuerta
40 μA	2	0.4 mA	2
20 μA	3	0.4 mA	3
60 μA	4	1.2 mA	4
120 μA	Total	2.0 mA	Total

FIGURA 8-16 Ejemplo 8-10.

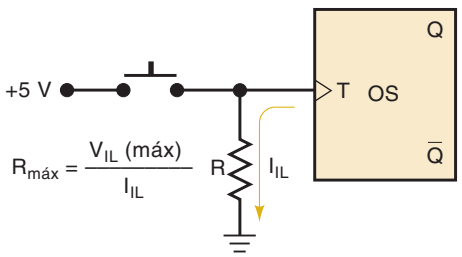
Polarización de las entradas TTL en bajo

En ocasiones se produce la situación en la que una entrada TTL debe mantenerse, por lo general, en BAJO y después se hace que cambie a ALTO por medio de la actuación de un interruptor mecánico. Esta situación se ilustra en la figura 8-17 para la entrada a un monoestable. Este monoestable se dispara en una transición positiva que ocurre cuando el interruptor se cierra por un instante. La resistencia R sirve para mantener la entrada T en BAJO mientras el interruptor está abierto. Hay que tener cuidado de mantener el valor de R lo bastante bajo como para que el voltaje que se desarrolle a través de esta resistencia debido a la corriente I_{IL} que fluye hacia fuera de la entrada del monoestable hacia tierra no se exceda de V_{IL}(máx). Así, el valor más grande de R se obtiene mediante

$$I_{IL} \times R_{máx} = V_{IL}(máx)$$
$$R_{máx} = \frac{V_{IL}(máx)}{I_{IL}} \tag{8-3}$$

R debe mantenerse por debajo de este valor para asegurar que la entrada del monoestable se encuentre en un nivel BAJO aceptable mientras el interruptor está abierto. El valor mínimo de R se determina en base al drenado de corriente en la alimentación de 5 V cuando el interruptor se cierra. En la práctica, este drenado de corriente debe minimizarse; para ello hay que mantener a R justo por debajo de $R_{máx}$.

FIGURA 8-17



EJEMPLO 8-11

Determine un valor aceptable para R si el monoestable es un CI TTL 74LS con una clasificación de I_{IL} de entrada de 0.4 mA.

Solución

El valor de I_{IL} será un máximo de 0.4 mA. Este valor máximo debe usarse para calcular $R_{máx}$. De la tabla 8-6, V_{IL}(máx) = 0.8 V para la serie 74LS. Por lo tanto, tenemos que

$$R_{máx} = \frac{0.8\text{ V}}{0.4\text{ mA}} = 2000\ \Omega$$

Aquí una buena elección sería R = 1.8 kΩ, un valor de resistencia estándar.

Transitorios de corriente

Los circuitos lógicos TTL sufren de transitorios o picos de corriente generados en forma interna, debido a la estructura de salida en forma de tótem. Cuando la salida cambia del estado BAJO al estado ALTO (vea la figura 8-18), los dos transistores de salida cambian de estado: Q_3 de APAGADO a ENCENDIDO y Q_4 de ENCENDIDO a APAGADO. Como Q_4 está cambiando de la condición saturada, tardará más tiempo

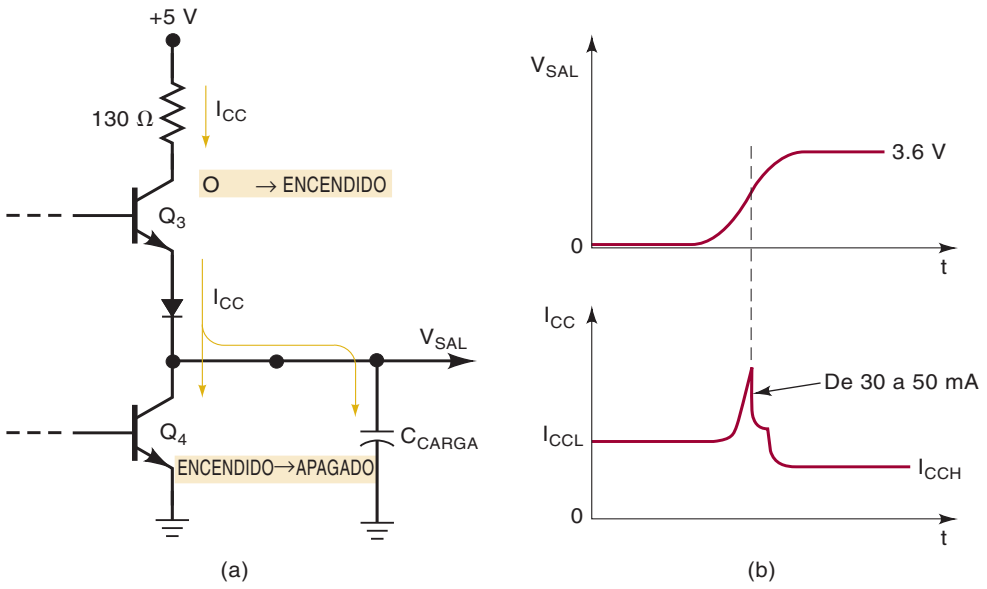


FIGURA 8-18 Un pico grande de corriente se drena de V_{CC} cuando una salida en forma de tótem cambia de BAJO a ALTO.

que Q_3 para cambiar de estado. Por lo tanto, hay un intervalo corto de tiempo (cerca de 2 ms) durante la transición de conmutación cuando ambos transistores están conduciendo y se drena una sobretensión bastante grande de corriente (de 30 a 50 mA) de la alimentación de +5 V. La duración de este transitorio de corriente se extiende por los efectos de cualquier capacitancia de carga en la salida del circuito. Ésta consiste de la capacitancia dispersa en el conductor y de la capacitancia de entrada de cualquier circuito de carga, y debe cargarse hasta llegar al voltaje de salida en estado ALTO. Este efecto en general puede sintetizarse de la siguiente manera:

Cada vez que una salida TTL en forma de tótem cambia de BAJO a ALTO, se drena un pico de corriente de alta amplitud de V_{CC}.

En un circuito o sistema digital complejo, puede haber muchas salidas TTL que cambien de estado al mismo tiempo; cada una de ellas drenará un pico estrecho de corriente de la fuente de energía. El efecto acumulativo de todos estos picos de corriente será producir un pico de voltaje en la línea común de V_{CC}, en gran parte debido a la inductancia distribuida en la línea de alimentación [recuerde que $V = L(di/dt)$ para inductancia, y que di/dt es muy grande para un pico de corriente de 2 ns]. Este pico de voltaje puede producir varias fallas graves durante las transiciones de conmutación, a menos que se utilice algún tipo de filtrado. La técnica más común utiliza pequeños capacitores de radiofrecuencia conectados de V_{CC} a TIERRA, en esencia para "poner en corto" estos picos de alta frecuencia. A esto se le conoce como **desacoplamiento de fuente de poder**.

Es práctica común conectar un capacitor de disco de cerámica con baja inductancia de 0.01-μF o 0.1-μF entre V_{CC} y tierra, cerca de cada CI TTL en un circuito impreso. Las terminales del capacitor se deben mantener muy cortas para minimizar la inductancia en serie.

También es común conectar un capacitor grande individual (de 2 a 20 μF) entre V_{CC} y tierra en cada tablero para filtrar las variaciones de una frecuencia relativamente baja en V_{CC}, las cuales se producen debido a los grandes cambios en los niveles de I_{CC} a medida que las salidas cambian de estado.

PREGUNTAS DE REPASO

1. ¿Cuál será la salida lógica de una compuerta NAND TTL que tiene todas sus entradas desconectadas?
2. ¿Cuáles son dos formas aceptables de manejar las entradas sin usar de una compuerta AND?
3. Repita la pregunta 2 para una compuerta NOR.
4. *Verdadero o falso:* cuando las entradas de una compuerta NAND se conectan entre sí, siempre se tratan como una carga individual en la fuente de la señal.
5. ¿Qué es el desacoplamiento de fuentes de energía? ¿Por qué se utiliza?

8-7 TECNOLOGÍA MOS

La tecnología MOS (metal óxido semiconductor) deriva su nombre de la estructura MOS básica de un electrodo metálico sobre un aislante de óxido, sobre un sustrato semiconductor. Los transistores de la tecnología MOS son transistores de efecto de campo, a los cuales se les conoce como **MOSFETs**. Esto significa que el *campo* eléctrico en el lado del electrodo *metálico* del aislante de *óxido* tiene un *efecto* sobre la resistencia del sustrato. La mayoría de los CIs digitales MOS están construidos en su totalidad a partir de MOSFETs y de ningún otro componente.

Las principales ventajas del MOSFET son que su fabricación es relativamente simple y económica; es pequeño y consume muy poca energía. La fabricación de los CIs MOS es equivalente a casi una tercera parte de la fabricación de los CIs bipolares (TTL, ECL, etc.). Además, los dispositivos MOS ocupan mucho menos espacio en un chip que los transistores bipolares; lo que es más importante: los CIs digitales MOS, por lo general, no utilizan los elementos tipo resistencia de los CIs, que ocupan la mayor parte del área del chip de los CIs bipolares.

Todo esto significa que los CIs MOS pueden alojar un número mucho mayor de elementos de circuito en un solo chip, en comparación con los CIs bipolares. Esta ventaja se ilustra mediante el hecho de que los CIs MOS han dominado a los CIs bipolares en el área de la integración a gran escala (LSI, VLSI). La alta densidad de encapsulamiento de los CIs MOS los hace ser en especial muy apropiados para los CIs complejos, tales como los microprocesadores y los chips de memoria. Las mejoras en la tecnología de CIs MOS nos han llevado a idear dispositivos que son más veloces que los dispositivos TTL 74, 74LS y 74ALS, con características de control de corriente comparables. En consecuencia, los dispositivos MOS (en específico, CMOS) también se han convertido en parte dominante del mercado SSI y MSI. La familia TTL 74AS aún sigue siendo tan rápida como los mejores dispositivos CMOS, pero a costa de una disipación de potencia mucho mayor.

La principal desventaja de los dispositivos MOS es su susceptibilidad al daño por la electricidad estática. Aunque esto puede minimizarse mediante los procedimientos de manejo apropiados, la familia TTL es mucho más resistente para la experimentación en el laboratorio. Por consiguiente, es muy probable que usted vea dispositivos TTL en usos educativos, siempre y cuando estén disponibles.

El MOSFET

Hasta ahora hay dos tipos generales de MOSFETs: de *depleción* y de *enriquecimiento*. Los CIs digitales MOS utilizan sólo MOSFETs de enriquecimiento, por lo que sólo consideraremos este tipo de MOSFETs en el siguiente análisis. Lo que es más, sólo nos preocuparemos por la operación de estos MOSFETs en los interruptores de encendido/apagado.

La figura 8-19 muestra los símbolos esquemáticos para los MOSFETs de enriquecimiento de canal N y canal P, en donde la dirección de la flecha indica ya sea el canal P o el canal N. Los símbolos muestran una línea quebrada entre la *fuente*

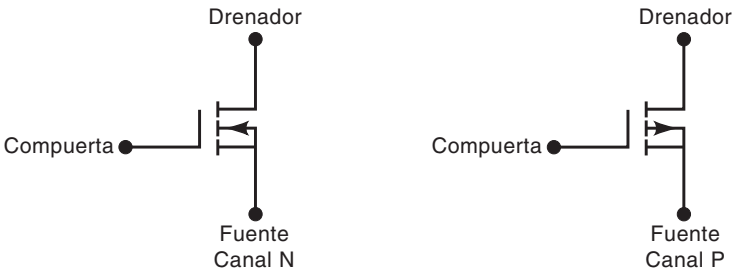


FIGURA 8-19 Símbolos esquemáticos para los MOSFETs de enriquecimiento.

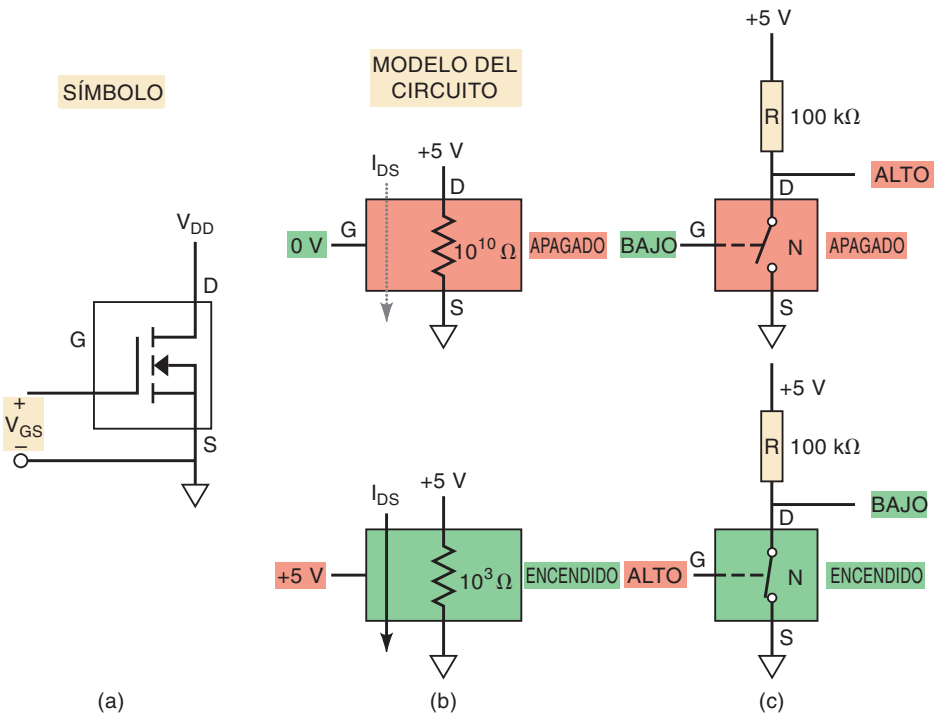
y el *drenador* para indicar que *por lo general* no hay un canal conductor entre estos dos electrodos. El símbolo también muestra una separación entre la *compuerta* y las demás terminales para indicar la resistencia muy alta (por lo general, alrededor de $10^{12}\Omega$) de la capa de óxido entre la compuerta y el canal, la cual se forma en el sustrato.

Interruptor MOSFET básico

La figura 8-20 muestra la operación de conmutación de un MOSFET de canal N, el elemento básico en una familia de dispositivos conocida como **N-MOS**. Para el dispositivo de canal N, el drenador siempre se polariza en forma directa con respecto a la fuente. El voltaje de compuerta a fuente V_{GS} es el de entrada, el cual se utiliza para controlar la resistencia entre drenador y fuente (es decir, la resistencia del canal), y por lo tanto determina si el dispositivo está encendido o apagado.

Cuando $V_{GS} = 0$, no hay un canal conductor entre fuente y drenador, y el dispositivo está apagado como se muestra en la figura 8-20(b). Por lo general, la resistencia del canal en este estado APAGADO es de $10^{10}\Omega$, que para la mayoría de los fines es un *circuito abierto*. El MOSFET permanecerá apagado siempre y cuando V_{GS} sea cero o negativo. A medida que V_{GS} se hace positivo (compuerta positiva, relativa a la fuente) se llega a un voltaje de umbral (V_T), punto en el cual se comienza a formar

FIGURA 8-20
MOSFET de canal N que se utiliza como un interruptor: (a) símbolo; (b) modelo del circuito; (c) operación de un inversor N-MOS.



un canal conductor entre la fuente y el drenador. Por lo general, $V_T = +1.5$ V para un N-MOSFET y, por lo tanto, cualquier $V_{GS} \geq 1.5$ V hará que el MOSFET conduzca. Por lo general, se utiliza un valor de V_{GS} mucho mayor que V_T para encender el MOSFET en forma más completa. Como se muestra en la figura 8-20(b), cuando $V_{GS} = +5$ V la resistencia del canal entre fuente y drenador cae hasta llegar a un valor de $R_{ON} = 1000\ \Omega$.

En esencia, el N-MOS cambiará de una resistencia muy alta a una baja a medida que el voltaje de la compuerta cambie de BAJO a ALTO. Es útil tan sólo pensar en el MOSFET como un interruptor que está abierto o cerrado entre la fuente y el drenador. La figura 8-20(c) muestra cómo puede formarse un inversor mediante el uso de un transistor N-MOS como un interruptor. Los primeros dispositivos lógicos N-MOS se construyeron con base en este método. La desventaja de este circuito (al igual que en TTL) es que cuando el transistor está ENCENDIDO, siempre habrá corriente fluyendo de la fuente hacia tierra, con lo que se producirá calor.

El MOSFET de canal P (o P-MOS) que se muestra en la figura 8-21(a) opera en forma idéntica al de canal N, sólo que utiliza voltajes de polaridad opuesta. Para los MOSFETs el drenador está conectado al lado inferior del circuito, por lo que se polariza con un voltaje más negativo en forma relativa a la fuente. Para ENCENDER el P-MOSFET, debe aplicarse a la compuerta un voltaje *inferior* al de la fuente por V_T, lo cual significa que el voltaje en la compuerta, con respecto a la fuente, debe ser negativo.

La figura 8-21(b) muestra que cuando la compuerta está a 5 V con respecto a tierra (el mismo voltaje que se aplica a la fuente), el transistor está APAGADO y tiene una resistencia muy alta de drenador a fuente. Cuando la compuerta está a 0 V (con respecto a tierra), entonces el voltaje de compuerta a fuente $V_{GS} = -5$ V y ENCIENDE el transistor, con lo cual se reduce su resistencia de drenador a fuente. El circuito de la figura 8-20(c) muestra la acción de conmutación de un inversor que utiliza lógica P-MOS.

La tabla 8-8 sintetiza las características de conmutación de los MOSFETs con canal P y N.



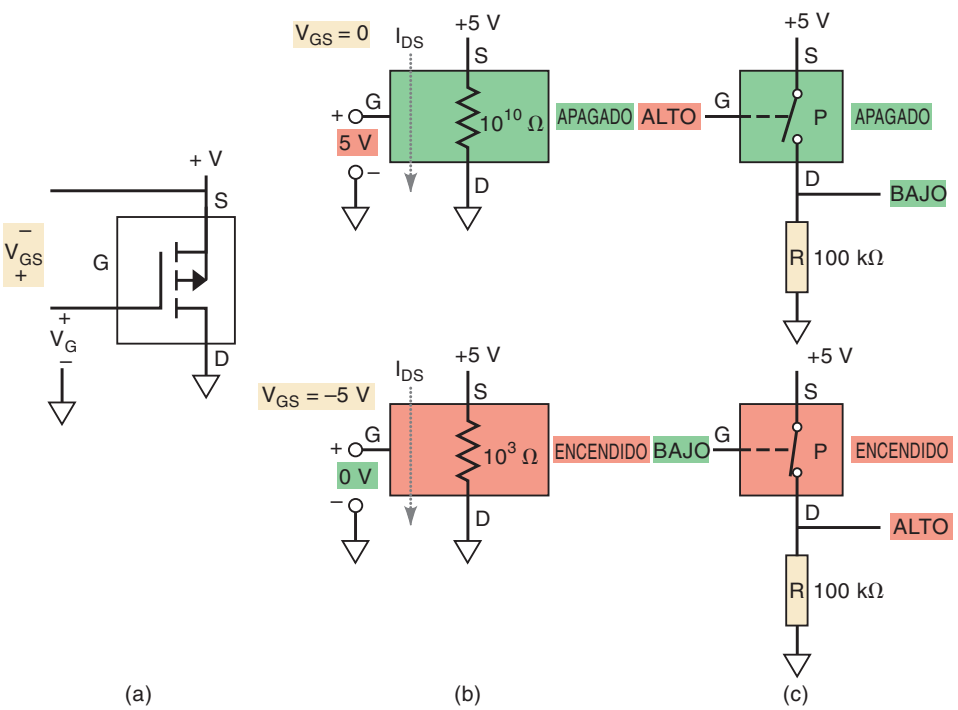


FIGURA 8-21
MOSFET de canal P utilizado como un interruptor: (a) símbolo; (b) modelo del circuito para ENCENDIDO y APAGADO; (c) circuito inversor P-MOS.

TABLA 8-8

	Polarización de drenador a fuente	Voltaje de compuerta a fuente (V_{GS}) necesario para la conducción	R_{ENC} (Ω)	R_{APAG} (Ω)
Canal P	Negativa	Por lo general, más negativo que −1.5 V	1000 (típico)	10^{10}
Canal N	Positiva	Por lo general, más positivo que +1.5 V	1000 (típico)	10^{10}

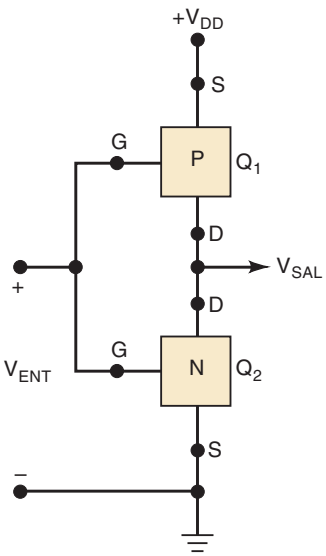
8-8 LÓGICA DE MOS COMPLEMENTARIO

Los circuitos lógicos P-MOS y N-MOS utilizan menos componentes y son mucho más simples de fabricar que los circuitos TTL. Como resultado, comenzaron a dominar los mercados LSI y VLSI en las décadas de 1970 y 1980. Durante esta era surgió una nueva tecnología que utilizaba tanto transistores P-MOS (como interruptores del lado alto) como N-MOS (como interruptores del lado bajo) en el mismo circuito lógico. A esta tecnología se le conoce como MOS complementario, o **CMOS**. Los circuitos lógicos CMOS no son tan simples y fáciles de fabricar como un P-MOS o un N-MOS, pero son más rápidos, utilizan mucho menos energía y son la tecnología dominante en el mercado actual.

Inversor CMOS

La figura 8-22 muestra el circuito para el INVERSOR CMOS básico. Para este diagrama y los siguientes, hemos sustituido los símbolos estándar para el MOSFET por bloques etiquetados como P y N para indicar un P-MOS y un N-MOS, respectivamente. Esto se hace sólo por conveniencia al analizar los circuitos. El INVERSOR CMOS tiene dos MOSFETs en serie, de manera que el dispositivo de canal P tiene conectada su fuente a $+V_{DD}$ (un voltaje positivo) y el dispositivo de canal N tiene su fuente conectada a tierra.* Las compuertas de los dos dispositivos se conectan entre sí para formar una entrada común. Los drenadores de los dos dispositivos se conectan entre sí para formar la salida común.

FIGURA 8-22
INVERSOR CMOS básico.



V_{ENT}	Q_1	Q_2	V_{SAL}
$+V_{DD}$ (1 lógico)	APAGADO $R_{APAG} = 10^{10}\ \Omega$	ENCENDIDO $R_{ENC} = 1\ k\Omega$	≃ 0 V
0 V (0 lógico)	ENCENDIDO $R_{ENC} = 1\ k\Omega$	APAGADO $R_{APAG} = 10^{10}\ \Omega$	≃ $+V_{DD}$

$$V_{SAL} = \overline{V_{ENT}}$$

* La mayoría de los fabricantes etiqueta esta terminal como V_{SS}.

Los niveles lógicos para CMOS son $+V_{DD}$ para el 1 lógico y 0 V para el 0 lógico. Considere primero el caso en el que $V_{ENT} = +V_{DD}$. En esta situación, la compuerta de Q_1 (canal P) está a 0 V con respecto a la fuente de Q_1. Por ende, Q_1 estará en el estado APAGADO con $R_{APAG} \approx 10^{10}\ \Omega$. La compuerta de Q_2 (canal N) estará a $+V_{DD}$ con respecto a su fuente. Por ende, Q_2 estará encendido con un valor típico de $R_{ENC} = 1\ k\Omega$. El divisor de voltaje entre R_{APAG} de Q_1 y R_{ENC} de Q_2 producirá $V_{SAL} \approx 0$ V.

Ahora considere el caso en el que $V_{ENT} = 0$ V. Q_1 ahora tiene su compuerta en un potencial negativo con respecto a su fuente, mientras que Q_2 tiene $V_{GS} = 0$ V. Por lo tanto, Q_1 estará encendido con $R_{ENC} = 1\ k\Omega$ y Q_2 estará apagado con $R_{APAG} = 10^{10}\Omega$, y producirá en V_{SAL} un valor aproximado a $+ V_{DD}$. En la tabla de la figura 8-22 se sintetizan estos dos estados de operación, en donde se muestra que el circuito actúa como un INVERSOR lógico.

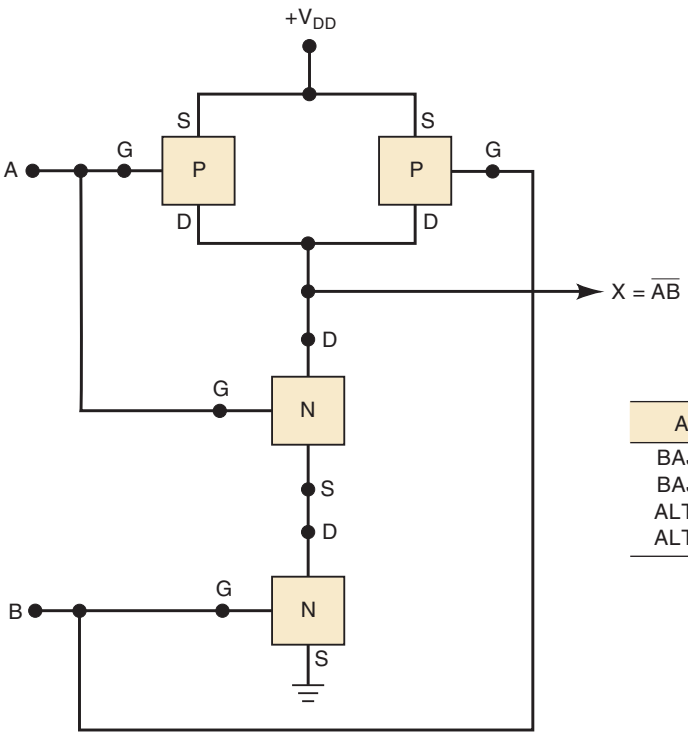
Compuerta NAND CMOS

Pueden construirse otras funciones lógicas si se modifica el INVERSOR básico. La figura 8-23 muestra una compuerta NAND que se forma al agregar un MOSFET de canal P en paralelo y un MOSFET de canal N en serie al INVERSOR básico. Para analizar este circuito, es útil tener en cuenta que una entrada de 0 V enciende su correspondiente P-MOS y apaga su correspondiente N-MOS, y viceversa, para una entrada de $+V_{DD}$. Así, podemos ver que la única ocasión en que se producirá una salida en BAJO será cuando las entradas A y B estén ambas en ALTO ($+V_{DD}$) para encender ambos N-MOSFETs, con lo cual se proporciona una resistencia baja de la terminal de salida a tierra. Para todas las demás condiciones de entrada, por lo menos, habrá un P-MOS encendido mientras que al menos un N-MOS estará apagado. Esto produce una salida en ALTO.

Compuerta NOR CMOS

Para formar una compuerta NOR CMOS se agrega un P-MOS en serie y un N-MOS en paralelo al INVERSOR básico, como se muestra en la figura 8-24. Una vez más, este circuito puede analizarse si se toma en cuenta que un nivel BAJO en cualquiera de las entradas encenderá su P-MOS correspondiente y apagará su N-MOS corres-

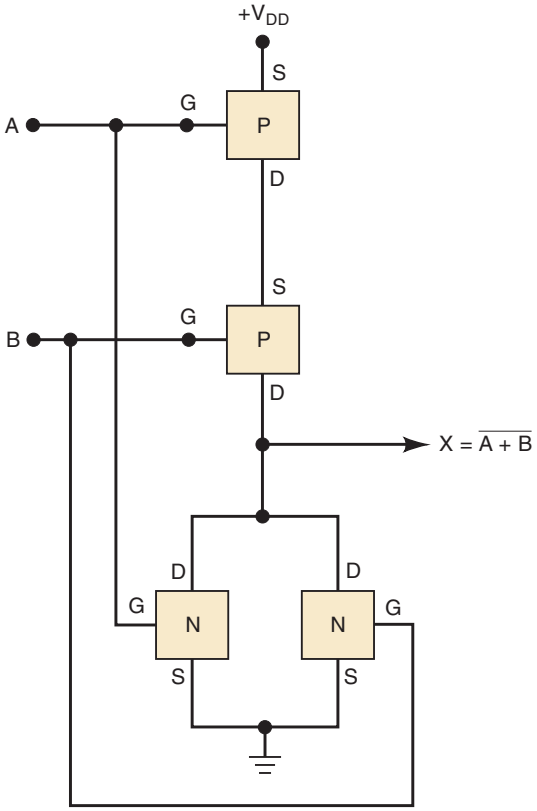
FIGURA 8-23
Compuerta NAND CMOS.



A	B	X
BAJO	BAJO	ALTO
BAJO	ALTO	ALTO
ALTO	BAJO	ALTO
ALTO	ALTO	BAJO



FIGURA 8-24
Compuerta NOR CMOS.



A	B	X
BAJO	BAJO	ALTO
BAJO	ALTO	BAJO
ALTO	BAJO	BAJO
ALTO	ALTO	BAJO



pondiente, y viceversa, para una entrada en ALTO. Dejaremos al lector la opción de verificar que este circuito opere como una compuerta NOR.

Para formar compuertas AND y OR CMOS se combinan las compuertas NAND y NOR con INVERSORes.

FF SET-RESET CMOS

Dos compuertas NOR o NAND CMOS pueden acoplarse en cruz para formar un latch SET-RESET simple. Se utilizan circuitos de compuertas adicionales para convertir el latch SET-RESET básico en flip-flops sincronizados por reloj en D y en J-K.

PREGUNTAS DE REPASO

1. ¿Qué diferencia hay entre los circuitos internos CMOS y los N-MOS?
2. ¿Cuántos MOSFETs de canal P hay en un INVERSOR CMOS?
3. ¿Cuántos MOSFETs hay en una compuerta AND CMOS de tres entradas?

8-9 CARACTERÍSTICAS DE LAS SERIES CMOS

Los CIs CMOS no sólo proporcionan todas las mismas funciones lógicas están disponibles en TTL, sino también varias funciones de propósito especial que no proporciona TTL. Con el paso de los años se han desarrollado varias series CMOS distintas, ya que los fabricantes han buscado mejorar las características de rendimiento. Antes de analizar las diversas series CMOS, será útil definir unos cuantos términos que se utilizan

cuando se van a utilizar CIs de distintas familias o series en conjunto, o como reemplazo por otro.

- **Compatibilidad de terminales.** Dos CIs tienen terminales compatibles cuando sus configuraciones de terminales son iguales. Por ejemplo, la terminal 7 en ambos CIs es TIERRA, la terminal 1 en ambos es una entrada para el primer INVERSOR, y así en lo sucesivo.
- **Equivalente funcional.** Dos CIs son equivalentes funcionales si las funciones lógicas que realizan son idénticas. Por ejemplo, ambos contienen cuatro compuertas NAND de dos entradas, o ambos contienen seis flip-flops D con disparo en el flanco positivo del reloj.
- **Compatible eléctricamente.** Dos CIs son compatibles eléctricamente cuando pueden conectarse en forma directa uno con el otro, sin necesidad de tomar medidas especiales para asegurar su operación correcta.

Series 4000/14000

La serie CMOS más antigua es la serie 4000, introducida por primera vez por RCA, y su equivalente funcional, la serie 14000 de Motorola. Los dispositivos en las series 4000/14000 tienen una disipación de potencia muy baja y pueden operar sobre un amplio intervalo de voltajes de alimentación de energía (de 3 a 15 V). Son muy lentos si se les compara con dispositivos TTL y las demás series CMOS, y tienen capacidades de corriente de salida muy bajas. No son compatibles en las terminales ni compatibles eléctricamente con las series TTL. Los dispositivos de las series 4000/14000 raras veces se utilizan en los nuevos diseños, excepto cuando hay un CI de propósito especial disponible, que no se encuentre disponible en otras series.

74HC/HCT (CMOS de alta velocidad)

La serie 74HC tiene un incremento de 10 veces la velocidad de conmutación, en comparación con la de los dispositivos 74LS, y una capacidad de corriente de salida mucho mayor que los primeros CIs de la serie 7400 CMOS. Los CIs 74HC/HCT son compatibles en terminales y equivalentes funcionales de los CIs TTL con el mismo número de dispositivo. Los dispositivos 74HCT son compatibles eléctricamente con TTL, pero los dispositivos 74HC no. Por ejemplo, esto significa que un chip INVERSOR hexadecimal 74HCT04 puede sustituir un chip 74LS04, y viceversa. También significa que un CI 74HCT puede conectarse directamente a cualquier CI TTL.

74AC/ACT (CMOS avanzado)

A esta serie se le conoce por lo común como ACL, por lógica CMOS avanzada. La serie es equivalente funcional de las diversas series TTL, pero *no* es compatible en terminales con TTL ya que las posiciones de las terminales en los chips 74AC o 74ACT se eligieron de manera que se mejorara la inmunidad al ruido, por lo que las entradas de los dispositivos son menos sensibles a los cambios de señal que ocurren en las terminales de otros CIs. Los dispositivos 74AC no son compatibles eléctricamente con TTL; los dispositivos 74ACT pueden conectarse directamente a los dispositivos TTL. ACL ofrece ciertas ventajas sobre la serie HC en cuanto a inmunidad al ruido, tiempos de propagación y velocidad máxima del reloj.

La numeración de los dispositivos para esta serie difiere un poco de la correspondiente para las series TTL, 74C y 74HC/HCT, ya que utiliza un número de dispositivo de cinco dígitos, el cual comienza con los dígitos 11. Los siguientes ejemplos ilustran esto:

74AC11 004 ≡ 74HC 04
74ACT11 293 ≡ 74HCT 293

74AHC/AHCT (CMOS avanzado de alta velocidad)

Esta serie de dispositivos CMOS ofrece una ruta de migración natural de la serie HC a las aplicaciones más rápidas, de menor potencia y de control bajo. Los dispositivos en esta serie son tres veces más rápidos y pueden utilizarse como reemplazos directos para los dispositivos de la serie HC. Ofrecen una inmunidad al ruido similar a la serie HC, sin los problemas de sobredisparo/subdisparo que a menudo se asocian con las características de control más altas requeridas para una velocidad comparable.

Lógica BiCMOS de 5V

Varios fabricantes de CIs han desarrollado series lógicas que combinan las mejores características de las lógicas bipolar y CMOS; a esto se le conoce como lógica BiCMOS. Las características de baja potencia de CMOS y las características de alta velocidad de los circuitos bipolares se integran para producir una familia lógica con una potencia extremadamente baja y una velocidad en extremo alta. Los CIs BiCMOS no están disponibles en la mayoría de las funciones SSI y MSI, sino que se limitan a funciones que se utilizan en aplicaciones de microprocesadores y de interfase de bus tales como latches, búferes, reforzadores y transceptores. La serie 74BCT (tecnología de interfase de bus BiCMOS) ofrece una reducción del 75 por ciento en consumo de energía en comparación con la familia 74F, mientras mantiene una velocidad y características de control similares. Las piezas en esta serie son compatibles en terminales con las piezas TTL estándar de la industria y operan con niveles lógicos de 5 V estándar. La serie 74ABT (tecnología BiCMOS avanzada) es la segunda generación de dispositivos de interfase de bus BiCMOS. En la sección 8-13 presentaremos los detalles sobre la lógica de la interfase de bus.

Voltaje de alimentación

Los dispositivos de las series 4000/14000 y 74C operan con valores de V_{DD} que varían entre 3 y 15 V, lo cual los hace muy versátiles. Pueden utilizarse en circuitos de bajo voltaje operados por baterías, en circuitos de 5 V estándar y en circuitos en los que se utiliza un voltaje de alimentación más alto para obtener los márgenes de ruido requeridos para su operación en un ambiente con mucho ruido. Las series 74HC/HCT, 74AC/ACT y 74AHC/AHCT operan a través de un intervalo mucho más estrecho de voltajes de alimentación, por lo general entre 2 y 6 V.

También hay disponibles series lógicas que están diseñadas para operar a voltajes menores (por ejemplo, 2.5 o 3.3 V). Cada vez que se interconectan dispositivos que utilizan distintos voltajes de alimentación en el mismo sistema digital, hay que tomar medidas especiales. En la sección 8-10 veremos los dispositivos de bajo voltaje y las técnicas de integración especiales.

Niveles lógicos de voltaje

Los niveles de voltaje de entrada y salida son distintos para las diversas series CMOS. La tabla 8-9 lista estos valores de voltaje para las diversas series CMOS, así como para las series TTL. Los valores listados en la tabla suponen que todos los dispositivos están operando a partir de un voltaje de alimentación de 5 V y que todas las salidas de los dispositivos manejan entradas de la misma familia lógica.

Un análisis de esta tabla nos muestra ciertos puntos importantes. En primer lugar, observe que V_{OL} para los dispositivos CMOS está muy cerca de 0 V, y que V_{OH} está muy cerca de 5 V. La razón se debe a que las salidas CMOS no tienen que suministrar o drenar ninguna cantidad considerable de corriente cuando manejan entradas CMOS con su resistencia de entrada extremadamente alta (10^{12} Ω). Observe además que, con excepción de las series 74HCT y 74ACT, los niveles de voltaje de entrada requeridos son mayores para CMOS que para TTL. Recuerde que las series

TABLA 8-9 Niveles de voltaje de entrada/salida (en volts) con $V_{DD} = V_{CC} = +5$ V.

	CMOS							TTL			
Parámetro	**4000B**	**74HC**	**74HCT**	**74AC**	**74ACT**	**74AHC**	**74AHCT**	**74**	**74LS**	**74AS**	**74ALS**
V_{IH}(mín)	3.5	3.5	2.0	3.5	2.0	3.85	2.0	2.0	2.0	2.0	2.0
V_{IL}(máx)	1.5	1.0	0.8	1.5	0.8	1.65	0.8	0.8	0.8	0.8	0.8
V_{OH}(mín)	4.95	4.9	4.9	4.9	4.9	4.4	3.15	2.4	2.7	2.7	2.5
V_{OL}(máx)	0.05	0.1	0.1	0.1	0.1	0.44	0.1	0.4	0.5	0.5	0.5
V_{NH}	1.45	1.4	2.9	1.4	2.9	0.55	1.15	0.4	0.7	0.7	0.7
V_{NL}	1.45	0.9	0.7	1.4	0.7	1.21	0.7	0.4	0.3	0.3	0.4

74HCT y 74ACT están diseñadas para ser compatibles eléctricamente con TTL, por lo que deben poder aceptar los mismos niveles de voltaje de entrada que TTL.

Márgenes de ruido

En la tabla 8-9 también se proporcionan los márgenes de ruido para cada serie; los cuales se calculan de la siguiente forma:

$$V_{NH} = V_{OH}(\text{mín}) - V_{IH}(\text{mín})$$
$$V_{NL} = V_{IL}(\text{máx}) - V_{OL}(\text{máx})$$

Observe que, en general, los dispositivos CMOS tienen mayores márgenes de ruido que TTL. La diferencia sería aún mayor si los dispositivos CMOS se operaran a un voltaje de alimentación mayor de 5 V.

Disipación de potencia

Cuando un circuito lógico CMOS se encuentra en un estado estático (que no cambia), su disipación de potencia es en extremo baja. Para ver la razón por la que hay que analizar cada uno de los circuitos de las figuras 8-22 a 8-24. Observe que, sin importar el estado de la salida, siempre hay una resistencia muy alta entre la terminal V_{DD} y tierra debido a que siempre hay un MOSFET apagado en la ruta de la corriente. Esto produce una disipación de potencia de corriente directa típica en el CMOS de sólo 2.5 nW por compuerta cuando $V_{DD} = 5$ V; aún con $V_{DD} = 10$ V, esta potencia se incrementa sólo a 10 nW. Con estos valores para P_D es fácil ver por qué CMOS se adapta en forma ideal a las aplicaciones en las que se utiliza energía por medio de baterías o energía de respaldo de baterías.

P_D se incrementa con la frecuencia

La disipación de potencia de un CI CMOS sería muy baja siempre y cuando se encuentre en una condición de corriente directa. Desafortunadamente, P_D se incrementará en proporción con la frecuencia en la que los circuitos cambian de estado. Por ejemplo, una compuerta NAND CMOS que tiene $P_D = 10$ nW bajo condiciones de corriente directa tendrá $P_D = 0.1$ mW a una frecuencia de 100 kpps, y de 1 mW a 1 MHz. La razón de esta dependencia en relación con la frecuencia se ilustra en la figura 8-25.

Cada vez que una salida CMOS cambia de BAJO a ALTO debe suministrarse una corriente de carga transitoria a la capacitancia de carga, la cual consiste de las capacitancias de entrada combinadas de todas las cargas que se estén manejando y de la propia capacitancia de salida del dispositivo. Estos picos estrechos de corriente se suministran a través de V_{DD}; pueden tener una amplitud típica de 5 mA y una duración de 20 a 30 ns. Es evidente que a medida que aumenta la frecuencia de

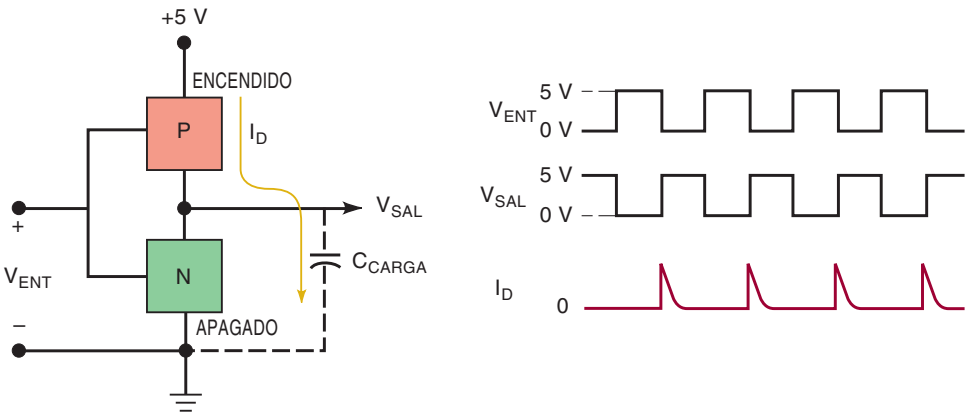


FIGURA 8-25 Los picos de corriente se drenan de la alimentación V_{DD} cada vez que la salida cambia de BAJO a ALTO. Esto se debe en su mayor parte a la corriente de carga de la capacitancia de carga.

conmutación, se producirán más de estos picos de corriente por segundo, y se incrementará la corriente actual promedio de V_{DD}. Aún con cargas capacitivas muy bajas hay un punto breve en la transición de BAJO a ALTO o de ALTO a BAJO cuando los dos transistores de salida se encienden en forma parcial. Esto reduce efectivamente la resistencia del voltaje de alimentación a tierra, con lo que se produce también un pico de corriente.

Por ende, a frecuencias mayores el circuito CMOS empieza a perder algunas de sus ventajas en comparación con otras familias lógicas. Como regla general, una compuerta C-MOS tendrá la misma P_D promedio que una compuerta 74LS a frecuencias cercanas de 2 a 3 MHz. Por encima de estas frecuencias también se incrementa la potencia de TTL con la frecuencia, debido a la corriente requerida para invertir la carga en la capacitancia de carga. Para los chips MSI, la situación es algo más compleja de como la describimos aquí, por lo que un diseñador lógico tendrá que realizar un análisis detallado para determinar si CMOS tiene o no una ventaja en cuanto a la disipación de potencia a una frecuencia de operación específica.

Capacidad de carga

Al igual que N-MOS y P-MOS, las entradas CMOS tienen una resistencia en extremo grande ($10^{12}\ \Omega$), la cual en esencia no drena corriente de la alimentación de la señal. No obstante, cada entrada CMOS presenta, por lo general, una carga de 5 pF a tierra. Esta capacitancia de entrada limita el número de entradas CMOS que puede

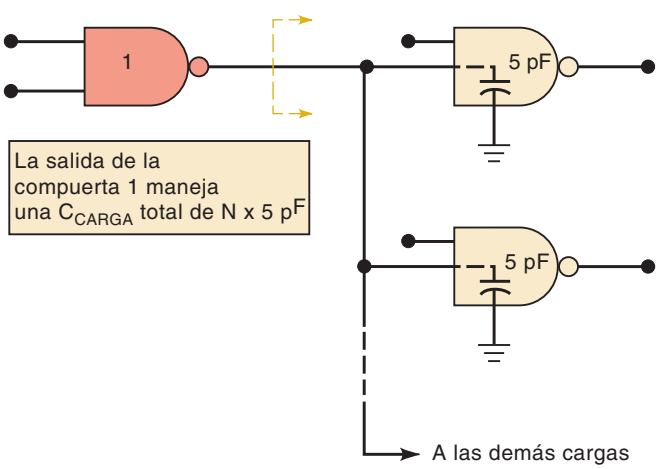


FIGURA 8-26 Cada entrada CMOS se suma a la capacitancia de carga total que ve la salida de la compuerta reforzadora.

manejar una salida CMOS (vea la figura 8-26). La salida CMOS debe cargar y descargar la combinación en paralelo de todas las capacitancias de entrada, de forma que se incremente el tiempo de conmutación de la salida en proporción al número de cargas que se controlan. Por lo general, cada carga CMOS incrementa el tiempo de propagación del circuito reforzador por 3 ns. Por ejemplo, la compuerta NAND 1 en la figura 8-26 podría tener un t_{PLH} de 25 ns si no manejara cargas; esto se incrementaría a 25 ns + 20(3 ns) = 85 ns si tuviera *veinte* cargas.

Así, la capacidad de carga CMOS depende del máximo tiempo de propagación permisible. Por lo general, las salidas CMOS están limitadas a una capacidad de carga de 50 para la operación en baja frecuencia (≤1 MHz). Desde luego que para la operación a frecuencias más altas, la capacidad de carga tendría que ser menor.

Velocidad de conmutación

Aunque los dispositivos CMOS al igual que N-MOS y P-MOS, tienen que controlar capacitancias de carga relativamente grandes, su velocidad de conmutación es un poco más rápida debido a su baja resistencia de salida en cada estado. Una salida N-MOS debe cargar la capacitancia de carga a través de una resistencia relativamente grande (100-kΩ). En el circuito CMOS, la resistencia de salida en el estado ALTO es la R_{ENC} del P-MOSFET, que, por lo general, es de 1 kΩ o menor. Esto permite una carga más rápida de la capacitancia de carga.

Por lo general, una compuerta NAND de la serie 4000 tiene un t_{pd} promedio de 50 ns a V_{DD} = 5 V, y de 25 ns a V_{DD} = 10 V. La razón de la mejora en t_{pd} a medida que se incrementa V_{DD} es que la R_{ENC} en los MOSFETs se reduce en forma considerable a voltajes de alimentación más alto. Por lo tanto, parece ser que V_{DD} debería hacerse lo más grande posible para la operación a frecuencias más altas. No obstante, el valor de V_{DD} más grande producirá un aumento en la disipación de potencia.

Una compuerta NAND ordinaria en la serie 74HC o 74HCT tiene un t_{pd} promedio aproximado de 8 ns cuando se opera a V_{DD} = 5 V. Una compuerta NAND 74AC/ACT tiene un t_{pd} promedio aproximado de 4.7 ns. Una compuerta NAND 74AHC tiene un t_{pd} promedio aproximado de 4.3 ns.

Entradas sin utilizar

Las entradas CMOS nunca se deben dejar desconectadas. Todas las entradas CMOS deben conectarse ya sea a un nivel de voltaje fijo (0 V o V_{DD}) o a otra entrada.

Esta regla se aplica incluso a las entradas de las compuertas lógicas adicionales que no se utilicen en un chip. Una entrada CMOS desconectada está susceptible al ruido y las cargas estáticas que podrían polarizar fácilmente los MOSFETs de canal P y de canal N en el estado conductivo, lo cual produciría un aumento en la disipación de potencia y un posible sobrecalentamiento.

Sensibilidad estática

Todos los dispositivos electrónicos, en mayor o menor grado, son sensibles al daño debido a la electricidad estática. El cuerpo humano es un excelente almacén de cargas electrostáticas. Por ejemplo, cuando usted camina a lo largo de una carpeta puede acumularse en su cuerpo una carga estática de más de 30,000 V. Si después toca un dispositivo electrónico, parte de esta carga puede llegar a transferirse al dispositivo. Las familias lógicas MOS (y todos los MOSFETs) son en especial susceptibles al daño por carga electrostática. Toda esta diferencia de potencial (carga estática) que se aplica a través de la película de óxido sobrepasa la capacidad de

aislamiento dieléctrico de la película. Cuando se rompe, el flujo de corriente resultante (descarga) es como un golpe de rayo, el cual hace un hoyo en la capa de óxido y daña el dispositivo en forma permanente.

La **descarga electrostática (ESD)** es responsable de los billones de dólares de daños a equipos electrónicos que se gastan por año, por lo que los fabricantes de equipo han dedicado una gran parte de su atención al desarrollo de procedimientos de manejo especial para todos los dispositivos y circuitos electrónicos. Aún y cuando la mayoría de los CIs modernos tienen redes de resistencias-diodos dentro del chip para proteger las entradas y las salidas de los efectos de la ESD, la mayoría de los laboratorios de ingeniería, las instalaciones de producción y los departamentos de servicio en el campo utilizan las siguientes precauciones:

1. Conecte el chasis de todos los instrumentos de prueba, las puntas de los cautines y su banco de trabajo (si es de metal) a tierra física (es decir, la terminal redonda en el enchufe de 120 VAC). Esto evita la acumulación de carga estática en estos dispositivos, la cual podría transferirse a cualquier tablero de circuitos o CI con el que pudieran tener contacto.
2. Conéctese usted mismo a tierra física con una muñequera especial. Esto permitirá que las cargas potencialmente dañinas de su cuerpo se descarguen a tierra. La muñequera contiene una resistencia de 1-MΩ que limita la corriente a un valor no letal, en caso de que usted toque por accidente un voltaje "vivo" mientras trabaja con el equipo.
3. Mantenga los CIs (en especial los MOS) en espuma conductora o en papel aluminio. Esto mantendrá todas las terminales del CI en corto, de manera que no puedan desarrollarse voltajes peligrosos entre dos terminales cualesquiera.
4. Evite tocar las terminales del CI e insértelo en el circuito justo después de removerlo de la caja protectora.
5. Coloque cintas para poner en corto los conectores de los bordes de las tarjetas de PC cuando los cargue o los transporte. Evite tocar los conectores del borde. Almacene los tableros de PC en plástico conductor o en sobres metálicos.
6. No deje entradas de CI sin usar desconectadas, ya que las entradas abiertas tienden a atraer cargas estáticas dispersas.

Efecto de cierre

Debido a la existencia inevitable de transistores PNP y NPN *parásitos* (indeseables) integrados en el sustrato de los CIs CMOS, puede producirse una condición conocida como **efecto de cierre** bajo ciertas circunstancias. Si estos transistores parásitos en un chip CMOS se disparan y empiezan a conducir, se cerrarán (permanecerán ENCENDIDOS en forma permanente) y podría fluir una corriente muy grande, con lo que se destruiría el CI. La mayoría de los CIs CMOS modernos están diseñados con circuitos de protección que ayudan a prevenir el efecto de cierre, pero de todas formas puede llegar a ocurrir cuando se exceden las clasificaciones de voltaje máximo del dispositivo. El efecto de cierre puede dispararse mediante picos de alto voltaje o sobretiros de voltaje en las entradas y salidas del dispositivo. Pueden conectarse diodos fijadores de nivel de manera externa para proteger el dispositivo contra los transitorios, en especial cuando los CIs se utilizan en ambientes industriales en los que se lleva a cabo la conmutación de cargas en alto voltaje y/o alta corriente (controladores de motores, relevadores, etc.). Una fuente de energía bien regulada minimizará los picos en la línea V_{DD}; si la fuente también tiene limitación de corriente, limitará la corriente en caso de que ocurra el efecto de cierre. Las técnicas de fabricación de CMOS modernas han reducido en forma considerable la susceptibilidad del CI al efecto de cierre.

PREGUNTAS DE REPASO

1. ¿Cuál de las series CMOS tiene compatibilidad de terminales con TTL?
2. ¿Cuál de las series CMOS es compatible eléctricamente con TTL?
3. ¿Cuál de las series CMOS es equivalente funcional de TTL?
4. ¿Qué familia lógica combina las mejores características de las lógicas CMOS y bipolar?
5. ¿Qué factores determinan la capacidad de carga de CMOS?
6. ¿Qué precauciones hay que tomar cuando se manejan CIs CMOS?
7. ¿Cuál de las familias de CIs (CMOS, TTL) es la que mejor se adapta a las aplicaciones operadas por baterías?
8. *Verdadero o falso:*
   (a) El drenado de energía de CMOS aumenta con la frecuencia de operación.
   (b) Las entradas CMOS que no se utilicen pueden dejarse desconectadas.
   (c) TTL se adapta mejor que CMOS para la operación en ambientes con mucho ruido.
   (d) La velocidad de conmutación de CMOS aumenta con la frecuencia de operación.
   (e) La velocidad de conmutación de CMOS aumenta con el voltaje de alimentación.
   (f) La condición de efecto de cierre es una ventaja de CMOS sobre TTL.

8-10 TECNOLOGÍA DE BAJO VOLTAJE

Los fabricantes de CIs están continuamente buscando formas de colocar dispositivos semiconductores (diodos, resistencias, transistores, etc.) lo más junto posible en un chip; es decir, buscan incrementar la densidad del chip. Esta mayor densidad tiene cuando menos dos beneficios principales. En primer lugar, permite encapsular más circuitos en el chip; en segundo lugar, como los circuitos están más cerca unos de otros, el tiempo para que las señales se propaguen de un circuito a otro disminuirá, con lo cual se mejorará la velocidad de operación del circuito en general. También hay desventajas en cuanto a tener una mayor densidad de chip. Cuando los circuitos se colocan más juntos, el material aislante que separa un circuito de otro es más estrecho. Esto reduce la cantidad de voltaje que puede soportar el circuito antes de que se produzca un rompimiento dieléctrico. Al aumentar la densidad del chip se aumenta la disipación de potencia total del chip, con lo cual se puede elevar la temperatura del chip por encima del máximo nivel permitido para una operación confiable.

Estas desventajas se pueden neutralizar si el chip se opera a niveles de voltaje mejores, con lo cual se reduce la disipación de potencia. Varias series lógicas en el mercado operan con 3.3 V. Las series más recientes están optimizadas para operar con 2.5 V. Esta tecnología de bajo voltaje podría muy bien significar el comienzo de una transición gradual en el campo de equipos digitales, en la que en un momento dado todos los CIs digitales operarán a partir de un nuevo estándar de bajo voltaje.

En la actualidad, los dispositivos de bajo voltaje están diseñados para aplicaciones que varían desde los juegos electrónicos hasta las estaciones de trabajo de ingeniería. Las CPUs más recientes son dispositivos de 2.5 V, y los chips de RAM dinámica de 3.3 V se utilizan en módulos de memoria para computadoras personales.

Actualmente existen varias series lógicas de bajo voltaje; desafortunadamente no nos es posible cubrir aquí todas las familias y series de todos los fabricantes, por lo que sólo describiremos las que ofrece actualmente Texas Instruments.

Familia CMOS

- La serie *74VLC (CMOS de bajo voltaje)* contiene el surtido más amplio de las conocidas compuertas SSI y funciones MSI de las familias de 5 V, junto con

muchos dispositivos de interfase de bus tales como búferes, latches, reforzadores, etcétera. Esta serie puede manejar niveles lógicos de 5-V en sus entradas, por lo que puede realizar conversiones de sistemas de 5-V a sistemas d 3-V. Siempre y cuando el control de corriente se mantenga lo suficientemente bajo como para mantener el voltaje de salida dentro de los límites aceptables, la serie 74VLC también podrá manejar entradas TTL de 5 V. Los requerimientos de entrada de V_{IH} de piezas CMOS de 5-V tales como la serie 74HC/AHC no permiten que los dispositivos LVC los controlen.

- En la actualidad, la serie *74ALVC (CMOS avanzado de bajo voltaje)* ofrece el rendimiento más alto. Los dispositivos de esta serie están diseñados principalmente para aplicaciones de interfase de bus en las que se utilice sólo lógica de 3.3 V.
- La serie *74LV (Bajo voltaje)* ofrece tecnología CMOS y muchas de las compuertas SSI y funciones lógicas MSI comunes, junto con algunos búferes octales populares, latches y flip-flops. Está diseñada para operar sólo con otros dispositivos de 3.3 V.
- La serie *74AVC (CMOS avanzado de muy bajo voltaje*) se introdujo teniendo en mente los sistemas del mañana. Está optimizada para sistemas de 2.5 V, pero puede operar con voltajes de alimentación con valores tan bajos como 1.2 V o de hasta 3.3 V. Este amplio intervalo de voltaje de alimentación la hace útil en sistemas con voltajes mixtos. Tiene tiempos de propagación de menos de 2 ns, lo cual hace de esta serie un digno rival de los dispositivos bipolares 74AS. Tiene muchas de las características de interfase de bus de las series BiCMOS que la harán útil en futuras generaciones de estaciones de trabajo de bajo voltaje, PCs, redes y equipo de telecomunicaciones.
- La serie *74AUC (CMOS avanzado de ultra bajo voltaje)* está optimizada para operar a niveles lógicos de 1.8 V.
- La serie *74AUP (Ultra baja potencia avanzada)* es la serie lógica de menor potencia de la industria y se utiliza en aplicaciones portátiles, operadas por baterías.
- La serie *74CBT (Tecnología de interconexión cruzada)* ofrece circuitos de interfase de bus de alta velocidad que pueden conmutar con rapidez cuando se habilitan, y no cargan el bus cuando se deshabilitan.
- La serie *74CBTLV (Tecnología transversal de bajo voltaje)* es el complemento de 3.3 V para la serie 74CBT.
- La serie *74GTLP (Lógica de transceptor de disparo Plus)* está hecha para aplicaciones de segundo plano en paralelo de alta velocidad. Veremos esta serie en una sección posterior del libro.
- La serie *74SSTV (Lógica terminada en series de acomplamiento)* es útil en los sistemas de memoria avanzada de alta velocidad de las computadoras modernas.
- La serie *Interruptor TS (Interruptor de señal TI)* está hecha para aplicaciones de señales mixtas y ofrece algunas soluciones de conmutación analógica y digital, y de multiplexeo.
- La serie *74TVC (Circuito de sujeción de traducción de voltaje)* se utiliza para proteger las entradas y salidas de los dispositivos sensibles contra el exceso de voltaje en las líneas de bus.

Familia BiCMOS

- La serie *74LVT (Tecnología BiCMOS de bajo voltaje)* contiene piezas BiCMOS que están diseñadas para aplicaciones de interfase de bus de 8 bits y de 16 bits. Al igual que en la serie LVC, las entradas pueden manejar niveles lógicos de 5 V y sirven como un traductor de 5 V a 3 V. Como los niveles de salida [V_{OH}(mín) y V_{OL}(máx)] son equivalentes a los niveles TTL, son totalmente compatibles eléctricamente con TTL. La tabla 8-10 compara las diversas características.
- La serie *74ALVT (Tecnología BiCMOS avanzada de bajo voltaje)* es una mejora en relación con la serie LVT. Ofrece la operación con 3.3 V o 2.2 V a 3 ns y sus terminales son compatibles con las de las series ABT y LVT existentes. También está diseñada para aplicaciones de interfase de bus.

TABLA 8-10 Características de las series de bajo voltaje.

	LV	ALVC	AVC	ALVT	ALB
V_{CC} (recomendado)	2.7–3.6	2.3–3.6	1.65–3.6	2.3–2.7	3–3.6
T_{PD} (ns)	18	3	1.9	3.5	2
V_{IH} (V)	De 2 a V_{CC} + 0.5	De 2 a 4.6	De 1.2 a 4.6	De 2 a 7	D 2.2 a 4.6
V_{IL} (V)	0.8	0.8	0.7	0.8	0.6
I_{OH} (mA)	6	12	8	32	25
I_{OL} (mA)	6	12	8	32	25

- La serie *74ALB (BiCMOS avanzada de bajo voltaje)* está diseñada para aplicaciones de interfase de bus de 3.3 V. Proporciona un control de salida de 25 mA y tiempos de propagación de sólo 2.2 ns.
- La serie *74VME (Módulo VERSA Eurocard)* está diseñada para operar con la tecnología de bus VME estándar.

Los técnicos e ingenieros en servicios digitales ya no pueden suponer que todos los CIs en un circuito, sistema o pieza de equipo digital está operando a 5, y deben estar preparados para tratar con las consideraciones de interfase necesarias en los sistemas de voltajes mixtos. Las habilidades de integración que usted aprenderá en este capítulo le permitirán lograr esto, sin importar lo que se desarrolle a medida que se vuelvan más comunes los sistemas de bajo voltaje.

El continuo desarrollo de la tecnología de bajo voltaje promete provocar una completa revolución desde el sistema original de 5 V, a los sistemas de voltajes mixtos y, finalmente, a los sistemas digitales puros de 3.3 V, de 2.5 V o incluso de voltajes más bajos. Para poner todo esto en perspectiva, en la figura 8-27 se muestra la percepción de Texas Instruments del ciclo de vida de las diversas familias lógicas.

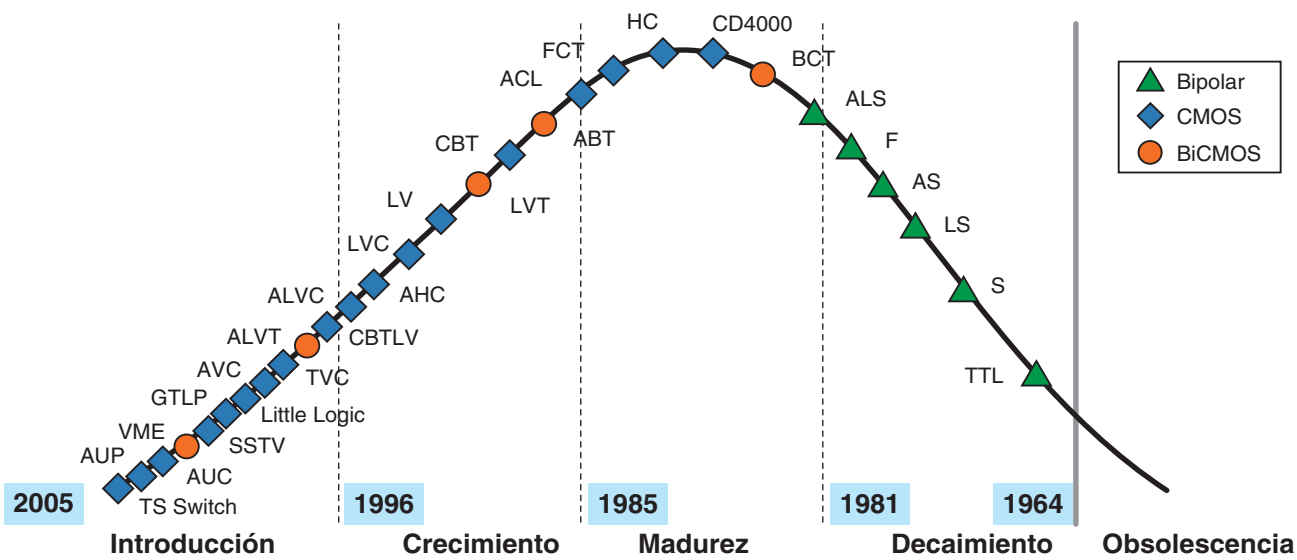


FIGURA 8-27 Ciclo de vida de los productos lógicos. (Cortesía de Texas Instruments.)

PREGUNTAS DE REPASO

1. ¿Cuáles son las dos ventajas de los CIs de mayor densidad?
2. ¿Cuáles son las desventajas?
3. ¿Cuál es el mínimo voltaje en ALTO en la entrada de un 74LVT?
4. ¿Cuál de las series de bajo voltaje puede trabajar sólo con otros CIs de series de bajo voltaje?
5. ¿Cuál de las series de bajo voltaje es totalmente compatible eléctricamente con TTL?

8-11 SALIDAS DE COLECTOR ABIERTO/DRENADOR ABIERTO

Algunas veces, varios dispositivos digitales comparten el uso de un solo conductor para poder transmitir una señal hacia algún dispositivo de destino, en forma muy parecida a como los vecinos comparten la misma calle. Esto significa que varios dispositivos deben tener sus salidas conectadas al mismo conductor, lo que en esencia los conecta a unos con otros. Para todos los dispositivos lógicos que hemos considerado hasta ahora, esto representa un problema. Cada salida tiene dos estados, ALTO y BAJO. Cuando una salida está en ALTO mientras la otra está en BAJO y cuando se conectan juntas tenemos un conflicto ALTO/BAJO. ¿Cuál de los dos ganará? Al igual que las vencidas, el más fuerte de los dos es el que ganará. En este caso, el circuito de transistores cuyo transistor de salida tenga la resistencia en "ENCENDIDO" más baja arrastrará el voltaje de salida hacia su dirección.

La figura 8.28 muestra un diagrama de bloques genérico de dos dispositivos lógicos con sus salidas conectadas a un conductor común. Si los dos dispositivos lógicos fueran CMOS, entonces la resistencia en ENCENDIDO del circuito de puesta en alto que produzca el nivel ALTO como salida sería aproximadamente igual a la resistencia en ENCENDIDO del circuito de puesta en bajo que produzca el nivel BAJO como salida. El voltaje en el conductor común será de un valor aproximado a la mitad del voltaje de alimentación. Este voltaje se encuentra en el intervalo indeterminado para la mayoría de las series CMOS y es inaceptable para manejar una entrada CMOS. Lo que es más, la corriente a través de los dos MOSFETs conductores será mucho mayor de lo normal, en especial con valores más altos de V_{DD}, y podría dañar los CIs.

Las salidas CMOS convencionales nunca deben conectarse juntas.

Si los dos dispositivos fueran salidas TTL en forma de tótem como se muestra en la figura 8-29, se produciría una situación similar pero con distintos resultados

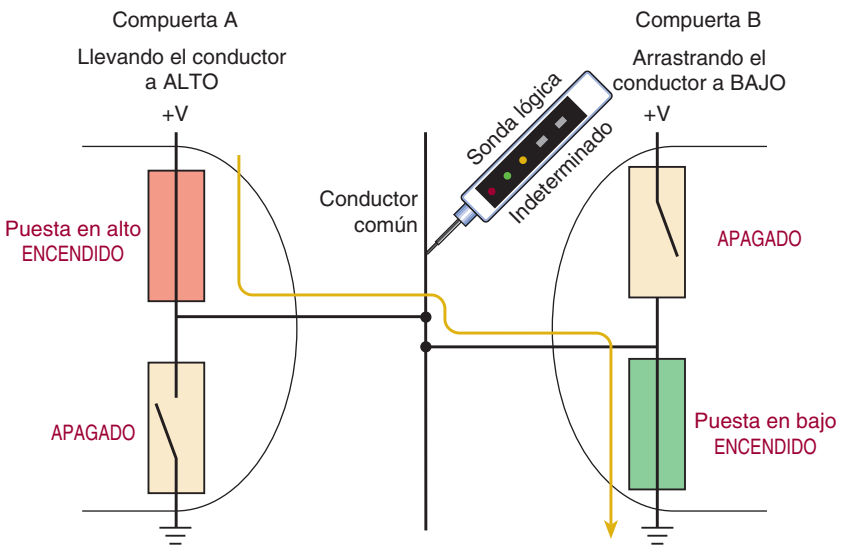


FIGURA 8-28 Dos salidas que luchan por el control de un conductor.

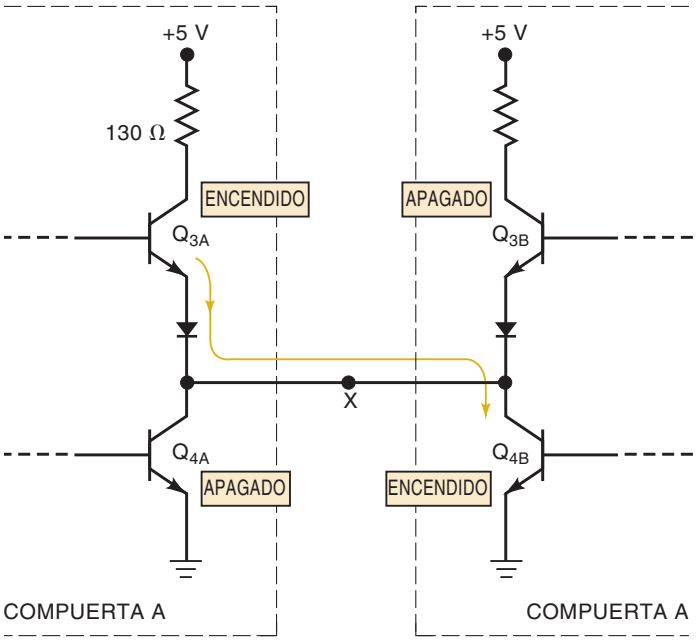


FIGURA 8-29 Las salidas en forma de tótem que se conectan juntas pueden producir una corriente dañina a través de Q_4.

debido a la diferencia en los circuitos de salida. Suponga que la salida de la compuerta *A* se encuentra en el estado ALTO (Q_{3A} ENCENDIDO, Q_{4A} APAGADO) y que la salida de la compuerta *B* se encuentra en el estado BAJO (Q_{3B} APAGADO, Q_{4B} ENCENDIDO). En esta situación, Q_{4B} representa una carga de muy baja resistencia en Q_{3A} y drenará una corriente mucho mayor de la que está clasificado para manejar. Esta corriente podría no dañar de inmediato a Q_{3A} o Q_{3B}, pero durante un periodo de tiempo podría causar un sobrecalentamiento y deterioro en el rendimiento, con la eventual falla del dispositivo.

Otro problema que se produce por esta corriente relativamente alta que fluye a través de Q_{4B} es que producirá una caída mayor de voltaje a través del colector emisor del transistor y el valor de V_{OL} estaría entre 0.5 y 1 V. Esto es mayor que el V_{OL}(máx) permisible. Por estas razones:

Las salidas TTL en forma de tótem nunca deben conectarse juntas.

Salidas de colector abierto/drenador abierto

Una solución al problema de compartir un conductor común entre compuertas es remover el transistor de puesta en alto activo del circuito de salida de cada compuerta. De esta forma, ninguna de las compuertas tratará siquiera de asegurar un nivel ALTO lógico. Las salidas TTL que están modificadas de esta forma se llaman **salidas de colector abierto**. Los circuitos CMOS de salida que se modifican de esta forma se llaman salidas de drenador abierto. La salida se toma en el drenador del MOSFET de puesta en bajo de canal N, el cual es un circuito abierto (es decir, no está conectado a ningún otro circuito).

El equivalente de TTL se llama salida de colector abierto ya que el colector del transistor inferior en el tótem se conecta en forma directa a la terminal de salida y en ninguna otra parte, como se muestra en la figura 8-30(a). La estructura de colector abierto elimina los transistores de puesta en alto Q_3, D_1 y R_4. En el estado BAJO de salida, Q_4 está ENCENDIDO (tiene corriente de base y en esencia es un corto entre colector y emisor); en el estado ALTO de salida, Q_4 está APAGADO (no tiene corriente de base y es en esencia un circuito abierto entre colector y emisor). Como este circuito no tiene una forma interna de elevar la salida a ALTO, el diseñador del circuito debe conectar una resistencia de puesta en alto externa R_P a la salida, como se muestra en la figura 8-30(b).

Cuando Q_4 está ENCENDIDO, lleva el voltaje de salida a BAJO. Cuando Q_4 está APAGADO, R_P lleva la salida de la compuerta a ALTO. Observe que sin la resistencia

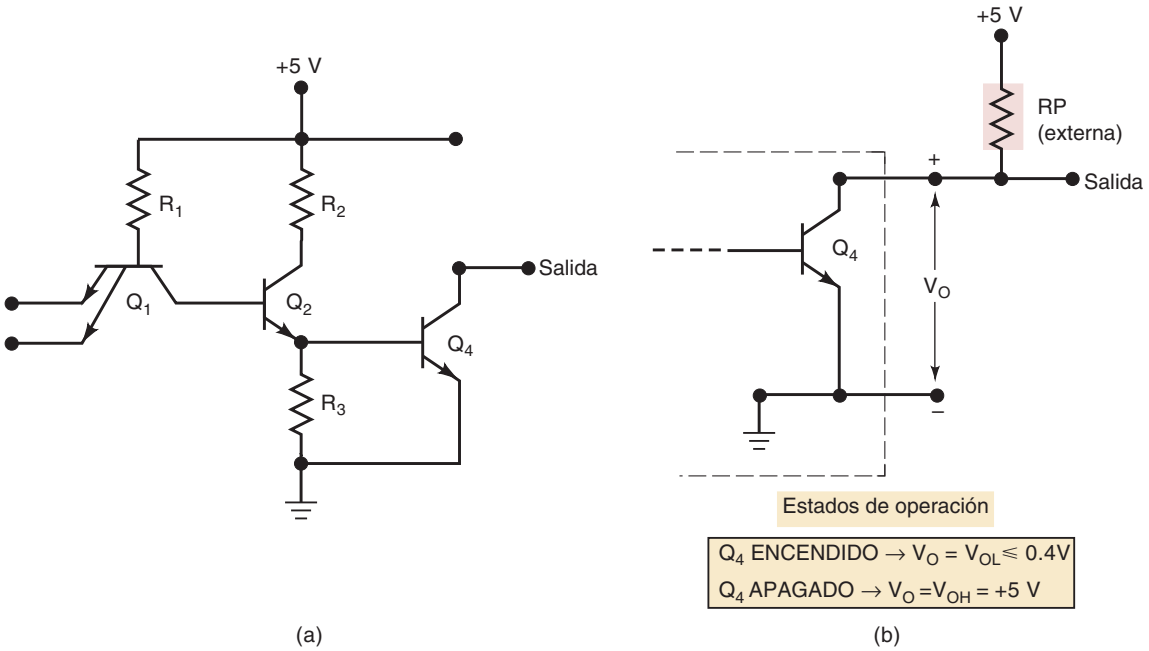


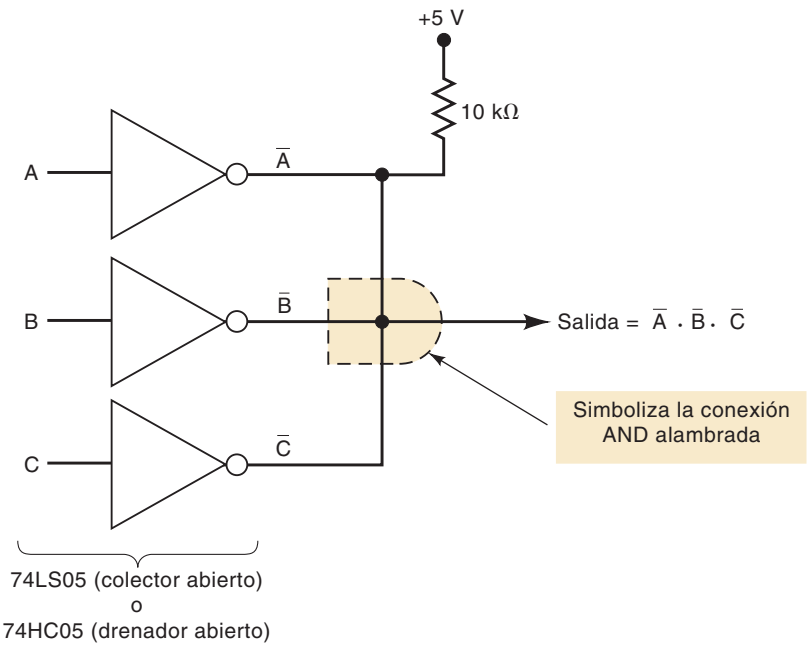
FIGURA 8-30 (a) Circuito TTL de colector abierto; (b) con resistencia de puesta en alto externa.

de puesta en alto, el voltaje de salida sería indeterminado (flotante). El valor de la resistencia R_P se elije, por lo general, de forma que sea 10 kΩ. Este valor es lo bastante pequeño como para que, en el estado ALTO, la caída de voltaje a través de la resistencia debido a la corriente de carga no reducirá el voltaje de salida por debajo del V_{OH} mínimo. Es lo bastante grande como para que, en el estado BAJO, limite la corriente que pasa a través de Q_4 a un valor por debajo de I_{OL}(máx).

Cuando varias compuertas de colector abierto o de drenador abierto comparten una conexión común, como se muestra en la figura 8-31, el conductor común está en ALTO de manera predeterminada, debido a la resistencia de puesta en alto. Cuando cualquiera (una o más) de las salidas de las compuertas lo lleva a BAJO, los 5 V caen

FIGURA 8-31
Operación AND alambrado mediante el uso de compuertas de colector abierto.





a través de R_P y la conexión común se encuentra en el estado BAJO. Como la salida común está en ALTO sólo cuando todas las salidas se encuentran en el estado ALTO, si se conectan las salidas de esta forma se implementa en forma básica la función AND lógica. A ésta se le conoce como conexión **AND alambrada** y se muestra en forma simbólica mediante el símbolo de compuerta AND punteado. En realidad ahí no hay una compuerta AND. *Una compuerta AND alambrada sólo puede implementarse con dispositivos lógicos TTL de colector abierto y CMOS de drenador abierto*.

Para sintetizar, los circuitos de colector abierto/drenador abierto no pueden hacer que sus salidas estén en ALTO; sólo pueden llevarlas a BAJO. Podemos utilizar esta característica para permitir que varios dispositivos compartan el mismo conductor para transmitir un nivel lógico a otro dispositivo, o para combinar las salidas de los dispositivos con efectividad, en una función AND lógica. Como dijimos antes, el propósito del transistor de puesta en alto activo en el circuito de salida de las compuertas convencionales es cargar con rapidez la capacitancia de carga y permitir una conmutación más rápida. Los dispositivos de colector abierto y drenador abierto tienen una velocidad de conmutación mucho más lenta de BAJO a ALTO y en consecuencia no se utilizan en aplicaciones de alta velocidad.

Búfer/Reforzadores de colector abierto/drenador abierto

Las aplicaciones de las salidas de colector/drenador abierto que hemos descrito se utilizaban más en los primeros días de los circuitos lógicos de lo que se emplean actualmente. Un uso más común de estos circuitos ahora es como un **búfer/reforzador**. Un búfer o reforzador es un circuito lógico que está diseñado para tener una mayor capacidad de corriente y/o voltaje de salida que un circuito lógico ordinario. Estos dispositivos permiten que un circuito de salida más débil controle una carga pesada. Los circuitos de colector/drenador abierto ofrecen cierta flexibilidad única como búfer/reforzadores.

Debido a sus especificaciones altas de I_{OL} y V_{OH}, el 7406 y el 7407 son los únicos dispositivos TTL estándar que aún se recomiendan para nuevos diseños. El 7406 es un CI búfer/reforzador de colector abierto, que contiene seis INVERSORes con salida de colector abierto que pueden drenar hasta 40 mA en el estado BAJO. Además, el 7406 puede manejar voltajes de salida de hasta 30 V en el estado ALTO. Esto significa que la salida puede conectarse a una carga que opere con un voltaje mayor de 5 V. En la figura 8-32 se ilustra este caso, en donde se utiliza un 7406 como búfer entre un flip-flop 74LS112 y una lámpara incandescente con valores nominales de 24 V, 25 mA. El 7406 controla el estado ENCENDIDO/APAGADO de la lámpara para indicar el estado de la salida Q del FF. Observe que la lámpara se energiza a través de +24 V y actúa como resistencia de puesta en alto para la salida de colector abierto.

Cuando $Q = 1$ la salida del 7406 cambia a BAJO, su transistor de salida drena los 25 mA de corriente de la lámpara que suministra la fuente de 24 V y la lámpara se enciende. Cuando $Q = 0$ el transistor de salida del 7406 se apaga; no hay camino para la corriente y la lámpara se apaga. En este estado aparecerán todos los 24 V a

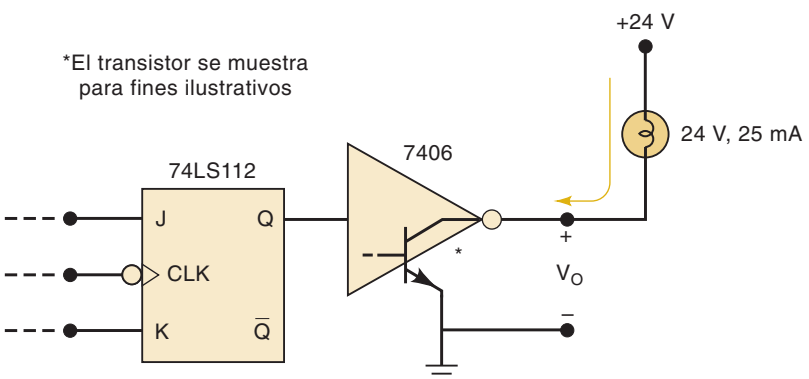


FIGURA 8-32 Un búfer/reforzador de colector abierto controla una carga con alta corriente y alto voltaje.

través del transistor de salida APAGADO, de manera que $V_{OH} = 24\,V$, un valor menor que la clasificación máxima de V_{OH} para el 7406.

Las salidas de colector abierto se utilizan con frecuencia para controlar LEDs indicadores, como se muestra en la figura 8-33(a). La resistencia se utiliza para limitar la corriente a un valor seguro. Cuando la salida del INVERSOR está en BAJO, su transistor de salida proporciona una ruta de baja resistencia a tierra para la corriente del LED, por lo que éste se enciende. Cuando la salida del INVERSOR está en ALTO, su transistor de salida está apagado y no hay ruta para la corriente del LED; en este estado, el LED se apaga.

El 7407 es un búfer no inversor de colector abierto, con las mismas clasificaciones de voltaje y corriente que un 7406.

El 74HC05 es un inversor hexadecimal de colector abierto con capacidad de drenar 25 mA de corriente. La figura 8-33(b) muestra una forma de integrar un FF-D 74AHC74 con un relevador de control, el cual es un interruptor electromagnético. Los contactos se cierran magnéticamente cuando la corriente nominal fluye a través de la bobina. El 74HC05 puede manejar el voltaje y corriente relativamente altos del relevador, de manera que la salida del 74AHC74 pueda encender y apagar el relevador.

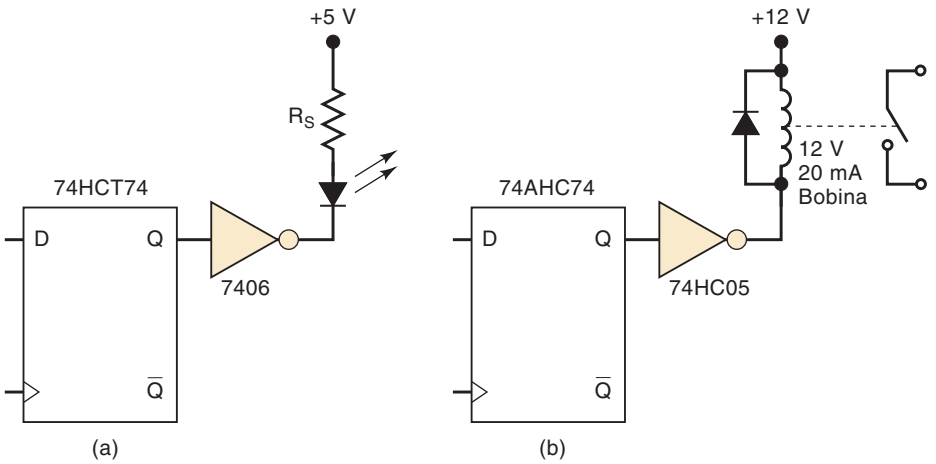


FIGURA 8-33
(a) Una salida de colector abierto puede usarse para controlar un indicador de LED; (b) una salida CMOS de colector abierto.

Símbolo IEEE/ANSI para salidas de colector/drenador abierto

La nueva simbología IEEE/ANSI utiliza una notación distinta para identificar salidas de colector/drenador abierto. La figura 8-34 muestra la designación IEEE/ANSI estándar para una salida de colector/drenador abierto. Es un diamante subrayado. Aunque por lo general no utilizaremos la simbología IEEE/ANSI completa en este libro, sí usaremos este diamante subrayado para indicar salidas de colector abierto y drenador abierto.

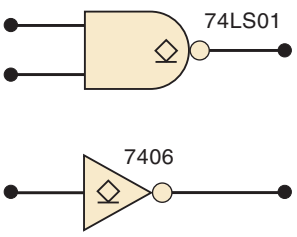


FIGURA 8-34 Notación IEEE/ANSI para salidas de colector abierto y drenador abierto.

PREGUNTAS DE REPASO

1. ¿Cuándo ocurre un conflicto ALTO/BAJO?
2. ¿Por qué no deben conectarse juntas las salidas en forma de tótem?
3. ¿Qué diferencia hay entre las salidas de colector abierto y las salidas en forma de tótem?
4. ¿Por qué las salidas de colector abierto necesitan una resistencia de puesta en alto?
5. ¿Cuál es la expresión lógica para la conexión AND alambrada de seis salidas de 7406?
6. ¿Por qué las salidas de colector abierto son, por lo general, más lentas que las salidas en forma de tótem?
7. ¿Cuál es el símbolo IEEE/ANSI para las salidas de colector abierto?

8-12 SALIDAS LÓGICAS TRIESTADO (TRES ESTADOS)

La configuración **triestado** es un tercer tipo de circuitos de salida que se utilizan en las familias TTL y CMOS. Aprovechan la operación de alta velocidad del arreglo de salida puesta en alto/puesta en bajo, al tiempo que permite conectar las salidas juntas para compartir un conductor común. Se llama triestado porque permite tres posibles los estados de salida: ALTO, BAJO y alta impedancia (Hi-Z). El estado Hi-Z es una condición en la que tanto el transistor de puesta en alto como el de puesta en bajo se APAGAN, de forma que la terminal de salida sea una alta impedancia tanto para tierra como para el voltaje de alimentación +V. La figura 8-35 ilustra estos tres estados para un circuito inversor simple.

Los dispositivos con salidas triestado tienen una entrada de *habilitación*. A menudo esta entrada se etiqueta como *E* (habilita) o como *OE* (habilita salida).

Cuando *OE* = 1, como se muestra en las figuras 8-35(a) y (b), el circuito opera como INVERSOR normal, ya que el nivel lógico ALTO en *OE* habilita la salida. Ésta se encontrará en ALTO o en BAJO, dependiendo del nivel de entrada. Cuando *OE* = 0, como se muestra en la figura 8-35(c), la salida del circuito se *deshabilita*. Cambia a su estado Hi-Z con ambos transistores en el estado no conductor. En este estado, la terminal de salida es en esencia un circuito abierto (no está conectada a nada).

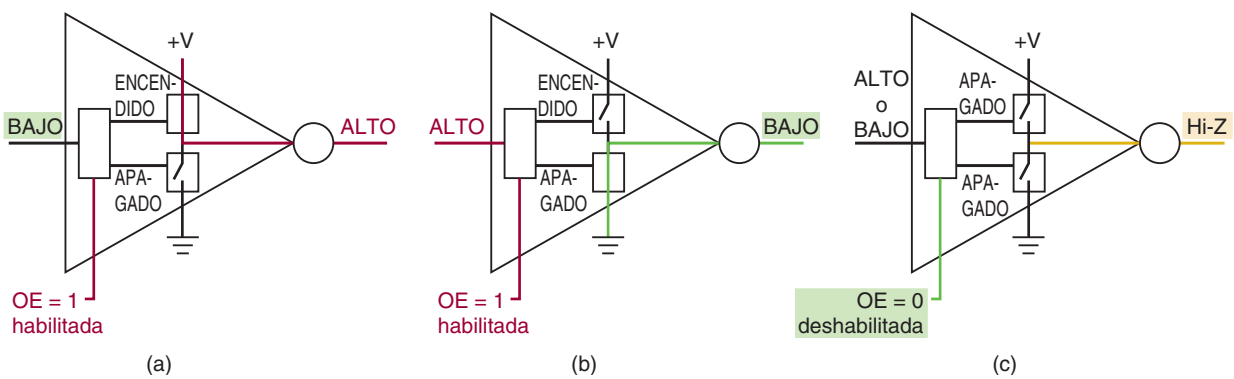


FIGURA 8-35 Las tres posibles condiciones de una salida triestado.

Ventaja de las salidas triestado

Las salidas de los CIs triestado pueden conectarse juntas (comparten el uso de un conductor común) sin sacrificar la velocidad de conmutación, ya que cuando se habilita una salida triestado opera como una salida en forma de tótem para TTL, o como

una salida CMOS de puesta en alto/puesta en bajo activa con sus características asociadas de alta impedancia y alta velocidad. No obstante, es importante tener en cuenta que cuando las salidas triestado se conectan juntas, sólo una de ellas debe estar habilitada en un momento dado. En caso contrario, dos salidas activas podrían luchar por el control del conductor común, como vimos antes, con lo cual fluirían corrientes dañinas y se producirían niveles lógicos inválidos.

Al hablar sobre los circuitos de colector/drenador abierto y triestado, nos hemos referido a casos en los que las salidas de varios dispositivos deben compartir un solo conductor para transmitir la información a otro dispositivo. A ese conductor compartido se le conoce como línea de bus. Un bus completo está compuesto de varios líneas conductoras que se utilizan para transportar información digital entre dos o más dispositivos que comparten el uso del bus.

Búferes triestado

Un *búfer triestado* es un circuito que se utiliza para controlar el paso de una señal lógica de la entrada a la salida. Algunos búferes triestado también invierten la señal a medida que va pasando. A los circuitos de la figura 8-35 se les puede llamar *búferes triestado inversores*.

Dos de los CIs de búfer triestado que se utilizan comúnmente son el 74LS125 y el 74LS126. Ambos contienen cuatro búferes triestado *no inversores*, como los que se muestran en la figura 8-36. El 74LS125 y el 74LS126 difieren sólo en el estado activo de sus entradas HABILITAR (ENABLE). El 74LS125 permite que la señal de entrada A llegue a la salida cuando $E = 0$, mientras que el 74LS126 pasa la entrada cuando $E = 1$.

Los búferes triestado tienen muchas aplicaciones en circuitos en los que se conectan varias señales a líneas comunes (buses). En el capítulo 9 examinaremos algunas de estas aplicaciones, pero podemos darnos una idea básica a través de la figura 8-37(a). Aquí tenemos tres señales lógicas A, B y C, las cuales se conectan a una línea de bus común a través de búferes triestado 74AHC126. Este arreglo nos permite transmitir cualquiera de estas señales a través de la línea del bus hacia otros circuitos; para ello hay que habilitar el búfer apropiado.

Por ejemplo, considere la situación en la figura 8-37(b), en donde $E_B = 1$ y $E_A = E_C = 0$. Esto deshabilita los búferes superior e inferior, de manera que sus salidas se encuentran en el estado Hi-Z y se desconectan del bus. Esto se simboliza mediante las letras X en el diagrama. El búfer medio se habilita de manera que su entrada B pase hacia su salida y hacia el bus, del cual se enruta hacia otros circuitos conectados al bus. Cuando se conectan salidas triestado juntas como en la figura 8-37, es importante recordar que no debe habilitarse más de una salida a la vez. En caso contrario se conectarían dos o más salidas en forma de tótem, lo cual podría producir corrientes dañinas. Aún si no se provocaran daños, esta situación produciría

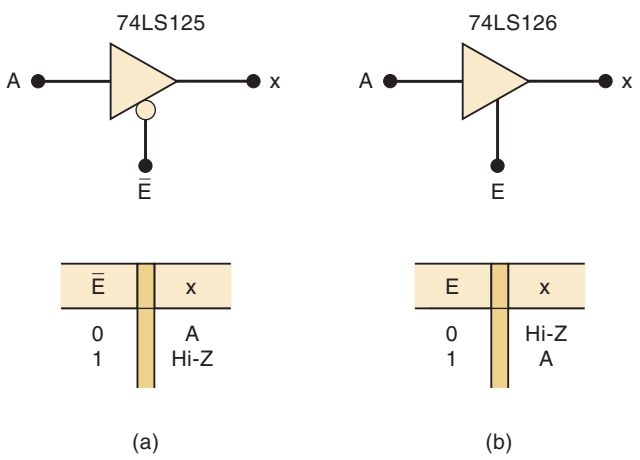


FIGURA 8-36 Búferes triestado no inversores.

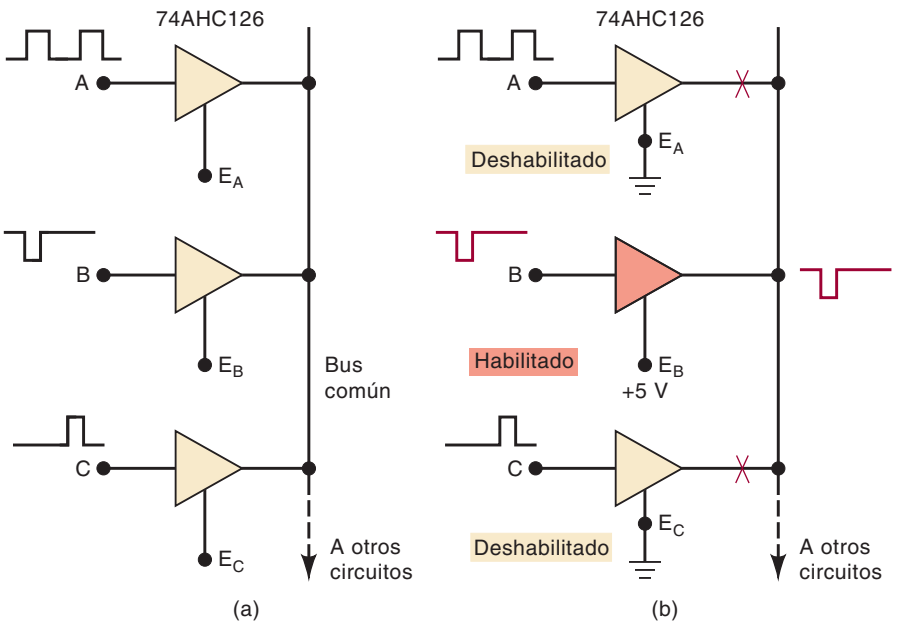


FIGURA 8-37
(a) Búferes triestado que se utilizan para conectar varias señales a un bus común; (b) condiciones para transmitir *B* al bus.

una señal en el bus, la cual es una combinación de dos o más señales. A esto se le conoce comúnmente como **colisión de bus**. La figura 8-38 muestra el efecto de habilitar las salidas *A* y *B* en forma simultánea. En la figura 8-37, cuando las entradas *A* y *B* se encuentran en estados opuestos, contienden por el control del bus. El voltaje resultante en el bus es un estado lógico inválido. En los sistemas de bus triestado, el diseñador debe asegurarse que las señales de habilitación no permitan la colisión de bus.

CIs triestado

Además de los búferes triestado, muchos CIs están diseñados con salidas triestado. Por ejemplo, el 74LS374 es un CI de registro octal de FFs tipo D con salidas triestado. Esto significa que es un registro de ocho bits compuesto de FFs tipo D, cuyas salidas están conectadas a búferes triestado. Este tipo de registro puede conectarse a líneas de bus comunes junto con las salidas de otros dispositivos similares para permitir la transferencia eficiente de datos a través del bus. En el capítulo 9 examinaremos este arreglo de *bus de datos triestado*. Entre otros tipos de dispositivos lógicos que están disponibles con salidas triestado están los decodificadores, los multiplexores, los convertidores analógicos/digitales, los chips de memoria y los microprocesadores.

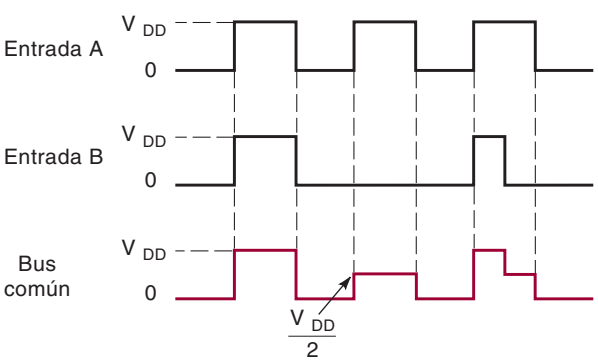


FIGURA 8-38 SI dos salidas CMOS habilitadas se conectan juntas, el bus estará a un voltaje aproximado a $V_{DD}/2$ cuando las salidas estén tratando de ser distintas.

Símbolo IEEE/ANSI para salidas triestado

La simbología lógica tradicional no tiene una notación especial para las salidas triestado. La figura 8-39 muestra la notación que se utiliza en la simbología IEEE/ANSI para indicar una salida triestado. Es un triángulo que apunta hacia abajo. Aunque no forma parte de la simbología tradicional, utilizaremos este triángulo para designar las salidas triestado durante el resto del libro.

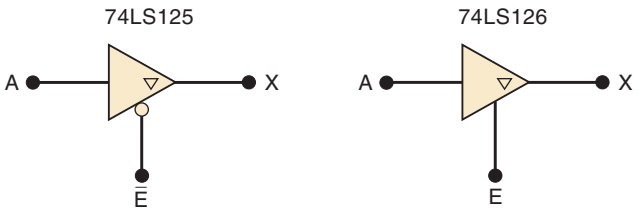


FIGURA 8-39 Notación IEEE/ANSI para las salidas triestado.

PREGUNTAS DE REPASO

1. ¿Cuáles son los tres estados posibles de una salida triestado?
2. ¿Cuál es el estado de una salida triestado cuando está deshabilitada?
3. ¿Qué es la colisión de bus?
4. ¿Qué condiciones son necesarias para transmitir la señal *C* al bus en la figura 8-37?
5. ¿Cuál es la designación IEEE/ANSI para las salidas triestado?

8-13 LÓGICA DE INTERFASE DE BUS DE ALTA VELOCIDAD

Muchos sistemas digitales utilizan un bus compartido para transferir señales y datos digitales entre los diversos componentes del sistema. Como puede ver de nuestra discusión sobre el desarrollo de la tecnología CMOS, los sistemas se están haciendo cada vez más rápidos. Muchas de las series lógicas de alta velocidad más recientes están diseñadas de manera específica para integrarse a un sistema de bus triestado. Los componentes en estas series son principalmente búferes triestado, transceptores bidireccionales, latches y reforzadores de línea de alta corriente.

A menudo hay una distancia considerable para separar físicamente los componentes en estos sistemas. Si esta distancia es mayor a unas 4 pulgadas, los conductores del bus entre ellos necesitan verse como una línea de transmisión. Aunque la teoría sobre la línea de transmisión podría ocupar todo un libro y se encuentra más allá del alcance de este texto, la idea general es bastante simple. Los conductores tienen inductancia, capacitancia y resistencia, lo cual significa que para las señales cambiantes (corriente alterna) tienen una impedancia característica que puede afectar a una señal que se coloca en un extremo y distorsionarla cuando llega al otro extremo. A las altas velocidades de las que estamos hablando, el tiempo de recorrido a través del conductor, los efectos de las ondas reflejadas (como ecos) y la sobreoscilación se vuelven preocupaciones reales. Existen varias formas de combatir los problemas asociados con las líneas de transmisión. Para poder evitar las ondas de pulso reflejadas, el extremo del bus se debe terminar con una resistencia que sea igual a la impedancia de la línea (cerca de 50 Ω), como se muestra en la figura 8-40(a). Este método no es factible, ya que se requiere demasiada corriente para mantener los voltajes de los niveles lógicos a través de una resistencia de ese tamaño. Otra técnica utiliza un capacitor para bloquear la corriente directa cuando la línea no está cambiando, pero en efecto aparece justo como una resistencia al pulso de elevación o de caída. En la figura 8-40(b) se muestra este método.

Si se utiliza un divisor de voltaje como en la figura 8-40(c), con resistencias más grandes que la de la línea, la impedancia ayuda a reducir las reflexiones pero con cientos de líneas de bus individuales es obvio que representa una carga pesada en la fuente de energía del sistema. La terminación con diodo que se muestra en la

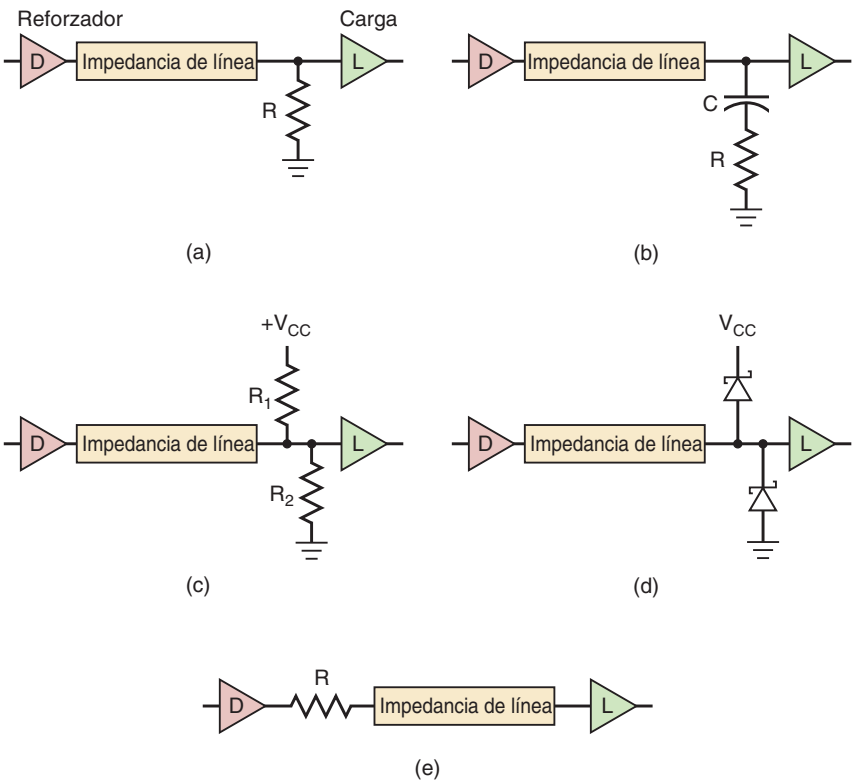


FIGURA 8-40 Técnicas de terminación de bus.

figura 8-40(d) sólo recorta o fija el sobredisparo/subdisparo de la sobreoscilación ocasionada por la naturaleza LC reactiva de la línea. La terminación en serie en la fuente, como se muestra en la figura 8-40(e), reduce la velocidad de conmutación y por consecuencia reduce los límites de frecuencia del bus, pero mejora en forma considerable la confiabilidad de las señales de bus.

Como puede ver, ninguno de estos métodos es ideal. Los fabricantes de CIs están diseñando nuevas series de circuitos lógicos que pueden sobrepasar muchos de estos problemas. La serie lógica de interfase de bus de Texas Instruments ofrece circuitos de salida que reducen en forma dinámica la impedancia de salida durante el estado estable (como una terminación en serie) para amortiguar cualquier sobreoscilación y reducir las reflexiones en la línea de bus. La serie GTLP (Lógica de transceptor de disparo Plus) de dispositivos de interfase de bus está diseñada especialmente para reforzar los buses relativamente extensos que conectan módulos de un sistema digital grande. El segundo plano se refiere a las interconexiones entre los módulos en la parte posterior de un sistema de montaje en bastidor de 19 pulgadas, estándar en la industria.

Otro de los principales participantes en el área de la interfase de bus de alta velocidad es la denominada **señalización diferencial de bajo voltaje (LVDS)**, la cual utiliza dos conductores para cada señal; señalización diferencial significa que responde a la diferencia entre los dos conductores. Por lo general se presentan señales de ruido indeseables en ambas líneas, las cuales no tienen efecto sobre la diferencia entre las dos señales. Para representar los dos estados lógicos, la LVDS utiliza una oscilación de bajo voltaje pero cambia de polaridad para distinguir con claridad un 1 de un 0.

PREGUNTAS DE REPASO

1. ¿Qué tan cerca necesitan estar los componentes unos de otros para ignorar los efectos de la "línea de transmisión"?
2. ¿Cuáles son las tres características de los conductores reales que se agregan para distorsionar señales que pasan a través de ellos?
3. ¿Cuál es el propósito de las terminaciones de bus?

8-14 LA FAMILIA ECL DE CIS DIGITALES

La familia TTL utiliza transistores que operan en el modo saturado. Como resultado, su velocidad de conmutación está limitada en base al tiempo de retraso de almacenamiento asociado con un transistor que se lleva a saturación. Existe otra familia lógica *bipolar* que evita la saturación del transistor, con lo que se incrementa la velocidad de conmutación en general. A esta familia lógica se le conoce como **lógica de acoplamiento por emisor (ECL)**, la cual opera en base al principio de conmutación de corriente, en el cual una corriente de polarización fija menor que I_C(sat) se cambia del colector de un transistor a otro. Debido a esta operación en modo de corriente, a esta forma lógica también se le conoce como *lógica en modo de corriente* (CML).

Circuito ECL básico

El circuito básico para la lógica de acoplamiento por emisor es en esencia la configuración del amplificador diferencial de la figura 8-41(a). El voltaje de alimentación V_{EE} produce una corriente I_E básicamente fija, la cual permanece alrededor de los 3 mA durante la operación normal. A esta corriente se le permite fluir a través de Q_1 o de Q_2, dependiendo del nivel de voltaje en V_{ENT}. En otras palabras, esta corriente cambia entre el colector de Q_1 y el colector de Q_2 a medida que V_{ENT} cambia entre sus dos niveles lógicos de -1.7 V (0 lógico para ECL) y -0.8 V (1 lógico para ECL). La tabla en la figura 8-41(a) muestra los voltajes de salida resultantes para estas dos condiciones en V_{ENT}. Hay que considerar dos puntos importantes: (1) V_{C1} y V_{C2} son los *complementos* uno del otro, y (2) los niveles de voltaje de salida no son iguales que los niveles lógicos de entrada.

El segundo punto anterior se resuelve con facilidad al conectar a V_{C1} y V_{C2} a etapas de seguidores de emisor (Q_3 y Q_4), como muestra la figura 8-41(b). Los seguidores de emisor realizan dos funciones: (1) restan una cantidad aproximada de 0.8 V de V_{C1} y V_{C2} para desplazar los niveles de salida a los niveles lógicos ECL correctos; y (2) proporcionan una impedancia de salida muy baja (por lo general de 7 Ω), la cual provee una capacidad de carga grande y una carga rápida de la capacitancia de carga. Este circuito produce dos salidas complementarias: V_{SAL1}, que es igual a $\overline{V_{\text{ENT}}}$, y V_{SAL2}, que es igual a V_{ENT}.

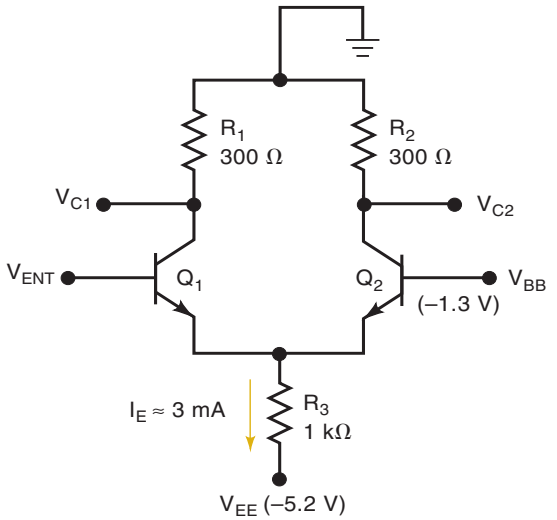
Compuerta OR/NOR ECL

El circuito ECL básico de la figura 8-41(b) puede usarse como un INVERSOR si la salida se toma en V_{SAL1}. Este circuito básico puede expandirse a más de una entrada si se conecta en paralelo el transistor Q_1 con otros transistores para las demás entradas, como en la figura 8-42(a). Aquí, Q_1 o Q_2 pueden provocar que la corriente se quite de Q_2, y como resultado las dos salidas V_{SAL1} y V_{SAL2} serían las operaciones NOR y OR, respectivamente. Esta compuerta OR/NOR se simboliza en la figura 8-42(b) y es la compuerta ECL fundamental.

Características de la familia ECL

La serie ECL más reciente de Motorola se llama *ECLin PS*, que significa ECL en pico segundos. Esta serie lógica presume de un tiempo de propagación máximo en las compuertas de 500 ps (que equivale a la mitad de un nanosegundo) y velocidades de conmutación de los FFs de 1.4 GHz. Algunos dispositivos en esta serie tienen tiempos en las compuertas de sólo 100 ps, a una potencia promedio de 5 mW. A continuación se listan las características más importantes de la serie ECLin PS de la familia de circuitos lógicos MECL de Motorola:

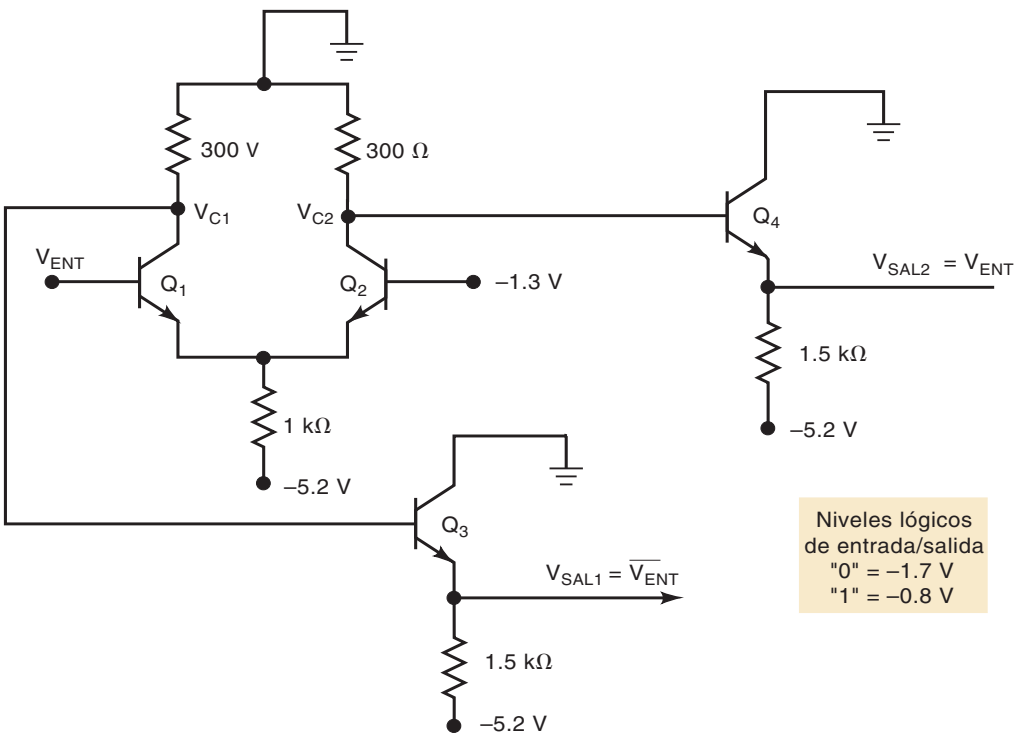
1. Los transistores nunca se saturan, por lo que la velocidad de conmutación es muy alta. El tiempo de propagación típico es de 360 ps, con lo cual ECL es más rápida que cualquiera de los miembros de las familias TTL o CMOS.



(a)

Estados de operación

V_{ENT}	Salidas
–1.7 V (0 lógico)	V_{C1} = 0 V, V_{C2} = –0.9 V } Q_2 conduce
–0.8 V (1 lógico)	V_{C1} = –0.9 V, V_{C2} = 0 V } Q_1 conduce



(b)

FIGURA 8-41 (a) Circuito ECL básico; (b) con la adición de los seguidores de emisor.

2. Los niveles lógicos nominales son −0.8 V y −1.7 V para el 1 y 0 lógicos, respectivamente. La serie ECLin PS es totalmente compatible en voltaje con las series anteriores de ECL.
3. Los márgenes de ruido de ECL para el peor de los casos son de aproximadamente 150 mV. Estos márgenes de ruido tan bajos hacen que ECL sea poco confiable para usarse en ambientes industriales pesados.
4. Por lo general, un bloque lógico ECL produce una salida y su complemento. Esto elimina la necesidad de inversores. El control complementario de alta corriente

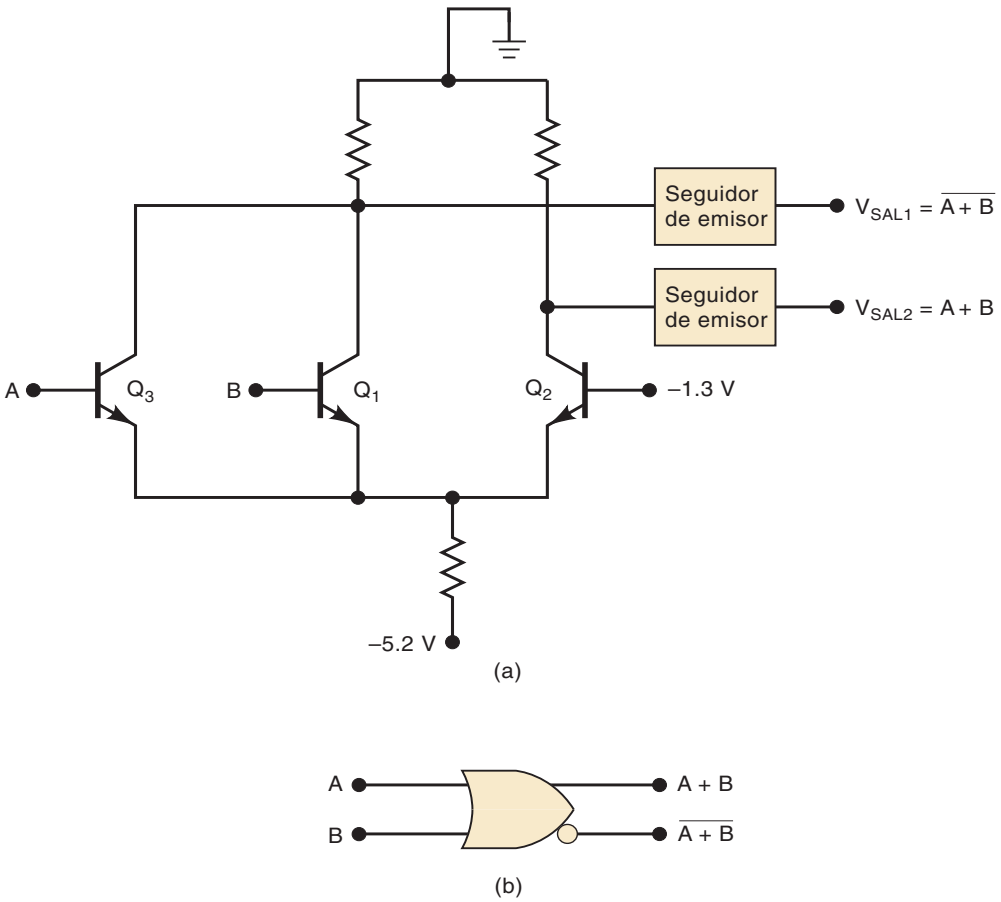


FIGURA 8-42 (a) Circuito NOR/OR ECL; (b) símbolo lógico.

también hace de ECL un excelente reforzador de líneas para conductor de par trenzado (como los conductores telefónicos).

5. Por lo general, la capacidad de carga es un valor cercano a 25, debido a las salidas de seguidor de emisor de baja impedancia.
6. La disipación de potencia típica es de 25 mW, un poco más alta que la de la serie 74AS.
7. El flujo total de corriente en un circuito ECL permanece relativamente constante, sin importar su estado lógico. Esto ayuda a mantener un drenado de corriente sin variaciones en la fuente de energía, incluso durante las transiciones de conmutación. Por ende, no se generarán picos de ruido en forma interna, como los que se producen debido a la conmutación en TTL y CMOS.

La tabla 8-11 muestra cómo se compara ECL con las familias lógicas TTL importantes. La familia ECL de CIs no incluye un amplio intervalo de dispositivos lógicos de propósito general, como las familias TTL y CMOS. ECL tampoco incluye CIs complejos de propósito especial que se utilizan en aplicaciones tales como la transmisión de datos en alta velocidad, las memorias de alta velocidad y las unidades aritméticas de alta velocidad. Los márgenes de ruido relativamente bajos y el alto drenado de potencia de ECL son desventajas, si se le compara con TTL y CMOS. Otra de las desventajas es su voltaje de alimentación negativo y sus niveles lógicos, que no son compatibles con los de las demás familias lógicas. Esto dificulta el uso de dispositivos ECL en conjunto con CIs TTL y/o CMOS; hay que conectar circuitos especiales de desplazamiento de nivel entre los dispositivos ECL y los dispositivos TTL (o CMOS), tanto en la entrada como en la salida.

TABLA 8-11 Comparación de lógicas de alta velocidad.

Familia lógica	t_{pd} (ns)	P_D(mW) 100 kHz	Margen de ruido para el peor caso (mV)	Velocidad máxima de reloj (MHz)
74AS	1.7	8	300	200
74F	3.8	6	300	100
74AHC	3.7	0.006	550	130
74AVC	2	0.006	250	*
74ALVT	2.4	0.33	400	*
74ALB	2.2	1	400	*
ECL	0.3	25	150	1400

*No hay flip-flops disponibles en esta serie.

PREGUNTAS DE REPASO

1. *Verdadero o falso:*
   (a) ECL obtiene su operación en alta velocidad al prevenir la saturación de los transistores.
   (b) Por lo general, los circuitos ECL tienen salidas complementarias.
   (c) Los márgenes de ruido para los circuitos ECL son más grandes que los márgenes de ruido para TTL.
   (d) Los circuitos ECL no generan picos de ruido durante las transiciones de estado.
   (e) Los dispositivos ECL requieren menos potencia que los dispositivos TTL estándar.
   (f) ECL puede utilizarse fácilmente con TTL.

8-15 COMPUERTA DE TRANSMISIÓN CMOS (INTERRUPTOR BILATERAL)

Un circuito CMOS especial que no tiene su contraparte en TTL o en ECL es la **compuerta de transmisión** o **interruptor bilateral**, el cual actúa en esencia como un interruptor de un polo, un tiro y se controla mediante un nivel lógico de entrada. Esta compuerta de transmisión pasa las señales en ambas direcciones y es útil para las aplicaciones analógicas y digitales.

La figura 8-43(a) es el arreglo básico para el interruptor bilateral. Consiste de un P-MOSFET y un N-MOSFET en paralelo, de manera que puedan cambiarse ambas polaridades del voltaje de entrada. La entrada CONTROL y su inverso se utilizan para encender (cerrado) y apagar (abierto) el interruptor. Cuando el CONTROL está en ALTO, ambos MOSFETs están apagados y el interruptor está abierto. En teoría, este circuito opera como un relevador electromecánico. No obstante, en la práctica no es un cortocircuito perfecto cuando el interruptor está cerrado; la resistencia R_{ENC} del interruptor es, por lo general, de 200 Ω. En el estado abierto, la resistencia del interruptor es muy grande, por lo general de 10^{12} Ω, que para la mayoría de los fines se puede considerar como circuito abierto. El símbolo en la figura 8-43(b) se utiliza para representar el interruptor bilateral.

A este circuito se le conoce como interruptor *bilateral* debido a que las terminales de entrada y de salida pueden intercambiarse. Las señales que se aplican a la entrada del interruptor pueden ser digitales o analógicas, siempre y cuando se mantengan dentro de los límites de 0 a V_{DD} volts.

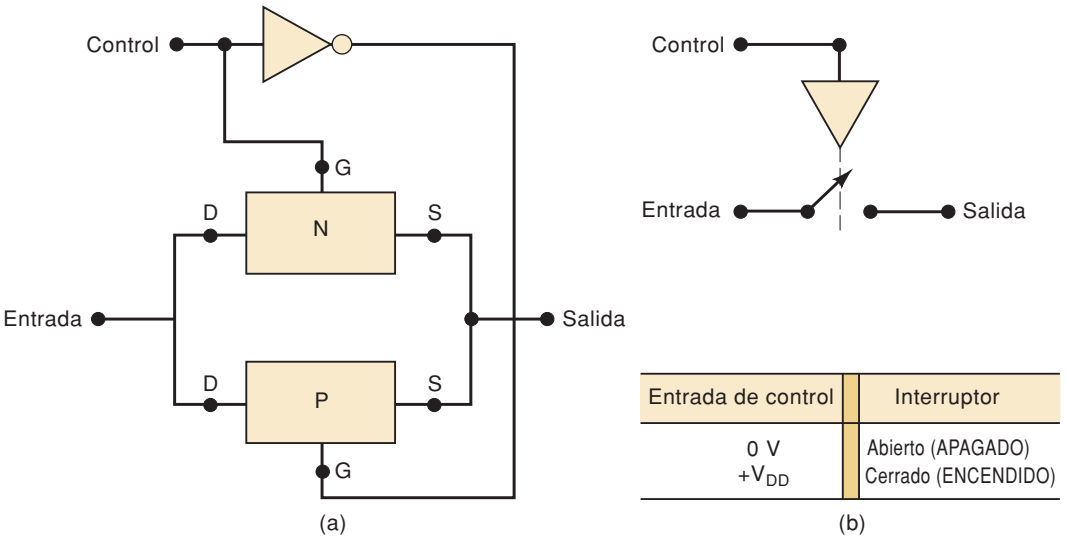
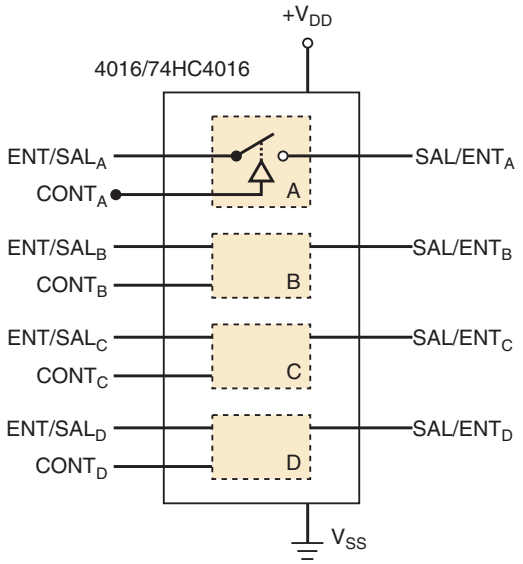


FIGURA 8-43 Interruptor bilateral CMOS (compuerta de transmisión).

La figura 8-44(a) muestra el diagrama lógico tradicional para un CI 4016 el cual contiene cuatro interruptores bilaterales, el cual también está disponible en la serie 74HC como un 74HC4016. Este CI contiene cuatro interruptores bilaterales que operan de la forma antes descrita. Cada interruptor se controla en forma independiente mediante su propia entrada de control. Por ejemplo, el estado ENCENDIDO/APAGADO del interruptor superior se controla mediante la entrada $CONT_A$. Como los interruptores son bidireccionales, cualquiera de las terminales del interruptor puede servir como entrada o como salida, según lo indiquen las etiquetas.

FIGURA 8-44 El interruptor bilateral cuádruple 4016/74HC4016.



EJEMPLO 8-12

Describa la operación del circuito en la figura 8-45.

Solución

Aquí, dos de los interruptores bilaterales están conectados de forma que una señal de entrada analógica común pueda cambiarse a la entrada X o a la entrada Y, dependiendo del estado lógico de la entrada SELECCIÓN DE SALIDA (OUTPUT SELECT). Cuando SELECCIÓN DE SALIDA está en BAJO, el interruptor superior

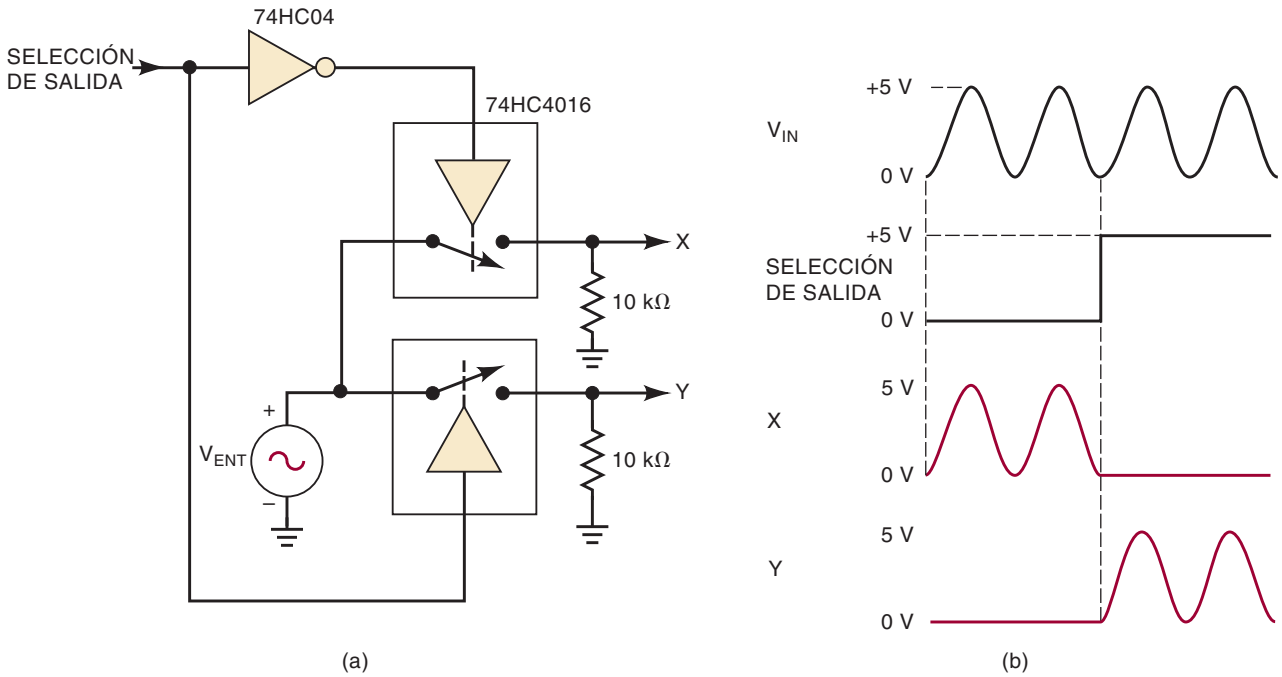


FIGURA 8-45 Ejemplo 8-12: interruptores bilaterales 74HC4016 que se utilizan para conmutar una señal analógica entre dos salidas distintas.

se cierra y el inferior se abre de manera que V_{ENT} se conecte a la salida X. Cuando SELECCIÓN DE SALIDA está en ALTO, el interruptor superior se abre y el inferior se cierra de forma que V_{ENT} se conecte a la salida Y. La figura 8-45(b) muestra algunas formas de onda típicas. Para una operación apropiada, V_{ENT} debe estar dentro del intervalo de 0 V a $+V_{DD}$.

El interruptor bilateral 4016/74HC4016 sólo puede conmutar voltajes de entrada que se encuentren entre 0 V y V_{DD} y, por lo tanto no podría utilizarse para señales que fueran tanto positivas como negativas, con respecto a tierra. El 4316 y el 74HC4316 son interruptores bilaterales cuádruples que pueden conmutar señales analógicas *bipolares*. Estos dispositivos tienen una segunda terminal de alimentación llamada V_{EE}, la cual puede hacerse negativa con respecto a tierra. Esto permite señales de entrada que pueden variar de V_{EE} a V_{DD}. Por ejemplo, si $V_{EE} = -5$ V y $V_{DD} = +5$ V, la señal de entrada analógica puede tener cualquier valor entre −5 V y +5 V.

PREGUNTAS DE REPASO

1. Describa la operación de un interruptor bilateral CMOS.
2. *Verdadero o falso:* no hay un interruptor bilateral TTL.

8-16 INTERFASE DE CIS

Interfase significa conectar la(s) salida(s) de un circuito o sistema a la(s) entrada(s) de otro circuito o sistema que tiene distintas características eléctricas. Es común que no se pueda hacer una conexión directa debido a la diferencia en las características eléctricas del circuito *reforzador* que proporciona la señal de salida y del circuito de *carga* que va a recibir la señal. Un circuito de interfase se conecta entre el reforzador y la carga, como se muestra en la figura 8-46. Su función es tomar la señal de salida del reforzador y acondicionarla de manera que sea compatible con

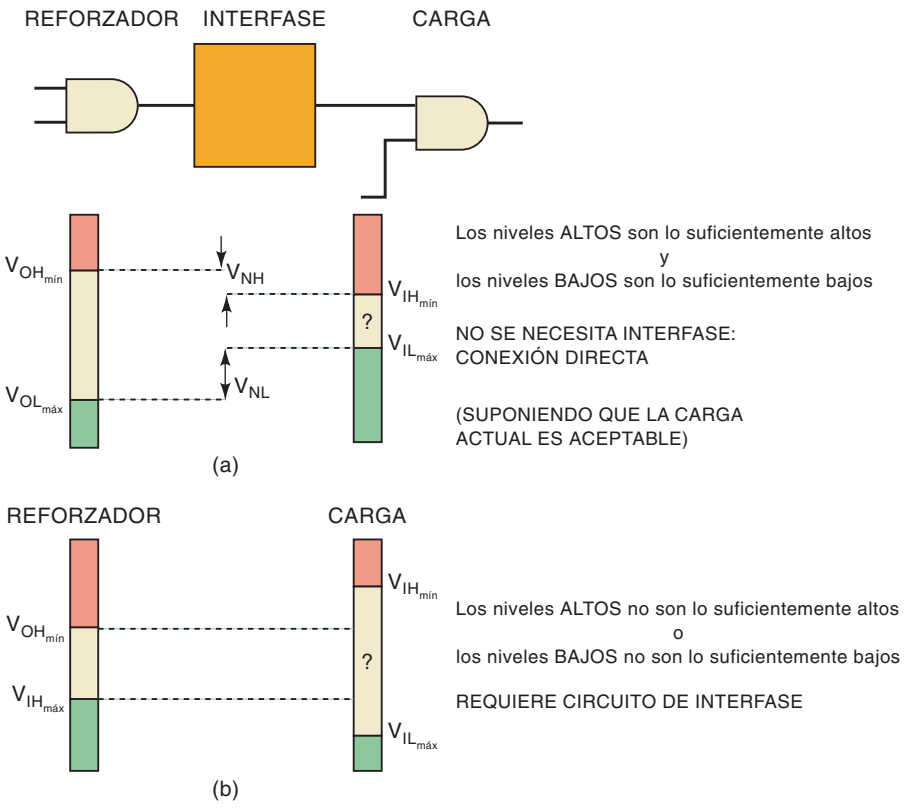


FIGURA 8-46 Interfase de CIs lógicos: (a) no se necesita una interfase; (b) se requiere una interfase.

los requerimientos de la carga. En los sistemas digitales esto es muy sencillo, ya que cada dispositivo sólo puede estar encendido o apagado. La interfase debe asegurar que cuando el reforzador produzca un nivel ALTO de salida, la carga reciba una señal que reconozca como nivel ALTO; *y* que cuando el reforzador produzca un nivel BAJO de salida, la carga reciba una señal que reconozca como nivel BAJO.

El circuito de interfase más simple y conveniente entre un controlador y una carga es una conexión directa. Desde luego que los dispositivos que se encuentran en la misma serie están diseñados para integrarse en forma directa unos con otros. No obstante, en la actualidad muchos sistemas implican el uso de familias mixtas, voltajes mixtos y series mixtas. En estos sistemas el reto es asegurarse que el reforzador sea capaz de activar en forma consistente la carga en los estados BAJO y ALTO.

Para cualquier caso tal como el que se muestra en la figura 8-46(a) en donde el V_{OH} del reforzador es lo *suficientemente mayor* que el V_{OH}(mín) de la carga, y el V_{OL} del reforzador es lo *suficientemente menor* que el V_{IL}(máx) de la carga, no hay necesidad de un circuito de interfase más que una conexión directa. "¿Qué tan *mayor*?", y "¿Qué tan *menor*?", son preguntas relacionadas con la cantidad de ruido que se espera en el sistema. Recuerde que los márgenes de ruido (V_{NH} y V_{NL}) son medidas de esta diferencia entre las características de salida y de entrada. (Consulte de nuevo la figura 8-4.) El mínimo margen de ruido aceptable para cualquier sistema queda a juicio del diseñador del sistema. Siempre que se determina que V_{NH} o V_{NL} son demasiado pequeños (o incluso negativos), entonces se requiere de un circuito de interfase para poder asegurar que el reforzador y la carga puedan trabajar en conjunto. En la figura 8-46(b) se describe esta situación. Para sintetizar esto:

Reforzador	***Carga***
V_{OH} (mín)	$> V_{IH}$(mín) + V_{NH}
V_{OL} (máx) + V_{NL}	$< V_{IL}$(máx)

Otra cosa que debemos tener en cuenta es que, en especial cuando se utilizan familias antiguas, las características de corriente (a diferencia del voltaje) del reforzador y la carga deben concordar también. La I_{OH} del reforzador debe ser capaz de suministrar suficiente corriente para proveer la I_{IH} necesaria de la carga, y la I_{OL} del reforzador debe ser capaz de drenar suficiente corriente como para soportar la I_{IH} de la carga. En la sección 8-5 vimos este tema, cuando hablamos sobre la capacidad de carga. La mayoría de los dispositivos lógicos modernos tienen un control de salida lo suficientemente alto y una corriente de entrada lo suficientemente baja como para que la carga represente un problema frecuente. No obstante, esto es muy importante cuando se utilizan interfases para dispositivos de entrada/salida externos tales como motores, luces o calentadores. Para sintetizar los requerimientos de carga:

Reforzador	*Carga*
I_{OH}(máx)	$> I_{IH}$(total)
I_{OL}(máx)	$> I_{IL}$(total)

La tabla 8-12 lista algunos valores nominales para varias familias lógicas y series de dispositivos digitales. Dentro de cada familia habrá excepciones a esos valores listados, por lo que en la práctica es importante que busque en los valores de las hojas técnicas los CIs específicos con los que vaya a trabajar. Por cuestión de conveniencia utilizaremos estos valores en los ejemplos que siguen.

Interfase de 5 V entre TTL y CMOS

Cuando se integran distintos tipos de CIs, debemos revisar que el dispositivo reforzador pueda cumplir con los requerimientos de corriente y de voltaje del dispositivo de carga. Un análisis de la tabla 8-12 indica que los valores de corriente de entrada para CMOS son en extremo bajos, en comparación con las capacidades de corriente de salida de cualquiera de las series TTL. Por ende, TTL no tiene problemas para cumplir con los requerimientos de corriente de entrada de CMOS.

No obstante, hay un problema cuando comparamos los voltajes de salida de TTL con los requerimientos de voltaje de entrada de CMOS. La tabla 8-9 muestra que el V_{OH}(mín) de cada una de las series TTL es demasiado bajo cuando se le compara con el requerimiento de V_{IH}(mín) de las series 4000B, 74HC y 74AC. Para estas situaciones debemos hacer algo para elevar el voltaje de salida TTL a un nivel aceptable para CMOS.

La solución más común a este problema de interfase se muestra en la figura 8-47, en donde la salida TTL se conecta a +5 V con una resistencia de puesta en alto. La presencia de esta resistencia de puesta en alto hace que la salida TTL se eleve a un valor aproximado a los 5 V en el estado ALTO, con lo cual se proporciona un nivel de voltaje de entrada adecuado para CMOS. La resistencia de puesta en alto no se requiere si el dispositivo CMOS es un 74HCT o un 74ACT, ya que estas series están diseñadas para aceptar salidas TTL en forma directa, como se muestra en la tabla 8-9.

TABLA 8-12 Corrientes de entrada/salida para dispositivos estándar con un voltaje de alimentación de +5 V.

	CMOS				TTL				
Parámetro	**4000B**	**74HC/HCT**	**74AC/ACT**	**74AHC/AHCT**	**74**	**74LS**	**74AS**	**74ALS**	**74F**
I_{IH}(máx)	1 μA	1 μA	1 μA	1 μA	40 A	20 μA	20 μA	20 μA	20 μA
I_{IL}(máx)	1 μA	1 μA	1 μA	1 μA	1.6 mA	0.4 mA	0.5 mA	100 A	0.6 mA
I_{OH}(máx)	0.4 mA	4 mA	24 mA	8 mA	0.4 mA	0.4 mA	2 mA	400 mA	1.0 mA
I_{OL}(máx)	0.4 mA	4 mA	24 mA	8 mA	16 mA	8 mA	20 mA	8 mA	20 mA

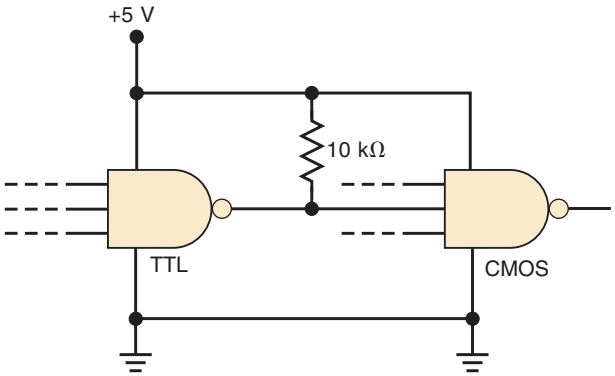


FIGURA 8-47 Se utiliza una resistencia de puesta en alto externa cuando una compuerta TTL maneja a una CMOS.



CMOS MANEJANDO TTL

Antes de considerar el problema de integrar las salidas CMOS con las entradas TTL, será útil revisar las características de salida de CMOS para los dos estados lógicos. La figura 8-48(a) muestra el circuito de salida equivalente en el estado ALTO. La R_{ENC} del P-MOSFET conecta la terminal de salida a V_{DD} (recuerde que el N-MOSFET está apagado). En consecuencia, el circuito de salida CMOS actúa como una fuente de V_{DD} con una resistencia de R_{ENC}. Por lo general el valor de R_{ENC} varía de 100 a 1000 Ω.

La figura 8-48(b) muestra el circuito de salida equivalente en el estado BAJO. La R_{ENC} del N-MOSFET conecta la terminal de salida a tierra (recuerde que el P-MOSFET está apagado). Por ende, la salida CMOS actúa como una resistencia baja a tierra; es decir, actúa como colector de corriente.

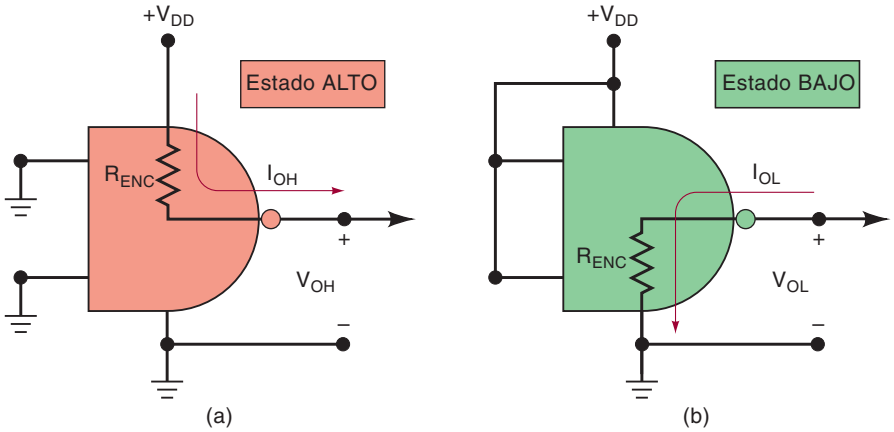


FIGURA 8-48 Circuitos de salida CMOS equivalentes para ambos estados lógicos.

CMOS manejando TTL en el estado ALTO

La tabla 8-9 muestra que las salidas CMOS pueden suministrar con facilidad el suficiente voltaje (V_{OH}) como para satisfacer el requerimiento de entrada de TTL en el estado ALTO (V_{IH}). La tabla 8-12 muestra que las salidas CMOS pueden suministrar más de la suficiente corriente (I_{OH}) como para cumplir con los requerimientos de corriente de entrada de TTL (I_{IH}). Por lo tanto, no se necesita ninguna consideración especial para el estado ALTO.

CMOS manejando TTL en el estado BAJO

La tabla 8-12 muestra que las entradas TTL tienen una corriente de entrada relativamente alta en el estado BAJO, cuyo intervalo es de 100 μA a 2 mA. Las series 74HC y 74HCT pueden drenar hasta 4 mA, por lo que no tendrían problema para manejar una carga TTL *individual* de cualquier serie. No obstante, la serie 4000B es mucho más limitada. Su capacidad de I_{OL} en nivel bajo no es suficiente como para manejar incluso una entrada de las series 74S o 74AS. La serie 74AHC tiene un control de salida comparable a de la serie 74LS.

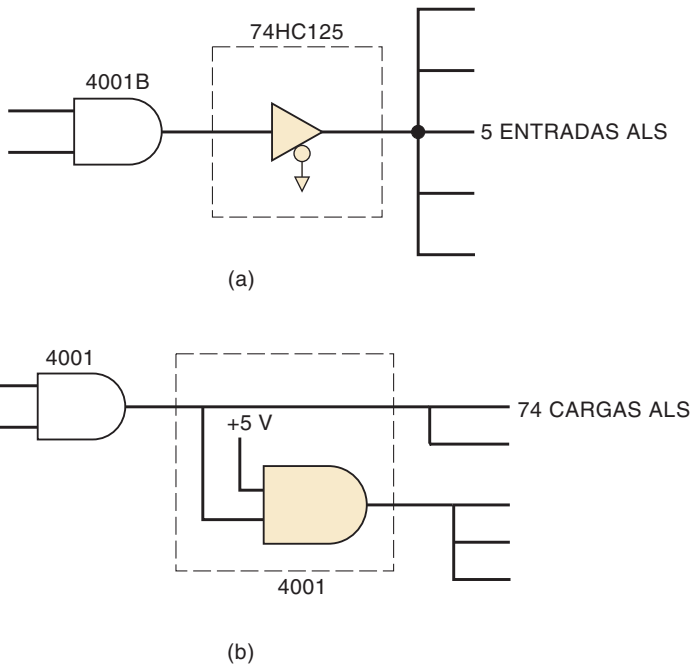


FIGURA 8-49 (a) Uso de un CI de interfase en serie HC. (b) Uso de una compuerta similar par compartir la carga.

Para la situación en la que un reforzador no puede suministrar suficiente corriente para la carga, la solución de interfase es seleccionar un búfer que tenga especificaciones de carga que sean compatibles con el reforzador y la suficiente corriente de control de la salida como para suministrarla a la carga. La figura 8-49(a) muestra un ejemplo de esta situación. La corriente de salida máxima del 4001B no es suficiente como para manejar cinco entradas ALS. Sí puede manejar la entrada 74HC125, que a su vez puede manejar las demás entradas. Otra posible solución se muestra en la figura 8-49(b), en donde la carga se divide entre varias piezas de la serie 4001, de manera tal que ninguna salida necesite manejar más de tres cargas.

EJEMPLO 8-13

La salida de un 74HC maneja tres entradas de un 7406. ¿Es éste un buen diseño?

Solución

¡NO! El 74HC00 puede drenar 4 mA, pero la I_{IL} de la entrada del 7406 es de 1.6 mA. La corriente de carga total cuando está en BAJO es de 1.6 mA $\times$ 3 = 4.8 mA... Demasiada corriente de carga.

EJEMPLO 8-14

La salida de un 4001B maneja tres entradas 74LS. ¿Es éste un circuito bien diseñado?

Solución

¡NO! El 4001B puede drenar 0.4 mA, pero cada entrada 74LS abarca 0.4 mA $\times$ 3 = 1.2 mA... Demasiada corriente de carga.

PREGUNTAS DE REPASO

1. ¿Qué se debe hacer para integrar una salida TTL estándar con una entrada 74AC o 74HC? Suponga que $V_{DD} = +5$ V.
2. ¿Cuál es el problema frecuente cuando CMOS maneja TTL?

8-17 INTERFASE DE VOLTAJE MIXTO

Como vimos en la sección 8-10, muchos dispositivos lógicos nuevos operan con menos de 5 V. En muchas situaciones, estos dispositivos necesitan comunicarse entre sí. En esta sección veremos específicamente cómo integrar dispositivos lógicos que operen con distintos estándares de voltaje.

Salidas de bajo voltaje que controlan cargas de alto voltaje

En ciertas situaciones, el V_{OH} del reforzador es un poco menor que lo que requiere la carga para reconocerlo como un nivel ALTO. Hablamos sobre esta situación cuando vimos cómo integrar salidas TTL con entradas CMOS de 5 V. El único componente de interfase necesario era una resistencia de puesta en alto de 10-kΩ, la cual provoca que el voltaje de salida TTL se impulse por encima del nivel de 3.3 V cuando la salida está en ALTO.

Cuando existe la necesidad de un desplazamiento más sustancial en el voltaje debido a que el reforzador y la carga operan con distintos voltajes de la fuente de energía, se requiere un circuito de interfase conocido como **traductor de nivel de voltaje**. Un ejemplo es un dispositivo CMOS de bajo voltaje (1.8 V) que maneja una entrada CMOS de 5 V. El reforzador puede producir un máximo de sólo 1.8 V como nivel ALTO, y la compuerta de carga requiere 3.33 V para un nivel ALTO. Necesitamos una interfase que pueda aceptar niveles lógicos de 1.8 V y que los traduzca a niveles de CMOS de 5 V. La manera más simple de logar esto es mediante un búfer que tenga un colector abierto, tal como el 74LVC07 que se muestra en la figura 8-50(a). Observe que la resistencia de puesta en alto se conecta a la alimentación de 5 V, mientras que la fuente de energía para el búfer de interfase es de 1.8 V. Otra solución es utilizar un circuito traductor de nivel de voltaje de alimentación dual tal como el 74AVC1T45, como se muestra en la figura 8-50(b). Este dispositivo utiliza dos voltajes distintos de la fuente de poder, uno para las entradas y otro para las salidas, y traduce entre los dos niveles.

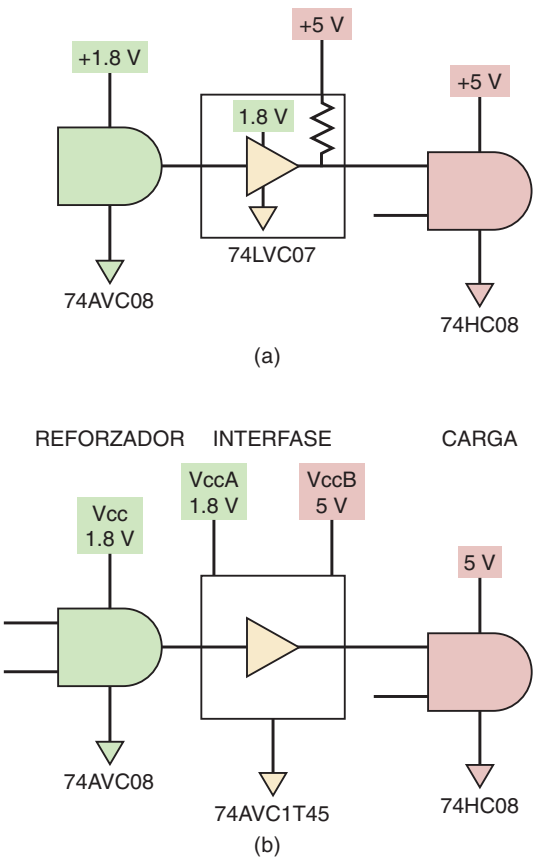


FIGURA 8-50 (a) Uso de un colector abierto con elevación a un alto voltaje. (b) Uso de un traductor de nivel de voltaje.

Salidas de alto voltaje que manejan cargas de bajo voltaje

Cuando los circuitos lógicos que operan con un voltaje mayor deben manejar otros circuitos lógicos que operan con un voltaje menor, a menudo el voltaje de salida del reforzador se excede de los límites seguros que puede manejar la compuerta de carga. En estas situaciones puede utilizarse un traductor de nivel de voltaje de alimentación dual, así como en la figura 8-50(b). Otra solución común a este problema es usar como interfase un búfer de una serie que pueda soportar el voltaje de entrada más alto. La figura 8-51 demuestra esto con una pieza CMOS de 5 V que maneja una entrada de 1.8 V de la serie AUC. El máximo voltaje que puede manejar la entrada AUC (compuerta de carga) es de 3.6 V. No obstante, un 74LVC07A puede manejar hasta 5.5 V en su entrada sin dañarla, aún y cuando opera con 1.8 V. La figura 8-51 muestra cómo podemos usar la tolerancia más alta de voltaje del 74LVC07A para traducir un nivel lógico de 5 V en un nivel lógico de 1.8 V.

En este momento tal vez se esté preguntando, "¿Por qué querría alguien utilizar tal variedad de piezas incompatibles?" La respuesta se da si consideramos sistemas más grandes y tratamos de balancear el rendimiento y el costo. Por ejemplo, en un sistema computacional tal vez haya un CPU de 2.5 V, un módulo de memoria de 3.3 V y un controlador de disco duro de 5 V, todos trabajando en la misma tarjeta madre. Los componentes de bajo voltaje pueden ser necesarios para obtener el rendimiento deseado, pero el disco duro de 5 V puede ser el menos costoso o el único tipo disponible. Los dispositivos reforzador y de carga tal vez no sean compuertas lógicas estándar, pero pueden ser un componente VLSI en nuestro sistema. Si nos basamos en la hoja técnica para esos dispositivos, debemos buscar las características de salida e integrarlos mediante el uso de las técnicas que hemos mostrado. Como los estándares lógicos siguen evolucionando, es importante que podamos hacer funcionar los sistemas utilizando cualquiera de los diversos componentes que estén a nuestra disposición.

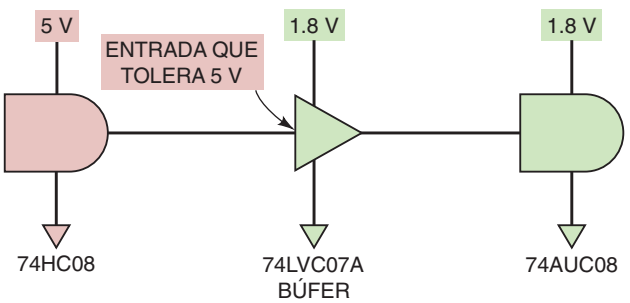


FIGURA 8-51 (a) Una serie de bajo voltaje con entradas que toleran 5 V como interfase.

PREGUNTAS DE REPASO

1. ¿Cuál es la función de un circuito de *interfase*?
2. *Verdadero o falso:* todas las salidas CMOS pueden manejar entradas TTL en el estado ALTO.
3. *Verdadero o falso:* Cualquier salida CMOS puede manejar cualquier entrada TTL individual.
4. ¿Cuál de las series CMOS puede manejar la familia TTL sin necesidad de una resistencia de puesta en alto?
5. ¿Cuántas entradas 7400 pueden manejarse mediante una salida 74HCT00?

8-18 COMPARADORES DE VOLTAJE ANALÓGICO

Otro dispositivo muy útil como interfase para los sistemas digitales es el **comparador de voltaje analógico**. En especial, es útil en sistemas que contienen tanto voltajes analógicos como componentes digitales. Un comparador de voltaje analógico

compara dos voltajes. Si el voltaje en la entrada (+) es mayor que el voltaje en la entrada (−), la salida es un nivel ALTO. Si el voltaje en la entrada (−) es mayor que el voltaje en la entrada (+), la salida es un nivel BAJO. Podemos considerar las entradas de un comparador como entradas analógicas, pero la salida es digital debido a que siempre estará en ALTO o en BAJO. Por esta razón, a este comparador se le conoce comúnmente como convertidor analógico/digital (A/D) de un bit. En el capítulo 10 examinaremos con detalle los convertidores A/D.

Un LM339 es un CI lineal analógico que contiene cuatro comparadores de voltaje. La salida de cada comparador es un transistor de colector abierto, justo igual que una salida TTL de colector abierto. El V_{CC} puede variar de 2 a 36 V, pero por lo general se establece en un valor mayor que los voltajes de entrada analógicos que se van a comparar. Hay que conectar una resistencia de puesta en alto de la salida a la misma fuente que utilizan los circuitos digitales (por lo general de 5 V).

EJEMPLO 8-15

Suponga que una incubadora debe tener una alarma de emergencia para avisar si la temperatura se excede de un nivel peligroso. El dispositivo para medir la temperatura es un LM34 que produce un voltaje directamente proporcional a la temperatura. El voltaje de salida aumenta 10 mV por cada grado F. La alarma del sistema digital debe sonar cuando la temperatura se exceda de 100 °F. Diseñe un circuito que actúe como interfase entre el sensor de temperatura y el circuito digital.

Solución

Necesitamos comparar el voltaje del sensor con un voltaje de referencia fijo. Primero debemos calcular el voltaje de referencia apropiado. Queremos que la salida del comparador cambie a ALTO cuando la temperatura se exceda de 100 °F. El voltaje de salida del LM34 a 100 °F será

$$100\ °\text{F} \cdot 10\ \text{mV/°F} = 1.0\ \text{V}$$

Esto significa que debemos establecer la terminal de entrada (−) del comparador en 1.0 V y debemos conectar el LM34 a la entrada (+). Para poder crear un voltaje de referencia de 1.0 V podemos usar un circuito divisor de voltaje y debemos seleccionar una corriente de polarización de 100 μA. La corriente de entrada del LM339 será en cierto modo despreciable, ya que será menor de 1 μA. Esto significa que $R_1 + R_2$ debe ser igual a 10 kΩ. En este ejemplo podemos operar todo desde una

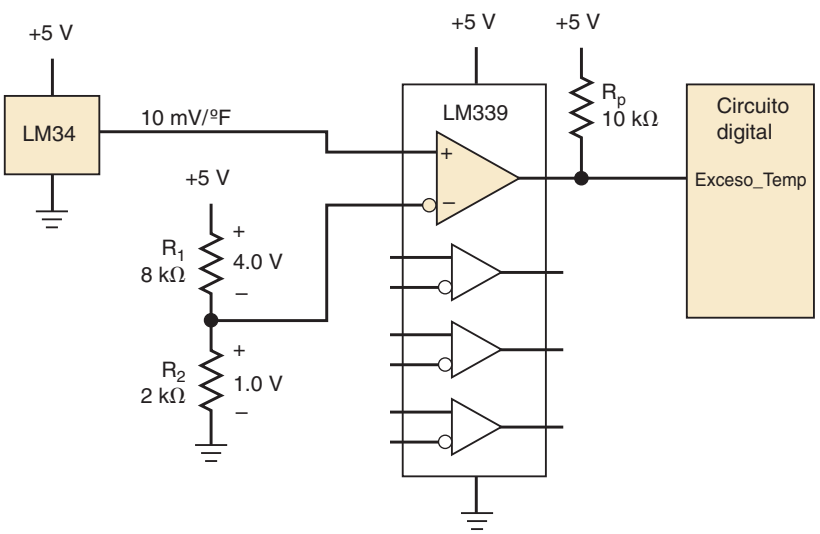


FIGURA 8-52 Un detector de límite de temperatura que utiliza un comparador de voltaje analógico LM339.

fuente de energía de +5 V. La figura 8-52 muestra el circuito completo. Los cálculos se muestran a continuación:

$$V_{R2} = V_{CC} \cdot \frac{R_2}{R_1 + R_2}$$
$$R_2 = V_{R2} \cdot (R_1 + R_2)/V_{CC}$$
$$= 1.0\ \text{V}(10\ \text{k}\Omega)/5\ \text{V} = 2\ \text{k}\Omega$$
$$R_1 = 10\ \text{k}\Omega - R_2 = 10\ \text{k}\Omega - 2\ \text{k}\Omega = 8\ \text{k}\Omega$$

PREGUNTAS DE REPASO

1. ¿Qué hace que la salida de un comparador cambie al estado lógico ALTO?
2. ¿Qué hace que la salida de un comparador cambie al estado lógico BAJO?
3. ¿Es la salida de un LM339 más similar a una salida TTL en forma de tótem o a una salida de colector abierto?

8-19 DIAGNÓSTICO DE FALLAS

Un **pulsador lógico** es una herramienta de prueba y diagnóstico de fallas que genera un pulso de corta duración cuando se opera en forma manual, por lo general al oprimir un botón. El pulsador lógico que se muestra en la figura 8-53 tiene una punta en forma de aguja, la cual toca el nodo del circuito en el que se va a emitir el pulso. El pulsador lógico está diseñado de forma que detecte el nivel de voltaje existente en el nodo y produzca un pulso de voltaje en sentido opuesto. En otras palabras, si el nodo está en BAJO el pulsador lógico produce un pulso estrecho de tendencia positiva; si el nodo está en ALTO, el pulsador lógico produce un pulso estrecho de tendencia negativa.

El pulsador lógico se utiliza para cambiar por unos momentos el nivel lógico en el nodo de un circuito, aún y cuando la salida de otro circuito pueda estar conectada al mismo nodo. En la figura 8-53 el pulsador lógico está en contacto con el nodo X, el cual también está conectado a la salida de la compuerta NAND. El pulsador lógico tiene una impedancia de salida muy baja (por lo general, de 2 Ω o menor) para que pueda sobrepasar la salida de la compuerta NAND y para que pueda cambiar el voltaje en el nodo. No obstante, el pulsador lógico no puede producir un pulso de voltaje en un nodo que esté en corto directo con tierra o V_{CC} (por ejemplo, a través de un puente de soldadura).

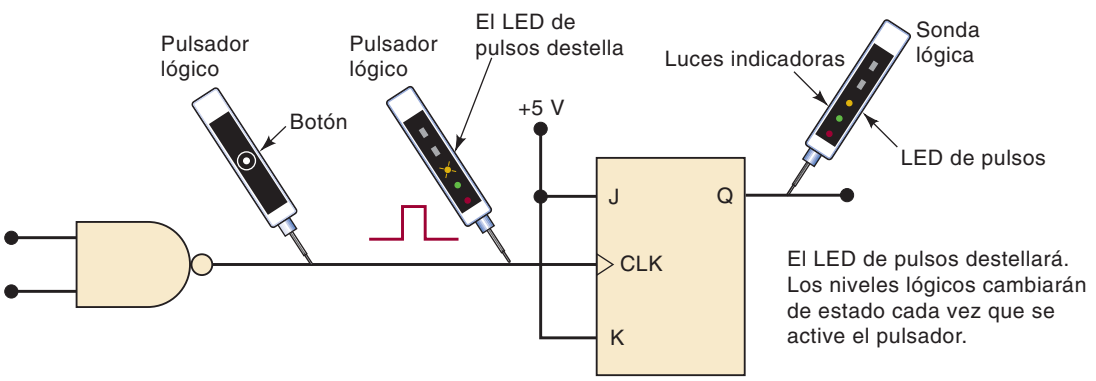


FIGURA 8-53 Un pulsador lógico puede inyectar un pulso en cualquier nodo que no esté en corto directo con tierra o V_{CC}.

Uso de un pulsador lógico y una sonda para probar un circuito

Se puede utilizar un pulsador lógico para inyectar en forma manual un pulso o una serie de pulsos en un circuito, para poder probar la respuesta del mismo. Casi siempre se utiliza una sonda lógica para monitorear la forma en que el circuito responde al pulsador lógico. En la figura 8.53 se está probando la operación de conmutación del flip-flop J-K; para ello se aplican pulsos con el pulsador lógico y se monitorea *Q* con la sonda lógica. Esta combinación de pulsador lógico/sonda lógica es muy útil para comprobar la operación de un dispositivo lógico mientras se conecta a un circuito. Observe que el pulsador lógico se aplica al nodo del circuito *sin* desconectar la salida de la compuerta NAND que controla a ese nodo. Cuando la sonda se coloca en el mismo nodo que el pulsador, los indicadores de nivel lógico parecen estar sin cambios (nivel BAJO en este ejemplo), pero el indicador de pulso amarillo destella una vez cada que se oprime el botón del pulsador. Cuando la sonda se coloca en la salida *Q*, el LED de pulsos destella una vez (con lo cual indica una transición) y los indicadores de nivel lógico cambian de estado cada vez que se oprime el botón del pulsador.

Búsqueda de nodos en corto

El pulsador lógico y la sonda lógica pueden utilizarse para revisar si hay nodos en corto directo a tierra o V_{CC}, como se muestra en la figura 8-54. Cuando usted toca el mismo nodo con un pulsador lógico y una sonda lógica, y después oprime el botón del pulsador lógico, la sonda lógica debe indicar la ocurrencia de un pulso en el nodo. Si la sonda indica un nivel BAJO constante y el LED de pulsos no destella, entonces el nodo está en corto con tierra, como se muestra en la figura 8-54(a). Si la sonda indica un nivel ALTO constante y el LED de pulsos no destella, entonces el nodo está en corto con V_{CC}, como se muestra en la figura 8-54(b).

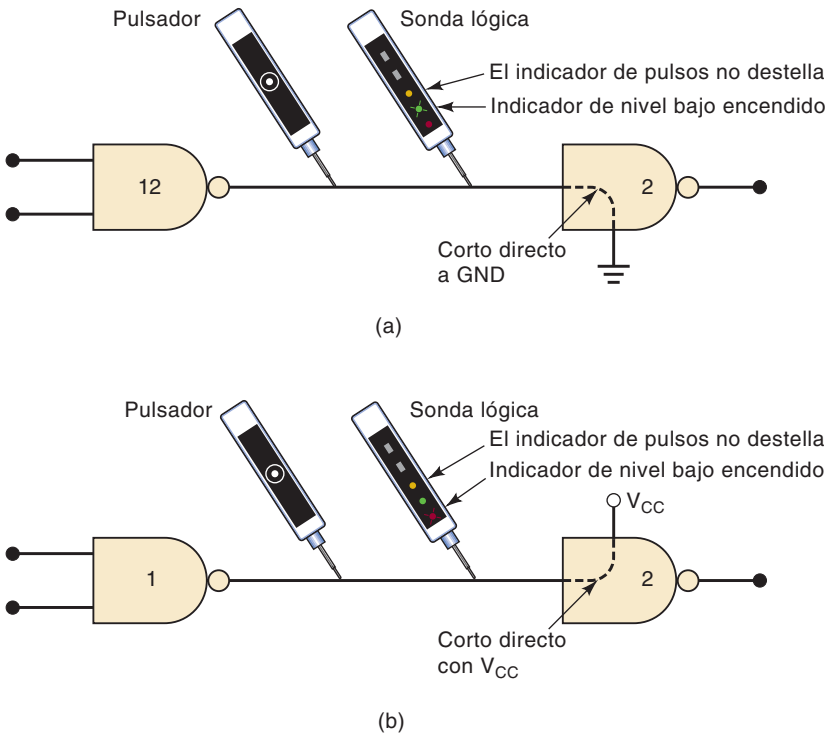


FIGURA 8-54 Un pulsador lógico y una sonda lógica pueden utilizarse para rastrear nodos en corto.

PREGUNTAS DE REPASO

1. ¿Cuál es la función de un pulsador lógico?
2. *Verdadero o falso:* un pulsador lógico producirá un pulso de voltaje en cualquier nodo.
3. *Verdadero o falso:* un pulsador lógico puede forzar a un nodo al nivel BAJO o ALTO durante periodos extendidos de tiempo.
4. ¿Cómo responde una sonda lógica al pulsador lógico?

RESUMEN

1. Todos los dispositivos lógicos digitales son similares en naturaleza, pero muy distintos en relación con los detalles sobre sus características. Es importante comprender los términos que se utilizan para describir estas características, ya que nos permitirá comparar y contrastar el rendimiento de los dispositivos. Al comprender las capacidades y limitaciones de cada tipo de dispositivo podremos combinar dispositivos en forma inteligente para aprovechar los puntos fuertes de cada dispositivo a la hora de construir sistemas digitales confiables.
2. La familia TTL de dispositivos lógicos ha estado en uso durante más de 30 años. Los circuitos utilizan transistores bipolares. Esta familia ofrece muchas compuertas lógicas SSI y dispositivos MSI. Se han desarrollado numerosas series de dispositivos con numeraciones similares, ya que los avances en la tecnología han ofrecido características mejoradas.
3. Cuando se conectan dispositivos en conjunto, es vital saber cuántas entradas puede manejar una salida dada sin comprometer la confiabilidad. A esto se le conoce como *capacidad de carga*.
4. Las salidas de colector abierto y drenador abierto pueden conectarse entre sí para implementar una función AND alambrada. Las salidas triestado pueden conectarse en conjunto para permitir que muchos dispositivos compartan una ruta común de datos, a la cual se le conoce como *bus*. En tal caso, sólo se permite que un dispositivo asegure un nivel lógico en el bus (es decir, que controle el bus) en un momento dado.
5. Los dispositivos lógicos *más rápidos* son de una familia que utiliza lógica de acoplamiento por emisor (ECL). Esta tecnología utiliza también transistores bipolares, pero es tan popular como TTL debido a la inconveniencia de sus características de entrada/salida.
6. Los traductores MOSFET también pueden utilizarse para implementar funciones lógicas. La principal ventaja de la lógica MOS es una menor potencia y una mayor densidad de encapsulado.
7. El uso de MOSFETs complementarios produjo familias lógicas CMOS. La tecnología CMOS capturó el mercado debido a su muy bajo consumo de potencia y a su velocidad competitiva.
8. La necesidad continua de reducir el consumo de potencia y el tamaño ha originado varias series nuevas de dispositivos que operan con 3.3 V y 2.5 V.
9. Los dispositivos lógicos que utilizan varias tecnologías no siempre pueden conectarse directamente en conjunto y operar en forma confiable. Hay que tener en consideración las características de voltaje y de corriente de las entradas y las salidas; además se deben tomar precauciones para asegurar una operación apropiada.
10. La tecnología CMOS permite que un sistema digital controle interruptores analógicos conocidos también como *compuertas de transmisión*. Estos dispositivos pueden pasar o bloquear una señal analógica, dependiendo del nivel lógico digital que la controla.
11. Los comparadores de voltajes analógicos ofrecen otro puente entre las señales analógicas y los sistemas digitales. Estos dispositivos comparan los voltajes analógicos y producen como salida un nivel lógico digital, con base en cuál voltaje es mayor. Permiten que un sistema analógico controle un sistema digital.

TÉRMINOS IMPORTANTES

AND alambrada
búfer/reforzador
capacidad de carga
CMOS
colisión de bus
comparador de voltaje analógico
compuerta de transmisión (interruptor bilateral)
desacoplamiento de fuente de poder
descarga electrostática (ESD)
DIP
distancia entre terminales
drenado de corriente
efecto de cierre
entradas flotantes
forma de tótem
inmunidad al ruido
interfase
lógica de acoplamiento por emisor (ECL)
margen de ruido
MOSFETs
N-MOS
P-MOS
pulsador lógico
salida de colector abierto
señalización diferencial de bajo voltaje (LVDS)
suministro de corriente
tecnología de montaje superficial
traductor de nivel de voltaje
transistor de drenado de corriente (transistor de puesta en bajo)
transistor de suministro de corriente (transistor de puesta en alto)
triestado
TTL

PROBLEMAS

SECCIONES 8-1 A 8-3

B 8.1.* Dos circuitos lógicos distintos tienen las características que se muestran en la tabla 8-13.

(a) ¿Cuál circuito tiene la mejor inmunidad al ruido de corriente directa en estado BAJO? ¿La mejor inmunidad al ruido de corriente directa en estado ALTO?

(b) ¿Cuál circuito puede operar a mayores frecuencias?

(c) ¿Cuál circuito consume la mayor cantidad de corriente de alimentación?

TABLA 8-13

	Circuito *A*	Circuito *B*
V_{fuente} (V)	6	5
V_{IH}(mín) (V)	1.6	1.8
V_{IL}(máx) (V)	0.9	0.7
V_{OH}(mín) (V)	2.2	2.5
V_{OL}(máx) (V)	0.4	0.3
t_{PLH} (ns)	10	18
t_{PHL} (ns)	8	14
P_D (mW)	16	10

B 8-2. Busque en las hojas técnicas de CIs y utilice los valores *máximos* para determinar P_D (prom) y t_{pd} (prom) para una compuerta en cada uno de los siguientes CIs TTL. (Vea el ejemplo 8-2 en la sección 8-3.)

(a)* 7432

(b)* 74S32

(c) 74LS20

(d) 74ALS20

(e) 74AS20

* Encontrará las respuestas a los problemas marcados con un asterisco al final del capítulo.

8-3. Cierta familia lógica tiene los siguientes parámetros de voltaje:

V_{IH}(mín) = 3.5 V V_{IL}(máx) = 1.0 V
V_{OH}(mín) = 4.9 V V_{OL}(máx) = 0.1 V

(a)* ¿Cuál es el pico de ruido con tendencia positiva más grande que puede tolerarse?

(b) ¿Cuál es el pico de ruido con tendencia negativa más grande que puede tolerarse?

PREGUNTA DE PRÁCTICA

B 8-4.* Para cada enunciado, indique el término o parámetro que se está describiendo:

(a) La corriente en una entrada, cuando se aplica un 1 lógico a esa entrada.

(b) La corriente que se consume de la fuente V_{CC} cuando todas las entradas están en BAJO.

(c) El tiempo requerido para que una salida cambie del estado 1 al estado 0.

(d) El tamaño del pico de voltaje que puede tolerarse en una entrada en ALTO sin ocasionar una operación indeterminada.

(e) Un encapsulado de CI que no requiere la perforación de hoyos en el tablero de circuito impreso.

(f) Cuando una salida en BAJO recibe corriente de la entrada del circuito que maneja.

(g) El número de entradas distintas que puede manejar una salida en forma segura.

(h) El arreglo de los transistores de salida en un circuito TTL estándar.

(i) Otro término que describe el transistor de puesta en alto Q_4.

(j) El intervalo de valores de V_{CC} permitidos para TTL.

(k) V_{OH} (mín) y V_{IH} (mín) para la serie 74ALS.

(l) V_{IL} (máx) y V_{OL} (máx) para la serie 74ALS.

(m) Cuando una salida en ALTO suministra corriente a una carga.

SECCIÓN 8-4

8-5.* (a) De la tabla 8-6, determine los márgenes de ruido cuando un dispositivo 74LS maneja una entrada 74ALS.

(b) Repita la parte (a) para un 74ALS que maneja a un 74LS.

(c) ¿Cuál será el margen de ruido total de un circuito lógico que utiliza circuitos 74LS y 74ALS en combinación?

(d) Cierto circuito lógico tiene un V_{IL} (máx) = 450 mV. ¿Cuál de las series TTL puede utilizarse con este circuito?

SECCIONES 8-5 Y 8-6

B 8.6 **PREGUNTA DE PRÁCTICA**

(a) Defina el concepto de *capacidad de carga*.

(b)* ¿En qué tipo de compuertas las entradas conectadas entre sí siempre cuentan como una sola carga de entrada en el estado BAJO?

(c)* Defina el concepto de entradas “flotantes”.

(d) ¿Qué es lo que produce picos de corriente en TTL? ¿Qué efecto indeseable se puede producir? ¿Qué puede hacerse para reducir este efecto?

(e) Cuando una salida TTL maneja una entrada TTL, ¿de dónde proviene I_{OL}? ¿A dónde va I_{OH}?

8-7. Utilice la tabla 8-12 para encontrar la capacidad de carga para crear una interfase que permita que la primera familia lógica controle a la segunda.

(a)* 74AS a 74AS

(b) +74F a 74F

(c) 74AHC a 74AS

(d) 74HC a 74ALS

B 8-8. Consulte la hoja técnica para el flip-flop J-K 74LS112.

(a)* Determine la corriente de carga en ALTO y en BAJO en las entradas *J* y *K*.

(b) Determine la corriente de carga en ALTO y en BAJO en las entradas reloj (CLK) y borrar (CLR).

(c) ¿Cuántas entradas de reloj 74LS112 puede manejar la salida de un 74LS112?

B 8-9.* La figura 8-55(a) muestra un flip-flop J-K 74LS112 cuya salida debe manejar un total de ocho entradas TTL estándar. Como esto se excede la capacidad de carga del 74LS112, se necesita algún tipo de búfer. La figura 8-55(b) muestra una posibilidad mediante el uso de una de las compuertas NAND del búfer NAND cuádruple 74LS37, el cual tiene una capacidad de carga mucho más alto que el 74LS112. Observe que se utiliza $\overline{Q}$ debido a que la compuerta NAND actúa como un INVERSOR. Consulte la hoja técnica para el 74LS37.

(a) Determine su máximo capacidad de carga para TTL estándar.

(b) Determine su máxima corriente de colector en el estado BAJO.

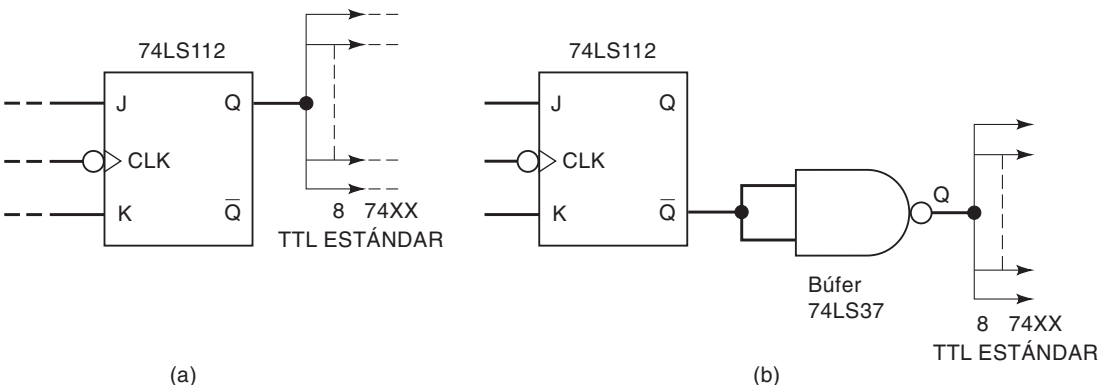


FIGURA 8-55 Problemas 8-9 y 8-10.

D 8-10. Por lo general, las compuertas de los búferes son más costosas que las compuertas ordinarias, y algunas veces hay compuertas ordinarias sin usar disponibles, las cuales pueden emplearse para resolver un problema de carga, tal como el de la figura 8-55(a). Muestre cómo pueden utilizarse compuertas NAND 74LS00 para resolver este problema.

B 8-11.* Consulte el diagrama lógico de la figura 8-56, en donde la salida del OR exclusivo 74LS86 maneja varias entradas 74LS20. Determine si se está excediendo la capacidad de carga del 74LS86 y explique por qué. Repita el procedimiento utilizando dispositivos 74AS. Use la tabla 8-7.

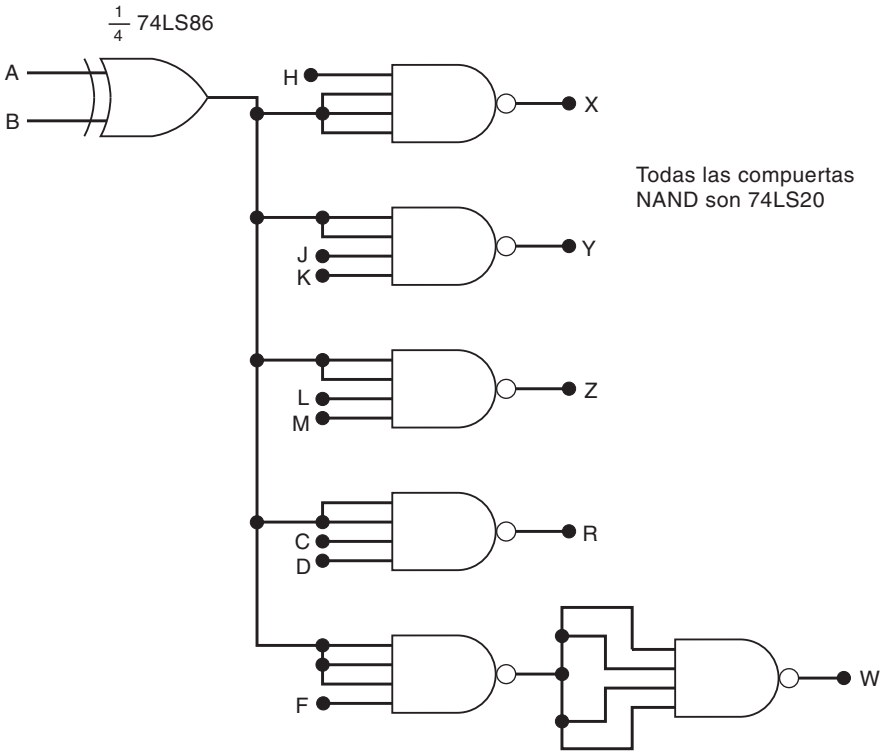


FIGURA 8-56 Problemas 8-11 y 8-13.

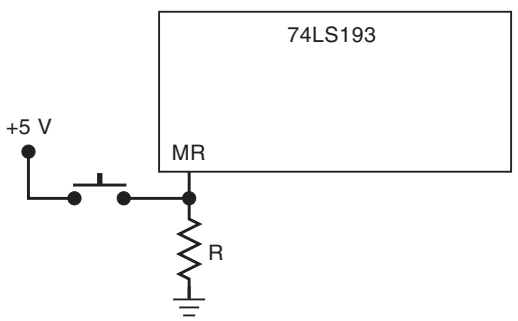
B 8-12. ¿Cuánto tiempo tarda la salida de un 74LS04 común en cambiar de estado, en respuesta a una transición de tendencia positiva en su entrada?

A 8.13.* Para el circuito de la figura 8-56, determine el tiempo más largo que tardará en reflejarse en la salida *W* un cambio en la entrada *A*. Use todas las condiciones del peor caso y los valores máximos de los tiempos de propagación en las compuertas. (*Sugerencia:* recuerde que las compuertas NAND son compuertas inversoras.) Repita el procedimiento utilizando sólo dispositivos 74ALS.

8-14.* (a) La figura 8-57 muestra un contador 74LS193 con su entrada RESET maestra activa en ALTO, la cual se activa mediante un interruptor de botón. La resistencia *R* se utiliza para retener a *MR* en BAJO mientras que el interruptor está abierto. ¿Cuál es el valor máximo que puede usarse para *R*?

(b) Repita la parte (a) para el 74ALS193.

FIGURA 8-57
Problema 8-14.



A, F 8-15. La figura 8-58(a) muestra un circuito que se utiliza para convertir una onda sinodal de 60-Hz en una señal de 60-pps, la cual puede disparar FFs y contadores en forma confiable. Este tipo de circuito podría utilizarse en un reloj digital.

(a) Explique la operación del circuito.

(b)* Un técnico está probando este circuito y observa que la salida del 74LS14 permanece en BAJO. Revisa la forma de onda en la entrada del INVERSOR y aparece como se muestra en la figura 8-58(b). Creyendo que el INVERSOR tiene falla, reemplaza el chip y observa los mismos resultados. ¿Qué cree usted que esté ocasionando el problema y cómo puede resolverse? (*Sugerencia:* examine la forma de onda v_x con cuidado.)

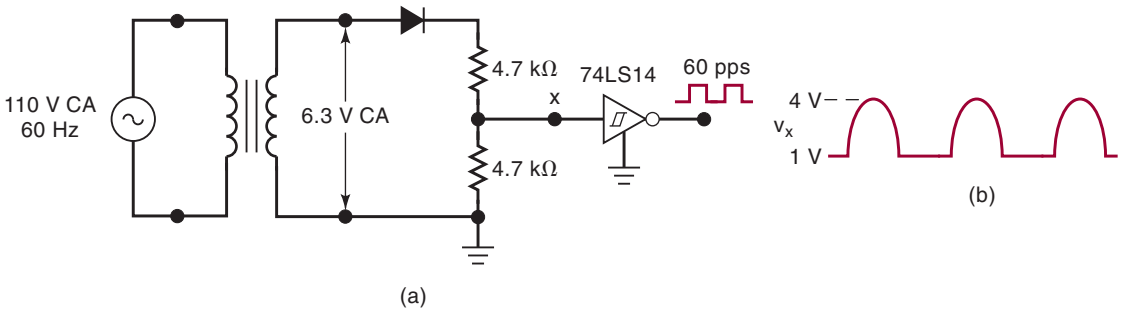


FIGURA 8-58 Problema 8-15.

F 8-16. Para cada una de las formas de onda en la figura 8-59, determine *por qué no* disparará en forma confiable un flip-flop 74LS112 en su entrada *CLK*.

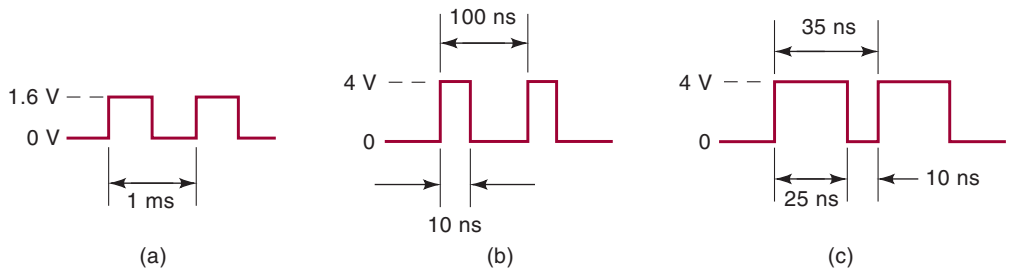


FIGURA 8-59 Problema 8-16.

F 8-17. Un técnico implementa un circuito en un tablero experimental para fines de prueba. Al probar la operación del circuito, descubre que muchos de los FFs y contadores se disparan en forma errática. Como todo buen técnico, revisa la línea V_{CC} con un multímetro y obtiene una lectura de 4.97 V, que es aceptable para TTL. Después revisa todo el alambrado del circuito y reemplaza cada CI uno por uno, pero el problema persiste. Por último decide observar V_{CC} en el osciloscopio y obtiene la forma de onda que se muestra en la figura 8-60. ¿Cuál es la probable causa del ruido en V_{CC}? ¿Qué olvido incluir el técnico cuando implementó el circuito en el tablero experimental?

FIGURA 8-60
Problema 8-17.



SECCIONES 8-7 A 8-10

B 8-18. ¿Qué tipo de MOSFET se enciende al aplicar

(a) 5 V en la compuerta y 0 V en la fuente?

(b) 0 V en la compuerta y 5 V en la fuente?

B 8-19.* ¿Cuáles de las siguientes son ventajas que, por lo general, CMOS tiene en comparación con TTL?

(a) Mayor densidad del encapsulado.

(b) Mayor velocidad.

(c) Mayor capacidad de carga.

(d) Menor impedancia de salida.

(e) Proceso de fabricación más simple.

(f) Es más apropiado para la LSI.

(g) Menor P_D (por debajo de 1 MHz).

(h) Los transistores son el único elemento del circuito.

(i) Menor capacitancia de entrada.

(j) Menos sensible a la ESD.

8-20. ¿Cuáles de las siguientes condiciones de operación tienen probabilidad de producir el valor promedio más bajo para P_D en un sistema lógico CMOS? Explique.

(a) $V_{DD} = 5$ V, frecuencia de conmutación $f_{máx} = 1$ MHz.

(b) $V_{DD} = 5$ V, $f_{máx} = 10$ kHz.

(c) $V_{DD} = 10$ V, $f_{máx} = 10$ kHz.

A 8-21.* La salida de cada INVERSOR en un CI 74LS04 maneja dos entradas 74HCT08. La entrada a cada INVERSOR está en BAJO durante un 99% del tiempo. ¿Cuál es la potencia máxima que disipa el chip 74LS04?

8-22. Utilice los valores de la tabla 8-9 para calcular el margen de ruido en estado ALTO cuando una compuerta 74HC maneja una entrada 74LS estándar.

8-23. ¿Qué es lo que puede producir el efecto de cierre en un CI CMOS? ¿Qué podría ocurrir en esa condición? ¿Qué precauciones hay que tomar para evitar el efecto de cierre?

8-24. Consulte la hoja técnica para el CI de compuerta NAND 74HC20. Use los valores máximos para calcular P_D(prom) y t_{pd}(prom). Compare los resultados con los valores calculados en el problema 8-2 para TTL.

SECCIONES 8-11 Y 8-12

B 8-25. **PREGUNTA DE PRÁCTICA**

(a) Defina el concepto de AND alambrado.

(b) ¿Qué es una resistencia de puesta en alto? ¿Por qué se utiliza?

(c) ¿Qué tipos de salidas TTL pueden conectarse juntas en forma segura?

(d) ¿Qué es la colisión de bus?

D 8-26. El 74LS09 es un CI TTL con cuatro compuertas AND de dos entradas con salidas de colector abierto. Muestre cómo pueden usarse chips 74LS09 para implementar la operación $x = A \cdot B \cdot C \cdot D \cdot E \cdot F \cdot G \cdot H \cdot I \cdot J \cdot K \cdot M$.

B 8-27.* Determine la expresión lógica para la salida X en la figura 8-61.

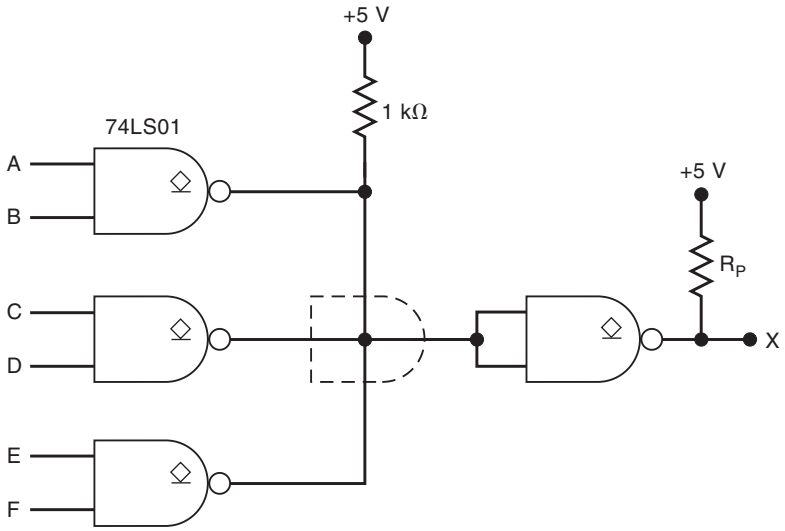


FIGURA 8-61
Problema 8-27.

A 8-28. ¿Cuál de las siguientes acciones tiene la mayor probabilidad de destruir una salida TTL en forma de tótem, mientras trata de cambiar de ALTO a BAJO?

(a) Conectar la salida a +5 V.

(b) Conectar la salida a tierra.

(c) Aplicar una entrada de 7 V.

(d) Conectar la salida a otra salida TTL en forma de tótem.

D 8-29.* La figura 8-62(a) muestra un búfer inversor 7406 de colector abierto que se utiliza para manejar el estado ENCENDIDO/APAGADO de un LED para indicar el estado de la salida Q de un FF. La especificación nominal del LED es $V_F = 2.4$ V a $I_F = 20$ mA, e I_F(máx) = 30 mA.

(a) ¿Qué voltaje aparecerá en la salida del 7406 cuando $Q = 0$?

(b) Seleccione un valor apropiado para la resistencia en serie, para una operación apropiada.

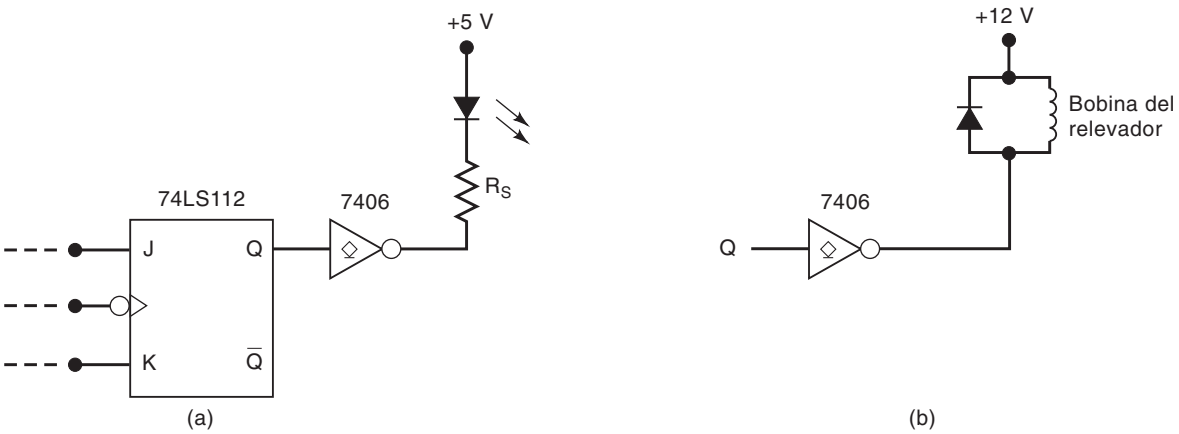


FIGURA 8-62 Problemas 8-29 y 8-30.

8-30. En la figura 8-62(b), la salida del 7406 se utiliza para cambiar la corriente hacia un relevador.

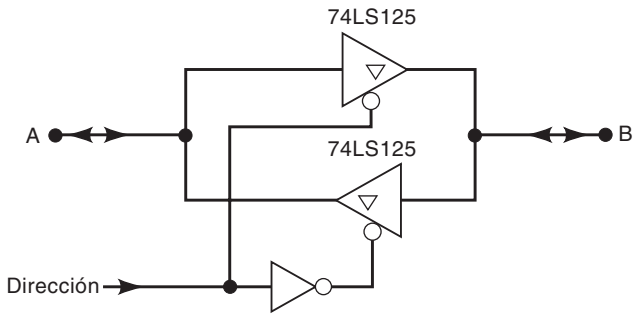
(a)* ¿Qué voltaje estará en la salida del 7406 cuando $Q = 0$?

(b)* ¿Cuál es el relevador de corriente más grande que puede usarse?

(c) ¿Cómo podemos modificar este circuito para utilizar un 7407?

N 8-31. La figura 8-63 muestra cómo pueden utilizarse dos búferes triestado para construir un *transceptor bidireccional* que permita transmitir datos digitales en cualquier dirección (de *A* hacia *B* o de *B* hacia *A*). Describa la operación del circuito para los dos estados de la entrada DIRECCION.

FIGURA 8-63
Problema 8-31.

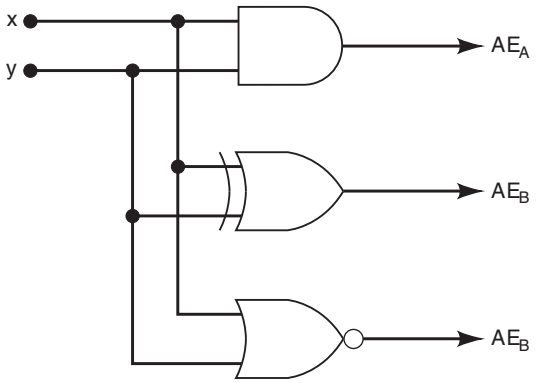


8-32. El circuito de la figura 8-64 se utiliza para proporcionar las entradas de habilitación para el circuito de la figura 8-37.

(a) Determine cuáles de las entradas de datos (*A*, *B* o *C*) aparecerán en el bus para cada combinación de entradas *x* y *y*.

(b) Explique por qué el circuito no funcionará si la compuerta NOR se cambia por una compuerta XNOR.

FIGURA 8-64
Problema 8-32.



8.33.* ¿Qué tipo de circuito contador del capítulo 7 podría manejar las entradas de habilitación en la figura 8-37, de manera que sólo un búfer esté encendido en un momento dado, y que los búferes se habiliten en forma secuencial?

SECCIÓN 8-14

B 8-34. **PREGUNTAS DE PRÁCTICA**

(a) ¿Qué familia lógica debe utilizarse si la velocidad máxima es de extrema importancia?

(b) ¿Qué familia lógica utiliza la mayor potencia?

(c) ¿Qué serie TTL es la más veloz?

(d) ¿Cuál de las series CMOS puras es la más veloz?

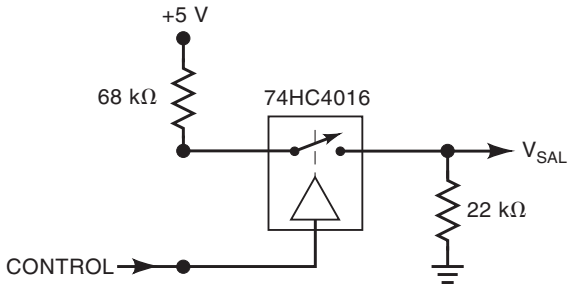
(e) ¿Qué familia tiene el mejor producto velocidad-potencia?

8-35. Nombre dos diferencias radicales entre las salidas ECL y las salidas TTL o CMOS.

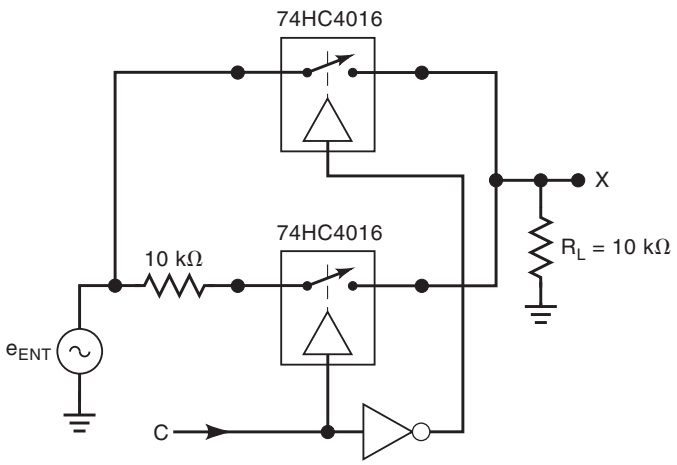
SECCIÓN 8-15

8-36.* Determine los valores aproximados de V_{SAL} para ambos estados de la entrada CONTROL en la figura 8-65.

FIGURA 8-65
Problema 8-36.



8-37.* Determine la forma de onda en la salida X de la figura 8-66 para las formas de onda de entrada dadas. Suponga que $R_{ENC} = 200\ \Omega$ para el interruptor bilateral.



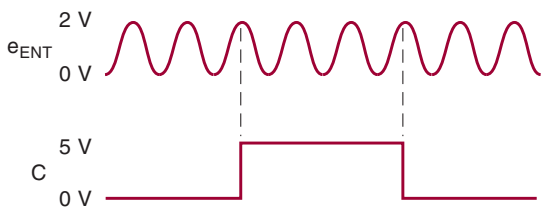
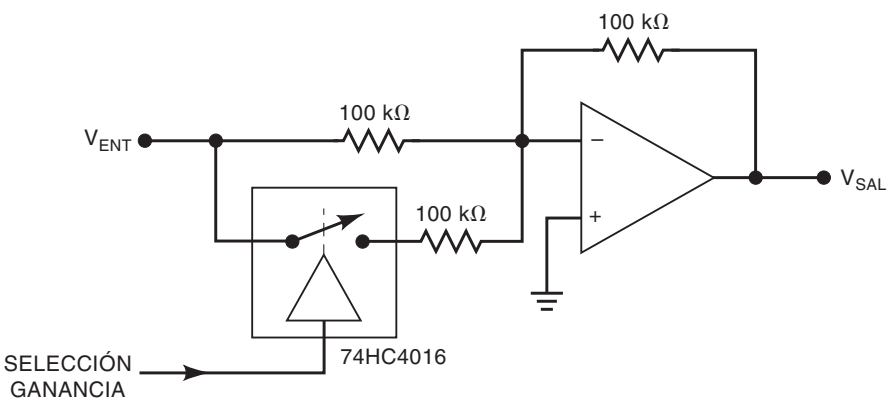


FIGURA 8-66 Problema 8-37.

N, D, A

8-38.* Determine la ganancia del circuito amplificador operacional de la figura 8-67 para los dos estados de la entrada SELECCIÓN GANANCIA. Este circuito muestra el principio básico de la amplificación de señales mediante el control digital.

FIGURA 8-67
Problema 8-38.



SECCIÓN 8-16

B 8-39. **PREGUNTA DE PRÁCTICA**

(a) ¿En cuál de las series CMOS se pueden manejar sus entradas directamente desde una salida TTL?

(b) ¿Cuál es la función de un traductor de nivel? ¿Cuándo se utiliza?

(c) *Verdadero o falso:* la mayoría de las salidas CMOS tienen problemas al suministrar la corriente de entrada de estado ALTO en TTL.

F 8-40. Consulte la figura 8-68(a), en donde una salida TTL 74LS (Q) maneja un INVERSOR CMOS que opera a V_{DD} = 10 V. Las formas de onda en Q y X aparecen como se muestra en la figura 8-68(b). ¿Cuál de las siguientes es una posible razón del por qué X permanece en ALTO?

(a) La fuente de 10 V está defectuosa.

(b) La resistencia de puesta en alto es demasiado grande.

(c) La salida del 74LS112 falla y produce un voltaje muy por debajo de los 10 V, manteniendo un nivel de 5.5 V en el estado ALTO. Este valor se encuentra en el intervalo indeterminado para la entrada CMOS.

(d) La entrada CMOS está sobrecargando la salida TTL.

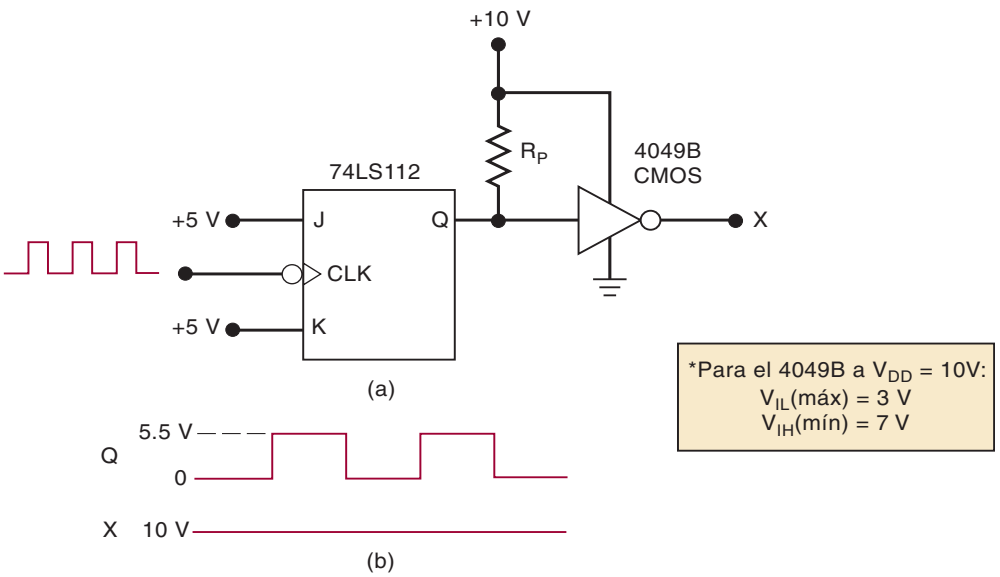


FIGURA 8-68 Problema 8-40.

8-41. (a)* Use la tabla 8-12 para determinar cuántas entradas 74AS pueden manejarse por lo general mediante una salida 4000B.

(b) Repita la parte (a) para una salida 74HC.

F 8-42. La figura 8-69 es un circuito lógico con un mal diseño. Contiene cuando menos ocho casos en donde las características de los CIs no se han tomado en cuenta de la manera correcta. Encuentre todas las que pueda.

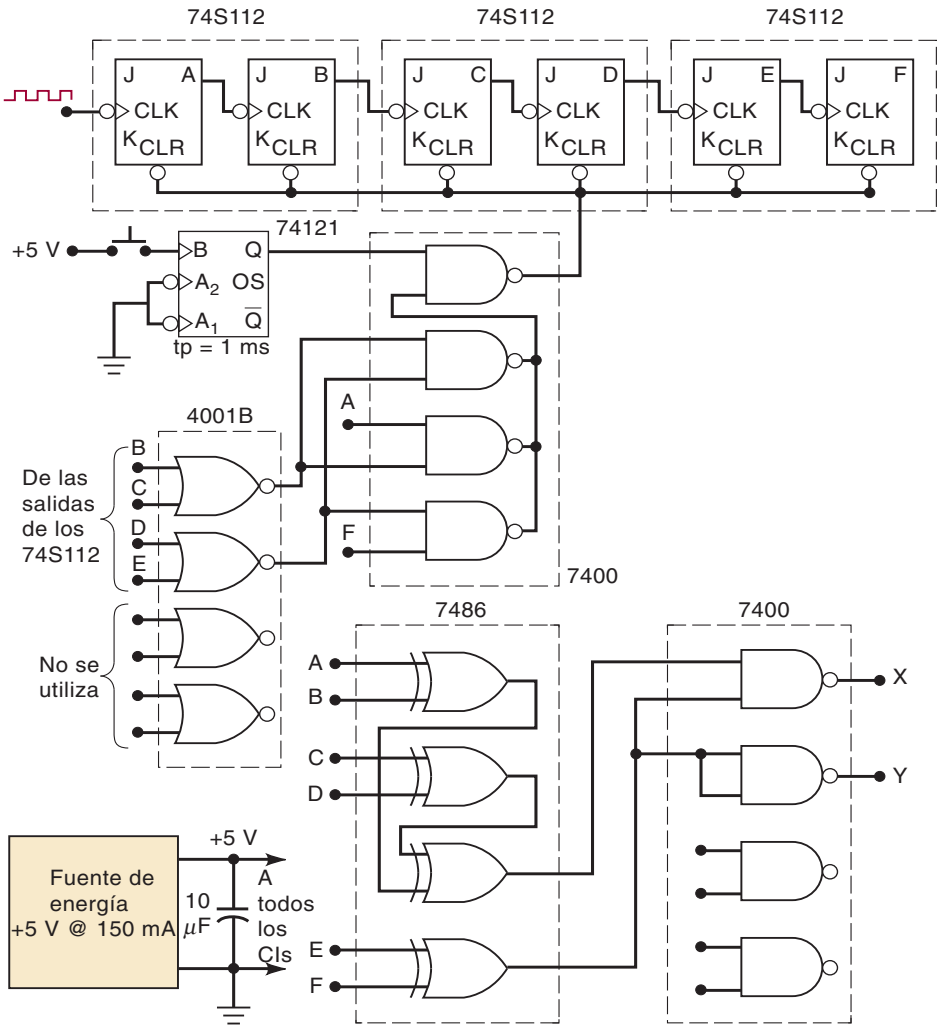


FIGURA 8-69 Problemas 8-42 y 8-43.

F 8-43. Repita el problema 8-42 con los siguientes cambios en el circuito:

- Cada CI TTL se sustituye con su equivalente 74LS.
- El 4001B se sustituye con un 74HCT02.

8-44.* Use la tabla 8-12 para explicar por qué el circuito de la figura 8-70 no funcionará así como está. ¿Cómo puede corregirse el problema?

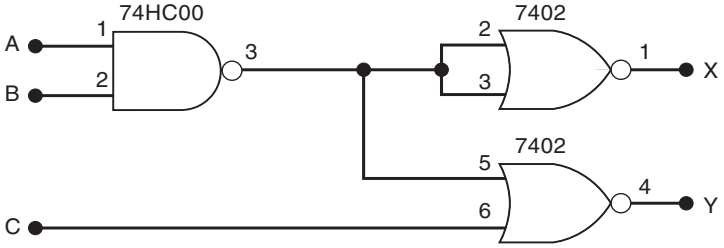


FIGURA 8-70 Problema 8-44.

SECCIÓN 8-18

D 8-45. El tanque de gasolina en su automóvil tiene una unidad de envío de nivel de combustible que trabaja igual que un potenciómetro. Un flotador se mueve hacia arriba y hacia abajo con el nivel de gasolina, y cambia el ajuste de la resistencia variable para producir un voltaje proporcional al nivel de gasolina. Un tanque lleno produce 12 V y un tanque vacío produce 0 V. Diseñe un circuito en el que utilice un LM339 para encender la luz indicadora de "Nivel de combustible bajo" cuando el nivel de voltaje de la unidad de envío esté por debajo de 0.5 V.

D 8-46.* Para modificar el circuito comparador de temperatura excesiva de la figura 8-52 se sustituye el sensor de temperatura LM34 por un LM35 que produzca como salida 10 mV por cada grado centígrado. La alarma debe aún activarse (ALTO) cuando la temperatura exceda los 100°F, que son aproximadamente 38°C. Recalcule los valores de R_1 y R_2 para completar la modificación.

SECCIÓN 8-19

F 8-47. El circuito de la figura 8-71 utiliza un CI 74HC05 que contiene seis INVERSORES de colector abierto. Los inversores están conectados en un arreglo de AND alambrado. La salida de la compuerta NAND siempre está en ALTO, sin importar las entradas A-H. Describa un procedimiento que utilice una sonda lógica y un pulsador lógico para aislar esta falla.

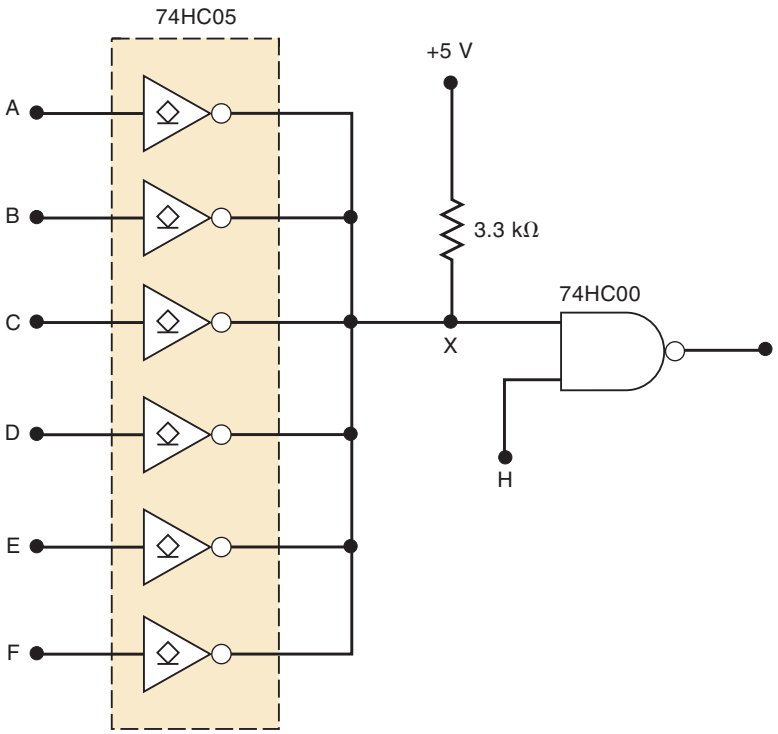


FIGURA 8-71
Problema 8-47.

F 8-48. El circuito de la figura 8-53 tiene un puente de soldadura a tierra en alguna parte entre la salida de la compuerta NAND y la entrada del FF. Describa un procedimiento para una prueba que pueda realizarse para indicar que la falla está en el tablero del circuito, y que probablemente no esté en la compuerta NAND o en los CIs de FF.

F 8-49.* En la figura 8-46, una sonda lógica indica que el extremo inferior de la resistencia de puesta en alto está fija en el estado ALTO. ¿Cuál de las siguientes fallas es la más probable?

(a) El transistor de suministro de corriente de la compuerta TTL está abierto.

(b) El transistor de drenado de corriente de la compuerta TTL tiene un corto en el colector-emisor.

(c) Hay una rotura en la conexión de R_P a la compuerta CMOS.

APLICACIÓN DE MICROCOMPUTADORA

A, N 8-50.* En el capítulo 5 vimos cómo un microprocesador (MPU), mediante el control de software, transfiere datos a un registro externo. El diagrama del circuito se repite en la figura 8-72. Una vez que los datos están en el registro, se almacenan ahí y se utilizan para cualquiera que sea su propósito. Algunas veces, cada bit individual en el registro tiene una función única. Por ejemplo, en las computadoras de los automóviles cada bit podría representar el estado de una variable física distinta que el MPU esté monitoreando. Un bit podría indicar cuando la temperatura del motor es demasiado alta. Otro bit podría indicar cuando la presión del aceite es demasiado baja. En otras aplicaciones, los bits en el registro se utilizan para producir una salida analógica que puede usarse para controlar dispositivos que requieran entradas analógicas que tengan muchos niveles de voltaje distintos.

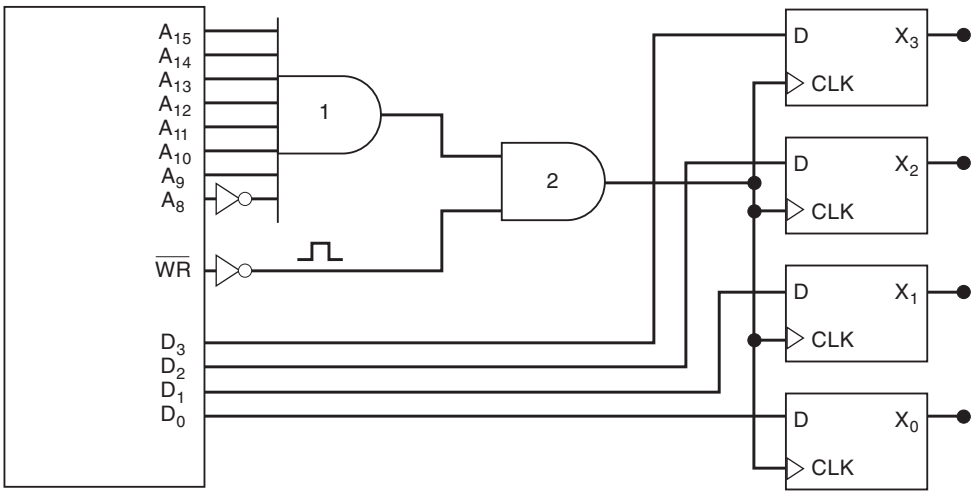


FIGURA 8-72 Problema 8-50.

La figura 8-73 muestra cómo podemos usar la MPU para generar un voltaje analógico; para ello tomamos los datos del registro de la figura 8-72 y los usamos para manejar las entradas a un amplificador de suma. Suponga que la MPU está ejecutando un programa que transfiere un nuevo conjunto de datos al registro cada 10 μs, de acuerdo con la tabla 8-14. Bosqueje la forma de onda resultante en V_{SAL}.

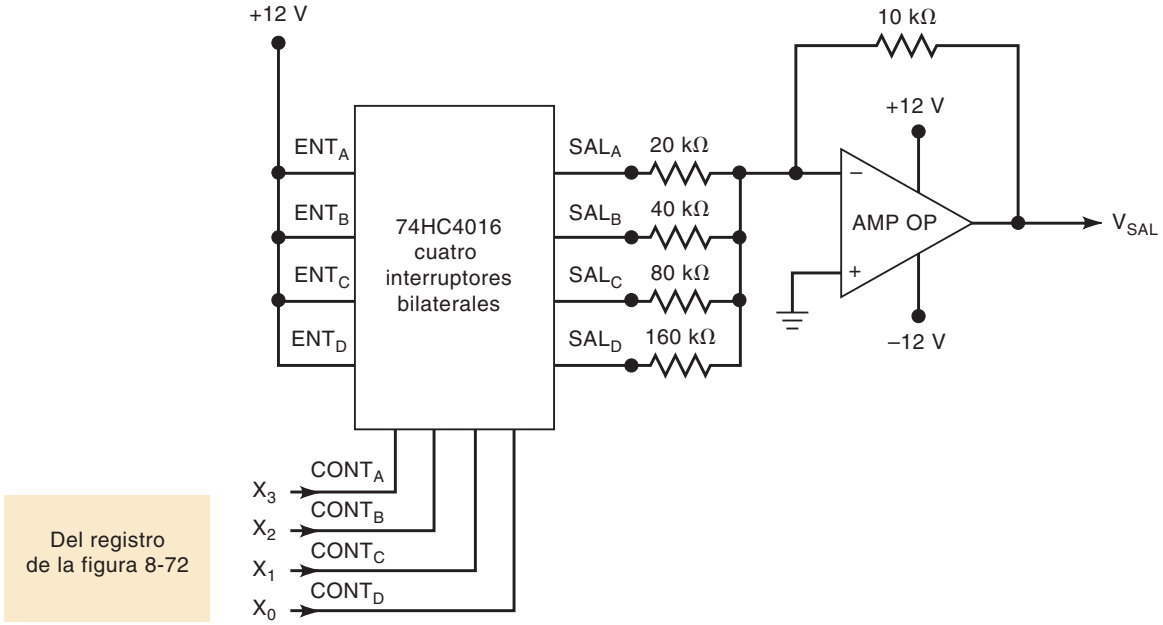


FIGURA 8-73. Problema 8-50..

TABLA 8-14 Problema 8-50.

Tiempo (μs)	Datos de la MPU
0	0000
10	0010
20	0100
30	0111
40	1010
50	1110
60	1111
70	1111
80	1110
90	1100
100	1000

RESPUESTAS A LAS PREGUNTAS DE REPASO DE LAS SECCIONES

SECCIÓN 8-1

1. Vea el texto. 2. Falso 3. Falso; V_{NH} es la diferencia entre V_{OH}(mín) y V_{IH}(mín). 4. Drenado de corriente: una salida recibe (drena) corriente de la entrada del circuito que controla. Suministro de corriente: una salida suministra (provee) corriente al circuito que controla. 5. DIP 6. Terminal tipo J. 7. Sus puntas están rotas. 8. No

SECCIÓN 8-2

1. Verdadero 2. BAJO 3. Tiempos de conmutación rápidos, baja disipación de potencia; pico de corriente grande durante el cambio de BAJO a ALTO. 4. Q_3 5. Q_3 6. No hay transistor con múltiples emisores.

SECCIÓN 8-4

1. (a) 74AS (b) 74S, 74LS (c) 74 estándar (d) 74S, 74LS, 74AS, 74ALS (e) 74ALS 2. Los tres pueden operar a 40 MHz, pero el 74ALS193 utilizará menos potencia. 3. Q_4, Q_5, en forma respectiva.

SECCIÓN 8-5

1. La resistencia en estado ENCENDIDO de Q_4 y V_{OL}(máx). 2. 12 3. Sus voltajes de entrada tal vez no permanezcan en los intervalos permitidos para el 0 y el 1 lógicos. 4. Dos; cinco

SECCIÓN 8-6

1. BAJO 2. Conectarlas a $+V_{CC}$ por medio de una resistencia de 1kΩ; conectarlas a otra entrada usada. 3. Conectarlas a tierra; conectarlas a otra entrada usada. 4. Falso; sólo en el estado BAJO. 5. La acción de conectar pequeños capacitares RF entre V_{CC} y tierra cerca de cada CI TTL, para filtrar los picos de voltaje ocasionados por cambios bruscos en la corriente durante las transiciones de la salida de BAJO a ALTO.

SECCIÓN 8-8

1. CMOS utiliza MOSFETs de canal P y de canal N. 2. Uno 3. Seis

SECCIÓN 8-9

1. 74C, HC, HCT, AHC, AHCT 2. 74ACT, HCT, AHCT 3. 74C, HC/HCT, AC/ACT, AHC/AHCT 4. BiCMOS 5. Máximo tiempo de propagación permisible; capacitancia de entrada de cada carga. 6. Consulte el texto. 7. CMOS 8. (a) Verdadero (b) Falso (c) Falso (d) Falso (e) Verdadero (f) Falso

SECCIÓN 8-10

1. Más circuitos en el chip; mayor velocidad de operación. 2. No puede soportar altos voltajes; el aumento en la disipación de potencia puede sobrecalentar el chip. 3. Igual que el de TTL estándar: 2.0 V. 4. 74ALVC, 74LV 5. 74LVT

SECCIÓN 8-11

1. Cuando dos o más salidas de un circuito se conectan juntas. 2. Puede fluir una corriente dañina; V_{OL} se excede de V_{OL}(máx). 3. El colector del transistor de drenado de corriente Q_4 está desconectado (no hay Q_3). 4. Para producir un nivel V_{OH}. 5. $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}\,\overline{E}\,\overline{F}$ 6. Porque no hay transistor de puesta en alto activo. 7. Vea la figura 8-34.

SECCIÓN 8-12

1. ALTO, BAJO, Hi-Z 2. Hi-Z 3. Cuando dos o más salidas triestado conectadas a un mismo bus se habilitan al mismo tiempo. 4. $E_A = E_B = 0, E_C = 1$ 5. Vea la figura 8-39.

SECCIÓN 8-13

1. Menos de 4 pulgadas. 2. Resistencia, capacitancia, inductancia. 3. Para reducir las reflexiones y la sobreoscilación negativa en la línea.

SECCIÓN 8-14

1. (a) Verdadero (b) Verdadero (c) Falso (d) Verdadero (e) Faso (f) Falso

SECCIÓN 8-15

1. El nivel lógico en la entrada de control determina el estado abierto/cerrado de un interruptor bidireccional, el cual puede pasar señales analógicas en cualquier dirección. 2. Verdadero

SECCIÓN 8-16

1. Se debe conectar una resistencia de puesta en alto a +5 V en la salida TTL.
2. Los valores de I_{OH} o I_{OL} del CMOS pueden llegar a ser demasiado bajos.

SECCIÓN 8-17

1. Toma la salida de un circuito reforzador y la condiciona de tal forma que sea compatible con los requerimientos de entrada de la carga. 2. Verdadero 3. Falso; por ejemplo, la serie 4000B no puede drenar I_{IL} de un dispositivo 74 o 74AS. 4. 74HCT y ACT

SECCIÓN 8-18

1. $V^{(+)} > V^{(-)}$ 2. $V^{(-)} > V^{(+)}$ 3. A la de colector abierto.

SECCIÓN 8-19

1. Inyecta un pulso de voltaje de la polaridad seleccionada en un nodo que no esté en corto con V_{CC} o tierra. 2. Falso 3. Falso 4. El LED del pulso emite un destello cada vez que se activa el pulsador.



CAPÍTULO 9

CIRCUITOS LÓGICOS MSI

CONTENIDO

9-1 Decodificadores
9-2 Decodificadores/ controladores de BCD a 7 segmentos
9-3 Pantallas de cristal líquido
9-4 Codificadores
9-5 Diagnóstico de fallas
9-6 Multiplexores (selectores de datos)
9-7 Aplicaciones de los multiplexores
9-8 Demultiplexores (distribuidores de datos)
9-9 Más diagnóstico de fallas
9-10 Comparador de magnitud
9-11 Convertidores de código
9-12 Buses de datos
9-13 El registro triestado 74ALS173/HC173
9-14 Operación del bus de datos
9-15 Decodificadores mediante el uso de HDL
9-16 El decodificador/controlador de 7 segmentos en HDL
9-17 Codificadores mediante el uso de HDL
9-18 Multiplexores y demultiplexores en HDL
9-19 Comparadores de magnitud en HDL
9-20 Convertidores de código en HDL

OBJETIVOS

Al completar este capítulo, usted podrá:

- Analizar y utilizar decodificadores y codificadores en varios tipos de aplicaciones de circuitos.
- Comparar las ventajas y desventajas de los LEDs y las LCDs.
- Utilizar la técnica de observación/análisis para diagnosticar fallas en circuitos digitales.
- Comprender la operación de los multiplexores y demultiplexores mediante el análisis de varias aplicaciones de circuitos.
- Comparar dos números binarios mediante el uso del circuito comparador de magnitud.
- Comprender la función y operación de los convertidores de código.
- Citar las precauciones que deben considerarse al conectar circuitos digitales mediante el uso del concepto del bus de datos.
- Usar HDL para implementar el equivalente de los circuitos lógicos MSI.

INTRODUCCIÓN

Los sistemas digitales obtienen datos codificados en binario e información con la que operan en cierta forma continua. Algunas de las operaciones son: (1) *decodificación y codificación*, (2) *multiplexaje*, (3) *demultiplexaje*, (4) *comparación*, (5) *conversión de código* y (6) *buses de datos*. Todas estas operaciones y otras más se han facilitado gracias a la disponibilidad de numerosos CIs en la categoría MSI (integración a mediana escala).

En este capítulo estudiaremos muchos de los tipos comunes de dispositivos MSI. Para cada tipo empezaremos con una breve discusión de su principio básico de operación y después presentaremos CIs específicos. Después le mostraremos cómo pueden usarse por sí solos o en combinación con otros CIs en diversas aplicaciones.

9-1 DECODIFICADORES

Un **decodificador** es un circuito lógico que acepta un conjunto de entradas que representan un número binario y activa sólo la salida que corresponde a ese número de entrada. En otras palabras, un circuito decodificador analiza sus entradas, determina cuál número binario está presente y activa la única salida que corresponde a ese número; todas las demás salidas permanecen inactivas. La figura 9-1 muestra el diagrama para un decodificador en general, con N entradas y M salidas. Como cada una de las N entradas puede ser 0 o 1, existen 2^N combinaciones posibles de entrada, o códigos. Para cada una de estas combinaciones, sólo una de las M salidas estará activa (en ALTO); todas las demás entradas estarán en BAJO. Muchos decodificadores

están diseñados para producir salidas activas en BAJO, en donde sólo la salida seleccionada está en BAJO mientras que las demás están en ALTO. Esta situación se indica mediante la presencia de pequeños círculos en las líneas de salida, en el diagrama del decodificador.

Algunos decodificadores no utilizan todos los 2^N posibles códigos de entrada, sólo unos cuantos. Por ejemplo, un decodificador de BCD a decimal tiene un código de entrada de cuatro bits y *diez* líneas de salida que corresponden a los *diez* grupos de código BCD, del 0000 al 1001. Los decodificadores de este tipo por lo general se diseñan de manera que si se aplica uno de los códigos no utilizados en la entrada, *ninguna* de las salidas se activará.

En el capítulo 7 vimos cómo se utilizan los decodificadores en conjunto con los contadores para detectar los diversos estados del contador. En esa aplicación, el FF en el contador proporcionó las entradas en código binario para el decodificador. Se utiliza el mismo circuito de decodificador básico, sin importar de dónde provengan las entradas. La figura 9-2 muestra el circuito para un decodificador con tres entradas y $2^3 = 8$ salidas. Utiliza sólo compuertas AND, por lo que las salidas son activas en ALTO. Observe que para un código de entrada dado, la única salida activa (ALTO) es la que corresponde al equivalente decimal del código de entrada binario (por ejemplo, la salida O_6 cambia a ALTO sólo cuando $CBA = 110_2 = 6_{10}$).

Se puede hacer referencia a este decodificador de varias formas. Se le puede llamar *decodificador de 3 a 8 líneas*, ya que tiene tres líneas de entrada y ocho de salida. También se le puede llamar *decodificador* o *convertidor de binario a octal*, ya que recibe un código binario de entrada de tres bits y activa una de las ocho salidas (octales) que corresponden a ese código. También se le conoce como *decodificador 1 de 8*, ya que sólo 1 de las 8 salidas se activa en un momento dado.



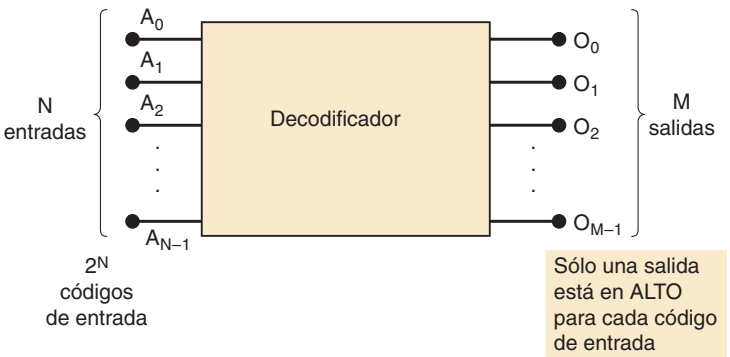


FIGURA 9-1 Diagrama general de un decodificador.

Entradas de HABILITACIÓN

Algunos decodificadores tienen una o más entradas de HABILITACIÓN, las cuales se utilizan para controlar la operación del decodificador. Por ejemplo, consulte el decodificador de la figura 9-2 y observe cómo tiene una línea de HABILITACIÓN común conectada a una cuarta entrada de cada compuerta. Si esta línea de HABILITACIÓN se mantiene en ALTO, el decodificador funcionará en forma normal, y el código de entrada *A*, *B*, *C* determinará cuál de las salidas está en ALTO. Pero si la HABILITACIÓN se mantiene en BAJO, *todas* las salidas se forzarán a quedar en el estado BAJO, sin importar los niveles en las entradas *A, B, C*. Por lo tanto, el decodificador se habilita sólo si la HABILITACIÓN está en ALTO.

La figura 9-3(a) muestra el diagrama lógico para el decodificador 74ALS138. Si examinamos este diagrama con cuidado, podremos determinar en forma exacta cómo funciona este decodificador. En primer lugar podemos observar que tiene salidas de compuertas NAND, por lo que sus salidas son activas en BAJO. Otra indi-

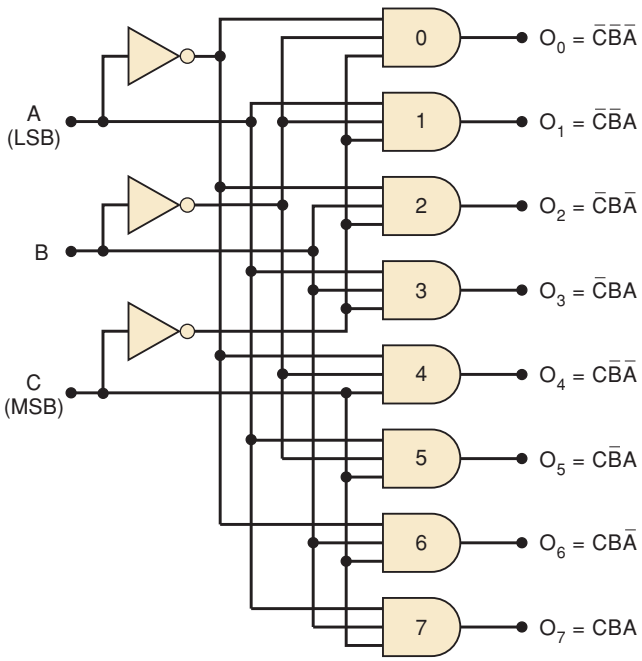


FIGURA 9-2
Decodificador de tres a 8 líneas (o 1 de 8).

C	B	A	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0



cación es el etiquetado de las salidas como $\overline{O}_7$, $\overline{O}_6$, $\overline{O}_5$ y así sucesivamente; la barra superior indica salidas activas en BAJO.

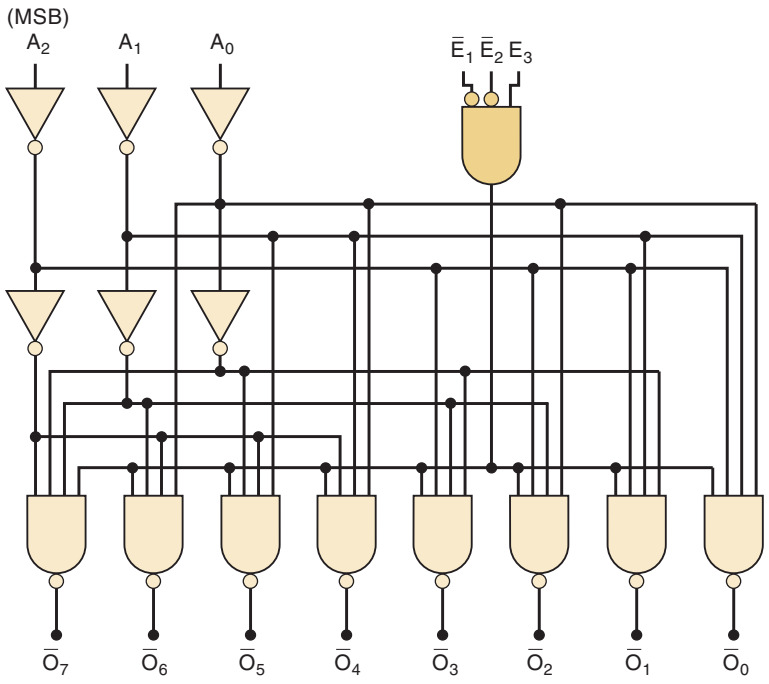
El código de entrada se aplica en A_2, A_1 y A_0, en donde A_2 es el MSB. Con tres entradas y ocho salidas, éste es un decodificador de 3 a 8 o, lo que es equivalente, un decodificador de 1 a 8.

Las entradas $\overline{E}_1$, $\overline{E}_2$ y E_3 son entradas de habilitación separadas que se combinan en la compuerta AND. Para poder habilitar las compuertas NAND de salida para que respondan al código en $A_2A_1A_0$, la salida de esta compuerta AND debe estar en ALTO. Esto ocurrirá sólo cuando $\overline{E}_1 = \overline{E}_2 = 0$ y $E_3 = 1$. En otras palabras, $\overline{E}_1$ y $\overline{E}_2$ son activas en BAJO, E_3 es activa en ALTO y las tres deben estar en sus estados activos para poder activar las salidas del decodificador. Si una o más de las entradas de habilitación se encuentra en su estado inactivo, la salida AND estará en BAJO, lo cual forzará a que todas las salidas NAND cambien a su estado ALTO inactivo, sin importar el código de entrada. En la tabla de verdad de la figura 9-3(b) se sintetiza esta operación. Recuerde que *x* representa la condición "no importa".

El símbolo lógico para el 74ALS138 se muestra en la figura 9-3(c). Observe cómo se representan las salidas activas en BAJO y cómo se representan las entradas de habilitación. Aun cuando la compuerta AND de habilitación se muestra como externa para el bloque del decodificador, forma parte de los circuitos internos del CI. El 74HC138 es la versión CMOS de alta velocidad de este decodificador.

FIGURA 9-3
(a) Diagrama lógico para el decodificador 74ALS138; (b) tabla de verdad; (c) símbolo lógico.

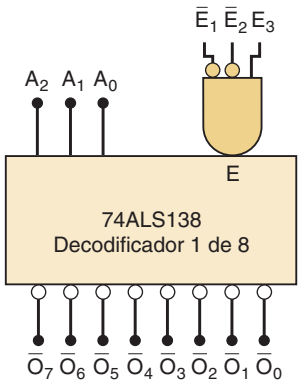




(a)

$\bar{E}_1$	$\bar{E}_2$	E_3	Salidas
0	0	1	Responde al código de entrada $A_2A_1A_0$
1	X	X	Deshabilitada – todas en ALTO
X	1	X	Deshabilitada – todas en ALTO
X	X	0	Deshabilitada – todas en ALTO

(b)



(c)

EJEMPLO 9-1

Indique los estados de las salidas de un 74ALS138 para cada uno de los siguientes conjuntos de entradas.

(a) $E_3 = \bar{E}_2 = 1, \bar{E}_1 = 0, A_2 = A_1 = 1, A_0 = 0$
(b) $E_3 = 1, \bar{E}_2 = \bar{E}_1 = 0, A_2 = 0, A_1 = A_0 = 1$

Solución

(a) Cuando $\bar{E}_2 = 1$, el decodificador se deshabilita y todas sus salidas estarán en su estado ALTO inactivo. Esto puede determinarse a partir de la tabla de verdad o siguiendo los niveles de entrada a través del circuito lógico.

(b) Todas las entradas de habilitación están activadas, por lo que la porción de decodificación está habilitada. Decodificará el código de entrada $011_2 = 3_{10}$ para activar la salida O_3. Por ende, $\bar{O}_3$ estará en BAJO y todas las demás salidas estarán en ALTO.

EJEMPLO 9-2

La figura 9-4 muestra cómo pueden ordenarse cuatro chips 74ALS138 y un INVERSOR para funcionar como un decodificador 1 de 32. Los decodificadores se etiquetan de Z_1 a Z_4 para una fácil referencia y las ochos salidas de cada uno de ellos se combinan en 32 salidas. Las salidas de Z_1 son de $\overline{O}_0$ a $\overline{O}_7$; a las salidas $\overline{O}_0$ – $\overline{O}_7$ de Z_2 se les cambia el nombre de $\overline{O}_8$ a $\overline{O}_{15}$, respectivamente; a las salidas de Z_3 se les cambia el nombre de $\overline{O}_{16}$ a $\overline{O}_{23}$ y a las salidas de Z_4 se les cambia el nombre de $\overline{O}_{24}$ a $\overline{O}_{31}$. Un código de entrada de cinco bits $A_4A_3A_2A_1A_0$ activará sólo una de estas 32 salidas para cada uno de los 32 posibles códigos de entrada.

(a) ¿Cuál salida se activará para $A_4A_3A_2A_1A_0 = 01101$?

(b) ¿Qué intervalo de códigos de entrada activará el circuito integrado Z_4?

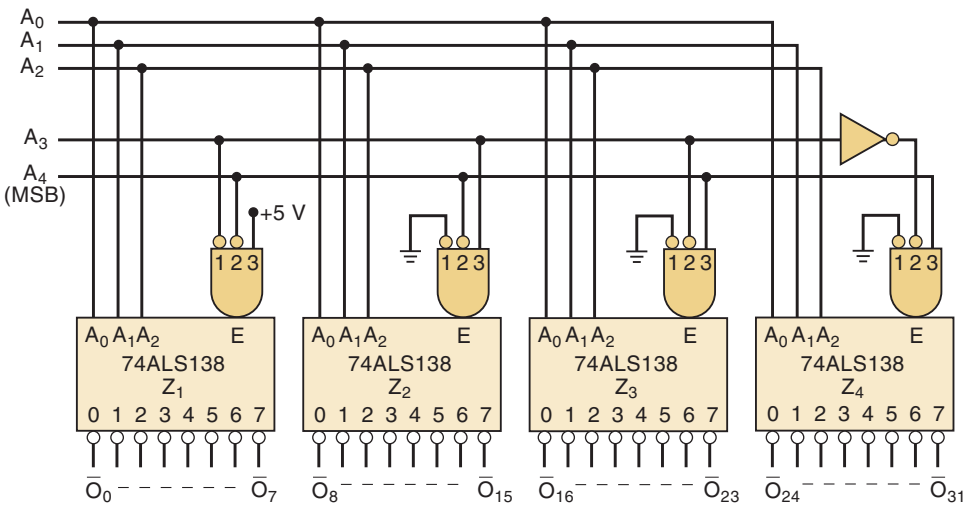


FIGURA 9-4 Cuatro circuitos integrados 74ALS138 que forman un decodificador 1 de 32.



Solución

(a) El código de cinco bits tiene dos porciones distintas. Los bits A_4 y A_3 determinan cuál de los circuitos Z_1 *a* Z_4 del decodificador se habilitará, mientras que $A_2A_1A_0$ determinan cuál salida del circuito habilitado se activará. Cuando $A_4A_3 = 01$, sólo Z_2 tiene todas sus entradas de habilitación activadas. Por lo tanto, Z_2 responde al código $A_2A_1A_0 = 101$ y activa su salida $\overline{O}_5$, la cual cambió su nombre a $\overline{O}_{13}$. Por ende, el código de entrada 01101 (que es el equivalente binario del 13 decimal) hará que la salida $\overline{O}_{13}$ cambie a BAJO, mientras que todas las demás permanecerán en ALTO.

(b) Para habilitar Z_4, tanto A_4 como A_3 deben estar en ALTO. En consecuencia, todos los códigos de entrada que varíen de 11000 (24_{10}) a 11111 (31_{10}) activarán Z_4. Esto corresponde a las salidas $\overline{O}_{24}$ a $\overline{O}_{31}$.

Decodificadores de BCD a decimal

La figura 9-5(a) muestra el diagrama lógico para un **decodificador de BCD a decimal** 7442. También está disponible como un 74LS42 y como un 74HC42. Cada salida cambia a BAJO sólo cuando se aplica su entrada BCD correspondiente. Por ejemplo, $\overline{O}_5$ cambiará a BAJO sólo cuando las entradas $DCBA = 0101$; $\overline{O}_8$ cambiará a BAJO sólo cuando $DCBA = 1000$. En cuanto a las combinaciones de entrada que sean inválidas para BCD, ninguna de las salidas se activará. A este decodificador también se le conoce como *decodificador de 4 a 10* o *decodificador 1 de 10*. La figura también muestra el símbolo lógico y la tabla de verdad para el 7442. Observe que este decodificador no tiene una entrada de habilitación. En el problema 9-7 veremos cómo

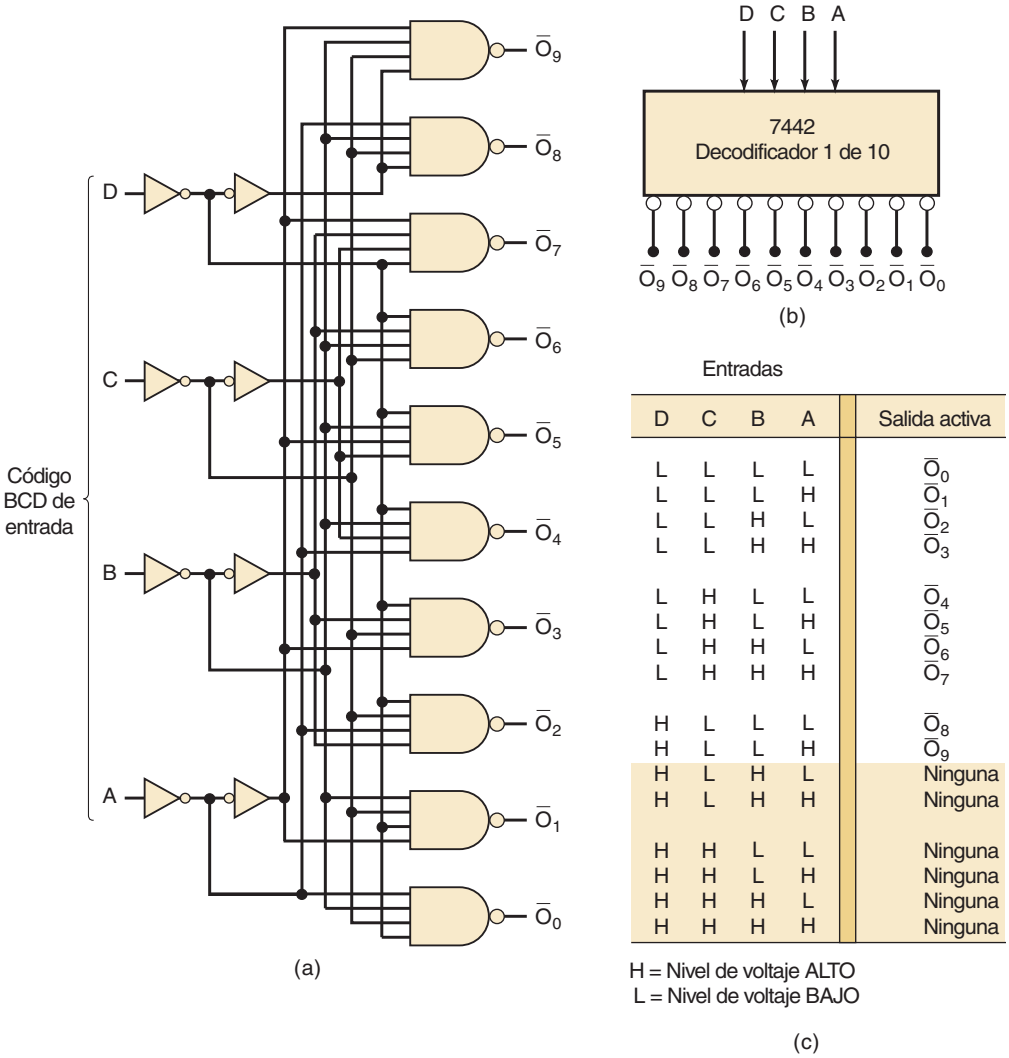


FIGURA 9-5 (a) Diagrama lógico para el decodificador de BCD a decimal 7442; (b) símbolo lógico; (c) tabla de verdad.

puede usarse el 7442 como decodificador de 3 a 8, utilizando la entrada *D* como entrada de habilitación.

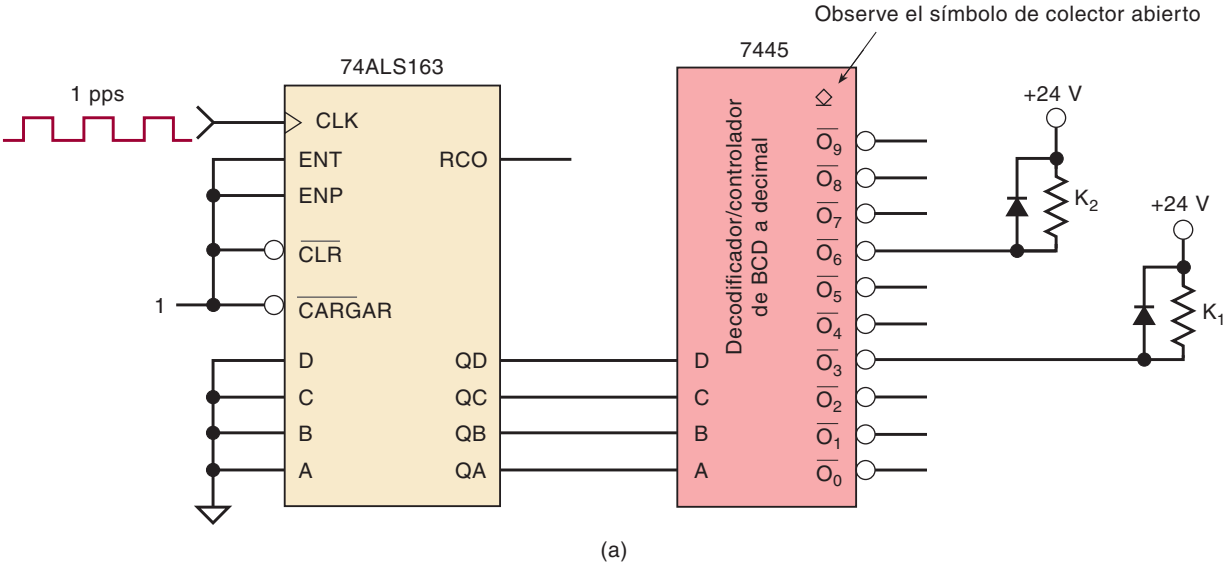
Decodificador/controlador de BCD a decimal

El TTL 7445 es un decodificador/**controlador** de BCD a decimal. El término *controlador* se agrega a su descripción debido a que este CI tiene salidas de colector abierto, las cuales pueden operar con límites de corriente y voltaje más altos que los de una salida TTL normal. Las salidas del 7445 pueden drenar hasta 80 mA en el estado BAJO, y pueden elevarse hasta 30 V en el estado ALTO. Esto los hace adecuados para manejar cargas en forma directa, tales como LEDs o lámparas indicadoras, relevadores o motores de corriente directa.

Aplicaciones de los decodificadores

Los decodificadores se utilizan siempre que debe activarse una salida o un grupo de salidas sólo cuando ocurra una combinación específica de niveles de entrada. A menudo estos niveles de entrada se proporcionan mediante las salidas de un contador o registro.

Cuando las entradas del decodificador provienen de un contador al que se aplican pulsos continuos, las salidas del decodificador se activarán en forma secuencial y podrán usarse como señales de sincronización o de secuencia para encender o apagar dispositivos en tiempos específicos. En la figura 9-6 se muestra un ejemplo de esta operación; aquí se utiliza el contador 74ALS163 y el decodificador/controlador 7445 que describimos antes.



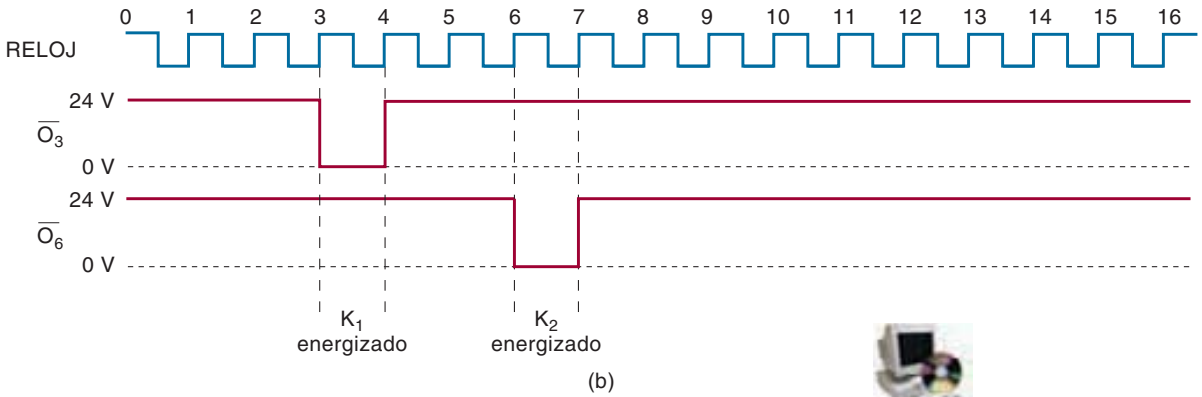


FIGURA 9-6 Ejemplo 9-3: combinación de contador/decodificador utilizada para proporcionar operaciones de sincronización y secuencia.

EJEMPLO 9-3

Describa la operación del circuito de la figura 9-6(a).

Solución

Se está aplicando un pulso al contador mediante una señal de 1-pps, de manera que avance en secuencia por los conteos binarios a la velocidad de 1 conteo/segundo. Las salidas de los FFs del contador se conectan como entradas para el decodificador. Las salidas de colector abierto O_3 y O_6 se utilizan para encender y apagar los relevadores K_1 y K_2. Por ejemplo, cuando O_3 se encuentre en su estado ALTO inactivo, su transistor de salida estará apagado (no conducirá), por lo que no podrá fluir corriente a través del relevador K_1 y se desenergizará. Cuando O_3 se encuentre en su estado activo en BAJO, su transistor de salida se encenderá y actuará como drenador para la corriente que fluye a través de K_1, de manera que este relevador se energizará. Observe que los relevadores operan con +24 V. Observe además la pre-

sencia de los diodos conectados en paralelo con los serpentines de los relevadores; estos diodos protegen los transistores de salida del decodificador contra el extenso voltaje de "patada inductiva" que se produce cuando la corriente del serpentín se detiene bruscamente.

El diagrama de sincronización en la figura 9-6(b) muestra la secuencia de eventos. Si suponemos que el contador se encuentra en el estado 0000 en el tiempo 0, entonces ambas salidas $\overline{O}_3$ y $\overline{O}_6$ se encuentran inicialmente en el estado ALTO inactivo, en donde sus transistores de entrada están apagados y ambos relevadores están desenergizados. A medida que se apliquen pulsos de reloj, el contador se incrementará una vez por segundo. En la NGT del cuarto pulso, el contador cambiará al estado 0100 (4). Esto desactivará a $\overline{O}_3$ y se desenergizará el relevador K_1.

De manera similar, en el tiempo 6 el contador cambiará al estado 0110 (6); esto hará que $\overline{O}_6 = 0$ y se energizará K_2. En el tiempo 7, el contador cambiará al estado 0111 (7) y desactivará O_6 para desenergizar a K_2.

El contador seguirá contando a medida que se le apliquen pulsos. Después de 16 pulsos, la secuencia que acabamos de describir volverá a comenzar.

Los decodificadores se utilizan mucho en el sistema de memoria de una computadora, en donde responden al código de dirección que genera el procesador central para activar una posición de memoria específica. Cada CI de memoria contiene muchos registros, los cuales pueden almacenar números binarios (datos). Cada registro necesita tener su propia dirección única para distinguirlo de los demás registros. Un decodificador se integra en los circuitos integrados de la memoria y permite que se active un registro de almacenamiento específico cuando se aplica una combinación única de entradas (es decir, su dirección). Por lo general, en un sistema hay varios CIs de memoria que se combinan para conformar toda la capacidad de almacenamiento completa. Un decodificador se utiliza para seleccionar un CI de memoria en respuesta a un intervalo de direcciones; para ello se decodifican los bits más significativos de la dirección del sistema y se habilita (selecciona) un circuito específico. En el problema 9-63 examinaremos esta aplicación y la estudiaremos con mucho más detalle cuando leamos sobre las memorias en el capítulo 12.

En los sistemas de memoria más complicados, los circuitos de memoria se ordenan en múltiples bancos que deben seleccionarse en forma individual o simultánea, dependiendo del número de bytes que requiere el microprocesador. Esto significa que bajo ciertas circunstancias, deberá activarse más de una salida del decodificador. Para sistemas como éste, a menudo se utiliza un dispositivo de lógica programable para implementar el decodificador, ya que no basta con un simple decodificador de 1 a 8. Los dispositivos lógicos programables pueden usarse con facilidad para aplicaciones de decodificación personalizadas.

PREGUNTAS DE REPASO

1. ¿Puede activarse más de una salida de un decodificador a la vez?
2. ¿Cuál es la función de la(s) entrada(s) de habilitación de un decodificador?
3. ¿Qué diferencia hay entre el 7445 y el 7442?
4. El 74154 es un decodificador de 4 a 16 con dos entradas de habilitación activas en BAJO. ¿Cuántas terminales (incluyendo energía y tierra) tiene este CI?

9-2 DECODIFICADORES/CONTROLADORES DE BCD A 7 SEGMENTOS

La mayoría del equipo digital cuenta con cierto medio para visualizar la información en un formato que el usuario u operador pueda comprender con facilidad.

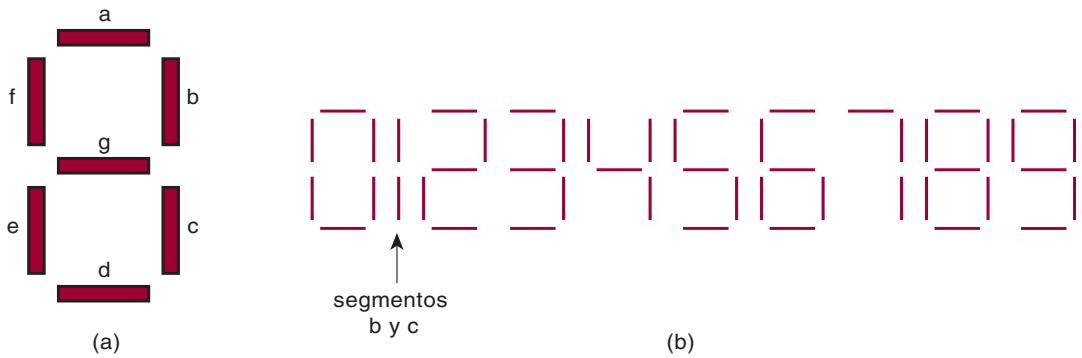


FIGURA 9-7 (a) Arreglo de 7 segmentos; (b) segmentos activos para cada dígito.

A menudo esta información consta de datos numéricos o alfanuméricos (números y letras). Uno de los métodos más simples y populares para visualizar dígitos numéricos utiliza una configuración de 7 segmentos [figura 9-7(a)] para formar los caracteres decimales del 0 al 9 y algunas veces los caracteres hexadecimales A-F. Un arreglo común utiliza diodos emisores de luz (LEDs) para cada segmento. Al controlar la corriente que fluye a través de cada LED, algunos segmentos se encenderán y otros se oscurecerán de manera que se genere el patrón del carácter deseado. La figura 9-7(b) muestra los patrones de los segmentos que se utilizan para visualizar los diversos dígitos. Por ejemplo, para un "6" se alumbran los segmentos *a*, *c*, *d*, *e*, *f* y *g*, mientras que se oscurece el segmento *b*.

Un **decodificador/controlador de BCD a 7 segmentos** se utiliza para tomar una entrada BCD de cuatro bits y proporcionar las salidas que pasarán corriente a través de los segmentos apropiados para que se visualice el dígito decimal. La lógica para este decodificador es más complicada que la de los decodificadores que hemos visto hasta ahora, ya que cada salida se activa para más de una combinación de entradas. Por ejemplo, el segmento *e* se debe activar para cualquiera de los dígitos 0, 2, 6 y 8, lo cual significa que se activará cada vez que ocurra uno de los códigos 0000, 0010, 0110 o 1000.

La figura 9-8(a) muestra un decodificador/controlador de BCD a 7 segmentos (TTL 7446 o 7447) que se utiliza para controlar una lectura con LEDs de 7 segmentos. Cada segmento consiste en un LED (diodo emisor de luz). Los diodos son dispositivos de estado sólido que permiten que la corriente fluya a través de ellos en una dirección, pero bloquean el flujo en la otra dirección. Cada vez que el ánodo de un LED sea más positivo que el cátodo por un valor aproximado a los 2 V, el LED se encenderá. Todos los ánodos de los LEDs están conectados a V_{CC} (+5 V). Los cátodos de los LEDs se conectan, a través de resistencias limitadoras de corriente, a las salidas apropiadas del decodificador/controlador, el cual tiene salidas activas en BAJO que son transistores controladores de colector abierto y pueden drenar una corriente bastante grande, ya que las lecturas de los LEDs pueden requerir de 10 a 40 mA por segmento, dependiendo de su tipo y de su tamaño.

Para ilustrar la operación de este circuito, vamos a suponer que la entrada BCD es $D = 0$, $C = 1$, $B = 0$, $A = 1$, que en BCD es el número 5. Con estas entradas, las salidas del decodificador/controlador *a*, *f*, *g*, *c* y *d* se cambiarán al nivel BAJO (se conectarán a tierra), con lo cual se permitirá el flujo de la corriente a través de los segmentos de LED *a*, $\overline{f}$, $\overline{g}$, $\overline{c}$ y $\overline{d}$, y en consecuencia se visualizará el número 5. Las salidas *b* y *e* estarán en ALTO (abiertas), por lo que los segmentos de LED $\overline{b}$ y $\overline{e}$ no podrán conducir.

Los decodificadores/controladores 7446/47 están diseñados para activar segmentos específicos, aún para códigos de entrada que no sean BCD (mayores de 1001). La figura 9-8(b) muestra los patrones de segmentos activados para todos los posibles códigos de entrada, desde 0000 hasta 1111. Observe que un código de entrada de 1111 (15) borrará todos los segmentos.

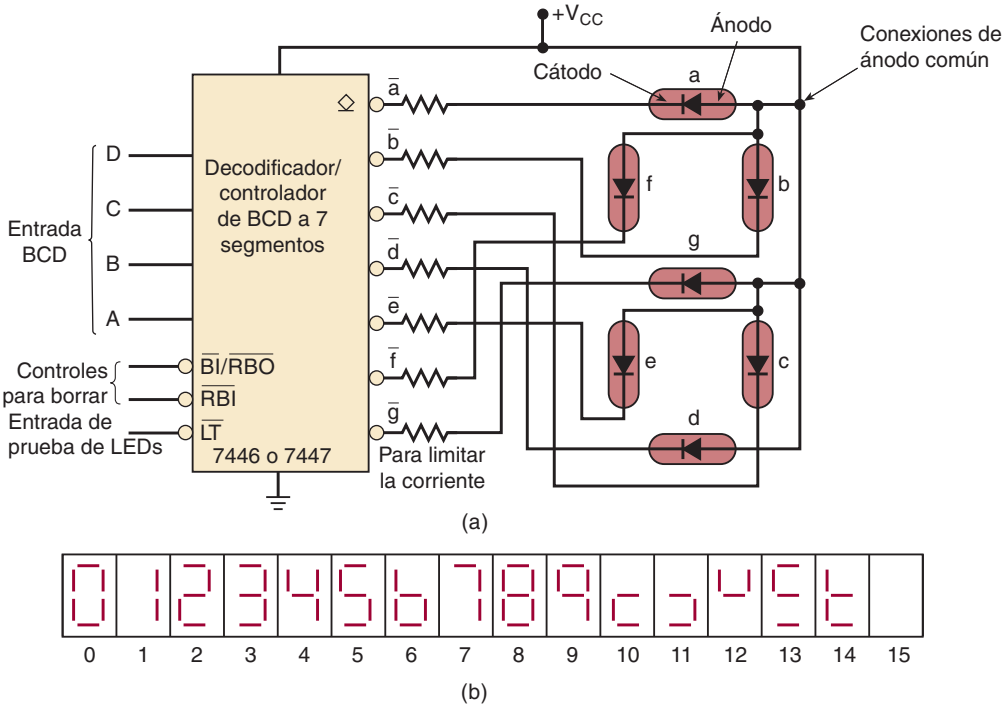


FIGURA 9-8
(a) Decodificador/ controlador de BCD a 7 segmentos que controla una pantalla de LEDs de 8 segmentos con ánodo común; (b) patrones de segmentos para todos los posibles códigos de entrada.

Los decodificadores/controladores de siete segmentos tales como el 7446/47 son excepciones a la regla, en cuanto a que los circuitos del decodificador sólo activan una salida para cada combinación de entradas. En realidad, activan un patrón único de salidas para cada combinación de entradas.

Comparación entre pantallas de LEDs de ánodo común y de cátodo común

La pantalla de LEDs que se utilizó en la figura 9-8 es del tipo **ánodo común**, ya que los ánodos de todos los segmentos están conectados a V_{CC}. Otro tipo de pantalla de LEDs de 7 segmentos utiliza un arreglo de **cátodo común**, en el cual los cátodos de todos los segmentos se conectan, juntos, a tierra. Este tipo de pantalla debe controlarse mediante un decodificador/controlador de BCD a 7 segmentos con salidas activas en ALTO que apliquen un voltaje ALTO a los ánodos de los segmentos que se deben activar. Como cada segmento requiere de 10 a 20 mA de corriente para encenderse, por lo general, no se utilizan dispositivos TTL y CMOS para controlar la pantalla de cátodo común en forma directa. Si recuerda, en el capítulo 8 vimos que las salidas TTL y CMOS no pueden suministrar grandes cantidades de corriente. A menudo se utiliza un circuito de interfaz de transistores entre los chips decodificadores y la pantalla de cátodo común.

EJEMPLO 9-4

Cada segmento de una pantalla de LEDs de 7 segmentos ordinaria está clasificado para operar a 10 mA y 2.7 V para un brillo normal. Calcule el valor de la resistencia limitadora de corriente necesaria para producir una corriente aproximada de 10 mA por segmento.

Solución

Si hacemos referencia a la figura 9-8(a), podemos ver que la resistencia en serie debe tener una caída de voltaje igual a la diferencia entre $V_{CC} = 5$ V y el voltaje de segmento de 2.7 V. Estos 2.3 V a través de la resistencia deben producir una corriente aproximada de 10 mA. Por ende, tenemos que

$$R_S = \frac{2.3\text{ V}}{10\text{ mA}} = 230\ \Omega$$

Puede utilizarse un valor de resistencia estándar que sea aproximado a éste. Una resistencia de 220 Ω sería una excelente opción.

PREGUNTAS DE REPASO

1. ¿Cuáles segmentos de LED estarán encendidos para una entrada de 1001 en el decodificador/controlador?
2. *Verdadero o falso:* más de una salida de un decodificador/controlador de BCD a 7 segmentos puede estar activa en un momento dado.

9-3 PANTALLAS DE CRISTAL LÍQUIDO

Una pantalla de LEDs genera o emite energía de luz a medida que se pasa corriente a través de los segmentos individuales. Una pantalla de cristal líquido (**LCD**) controla la reflexión de la luz disponible, que puede ser la luz ambiental (de los alrededores) tal como la luz del sol, o la luz normal de una habitación; las LCDs *reflectivas* utilizan la luz ambiental. La luz disponible podría proporcionarse a través de una pequeña fuente de luz que forme parte de la unidad de visualización, las LCDs con *retroiluminación* utilizan este método. En cualquier caso, las LCDs han ganado una amplia aceptación debido a su extremadamente bajo consumo de energía en comparación con los LEDs, en especial en los equipos operados por baterías tales como calculadoras, relojes digitales e instrumentos portátiles de medición. Por otro lado, los LEDs tienen la ventaja de tener una pantalla mucho más brillante que las LCDs reflectivas, por lo que pueden verse con facilidad en áreas oscuras o con poca iluminación.

En esencia, las LCDs operan a partir de una señal de corriente alterna de bajo voltaje (por lo general de 3 a 15 V rms) y baja frecuencia (de 25 a 60 Hz), y consumen muy poca corriente. Por lo general, se distribuyen como pantallas de 7 segmentos para lecturas numéricas, como se muestra en la figura 9-9(a). El voltaje de corriente alterna necesario para encender un segmento se aplica entre el segmento y el **plano posterior**, el cual es común para todos los segmentos. El segmento y el plano posterior forman un capacitor que consume muy poca corriente, siempre y cuando la frecuencia de corriente alterna se mantenga baja. Por lo general, no es menor de 25 Hz, ya que se produciría un parpadeo visible.

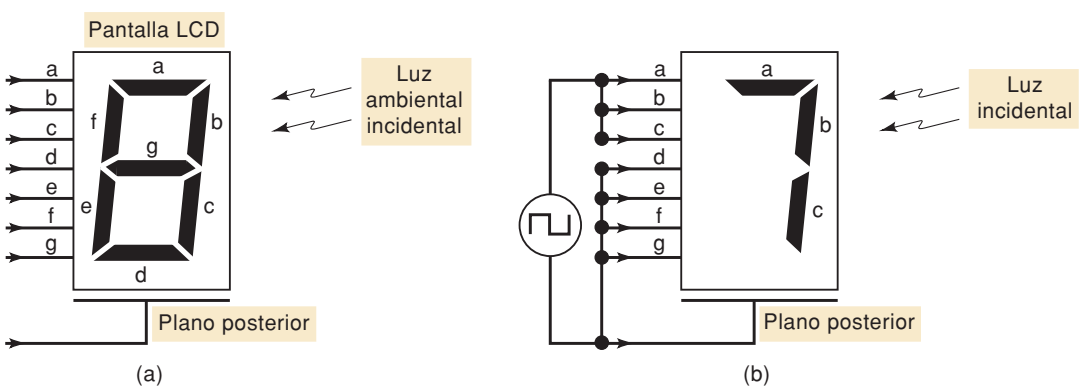


FIGURA 9-9 Pantalla de cristal líquido: (a) disposición básica; (b) al aplicar un voltaje entre el segmento y el plano posterior se ENCIENDE el segmento. Un voltaje de cero APAGA el segmento.

Ahora veremos una explicación bastante simplificada de cómo opera una LCD. Cuando no hay diferencia en el voltaje entre un segmento y el plano posterior, se dice que el segmento está *desactivado* (APAGADO). Los segmentos *d*, *e*, *f* y *g* en la figura 9-9(b) están APAGADOS y reflejarán la luz incidental de manera que aparezcan invisibles al contrastar con su plano posterior. Cuando se aplica un voltaje de corriente alterna apropiado entre un segmento y el plano posterior, el segmento se activa (ENCENDIDO). Los segmentos *a*, *b* y *c* en la figura 9-9(b) están ENCENDIDOS y no reflejarán la luz incidental, por lo cual aparecerán oscuros al contrastarlos con su fondo.

Control de una LCD

Un segmento LCD se ENCENDERÁ cuando se aplique un voltaje de corriente alterna entre el segmento y el plano posterior, y se APAGARÁ cuando no haya voltaje entre los dos. En vez de generar una señal de corriente alterna, es práctica común producir el voltaje de corriente alterna requerido mediante la aplicación de ondas cuadradas desfasadas al segmento y al plano posterior. Esto se ilustra en la figura 9-10(a) para un segmento. Se aplica una onda cuadrada de 40 Hz al plano posterior y también a la entrada que controlará si el segmento está ENCENDIDO o APAGADO.

Cuando la entrada de CONTROL esté en BAJO, la salida de la compuerta XOR será idéntica a la onda cuadrada de 40 Hz, para que las señales que se apliquen al segmento y al plano posterior sean iguales. Como no hay diferencia en el voltaje, el seg-

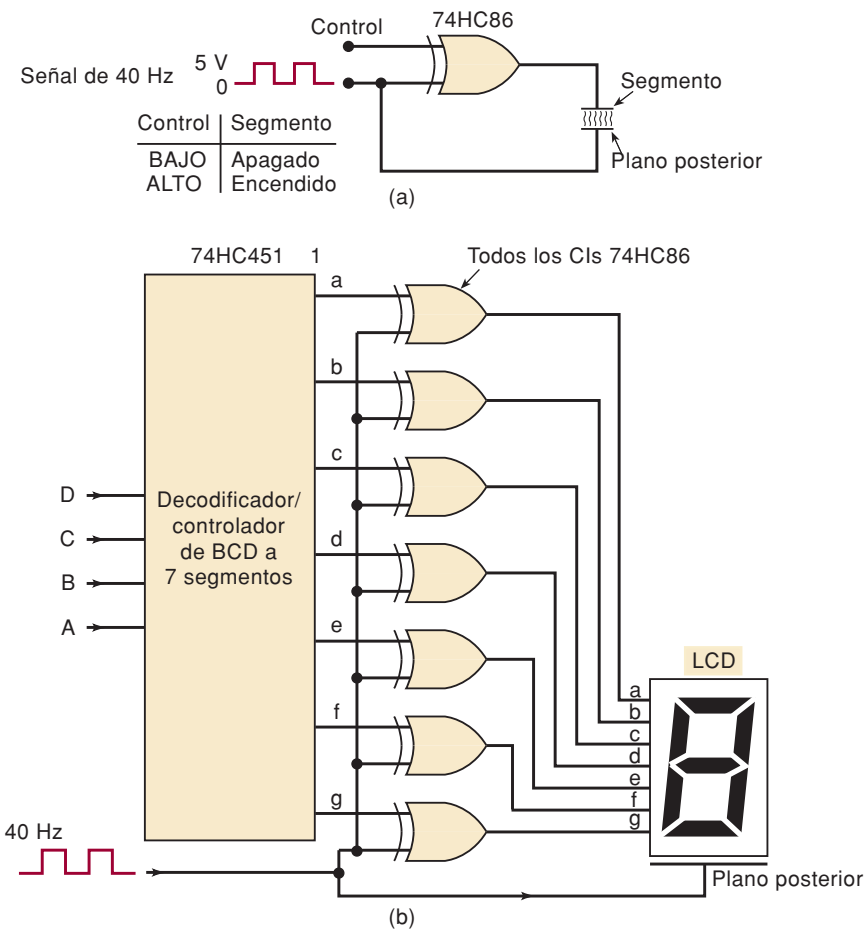


FIGURA 9-10
(a) Método para controlar un segmento LCD; (b) control de una pantalla de 7 segmentos.

mento estará APAGADO. Cuando la entrada de CONTROL esté en ALTO, la salida de la compuerta XOR será el INVERSO de la onda cuadrada de 40 Hz, para que la señal que se aplique al segmento esté desfasada en comparación con la señal que se aplique al plano posterior. Como resultado, el voltaje del segmento alternará entre +5 V y −5 V con respecto al plano posterior. Este voltaje de corriente alterna ENCENDERÁ el segmento.

Esta misma idea puede extenderse a una pantalla LCD completa de 7 segmentos, como se muestra en la figura 9-10(b). Aquí, el decodificador/controlador de BCD a 7 segmentos CMOS 74HC4511 suministra las señales de CONTROL para cada una de las siete compuertas XOR, para los siete segmentos. El 74HC4511 tiene salidas activas en ALTO, ya que se requiere un nivel ALTO para encender un segmento. El decodificador/controlador y las compuertas XOR de la figura 9-10(b) están disponibles en un encapsulado. El CMOS 74HC4543 es uno de esos dispositivos. Toma el código BCD de entrada y proporciona las salidas para controlar los segmentos LCD en forma directa.

En general se utilizan dispositivos CMOS para controlar LCDs por dos razones: (1) requieren mucho menos energía que TTL y se adaptan mejor a las aplicaciones operadas por baterías en las que se utilizan LCDs; (2) el voltaje TTL en estado BAJO no es exactamente igual a 0 V, y puede llegar hasta 0.4 V. Esto producirá un componente de voltaje de corriente directa entre el segmento y el plano posterior, lo cual reducirá en forma considerable la vida de una LCD.

Tipos de LCDs

Los cristales líquidos están disponibles como pantallas numéricas decimales de 7 segmentos. Vienen en muchos tamaños y con muchas características especiales, tales como signos de dos puntos (:) para las pantallas de reloj, indicadores de + y de − para los voltímetros digitales, puntos decimales para las calculadoras e indicadores de batería baja debido a que muchos dispositivos LCD son operados por baterías. Estas pantallas deben controlarse mediante un CI decodificador/controlador tal como el 74HC4543.

Una pantalla LCD más complicada pero que ya está disponible es el módulo LCD alfanumérico. Estos módulos están disponibles a través de muchas compañías y en numerosos formatos tales como los de 1 línea por 16 caracteres, hasta los de 4 líneas por 40 caracteres. La interfaz para estos módulos se estandarizó, de manera que un módulo LCD de cualquier fabricante utilizará las mismas señales y el mismo formato de datos. El módulo incluye algunos circuitos VLSI que facilitan el uso de este dispositivo. Se utilizan ocho líneas de datos para enviar el código ASCII para lo que se desee visualizar. Estas líneas de datos también llevan códigos de control especiales para el registro de comandos de la LCD. Se utilizan otras tres entradas (Selección de registro, Lectura/Escritura y Habilitación) para controlar la ubicación, dirección y sincronización de la transferencia de datos. A medida que se envían caracteres al módulo, los almacena en su propia memoria y los escribe en la pantalla de visualización.

Otros módulos LCD permiten al usuario crear una pantalla de gráficos mediante el control de los puntos individuales en la pantalla, a los cuales se les conoce como **píxeles**. Los paneles LCD más grandes pueden explorarse a una velocidad alta, con lo cual se producen películas de video de alta calidad. En estas pantallas, las líneas de control se distribuyen en una rejilla de filas y columnas. En la intersección de cada fila y columna hay un píxel que actúa como una "ventana" u "obturador", el cual se puede abrir y cerrar por medios electrónicos para controlar la cantidad de luz que se transmite a través de la celda. El voltaje de una fila a una columna determina el brillo de cada píxel. En una computadora portátil se almacena un número binario para cada píxel en la memoria de "video". Estos números se convierten en voltajes que se aplican a la pantalla.

Cada píxel en una pantalla a color está compuesto de tres subpíxeles. Estos subpíxeles controlan la luz que pasa a través de un filtro rojo, verde o azul para producir el color de cada píxel. En una pantalla LCD de 640 por 480 habría 640×3 conexiones para las columnas y 480 conexiones para las filas, para un total de 2400

conexiones a la LCD. Es obvio que el circuito controlador para un dispositivo de este tipo es un circuito VLSI muy complicado.

Los avances en la tecnología para las pantallas LCD han incrementado la velocidad a la cual pueden encenderse y apagarse los píxeles. A las pantallas antiguas se les conoce como Trenzada Nemática (TN) o Super Trenzada Nemática (STN). A estos dispositivos se les denomina LCDs pasivas. En vez de usar un plano posterior uniforme como el de las pantallas LCD de 7 segmentos, tienen líneas paralelas conductoras, fabricadas en dos piezas de vidrio. Las dos hojas de vidrio se utilizan para aprisionar el material de cristal líquido con las líneas conductoras a 90°, con lo cual se forma una rejilla de filas y columnas, como se muestra en la figura 9-11. La intersección de cada fila y columna forma un píxel. La conmutación de la corriente entre encendido y apagado se realiza en el CI controlador que se conecta a las filas y columnas de la pantalla. Las pantallas de matriz pasiva son bastante lentas para apagarse. Esto limita la velocidad a la cual pueden desplazarse los objetos en la pantalla sin dejar un rastro sombreado detrás de ellos.

Las pantallas más recientes se llaman LCDs TFT de matriz activa. Este último término significa que se utiliza un elemento activo en la pantalla para encender y apagar los píxeles. El componente activo es un transistor de película delgada (TFT) que se fabrica directamente en una pieza de vidrio. La otra pieza de vidrio tiene un recubrimiento uniforme para formar un plano posterior. Las líneas de control para estos transistores se distribuyen en filas y columnas entre los píxeles. Estas pantallas se han hecho posibles gracias a la tecnología que permite fabricar estos transistores en una matriz sobre una película delgada del tamaño de una pantalla de computadora portátil. Este tipo de pantallas proporcionan una respuesta mucho más rápida y una mayor resolución. El uso de la tecnología de polisilicio permite integrar los circuitos controladores en la unidad de visualización, con lo cual se reducen los problemas de conexión y se requiere un espacio perimetral muy pequeño alrededor de la LCD.

Hay otras tecnologías de visualización que se están refinando, incluyendo la fluorescencia en vacío, el plasma de descarga de gas y la electroluminiscencia. La física óptica para cada una de estas tecnologías varía, pero el medio para controlarlas todas es el mismo. Un sistema digital debe activar una fila y una columna de una matriz para poder controlar la cantidad de luz en el píxel ubicado en la intersección de esa fila/columna.

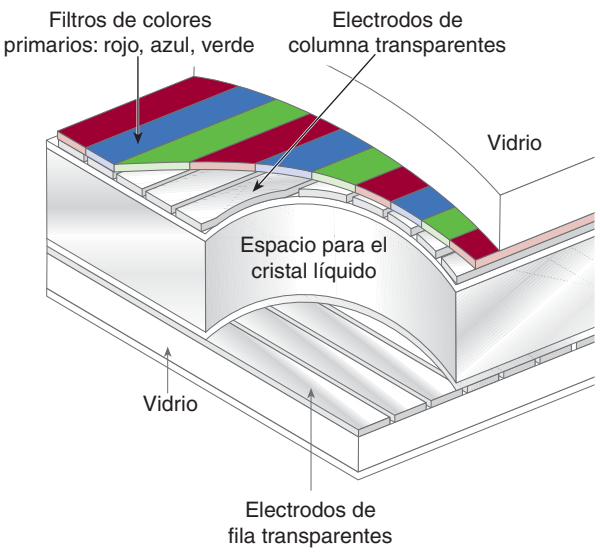


FIGURA 9-11 Un panel LCD de matriz activa.

PREGUNTAS DE REPASO

1. Indique cuáles de los siguientes enunciados se refieren a las pantallas LCD y cuáles se refieren a las pantallas de LEDs.
   (a) Emiten luz.
   (b) Reflejan la luz ambiental.
   (c) Son mejores para las aplicaciones de baja potencia.
   (d) Requieren un voltaje de corriente alterna.
   (e) Usan un arreglo de 7 segmentos para producir dígitos.
   (f) Requieren resistencias limitadoras de corriente.
2. ¿Qué forma de datos se envía a cada uno de los siguientes dispositivos?
   (a) Una pantalla LCD de 7 segmentos con un decodificador/controlador.
   (b) Un módulo LCD alfanumérico.
   (c) Una pantalla de computadora LCD.

9-4 CODIFICADORES

La mayoría de los decodificadores acepta un código de entrada y produce un nivel ALTO (o BAJO) en *una y sólo una* línea de salida. En otras palabras, podemos decir que un decodificador identifica, reconoce o detecta un código específico. Al opuesto de este proceso de decodificación se le conoce como **codificación** y se lleva a cabo mediante un circuito lógico llamado **codificador**, el cual tiene cierto número de líneas de entrada, de las cuales sólo una se activa en un momento dado y produce un código de salida de N bits, dependiendo de la entrada que se active. La figura 9-12 es el diagrama general para un codificador con M entradas y N salidas. En este codificador las entradas son activas en ALTO, lo cual significa que por lo general están en BAJO.

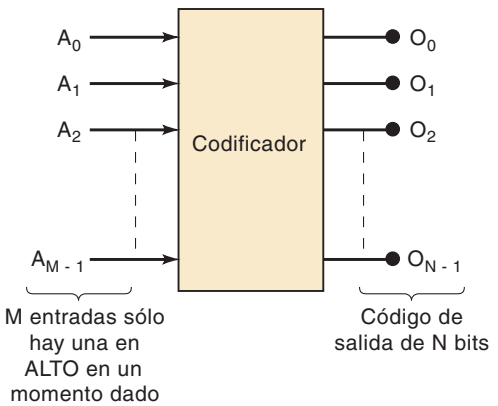
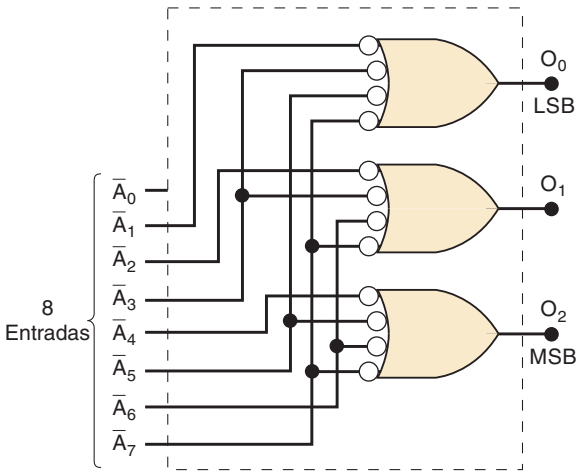


FIGURA 9-12 Diagrama general de un codificador.

Anteriormente vimos que un *decodificador de binario a octal (decodificador de 3 a 8 líneas)* acepta un código de entrada de tres bits y activa uno de ocho líneas de salida correspondientes a ese código. Un *codificador de octal a binario (codificador de 8 a 3 líneas)* realiza la función opuesta: acepta ocho líneas de entrada y produce un código de salida de tres bits, correspondiente a la entrada que se activó. La figura 9-13 muestra el circuito lógico y la tabla de verdad para un codificador de octal a binario con entradas activas en BAJO.

Si rastrea la lógica podrá verificar que un nivel BAJO en cualquier entrada individual producirá el código binario de salida correspondiente a esa entrada. Por



Entradas								Salidas		
$\overline{A}_0$	$\overline{A}_1$	$\overline{A}_2$	$\overline{A}_3$	$\overline{A}_4$	$\overline{A}_5$	$\overline{A}_6$	$\overline{A}_7$	O_2	O_1	O_0
X	1	1	1	1	1	1	1	0	0	0
X	0	1	1	1	1	1	1	0	0	1
X	1	0	1	1	1	1	1	0	1	0
X	1	1	0	1	1	1	1	0	1	1
X	1	1	1	0	1	1	1	1	0	0
X	1	1	1	1	0	1	1	1	0	1
X	1	1	1	1	1	0	1	1	1	0
X	1	1	1	1	1	1	0	1	1	1

FIGURA 9-13 Circuito lógico para un codificador de octal a binario (de 8 a 3 líneas). Para una operación apropiada, sólo debe haber una entrada activa en un momento dado.

ejemplo, un nivel BAJO en $\overline{A}_3$ (mientras que las demás entradas se encuentran en ALTO) producirá $O_2 = 0$, $O_1 = 1$ y $O_0 = 1$, que es el código binario para el 3. Observe que $\overline{A}_0$ no se conecta a las compuertas lógicas, ya que por lo general las salidas del codificador estarán en 000 cuando ninguna de las entradas de $\overline{A}_1$ hasta $\overline{A}_9$ esté en BAJO.

EJEMPLO 9-5

Determine las salidas del codificador de la figura 9-13 cuando $\overline{A}_3$ y $\overline{A}_5$ están al mismo tiempo en BAJO.

Solución

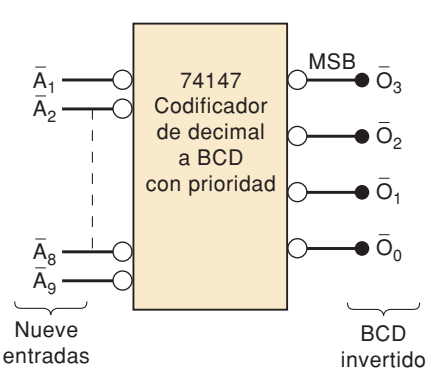
Si seguimos la ruta de las compuertas lógicas, podremos ver que los niveles BAJOs en estas dos entradas producirán niveles ALTOs en cada salida; en otras palabras, el código binario 111. Es evidente que éste no es el código para ninguna de las entradas que se activaron.

Codificadores con prioridad

Este último ejemplo identifica una desventaja del circuito codificador simple de la figura 9-13 cuando se activa más de una entrada a la vez. Una versión modificada de este circuito, conocida como **codificador con prioridad**, incluye la lógica necesaria para asegurar que cuando se activen dos o más entradas, el código de salida corresponda a la entrada de mayor numeración. Por ejemplo, cuando $\overline{A}_3$ y $\overline{A}_5$ están en BAJO, el código de salida será 101 (5). De manera similar, cuando $\overline{A}_6$, $\overline{A}_2$ y $\overline{A}_0$ estén todas en BAJO, el código de salida será 110 (6). Los circuitos 74148, 74LS148 y 84HC148 son todos codificadores de octal a binario con prioridad.

74147 Codificador de decimal a BCD con prioridad

La figura 9-14 muestra el símbolo lógico y la tabla de verdad para el 74147 (74LS147, 74HC147), el cual funciona como un codificador de decimal a BCD con prioridad. Tiene nueve entradas activas en BAJO, las cuales representan los dígitos decimales del 1 al 9, y produce el código BCD *invertido* que corresponde a la entrada de mayor numeración que se activó.



$\overline{A}_1$	$\overline{A}_2$	$\overline{A}_3$	$\overline{A}_4$	$\overline{A}_5$	$\overline{A}_6$	$\overline{A}_7$	$\overline{A}_8$	$\overline{A}_9$	$\overline{O}_3$	$\overline{O}_2$	$\overline{O}_1$	$\overline{O}_0$
1	1	1	1	1	1	1	1	1	1	1	1	1
X	X	X	X	X	X	X	X	0	0	1	1	0
X	X	X	X	X	X	X	0	1	0	1	1	1
X	X	X	X	X	X	0	1	1	1	0	0	0
X	X	X	X	X	0	1	1	1	1	0	0	1
X	X	X	X	0	1	1	1	1	1	0	1	0
X	X	X	0	1	1	1	1	1	1	0	1	1
X	X	0	1	1	1	1	1	1	1	1	0	0
X	0	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	0

X = ya sea 0 o 1

FIGURA 9-14 Codificador 74147 de decimal a BCD con prioridad.



Vamos a examinar la tabla de verdad para ver cómo funciona este CI. La primera línea en la tabla muestra a todas las entradas en su estado inactivo en ALTO. Para esta condición las salidas son 1111, que viene siendo el inverso de 0000, el código BCD para el 0. La segunda línea en la tabla indica que un nivel BAJO en $\overline{A}_9$, sin importar los estados de las demás entradas, producirá un código de salida de 0110, que viene siendo el inverso de 1001, el código BCD para el 9. La tercera línea muestra que un nivel BAJO en $\overline{A}_8$, siempre y cuando $\overline{A}_9$ esté en ALTO, producirá un código de salida de 0111, el inverso de 1000, el código BCD para el 8. De manera similar, las líneas restantes en la tabla muestran que un nivel BAJO en cualquier entrada, siempre y cuando todas las entradas de mayor numeración estén en ALTO, producirá el inverso del código BCD para esa entrada.

Las salidas del 74147 estarán, por lo general, en ALTO cuando ninguna de las entradas esté activada. Esto corresponde a la condición de entrada de 0 decimal. No hay entrada A_0 debido a que el codificador asume el estado de entrada de 0 decimal cuando todas las demás entradas están en ALTO. Para convertir las salidas BCD invertidas del 74147 en BCD normal se pasa cada una de ellas a través de un INVERSOR.

EJEMPLO 9-6

Determine los estados de las salidas en la figura 9-14 cuando $\overline{A}_5$, $\overline{A}_7$ y $\overline{A}_3$ están en BAJO y todas las demás entradas están en ALTO.

Solución

La tabla de verdad muestra que cuando $\overline{A}_7$ está en BAJO, los niveles en $\overline{A}_5$ y $\overline{A}_3$ no importan. Por lo tanto, las salidas serán cada una 1000, el inverso de 0111 (7).

Codificador interruptor

La figura 9-15 muestra cómo puede utilizarse un 74147 como un *codificador interruptor*. Los 10 interruptores podrían ser los interruptores del teclado en una calculadora que representan los dígitos del 0 al 9. Los interruptores son del tipo normalmente abierto, por lo que las entradas del interruptor están todas, por lo general, en ALTO y la salida BCD es 0000 (observe los INVERSORes). Cuando se oprima un dígito, el circuito producirá el código BCD para ese dígito. Como el 74LS147 es un codificador con prioridad, si se oprimen varias teclas al mismo tiempo se producirá el código BCD para la tecla con mayor numeración.

El codificador interruptor de la figura 9-15 puede usarse siempre que deban introducirse datos BCD en forma manual en un sistema digital. Uno de los mejores ejemplos sería una calculadora electrónica, en la cual el operador oprime varios interruptores de teclado en forma sucesiva para introducir un número decimal. En una calculadora básica simple, el código BCD para cada dígito decimal se introduce en un registro de almacenamiento de cuatro bits. En otras palabras, cuando se oprime la primera tecla se envía el código BCD para ese dígito a un registro de FFs

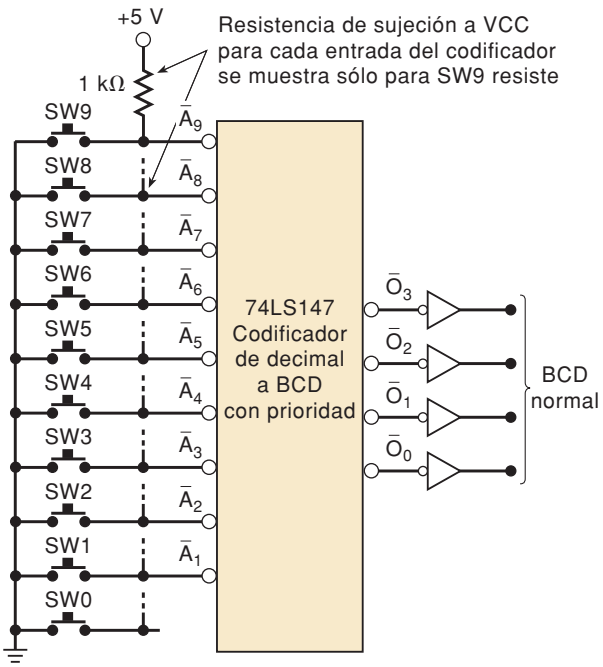


FIGURA 9-15 Codificador interruptor de decimal a BCD.

de cuatro bits; cuando se oprime el segundo interruptor envía el código BCD para ese dígito a *otro* registro de FFs de cuatro bits, y así en lo sucesivo. Por ende, una calculadora que pueda manejar ocho dígitos tendrá ocho registros de cuatro bits para almacenar los códigos BCD para estos dígitos. Cada registro de cuatro bits controla un decodificador/controlador y una pantalla numérica de manera que pueda visualizarse el número de ocho dígitos.

La operación descrita puede lograrse con el circuito de la figura 9-16. Este circuito tomará tres dígitos decimales que se introducen desde el teclado en secuencia, los codificará en BCD y almacenará el BCD en tres registros de salida de FF. Los 12 flip-flops tipo *D* identificados como Q_0 a Q_{11} se utilizan para recibir y almacenar los códigos BCD para los dígitos. Los flip-flops Q_8 a Q_{11} almacenan el código BCD para el dígito más significativo (MSD), que es el primero en introducirse desde el teclado. Los flip-flops Q_4 a Q_7 almacenan el segundo dígito introducido y los flip-flops Q_0 a Q_3 almacenan el tercer dígito introducido. Los flip-flops *X*, *Y* y *Z* forman un contador de anillo (capítulo 7) que controla la transferencia de datos de las salidas del codificador al registro de salida apropiado. La compuerta OR produce una salida en ALTO cada vez que se oprime una de las teclas. Esta salida puede verse afectada por el rebote de los contactos de los interruptores, con lo cual se producirían varios pulsos antes de establecerse en el estado ALTO. El OS se utiliza para neutralizar el rebote de los interruptores mediante un disparo en la primera transición positiva desde la compuerta OR y permaneciendo en nivel ALTO durante 20 ms, mucho más allá del tiempo que dura el rebote del interruptor. La salida del OS aplica pulsos de reloj al contador de anillo.

A continuación se describe la operación del circuito para el caso en el que se introduce el número 309 decimal:

1. Se oprime la tecla BORRAR, con lo cual se borran todos los flip-flops de almacenamiento Q_0 a Q_{11} para quedar en 0. También se borran los flip-flops *X* y *Y* y se prestablece el flip-flop *Z* en 1, de manera que el contador de anillo comience en el estado 001.
2. La tecla BORRAR se deja de oprimir y se oprime el "3". Las salidas 1100 del codificador se invierten para producir 0011, el código BCD para el 3. Estos valores binarios se envían a las entradas *D* de los tres registros de salida de cuatro bits.
3. La salida de la compuerta OR cambia a ALTO (debido a que dos de sus entradas están en ALTO) y dispara la salida del OS $Q = 1$ durante 20 ms. *Q* regresa a BAJO y aplica pulsos de reloj al contador de anillo hasta el estado 100 (*X*

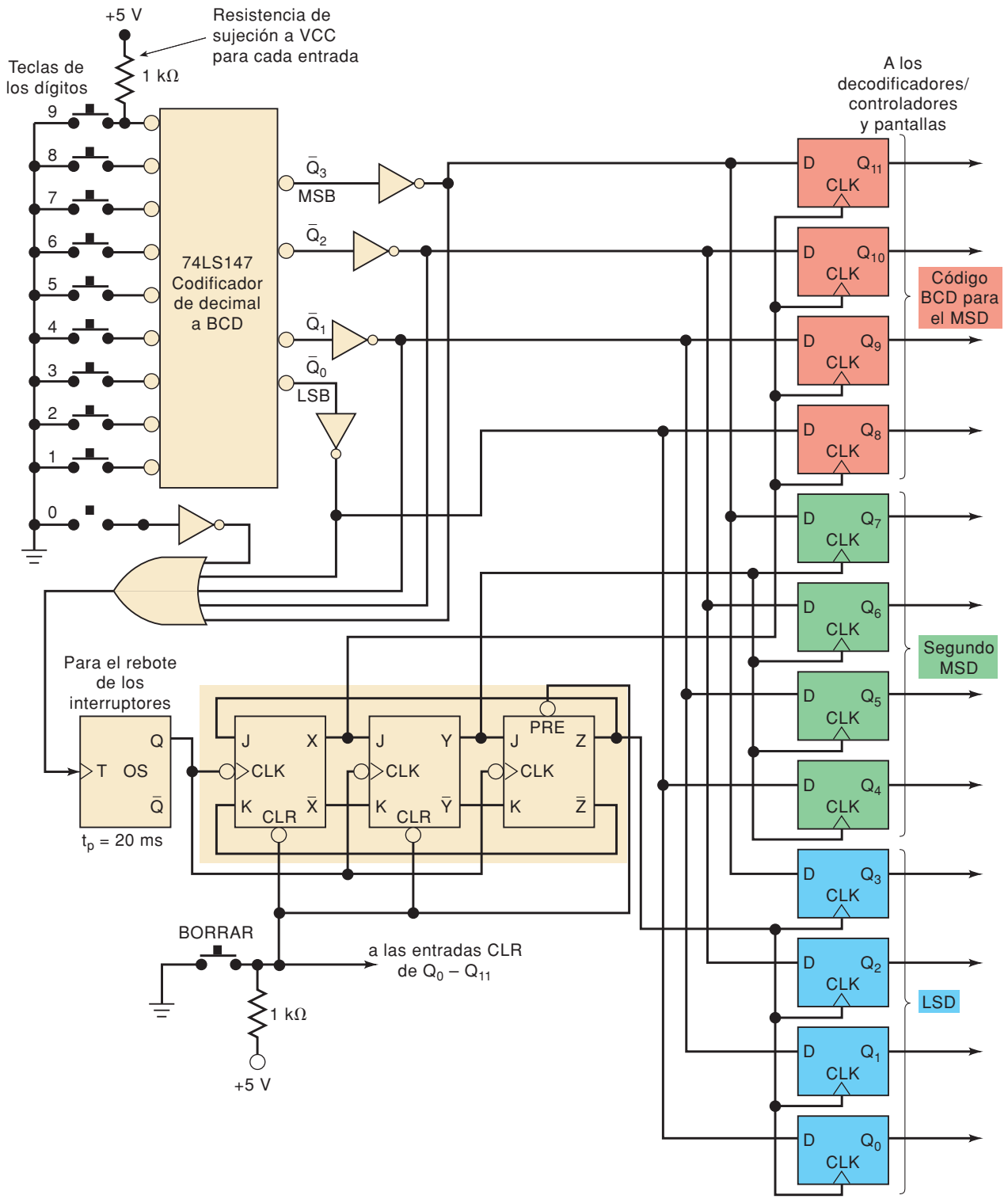


FIGURA 9-16 Circuito para introducir mediante el teclado un número de tres dígitos en los registros de almacenamiento.

cambia a ALTO). La transición positiva en *X* se alimenta a las entradas *CLK* de los flip-flops Q_8 a Q_{11}, de manera que las salidas del codificador se transfieran a esos FFs. Esto es, $Q_{11} = 0$, $Q_{10} = 0$, $Q_9 = 1$ y $Q_8 = 1$. Observe que los flip-flops Q_0 a Q_7 no se ven afectados, ya que sus entradas *CLK* no han recibido una transición positiva.

4. La tecla "3" se deja de oprimir y la salida de la compuerta OR regresa a BAJO. Después se oprime la tecla "0". Esto produce el código BCD de 0000, que se alimenta a las entradas de los tres registros.
5. La salida de la compuerta OR cambia a ALTO en respuesta a la tecla "0" (observe el INVERSOR) y dispara el OS durante 20 ms. Después de ese tiempo, el contador de anillo se desplaza al estado 010 (Y cambia a ALTO). La transición positiva en Y se alimenta a las entradas *CLK* de Q_4 a Q_7 y transfiere el 0000 a estos FFs. Observe que los flip-flops de Q_0 a Q_3 y de Q_8 a Q_{11} no se ven afectados por la transición en Y.
6. La tecla "0" se deja de oprimir y la salida de la compuerta OR regresa a BAJO. Se oprime la tecla "9" y se producen las salidas BCD 1001, las cuales se alimentan a los registros de almacenamiento.
7. La salida de la compuerta OR cambia a ALTO de nuevo y se dispara el OS, el cual a su vez aplica pulsos de reloj al contador de anillo para que cambie al estado 001 (Z cambia a ALTO). La transición positiva en Z se alimenta a las entradas *CLK* de Q_0 a Q_3 y se transfiere el 1001 hacia estos FFs. Los demás FFs de almacenamiento no se ven afectados.
8. En este punto, el registro de almacenamiento contiene 001100001001, comenzando con Q_{11}. Éste es el código BCD para el 309. Las salidas de estos registros alimentan los decodificadores/controladores que controlan las pantallas apropiadas para indicar los dígitos decimales 309.
9. Las salidas de los FFs de almacenamiento también se alimentan a otros circuitos en el sistema. Por ejemplo, en una calculadora estas salidas se enviarían a la sección aritmética para su procesamiento.

Varios problemas al final del capítulo tratarán con algunos otros aspectos de este circuito, incluyendo los ejercicios de diagnóstico de fallas.

El 74ALS148 es un poco más sofisticado que el '147. Tiene ocho entradas que están codificadas en un número binario de tres bits. Este CI también proporciona tres terminales de control, como se indica en la tabla 9-1. La entrada de habilitación ($\overline{EI}$) y la salida habilitación ($\overline{EO}$) pueden usarse para conectar en cascada dos CIs, para producir un codificador de hexadecimal a binario. La terminal $\overline{EI}$ debe estar en BAJO para que cualquier terminal de salida pueda cambiar a BAJO, y la terminal $\overline{EO}$ cambiará a BAJO sólo cuando ninguna de las ocho entradas esté activa y $\overline{EI}$ esté activa. La salida $\overline{GS}$ se utiliza para indicar cuando por lo menos está activada una de las ocho entradas. Hay que recalcar que las salidas de A_2 hasta A_9 están invertidas, como en el 74147.

TABLA 9-1 Tabla de funciones del 74ALS148.

	ENTRADAS								SALIDAS				
$\overline{EI}$	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$	$\overline{4}$	$\overline{5}$	$\overline{6}$	$\overline{8}$	$\overline{A}_2$	$\overline{A}_1$	$\overline{A}_0$	$\overline{GS}$	$\overline{EO}$
H	x	x	x	x	x	x	x	x	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
L	x	x	x	x	x	x	x	L	L	L	L	L	H
L	x	x	x	x	x	x	L	H	L	L	H	L	H
L	x	x	x	x	x	L	H	H	L	H	L	L	H
L	x	x	x	x	L	H	H	H	L	H	H	L	H
L	x	x	x	L	H	H	H	H	H	L	L	L	H
L	x	x	L	H	H	H	H	H	H	L	H	L	H
L	x	L	H	H	H	H	H	H	H	H	L	L	H
L	L	H	H	H	H	H	H	H	H	H	H	L	H

PREGUNTAS DE REPASO

1. ¿Qué diferencia hay entre un codificador y un decodificador?
2. ¿Qué diferencia hay entre un codificador con prioridad y un codificador ordinario?
3. ¿Cuáles serán las salidas en la figura 9-15 cuando SW6, SW5 y SW2 están todos cerrados?
4. Describa las funciones de cada una de las siguientes partes del circuito para introducir datos desde el teclado de la figura 9-16.
   (a) compuerta OR
   (b) codificador 74147
   (c) Monestable (one-shot)
   (d) Flip-flops X, Y, Z
   (e) Flip-flops Q_0 a Q_{11}
5. ¿Cuál es el propósito de cada entrada y salida de control en un codificador 74148?

9-5 DIAGNÓSTICO DE FALLAS

A medida que los circuitos y los sistemas se hacen más complejos, es obvio que se incrementa el número de posibles causas de falla. Mientras que el procedimiento para aislar y corregir fallas sigue siendo en esencia el mismo, la aplicación del proceso de **análisis/observación** es más importante para los circuitos complejos, ya que ayuda al técnico de diagnóstico de fallas a reducir la ubicación de la falla a un área pequeña del circuito. Con esto se reducen en una cantidad considerable los pasos de prueba y los datos resultantes que deben analizarse. Al comprender la operación del circuito, observar los síntomas de la falla y razonar a través de la operación, el técnico de diagnóstico puede a menudo predecir las posibles fallas antes de siquiera utilizar una sonda lógica o un osciloscopio. El proceso de observación/análisis es uno que los técnicos de diagnóstico sin experiencia no quieren aplicar, tal vez debido a la extensa variedad y capacidades del equipo de prueba moderno a su disposición. Es fácil volverse extremadamente confiado con esas herramientas y no utilizar en forma adecuada el razonamiento y las habilidades analíticas del cerebro humano.

Los siguientes ejemplos ilustran cómo puede aplicarse el proceso de observación/análisis. Muchos de los problemas de diagnóstico de fallas al final del capítulo le brindarán la oportunidad de desarrollar su habilidad para aplicar este proceso.

Otra de las estrategias vitales en el diagnóstico de fallas es lo que se denomina como **dividir y conquistar**. Esta estrategia se utiliza para identificar la ubicación del problema una vez que el proceso de observación/análisis ha generado varias posibilidades. Un método menos eficiente sería investigar todas las causas posibles, una por una. El método de dividir y conquistar busca un punto en el circuito que pueda probarse, dividiendo así el número total de posibles causas a la mitad. En los sistemas simples esto podría parecer innecesario, pero a medida que aumenta la complejidad también se incrementa el número de posibles causas. Si hay ocho posibles causas, entonces debe realizarse una prueba que elimine cuatro de ellas. La siguiente prueba debería eliminar dos más y la tercera debería identificar el problema.

EJEMPLO 9-7

Un técnico evalúa el circuito de la figura 9-4 mediante el uso de un conjunto de interruptores para aplicar el código de entrada desde A_4 hasta A_0. El técnico recorre cada uno de los posibles códigos de entrada y comprueba la correspondiente salida del decodificador para ver si está activada. Luego observa que todas las salidas con numeración impar responden en forma correcta, pero todas las salidas con numeración par no responden cuando se aplica su código. ¿Cuáles son las fallas más probables?

Solución

En una situación en la que fallan tantas salidas, sería irrazonable esperar que cada una de estas salidas tenga una falla. Es mucho más probable que alguna otra condición de entrada fallida esté produciendo las fallas en las salidas. ¿Qué tienen en común todas las salidas con numeración par? En la tabla 9-2 se listan los códigos de entrada para varias de ellas.

TABLA 9-2

Salida	Código de entrada
$\overline{O}_0$	00000
$\overline{O}_4$	00100
$\overline{O}_{14}$	01110
$\overline{O}_{18}$	10010

Es evidente que cada salida con numeración par requiere un código de entrada con $A_0 = 0$ para poder activarse. Por ende, las fallas más probables serían las que evitan que A_0 cambie a BAJO. Éstas incluyen:

1. Un interruptor defectuoso conectado a la entrada A_0.
2. Una interrupción en la ruta entre el interruptor y la línea A_0.
3. Un corto externo de la línea A_0 a V_{CC}.
4. Un corto interno con V_{CC} en las entradas A_0 de cualquiera de los chips decodificadores

Por medio de la observación y el análisis, el técnico ha identificado varias causas posibles. Las causas potenciales 1 y 2 se encuentran en los interruptores que generan la dirección. Las causas 3 y 4 se encuentran en el circuito decodificador en sí. Podemos dividir el circuito si abrimos la conexión entre el interruptor menos significativo y la entrada A_0, como se muestra en la figura 9-17. Podemos utilizar una sonda lógica para ver si el interruptor puede generar un nivel BAJO al igual que un ALTO. Sin importar el resultado, hemos eliminado dos de las posibles causas.

En consecuencia, la falla se reduce a un área específica del circuito. La falla exacta puede rastrearse con las técnicas de prueba y medición con las que ya nos hemos familiarizado.

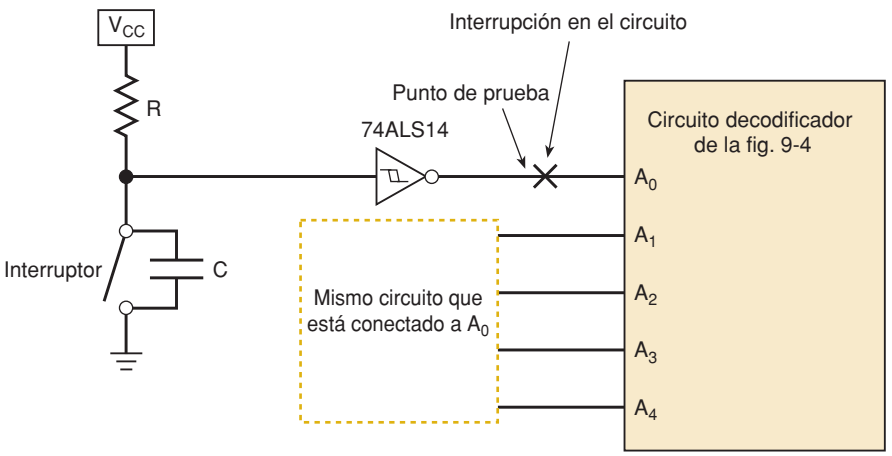


FIGURA 9-17
Diagnóstico de fallas en el circuito del ejemplo 9-7.

EJEMPLO 9-8

Un técnico alambra las salidas de un contador BCD a las entradas del decodificador/controlador de la figura 9-8. Aplica pulsos al contador a una velocidad muy lenta y observa la pantalla de LEDs, que se muestra a continuación, a medida que el contador avanza de 0000 a 1001. Examine esta secuencia observada con cuidado y trate de predecir la falla más probable.

CONTEO	0	1	2	3	4	5	6	7	8	9
Observado en pantalla										
Esperado en pantalla										

Solución

Si comparamos lo observado con lo esperado para cada conteo, podemos ver varios puntos importantes:

- Para los conteos en los que el dato observado es incorrecto, no es uno de los patrones de segmento que corresponden a conteos mayores que 1001.
- Esto descarta una falla en el contador o en los alambres que conectan al contador con el decodificador/controlador.
- Los patrones de segmento correctos (0, 1, 3, 6, 7 y 8) tienen la propiedad común de que los segmentos *e* y *f* están encendidos o apagados.
- Los patrones de segmento incorrectos tienen la propiedad común de que los segmentos *e* y *f* se encuentran en estados opuestos, y si intercambiamos los estados de estos dos segmentos se obtiene el patrón correcto.

Si analizamos estos puntos llegaremos a la conclusión de que es probable que el técnico haya "cruzado" las conexiones de los segmentos *e* y *f*.

9-6 MULTIPLEXORES (SELECTORES DE DATOS)

Un sistema estéreo doméstico moderno puede tener un interruptor que seleccione música de una de cuatro fuentes: una cinta de cassete, un disco compacto (CD), un sintonizador de radio o una entrada auxiliar tal como el audio de una VCR o un DVD. El interruptor selecciona una de las señales electrónicas de una de estas cuatro fuentes y la envía al amplificador de poder y las bocinas. En términos simples, esto es lo que hace un **multiplexor (MUX)**: selecciona una de varias señales de entrada y la pasa a la salida.

Un *multiplexor digital* o *selector de datos* es un circuito lógico que acepta varias entradas de datos digitales y selecciona una de ellas en un momento dado para pasarla a la salida. El enrutamiento de la entrada de datos deseada hacia la salida se controla mediante entradas de SELECCIÓN (también conocidos como entradas de DIRECCIÓN). La figura 9-18 muestra el diagrama funcional de un multiplexor digital en general. Las entradas y las salidas se dibujan como flechas gruesas en vez de líneas; esto indica que pueden tener más de una línea de señal.

El multiplexor actúa como un interruptor de múltiples posiciones controlado en forma digital, en el que el código digital que se aplica a las entradas de SELECCIÓN controla qué entradas de datos se conmutarán hacia la salida. Por ejemplo, la salida *Z* será igual a la entrada de datos I_0 para cierto código de entrada de SELECCIÓN específico, *Z* será igual a I_1 para otro código de entrada de SELECCIÓN específico, y así en lo sucesivo. Dicho de otra forma, un multiplexor selecciona 1 de *N* fuentes de datos de entrada y transmite los datos seleccionados a un solo canal de salida. A esto se le conoce como **multiplexaje**.

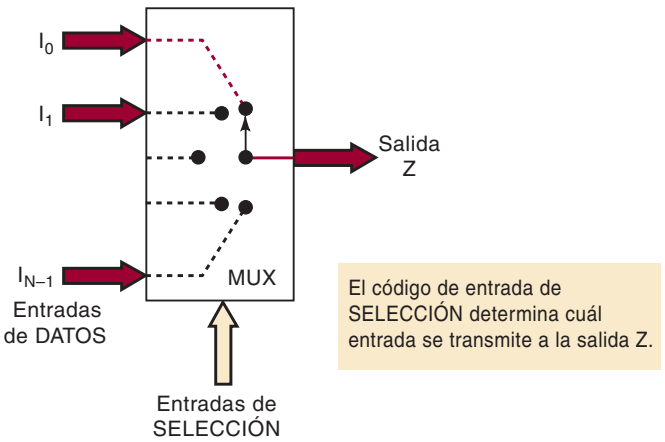


FIGURA 9-18
Diagrama funcional de un multiplexor digital (MUX).

Multiplexor básico de dos entradas

La figura 9-19 muestra el circuito lógico para un multiplexor con dos entradas de datos (I_0 e I_1) y la entrada de SELECCIÓN S. El nivel lógico que se aplica a la entrada S determina cuál compuerta AND está habilitada, de manera que su entrada de datos pase a través de la compuerta OR, a la salida Z. Si lo vemos de otra forma, la expresión Booleana para la salida es

$$Z = I_0\bar{S} + I_1S$$

Cuando $S = 0$, esta expresión se vuelve

$$\begin{aligned} Z &= I_0 \cdot 1 + I_1 \cdot 0 \\ &= I_0 \end{aligned} \qquad \text{[compuerta 2 habilitada]}$$

lo cual indica que Z será idéntica a la señal de entrada I_0, la que a su vez puede ser un nivel lógico fijo o una señal lógica que varía con el tiempo. Cuando $S = 1$, la expresión se vuelve

$$Z = I_0 \cdot 0 + I_1 \cdot 1 = I_1 \qquad \text{[compuerta 1 habilitada]}$$

lo cual nos muestra que la salida Z será idéntica a la señal de entrada I_1.

Un ejemplo en el que se podría utilizar un MUX de dos entradas es en un sistema digital con dos señales RELOJ MAESTRO distintas: un reloj de alta velocidad (por decir, 10-MHz) en un modo y un reloj de baja velocidad (por decir, 4.77-MHz) para el otro. Si utilizamos el circuito de la figura 9-19, el reloj de 10-MHz se conec-

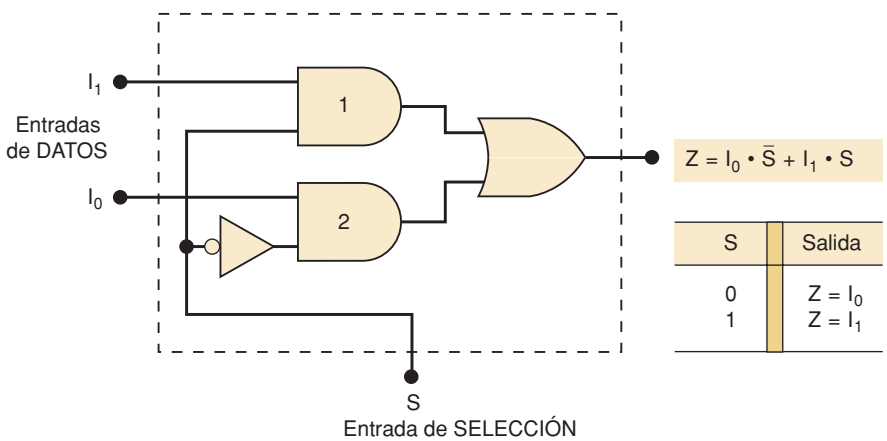


FIGURA 9-19
Multiplexor de dos entradas.



taría a I_0 y el reloj de 4.77-MHz se conectaría a I_1. Una señal de la sección lógica de control del sistema controlaría la entrada de SELECCIÓN para definir cuál de las señales de reloj debe aparecer en la salida Z para enrutarla a las demás partes del circuito.

Multiplexor de cuatro entradas

La misma idea básica puede utilizarse para formar el multiplexor de cuatro entradas que se muestra en la figura 9-20(a). Aquí, cuatro entradas se transmiten en forma selectiva a la salida, de acuerdo con las cuatro posibles combinaciones de las entradas de selección S_1S_0. Cada entrada de datos se conecta a una compuerta con una combinación distinta de niveles de entrada de selección. I_0 se conecta en una compuerta con $\overline{S}_1\overline{S}_0$, de manera que I_0 pasará a través de su compuerta AND hacia la salida Z sólo cuando $S_1 = 0$ y $S_0 = 0$. La tabla en la figura proporciona las salidas para los otros tres códigos de selección de entrada.

La figura 9-20(b) muestra otro circuito que realiza la misma función exacta. En este ejemplo se utilizan búferes triestado para seleccionar una de las señales. El decodificador asegura que sólo pueda estar un búfer habilitado a la vez. S_1 y S_0 se utilizan para especificar a cuál de las señales de entrada se le permite pasar a través de su búfer y llegar a la salida.

En las familias lógicas TTL y CMOS hay multiplexores de dos, cuatro, ocho y 16 entradas. Estos CIs básicos pueden combinarse para multiplexar un mayor número de entradas.

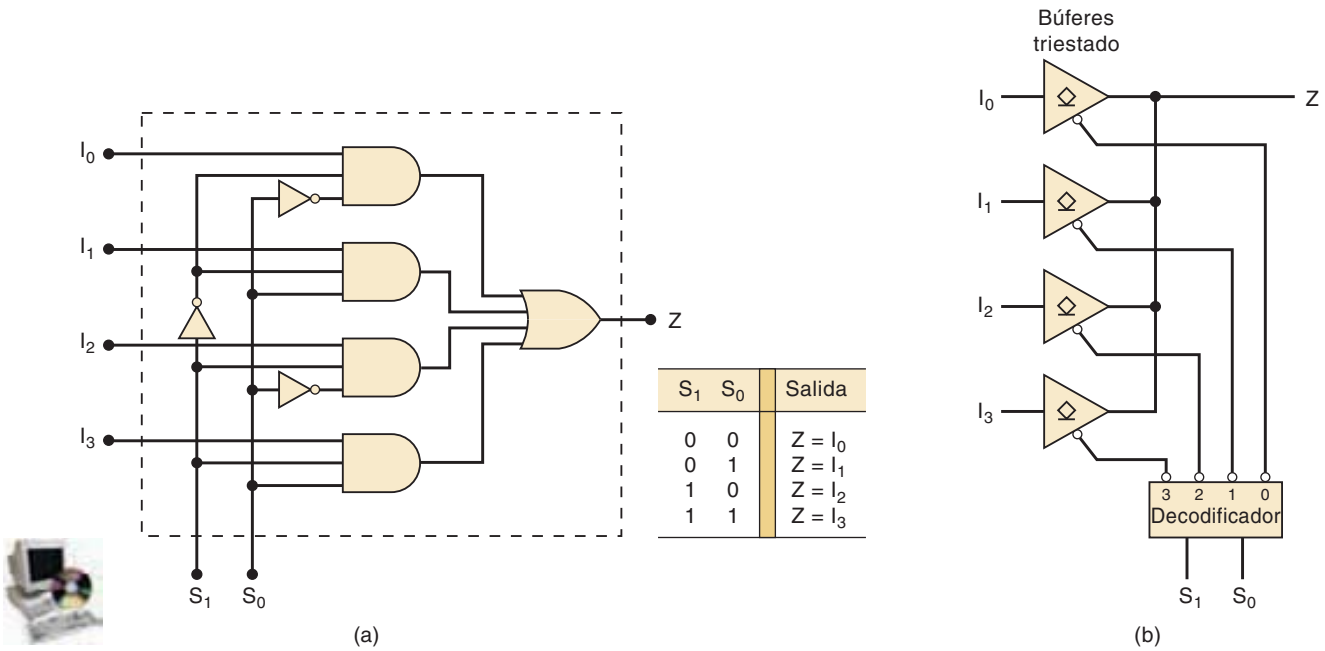
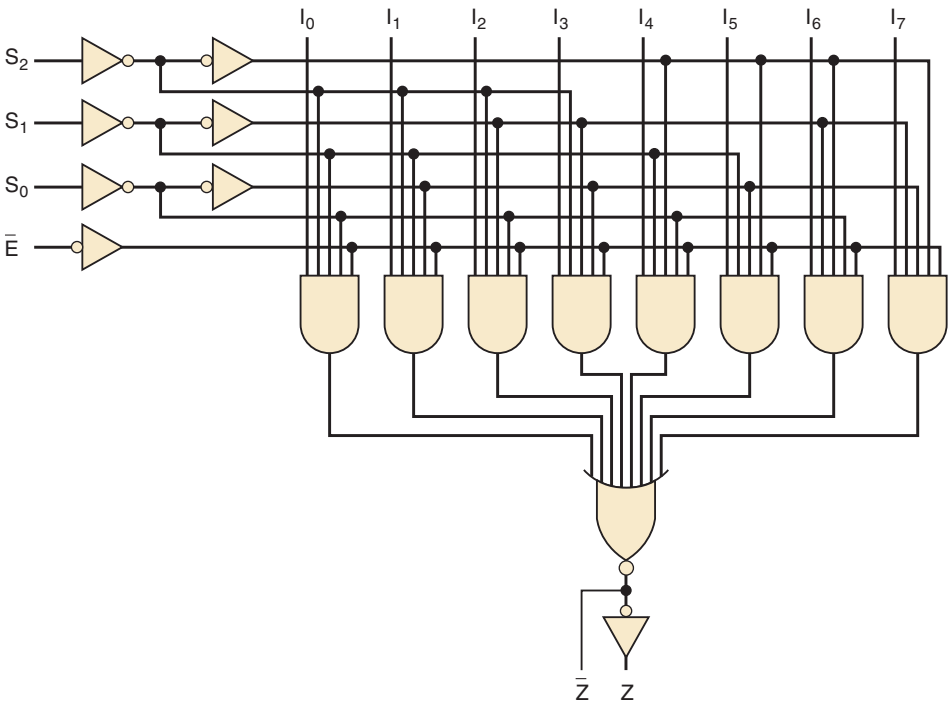


FIGURA 9-20 Multiplexor de cuatro entradas; (a) uso de la lógica de suma de productos; (b) uso de búferes triestado.

Multiplexor de ocho entradas

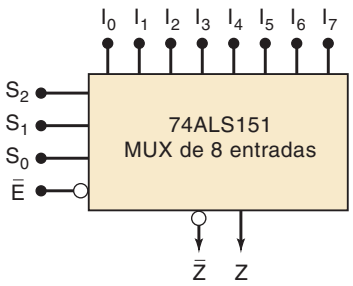
La figura 9-21(a) muestra el diagrama lógico para el multiplexor 74ALS151 (74HC151) de ocho entradas. Este multiplexor tiene una entrada de habilitación $\overline{E}$ y proporciona tanto la salida normal como la invertida. Cuando $\overline{E} = 0$, las entradas de selección $S_2S_1S_0$ seleccionarán una entrada de datos (de I_0 a I_7) para que pasen a la salida Z. Cuando $\overline{E} = 1$ el multiplexor se deshabilita, de manera que $Z = 0$ sin importar el código de entrada de selección. En la figura 9-21(b) se sintetiza esta operación, y en la figura 9-21(c) se muestra el símbolo lógico del 74151.



(a)

Entradas				Salidas	
$\bar{E}$	S_2	S_1	S_0	$\bar{Z}$	Z
H	X	X	X	H	L
L	L	L	L	$\bar{I}_0$	I_0
L	L	L	H	$\bar{I}_1$	I_1
L	L	H	L	$\bar{I}_2$	I_2
L	L	H	H	$\bar{I}_3$	I_3
L	H	L	L	$\bar{I}_4$	I_4
L	H	L	H	$\bar{I}_5$	I_5
L	H	H	L	$\bar{I}_6$	I_6
L	H	H	H	$\bar{I}_7$	I_7

(b)



(c)



FIGURA 9-21 (a) Diagrama lógico para el multiplexor 74ALS151; (b) tabla de verdad; (c) símbolo lógico.

EJEMPLO 9-9

El circuito en la figura 9-22 utiliza dos CIs 74HC151, un INVERSOR y una compuerta OR. Describa la operación de este circuito.

Solución

Este circuito tiene un total de 16 entradas de datos, de las cuales ocho se aplican a cada multiplexor. Las salidas de los dos multiplexores se combinan en la compuerta OR para producir una sola salida *X*. El circuito funciona como un multiplexor de 16 entradas. Las cuatro entradas de selección $S_3S_2S_1S_0$ seleccionarán una de las 16 entradas para que pase hacia *X*.

La entrada S_3 determina cuál multiplexor está habilitado. Cuando $S_3 = 0$, el multiplexor superior está habilitado y las entradas $S_2S_1S_0$ determinan cuál de sus entradas de datos aparecerá en su salida y pasará a través de la compuerta OR, hacia *X*. Cuando $S_3 = 1$, se habilita el multiplexor inferior y las entradas $S_2S_1S_0$ seleccionan una de sus entradas de datos para que pase a la salida *X*.

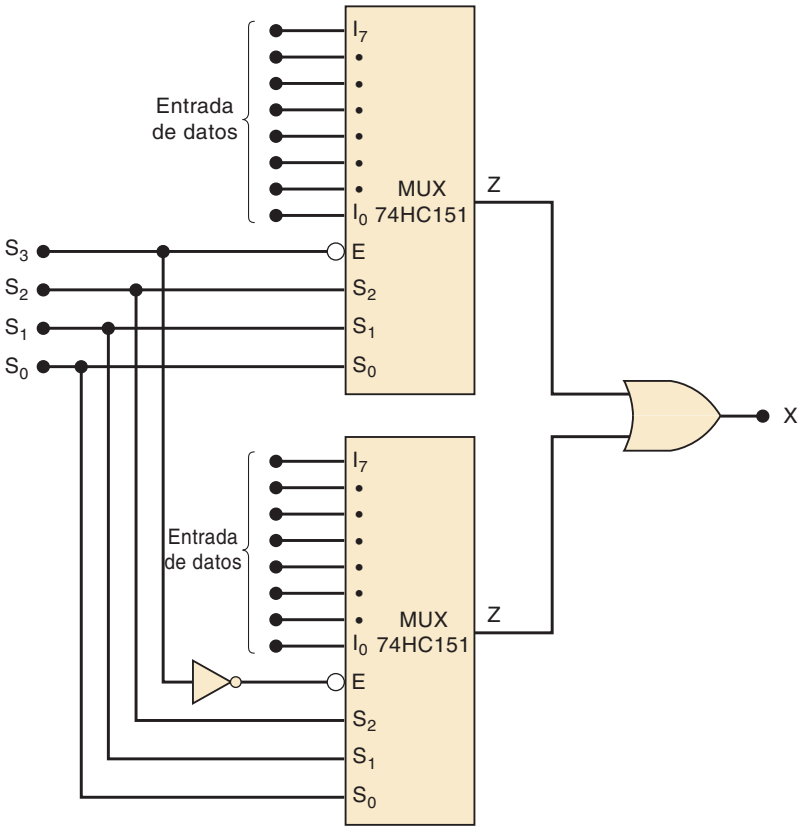


FIGURA 9-22 Ejemplo 9-9: dos CIs 74HC151 combinados para formar un multiplexor de 16 entradas.

MUX cuádruple de dos entradas (74ALS157/HC157)

El CI 74ALS157 es un multiplexor muy útil, el cual contiene cuatro multiplexores de dos entradas, como el que se muestra en la figura 9-19. El diagrama lógico para el 74ALS157 se muestra en la figura 9-23(a). Observe la forma en que se etiquetan las entradas y salidas de datos.

EJEMPLO 9-10

Determine las condiciones de entrada requeridas para que cada salida Z tome el nivel lógico de su correspondiente entrada I_0. Repita el proceso para I_1.

Solución

Primero que nada, la entrada de habilitación debe estar activa; esto es, $\overline{E} = 0$. Para que Z_a sea igual a I_{0a}, la entrada de selección debe estar en BAJO. Estas mismas condiciones producirán $Z_b = I_{0b}$, $Z_c = I_{0c}$ y $Z_d = I_{0d}$.

Cuando $\overline{E} = 0$ y $S = 1$, las salidas Z seguirán el conjunto de entradas I_1; esto es, $Z_a = I_{1a}$, $Z_b = I_{1b}$, $Z_c = I_{1c}$ y $Z_d = I_{1d}$.

Todas las salidas se deshabilitarán (en BAJO) cuando $\overline{E} = 1$.

Es útil pensar en este multiplexor como si fuera un simple multiplexor de dos entradas, pero uno en el cual cada entrada tenga cuatro líneas y que la salida tenga cuatro líneas. Las cuatro líneas de salida conmutan entre los dos conjuntos de cuatro líneas de entrada, bajo el control de la entrada de selección. Esta operación se representa mediante el símbolo lógico del 74ALS157 en la figura 9-23(b).

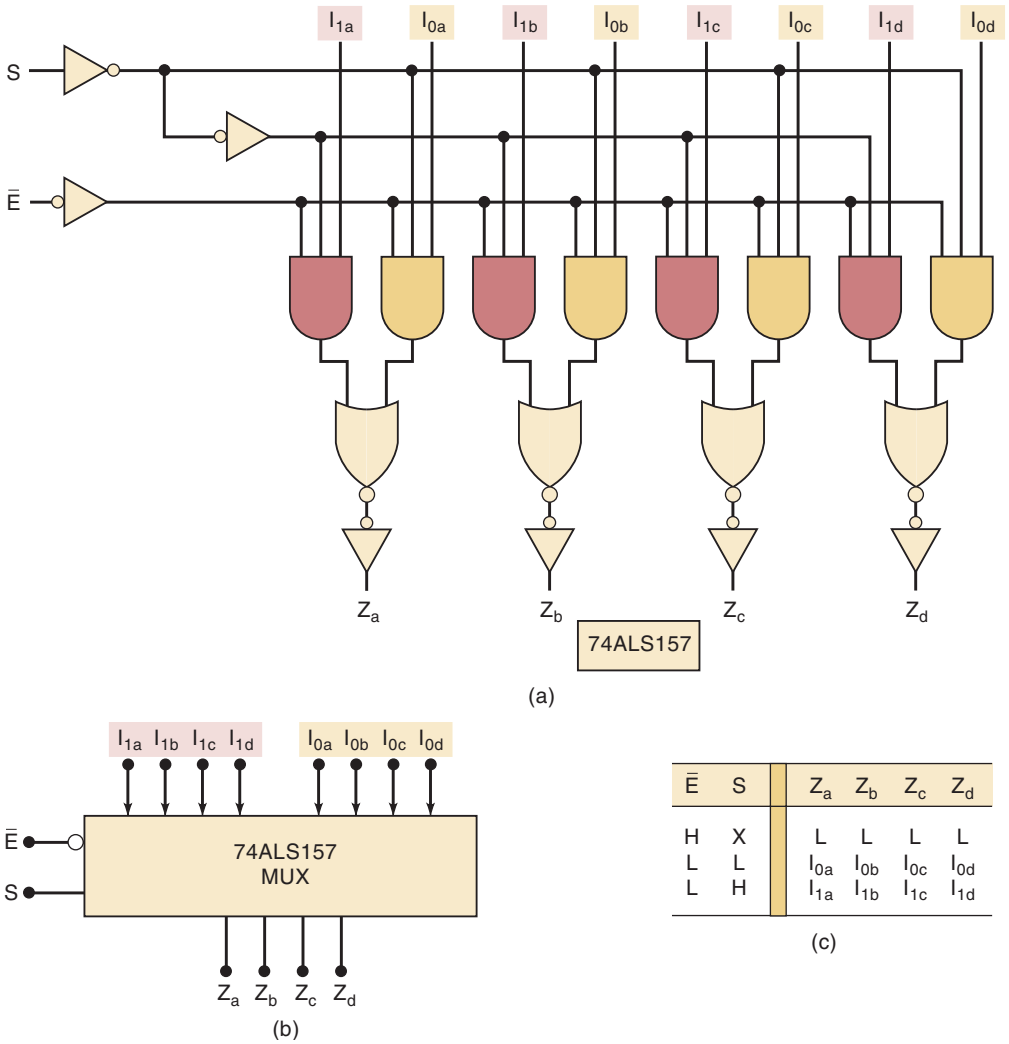


FIGURA 9-23 (a) Diagrama lógico para el multiplexor 74ALS157; (b) símbolo lógico; (c) tabla de verdad.

PREGUNTAS DE REPASO

1. ¿Cuál es la función de las entradas de selección de un multiplexor?
2. Cierto multiplexor puede conmutar una de 32 entradas de datos hacia su salida. ¿Cuántas entradas distintas tiene este MUX?

9-7 APLICACIONES DE LOS MULTIPLEXORES

Los circuitos multiplexores tienen muchas y variadas aplicaciones en los sistemas digitales de todo tipo. Entre estas aplicaciones están la selección de datos, el enrutamiento de datos, la secuencia de operaciones, la conversión de paralelo a serial, la generación de formas de onda y la generación de funciones lógicas. Aquí veremos algunas de estas aplicaciones y varias más en los problemas al final del capítulo.

Enrutamiento de datos

Los multiplexores pueden enrutar datos de una de varias fuentes hacia un destino. Una aplicación típica utiliza multiplexores 74ALS157 para seleccionar y mostrar el

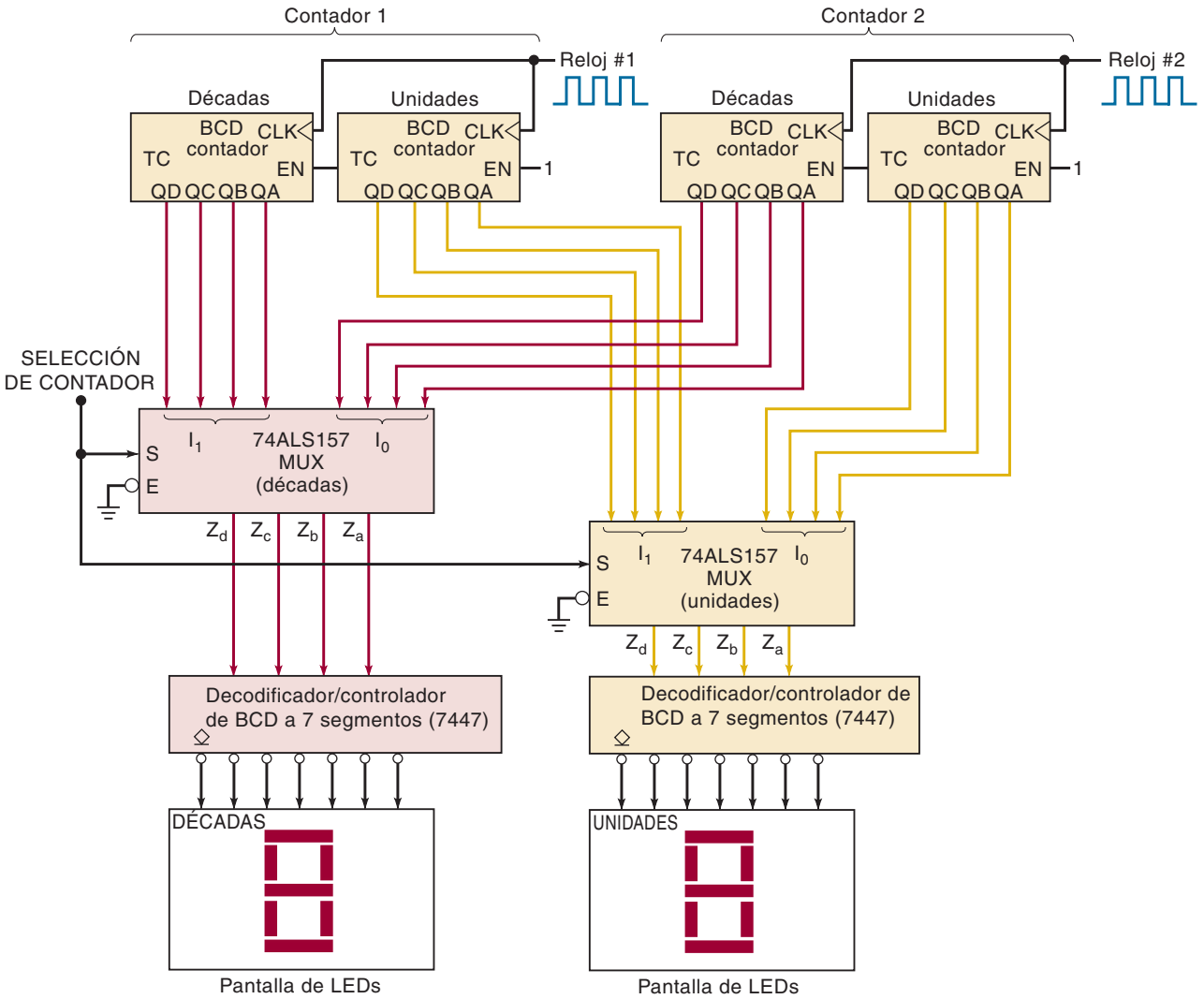


FIGURA 9-24 Sistema para visualizar dos contadores BCD de múltiples dígitos, uno a la vez.

contenido de uno de dos contadores BCD, mediante el uso de un conjunto *individual* de decodificadores/controladores y pantallas de LEDs. En la figura 9-24 se muestra el arreglo de este circuito.

Cada contador consiste de dos etapas BCD conectadas en cascada, y cada una de ellas se controla mediante su propia señal de reloj. Cuando la línea SELECCIÓN DE CONTADOR está en ALTO, las salidas del contador 1 podrán pasar a través de los multiplexores, hacia los decodificadores/controladores para visualizarse en las pantallas de LEDs. Cuando SELECCIÓN DE CONTADOR = 0, las salidas del contador 2 pasarán a través de los multiplexores, hacia las pantallas. De esta forma se visualizará el contenido decimal de un contador o del otro, bajo el control de la entrada SELECCIÓN DE CONTADOR. Una situación común en la que podría utilizarse este circuito es en un reloj digital. Los circuitos del reloj digital contienen muchos contadores y registros que llevan la cuenta de los segundos, minutos, horas, días, meses, configuración de las alarmas, etcétera. Un esquema de multiplexaje tal como éste permite visualizar diferentes datos en el número limitado de pantallas decimales.

El propósito de la técnica de multiplexaje, como la utilizamos aquí, es *compartir en el tiempo* los decodificadores/controladores y los circuitos de visualización entre los dos contadores, en vez de tener un conjunto separado de decodificadores/controladores y pantallas para cada contador. Esto produce un ahorro considerable en

el número de conexiones de alambres, en especial cuando se agregan más etapas BCD a cada contador. Lo que es más importante, representa una considerable disminución en el consumo de energía, ya que por lo general los decodificadores/controladores y las pantallas de LEDs consumen cantidades relativamente grandes de corriente de la fuente V_{CC}. Desde luego que esta técnica tiene la limitación de que sólo se puede visualizar el contenido de un contador a la vez. No obstante, en muchas aplicaciones esta limitación no es una desventaja. Podría utilizarse un arreglo de conmutación mecánica para realizar la función de conmutar primero un contador y después el otro con los decodificadores/controladores y las pantallas, pero el número de contactos requeridos para los interruptores, la complejidad del alambrado y el tamaño físico podrían constituir desventajas en comparación con el método completamente lógico de la figura 9-24.

Conversión de paralelo a serial

Muchos sistemas digitales procesan datos binarios en formato paralelo (todos los bits al mismo tiempo) porque es más rápido. No obstante, cuando los datos se van a transmitir a través de distancias relativamente grandes no es conveniente el arreglo en paralelo, ya que se requiere un gran número de líneas de transmisión. Por esta razón, es común convertir los datos binarios o la información en formato paralelo a



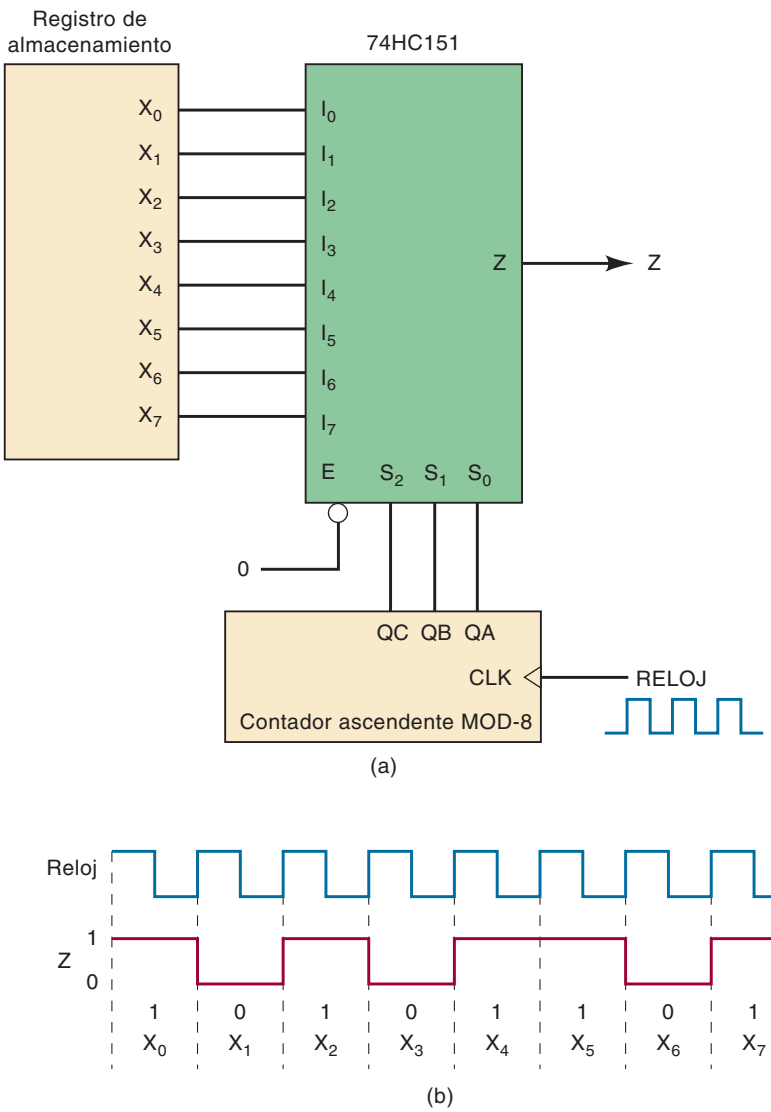


FIGURA 9-25
(a) Convertidor de paralelo a serial; (b) formas de onda para $X_7\,X_6X_5X_4X_3X_2X_1X_0 = 10110101$.

formato serial antes de transmitirlos a un destino remoto. Un método para realizar esta **conversión de paralelo a serial** utiliza un multiplexor, como se ilustra en la figura 9-25.

Los datos están presentes en formato paralelo, en las salidas del registro X y se alimentan al multiplexor de ocho entradas. Se utiliza un contador de tres bits (MOD-8) para proporcionar los bits de código de selección $S_2S_1S_0$ de manera que cuente desde 000 hasta 111 a medida que se aplican pulsos de reloj. De esta forma, la salida del multiplexor será X_7 durante el primer periodo de reloj, X_1 durante el segundo periodo de reloj, y así en lo sucesivo. La salida Z es una forma de onda que viene siendo una representación en serie de los datos de entrada en paralelo. Las formas de onda en la figura son para los casos en los que $X_7X_6X_5X_4X_3X_2X_1X_0 = 10110101$. Este proceso de conversión tarda un total de ocho ciclos de reloj. Observe que X_0 (el LSB) se transmite primero y X_7 (el MSB) se transmite al último.

Secuencia de las operaciones

El circuito de la figura 9-26 utiliza un multiplexor de ocho entradas como parte de un secuenciador de control que avanza a través de ocho pasos, cada uno de los cuales realiza cierta porción del proceso físico que se está controlando. Por ejemplo, este proceso podría ser uno en el que se mezclen dos ingredientes líquidos y después se cocine la mezcla. El circuito también utiliza un decodificador de 3 a 8 líneas y un contador binario MOD-8. La operación se describe de la siguiente manera.

1. Al principio el contador se restablece al estado 000. Las salidas del contador se alimentan a las entradas de selección del multiplexor y a las entradas del decodificador. Por lo tanto, la salida del decodificador $\overline{O}_0 = 0$ y todas las demás son 1, de manera que todas las entradas del ACTUADOR del proceso están en BAJO. Las salidas del SENSOR del proceso empiezan todas en BAJO. La salida del multiplexor $\overline{Z} = \overline{I}_0 = 1$, ya que las entradas S son 000.
2. El pulso INICIO comienza la operación secuencial al establecer el flip-flop Q_0 en ALTO, con lo cual el contador se lleva al estado 001. Esto hace que la salida del decodificador O_1 cambie a BAJO, con lo cual se activa el actuador 1; éste es el primer paso en el proceso (abrir la válvula de llenado 1).
3. Cierto tiempo después la salida 1 del SENSOR cambia a ALTO, indicando que se completó el primer paso (el interruptor flotador indica que el tanque está lleno). Este nivel ALTO está ahora presente en la entrada I_1 del multiplexor. Se invierte y llega a la salida $\overline{Z}$ porque el código de selección del contador es 001.
4. La transición a nivel BAJO en $\overline{Z}$ se alimenta a la entrada *CLK* del flip-flop Q_0. Esta transición negativa avanza el contador al estado 010.
5. La salida del decodificador $\overline{O}_2$ ahora cambia a BAJO, con lo cual se activa el actuador 2; éste es el segundo paso en el proceso (abrir la válvula de llenado 2). Ahora $\overline{Z}$ es igual a I_2 (el código de selección es 010). Como la salida 2 del SENSOR sigue estando en BAJO, $\overline{Z}$ cambiará a ALTO.
6. Cuando se completa el segundo paso del proceso la salida 2 del SENSOR cambia a ALTO, se produce un nivel BAJO en $\overline{Z}$ y el contador avanza a 011.
7. Esta misma acción se repite para cada uno de los otros pasos. Cuando se completa el séptimo paso la salida 7 del SENSOR cambia a ALTO, el contador avanza de 111 a 000 y permanece ahí hasta que otro pulso INICIO reinicie la secuencia.

Generación de funciones lógicas

Los multiplexores pueden utilizarse para implementar funciones lógicas en forma directa a partir de una tabla de verdad, sin necesidad de simplificación. Cuando se utiliza un multiplexor para este propósito, las entradas de selección se utilizan como variables lógicas y cada entrada de datos se conecta de manera permanente en ALTO o en BAJO según sea necesario para satisfacer la tabla de verdad.

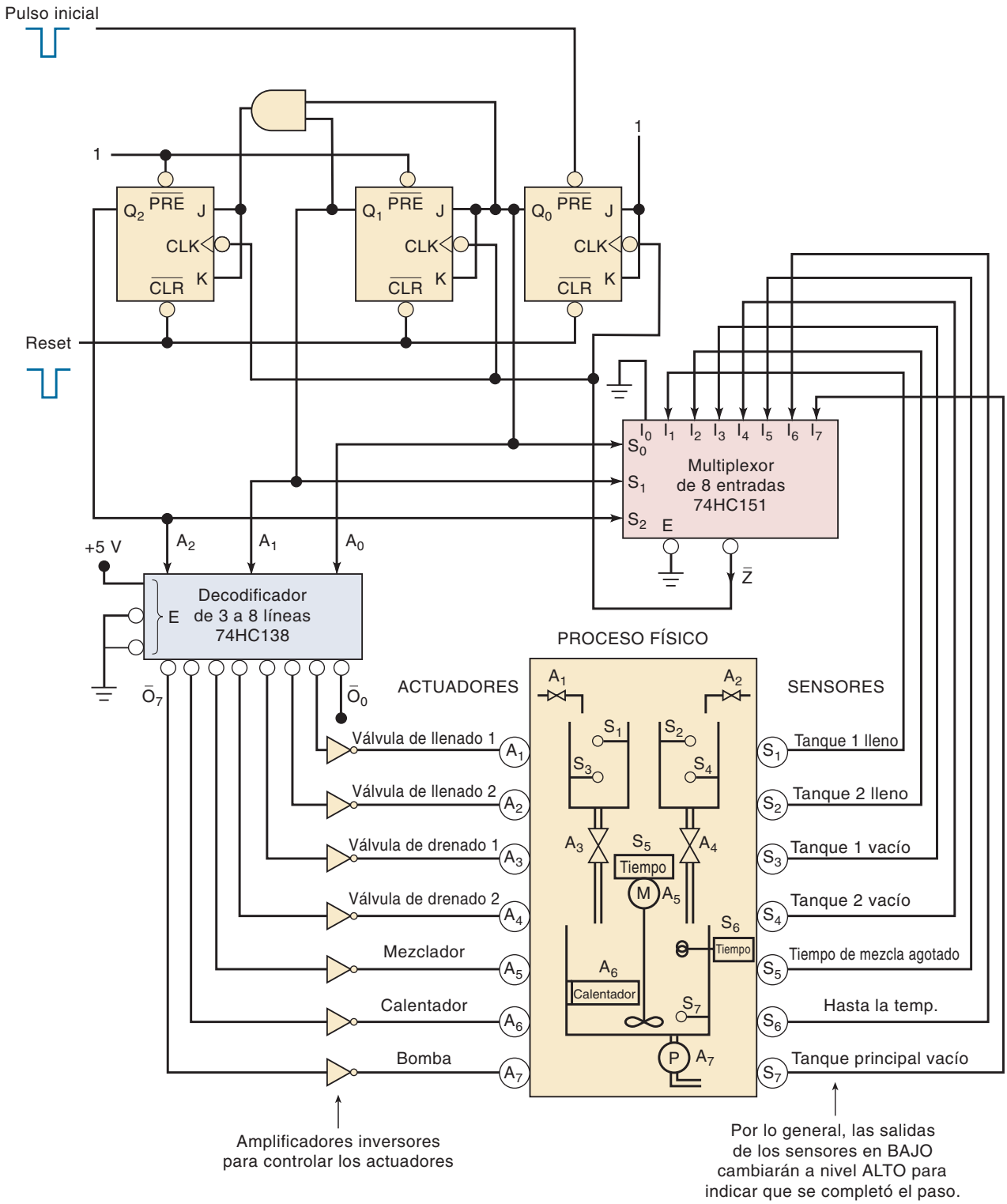
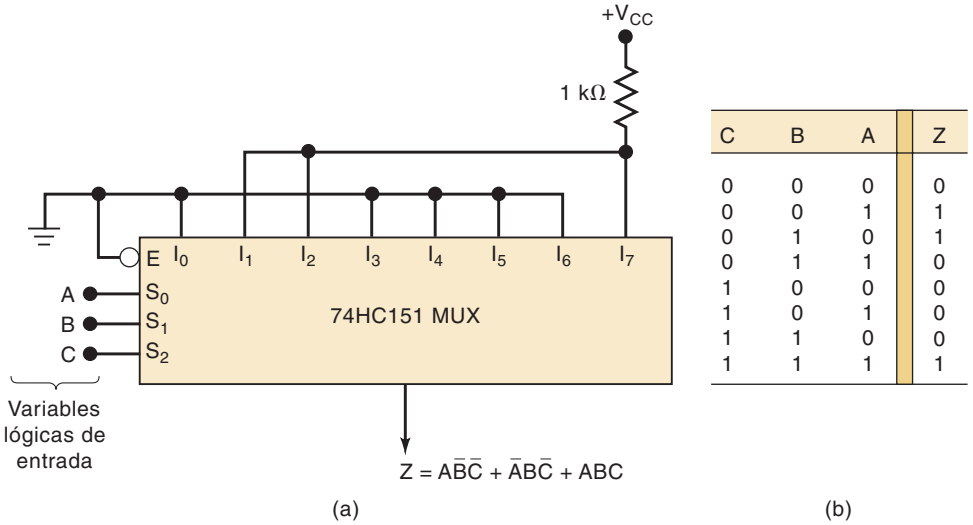


FIGURA 9-26 Secuenciador de control de siete pasos.

La figura 9-27 ilustra cómo puede usarse un multiplexor para implementar el circuito lógico que satisfaga la tabla de verdad dada. Las variables de entrada *A*, *B*, *C* se conectan con S_0,S_1,S_2, respectivamente, de manera que los niveles en estas entradas determinen cuáles datos de entrada deben aparecer en la salida *Z*. De acuerdo con la tabla de verdad, se supone que *Z* debe estar en BAJO cuando





C	B	A	Z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

FIGURA 9-27
Multiplexor que se utiliza para implementar una función lógica descrita por la tabla de verdad.

CBA = 000. Por ende, la entrada I_0 del multiplexor debe conectarse en BAJO. De igual forma, se supone que *Z* debe estar en BAJO para *CBA* = 011, 100, 101 y 110, por lo que las entradas I_3, I_4, I_5 e I_6 también deben conectarse en BAJO. Los otros conjuntos de condiciones de *CBA* deben producir *Z* = 1, por lo que las entradas I_1, I_2 e I_7 del multiplexor deben conectarse de manera permanente en ALTO.

Es fácil ver que puede implementarse cualquier tabla de verdad con tres variables con este multiplexor de ocho entradas. A menudo, este método de implementación es más eficiente que el uso de compuertas lógicas separadas. Por ejemplo, si podemos escribir la expresión de suma de productos para la tabla de verdad de la figura 9-27, tenemos que

$$Z = A\overline{B}\,\overline{C} + \overline{A}B\overline{C} + ABC$$

Esta ecuación *no puede* simplificarse ni por medios algebraicos, ni por el mapeo K, por lo que para implementar sus compuertas se requerirían tres INVERSORes y cuatro compuertas NAND, para un total de tres CIs.

Hay un método aún más eficiente para usar multiplexores para implementar funciones lógicas. Este método permite al diseñador lógico utilizar un multiplexor con tres entradas de selección (por ejemplo, un 74HC151) para implementar una función lógica de *cuatro variables*. En el problema 9-37 presentaremos este método.

El concepto más importante que podemos obtener del uso de un MUX para implementar una expresión de suma de productos es el hecho de que la función lógica puede cambiarse muy fácilmente, con sólo modificar los 1s y 0s en las entradas del MUX. En otras palabras, es muy fácil utilizar un MUX como un dispositivo lógico programable (PLD). Muchos PLDs utilizan esta estrategia en bloques de hardware, a los que, por lo general, se les conoce como tablas de búsqueda (LUTs). En los capítulos 12 y 13 hablaremos con más detalle acerca de las tablas de búsqueda.

PREGUNTAS DE REPASO

1. ¿Cuáles son algunas de las principales aplicaciones de los multiplexores?
2. *Verdadero o falso:* cuando se utiliza un multiplexor para implementar una función lógica, las variables lógicas se aplican a las entradas de datos del multiplexor.
3. ¿Qué tipo de circuito proporciona las entradas de selección cuando un MUX se utiliza como convertidor de paralelo a serial?

9-8 DEMULTIPLEXORES (DISTRIBUIDORES DE DATOS)

Un multiplexor recibe varias entradas y transmite *una* de ellas a la salida. Un **demultiplexor** (**DEMUX**) realiza la operación inversa: recibe una sola entrada y la distribuye a través de varias salidas. La figura 9-28 muestra el diagrama funcional para un demultiplexor digital. Las flechas gruesas para las entradas y las salidas pueden representar una o más líneas. El código de entrada de selección determina cuál de las salidas transmitirá la entrada de DATOS. En otras palabras, el demultiplexor recibe una fuente de datos de entrada y la distribuye en forma selectiva a 1 de N canales de salida, justo igual que un interruptor de múltiples posiciones.

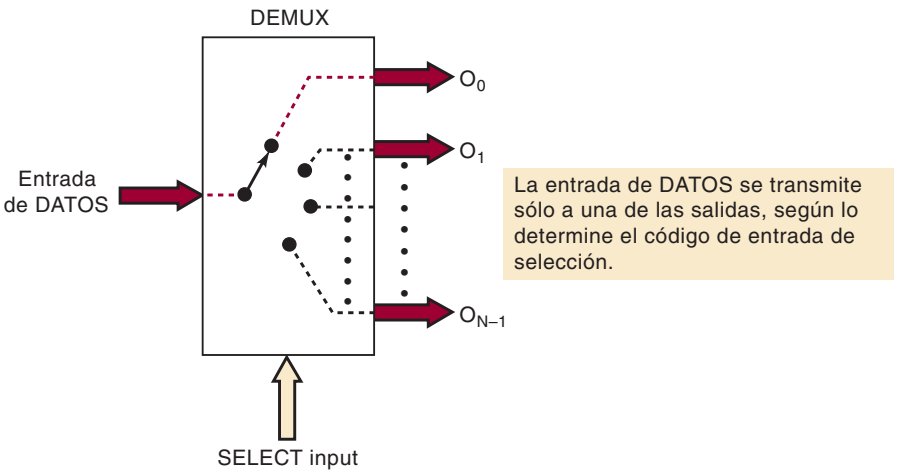


FIGURA 9-28
Demultiplexor general.

Demultiplexor de 1 a 8 líneas

La figura 9-29 muestra el diagrama lógico para un demultiplexor que distribuye una línea de entrada a ocho líneas de salida. La única línea de entrada de datos I se conecta a todas las ocho compuertas AND, pero sólo una de estas compuertas se habilitará mediante las líneas de entrada de SELECCIÓN. Por ejemplo, cuando $S_2S_1S_0 = 000$, sólo se habilitará la compuerta AND 0 y la entrada de datos I aparecerá en la salida O_0. Otros códigos de SELECCIÓN hacen que la entrada I llegue a las demás salidas. La tabla de verdad sintetiza la operación.

El circuito demultiplexor de la figura 9-29 es muy similar al circuito decodificador de 3 a 8 líneas de la figura 9-2, sólo que se agregó una cuarta entrada (I) en cada compuerta. Anteriormente señalamos que muchos decodificadores de CI tienen una entrada de HABILITACIÓN, la cual es una entrada extra que se agrega a las compuertas del decodificador. Por lo tanto, este tipo de circuito integrado decodificador puede utilizarse como demultiplexor, en donde las entradas de código binario (por ejemplo, A, B, C en la figura 9-2) sirven como entradas de SELECCIÓN y la entrada de HABILITACIÓN sirve como la entrada de datos I. Por esta razón, es común que los fabricantes de CIs llamen a este tipo de dispositivo un *decodificador/demultiplexor*, el cual puede usarse para cualquier función.

Anteriormente vimos cómo se utiliza el 74ALS138 como decodificador de 1 a 8. La figura 9-30 muestra cómo se puede utilizar como demultiplexor. La entrada de habilitación E_1 se utiliza como la entrada de datos I, mientras que las otras dos entradas de habilitación se mantienen en sus estados activos. Las entradas $A_2A_1A_0$ se utilizan como el código de selección. Para ilustrar la operación, vamos a suponer que las entradas de selección son 000. Con este código de entrada, la única salida que puede activarse es $\overline{O}_0$, mientras que todas las demás entradas están en ALTO. $\overline{O}_0$ cambiará a BAJO sólo si $\overline{E}_1$ cambia a BAJO y cambiará a ALTO si $\overline{E}_1$ cambia a ALTO. En otras palabras, $\overline{O}_0$ *seguirá a la señal en* $\overline{E}_1$ (es decir, la entrada de datos I) mientras que todas las demás salidas permanecerán en ALTO. De manera similar, si se aplica un código de selección distinto a $A_2A_1A_0$ la salida correspondiente seguirá a la entrada de datos, I.



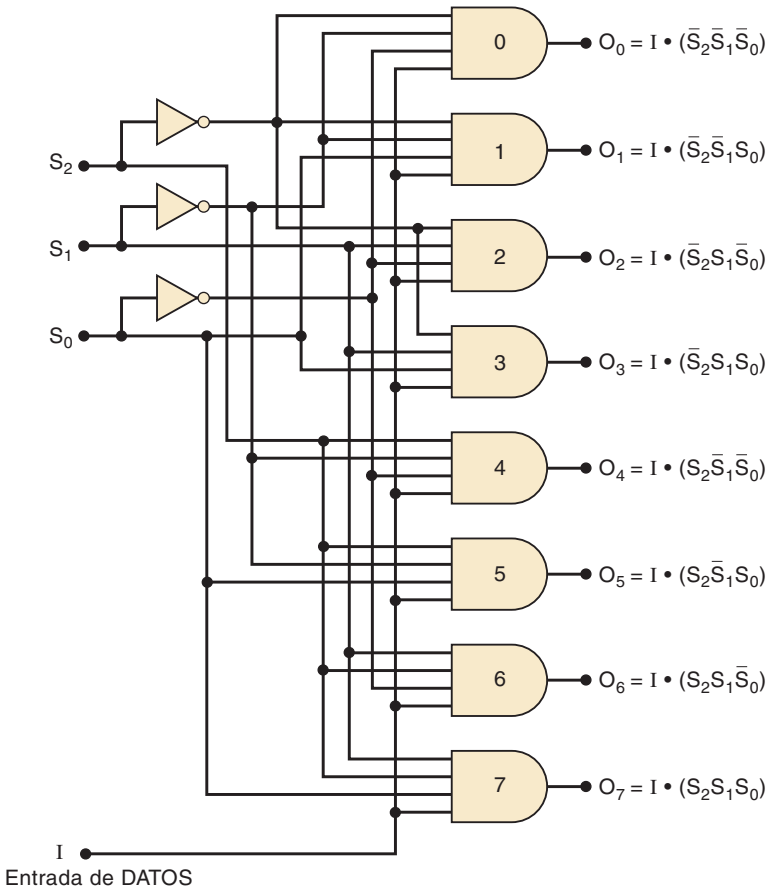


FIGURA 9-29
Demultiplexor de 1 a 8 líneas.

Código de SELECCIÓN S_2	S_1	S_0	SALIDAS O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	0	0	0	0	0	0	0	I
0	0	1	0	0	0	0	0	0	I	0
0	1	0	0	0	0	0	0	I	0	0
0	1	1	0	0	0	0	I	0	0	0
1	0	0	0	0	0	I	0	0	0	0
1	0	1	0	0	I	0	0	0	0	0
1	1	0	0	I	0	0	0	0	0	0
1	1	1	I	0	0	0	0	0	0	0

Nota: I es la entrada de datos

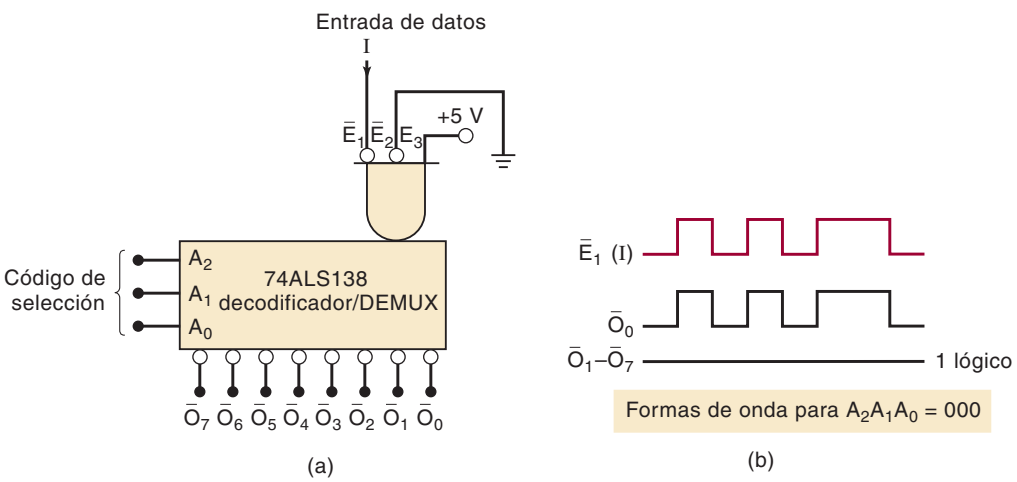


FIGURA 9-30 (a) El decodificador 74ALS138 puede funcionar como un demultiplexor, en donde E_1 se utiliza como la entrada de datos; (b) las formas de onda típicas para un código de selección de $A_2A_1A_0$ = 000 muestran que O_0 es idéntica a la entrada de datos I en E_1.

La figura 9-30(b) muestra las formas de onda típicas para el caso en el que $A_2A_1A_0 = 000$ selecciona la salida $\overline{O}_0$. Para este caso, la señal de datos que se aplica a $\overline{E}_1$ se transmitirá a $\overline{O}_0$ y todas las demás salidas permanecerán en su estado ALTO inactivo.

Sistema de monitoreo de seguridad

Considere el caso de un sistema de monitoreo de seguridad en una planta industrial, en donde se debe monitorear el estado abierto/cerrado de muchas puertas de acceso. Cada puerta controla el estado de un interruptor y es necesario mostrar el estado de cada interruptor en LEDs que se montan en un panel de monitoreo remoto en la estación del guardia de seguridad. Una manera de hacerlo sería extender una señal separada del interruptor de cada puerta hasta un LED en el panel de monitoreo. Para esta configuración se tendrían que extender muchos alambres sobre una gran distancia. Un mejor método que reduciría la cantidad de alambrado conectado al panel de monitoreo utiliza una combinación de multiplexor/demultiplexor. La figura 9-31 muestra un sistema que puede manejar ocho puertas, pero la idea básica puede expandirse a cualquier número de puertas.

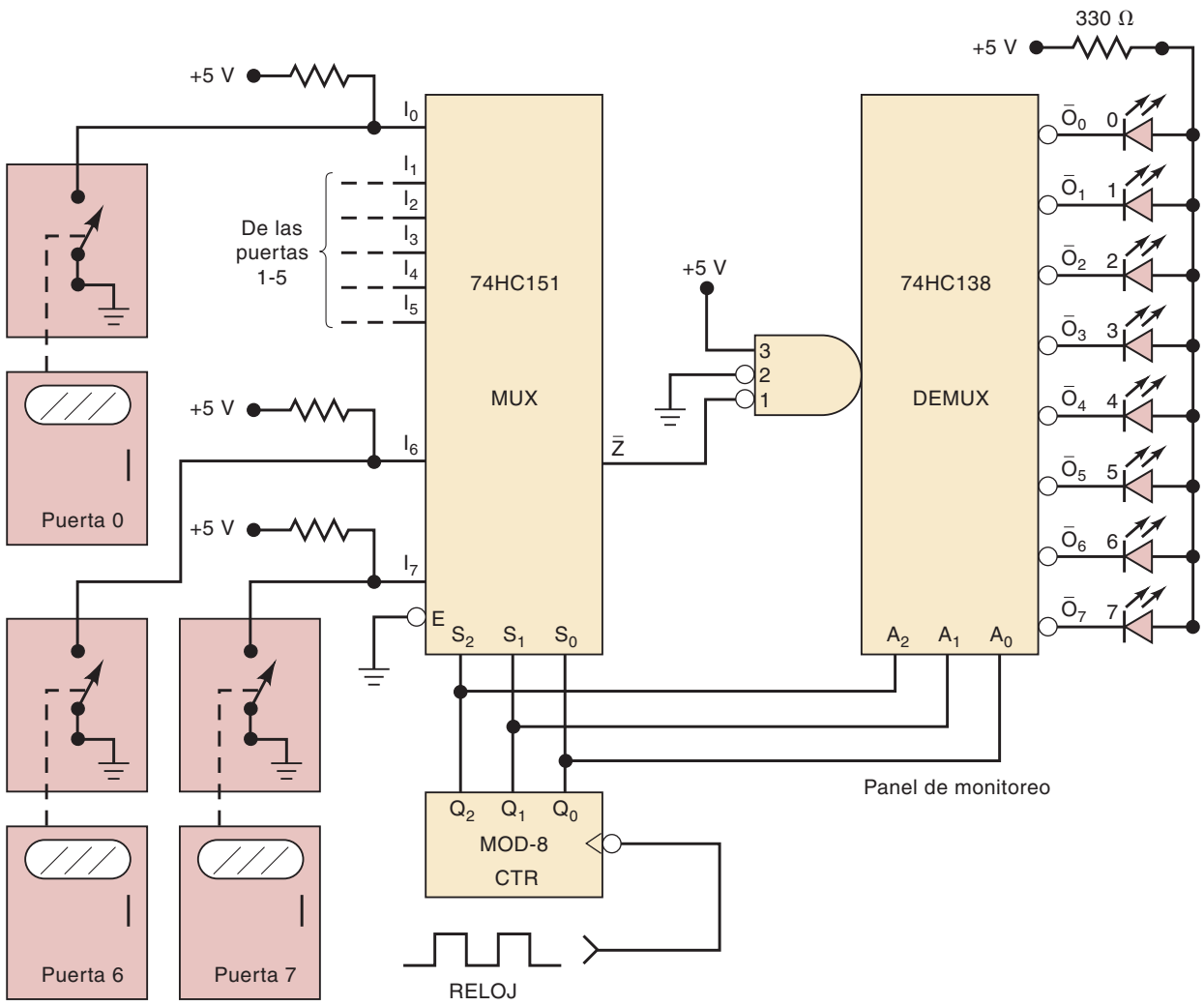


FIGURA 9-31 Sistema de monitoreo de seguridad.

EJEMPLO 9-11

Examine la figura 9-31 con cuidado y describa su operación completa.

Solución

Los ocho interruptores de las puertas son las entradas de datos para el MUX; producen un nivel ALTO cuando una puerta está abierta y un nivel BAJO cuando está cerrada. El contador MOD-8 proporciona las entradas de selección para el MUX y también para el DEMUX en el panel de monitoreo remoto. Cada salida del DEMUX se conecta a un LED indicador que se encenderá cuando la salida esté en BAJO. Los pulsos de reloj que se aplican al contador harán que las entradas de selección avancen en secuencia a través de todos los posibles estados, del 000 al 111. En cada número del contador se invertirá mediante el MUX el estado del interruptor para la puerta del mismo número y se pasará a la salida Z. De ahí se transmitirá a la entrada del DEMUX, el cual la pasará hacia la salida que corresponda al mismo número.

Por ejemplo, digamos que el contador se encuentra en el conteo de 110 (6). Mientras el contador se encuentra en este estado, digamos que la puerta 6 se cierra. El nivel BAJO en I_6 pasará a través del MUX y se invertirá para producir un nivel ALTO en $\overline{Z}$. Este nivel ALTO se pasará a través del DEMUX hacia la salida $\overline{O}_6$, por lo que el LED 6 se apagará, indicando que la puerta 6 está cerrada. Ahora digamos que la puerta 6 está abierta. Aparecerá un nivel BAJO en $\overline{Z}$ y en $\overline{O}_6$, por lo que el LED 6 se encenderá para indicar que la puerta 6 está abierta. Desde luego que todos los demás LEDs se apagarán durante este tiempo, ya que $\overline{O}_6$ es la única salida activa.

A medida que se aplican pulsos de reloj al contador para que recorra sus ocho estados del 000 al 111, los LEDs indicarán en forma secuencial el estado de las ocho puertas. Si todas las puertas están cerradas, ninguno de los LEDs se encenderá aún y cuando se seleccione la correspondiente salida del DEMUX. Si una puerta está abierta, su LED se encenderá sólo durante el intervalo de tiempo en el que el contador se encuentre en el conteo apropiado; estará apagado en todos los demás conteos. Por ende, el LED se encenderá y se apagará con destellos si su puerta se abre. La velocidad de destello puede ajustarse si se cambia la frecuencia del reloj.

Observe que sólo hay cuatro líneas de señal que van del circuito "detector de puertas" hasta el panel de monitoreo remoto: la salida $\overline{Z}$ y las tres líneas de selección. Esto representa un ahorro de cuatro líneas si se le compara con la alternativa de tener una línea por puerta. La combinación de MUX/DEMUX se utiliza para transmitir el estado de cada puerta a su LED, una a la vez (en serie) en vez de todas a la vez (en paralelo).

Sistema de transmisión de datos en forma síncrona

Las figuras 9-32 y 9-33 muestran los diagramas lógicos para un sistema de transmisión de datos en forma síncrona que se utiliza para transmitir cuatro palabras de cuatro bits en serie, desde un transmisor hacia un receptor remoto. Para operar este sistema se cargan en paralelo cuatro palabras de datos en los registros de entrada del bloque transmisor y se activa la señal para transmitir. Después los 16 bits de datos se envían a través de una sola línea de datos, un bit a la vez, para que el receptor los vuelva a ensamblar y los almacene en registros de salida. Primero veamos los detalles del transmisor en la figura 9-32. La entrada *reloj* es una señal de reloj periódica de alta frecuencia y operación constante, la cual sincroniza todas las actividades en el sistema. Las palabras de datos de cuatro bits se almacenan en forma individual (síncrona) en los registros PISO cuando se habilitan mediante la entrada *ld_x* apropiada. Las entradas de datos en paralelo para los registros PISO no se muestran en el diagrama por cuestión de simplificación. Estos registros de entrada están diseñados para desplazar los datos a la derecha y también para recircular el LSB (bit de más a la derecha) hacia el MSB (bit de más a la izquierda). Con este

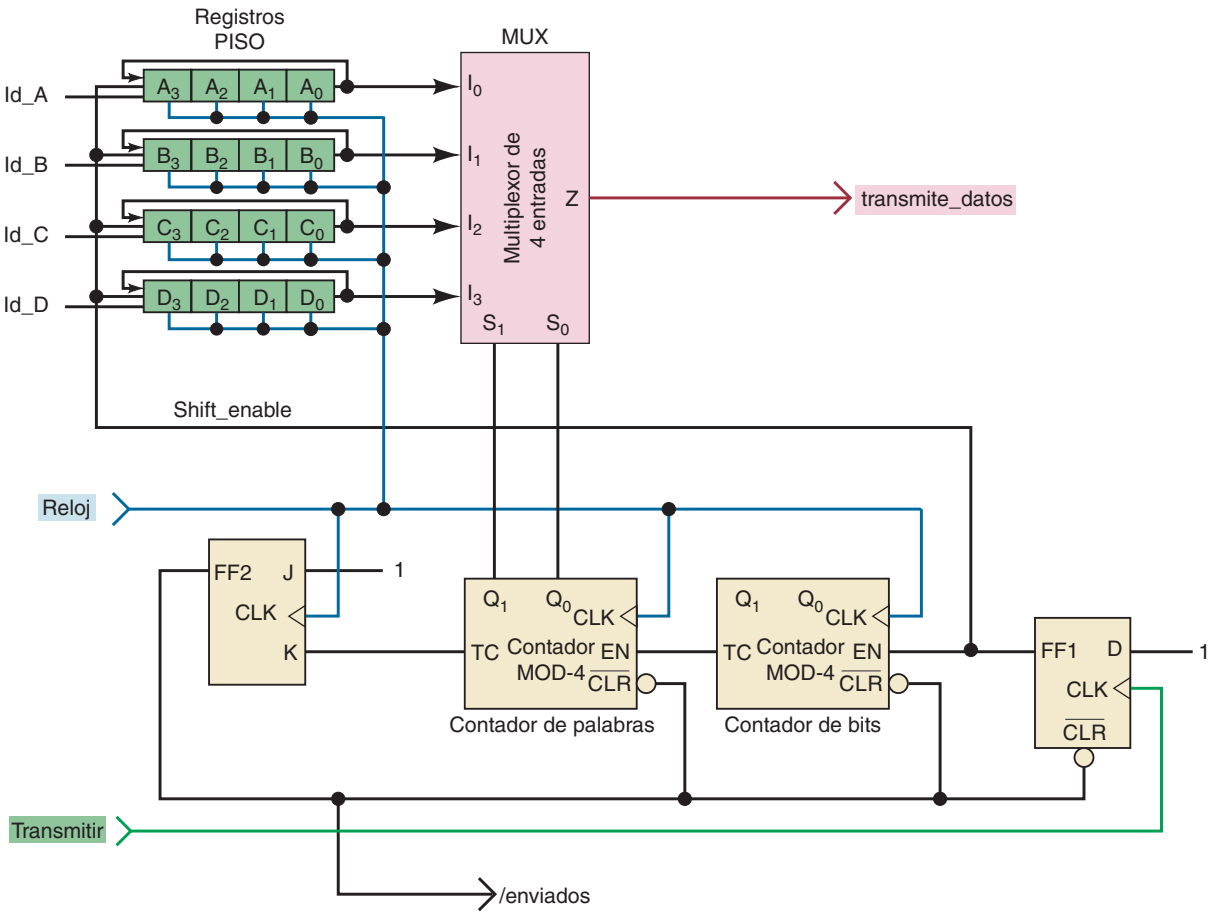


FIGURA 9-32 Bloque transmisor en un sistema de transmisión de datos en forma síncrona.

arreglo todos los bits se desplazan hacia la entrada en serie y también terminan de regreso en sus posiciones correctas después de cuatro pulsos de reloj.

OPERACIÓN DEL TRANSMISOR Vamos a suponer que al principio todos los flip-flops y los dos contadores MOD-4 de la figura 9-32 están en ceros. En la siguiente PGT del reloj se ESTABLECE *FF2*, con lo cual se quita el comando borrar asíncrono de los contadores y de *FF1*. Cuando la señal *transmitir* cambia a ALTO se ESTABLECE *FF1*, con lo cual todos los registros de desplazamiento se colocan en el modo de desplazamiento. El MUX selecciona la entrada 0 (registro *A*), ya que el contador de Palabras MOD-4 está en 0. En este punto, el LSB del registro *A* está en la línea *transmite_datos*. Los siguientes tres pulsos de reloj (el contador de Bits se encarga del conteo) desplazan los demás bits del registro *A* hacia la salida en serie. Como resultado, la línea *transmite_datos* produce como salida cada uno de los bits del registro *A*, un bit a la vez desde el menos significativo hasta el más significativo. En la cuarta PGT, el contador de Bits se regresa a cero, el contador de Palabras se incrementa en 1, todos los registros de desplazamiento han reciclado sus datos de vuelta a la posición original y el MUX ahora selecciona los datos del LSB del registro *B* para enviarlos como salida a la línea *transmite_datos*. En los siguientes tres ciclos de reloj se desplaza el contenido del registro *B*, seguido de los registros *C* y *D*. En la 16va PGT, *FF2* conmuta al estado cero, en el cual se restablecen todos los contadores y se deshabilita cualquier conteo posterior al borrar también *FF1*. La siguiente PGT establece el *FF2* de nuevo, y el sistema espera a que se carguen nuevos datos y a la siguiente señal *transmitir*.

OPERACIÓN DEL RECEPTOR El circuito receptor que se muestra en la figura 9-33 es muy similar en operación al transmisor. Observe que todos los flip-flops,

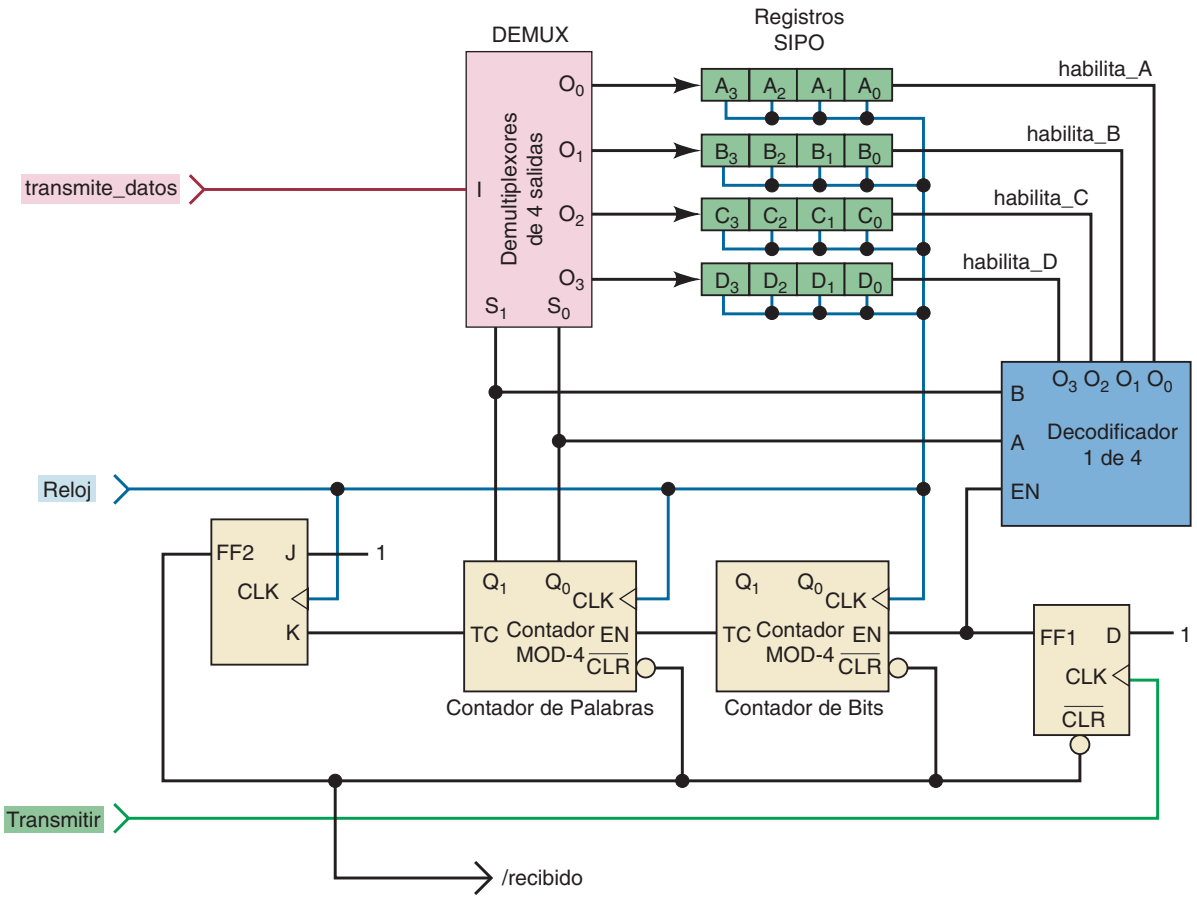


FIGURA 9-33 El bloque receptor en un sistema de transmisión de datos en forma asíncrona.

contadores y registros utilizan el mismo reloj que el transmisor. El receptor utiliza un DEMUX para distribuir los datos en serie al registro SIPO apropiado, y un decodificador para habilitar un registro a la vez. Empezaremos el análisis de este circuito con todos los contadores y los flip-flops en cero. El siguiente ciclo de *reloj* establece *FF2*, con lo cual se quita el comando borrar asíncrono de los contadores y de *FF1*. Cuando la línea *transmitir* cambia a ALTO se ESTABLECE *FF1*, con lo cual se habilitan el contador de Bits, el contador de Palabras y también el decodificador. Con el contador de Palabras en cero, el decodificador habilita el registro *A* y el DEMUX conecta la línea de datos en serie (que actualmente contiene el LSB del registro de transmisión *A*) a la entrada de datos en serie del registro de recepción *A*. La siguiente PGT desplaza el bit de datos menos significativo hacia el registro *A* y avanza el contador de Bits. Las siguientes tres PGTs desplazan los siguientes tres bits de datos hacia el registro *A*, el contador de Bits se regresa a cero, el contador de Palabras se incrementa en 1 y el decodificador y el DEMUX cambian al registro *B*. Después de la 16va PGT los cuatro registros contienen los datos apropiados, *FF2* ha conmutado a un estado cero, *FF1* se ha borrado y deshabilita el decodificador, lo cual deshabilita a todos los registros SIPO. En la siguiente PGT se establece *FF2* y el sistema espera la siguiente transmisión de datos.

SINCRONIZACIÓN DEL SISTEMA El diagrama de sincronización de la figura 9-34 muestra los datos en paralelo que se cargan en el transmisor, el flujo de datos en serie y la distribución y almacenamiento de los cuatro valores de datos en los registros receptores. Durante los tiempos $t_{1\text{-}4}$, los valores de datos binarios (que se muestran como 3, 5, 6 y D hexadecimal) se cargan en los registros de transmisión *A*, *B*, *C* y *D*, en forma respectiva. El sistema está inactivo hasta que la línea *transmitir* cambia a ALTO en t_5. En este punto, el LSB del registro $A(A_0)$ ya se encuentra en la

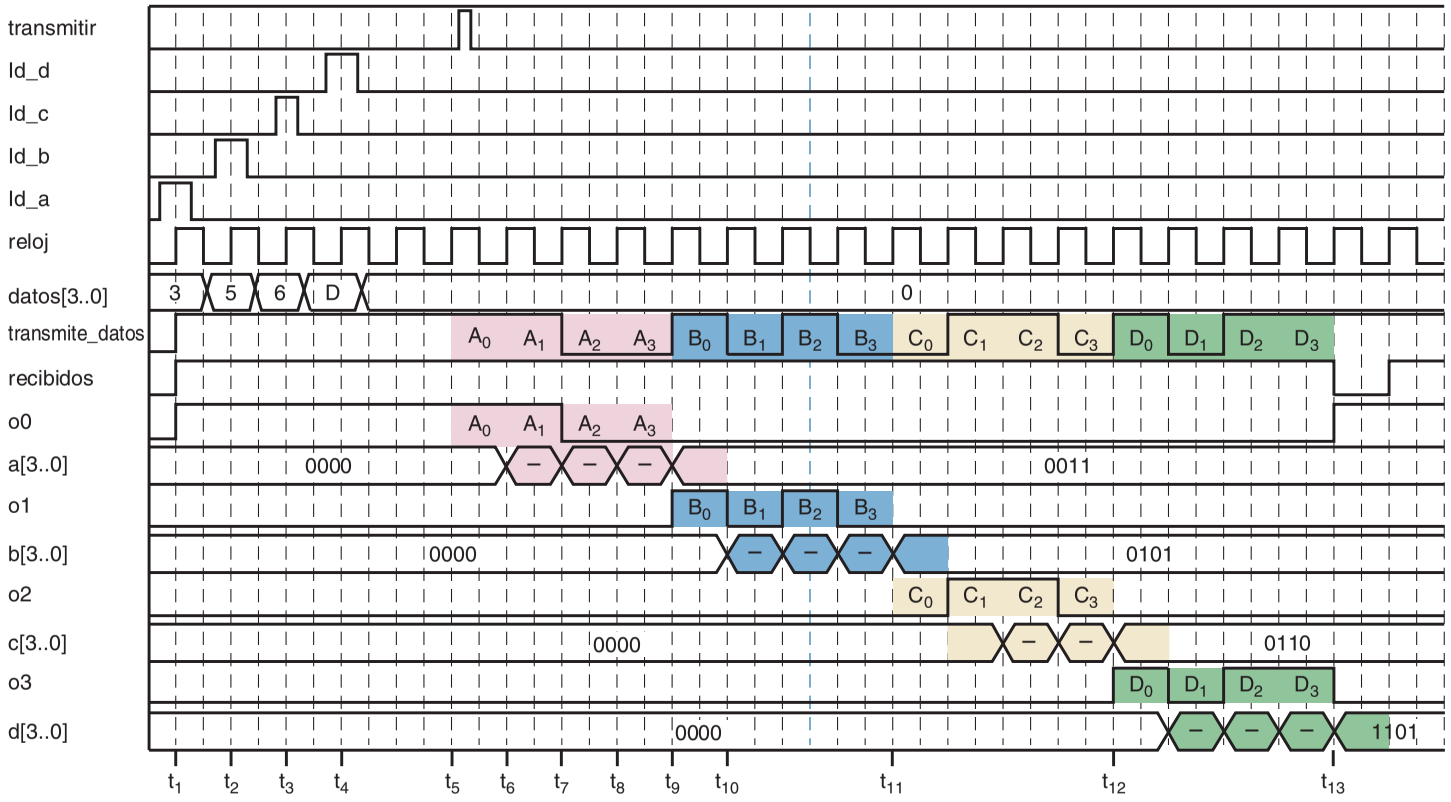


FIGURA 9-34 Diagrama de sincronización para un ciclo de transmisión completo.

línea *transmite_datos*. Observe además que en $t_5 - t_8$ los datos en la salida O_0 del DEMUX es idéntica a la línea *transmite_datos*. Esto demuestra que DEMUX ha distribuido la línea *transmite_datos* al registro de desplazamiento A. En t_6, la PGT del reloj desplaza a A_0 hacia el MSB del registro de recepción A, se desplazan todos los registros de datos (no se muestran en la sincronización) y el bit de datos A_1 aparece en la línea *transmite_datos*. En los tiempos t_7, t_8 y t_9 los otros tres bits se desplazan hacia el registro A de tal forma que después t_9, el registro de recepción A contiene los bits de datos que se almacenaron en el registro de transmisión A. El diagrama muestra que el DEMUX ha cambiado para distribuir datos al registro B, ya que la salida O_1 del DEMUX es ahora idéntica a *transmite_datos*, de t_9 a t_{11}. Comenzando en t_{10}, los datos se desplazan hacia el registro de recepción B, el cual en t_{11} contiene el valor que se almacenó en un principio en el registro de transmisión B. Los registros C y D se envían y almacenan de t_{11} a t_{12} y de t_{12} a t_{13}, respectivamente.

PREGUNTAS DE REPASO

1. Explique la diferencia entre un DEMUX y un MUX.
2. *Verdadero o falso:* el circuito para un DEMUX es en esencia el mismo que para un decodificador.
3. Para el sistema de la figura 9-31, ¿qué verá el guardia de seguridad en el panel de monitoreo cuando todas las puertas estén abiertas?

9-9 MÁS DIAGNÓSTICO DE FALLAS

He aquí tres ejemplos más para ilustrar el proceso de observación/razonamiento, que es un paso inicial tan importante cuando se diagnostican fallas. Para cada caso trate de determinar la falla del circuito antes de analizar la solución.

EJEMPLO 9-12

Considere el circuito de la figura 9-24. Una prueba realizada en este circuito produce el resultado que se muestra en la tabla 9-3. ¿Cuál es la probable falla del circuito?

TABLA 9-3

		Conteo actual	Conteo en pantalla
Caso 1	Contador 1	25	25
	Contador 2	37	35
Caso 2	Contador 1	49	49
	Contador 2	72	79
Caso 3	Contador 1	96	96
	Contador 2	14	16

Solución

En cada uno de los casos de prueba, la pantalla del contador 1 concuerda con el conteo real del contador. Esto indica que es probable que las entradas I_1, todas las salidas MUX y ambas pantallas funcionen de manera correcta. Por otro lado, cada caso de prueba demuestra que el dígito de las *décadas* del contador 2 se visualiza en forma correcta, pero su dígito de *unidades* no. Esto podría significar que hay una

falla en alguna parte entre la salida de la sección de las unidades del contador 2 y las entradas I_0 del MUX de las unidades. Deberíamos comparar los patrones de bits de los valores reales y en pantalla de las unidades para el contador 2 (tabla 9-4). La idea es buscar cosas tales como un bit que no cambie (que permanezca en BAJO o en ALTO) o dos bits que se inviertan (conexiones cruzadas). Los datos en la tabla 9-4 no revelan un patrón obvio.

TABLA 9-4

	Unidades actuales	Unidades en pantalla
Caso 1	0111 (7)	0101 (5)
Caso 2	0010 (2)	1001 (9)
Caso 3	0100 (4)	0110 (6)

Si analizamos de nuevo los resultados registrados de la prueba, veremos que el dígito de las unidades en pantalla del contador 2 es siempre el mismo que el dígito de las unidades del contador 1. Es probable que este síntoma sea el resultado de un nivel ALTO lógico constante en la entrada de selección del MUX de las unidades, ya que se pasaría en forma continua el dígito de las unidades del contador 1 a la salida del MUX de las unidades. Es muy probable que este nivel ALTO constante en la entrada de selección lo ocasione una ruta abierta en alguna parte entre la entrada de selección del MUX de las décadas y la entrada de selección del MUX de las unidades. No podría ser ocasionado por un corto con V_{CC}, ya que la entrada de selección del MUX de las décadas también se mantendría en un nivel ALTO constante y sabemos que el MUX de las décadas está funcionando.

EJEMPLO 9-13

El sistema de monitoreo de seguridad de la figura 9-31 se probó y los resultados se registraron en la tabla 9-5. ¿Cuáles son las posibles fallas que podrían producir estos resultados?

TABLA 9-5

Condición	LEDs
Todas las puertas cerradas	Todos los LEDs apagados
Puerta 0 abierta	LED 4 destellando
Puerta 1 abierta	LED 5 destellando
Puerta 2 abierta	LED 6 destellando
Puerta 3 abierta	LED 7 destellando
Puerta 4 abierta	LED 4 destellando
Puerta 5 abierta	LED 5 destellando
Puerta 6 abierta	LED 6 destellando
Puerta 7 abierta	LED 7 destellando

Solución

De nuevo debemos revisar los datos para ver si hay algún patrón que pudiera ayudar a reducir la búsqueda de la falla a una pequeña área del circuito. Los datos de la tabla 9-5 revelan que destellan los LEDs correctos para las puertas abiertas de la 4 a la 7. También muestran que para las puertas abiertas de la 0 a la 3, el número

del LED que destella es *cuatro números* más que el número de la puerta, y los LEDs del 0 al 3 siempre están apagados. La causa más probable de esto es un nivel lógico ALTO constante en A_2, el MSB de la entrada de selección del DEMUX, porque esto siempre hará que el código de selección sea 4 o mayor; además agregaría 4 números a los códigos de selección del 0 al 3.

Por ende, tenemos dos posibilidades: que A_2 esté de alguna forma en corto con V_{CC}, o que haya una conexión abierta en A_2. Después de pensarlo un poco eliminamos la primera opción como una posibilidad, ya que esto también significaría que la señal S_2 del MUX también permanecería en ALTO. Si así fuera, entonces el estado de las puertas de la 0 a la 3 no pasaría a través del MUX y luego hacia el EMUX. Sabemos que esto no es cierto ya que los datos muestran que cuando se abre cualquiera de estas puertas, afecta a una de las salidas del DEMUX.

EJEMPLO 9-14

En la sección 9-5 presentamos un principio extremadamente importante del diagnóstico de fallas, al cual se le conoce como *dividir y conquistar*. Este principio no trata sobre una estrategia militar, sino que describe la manera más eficiente de eliminar, mediante la consideración, todas las partes del circuito que trabajan en forma correcta. Suponga que se han cargado datos en los cuatro registros de transmisión de la figura 9-32 y que se ha producido el pulso para transmitir, pero después de los siguientes 16 pulsos de reloj no han aparecido datos en los registros de recepción que se muestran en la figura 9-33. ¿Podremos encontrar el problema de la manera más eficiente?

Solución

En un sistema digital síncrono que simplemente no funciona, es razonable primero revisar que la fuente de energía y el reloj estén funcionando, así como revisaríamos el pulso de una persona si la encontráramos tirada en el piso. No obstante y suponiendo que el reloj está oscilando, hay una manera mucho más eficiente de aislar el problema, en vez de elegir puntos al azar en el circuito y determinar si está presente la señal correcta. Queremos realizar una prueba en este circuito de tal forma que, si obtenemos los resultados que esperamos, sabremos que la mitad del circuito estará funcionando bien y podremos dejar de considerar esa mitad como posible lugar de la falla. En este circuito, el mejor lugar para buscar es en la línea *transmite_datos*. Hay que colocar una sonda lógica en la línea *transmite_datos* y activar la señal *transmitir*. Si observamos una ráfaga de pulsos en la sonda lógica, significa que la sección de transmisión está funcionando. Tal vez no sepamos si los datos son correctos, pero hay que recordar que el receptor no está recibiendo datos incorrectos, sino que no está recibiendo ningún tipo de dato. Pero si no observamos ninguna ráfaga de pulsos, sin duda tenemos un problema en la sección de transmisión.

Un diagrama de árbol para diagnóstico de fallas como el que se muestra en la figura 9-35 es útil para aislar los problemas en un sistema. Vamos a suponer que no hay pulsos en *transmite_datos*. Ahora necesitamos realizar una prueba en el transmisor para demostrar que la mitad de éste funciona en forma correcta. En este caso no es fácil dividir el circuito exactamente a la mitad. Una buena opción podría ser examinar la salida del contador de palabras. Hay que colocar una sonda lógica en las entradas de selección del MUX y activar la señal *transmitir*. Si se producen pulsos breves justo después de transmitir, entonces es probable que toda la sección de control (compuesta por dos contadores y dos flip-flops) esté funcionando en forma correcta y podemos buscar en otra parte. El siguiente lugar para buscar es en la salida de los registros PISO (o las entradas de datos del MUX). Si hay pulsos de datos presentes en cada línea después de activar la señal *transmitir*, el problema debe estar en el MUX. En caso contrario, podemos dividir aún más la sección PISO. Cada prueba que realicemos deberá eliminar la mayor cantidad posible del resto del circuito, hasta que todo lo que quede sea un pequeño bloque que contenga la falla.

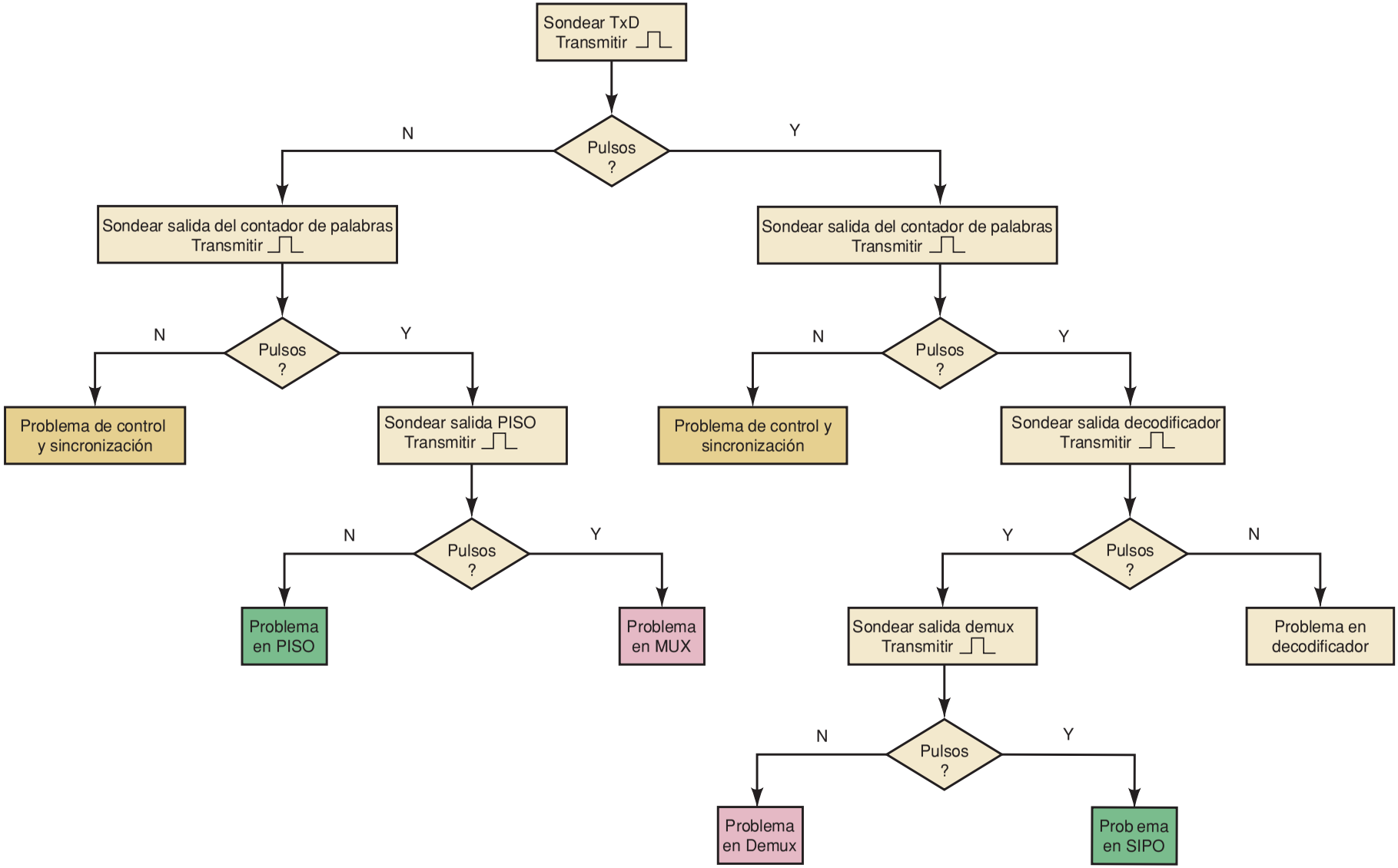


FIGURA 9-35 Ejemplo 9-14A: un diagrama de árbol para diagnóstico de fallas.

9-10 COMPARADOR DE MAGNITUD

Otro miembro útil de la categoría MSI de CIs es el **comparador de magnitud**. Es un circuito lógico combinacional que compara dos cantidades binarias de entrada y genera salidas para indicar cuál tiene la mayor magnitud. La figura 9-36 muestra el símbolo lógico y la tabla de verdad para el comparador de magnitud 74HC84 de cuatro bits, que también está disponible como el 74LS85.



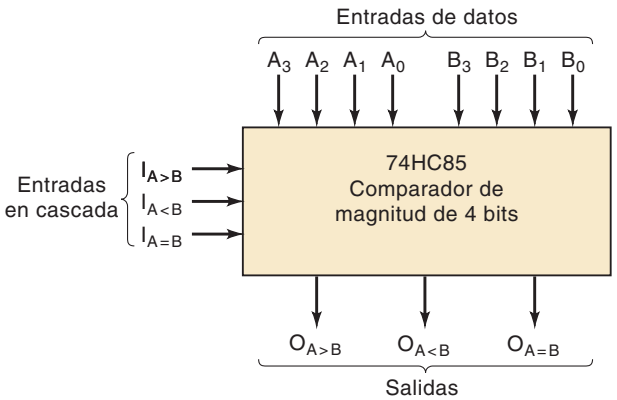


TABLA DE VERDAD

COMPARACIÓN DE ENTRADAS				COMPARACIÓN DE ENTRADAS			SALIDAS		
A_3, B_3	A_2, B_2	A_1, B_1	A_0, B_0	$I_{A>B}$	$I_{A<B}$	$I_{A=B}$	$O_{A>B}$	$O_{A<B}$	$O_{A=B}$
$A_3>B_3$	X	X	X	X	X	X	H	L	L
$A_3<B_3$	X	X	X	X	X	X	L	H	L
$A_3=B_3$	$A_2>B_2$	X	X	X	X	X	H	L	L
$A_3=B_3$	$A_2<B_2$	X	X	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1>B_1$	X	X	X	X	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1<B_1$	X	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0>B_0$	X	X	X	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0<B_0$	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	H	L	L	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	L	H	L	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	X	X	H	L	L	H
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	L	L	L	H	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	H	H	L	L	L	L

H = Nivel de voltaje ALTO
L = Nivel de voltaje BAJO
X = No importa

FIGURA 9-36 Símbolo lógico y tabla de verdad para un comparador de magnitud 74HC85(7485, 74LS85) de cuatro bits.

Entradas de datos

El 74HC85 compara dos números binarios *sin signo* de cuatro bits. Uno de ellos es $A_3A_2A_1A_0$, al cual se le llama palabra *A*; el otro es $B_3B_2B_1B_0$, al cual se le llama palabra *B*. El término *palabra* se utiliza en el campo de las computadoras digitales para designar un grupo de bits que represente cierto tipo específico de información. Aquí, las palabras *A* y *B* representan cantidades numéricas.

Salidas

El 74HC85 tiene tres salidas activas en ALTO. La salida $O_{A>B}$ estará en ALTO cuando la magnitud de la palabra A sea mayor que la magnitud de la palabra B. La salida de $O_{A<B}$ será ALTO cuando la magnitud de la palabra A sea menor que la magnitud de la palabra B. La salida $O_{A=B}$ estará en ALTO cuando las palabras A y B sean idénticas.

Entradas en cascada

La conexión de las entradas en cascada nos proporciona un medio para expandir la operación de comparación a más de cuatro bits, mediante la conexión en cascada de dos o más comparadores de cuatro bits. Observe que las entradas en cascada se etiquetan de la misma manera que las salidas. Cuando se está haciendo una comparación de cuatro bits, como en la figura 9-37(a), las entradas en cascada deben conectarse como se muestra para que el comparador produzca las salidas correctas.

Cuando se van a conectar en cascada dos comparadores, las salidas de menor orden se conectan con las entradas correspondientes del comparador de mayor orden. Esto se muestra en la figura 9-37(b), en donde el comparador a la izquierda compara los cuatro bits de menor orden de las dos palabras de ocho bits: $A_7A_6A_5A_4A_3A_2A_1A_0$ y $B_7B_6B_5B_4B_3B_2B_1B_0$. Sus salidas se alimentan a las entradas en cascada del comparador a la derecha, el cual compara los bits de mayor orden. Las salidas del comparador de mayor orden son las salidas finales que indican el resultado de la comparación de ocho bits.

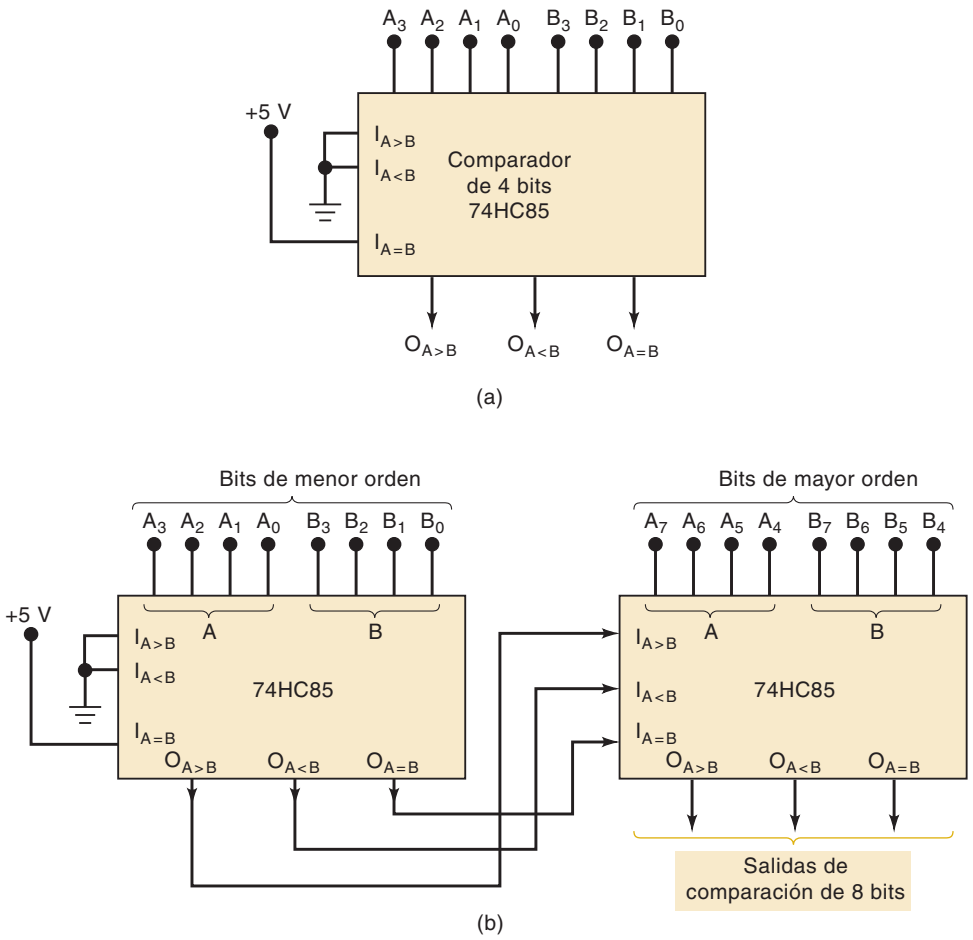


FIGURA 9-37 (a) Un 74HC85 alambrado como un comparador de cuatro bits; (b) dos chips 74HC85 en cascada para realizar una comparación de ocho bits.

EJEMPLO 9-15

Describa la operación del arreglo de comparación de ocho bits de la figura 9-37(b) para los siguientes casos:

(a) $A_7A_6A_5A_4A_3A_2A_1A_0 = 10101111; B_7B_6B_5B_4B_3B_2B_1B_0 = 10110001$
(b) $A_7A_6A_5A_4A_3A_2A_1A_0 = 10101111; B_7B_6B_5B_4B_3B_2B_1B_0 = 10101001$

Solución

(a) El comparador de mayor orden compara sus entradas $A_7A_6A_5A_4 = 1010$ y $B_7B_6B_5B_4 = 1011$, y produce $O_{A<B} = 1$ sin importar qué niveles se apliquen a sus entradas en cascada que provienen del comparador de menor orden. En otras palabras, una vez que el comparador de mayor orden detecte una diferencia en los bits de mayor orden de las dos palabras de ocho bits, sabrá cuál palabra de ocho bits es mayor sin tener que analizar los resultados de la comparación de menor orden.

(b) El comparador de mayor orden ve $A_7A_6A_5A_4 = B_7B_6B_5B_4 = 1010$, por lo que debe analizar sus entradas en cascada para ver el resultado de la comparación de menor orden. El comparador de menor orden tiene $A_3A_2A_1A_0 = 1111$ y $B_3B_2B_1B_0 = 1001$, lo cual produce un 1 en su salida $O_{A>B}$ y en la entrada $I_{A>B}$ del comparador de mayor orden. Este comparador detecta el 1 y como sus entradas de datos son iguales, produce un nivel ALTO en su salida $O_{A>B}$ para indicar el resultado de la comparación de ocho bits.

Aplicaciones

Los comparadores de magnitud también son útiles en las aplicaciones de control en las que un número binario que representa la variable física que se está controlando (por ejemplo, posición, velocidad o temperatura) se compara con un valor de referencia. Las salidas del comparador se utilizan para operar circuitos que lleven la variable física hacia el valor de referencia. El siguiente ejemplo ilustrará una aplicación. En el problema 9-52 examinaremos otra aplicación de los comparadores.

EJEMPLO 9-16

Considere un termostato digital en el que la medición de la temperatura ambiental se convierte en un número digital y se aplica a las entradas A de un comparador. La temperatura ambiental deseada se introduce desde el teclado y se almacena en un registro, el cual se conecta a las entradas B. Si $A < B$ deberá activarse el horno para calentar la habitación. El horno deberá seguir calentando mientras que $A = B$, y se apagará cuando $A > B$. Cuando la habitación empiece a enfriarse, el horno deberá permanecer apagado mientras $A = B$, y se deberá encender otra vez cuando $A > B$. ¿Qué circuito digital puede utilizarse para integrar un comparador de magnitud con un horno para realizar la aplicación de control de un termostato antes descrita?

Solución

Si utilizamos la salida $O_{A<B}$ para controlar el horno en forma directa, éste se apagaría tan pronto como los valores se hicieran iguales. Esto puede ocasionar un reciclado severo entre el estado encendido/apagado del horno cuando la temperatura actual esté muy cerca del límite entre $A < B$ y $A = B$. Si utilizamos un circuito de latch SET-CLEAR con compuertas NOR (consulte el capítulo 5) como se muestra en la figura 9-38, el sistema operará en la forma descrita. Observe que $O_{A<B}$ se conecta a la entrada SET y que $O_{A>B}$ se conecta a la entrada BORRAR del latch. Cuando la temperatura es más caliente de lo deseado se borra el latch y se apaga el horno. Cuando la temperatura es más fría de lo deseado se establece el latch y se enciende el horno.

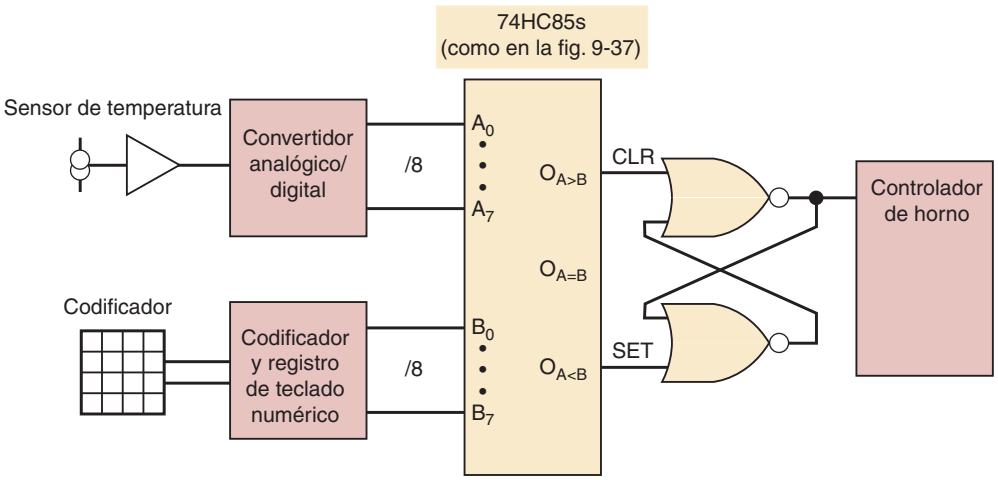


FIGURA 9-38
Comparador de magnitud que se utiliza en un termostato digital.

PREGUNTAS DE REPASO

1. ¿Cuál es el propósito de las entradas en cascada del 74HC85?
2. ¿Cuáles son las salidas de un 74HC85 con las siguientes entradas: $A_3A_2A_1A_0 = B_3B_2B_1B_0 = 1001$, $I_{A>B} = I_{A<B} = 0$, e $I_{A=B} = 1$?

9-11 CONVERTIDORES DE CÓDIGO

Un convertidor de código es un circuito lógico que cambia los datos que se presentan en cierto tipo de código binario, a otro tipo de código binario. El decodificador-controlador de BCD a 7 segmentos que presentamos antes es un convertidor de código, ya que cambia un código de entrada BCD al código de 7 segmentos que requiere la pantalla de LEDs. En la tabla 9-6 se muestra una lista parcial de algunas de las conversiones de código más comunes.

TABLA 9-6

BCD a 7 segmentos
BCD a binario
Binario a BCD
Binario a código de Gray
Código de Gray a binario
ASCII a EBCDIC*
EBCDIC a ASCII

*EBCDIC es un código alfanumérico que desarrolló IBM; es similar al código ASCII.

Como ejemplo de un circuito convertidor de código, consideremos un convertidor de BCD a binario. Antes de comenzar con la implementación del circuito, debemos repasar la representación BCD.

Los valores decimales de dos dígitos que varían del 00 al 99 pueden representarse en BCD mediante dos grupos de código de cuatro bits. Por ejemplo, el 57_{10} se representa como:

$$\overbrace{0101}^{5}\quad\overbrace{0111}^{7}\qquad\text{(BCD)}$$

La representación binaria directa para el 57 decimal es:

$$57_{10} = 111001_2$$

El valor decimal de dos dígitos más grande (99) tiene la siguiente representación:

$$99_{10} = 10011001\,\text{(BCD)} = 1100011_2$$

Observe que la representación binaria sólo requiere siete bits.

Idea básica

El diagrama de la figura 9-39 muestra la idea básica para un convertidor de BCD a binario de dos dígitos. Las entradas para el convertidor son los dos grupos de

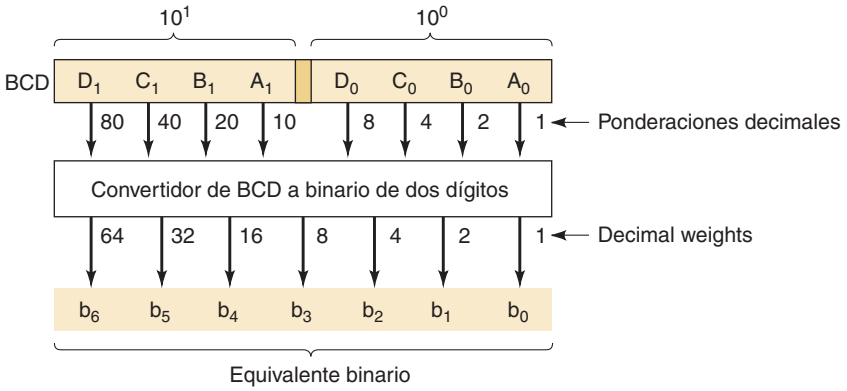


FIGURA 9-39 Idea básica de un convertidor de BCD a binario de dos dígitos.

código de cuatro bits $D_0C_0B_0A_0$, los cuales representan el dígito de las unidades o 10^0, y $D_1C_1B_1A_1$ representan el dígito de las décadas o 10^1 del valor decimal. Las salidas del convertidor son $b_6b_5b_4b_3b_2b_1b_0$, los siete bits del equivalente binario al valor decimal. Observe la diferencia en los pesos de los bits BCD y en los pesos de los bits binarios.

Un uso común del convertidor de BCD a binario sería en el que los datos BCD que provienen de un instrumento tal como un DMM (multímetro digital) se van a transferir a una computadora para su almacenamiento o procesamiento. Los datos deben convertirse en binario para que la ALU de la computadora pueda trabajar con ellos, ya que ésta tal vez no tenga la capacidad de realizar operaciones aritméticas con datos BCD. La conversión de BCD a binario puede llevarse a cabo con hardware o con software. Por lo general, el método de hardware (el cual veremos en un momento) es más rápido pero requiere de circuitos adicionales. El método de software no utiliza circuitos adicionales, pero requiere de más tiempo ya que el software no realiza la conversión paso a paso. El método a elegir en una aplicación específica dependerá de si el tiempo de conversión es o no una consideración importante.

Proceso de conversión

Los bits en una representación en BCD tienen los pesos decimales que son 8, 4, 2, 1 dentro de cada grupo de código, pero que difieren por un factor de 10 de un grupo de código (dígito decimal) al siguiente. La figura 9-39 muestra los pesos de los bits para la representación en BCD de dos dígitos.

El peso decimal de cada bit en la representación en BCD puede convertirse en su equivalente binario. En la tabla 9-7 se muestran los resultados. Si utilizamos estos pesos, podremos realizar la conversión de BCD a binario con sólo hacer lo siguiente:

TABLA 9-7
Equivalentes binarios de los pesos decimales de cada bit BCD.

Bit BCD	Peso decimal	Equivalente binario						
		b_6	b_5	b_4	b_3	b_2	b_1	b_0
A_0	1	0	0	0	0	0	0	1
B_0	2	0	0	0	0	0	1	0
C_0	4	0	0	0	0	1	0	0
D_0	8	0	0	0	1	0	0	0
A_1	10	0	0	0	1	0	1	0
B_1	20	0	0	1	0	1	0	0
C_1	40	0	1	0	1	0	0	0
D_1	80	1	0	1	0	0	0	0

Calcule la suma binaria de los equivalentes binarios de todos los bits en la representación en BCD que sean 1s.

El siguiente ejemplo ilustrará esto.

EJEMPLO 9-17

Convierta el número 01010010 (BCD para el 52 decimal) en binario. Repita para el 10010101 (95 decimal).

Solución

Anote los equivalentes en binario para todos los 1s en la representación en BCD. Después súmelos en binario.

```
0 1 0 1 0 0 1 0   (BCD)
      →0000010 (binario para el 2)
      →0001010 (binario para el 10)
    →+ 0101000 (binario para el 40)
      0110100 (binario para el 52)
1 0 0 1 0 1 0 1   (BCD)
      →0000001 (binario para el 1)
      →0000100 (binario para el 4)
      →0001010 (binario para el 10)
    →+ 1010000 (binario para el 80)
      1011111 (binario para el 95)
```

Implementación del circuito

Sin duda, una manera de implementar el circuito lógico que realice este proceso de conversión es mediante el uso de circuitos sumadores binarios. La figura 9-40

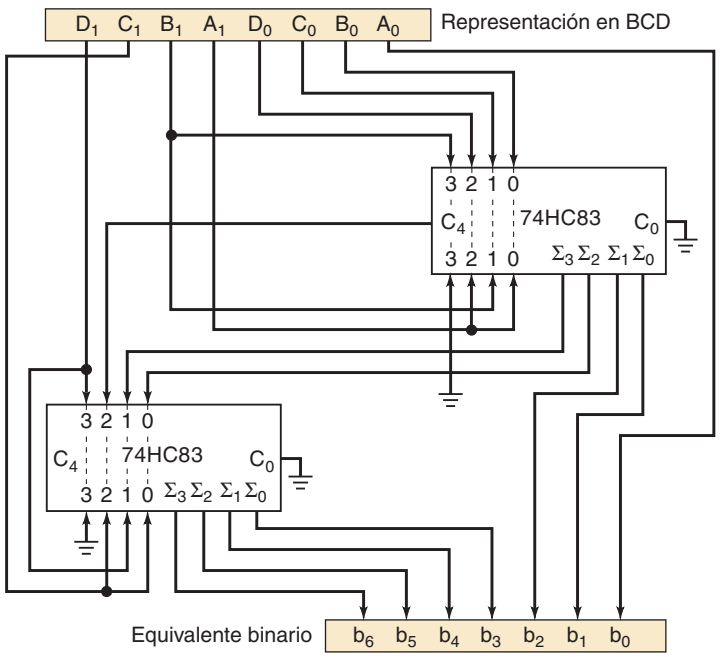


FIGURA 9-40
Convertidor de BCD a binario, implementado con dos sumadores en paralelo 74HC83 de cuatro bits.

muestra cómo dos sumadores en paralelo 74HC83 de cuatro bits pueden conectarse para realizar la conversión. Éste es uno de varios arreglos posibles de sumadores que funcionará. Tal vez quiera repasar la operación de este CI en la sección 6-14.

Los dos CIs sumadores realizan la suma de los bits BCD en las combinaciones apropiadas, de acuerdo con la tabla 9-7. Por ejemplo, esta tabla muestra que A_0 es el único bit BCD que contribuye para el LSB (b_0) del equivalente binario. Como no hay acarreo hacia esta posición de bit, A_0 se conecta en forma directa como la salida b_0. La tabla también muestra que sólo los bits BCD b_0 y A_1 contribuyen para el bit b_1 de la salida binaria. Estos dos bits se combinan en el sumador superior para producir la salida b_1. De igual forma, sólo los bits BCD D_0, A_1 y C_1 contribuyen para el bit b_3. El sumador superior combina D_0 y A_1 para generar Σ_2, que se conecta al sumador inferior, en donde se le suma C_1 para producir b_3.

EJEMPLO 9-18

La representación en BCD para el 56 decimal se aplica al convertidor de la figura 9-40. Determine las salidas Σ de cada sumador y la salida binaria final.

Solución

Anote los bits de la representación en BCD 01010110 en el diagrama del circuito. Como $A_0 = 0$, el bit b_0 de la salida es 0.

Las entradas superiores para el sumador superior son 0011. Las entradas inferiores son 0101. El sumador suma estas entradas para producir

$$\begin{array}{l} 0011 \\ +\underline{0101} \\ 1000 = \Sigma_3\Sigma_2\Sigma_1\Sigma_0 \text{ salidas del sumador superior} \end{array}$$

Los bits Σ_1 y Σ_0 se convierten en las salidas binarias b_2 y b_1, respectivamente. Los bits Σ_3 y Σ_2 se alimentan al sumador inferior. Por lo tanto, las entradas superiores para el sumador inferior son 0010 y las entradas inferiores son 0101. El sumador suma estas entradas para producir

$$\begin{array}{l} 0011 \\ +\underline{0101} \\ 1000 = \Sigma_3\Sigma_2\Sigma_1\Sigma_0 \text{ salidas del sumador inferior} \end{array}$$

Estos bits se convierten en $b_6b_5b_4b_3b_2b_1b_0$, respectivamente.

Por ende, tenemos que $b_6b_5b_4b_3b_2b_1b_0 = 0111000$ como el equivalente binario correcto para el 56 decimal.

Otras implementaciones del convertidor de código

Mientras que pueden crearse todo tipo de convertidores de código mediante la combinación de compuertas lógicas, circuitos sumadores o demás lógica combinacional, el circuito puede volverse bastante complejo al requerir muchos CIs. A menudo es más eficiente utilizar una memoria de sólo lectura (ROM) o un dispositivo lógico programable (PLD) para funcionar como un convertidor de código. Como veremos en los capítulos 12 y 13, estos dispositivos contienen el equivalente de cientos de compuertas lógicas, por lo que pueden programarse para proporcionar una amplia variedad de funciones lógicas.

PREGUNTAS DE REPASO

1. ¿Qué es un convertidor de código?
2. ¿Cuántas salidas binarias tendría un convertidor de BCD a binario de tres dígitos?

9-12 BUSES DE DATOS

En la mayoría de las computadoras modernas, la transferencia de datos se lleva a cabo a través de un conjunto común de líneas de conexión, al cual se le conoce como **bus de datos**. En estas computadoras organizadas por buses, muchos dispositivos distintos pueden tener sus salidas y entradas conectadas a las líneas del bus de datos común. Debido a esto, los dispositivos que están conectados al bus de datos tendrán, por lo general, salidas triestado, o estarán conectados al bus de datos a través de búferes triestado.

Algunos de los dispositivos que se conectan, por lo común, a un bus de datos son (1) los microprocesadores; (2) los circuitos semiconductores de memoria, que cubriremos en el capítulo 12; (3) los convertidores digitales/analógicos (DACs) y los convertidores analógicos/digitales (ADCs), que describiremos en el capítulo 11.

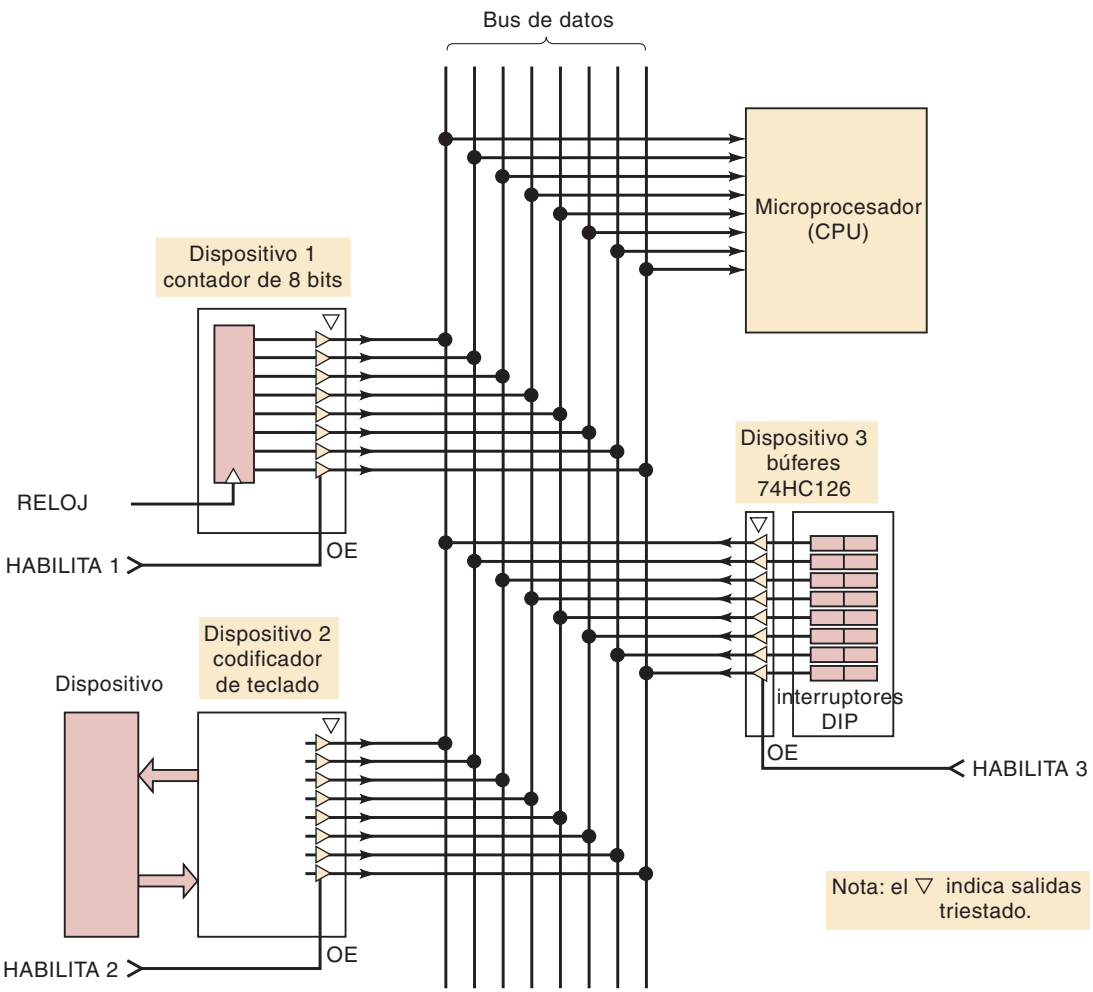


FIGURA 9-41 Tres dispositivos distintos pueden transmitir datos de ocho bits hacia un microprocesador a través de un bus de datos de ocho líneas; sólo un dispositivo se habilita en un momento dado, para evitar la colisión de bus.

La figura 9-41 ilustra una situación común en la que un microprocesador (el circuito integrado CPU en una microcomputadora) se conecta a varios dispositivos a través de un bus de datos de ocho líneas. El bus de datos es tan sólo una colección de rutas conductoras a través de las cuales se transmiten datos digitales de un dispositivo a otro. Cada dispositivo proporciona una salida de ocho bits que se envía a las entradas del microprocesador a través del bus de datos de ocho líneas. Sin duda, debido a que las salidas de cada uno de los tres dispositivos están conectadas a las mismas entradas del microprocesador a través de las rutas conductoras del bus de datos, debemos estar al pendiente de los problemas de colisión de bus (sección 8-12), en donde dos o más señales conectadas a la misma línea de bus están activas y en esencia luchan entre sí. La colisión de bus puede evitarse si los dispositivos tienen salidas triestado o si están conectados al bus a través de búferes triestado (sección 8-12). Las entradas de habilitación de la salida (*OE*) para cada dispositivo (o su búfer) se utilizan para asegurar que no haya más de una salida de dispositivo activa en un momento dado.

EJEMPLO 9-19

(a) Para la figura 9-41, describa las condiciones necesarias para transmitir datos del dispositivo 3 al microprocesador.

(b) ¿Cuál será el estado del bus de datos cuando ninguno de los dispositivos esté habilitado?

Solución

(a) HABILITA 3 debe estar activada; HABILITA 1 y HABILITA 2 deben encontrarse en su estado inactivo. Esto colocará a las salidas de los dispositivos 1 y 2 en el estado Hi-Z, con lo cual se desconectarán del bus. Las salidas del dispositivo 3 se activarán de manera que sus niveles lógicos aparezcan en las líneas del bus de datos y se transmitan a las entradas del microprocesador. Podemos visualizar esto si cubrimos los dispositivos 1 y 2, como si no fueran parte del circuito; entonces nos quedamos con el dispositivo 3, que será el único conectado al microprocesador a través del bus de datos.

(b) Si ninguna de las entradas de habilitación de los dispositivos está activada, todas las salidas de los dispositivos se encuentran en el estado Hi-Z. Esto desconecta del bus a todas las salidas de los dispositivos. Por ende, no hay un nivel lógico definido en ninguna de las líneas del bus de datos; se encuentran en el estado indeterminado. Esta condición se conoce como **bus flotante**, y se dice que cada una de las líneas del bus de datos se encuentra en un estado *flotante* (indeterminado). La visualización en el osciloscopio de una línea de bus flotante sería impredecible. Una sonda lógica indicaría un nivel lógico indeterminado.

PREGUNTAS DE REPASO

1. ¿Qué significa el término *bus de datos*?
2. ¿Qué es la *colisión de bus* y qué debe hacerse para evitarla?
3. ¿Qué es un *bus flotante*?

9-13 EL REGISTRO TRIESTADO 74ALS173/HC173

Los dispositivos conectados a un bus de datos contendrán registros (por lo general flip-flops) que almacenen los datos de cada dispositivo. Por lo general, las salidas de estos registros se conectan a búferes triestado que les permiten enlazarse a un bus de datos. Para demostrar los detalles de la operación de un bus de datos utilizaremos un registro de CI que incluye los búferes triestado en el mismo encapsulado: el TTL 74ALS173 (también disponible en versiones CMOS 74HC173). En la figura 9-42 se muestra su diagrama lógico y la tabla de verdad.

Entradas					Salidas de FF
MR	CP	$\overline{IE}_1$	$\overline{IE}_2$	D_n	Q
H	X	X	X	X	L
L	L	X	X	X	Q_0
L	↑	H	X	X	Q_0
L	↑	X	H	X	Q_0
L	↑	L	L	L	L
L	↑	L	L	H	H

Cuando $\overline{OE}_1$ o $\overline{OE}_2$ están en ALTO, la salida se encuentra en el estado APAGADO (alta impedancia); sin embargo, esto no afecta al contenido ni a la operación secuencial del registro.

H = Nivel de voltaje ALTO
L = Nivel de voltaje BAJO
X = No importa

Q_0 = salida antes del PGT

Diagrama lógico

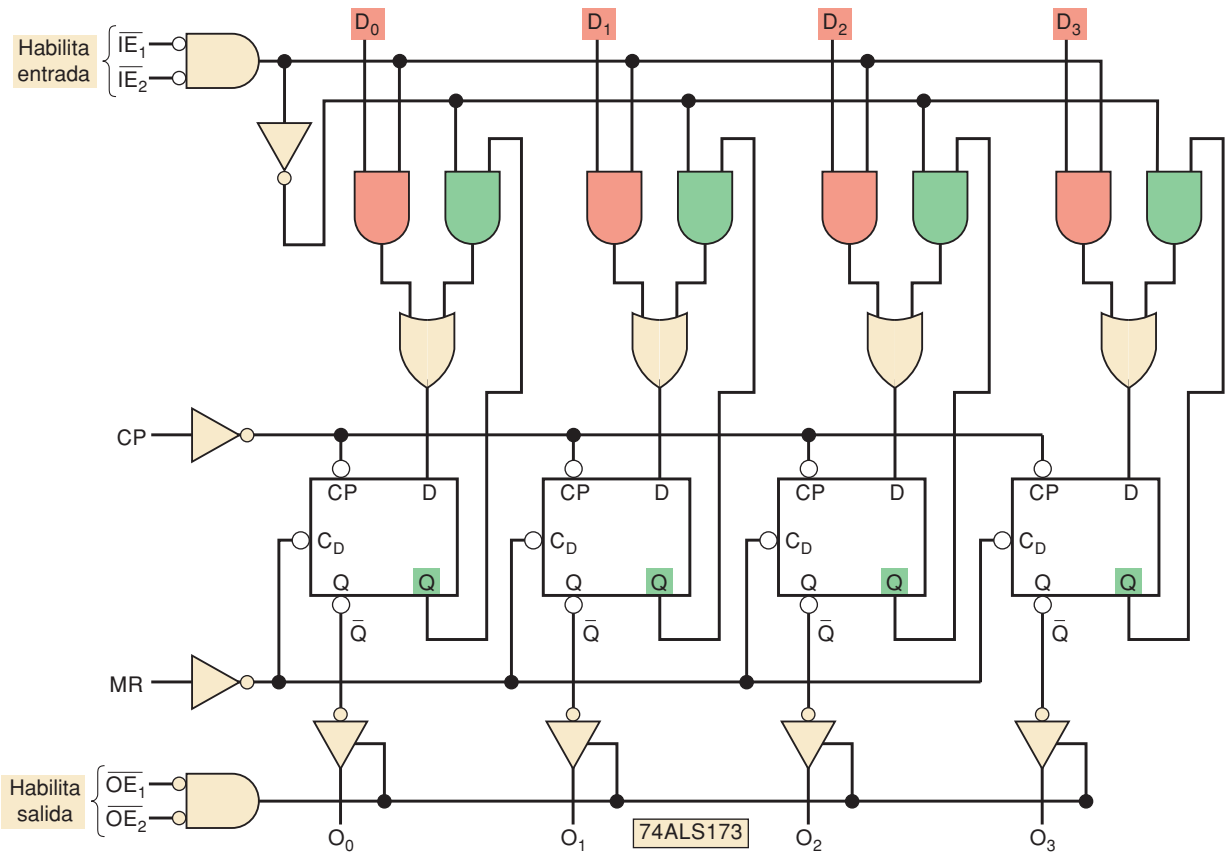


FIGURA 9-42 Tabla de verdad y diagrama lógico para el registro triestado 74ALS173.

El 74ALS173 es un registro de cuatro bits con capacidad de entrada en paralelo/salida en paralelo. Observe que las salidas de los FFs están conectadas a búferes triestado, los cuales proporcionan las salidas O_0 a la O_3. Observe además que las entradas de datos D_0 a D_3 se conectan a las entradas D de los FFs de los registros a través de circuitos lógicos. Esta lógica permite dos modos de operación: (1) *carga*, en donde los datos en las entradas D_0 a D_3 se transfieren hacia los FFs en el PGT del pulso de reloj en *CP*; y (2) *retención*, en donde los datos en el registro no cambian cuando ocurre el PGT de *CP*.

EJEMPLO 9-20

(a) ¿Qué condiciones de entrada producirán la operación de carga?

(b) ¿Qué condiciones de entrada producirán la operación de retención?

(c) ¿Qué condiciones de entrada permitirán que aparezcan las salidas internas de los registros en O_0 a O_3?

Solución

(a) Las últimas dos entradas en la tabla de verdad muestran que cada salida Q toma el valor presente en su entrada D cuando ocurre una PGT en CP, siempre y cuando MR esté en BAJO y *ambas* entradas de habilitación de entrada $\overline{IE}_1$ e $\overline{IE}_2$ estén en BAJO.

(b) Las líneas tercera y cuarta de la tabla de verdad establecen que cuando cualquiera de las entradas $\overline{IE}$ está en ALTO, las entradas D no tienen efecto y las salidas Q retendrán sus valores actuales cuando ocurra el PGT.

(c) Los búferes de salida se habilitan cuando *ambas* entradas de habilitación de salida $\overline{OE}_1$ y $\overline{OE}_2$ están en BAJO. Esto hará que se pasen las salidas del registro hacia las salidas externas O_0 a O_3. Si cualquiera de las entradas de habilitación de salida está en ALTO, los búferes se deshabilitarán y las salidas se encontrarán en el estado Hi-Z.

Observe que las entradas $\overline{OE}$ no tienen efecto sobre la operación de carga de datos. Sólo se utilizan para controlar si se pasan o no las salidas de los registros a las salidas externas.

En la figura 9-43 se muestra el símbolo lógico para el 74ALS173/HC173. Hemos incluido la notación IEEE/ANSI "&" para indicar la relación AND de los dos pares de entradas de habilitación.

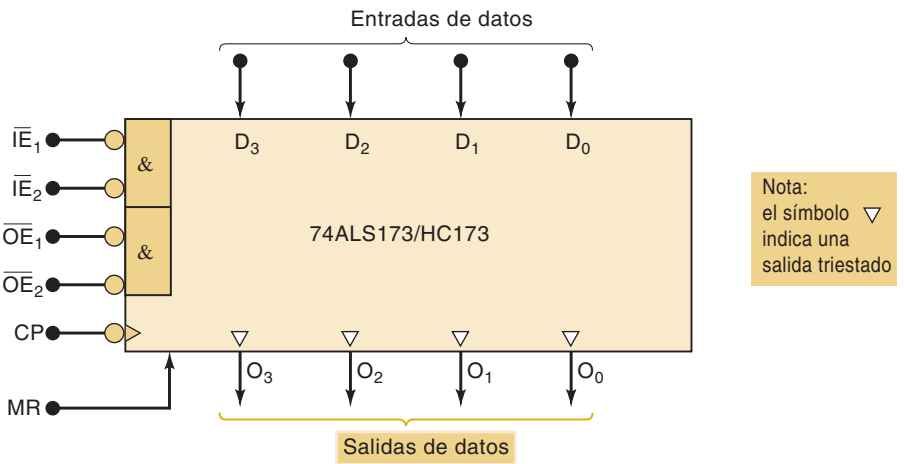


FIGURA 9-43 Símbolo lógico para el CI 74ALS173/HC173.

PREGUNTAS DE REPASO

1. Suponga que ambas entradas *IE* están en BAJO y que $D_0D_1D_2D_3 = 1011$. ¿Qué niveles lógicos están presentes en las entradas D de los FFs?
2. *Verdadero o falso:* el registro no puede cargarse cuando la entrada restablecer maestra (*MR*) se mantiene en ALTO.
3. ¿Cuáles serán los niveles de salida cuando *MR* = ALTO y ambas entradas *OE* se mantengan en bajo?

9-14 OPERACIÓN DEL BUS DE DATOS

El bus de datos es muy importante en los sistemas computacionales, pero no apreciaremos su significado sino hasta después de que estudiemos las memorias y los microprocesadores. Por ahora vamos a ilustrar la operación del bus de datos para la transferencia de datos de un registro a otro. La figura 9-44 muestra un sistema organizado por bus para tres registros triestado 74HC173. Observe que cada registro tiene su par de entradas $\overline{OE}$ conectadas entre sí como una entrada $\overline{OE}$, y de igual forma pasa con las entradas $\overline{IE}$. Observe además que nos referiremos a los registros como *A*, *B* y *C* de arriba hacia abajo. Esto se indica mediante los subíndices en cada entrada y salida.

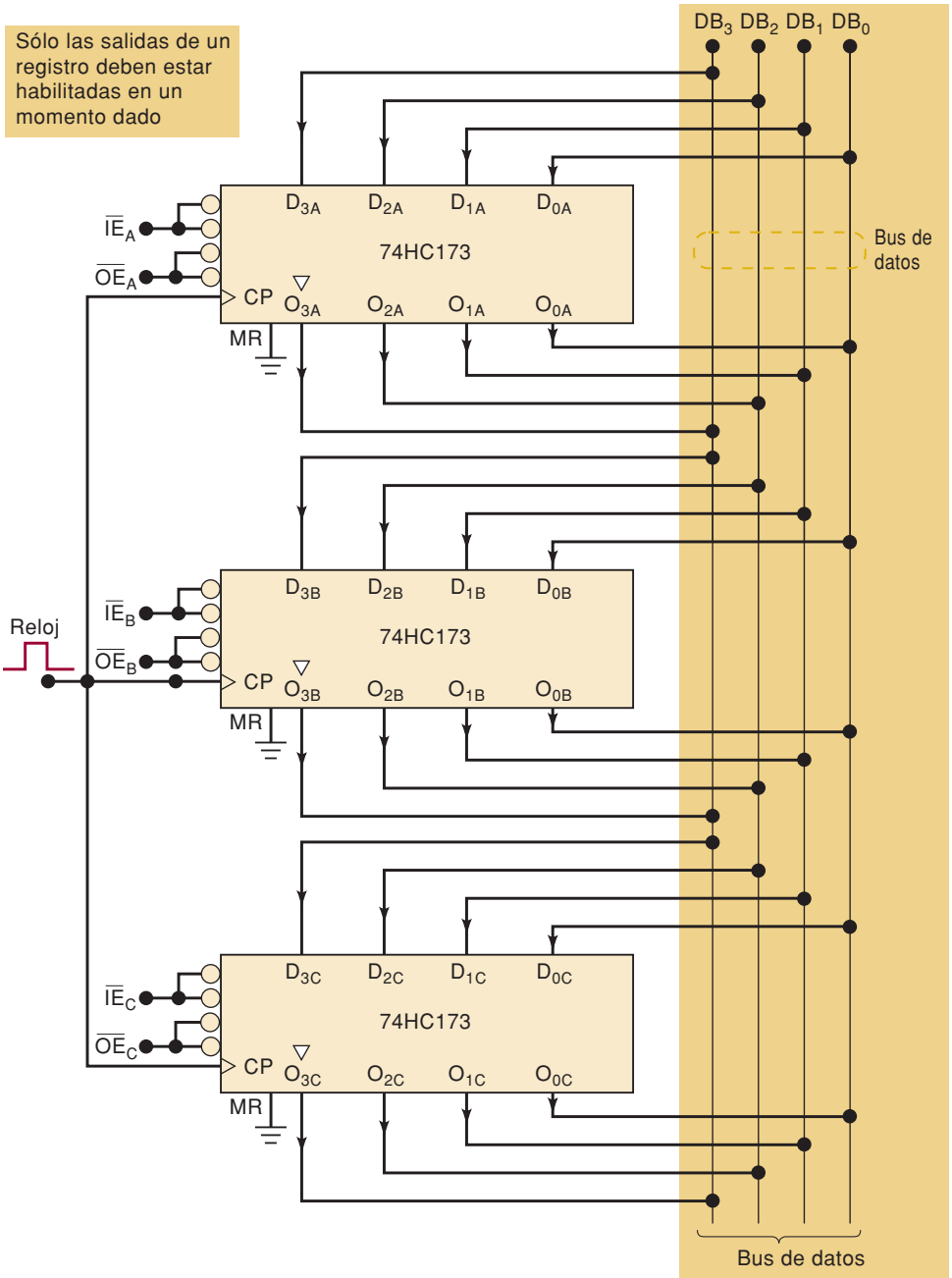


FIGURA 9-44 Registros triestado conectados a un bus de datos.

En este arreglo, el bus de datos consiste de cuatro líneas que se identifican como DB_0 a DB_3. Las correspondientes salidas de cada registro se conectan a la misma línea del bus de datos (por ejemplo, O_{3A}, O_{3B} y O_{3C} se conectan a DB_3). Como los tres registros tienen sus salidas conectadas entre sí, es imperativo que sólo un registro tenga sus salidas habilitadas y que las salidas de los otros dos registros permanezcan en el estado Hi-Z. En caso contrario habrá colisión de bus (dos o más conjuntos de salidas luchando entre sí), con lo cual se producirán niveles inciertos en el bus y un posible daño a los búferes de salida de los registros.

Las entradas de los registros correspondientes también se conectan a la misma línea de bus (por ejemplo, D_{3A}, D_{3B} y D_{3C} se conectan a DB_3). Por ende, los niveles en el bus siempre estarán listos para transferirse a uno o más de los registros, dependiendo de las entradas *IE*.

Operación de transferencia de datos

El contenido de cualquiera de los tres registros puede transferirse en paralelo a través del bus de datos y hacia uno de los otros registros, por medio de la aplicación apropiada de niveles lógicos a las entradas de habilitación de registro. En un sistema ordinario, la unidad de control de una computadora (es decir, la CPU) generará las señales que seleccionen cuál registro colocará sus datos en el bus de datos y cuál tomará los datos del bus de datos. El siguiente ejemplo ilustrará esto.

EJEMPLO 9-21

Describa los requerimientos de señal de entrada para transferir $[A] \rightarrow [C]$.

Solución

Antes que nada, sólo el registro *A* deberá tener habilitadas sus salidas. Esto es, necesitamos que:

$$\overline{OE}_A = 0 \qquad \overline{OE}_B = \overline{OE}_C = 1$$

Esto colocará el contenido del registro *A* en las líneas del bus de datos.

A continuación, sólo el registro *C* deberá tener habilitadas sus entradas. Para ello queremos que:

$$\overline{IE}_C = 0 \qquad \overline{IE}_A = \overline{IE}_B = 1$$

Esto permitirá que sólo el registro *C* acepte datos del bus de datos cuando se produzca el PGT de la señal de reloj.

Por último, se requiere un pulso de reloj para transferir los datos del bus hacia los flip-flops del registro *C*.

Señales de bus

El diagrama de sincronización de la figura 9-45 muestra las diversas señales involucradas en la transferencia de los datos 1011 del registro *A* al registro *C*. Asumiremos que las líneas $\overline{IE}$ y $\overline{OE}$ que no se muestran se encuentran en su estado ALTO inactivo. Antes del tiempo t_1 las líneas $\overline{IE}_C$ y $\overline{OE}_A$ también están en ALTO, por lo que se deshabilitan las salidas de todos los registros y ninguno de ellos colocará sus datos en las líneas de bus. En otras palabras, la líneas del bus de datos se encuentran en el estado Hi-Z o "flotante", según la representación de las líneas punteadas en el diagrama de sincronización. El estado Hi-Z no corresponde a ningún nivel de voltaje específico.

En t_1 se activan las entradas $\overline{IE}_C$ y $\overline{OE}_A$. Las salidas del registro *A* se habilitan y comienzan a cambiar las líneas DB_3 a DB_0 del bus de datos, del estado Hi-Z a los niveles lógicos 1011. Después de permitir tiempo para que los niveles lógicos se

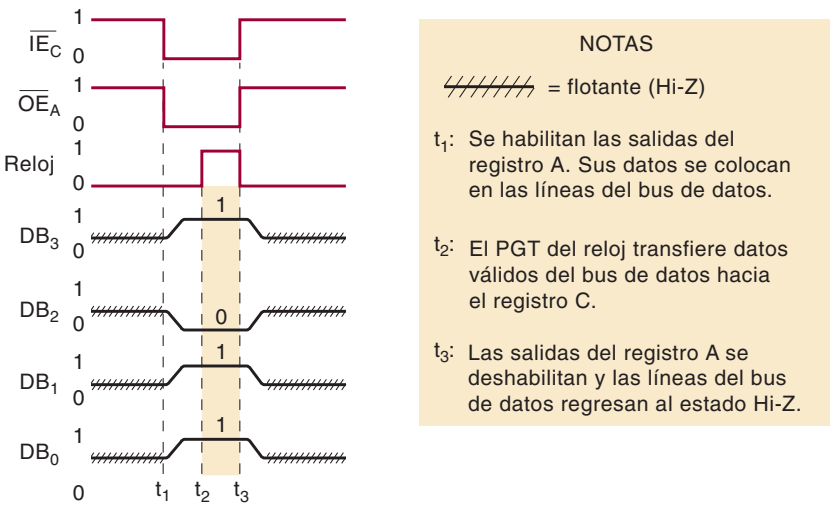


FIGURA 9-45
Actividad de las señales durante la transferencia de los datos 1011 del registro *A* al registro *C*.

estabilicen en el bus, se aplica el PGT del reloj en t_2. Este PGT transferirá estos niveles lógicos al registro *C*, ya que $\overline{IE}_C$ está activa. Si el PGT ocurre antes de que el bus de datos tenga niveles lógicos válidos, se transferirán datos impredecibles hacia *C*.

En t_3, las líneas $\overline{IE}_C$ y $\overline{OE}_A$ regresan al estado inactivo. Como resultado, las salidas del registro *A* cambian al estado Hi-Z. Con esto se quitan los datos de salida del registro *A* de las líneas del bus y éstas regresan al estado Hi-Z.

Observe que las líneas del bus de datos muestran niveles lógicos sólo durante el intervalo de tiempo en el que se habilitan las salidas del registro *A*. En todos los demás tiempos las líneas del bus de datos están flotando y no hay manera de predecir con facilidad cuál sería su apariencia si las visualizáramos en un osciloscopio. Una sonda lógica mostraría una indicación "indeterminada" si estuviera monitoreando una línea de bus flotante. Observe además la velocidad relativamente lenta con la que cambian las señales en las líneas del bus de datos. Aunque hemos exagerado un poco este efecto en el diagrama, es una característica común a los sistemas de bus y se produce debido a la carga capacitiva en cada línea. Esta carga consiste de una combinación de la capacitancia parásita y las capacitancias que contribuyen cada una de las entradas y salidas que se conectan a la línea.

Diagrama de sincronización de bus simplificado

El diagrama de sincronización de la figura 9-45 muestra las señales en cada una de las cuatro líneas del bus de datos. Este mismo tipo de actividad de las señales ocurre en los sistemas digitales que utilizan los buses de datos más comunes de 8, 16 o 32 líneas. Para estos buses más grandes, los diagramas de sincronización como el de la figura 9-45 serían demasiado extensos y voluminosos. Existe un método simplificado para mostrar la actividad de las señales que se producen en un conjunto de líneas de bus que utiliza sólo una forma de onda de sincronización individual para representar el conjunto completo de líneas de bus. Esto se ilustra en la figura 9-46 para la misma situación de transferencia de datos descrita en la figura 9-45. Observe cómo se representa la actividad del bus de datos; en especial cómo se indican los datos válidos 1011 en el diagrama, durante el intervalo t_2 - t_3. Por lo general utilizaremos este diagrama de sincronización de bus simplificado en lo sucesivo.

Expansión del bus

La operación de transferencia de datos del bus de datos de cuatro líneas de la figura 9-44 es un ejemplo típico de la operación de los buses de datos más grandes que se encuentran en la mayoría de las computadoras y demás sistemas digitales; por lo

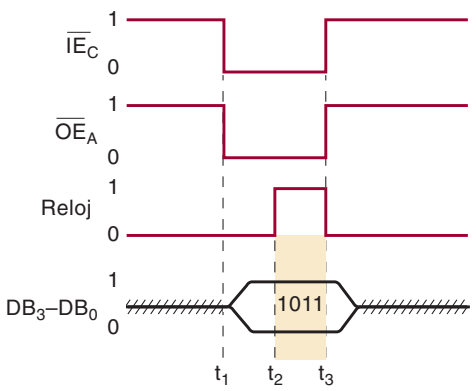


FIGURA 9-46 Forma simplificada de mostrar la actividad de las señales en las líneas del bus de datos.

general estos buses de datos son de 8, 16 o 32 líneas, y tienen mucho más de tres dispositivos conectados al bus, pero la operación de transferencia básica de datos es la misma: *un dispositivo tiene sus entradas habilitadas de manera que sus datos se coloquen en el bus de datos; otro dispositivo tiene sus entradas habilitadas de manera que pueda extraer estos datos del bus y fijarlos en sus circuitos internos, en el flanco de reloj apropiado*.

El número de líneas en el bus de datos dependerá del tamaño de la **palabra** de datos (unidad de datos) que se vaya a transferir a través del bus. Una computadora con un tamaño de palabra de 8 bits tendrá un bus de datos de ocho líneas, una computadora que tenga un tamaño de palabra de 16 bits tendrá un bus de datos de 16 líneas y así, en lo sucesivo. El número de dispositivos conectados a un bus de datos varía de una computadora a otra y depende de factores como la cantidad de memoria que tiene la computadora y el número de dispositivos de entrada y salida que deben comunicarse con la CPU a través del bus de datos.

Las salidas de todos los dispositivos deben conectarse al bus por medio de búferes triestado. Algunos dispositivos, como el registro 74173 tienen estos búferes en el mismo chip. Otros dispositivos tendrán que conectarse al bus a través de un CI llamado **controlador de bus**. Un CI controlador de bus tiene salidas triestado con una impedancia de salida muy baja, la cual puede cargar y descargar con rapidez la capacitancia del bus. Esta capacitancia representa el efecto acumulado de todas las capacitancias parásitas de las distintas entradas y salidas conectadas al bus, y puede ocasionar un deterioro de los tiempos de transición de la señal del bus si no se controlan desde una fuente de señal de baja impedancia. La figura 9-47 muestra un CI controlador de bus octal 74HC541, el cual conecta las salidas de un convertidor analógico/digital (ADC) de ocho bits a un bus de datos. El ADC tiene salidas triestado, pero carece de la capacidad para controlar la carga de la capacitancia del bus (que en el dibujo se muestra como capacitores conectados a tierra.) Observe que el bit de datos 0 controla el bus directamente, sin la asistencia del controlador de bus. Si el tiempo de transición es lo bastante lento, tal vez el voltaje nunca llegue a un nivel lógico ALTO en el tiempo de habilitación asignado. Las dos entradas de habilitación del controlador de bus están conectadas entre sí, de manera que un nivel BAJO en la línea de habilitación común permitirá que las salidas del ADC pasen a través de los búferes hacia el bus de datos, desde donde podrán transferirse hacia otro dispositivo.

Representación de bus simplificada

Por lo general, muchos dispositivos están conectados al mismo bus de datos. En el diagrama esquemático de un circuito esto puede producir un arreglo confuso de líneas y conexiones. Por esta razón, a menudo se utiliza una representación más simplificada de conexiones al bus de datos en los diagramas de bloques y en los diagramas esquemáticos de algunos circuitos. En la figura 9-48 se muestra un tipo de representación simplificada para un bus de datos de ocho líneas.

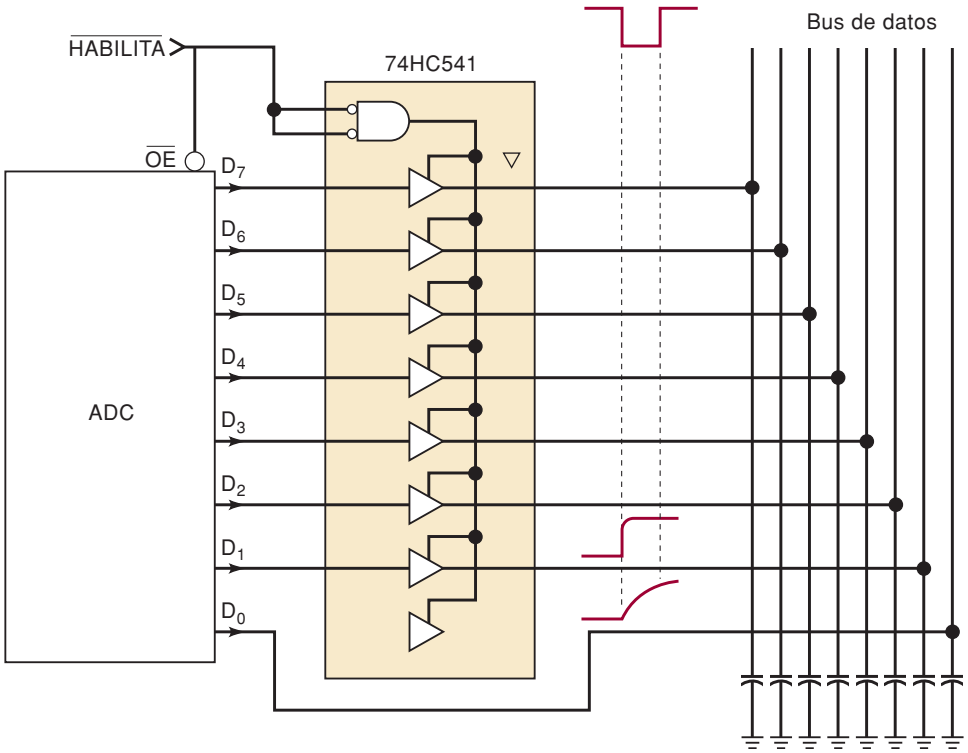


FIGURA 9-47 Un controlador de bus octal 74HC541 conecta las salidas de un convertidor analógico/digital (ADC) a un bus de datos de ocho líneas. La salida D_0 se conecta en forma directa al bus para mostrar los efectos capacitivos.

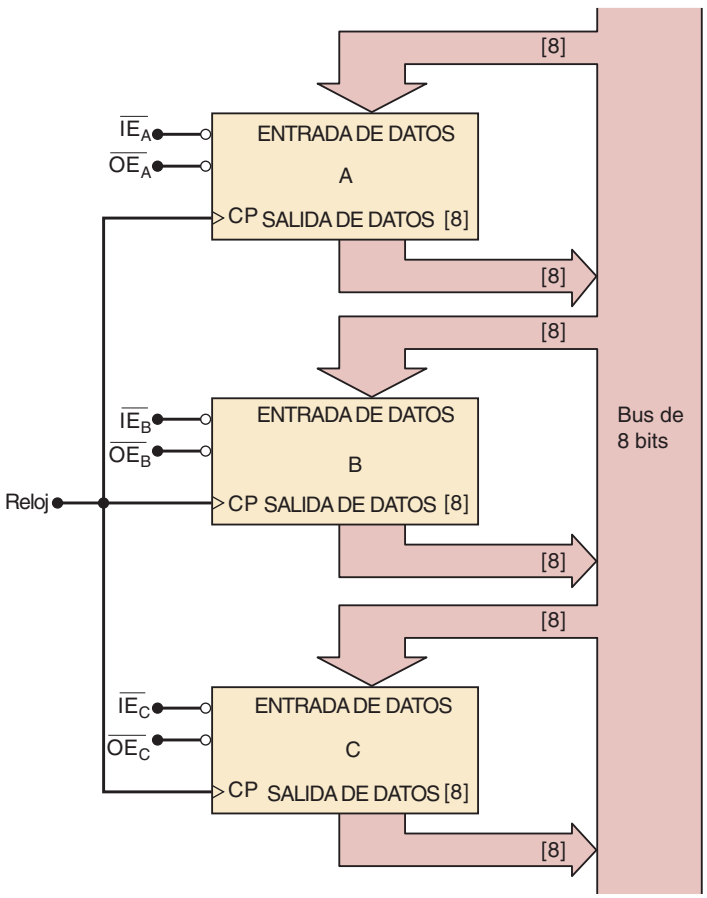


FIGURA 9-48
Representación simplificada de la distribución de un bus.

Las conexiones hacia y desde el bus de datos se representan mediante flechas gruesas. Los números entre corchetes indican el número de bits que contiene cada registro, así como el número de líneas que conectan las entradas y salidas del registro con el bus.

En la figura 9-49 se presenta otro método común para representar buses en un diagrama esquemático, para un bus de datos de ocho líneas. En esta figura se muestran las ocho líneas individuales de salida de un controlador de bus 74HC541, las cuales se identifican como $D_7 - D_0$; estas líneas están agrupadas (no conectadas entre sí) y se muestran como una sola línea. Las líneas de salida de datos agrupadas se conectan al bus de datos, el cual también se muestra como una línea (es decir, las ocho líneas del bus de datos se agrupan). La notación "/8" indica el número de líneas que representa cada agrupamiento. Este método de agrupamiento se utiliza para representar las conexiones del bus de datos a las ocho entradas de datos del microprocesador. Cuando se utiliza el método del agrupamiento es muy importante etiquetar ambos extremos de todos los alambres que se utilicen, ya que la conexión no puede rastrearse en forma visual en el diagrama.

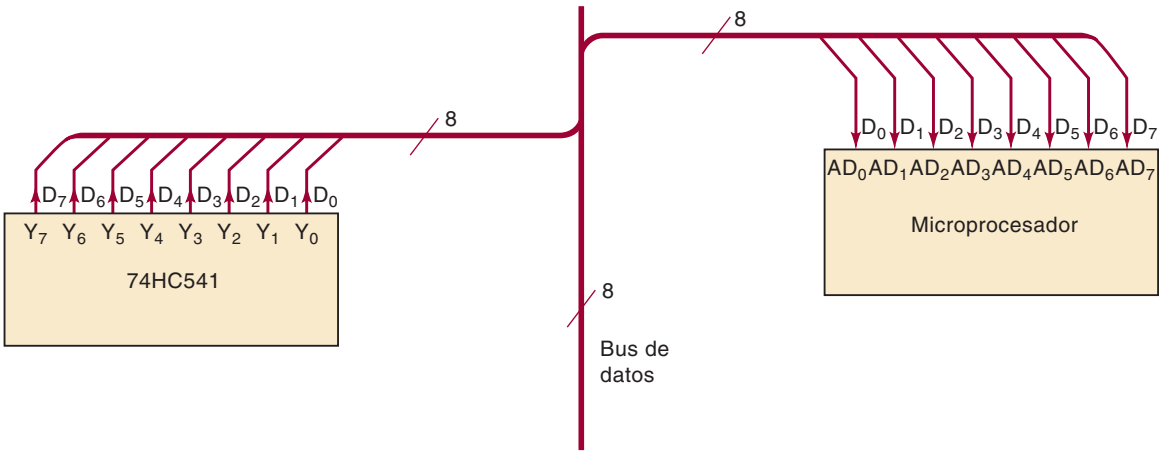


FIGURA 9-49 Método de agrupamiento para la representación simplificada de las conexiones del bus de datos. El "/8" denota un bus de datos de ocho líneas.

Buses bidireccionales

Cada registro en la figura 9-44 tiene sus entradas y salidas conectadas al bus de datos, de manera que las correspondientes entradas y salidas están en corto. Por ejemplo, cada registro tiene la salida O_2 conectada a la entrada D_2, debido a su conexión común con DB_2. Desde luego esto no sería cierto si se conectaran controladores de bus externos entre las salidas del registro y el bus de datos.

Como es común que las entradas y salidas se conecten entre sí en los sistemas de bus, los fabricantes de CIs han desarrollado circuitos integrados que conectan las entradas y salidas entre sí de manera *interna* al chip, para poder reducir el número de terminales del CI y el número de conexiones al bus. La figura 9-50 ilustra esto para un registro de cuatro bits. Las líneas de entrada de datos separadas (D_0 a D_3) y las líneas de salida (O_0 a O_3) se sustituyeron por las líneas de entrada/salida (I/O_0 a I/O_3).

Cada línea de E/S funcionará como una entrada o una salida, dependiendo de los estados de las líneas de habilitación. Por ende, se les conoce como **líneas bidireccionales de datos**. El 74ALS299 es un registro de ocho bits con líneas de E/S comunes. Muchos CIs de memoria y microprocesadores tienen transferencia bidireccional de datos.

En el capítulo 12 retomaremos el importante tema de los buses de datos, y veremos con detalle los sistemas de memoria.

FIGURA 9-50 Registro bidireccional conectado al bus de datos.

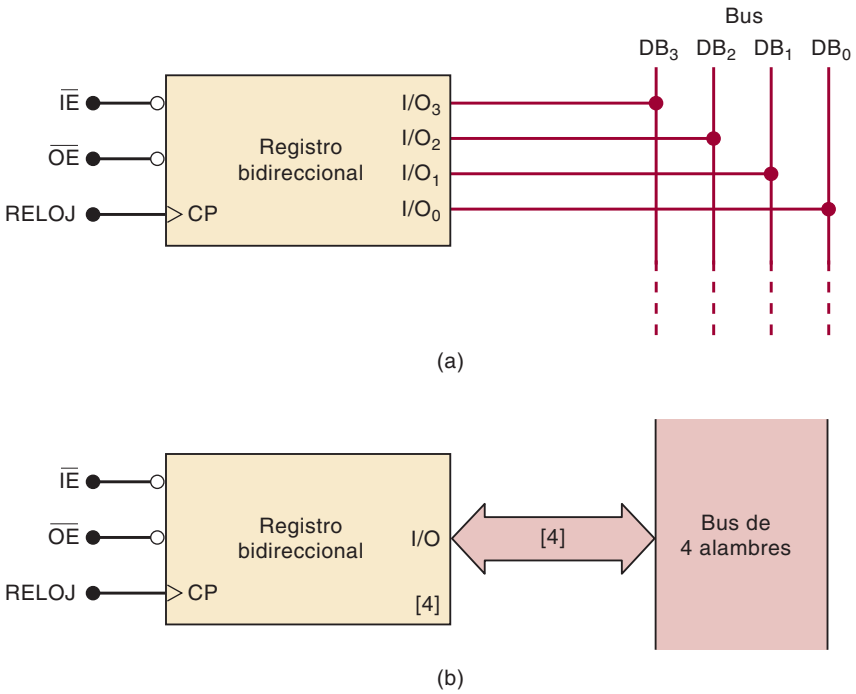
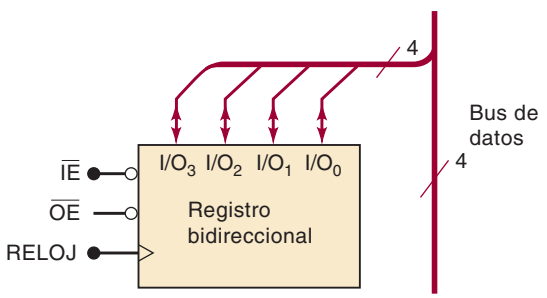


FIGURA 9-51



PREGUNTAS DE REPASO

1. ¿Qué ocurrirá si $\overline{OE}_A = \overline{OE}_B$ = BAJO en la figura 9-44?
2. ¿Qué nivel lógico hay en una línea del bus de datos cuando todos los dispositivos conectados al bus están deshabilitados?
3. ¿Cuál es la función de un controlador de bus?
4. ¿Cuáles son las razones de tener registros con líneas de E/S comunes?
5. Vuelva a dibujar la figura 9-50(a) utilizando la representación de líneas agrupadas. (La respuesta se muestra en la figura 9-51.)

9-15 DECODIFICADORES MEDIANTE EL USO DE HDL

En la sección 9-1 se introdujo el decodificador como un dispositivo que puede reconocer un número binario en su entrada y activar su correspondiente salida. En específico se presentó el decodificador 74138 1 de 8, el cual utiliza tres entradas binarias para activar una de las ocho salidas cuando el chip está habilitado. Para poder estudiar los métodos de HDL para implementar los tipos de dispositivos digitales que se cubren en este capítulo nos enfocaremos principalmente en las

piezas MSI convencionales, las cuales describimos antes. No sólo hemos descrito ya la operación de estos dispositivos en este libro; también tenemos a nuestra disposición más material en los libros de datos. En todos estos casos es vital que usted comprenda lo que se supone debe hacer el dispositivo antes de tratar de disectar el código de HDL que lo describe.

Por ejemplo, en la práctica actual no recomendamos que se escriba nuevo código para realizar la tarea de un 74138. Después de todo hay una macrofunción disponible que funciona justo igual que esta pieza estándar. El uso de estos dispositivos como ejemplos y la demostración de las técnicas de HDL que se utilizan para crearlos nos abre la puerta para adornar estos dispositivos de tal forma que pueda describirse un circuito que se adapte en forma única a la aplicación que se tenga en mente. En algunos casos añadiremos nuestros propios adornos a un circuito que ya se haya descrito; en otros casos describiremos una versión más simple de una pieza para poder enfocarnos en el principio básico en HDL y evitar otras características confusas.

Los métodos utilizados para definir las entradas y salidas deben tomar en cuenta el propósito de estas señales. En el caso de un decodificador 1 de 8 como el 74138 que se describe en la figura 9-3, existen tres entradas de habilitación ($\overline{E}_1$, $\overline{E}_2$ y $\overline{E}_3$) que deben describirse como entradas individuales para el dispositivo. Por otro lado, las entradas binarias que se van a decodificar (A_2, A_1, A_0) deben describirse como números de tres bits. Las salidas pueden describirse como ocho bits individuales. También se pueden describir como un arreglo de ocho bits, en donde la salida 0 se representa mediante el elemento 0 en el arreglo, y así sucesivamente hasta la salida 7, representada por el elemento 7. Dependiendo de la manera en que se escriba el código, una estrategia puede ser más fácil de escribir que la otra. Por lo general el uso de nombres individuales puede hacer más claro el propósito de cada bit de E/S y el uso de arreglos de bits facilita la escritura del código.

Cuando una aplicación tal como un decodificador exige una respuesta única por parte del circuito en correspondencia a cada combinación de sus variables de entrada, los dos métodos que se adaptan mejor a este propósito son la instrucción CASE y la tabla de verdad (TABLE). El aspecto interesante de este decodificador es que la respuesta de salida debe ocurrir sólo cuando *todas* las señales de habilitación estén activadas. Si cualquiera de las señales de habilitación no se encuentra en su estado activo, todas las salidas deberán cambiar al nivel ALTO. Cada uno de los ejemplos que veremos a continuación demostrará métodos para decodificar el número de entrada sólo cuando *todas* las señales de habilitación estén activadas.

DECODIFICADORES EN AHDL

AHDL

En la figura 9-52 se muestra la primera ilustración de un decodificador en AHDL, la cual tiene el propósito de demostrar el uso de una instrucción CASE que se evalúa sólo bajo la condición de que todas las señales de habilitación estén activas. Todas las salidas deben regresar al nivel ALTO tan pronto como se desactive cualquier señal de habilitación. Este ejemplo también ilustra una forma de lograr eso sin tener que asignar en forma explícita un valor a cada salida para cada caso, y se utilizan bits de salida con nombres individuales.

La línea 3 define el número binario de tres bits que se decodificará. La línea 4 define las tres entradas de habilitación y la línea 5 nombra cada salida de manera específica. La propiedad única de esta solución es el uso de la palabra clave **DEFAULTS** en AHDL (líneas 10 a 13) para establecer un valor para las variables que no se especifican en otras partes del código. Esta maniobra permite que cada caso obligue a que un bit cambie a BAJO sin declarar en forma específica que los demás deben cambiar al nivel ALTO.

La siguiente ilustración, en la figura 9-53, tiene el propósito de demostrar el mismo decodificador, pero mediante el uso del método de la tabla de verdad. Observe que las salidas se definen como arreglos de bits, pero se siguen numerando de *y[7]* hasta *y[0]*. El aspecto único de este código es el uso de los valores "no importa" en la tabla de verdad. La línea 11 se utiliza para concatenar los seis bits

FIGURA 9-52
Equivalente del decodificador 74138 en AHDL.

```
 1  SUBDESIGN fig9 52
 2  (
 3      a[2..0]                  :INPUT;        entradas binarias
 4      e3, e2bar, e1bar         :INPUT;        entradas de habilitación
 5      y7,y6,y5,y4,y3,y2,y1,y0  :OUTPUT;       salidas decodificadas
 6  )
 7  VARIABLE
 8      habilita                   :NODE;
 9  BEGIN
10      DEFAULTS
11         y7=VCC;y6=VCC;y5=VCC;y4=VCC;
12         y3=VCC;y2=VCC;y1=VCC;y0=VCC;    valores predeterminados:
                                           todas las salidas en ALTO
13      END DEFAULTS;
14      habilita = e3 & !e2bar & !e1bar;    todas las entradas de
                                           habilitacion activadas
15      IF habilita    THEN
16         CASE a[] IS
17            WHEN 0    =>    y0 = GND;
18            WHEN 1    =>    y1 = GND;
19            WHEN 2    =>    y2 = GND;
20            WHEN 3    =>    y3 = GND;
21            WHEN 4    =>    y4 = GND;
22            WHEN 5    =>    y5 = GND;
23            WHEN 6    =>    y6 = GND;
24            WHEN 7    =>    y7 = GND;
25         END CASE;
26      END IF;
27  END;
```

FIGURA 9-53
Decodificador en AHDL mediante el uso de TABLE.

```
 1  SUBDESIGN fig9 53
 2  (
 3      a[2..0]           :INPUT;     entradas del decodificador
 4      e3, e2bar, e1bar  :INPUT;     entradas de habilitación
 5      y[7..0]           :OUTPUT;    salidas decodificadas
 6  )
 7  VARIABLE
 8     entradas[5..0]     :NODE;      todas las 6 entradas combinadas
 9
10  BEGIN
11     entradas[] = (e3, e2bar, e1bar, a[]);    concatena las entradas
12     TABLE
13        entradas[]       =>     y[];
14        B"0XXXXX"       =>     B"11111111";       e1 no está habilitada
15        B"X1XXXX"       =>     B"11111111";       e2bar deshabilitada
16        B"XX1XXX"       =>     B"11111111";       e3bar deshabilitada
17        B"100000"       =>     B"11111110";       Y0 activa
18        B"100001"       =>     B"11111101";       Y1 activa
19        B"100010"       =>     B"11111011";       Y2 activa
20        B"100011"       =>     B"11110111";       Y3 activa
21        B"100100"       =>     B"11101111";       Y4 activa
22        B"100101"       =>     B"11011111";       Y5 activa
23        B"100110"       =>     B"10111111";       Y6 activa
24        B"100111"       =>     B"01111111";       Y7 activa
25     END TABLE;
26  END;
```

de entrada en una sola variable (arreglo de bits) llamada *entradas[ ]*. Observe que en las líneas 14, 15 y 16 de la tabla sólo se especifica el valor de un bit como 1 o 0. Los demás se encuentran en el estado "no importa" (X). La línea 14 dice, "mientras que *e3* no esté habilitada, no importa lo que estén haciendo las demás entradas; las salidas estarán en ALTO". Las líneas 15 y 16 hacen lo mismo, asegurando que si *e2bar* o *e1bar* están en ALTO (deshabilitadas), las salidas estarán en ALTO. Las líneas 17 a la 24 establecen que mientras los primeros tres bits (de habilitación) sean "100", se activará la salida apropiada del decodificador para corresponder con los tres bits inferiores de *entradas[ ]*.

DECODIFICADORES EN VHDL

La solución con VHDL que se presenta en la figura 9-54 utiliza en esencia el método de la tabla de verdad. La estrategia clave en esta solución implica la concatenación de los tres bits de habilitación (*e3, e2bar, e1bar*) con la entrada binaria *a* en la línea 11. La asignación de señal selecta en VHDL se utiliza para asignar un valor a una señal cuando está presente una combinación específica de entradas. La línea 12 (WITH entradas SELECT) indica que estamos usando el valor de la señal intermedia *entradas* para determinar cuál valor se asigna a *y*. Cada una de las salidas *y* se lista en las líneas 13 a 20. Observe que sólo las combinaciones que comienzan con 100 van después de la cláusula WHEN en las líneas 13 a 20. Esta combinación de *e3*, *e2bar* y *e1bar* es necesaria para activar cada una de las señales de habilitación. La línea 21 asigna un estado deshabilitado a cada salida cuando hay una combinación distinta de 100 presente en las entradas de habilitación.

```
1   ENTITY fig9 54 IS
2   PORT(
3        a                 :IN BIT VECTOR (2 DOWNTO 0);
4        e3, e2bar, e1bar  :IN BIT;
5        u                 :OUT BIT VECTOR (7 DOWNTO 0)
6       );
7   END fig9 54 ;
8   ARCHITECTURE verdad OF fig9 54 IS
9   SIGNAL entradas:  BIT VECTOR (5 DOWNTO 0);   combina señales de habilitación con
10     BEGIN
11        entradas <= e3 & e2bar & e1bar & a;
12        WITH entradas SELECT
13           y  <= "11111110"  WHEN "100000",    Y0 activa
14                 "11111101"  WHEN "100001",    Y1 activa
15                 "11111011"  WHEN "100010",    Y2 activa
16                 "11110111"  WHEN "100011",    Y3 activa
17                 "11101111"  WHEN "100100",    Y4 activa
18                 "11011111"  WHEN "100101",    Y5 activa
19                 "10111111"  WHEN "100110",    Y6 activa
20                 "01111111"  WHEN "100111",    Y7 activa
21                 "11111111"  WHEN OTHERS;      deshabilitada
22      END verdad;
```

FIGURA 9-54 Equivalente en VHDL del decodificador 74138.

PREGUNTAS DE REPASO

1. ¿Cuál es el propósito de las tres entradas *e3*, *e2bar* y *e1bar*?
2. Nombre dos métodos en AHDL para describir la operación de un decodificador.
3. Nombre dos métodos en VHDL para describir la operación de un decodificador.

9-16 EL DECODIFICADOR/CONTROLADOR DE 7 SEGMENTOS EN HDL

En la sección 9-2 describimos un decodificador/controlador de BCD a 7 segmentos. El número de pieza estándar para el circuito descrito es 7447. En esta sección veremos el código de HDL necesario para producir un dispositivo que cumple con los mismos criterios que el 7447. Recuerde que $\overline{BI}$ (entrada de borrado) es el control de anulación que apaga todos los segmentos sin importar los demás niveles de entrada. La entrada $\overline{LT}$ (prueba de lámpara) se utiliza para probar todos los segmentos en la pantalla; para ello se alumbran. La señal $\overline{RBO}$ (salida de borrado de rizo) está diseñada para cambiar a BAJO cuando $\overline{RBI}$ (entrada de borrado de rizo) está en BAJO y el valor de entrada BCD es 0. Por lo general, en aplicaciones de pantalla con múltiples dígitos cada terminal $\overline{RBO}$ se conecta a la terminal $\overline{RBI}$ del siguiente dígito a la derecha. Esta configuración crea la característica de borrar todos los ceros a la izquierda en un valor de visualización sin borrar los ceros en medio de un número. Por ejemplo, el número 2002 se visualizaría como 2002, pero el número 0002 no se visualizaría como 0002, sino como _ _ _ 2. Una característica del 7447 que sería difícil de replicar en HDL es la terminal de combinación de entrada/salida llamada $\overline{BI}/\overline{RBO}$. En vez de complicar el código hemos decidido crear una entrada ($\overline{BI}$) y una salida ($\overline{RBO}$) separadas, en dos terminales distintas. En esta discusión no intentamos replicar los caracteres de visualización que no sean BCD de un 7447; tan sólo queremos borrar todos los segmentos para los valores mayores de 9.

Hay que realizar varias decisiones al diseñar un circuito tal como éste. La primera implica el tipo de pantalla que deseamos usar. Si es un cátodo común, entonces un 1 lógico enciende el segmento de LED. Si es un ánodo común, entonces se requiere un 0 lógico para encender un segmento. A continuación debemos decidir sobre el tipo de entradas, salidas y variables intermedias. Hemos decidido que a las salidas para cada segmento individual se les debe asignar un nombre de bit (*a*-*g*) en vez de usar un arreglo de bits. Esta distribución ayudará a ver todo más claro cuando se conecte la pantalla con el CI. Estos bits individuales pueden agruparse como un conjunto de bits y valores binarios asignados como se hizo en AHDL, o puede utilizarse un arreglo de bits de variable intermedia para que sea conveniente asignar los siete niveles de bits en una sola instrucción, como se hizo en VHDL. Las entradas BCD se tratan como un número de cuatro bits y los controles de borrado son bits individuales. La otra cuestión que afecta en forma considerable a los patrones de bits en el código de HDL es la decisión arbitraria del orden de los nombres de segmento *a*-*g*. En esta discusión hemos asignado el segmento *a* al bit de más a la izquierda en el patrón de bits binario, y los bits se mueven en orden alfabético de izquierda a derecha.

Algunos de los controles deben tener precedencia sobre los demás controles. Por ejemplo, el control $\overline{LT}$ (prueba de lámpara) debe anular cualquier visualización de dígitos ordinaria y el control $\overline{BI}$ (entrada de borrado) debe anular incluso hasta la entrada de prueba de lámpara. En estas ilustraciones se utiliza la estructura de control IF/ELSE para establecer la precedencia. La primera condición que se evalúe como verdadera determinará la salida resultante, sin importar los demás niveles de entrada. Las instrucciones ELSE subsiguientes no tendrán efecto, lo cual explica por qué el código prueba primero para *BI*, después *LT*, luego *RBI* y por último determina el patrón de segmentos correcto.

AHDL

DECODIFICADOR/CONTROLADOR EN AHDL

En la figura 9-55 se muestra el código de AHDL para este circuito. AHDL permite agrupar los bits de salida en un conjunto; para ello se separan los bits con comas y se encierran entre paréntesis. Un grupo de estados binarios puede asignarse en forma directa a estos conjuntos de bits, como se muestra en las líneas 9, 11, 13 y 15. Esta convención evita la necesidad de una variable intermedia y es mucho más corta que ocho instrucciones de asignación separadas. La característica TABLE de AHDL es útil en esta aplicación para correlacionar un valor BCD de entrada con un patrón de bits de 7 segmentos.

```
1   SUBDESIGN fig9 55
2   (
3      bcd[3..0]          :INPUT;          número de 4 bits
4      lt, bi, rbi        :INPUT;          3 controles independientes
5      a,b,c,d,e,f,g,rbo  :OUTPUT;         salidas individuales
6   )
7   BEGIN
8       IF !bi THEN
9           (a,b,c,d,e,f,g,rbo) = (1,1,1,1,1,1,1,0);    % borra todos %
10      ELSIF     !lt THEN
11          (a,b,c,d,e,f,g,rbo) = (0,0,0,0,0,0,0,1);    % prueba segmentos %
12      ELSIF !rbi & bcd[] == 0  THEN
13          (a,b,c,d,e,f,g,rbo) = (1,1,1,1,1,1,1,0);    % borra 0 s a la izquierda %
14      ELSIF bcd[] > 9 THEN
15          (a,b,c,d,e,f,g,rbo) = (1,1,1,1,1,1,1,1);    % borra entrada no BCD %
16      ELSE
17          TABLE                    % visualiza patrón de Anodo común de 7 segmentos %
18          bcd[]     =>    a,b,c,d,e,f,g,rbo;
19          0         =>    0,0,0,0,0,0,1,1;
20          1         =>    1,0,0,1,1,1,1,1;
21          2         =>    0,0,1,0,0,1,0,1;
22          3         =>    0,0,0,0,1,1,0,1;
23          4         =>    1,0,0,1,1,0,0,1;
24          5         =>    0,1,0,0,1,0,0,1;
25          6         =>    1,1,0,0,0,0,0,1;
26          7         =>    0,0,0,1,1,1,1,1;
27          8         =>    0,0,0,0,0,0,0,1;
28          9         =>    0,0,0,1,1,0,0,1;
29          END TABLE;
30      END IF;
31  END;
```

FIGURA 9-55 Decodificador de pantalla BCD de 7 segmentos en AHDL.

DECODIFICADOR/CONTROLADOR EN VHDL

La figura 9-56 muestra el código de VHDL para este circuito. Esta ilustración demuestra el uso de VARIABLE en vez de SIGNAL. Una VARIABLE puede considerarse como un pedazo de papel que se utiliza para anotar algunos números que se necesitarán después. Por otro lado, una señal (SIGNAL) se considera por lo general como un alambre que conecta dos puntos en el circuito. En la línea 12 se utiliza la palabra clave VARIABLE para declarar *segmentos* como un vector de siete bits. En especial, observe el orden de los índices para esta variable, los cuales se declaran como 0 TO 6. En VHDL esto significa que el elemento 0 aparece en el extremo izquierdo del patrón binario de bits y que el elemento 6 aparece en el extremo derecho. Esto es el opuesto exacto de la manera en la que se han presentado las variables en la mayoría de los ejemplos en este texto, pero es importante tener en cuenta el significado de la instrucción de declaración en VHDL. Para esta ilustración, el segmento *a* es el bit 0 (a la izquierda), el segmento *b* es el bit 1 (nos desplazamos hacia la derecha), y así en lo sucesivo.

Observe que en la línea 3 la entrada BCD se declara como un entero (INTEGER). Esto nos permite referirnos a ella como un valor numérico en decimal, en vez de limitarnos a las referencias de los patrones de bits. Aquí se emplea un bloque PROCESS para poder utilizar las instrucciones IF/ELSE y establecer la precedencia de una entrada sobre la otra. Observe que la lista de sensibilidad contiene todas las entradas. El código dentro del bloque PROCESS describe la operación del compor-

VHDL

```
1   ENTITY fig9 56 IS
2   PORT  (
3            bcd                :IN INTEGER RANGE 0 TO 15;
4            lt, bi, rbi        :IN BIT;
5            a,b,c,d,e,f,g,rbo :OUT BIT
6         );
7   END fig9 56 ;
8
9   ARCHITECTURE vhdl OF fig9 56 IS
10  BEGIN
11  PROCESS  (bcd, lt, bi, rbi)
12  VARIABLE  segmentos       :BIT VECTOR (0 TO 6);
13      BEGIN
14         IF  bi = '0' THEN
15            segmentos := "1111111";  rbo <= '0';    borra todo
16         ELSIF lt = '0' THEN
17            segmentos := "0000000";  rbo <= '1';    prueba segmentos
18         ELSIF (rbi = '0' AND bcd = 0) THEN
19            segmentos := "1111111";  rbo <= '0';    borra 0's a la izquierda
20         ELSE
21            rbo <= '1';
22            CASE bcd IS       visualiza patrón de Anodo común de 7 segmentos
23               WHEN 0      => segmentos := "0000001";
24               WHEN 1      => segmentos := "1001111";
25               WHEN 2      => segmentos := "0010010";
26               WHEN 3      => segmentos := "0000110";
27               WHEN 4      => segmentos := "1001100";
28               WHEN 5      => segmentos := "0100100";
29               WHEN 6      => segmentos := "1100000";
30               WHEN 7      => segmentos := "0001111";
31               WHEN 8      => segmentos := "0000000";
32               WHEN 9      => segmentos := "0001100";
33               WHEN OTHERS => segmentos := "1111111";
34            END CASE;
35         END IF;
36      a <= segmentos(0);   asigna bits del arreglo a las terminales de salida
37      b <= segmentos(1);
38      c <= segmentos(2);
39      d <= segmentos(3);
40      e <= segmentos(4);
41      f <= segmentos(5);
42      g <= segmentos(6);
43          END PROCESS;
44  END vhdl;
```

FIGURA 9-56 Decodificador de pantalla BCD de 7 segmentos en VHDL.

tamiento del circuito que se necesita cada vez que una de las entradas en la lista de sensibilidad cambia de estado. Otro punto muy importante en esta ilustración es el operador de asignación para las variables. Por ejemplo, observe la instrucción *segmentos := "1111111"* en la línea 15. El operador de asignación de variables := se utiliza para las variables en vez del operador <= que se utilizaba para las asignaciones de las señales. En las líneas 36 a 42, los bits individuales que se establecieron en las decisiones IF/ELSE se asignan a los bits de salida apropiados.

PREGUNTAS DE REPASO

1. ¿Qué característica de un 7447 es muy difícil de duplicar en el hardware de un PLD y en código de HDL?
2. ¿Están diseñadas estas ilustraciones para controlar pantallas de 7 segmentos de ánodo común o de cátodo común?
3. ¿Cómo se da precedencia a ciertas entradas (por ejemplo, prueba de lámpara) sobre otras (por ejemplo, RBI) en el código de HDL en esta sección?

9-17 CODIFICADORES MEDIANTE EL USO DE HDL

En la sección 9-4 hablamos sobre los codificadores y los codificadores con prioridad. Sin duda existen semejanzas entre los decodificadores y los codificadores. Los decodificadores toman un número binario y activan una salida que corresponde a ese número. Un codificador funciona en sentido opuesto, ya que monitorea una de sus diversas entradas; cuando una de las entradas se activa, produce un número binario que corresponde a esa entrada. Si se activa más de una de sus entradas al mismo tiempo, un codificador con prioridad ignora la entrada de menor significado y produce el valor binario que corresponde a la entrada más significativa. En otras palabras, da prioridad a las entradas más significativas sobre las menos significativas. Esta sección se enfoca en los métodos que pueden utilizarse en HDL para describir circuitos que tienen esta característica de prioridad para algunas entradas sobre las otras.

Otro concepto muy importante que presentamos en el capítulo 8 es el circuito de salida triestado. Los dispositivos con salidas triestado puede producir un nivel ALTO o BAJO lógico, justo igual que un circuito normal, cuando su salida se habilita. No obstante, estos dispositivos pueden deshabilitar sus salidas, lo cual los coloca en un estado "desconectado" o de alta impedancia. Esto es muy importante para los dispositivos que se conectan a buses comunes, como se describió en la sección 9-12. La siguiente pregunta lógica es "¿cómo se describen las salidas triestado mediante el uso de HDL?" Esta sección incorpora las entradas triestado en el diseño del codificador para tratar con esta cuestión. Para poder mantener la discusión enfocada en los aspectos básicos creamos un circuito que emula al codificador con prioridad 74147, pero le agregamos la característica de tener salidas triestado en ALTO. Dejaremos las otras características tales como las entradas y salidas en cascada (como las que se encuentran en un 74148) para que usted las implemente después. En la figura 9-57 se muestra un símbolo para el circuito que estamos describiendo. Como todas las entradas están etiquetadas de una manera muy similar a la notación en arreglos de bits, tiene sentido utilizar un arreglo de bits para describir las entradas del codificador. La señal de habilitación triestado debe ser un solo bit y las salidas codificadas pueden describirse como un valor numérico entero.

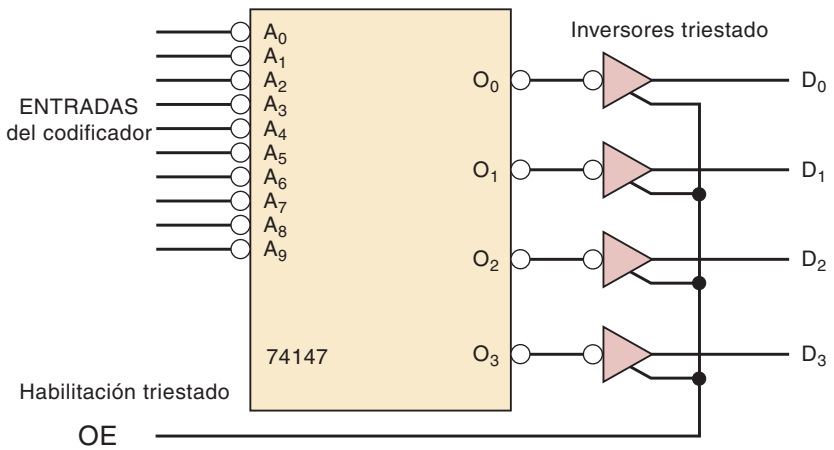


FIGURA 9-57
Descripción gráfica de un codificador con salidas triestado.

AHDL

CODIFICADOR EN AHDL

El punto más importante a deducir de la figura 9-58 es el método de establecer la prioridad, pero hay que observar también las asignaciones de E/S. Las descripciones de entrada/salida en AHDL no proporcionan un tipo separado para los enteros, pero permiten hacer referencia a un arreglo de bits como si fuera un entero. En consecuencia, la línea 4 describe las salidas como un arreglo de bits. En esta ilustración se utiliza una estructura TABLE que es muy similar a las tablas que se encuentran con frecuencia en los libros de datos, en la cual se describe la operación de este circuito. La clave para esta tabla es el uso del estado "no importa" (X) en las entradas. La prioridad se describe mediante la forma en que posicionamos los estados "no importa" en la tabla de verdad. Por ejemplo, al leer la línea 15 podemos ver que tan pronto como nos encontramos una entrada activa (en BAJO en la entrada *a[4]*), los bits de entrada de menor orden no importan. Se ha determinado que la salida es un 4. Las salidas triestado son posibles gracias al uso de la función primitiva integrada : TRI en la línea 6. Esta línea asigna los atributos de un búfer triestado a la variable de nombre *bufer*. Recuerde que ésta es la misma forma en que se describe un flip-flop en AHDL. Los puertos de un búfer triestado son bastante simples. Representan la entrada (*in*), la salida (*out*) y la habilitación de salida triestado (*oe*).

FIGURA 9-58
Codificador de prioridad con salidas triestado en AHDL.

```
1   SUBDESIGN fig9 58
2   (
3      a[9..0], oe          :INPUT;
4      d[3..0]              :OUTPUT;
5   )
6   VARIABLE bufer[3..0]   :TRI;
7   BEGIN
8      TABLE
9         a[]             => bufer[].in;
10        B"1111111111"   => B"1111";       no hay entradas activas
11        B"1111111110"   => B"0000";       0
12        B"111111110X"   => B"0001";       1
13        B"11111110XX"   => B"0010";       2
14        B"1111110XXX"   => B"0011";       3
15        B"111110XXXX"   => B"0100";       4
16        B"11110XXXXX"   => B"0101";       5
17        B"1110XXXXXX"   => B"0110";       6
18        B"110XXXXXXX"   => B"0111";       7
19        B"10XXXXXXXX"   => B"1000";       8
20        B"0XXXXXXXXX"   => B"1001";       9
21     END TABLE;
22     bufer[].oe = oe;         conecta línea de habilitación
23     d[] = buffer[].out;      conecta salidas
24  END;
```

La siguiente ilustración (figura 9-59) utiliza la instrucción IF/ELSE para establecer la prioridad, en forma muy parecida al método que demostramos en el ejemplo del decodificador de 7 segmentos. La primera condición IF que se evalúe como VERDADERO hará entonces (THEN) que se aplique el valor correspondiente a las entradas del búfer triestado. La prioridad se establecerá mediante el orden en el cual listemos las condiciones IF. Observe que empiezan con la entrada 9, la entrada de mayor orden. Esta ilustración agrega otra característica de colocar las salidas en el estado de alta impedancia cuando no se activa ninguna entrada. La línea 20

```
1   SUBDESIGN fig9 59
2   (
3      sw[9..0], oe   :INPUT;
4      d[3..0]        :OUTPUT;
5   )
6   VARIABLE
7      buferes[3..0]  :TRI;
8   BEGIN
9      IF    !sw[9]   THEN  buferes[].in = 9;
10     ELSIF !sw[8]   THEN  buferes[].in = 8;
11     ELSIF !sw[7]   THEN  buferes[].in = 7;
12     ELSIF !sw[6]   THEN  buferes[].in = 6;
13     ELSIF !sw[5]   THEN  buferes[].in = 5;
14     ELSIF !sw[4]   THEN  buferes[].in = 4;
15     ELSIF !sw[3]   THEN  buferes[].in = 3;
16     ELSIF !sw[2]   THEN  buferes[].in = 2;
17     ELSIF !sw[1]   THEN  buferes[].in = 1;
18     ELSE                buferes[].in = 0;
19     END IF;
20     buferes[].oe = oe & sw[]!=b"1111111111";     se habilita en cualquier entrada
21     d[] = buffers[].out;                         se conecta a las salidas
22  END;
```

FIGURA 9-59 Codificador de prioridad en AHDL, mediante el uso de IF/ELSE.

muestra que las señales de habilitación de salida se activarán solo cuando se activen la terminal *oe* y una de las entradas. Otro punto de interés en esta ilustración es el uso de la notación en arreglo de bits para describir entradas individuales. Por ejemplo, la línea 9 establece que si (IF) se activa (en BAJO) la entrada del interruptor 9, entonces (THEN) se asignará el valor 9 (en binario, por supuesto) a las entradas del búfer triestado.

CODIFICADOR EN VHDL

En esta descripción de un codificador de prioridad se demuestran dos técnicas muy importantes en VHDL. La primera es el uso de salidas triestado en VHDL, y la segunda es un nuevo método de describir la prioridad. La figura 9-60 muestra las definiciones de entradas/salidas para este circuito codificador. Observe en la línea 6 que los interruptores de entrada se definen como vectores de bit con índices del 9 al 0. Observe además que la salida *d* se define como un arreglo de bits del estándar IEEE (tipo std_logic_vector). Esta definición es necesaria para permitir el uso de estados de alta impedancia (triestado) en las salidas y también explica la necesidad de las instrucciones LIBRARY y USE en las líneas 1 y 2. Como se mencionó antes, un punto muy importante de esta ilustración es el método de describir la precedencia para las entradas. Este código utiliza la **instrucción de asignación de señal condicional**, que empieza en la línea 14 y continúa hasta la línea 24. En la línea 14 asigna el valor que aparece a la derecha de <= a la variable *d* que está a la izquierda, suponiendo que la condición que va después de WHEN sea verdadera. Si esta cláusula no es verdadera se evalúan las cláusulas que van después de ELSE una a la vez, hasta encontrar una que sea verdadera. Entonces se asignará a *d* el valor que va antes de WHEN. Un atributo muy importante de la instrucción de asignación de señal condicional es la evaluación secuencial. La precedencia de estas instrucciones se establece en base al orden en el que se listan. Observe que en esta ilustración, la primera condición que se evalúa (línea 14) es la habilitación de las salidas triestado. Si recuerda del capítulo 8, los tres estados de una salida triestado son ALTO, BAJO y alta impedancia, que se conoce también como Hi-Z. Cuando se asigna el valor "ZZZZ" a la salida, cada salida se encuentra en el estado de alta impedancia. Si las

```
1   LIBRARY    ieee;
2   USE ieee.std logic 1164.ALL;
3
4   ENTITY fig9 60 IS
5   PORT(
6        sw    :IN BIT VECTOR (9 DOWNTO 0);     no se necesita lógica estándar
7        oe    :IN BIT;                         no se necesita lógica estándar
8        d     :OUT STD LOGIC VECTOR (3 DOWNTO 0)     lógica estándar para hi z
9       );
10  END fig9 60;
11
12  ARCHITECTURE a OF fig9 60 IS
13     BEGIN
14        d  <= "ZZZZ"  WHEN  ((oe = '0') OR (sw = "1111111111")) ELSE
15              "1001"  WHEN  sw(9) = '0' ELSE
16              "1000"  WHEN  sw(8) = '0' ELSE
17              "0111"  WHEN  sw(7) = '0' ELSE
18              "0110"  WHEN  sw(6) = '0' ELSE
19              "0101"  WHEN  sw(5) = '0' ELSE
20              "0100"  WHEN  sw(4) = '0' ELSE
21              "0011"  WHEN  sw(3) = '0' ELSE
22              "0010"  WHEN  sw(2) = '0' ELSE
23              "0001"  WHEN  sw(1) = '0' ELSE
24              "0000"  WHEN  sw(0) = '0';
25  END a;
```

FIGURA 9-60 Codificador con prioridad en VHDL, mediante el uso de asignaciones de señales condicionales.

salidas se van a deshabilitar (Hi-Z), entonces no importa ninguna de las otras codificaciones. La línea 15 evalúa la entrada de mayor prioridad, la cual es el bit 9 del arreglo de entrada *sw*. Si está activa (en BAJO), entonces se produce como salida un valor de 9 sin importar que se activen otras entradas al mismo tiempo.

PREGUNTAS DE REPASO

1. Nombre dos métodos en AHDL para dar prioridad a unas entradas sobre otras.
2. Nombre dos métodos en VHDL para dar prioridad a unas entradas sobre otras.
3. ¿Cómo se implementan las salidas triestado en AHDL?
4. ¿Cómo se implementan las salidas triestado en VHDL?

9-18 MULTIPLEXORES Y DEMULTIPLEXORES EN HDL

Un multiplexor es un dispositivo que actúa como un interruptor selector para señales digitales. Las entradas de selección se utilizan para especificar el canal de entrada que se va a "conectar" a las terminales de salida. Un demultiplexor funciona en sentido opuesto, al tomar una señal digital como entrada y distribuirla a una de sus salidas. La figura 9-61 muestra un sistema de multiplexor/demultiplexor con cuatro canales de entrada de datos. Cada entrada es un número de cuatro bits. Estos dispositivos no son exactamente iguales a cualquiera de los multiplexores o demultiplexores que describimos antes en este capítulo, pero operan de la misma forma. El sistema en esta ilustración permite que las cuatro señales digitales compartan una "canalización" común para poder transportar datos de un punto al otro. Las entradas de selección se utilizan para decidir cuál señal va a pasar por la canalización en un momento dado.

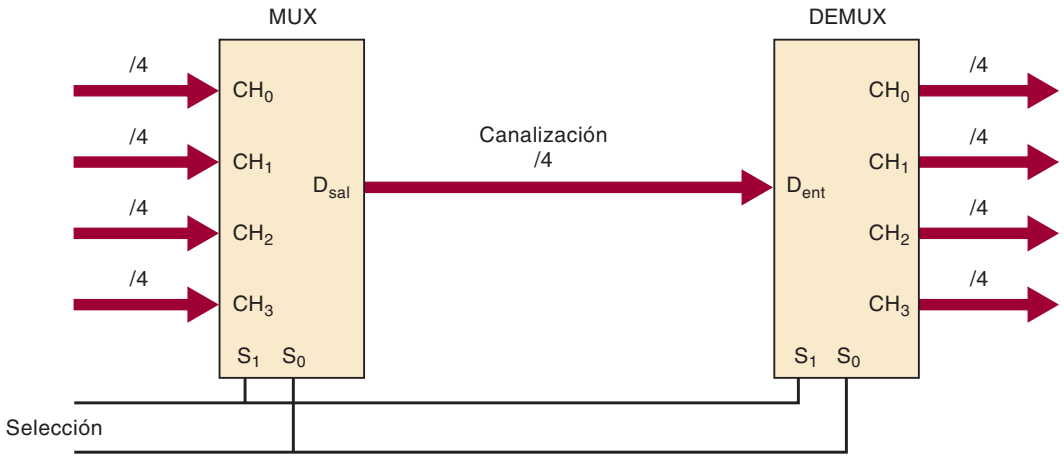


FIGURA 9-61 Cuatro canales de datos que comparten una ruta de datos común.

En esta sección examinaremos código para implementar tanto el multiplexor como el demultiplexor. La cuestión clave de HDL tanto en el MUX como en el DEMUX es asignar señales bajo ciertas condiciones. Para el demux, otra de las cuestiones es asignar un estado a las salidas que no estén seleccionadas en cierto momento para distribuir datos. En otras palabras, cuando una salida no se está utilizando para datos (no está seleccionada), ¿queremos que tenga todos los bits en ALTO, todos los bits en BAJO o el estado triestado deshabilitado? En las siguientes descripciones hemos optado por poner a todos los bits en ALTO cuando no esté la salida seleccionada, pero con la estructura que se muestra sería muy sencillo cambiar a una de las otras posibilidades.

MUX Y DEMUX EN AHDL

Primero vamos a implementar el multiplexor. La figura 9-62 describe un multiplexor con cuatro entradas de cuatro bits cada una. A cada canal de entrada se le asigna un nombre de forma que identifique su número de canal. En esta figura, cada entrada se describe como un arreglo de cuatro bits. La entrada de selección (*s[ ]*) requiere dos bits para especificar los cuatro números de canal (0-3). Aquí se utiliza una instrucción CASE para asignar un canal de entrada en forma condicional a las terminales de salida. Por ejemplo, la línea 9 establece que en el caso en el que las entradas de selección (*s[ ]*) se establecen en 0 (es decir, 00 binario), el circuito debe conectar la entrada del canal 0 con la salida de datos. Observe que al asignar

```
1   SUBDESIGN fig9 62
2   (
3     ch0[3..0], ch1[3..0], ch2[3..0], ch3[3..0]:INPUT;
4     s[1..0]                                    :INPUT;    entradas de selección
5     dout[3..0]                                 :OUTPUT;
6   )
7   BEGIN
8      CASE s[] IS
9           WHEN 0 =>     dsal[] = ch0[];
10          WHEN 1 =>     dsal[] = ch1[];
11          WHEN 2 =>     dsal[] = ch2[];
12          WHEN 3 =>     dsal[] = ch3[];
13     END CASE;
14  END;
```

FIGURA 9-62 MUX de cuatro canales × cuatro bits en AHDL.

AHDL

FIGURA 9-63 DEMUX de cuatro canales × cuatro bits en AHDL.

```
1   SUBDESIGN fig9 63
2   (
3      ch0[3..0], ch1[3..0], ch2[3..0], ch3[3..0]  :OUTPUT;
4      s[1..0]                                     :INPUT;
5      dent[3..0]                                  :INPUT;
6   )
7   BEGIN
8      DEFAULTS
9         ch0[] = B"1111";
10        ch1[] = B"1111";
11        ch2[] = B"1111";
12        ch3[] = B"1111";
13     END DEFAULTS;
14
15     CASE S[] IS
16           WHEN 0 =>     ch0[] = dent[];
17           WHEN 1 =>     ch1[] = dent[];
18           WHEN 2 =>     ch2[] = dent[];
19           WHEN 3 =>     ch3[] = dent[];
20      END CASE;
21  END;
```

conexiones, el destino (salida) de la señal está a la izquierda del signo = y el origen (entrada) está a la derecha.

El código del demultiplexor funciona de manera similar, pero sólo tiene un canal de entrada y cuatro canales de salida. También debe asegurar que todas las salidas estén en ALTO cuando no están seleccionadas. En la figura 9-63, las entradas y salidas se declaran como siempre en las líneas 3 a 5. La condición predeterminada para cada canal se especifica después de la palabra clave DEFAULTS, la cual indica al compilador que debe generar un circuito que tendrá niveles ALTOs en las salidas, a menos que se les asigne un valor en forma específica en cualquier otra parte en el código. Si no se especifica esta sección predeterminada, todas las salidas cambiarán en forma automática a su valor predeterminado de BAJO. Observe en las líneas 16 a 19 que la señal de entrada se asigna en forma condicional a uno de los canales de salida. En consecuencia, el canal de salida está a la izquierda del signo = y la señal de entrada está a la derecha.

VHDL

MUX Y DEMUX EN VHDL

La figura 9-64 muestra el código para crear un MUX de cuatro canales con cuatro bits por canal. Las entradas se declaran como arreglos de bits en la línea 3. Podrían haberse declarado con la misma facilidad como enteros, del 0 al 15. De cualquier forma que se declaren las entradas, las salidas deberán ser del mismo tipo. Observe en la línea 4 que la(s) entrada(s) de selección se declaran como un entero decimal del 0 al 3 (el equivalente de los valores binarios 00 a 11). Esto nos permite hacer referencia a este tipo de entrada por su número de canal decimal en el código, con lo cual se facilita su comprensión. Las líneas 11 a 15 utilizan la instrucción de asignación de señal selecta para "conectar" la entrada apropiada con la salida, dependiendo del valor en las entradas de selección. Por ejemplo, la línea 15 establece que debe seleccionarse el canal 3 para conectar las salidas de datos cuando las entradas de selección se establezcan en 3.

El código del demultiplexor funciona de manera similar, pero sólo tiene un canal de entrada y cuatro canales de salida. En la figura 9-65, las entradas y salidas se declaran como siempre en las líneas 3 a 5. Observe que en la línea 3 el tipo de la(s) entrada(s) de selección es entero, justo igual que en el código del MUX en la figura 9-64. La operación de un DEMUX se describe con la misma facilidad, median-

FIGURA 9-64 MUX de cuatro canales × cuatro bits en VHDL.

```
1   ENTITY fig9 64 IS
2   PORT    (
3            ch0, ch1, ch2, ch3  :IN BIT VECTOR (3 DOWNTO 0);
4            s                   :IN INTEGER RANGE 0 TO 3;
5            dsal                :OUT BIT VECTOR (3 DOWNTO 0)
6           );
7   END fig9 64;
8
9   ARCHITECTURE selector OF fig9 64 IS
10      BEGIN
11         WITH s SELECT
12         dsal  <=  ch0 WHEN 0,     cambia el canal 0 a la salida
13                   ch1 WHEN 1,     cambia el canal 1 a la salida
14                   ch2 WHEN 2,     cambia el canal 2 a la salida
15                   ch3 WHEN 3;     cambia el canal 3 a la salida
16      END selector;
```

FIGURA 9-65 DEMUX de cuatro canales × cuatro bits en VHDL.

```
1   ENTITY fig9 65 IS
2   PORT   (
3            s                   :IN INTEGER RANGE 0 TO 3;
4            dent                :IN BIT VECTOR (3 DOWNTO 0);
5            ch0, ch1, ch2, ch3  :OUT BIT VECTOR(3 DOWNTO 0)
6          );
7   END fig9 65;
8
9   ARCHITECTURE selector OF fig9 65 IS
10      BEGIN
11         ch0 <= dent WHEN s = 0  ELSE "1111";
12         ch1 <= dent WHEN s = 1  ELSE "1111";
13         ch2 <= dent WHEN s = 2  ELSE "1111";
14         ch3 <= dent WHEN s = 3  ELSE "1111";
15      END selector;
```

te el uso de varias instrucciones de asignación de señal condicional, como se muestra en las líneas 11 a 14. Anteriormente decidimos que el código para este DEMUX debe asegurar que todas las salidas estén en ALTO cuando no están seleccionadas. Esto se logra mediante la cláusula ELSE de cada asignación de señal condicional. Si no se utiliza la cláusula ELSE, los valores de salida cambiarán en forma automática a su valor predeterminado de BAJO. Por ejemplo, la línea 13 establece que el canal 2 se conectará a las entradas de datos siempre que las entradas de selección se establezcan en 2. Si *s* se establece a cualquier otro valor, entonces el canal 2 se verá obligado a tener todos los bits en ALTO.

PREGUNTAS DE REPASO

1. Para el MUX de cuatro canales por cuatro bits, nombre las entradas de datos, las salidas de datos y las entradas de control que seleccionan un canal de los cuatro.
2. Para el DEMUX de cuatro canales por cuatro bits, nombre las entradas de datos, las salidas de datos y las entradas de control que seleccionan un canal de los cuatro.
3. ¿Cómo se determinan los estados lógicos para los canales que no se seleccionan en el ejemplo de AHDL?
4. ¿Cómo se determinan los estados lógicos para los canales que no se seleccionan en el ejemplo de VHDL?

9-19 COMPARADORES DE MAGNITUD EN HDL

En la sección 9-10 estudiamos un chip comparador de magnitud 7485. Como su nombre lo indica, este dispositivo compara la magnitud de dos números binarios e indica la relación entre los dos (mayor que, menor que, igual a). Se proporcionan entradas de control para el propósito de conectar estos chips en cascada. Los chips se interconectan de manera que el chip que compara los bits de menor orden tenga sus entradas conectadas a las entradas de control del siguiente chip de mayor orden, como se muestra en la figura 9-37. Cuando la etapa del orden más alto detecte que sus entradas de datos tienen igual magnitud, buscará en la siguiente etapa de menor orden y utilizará estas entradas de control para tomar la decisión final. Esto nos da la oportunidad de ver una de las diferencias contundentes entre el uso de los CIs de lógica tradicional y el uso de HDL para diseñar un circuito. Si necesitamos comparar valores más grandes mediante el uso de HDL, lo único que tenemos que hacer es ajustar el tamaño de los puertos de entrada del comparador para que se ajuste a lo que requerimos, en vez de tratar de conectar en cascada varios comparadores de cuatro bits. En consecuencia, no hay necesidad de conectar en cascada los controles de entrada en la versión con HDL.

Hay muchas posibles formas de describir la operación de un comparador. No obstante, es mejor utilizar una instrucción IF/ELSE, ya que cada cláusula IF evalúa una relación entre dos valores, en vez de buscar el valor individual de una variable, como en la instrucción CASE. Las dos entradas que se van a comparar deben declararse en definitiva como valores numéricos. Las tres salidas del comparador deben declararse como bits individuales para poder etiquetar el propósito de cada bit con claridad.

AHDL

COMPARADOR EN AHDL

El código de AHDL en la figura 9-66 sigue el algoritmo que describimos mediante el uso de instrucciones IF/ELSE. Observe en la línea 3 que los valores de datos se declaran como números de cuatro bits. Observe además en las líneas 8, 10 y 11 que pueden utilizarse varias instrucciones para especificar la operación del circuito cuando la cláusula IF es verdadera. Cada instrucción se utiliza para establecer el nivel en una de las salidas. Estas tres instrucciones se consideran concurrentes, y el orden en el que se listan no hace ninguna diferencia. Por ejemplo, en la línea 8 cuando *A* es mayor que *B*, la salida *agtb* cambiará a ALTO al mismo tiempo que las otras dos salidas (*altb*, *aeqb*) cambien a BAJO.

```
1   SUBDESIGN fig9_66
2   (
3      a[3..0], b[3..0]     :INPUT;
4      agtb, altb, aeqb     :OUTPUT;
5   )
6   BEGIN
7      IF    a[] > b[] THEN
8               agtb = VCC;    altb = GND;   aeqb = GND;
9      ELSIF a[] < b[] THEN
10              agtb = GND;    altb = VCC;   aeqb = GND;
11     ELSE     agtb = GND ;   altb = GND ;  aeqb = VCC;
12     END IF;
13  END;
```

FIGURA 9-66 Comparador de magnitud en AHDL.

COMPARADOR EN VHDL

El código de VHDL de la figura 9-67 sigue el algoritmo que describimos mediante el uso de instrucciones IF/ELSE. Observe en la línea 2 que los valores de datos se declaran como enteros de cuatro bits. Recuerde que en VHDL las instrucciones IF/ELSE pueden utilizarse sólo dentro de un bloque PROCESS. En este caso queremos evaluar este bloque PROCESS cada vez que cualquiera de las entradas cambian de estado. En consecuencia, cada entrada se incluye en la lista de sensibilidad dentro de los paréntesis. Observe además que en las líneas 10, 11 y 12 pueden utilizarse varias instrucciones para especificar la operación del circuito cuando la cláusula IF es verdadera. Cada instrucción se utiliza para establecer el nivel en una de las salidas. Estas tres instrucciones se consideran concurrentes, y el orden en el que se listan no hace ninguna diferencia. Por ejemplo, en la línea 11 cuando *A* es mayor que *B*, la salida *agtb* cambiará a ALTO al mismo tiempo que las otras dos salidas (*altb*, *aeqb*) cambian a BAJO.

```
1   ENTITY  fig9_67 IS
2   PORT (    a, b                    : IN INTEGER RANGE 0 TO 15;
3             agtb, altb, aeqb        : OUT BIT);
4   END  fig9_67;
5
6   ARCHITECTURE  vhdl OF fig9_67  IS
7   BEGIN
8       PROCESS (a, b)
9       BEGIN
10            IF        a < b THEN     altb <= '1';   agtb <= '0';   aeqb <= '0';
11            ELSIF     a > b THEN     altb <= '0';   agtb <= '1';   aeqb <= '0';
12            ELSE                     altb <= '0';   agtb z= '0';   aeqb <= '1';
13            END IF;
14      END PROCESS;
16  END vhdl;
```

FIGURA 9-67 Comparador de magnitud en VHDL.

PREGUNTAS DE REPASO

1. ¿Qué tipo de objetos de datos deben declararse para las entradas de datos de un comparador?
2. ¿Cuál es la estructura de control clave que se utiliza para describir un comparador?
3. ¿Cuáles son los operadores clave que se utilizan?

9-20 CONVERTIDORES DE CÓDIGO EN HDL

En la sección 9-11 se demostraron algunos métodos en los que se utilizan circuitos sumadores de una manera interesante, pero nada intuitiva, de crear un convertidor de BCD a binario. En el capítulo 6 vimos los circuitos sumadores, y es evidente que el circuito de la figura 9-40 puede implementarse mediante el uso de HDL y las macrofunciones del 7483 o las descripciones de sumadores que sabemos cómo escribir. No obstante, ésta es una excelente oportunidad para señalar la enorme ventaja que puede ofrecer el HDL, ya que permite describir un circuito de una manera que sea la más apropiada. En el caso de la conversión de BCD a binario, el

método sensible de conversión es utilizar los conceptos que todos aprendimos en tercer grado acerca del sistema numérico decimal. Alguna vez nos enseñaron que el número 275 era en realidad:

$$\begin{array}{rrrrrr} & 2 & \times & 100 & = & 200 \\ + & 7 & \times & 10 & = & 70 \\ + & 5 & \times & 1 & = & 5 \\ \hline & & & & & 275 \end{array}$$

Ya estudiamos el sistema numérico BCD y podemos deducir que el 275 se representa en BCD como 0010 0111 0101. Es tan simple como que cada dígito se representa en binario. Si pudiéramos multiplicar estos dígitos binarios por el peso decimal (representado en binario) y sumarlos, obtendríamos una respuesta en binario que sería equivalente a la cantidad en BCD. Por ejemplo, probemos a utilizar la representación en BCD para el 275:

BCD	*Peso decimal (en binario)*	*Producto parcial (en binario)*
0010 ×	1100100 =	11001000
+ 0111 ×	1010 =	01000110
+ 0101 ×	1 =	0101
		$100010011 = 275_{10}$

La solución que presentamos aquí para nuestro convertidor de código de ocho bits (dos dígitos BCD) en HDL utiliza la siguiente estrategia:

> **Tome el dígito BCD más significativo (la posición de las décadas) y multiplíquelo por 10. Sume este producto al dígito BCD menos significativo (la posición de las unidades).**

La respuesta será un número binario que represente la cantidad en BCD. Es importante tener en cuenta que el compilador de HDL no necesariamente tratará de implementar un verdadero circuito multiplicador en su solución, sino que creará el circuito más eficiente que pueda realizar el trabajo, lo cual permite al diseñador describir su comportamiento de la manera más sensible.

AHDL

CONVERTIDOR DE CÓDIGO BCD A BINARIO EN AHDL

La clave para esta estrategia es poder multiplicar por 10. El AHDL no ofrece un operador de multiplicación, por lo que para poder utilizar esta estrategia en general, necesitamos realizar algunos trucos de matemáticas. Utilizaremos el desplazamiento de bits para realizar la multiplicación y después emplearemos la propiedad distributiva del álgebra para multiplicar por 10. De la misma forma en la que podemos desplazar un número decimal a la izquierda por un dígito (es decir, multiplicarlo por 10), también podemos desplazar un número binario una posición a la izquierda y multiplicarlo por 2. Si el desplazamiento es de dos posiciones, el número binario se multiplica por 4, y si el desplazamiento es de tres posiciones se multiplica por 8. La propiedad distributiva nos dice que:

$$\text{núm} \times 10 = \text{núm} \times (8 + 2) = (\text{núm} \times 8) + (\text{núm} \times 2)$$

Si podemos tomar el dígito de las décadas en BCD y lo desplazamos tres posiciones de bit a la izquierda (es decir, lo multiplicamos por 8), después tomamos el mismo número y lo desplazamos una posición a la izquierda (es decir, lo multiplicamos por 2) y después sumamos los productos, el resultado será el mismo que multiplicar el dígito BCD por 10. Después este valor se suma al dígito de las unidades en BCD para producir el equivalente binario de la entrada BCD de dos dígitos.

El siguiente reto es desplazar el dígito BCD a la izquierda mediante el uso de AHDL. Como AHDL nos permite crear conjuntos de variables, para desplazar los bits podemos agregar ceros en el extremo derecho del arreglo. Por ejemplo, si tenemos el número 5 en BCD (0101) y queremos desplazarlo tres posiciones, podemos concatenar el número 0101 con el número 000 en un conjunto, como se muestra a continuación:

(B“0101”, B“000”) = B“0101000”

El código de AHDL en la figura 9-68 comienza por declarar las entradas para los dígitos de las unidades y las décadas en BCD. La salida binaria debe poder representar el número 99_{10}, el cual requiere siete bits. También necesitamos una variable para retener el producto del dígito BCD multiplicado por 10. La línea 5 declara esta variable como un número de siete bits. La línea 8 realiza el desplazamiento del arreglo *decadas[ ]* tres veces y lo suma al arreglo *decenas[ ]* desplazado una posición a la izquierda. Observe que este último conjunto debe tener siete bits para poder sumarse al primer conjunto, por lo cual necesitamos concatenar B“00” en el extremo izquierdo. Por último, en la línea 10 se suma el resultado de la línea 8 al dígito de las unidades en BCD con extensiones de ceros a la izquierda (para formar siete bits), para formar la salida binaria.

```
1   SUBDESIGN fig9 68
2   (  unidades[3..0], decenas[3..0] :INPUT;
3      binario[6..0]                 :OUTPUT;  )
4
5   VARIABLE por10[6..0]       :NODE;    % variable para el digito de las decenas por 10
6
7   BEGIN
8      por10[] = (decenas[],B"000") + (B"00",decenas[],B"0");
9           % desplaza a la izquierda 3X (por 8) + desplaza a la izquierda 1X (por 2) %
10     binario[] = por10[] + (B"000",unidades[]);
11          % digito de las decadas por 10 + digito de las unidades %
12  END;
```

FIGURA 9-68 Convertidor de código BCD a binario en AHDL.

CONVERTIDOR DE CÓDIGO BCD A BINARIO EN VHDL

La solución de VHDL en la figura 9-59 es muy simple, debido a las poderosas operaciones matemáticas disponibles en este lenguaje. Las entradas y salidas deben declararse como enteros, ya que es nuestra intención realizar operaciones aritméticas con ellos. Observe que el intervalo se especifica con base en el mayor número BCD que sea válido, utilizando sólo dos dígitos. En la línea 9 el dígito de las décadas se multiplica por diez y en la línea 10 se suma el dígito de las unidades para formar el equivalente binario de la entrada BCD.

FIGURA 9-69
Convertidor de código BCD a binario en VHDL.

```
1   ENTITY fig9 69 IS
2   PORT (   unidades, decadas  :IN INTEGER RANGE 0 TO 9;
3            binary      :OUT INTEGER RANGE 0 TO 99);
4   END fig9 69;
5
6   ARCHITECTURE vhdl OF fig9 69 IS
7   SIGNAL por10      :INTEGER RANGE 0 TO 90;
8   BEGIN
9      por10 <= decadas * 10;
10     binario <= por10 + unidades;
11  END vhdl;
```

PREGUNTAS DE REPASO

1. ¿Cuál es el peso decimal del dígito más significativo para un número BCD (ocho bits) de dos dígitos?
2. ¿Cómo se logra la multiplicación por 10 en AHDL?
3. ¿Cómo se logra la multiplicación por 10 en VHDL?

RESUMEN

1. Un decodificador es un dispositivo cuya salida se activa sólo cuando hay una combinación (código) binaria única presente en sus entradas. Muchos decodificadores MSI tienen varias salidas, cada una de las cuales corresponde a sólo una de las muchas combinaciones de entrada posibles.
2. A menudo los sistemas digitales necesitan visualizar números decimales. Para ello se utilizan pantallas de 7 segmentos que se controlan mediante chips especiales que decodifican los números binarios y los traducen en patrones de segmentos que representan números decimales para las personas. Los elementos de segmento pueden ser diodos emisores de luz, cristales líquidos o electrodos fosforescentes rodeados de gas neón.
3. Los LCDs gráficos utilizan una matriz de elementos de imagen llamados píxeles, para crear una imagen en una pantalla grande. Cada píxel se controla mediante la activación de la fila y la columna que tienen ese píxel en común. El nivel de brillo de cada píxel se almacena como un número binario en la memoria de video. Un circuito digital bastante complejo debe explorar la memoria de video y todas las combinaciones de filas/columnas, controlando la cantidad de luz que puede pasar a través de cada píxel.
4. Un codificador es un dispositivo que genera un código binario único, en respuesta a la activación de cada entrada individual.
5. Para diagnosticar fallas en un sistema digital se requiere el proceso de *observación/análisis* para identificar las posibles causas, y un proceso de eliminación llamado *dividir y conquistar* para aislar e identificar la causa.
6. Los multiplexores actúan como interruptores controlados en forma digital, los cuales seleccionan y conectan una entrada lógica con la terminal de salida, en un momento dado. Al tomar turnos, muchas señales de datos distintas pueden compartir la misma ruta de datos mediante el uso de multiplexores. Los demultiplexores se utilizan en el otro extremo de la ruta de datos para separar las señales que comparten una ruta de datos y las distribuye hacia sus destinos respectivos.
7. Los comparadores de magnitud sirven como un indicador de la relación entre dos números binarios, con salidas que muestran >, < y =.
8. A menudo es necesario traducir entre varios métodos de representar cantidades con números binarios. Los convertidores de código son dispositivos que reciben una forma de representación binaria y la convierten en otra forma.
9. En los sistemas digitales, muchos dispositivos deben compartir a menudo la misma ruta de datos. A la cual se le conoce como *bus de datos*. Aún y cuando muchos dispositivos pueden estar "viajando" en el bus, sólo puede haber un "controlador" de bus en un momento dado. Por ende, los dispositivos deben tomar turnos para aplicar señales lógicas al bus de datos.
10. Para poder tomar turnos, los dispositivos deben tener *salidas triestado* que puedan deshabilitarse cuando otro dispositivo esté controlando el bus. En el estado deshabilitado, la salida del dispositivo se desconecta eléctricamente del bus al pasar a un estado que ofrece una ruta de alta impedancia tanto hacia tierra como hacia la fuente de energía positiva. Los dispositivos diseñados como interfaces para un bus tienen salidas que pueden estar en ALTO, BAJO o deshabilitadas (alta impedancia).

11. Los PLDs ofrecen una alternativa para el uso de circuitos MSI, para implementar sistemas digitales. Pueden utilizarse ecuaciones Booleanas para describir la operación de estos circuitos, pero los HDLs también ofrecen instrucciones en lenguaje de alto nivel.
12. Las macrofunciones de HDL están disponibles para muchas piezas MSI estándar que se describieron en este capítulo.
13. Puede escribirse código personalizado en HDL para describir cada una de las funciones lógicas comunes que se presentaron en este capítulo.
14. La prioridad y la precedencia pueden establecerse en AHDL mediante el uso de entradas "no importa" en las tablas de verdad y decisiones IF/ELSE. La prioridad y la precedencia pueden establecerse en VHDL mediante el uso de asignaciones de señal condicionales o mediante el uso de un bloque PROCESS que contenga decisiones IF/ELSE o CASE.
15. En AHDL pueden crearse salidas triestado. AHDL utiliza primitivas :TRI para controlar las salidas. VHDL asigna Z (alta impedancia) como un estado válido para las salidas STD_LOGIC.
16. La instrucción DEFAULTS en AHDL puede usarse para definir el nivel apropiado para las salidas que no se definen en forma explícita en el código.
17. La cláusula ELSE en la instrucción de asignación de señal condicional de VHDL puede usarse para definir el estado predeterminado de una salida.

TÉRMINOS IMPORTANTES

análisis/observación
ánodo común
bus de datos
bus flotante
cátodo común
codificación
codificador
codificador con prioridad
comparador de magnitud
controlador
controlador de bus
conversión de paralelo a serial
decodificador
decodificador de BCD a decimal
decodificador/controlador de BCD a 7 segmentos
demultiplexor (DEMUX)
dividir y conquistar
instrucción de asignación de señal condicional
LCD
líneas bidireccionales de datos
multiplexaje
multiplexor (MUX)
palabra
píxeles
plano posterior
valores predeterminados (DEFAULTS)

PROBLEMAS

SECCIÓN 9-1

B 9-1. Consulte la figura 9-3. Determine los niveles en la salida de cada decodificador para los siguientes conjuntos de condiciones de entrada.

(a)* Todas las entradas en BAJO.

(b)* Todas las entradas en bajo, excepto E_3 = ALTO.

(c) Todas las entradas en ALTO excepto $\overline{E}_1 = \overline{E}_2$ = BAJO.

(d) Todas las entradas en ALTO.

B 9-2.* ¿Cuál es el número de entradas y salidas de un decodificador que acepta 64 distintas combinaciones de entrada?

* Encontrará las respuestas a los problemas marcados con un asterisco al final del libro.

B 9-3. Para un 74ALS138, ¿qué condiciones de entrada producirán las siguientes salidas?

(a)* BAJO en $\overline{O}_6$

(b)* BAJO en $\overline{O}_3$

(c) BAJO en $\overline{O}_5$

(d) BAJO en $\overline{O}_0$ y $\overline{O}_7$, al mismo tiempo

D 9-4. Muestre cómo utilizar 74LS138s para formar un decodificador de 1 a 16.

9-5.* La figura 9-70 muestra cómo puede utilizarse un decodificador para generar señales de control. Suponga que se produjo un pulso de RESET en el tiempo t_0 y determine la forma de onda de CONTROL para 32 pulsos de reloj.

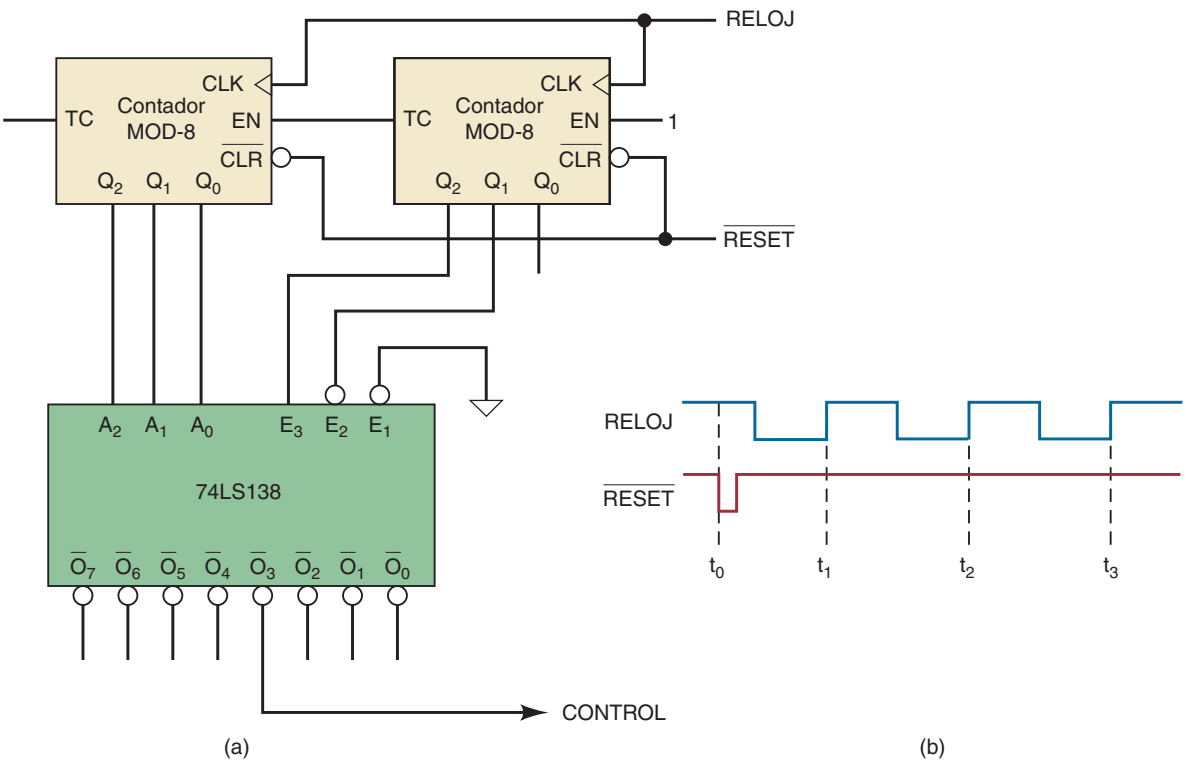


FIGURA 9-70 Problemas 9-5 y 9-6.

D 9-6. Modifique el circuito de la figura 9-70 para generar una forma de onda de CONTROL que cambie a BAJO de t_{20} a t_{24}. (*Sugerencia:* la modificación no requiere lógica adicional.)

9-7.* El decodificador 7442 de la figura 9-5 no tiene una entrada de HABILITACIÓN. No obstante, podemos operarlo como un decodificador de 1 a 8 si no utilizamos las salidas O_8 y O_9 y utilizando la entrada D como HABILITACIÓN. Esto se ilustra en la figura 9-71. Describa cómo funciona este arreglo como un decodificador de 1 a 8 habilitado, e indique cómo el nivel en D habilita o deshabilita las salidas.

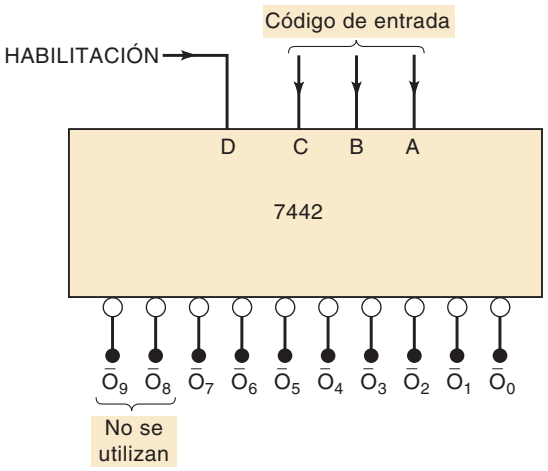


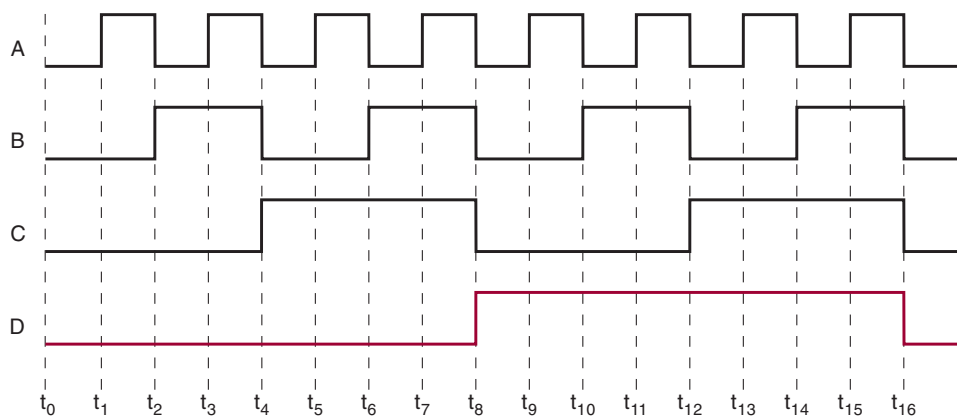
FIGURA 9-71 Problema 9-7.

9-8. Considere las formas de onda en la figura 9-72. Aplique estas señales al 74LS138 de la siguiente manera:

$$A \rightarrow A_0 \qquad B \rightarrow A_1 \qquad C \rightarrow A_2 \qquad D \rightarrow E_3$$

Suponga que $\overline{E}_1$ y $\overline{E}_2$ se conectan al nivel BAJO y dibuje las formas de onda para las salidas $\overline{O}_0$, $\overline{O}_3$, $\overline{O}_6$ y $\overline{O}_7$.

FIGURA 9-72
Problemas 9-8, 9-15 y 9-41.



D 9-9. Modifique el circuito de la figura 9-6 de manera que el relevador K_1 permanezca energizado de la PGT 3 a la 5, y que K_2 permanezca energizado del PGT 6 al 9. (*Sugerencia:* esta modificación no requiere circuitos adicionales.)

SECCIONES 9-2 Y 9-3

B, D 9-10.* Muestre cómo conectar decodificadores/controladores de BCD a 7 segmentos y pantallas de LEDs de 7 segmentos con el circuito contador de la figura 7-22. Suponga que cada segmento debe operar a una corriente aproximada de 10 mA, a 2.5 V.

B 9-11. (a) Consulte la figura 9-10 y dibuje las formas de onda del segmento y del plano posterior con respeto a tierra para CONTROL = 0. Después dibuje la forma de onda del voltaje del segmento con respecto al voltaje del plano posterior.

(b) Repita la parte (a) para CONTROL = 1.

A, D 9-12.* El decodificador/controlador de BCD a 7 segmentos de la figura 9-8 contiene la lógica para activar cada segmento, para las entradas BCD apropiadas. Diseñe la lógica para activar el segmento g.

SECCIÓN 9-4

B 9-13. **PREGUNTA DE PRÁCTICA**

Para cada inciso, indique si se refiere a un codificador o a un decodificador.

(a) Tiene más entradas que salidas.

(b) Es usado para convertir acciones clave a un código binario.

(c) Sólo una salida puede estar activa a la vez.

(d) Se puede usar para conectar una entrada BCD a un exhibidor de LEDs.

(e) Frecuentemente tiene salidas tipo amplificador para manejar alta corriente y voltaje.

9-14. Determine los niveles de salida para el codificador 74147 cuando $\overline{A}_8 = \overline{A}_4 = 0$ y todas las otras entradas están en ALTO.

9-15. Aplique las señales de la figura 9-72 a las entradas de un 74147 como se indica a continuación:

$$A \rightarrow \overline{A}_7 \qquad B \rightarrow \overline{A}_4 \qquad C \rightarrow \overline{A}_2 \qquad D \rightarrow \overline{A}_1$$

Dibuje las formas de onda para las salidas de los codificadores.

A, D 9-16. La figura 9-73 muestra el diagrama de bloques de un circuito lógico que se utiliza para controlar el número de copias que realiza una copiadora. Para seleccionar el número de copias deseadas, el operador de la máquina cierra uno de los interruptores selectores de S_1 a S_9. El codificador se encarga de codificar este número en BCD y luego se envía a un circuito comparador. Después el operador oprime un interruptor INICIO de contacto momentá-

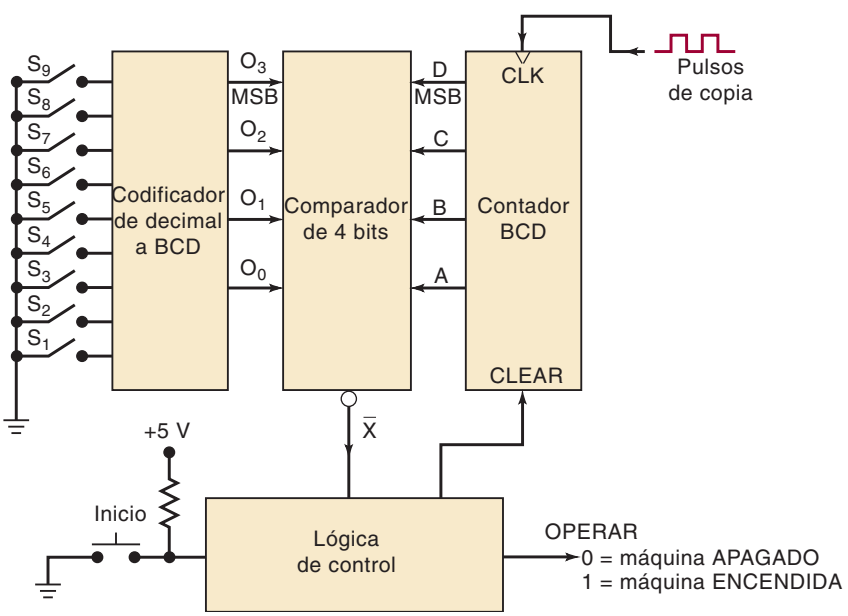


FIGURA 9-73 Problemas 9-16 y 9-52.

neo, el cual borra el contador e inicia una salida OPERAR EN NIVEL ALTO, la cual se envía a la máquina para indicarle que saque copias. A medida que la máquina va realizando cada copia, se genera un pulso para copiar y se alimenta al contador BCD. El comparador compara en forma continua las salidas del contador con las salidas del codificador de interruptor. Cuando los dos números BCD concuerdan, indicando que se ha completado el número deseado de copias, la salida *X* del comparador cambia a BAJO; esto hace que el nivel de OPERAR regrese a BAJO y se detiene la máquina para que ya no se realicen copias. Si se activa el interruptor INICIO se repetirá este proceso. Diseñe los circuitos lógicos completos para las secciones del comparador y de control de este sistema.

A, D 9-17.* El circuito de teclado de la figura 9-16 está diseñado para aceptar un número decimal de tres dígitos. ¿Qué ocurriría si se activaran *cuatro* teclas de dígitos (por ejemplo, 3095)? Diseñe la lógica necesaria que se debe agregar a este circuito de tal forma que, una vez que se hayan introducido tres dígitos, se ignoren todos los dígitos adicionales hasta que se oprima la tecla BORRAR. En otras palabras, si se introduce 3095 en el teclado, los registros de salida mostrarán 309 e ignorarán el 5 y cualquier dígito subsiguiente, hasta que el circuito se borre.

SECCIÓN 9-5

F 9-18.* Un técnico implementa el circuito de teclado de la figura 9-16 y evalúa su operación introduciendo una serie de números de tres dígitos. Mediante esta prueba descubre que algunas veces se introduce el dígito 0 en vez del dígito que oprimió. También observa que esto ocurre con todas las teclas de una forma más o menos aleatoria, aunque es peor para algunas teclas que para otras. Sustituye todos los CIs y la falla persiste. ¿Cuáles de las siguientes fallas del circuito podrían explicar sus observaciones? Explique cada una de las opciones.

(a) El técnico olvidó aterrizar las entradas que no se utilizan de la compuerta OR.

(b) Utilizó Q por error, en vez de $\overline{Q}$ del monoestable.

(c) El rebote del interruptor de las teclas de los dígitos dura más de 20 ms.

(d) Las salidas *Y* y *Z* están en corto.

F 9-19.* Repita el problema 9-18 con el siguiente síntoma: los registros y las pantallas permanecen en 0, sin importar cuántas veces se oprima una tecla.

F 9-20.* Al probar el circuito de la figura 9-16, un técnico descubre que al oprimir cualquier tecla con número impar se introduce el dígito correcto, pero si se oprime cualquier tecla con número par se introduce el dígito incorrecto de la siguiente manera: la tecla 0 hace que se introduzca un 1, la tecla 2 hace que se introduzca un 3, la tecla 4 hace que se introduzca un 5, y así en lo sucesivo. Considere cada una de las siguientes fallas como posibles causas del mal funcionamiento. Para cada una, explique por qué puede ser o no la verdadera causa.

(a) Hay una conexión abierta de la salida del inversor del LSB a las entradas *D* de los FFs.

(b) La entrada *D* del flip-flop Q_8 está en corto interno con V_{CC}.

(c) Un puente de soldadura está poniendo en corto a $\overline{O}_0$ con tierra.

F 9-21.* Un técnico evalúa el circuito de la figura 9-4 como se describe en el ejemplo 9-7 y obtiene los siguientes resultados: todas las salidas funcionan excepto de $\overline{O}_{16}$ a $\overline{O}_{19}$ y de $\overline{O}_{24}$ a $\overline{O}_{27}$, que están en ALTO de manera permanente. ¿Cuál es la falla más probable del circuito?

F 9-22. Un técnico evalúa el circuito de la figura 9-4 como se describe en el ejemplo 9-7 y descubre que se activa la salida correcta para cada código de entrada

TABLA 9-8

Código de entrada					
A_4	A_3	A_2	A_1	A_0	Salidas activadas
1	0	0	0	0	$\overline{O}_{16}$ y $\overline{O}_{24}$
1	0	0	0	1	$\overline{O}_{17}$ y $\overline{O}_{25}$
1	0	0	1	0	$\overline{O}_{18}$ y $\overline{O}_{26}$
1	0	0	1	1	$\overline{O}_{19}$ y $\overline{O}_{27}$
1	0	1	0	0	$\overline{O}_{20}$ y $\overline{O}_{28}$
1	0	1	0	1	$\overline{O}_{21}$ y $\overline{O}_{29}$
1	0	1	1	0	$\overline{O}_{22}$ y $\overline{O}_{30}$
1	0	1	1	1	$\overline{O}_{23}$ y $\overline{O}_{31}$

posible, excepto los que se listan en la tabla 9-8. Examine esta tabla y determine la causa probable del mal funcionamiento.

F 9-23.* Suponga que se utilizó por error una resistencia de 22 Ω para el segmento *g* en la figura 9-8. ¿Cómo afectaría esto a la pantalla? ¿Qué posibles problemas podrían ocurrir?

F 9-24. Repita el ejemplo 9-8 con la secuencia observada que se muestra a continuación:

CONTEO	0	1	2	3	4	5	6	7	8	9
Valor observado en exhibidor	0	1	2	3	8	9	c	ɔ	4	5

F 9-25.* Repita el ejemplo 9-8 con la secuencia observada que se muestra a continuación:

CONTEO	0	1	2	3	4	5	6	7	8	9
Valor observado en exhibidor	0	7	2	3	9	9	8	7	8	9

F 9-26.* Para evaluar el circuito de la figura 9-11, un técnico conecta un contador BCD a las entradas del 74HC4511 y aplica pulsos al contador a una frecuencia muy lenta. Después observa que el segmento *f* funciona en forma errática y no hay un patrón específico evidente. ¿Cuáles son algunas de las posibles causas del mal funcionamiento? (*Sugerencia:* recuerde que los CIs son CMOS.)

SECCIONES 9-6 Y 9-7

B 9-27. El diagrama de sincronización de la figura 9-74 se aplica a la figura 9-19. Dibuje la forma de onda de la salida *Z*.

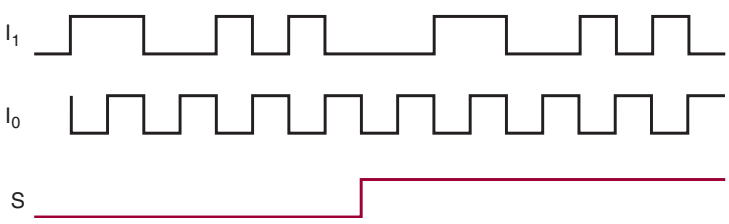
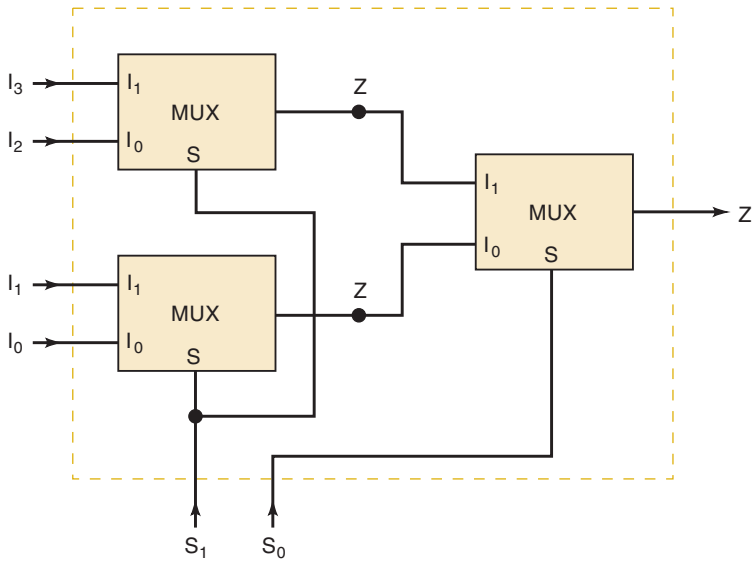


FIGURA 9-74 Problema 9-27.

9-28. La figura 7-68 muestra un registro de desplazamiento de ocho bits que podría utilizarse para retrasar una señal de 1 a 8 periodos de reloj. Muestre cómo alambrar un 74151 a este registro de desplazamiento, para poder seleccionar la salida Q deseada e indique el nivel lógico necesario en las entradas de selección para proporcionar un retraso de $6 \times T_{clk}$.

9-29.* El circuito de la figura 9-75 utiliza tres multiplexores de dos entradas (figura 9-19). Determine la función que realiza este circuito.

FIGURA 9-75 Problema 9-29.



D 9-30. Utilice la idea del problema 9.29 para conectar varios multiplexores 74151 1-de-8, para formar un multiplexor 1-de-64.

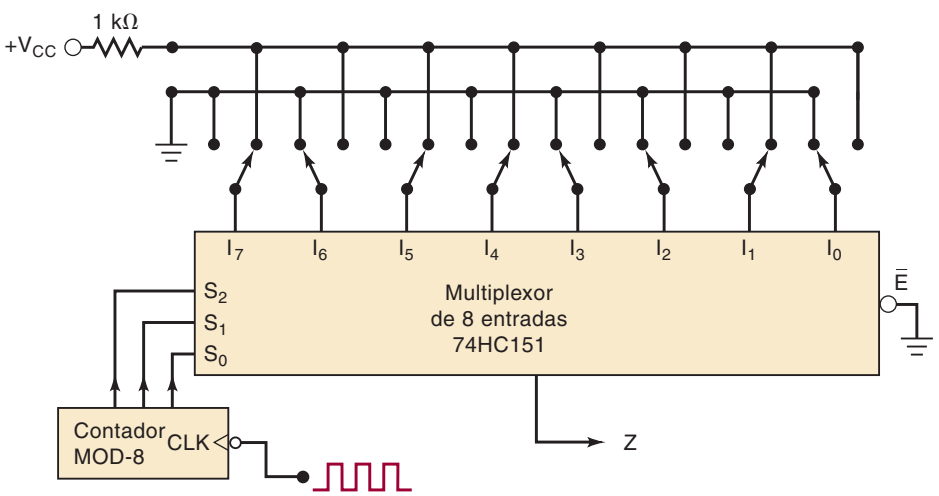
A, D 9-31.* Muestre cómo pueden conectarse dos chips 74157 y un 74151 para formar un multiplexor 1-de-16 sin requerir ninguna otra lógica. Etiquete las entradas de I_0 a I_{15} para mostrar cómo corresponden al código de selección.

D 9-32. (a) Expanda el circuito de la figura 9-24 para mostrar el contenido de dos contadores BCD de tres etapas.

(b)* Cuente el número de conexiones en este circuito y compárelo con el número requerido si se utilizaran un decodificador/controlador y una pantalla por separado para cada etapa de cada contador.

9-33.* La figura 9-76 muestra cómo puede utilizarse un multiplexor para generar formas de onda lógicas con cualquier patrón deseable. El patrón se pro-

FIGURA 9-76
Problemas 9-33 y 9-34.



grama mediante el uso de ocho interruptores SPDT y la forma de onda se produce en forma repetitiva al aplicar pulsos al contador MOD-8. Dibuje la forma de onda en Z para las posiciones dadas de los interruptores.

9-34. Cambie el contador MOD-8 en la figura 9-76 por un contador MOD-16, y conecte el MSB a la entrada $\overline{E}$ del multiplexor. Dibuje la forma de onda de Z.

D 9-35.* Muestre cómo puede usarse un 74151 para generar la función lógica $Z = AB + BC + AC$.

D 9-36. Muestre cómo puede usarse un multiplexor de 16 entradas tal como el 74150 para generar la función $Z = \overline{A}\,\overline{B}\,\overline{C}\,D + BCD + A\,\overline{B}\,\overline{D} + AB\overline{C}D$.

N 9-37.* El circuito de la figura 9-77 muestra cómo puede utilizarse un MUX de ocho entradas para generar una función lógica de cuatro variables, aún y cuando el MUX sólo tiene tres entradas de SELECCIÓN. Tres de las variables lógicas A, B y C se conectan a las entradas de SELECCIÓN. La cuarta variable D y su inverso $\overline{D}$ se conectan a las entradas de datos seleccionadas del MUX, según lo requiera la función lógica deseada. Las otras entradas de datos del MUX se conectan a un nivel BAJO o ALTO, según lo requiera la función.

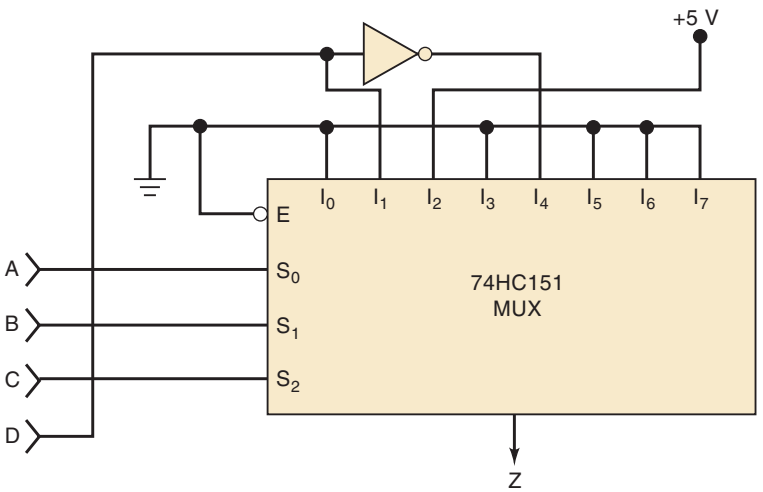


FIGURA 9-77 Problemas 9-37 y 9-38.

(a) Establezca una tabla de verdad que muestre la salida Z para las 16 combinaciones posibles de variables de entrada.

(b) Escriba la expresión de suma de productos para Z y simplifíquela para verificar que:

$$Z = \overline{C}B\overline{A} + D\overline{C}\,\overline{B}A + \overline{D}C\overline{B}\,\overline{A}$$

9-38. El método de hardware que se utiliza en la figura 9-77 puede usarse para generar cualquier función lógica de cuatro variables. Por ejemplo, la función $Z = \overline{D}\,\overline{B}\,\overline{C}\,A + \overline{C}BA + DC\overline{B}A + CB\overline{A}$ se implementa mediante los siguientes pasos:

1. Establezca una tabla de verdad en dos mitades, lado a lado como se muestra en la tabla 9-9. Observe que la mitad izquierda muestra todas las combinaciones de CBA cuando $D = 0$, y la mitad derecha muestra todas las combinaciones de CBA cuando $D = 1$.
2. Escriba el valor de Z para cada combinación de cuatro bits cuando $D = 0$ y también cuando $D = 1$.
3. Cree una columna del lado derecho como se muestra, en la cual se describa lo que debe conectarse a cada una de las ocho entradas I_n del MUX.

TABLA 9-9

	$D = 0$		$D = 1$	
DCBA	**Z**	**DCBA**	**Z**	**I_n**
0000	0	1000	0	$I_0 = 0$
0001	1	1001	0	$I_1 = \overline{D}$
0010	0	1010	0	$I_2 = 0$
0011	1	1011	1	$I_3 = 1$
0100	0	1100	0	$I_4 = 0$
0101	0	1101	1	$I_5 = D$
0110	1	1110	1	$I_6 = 1$
0111	0	1111	0	$I_7 = 0$

4. Para cada línea de esta tabla, compare el valor para Z cuando $D = 0$ con el valor para Z cuando $D = 1$. Introduzca la información apropiada para I_n, como se muestra a continuación:

   Cuando $Z = 0$ sin importar si $D = 0$ o 1, entonces $I_n = 0$ (GND).

   Cuando $Z = 1$ sin importar si $D = 0$ o 1, entonces $I_n = 1$ (VCC).

   Cuando $Z = 0$ cuando $D = 0$ y $Z = 1$ cuando $D = 1$, entonces $I_n = D$.

   Cuando $Z = 1$ cuando $D = 0$ y Z $= 0$ cuando $D = 1$, entonces $I_n = \overline{D}$.

(a) Utilice este método para verificar el diseño de la figura 9-77.

(b) Use este método para implementar una función que produzca un nivel ALTO sólo cuando las cuatro variables de entrada se encuentren en el mismo nivel, o cuando las variables B y C se encuentren en niveles distintos.

SECCIÓN 9-8

B 9-39.* **PREGUNTA DE PRÁCTICA**

Para cada inciso, indique si hace referencia a un decodificador, a un codificador, a un MUX o a un DEMUX.

(a) Tiene más entradas que salidas.

(b) Utiliza entradas de SELECCIÓN.

(c) Puede usarse en la conversión de paralelo a serial.

(d) Produce un código binario en su salida.

(e) Sólo una de sus salidas puede estar activa en un momento dado.

(f) Puede usarse para enrutar una señal de entrada hacia una de varias salidas posibles.

(g) Puede usarse para generar funciones lógicas arbitrarias.

9-40. Muestre cómo puede usarse el decodificador 7442 como un demultiplexor de 1-a-8. (*Sugerencia:* vea el problema 9-7.)

9-41.* Aplique las formas de onda de la figura 9-72 a las entradas del DEMUX 74LS138 de la figura 9-30(a) de la siguiente manera:

$$D \rightarrow A_2 \qquad C \rightarrow A_1 \qquad B \rightarrow A_0 \qquad A \rightarrow \overline{E}_1$$

Dibuje las formas de onda en las salidas del DEMUX.

9-42. Considere el sistema de la figura 9-31. Suponga que la frecuencia del reloj es de 10 pps. Describa cuáles serán las indicaciones del panel de monitoreo para cada uno de los siguientes casos.

(a) Todas las puertas cerradas.

(b) Todas las puertas abiertas.

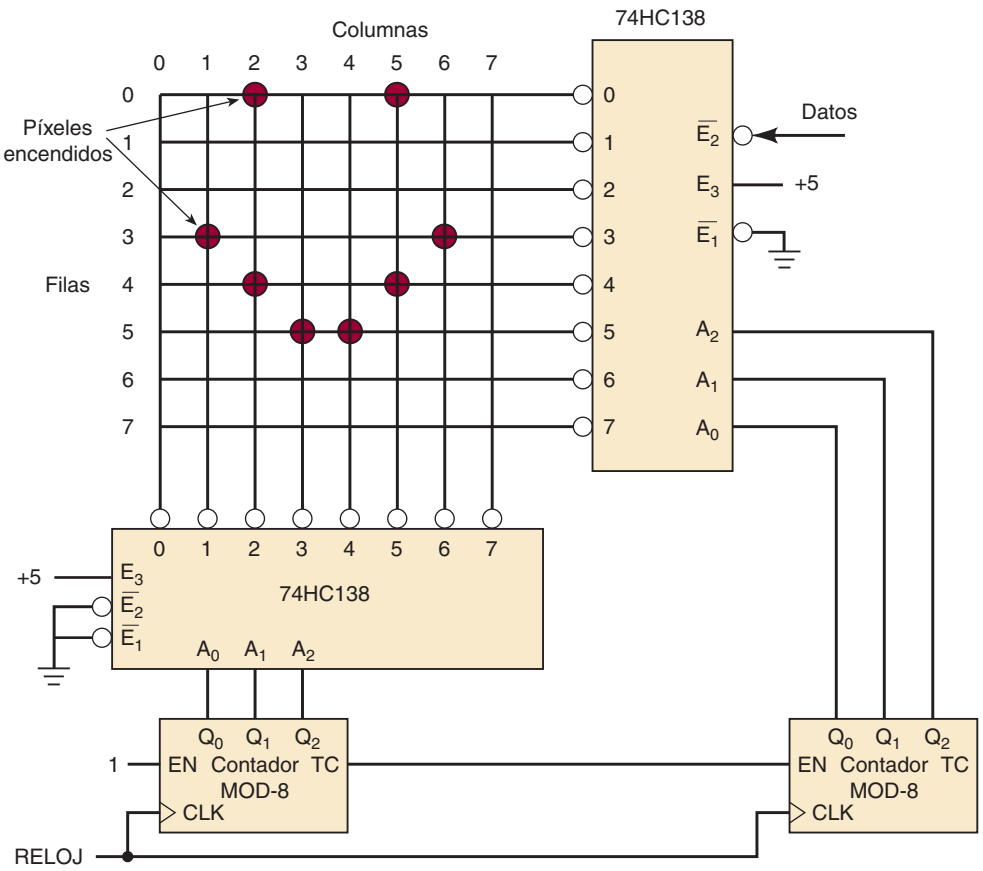
(c) Las puertas 2 y 6 abiertas.

C, D 9-43.* Modifique el sistema de la figura 9-31 para que maneje 16 puertas. Utilice un MUX 74150 de 16 entradas y dos DEMUX 74LS138. ¿Cuántas líneas van hacia el panel de monitoreo remoto?

9-44. Dibuje las formas de onda en transmite_datos y las salidas O_0,O_1,O_2 y O_3 en la figura 9-33 para los siguientes datos que se cargan en los registros de transmisión de la figura 9-32: $[A] = 0011$, $[B] = 0110$, $[C] = 1001$, $[D] = 0111$.

9-45. La figura 9-78 muestra una matriz de pantalla LCD de gráficos de 8×8, la cual se controla mediante un 74HC138 configurado como un decodificador y un 74HC138 configurado como un demultiplexor. Dibuje 48 ciclos del reloj y la entrada de datos necesaria para activar los píxeles que se muestran en la pantalla.

FIGURA 9-78
Problema 9-45.



SECCIÓN 9-9

F 9-46. Considere el secuenciador de control de la figura 9-26. Describa cómo afectará cada una de las siguientes fallas a la operación.

(a)* La entrada I_3 del MUX se pone en corto a tierra.

(b) Las conexiones de los sensores 3 y 4 con el MUX se invierten.

TABLA 9-10

		Conteo actual	Conteo en pantalla
Caso 1	Contador 1	33	33
	Contador 2	47	47
Caso 2	Contador 1	82	02
	Contador 2	64	64
Caso 3	Contador 1	63	63
	Contador 2	95	15

F 9-47.* Considere el circuito de la figura 9-24. Una prueba de este circuito produce los resultados que se muestran en la tabla 9-10. ¿Cuáles son las posibles causas del mal funcionamiento?

F 9-48.* Una prueba del sistema de monitoreo de seguridad de la figura 9-31 produce los resultados que se registran en la tabla 9-11. ¿Cuáles son las posibles fallas que podrían ocasionar esta operación?

TABLA 9-11

Condición	LEDs
Todas las puertas cerradas	All LEDs off
Puerta 0 abierta	LED 0 destellando
Puerta 1 abierta	LED 2 destellando
Puerta 2 abierta	LED 1 destellando
Puerta 3 abierta	LED 3 destellando
Puerta 4 abierta	LED 4 destellando
Puerta 5 abierta	LED 6 destellando
Puerta 6 abierta	LED 5 destellando
Puerta 7 abierta	LED 7 destellando

F 9-49.* Una prueba del sistema de monitoreo de seguridad de la figura 9-31 produce los resultados que se registran en la tabla 9-12. ¿Cuáles son las posibles fallas que podrían provocar esta operación? ¿Cómo puede verificarse o eliminarse esto como una falla?

TABLA 9-12

Condición	LEDs
Todas las puertas cerradas	Todos los LEDs apagados
Puerta 0 abierta	LED 0 destellando
Puerta 1 abierta	LED 1 destellando
Puerta 2 abierta	LED 2 destellando
Puerta 3 abierta	LED 3 destellando
Puerta 4 abierta	LED 4 destellando
Puerta 5 abierta	LED 5 destellando
Puerta 6 abierta	Ningún LED destellando
Puerta 7 abierta	Ningún LED destellando
Puertas 6 y 7 abiertas	LEDs 6 y 7 destellando

F 9-50.* El sistema de transmisión de datos en forma síncrona de las figuras 9-32 y 9-33 está fallando. Se utiliza un osciloscopio para analizar las salidas del

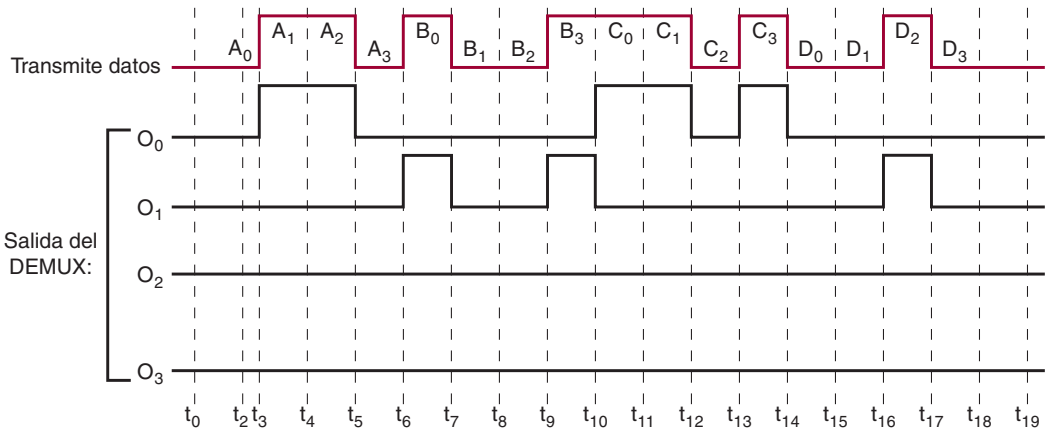


FIGURA 9-79 Problema 9-50.

MUX y del DEMUX durante el ciclo de transmisión, y los resultados se muestran en la figura 9-79. ¿Cuáles son las posibles causas del mal funcionamiento?

F 9-51. El sistema de transmisión de datos en forma síncrona de las figuras 9-32 y 9-33 no funciona en forma apropiada, por lo cual se utilizó el diagrama de árbol para diagnóstico de fallas de la figura 9-35 para aislar el problema en la sección de sincronización y control del receptor. Dibuje un diagrama de árbol para diagnóstico de fallas, para aislar el problema aún más en uno de los cuatro bloques de esa sección (*FF*1, contador de Bits, contador de Palabras o *FF*2). Suponga que todos los alambres se conectan como se muestra, sin errores en el alambrado.

SECCIÓN 9-10

A, D 9-52. Rediseñe el circuito del problema 9-16; para ello utilice un comparador de magnitud 74HC85. Agregue una característica de "desbordamiento de copias" que active una salida de ALARMA si la salida OPERAR no detiene la máquina cuando se termine el número de copias solicitadas.

D 9-53.* Muestre cómo conectar chips 74HC85 para comparar dos números de 10 bits.

SECCIÓN 9-11

9-54. Suponga una entrada BCD de 69 para el convertidor de código de la figura 9-40. Determine los niveles en cada salida Σ y en la salida binaria final.

F 9-55.* Un técnico evalúa el convertidor de código de la figura 9-40 y observa los siguientes resultados:

Entrada BCD	*Salida binaria*
52	0110011
95	1100000
27	0011011

¿Cuál es la probable falla del circuito?

SECCIONES 9-12 A 9-14

B 9-56. **PREGUNTA DE PRÁCTICA**

Verdadero o falso:

(a) Un dispositivo conectado a un bus de datos debe tener salidas triestado.

(b) La contención de bus ocurre cuando más de un dispositivo recibe datos del bus.

(c) Pueden transferirse unidades de carga mayores a través de un bus de datos de ocho líneas que a través de un bus de datos de cuatro líneas.

(d) Un CI controlador de bus tiene, por lo general, una impedancia de salida alta.

(e) Los registros y los búferes bidireccionales tienen líneas de E/S comunes.

9-57.* Para el arreglo del bus de la figura 9-44, describa los requerimientos de señal de entrada para transferir en forma simultánea el contenido del registro *C* hacia los otros dos registros.

9-58. Suponga que en un principio los registros de la figura 9-44 son $[A] = 1011$, $[B] = 1000$ y $[C] = 0111$. Las señales en la figura 9-80 se aplican a las entradas del registro.

(a) Determine el contenido de cada registro en los tiempos t_1, t_2, t_3 y t_4.

(b) Describa lo que ocurriría si $\overline{IE}_A$ estuviera en BAJO al producirse el tercer pulso de reloj.

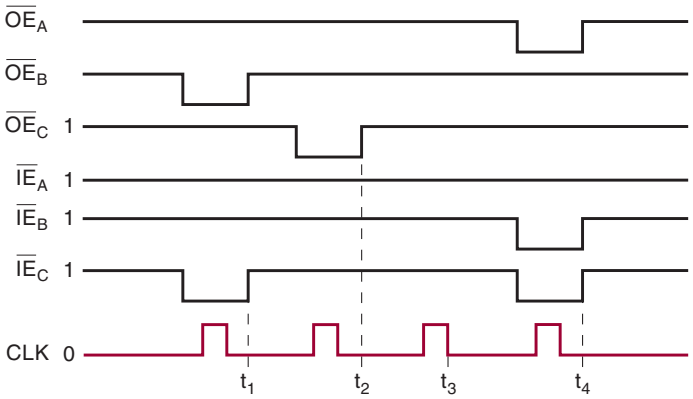


FIGURA 9-80 Problemas 9-58 y 9-59.

9-59. Suponga las mismas condiciones iniciales del problema 9-58 y haga un bosquejo de la señal en DB_3 para las formas de onda de la figura 9-80.

9-60. La figura 9-81 muestra otros dos dispositivos que se van a agregar al bus de datos de la figura 9-44. Uno de ellos es un conjunto de interruptores con búfer que puede usarse para introducir datos en forma manual hacia uno de los registros del bus. El otro dispositivo es un registro de salida que se utiliza para fijar cualquier dato que se encuentre en el bus durante una operación de transferencia de datos y visualizarlos en un conjunto de LEDs.

(a) Suponga que todos los registros contienen 0000. Describa la secuencia de operaciones necesarias para cargar los registros con los siguientes datos de los interruptores: $[A] = 1011$, $[B] = 0001$, $[C] = 1110$.

(b) ¿Cuál será el estado de los LEDs al terminar esta secuencia?

A 9-61. Ahora que se agregó el circuito de la figura 9-81 a la figura 9-44, hay un total de cinco dispositivos conectados al bus de datos. El circuito de la figura 9-82(a) se utilizará ahora para generar las señales de habilitación necesarias para realizar las distintas transferencias de datos a través del bus de datos. Este circuito utiliza un chip 74HC139 que contiene dos decodificadores independientes 1-de-4 idénticos, con una señal de habilitación activa en BAJO. El decodificador superior se utiliza para seleccionar el dispositivo que colocará datos en el bus de datos (selección de salida) y el decodificador inferior se utiliza para seleccionar el dispositivo que va a recibir los datos del bus de datos (selección de entrada). Suponga que las salidas del decodificador se conectan a las correspondientes entradas de habilitación de los dispositivos conectados al bus de datos. Suponga también que al principio

FIGURA 9-81
Problemas 9-60, 9-61 y 9-62.

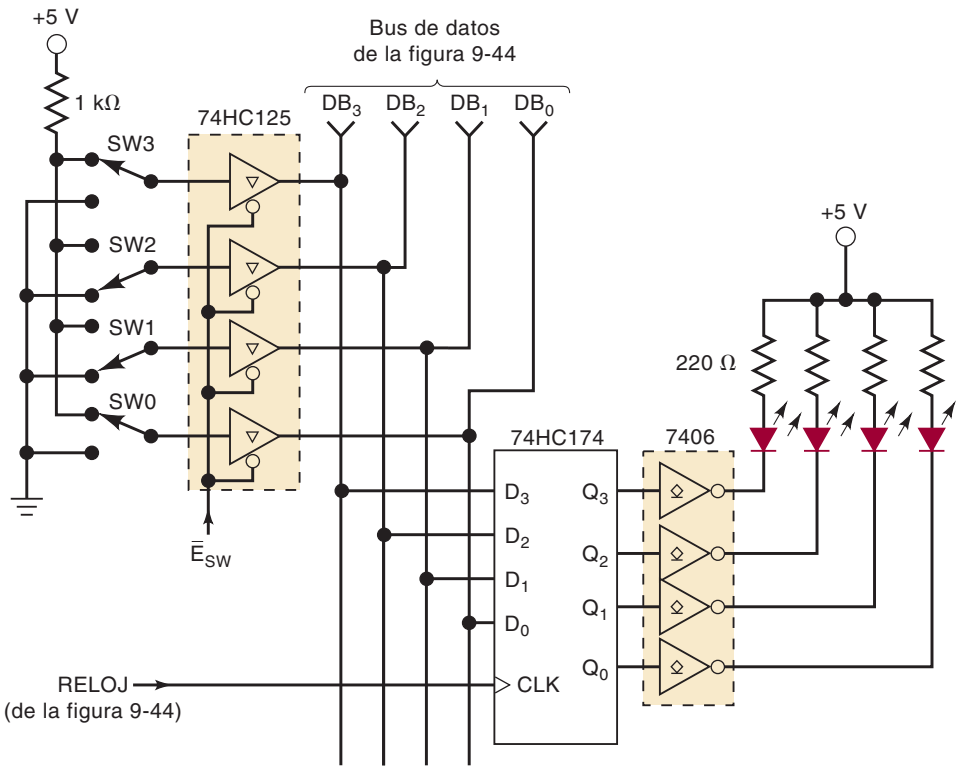
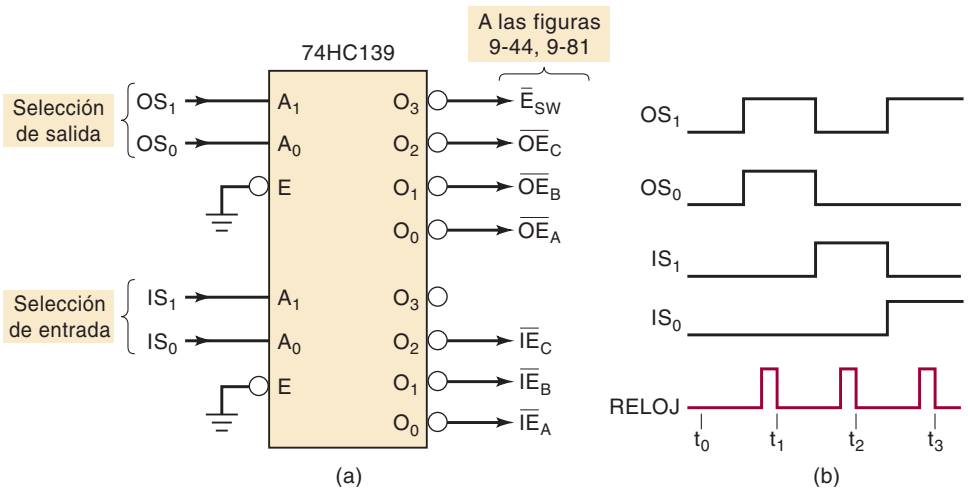


FIGURA 9-82
Problema 9-61.



todos los registros contienen 0000 en el tiempo t_0 y que los interruptores están en las posiciones que se muestran en la figura 9-81.

(a)* Determine el contenido de cada registro en los tiempos t_1, t_2 y t_3 en respuesta a las formas de onda de la figura 9-82(b).

(b) ¿Puede ocurrir una colisión de bus con este circuito? Explique.

9-62. Muestre cómo puede utilizarse un 74HC541 (figura 9-47) en el circuito de la figura 9-81.

APLICACIONES DE MICROCOMPUTADORA

A, N 9-63.* La figura 9-83 muestra el circuito básico que actúa como interfaz entre un microprocesador (MPU) y un módulo de memoria. El módulo de memoria deberá contener uno o más CIs de memoria (capítulo 12) que pueden reci-

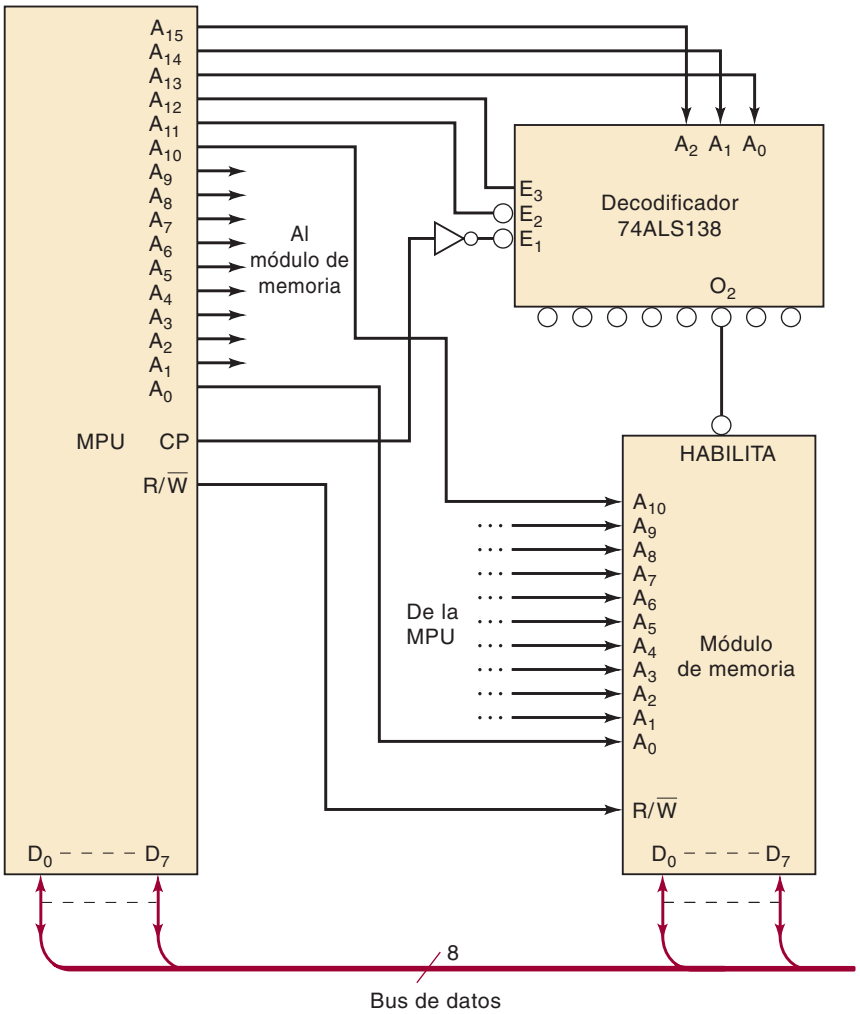


FIGURA 9-83 Circuito de interfaz básica de microprocesador a memoria para el problema 9-63.

bir datos de la MPU (una operación de ESCRITURA) o enviar datos hacia la MPU (operación de LECTURA). Los datos se transfieren a través del bus de datos de ocho líneas. Las líneas de datos de la MPU y las líneas de datos de E/S de la memoria se conectan a este bus común. Por ahora nos preocuparemos por la forma en que la MPU controla la selección del módulo de memoria para una operación de LECTURA o de ESCRITURA. Los pasos para hacer esto son:

1. La MPU coloca la dirección de memoria en sus líneas de salida de dirección $A_{15} - A_0$.
2. La MPU genera la señal $R/\overline{W}$ para informar al módulo de memoria qué operación se va a realizar: $R/\overline{W} = 1$ para LECTURA, $R/W = 0$ para ESCRITURA.
3. Los cinco bits superiores de las líneas de dirección de la MPU se decodifican mediante el 74ALS138, el cual controla la entrada de HABILITACIÓN del módulo de memoria. Esta entrada de HABILITACIÓN debe estar activa para que el módulo de memoria pueda realizar una operación de LECTURA o de ESCRITURA.
4. Los otros 11 bits de dirección se conectan al módulo de memoria, el cual los utiliza para seleccionar la posición de memoria *interna* específica a la que la MPU va a tener acceso, siempre y cuando HABILITACIÓN esté activa.

Para poder leer de, o escribir hacia el módulo de memoria, el MPU debe colocar la dirección correcta en las líneas de dirección para habilitar la memoria, y después se aplica un pulso a *CP* para que cambie al estado ALTO.

(a) Determine cuál de las siguientes direcciones hexadecimales activará el módulo de memoria: 607F, 57FA, 5F00.

(b) Determine el intervalo de direcciones hexadecimales que activará la memoria. (*Sugerencia:* las entradas A_0 a A_{10} para la memoria pueden tener cualquier combinación.)

(c) Suponga que se agrega un segundo módulo de memoria idéntico al circuito con sus líneas de dirección, $R/\overline{W}$ y de E/S de datos conectadas justo igual que el primer módulo, *excepto* que su entrada de HABILITACIÓN está conectada a la salida $\overline{O}_4$ del decodificador. ¿Qué intervalo de direcciones hexadecimales activará este segundo módulo?

(d) Es posible que la MPU lea de, o escriba hacia ambos módulos a la vez? Explique.

PROBLEMA DE DISEÑO

A, D 9-64. El circuito de introducción desde el teclado de la figura 9-16 se va a utilizar como parte de un candado digital electrónico que opera de la siguiente manera: cuando se activa, una salida DESBLOQUEO cambia a ALTO. Este nivel ALTO se utiliza para energizar un solenoide que retracta un perno y permite abrir una puerta. Para activar DESBLOQUEO, el operador debe oprimir la tecla BORRAR y después debe introducir la secuencia correcta de tres teclas.

(a) Muestre cómo pueden agregarse comparadores 74HC85 y demás lógica necesaria al circuito de introducción desde el teclado para producir la operación del candado digital que se describe antes para la secuencia de teclas BORRAR-3-5-8.

(b) Modifique el circuito para activar una salida de ALARMA si el operador introduce algo distinto a la secuencia correcta de tres teclas.

SECCIONES 9-15 A 9-20

H, D, N 9-65.* Escriba el código de HDL para un decodificador de BCD a decimal (el equivalente a un 7442).

H, D, N 9-66. Escriba el código de HDL para un decodificador/controlador HEX para una pantalla de 7 segmentos. Los primeros 10 caracteres deben aparecer como se muestra en la figura 9-7. Los últimos seis caracteres deben aparecer como se muestra en la figura 9-84.

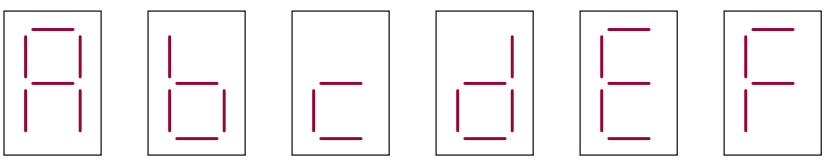


FIGURA 9-84 Caracteres HEX para el problema 9-66.

B, H, N 9-67. Escriba una descripción CODIFICADOR de baja prioridad que siempre codifique el número más bajo si dos entradas se activan en forma simultánea.

H, N 9-68. Vuelva a escribir el código del comparador de cuatro bits de las figuras 9-66 o 9-67 para crear un comparador de ocho bits, sin utilizar macrofunciones.

H, N 9-69. Use HDL para describir un convertidor de código de número binario de cuatro bits a código BCD de dos dígitos.

H, N 9-70. Utilice HDL para describir un convertidor de código BCD de tres dígitos a número binario de ocho bits. (La máxima entrada BCD es 255.)

RESPUESTAS A LAS PREGUNTAS DE REPASO DE SECCIÓN

SECCIÓN 9-1

1. No 2. La entrada de habilitación controla si la lógica del decodificador responde o no al código binario de entrada. 3. El 7445 tiene salidas de colector abierto que pueden manejar hasta 30 V y 80 mA. 4. 24 terminales: 2 de habilitación, 4 entradas, 16 salidas, V_{CC} y tierra.

SECCIÓN 9-2

1. *a, b, c, f, g* 2. Verdadero

SECCIÓN 9-3

1. LEDs: (a), (e), (f). LCDs: (b), (c), (d), (e) 2. (a) BCD de cuatro bits, (b) ASCII de siete u ocho bits, (c) valor binario para la intensidad de los píxeles.

SECCIÓN 9-4

1. Un codificador produce un código de salida que corresponde a la entrada que se activó. Un decodificador activa una salida que corresponde a un código de entrada que se aplicó. 2. En un codificador con prioridad, el código de salida corresponde a la entrada de *mayor* numeración que se activa. 3. BCD normal = 0110
4. (a) produce un PGT cuando se oprime una clave. (b) convierte la acción de oprimir la tecla en su código BCD, (c) genera un pulso libre de rebotes para disparar el contador de anillo, (d) forma un contador de anillo que aplica pulsos de reloj en forma secuencial a los registros de salida, (e) almacena los códigos BCD generados por las teclas oprimidas.
5. E_1 y E_0 se utilizan para la conexión en cascada y GS indica una entrada activa.

SECCIÓN 9-6

1. El número binario en las entradas de selección determina cuál entrada de datos pasará hacia la salida. 2. Treinta y dos entradas de datos y cinco entradas de selección.

SECCIÓN 9-7

1. Conversión de paralelo a serial, enrutamiento de datos, generación de funciones lógicas, secuenciamiento de operaciones. 2. Falso; se aplican a las entradas de selección. 3. Contador

SECCIÓN 9-8

1. Un MUX selecciona una de muchas señales de entrada para pasarla hacia su salida; un DEMUX selecciona una de muchas salidas para recibir la señal de entrada. 2. Verdadero, siempre y cuando el decodificador tenga una entrada de HABILITACIÓN 3. Los LEDs se encenderán y se apagarán en secuencia.

SECCIÓN 9-10

1. Para proveer el medio para expandir las operaciones de comparación a números con más de cuatro bits. 2. $O_{A=B} = 1$; las demás salidas son 0.

SECCIÓN 9-11

1. Un convertidor de código recibe los datos de entrada representados en un tipo de código binario y los convierte en otro tipo de código binario. 2. Tres dígitos pueden representar valores decimales de hasta 999. Para representar el 999 en binario simple se requieren 10 bits.

SECCIÓN 9-12

1. Un conjunto de líneas de conexión a las que pueden conectarse las entradas y salidas de muchos dispositivos distintos. 2. La colisión de bus se produce cuando las salidas de varios dispositivos conectados al bus se habilitan al mismo tiempo. Para evitarla se

controlan las entradas de habilitación de dispositivos, de manera que esto no pueda ocurrir. 3. Una condición en la que todos los dispositivos conectados a un bus se encuentran en el estado Hi-Z.

SECCIÓN 9-13

1. 1011 2. Verdadero 3. 0000

SECCIÓN 9-14

1. Colisión de bus 2. Flotante, Hi-Z 3. Proporciona salidas triestado de baja impedancia. 4. Reduce el número de terminales de CI y el número de conexiones al bus de datos. 5. Vea la figura 9-51.

SECCIÓN 9-15

1. Son entradas de habilitación. Todas deben estar activas para que el decodificador funcione. 2. CASE y TABLE 3. La instrucción de asignación de señal selecta y CASE.

SECCIÓN 9-16

1. La terminal de combinación de entrada/salida BI/RBO. 2. Ánodo común. Las salidas se conectan a los cátodos y cambian a BAJO para alumbrar los segmentos. 3. La estructura de control IF/ELSE se evalúa en secuencia y otorga precedencia en el orden en el que se listan las decisiones.

SECCIÓN 9-17

1. La celda de "no importa" en una tabla de verdad y la estructura de control IF/ELSE. 2. La estructura de control IF/ELSE y la instrucción de asignación de señal condicional. 3. Mediante el uso de la primitiva :TRI y la asignación de un valor a OE. 4. Mediante el uso del tipo de datos IEEE STD_LOGIC que tiene un valor posible de *Z*.

SECCIÓN 9-18

1. Entradas: *ch0, ch1, ch2, ch3*; salida: *dsal*; entradas de control (Selección): *s.*
2. Entrada *dent*; salidas: *ch0, ch1, ch2, ch3*; entradas de control (Selección): *s.*
3. DEFAULTS
4. ELSE

SECCIÓN 9-19

1. Objetos de datos numéricos (por ejemplo, INTEGER en VHDL). 2. IF/ELSE
3. Operadores relacionales (<, >).

SECCIÓN 9-20

1. 10 2. Mediante la multiplicación de 8 + 2. Si se desplaza el dígito BCD tres posiciones a la izquierda se multiplica por 8, y si se desplaza el mismo dígito BCD una posición a la izquierda se multiplica por 2. Al sumar estos resultados se produce el dígito BCD multiplicado por 10. 3. En VHDL simplemente se utiliza el operador * para multiplicar.

CAPÍTULO 10

PROYECTOS DE SISTEMAS DIGITALES MEDIANTE EL USO DE HDL

CONTENIDO

10-1 Administración de pequeños proyectos

10-2 Proyecto de controlador de motor de pasos

10-3 Proyecto de codificador de teclado numérico

10-4 Proyecto de reloj digital

10-5 Proyecto de contador de frecuencia

OBJETIVOS

Al terminar este capítulo, usted podrá:

- Analizar la operación de los sistemas compuestos por varios de los componentes que cubrimos en capítulos anteriores de este libro.
- Describir todo un proyecto con un archivo de HDL.
- Describir el proceso de la administración jerárquica de proyectos.
- Comprender cómo descomponer un proyecto en piezas manejables.
- Utilizar las herramientas de software MAX+PLUS II o Quartus II para implementar un proyecto modular jerárquico.
- Planear formas de evaluar la operación de los circuitos que usted construya.

INTRODUCCIÓN

Durante los primeros nueve capítulos de este libro le explicamos los bloques de construcción fundamentales de los sistemas digitales. Después de haber extraído cada bloque y analizarlo, no queremos hacerlos a un lado y olvidarlos; es tiempo de construir algo con los bloques. Algunos de los ejemplos que hemos utilizado para demostrar la operación de circuitos individuales son en realidad sistemas digitales por sí solos, y hemos estudiado la manera en que funcionan. En este capítulo queremos enfocarnos más en el proceso de construcción.

Encuestas a graduados nos indican que la mayoría de los profesionales en los campos de la tecnología e ingeniería eléctrica y computacional tienen la responsabilidad de administrar proyectos. La experiencia con los estudiantes también nos ha demostrado que la administración eficiente de un proyecto no es una habilidad obvia o instintiva para todo mundo, lo cual explica por qué muchos de nosotros terminamos aprendiendo con el método de prueba y error (la escuela de los golpes duros). Este capítulo tiene el propósito de brindarle un plan estratégico para administrar proyectos mientras aprende acerca de los sistemas digitales y las herramientas modernas que se utilizan para desarrollarlos. Los principios aquí no se limitan a proyectos digitales, o incluso electrónicos, en general. Podrían aplicarse para construir una casa o su propio negocio. Sin duda mejorarán su probabilidad de tener éxito y reducirán el factor de frustración.

En realidad, los lenguajes de descripción de hardware se crearon con el fin de administrar sistemas digitales extensos: para la documentación, la prueba de simulación y para sintetizar los circuitos funcionales. De igual forma, las herramientas de software de Altera están diseñadas de manera específica para trabajar con proyectos de administración que van más allá del alcance de este libro. A medida que avancemos describiremos algunas de las características de los paquetes de software de Altera a través de los pasos para desarrollar estos proyectos pequeños. Este concepto de desarrollo modular de proyectos, que introdujimos en el capítulo 4, se demostrará aquí mediante una serie de ejemplos.

10-1 ADMINISTRACIÓN DE PEQUEÑOS PROYECTOS

Los primeros proyectos que describiremos aquí son sistemas relativamente pequeños que consisten de un pequeño número de bloques de construcción. Estos proyectos pueden desarrollarse en módulos separados, pero esto sólo aumentaría la complejidad. Son tan pequeños que tiene sentido implementar el proyecto completo en un solo archivo de diseño de HDL. Sin embargo, esto no significa que *no* haya que seguir un proceso estructurado para completar el proyecto. De hecho, la mayoría de los mismos pasos que deben emplearse en un proyecto modular extenso pueden aplicarse también en estos ejemplos. Los pasos a seguir son (1) definición general, (2) planeación estratégica para descomponer el proyecto en piezas pequeñas, (3) síntesis y prueba de cada pieza e (4) integración y prueba del sistema.

Definición

El primer paso en cualquier proyecto es la definición detallada de su alcance. En este paso hay que determinar lo siguiente:

- ¿Cuántos bits de datos se requieren?
- ¿Cuántos dispositivos se controlan mediante las salidas?
- ¿Cuáles son los nombres de cada entrada y salida?
- ¿Son las entradas y salidas activas en ALTO o activas en BAJO?
- ¿Cuáles son los requerimientos de velocidad?
- ¿Comprendo por completo la forma en que debe operar este dispositivo?
- ¿Qué es lo que definirá la conclusión exitosa de este proyecto?

De este paso debe producirse una descripción completa y detallada de la operación del proyecto en general, una definición de sus entradas y salidas, y las especificaciones numéricas completas que definan sus capacidades y limitaciones.

Planeación estratégica

El segundo paso implica desarrollar una estrategia para dividir este proyecto completo en piezas manejables. Los requerimientos de estas piezas son:

- Hay que desarrollar una forma de probar cada pieza.
- Cada pieza debe ajustarse con las demás para conformar el sistema completo.
- Debemos conocer la naturaleza de todas las señales que conectan las piezas.
- Se debe definir y comprender con detalle la operación exacta de cada bloque.
- Debemos tener una visión clara de cómo hacer que cada bloque funcione.

El último requerimiento podría parecer obvio, pero es sorprendente ver cuántos proyectos se planean alrededor de un bloque central que involucra un milagro técnico que aún no se ha descubierto, o que torpemente viola leyes elementales tales como la de la conservación de la energía. En esta etapa, cada subsistema (bloque de sección) se convierte en algo así como un proyecto por sí solo, con la posibilidad de definir subsistemas adicionales dentro de sus límites. Éste es el concepto del diseño jerárquico.

Síntesis y prueba

Cada subsistema debe construirse empezando en el nivel más simple. En el caso de un sistema digital diseñado mediante el uso de HDL, significa escribir piezas de código. También significa desarrollar un plan para evaluar ese código y asegurarse de que cumpla con todos los criterios. A menudo esto se logra a través de cierto tipo de simulación. Cuando se simula un circuito en una computadora, el diseñador

debe crear todos los escenarios que experimentará el circuito real y también debe conocer la respuesta apropiada a esas entradas. Con frecuencia, esta evaluación requiere pensarse mucho y no es un área que deba subestimarse. El peor error que usted puede cometer es concluir que un bloque fundamental funciona de manera perfecta, sólo para descubrir después aquellas pocas situaciones en las que falla. Es común que este predicamento lo obligue a rediseñar muchos de los otros bloques, con lo cual se anula la mayor parte de su trabajo.

Integración y prueba del sistema

El último paso es juntar los bloques y evaluarlos como una unidad. Se agregan y se evalúan bloques en cada etapa, hasta que todo el proyecto está funcionando. A menudo esta área se vuelve trivial pero raras veces se lleva a cabo sin problemas. Aún y cuando usted se haga cargo de todos los detalles que consideró, siempre existirán las excepciones que nadie había considerado.

Algunos aspectos de la planeación y administración de proyectos van más allá del alcance de este libro. Uno de ellos es la selección de la plataforma de hardware que se adapte mejor en su aplicación. En el capítulo 13 exploraremos el extenso campo de los sistemas digitales y analizaremos en forma específica las capacidades y limitaciones de los PLDs en diversas categorías. Otra dimensión muy crítica en la administración de proyectos es el tiempo. Su jefe sólo le dará cierta cantidad de tiempo para completar su proyecto, por lo cual usted debe planear su trabajo (y esfuerzo) para cumplir con esta fecha de entrega. No vamos a cubrir la administración del tiempo en este libro, pero como regla general descubrirá que la mayoría de las facetas del proyecto requerirán de dos a tres veces más de lo que usted pensaba cuando comenzó.

PREGUNTAS DE REPASO

1. Nombre los pasos de la administración de proyectos.
2. ¿En qué etapa debe decidir cómo medir el éxito?

10-2 PROYECTO DE CONTROLADOR DE MOTOR DE PASOS

El propósito de esta sección es demostrar una aplicación ordinaria de los contadores combinados con los circuitos decodificadores. A menudo un sistema digital contiene un contador que avanza a través de una secuencia especificada y cuyos estados de salida se decodifican mediante un circuito lógico combinacional, el cual a su vez controla la ponderación del sistema. Muchas aplicaciones tienen también entradas externas que se utilizan para colocar el sistema en varios modos de operación. En esta sección hablaremos de todas estas características para controlar un motor de pasos.

En un proyecto real, el primer paso de definición involucra con frecuencia cierta investigación por parte del administrador del proyecto. En esta sección (o proyecto) es vital que comprendamos lo que es un motor de pasos y cómo funciona antes de tratar de idear un supuesto circuito para controlarlo. En la sección 7-10 le mostramos cómo diseñar un contador síncrono simple que podría utilizarse para controlar un motor de pasos. La secuencia que se demostró en esa sección se conoce como *secuencia de paso completo*. Como podrá recordar, se requieren dos flip-flops y sus salidas Q y $\overline{Q}$ para controlar los cuatro devanados del motor. La secuencia de paso completo siempre tiene energizados dos devanados del motor en cualquier estado de la secuencia y, por lo general, produce 15° de rotación del eje por cada paso. No obstante, otras secuencias también harán que un motor de pasos gire. Si analiza la secuencia de paso completo, podrá observar que cada transición de estado implica apagar un devanado y encender otro en forma simultánea. Por ejemplo, analice el primer estado (1010) en la secuencia de paso completo de la tabla 10-1.

Cuando cambia al segundo estado en la secuencia, el *devanado 1* se apaga y el *devanado 0* se enciende. La *secuencia de medio paso* se crea mediante la inserción de un estado con sólo un devanado energizado entre los pasos completos, como se muestra en la columna central de la tabla 10-1. En esta secuencia se desenergiza un devanado antes de energizar el otro. El primer estado es 1010 y el segundo es 1000, lo cual significa que el *devanado 1* se apaga para un estado antes de que se encienda el *devanado 0*. Este estado intermedio hace que el eje del motor de pasos gire la mitad del recorrido (7.5°) que tendría en la secuencia de paso completo (15°). La secuencia de medio paso se utiliza cuando conviene tener pasos más pequeños y es aceptable tener más pasos por revolución. En consecuencia, el motor de pasos girará en forma similar a la secuencia de paso completo (15° por paso) si sólo aplica la secuencia de estados intermedios, con un devanado energizado en un momento dado. Esta secuencia, a la cual se le conoce como *secuencia de control de onda*, tiene menos torsión pero opera de manera más uniforme que la secuencia de paso completo a velocidades moderadas. La secuencia de control de onda se muestra en la columna de la derecha de la tabla 10-1.

TABLA 10-1 Secuencias de control del devanado del motor de pasos.

Paso completo	Medio paso	Control de onda
1010	1010	
	1000	1000
1001	1001	
	0001	0001
0101	0101	
	0100	0100
0110	0110	
	0010	0010

Enunciado del problema

Un laboratorio de microprocesadores necesita una interfase universal para controlar un motor de pasos. Para poder experimentar con microcontroladores que controlan motores de pasos, sería conveniente tener un CI de interfase universal conectado al motor de pasos. Este circuito necesita aceptar cualquiera de las formas típicas de señales de control de motores de pasos provenientes de un microcontrolador, y debe activar los devanados del motor para hacerlo que se mueva de la manera deseada. La interfase necesita operar en uno de cuatro modos: paso completo decodificado, medio paso decodificado, control de onda decodificado o control directo no decodificado. El modo se selecciona mediante el control de los niveles lógicos en las terminales de entrada *M1*, *M0*. En los primeros tres modos la interfase recibe sólo dos bits de control (un bit de pulso de paso y un bit de control de sentido del giro) del microcontrolador. Cada vez que detecta un flanco positivo en la entrada de paso, el circuito debe hacer que el motor avance un incremento de movimiento ya sea en sentido de las manecillas del reloj o en contra, de acuerdo con el nivel presente en el bit de sentido. Dependiendo del modo en el que se encuentre el CI, las salidas responderán a cada pulso de paso con un cambio de estado, de acuerdo con las secuencias que se muestran en la tabla 10-1. El cuarto modo de operación de este circuito debe permitir que el microcontrolador controle cada devanado del motor en forma directa. En este modo, el circuito acepta cuatro bits de control del microcontrolador y pasa estos niveles lógicos directamente hacia sus salidas, las cuales se utilizan para energizar los devanados del motor de pasos. La tabla 10-2 sintetiza los cuatro modos.

TABLA 10-2

Modo	M1	M0	Señales de entrada	Salida
0	0	0	Paso, sentido	Secuencia de conteo de paso completo
1	0	1	Paso, sentido	Secuencia de conteo de control de onda
2	1	0	Paso, sentido	Secuencia de conteo de medio paso
3	1	1	Cuatro entradas de control	Control directo de las entradas de control

En los modos 0, 1 y 2, las salidas cuentan a través de la secuencia de conteo correspondiente en cada flanco de subida de la entrada de paso. La entrada de sentido determina si la secuencia avanza hacia delante o hacia atrás, a través de los estados en la tabla 10-1, con lo cual el motor gira en sentido de las manecillas del reloj o en contra. Con base en esta decisión podemos tomar algunas decisiones acerca del proyecto.

Entradas

paso: disparo en flanco de subida

sentido: 0 = hacia atrás a través de la tabla, 1 = hacia delante a través de la tabla

cent0, cent1, cent2, cent3, m1, m0: entradas de control activas en ALTO

Salidas

csal0, csal1, csal2, csal3: salidas de control activas en ALTO

Planeación estratégica

Este proyecto tiene dos requerimientos clave. Requiere un circuito contador secuencial para controlar las salidas en tres de los modos. En el último modo la salida no sigue a un contador, sino a las entradas de control. Aunque existen muchas formas de dividir este proyecto y de todas formas cumplir con estos requerimientos, optaremos por tener un contador binario ascendente/descendente simple que responda a las entradas de paso y de sentido. Un circuito lógico combinacional separado traducirá (decodificará) el conteo binario en el estado de salida apropiado, dependiendo de la configuración de entrada de modo. Este circuito también ignorará las entradas del contador y pasará las entradas de control en forma directa hacia las salidas cuando el modo se establezca en 3. En la figura 10-1 se muestra el diagrama del circuito.

También es bastante simple el proceso de descomponer este problema en piezas manejables. El primer paso es construir un contador ascendente/descendente. Deberemos probar este contador en un simulador, utilizando sólo las entradas de sentido y de paso. Después hay que tratar de hacer que cada secuencia decodificada funcione en forma individual con el contador. Luego hay que tratar de hacer que las entradas de modo seleccionen una de las secuencias del decodificador y que se agregue la opción de control directo (que es bastante trivial). Cuando el circuito pueda seguir los estados que se muestran en la tabla 10-1 en cualquier sentido, para cada secuencia de modo y que pueda pasar las cuatro señales *cent* de manera directa hacia *csal* en el modo 3, tendremos éxito.

Síntesis y prueba

El código en las figuras 10-2 y 10-3 muestra la primera etapa de desarrollo: diseñar y probar un contador ascendente/descendente. Vamos a utilizar una variable entera intermedia para el valor del contador y lo evaluaremos enviando de salida el conteo en forma directa hacia *q*. Para evaluar esta parte del diseño tan sólo necesitamos asegurarnos de que pueda contar hacia arriba y hacia abajo, a través de los ocho estados. La figura 10-4 muestra los resultados de la simulación. Sólo necesitamos proporcionar los pulsos de reloj y crear una señal de control de sentido del giro, y el simulador demuestra la respuesta del contador.

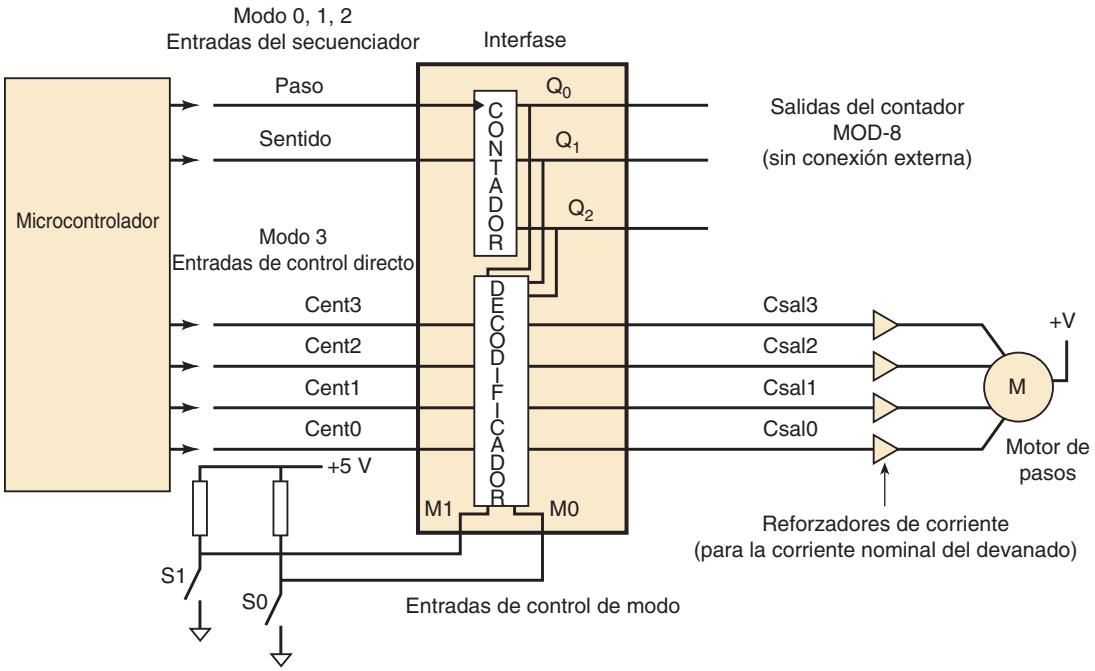


FIGURA 10-1 Un circuito de interfase para un motor de pasos universal.

```
SUBDESIGN  fig10 2
(
  paso, dir      :INPUT;
  q[2..0]        :OUTPUT;
)
VARIABLE
conteo[2..0]      : DFF;

BEGIN
  conteo[].clk = paso;
  IF dir THEN conteo[].d = conteo[].q + 1;
  ELSE        conteo[].d = conteo[].q   1;
  END IF;
  q[] = conteo[].q;
END;
```

FIGURA 10-2 MOD-8 en AHDL.

```
ENTITY fig10 3 IS
PORT( paso, dir   :IN BIT;
      q           :OUT INTEGER RANGE 0 TO 7);
END fig10 3;

ARCHITECTURE vhdl OF fig10 3 IS
BEGIN
   PROCESS (paso)
   VARIABLE conteo     :INTEGER RANGE 0 TO 7;
   BEGIN
      IF (paso'EVENT AND paso = '1') THEN
        IF dir = '1' THEN conteo := conteo + 1;
        ELSE              conteo := conteo   1;
        END IF;
      END IF;
      q <= conteo;
   END PROCESS;
END vhdl;
```

FIGURA 10-3 MOD-8 en VHDL.

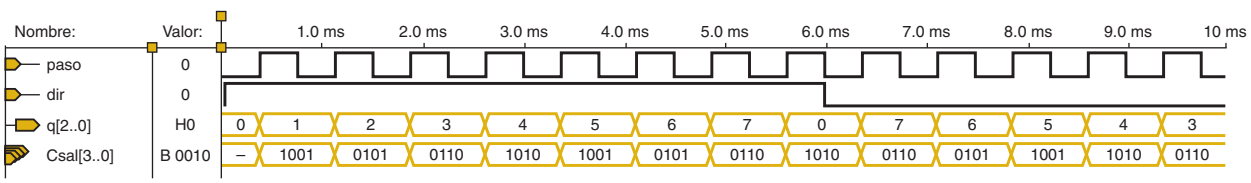


FIGURA 10-4 Prueba de simulación de un MOD-8 básico.

El siguiente paso es agregar una de las salidas decodificadas y probarla, para lo cual tendrá que agregar la especificación de salida *csal* de cuatro bits. Los bits de la salida *q* del contador MOD-8 se mantienen por cuestión de continuidad. La figura 10-5 muestra el código de AHDL para esta etapa de prueba y la figura 10-6 muestra el código de VHDL para la misma etapa de prueba. Observe que se utiliza una instrucción CASE para decodificar el contador y controlar las salidas. En el código de VHDL, las salidas *csal* se declaran como de tipo bit_vector, ya que ahora queremos asignarles patrones binarios de bits. La figura 10-7 muestra la prueba simulada de su operación con suficientes ciclos de reloj incluidos como para evaluar un ciclo ascendente y descendente completo de un contador.

Las demás secuencias de conteo son simples variaciones del código que acabamos de evaluar. Tal vez no sea necesario evaluar cada uno de manera independiente, por lo que ahora es un buen momento para incluir las entradas de selector (*m*) y las entradas de control directo del devanado (*cent*). Observe que las nuevas entradas se definen en las figuras 10-8 (AHDL) y 10-9 (VHDL). Como el control de modo tiene cuatro estados posibles y queremos hacer algo distinto para cada estado, es mejor utilizar otra instrucción CASE. En otras palabras, hemos optado por usar una estructura CASE para seleccionar el modo y una estructura CASE dentro de cada modo para

```
SUBDESIGN  fig10_5
(
   paso, dir      :INPUT;
   q[2..0]        :OUTPUT;
   csal[3..0]     :OUTPUT;
)
VARIABLE
   conteo[2..0]    : DFF;

BEGIN
   conteo[].clk   paso;
   IF dir THEN conteo[].d   conteo[].q + 1;
   ELSE        conteo[].d   conteo[].q - 1;
   END IF;
   q[]   conteo[].q;
   CASE conteo[] IS
      WHEN  B"000"    > csal[]   B"1010";
      WHEN  B"001"    > csal[]   B"1001";
      WHEN  B"010"    > csal[]   B"0101";
      WHEN  B"011"    > csal[]   B"0110";
      WHEN  B"100"    > csal[]   B"1010";
      WHEN  B"101"    > csal[]   B"1001";
      WHEN  B"110"    > csal[]   B"0101";
      WHEN  B"111"    > csal[]   B"0110";
   END CASE;
END;
```

FIGURA 10-5 Decodificador de secuencia de paso completo en AHDL.

```
ENTITY fig10_6 IS
PORT (   paso, dir :IN BIT;
         q         :OUT INTEGER RANGE 0 TO 7;
         csal      :OUT BIT_VECTOR (3 downto 0));
END fig10_6;

ARCHITECTURE vhdl OF fig10_6 IS
BEGIN
   PROCESS (paso)
   VARIABLE conteo   :INTEGER RANGE 0 TO 7;
   BEGIN
      IF (paso'EVENT AND paso   '1') THEN
         IF dir   '1' THEN conteo :  conteo + 1;
         ELSE              conteo :  conteo - 1;
         END IF;
         q <  conteo;
      END IF;
         CASE conteo IS
            WHEN 0    > csal <  B"1010";
            WHEN 1    > csal <  B"1001";
            WHEN 2    > csal <  B"0101";
            WHEN 3    > csal <  B"0110";
            WHEN 4    > csal <  B"1010";
            WHEN 5    > csal <  B"1001";
            WHEN 6    > csal <  B"0101";
            WHEN 7    > csal <  B"0110";
         END CASE;
       END PROCESS;
END vhdl;
```

FIGURA 10-6 Decodificador de secuencia de paso completo en VHDL.

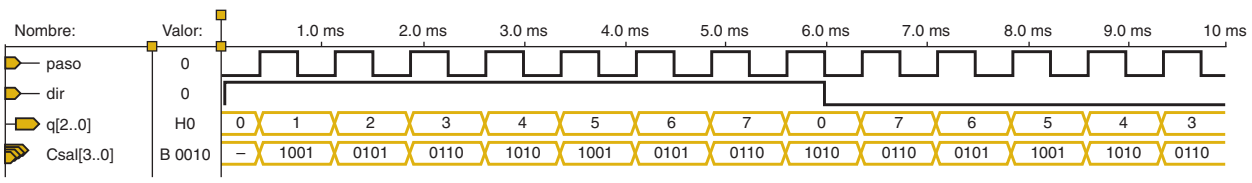


FIGURA 10-7 Prueba de simulación de una secuencia decodificada.

FIGURA 10-8 Controlador de motor de pasos en AHDL.

```
SUBDESIGN  fig10 8
(
   paso, dir              :INPUT;
   m[1..0], cent[3..0]    :INPUT;
   csal[3..0], q[2..0]    :OUTPUT;
)
VARIABLE
   conteo[2..0]    : DFF;
BEGIN
   conteo[].clk = paso;
   IF dir THEN conteo[].d = conteo[].q + 1;
   ELSE        conteo[].d = conteo[].q   1;
   END IF;
   q[] = conteo[].q;
   CASE m[] IS
   WHEN 0 =>
         CASE conteo[] IS            PASO COMPLETO
         WHEN  B"000"   => csal[] = B"1010";
         WHEN  B"001"   => csal[] = B"1001";
         WHEN  B"010"   => csal[] = B"0101";
         WHEN  B"011"   => csal[] = B"0110";
         WHEN  B"100"   => csal[] = B"1010";
         WHEN  B"101"   => csal[] = B"1001";
         WHEN  B"110"   => csal[] = B"0101";
         WHEN  B"111"   => csal[] = B"0110";
         END CASE;
   WHEN 1 =>
         CASE conteo[] IS            CONTROL DE ONDA
         WHEN  B"000"   => csal[] = B"1000";
         WHEN  B"001"   => csal[] = B"0001";
         WHEN  B"010"   => csal[] = B"0100";
         WHEN  B"011"   => csal[] = B"0010";
         WHEN  B"100"   => csal[] = B"1000";
         WHEN  B"101"   => csal[] = B"0001";
         WHEN  B"110"   => csal[] = B"0100";
         WHEN  B"111"   => csal[] = B"0010";
         END CASE;
   WHEN 2 =>
         CASE conteo[] IS            MEDIO PASO
         WHEN  B"000"   => csal[] = B"1010";
         WHEN  B"001"   => csal[] = B"1000";
         WHEN  B"010"   => csal[] = B"1001";
         WHEN  B"011"   => csal[] = B"0001";
         WHEN  B"100"   => csal[] = B"0101";
         WHEN  B"101"   => csal[] = B"0100";
         WHEN  B"110"   => csal[] = B"0110";
         WHEN  B"111"   => csal[] = B"0010";
         END CASE;
   WHEN 3 =>   csal[] = cent[];      Direct Drive
   END CASE;
END;
```

FIGURA 10-9 Controlador de motor de pasos en VHDL.

```
ENTITY fig10 9 IS
PORT (   paso, dir   :IN BIT;
         m           :IN BIT VECTOR (1 DOWNTO 0);
         cent        :IN BIT VECTOR (3 DOWNTO 0);
         q           :OUT INTEGER RANGE 0 TO 7;
         csal        :OUT BIT VECTOR (3 DOWNTO 0));
END fig10 9;

ARCHITECTURE vhdl OF fig10 9 IS
BEGIN
   PROCESS (paso)
   VARIABLE conteo    :INTEGER RANGE 0 TO 7;
   BEGIN
      IF (paso'EVENT AND step = '1') THEN
         IF dir = '1' THEN conteo := conteo + 1;
         ELSE              conteo := conteo   1;
         END IF;
      END IF;
      q <= conteo;
   CASE m IS
      WHEN "00" =>                    PASO COMPLETO
         CASE conteo IS
            WHEN 0   => csal <= "1010";
            WHEN 1   => csal <= "1001";
            WHEN 2   => csal <= "0101";
            WHEN 3   => csal <= "0110";
            WHEN 4   => csal <= "1010";
            WHEN 5   => csal <= "1001";
            WHEN 6   => csal <= "0101";
            WHEN 7   => csal <= "0110";
         END CASE;
      WHEN "01" =>                    CONTROL DE ONDA
         CASE conteo IS
            WHEN 0   => csal <= "1000";
            WHEN 1   => csal <= "0001";
            WHEN 2   => csal <= "0100";
            WHEN 3   => csal <= "0010";
            WHEN 4   => csal <= "1000";
            WHEN 5   => csal <= "0001";
            WHEN 6   => csal <= "0100";
            WHEN 7   => csal <= "0010";
         END CASE;
      WHEN "10" =>                    MEDIO PASO
         CASE conteo IS
            WHEN 0   => csal <= "1010";
            WHEN 1   => csal <= "1000";
            WHEN 2   => csal <= "1001";
            WHEN 3   => csal <= "0001";
            WHEN 4   => csal <= "0101";
            WHEN 5   => csal <= "0100";
            WHEN 6   => csal <= "0110";
            WHEN 7   => csal <= "0010";
         END CASE;
      WHEN "11" =>   csal <= cent;  Control directo
   END CASE;
   END PROCESS;
END vhdl;;
```

seleccionar la salida apropiada. Al proceso de utilizar una instrucción dentro de otra se le conoce como **anidamiento**. El uso de sangrías es muy importante para mostrar la estructura y la lógica del código, en especial cuando se utiliza el anidamiento.

Las simulaciones de la figura 10-10 verifican que el circuito parece estar funcionando en forma apropiada. Figura 10-10(a) muestra como cada estado decodifica en modo 0 (paso completo) y completa el ciclo en ambos sentidos. Observe que una vez que el modo (*m*) cambia a 01_2, la salida (*csal*) se decodifica como la secuencia de control de onda. La figura 10-10(b) muestra la secuencia de control de onda (modo 1) en ambos sentidos y después cambia el modo a 10_2, lo cual produce la secuencia de medio paso que se está decodificando del contador MOD-8. Por último, la figura 10-10(c) muestra cómo el modo de medio paso cuenta en forma cíclica hacia arriba y empieza de nuevo desde abajo. Después cambia al modo 3 (control directo) en 7.5 ms, con lo cual demuestra que los datos en *cent* se transfieren en forma asíncrona a las salidas. Observe que los valores elegidos para *cent* aseguran que cada bit pueda cambiar a ALTO y a BAJO.

La integración y prueba finales deben implicar algo más que sólo la simulación. Hay que conectar al circuito un verdadero motor de pasos y el reforzador de corriente para probarlos. En este caso, la velocidad de cada paso que utilice la simulación será probablemente más rápida de lo que el motor de pasos pueda soportar, por lo que tendría que reducirse para una verdadera prueba funcional de hardware.

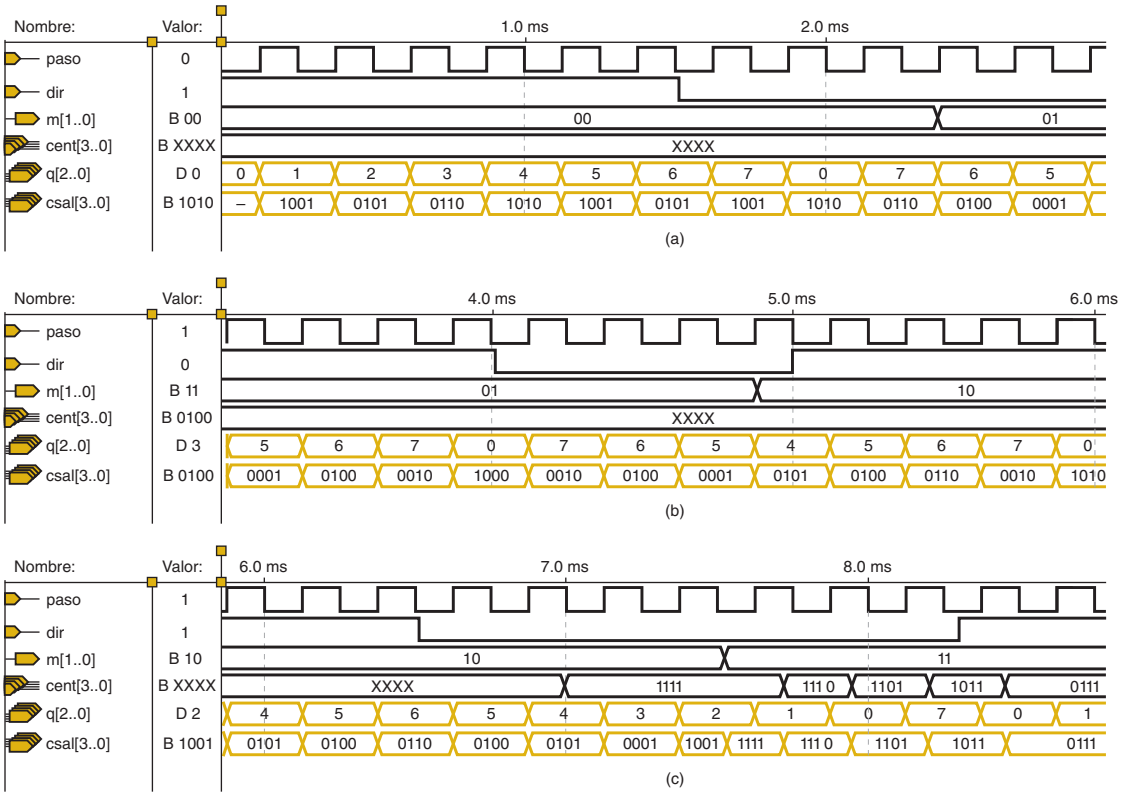


FIGURA 10-10 Prueba de simulación del controlador del motor de pasos completo.

PREGUNTAS DE REPASO

1. ¿Cuáles son los cuatro modos de operación para este controlador de motor de pasos?
2. ¿Cuáles son las entradas para el modo de control directo?
3. ¿Cuáles son las entradas para el modo de control de onda?
4. ¿Cuántos estados hay en la secuencia de medio paso?

10-3 PROYECTO DE CODIFICADOR DE TECLADO NUMÉRICO

Otra importante habilidad que tratamos de reforzar es el análisis de circuitos. Esto podría sonar como algo sacado de un libro de texto analógico, pero en realidad necesitamos poder analizar y comprender la manera en que operan los circuitos digitales existentes. En esta sección presentaremos un circuito y analizaremos la manera en que opera. Después utilizaremos las habilidades que adquirimos para rediseñar el circuito y escribir el código para éste en HDL.

Análisis del problema

Para reforzar los conceptos de codificación del capítulo 9, vamos a presentar un circuito digital muy útil, en el cual se codifica un teclado numérico hexadecimal (16 teclas) en una salida binaria de cuatro bits. Por lo general, los codificadores como éste tienen una salida de estrobo, la cual indica cuando alguien oprime y suelta una tecla. Como los teclados numéricos se integran con frecuencia al sistema de bus de una microcomputadora, las salidas codificadas deben tener señales de habilitación triestado. La figura 10-11 muestra el diagrama de bloques del codificador de teclado numérico.

El método del codificador de prioridad que se muestra en el capítulo 9 (figura 9-15) es eficiente para teclados numéricos pequeños, pero teclados grandes como los que se utilizan en las computadoras personales deben utilizar una técnica distinta. En éstos, cada tecla no es un interruptor independiente conectado a V_{CC} o a tierra; sino que el interruptor de cada tecla se utiliza para conectar una fila con una columna en la matriz del teclado. Cuando no se oprimen teclas, no hay

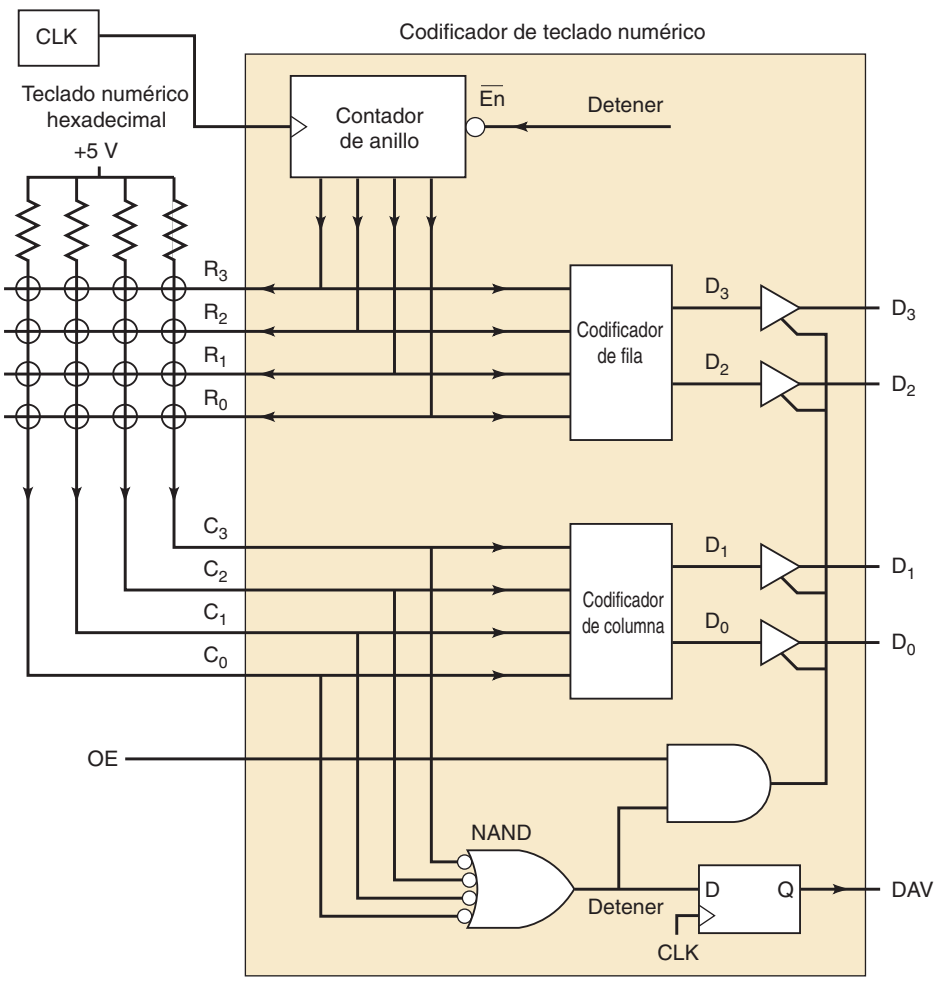


FIGURA 10-11
Diagrama de bloques del codificador de teclado numérico.

conexiones entre las filas y las columnas. El truco de saber cuál tecla se oprimió se logra mediante la acción de activar (llevar a BAJO) una fila a la vez y después se revisa para ver si alguna de las columnas cambió a BAJO. Si una de las columnas tiene un nivel BAJO, entonces la tecla que se está oprimiendo se encuentra en la intersección de la fila y la columna activadas que se encuentran en ese momento en BAJO. Si no hay columnas en BAJO, sabremos que no se están oprimiendo teclas en la fila activada y podemos pasar a revisar la siguiente fila, llevándola al nivel BAJO. Al proceso de activar filas en forma secuencial se le conoce como *explorar* el teclado. La ventaja de este método es la reducción de las conexiones que van al teclado numérico. En este caso, 16 teclas pueden codificarse mediante el uso de ocho entradas/salidas.

Cada tecla representa una combinación única de un número de fila y un número de columna. Al numerar las filas y columnas en forma estratégica, podemos combinar los números binarios de fila y columna para crear el valor binario de las teclas hexadecimales, como se muestra en la figura 10-12. En esa figura, la fila 1 (01_2) se lleva a BAJO y los datos en el codificador de columna son 10_2, por lo que es evidente que se oprimió el botón en la fila 1, columna 2. La compuerta NAND en la figura 10-11 se utiliza para determinar si alguna columna está en BAJO, lo cual indica que se oprimió una tecla en la fila que esté activa en ese momento. La salida de esta compuerta se llama *DETENER* porque cuando se oprime una tecla, queremos detener el contador de anillo y dejar de explorar hasta que se libere la tecla. A medida que los codificadores pasan a través de su retraso de propagación y que los búferes triestado se habilitan, las salidas de datos se encuentran en un estado transitorio. En el siguiente flanco de subida del reloj, el flip-flop D transferirá un nivel ALTO de *DETENER* hacia la salida *DAV*, lo cual indicará que se está oprimiendo una tecla y que los datos válidos están disponibles.

Como vimos en el capítulo 7, un contador de registro de desplazamiento (contador de anillo) se utiliza para generar la exploración secuencial de las cuatro filas. La secuencia de conteo utiliza cuatro estados, en donde cada uno tiene un bit distinto que se lleva al nivel BAJO. Cuando se detecta una tecla oprimida, el contador de anillo debe retener su estado actual (detener) hasta que se libera la tecla. La figura 10-13 muestra el diagrama de transiciones de estado. Cada estado de este contador debe codificarse para generar un número de fila binario de dos bits. El valor de

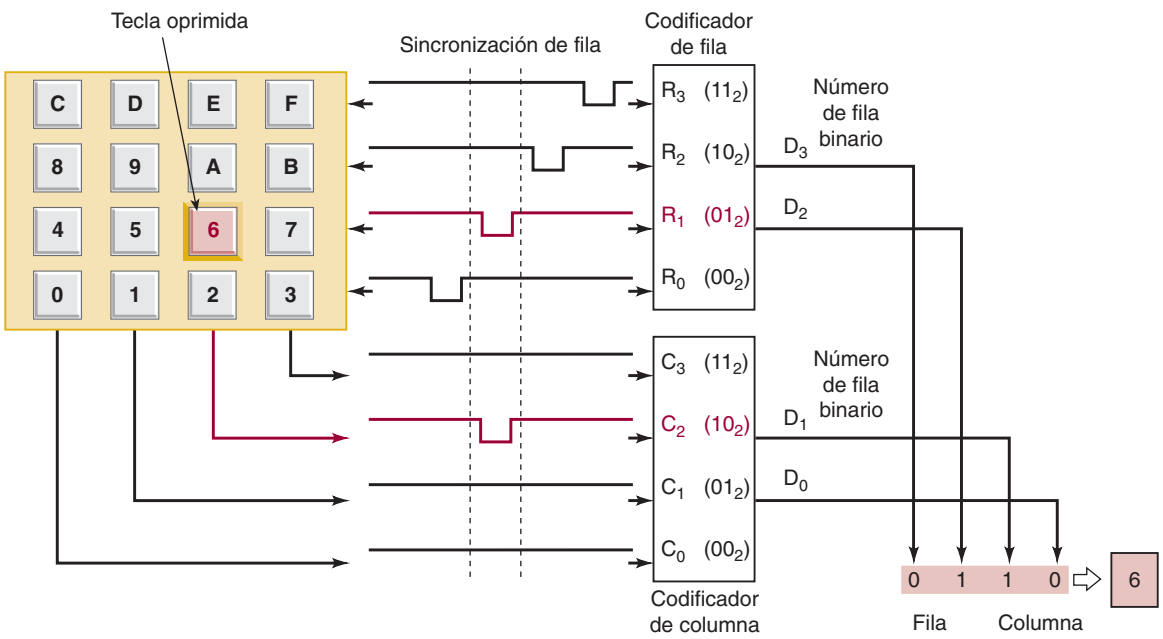


FIGURA 10-12 Operación del codificador cuando se oprime la tecla "6".

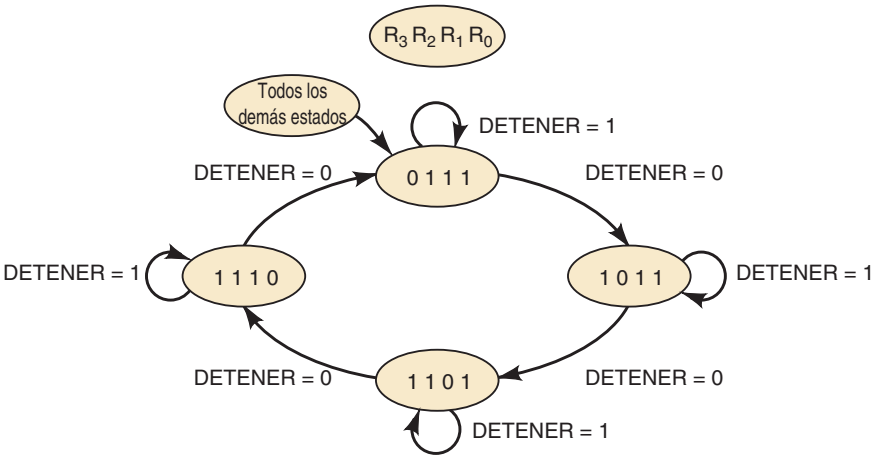


FIGURA 10-13
Diagrama de estado del contador de anillo con control de fila.

cada columna también debe codificarse para generar un número de columna binario de dos bits. El sistema requerirá las siguientes entradas y salidas.

4	Salidas de control de fila	R_0–R_3
4	Entradas de lectura de columna	C_0–C_3
4	Salidas de datos codificadas	D_0–D_3
1	Salida de estrobo disponible para datos	*DAV*
1	Entrada de habilitación triestado	*OE*
1	Entrada de reloj	*CLK*

Planeación estratégica

Este circuito ya se encuentra estructurado, por lo cual podemos escribir con facilidad piezas de código de HDL para emular cada una de las secciones del sistema. Los principales bloques son:

Un contador de anillo con salidas activas en BAJO.

Dos codificadores para los números de fila y de columna.

Detección de pulsación de tecla y circuitos de habilitación triestado.

En capítulos anteriores ya exploramos estos circuitos, por lo que no mostraremos el desarrollo y la prueba de cada bloque aquí. Las soluciones que se muestran a continuación saltan directamente a la fase de integración y prueba del proyecto.

SOLUCIÓN EN AHDL

Las entradas y salidas (vea la figura 10-14) se definen en las líneas 3 a 8 y van después de la descripción que se obtiene del análisis del diagrama esquemático. La sección VARIABLE define varias características de este circuito codificador. El bit detener detecta cuando se oprime una tecla. El nodo de datos se utiliza para combinar los datos del codificador de la fila y la columna. El arreglo de bits *ts* (línea 13) representa un búfer triestado, como vimos en el capítulo 9. Recuerde que cada bit de este búfer tiene una entrada (*ts[ ].IN*), una salida (*ts.OUT*) y una señal de habilitación de salida (*ts[ ].OE*). El bit *datos_disp* (línea 14) representa un flip-flop D con las entradas *datos_disp.CLK*, *datos_disp.D* y la salida *datos_disp.Q*.

AHDL

```
SUBDESIGN fig10 14
(
   clk           :INPUT;
   col[3..0]     :INPUT;
   oe            :INPUT;      habilitación de salida triestado
   fila[3..0]    :OUTPUT;
   d[3..0]       :OUTPUT;
   dav           :OUTPUT;     datos disponibles
)
VARIABLE
detener          :NODE;
datos[3..0]      :NODE;
ts[3..0]         :TRI;
datos disp       :DFF;
anillo: MACHINE OF BITS (fila[3..0])
WITH STATES (s1 = B"1110", s2 = B"1101", s3 = B"1011", s4 = B"0111",
                    % s = estados del anillo %
             f1 = B"0001", f2 = B"0010", f3 = B"0011", f4 = B"0100",
             f5 = B"0101", f6 = B"0110", f7 = B"1000", f8 = B"1001",
             f9 = B"1010", fa = B"1100", fb = B"1111", fc = B"0000");
         % f = estados que no se utilizan   > diseño autocorregible %
BEGIN
   anillo.CLK = clk;
   anillo.ENA = !detener;
   datos disp.CLK = clk;
   datos disp.D = detener;
   dav = datos disp.Q;
   ts[].OE = oe & detener;
   ts[].IN = datos[];
   d[] = ts[].OUT;

   CASE anillo IS
      WHEN s1 =>   anillo = s2;    datos[3..2] = B"00";
      WHEN s2 =>   anillo = s3;    datos[3..2] = B"01";
      WHEN s3 =>   anillo = s4;    datos[3..2] = B"10";
      WHEN s4 =>   anillo = s1;    datos[3..2] = B"11";
      WHEN OTHERS => anillo = s1;
   END CASE;

   CASE col[] IS
      WHEN B"1110" =>   datos[1..0] = B"00";     detener = VCC;
      WHEN B"1101" =>   datos[1..0] = B"01";     detener = VCC;
      WHEN B"1011" =>   datos[1..0] = B"10";     detener = VCC;
      WHEN B"0111" =>   datos[1..0] = B"11";     detener = VCC;
      WHEN OTHERS  =>   datos[1..0] = B"00";     detener = GND;
   END CASE;
END;
```

FIGURA 10-14 Codificador para exploración de teclado numérico en AHDL.

Las líneas 15 a 20 demuestran una poderosa característica de AHDL, la cual nos permite definir una máquina de estados, en donde cada estado se conforma por el patrón de bits que necesitamos. En la línea 15 se dio el nombre *anillo* a esta máquina de estados, ya que actúa como un contador de anillo. Los bits que conforman esta máquina contador de anillo son los cuatro bits de fila que se definieron en la línea 6. Estos estados se etiquetan como *s1-s4* y tienen sus patrones de bits asignados a ellos, de manera que uno de los cuatro bits esté en BAJO para cada estado, como un contador de anillo activo en BAJO. Los otros doce estados se especifican mediante un nivel arbitrario que comienza con *f* para indicar que no son estados válidos. En esencia, las líneas 23 a 30 conectan todos los componentes, como se muestra en el dibujo del circuito de la figura 10-11. En las líneas 32 a 38 se describen tanto la secuencia de conteo de anillo como la codificación del valor de la fila. Para cada valor de estado PRESENTE de *anillo* se define el estado SIGUIENTE, así como la salida apropiada del codificador de fila (*datos[3..2]*). La línea 37 asegura que este contador se inicie en forma automática al enviarlo a *s1* desde cualquier otro estado que no sea *s1-s4*. En las líneas 40 a 46 se describe la codificación del valor de la columna. Observe que la generación de la señal *detener* en este diseño no sigue el diagrama de la figura 10-11 con exactitud. En este diseño, en vez de aplicar una operación AND a las columnas, la estructura CASE activa a *detener* sólo cuando una (y sólo una) columna está en BAJO. Por ende, si se oprimieran varias teclas en la misma fila el codificador no reconocería a ninguna como válida y no activaría a *dav*.

SOLUCIÓN EN VHDL

Compare la descripción de VHDL de la figura 10-15 con el dibujo del circuito de la figura 10-11. Las entradas y salidas se definen en las líneas 5 a 9 y siguen la descripción que se obtuvo de analizar el diagrama esquemático. Se definen dos señales (SIGNAL) en las líneas 13 y 14 para este diseño. El bit *detener* detecta cuando se oprime una tecla. La señal *datos* se usa para combinar los datos del codificador relacionados con la fila y la columna para formar un valor de cuatro bits que represente la tecla que se oprimió. El contador de anillo se implementa mediante el uso de un bloque PROCESS, el cual responde a la entrada *clk*. La línea 26 asegura que este contador se inicie en forma automática al enviarlo al estado "1110" desde cualquier otro estado que no sean los que se encuentran en la secuencia de *anillo*. Observe que en la línea 20 se comprueba el estado de *detener* antes de utilizar una instrucción CASE para asignar un estado SIGUIENTE a *anillo*. Ésta es la forma en que se implementa la señal de habilitación de conteo en este diseño. En la línea 29 se actualiza en forma síncrona la salida de datos disponibles (*dav*) con el valor de *detener*. Es síncrona debido a que se encuentra dentro de la estructura IF (líneas 19 a 30) que detecta el flanco activo de reloj. Las instrucciones restantes (líneas 31 a 52) no dependen del flanco activo de reloj, sino que describen lo que hará el circuito en cualquier flanco del reloj.

La codificación del valor de fila se describe en las líneas 33 a 39. Para cada valor de estado PRESENTE de *anillo* se define la salida *data(*3 DOWNTO 2) del codificador de fila. La codificación del valor de columna se describe en las líneas 41 a 47. Observe que la generación de la señal *detener* en este diseño no sigue con exactitud el diagrama de la figura 10-11. En este diseño, en vez de aplicar una operación NAND a las columnas, la estructura CASE activa a *detener* sólo cuando una (y sólo una) columna está en BAJO. Por ende, si se oprimieran varias teclas en la misma fila, el codificador no reconocería ninguna como válida y no activaría a *dav*.

FIGURA 10-15
Codificador para exploración de teclado numérico en VHDL.

```
LIBRARY ieee;
USE ieee.std logic 1164.all;

ENTITY fig10 15 IS
PORT (   clk         :IN STD LOGIC;
         col         :IN STD LOGIC VECTOR (3 DOWNTO 0);
         fila        :OUT STD LOGIC VECTOR (3 DOWNTO 0);
         d           :OUT STD LOGIC VECTOR (3 DOWNTO 0);
         dav         :OUT STD LOGIC                      );
END fig10 15;

ARCHITECTURE vhdl OF fig10 15 IS
SIGNAL detener        :STD LOGIC;
SIGNAL datos          :STD LOGIC VECTOR (3 DOWNTO 0);
BEGIN
   PROCESS (clk)
   VARIABLE anillo    :STD LOGIC VECTOR (3 DOWNTO 0);
   BEGIN
      IF (clk'EVENT AND clk = '1')  THEN
         IF detener = '0'   THEN
            CASE anillo IS
               WHEN "1110" => anillo := "1101";
               WHEN "1101" => anillo := "1011";
               WHEN "1011" => anillo := "0111";
               WHEN "0111" => anillo := "1110";
               WHEN OTHERS => anillo := "1110";
            END CASE;
         END IF;
         dav <= detener;
      END IF;
      fila <= anillo;

      CASE anillo IS
         WHEN "1110" => datos(3 DOWNTO 2) <= "00";
         WHEN "1101" => datos(3 DOWNTO 2) <= "01";
         WHEN "1011" => datos(3 DOWNTO 2) <= "10";
         WHEN "0111" => datos(3 DOWNTO 2) <= "11";
         WHEN OTHERS => datos(3 DOWNTO 2) <= "00";
      END CASE;

      CASE col IS
         WHEN "1110" => datos(1 DOWNTO 0) <= "00";    detener <= '1';
         WHEN "1101" => datos(1 DOWNTO 0) <= "01";    detener <= '1';
         WHEN "1011" => datos(1 DOWNTO 0) <= "10";    detener <= '1';
         WHEN "0111" => datos(1 DOWNTO 0) <= "11";    detener <= '1';
         WHEN OTHERS => datos(1 DOWNTO 0) <= "00";    detener <= '0';
      END CASE;

      IF detener = '1'   THEN  d <= datos;
      ELSE                d <= "ZZZZ";
      END IF;
   END PROCESS;
END vhdl;
```

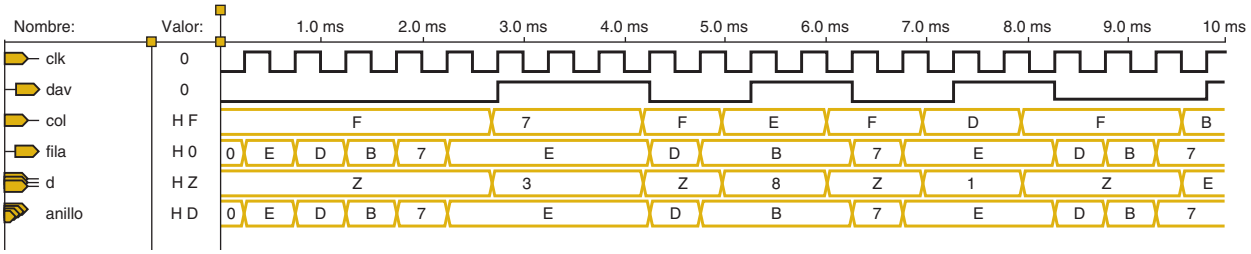


FIGURA 10-16 Simulación del codificador para exploración de teclado numérico.

La figura 10-16 muestra la simulación del proyecto. El diseñador introduce los valores de columna (*col*) como una entrada de prueba para simular el valor que se lee de las columnas del teclado a medida que se exploran las filas. Mientras que todas las columnas estén en ALTO (es decir, que en *col* esté el valor hexadecimal F), el contador de *anillo* estará habilitado y contando, *dav* estará en BAJO y las salidas *d* se encontrarán en el estado Hi-Z. Justo antes de la marca de los 3.0 ms se simula un 7 como entrada *col*, lo cual significa que una de las columnas cambió a BAJO. Esto simula que se detecta una tecla en la columna más significativa (C3) de la matriz de teclado numérico. Observe que como resultado de que la columna cambia a BAJO, en el siguiente flanco activo (de subida) del reloj la línea *dav* cambia a ALTO y el contador de anillo no cambia de estado. Se deshabilita para que no pase a su estado SIGUIENTE mientras que la tecla esté oprimida. En este punto, el valor de *fila* es E hexadecimal (1110_2), lo cual significa que el contador de anillo está llevando la fila menos significativa (R0) al nivel BAJO. El codificador de fila traduce esto en el número de fila binario (00). La tecla ubicada en la intersección de la fila menos significativa (00_2) y la columna más significativa (11_2) es la tecla 3 (vea la figura 10-12). En este punto, las salidas *d* retienen el valor de tecla codificado de 3 (0011_2). Justo antes de la marca de los 4 ms, la simulación imita la liberación de la tecla al cambiar el valor de columna de vuelta al 4 hexadecimal, lo cual hace que la salida *d* cambie a su estado Hi-Z. En el siguiente flanco de subida del reloj, la línea *dav* cambia a BAJO y el contador de anillo continúa con su secuencia de conteo.

PREGUNTAS DE REPASO

1. ¿Cuántas filas en el teclado explorado se activan en un momento dado?
2. Si se oprimen dos teclas en la misma columna en forma simultánea, ¿cuál de las teclas se codificará?
3. ¿Cuál es el propósito del flip-flop D en la terminal DAV?
4. ¿Será el tiempo entre la tecla que se está oprimiendo y cuando la señal DAV cambia al nivel ALTO siempre el mismo?
5. ¿Cuándo se encuentran las terminales de salida de datos en el estado Hi-Z?

10-4 PROYECTO DE RELOJ DIGITAL

Una de las aplicaciones más comunes de los contadores es el reloj digital; un reloj de tiempo real que visualiza la hora del día en horas, minutos y algunas veces segundos. Para construir un reloj digital preciso se requiere una señal de reloj con su frecuencia controlada eficientemente. Para los relojes operados por batería, la frecuencia básica se obtiene, por lo general, mediante un oscilador de cristal de cuarzo. Los relojes digitales que se operan mediante la línea de energía de corriente alterna pueden utilizar la frecuencia de *60 Hz* como la frecuencia básica de reloj. En cualquier caso, la frecuencia básica debe dividirse para obtener una frecuencia de 1 Hz o 1 pulso por segundo (pps). La figura 10-17 muestra el diagrama de bloques básico para un reloj digital que opera con 60 Hz.

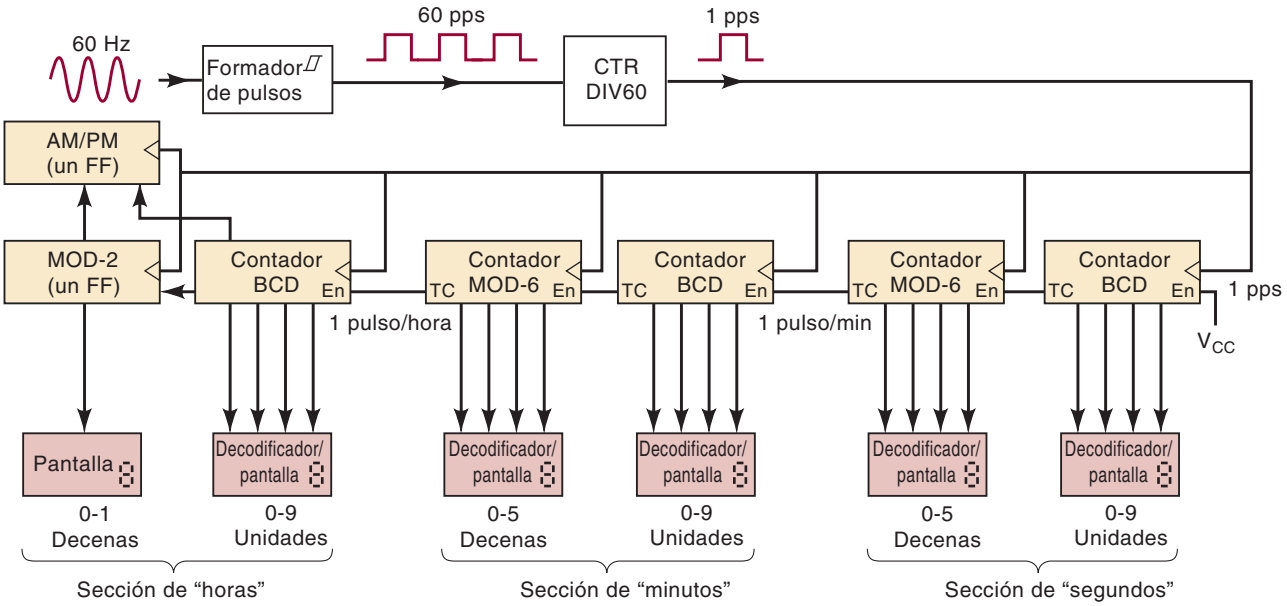


FIGURA 10-17 Diagrama de bloques para un reloj digital.

La señal de *60 Hz* se envía a través de un circuito disparador de Schmitt para producir pulsos cuadrados a la velocidad de 60 pps. Esta forma de onda de 60 pps se alimenta hacia un contador MOD-60, el cual se utiliza para dividir los 60 pps hasta 1 pps. La señal de 1 pps se utiliza como un reloj síncrono para todas las etapas del contador, las cuales se conectan en cascada en forma síncrona. La primera etapa es la sección SEGUNDOS, la cual se utiliza para contar y visualizar los segundos del 0 al 9. El contador *BCD* avanza un conteo por segundo. Cuando esta etapa llega a los 9 segundos, el contador BCD activa su salida de conteo terminal (*tc*) y en el siguiente flanco activo del reloj se recicla a 0. La salida de conteo terminal BCD habilita al contador MOD-6 y lo hace avanzar por un conteo al mismo tiempo que el contador BCD se recicla. Este proceso continúa durante 59 segundos, punto en el cual el contador MOD-6 se encuentra en el conteo 101 (5) y el contador BCD se encuentra en 1001 (9), de manera que la pantalla indique 59 s y que *tc* de MOD-6 se encuentre en ALTO. El siguiente pulso recicla el contador BCD y el contador MOD-6 a cero (recuerde que el MOD-6 cuenta de 0 a 5).

La salida *tc* del contador MOD-6 en la sección SEGUNDOS tiene una frecuencia de 1 pulso por minuto (es decir, el MOD-6 se recicla cada 60 s). Esta señal se alimenta a la sección MINUTOS, la cual cuenta y visualiza los minutos del 0 al 59. La sección MINUTOS es idéntica a la sección SEGUNDOS y opera de la misma forma exacta.

La salida *tc* del contador MOD-6 en la sección MINUTOS tiene una frecuencia de 1 pulso por hora (es decir, el MOD-6 se recicla cada 60 min). Esta señal se alimenta a la sección HORAS, la cual cuenta y visualiza las horas de 1 a 12. Esta sección HORAS es distinta de las secciones MINUTOS y SEGUNDOS; ya que nunca llega al estado 0. Los circuitos en esta sección son bastante inusuales como para garantizar la necesidad de una investigación más detallada.

La figura 10-18 muestra el circuito detallado correspondiente a la sección HORAS. Esta sección incluye un contador BCD para contar unidades de horas y un solo FF (MOD-2) para contar decenas de horas. El contador BCD es un 74160, el cual tiene dos entradas activas en ALTO llamadas ENT y ENP, a las cuales se les aplica una operación AND en forma interna para habilitar el conteo. La entrada ENT también habilita la señal de acarreo activa en ALTO (RCO), la cual detecta el conteo terminal BCD (la transición de 9 a 0). En consecuencia, la entrada ENT y la salida RCO pueden utilizarse para la conexión en cascada de contadores síncronos.

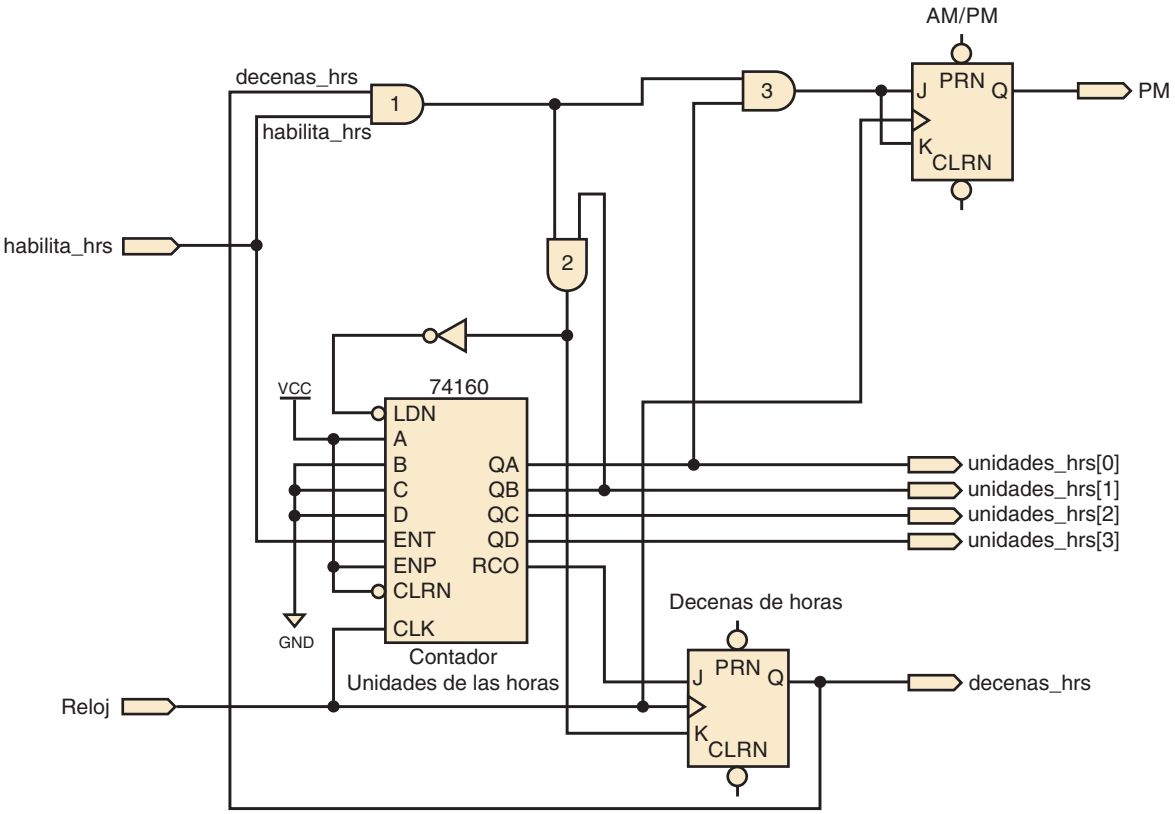


FIGURA 10-18 Circuito detallado para la sección HORAS.

La entrada ENP se conecta al nivel ALTO, de manera que el contador se incrementará cada vez que ENT se encuentre en ALTO.

El contador de horas se habilita para las etapas de los minutos y los segundos sólo durante un pulso de reloj cada hora. Cuando ocurre esta condición ENT se encuentra en ALTO, lo cual significa que las etapas de minutos:segundos se encuentran en 59:59. Por ejemplo, a las 9:59:59 el flip-flop de decenas de horas contiene un 0, el 74160 contiene 1001_2 (9) y la salida RCO se encuentra en ALTO, con lo cual el flip-flop de decenas de horas se coloca en el modo SET. Los dos dígitos de visualización para las horas muestran 09. En el siguiente flanco de subida del reloj, el contador BCD avanza a su SIGUIENTE estado natural de 0000_2, RCO cambia a BAJO y el flip-flop de decenas de horas avanza a 1, por lo que los dígitos de visualización de horas ahora muestran 10.

Cuando son las 11:59:59, la compuerta AND 1 detecta que el flip-flop de decenas de horas contiene 1 y la entrada de habilitación está activa (las etapas anteriores están en 59:59). La compuerta AND 3 combina las condiciones de la compuerta AND 1 y la condición de que el contador BCD se encuentra en el estado 0001_2. La salida de la compuerta AND 3 estará en ALTO sólo a las 11:59:59 en la secuencia de conteo de las horas. En el siguiente pulso de reloj el flip-flop AM/PM conmuta para indicar mediodía (ALTO) o medianoche (BAJO). Al mismo tiempo, el contador BCD avanza a 2 y las etapas de minutos:segundos se regresan a 00:00, para que la pantalla BCD indique las 12:00:00. A las 12:59:59 la compuerta AND 1 detecta que el dígito de las decenas es 1, por lo cual es tiempo de avanzar las horas. La compuerta AND 2 detecta que el contador BCD está en 2. La salida de la compuerta AND 2 se prepara para llevar a cabo dos tareas en el siguiente flanco de reloj: restablecer el flip-flop de decenas de horas y cargar el contador 74160 con el valor 0001_2. Después del siguiente pulso de reloj es la 01:00:00 en punto.

Ahora deberá tener sentido la operación de los circuitos contadores para usted; además deberá tener una buena idea acerca de cómo conectar chips MSI para for-

mar este reloj digital. Observe que en realidad está compuesto de varios circuitos pequeños y relativamente simples, los cuales se interconectan de manera estratégica para formar el reloj. Si recuerda, en el capítulo 4 mencionamos brevemente el concepto del diseño y desarrollo modular y jerárquico de sistemas digitales. Ahora es tiempo de aplicar estos principios a un proyecto que se encuentre dentro del alcance de su comprensión, para el cual puede utilizar el sistema de desarrollo MAX +PLUS II o Quartus II de Altera. Debe entender la operación de los circuitos que acabamos de describir antes de proceder con el diseño de este reloj mediante el uso de HDL. Tome su tiempo para repasar este material.

Diseño jerárquico de arriba-abajo

El diseño de arriba-abajo significa que queremos empezar en el nivel más alto de complejidad en la jerarquía, o que se considera que todo el proyecto completo existe en una caja cerrada y oscura con entradas y salidas. Aún no se conocen los detalles en relación con lo que hay en la caja. Por ahora sólo podemos decir cómo queremos que sea. Escogimos el reloj digital ya que todos estamos familiarizados con el resultado final de la operación de este dispositivo. Un aspecto importante de esta etapa del proceso de diseño es el establecimiento del alcance del proyecto. Por ejemplo, este reloj digital no va a tener una manera de establecer el tiempo, de establecer un tiempo de alarma, de apagar la alarma, de utilizar el modo de siesta (snooze) o de incorporar otras características que tal vez encuentre en el reloj que tiene al lado de su cama. Si agregáramos todas estas características ahora, sólo se saturaría el ejemplo con una complejidad innecesaria para nuestro fin inmediato. Tampoco vamos a incluir el condicionamiento de señales que transforma una onda senoidal de 60 Hz en una forma de onda digital de 60 pulsos por segundo, o los circuitos de decodificador/pantalla. El proyecto que vamos a ver contiene las siguientes especificaciones:

Entradas: forma de onda compatible con CMOS de 60 pps (su precisión depende de la frecuencia de línea)

Salidas: Horas BCD: 1 bit DECENAS 4 bits UNIDADES
Minutos BCD: 3 bits DECENAS 4 bits UNIDADES
Segundos BCD: 3 bits DECENAS 4 bits UNIDADES
Indicador PM

Secuencia de minutos y segundos: BCD MOD 60
00-59 (representación decimal de BCD)

Secuencia de horas BCD MOD 12
01-12 (representación decimal de BCD)

Intervalo total de visualización
01:00:00-12:59:59

El indicador AM/PM cambia a las 12:00:00

Una **jerarquía** es un grupo de objetos ordenados por su intervalo de magnitud, su importancia o su complejidad. La figura 10-19 muestra un diagrama de bloques del proyecto en general (nivel mayor de la jerarquía). Observe que hay cuatro bits para cada una de las salidas de las unidades BCD y sólo tres bits para cada una de

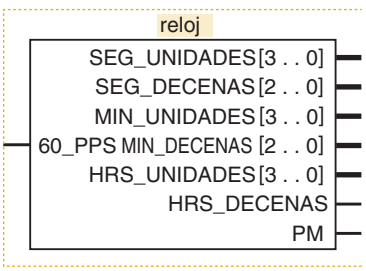


FIGURA 10-19 El bloque de nivel superior de la jerarquía.

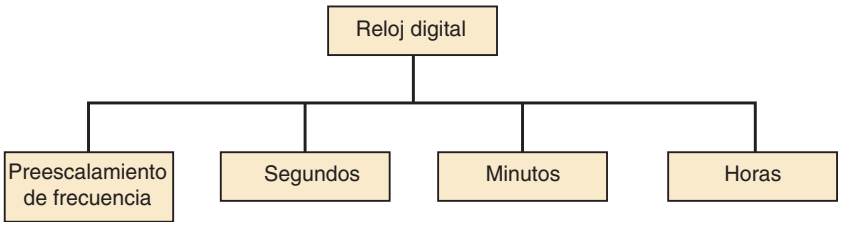


FIGURA 10-20 El nivel de secciones de la jerarquía.

las salidas de decenas BCD para los minutos y los segundos. Como el dígito BCD más significativo para la posición de las decenas es 5 (101_2), sólo se necesitan tres bits. Observe también que la posición de las decenas para las horas (HRS_DECENAS) sólo tiene un bit. Nunca tendrá un valor distinto de 0 o de 1.

La siguiente fase es descomponer el problema en secciones más simples. En primer lugar, necesitamos tomar la entrada de 60 pps y transformarla en una señal de sincronización de 1 pulso por segundo. A un circuito que divide una frecuencia de referencia en una velocidad requerida por el sistema se le llama **preescalador**. En segundo lugar, tiene sentido tener secciones individuales para un contador de segundos, un contador de minutos y un contador de horas. Hasta ahora el diagrama de jerarquía se ve como en la figura 10-20, la cual muestra el proyecto dividido en cuatro subsecciones.

La función de la sección del preescalador de frecuencia es dividir la entrada de 60 pps en una frecuencia de un pulso por segundo. Para ello se requiere un contador MOD-60, y la secuencia del conteo en realidad no importa. En este ejemplo, tanto la sección de minutos como la de segundos requieren contadores MOD-60 para contar de 00-59 en BCD. Es muy importante buscar similitudes como ésta en el proceso de diseño. En este caso, podemos utilizar exactamente el mismo diseño de circuito para implementar el preescalador de frecuencia, el contador de minutos y el contador de segundos.

Es fácil crear un contador BCD MOD-60 a partir de un contador MOD-10 (decenas) conectado en cascada a un contador BCD MOD-6, como vimos en el diagrama de la figura 10-17. Esto significa que dentro de cada uno de estos bloques MOD-10 encontraríamos un diagrama similar al de la figura 10-21. Ahora la jerarquía del proyecto aparece como se muestra en la figura 10-22.

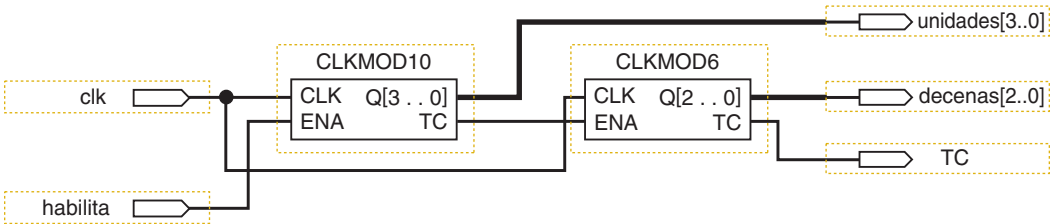


FIGURA 10-21 Los bloques dentro de la sección MOD-60.

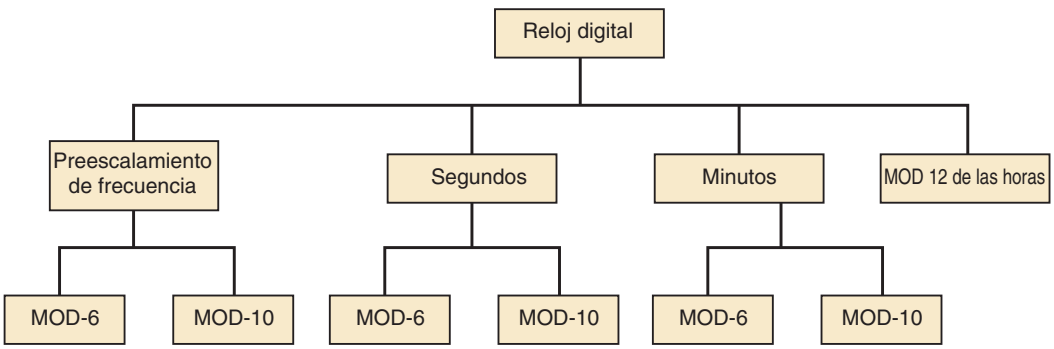


FIGURA 10-22 La jerarquía completa del proyecto del reloj.

La última decisión en el diseño es si se va a descomponer o no la sección MOD-12 para las Horas en dos etapas, como se muestra en la figura 10-18. Una opción es conectar las macrofunciones de estas piezas estándar de la biblioteca HDL, como hemos visto en capítulos anteriores. Dado que este circuito es algo inusual, hemos decidido mejor describir el contador MOD-12 de las horas mediante el uso de un módulo individual de HDL. También describiremos los bloques de construcción del MOD-6 y del MOD-10 mediante HDL. Así, todo el circuito de reloj completo puede crearse mediante estas tres descripciones de circuitos básicos. Desde luego que estos bloques también pueden descomponerse en bloques de flip-flops mucho más pequeños, y pueden diseñarse mediante el uso de un diagrama esquemático, pero será más sencillo usar HDL en este nivel.

Construcción de los bloques desde cero

Ahora presentaremos cada uno de los bloques básicos tanto en AHDL como en VHDL. Vamos a presentar el MOD-6 como una simple modificación de las descripciones del contador síncrono MOD-5 que presentamos en el capítulo 7 (vea las figuras 7-39 y 7-40). Después modificaremos aún más este código para crear el contador MOD-10 y por último diseñaremos el contador de Horas MOD-12 desde cero. Construiremos el reloj completo a partir de estos tres bloques básicos.

AHDL

CONTADOR MOD-6 EN AHDL

Las únicas características adicionales que requiere este diseño y que no se cubren en la figura 7-39 son la entrada *habilita* conteo y la salida de conteo terminal (*tc*) que se muestran en la figura 10-23. Observe que en la definición de E/S se incluyen la entrada (*habilita*, línea 3) y la salida (*tc*, línea 4) adicionales. Una nueva línea (línea 11) en la descripción de la arquitectura evalúa *habilita* antes de decidir cómo actualizar el valor de *conteo* (líneas 12 a15). Si *habilita* está en BAJO, *conteo* conten-

```
1   SUBDESIGN fig10 23
2   (
3      reloj, habilita      :INPUT;        sincroniza reloj y habilita.
4      q[2..0], tc          :OUTPUT;       3 contador de 3 bits
5   )
6   VARIABLE
7      conteo[2..0]   :DFF;      declara un registro de flip flops D.
8
9   BEGIN
10     conteo[].clk = reloj;        conecta todos los relojes a una fuente síncrona
11     IF habilita THEN
12        IF conteo[].q < 5 THEN
13           conteo[].d = conteo[].q + 1;     incrementa el valor actual en uno
14        ELSE conteo[].d = 0;           recicla, hace que los estados sin usar sean 0
15        END IF;
16     ELSE conteo[].d = conteo[].q;       no habilitado: se detiene en este conteo
17     END IF;
18     tc = habilita & conteo[].q == 5;    detecta conteo máximo si está deshabilitado
19     q[] = conteo[].q;                   conecta el registro a las salidas
20  END;
```

FIGURA 10-23 El diseño del MOD-6 en AHDL.

drá el mismo valor en cada flanco del reloj mediante la bifurcación ELSE (línea 16). Recuerde que siempre hay que relacionar una instrucción IF con una instrucción END IF, como hicimos en las líneas 15 y 17. La salida de conteo terminal (*tc*, línea 18) estará en ALTO cuando se cumpla que *conteo* = = 5 y (AND) *habilita* esté activa. Observe el uso de dobles signos de igual (= =) para evaluar la igualdad en AHDL.

CONTADOR MOD-6 EN VHDL

Las únicas características adicionales que necesita este diseño y que no se cubren en la figura 7-40 son la entrada *habilita* conteo y la salida de conteo terminal (*tc*) que se muestran en la figura 10-24. Observe que la entrada (*habilita*, línea 2) y la salida (*tc*, línea 4) adicionales se incluyen en la definición de E/S. Una nueva línea (línea 15) en la descripción de la arquitectura evalúa a *habilita* antes de decidir cómo actualizar el valor de *conteo* (líneas 16 a 20). En el caso en el que *habilita* está en BAJO, el valor actual se retiene en la variable *conteo* y el contador no sigue contando. Recuerde que siempre hay que relacionar una instrucción IF con una instrucción END IF, como hicimos en las líneas 20 a 22. El indicador de conteo terminal (*tc*, líneas 23 y 24) estará en ALTO cuando se cumpla que *conteo* = 5 y (AND) *habilita* esté activa.

```
1   ENTITY fig10 24 IS
2   PORT( reloj, habilita    :IN BIT ;
3         q                  :OUT INTEGER RANGE 0 TO 5;
4         tc                 :OUT BIT
5       );
6   END fig10 24;
7
8   ARCHITECTURE a OF fig10 24 IS
9   BEGIN
10     PROCESS (reloj)                          responde al reloj
11     VARIABLE conteo   :INTEGER RANGE 0 TO 5;
12
13     BEGIN
14        IF (reloj = '1' AND reloj'event) THEN
15           IF habilita = '1' THEN             entrada síncrona en cascada
16              IF conteo < 5 THEN              < conteo máx (terminal)?
17                 conteo := conteo + 1;
18              ELSE
19                 conteo := 0;
20              END IF;
21           END IF;
22        END IF;
23        IF (conteo = 5) AND (habilita = '1') THEN    entrada síncrona en cascada
24              tc <= '1';                             indica ct terminal
25        ELSE  tc <= '0';
26        END IF;
27        q <= conteo;                                 actualiza salidas
28     END PROCESS;
29  END a;
```

FIGURA 10-24 El diseño de un MOD-6 en VHDL.

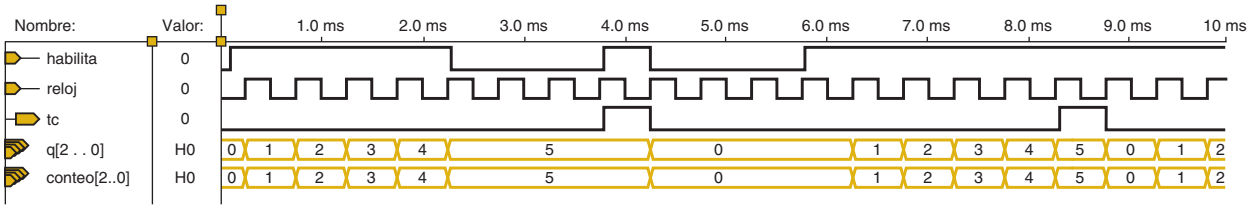


FIGURA 10-25 Simulación del contador MOD-6.

La prueba de la simulación del contador MOD-6 en la figura 10-25 verifica que cuente de 0 a 5 y que responda a la entrada de habilitación de conteo, ignorando los pulsos de reloj y deteniendo el contador cada vez que *habilita* está en BAJO. También genera la salida *tc* cuando se habilita a su conteo máximo de 5.

AHDL

CONTADOR MOD-10 EN AHDL

El contador MOD-10 varía sólo un poco en comparación con el contador MOD-6 que describimos en la figura 10-23. Los únicos cambios necesarios implican la modificación del número de bits en el puerto de salida y en el registro (en la sección VARIABLE), junto con el valor máximo al que debería llegar el contador antes de regresarse a cero. La figura 10-26 presenta el diseño de un MOD-10.

```
1   SUBDESIGN fig10_26
2   (
3      reloj, habilita      :INPUT;          sincroniza reloj y habilita.
4      q[3..0], tc          :OUTPUT;         contador de Decenas de 4 bits
5   )
6   VARIABLE
7      conteo[3..0] :DFF;        declara un registro de flip flops D.
8
9   BEGIN
10     conteo[].clk = reloj;     conecta todos los relojes a una fuente síncrona
11     IF habilita THEN
12        IF conteo[].q < 9 THEN
13           conteo[].d = conteo[].q + 1;    incrementa el valor actual en uno
14        ELSE conteo[].d = 0;    recicla, hace que los estados no usados sean 0
15        END IF;
16     ELSE conteo[].d = conteo[].q;    no habilitado: se detiene en este conteo
17     END IF;
18     tc = habilita & conteo[].q == 9;       detecta el conteo máximo
19     q[] = conteo[].q;                      conecta el registro a la salida
20  END;
```

FIGURA 10-26 Diseño de un MOD-10 en AHDL.

VHDL

CONTADOR MOD-10 EN VHDL

El contador MOD-10 varía sólo un poco en comparación con el contador MOD-6 que describimos en la figura 10-24. Los únicos cambios necesarios implican modificar el número de bits en el puerto de salida y la variable *conteo* (mediante el uso de INTEGER RANGE), junto con el valor máximo al que debe llegar el contador antes de regresar a cero. La figura 10-27 presenta el diseño del MOD-10.

```
1   ENTITY fig10 27 IS
2   PORT( reloj, habilita   :IN BIT ;
3         q                 :OUT INTEGER RANGE 0 TO 9;
4         tc                :OUT BIT
5       );
6   END fig10 27;
7
8   ARCHITECTURE a OF fig10 27 IS
9   BEGIN
10     PROCESS (reloj)                          responde al reloj
11     VARIABLE conteo   :INTEGER RANGE 0 TO 9;
12
13     BEGIN
14        IF (reloj = '1' AND reloj'event) THEN
15           IF habilita = '1' THEN            entrada síncrona en cascada
16              IF conteo < 9 THEN             contador de decenas
17                 conteo := conteo + 1;
18              ELSE
19                 conteo := 0;
20              END IF;
21           END IF;
22        END IF;
23        IF (conteo = 9) AND (habilita = '1') THEN     salida síncrona en cascada
24              tc <= '1';
25        ELSE  tc <= '0';
26        END IF;
27        q <= conteo;                              actualiza salidas
28     END PROCESS;
29  END a;
```

FIGURA 10-27 El diseño del MOD-10 EN VHDL.

Diseño del contador MOD 12

Ya hemos decidido que el contador de horas se va a implementar en un archivo de diseño mediante HDL. Debe ser un contador BCD MOD-12 que siga la secuencia de horas de un reloj (1-12) y que proporcione el indicador AM/PM. Si recuerda, en el paso de diseño inicial las salidas BCD necesitan ser un arreglo de cuatro bits para el dígito de menor orden y de un solo bit para el dígito de mayor orden. Para diseñar este circuito contador, considere la forma en que necesita operar. Su secuencia es:

01 02 03 04 05 06 07 08 09 10 11 12 01 …

Si observamos esta secuencia podemos concluir que hay cuatro áreas críticas que definen las operaciones necesarias para producir el estado SIGUIENTE que sea apropiado:

1. Cuando el valor es de 01 a 08, se incrementa el dígito inferior y el dígito superior se mantiene igual.
2. Cuando el valor es 09, el dígito inferior se restablece a 0 y el dígito superior se hace 1.
3. Cuando el valor es 10 o 11, se incrementa el dígito inferior y el dígito superior se mantiene igual.
4. Cuando el valor es 12, el dígito inferior se restablece a 1 y el dígito superior a 0.

Como estas condiciones necesitan evaluar un intervalo de valores, es más apropiado utilizar una instrucción IF/ELSIF en vez de una instrucción CASE. También existe

la necesidad de identificar cuando es tiempo de cambiar el indicador AM/PM. Este tiempo ocurre cuando el estado de la hora es 11 y la señal de habilitación está en ALTO, lo cual significa que los contadores de menor orden se encuentran en su valor máximo (59-59).

AHDL

CONTADOR MOD-12 EN AHDL

El contador en AHDL necesita un banco de cuatro flip-flops para el dígito BCD de menor orden y sólo un flip-flop D para el dígito BCD de mayor orden, ya que este valor siempre será 0 o 1. También se necesita un flip-flop para llevar el registro de A.M. y P.M. Estas primitivas se declaran en las líneas 7 a 9 de la figura 10-28. Observe además que en este diseño se utilizan los mismos nombres para los puertos de salida. Ésta es una característica conveniente de AHDL. Cuando la entrada habilita (*ena*) está activa, el circuito evalúa las instrucciones IF/ELSE de las líneas 16 a

```
1   SUBDESIGN fig10 28
2   (
3      clk, ena           :INPUT;
4      inf[3..0], sup, pm :OUTPUT;
5   )
6   VARIABLE
7      inf[3..0]    :DFF;
8      sup          :DFF;
9      am pm        :JKFF;
10     tiempo       :NODE;
11  BEGIN
12     inf[].clk = clk;           pulsos de reloj síncronos
13     sup.clk = clk;
14     am pm.clk = clk;
15     IF ena THEN                utiliza habilitar para contar
16         IF inf[].q < 9 & alto.q == 0   THEN
17           inf[].d = bajo[].q + 1;   inc dígito inf
18           sup.d = sup.q;      retiene dígito sup
19        ELSIF inf[].q == 9 THEN
20           inf[].d = 0;
21           sup.d = VCC;
22        ELSIF sup.q == 1 & inf[].q < 2 THEN
23                inf[].d = inf[].q + 1;
24                sup.d = sup.q;
25        ELSIF sup.q == 1 & inf[].q == 2 THEN
26                inf[].d = 1;
27                sup.d = GND;
28        END IF;
29     ELSE
30        inf[].d = inf[].q;
31        sup.d = sup.q;
32     END IF;
33     tiempo = sup.q == 1 & inf[3..0].q == 1 & ena;      detecta 11:59:59
34     am pm.j = tiempo;          alterna am/pm a mediodía y a medianoche
35     am pm.k = tiempo;
36     pm = am pm.q;
37  END;
```

FIGURA 10-28 El contador de horas MOD-12 en AHDL.

28 y realiza la operación apropiada sobre los nibbles superior e inferior del número BCD. Cada vez que la entrada de habilitación está en BAJO el valor permanece igual, como se muestra en las líneas 30 y 31. La línea 33 detecta cuando el conteo llega a 11, mientras el contador está habilitado. Esta señal se aplica a las entradas *J* y *K* del flip-flop am_pm para hacer que cambie a las 11:59:59.

CONTADOR MOD-12 EN VHDL

El contador de VHDL de la figura 10-29 necesita una salida de cuatro bits para el dígito BCD de menor orden y un solo bit de salida para el dígito BCD de mayor

```
1  ENTITY fig10 29 IS
2  PORT( clk, ena       :IN BIT ;
3        inf            :OUT INTEGER RANGE 0 TO 9;
4        sup            :OUT INTEGER RANGE 0 TO 1;
5        pm             :OUT BIT            );
6  END fig10 29;
7
8  ARCHITECTURE a OF fig10 29 IS
9  BEGIN
10    PROCESS (clk)                              responde al reloj
11    VARIABLE am pm :BIT;
12    VARIABLE unidades  :INTEGER RANGE 0 TO 9;    señal de unidades de 4 bits
13    VARIABLE decenas  :INTEGER RANGE 0 TO 1;     señal de decenas de 1 bit
14    BEGIN
15       IF (clk = '1' AND clk'EVENT) THEN
16          IF ena = '1' THEN                     entrada síncrona en cascada
17             IF (unidades = 1) AND (decenas = 1) THEN   -- a las 11:59:59
18                am_pm := NOT am_pm;               -- cambia am/pm
19             END IF;
20             IF (unidades < 9) AND  (decenas = 0) THEN     estados 00 08
21                unidades := ones + 1;           incrementa unidades
22             ELSIF unidades = 9 THEN            estado 09… cambiar a 10:00
23                unidades := 0;                  restablecer unidades a cero
24                decenas := 1;                   decenas cambia a 1
25             ELSIF (decenas = 1) AND (unidades < 2) THEN     estados 10, 11
26                unidades := unidades + 1;             incrementa unidades
27             ELSIF (decenas = 1) AND (unidades = 2) THEN    estado 12
28                unidades := 1;                    establece en 01:00
29                decenas := 0;
30             END IF;
31
32       Este espacio es la ubicación alternativa para actualizar am/pm
33
34          END IF;
35       END IF;
36       pm <= am pm;
37       inf <= unidades;                                actualiza salidas
38       sup <= decenas;
39    END PROCESS;
40 END a;
```

FIGURA 10-29 El contador de horas MOD-12 en VHDL.

orden, ya que su valor siempre será 0 o 1. Estas salidas (líneas 3 y 4), junto con las variables que producirán las salidas (líneas 12 y 13), se declaran como enteros debido a que el "conteo" es posible con sólo sumar 1 al valor de la variable. En cada flanco activo del reloj, cuando la entrada de habilitación está activa, el circuito necesita decidir qué hacer con el contador BCD de unidades de horas, con el flip-flop de un solo bit para las decenas de horas y también con el flip-flop AM/PM.

Este ejemplo es una excelente oportunidad para recalcar algunas de las características avanzadas de VHDL, las cuales permiten al diseñador describir con precisión la operación del circuito de hardware final. En capítulos anteriores hablamos sobre la cuestión de las instrucciones dentro de un bloque PROCESS que se evaluaban en forma secuencial. Recuerde que las instrucciones fuera del bloque PROCESS se consideran concurrentes y el orden en el que se escriben en el archivo de diseño no tiene efecto sobre la operación del circuito final. En este ejemplo debemos evaluar el estado actual para decidir si debemos cambiar el indicador AM/PM y avanzar el contador al estado SIGUIENTE. Algunas de las cuestiones que se involucran aquí son:

1. ¿Cómo "recordamos" el valor de conteo actual en VHDL?
2. ¿Evaluamos el conteo actual para ver si es 11 (y determinar si necesitamos cambiar de estado el flip-flop AM/PM) y después incrementarlo a 12, o incrementamos el estado del contador de 11 a 12 y después evaluamos el conteo para ver si es 12 (para saber si necesitamos cambiar el estado del flip-flop AM/PM)?

En relación con la primera cuestión, hay dos formas de recordar el estado actual de un contador en VHDL. Tanto las señales (SIGNAL) como las variables (VARIABLE) retienen su valor hasta que se actualizan. Por lo general, las señales se utilizan para conectar nodos en el circuito como alambres, y las variables se utilizan como un registro para almacenar datos entre actualizaciones. En consecuencia, las variables se utilizan, por lo general, para implementar contadores. Las principales diferencias son que las variables son locales para el bloque PROCESS en el cual se declaran y las señales son globales. Además, las variables se actualizan en forma inmediata dentro de la secuencia de instrucciones en un bloque PROCESS, mientras que las señales a las que se hace referencia dentro de un bloque PROCESS se actualizan cuando éste se suspende. En este ejemplo hemos optado por utilizar variables, las cuales son locales para el bloque PROCESS que describe lo que debe ocurrir cuando llega el flanco activo del reloj.

En cuanto a la segunda pregunta, cualquiera de estas estrategias funcionará pero ¿cómo lo describimos mediante el uso de VHDL? Si queremos que el circuito cambie entre A.M. y P.M. al detectar un 11 antes de que se actualice el contador (como una conexión síncrona en cascada), entonces la evaluación debe ocurrir en el código antes de actualizar las variables. Esta evaluación se demuestra en el archivo de diseño de la figura 10-29, en las líneas 17 a 19. Por otro lado, si queremos que el circuito cambie entre A.M. y P.M. cuando detecte que haya llegado la hora 12 después del flanco de reloj (algo más parecido a una conexión en cascada), entonces hay que actualizar las variables antes de evaluar el valor 12. Para modificar el diseño en la figura 10-29 para realizar esta tarea, podemos mover la instrucción IF de las líneas 17 a 19 hacia el área en blanco de las líneas 31 a 33 y editarla como se muestra a continuación, en negritas:

```
31    IF (unidades = 2) AND (decenas = 1)  THEN     a las 12:00:00
32    am pm := NOT am pm;                        cambiar entre am/pm
33    END IF;
```

El orden de las instrucciones y el valor que se decodifica marca toda la diferencia en cuanto a la forma en que opera el circuito. En las líneas 36 a 38, la VARIABLE *am_pm* se conecta al puerto *pm*, el dígito BCD de las unidades se aplica a los cuatro bits inferiores de la salida (*inf*) y el dígito de las decenas (una variable de un solo bit) se aplica al dígito más significativo (*sup*) del puerto de salida. Como todas estas

variables son locales, estas instrucciones deben ejecutarse antes de la instrucción END PROCESS en la línea 39.

Una vez compilado el diseño, debe simularse para verificar su operación, en especial en las áreas críticas. La figura 10-30 muestra un ejemplo de una simulación para probar este contador. En el lado izquierdo del diagrama de sincronización el contador está deshabilitado y retiene la hora 11 debido a que el dígito *sup* está en 1 y el dígito *inf[ ]* está en 1. En el flanco de subida del reloj, después de que la señal de habilitación cambia a ALTO, la hora cambia de 11 a 12 y hace que el indicador PM cambie a ALTO, lo que significa que es mediodía. El siguiente flanco activo hace que el conteo regrese de 12 a 01. En la mitad derecha de la sincronización se simula la misma secuencia, con lo cual se demuestra que en realidad habría muchos pulsos de reloj entre los tiempos en que se incrementa la hora. En el ciclo de reloj antes del incremento, la señal de habilitación se lleva a ALTO mediante el conteo terminal de la etapa anterior.

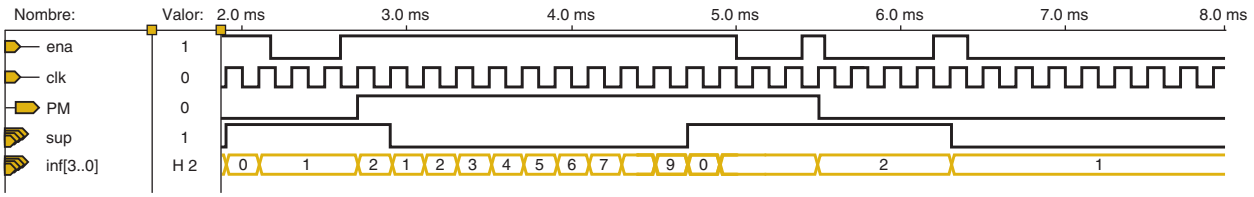


FIGURA 10-30 Simulación del contador de horas MOD-12.

Combinación de bloques en forma gráfica

Ya hemos definido, creado y simulado en forma individual los bloques de construcción del proyecto para verificar que funcionen sin problemas. Ahora es tiempo de combinar los bloques para formar secciones y combinarlas para formar el producto final. El software de Altera ofrece varias formas de lograr la integración de todas las piezas de un proyecto. En el capítulo 4 mencionamos que todos los distintos tipos de archivos de diseño (AHDL, VHDL, VERILOG, Esquemático) pueden combinarse en forma gráfica. Esta técnica se hace posible gracias a una característica que nos permite crear un "símbolo" para representar un archivo de diseño específico. Por ejemplo, el archivo de diseño del contador MOD-6 que se escribió en el archivo de diseño de VHDL fig10_24 puede representarse en el software como el bloque del circuito, como se muestra en la figura 10-31(a). El software MAX+PLUS II o Quartus II crea este símbolo con el clic de un botón. A partir de ese punto, reconocerá que ese símbolo opera de acuerdo con el diseño especificado en el código de HDL. El símbolo de la figura 10-31(b) se creó a partir del archivo de AHDL para el contador MOD-10 de la figura 10-26, y el símbolo de la figura 10-31(c) se creó a partir del archivo de VHDL para el contador MOD-12 de la figura 10-29. (La razón por la que estos bloques se identifican según el número de figura es sólo para facilitar la ubicación de los archivos de diseño en el CD que se incluye en este libro. En un ambiente de diseño [y no en un libro de texto] deberán nombrarse de acuerdo con su propósito, con nombres tales como MOD6, MOD10 y RELOJ_HORAS.)

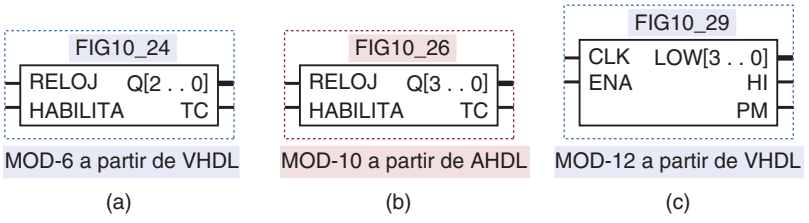


FIGURA 10-31 Símbolos de bloque gráficos que se generan a partir de archivos de diseño de HDL: (a) MOD-6 a partir de VHDL; (b) MOD-10 a partir de AHDL; (c) MOD-12 a partir de VHDL.

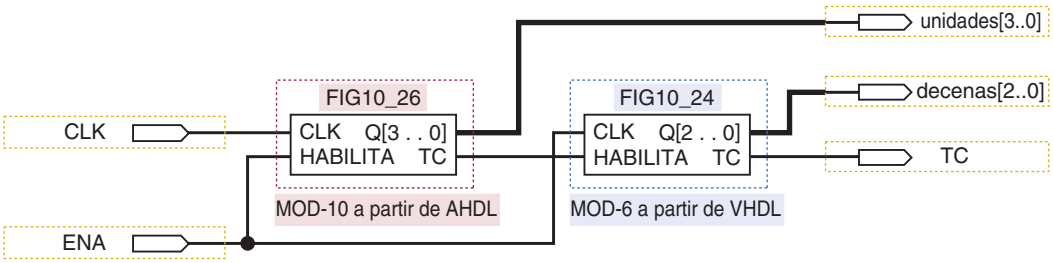


FIGURA 10-32 Combinación gráfica de bloques de HDL para formar un MOD-60.

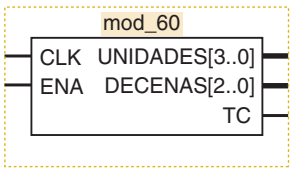


FIGURA 10-33 El contador MOD-60.

Si seguimos la jerarquía de diseño que acabamos de establecer, el siguiente paso sería combinar los contadores MOD-6 y MOD-10 para formar un bloque MOD-60. El software MAX+PLUS II utiliza archivos de diseño gráfico (.gdf) para integrar los símbolos de bloque mediante el dibujo de líneas que conectan los puertos de entrada, los símbolos y los puertos de salida. El software Quartus II proporciona la misma característica, pero utiliza archivos de diseño de bloque (.bdf). El resultado se muestra en la figura 10-32, la cual representa a un archivo GDF en MAX+PLUS II o a un archivo BDF en Quartus II. Puede compilar y utilizar este archivo de diseño gráfico o de bloque para simular la operación del contador MOD-60. Después de verificar que el diseño funciona en forma apropiada, el sistema MAX+PLUS II o Quartus II nos permitirá tomar este circuito y crear un símbolo de bloque para el mismo, como se muestra en la figura 10-33.

El símbolo del MOD-60 puede usarse en forma repetitiva junto con el símbolo MOD-12 para crear el diagrama de símbolos de bloque a nivel de sistema que se muestra en la figura 10-34. Incluso este diagrama a nivel de sistema puede representarse mediante un símbolo de bloque para todo el proyecto, como se muestra en la figura 10-35.

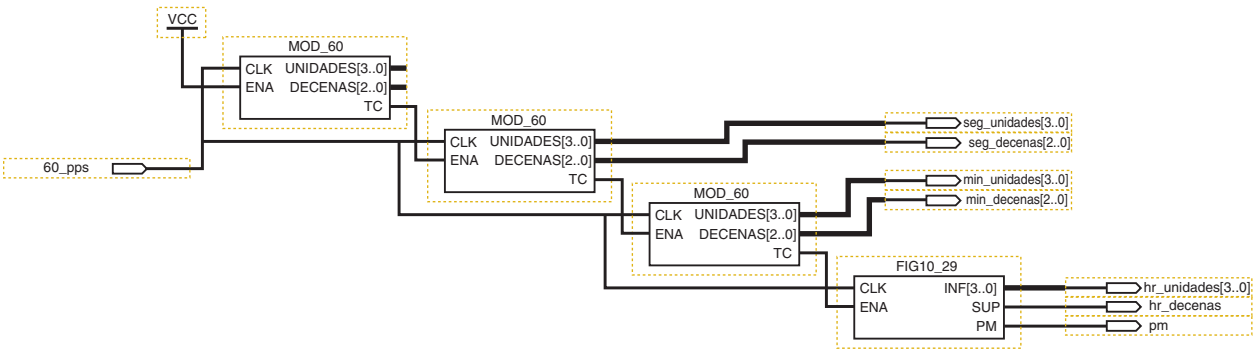
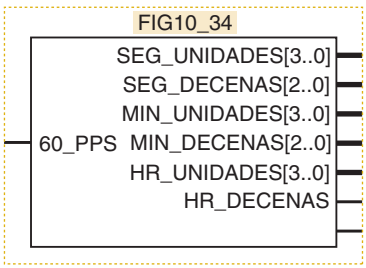


FIGURA 10-34 El proyecto de reloj completo, conectado mediante el uso de símbolos de bloque.

FIGURA 10-35 El reloj completo, representado mediante un solo símbolo.



Combinación de bloques utilizando sólo HDL

El método gráfico funciona bien, siempre y cuando esté disponible y sea adecuado para un determinado propósito. Como dijimos antes, el HDL se desarrolló para proporcionar un método eficiente de documentar sistemas complejos y almacenar

la información de una manera menos temporal e independiente del software. Es razonable suponer que con AHDL, la opción de la integración gráfica de los subdiseños siempre estará disponible con las herramientas de Altera; no obstante, esta suposición no es razonable para los usuarios de VHDL. Muchos sistemas de desarrollo de VHDL no ofrecen ningún equivalente para la integración de bloques gráficos de Altera, lo cual explica el por qué es importante tratar el mismo concepto de desarrollo modular, jerárquico y la integración de proyectos utilizando sólo herramientas de lenguaje basadas en texto. Nuestra cobertura de la integración en AHDL no será tan detallada como nuestra cobertura de VHDL, debido a que, por lo general, se prefiere el método gráfico.

INTEGRACIÓN DE MÓDULOS EN AHDL

Regresemos a los dos archivos de AHDL para los contadores MOD-6 y MOD-10. ¿Cómo combinamos estos archivos en un contador MOD-60 utilizando sólo AHDL basado en texto? En realidad el método es muy similar al de la integración gráfica. En vez de crear una representación "simbólica" de los archivos del MOD-6 y del MOD-10, se crea un nuevo tipo de archivo conocido como "archivo de inclusión (INCLUDE)", el cual contiene toda la información importante acerca del archivo AHDL que representa. Para describir un contador MOD-60 se abre un nuevo archivo TDF, como el que se muestra en la figura 10-36. Los bloques de construcción se "incluyen" en la parte superior, como se muestra en las líneas 1 y 2. A continuación, los nombres que se utilizaban para los bloques de construcción se utilizan como componentes o primitivas de biblioteca para definir la naturaleza de una variable. En la línea 10, la variable *mod10* se utiliza ahora para representar al contador MOD-10 en el otro módulo (fig10_26). *MOD10* tiene ahora todos los atributos (entradas, salidas, operación funcional) descritos en fig10_26.tdf. De igual forma, en la línea 10 la variable *mod6* recibe los atributos del contador MOD-6 de fig10_23.tdf. Las líneas 13 a 19 realizan la misma tarea que al dibujar líneas en el archivo GDF o BDF para conectar los componentes entre sí y con los puertos de entrada/salida.

```
1   INCLUDE "fig10 26.inc";          módulo del contador mod 10
2   INCLUDE "fig10 23.inc";          módulo del contador mod 6
3
4   SUBDESIGN fig10 36
5   (
6      clk, ena                                :INPUT;
7      unidades[3..0], decenas[2..0], tc       :OUTPUT;
8   )
9   VARIABLE
10     mod10          :fig10 26;          mod 10 para las unidades
11     mod6           :fig10 23;          mod 6 para las decenas
12  BEGIN
13     mod10.reloj = clk;                 pulsos de reloj síncronos
14     mod6.reloj = clk;
15     mod10.habilita = ena;
16     mod6.habilita = mod10.tc;            cascada
17     unidades[3..0] = mod10.q[3..0];      1s
18     decenas[2..0] = mod6.q[2..0];        10s
19     tc = mod6.tc;                        Hace conteo terminal en 59
20  END;
```

FIGURA 10-36 El MOD-60 formado a partir de MOD-10 y MOD-6 en AHDL.

El compilador puede traducir este archivo (FIG10_36.TDF) en uno de "inclusión" (fig10_36.inc), para luego utilizarlo en otro archivo tdf que describa la interconexión de las secciones principales que conforman el sistema. Cada nivel de la jerarquía hace referencia a los módulos que constituyen los niveles inferiores.

INTEGRACIÓN DE MÓDULOS EN VHDL

Regresemos a los dos archivos de VHDL para los contadores MOD-6 y MOD-10, los cuales se muestran en las figuras 10-24 y 10-27 respectivamente. ¿Cómo combinamos estos archivos en un contador MOD-60, utilizando sólo VHDL basado en texto? En realidad el método es muy similar al de la integración gráfica. En vez de crear una representación "simbólica" de los archivos MOD-6 y MOD-10, estos archivos de diseño se describen como un componente (COMPONENT), como vimos en el capítulo 5. El componente contiene toda la información importante acerca del archivo de VHDL que representa. Para describir un contador MOD-60 se abre un nuevo archivo de VHDL, el cual se muestra en la figura 10-37. Los archivos de los bloques de construcción se describen como "componentes", como muestran las líneas 10 a 14 y en las líneas 15 a 19 en la descripción de la arquitectura. A continuación,

```
1   ENTITY fig10 37 IS
2   PORT( clk, ena           :IN BIT ;
3         decenas            :OUT INTEGER RANGE 0 TO 5;
4         unidades           :OUT INTEGER RANGE 0 TO 9;
5         tc                 :OUT BIT            );
6   END fig10 37;
7
8   ARCHITECTURE a OF fig10 37 IS
9   SIGNAL alambre cascada  :BIT;
10  COMPONENT fig10 24                              módulo MOD 6
11  PORT( reloj, habilita  :IN BIT ;
12        q                :OUT INTEGER RANGE 0 TO 5;
13        tc               :OUT BIT);
14  END COMPONENT;
15  COMPONENT fig10 27                              módulo MOD 10
16  PORT( reloj, habilita  :IN BIT ;
17        q                :OUT INTEGER RANGE 0 TO 9;
18        tc               :OUT BIT);
19  END COMPONENT;
20
21  BEGIN
22     mod10:fig10 27
23        PORT MAP(   reloj => clk,
24                    habilita => ena,
25                    q  => unidades,
26                    tc => alambre cascada);
27
28     mod6:fig10 24
29        PORT MAP(   reloj => clk,
30                    habilita => alambre cascada,
31                    q  => decenas,
32                    tc => tc);
33  END a;
```

FIGURA 10-37 El MOD-60 formado a partir de MOD-10 y MOD-6 en VHDL.

```
ENTITY fig10 38 IS
PORT( pps 60                                        :IN BIT ;
      hora decenas                                  :OUT INTEGER RANGE 0 TO 1;
      hora unidades, min unidades, seg unidades     :OUT INTEGER RANGE 0 TO 9;
      min decenas, seg decenas                      :OUT INTEGER RANGE 0 to 5;
      pm                                            :OUT BIT               );
END fig10 38;

ARCHITECTURE a OF fig10 38 IS
SIGNAL alambre cascada1, alambre cascada2, alambre cascada3    :BIT;
SIGNAL habilitado                                              :BIT;
COMPONENT fig10 37      MOD 60
PORT( clk, ena       :IN BIT ;
      decenas        :OUT INTEGER RANGE 0 TO 5;
      unidades       :OUT INTEGER RANGE 0 TO 9;
      tc             :OUT BIT          );
END COMPONENT;
COMPONENT fig10 29      MOD 12
PORT( clk, ena       :IN BIT ;
      inf            :OUT INTEGER RANGE 0 TO 9;
      sup            :OUT INTEGER RANGE 0 TO 1;
      pm             :OUT BIT          );
END COMPONENT;
BEGIN
   habilitado <= '1';

   preescala:fig10 37              preescalador MOD 60
      PORT MAP(   clk      =>   pps 60,
                  ena      =>   habilitado,
                  tc       =>   alambre cascada1);

   segundo:fig10 37                contador de segundos MOD 60
      PORT MAP(   clk      =>   pps 60,
                  ena      =>   alambre cascada1,
                  nidades  =>   seg unidades,
                  decenas  =>   seg decenas,
                  tc       =>   alambre cascada2);

   minuto:fig10 37                 contador de minutos MOD 60
      PORT MAP(   clk      =>   pps 60,
                  ena      =>   alambre cascada2,
                  unidades =>   min unidades,
                  tens     =>   min decenas,
                  tc       =>   alambre cascada3);

   hora:fig10 29                   contador de horas MOD 12
      PORT MAP(   clk      =>   pps 60,
                  ena      =>   alambre cascada3,
                  inf      =>   hora unidades,
                  sup      =>   hora decenas,
                  pm       =>   pm);
END a;
```

FIGURA 10-38 El reloj completo en VHDL.

los nombres que se utilizaban para los bloques de construcción (componentes) se utilizan junto con las palabras clave PORT MAP para describir la interconexión de estos componentes. La información en las secciones PORT MAP describe las mismas operaciones exactas que el dibujar alambres en un diagrama esquemático en un archivo GDF o BDF.

Por último, el archivo VHDL que representa el bloque en la parte superior de la jerarquía se crea mediante el uso de componentes de la figura 10-37 (MOD-60) y de la figura 10-29 (MOD-12). Este archivo se muestra en la figura 10-38. Observe que el formato general es el siguiente:

Definir E/S: líneas 1 a 7.

Definir señales: líneas 10 a 11.

Definir componentes: líneas 12 a 23.

Instanciar componentes y conectarlos entre sí: líneas 27 a 52.

PREGUNTAS DE REPASO

1. ¿Qué se define en el nivel superior de un diseño jerárquico?
2. ¿En dónde comienza el proceso de diseño?
3. ¿En dónde comienza el proceso de construcción?
4. ¿En qué etapa(s) debe realizarse la prueba de simulación?

10-5 PROYECTO DE CONTADOR DE FRECUENCIA

El proyecto en esta sección demuestra el uso de los contadores y otras funciones lógicas estándar para implementar un sistema denominado como contador de frecuencia, similar al instrumento de prueba que tal vez haya utilizado en el laboratorio. Describiremos la teoría de operación en términos de dispositivos lógicos MSI convencionales y después la relacionaremos con los bloques de construcción que pueden desarrollarse mediante el uso de HDL. Al igual que con la mayoría de los proyectos, este ejemplo consiste de varios circuitos que ya hemos estudiado en capítulos anteriores. Aquí combinamos estos circuitos para formar un sistema digital con un propósito único. Primero definamos lo que es un contador de frecuencia.

Un **contador de frecuencia** es un circuito que puede medir y visualizar la frecuencia de una señal. Como sabe, la frecuencia de una forma de onda periódica es en sí el número de ciclos por segundo. La acción de formar cada ciclo de la frecuencia desconocida en un pulso digital nos permite utilizar un circuito digital para contar los ciclos. La idea general detrás de la medición de la frecuencia implica la acción de habilitar un contador para que cuente el número de ciclos (pulsos) de la forma de onda entrante durante un periodo de tiempo especificado con precisión, al cual se le conoce como **intervalo de muestreo**. La longitud del intervalo de muestreo determina el intervalo de frecuencias que pueden medirse. Un intervalo más largo proporciona una mejor precisión para las bajas frecuencias, pero hará que el contador se desborde en las frecuencias altas. Un intervalo de muestreo más corto proporciona una medición menos precisa de las frecuencias bajas, pero puede medir una frecuencia máxima mucho más alta sin exceder el límite superior del contador.

EJEMPLO 10-1

Suponga que un contador de frecuencia utiliza un contador BCD de cuatro dígitos. Determine la frecuencia máxima que se puede medir utilizando cada uno de los siguientes intervalos de muestreo:

(a) 1 segundo (b) 0.1 segundo (c) 0.01 segundo

Solución

(a) Con un intervalo de muestreo de 1 segundo, el contador de cuatro dígitos puede contar hasta 9999 pulsos. La frecuencia es de 9999 pulsos por segundo, o 9.999 kHz.

(b) El contador puede contar hasta 9999 pulsos dentro del intervalo de muestreo de 0.1 segundo. Esto se traduce en una frecuencia de 99,990 pulsos por segundo, o 99.99 kHz.

(c) El contador puede contar hasta 9999 pulsos dentro del intervalo de muestreo de 0.01 segundo. Esto se traduce en una frecuencia de 999,900 pulsos por segundo, o 999.9 kHz.

EJEMPLO 10-2

Si se aplica una frecuencia de 3792 pps a la entrada de un contador de frecuencia, ¿qué lectura indicará el contador bajo cada uno de los siguientes intervalos de muestreo?

(a) 1 segundo (b) 0.1 segundo (c) 10 ms

Solución

(a) Durante un intervalo de muestreo de 1 segundo, el contador contará 3792 ciclos. La frecuencia será de 3.792 kpps.

(b) Durante un intervalo de muestreo de 0.1 segundo, el número de pulsos que se contarán será de 379 o 380 ciclos, dependiendo de la posición en la que comience el intervalo de muestreo. La frecuencia será de 03.79 kpps o 03.80 kpps.

(c) Durante un intervalo de muestreo de 0.01 segundo, el número de pulsos que se contará será de 37 o 38 ciclos, dependiendo de la posición en la que comience el intervalo de muestreo. La frecuencia será de 003.7 kpps o 003.8 kpps.

La figura 10-39 muestra uno de los métodos más simples para construir un contador de frecuencia en forma de diagrama de bloques. Los bloques principales son el contador, el registro de visualización, el decodificador/pantalla y la unidad de sincronización y de control. El bloque del contador contiene varios contadores BCD en cascada, los cuales se utilizan para contar el número de pulsos producidos por la señal desconocida que se aplica a la entrada del reloj. El bloque del contador tiene controles de habilitación de conteo y de borrado. El periodo de tiempo para el conteo (intervalo de muestreo) se controla mediante una señal de habilitación

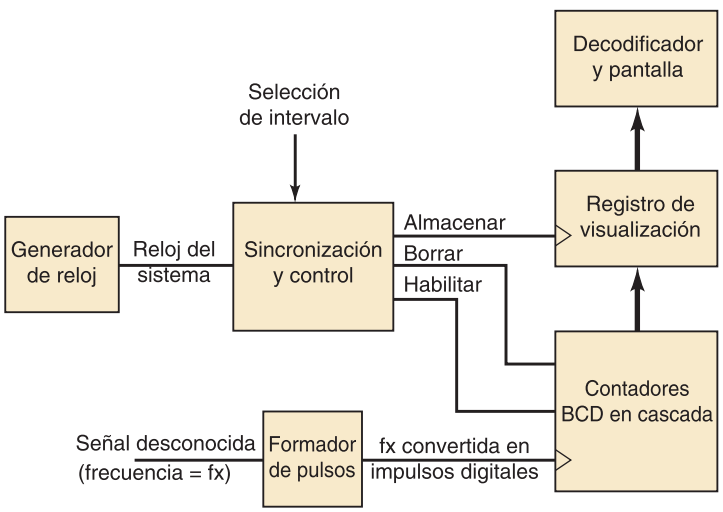


FIGURA 10-39
Diagrama de bloques de un contador de frecuencia básico.

producida por el bloque de sincronización y control. La longitud de tiempo para que los contadores BCD se habiliten se puede seleccionar con la entrada de selección de intervalo para el bloque de sincronización y control. Esto permite al usuario seleccionar el intervalo de frecuencia deseada a medir y determina en efecto la posición del punto decimal en la lectura digital. La anchura del pulso de la señal de habilitación (intervalo de muestreo) es imprescindible para tomar una medición de frecuencia precisa. El contador debe borrarse antes de habilitarlo para una nueva medición de frecuencia de la señal desconocida. Después de tomar un nuevo conteo se deshabilita el contador y se almacena la medición de frecuencia más reciente en el registro de visualización. La salida de este registro es una entrada para el bloque del decodificador y pantalla, en donde los valores BCD se convierten en decimal para la lectura digital. El uso de un registro de visualización separado permite al contador de frecuencia tomar una nueva medición en segundo plano, de manera que el usuario no observe el contador mientras éste calcula el total del número de pulsos para una nueva lectura. En vez de ello, la pantalla se actualiza en forma periódica con la última lectura de frecuencia.

La precisión de este contador de frecuencia depende casi en su totalidad de la precisión de la frecuencia del reloj del sistema, la cual se utiliza para crear el ancho de pulso apropiado para la señal de habilitación del contador. En la figura 10-39 se utiliza un generador de reloj controlado por cristal para producir un reloj de sistema preciso para el bloque de sincronización y control.

Se necesita un bloque "formador de pulsos" para asegurar que la señal de forma desconocida cuya frecuencia se va a medir sea compatible con la entrada de reloj para el bloque del contador. Puede utilizarse un circuito disparador de Schmitt para convertir las formas de onda "no cuadradas" (senoidal, triangular, etc.) siempre y cuando la señal de entrada desconocida esté en un intervalo de amplitud compatible. Si la señal desconocida pudiera tener una amplitud más grande o más pequeña de lo que se considera como compatible con un disparador de Schmitt, entonces se requerirán circuitos adicionales de acondicionamiento de señales analógicas, como el control de ganancia automático, para el bloque del formador de pulsos.

En la figura 10-40 se muestra el diagrama de sincronización para el control del contador de frecuencia. El reloj de control se deriva de la señal de reloj del sistema mediante unos divisores de frecuencia contenidos en el bloque de sincronización y control. El periodo de la señal de reloj de control se utiliza para crear la anchura de pulso de habilitación deseada. Un contador de control reciclable dentro del bloque de sincronización y control se sincroniza mediante la señal del reloj de control. Tiene los estados seleccionados decodificados para producir la secuencia de señal

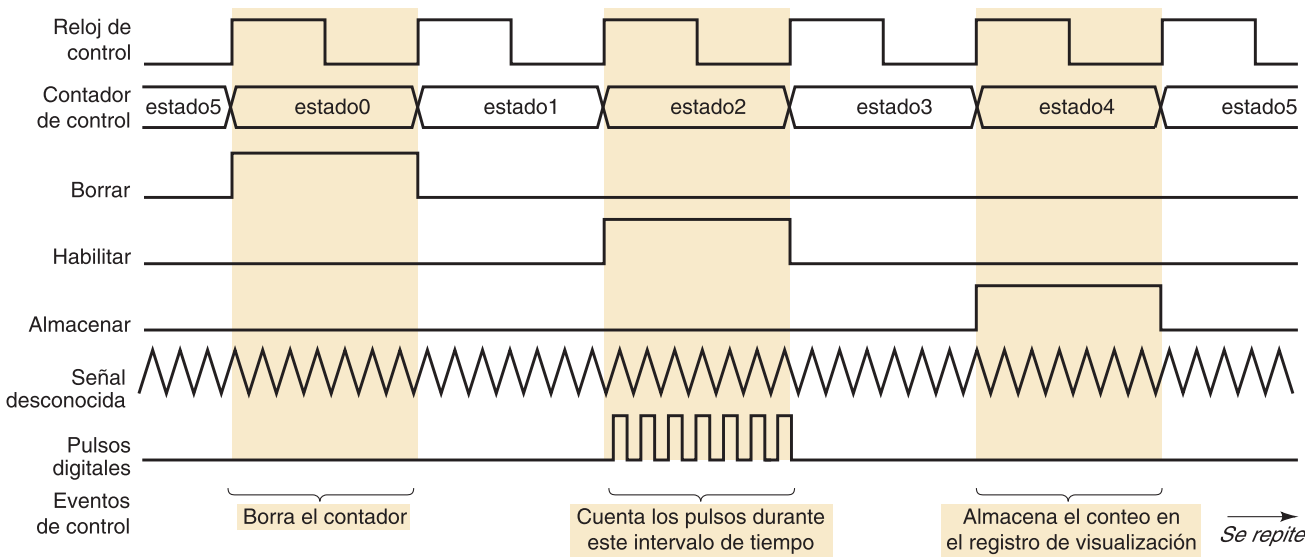


FIGURA 10-40 Diagrama de sincronización del contador de frecuencia.

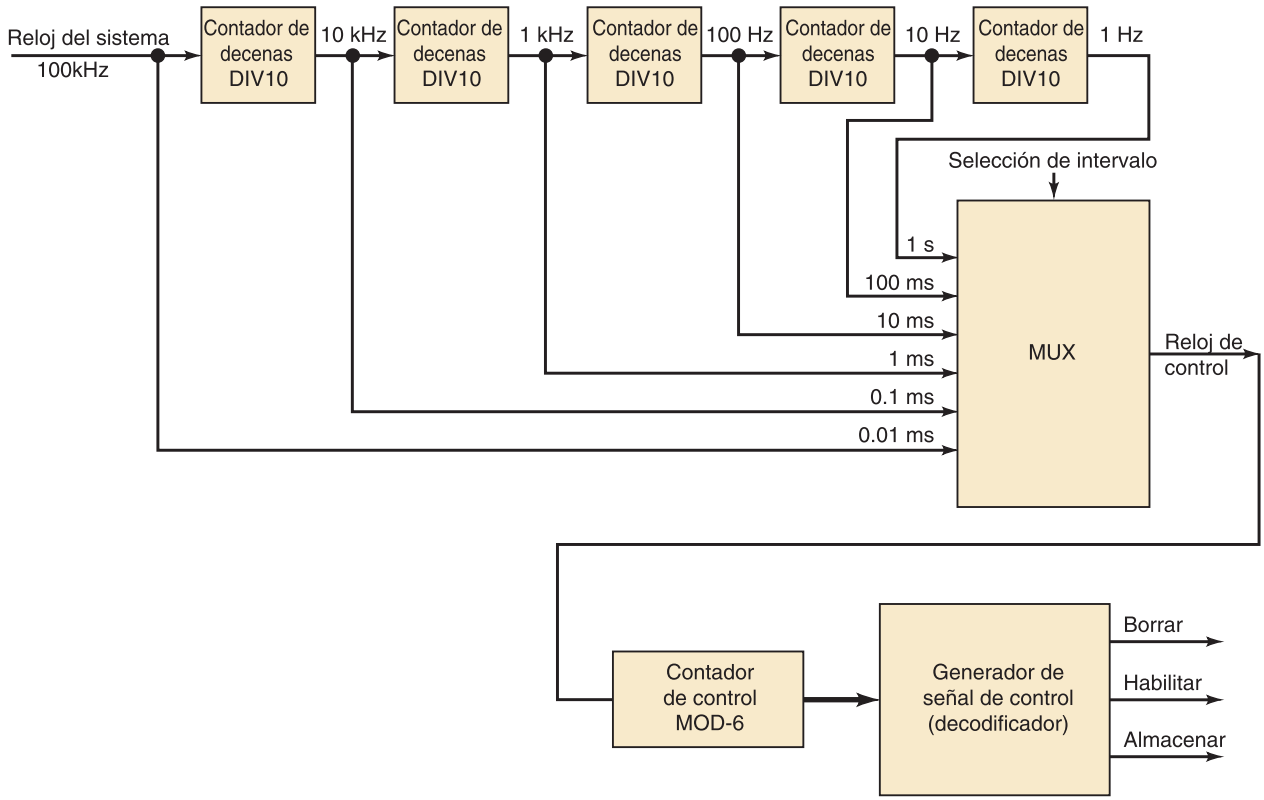


FIGURA 10-41 Bloque de sincronización y control para el contador de frecuencia.

de control repetida (borrar, habilitar y almacenar). El contador (etapas BCD en cascada) se borra primero. Después se habilita en el intervalo de muestreo apropiado para contar los pulsos digitales, los cuales tienen la misma frecuencia que la señal desconocida. Después de deshabilitar el contador, el nuevo conteo se almacena en el registro de visualización.

Las secciones del contador, del registro de visualización y del decodificador/pantalla son simples, por lo cual no las describiremos con más detalle. El bloque de sincronización y control proporciona el "cerebro" de nuestro contador de frecuencia y merece que hablemos un poco más sobre él para explicar su operación. La figura 10-41 muestra los subbloques dentro del bloque de sincronización y control. Para nuestro ejemplo de diseño vamos a suponer que el generador de reloj produce una señal de reloj de sistema de 100 kHz. La frecuencia del reloj de sistema se divide mediante un conjunto de cinco contadores de decenas (MOD-10). Esto proporciona al usuario seis frecuencias distintas que pueden seleccionarse mediante el multiplexor, para la frecuencia del reloj de control mediante el uso del control de selección de intervalo. Como el periodo del reloj de control es igual que la anchura del pulso de la señal de habilitación del contador, esta configuración permite al contador de frecuencia tener seis intervalos de medición de frecuencia distintos. El contador de control es un contador MOD-6 con tres estados seleccionados, los cuales se decodifican mediante el generador de señal de control para producir las señales de control borrar, habilitar y almacenar.

EJEMPLO 10-3

Suponga que el contador BCD de la figura 10-39 consiste de tres etapas BCD en cascada y sus pantallas asociadas. Si la frecuencia desconocida está entre 1 kpps y 9,99 kpps, ¿cuál intervalo de muestreo debe seleccionarse mediante el MUX de la figura 10-41?

Solución

Con tres contadores BCD, la capacidad total del contador es 999. Una frecuencia de 999 kpps produce un conteo de 999 si se utiliza un intervalo de muestreo de 0.1 segundos. Por ende, para poder utilizar la capacidad completa del contador, el MUX debe seleccionar el periodo de reloj de 0.1 s (10 Hz). Si se utilizara un intervalo de muestreo de 1 s, la capacidad del contador siempre se excedería para las frecuencias en el intervalo especificado. Si se utilizara un intervalo de muestreo más corto, el contador sólo contaría entre 1 y 99, lo cual produciría una lectura de sólo dos cifras significativas y sería un desperdicio de la capacidad del contador.

PREGUNTAS DE REPASO

1. ¿Cuál es el propósito de pasar la señal desconocida a través de un formador de pulsos?
2. ¿Cuáles son las unidades de una medición de frecuencia?
3. ¿Qué muestra la pantalla durante el intervalo de muestreo?

RESUMEN

1. Para lograr la administración exitosa de un proyecto se pueden realizar los siguientes pasos: definición del proyecto en general; descomposición del proyecto en piezas pequeñas y estratégicas; síntesis y prueba de cada una de las piezas; e integración del sistema.
2. Los proyectos pequeños tales como el controlador de motor de pasos pueden completarse en un solo archivo de diseño, aún y cuando estos proyectos se desarrollan en forma modular.
3. Los proyectos que consisten de varios bloques de construcción simples, como el codificador de teclado, pueden producir sistemas muy útiles.
4. Los proyectos más grandes tales como el reloj digital pueden aprovechar a menudo los módulos comunes estándar que se pueden utilizar en repetidas ocasiones en el diseño general.
5. Los proyectos deben construirse y probarse en módulos, comenzando en los niveles más bajos de jerarquía.
6. Los módulos preexistentes pueden combinarse fácilmente con nuevos módulos personalizados, mediante el uso de los métodos de descripción gráfico y basado en texto.
7. Los módulos pueden combinarse y representarse como un bloque individual en el siguiente nivel superior de la jerarquía mediante las herramientas de diseño de Altera.

TÉRMINOS IMPORTANTES

anidamiento
contador de frecuencia
intervalo de muestreo
jerarquía
preescalador

PROBLEMAS

SECCIÓN 10-1

B 10-1. El sistema de monitoreo de seguridad de la sección 9-8 del capítulo 9 puede desarrollarse como un proyecto.

(a) Escriba la definición de un proyecto con especificaciones para este sistema.

(b) Defina tres bloques principales de este proyecto.

(c) Identifique las señales que interconectan a los bloques.

(d)* ¿A qué frecuencia debe operar el oscilador para una velocidad de destello de 2.5 Hz?

(e)* ¿Por qué es razonable utilizar sólo una resistencia limitadora de corriente para los ocho LEDs?

SECCIÓN 10-2

Los problemas 10-2 a 10-7 hacen referencia a los motores de pasos que describimos en la sección 10-2.

B 10-2.* ¿Cuántos pasos completos deben ocurrir para una revolución completa?

B 10-3.* ¿Cuántos grados de rotación se producen a partir de un ciclo completo a través de la secuencia de paso completo de la tabla 10-1?

B 10-4. ¿Cuántos grados de rotación se producen a partir de un ciclo completo a través de la secuencia de medio paso de la tabla 10-1?

B 10-5. Las líneas *csal* de la figura 10-1 comenzaron en 1010 y han progresado a través de la siguiente frecuencia: 1010, 1001, 0101, 0110.

(a)* ¿Cuántos grados ha girado el eje?

(b) ¿Qué secuencia invertirá la rotación y regresará el eje a su posición original?

B 10-6. Describa un método para probar el controlador del motor de pasos en:

(a) Modo de paso completo.

(b) Modo de medio paso.

(c) Modo de control de onda.

(d) Modo de control directo.

D, H 10-7. Reformule el archivo de diseño del controlador del motor de pasos de la figura 10-8 o 10-9 sin utilizar una instrucción CASE. Use su HDL preferido.

D, H 10-8. Modifique el archivo de diseño del motor de pasos de la figura 10-8 o 10-9 para agregar una entrada de habilitación que coloque las salidas en el estado Hi-Z (triestado) cuando habilitar = 0.

SECCIÓN 10-3

B 10-9. Escriba la tabla de estados para el contador de anillo que se muestra en la figura 10-1 y que se describe en la figura 10-13.

B 10-10.* Sin oprimir teclas, ¿cuál es el valor en c[3..0]?

B 10-11. Suponga que el contador de anillo se encuentra en el estado 0111 cuando alguien oprime la tecla 7. ¿Avanzará el contador de anillo al estado SIGUIENTE?

B 10-12. Suponga que se oprime la tecla 9 y se sostiene hasta que DAV = 1.

(a)* ¿Cuál es el valor en el contador de anillo?

(b) ¿Cuál es el valor codificado por el codificador de fila?

(c) ¿Cuál es el valor codificado por el codificador de columna?

(d) ¿Qué número binario hay en las líneas D[3..0]?

B 10-13.* En el problema 10-12, ¿serán válidos los datos en el flanco de caída de DAV?

B, D 10-14. Si quisiera fijar los datos del teclado numérico en un registro 74174, ¿qué señal del teclado numérico conectaría al reloj del registro? Dibuje el circuito.

F 10-15.* El teclado numérico está conectado a un latch transparente octal 74373, como se muestra en la figura 10-42. La salida es correcta siempre y cuando se mantenga oprimida una tecla. Sin embargo, no puede fijar los datos entre pulsaciones de teclas. ¿Por qué este circuito *no* funcionará en forma correcta?

---

* Encontrará las respuestas a los problemas marcados con un asterisco al final del libro.

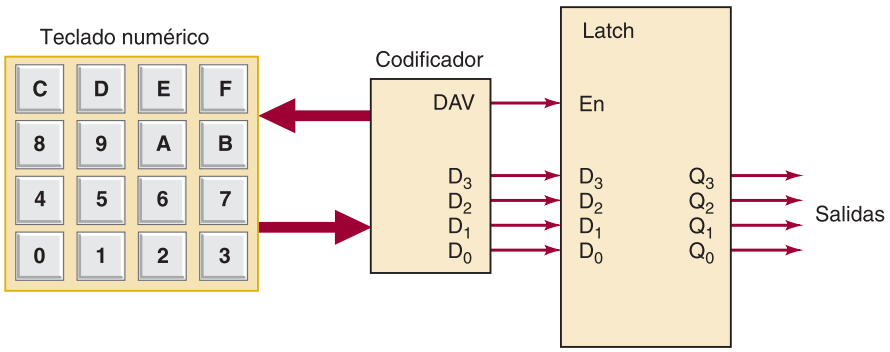


FIGURA 10-42 Problema 10-15.

SECCIÓN 10-4

B 10-16. Suponga que se aplica un reloj de 1 Hz a la etapa de los segundos del reloj en la figura 10-17. La salida de conteo terminal (*tc*) del contador de *unidades de segundos* MOD-10 se muestra en la figura 10-43. Dibuje un diagrama similar en el que se muestre el número de ciclos de reloj entre los pulsos de la salida *tc* de cada uno de los siguientes:

(a)* Contador de decenas de segundos.

(b) Contador de unidades de minutos.

(c) Contador de decenas de minutos.

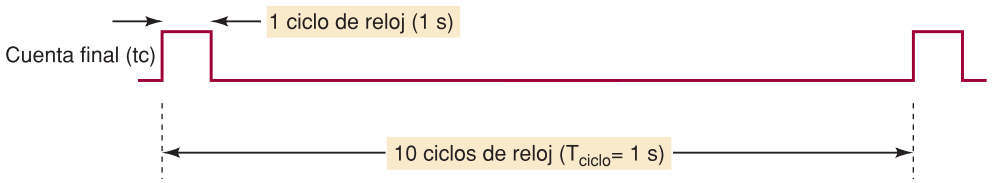


FIGURA 10-43 Problema 10-16.

B 10-17.* ¿Cuántos ciclos de la línea de energía de 60 Hz se producirán en un periodo de 24 horas? ¿Qué problema cree usted que resultará si tratamos de simular la operación de todo el circuito de reloj completo?

D 10-18.* Muchos relojes digitales se ajustan con sólo hacer que cuenten más rápido mientras se mantiene oprimido un botón. Modifique el diseño para agregar esta característica.

D, H 10-19. Modifique la etapa de las horas de la figura 10-18 para mantener el tiempo militar (00-23 horas).

SECCIÓN 10-5

B 10-20. Dibuje el diagrama de jerarquía para el proyecto del contador de frecuencia.

D, H 10-21. Escriba el código de HDL para el contador de control MOD-6 y el generador de señal de control de la figura 10-41.

D, H 10-22.* Escriba el código de HDL para el MUX de la figura 10-41.

D 10-23. Utilice las técnicas de diseño gráfico y el contador BCD descrito en la figura 10-31, el MUX y el diseño del generador de señal de control para crear todo el bloque de sincronización y control completo para el proyecto del contador de frecuencia.

D, H 10-24. Escriba el código de HDL para la sección de sincronización y control del contador de frecuencia.

RESPUESTAS A LAS PREGUNTAS DE REPASO DE SECCIÓN

SECCIÓN 10-1

1. Definición, planeación estratégica, síntesis y prueba, integración y prueba del sistema. 2. La etapa de definición.

SECCIÓN 10-2

1. Paso completo, medio paso, control de onda y control directo. 2. $cent_0$-$cent_3$ [interruptores del selector de modo establecidos en (1,1)]. 3. Paso, sentido [interruptores del selector de modo establecidos en (0,1)]. 4. Ocho etapas

SECCIÓN 10-3

1. Sólo una 2. La primera que se exploró después de oprimirse (por lo general, la primera que se oprimió). 3. Hacer que DAV cambie a ALTO después de que se estabilicen los datos. 4. No, cambia a ALTO en el siguiente pulso de reloj después de oprimir la tecla. 5. Cada vez que OE está en BAJO o cuando no se oprime ninguna tecla.

SECCIÓN 10-4

1. Las especificaciones de operación en general y las entradas y salidas del sistema.
2. En la parte superior de la jerarquía. 3. En la parte inferior; se construyen primero los bloques más simples. 4. En cada etapa de implementación modular.

SECCIÓN 10-5

1. Cambiar la forma de la señal analógica por una señal digital de la misma frecuencia. 2. Ciclos por segundo (Hz) o pulsos por segundo (pps). 3. La pantalla muestra la frecuencia medida durante el intervalo de muestreo anterior.

CAPÍTULO 11

INTERFACE CON EL MUNDO ANALÓGICO

CONTENIDO

11-1 Repaso de la comparación entre digital y analógico
11-2 Conversión digital-analógica (DAC)
11-3 Circuitos convertidores D/A
11-4 Especificaciones de un DAC
11-5 Un DAC de circuito integrado
11-6 Aplicaciones de los DACs
11-7 Diagnóstico de fallas en los DACs
11-8 Conversión analógica-digital (ADC)
11-9 ADC de rampa digital
11-10 Adquisición de datos
11-11 ADC de aproximaciones sucesivas
11-12 ADCs tipo Flash
11-13 Otros métodos de conversión A/D
11-14 Circuitos de muestreo y retención
11-15 Multiplexaje
11-16 Osciloscopio de almacenamiento digital
11-17 Procesamiento digital de señales (DSP)

OBJETIVOS

Al completar este capítulo, usted podrá:

- Comprender la teoría de operación y las limitaciones en los circuitos de varios tipos de convertidores digital-analógico (DACs).
- Leer y comprender las diversas especificaciones de los fabricantes de DACs.
- Utilizar distintos procedimientos de prueba para diagnosticar fallas en circuitos DAC.
- Comparar las ventajas y desventajas entre el convertidor analógico-digital de rampa digital (ADC), el ADC de aproximaciones sucesivas y el ADC flash.
- Analizar el proceso por el cual una computadora, en conjunto con un ADC, digitaliza una señal analógica y después la reconstruye a partir de datos digitales.
- Describir la operación básica de un voltímetro digital.
- Comprender la necesidad de utilizar circuitos de muestreo y retención en conjunto con los ADCs.
- Describir la operación de un sistema de multiplexaje analógico.
- Comprender las características y la operación básica de un osciloscopio de almacenamiento digital.
- Comprender los conceptos básicos del procesamiento digital de señales.

11-1 REPASO DE LA COMPARACIÓN ENTRE DIGITAL Y ANALÓGICO

Una **cantidad digital** tiene un valor que se especifica como una de dos posibilidades tales como 0 o 1, BAJO o ALTO, verdadero o falso, y así sucesivamente. En la práctica, una cantidad digital tal como un voltaje puede en realidad tener un valor que se encuentre en cualquier punto dentro de intervalos especificados, para lo cual definimos valores dentro de un intervalo dado para que tengan el mismo valor digital. Por ejemplo, para la lógica TTL sabemos que

De 0 V a 0.8 V = 0 lógico
De 2 V a 5 V = 1 lógico

Cualquier voltaje que se encuentre dentro del intervalo de 0 a 0.8 V recibe el valor digital de 0, y cualquier voltaje dentro del intervalo de 2 a 5 V recibe el valor digital de 1. Los valores de voltaje exactos no son importantes ya que los circuitos digitales responden de la misma forma a todos los valores de voltaje dentro de un intervalo dado.

Por el contrario, una **cantidad analógica** puede tomar cualquier valor sobre un intervalo continuo de valores y, lo que es más importante, su valor exacto es significativo. Por ejemplo, la salida de un convertidor de temperatura a voltaje analógico podría medirse como 2.76 V, lo cual podría representar una temperatura específica de 27.6 °C. Si la medición de voltaje fuera un poco distinta, como 2.34 V o 3.78 V, esto representaría una temperatura por completo distinta. En otras palabras, cada

uno de los posibles valores de una cantidad analógica tiene un significado distinto. Otro ejemplo sería el voltaje de salida de un amplificador de audio hacia una bocina. Este voltaje es una cantidad analógica, ya que cada uno de sus posibles valores produce una respuesta distinta en la bocina.

La mayoría de las variables físicas son analógicas por naturaleza y pueden tomar cualquier valor dentro de un intervalo continuo de valores. Algunos ejemplos son la temperatura, la presión, la intensidad de la luz, las señales de audio, la posición, la velocidad de rotación y la velocidad de flujo. Los sistemas digitales realizan todas sus operaciones internas mediante el uso de circuitos y operaciones digitales. Cualquier información que deba introducirse en un sistema digital debe colocarse primero en forma digital. De manera similar, las salidas de un sistema digital siempre se encuentran en forma digital. Cuando un sistema digital tal como una computadora se va a utilizar para monitorear y/o controlar un proceso físico, debemos tratar con la diferencia entre la naturaleza digital de la computadora y la naturaleza analógica de las variables del proceso. La figura 11-1 ilustra esta situación. Este diagrama muestra los cinco elementos que se involucran cuando una computadora está monitoreando y controlando una variable física que se supone es analógica:

1. **Transductor.** Por lo general, la variable física es una cantidad no eléctrica. Un **transductor** es un dispositivo que convierte la variable física en eléctrica. Algunos transductores comunes son los termistores, las fotoceldas, los fotodiodos, los medidores de flujo, los transductores de presión y los tacómetros. La salida eléctrica del transductor es una corriente o voltaje analógico proporcional a la variable física que está monitoreando. Por ejemplo, la variable física podría ser la temperatura del agua en un tanque grande que se esté llenando a través de tubería de agua fría y caliente. Digamos que la temperatura del agua varía de 80 a 150 °F y que un termistor y sus circuitos asociados convierten esta temperatura del agua en un voltaje que varía entre 800 y 1500 mV. Observe que la salida del transductor es directamente proporcional a la temperatura, de tal forma que cada 1 °F produce una salida de 10 mV. Elegimos este factor de proporcionalidad por conveniencia.
2. **Convertidor analógico-digital (ADC).** La salida eléctrica analógica del transductor sirve como entrada analógica para el **convertidor analógico-digital (ADC)**. El ADC convierte esta entrada analógica en una salida digital, la cual consiste de un número de bits que representa el valor de la entrada analógica. Por ejemplo, el ADC podría convertir los valores analógicos entre 800 y 1500 mV del transductor en valores binarios que varíen desde 01010000 (80) hasta 10010110 (150). Observe que la salida binaria del ADC es proporcional al voltaje de entrada analógico de tal forma que cada unidad de la salida digital representa 10 mV.

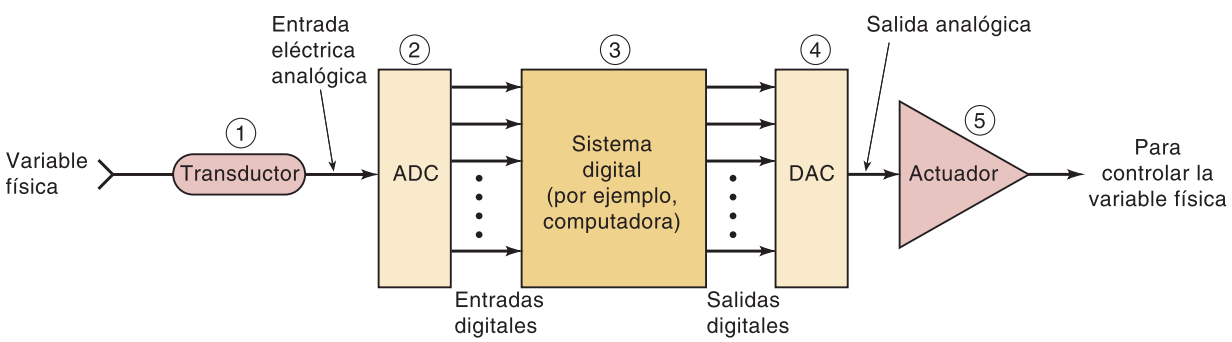


FIGURA 11-1 El convertidor analógico-digital (ADC) y el convertidor digital-analógico (DAC) se utilizan como interfaz para conectar una computadora con el mundo analógico, de manera que ésta pueda monitorear y controlar una variable física.

3. **Computadora.** La representación digital de la variable del proceso se transmite desde el ADC hasta la computadora digital, la cual almacena el valor digital y lo procesa de acuerdo con un programa de instrucciones que está ejecutando. El programa podría realizar cálculos u otras operaciones sobre esta representación digital de la temperatura, para obtener una salida digital que se utilice en cierto momento para controlar la temperatura.
4. **Convertidor digital-analógico (DAC).** La salida digital de la computadora se conecta a un **convertidor digital-analógico (DAC)**, el cual la convierte en un voltaje o corriente analógica proporcional. Por ejemplo, la computadora podría producir una salida digital que varíe entre 00000000 y 11111111; el DAC convertirá esta salida digital en un voltaje que varíe entre 0 y 10 V.
5. **Actuador.** La señal analógica del DAC se conecta con frecuencia a un dispositivo o circuito que sirve como actuador para controlar la variable física. Para nuestro ejemplo de la temperatura del agua, el actuador podría ser una válvula con control eléctrico que regule el flujo de agua caliente hacia el tanque, de acuerdo con el voltaje analógico proveniente del DAC. La velocidad del flujo variaría en proporción a este voltaje analógico, en donde 0 V no producirían flujo y 10 V producirían el máximo flujo.

De todo esto podemos ver que los ADCs y los DACs funcionan como *interfaces* entre un sistema completamente digital, como una computadora, y el mundo analógico. Esta función se ha vuelto cada vez más importante a medida que las microcomputadoras se han desplazado hacia áreas de control de procesos, en donde antes no era factible el control mediante computadoras.

PREGUNTAS DE REPASO

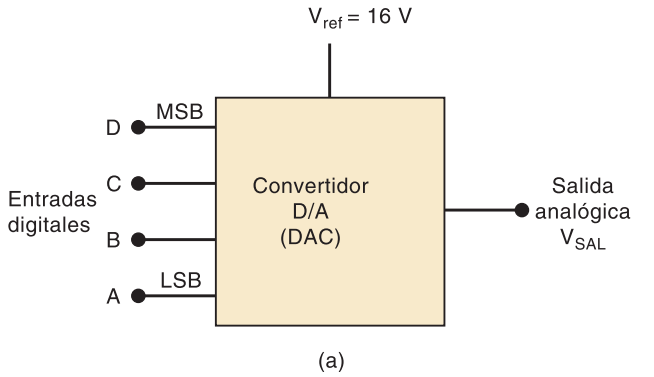
1. ¿Cuál es la función de un transductor?
2. ¿Cuál es la función de un ADC?
3. ¿Qué hace una computadora a menudo con los datos que recibe de un ADC?
4. ¿Qué función realiza un DAC?
5. ¿Cuál es la función de un actuador?

11-2 CONVERSIÓN DIGITAL-ANALÓGICA (DAC)

Ahora comenzaremos nuestro estudio de la conversión digital-analógica (D/A) y analógica-digital (A/D). Muchos métodos de conversión A/D utilizan el proceso de conversión D/A, por lo que primero examinaremos la conversión D/A.

En esencia, la *conversión* D/A es el proceso de tomar un valor representado en código *digital* (como binario directo o BCD) y convertirlo en un voltaje o corriente proporcional al valor digital. La figura 11-2(a) muestra el símbolo para un convertidor D/A ordinario de cuatro bits. No nos preocuparemos por los circuitos internos sino hasta después. Por ahora examinaremos las diversas relaciones de entrada/salida.

Observe que hay una entrada para una referencia de voltaje, V_{ref}. Esta entrada se utiliza para determinar la **salida a escala completa** o valor máximo que puede producir el convertidor D/A. Las entradas digitales D, C, B y A se derivan, por lo general, del registro de salida de un sistema digital. Los $2^4 = 16$ números binarios distintos representados por estos cuatro bits se listan en la figura 11-2(b). Para cada número de entrada, el voltaje de salida del convertidor D/A es un valor único. De hecho, para este caso el voltaje de salida analógico V_{SAL} es igual en volts al número binario. También podría haber sido el doble del número binario o cualquier otro factor de proporcionalidad. La misma idea se aplicaría si la salida D/A fuera una corriente I_{SAL}.



(a)

D	C	B	A	V_{SAL}	
0	0	0	0	0	Volts
0	0	0	1	1	
0	0	1	0	2	
0	0	1	1	3	
0	1	0	0	4	
0	1	0	1	5	
0	1	1	0	6	
0	1	1	1	7	
1	0	0	0	8	
1	0	0	1	9	
1	0	1	0	10	
1	0	1	1	11	
1	1	0	0	12	
1	1	0	1	13	
1	1	1	0	14	
1	1	1	1	15	Volts

(b)



FIGURA 11-2 DAC de cuatro bits con salida de voltaje.

En general,

$$\text{salida analógica} = K \times \text{entrada digital} \qquad \textbf{(11-1)}$$

en donde K es el factor de proporcionalidad y es un valor constante para un DAC dado que se conecta a un voltaje de referencia fijo. Desde luego que la salida analógica puede ser un voltaje o una corriente. Cuando sea un voltaje, K estará en unidades de voltaje (volts) y cuando sea una corriente, K estará en las unidades de corriente (amperes). Para el DAC de la figura 11-2, $K = 1\,\text{V}$, de manera que

$$V_{SAL} = (1\,\text{V}) \times \text{entrada digital}$$

Podemos utilizar esto para calcular V_{SAL} para cualquier valor de entrada digital. Por ejemplo, con una entrada digital de $1100_2 = 12_{10}$, obtenemos

$$V_{SAL} = 1\,\text{V} \times 12 = 12\,\text{V}$$

EJEMPLO 11-1A

Un DAC de cinco bits tiene una salida de corriente. Para una entrada digital de 10100 se produce una corriente de salida de 10 mA. ¿Cuál será el valor de I_{SAL} para una entrada digital de 11101?

Solución

La entrada digital 10100_2 es igual al 20 decimal. Como $I_{SAL} = 10$ mA para este caso, el factor de proporcionalidad debe ser 0.5 mA. Por lo tanto, podemos encontrar I_{SAL} para cualquier entrada digital tal como $11101_2 = 29_{10}$ de la siguiente manera:

$$\begin{aligned} I_{SAL} &= (0.5\text{ mA}) \times 29 \\ &= 14.5\text{ mA} \end{aligned}$$

Recuerde que el factor de proporcionalidad K varía de un DAC a otro y depende del voltaje de referencia.

EJEMPLO 11-1B

¿Cuál es el valor más grande del voltaje de salida proveniente de un DAC de ocho bits que produce 1.0 V para una entrada digital de 00110010?

Solución

$$00110010_2 = 50_{10}$$
$$1.0\,\text{V} = K \times 50$$

Por lo tanto,

$$K = 20\,\text{mV}$$

La salida más grande se producirá para una entrada de $11111111_2 = 255_{10}$.

$$V_{SAL}\,(\text{máx}) = 20\,\text{mV} \times 255$$
$$= 5.10\,\text{V}$$

Salida analógica

En el sentido estricto de la palabra, la salida de un DAC no es una cantidad analógica ya que puede tomar sólo valores específicos tales como los 16 niveles de voltaje posibles para V_{SAL} en la figura 11-2, siempre y cuando V_{ref} sea constante. Por lo tanto, en ese sentido es en realidad digital. No obstante y como veremos, el número de distintos valores de salida posibles puede aumentar y la diferencia entre los valores sucesivos puede disminuir si se incrementa el número de bits de entrada. Esto nos permitirá producir una salida que sea cada vez más como una cantidad analógica que varíe en forma continua, a través de un intervalo de valores. En otras palabras, la salida del DAC es una cantidad "pseudo-analógica". Para nosotros seguirá siendo analógica, teniendo en cuenta que es una aproximación a una cantidad analógica auténtica.

Pesos de entrada

Para el DAC de la figura 11-2, observe que cada entrada digital contribuye con una cantidad distinta para la salida analógica. Esto se puede ver con facilidad si examinamos los casos en los que sólo una entrada está en ALTO (tabla 11-1). Las contribuciones de cada entrada digital se *ponderan* de acuerdo con su posición en el número binario. Por lo tanto, A (que es el LSB) tiene una *ponderación* de 1 V; B una ponderación de 2 V; C una ponderación de 4 V; y D (el MSB) la ponderación más grande, 8 V. Las ponderaciones se duplican sin problemas para cada bit, comenzando con el LSB. De esta manera, podemos considerar a V_{SAL} como la suma ponderada de las entradas digitales. Por ejemplo, para encontrar el valor de V_{SAL} para la entrada digital 0111 podemos sumar las ponderaciones de los bits C, B y A para obtener 4V + 2V + 1V = 7V.

TABLA 11-1

D	*C*	*B*	*A*		V_{SAL} (V)
0	0	0	1	→	1
0	0	1	0	→	2
0	1	0	0	→	4
1	0	0	0	→	8

EJEMPLO 11-2

Un convertidor D/A de cinco bits produce un $V_{SAL} = 0.2$ V para una entrada digital de 00001. Encuentre el valor de V_{SAL} para una entrada de 11111.

Solución

Es obvio que 0.2 V es la ponderación del LSB. Así, las ponderaciones de los demás bits deben ser 0.4 V, 0.8 V, 1.6 V y 3.2 V respectivamente. Entonces, para una entrada digital de 11111 el valor de V_{SAL} será de 3.2 V + 1.6 V + 0.8 V + 0.4 V + 0.2 V = 6.2 V.

Resolución

La **resolución** de un convertidor D/A se define como el cambio más pequeño que puede ocurrir en la salida analógica como resultado de un cambio en la entrada digital. Si hacemos referencia a la tabla de la figura 11-2, podremos ver que la resolución es de 1V, ya que V_{SAL} puede cambiar por no más de 1 V cuando cambia el valor de la entrada digital. La resolución siempre es igual a la ponderación del LSB y también se le conoce como el **tamaño del escalón**, ya que es la cantidad que cambiará V_{SAL} a medida que cambie el valor de entrada digital de un intervalo al siguiente. Esto se ilustra mejor en la figura 11-3, en donde las salidas de un contador binario de cuatro bits proporcionan las entradas para nuestro DAC. A medida que el contador avanza en ciclo en forma continua a través de sus 16 estados mediante la señal de reloj, la salida del DAC es una forma de onda tipo **escalera** que avanza 1 V en cada escalón. Cuando el contador se encuentra en 1111, la salida del DAC se encuentra en su valor máximo de 15 V; ésta es la salida a escala completa. Cuando el contador se recicla a 0000, la salida del DAC regresa a 0 V. La resolución (o tamaño del escalón) es el tamaño de los saltos en la forma de onda de escalera; en este caso es de 1 V.

Observe que la escalera tiene 16 niveles, los cuales corresponden a los 16 estados de entrada, pero sólo hay 15 escalones o saltos entre el nivel de 0 V y la escala completa. En general, para un DAC de N bits el número de niveles distintos será de 2^N y el número de pasos será de $2^N - 1$.

Tal vez ya se haya dado cuenta de que la resolución es lo mismo que el factor de proporcionalidad en la relación de entrada/salida del DAC:

$$\text{salida analógica} = K \times \text{entrada digital}$$

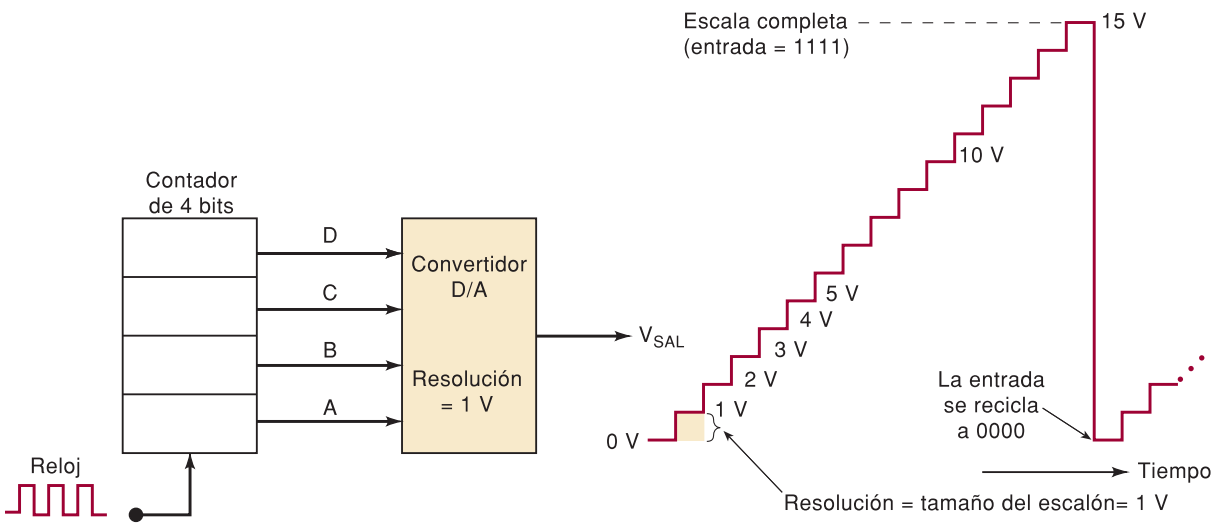


FIGURA 11-3 Formas de onda de salida de un DAC cuando un contador binario proporciona las entradas.

Una nueva interpretación de esta expresión sería que la entrada digital es igual al número de intervalos, K es la cantidad de voltaje (o corriente) por escalón y la salida analógica es el producto de las dos. Ahora tenemos una forma conveniente de calcular el valor de K para el convertidor D/A:

$$\text{resolución} = K\frac{A_{\text{fs}}}{(2^n - 1)} \tag{11-2}$$

en donde A_{fs} es la salida analógica a escala completa y n es el número de bits.

EJEMPLO 11-3A

¿Cuál es la resolución del DAC del ejemplo 11-2? Describa la señal de escalera que sale de este DAC.

Solución

El LSB para este convertidor tiene una ponderación de 0.2 V. Ésta es la resolución, o el tamaño del escalón. Puede generarse una forma de onda tipo escalera si se conecta un contador de cinco bits a las entradas del DAC. La escalera tendrá 32 niveles, desde 0 V hasta una salida a escala completa de 6.2 V, y 31 intervalos de 0.2 V cada uno.

EJEMPLO 11-3B

Para el DAC del ejemplo 11-2, determine V_{SAL} para una entrada digital de 10001.

Solución

El tamaño del escalón es de 0.2 V, que viene siendo el factor de proporcionalidad K. La entrada digital es $10001 = 17_{10}$. Por lo tanto, tenemos que

$$\begin{aligned} V_{SAL} &= (0.2\text{ V}) \times 17 \\ &= 3.4\text{ V} \end{aligned}$$

Resolución porcentual

Aunque la resolución puede expresarse como la cantidad de voltaje o de corriente por intervalo, también es útil expresarla como un porcentaje de la *salida a escala completa*. Para ilustrar esto, el DAC de la figura 11-3 tiene una salida a escala completa máxima de 15 V (cuando la entrada digital es 1111). El tamaño del escalón es de 1 V, lo cual nos proporciona una resolución porcentual de:

$$\begin{aligned} \text{Porcentaje de resolución} &= \frac{\text{tamaño del escalón}}{\text{escala completa(F.S.)}} \times 100\% \\ &= \frac{1\text{ V}}{15\text{ V}} \times 100\% = 6.67\% \end{aligned} \tag{11-3}$$

EJEMPLO 11-4

Un DAC de 10 bits tiene un tamaño de escalón de 10 mV. Determine el voltaje de salida a escala completa y la resolución porcentual.

Solución

Con 10 bits habría $2^{10} - 1 = 1023$ intervalos de 10 mV cada uno. Por lo tanto, la salida a escala completa sería de 10 mV × 1023 = 10.23 V, y

$$\text{porcentaje de resolución} = \frac{10\text{ mV}}{10.23\text{ V}} \times 100\% \approx 0.1\%$$

El ejemplo 11-4 ayuda a ilustrar el hecho de que la resolución porcentual se vuelve más pequeño a medida que aumenta el número de bits de entrada. De hecho, la resolución porcentual también puede calcularse a partir de

$$\text{porcentaje de resolución} = \frac{1}{\text{número total de escalones}} \times 100\% \qquad \textbf{(11-4)}$$

Para un código de entrada binaria de N bits, el número total de3 escalones es $2^N - 1$. Por lo tanto, para el ejercicio anterior,

$$\begin{aligned}\text{porcentaje de resolución} &= \frac{1}{2^{10}-1} \times 100\% \\ &= \frac{1}{1023} \times 100\% \\ &\approx 0.1\%\end{aligned}$$

Esto significa que es *sólo el número de bits* lo que determina la resolución porcentual. Al aumentar el número de bits aumenta el número de escalones para llegar a la escala completa, de manera que cada escalón es una parte más pequeña del voltaje a escala completa. La mayoría de los fabricantes de DACs especifican la resolución como el número de bits.

¿Qué significa resolución?

Un DAC no puede producir un intervalo continuo de valores de salida, por lo cual (hablando en sentido estricto) su salida no es verdaderamente analógica. Un DAC produce un conjunto finito de valores de salida. En nuestro ejemplo sobre la temperatura del agua en la sección 11-1, la computadora genera una salida digital para proporcionar un voltaje analógico entre 0 y 10 V para una válvula con control eléctrico. La resolución (número de bits) del DAC determina cuántos posibles valores de voltaje puede enviar la computadora a la válvula. Si se utiliza un DAC de seis bits, habrá 63 posibles valores en un intervalo de 0.159 V entre 0 y 10 V. Si se utiliza un DAC de ocho bits, habrá 255 posibles valores en un intervalo de 0.039 V entre 0 y 10 V. Entre mayor sea el número de bits, más fina será la resolución (más pequeño el tamaño del escalón).

El diseñador del sistema debe decidir qué resolución se necesita con base en el rendimiento requerido del sistema. La resolución limita lo cerca que puede llegar la salida del DAC a un valor analógico dado. En general, el costo de los DACs aumenta con el número de bits, por lo que el diseñador sólo deberá utilizar los bits que sean necesarios.

EJEMPLO 11-5

La figura 11-4 muestra a una computadora que controla la velocidad de un motor. La corriente analógica de 0 a 2 mA que proviene del DAC se amplifica para producir velocidades de motor de 0 a 1000 rpm (revoluciones por minuto). ¿Cuántos bits deben utilizarse si la computadora puede producir una velocidad de motor que se encuentre a 2 rpm de la velocidad deseada?

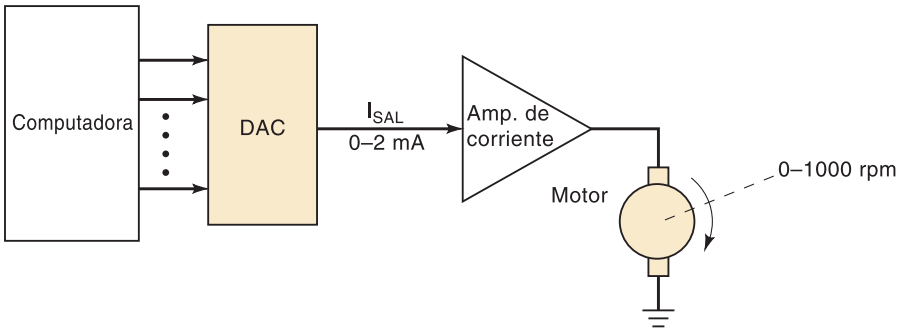


FIGURA 11-4 Ejemplo 11-5.

Solución

La velocidad del motor variará de 0 a 1000 rpm a medida que el DAC vaya de cero a escala completa. Cada intervalo en la salida del DAC producirá un paso más en la velocidad de motor. Queremos que la resolución no sea mayor de 2 rpm. Por lo tanto, necesitamos cuando menos 500 escalones (1000/2). Ahora debemos determinar cuántos bits se requieren, de manera que haya cuando menos 500 intervalos desde cero hasta la escala completa. Sabemos que el número de pasos es $2^N - 1$ y, por lo tanto, podemos decir que

$$2^N - 1 \geq 500$$

o que

$$2^N \geq 501$$

Como $2^8 = 256$ y $2^9 = 512$, el menor número de bits que puede producir cuando menos 500 pasos es *nueve*. Podríamos usar más de nueve bits, pero esto podría aumentar el costo del DAC.

EJEMPLO 11-6

Si utilizamos nueve bits, ¿qué tan cerca de 326 rpm puede ajustarse la velocidad del motor?

Solución

Con nueve bits habrá 511 intervalos ($2^9 - 1$). Así, la velocidad del motor avanzará en intervalos de 1000 rpms/511 = 1.957 rpm. El número de intervalos necesarios para llegar a 326 rpm es 326/1.957 = 166.58. Como éste no es un número entero de intervalos, lo redondearemos a 167. La velocidad real del motor en el intervalo 167 será $167 \times 1.957 = 326.8$ rpm. Por lo tanto, la computadora debe producir como salida el equivalente binario de nueve bits del número 167_{10} para obtener la velocidad de motor deseada dentro de la resolución del sistema.

En todos nuestros ejemplos hemos asumido que los DACs son de una precisión perfecta al producir una salida analógica directamente proporcional a la entrada binaria, y que la resolución es lo único que limita qué tan cerca podemos llegar al valor analógico deseado. Desde luego que esto no es real, ya que todos los dispositivos contienen imprecisiones. En las secciones 11-3 y 11-4 examinaremos las causas y efectos de la imprecisión en el DAC.

DACs bipolares

Hasta este momento hemos asumido que la entrada binaria para un DAC es un número sin signo y que la salida del DAC es un voltaje o corriente positiva. Muchos DACs también pueden producir voltajes negativos al hacer pequeños cambios a los circuitos analógicos en la salida del DAC. En este caso, el intervalo de entradas binarias (por ejemplo, de 00000000 a 11111111) abarca un intervalo de $-V_{ref}$ hasta un valor aproximado a $+V_{ref}$. El valor de 10000000 se convierte en 0 V de salida. La salida de un sistema digital de complemento a 2 con signo puede controlar este tipo de DAC; para ello invierte el MSB y los números binarios con signo se convierten en el valor apropiado para el DAC, como se muestra en la tabla 11-2.

TABLA 11-2

	Complemento a 2 con signo	Entradas del DAC	V_{SAL} del DAC
Más positivo	01111111	11111111	$\sim+ V_{ref}$
Cero	00000000	10000000	0 V
Más negativo	10000000	00000000	$- V_{ref}$

Otros DACs pueden tener integrados los circuitos adicionales para aceptar números con signo en complemento a 2 como entradas. Por ejemplo, suponga que tenemos un DAC bipolar de seis bits que utiliza el sistema de complemento a 2 y que tiene una resolución de 0.2 V. Los valores de entrada binarios varían de 100000 (−32) hasta 011111 (+31) para producir salidas analógicas en el intervalo de −6.4 V a +6.2 V. Hay 63 intervalos ($2^6 - 1$) de 0.2 V entre estos límites positivo y negativo.

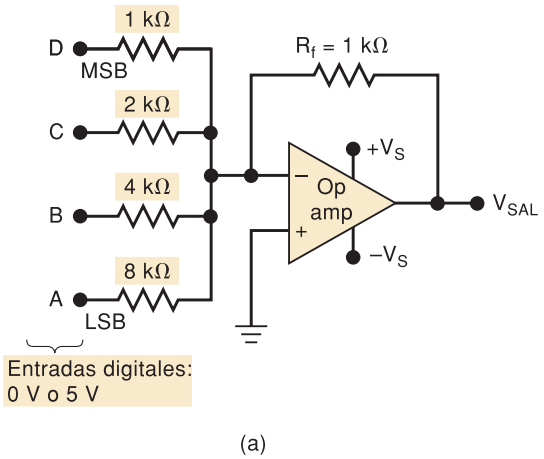
PREGUNTAS DE REPASO

1. Un DAC de ocho bits tiene una salida de 3.92 mA para una entrada de 01100010. ¿Cuál es la resolución y la salida a escala completa del DAC?
2. ¿Cuál es la ponderación del MSB del DAC de la pregunta 1?
3. ¿Cuál es la resolución porcentual de un DAC de ocho bits?
4. ¿Cuántos voltajes de salida distintos puede producir un DAC de 12 bits?
5. Para el sistema de la figura 11-4, ¿cuántos bits se deben utilizar si la computadora va a controlar la velocidad del motor con una tolerancia no mayor de 0.4 rpm?
6. *Verdadero o falso:* la resolución porcentual de un DAC depende *sólo* del número de bits.
7. ¿Cuál es la ventaja de una resolución más pequeña (más fina)?

11-3 CIRCUITOS CONVERTIDORES D/A

Existen varios métodos y circuitos para realizar la operación D/A que hemos descrito. Vamos a examinar varios de los esquemas básicos para ver las generalidades de las ideas utilizadas. No es importante estar familiarizado con todos los esquemas de circuitos, ya que los convertidores D/A están disponibles como CIs o como paquetes encapsulados que no requieren ningún conocimiento sobre sus circuitos. Lo importante es conocer las características de desempeño vitales de los DACs en general, para poder usarlos correctamente. En la sección 11-4 veremos este tema.

La figura 11-5(a) muestra el circuito básico para un tipo de DAC de cuatro bits. *A*, *B*, *C* y *D* son entradas binarias que asumen valores de 0 o de 5 V. El *amplificador operacional* se emplea como amplificador sumador, el cual produce la suma ponde-



(a)

Código de entrada

D	C	B	A	V_{SAL} (volts)	
0	0	0	0	0	
0	0	0	1	−0.625	← LSB
0	0	1	0	−1.250	
0	0	1	1	−1.875	
0	1	0	0	−2.500	
0	1	0	1	−3.125	
0	1	1	0	−3.750	
0	1	1	1	−4.375	
1	0	0	0	−5.000	
1	0	0	1	−5.625	
1	0	1	0	−6.250	
1	0	1	1	−6.875	
1	1	0	0	−7.500	
1	1	0	1	−8.125	
1	1	1	0	−8.750	
1	1	1	1	−9.375	← Escala completa

(b)



FIGURA 11-5 Un DAC simple que utiliza un amplificador operacional sumador con resistencias de ponderación binaria.

rada de estos voltajes de entrada. Recuerde que el amplificador sumador multiplica cada voltaje de entrada por la relación entre la resistencia de retroalimentación R_F y la correspondiente resistencia de entrada R_{ENT}. En este circuito $R_F = 1\ k\Omega$ y las resistencias de entrada varían de 1 a 8 kΩ. Como la entrada D tiene $R_{ENT} = 1\ k\Omega$, el amplificador sumador pasa el voltaje en D sin atenuación. Como la entrada C tiene $R_{ENT} = 2\ k\Omega$, se atenuará por ½. De manera similar, la entrada B se atenuará por ¼ y la entrada A por $^1/_8$. Por lo tanto, la salida del amplificador puede expresarse como

$$V_{SAL} = -(V_D + \tfrac{1}{2} V_C + \tfrac{1}{4} V_B + \tfrac{1}{8} V_A) \qquad \textbf{(11-5)}$$

El signo negativo está presente debido a que el amplificador sumador es un amplificador inversor de polaridad, lo cual no es importante aquí.

Es evidente que la salida del amplificador sumador es un voltaje analógico que representa una suma ponderada de las entradas digitales, como se muestra en la tabla de la figura 11-5(b). Esta tabla lista todas las posibles condiciones de entrada y el voltaje de salida resultante en el amplificador. La salida se evalúa para cualquier condición de entrada; para ello se establecen las entradas apropiadas a 0 o a 5 V. Por ejemplo, si la entrada digital es 1010 entonces $V_D = V_B = 5\ V$ y $V_C = V_A = 0\ V$. Ahora, si utilizamos la ecuación (11-5),

$$V_{SAL} = -(5\ V + 0\ V + \tfrac{1}{4} \times 5\ V + 0\ V)$$
$$= -6.25\ V$$

La resolución de este convertidor D/A es igual a la ponderación del LSB, que viene siendo $^1/_8 \times 5\ V = 0.625\ V$. Como se muestra en la tabla, la salida analógica se incrementa 0.625 V cuando el número de entrada binario se incrementa en uno.

EJEMPLO 11-7

(a) Determine la ponderación de cada bit de entrada de la figura 11-5(a).

(b) Cambie R_F a 250 Ω y determine la salida a escala completa.

Solución

(a) El MSB pasa con ganancia = 1, por lo que su ponderación en la salida es de 5 V. En conclusión,

MSB → 5 V
2do. MSB → 2.5 V
3er. MSB → 1.25 V
4to. MSB → 0.625 V

(b) Si R_F se reduce por un factor de 4 a 250 Ω, la ponderación de cada entrada será cuatro veces *menor* que los valores antes mencionados. Así, la salida a escala completa se reducirá por el mismo factor y se convertirá en −9.375/4= −2.344 V.

Si analizamos los valores de las resistencias de entrada en la figura 11-5, no deberá sorprendernos que tengan *ponderación binaria*. En otras palabras y comenzando con la resistencia del MSB, los valores de las resistencias se incrementan por un factor de 2. Desde luego que esto produce la ponderación deseada en la salida del voltaje.

Precisión en la conversión

La tabla de la figura 11-5(b) proporciona los valores *ideales* de V_{SAL} para los diversos casos de entrada. El grado de aproximación con el que el circuito produzca estos valores depende en primera instancia de dos factores: (1) la precisión de las resistencias de entrada y de retroalimentación y (2) la precisión de los niveles de voltaje de entrada. Las resistencias pueden hacerse muy precisas (dentro de un intervalo no mayor del 0.01 por ciento de los valores deseados) mediante un ajuste fino, pero los niveles de voltaje de entrada deben manejarse de manera distinta. Hay que dejar claro que las entradas digitales no pueden tomarse directamente de las salidas de los FFs o compuertas lógicas, ya que los niveles lógicos de salida de estos dispositivos no son valores precisos como 0 V y 5 V, sino que varían dentro de intervalos específicos. Por esta razón es necesario agregar más circuitos entre cada entrada digital y su resistencia de entrada al amplificador sumador, como se muestra en la figura 11-6.

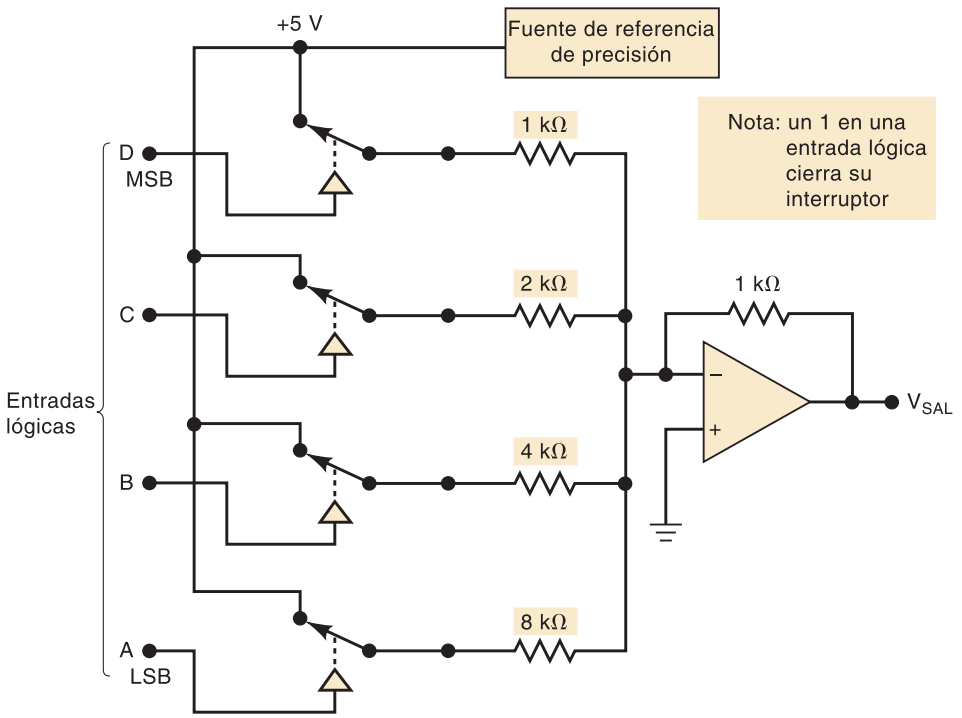


FIGURA 11-6 DAC completo de cuatro bits que incluye una fuente de referencia de precisión.

Cada entrada digital controla un interruptor semiconductor, tal como la compuerta de transmisión CMOS que estudiamos en el capítulo 8. Cuando la entrada está en ALTO, el interruptor se cierra y conecta una *fuente de referencia de precisión* a la resistencia de entrada; cuando la entrada está en BAJO el interruptor está abierto. La fuente de referencia produce un voltaje muy estable y preciso, necesario para generar una salida analógica precisa.

DAC con salida de corriente

La figura 11-7(a) muestra un esquema básico para generar una corriente de salida analógica proporcional a una entrada binaria. El circuito que se muestra es un DAC de cuatro bits, el cual utiliza resistencias con ponderación binaria. El circuito utiliza cuatro rutas de corriente en paralelo, cada una de las cuales se controla mediante un interruptor semiconductor tal como la compuerta de transmisión CMOS. El estado de cada interruptor se controla mediante niveles lógicos en las entradas binarias. La corriente que fluye a través de cada ruta se determina mediante un voltaje de referencia preciso (V_{REF}) y una resistencia de precisión en la ruta. Las resistencias se ponderan en forma binaria, de manera que las diversas corrientes tendrán ponderación binaria y la corriente total (I_{SAL}) será la suma de las corrientes individuales. La ruta del MSB tiene la resistencia más pequeña, R; la siguiente ruta tiene una resistencia del doble de ese valor; y así en lo sucesivo. Se puede hacer que la corriente de salida fluya a través de una carga R_L que sea más pequeña que R, para que no tenga efecto sobre el valor de la corriente. En teoría, R_L debe ser un corto a tierra.

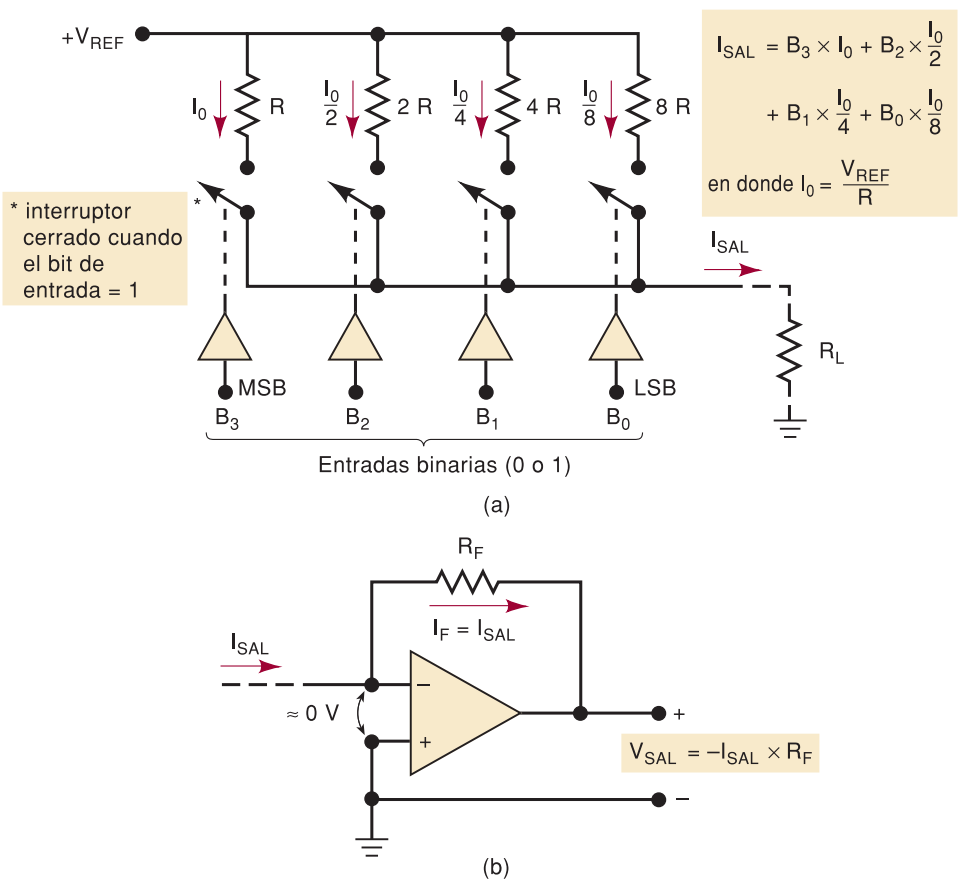


FIGURA 11-7 (a) DAC básico con salida de corriente; (b) conectado a un amplificador operacional convertidor de corriente a voltaje.

EJEMPLO 11-8

Suponga que $V_{REF} = 10$ V y $R = 10$ kΩ. Determine la resolución y la salida a escala completa para este DAC. Suponga que R_L es mucho más pequeña que R.

Solución

$I_{SAL} = V_{REF}/R = 1$ mA. Éste es el peso del MSB. Las otras tres corrientes serán 0.5, 0.25 y 0.125 mA. El LSB es de 0.125 mA, que también viene siendo la resolución.

La salida a escala completa se producirá cuando todas las entradas binarias se encuentren en ALTO, de manera que se cierre cada uno de los interruptores de corriente y

$$I_{SAL} = 1 + 0.5 + 0.25 + 0.125 = 1.875 \text{ mA}$$

Observe que la corriente de salida es proporcional a V_{REF}. Si V_{REF} se incrementa o se decrementa, la resolución y la salida a escala completa cambiarán en forma proporcional.

Para que I_{SAL} sea precisa, R_L debe ser un corto a tierra. Una manera común de lograr esto es utilizar un amplificador operacional como un convertidor de corriente a voltaje, como se muestra en la figura 11-7(b). Aquí, la corriente I_{SAL} del DAC se conecta a la entrada "−" del amplificador operacional, que se encuentra virtualmente conectada a tierra. La retroalimentación negativa del amplificador operacional hace que fluya una corriente igual a I_{SAL} a través de R_F para producir $V_{SAL} = -I_{OUT} \times R_F$. Por lo tanto, V_{SAL} será un voltaje analógico proporcional a la entrada binaria para el DAC. Esta salida analógica puede controlar sin problemas una amplia variedad de cargas.

Red *R/2R*

Los circuitos DAC que hemos visto hasta ahora utilizan resistencias con ponderación binaria para producir la ponderación apropiada de cada bit. Mientras que este método funciona en teoría, tiene ciertas limitaciones prácticas. El mayor problema es la gran diferencia en los valores de las resistencias entre el LSB y el MSB, en especial en los DACs de alta resolución (es decir, con muchos bits). Por ejemplo, si la resistencia del MSB es de 1 kΩ en un DAC de 12 bits, la resistencia del LSB será mayor de 2 MΩ. Con la tecnología actual de fabricación de CIs, es muy difícil producir valores de resistencias con un amplio intervalo y que mantengan una relación precisa, en especial con las variaciones en la temperatura.

Por esta razón es preferible tener un circuito que utilice resistencias que estén muy cerca del valor deseado. Uno de los circuitos DAC con más amplio uso que cumple con este requerimiento es la red de *escalera R/2R*, en la cual los valores de las resistencias guardan una relación de sólo 2 a 1. En la figura 11-8 se muestra un DAC de este tipo.

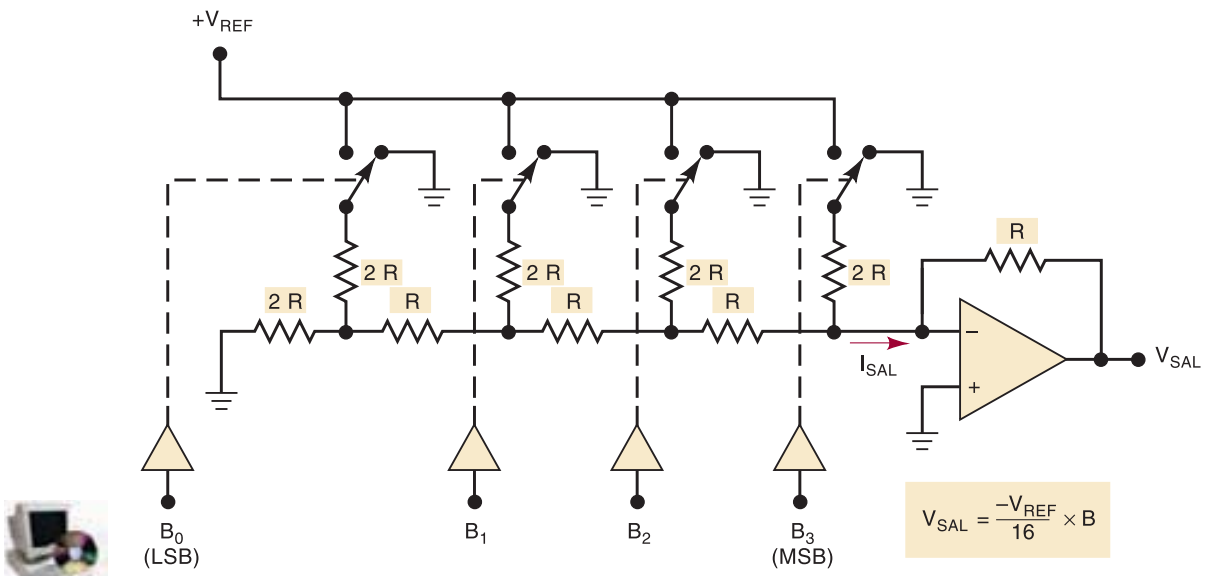


FIGURA 11-8 DAC básico de escalera *R/2R*.

Note cómo están ordenadas las resistencias; en especial observe que sólo se utilizan dos valores distintos: R y $2R$. La corriente I_{SAL} depende de las posiciones de los cuatro interruptores y las entradas binarias $B_3B_2B_1B_0$ controlan los estados de los interruptores. A esta corriente se le permite fluir a través de un amplificador operacional convertidor de corriente a voltaje para desarrollar V_{SAL}. No realizaremos un análisis detallado de este circuito aquí, pero podemos demostrar que el valor de V_{SAL} se obtiene mediante la expresión

$$V_{\text{SAL}} = \frac{-V_{\text{REF}}}{16} \times B \tag{11-6}$$

en donde B es el valor de la entrada binaria, la cual puede variar de 0000 (0) a 1111 (15).

EJEMPLO 11-9

Suponga que $V_{REF} = 10$ V para el DAC en la figura 11-8. ¿Cuál es la resolución y la salida a escala completa de este convertidor?

Solución

La resolución es igual a la ponderación del LSB, la cual podemos determinar si establecemos $B = 0001 = 1$ en la ecuación (11-6):

$$\text{resolución} = \frac{-10\text{ V} \times 1}{16} = -0.624\text{ V}$$

La salida a escala completa ocurre para $B = 1111 = 15_{10}$. Si utilizamos de nuevo la ecuación (11-6),

$$\text{escala completa} = \frac{-10\text{ V} \times 15}{16} = -9.375\text{ V}$$

PREGUNTAS DE REPASO

1. ¿Cuál es la ventaja de los DACs tipo escalera $R/2R$, en comparación con los que utilizan resistencias con ponderación binaria?
2. Cierto DAC de seis bits utiliza resistencias con ponderación binaria. Si la resistencia del MSB es de 20 kΩ, ¿cuál es la resistencia del LSB?
3. ¿Cuál será la resolución si el valor de R_F en la figura 1-5 se cambia a 800 Ω?
4. ¿Qué ocurrirá con la resolución y la salida a escala completa cuando V_{REF} se incremente en un 20 por ciento?

11-4 ESPECIFICACIONES DE UN DAC

Hay una amplia variedad de DACs disponibles en la actualidad como CIs o paquetes encapsulados, autocontenidos. Uno debe estar familiarizado con las especificaciones más importantes de los fabricantes para poder evaluar un DAC para una aplicación específica.

Resolución

Como dijimos antes, la resolución porcentual de un DAC depende únicamente del número de bits. Debido a ello, por lo general, los fabricantes especifican la resolu-

ción de un DAC como el número de bits. Un DAC de 10 bits tiene una resolución más fina (pequeña) que un DAC de ocho bits.

Precisión

Los fabricantes de DACs pueden especificar la precisión de varias formas. Las dos más comunes son el **error a escala completa** y el **error de linealidad**, que, por lo general, se expresan como un porcentaje de la salida a escala completa del convertidor (% F.S.).

El error a escala completa es la desviación máxima de la salida del DAC de su valor esperado (ideal), lo cual se expresa como un porcentaje de la escala completa. Por ejemplo, suponga que el DAC de la figura 11-5 tiene una precisión de ±0.01% F.S. Como este convertidor tiene una salida a escala completa de 9.375 V, este porcentaje se convierte en

$$\pm 0.01\% \times 9.375\,\text{V} = \pm 0.9375\,\text{mV}$$

Esto significa que en cualquier momento, la salida de este DAC puede desviarse hasta 0.9375 mV de su valor esperado.

El error de linealidad es la máxima desviación en el tamaño del escalón, en base al tamaño de escalón ideal. Por ejemplo, el DAC de la figura 11-5 tiene un tamaño de escalón esperado de 0.625 V. Si este convertidor tiene un error de linealidad de ±0.01% F.S., esto significaría que el *tamaño de escalón* actual podría desviarse hasta 0.9375 mV.

Es importante comprender que la precisión y la resolución de un DAC deben ser compatibles. Es ilógico tener una resolución de, por decir, 1 por ciento y una precisión de 0.1 por ciento. Para ilustrar esto, podemos considerar un DAC con una resolución de 1% y una salida F.S. de 10 V, el cual puede producir un voltaje analógico de salida con una desviación máxima de 0.1 V de cualquier valor deseado, suponiendo una precisión perfecta. No tiene sentido tener una precisión costosa de 0.01% F.S. (o 1 mV) si la resolución limita de antemano la cercanía del valor deseado a 0.1 V. Lo mismo puede decirse cuando se tiene una resolución que es muy pequeña (muchos bits) mientras que la precisión es pobre; es un desperdicio de bits de entrada.

EJEMPLO 11-10

Cierto DAC de ocho bits tiene una salida a escala completa de 2 mA y un error a escala completa de ±0.5 F.S. ¿Cuál es el intervalo de posibles salidas para una entrada de 10000000?

Solución

El tamaño de escalón es 2 mA/255 = 7 μA. Como $10000000 = 128_{10}$, la salida ideal debería ser 128 × 7.84 μA = 1004 μA. El error puede ser de hasta

$$\pm 0.5\% \times 2\,\text{mA} = \pm 10\,\mu\text{A}$$

Por lo tanto, la salida actual puede desviarse esta cantidad de los 1000 μA ideales y la salida puede ser cualquier valor entre 994 y 1014 μA.

Error de desplazamiento

En teoría, la salida de un DAC será cero volts cuando todos los bits de la entrada binaria sean 0. No obstante, en la práctica habrá un voltaje de salida muy pequeño para este caso; a esto se le conoce como **error de desplazamiento**. Si este error de desplazamiento no se corrige, se agregará a la salida esperada del DAC para *todos* los casos de entrada. Por ejemplo, digamos que un DAC de cuatro bits tiene un error de desplazamiento de +2 mV y un tamaño de escalón *perfecto* de 100 mV. La tabla 11-3 muestra la salida ideal y actual del DAC para varios casos de entrada. Observe que la salida actual es 2 mV mayor de la esperada; esto se debe al error de desplazamiento. Este error puede ser tanto negativo como positivo.

TABLA 11-3

Código de entrada	Salida ideal (mV)	Salida actual (mV)
0000	0	2
0001	100	102
1000	800	802
1111	1500	1502

Muchos DACs tienen un ajuste de desplazamiento externo, el cual nos permite poner el desplazamiento en cero. Para ello se aplican, por lo general, sólo ceros a la entrada del DAC y se monitorea la salida mientras se ajusta un *potenciómetro de ajuste de desplazamiento* hasta que la salida esté lo más cerca posible de 0 V.

Tiempo de estabilización

La velocidad de operación de un DAC se especifica, por lo general, mediante su **tiempo de estabilización**, el cual es el tiempo requerido para que la salida del DAC cambie de cero a escala completa, a medida que la entrada binaria se cambia de todos los bits en 0 a todos los bits en 1. En realidad, el tiempo de estabilización se mide como el tiempo que tarda la salida del DAC en estabilizarse dentro de un intervalo no mayor a ±1/2 del tamaño del escalón (resolución) de su valor final. Por ejemplo, si un DAC tiene una resolución de 10 mV, el tiempo de estabilización se mide como el tiempo que tarda la salida en estabilizarse dentro de un intervalo no mayor a 5 mV del valor de su escala completa.

Los valores típicos para el tiempo de estabilización varían entre 50 ns y 10 µs. Hablando en general, los DACs con una salida de corriente tendrán tiempos de estabilización más cortos que los DACs con salidas de voltaje. La principal razón de esta diferencia es el tiempo de respuesta del amplificador operacional que se utiliza como convertidor de corriente a voltaje.

Monotonicidad

Un DAC es **monotónico** si su salida se incrementa a medida que se incrementa la entrada binaria de un valor al siguiente. Otra forma de describir esto es que la salida tipo escalera no tendrá escalones descendentes a medida que se incremente la entrada binaria de cero a escala completa.

PREGUNTAS DE REPASO

1. Defina *error a escala completa*.
2. ¿Qué es el *tiempo de estabilización*?
3. Describa lo que es el error de desplazamiento y su efecto sobre la salida de un DAC.
4. ¿Por qué los DACs de voltaje son en general más lentos que los DACs de corriente?

11-5 UN DAC DE CIRCUITO INTEGRADO

El AD7524, un CI CMOS disponible de varios fabricantes, es un convertidor D/A de ocho bits que utiliza una red de escalera *R/2R*. Su símbolo de bloque se muestra en la figura 11-9(a). Este DAC tiene una entrada de ocho bits que puede fijarse en forma interna bajo el control de las entradas Selección de chip (*CS*) y WRITE (*WR*). Cuando ambas entradas de control están en BAJO, las entradas de datos digitales D_7-D_0 producen la corriente de salida analógica *OUT 1* (por lo general, la terminal *OUT 2* está aterrizada). Cuando una de las entradas de control cambia a ALTO se fijan los datos de entrada digitales y la salida analógica permanece en el nivel correspondiente a esos datos digitales fijos. Los cambios subsiguientes en las entradas digitales no tendrán efecto sobre *OUT 1* en este estado de fijación.

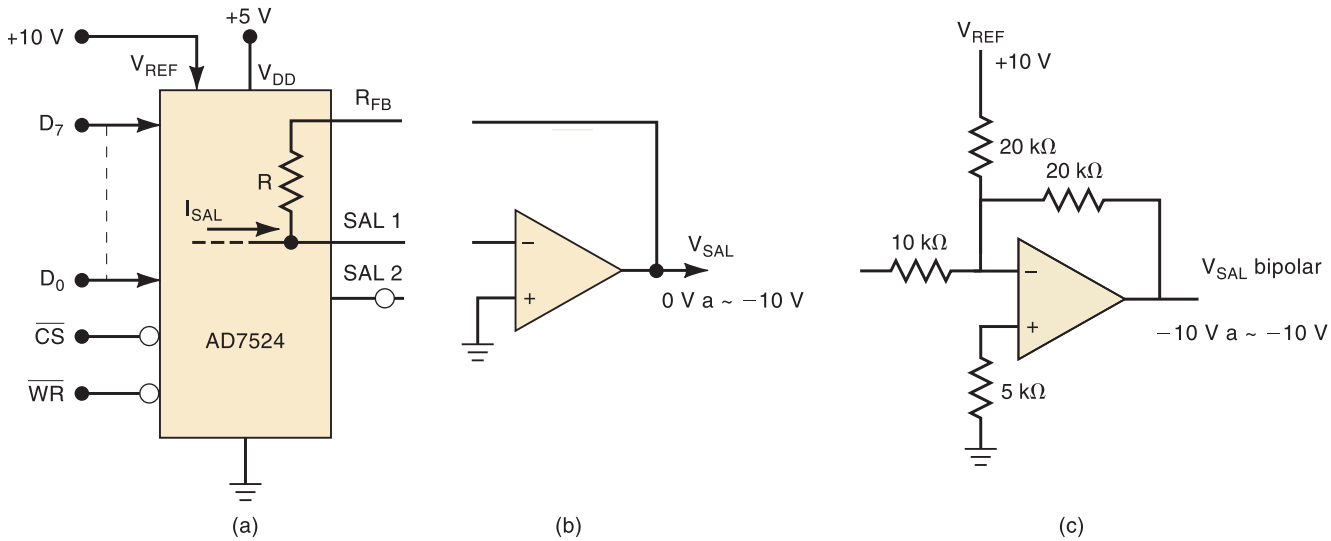


FIGURA 11-9 (A) Un DAC AD7524 de 8 bits con entradas tipo latch; (b) amplificador operacional convertidor de corriente a voltaje que proporciona de 0 a un valor aproximado a los 10 V de salida; (c) circuito de amplificador operacional para producir una salida bipolar desde −10 V hasta un valor aproximado a los +10 V.

El tiempo de estabilización máximo para el AD7524 es, por lo general, de 100 ns, y su precisión de escala completa está clasificada a ±0.2% F.S. El V_{REF} puede variar entre voltajes tanto negativos como positivos desde 0 a 25 V, por lo que pueden producirse corrientes de salida analógicas de ambas polaridades. La corriente de salida puede convertirse en un voltaje mediante el uso de un amplificador operacional conectado como en la figura 11-9(b). Observe que la resistencia de retroalimentación del amplificador operacional ya se encuentra en el chip DAC. Puede agregarse el circuito del amplificador operacional que se muestra en la figura 11-9(c) para producir una salida bipolar que varíe de $-V_{ref}$ (cuando la entrada = 00000000) a casi $+V_{ref}$ (cuando la entrada + 11111111).

11-6 APLICACIONES DE LOS DACS

Los DACs se utilizan cada vez que la salida de un circuito digital debe proporcionar un voltaje o corriente analógico para controlar un dispositivo analógico. En los siguientes párrafos describiremos algunas de las aplicaciones más comunes.

Control

La salida digital de una computadora puede convertirse en una señal de control analógica para ajustar la velocidad de un motor o la temperatura de un horno, o para controlar casi cualquier variable física.

Prueba automática

Las computadoras pueden programarse para generar señales analógicas (a través de un DAC) necesarias para probar circuitos analógicos. Por lo general, la respuesta de la salida analógica del circuito de prueba se convierte en un valor digital mediante un ADC y después se alimenta a la computadora para almacenarla, visualizarla y algunas veces analizarla.

Reconstrucción de señales

En muchas aplicaciones, una señal analógica se **digitaliza**; esto es, puntos sucesivos en la señal se convierten en sus equivalentes digitales y se almacenan en memoria. Esta conversión se realiza a través de un convertidor analógico-digital (ADC). Des-

pués se puede utilizar un DAC para convertir los datos digitalizados que están almacenados de vuelta a datos analógicos (un punto a la vez), con lo cual se reconstruye la señal original. Esta combinación de digitalización y reconstrucción se utiliza en los osciloscopios de almacenamiento digital, en los sistemas de discos compactos y en la grabación de audio y video digital. Más adelante, cuando veamos los ADCs, analizaremos estas aplicaciones.

Conversión A/D

Varios tipos de ADCs utilizan DACs como parte de su circuito, como veremos en la sección 11-8.

Control de amplitud digital

Los DACs también pueden utilizarse para reducir la amplitud de una señal analógica, conectando esta señal a la entrada V_{REF} como se muestra en la figura 11-10. La entrada binaria escala la señal en V_{REF}: $V_{SAL} = V_{REF} \times$ ent. binaria/2^N. Cuando se aplica el máximo valor de entrada binario, la salida es casi igual que la entrada V_{REF}. No obstante, cuando un valor que representa casi la mitad del máximo (por ejemplo, 1000 000 para un convertidor de ocho bits unipolar) se aplica a las entradas, la salida es aproximadamente la mitad de V_{REF}. Si V_{REF} es una señal (por ejemplo, una onda senoidal) que varía dentro del intervalo del voltaje de referencia, la salida será la misma forma de onda completamente analógica cuya amplitud depende del número digital que se aplique al DAC. De esta forma, un sistema digital puede controlar cosas tales como el volumen de un sistema de audio o la amplitud de un generador de funciones.

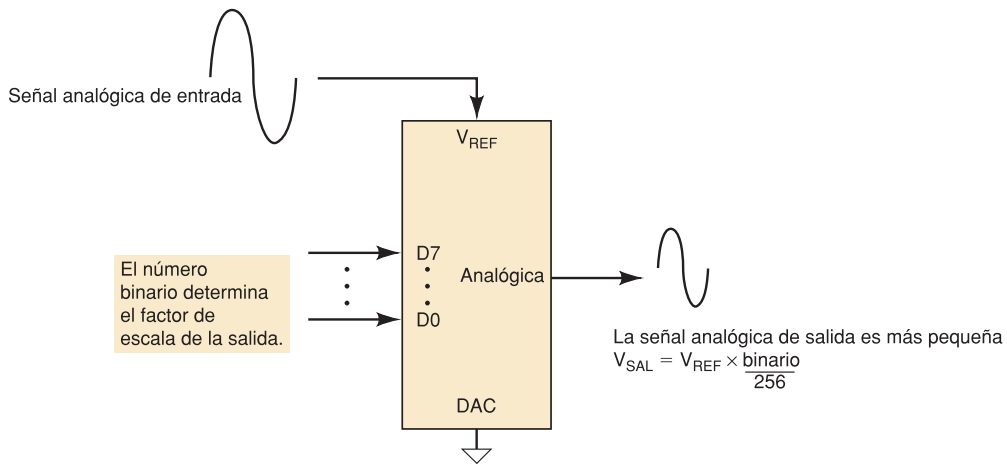


FIGURA 11-10 Un DAC que se utiliza para controlar la amplitud de una señal analógica.

DACs en serie

Muchas de estas aplicaciones requieren el uso de un microprocesador. El principal problema con el uso de los DACs de datos en paralelo que hemos descrito hasta ahora es que ocupan demasiados bits de puerto de la microcomputadora. En los casos en los que la velocidad de la transferencia de datos no es muy importante, un microprocesador puede enviar el valor digital a un DAC a través de una interface en serie. Actualmente existen DACs en serie disponibles con un registro de desplazamiento de entrada en serie/salida en paralelo integrado. Muchos de estos dispositivos tienen más de un DAC en el mismo chip. Los datos digitales, junto con un código que especifica cuál es la DAC que uno quiere, se envían al chip bit por bit. A medida que se presenta cada bit en la entrada del DAC, se aplica un pulso a la entrada de reloj en serie para desplazar los datos hacia la entrada. Después del número apropiado de pulsos de reloj, el valor de los datos se fija y se convierte en su valor analógico.

11-7 DIAGNÓSTICO DE FALLAS EN LOS DACs

Los DACs son tanto digitales como analógicos. Pueden utilizarse sondas y pulsadores lógicos en las entradas digitales, pero se debe usar un multímetro o un osciloscopio en la salida analógica. Básicamente hay dos formas de probar la operación de un DAC: una *prueba de precisión estática* y una *prueba de escalera*.

La prueba estática implica establecer la entrada binaria a un valor fijo y medir la salida analógica con un multímetro de alta precisión. Esta prueba se utiliza para comprobar que el valor de salida se encuentre dentro del intervalo esperado y que sea consistente con la precisión especificada del DAC. Si no es así, puede deberse a varias causas. He aquí algunas de ellas:

- Variación en los valores de los componentes internos del DAC (por ejemplo, los valores de las resistencias) ocasionada por la temperatura, el desgaste u otros factores. Esta condición puede producir con facilidad valores de salida fuera del intervalo de precisión esperado.
- Conexiones abiertas o cortos en cualquiera de las entradas binarias. Esto podría evitar que una entrada agregue su ponderación a la salida analógica o podría hacer que su peso estuviera presente en forma permanente en la salida. Esta situación es en especial difícil de detectar cuando la falla se encuentra en las entradas menos significativas.
- Una referencia de voltaje defectuosa. Como la salida analógica depende en forma directa de V_{REF}, esto podría dar resultados demasiado alejados. Para los DACs que utilizan fuentes de referencia externas, el voltaje de referencia puede comprobarse con facilidad mediante un voltímetro digital. Muchos DACs cuentan con voltajes de referencia internos que no pueden comprobarse, excepto en algunos DACs que sacan el voltaje de referencia en una terminal del CI.
- Error de desplazamiento excesivo, provocado por el desgaste de un componente o la temperatura. Esto produciría salidas desviadas por una cantidad fija. Si el DAC tiene de ajuste de desplazamiento externa, este tipo de error puede tener la capacidad de eliminarse al principio, pero los cambios en la temperatura de operación pueden ocasionar que vuelva a aparecer el error de desplazamiento.

La prueba de escalera se utiliza para comprobar la monotonicidad del DAC; esto es, comprueba que la salida se incremente de intervalo a intervalo, a medida que se incrementa la entrada binaria, como en la figura 11-3. Los intervalos en la escalera deben ser del mismo tamaño y no debe omitirse ninguno ni debe haber intervalos descendentes hasta llegar a la escala completa. Esta prueba puede ayudar a detectar fallas internas o externas que hagan que una entrada no tenga contribución o que tenga una contribución permanente a la salida analógica. El siguiente ejemplo ilustrará esto.

EJEMPLO 11-11

¿Cómo aparecería la forma de onda de escalera si la entrada *C* para el DAC de la figura 11-3 está abierta? Suponga que las entradas del DAC son compatibles con TTL.

Solución

El DAC interpretará una conexión abierta en *C* como un 1 lógico constante. Por lo tanto, esto contribuirá un valor constante de 4 V a la salida del DAC, de manera que la forma de onda de salida aparecerá como se muestra en la figura 11-11. Las líneas punteadas son la escalera que aparecería si el DAC estuviera funcionando en forma correcta. Observe que la forma de onda de salida defectuosa concuerda con la correcta durante los tiempos en los que la entrada del bit *C* estaría, por lo general, en ALTO.

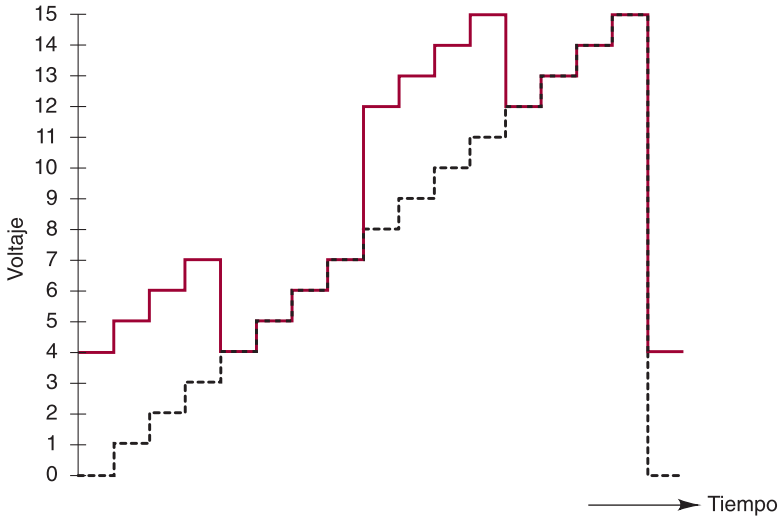


FIGURA 11-11
Ejemplo 11-11.

11-8 CONVERSIÓN ANALÓGICA-DIGITAL (ADC)

Un convertidor analógico-digital toma un voltaje de entrada analógico, y después de cierto tiempo produce un código de salida digital que representa a la entrada analógica. Por lo general, el proceso de conversión A/D es más complejo y consume más tiempo que el proceso D/A, por lo cual se han desarrollado y utilizado varios métodos. Examinaremos algunos de ellos con detalle, aún y cuando tal vez nunca sea necesario diseñar o construir ADCs (dado que están disponibles como unidades encapsuladas en IC). No obstante, las técnicas que se utilizan proporcionan una visión interna de los factores que determinan el desempeño de un ADC.

Varios tipos importantes de ADCs utilizan un DAC como parte de sus circuitos. La figura 11-12 es un diagrama de bloques general para esta clase de ADC. La sincronización para la operación se proporciona mediante la señal del reloj de entrada. La unidad de control contiene los circuitos lógicos para generar la secuencia apropiada de operaciones en respuesta a la SEÑAL DE INICIO, el cual inicia el proceso de conversión. El amplificador operacional comparador tiene dos entradas *analógicas* y una salida *digital* que cambia de estado, dependiendo de cuál de las entradas analógicas sea mayor.

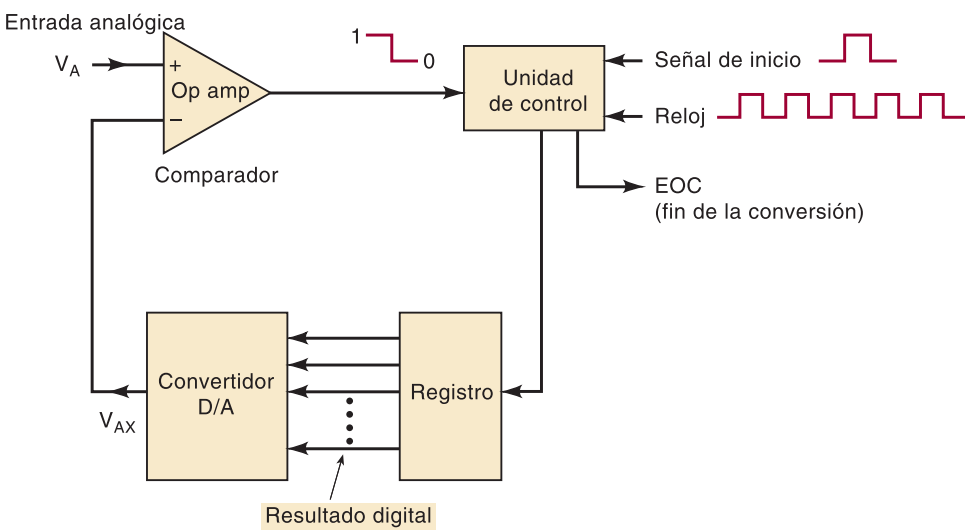


FIGURA 11-12
Diagrama general de una clase de ADCs.

La operación básica de los ADCs de este tipo consiste en los siguientes pasos:

1. Pulsar SEÑAL DE INICIO inicia la operación.
2. A una velocidad determinada por el reloj, la unidad de control modifica en forma continua el número binario que se almacena en el registro.
3. El DAC convierte el número binario del registro en un voltaje analógico, V_{AX}.
4. El comparador compara V_{AX} con la entrada analógica V_A. Mientras que $V_{AX} < V_A$, la salida del comparador permanecerá en ALTO. Cuando V_{AX} se exceda de V_A por cuando menos una cantidad igual a V_T (voltaje de umbral), la salida del comparador cambiará a BAJO y detendrá el proceso de modificación del número del registro. En este punto V_{AX} será una aproximación cercana a V_A. El número digital en el registro, que viene siendo el equivalente digital de V_{AX}, es también el equivalente digital aproximado de V_A, dentro de la resolución y la precisión del sistema.
5. La lógica de control activa la señal de fin de conversión (*EOC*) cuando se completa ésta.

Las diversas situaciones de este esquema de conversión A/D difieren principalmente en la forma en la cual la sección de control modifica de manera continua los números en el registro. En cualquier otro caso la idea básica es la misma, ya que el registro retiene la salida digital requerida cuando se completa el proceso de conversión.

PREGUNTAS DE REPASO

1. ¿Cuál es la función del comparador en el ADC?
2. ¿En dónde está el equivalente digital aproximado de V_A cuando se completa la conversión?
3. ¿Cuál es la función de la señal *EOC*?

11-9 ADC DE RAMPA DIGITAL

Una de las versiones más simples del ADC general de la figura 11-12 utiliza un contador binario como registro y permite que el reloj incremente el contador un intervalo a la vez, hasta que $V_{AX} \geqq V_A$. A este ADC se le conoce como **ADC de rampa digital** debido a que la forma de onda en V_{AX} es una rampa de intervalo por intervalo (en realidad una escalera), como la que se muestra en la figura 11-3. También se le conoce como ADC *tipo contador*.

La figura 11-13 es el diagrama para un ADC de rampa digital. Contiene un contador, un DAC, un comparador analógico y una compuerta AND de control. La salida del comparador sirve como la señal de fin de conversión $\overline{EOC}$, activa en BAJO. Si suponemos que V_A, el voltaje analógico que se va a convertir, es positivo, la operación sería la siguiente:

1. Se aplica un pulso de INICIO para restablecer el contador a 0. El nivel ALTO en INICIO también inhibe los pulsos de reloj para que no pasen a través de la compuerta AND y hacia el contador.
2. Si todos los bits de entrada son 0, la salida del DAC será $V_{AX} = 0$ V.
3. Como $V_A > V_{AX}$, la salida del comparador ($\overline{EOC}$) estará en ALTO.
4. Cuando INICIO regresa a BAJO, la compuerta AND se habilita y los pulsos de reloj pasan hacia el contador.

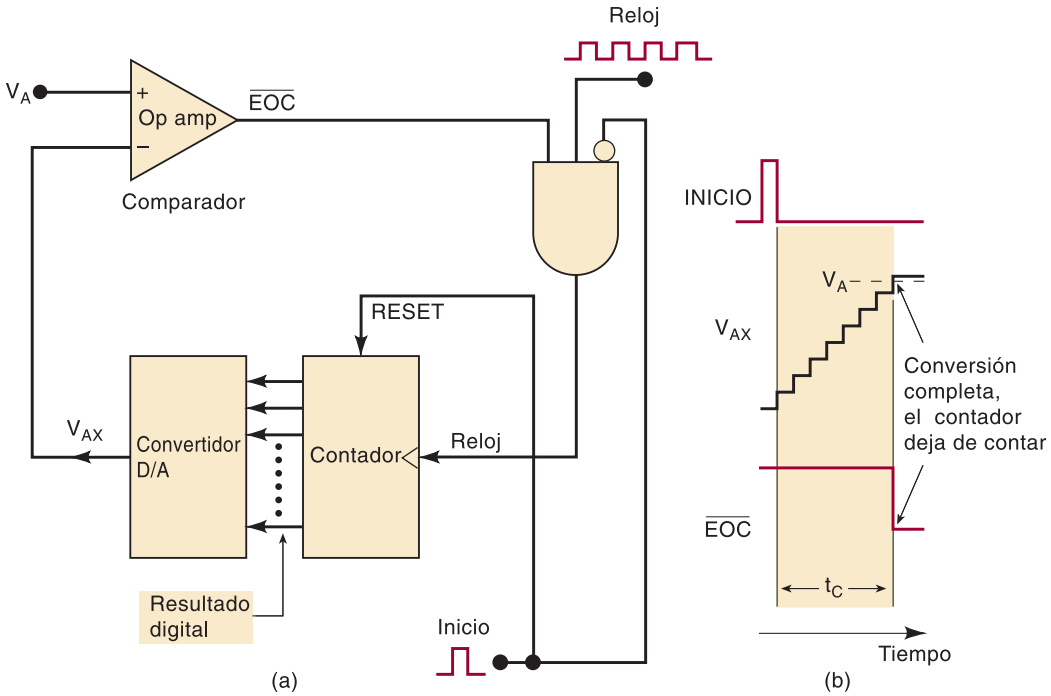


FIGURA 11-13 ADC de rampa digital.

5. A medida que el contador avanza, la salida V_{AX} del DAC se incrementa conforme a la cuenta, como se muestra en la figura 11-13(b).
6. Este proceso continúa hasta que V_{AX} llega a un valor mayor a V_A por una cantidad igual o mayor a V_T (por lo general, de 10 a 100 μV). En este punto, $\overline{EOC}$ cambiará a BAJO e inhibirá el flujo de pulsos que van hacia el contador, y éste dejará de contar.
7. El proceso de conversión está ahora completo, como lo indica la transición de ALTO a BAJO en $\overline{EOC}$ y el contenido del contador es la representación digital de V_A.
8. El contador retendrá el valor digital hasta que el siguiente pulso INICIO comience una nueva conversión.

EJEMPLO 11-12

Suponga los siguientes valores para el ADC de la figura 11-13: frecuencia de reloj = 1 MHz; V_T = 0.1 mV; el DAC tiene una salida F.S. = 10.23 V y una entrada de 10 bits. Determine los siguientes valores.

(a) El equivalente digital que se obtiene para V_A = 3.728 V.
(b) El tiempo de conversión.
(c) La resolución de este convertidor.

Solución

(a) El DAC tiene una entrada de 10 bits y una salida F.S. de 10.23 V. Por lo tanto, el número total de posibles intervalos es $2^{10} - 1 = 1023$ y, por lo tanto, el tamaño del escalón es

$$\frac{10.23\text{ V}}{1023} = 10\text{ mV}$$

Esto significa que V_{AX} se incrementa en intervalos de 10 mV a medida que el contador cuenta en forma ascendente desde 0. Como $V_A = 3.728$ V y $V_T = 0.1$ mV, V_{AX} debe llegar a 3.728 V o más antes de que el comparador cambie a BAJO. Para esto se requerirá lo siguiente:

$$\frac{3.7281\text{ V}}{10\text{ mV}} = 372.81 = 373 \text{ escalones}$$

Entonces, al final de la conversión el contador retendrá el equivalente binario de 373, que es 0101110101. El equivalente digital deseado de $V_A = 3.728$ V que produce este DAC.

(b) Se requirieron 373 intervalos para completar la conversión. Por lo tanto, se produjeron 373 pulsos de reloj a la velocidad de uno cada microsegundo. Esto nos da un tiempo total de conversión de 373 μs.

(c) La resolución de este convertidor es igual al tamaño del escalón del DAC, que es de 10 mV. Si lo expresamos como un porcentaje, es 1/1023 × 100% = 0.1%.

EJEMPLO 11-13

Para el mismo DAC del ejemplo 11-12, determine el intervalo aproximado de voltajes de entrada analógicos que producirán el mismo resultado digital de $0101110101_2 = 373_{10}$.

TABLA 11-4

Escalón	V_{AX}(V)
371	3.71
372	3.72
373	3.73
374	3.74
375	3.75

Solución

La tabla 11-4 muestra el voltaje de salida V_{AX} ideal del DAC para el paso 373 y varios valores cerca de éste. Si V_A es un poco menor que 3.72 V (por una cantidad $< V_T$), entonces $\overline{EOC}$ no cambiará a BAJO cuando V_{AX} llegue a 3.72 V, sino que cambiará a BAJO en 3.73 V. Si V_A es un poco menor que 3.73 V (por una cantidad $< V_T$), entonces $\overline{EOC}$ no cambiará a BAJO sino hasta que V_{AX} llegue a 3.74 V. Por lo tanto, mientras que V_A se encuentre en un valor aproximado entre 3.72 y 3.73 V, $\overline{EOC}$ cambiará a BAJO cuando V_{AX} llegue al intervalo de 3.73 V. El intervalo exacto de valores de V_A es:

$$3.72\text{ V} - V_T \qquad \text{a} \qquad 3.73\text{ V} - V_T$$

pero debido a que V_T es muy pequeño, basta decir que el intervalo es aproximadamente de 3.72 a 3.73 V; un escalón de10 mV, la resolución del DAC. Esto se ilustra en la figura 11-14.

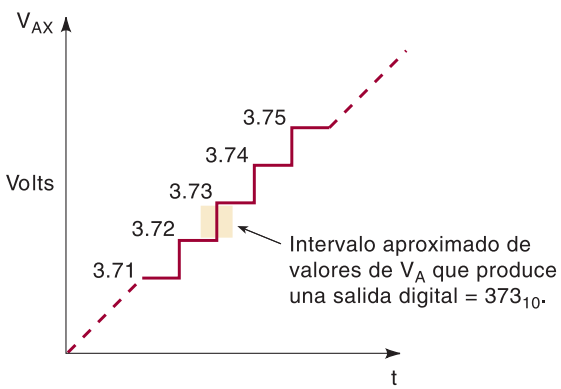


FIGURA 11-14
Ejemplo 11-13.

Resolución y precisión A/D

Es muy importante comprender los errores asociados con la realización de cualquier tipo de mediciones. Una fuente de error inevitable en el método de rampa digital es

que el tamaño del escalón o resolución del DAC interno es la unidad más pequeña de medición. Imagine tratar de medir la altura de los jugadores de básquetbol colocándolos en seguida de una escalera con intervalos de 12 pulgadas y asignándoles la altura del primer intervalo más alto que su cabeza. ¡Cualquier jugador con más de 6 pies de altura se mediría como si tuviera 7 pies! De igual forma, el voltaje de salida V_{AX} es una forma de onda de escalera que sube en intervalos discretos hasta que se excede del voltaje de entrada V_A. Al reducir el tamaño del escalón podemos reducir el error potencial, pero siempre habrá una diferencia entre la cantidad real (analógica) y el valor digital que se le asigne. A esto se le conoce como **error de cuantización**. Por lo tanto, V_{AX} es una aproximación al valor de V_A y lo mejor que podemos esperar es que V_{AX} se encuentre dentro de un intervalo menor a 10 mV de V_A si la resolución (tamaño del escalón) es de 10 mV. Este error de cuantización, que puede reducirse si se incrementa el número de bits en el contador y en el DAC, se especifica algunas veces como un error de +1 LSB, lo cual indica que el resultado podría desplazarse por un valor máximo equivalente a la ponderación del LSB.

Una práctica más común es hacer el error de cuantización simétrico alrededor de un múltiplo entero de la resolución, para hacer que sea de ±½ LSB. Para ello hay que asegurarse que la salida cambie a ½ unidad de resolución por debajo y por encima del voltaje de entrada nominal. Por ejemplo, si la resolución es de 10 mV entonces la salida A/D cambiará en teoría de 0 a 1 a 5 mV, y de 1 a 2 a 15 mV. El valor nominal (10 mV), que se representa mediante el valor digital de 1, se encuentra en teoría siempre dentro de un intervalo menor a 5 mV (½ LSB) del voltaje de entrada actual. El problema 11-28 explora un método para lograr esto. De cualquier forma, hay un pequeño intervalo de voltajes de entrada que producirán la misma salida digital.

La especificación de precisión refleja el hecho de que la salida de cualquier ADC no cambia de un valor binario al siguiente en el voltaje de entrada exacto que se haya prescrito. Algunos cambian a un voltaje un poco mayor de lo esperado, y algunos a un voltaje un poco menor. La imprecisión e inconsistencia se deben a los componentes imperfectos tales como las resistencias de precisión, los comparadores, los interruptores de corriente, etcétera. La precisión puede expresarse como un porcentaje de la escala completa, justo igual que para el DAC, pero se especifica con más frecuencia como $\pm n$ LSB, en donde n es un valor fraccionario o 1. Por ejemplo, si la precisión se especifica como ±¼ LSB con una resolución de 10 mV y suponemos que la salida deberá cambiar en teoría de 0 a 1 a 5 mV, entonces sabemos que la salida podría cambiar de 0 a 1 con cualquier voltaje de entrada entre 2.5 y 7.5 mV. En este caso estaríamos seguros de que cualquier voltaje entre 7.5 y 12.5 mV sin duda produciría el valor 1. No obstante, en el peor de los casos la salida de un 1 binario podría representar un valor nominal de 10 mV con un voltaje aplicado real de 2.5 mV, un error de ¾ de bit que es la suma del error de cuantización y la precisión.

EJEMPLO 11-14

Cierto ADC de ocho bits, similar al de la figura 11-13, tiene una entrada a escala completa de 2.55 V (es decir, V_A = 2.55 V produce una salida digital de 11111111). Tiene un error especificado de ±¼ LSB. Determine la máxima cantidad de error en la medición.

Solución

El tamaño del escalón es de $2.55\text{V}/(2^8 - 1)$, que es exactamente 10 mV. Esto significa que aunque el DAC no tenga imprecisiones, la salida V_{AX} podría desplazarse por un valor máximo de hasta 10 mV, debido a que V_{AX} sólo puede cambiar en intervalos de 10 mV; éste es el error de cuantización. El error especificado de ±¼ LSB es ¼ × 10 mV = 2.5 mV; lo que significa que el valor de V_{AX} puede desplazarse por una cantidad máxima de hasta 2.5 mV, debido a las imprecisiones de los componentes. Así, el error posible total podría ser de hasta 10 mV + 2.5 mV = 12.5 mV.

Por ejemplo, suponga que la entrada analógica es de 1.268 V. Si la salida del DAC fuera perfectamente precisa, la escalera se detendría en el intervalo 127 (1.27 V). Pero digamos que V_{AX} se desplazara por −2 mV, de forma que fuera de 1.268 V en el paso 127. Esto no sería lo bastante grande como para detener la conversión; se detendría en el intervalo 128. Por lo tanto, la salida digital sería de $10000000_2 = 128_{10}$ (que representa 12.8 V) para una entrada analógica de 1.268 V, un error de 12 mV.

Tiempo de conversión t_C

El tiempo de conversión se muestra en la figura 11-13(b) como el tiempo entre el final del pulso INICIO y la activación de la salida *EOC*. El contador empieza a contar desde cero y avanza en forma ascendente hasta que V_{AX} se excede de V_A, punto en el cual *EOC* cambia a BAJO para finalizar el proceso de conversión. Debe quedar claro que el valor del tiempo de conversión t_C depende de V_A. Un valor más grande requerirá más escalones antes de que el voltaje de la escalera exceda a V_A.

El tiempo máximo de conversión ocurrirá cuando V_A se encuentre justo debajo de la escala completa, de manera que V_{AX} deberá avanzar hasta el último intervalo para activar a *EOC*. Para un convertidor de N bits, esto será

$$t_C(\text{máx}) = (2^N - 1) \text{ ciclos de reloj}$$

Por ejemplo, el ADC en el ejemplo 11-12 tendría un tiempo máximo de conversión de

$$t_C(\text{máx}) = (2^{10} - 1) \times 1\ \mu\text{s} = 1023\ \mu\text{s}$$

Algunas veces se especifica el tiempo de conversión promedio; éste equivale a la mitad del tiempo máximo de conversión. Para el convertidor de rampa digital, esto sería

$$t_C(\text{prom}) = \frac{t_C(\text{máx})}{2} \approx 2^{N-1} \text{ ciclos de reloj}$$

La principal desventaja del método de rampa digital es que, en esencia, el tiempo de conversión se duplica por cada bit que se agrega al contador, por lo que la resolución sólo puede mejorarse a costa de un t_C más largo. Esto hace que este tipo de ADC sea inadecuado para aplicaciones en las que deban realizarse conversiones A/D repetitivas de una señal analógica que cambie con mucha rapidez. No obstante, para las aplicaciones de baja velocidad la relativa simpleza del convertidor de rampa digital es una ventaja en comparación con los ADCs más complejos de mayor velocidad.

EJEMPLO 11-15

¿Qué pasará con la operación de un ADC de rampa digital si la entrada analógica V_A es mayor que el valor de escala completa?

Solución

De la figura 11-13 debe quedar claro que la salida del comparador nunca cambiará a BAJO, ya que el voltaje de escalera no puede excederse de V_A. Así, se aplicarán pulsos en forma continua al contador, de manera que éste contará en forma repetitiva desde 0 al valor máximo, se reciclará de vuelta a 0, contará en forma ascendente, y así en lo sucesivo. Esto producirá formas de onda de escalera repetitivas en V_{AX} que vayan de 0 hasta la escala completa, y esto continuará hasta que V_A disminuya por debajo de la escala completa.

PREGUNTAS DE REPASO

1. Describa la operación básica del ADC de rampa digital.
2. Explique lo que es el *error de cuantización*.
3. ¿Por qué el tiempo de conversión se incrementa con el valor del voltaje de entrada analógico?
4. *Verdadero o falso:* Si todo lo demás es igual, un ADC de rampa digital de 10 bits tendrá una mejor resolución, pero un tiempo de conversión mayor, que un ADC de ocho bits.
5. Mencione una ventaja y una desventaja de un ADC de rampa digital.
6. Para el convertidor del ejemplo 11-12, determine la salida digital cuando V_A = 1.345 V. Repita para V_A = 1.342 V.

11-10 ADQUISICIÓN DE DATOS

Existen muchas aplicaciones en las que los datos analógicos deben *digitalizarse* (convertirse en digitales) y transferirse hacia la memoria de una computadora. El proceso mediante el cual la computadora adquiere estos datos analógicos digitalizados se llama *adquisición de datos*. Al proceso de adquirir el valor de un solo punto de datos se le conoce como **muestreo** de la señal analógica y, por lo general, a ese punto de datos se le conoce como *muestra*. La computadora puede realizar varias operaciones distintas con los datos, dependiendo de la aplicación. En una aplicación de almacenamiento tal como la grabación de audio digital, de video o de un osciloscopio digital, la microcomputadora interna almacenará los datos y después los transferirá hacia un DAC más adelante para reproducir la señal analógica original. En una aplicación de control de un proceso, la computadora puede examinar los datos o realizar cálculos sobre ellos para determinar qué salidas de control se van a generar.

La figura 11-15(a) muestra cómo se conecta una microcomputadora a un ADC de rampa digital con el propósito de adquirir datos. La computadora genera los pulsos de INICIO que comienzan cada nueva conversión A/D. La señal $\overline{EOC}$ (fin de conversión) del ADC se alimenta a la computadora, que a su vez monitorea $\overline{EOC}$ para averiguar cuando se complete la conversión A/D actual; después transfiere los datos digitales de la salida ADC hacia su memoria.

Las formas de onda de la figura 11-15(b) ilustran cómo la computadora adquiere una versión digital de la señal analógica V_A. La forma de onda de escalera de V_{AX} que se genera en forma interna para el ADC se muestra superimpuesta en la forma de onda de V_A para fines ilustrativos. El proceso comienza en t_0, cuando la computadora genera un pulso INICIO para comenzar un ciclo de conversión A/D. La conversión se completa en t_1, cuando la escalera primero se excede de V_A y $\overline{EOC}$ cambia a BAJO. Esta NGT en $\overline{EOC}$ indica a la computadora que el ADC tiene una salida digital que ahora representa el valor de V_A en el punto a, y la computadora cargará estos datos en su memoria.

La computadora genera un nuevo pulso INICIO poco después de t_1 para iniciar un segundo ciclo de conversión. Observe que esto restablece la escalera a 0 y $\overline{EOC}$ regresa a ALTO debido a que el pulso INICIO restablece el contador en el ADC. La segunda conversión termina en t_2, cuando la escalera se excede de nuevo de V_A. Después la computadora carga los datos digitales que corresponden al punto b en su memoria. Estos pasos se repiten en t_3, t_4 y así en lo sucesivo.

El proceso mediante el cual la computadora genera un pulso INICIO, monitorea $\overline{EOC}$ y carga los datos del ADC en la memoria se lleva a cabo bajo el control de un programa que ejecuta la computadora. Este programa de adquisición de datos determinará cuántos puntos de datos (muestras) de la señal analógica se almacenarán en la memoria de la computadora.

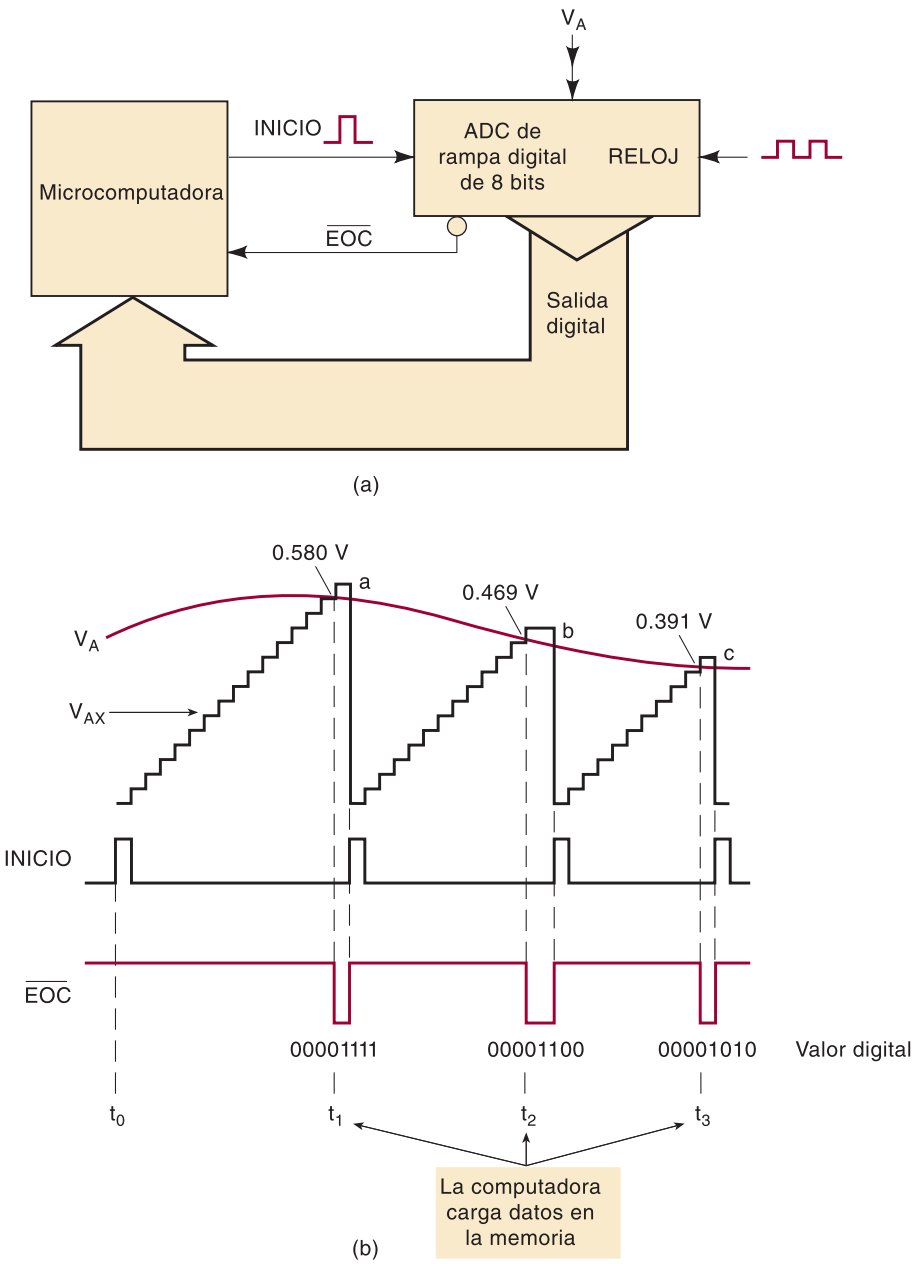


FIGURA 11-15
(a) Sistema típico de adquisición de datos de una computadora; (b) formas de onda que muestran cómo la computadora inicia cada nuevo ciclo de conversión y después carga los datos digitales en la memoria al final de la conversión.

Reconstrucción de una señal digitalizada

En la figura 11-15(b) el ADC opera a su máxima velocidad, ya que se genera un nuevo pulso INICIO justo después de que la computadora adquiere los datos de salida del ADC de la conversión anterior. Observe que los tiempos de conversión no son constantes, ya que el valor de entrada analógico está cambiando. El problema con este método para almacenar una forma de onda es que, para poder reconstruir la forma de onda, tendríamos que conocer el punto en el tiempo en el que se va a trazar cada uno de los valores de datos. Por lo general, cuando se almacena una forma de onda digitalizada las muestras se toman a intervalos fijos, a una velocidad que sea por lo menos dos veces mayor que la frecuencia más alta en la señal analógica. El sistema digital almacenará la forma de onda como una lista de valores de datos de muestra. La tabla 1-5 muestra la lista de datos que se almacenarían si se digitalizara la señal en la figura 11-16(a).

TABLA 11-5 Muestras de datos digitalizados.

Punto	Voltaje real (V)	Equivalente digital
a	1.22	01111010
b	1.47	10010011
c	1.74	10101110
d	1.70	10101010
e	1.35	10000111
f	1.12	01110000
g	0.91	01011011
h	0.82	01010010

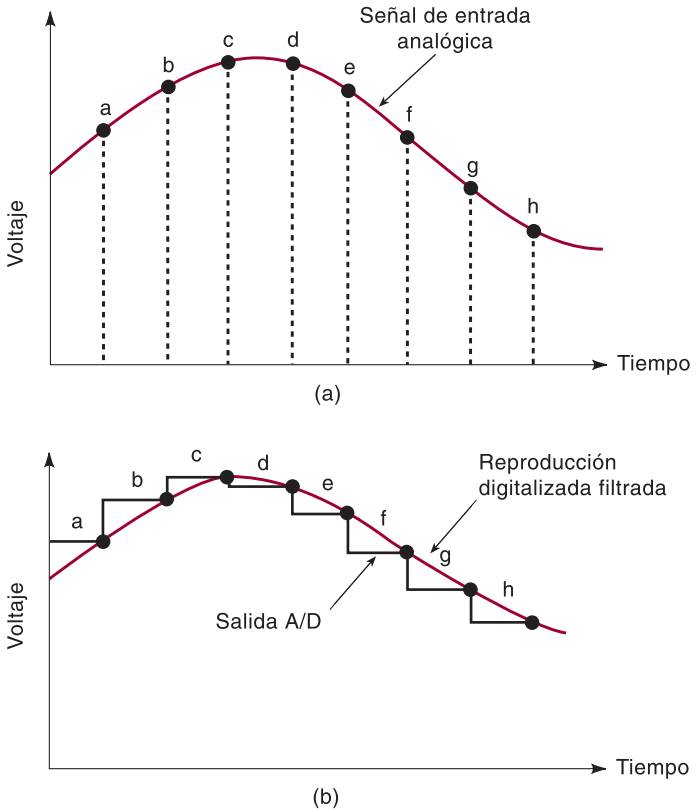


FIGURA 11-16
(a) Digitalización de una señal analógica; (b) reconstrucción de la señal a partir de los datos digitales.

En la figura 11-16(a) podemos ver cómo el ADC realiza conversiones en forma continua para digitalizar la señal de entrada en los puntos *a*, *b*, *c*, *d*, y así sucesivamente. Si se utilizan estos datos digitales para reconstruir la señal, el resultado se verá como en la figura 11-16(b). La línea negra representa la forma de onda de voltaje que saldría del convertidor D/A. La línea roja seria el resultado de pasar la señal a través de un filtro RC simple de paso bajo. Podemos ver que es una reproducción bastante buena de la señal analógica original, ya que la señal analógica no tiene cambios rápidos entre los puntos digitalizados. Si la señal analógica tuviera variaciones de alta frecuencia, el ADC no podría seguir las variaciones y la versión reproducida sería mucho menos precisa.

Traslape espectral

La meta obvia en la reconstrucción de señales es que la reconstrucción sea casi idéntica a la señal analógica original. Para poder evitar la pérdida de información,

según lo demostrado por un hombre llamado Harry Nyquist, la señal entrante debe muestrearse a una velocidad mayor que el doble del componente de frecuencia más alta en la señal entrante. Por ejemplo, si usted está muy seguro de que la frecuencia más alta en un sistema de audio será menor de 10 kHz, debe muestrear la señal de audio a 20,000 muestras por segundo para poder reconstruir la señal. La frecuencia a la cual se toman las muestras se conoce como **frecuencia de muestreo, F_S**. ¿Qué cree usted que ocurrirá si por alguna razón hay un tono de 12 kHz presente en la señal de entrada? Por desgracia el sistema *no* nada más lo ignora porque sea demasiado alto, sino que se produce un fenómeno conocido como *traslape espectral*. Un **traslape** de señal se produce cuando se muestrea la señal a una velocidad menor que la velocidad mínima identificada por Nyquist (el doble de la frecuencia entrante más alta). En este caso, cualquier frecuencia por encima de 10 kHz producirá una frecuencia *fantasma*. Esta frecuencia es siempre la diferencia entre cualquier múltiplo entero de la frecuencia de muestreo F_S (20 kHz) y la frecuencia entrante que se está digitalizando (12 kHz). En vez de escuchar un tono de 12 kHz en la señal reconstruida, escucharía un tono de 8 kHz que no estaba en la señal original.

Para ver cómo se puede producir la señal fantasma, considere la onda senoidal en la figura 11-17. Su frecuencia es de 1.9 kHz. Los puntos muestran en dónde se muestrea la forma de onda cada 500 µs (F_S = 2 kHz). Si conectamos los puntos que integran la forma de onda muestreada, descubriremos que forman una onda cosenoidal que tiene un periodo de 10 ms y una frecuencia de 100 Hz. Esto demuestra que la frecuencia de traslape es igual a la diferencia entre la frecuencia de muestreo y la frecuencia entrante. Si pudiéramos escuchar la salida que se produce de esta adquisición de datos, no sonaría como 1.9 kHz, sino como 100 Hz.

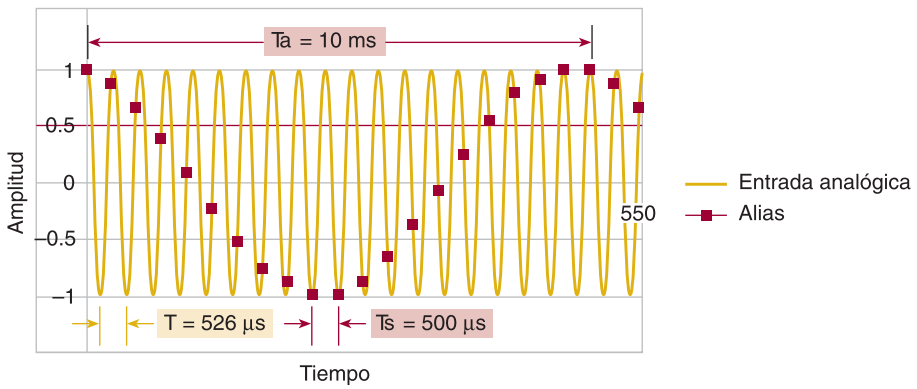


FIGURA 11-17 Una señal fantasma debido al submuestreo.

El problema con el **submuestreo** ($F_S < 2$ *Fent* máxima) es que el sistema digital no tiene idea de que en realidad había una frecuencia más alta en la entrada. Tan sólo muestrea la entrada y almacena los datos. Cuando reconstruye la señal la frecuencia fantasma (100 Hz) está presente, la frecuencia de 1.9 kHz original no está y la señal reconstruida no suena igual. Esto explica el por qué un sistema de adquisición de datos no debe permitir que se coloquen frecuencias mayores de la mitad de F_S en la entrada.

PREGUNTAS DE REPASO

1. ¿Qué es *digitalizar una señal*?
2. Describa los pasos en un proceso de adquisición de datos computacionales.
3. ¿Cuál es la mínima frecuencia de muestreo necesaria para reconstruir una señal analógica?
4. ¿Qué ocurre si la señal se muestrea a una frecuencia menor que la mínima frecuencia que se determinó en la pregunta 3?

11-11 ADC DE APROXIMACIONES SUCESIVAS

El **convertidor de aproximaciones sucesivas** es uno de los tipos de ADC más utilizados. Tiene circuitos más complejos que el ADC de rampa digital, pero un tiempo de conversión mucho más corto. Además, los convertidores de aproximaciones sucesivas (SACs) tienen un valor fijo para el tiempo de conversión, el cual no depende del valor de la entrada analógica.

El arreglo básico que se muestra en la figura 11-18(a) es similar al del ADC de rampa digital. Sin embargo, el SAC no utiliza un contador para proporcionar la entrada al bloque del DAC, sino que utiliza un registro. La lógica de control modifica el contenido del registro bit por bit, hasta que los datos del registro sean el equivalente digital de la entrada analógica V_A dentro de la resolución del convertidor. La secuencia básica de operación se da mediante el diagrama de flujo de la figura 11-18(b). Seguiremos este diagrama de flujo a medida que avancemos por el ejemplo que se ilustra en la figura 11-19.

Para este ejemplo hemos elegido un convertidor simple de cuatro bits con un tamaño de escalón de 1 V. Aun y cuando la mayoría de los SACs prácticos tienen muchos más bits y una resolución más pequeña que el de nuestro ejemplo, la operación será exactamente la misma. En este punto debemos ser capaces de determinar que los cuatro bits de registro que alimentan al DAC tienen ponderaciones de 8, 4, 2 y 1 V, en forma respectiva.

Supongamos que la entrada analógica es $V_A = 10.4$ V. La operación comienza cuando la lógica de control borra todos los bits del registro para dejarlos en 0, de manera que $Q_3 = Q_2 = Q_1 = Q_0 = 0$. Expresaremos esto como [Q] = 0000. Esto hace que la salida del DAC $V_{AX} = 0$, como se indica en el tiempo t_0 en el diagrama de sincronización de la figura 11-19. Cuando $V_{AX} < V_A$, la salida del comparador está en ALTO.

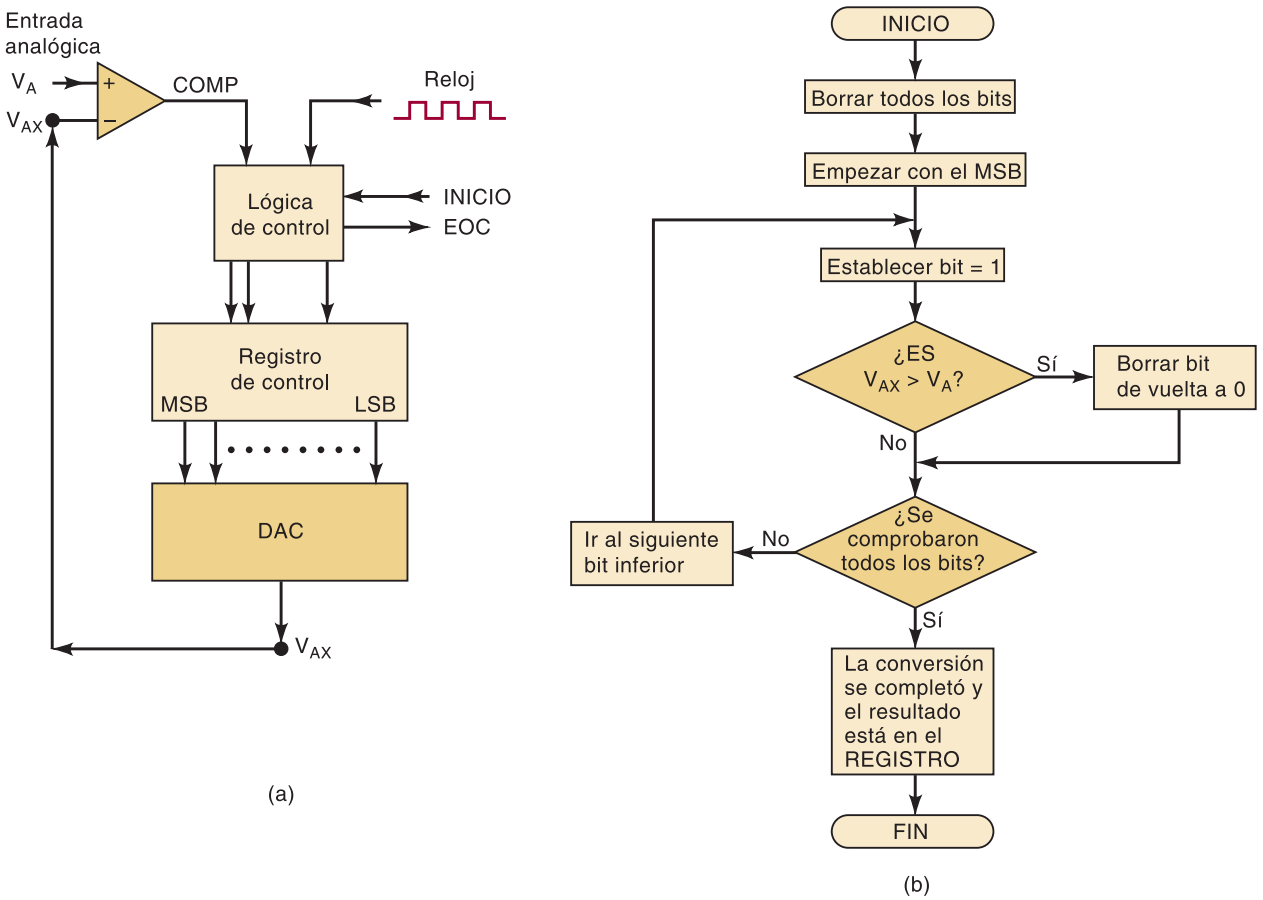


FIGURA 11-18 ADC de aproximaciones sucesivas: (a) diagrama de bloques simplificado; (b) diagrama de flujo de la operación.

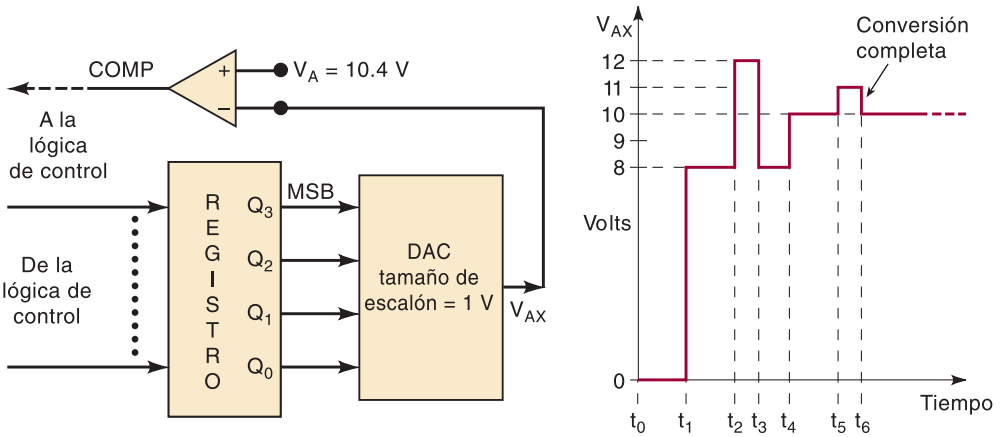


FIGURA 11-19
Ilustración de la operación de un SAC de cuatro bits mediante el uso de un tamaño de escalón de DAC de 1 V y V_A = 10.4 V.

En el siguiente intervalo (tiempo t_1) la lógica de control establece el MSB del registro en 1, de manera que [Q] = 1000. Esto produce un V_{AX} = 8 V. Como $V_{AX} < V_A$, la salida COMP aún se encuentra en ALTO. Este nivel ALTO indica a la lógica de control que el ajuste del MSB no hizo que V_{AX} se excediera de V_A, por lo cual el MSB se mantiene en 1.

Ahora la lógica de control procede con el siguiente bit inferior, Q_2, al que en 1 para producir [Q] = 1100 y V_{AX} = 12 V en el tiempo t_2. Como $V_{AX} > V_A$, la salida COMP cambia a BAJO. Este nivel BAJO indica a la lógica de control que el valor de V_{AX} es demasiado grande, por lo que la lógica de control procede a borrar Q_2 de vuelta a 0 en t_3. Por lo tanto, en t_3 el contenido del registro se regresa a 1000 y V_{AX} se regresa a 8 V.

El siguiente intervalo ocurre en t_4, en donde la lógica de control establece el siguiente bit inferior Q_1, de manera que [Q] = 1010 y V_{AX} = 10 V. Cuando $V_{AX} < V_A$, COMP está en ALTO e indica a la lógica de control que mantenga a Q_1 en 1.

El intervalo final ocurre en t_5, en donde la lógica de control establece el siguiente bit inferior Q_0, de manera que [Q] = 1011 y V_{AX} = 11 V. Como $V_{AX} > V_A$, COMP cambia a BAJO para indicar que V_{AX} es demasiado grande y la lógica de control borra a Q_0 de vuelta a 0 en t_6.

En este punto se han procesado todos los bits del registro, la conversión está completa y la lógica de control activa su salida $\overline{EOC}$ para indicar que el equivalente digital de V_A se encuentra ahora en el registro. Para este ejemplo, la salida digital para V_A = 10.4 V es [Q] = 1010. Observe que en realidad 1010 es equivalente a 10 V, que es un valor *menor que* la entrada analógica; ésta es una característica del método de aproximaciones sucesivas. Recuerde que en el método de la rampa digital, la salida digital siempre era equivalente a un voltaje que se encontraba en el intervalo por encima de V_A.

EJEMPLO 11-16

Un SAC de ocho bits tiene una resolución de 20 mV. ¿Cuál será su salida digital para una entrada analógica de 2.17 V?

Solución

$$2.17\text{ V}/20\text{ mV} = 108.5$$

de manera que el intervalo 108 produciría un V_{AX} = 2.16 V y el intervalo 109 produciría 2.18 V. El SAC siempre produce un V_{AX} final que se encuentra en el intervalo *por debajo* de V_A. Por lo tanto, para el caso de V_A = 2.17 V, el resultado digital sería $108_{10} = 01101100_2$.

Tiempo de conversión

En la operación que acabamos de describir, la lógica de control analiza cada bit del registro, lo establece en 1, decide si lo va a mantener o no en 1 y pasa al siguiente

bit. El procesamiento de cada bit requiere un ciclo de reloj, de manera que el tiempo de conversión total para un SAC de N bits será de N ciclos de reloj. Esto es,

$$t_C \text{ para un SAC} = N \times 1 \text{ ciclo de reloj}$$

Este tiempo de conversión será el mismo *sin importar el valor de* V_A, ya que la lógica de control debe procesar cada bit para ver si se necesita o no un 1.

EJEMPLO 11-17

Compare los tiempos máximos de conversión de un ADC de rampa digital de 10 bits y un ADC de aproximaciones sucesivas de 10 bits si ambos utilizan una frecuencia de reloj de 500 kHz.

Solución

Para el convertidor de rampa digital, el tiempo máximo de conversión es

$$(2^N - 1) \times (1 \text{ ciclo de reloj}) = 1023 \times 2\ \mu\text{s} = 2046\ \mu\text{s}$$

Para un convertidor de aproximaciones sucesivas, el tiempo de conversión siempre es de 10 periodos de reloj o

$$10 \times 2\mu\text{s} = 20\ \mu\text{s}$$

Por lo tanto, es aproximadamente 100 veces más rápido que el convertidor de rampa digital.

Como los SACs tienen tiempos de conversión relativamente rápidos, su uso en las aplicaciones de adquisición de datos permitirá adquirir más valores de datos en un intervalo dado. Esta característica puede ser muy importante cuando los datos analógicos cambian a una velocidad relativamente rápida.

Como muchos SACs están disponibles en CI, rara vez es necesario diseñar los circuitos de la lógica de control por lo que no los cubriremos aquí. Para aquellos que estén interesados en los detalles acerca de la lógica de control, muchos libros de datos de los fabricantes proporcionarán los detalles suficientes.

Un CI real: el ADC0804, un ADC de aproximaciones sucesivas

Los ADCs están disponibles a través de varios fabricantes de CIs con una amplia variedad de características de operación. Aquí daremos un vistazo a uno de los dispositivos más populares para darnos una idea de lo que en realidad se utiliza en las aplicaciones de sistema. La figura 11-20 es el diagrama de terminales para el ADC0804, el cual es un CI CMOS de 20 terminales que realiza la conversión A/D mediante el método de aproximaciones sucesivas. Algunas de sus características importantes son:

- Tiene dos entradas analógicas, $V_{\text{ENT}}(+)$ y $V_{\text{ENT}}(-)$, para permitir **entradas diferenciales**. En otras palabras, la entrada analógica real V_{ENT} es la diferencia en los voltajes que se aplican a estas terminales [V_{ENT} analógico = $V_{\text{ENT}}(+) - V_{\text{ENT}}(-)$]. En las mediciones de un solo extremo, es decir con referencia a tierra, la entrada analógica se aplica a $V_{\text{ENT}}(+)$, mientras que $V_{\text{ENT}}(-)$ se conecta a tierra analógica. Durante la operación normal, el convertidor utiliza $V_{CC} = +5$ V como su voltaje de referencia y la entrada analógica puede variar desde 0 hasta 5 V.
- Convierte el voltaje de entrada analógica diferencial en una salida digital con búfer triestado de ocho bits. Los circuitos internos son un poco más complejos

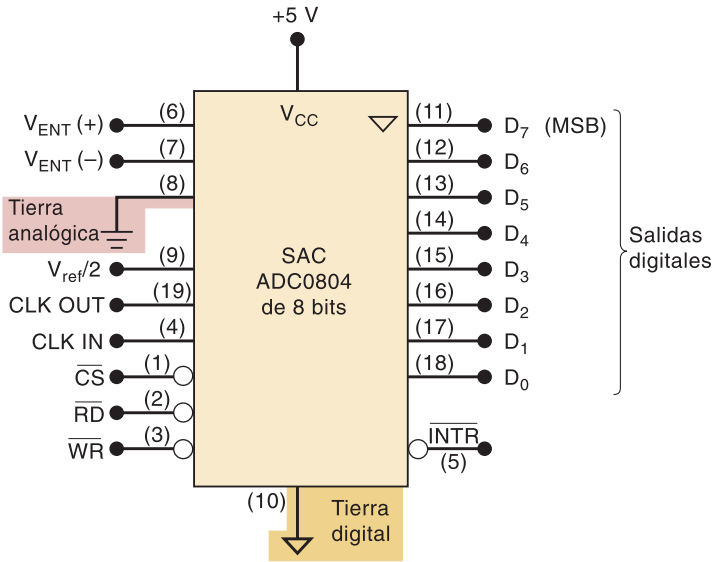


FIGURA 11-20 ADC de aproximaciones sucesivas ADC0804 de ocho bits con salidas triestado. Los números entre paréntesis indican los números de terminales del CI.

que los que se describen en la figura 11-19, para poder hacer que las transiciones entre los valores de salida ocurran en el valor nominal ± ½ LSB. Por ejemplo, con una resolución de 10 mV la salida A/D cambiaría de 0 a 1 a los 5 mV, de 1 a 2 a los 15 mV y así sucesivamente. Para este convertidor, la resolución se calcula como $V_{REF}/256$; cuando $V_{REF} = 5.00$ V, la resolución es de 19.53 mV. La entrada nominal a escala completa es de $255 \times 19.53 = 4.98$ V, lo cual debería producir una salida de 11111111. Este convertidor producirá como salida el valor 11111111 para cualquier entrada analógica entre los valores aproximados de 4.971 V y 4.990 V.

- Tiene un circuito generador de reloj interno, el cual produce una frecuencia de $f = 1/(1.1RC)$, en donde R y C son los valores de los componentes conectados externamente. Una frecuencia de reloj típica es de 606 kHz, utilizando $R = 10$ kΩ y $C = 150$ pF. Si lo desea puede utilizar una señal de reloj externa; para ello hay que conectarla a la terminal CLK IN.
- Si utilizamos una frecuencia de reloj de 606 kHz, el tiempo de conversión es de aproximadamente 100 μs.
- Tiene conexiones a tierra separadas para los voltajes digitales y analógicos. La terminal 8 es la tierra analógica que se conecta al punto de referencia común del circuito analógico que genera el voltaje analógico. La terminal 10 es la tierra digital que utilizan todos los dispositivos digitales en el sistema. (Observe los distintos símbolos utilizados para las distintas tierras.) La tierra digital es de por sí ruidosa, debido a los rápidos cambios de corriente que se producen a medida que los dispositivos digitales cambian de estado. Aunque no es necesario utilizar una tierra analógica separada, si lo hacemos nos aseguramos de evitar que el ruido de la tierra digital provoque un cambio prematuro en el comparador analógico dentro del ADC.

Este CI está diseñado para integrarse con facilidad al bus de datos de un microprocesador. Por esta razón, los nombres de algunas de las entradas y salidas del ADC0804 se basan en funciones comunes para todos los sistemas basados en microprocesador. Las funciones de estas entradas y salidas se definen de la siguiente manera:

- $\overline{CS}$ (Selección de chip). Esta entrada debe encontrarse en su estado activo en BAJO para que las entradas $\overline{RD}$ o $\overline{WR}$ tengan algún efecto. Cuando $\overline{CS}$ está en ALTO, las salidas digitales se encuentran en el estado Hi-Z, por lo que no puede realizarse ninguna conversión.

- $\overline{RD}$ (LECTURA). Esta entrada se utiliza para habilitar los búferes de salida digitales. Cuando $CS = \overline{RD}$ = BAJO, las terminales de salida digitales tendrán niveles lógicos que representen los resultados de la *última* conversión A/D. Después la microcomputadora podrá *leer* (obtener) este valor digital a través del bus de datos del sistema.
- $\overline{WR}$ (ESCRITURA). Se aplica un pulso BAJO a esta entrada para indicar el comienzo de una nueva conversión. En realidad ésta es una entrada de inicio de conversión. Se llama entrada **WRITE** porque en una aplicación común, la microcomputadora genera un pulso WRITE (similar al que se utiliza para escribir a la memoria) que controla esta entrada.
- $\overline{INTR}$ (INTERRUPCIÓN). Esta señal de salida cambiará al nivel ALTO cuando comience una conversión y regresará a BAJO para indicar el final de la conversión. En realidad es una señal de salida de fin de conversión, pero se le llama INTERRUPT debido a que en una situación ordinaria se envía a la entrada de interrupción de un microprocesador para obtener su atención y hacerle saber que los datos del ADC están listos para ser leídos.
- $V_{ref}/2$. Ésta es una entrada opcional que puede utilizarse para reducir el voltaje de referencia interno y por consecuencia cambiar el intervalo de entrada analógico que puede manejar el convertidor. Cuando esta entrada se deja desconectada, permanece en 2.5 V ($V_{CC}/2$) debido a que V_{CC} se utiliza como el voltaje de referencia. Al conectar un voltaje externo a esta terminal se cambia la referencia interna al doble de ese voltaje, y el intervalo de entrada analógico se cambia de manera acorde (consulte la tabla 11-6).
- *CLK OUT*. Se conecta una resistencia a esta terminal para utilizar el reloj interno. La señal de reloj aparece en esta terminal.
- *CLK IN*. Se utiliza para la entrada de reloj externo, o para conectar un capacitor cuando se utiliza el reloj interno.

TABLA 11-6

$V_{ref}/2$	Intervalo de entrada analógico (V)	Resolución (mV)
Abierto	0–5	19.5
2.25	0–4.5	17.6
2.0	0–4	15.6
1.5	0–3	11.7

La figura 11-21(a) muestra una conexión típica del ADC0804 con una microcomputadora, en una aplicación de adquisición de datos. Para controlar el momento en que se va a realizar una conversión, la microcomputadora genera las señales $\overline{CS}$ y $\overline{WR}$. Después, para adquirir los datos de salida del ADC, genera las señales $\overline{CS}$ y $\overline{RD}$ después de detectar una NGT en $\overline{INTR}$, lo cual indica el fin de la conversión. Las formas de onda de la figura 11-21(b) muestran la actividad de las señales durante el proceso de adquisición de datos. Observe que $\overline{INTR}$ cambia a ALTO cuando $\overline{CS}$ y $\overline{WR}$ están en BAJO, pero el proceso de conversión no comienza sino hasta que *WR* regresa a ALTO. Observe además que las líneas de datos de salida del ADC se encuentran en su estado Hi-Z hasta que la microcomputadora activa a $\overline{CS}$ y a $\overline{RD}$; en ese punto se habilitan los búferes de datos del ADC para que se envíen los datos del ADC a la microcomputadora a través del bus de datos. Las líneas de datos regresan al estado Hi-Z cuando *CS* o *RD* regresan al nivel ALTO.

En esta aplicación del ADC0804, la señal de entrada varía sobre un intervalo de 0.5 a 3.5 V. Para poder hacer un uso completo de la resolución de ocho bits, la conversión A/D debe concordar con las especificaciones de la señal analógica. En este caso, el intervalo a escala completa es de 3.0 V. Sin embargo, se desplaza con respecto a tierra por 0.5 V. Este desplazamiento de 0.5 V se aplica a la entrada negativa $V_{ENT}(-)$, con lo cual esto se establece como la referencia de valor 0. El intervalo

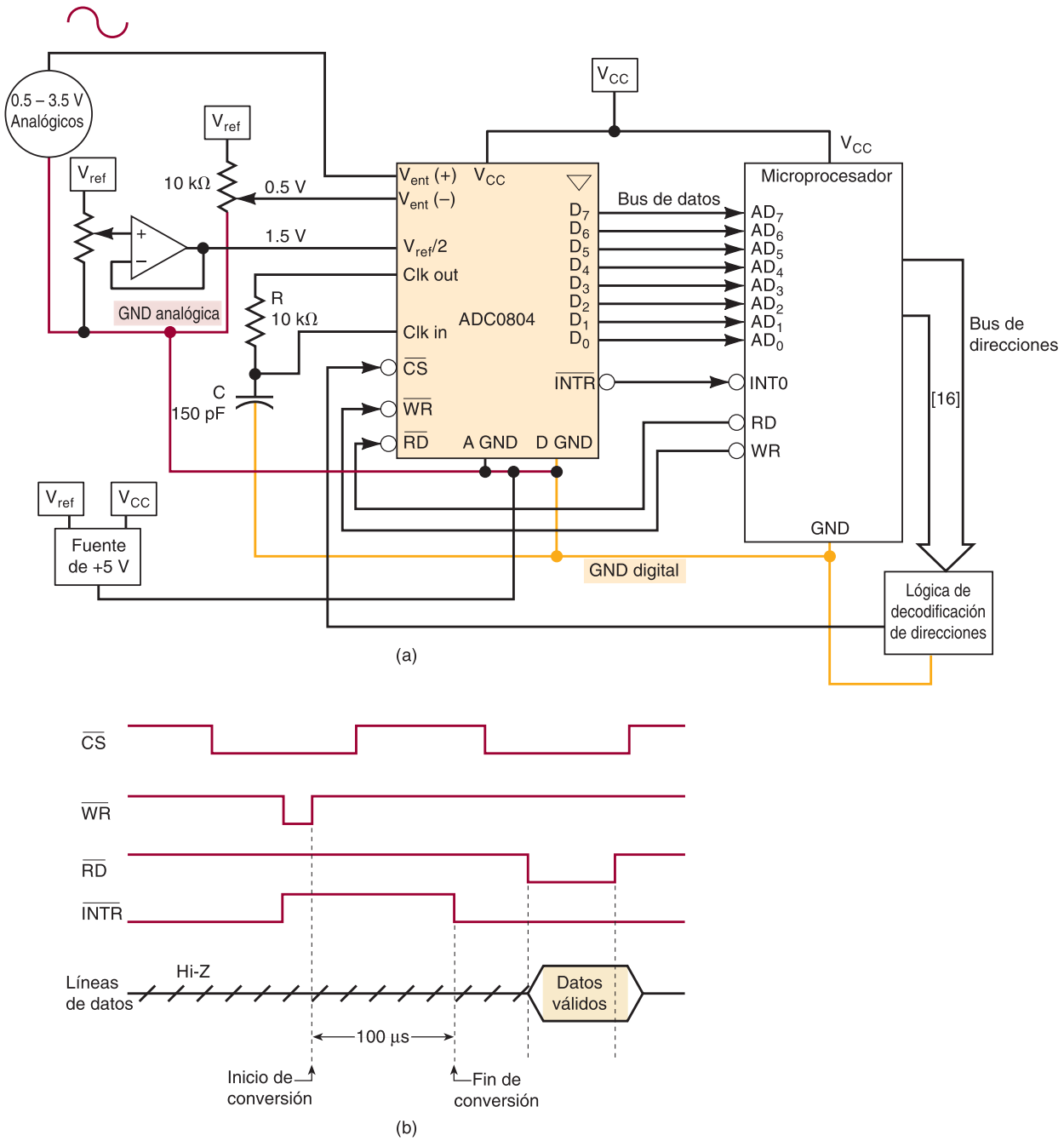


FIGURA 11-21 (a) Una aplicación del ADC0804; (b) señales de sincronización típicas durante la adquisición de datos.

de 3.0 V se establece al aplicar 1.5 V a $V_{ref}/2$, con lo cual se establece V_{ref} como 3.0 V. Una entrada de 0.5 V producirá un valor digital de 00000000 y una entrada de 3.5 V (o cualquier valor por encima de 3.482) producirá 11111111.

Otra de las principales preocupaciones al integrar señales digitales y analógicas es el *ruido*. Observe que las rutas hacia tierra digital y analógica están separadas. Las dos tierras se conectan entre sí en un punto que está muy cerca del convertidor A/D. Una ruta de muy baja resistencia conecta este punto en forma directa con la terminal negativa de la fuente de energía. También es conveniente enrutar las líneas de suministro positivas en forma separada para los dispositivos digitales y

los analógicos, y utilizar todos los capacitares de desacoplamiento (0.01 μF) que se puedan, desde la conexión al suministro de energía de cada chip hasta tierra.

PREGUNTAS DE REPASO

1. ¿Cuál es la principal ventaja de un SAC, en comparación un ADC de rampa digital?
2. ¿Cuál es la principal desventaja en comparación con el convertidor de rampa digital?
3. *Verdadero o falso:* el tiempo de conversión para un SAC se incrementa a medida que aumenta el voltaje analógico.
4. Responda a las siguientes preguntas en relación con el ADC0804.
   (a) ¿Cuál es su resolución en bits?
   (b) ¿Cuál es el intervalo normal de voltaje de entrada analógico?
   (c) Describa las funciones de las entradas $\overline{CS}$, $\overline{WR}$ y $\overline{RD}$.
   (d) ¿Cuál es la función de la salida $\overline{INTR}$?
   (e) ¿Por qué tiene dos tierras separadas?
   (f) ¿Cuál es el propósito de $V_{ENT}(-)$?

11-12 ADCS TIPO FLASH

El **convertidor Flash** es el ADC de mayor velocidad disponible, pero requiere muchos más circuitos que los otros tipos. Por ejemplo, un ADC Flash de seis bits requiere 63 comparadores analógicos, mientras que una unidad de ocho bits requiere 255 comparadores, y un convertidor de diez bits requiere 1023 comparadores. El extenso número de comparadores ha limitado el tamaño de los convertidores Flash. Por lo general, los convertidores Flash de CI están disponibles en unidades de dos a ocho bits, y la mayoría de los fabricantes ofrecen también unidades de nueve y diez bits.

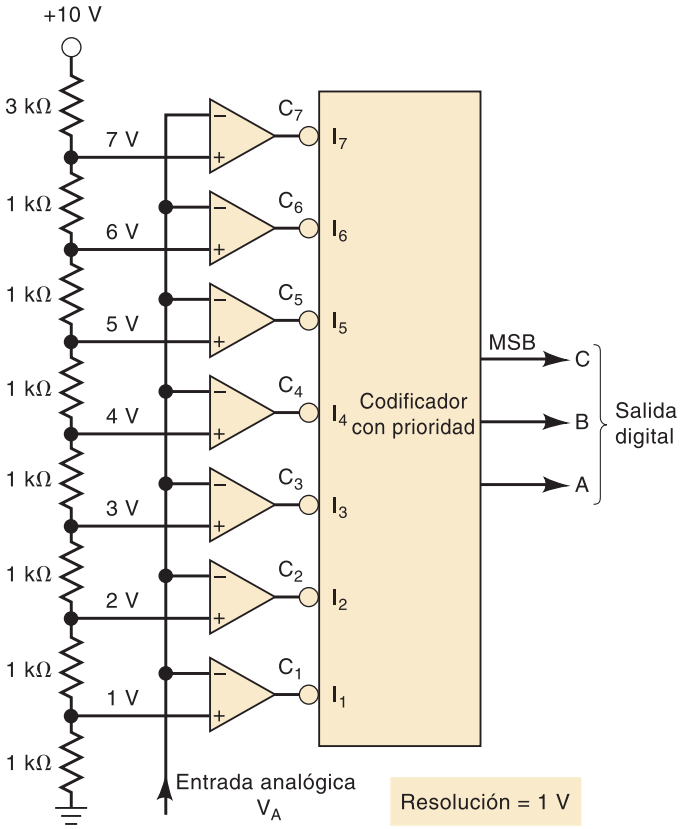
Describiremos el principio de operación de un convertidor Flash de tres bits, para poder mantener el circuito en un nivel con el que se pueda trabajar. Una vez que comprenda el funcionamiento del convertidor de tres bits, le será fácil extender la idea básica a los convertidores Flash de más bits.

El convertidor Flash de la figura 11-22(a) tiene una resolución de tres bits y una resolución de 1 V. El divisor de voltaje establece niveles de referencia para cada comparador, de manera que hay siete niveles que corresponden a 1 V (ponderación del LSB), 2 V, 3V, …, y 7 V (escala completa). La entrada analógica V_A se conecta a la otra entrada de cada comparador.

Cuando $V_A < 1$ V, todas las salidas del comparador (de C_1 a C_7) estarán en ALTO. Cuando $V_A > 1$V, una o más de las salidas del comparador estarán en BAJO. Las salidas del comparador se alimentan a un codificador con prioridad activo en BAJO, el cual genera una salida binaria que corresponde a la salida del comparador con mayor numeración que esté en BAJO. Por ejemplo, cuando V_A esté entre 3 y 4 V, las salidas C_1, C_2 y C_3 estarán en BAJO y las demás en ALTO. El codificador con prioridad sólo responderá al nivel BAJO en C_3 y producirá una salida binaria $CBA = 011$, la cual representa el equivalente digital de V_A, dentro de la resolución de 1 V. Cuando V_A es mayor de 7 V, todas las salidas de C_1 a C_7 estarán en BAJO y el codificador producirá $CBA = 111$ como el equivalente digital de V_A. La tabla de la figura 11-22(b) muestra las respuestas para todos los posibles valores de la entrada analógica.

El ADC tipo Flash de la figura 11-22 tiene una resolución de 1 V, ya que la entrada analógica debe cambiar por 1 V para poder llevar la salida digital hacia su siguiente valor. Para lograr resoluciones más finas tendríamos que incrementar el número de niveles de voltaje de entrada (es decir, utilizar más resistencias divisoras de voltaje) y el número de comparadores. Por ejemplo, un convertidor Flash de ocho bits reque-

FIGURA 11-22
(a) ADC tipo Flash de tres bits; (b) tabla de verdad.



(a)

Entrada analógica	Salidas del comparador							Salidas digitales		
V_A	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C	B	A
0–1 V	1	1	1	1	1	1	1	0	0	0
1–2 V	0	1	1	1	1	1	1	0	0	1
2–3 V	0	0	1	1	1	1	1	0	1	0
3–4 V	0	0	0	1	1	1	1	0	1	1
4–5 V	0	0	0	0	1	1	1	1	0	0
5–6 V	0	0	0	0	0	1	1	1	0	1
6–7 V	0	0	0	0	0	0	1	1	1	0
> 7 V	0	0	0	0	0	0	0	1	1	1

(b)



riría $2^8 = 256$ niveles de voltaje, incluyendo 0 V. Esto requeriría 256 resistencias y 255 comparadores (no hay comparador para el nivel de 0 V). Las 255 salidas de los comparadores alimentarían un circuito codificador con prioridad, el cual produciría un código de ocho bits correspondiente a la salida del comparador de mayor orden que estuviera en BAJO. En general, un convertidor Flash de N bits requiere $2^N - 1$ comparadores, 2^N resistencias y la lógica necesaria para el codificador.

Tiempo de conversión

El convertidor tipo Flash no utiliza señal de reloj, ya que no se requiere sincronización ni secuenciamiento. La conversión se lleva a cabo de manera continua. Cuando cambie el valor de la entrada analógica, cambiarán las salidas del codificador. El tiempo de conversión es el tiempo que tarda en aparecer una nueva salida digital en respuesta a un cambio en V_A, y depende sólo de los tiempos de propagación de los comparadores y de la lógica del codificador. Por esta razón, los convertidores Flash

tienen tiempos de conversión extremadamente cortos. Por ejemplo, el AD9020 de Analog Devices es un convertidor tipo Flash de 10 bits con un tiempo de conversión menor a 17 ns.

PREGUNTAS DE REPASO

1. *Verdadero o falso:* un ADC tipo Flash no contiene un DAC.
2. ¿Cuántos comparadores requiere un convertidor Flash de 12 bits? ¿Cuántas resistencias?
3. Mencione la principal ventaja y desventaja de un convertidor Flash.

11-13 OTROS MÉTODOS DE CONVERSIÓN A/D

Existen varios métodos de conversión A/D que han estado en uso durante cierto tiempo ya, cada uno con sus ventajas y desventajas relativas. Ahora describiremos en forma breve algunos de ellos.

ADC ascendente/descendente de rampa digital (ADC rastreador)

Como hemos visto, el ADC de rampa digital es relativamente lento, ya que el contador se restablece a 0 al principio de cada nueva conversión. La escalera siempre comienza en 0 V y avanza en intervalos hasta el "punto de conmutación", en donde V_{AX} se excede de V_A y la salida del comparador cambia a BAJO. El tiempo que requiere la escalera para restablecerse a 0 y volver a avanzar en intervalos hasta el nuevo punto de conmutación es en realidad un desperdicio. El **ADC ascendente/descendente de rampa digital** utiliza un contador ascendente/descendente para reducir este desperdicio de tiempo.

El contador ascendente/descendente sustituye al contador ascendente que alimenta al DAC. Está diseñado para contar en forma ascendente cada vez que la salida del comparador indica que $V_{AX} < V_A$ y para contar en forma descendente cada vez que $V_{AX} > V_A$. Por lo tanto, la salida del DAC siempre avanza en intervalos en la dirección del valor de V_A. Cada vez que la salida del comparador cambia de estado, indica que V_{AX} ha "cruzado" el valor de V_A, que el equivalente digital de V_A está en el contador y la conversión está completa.

Cuando va a comenzar una nueva conversión el contador *no se restablece a 0*, sino que comienza a contar ya sea en forma ascendente o descendente a partir de su último valor, dependiendo de la salida del comparador. Contará hasta que la escalera cruce V_A de nuevo para finalizar la conversión. Entonces, la forma de onda de V_{AX} contendrá señales de escalera de pendiente tanto positiva como negativa, las cuales "rastrearán" la señal de V_A. Por esta razón a este ADC se le conoce comúnmente como *ADC rastreador*.

Es evidente que los tiempos de conversión se reducirán en general, ya que el contador no empieza desde 0 cada vez, sino que tan sólo sigue contando en forma ascendente o descendente a partir de su valor anterior. Desde luego que el valor de t_C seguirá dependiendo del valor de V_A y, por lo tanto, no será constante.

ADC integrador de pendiente dual

El **convertidor de pendiente dual** tiene uno de los tiempos de conversión más lentos (por lo general, de 10 a 100 ms), pero tiene la ventaja de un costo relativamente bajo ya que no requiere componentes de precisión tales como un DAC o un VCO. La operación básica de este convertidor implica el proceso de carga y descarga *lineal* de un capacitor mediante el uso de corrientes constantes. En primer lugar, el capacitor se carga durante un intervalo de tiempo fijo mediante una corriente constante, la cual se deriva del voltaje de entrada analógico V_A. En consecuencia, al final del

intervalo de carga fijo el voltaje del capacitor será proporcional a V_A. En ese punto, el capacitor se descarga en forma lineal mediante una corriente constante que se deriva de un voltaje de referencia preciso, V_{ref}. Cuando el voltaje del capacitor llega a 0, se termina la descarga lineal. Durante el intervalo de descarga se alimenta una frecuencia de referencia digital a un contador y se cuenta. La duración del intervalo de descarga será proporcional al voltaje inicial del capacitor. Así, al final del intervalo de descarga el contador tendrá una cuenta proporcional al voltaje inicial del capacitor que a su vez, como dijimos, es proporcional a V_A.

Además de su bajo costo, otra ventaja del ADC de pendiente dual es su baja sensibilidad al ruido y a las variaciones en los valores de sus componentes, las cuales se producen debido a los cambios en la temperatura. Debido a sus tiempos de conversión bajos, el ADC de pendiente dual no se utiliza en aplicaciones de adquisición de datos. Sin embargo, los tiempos de conversión lentos no son problema en aplicaciones tales como los voltímetros o multímetros digitales, y es aquí en donde encuentran su principal aplicación.

ADC de voltaje a frecuencia

El **ADC de voltaje a frecuencia** es más simple que otros ADCs, ya que no utiliza un DAC, sino un *oscilador lineal controlado por voltaje (VCO)* que produce una frecuencia de salida proporcional a su voltaje de entrada. El voltaje analógico que va a convertirse se aplica al VCO para generar una frecuencia de salida. Esta frecuencia se alimenta a un contador para contar un intervalo de tiempo fijo. El conteo final es proporcional al valor del voltaje analógico.

Para ilustrar esto, suponga que el VCO genera una frecuencia de 10 kHz para cada volt de entrada (es decir, 1 V produce 10 kHz, 1.5 V producen 15 kHz, 2.73 V producen 27.3 kHz). Si el voltaje de entrada analógico es de 4.54 V, entonces la salida del VCO será una señal de 45.4 kHz que aplicará pulsos de reloj a un contador durante, por decir, 10 ms. Después del intervalo de conteo de 10 ms, el contador retendrá la cuenta de 454, que viene siendo la representación digital de 4.54 V.

Aunque éste es un método de conversión simple, es difícil lograr un alto nivel de precisión debido a la dificultad que se presenta al diseñar VCOs con precisiones de más del 0.1 por ciento.

Una de las principales aplicaciones de este tipo de convertidor es en ambientes industriales ruidosos, en donde deben transmitirse señales analógicas pequeñas desde circuitos transductores hacia una computadora de control. Las pequeñas señales analógicas pueden verse afectadas en forma drástica por el ruido, si se transmiten en forma directa a la computadora de control. Un método más conveniente es alimentar la señal analógica a un VCO, el cuál genera una señal digital cuya frecuencia de salida cambia de acuerdo con la entrada analógica. Esta señal digital se transmitirá a la computadora sin que le afecte el ruido. Los circuitos en la computadora de control contarán los pulsos digitales (es decir, realizarán una función de conteo de frecuencia) para producir un valor digital equivalente a la entrada analógica original.

Modulación Sigma/Delta*

Otro método para representar información analógica en forma digital es la **modulación sigma/delta**. Un convertidor A/D sigma/delta es un dispositivo de sobremuestreo, lo cual significa que en efecto muestrea la información analógica con más frecuencia que la velocidad de muestreo mínima. Esta velocidad mínima de muestreo es dos veces mayor que la frecuencia más alta en la onda analógica de entrada. Al igual que el método de voltaje a frecuencia, el método sigma/delta no produce en

* Como base para esta descripción utilizamos un excelente artículo publicado en la Web por Jim Thompson, de la Universidad de Washington. Visite el sitio Web complementario llamado *Digital Systems: Principles and Applications* en http://www.pearsoneducacion.net/Tocci para encontrar el vínculo hacia este artículo.

forma directa un número de varios bits para cada muestra, sino que para representar el voltaje analógico varía la densidad de 1s lógicos en una trama de bits de datos en serie. Para representar las porciones positivas de una forma de onda, el modulador genera una trama de bits con una alta densidad de 1s (por ejemplo, 01111101111110 111110111). Para representar las porciones negativas se genera una menor densidad de 1s (es decir, una mayor densidad de 0s) (por ejemplo, 00010001000010001000).

La modulación sigma/delta se utiliza en la conversión A/D y también en la conversión D/A. Se diseña una forma de un circuito modulador sigma/delta para convertir una señal analógica en un flujo de bits modulado (A/D). La otra forma convierte una secuencia de muestras digitales en el flujo de bits modulado (D/A). Como tenemos la perspectiva de los sistemas digitales, es más fácil comprender el último de estos dos circuitos, ya que consiste sólo de componentes digitales que hemos estudiado. La figura 11-23 muestra un circuito que recibe un valor digital con signo de cinco bits como entrada y lo convierte en un flujo de bits sigma/delta. Vamos a suponer que los números que pueden colocarse en la entrada de este circuito están dentro del intervalo de −8 a +8. El primer componente es tan sólo un restador (la sección delta) similar al que estudiamos en la figura 6-14. El restador determina qué tan alejado está el número de entrada de su valor mínimo o máximo. A menudo a esta diferencia se le conoce como señal de error. Los siguientes dos componentes (el sumador y el registro D) forman un acumulador muy similar al circuito de la figura 6-10 (la sección sigma). Para cada muestra que entra, el acumulador suma la diferencia (señal de error) al total de la operación. Cuando el error es pequeño, este total operativo (sigma) cambia en pequeños incrementos. Cuando el error es grande, el componente sigma cambia en incrementos grandes. El último componente compara el total operativo del acumulador con un umbral fijo, que en este caso es cero. En otras palabras, tan sólo determina si el total es positivo o negativo, Para ello se utiliza el MSB (bit de signo) de sigma. Tan pronto como el total se hace positivo, el MSB cambia a BAJO y retroalimenta a la sección delta el máximo valor positivo (+8). Cuando el MSB de sigma se hace negativo, retroalimenta el máximo valor negativo (−8).

Una hoja de cálculo es un excelente medio para analizar un circuito como éste. Las tablas en esta sección se obtuvieron de la hoja de cálculo que viene en el CD

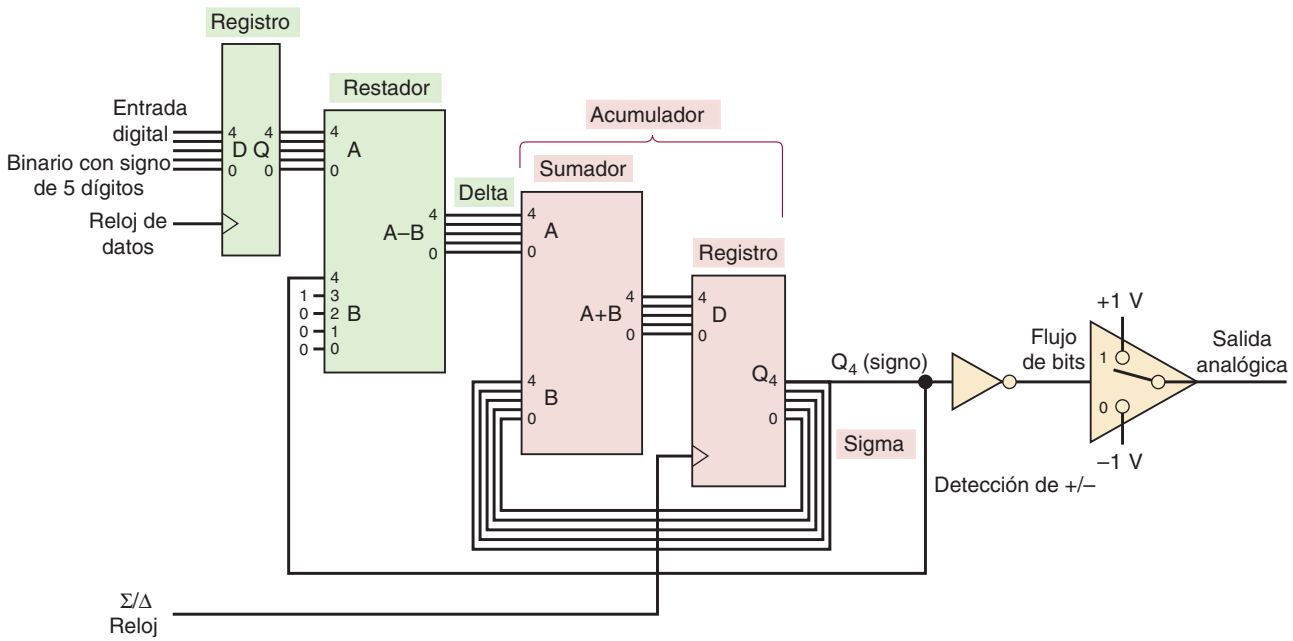


FIGURA 11-23 Modulador sigma/delta en un convertidor D/A.

TABLA 11-7
Modulador sigma/delta con una entrada de 0.

Muestra (s)	ENT digital	Delta	Sigma	Salida de trama de bits	SAL analógica	Retroalimentación
1	0	−8	0	1	1	8
2	0	8	−8	0	−1	−8
3	0	−8	0	1	1	8
4	0	8	−8	0	−1	−8
5	0	−8	0	1	1	8
6	0	8	−8	0	−1	−8
7	0	−8	0	1	1	8
8	0	8	−8	0	−1	−8

TABLA 11-8
Modulador sigma/delta con una entrada de 4.

Muestra (s)	ENT digital	Delta	Sigma	Trama de bits de salida	SAL analógica	Retroalimentación
1	4	−4	4	1	1	8
2	4	−4	0	1	1	8
3	4	12	−4	0	−1	−8
4	4	−4	8	1	1	8
5	4	−4	4	1	1	8
6	4	−4	0	1	1	8
7	4	12	−4	0	−1	−8
8	4	−4	8	1	1	8

que se incluye con este libro. La tabla 11-7 muestra la operación del convertidor cuando hay un valor de cero en la entrada. Observe que la salida de flujo de bits alterna entre 1 y 0, y que el valor promedio de la salida analógica es de 0 volts. La tabla 11-8 muestra lo que ocurre cuando la entrada digital es 4. Si suponemos que 8 es escala completa, esto representa $\frac{4}{8} = 0.5$. La salida está en ALTO para tres muestras y en BAJO para una muestra, un patrón que se repite cada cuatro muestras. El valor promedio de la salida analógica es $(1 + 1 + 1 - 1)/4 = 0.5$ V.

Como ejemplo final vamos a utilizar una entrada de −5, la cual representa $-5/8 = -0.625$. La tabla 11-9 muestra la salida resultante. El patrón en el flujo de bits no es periódico. De la columna sigma podemos ver que se requieren 16 muestras para que se repita el patrón. No obstante, si tomamos la densidad de bits total y calculamos el valor promedio de la salida analógica a través de 16 muestras, encontraremos que es igual a −0.625. Es probable que su reproductor de CD utilice un convertidor D/A sigma/delta que opere de esta forma. Los números digitales de 16 bits salen del CD en forma serial; después se les aplica un formato para convertirlos en patrones de datos en paralelo y se aplican mediante pulsos de reloj a un convertidor. A medida que entran los números cambiantes al convertidor, el valor promedio de la salida analógica cambia de manera acorde. A continuación, la salida analógica pasa a través de un circuito llamado filtro pasabajas, el cual suaviza los cambios repentinos y produce un voltaje con cambios suaves, que viene siendo el valor promedio de la trama de bits. En sus audífonos esta señal analógica cambiante suena justo igual que la grabación original. Un convertidor A/D sigma/delta funciona de una manera muy similar, sólo que convierte el voltaje analógico en el flujo de bits modulado. Para almacenar los datos digitalizados como una lista de números binarios de N bits, se calcula y se almacena la densidad de bits promedio de 2^N muestras de flujo de bits.

TABLA 11-9
Modulador sigma/delta con una entrada de −5.

Muestra (s)	ENT digital	Delta	Sigma	Trama de bits de salida	SAL analógica	Retroalimentación
1	−5	3	−5	0	−1	−8
2	−5	3	−2	0	−1	−8
3	−5	−13	1	1	1	8
4	−5	3	−12	0	−1	−8
5	−5	3	−9	0	−1	−8
6	−5	3	−6	0	−1	−8
7	−5	3	−3	0	−1	−8
8	−5	−13	0	1	1	8
9	−5	3	−13	0	−1	−8
10	−5	3	−10	0	−1	−8
11	−5	3	−7	0	−1	−8
12	−5	3	−4	0	−1	−8
13	−5	3	−1	0	−1	−8
14	−5	−13	2	1	1	8
15	−5	3	−11	0	−1	−8
16	−5	3	−8	0	−1	−8
17	−5	3	−5	0	−1	−8
18	−5	3	−2	0	−1	−8

PREGUNTAS DE REPASO

1. ¿En qué forma mejora el ADC ascendente/descendente de rampa digital al ADC de rampa digital?
2. ¿Cuál es el principal elemento de un ADC de voltaje a frecuencia?
3. Cite dos ventajas y una desventaja del ADC de pendiente dual.
4. Nombre tres tipos de ADCs que no utilizan un DAC.
5. ¿Cuántos bits de datos de salida utiliza un modulador sigma/delta?

11-14 CIRCUITOS DE MUESTREO Y RETENCIÓN

Cuando un voltaje analógico se conecta en forma directa a la entrada de un ADC, el proceso de conversión puede verse afectado si el voltaje analógico cambia durante el tiempo de conversión. La estabilidad del proceso de conversión puede mejorarse mediante el uso de un **circuito de muestreo y retención (S/H)** para mantener el voltaje analógico constante mientras se realiza la conversión A/D. En la figura 11-24 se muestra un diagrama simplificado de un circuito de muestreo y retención (S/H).

El circuito S/H contiene un amplificador con ganancia unitaria A_1, el cual presenta una alta impedancia a la señal analógica y tiene una baja impedancia de salida que puede cargar con rapidez el capacitor de retención C_h. Este capacitor se conecta a la salida de A_1 cuando el interruptor de control digital está cerrado. A esto se le conoce como la operación de *muestreo*. El interruptor estará cerrado el tiempo suficiente como para que C_h se cargue hasta el valor presente de la entrada analógica. Por ejemplo, si el interruptor se cierra en el tiempo t_0, la salida A_1 cargará rápidamente el capacitor C_h hasta un voltaje V_0. Cuando el interruptor se abra, C_h *retendrá* este voltaje de manera que la salida de A_2 aplique este voltaje al ADC. El amplificador búfer de ganancia unitaria A_2 presenta una alta impedancia de entrada que no descargará el voltaje

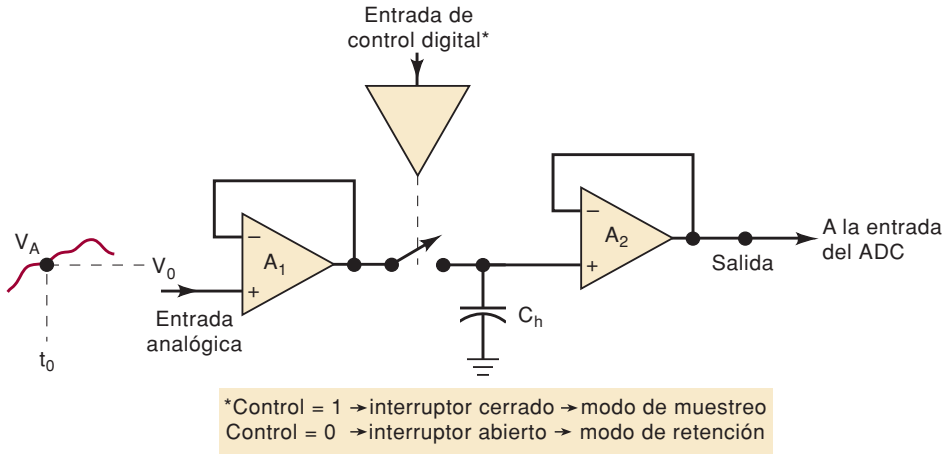


FIGURA 11-24
Diagrama simplificado de un circuito de muestreo y retención.

del capacitor en forma considerable durante el tiempo de conversión del ADC, por lo que éste recibirá en esencia un voltaje de entrada V_0 de corriente directa.

En un sistema de adquisición de datos controlado por computadora tal como el que vimos antes, el interruptor de muestreo y retención se controla mediante una señal digital proveniente de la computadora. Esta señal de la computadora cerraría el interruptor para cargar el capacitor C_h con una nueva muestra del voltaje analógico; la cantidad de tiempo que el interruptor tendría que permanecer cerrado se conoce como el **tiempo de adquisición**, el cual depende del valor de C_h y de las características del circuito S/H. La señal de la computadora abriría entonces el interruptor para permitir que C_h mantuviera su valor y proporcionara un voltaje analógico relativamente constante en la salida de A_2.

El ADC1154 es un circuito integrado de muestreo y retención que tiene un capacitor de retención interno con un tiempo de adquisición de 3.5 s. Durante el tiempo de retención, el voltaje del capacitor se perderá (descargará) a una velocidad de sólo 0.1 V/ s. La pérdida de voltaje dentro del intervalo de muestreo deberá ser menor que la ponderación del LSB. Por ejemplo, un convertidor de 10 bits con un intervalo de escala completa de 10 V tendría una ponderación del LSB aproximada de 10 mV. Se requerirían 100 ms para que la pérdida de voltaje del capacitor fuera igual a la ponderación del LSB del ADC. Sin embargo, es muy poco probable que alguna vez llegue a ser necesario retener la muestra durante un tiempo tan largo en el proceso de conversión.

PREGUNTAS DE REPASO

1. Describa la función de un circuito de muestreo y retención.
2. *Verdadero o falso:* los amplificadores en un circuito S/H se utilizan para proporcionar amplificación de voltaje.

11-15 MULTIPLEXAJE

Cuando van a convertirse entradas analógicas provenientes de varias fuentes, puede utilizarse una técnica de multiplexaje de manera que un ADC pueda compartirse en el tiempo. En la figura 11-25 se ilustra el esquema básico para un sistema de adquisición de cuatro canales. El interruptor giratorio *S* se utiliza para conmutar cada señal analógica con la entrada del ADC, una a la vez y en secuencia. El circuito de control se encarga de controlar la posición del interruptor, de acuerdo con los bits de *selección de dirección* (A_1, A_0) del contador MOD-4. Por ejemplo, cuando $A_1A_0 = 00$ el interruptor conecta a V_{A0} a la entrada del ADC; cuando $A_1A_0 = 01$ se conecta V_{A1} con la entrada del ADC; y así en lo sucesivo. Cada canal de entrada tiene un código de dirección específico que, cuando está presente, conecta ese canal con el ADC.

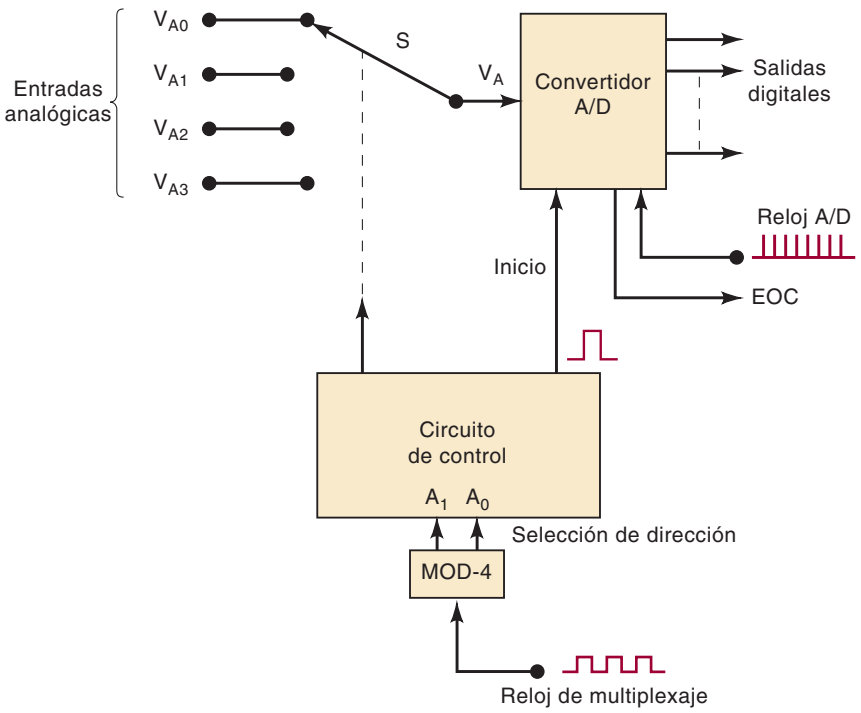


FIGURA 11-25
Conversión de cuatro entradas analógicas mediante el multiplexaje a través de un ADC.

La operación procede de la siguiente manera:

1. Cuando la selección de dirección = 00, V_{A0} se conecta a la entrada del ADC.
2. El circuito de control genera un pulso INICIO para iniciar la conversión de V_{A0} en su equivalente digital.
3. Cuando la conversión está completa, *EOC* indica que los datos de salida del ADC están listos. Por lo general, estos datos se transferirán a una computadora mediante un bus de datos.
4. El reloj de multiplexaje incrementa la selección de dirección a 01, con lo cual se conecta V_{A1} al ADC.
5. Se repiten los pasos 2 y 3 para que ahora esté presente el equivalente digital de V_{A1} en las salidas del ADC.
6. El reloj de multiplexaje incrementa la selección de dirección a 10 y V_{A2} se conecta al ADC.
7. Se repiten los pasos 2 y 3 para que ahora esté presente el equivalente digital de V_{A2} en las salidas del ADC.
8. El reloj de multiplexaje incrementa la selección de dirección a 11 y V_{A3} se conecta al ADC.
9. Se repiten los pasos 2 y 3 para que ahora esté presente el equivalente digital de V_{A3} en las salidas del ADC.

El reloj de multiplexaje controla la velocidad a la cual se conmutan las señales analógicas en secuencia con el ADC. La velocidad máxima se determina mediante el tiempo de retraso de los interruptores y el tiempo de conversión del ADC. El tiempo de retraso del interruptor puede minimizarse mediante el uso de interruptores semiconductores, como el interruptor CMOS bilateral que describimos en el capítulo 8. Puede ser necesario conectar un circuito de muestreo y retención en la entrada del ADC si las entradas analógicas pueden cambiar de manera considerable durante el tiempo de conversión del ADC.

Muchos ADCs integrados contienen los circuitos de multiplexaje en el mismo chip que el ADC. Por ejemplo, el ADC0808 puede multiplexar ocho entradas ana-

lógicas distintas en un ADC. Utiliza un código de selección de entrada de tres bits para determinar cuál entrada analógica está conectada al ADC.

PREGUNTAS DE REPASO

1. ¿Cuál es la ventaja de este esquema de multiplexaje?
2. ¿Cómo podría cambiarse el contador de direcciones si hubiera ocho entradas analógicas?

11-16 OSCILOSCOPIO DE ALMACENAMIENTO DIGITAL

Como ejemplo final de la aplicación de los convertidores D/A y A/D, daremos un breve vistazo al osciloscopio de almacenamiento digital (DSO). El DSO utiliza ambos tipos de dispositivos para digitalizar, almacenar y visualizar formas de ondas analógicas.

En la figura 11-26 se muestra el diagrama de bloques de un DSO. La operación en general se controla y sincroniza mediante los circuitos en el bloque CONTROL, que, por lo general, contiene un microprocesador que ejecuta un programa de control almacenado en la ROM (memoria de sólo lectura). La porción de la adquisición de datos del sistema contiene un circuito de muestreo y retención (S/H) y un ADC que muestrea y digitaliza en forma repetitiva la señal de entrada, a una velocidad determinada por el RELOJ DE MUESTREO y después envía los datos digitalizados a la memoria para su almacenamiento. El bloque CONTROL se asegura de que los puntos de datos subsiguientes se almacenen en ubicaciones de memoria subsiguientes; para ello actualiza en forma continua el CONTADOR DE DIRECCIONES de la memoria.

Cuando la memoria está llena, el siguiente punto de datos del ADC se almacena en la primera ubicación de memoria y se borran los datos anteriores; este proceso continúa para los subsiguientes puntos de datos. El proceso de adquisición y almacenamiento de datos continúa hasta que el bloque CONTROL recibe una señal de

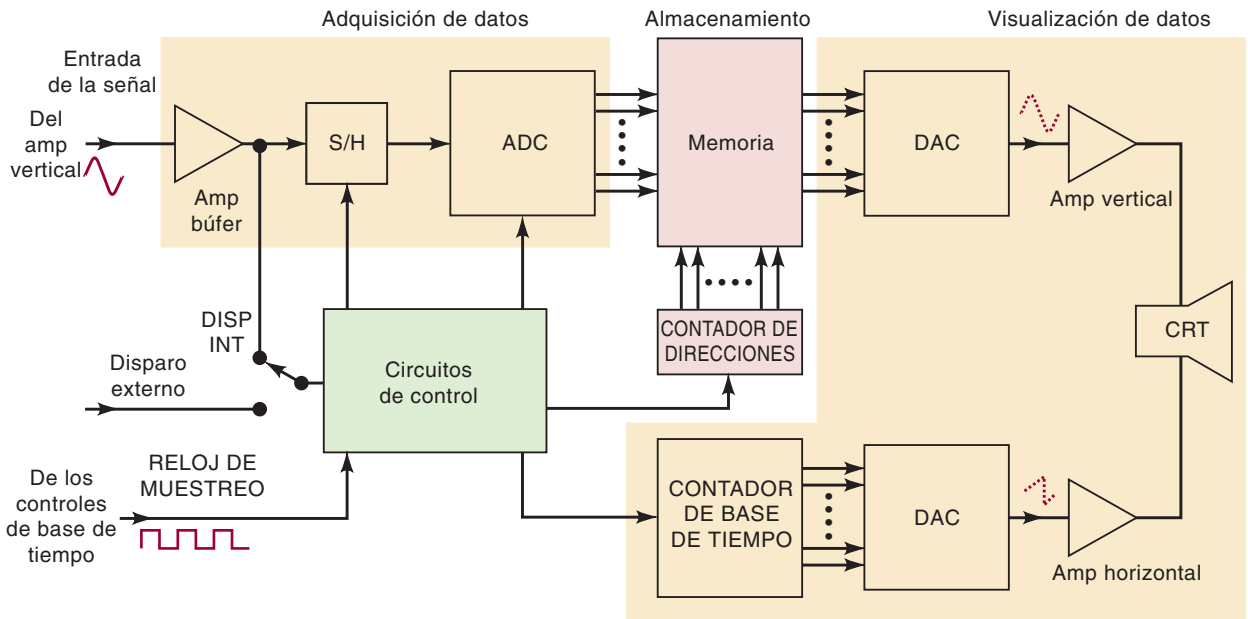


FIGURA 11-26 Diagrama de bloques de un osciloscopio de almacenamiento digital.

disparo, ya sea de la forma de onda de entrada (disparo INTERNO) o de una fuente de disparo EXTERNO. Cuando se produce el disparo, el sistema deja de adquirir nuevos datos y entra al modo de operación de visualización, en donde todos o parte de los datos de la memoria se visualizan en forma repetitiva en el CRT.

La operación de visualización utiliza dos DACs para proporcionar los voltajes de deflexión horizontal y vertical para el CRT. Los datos de la memoria producen la deflexión vertical del haz de electrones, mientras que el CONTADOR DE BASE DE TIEMPO proporciona la deflexión vertical en forma de una señal de barrido tipo escalera. El bloque CONTROL sincroniza la operación de visualización mediante el incremento del CONTADOR DE DIRECCIONES de memoria y el CONTADOR DE BASE DE TIEMPO al mismo tiempo, de manera que cada intervalo horizontal del haz de electrones esté acompañado de un nuevo valor de datos que vaya de la memoria hacia el DAC vertical. Los contadores se reciclan en forma continua, de manera que los puntos de datos almacenados se vuelvan a trazar en forma repetitiva en la pantalla de CRT. La visualización en pantalla consiste de puntos discretos que representan los diversos puntos de datos pero, por lo general, el número de puntos es tan grande (de 1000 o más) que tienden a mezclarse entre sí para dar la apariencia de una forma de onda continua y uniforme. La operación de visualización se termina cuando el operador oprime un botón en el panel frontal que indica al DSO que comience un nuevo ciclo de adquisición de datos.

Aplicaciones relacionadas

La misma secuencia de operaciones que se lleva a cabo en un DSO (adquisición de datos/digitalización/almacenamiento/salida de datos) se utiliza en otras aplicaciones de los DACs y los ADCs. Por ejemplo, los monitores del corazón que se encuentran en cualquier hospital son similares a los DSOs, sólo que muestran de manera constante una forma de onda en la que se indica la actividad cardiaca del paciente durante los últimos segundos. Como otro ejemplo, las cámaras de video digitales digitalizan una imagen, un elemento de imagen (píxel) a la vez y almacenan la información en cinta magnética o en DVD. Las cámaras fotográficas digitales digitalizan cada píxel y almacenan los datos en una tarjeta de memoria de estado sólido. Después los datos pueden transferirse en forma digital y después enviarse como salida hacia un dispositivo digital, en donde los datos se convierten en una señal de "brillo" analógica para cada píxel y se reensamblan para formar una imagen en la pantalla.

PREGUNTAS DE REPASO

1. Vea la figura 11-26. ¿Cómo se "almacenan" las formas de onda en un DSO?
2. Describa las funciones del ADC y los DACs que forman parte del DSO.

11-17 PROCESAMIENTO DIGITAL DE SEÑALES (DSP)

Una de las áreas más dinámicas de los sistemas digitales en la actualidad está en el campo de los **procesadores digitales de señales (DSP)**. Un DSP es una forma muy especializada de procesador, el cual está optimizado para realizar cálculos repetitivos con flujos de datos digitalizados. Por lo general, estos datos digitalizados se alimentan al DSP a través de un convertidor A/D. La explicación de las matemáticas que permiten a un DSP procesar estos valores de datos es un tema que se encuentra más allá del alcance de este libro, pero basta con decir que para cada nuevo punto de datos que llega como entrada, se realiza un cálculo (con mucha rapidez). Este cálculo involucra el punto de datos más reciente así como varias de las muestras de datos anteriores. El resultado del cálculo produce un nuevo punto de datos de salida, el cual, por lo general, se envía a un convertidor D/A. Un sistema DSP es similar al diagrama de bloques que se muestra en la figura 11-1. La principal diferencia se encuentra en el hardware especializado que contiene la sección computacional.

Una de las principales aplicaciones del DSP está en el filtrado y condicionamiento de señales analógicas. Como un ejemplo muy sencillo, un DSP puede programarse para recibir una forma de onda analógica tal como la salida de un preamplificador de audio, y pasar a la salida sólo aquellos componentes de frecuencia que se encuentren por debajo de cierta frecuencia. Todas las frecuencias más altas se atenúan mediante el filtro. Tal vez usted recuerde de su estudio sobre los circuitos analógicos que puede lograrse lo mismo mediante un filtro simple de paso bajo compuesto por una resistencia y un capacitor. La ventaja del DSP en comparación con las resistencias y capacitares es la flexibilidad de poder cambiar la frecuencia crítica sin tener que cambiar componentes. En vez de ello, sólo se cambian números en los cálculos para adaptar la respuesta dinámica del filtro. ¿Alguna vez ha estado en un auditorio cuando el sistema de audio empieza a rechinar? Esto puede evitarse si se filtra la frecuencia de retroalimentación degenerativa. Por desgracia, la frecuencia que provoca el rechinido cambia con el número de personas en el cuarto, la ropa que utilicen y muchos otros factores. Con un ecualizador de audio basado en DSP puede detectarse la frecuencia de oscilación y los filtros se pueden ajustar en forma dinámica para eliminar el rechinido.

Filtrado digital

Para ayudarlo a comprender el concepto de filtrado digital, imagine que va a comprar y vender acciones. Para decidir cuándo comprar y vender, necesita saber lo que está ocurriendo en el mercado. Es conveniente ignorar los cambios repentinos a corto plazo (alta frecuencia) y responder a las tendencias generales (promedios de 30 días). Cada día usted lee el periódico, toma una muestra del precio de cierre para sus acciones y lo anota. Después utiliza una fórmula para calcular el promedio de los precios en los últimos 30 días. Este valor promedio se traza como se muestra en la figura 11-27; el gráfico resultante se utiliza para tomar decisiones. Ésta es una forma de filtrar la señal digital (secuencia de muestras de datos) que representa la actividad del mercado de acciones.

Ahora imagine que en vez de muestrear los precios de las acciones, un sistema digital muestrea una señal de audio (analógica) proveniente de un micrófono, mediante el uso de un convertidor A/D. En vez de tomar una muestra una vez al día, toma una muestra 20,000 veces cada segundo (cada 50 μs). Para cada muestra se realiza un cálculo del promedio de ponderación, utilizando las últimas 256 muestras de datos y se produce un solo punto de datos de salida. Un **promedio ponderado** significa que algunos de los puntos de datos se consideran más importantes que otros. Cada una de las muestras se multiplica por un número fraccional (entre 0 y 1) antes de sumarlas entre sí. Este cálculo del promedio procesa (filtra) la señal de audio.

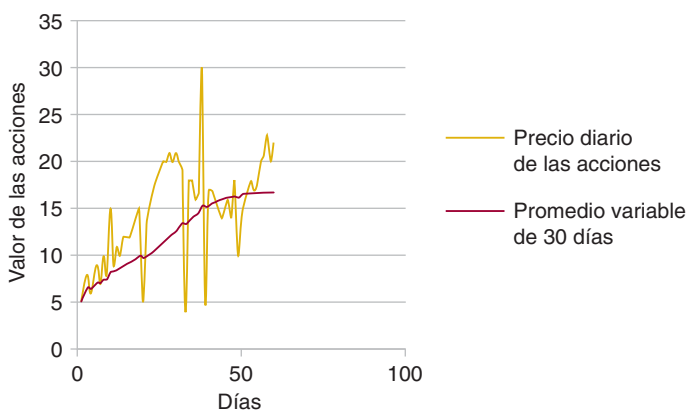


FIGURA 11-27
Filtrado digital de la actividad en el mercado de valores.

La parte más difícil de esta forma de DSP es determinar las constantes de ponderación correctas para el cálculo del promedio, para poder lograr las características deseadas del filtro. Por fortuna ya hay software disponible para PCs que facilita esto de manera considerable. El hardware DSP especial debe realizar las siguientes operaciones:

Leer la muestra más reciente (un nuevo miembro) de la conversión A/D.

Sustituir la muestra más antigua (de 256) con la nueva de la conversión A/D.

Multiplicar cada una de las 256 muestras por su constante de ponderación correspondiente.

Sumar todos estos productos.

Enviar como salida la suma resultante de productos (1 número) al convertidor D/A.

La figura 11-28 muestra la arquitectura básica de un DSP. La sección de multiplicación y acumulación (**MAC**) es central para todos los DSPs y se utiliza en la mayoría de las aplicaciones. Se utiliza hardware especial (como el que veremos en el capítulo 12) para implementar el sistema de memoria que almacena las muestras de datos y los valores de ponderación. La **unidad aritmética-lógica** y el **desplazador de barril** (registro de desplazamiento) proporcionan el soporte necesario para tratar con el sistema numérico binario al procesar las señales.

Hay otra aplicación útil del DSP, la cual se conoce como **sobremuestreo** o **filtrado por interpolación**. Como podrá recordar, la forma de onda reconstruida es siempre una aproximación de la original debido al error de cuantización. Los cambios repentinos en los intervalos de un punto de datos al siguiente también introducen ruido de alta frecuencia en la señal reconstruida. Un DSP puede insertar puntos de datos interpolados en la señal digital. La figura 11-29 muestra cómo el filtrado por interpolación con submuestreo de 4X suaviza la forma de onda y hace posible

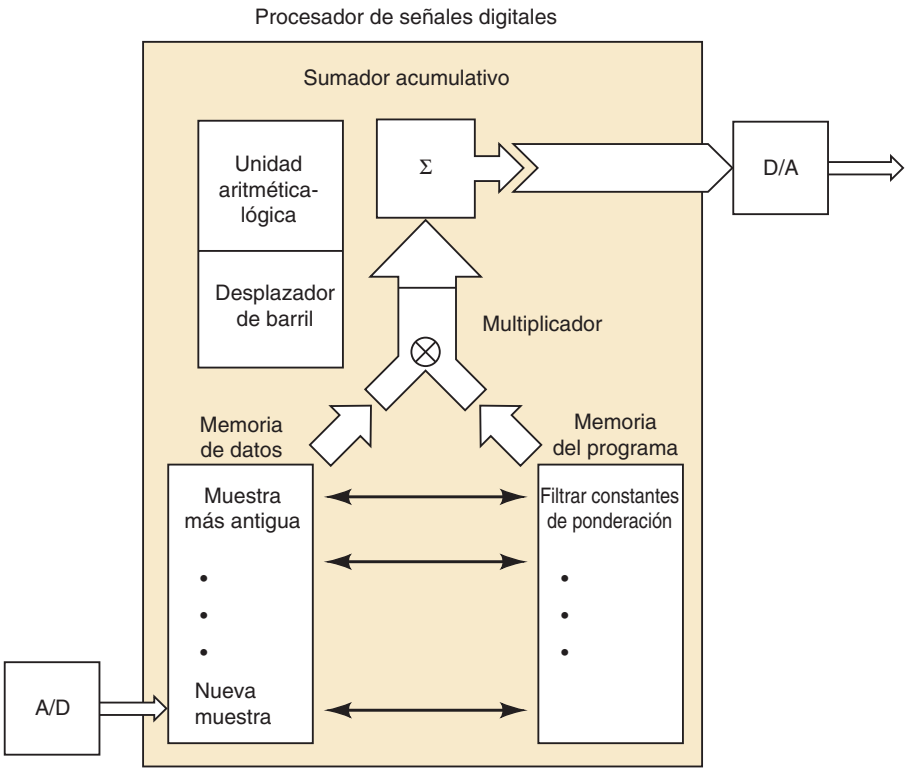


FIGURA 11-28
Arquitectura del procesador de señales digitales.

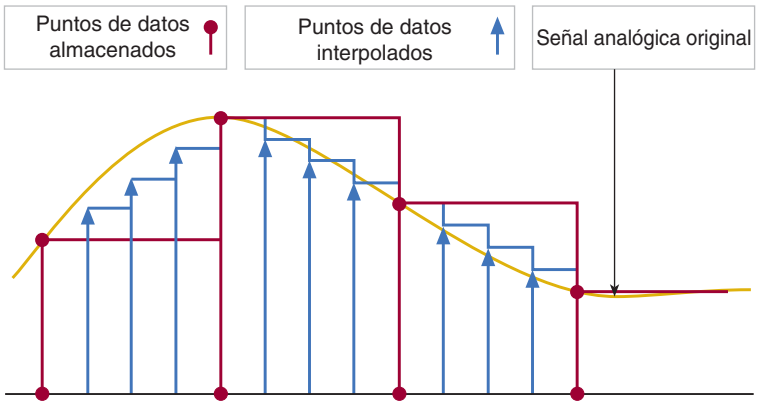


FIGURA 11-29
Inserción de un punto de datos interpolado en una señal digital para reducir el ruido.

el filtrado final con circuitos analógicos más simples. El DSP realiza este papel en su reproductor de CD para proporcionar una excelente reproducción de audio. Los puntos redondos representan los datos digitales grabados en su CD. Los triángulos representan los puntos de datos interpolados que inserta el filtro digital en su reproductor de CD antes del filtro de salida analógica final.

En este capítulo y en otros anteriores hemos presentado muchos de los conceptos importantes que usted necesita comprender para poder comenzar a utilizar el DSP. Son imprescindibles los métodos y el hardware de conversión A/D y D/A, junto con los conceptos de adquisición de datos y muestreo de señales. Los temas como las representaciones de números binarios con signo (incluyendo fracciones), suma y multiplicación de números binarios con signo (que cubrimos en el capítulo 6), y los registros de desplazamiento (capítulo 7) son necesarios para comprender el hardware y la programación de un DSP. Los conceptos del sistema de memoria, que presentaremos en el siguiente capítulo, también serán importantes.

El DSP se está integrando en muchos sistemas comunes con los que usted ya está familiarizado. Los reproductores de CD utilizan DSP para filtrar los datos digitales que se leen del disco y así minimizar el ruido de cuantización que se produce de manera inevitable cuando se digitaliza la música. Los sistemas telefónicos utilizan DSP para cancelar ecos en las líneas telefónicas. Los módems de alta velocidad que son estándar en las PCs son posibles y económicos gracias al DSP. Las cajas de efectos especiales para las guitarras y otros instrumentos realizan efectos de eco, reverberación, ajuste de fase y demás efectos mediante el uso del DSP. Las aplicaciones del DSP están creciendo justo ahora a la misma velocidad con que crecieron las aplicaciones del microprocesador a principios de la década de 1980. Proporcionan una solución digital para muchos problemas analógicos tradicionales. Algunos otros ejemplos de aplicaciones son el reconocimiento de voz, cifrado de datos de telecomunicaciones, transformadas rápidas de Fourier, procesamiento de imágenes en la televisión digital, la formación de rayos ultrasónicos en la electrónica médica y la cancelación de ruido en controles industriales. A medida que continúe esta tendencia, podemos esperar ver que casi todos los sistemas electrónicos contengan circuitos para procesamiento de señales digitales.

PREGUNTAS DE REPASO

1. ¿Cuál es una aplicación principal del DSP?
2. ¿Cuál es la fuente típica de datos digitales para que un DSP los procese?
3. ¿Qué ventaja tiene un filtro DSP en comparación con un circuito de filtro analógico?
4. ¿Cuál es la característica de hardware central de un DSP?
5. ¿Cuántos puntos de datos interpolados se insertan entre las muestras cuando se realiza un filtrado digital con sobremuestreo de 4X? ¿Cuántos para un sobremuestreo de 8X?

RESUMEN

1. Las variables físicas que queremos medir, tales como la temperatura, la presión, la humedad, la distancia, la velocidad, etcétera, son cantidades que varían en forma continua. Puede utilizarse un transductor para traducir estas cantidades en una señal eléctrica de voltaje o corriente que fluctúa en proporción a la variable física. Estas señales de voltaje o corriente que varían en forma continua se llaman señales *analógicas*.
2. Para medir una variable física, un sistema digital debe asignar un número binario al valor analógico que está presente en ese instante. Esto se logra mediante un convertidor A/D. Para generar valores de voltaje o corriente variables que puedan controlar procesos físicos, un sistema digital debe traducir los números binarios en una magnitud de voltaje o de corriente. Esto se logra mediante un convertidor D/A.
3. Un convertidor D/A con n bits divide un intervalo de valores analógicos (voltaje o corriente) en $2^n - 1$ piezas. El tamaño o la magnitud de cada pieza es la ponderación equivalente analógica del bit menos significativo. A esto se le conoce como la *resolución* o *tamaño del escalón*.
4. La mayoría de los convertidores D/A utiliza redes de resistencias que pueden hacer que fluyan cantidades ponderadas de corriente cuando se active cualquiera de sus entradas binarias. La cantidad de corriente que fluye es proporcional al peso binario de cada bit de entrada. Estas corrientes ponderadas se suman para crear la señal analógica de salida.
5. Un convertidor A/D debe asignar un número binario a una cantidad analógica (que varía en forma continua). La precisión con la que un convertidor A/D puede realizar esta conversión depende de cuántos números distintos puede asignar y qué tan amplio sea el intervalo analógico. El cambio más pequeño en el valor analógico que puede medir un convertidor A/D se conoce como su *resolución*, la ponderación de su bit menos significativo.
6. Al muestrear en forma repetida la señal analógica entrante, convertirla en digital y almacenar los valores digitales en un dispositivo de memoria, puede capturarse una forma de onda analógica. Para reconstruir la señal, los valores digitales se leen del dispositivo de memoria a la misma velocidad con la que se almacenaron, y después se alimentan en un convertidor D/A. Después la salida del convertidor D/A se filtra para suavizar los escalones en la salida y recrear la forma de onda original. El ancho de banda de las señales muestreadas se limita a ½ F_S. Las frecuencias entrantes mayores de ½ F_S crean una *señal fantasma* que tiene una frecuencia igual a la diferencia entre el múltiplo entero más cercano de F_S y la frecuencia entrante. Esta diferencia siempre será menor de ½ F_S.
7. Un convertidor A/D de rampa digital es el más simple de comprender, pero no se utiliza comúnmente debido a su tiempo de conversión variable. Un convertidor de aproximaciones sucesivas tiene un tiempo de conversión constante y es probablemente el convertidor de propósito general más común.
8. Los convertidores Flash utilizan comparadores analógicos y un codificador con prioridad para asignar un valor digital a la entrada analógica. Éstos son los convertidores más rápidos, ya que los únicos retrasos involucrados son los tiempos de propagación.
9. Otros métodos populares de conversión A/D son: rastreo ascendente/descendente, integración, conversión de voltaje a frecuencia y conversión sigma/delta. Cada tipo de convertidor tiene su propio nicho de aplicaciones.
10. Cualquier convertidor D/A puede utilizarse con otros circuitos tales como los multiplexores analógicos, los cuales seleccionan una de varias señales analógicas que se van a convertir, una a la vez. Los circuitos de muestreo y retención pueden utilizarse para "congelar" una señal analógica con cambios rápidos mientras se lleva a cabo la conversión.

11. Los procesadores digitales de señales es un nuevo y emocionante campo de crecimiento para la electrónica. Estos dispositivos permiten realizar cálculos con rapidez para poder emular la operación de muchos circuitos de filtros analógicos en forma digital. La principal característica arquitectónica de un DSP es un circuito multiplicador y sumador de hardware que puede multiplicar pares de números entre sí y acumular el total operativo (suma) de estos productos. Este circuito se utiliza para realizar con eficiencia el cálculo del promedio móvil ponderado que se utilizan para implementar filtros digitales y otras funciones del DSP. Éste es responsable de muchos de los avances recientes en el audio de alta fidelidad, en la TV de alta definición y en las telecomunicaciones.

TÉRMINOS IMPORTANTES

ADC ascendente/descendente de rampa digital
ADC de aproximaciones sucesivas
ADC de pendiente dual
ADC de rampa digital
ADC de voltaje a frecuencia
ADC tipo Flash
cantidad analógica
cantidad digital
circuito de muestreo y retención (S/H)
convertidor analógico-digital (ADC)
convertidor digital-analógico (DAC)
desplazador de barril
digitaliza
entradas diferenciales
error a escala completa
error de cuantización
error de desplazamiento
error de linealidad
escalera
filtrado por interpolación
frecuencia de muestreo, F_S
MAC
modulación sigma/delta
monotonicidad
muestreo
Procesadores digitales de señales (DSP)
promedio ponderado
resolución
salida a escala completa
sobremuestreo
submuestreo
tamaño del escalón
tiempo de adquisición
tiempo de estabilización
transductor
traslape espectral
unidad aritmética-lógica
WRITE

PROBLEMAS

SECCIONES 11-1 Y 11-2

B 11-1. **PREGUNTA DE PRÁCTICA**

(a) ¿Cuál es la expresión que relaciona la salida y las entradas de un DAC?

(b) Defina el *tamaño de escalón* de un DAC.

(c) Defina la *resolución* de un DAC.

(d) Defina *escala completa*.

(e) Defina *resolución porcentual*.

(f)* *Verdadero o falso:* un DAC de 10 bits tendrá una resolución más pequeña que un DAC de 12 bits para la misma salida a escala completa.

(g)* *Verdadero o falso:* un DAC de 10 bits con salida a escala completa de 10 V tiene una resolución porcentual más pequeña que un DAC de 10 bits con escala completa de 12 V.

B 11-2. Un DAC de ocho bits produce un voltaje de salida de 2.0 V para un código de entrada de 01100100. ¿Cuál será el valor de V_{SAL} para un código de entrada de 10110011?

B 11-3.* Determine el peso de cada bit de entrada para el DAC del problema 11-2.

* Encontrará las respuestas a los problemas marcados con un asterisco al final del libro.

B 11-4. ¿Cuál es la resolución del DAC del problema 11-2? Expréselo en volts y como un porcentaje.

B 11-5.* ¿Cuál es la resolución en volts de un DAC de 10 bits, cuya salida F.S. es de 5 V?

B 11-6. ¿Cuántos bits se requieren para un DAC, de manera que su salida F.S. sea de 10 mA y su resolución sea menor de 40 A?

B 11-7.* ¿Cuál es la resolución porcentual del DAC de la figura 11-30? ¿Cuál es el tamaño del escalón si el valor máximo es de 2 V?

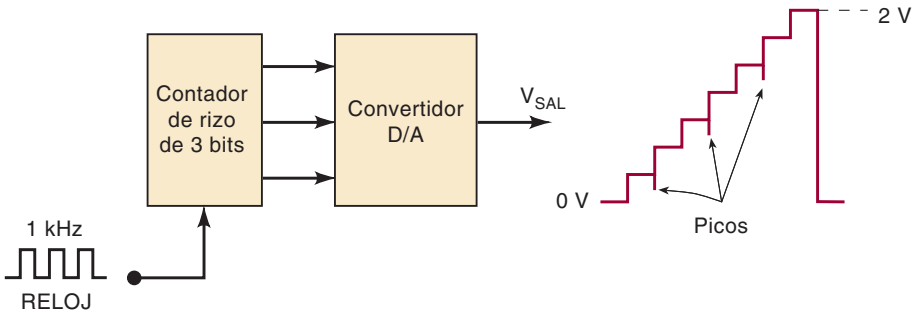


FIGURA 11-30
Problemas 11-7 y 11-8.

C 11-8. ¿Cuál es la causa de los picos de pendiente negativa en la forma de onda de V_{SAL} de la figura 11-30? (*Sugerencia:* observe que el contador es de rizo y que los picos ocurren en cualquier otro paso.)

B 11-9.* Suponiendo que tenemos un DAC de 12 bits con precisión perfecta, ¿qué tan cerca de los 250 rpm puede ajustarse la velocidad del motor en la figura 11-4?

11.10. Un DAC de 12 bits tiene una salida a escala completa de 15.0 V. Determine el tamaño del escalón, la resolución porcentual y el valor de V_{SAL} para un código de entrada de 011010010101.

11-11.* Un microcontrolador tiene un puerto de salida de ocho bits que debe utilizarse para controlar un DAC. El DAC disponible tiene 10 bits de entrada y una salida a escala completa de 10 V. La aplicación requiere un voltaje que varíe entre 0 y 10 V en intervalos de 50 mV o menores. ¿Qué ocho bits del DAC de 10 bits se conectarán al puerto de salida?

11-12. Usted necesita un DAC que pueda abarcar 12 V con una resolución de 20 mV o menor. ¿Cuántos bits se requieren?

SECCIÓN 11-3

D 11-13.* El tamaño del escalón del DAC de la figura 11-5 puede modificarse si se cambia el valor de R_F. Determine el valor requerido de R_F para un tamaño de escalón de 0.5 V. ¿Cambiará el nuevo valor de R_F la resolución porcentual?

D 11-14. Suponga que la salida del DAC de la figura 11-17(a) se conecta al amplificador operacional de la figura 11-7(b).

(a) Determine el tamaño del escalón y el voltaje a escala completa en V_{SAL} si $V_{REF} = 5$ V, $R = 20$ kΩ y $R_F = 10$ kΩ.
(b) Modifique el valor de R_F, de manera que el voltaje a escala completa en V_{SAL} sea de −2 V.
(c) Utilice este nuevo valor de R_F y determine el factor de proporcionalidad (K) en la relación $V_{SAL} = K(V_{REF} \times B)$.

11-15.* ¿Cuál es la ventaja del DAC de la figura 11-8, en comparación con el de la figura 11-17, en especial para un número más grande de bits de entrada?

SECCIONES 11-4 A 11-6

11-16. Un DAC de ocho bits tiene un error a escala completa de 0.2% F.S. Si el DAC tiene una salida a escala completa de 10 mA, ¿cuál es el error máximo que puede haber en cualquier entrada digital? Si la salida del convertidor D/A indica 50 A para una entrada digital de 00000001, ¿se encuentra dentro del intervalo especificado de precisión? (Suponga que no hay error de desplazamiento.)

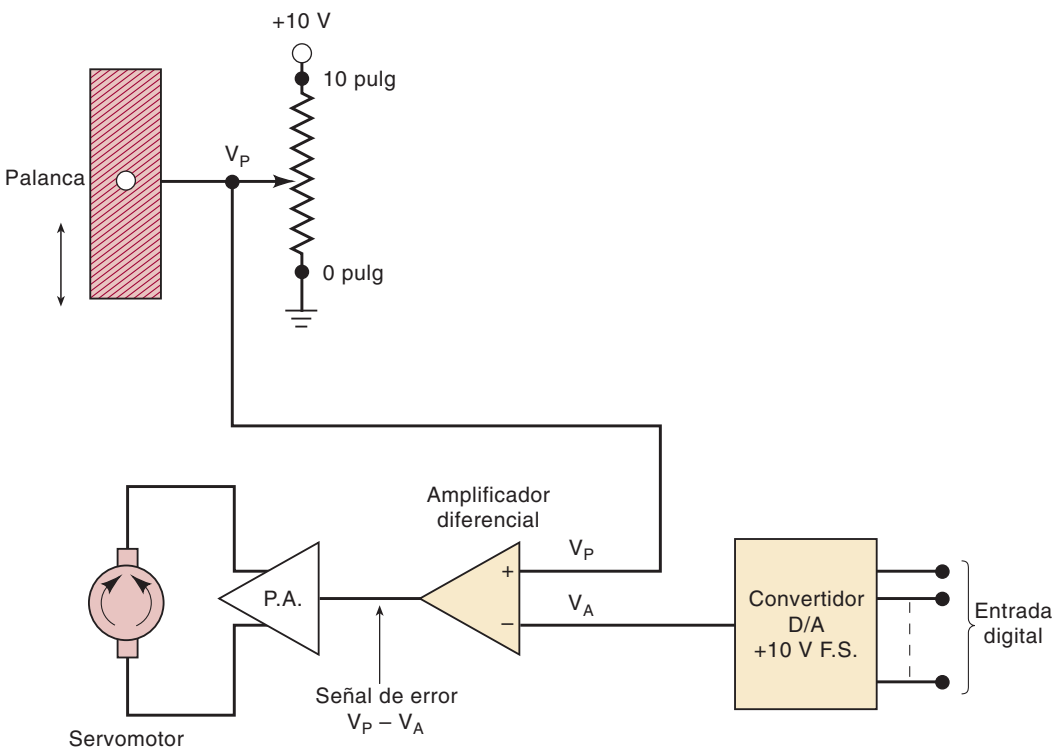
A, N 11-17.* El control de un dispositivo de posicionamiento puede lograrse mediante el uso de un *servomotor*, que viene siendo un motor diseñado para controlar un dispositivo mecánico, mientras no exista una señal de error. La figura 11-31 muestra un sistema servo-controlado simple, el cual se controla mediante una entrada digital que podría llegar directamente de una computadora o de un medio de salida tal como cinta magnética. El brazo de palanca se desplaza en forma vertical mediante el servomotor. El motor gira en sentido a favor o en contra de las manecillas del reloj, dependiendo de si el voltaje del amplificador de potencia (P.A.) es positivo o negativo. El motor se detiene cuando la salida del P.A. es de 0.

La posición mecánica de la palanca se convierte en un voltaje de corriente directa mediante el arreglo de potenciómetros que se muestra. Cuando la palanca se encuentra en su punto de referencia 0, V_P = 0 V. El valor de V_P se incrementa a la velocidad de 1 V/pulgada hasta que la palanca se encuentra en su punto más alto (10 pulg) y V_P = 10 V. La posición deseada de la palanca se proporciona como un código digital que proviene de la computadora y después se alimenta a un DAC, con lo cual se produce V_A. La *diferencia* entre V_P y V_A (que se conoce como *error*) se produce debido al amplificador *diferencial* y se amplifica mediante el P.A. para controlar el motor en la dirección que haga que la señal de error disminuya a 0; esto es, desplaza la palanca hasta que $V_P = V_A$.

(a) Si la palanca debe posicionarse dentro de una resolución de 0.1 pulgadas, ¿cuál es el número de bits necesarios en el código de entrada digital?

(b) En la operación real, el brazo de palanca podría oscilar un poco alrededor de la posición deseada, en especial si se utiliza un potenciómetro de alambre enrollado. ¿Puede explicar por qué?

FIGURA 11-31
Problema 11-17.



B 11-18. **PREGUNTA DE PRÁCTICA**

(a) Defina el concepto de *red de resistencias con ponderación binaria*.

(b) Defina el concepto de *red de escalera R/2R*.

(c) Defina el concepto de *tiempo de estabilización de un DAC*.

(d) Defina *error a escala completa*.

(e) Defina *error de desplazamiento*.

11-19.* Cierto DAC de seis bits tiene una salida a escala completa, clasificada en 1.260 V. Su precisión se especifica como ± 0.1% F.S., y tiene un error de desplazamiento de ± 1 mV. Suponga que el error de desplazamiento no se ignora. Considere las mediciones obtenidas de este DAC (tabla 11-10) y determine cuáles de ellas no se encuentran dentro de las especificaciones del dispositivo. (*Sugerencia:* el error de desplazamiento se suma al error producido por las imprecisiones de los componentes.)

TABLA 11-10

Código de entrada	Salida
000010	41.5 mV
000111	140.2 mV
001100	242.5 mV
111111	1.258 V

SECCIÓN 11-7

T 11-20. Cierto DAC tiene las siguientes especificaciones: resolución de ocho bits, escala completa = 2.55 V, desplazamiento ≦ 2 mV; precisión = ±0.1% F.S. Una prueba estática con este DAC produce los resultados que se muestran en la tabla 11-11. ¿Cuál es la probable causa de la falla?

TABLA 11-11

Código de entrada	Salida
00000000	8 mV
00000001	18.2 mV
00000010	28.5 mV
00000100	48.3 mV
00001111	158.3 mV
10000000	1.289 V

T 11-21.* Repita el problema 11-20 usando los datos de medición que se proporcionan en la tabla 11-12.

TABLA 11-12

Código de entrada	Salida
00000000	20.5 mV
00000001	30.5 mV
00000010	20.5 mV
00000100	60.6 mV
00001111	150.6 mV
10000000	1.300 V

B 11-22.* Un técnico conecta un contador al DAC de la figura 11-3 para realizar una prueba de escalera con un reloj de 1 kHz. El resultado se muestra en la figura 11-32. ¿Cuál es la causa probable de la señal de escalera incorrecta?

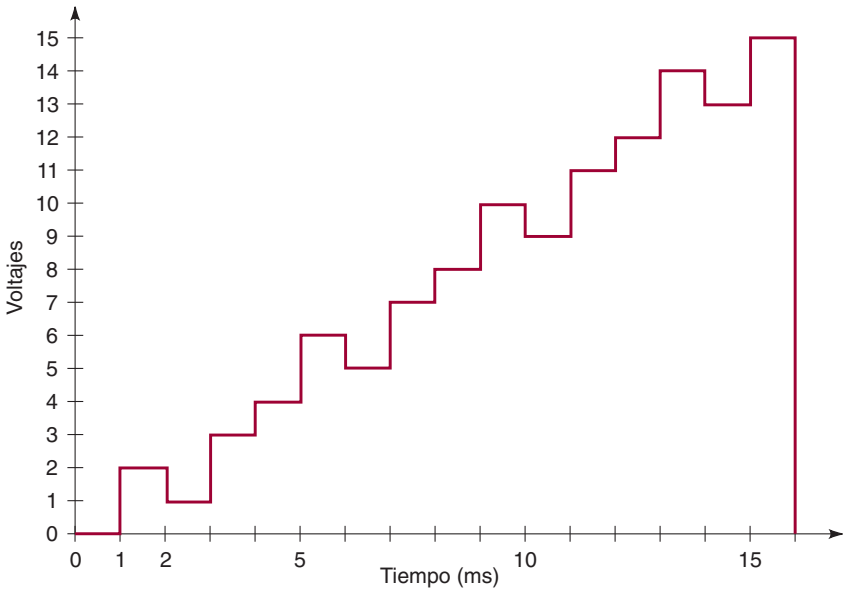


FIGURA 11-32 Problema 11-22.

SECCIONES 11-8 Y 11-9

11-23. **PREGUNTA DE PRÁCTICA**

En la siguiente descripción del ADC de la figura 11-13 llene los espacios en blanco. Cada espacio en blanco puede contener una o más palabras.

Un pulso INICIO se aplica para _____ el contador y evitar que pase _____ a través de la compuerta AND y hacia el _____. En este punto, la salida del DAC (V_{AX}) es _____ y $\overline{EOC}$ es _____.

Cuando INICIO regresa _____, la compuerta AND está en _____ y el contador puede _____. La señal V_{AX} se incrementa un _____ a la vez hasta que _____ V_A. En ese punto, _____ cambia a BAJO para _____ más pulsos de _____. Esto indica el fin de la conversión y el equivalente de V_A está presente en _____.

B 11-24. Un ADC de rampa digital de ocho bits con una resolución de 40 mV utiliza una frecuencia de reloj de 2.5 MHz y un comparador con V_T = 1 mV. Determine los siguientes valores.

(a)* La salida digital cuando V_A = 6.000 V.

(b) La salida digital para 6.035 V.

(c) Los tiempos de conversión mínimo y promedio para este ADC.

B 11-25. ¿Por qué las salidas digitales fueron iguales para las partes (a) y (b) del problema 11-24?

D 11-26. ¿Qué pasaría con el ADC del problema 11-24 si se aplicara un voltaje analógico de V_A = 10.853 V a la entrada? ¿Qué forma de onda aparecería en la salida del convertidor D/A? Incorpore la lógica necesaria en este ADC, de manera que se genere una indicación de "Fuera de escala" cada vez que V_A sea demasiado grande.

B 11-27.* Un ADC tiene las siguientes características: resolución: 12 bits, error a escala completa: 0.03% F.S; salida a escala completa: +5V.

(a) ¿Cuál es el error de cuantización en volts?

(b) ¿Cuál es el error posible total en volts?

C, N 11-28. El error de cuantización de un ADC como el que se muestra en la figura 11-13 es siempre positivo, ya que el valor de V_{AX} debe exceder de V_A para que la salida del comparador cambie de estados. Esto significa que el valor de V_{AX} podría ser cuando mucho un LSB más que V_A. Este error de cuantización puede modificarse de manera que V_{AX} se encuentre dentro de ± ½ LSB de V_A. Para ello se agrega al valor de V_A un voltaje fijo igual a ½ LSB (½ intervalo). La figura 11-33 muestra esto en forma simbólica para un convertidor que tiene una resolución de 10 mV/intervalo. Se agrega un voltaje fijo de +5 mV a la salida del convertidor D/A en el amplificador sumador, y el resultado (V_{AY}) se alimenta al comparador, el cual tiene V_T = 1 mV.

Para este convertidor modificado, determine la salida digital para los siguientes valores de V_A.

(a)* V_A = 5.022 V

(b) V_A = 50.28 V

Determine el error de cuantización en cada caso mediante la comparación entre V_{AX} y V_A. Observe que el error es positivo en un caso y negativo en el otro.

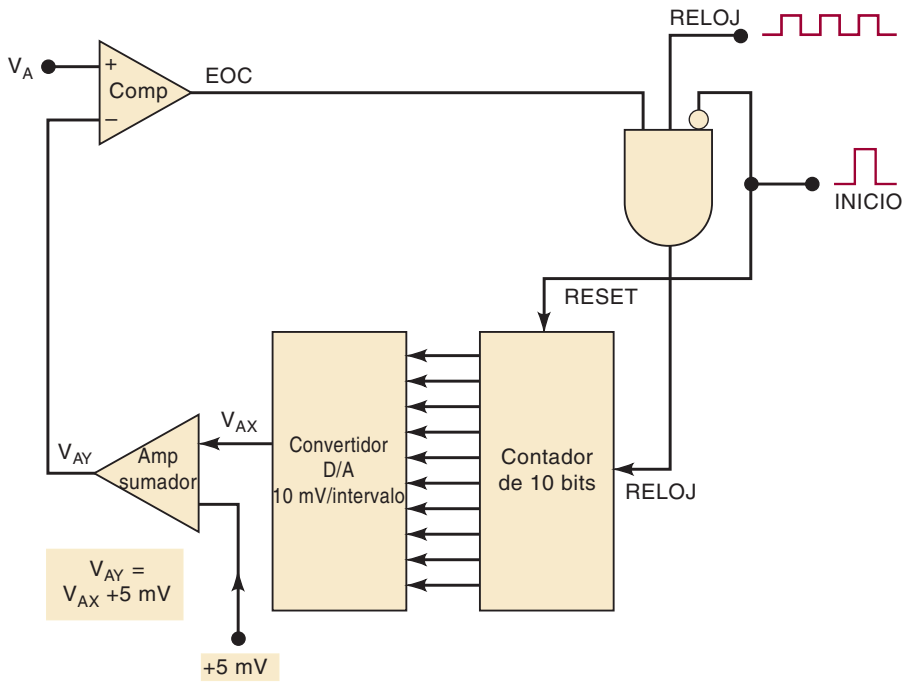


FIGURA 11-33 Problemas 11-28 y 11-29.

C 11-29. Para el ADC de la figura 11-33, determine el intervalo de valores de entrada analógicos que producirán una salida digital de 0100011100.

SECCIÓN 11-10

B 11-30. Suponga que la señal analógica de la figura 11-34(a) debe digitalizarse realizando conversiones A/D continuas mediante un convertidor de rampa digital de ocho bits, cuya escalera se eleva a la velocidad de 1 V cada 25 μs. Bosqueje la señal reconstruida utilizando los datos que se obtuvieron durante el proceso de digitalización. Compárela con la señal original y describa lo que podría hacerse para obtener una representación más precisa.

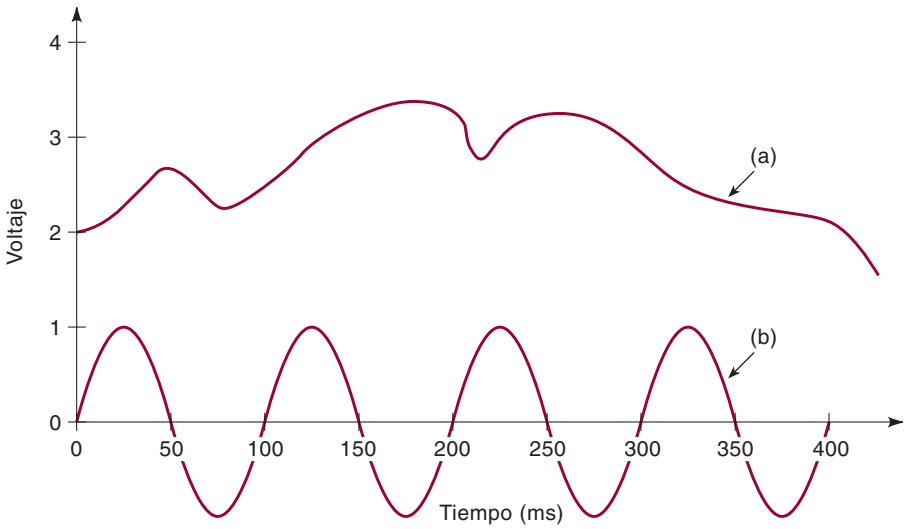


FIGURA 11-34 Problemas 11-30, 11-31 y 11-41.

C 11-31.* En la onda senoidal de la figura 11-34(b), marque los puntos en donde se toman las muestras mediante un convertidor A/D tipo Flash, a intervalos de 75 s (comenzando en el origen). Después dibuje la salida reconstruida del convertidor D/A (conecte los puntos de muestra con líneas rectas para mostrar el filtrado). Calcule la frecuencia de muestreo, la frecuencia de la entrada senoidal y la diferencia entre ellas. Después compare con la frecuencia de la forma de onda reconstruida resultante.

11-32. Un sistema de adquisición de datos muestreados se utiliza para digitalizar una señal de audio. Suponga que la frecuencia de muestreo F_S es de 20 kHz. Determine la frecuencia de salida que se escuchará para cada una de las siguientes frecuencias de entrada.

(a)* Señal de entrada = 5 kHz

(b)* Señal de entrada = 10.1 kHz

(c) Señal de entrada = 10.2 kHz

(d) Señal de entrada = 15 kHz

(e) Señal de entrada = 19.1 kHz

(f) Señal de entrada = 19.2 kHz

SECCIÓN 11-11

B 11-33.* **PREGUNTA DE PRÁCTICA**

Indique en dónde hace referencia cada una de las siguientes declaraciones al ADC de rampa digital, al ADC de aproximaciones sucesivas, o ambos.

(a) Produce una señal de escalera en la salida de su DAC.
(b) Tiene un tiempo de conversión constante, independiente de V_A.
(c) Tiene un tiempo de conversión promedio más corto.
(d) Utiliza un comparador analógico.
(e) Utiliza un DAC.
(f) Utiliza un contador.
(g) Tiene una lógica de control compleja.
(h) Tiene una salida *EOC.*

11-34. Dibuje la forma de onda para V_{AX} cuando el SAC de la figura 11-19 convierte a $V_A = 6.7$ V.

11-35. Repita el problema 11-34 para $V_A = 16$ V.

B 11-36.* Cierto convertidor de aproximaciones sucesivas de ocho bits tiene 2.55 V a escala completa. El tiempo de conversión para $V_A = 1$ V es de 80 μs. ¿Cuál será el tiempo de conversión para $V_A = 1.5$ V?

11-37. La figura 11-35 muestra la forma de onda en V_{AX} para un SAC de seis bits con un tamaño de escalón de 40 mV, durante un ciclo de conversión completo. Examine esta forma de onda y describa lo que ocurre en los tiempos t_0 a t_5. Después determine la entrada digital resultante.

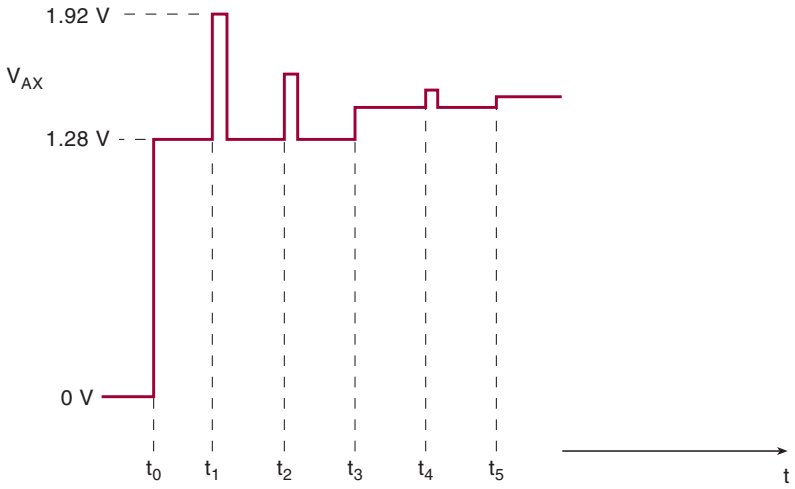


FIGURA 11-35 Problema 11-37.

B 11-38.* Consulte la figura 11-21. ¿Cuál es el valor aproximado de la entrada analógica si el bus de datos de la microcomputadora está en 10010111 cuando *RD* recibe un pulso para cambiar a BAJO?

D 11-39. Conecte una fuente de referencia de 2.0 V a $V_{ref}/2$ y repita el problema 11-38.

C,D 11-40.* Diseñe la interfaz de un ADC para un termostato digital, utilizando un sensor de temperatura LM35 y el ADC0804. Su sistema deberá realizar mediciones precisas (±0.1 °C) de 10 a 40 °C. El LM35 produce 0.01 V de salida por cada grado °C (0 °C = 0 V).

(a) ¿Cuál debe ser el valor digital para 10 °C, para la mejor resolución?
(b) ¿Qué voltaje debe aplicarse a V_{ent} (−)?
(c) ¿Cuál es el intervalo de voltaje de escala completa que entrará?
(d) ¿Qué voltaje debe aplicarse a $V_{ref}/2$?
(e) ¿Qué valor binario representará 22 °C?
(f) ¿Cuál es la resolución en °C? ¿En volts?

SECCIÓN 11-12

B 11-41. Describa cómo funcionaría un ADC tipo Flash con un tiempo de conversión de 1 μs para la situación del problema 11-30.

D 11-42. Dibuje el diagrama del circuito para un convertidor Flash de cuatro bits con salida BCD y una resolución de 0.1 V. Suponga que hay un voltaje de suministro de precisión de +5 V disponible.

PREGUNTA DE PRÁCTICA

B 11-43. Para cada una de las siguientes declaraciones, indique qué tipo de ADC (rampa digital, SAC o Flash) se describe.

(a) Método más rápido de conversión.
(b) Necesita un pulso de INICIO.
(c) Requiere la mayor cantidad de circuitos.
(d) No utiliza un DAC.
(e) Genera una señal de escalera.
(f) Utiliza un comparador analógico.
(g) Tiene un tiempo de conversión relativamente fijo, independiente de V_A.

PREGUNTA 11-13

B 11.44 **PREGUNTA DE PRÁCTICA**

Para cada declaración, indique qué tipo(s) de ADC se describe(n).

(a) Utiliza un contador que nunca se restablece a 0.
(b) Utiliza un número extenso de comparadores.
(c) Utiliza un VCO.
(d) Se utiliza en entornos industriales ruidosos.
(e) Utiliza un capacitor.
(f) Es relativamente insensible a la temperatura.

SECCIONES 11-14 Y 11-15

F 11-45.* Consulte el circuito de muestreo y retención de la figura 11-24. ¿Qué falla del circuito provocaría que V_{SAL} se viera idéntico a V_A? ¿Qué falla ocasionaría que V_{SAL} permaneciera en 0?

A, D 11-46. Utilice el CI CMOS 4016 (sección 8-16) para implementar la conmutación en la figura 11-25 y diseñe la lógica de control necesaria, de manera que cada entrada analógica se convierta en su equivalente digital en secuencia. El ADC es del tipo de aproximaciones sucesivas de 10 bits y utiliza una señal de reloj de 50 kHz; además requiere un pulso de inicio con duración de 10 μs para comenzar cada conversión. Las salidas digitales deben permanecer estables durante 100 μs después de que termina la conversión, para poder conmutar hacia la siguiente entrada analógica. Seleccione una frecuencia de reloj de multiplexaje apropiada.

APLICACIÓN DE MICROCOMPUTADORA

A, N, D 11-47.* La figura 11-21 muestra cómo se integra el ADC0804 con una microcomputadora. Muestra tres señales de control ($\overline{CS}$, $\overline{RD}$ y $\overline{WR}$) que provienen de la microcomputadora hacia el ADC. Estas señales se utilizan para comenzar cada nueva conversión A/D y leer (transferir) la salida de datos del ADC hacia la microcomputadora, a través del bus de datos.

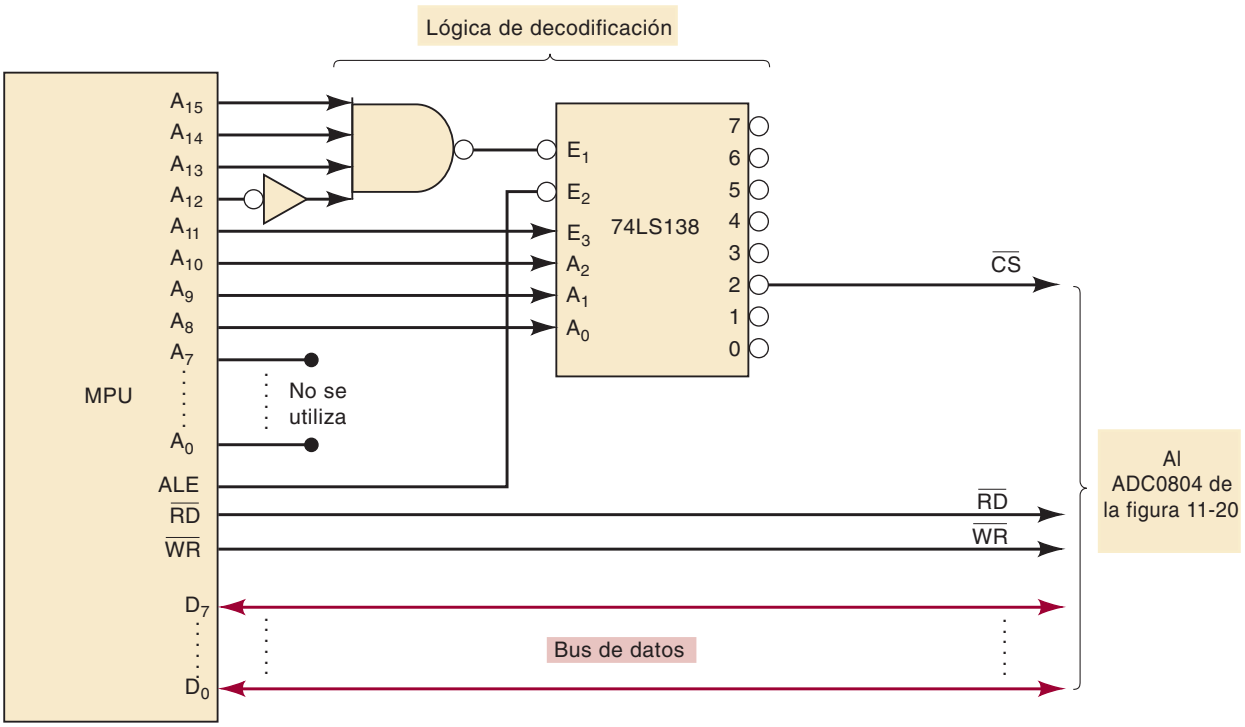


FIGURA 11-36 Problema 11-47: Interfaz entre el MPU y el ADC0804 de la figura 11-20.

La figura 11-36 muestra una manera en que podría implementarse la lógica de decodificación de direcciones. La señal $\overline{CS}$ que activa el ADC0804 se desarrolla a partir de las ocho líneas de dirección de mayor orden del bus de direcciones del MPU. Cada vez que el MPU quiere comunicarse con el ADC0804, coloca la dirección del ADC0804 en el bus de direcciones y la lógica de decodificación lleva a la señal $\overline{CS}$ al nivel BAJO. Observe que además de las líneas de dirección, se conecta una señal de sincronización y control (*ALE*) a la entrada de habilitación $\overline{E}_2$. Cada vez que *ALE* está en ALTO, significa que la dirección se encuentra en una transición potencial, por lo que el decodificador debe deshabilitarse hasta que *ALE* cambie a BAJO (en cuyo momento la dirección será válida y estable). Esto tiene un propósito para la sincronización, pero no tiene efecto sobre la dirección del ADC.

(a) Determine la dirección del ADC0804.

(b) Modifique el diagrama de la figura 11-36 para colocar el ADC0804 en la dirección E8XX hexadecimal.

(c) Modifique el diagrama de la figura 11-36 para colocar el ADC0804 en la dirección FFXX hexadecimal.

D 11-48. Usted tiene un convertidor A/D SAC de 10 bits (AD 573) disponible, pero su sistema sólo requiere ocho bits de resolución y sólo tiene ocho bits de puerto disponibles en su microprocesador. ¿Puede utilizar este convertidor A/D y, de ser así, cuáles de las 10 líneas de datos conectará al puerto?

SECCIÓN 11-17

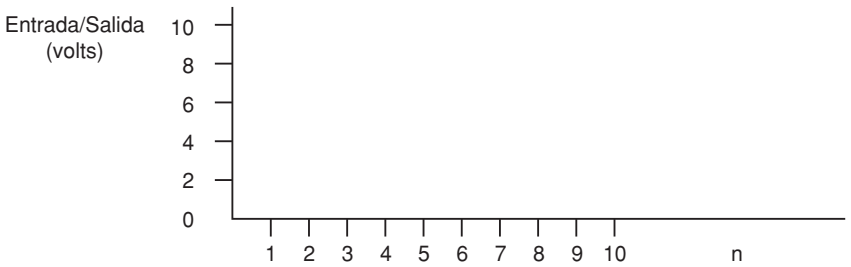
11-49. Los datos en la tabla 11-13 son muestras de entrada tomadas por un convertidor A/D. Observe que si se trazaran los datos de entrada, representarían una función simple de intervalos tal como el flanco ascendente de una señal digital. Calcule el promedio simple de los cuatro puntos de datos más recientes; comience con SAL[4] y proceda hasta SAL[10]. Trace los valores

para ENT y SAL contra el número de muestra *n*, como se muestra en la figura 11-37.

TABLA 11-13

Muestra *n*	1	2	3	4	5	6	7	8	9	10
ENT[*n*] (V)	0	0	0	0	10	10	10	10	10	10
SAL[*n*] (V)	0	0	0							

FIGURA 11-37
Formato gráfico para los problemas 11-49 y 11-50



Cálculos de las muestras:

$$\text{SAL}[n] = (\text{ENT}[n-3] + \text{ENT}[n-2] + \text{ENT}[n-1] + \text{ENT}[n])/4 = 0$$
$$\text{SAL}[4] = (\text{ENT}[1] + \text{ENT}[2] + \text{ENT}[3] + \text{ENT}[4])/4 = 0$$
$$\text{SAL}[5] = (\text{SAL}[2] + \text{SAL}[3] + \text{SAL}[4] + \text{SAL}[5])/4 = 2.5$$

(Observe que este cálculo equivale a multiplicar cada muestra por ¼ y sumar.)

11-50. Repita el problema anterior utilizando un promedio ponderado de las últimas cuatro muestras. Las ponderaciones en este caso ponen mucho énfasis en las muestras recientes y menos énfasis en las muestras más antiguas. Use las ponderaciones 0.1, 0.2, 0.3 y 0.4.

$$\text{SAL}[n] = 0.1(\text{ENT}[n-3]) + 0.2(\text{ENT}[n-2]) + 0.3(\text{ENT}[n-1]) + 0.4(\text{ENT}[n])$$
$$\text{SAL}[5] = 0.1(\text{ENT}[2]) + 0.2(\text{ENT}[3]) + 0.3(\text{ENT}[4]) + 0.4(\text{ENT}[5]) = 4$$

11-51. ¿Qué significa el término MAC?

11-52.* **PREGUNTA DE PRÁCTICA**

Verdadero o falso:

(a) Una señal digital es un voltaje que cambia en forma continua.

(b) Una señal digital es una secuencia de números que representan una señal analógica.

Cuando se procesa una señal analógica, la salida puede distorsionarse debido a:

(a) El error de cuantización cuando se convierte de analógica a digital.

(b) No muestrear la señal original con la suficiente frecuencia.

(c) Una variación de temperatura en los componentes del procesador.

(d) Los componentes de alta frecuencia asociados con cambios repentinos en el voltaje de salida del DAC.

(e) Ruido eléctrico en la fuente de energía.

(f) Señales fantasma introducidas por el sistema digital.

RESPUESTAS A LAS PREGUNTAS DE REPASO DE SECCIÓN

SECCIÓN 11-1

1. Convierte una cantidad física no eléctrica en una cantidad eléctrica. 2. Convierte un voltaje o corriente analógica en una representación digital. 3. Los almacena; realiza cálculos o alguna otra operación con ellos. 4. Convierte los datos digitales en su representación analógica. 5. Controla una variable física de acuerdo con una señal de entrada eléctrica.

SECCIÓN 11-2

1. 40 μA; 10.2 mA 2. 5.12 mA 3. 0.39 por ciento 4. 4096 5. 12 6. Verdadero 7. Produce un mayor número de salidas analógicas posibles entre 0 y escala completa.

SECCIÓN 11-3

1. Utiliza sólo dos tamaños distintos de resistencias. 2. 640 kΩ 3. 0.5 V 4. Se incrementa un 20 por ciento.

SECCIÓN 11-4

1. La desviación máxima de la salida de un DAC en relación con su valor ideal; se expresa como un porcentaje de la escala completa. 2. El tiempo que requiere la salida del DAC para estabilizarse en un intervalo no mayor de ½ tamaño de escalón de su valor de escala completa, cuando la entrada digital cambia de 0 a escala completa. 3. El error de desplazamiento agrega un pequeño valor constante positivo o negativo a la salida analógica esperada para cualquier entrada digital. 4. Debido al tiempo de respuesta del amplificador operacional convertidor de corriente a voltaje.

SECCIÓN 11-8

1. Indica a la lógica de control cuando la salida del DAC se excede de la entrada analógica. 2. En las salidas del registro. 3. Nos indica cuando se completa la conversión y el equivalente digital de V_A se encuentra en las salidas del registro.

SECCIÓN 11-9

1. La entrada digital parea un DAC se incrementa hasta que la salida de escalera del DAC se excede de la entrada analógica. 2. El error integrado que se produce debido al hecho de que V_{AX} no se incrementa en forma continua, sino que avanza en intervalos iguales a la resolución del DAC. El V_{AX} final puede ser distinto de V_A hasta por un tamaño de escalón. 3. Si V_A se incrementa, se requerirán más intervalos para que V_{AX} pueda llegar al intervalo que se exceda primero de V_A. 4. Verdadero 5. Circuito simple; tiempo de conversión relativamente largo que cambia con V_A. 6. $0010000111_2 = 135_{10}$ para ambos casos.

SECCIÓN 11-10

1. Es el proceso de convertir distintos puntos en una señal analógica en digitales y almacenar los datos digitales para usarlos después. 2. La computadora genera una señal INICIO para comenzar la conversión A/D de la señal analógica. Cuando *EOC* cambia a BAJO, indica a la computadora que se completó la conversión. Después la computadora carga la salida del ADC en memoria. El proceso se repite para el siguiente punto en la señal analógica. 3. El doble de la frecuencia más alta en la señal de entrada.
4. Una frecuencia alias estará presente en la entrada.

SECCIÓN 11-11

1. El SAC tiene un tiempo de conversión más corto que no cambia con V_A. 2. Tiene una lógica de control más compleja. 3. Falso 4. (a) 8 (b) 0-5 V (c) $\overline{CS}$ controla el efecto de las señales $\overline{RD}$ y $\overline{WR}$; $\overline{WR}$ se utiliza para iniciar una nueva conversión; $\overline{RD}$ habilita los búferes de salida. (d) Cuando está en BAJO, indica el final de una conversión. (e) Separa

la tierra digital (que por lo común contiene ruido) de la tierra analógica, de manera que no se contamine la señal de entrada analógica. (f) Todos los voltajes analógicos en $V_{ENT}(+)$ se miden con referencia a esta terminal. Esto permite que el intervalo de entrada tenga un desplazamiento con respecto a tierra.

SECCIÓN 11-12

1. Verdadero 2. 4095 comparadores y 4096 resistencias. 3. Su principal ventaja es la velocidad de conversión; la desventaja es el número de componentes requeridos en el circuito para una resolución práctica.

SECCIÓN 11-13

1. Reduce el tiempo de conversión mediante el uso de un contador ascendente/descendente, el cual permite que V_{AX} rastree a V_A sin empezar desde 0 para cada conversión.
2. Un VCO 3. Ventajas: bajo costo, inmunidad a la temperatura; desventaja: tiempo de conversión lento. 4. ADC tipo Flash, ADC de voltaje a frecuencia y ADC de pendiente dual. 5. Uno

SECCIÓN 11-14

1. Toma una muestra de una señal de voltaje analógica y la almacena en un capacitor. 2. Falso; son búferes con ganancia unitaria, alta impedancia de entrada y baja impedancia de salida.

SECCIÓN 11-15

1. Utiliza un solo ADC. 2. Se convertiría en un contador MOD-8.

SECCIÓN 11-16

1. Las formas de onda digitalizadas se almacenan en el bloque de memoria. 2. El ADC digitaliza los puntos en la forma de onda de entrada para almacenarlos en memoria; el DAC vertical convierte los puntos de datos almacenados nuevamente en voltajes analógicos para producir la deflexión vertical del haz de electrones; el DAC horizontal produce un voltaje de barrido de escalera, el cual proporciona la deflexión horizontal del haz de electrones.

SECCIÓN 11-17

1. El filtrado de señales analógicas. 2. Un convertidor A/D. 3. Para cambiar su respuesta dinámica sólo hay que cambiar los números en el programa de software, no los componentes de hardware. 4. La unidad de multiplicación y acumulación (MAC) 5. 3; 7

CAPÍTULO 12

DISPOSITIVOS DE MEMORIA

CONTENIDO

12-1 Terminología de memoria
12-2 Operación general de la memoria
12-3 Conexiones entre CPU y memoria
12-4 Memorias de sólo lectura
12-5 Arquitectura de la ROM
12-6 Sincronización de la ROM
12-7 Tipos de ROMs
12-8 Memoria Flash
12-9 Aplicaciones de la ROM
12-10 RAM semiconductora
12-11 Arquitectura de la RAM
12-12 RAM estática (SRAM)
12-13 RAM dinámica (DRAM)
12-14 Estructura y operación de la RAM dinámica
12-15 Ciclos de lectura/escritura de la DRAM
12-16 Refresco de la DRAM
12-17 Tecnología de la DRAM
12-18 Expansión de tamaño de palabra y capacidad
12-19 Funciones especiales de la memoria
12-20 Diagnóstico de fallas en sistemas de RAM
12-21 Prueba de la ROM

OBJETIVOS

Al terminar este capítulo, usted podrá:

- Comprender y utilizar en forma correcta la terminología asociada con los sistemas de memoria.
- Describir la diferencia entre memoria de lectura/escritura y memoria de sólo lectura.
- Hablar sobre la diferencia entre memoria volátil y no volátil.
- Determinar la capacidad de un dispositivo de memoria con base en sus entradas y salidas.
- Describir los pasos que ocurren cuando la CP lee o escribe en memoria.
- Distinguir entre los diversos tipos de ROMs y citar algunas aplicaciones comunes.
- Comprender y describir la organización y operación de las RAMs estáticas y dinámicas.
- Comparar las ventajas y desventajas relativas de la EPROM, EEPROM y la memoria Flash.
- Combinar CIs de memoria para formar módulos de memoria con un tamaño de palabra más grande y/o mayor capacidad.
- Utilizar los resultados de las pruebas en un sistema de RAM o ROM para determinar las posibles fallas en el sistema de memoria.

INTRODUCCIÓN

Una de las principales ventajas de los sistemas digitales sobre los analógicos es su habilidad para almacenar con facilidad grandes cantidades de información y datos digitales, durante periodos cortos o largos. Esta capacidad de memoria es lo que hace a los sistemas digitales tan versátiles y adaptables a muchas situaciones. Por ejemplo, en una computadora digital la memoria principal interna almacena instrucciones que indican a la computadora lo que debe hacer bajo *todas* las posibles circunstancias, de manera que la computadora pueda realizar su trabajo con una mínima cantidad de intervención humana.

En este capítulo estudiaremos de los tipos de dispositivos y sistemas de memoria que se utilizan con más frecuencia. Ya nos hemos familiarizado bastante con el flip-flop, que es un dispositivo de memoria electrónico. También hemos visto cómo pueden utilizarse grupos de FFs, a los cuales se les llama *registros*, para almacenar información y cómo esta información puede transferirse hacia otras ubicaciones. Los registros de FF son elementos de memoria de alta velocidad, los cuales se utilizan mucho en las operaciones internas de una computadora digital, en donde la información digital se desplaza continuamente de una ubicación a otra. Los avances en las tecnologías LSI y VLSI han hecho posible colocar grandes

cantidades de FFs en un solo chip, ordenados en diversos formatos de arreglos de memoria. Estas memorias semiconductoras bipolares y MOS son los dispositivos de memoria más veloces disponibles, y su costo se ha ido reduciendo en forma continua, a medida que mejora la tecnología LSI.

Los datos digitales también pueden almacenarse como cargas en capacitores; y un tipo muy importante de memoria semiconductora utiliza este principio para obtener un almacenamiento de alta densidad con niveles bajos de requerimiento de energía.

Las memorias semiconductoras se utilizan como **memoria principal** de una computadora (figura 12-1), en donde la operación rápida es importante. La memoria principal de una computadora (también conocida como su *memoria de trabajo*) está en comunicación constante con la unidad central de procesamiento (CPU) a medida que se ejecuta un programa de instrucciones. El programa y cualquier información que éste utilice residen en la memoria principal mientras la computadora trabaja con ese programa. La RAM y la ROM (que definiremos en breve) conforman la memoria principal.

La **memoria auxiliar** es otra forma de almacenamiento en la computadora (figura 12-1); esta memoria auxiliar (también conocida como almacenamiento masivo) está separada de la memoria de trabajo principal y tiene la capacidad de almacenar cantidades masivas de datos, sin necesidad de energía eléctrica. La memoria auxiliar opera a una velocidad mucho más lenta que la memoria principal; almacena programas y datos que la CPU no utiliza en ese momento. Esta información se transfiere a la memoria principal cuando la computadora la necesita. Los dispositivos de memoria auxiliar comunes son el disco magnético y el disco compacto (CD).

Veremos con detalle las características de los dispositivos de memoria más comunes que se utilizan como la memoria interna de una computadora. Primero definiremos algunos de los términos comunes que se utilizan en los sistemas de memoria.

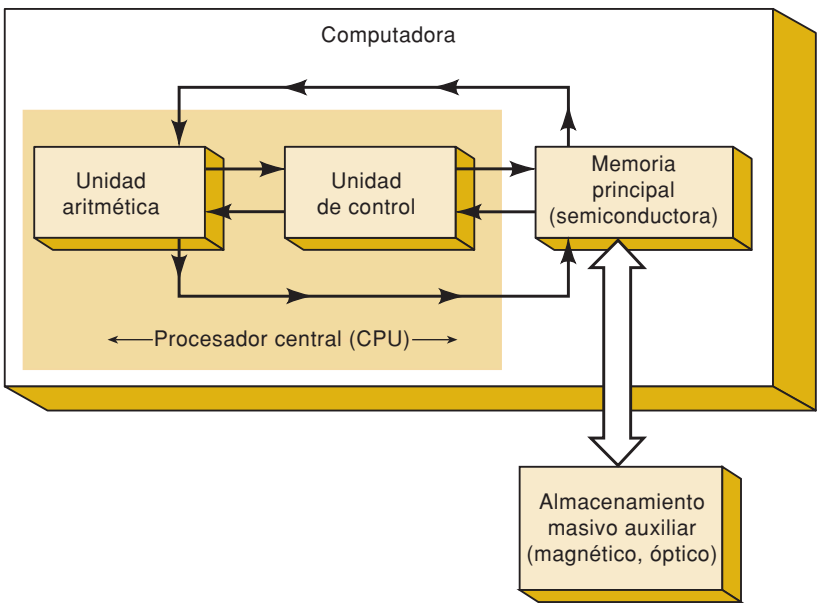


FIGURA 12-1 Un sistema computacional utiliza, por lo general, memoria principal de alta velocidad y memoria auxiliar externa más lenta.

12-1 TERMINOLOGÍA DE MEMORIA

El estudio de los dispositivos y sistemas de memoria está lleno de terminología que algunas veces puede ser abrumadora para un estudiante. Antes de ver una descripción detallada sobre las memorias, sería conveniente que entendiera el significado

de algunos de los términos más elementales. El resto de los términos los definiremos a medida que aparezcan en el capítulo.

- **Celda de memoria.** Un dispositivo o un circuito eléctrico utilizado para almacenar un bit (0 o 1). Algunos ejemplos de celdas de memoria son: un flip-flop, un capacitor cargado y un punto individual en cinta o disco magnético.
- **Palabra de memoria.** Un grupo de bits (celdas) en una memoria, el cual representa instrucciones o datos de cierto tipo. Por ejemplo, un registro que consiste de ocho FFs puede considerarse como una memoria que almacena una palabra de ocho bits. Los tamaños de las palabras en las computadoras modernas varían, por lo general, de 8 a 64 bits, dependiendo del tamaño de la computadora.
- **Byte.** Un término especial que se utiliza para un grupo de ocho bits. Un byte siempre consiste de ocho bits. Los tamaños de las palabras pueden expresarse en bytes o en bits. Por ejemplo, una palabra de un tamaño de ocho bits puede expresarse también cómo una palabra de un byte; una palabra de 16 bits equivale a 2 bytes, y así sucesivamente.
- **Capacidad.** Una manera de especificar cuántos bits pueden almacenarse en un dispositivo de memoria específico o en un sistema completo de memoria. Por ejemplo, suponga que tenemos una memoria que puede almacenar 4096 palabras de 20 bits. Esto representa una capacidad total de 81,920 bits. También podríamos expresar la capacidad de esta memoria como 4096 × 20. Cuando se expresa de esta manera, el primer número (4096) es el de palabras y el segundo número (20) es el de bits por palabra (tamaño de palabra). El número de palabras en una memoria es por lo común un múltiplo de 1024. Es común utilizar la designación "1K" para representar $1024 = 2^{10}$ cuando se hace referencia a la capacidad de la memoria. Por lo tanto, una memoria que tiene una capacidad de almacenamiento de 4K × 20 es en realidad una memoria de 4096 × 20. El desarrollo de memorias más grandes trajo consigo la designación "1M" o "1 meg" para representar $2^{20} = 1{,}048{,}576$. Por lo tanto, una memoria con una capacidad de 2M × 8 tiene en realidad una capacidad de 2,097,152 × 8. La designación "giga" se refiere a $2^{30} = 1{,}073{,}741{,}824$.

EJEMPLO 12-1A

Cierto chip de memoria semiconductora se especifica como de 2 K × 8. ¿Cuántas palabras pueden almacenarse en este chip? ¿Cuál es el tamaño de palabra? ¿Cuántos bits en total puede almacenar este chip?

Solución

$$2\text{K} = 2 \times 1024 = 2048 \text{ palabras}$$

Cada palabra es de ocho bits (un byte). Por lo tanto, el número total de bits es

$$2048 \times 8 = 16{,}384 \text{ bits}$$

EJEMPLO 12-1B

¿Qué memoria almacena más bits: una de 5M × 8 o una que almacena 1M palabras con un tamaño de palabra de 16 bits?

Solución

$$5\text{M} \times 8 = 5 \times 1{,}048{,}576 \times 8 = 41{,}943{,}010 \text{ bits}$$
$$1\text{M} \times 16 = 1{,}048{,}576 \times 16 = 16{,}777{,}216 \text{ bits}$$

La memoria de 5M × 8 almacena más bits.

- **Densidad.** Otro término para *capacidad*. Cuando decimos que un dispositivo de memoria tiene una mayor densidad que otra, queremos decir que puede almacenar más bits en la misma área de silicio; es más densa.
- **Dirección.** Un número que identifica la ubicación de una palabra en memoria. Cada palabra que se almacena en un dispositivo o sistema de memoria tiene una dirección única. Las direcciones siempre existen en un sistema digital en forma de un número binario, aunque a menudo se utilizan números octales, hexadecimales y decimales para representar la dirección. La figura 12-2 ilustra una pequeña memoria que consiste de ocho palabras. Cada una de estas ocho palabras tiene una dirección específica, la cual se representa como un número de tres bits que varía de 000 a 111. Cada vez que hacemos referencia a la ubicación de una palabra específica en memoria, utilizamos su código de dirección para identificarla.
- **Operación de lectura.** La operación mediante la cual la palabra binaria almacenada en una ubicación específica de memoria (dirección) se detecta y después se transfiere hacia otro dispositivo. Por ejemplo, si queremos utilizar la palabra 4 de la memoria de la figura 12-2 para cierto propósito, debemos realizar una operación de lectura en la dirección 100. A la operación de lectura, por lo general, se le conoce como operación *buscar (fetch)*, ya que se está buscando una palabra en memoria. Utilizaremos ambos términos de manera indistinta.
- **Operación de escritura.** La operación mediante la cual se coloca una nueva palabra en una ubicación de memoria específica. También se le conoce como operación *almacenar*. Cada vez que se escribe una nueva palabra en una ubicación de memoria, ésta sustituye a la palabra que estaba ahí almacenada.
- **Tiempo de acceso.** Una medida de la velocidad de operación de un dispositivo de memoria. Es la cantidad de tiempo que se requiere para realizar una operación de lectura. Dicho de manera más específica, es el tiempo desde que la memoria recibe una nueva dirección de entrada y hasta que los datos están disponibles en la salida de la memoria. Se utiliza el símbolo t_{ACC} para el tiempo de acceso.
- **Memoria volátil.** Cualquier tipo de memoria que requiere la aplicación de energía eléctrica para poder almacenar información. Si se quita la energía eléctrica, se perderá toda la información almacenada en la memoria. Muchas memorias semiconductoras son volátiles, mientras que todas las memorias magnéticas son *no volátiles*, lo cual significa que pueden almacenar información sin energía eléctrica.
- **Memoria de acceso aleatorio (RAM).** Memoria en la cual la ubicación física de una palabra de memoria no tiene efecto sobre el tiempo que tardan las opera-

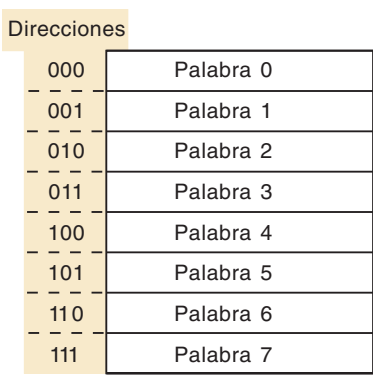


FIGURA 12-2 La ubicación de cada palabra tiene una dirección binaria específica.

ciones de lectura o escritura, en una ubicación determinada. En otras palabras, el tiempo de acceso es igual para cualquier dirección en memoria. La mayoría de las memorias semiconductoras son del tipo RAM.

- **Memoria de acceso secuencial (SAM).** Un tipo de memoria en la cual el tiempo de acceso no es constante, sino que varía dependiendo de la ubicación de la dirección. Una palabra específica que esté almacenada se encuentra mediante la búsqueda secuencial a través de todas las ubicaciones de las direcciones, hasta que se llega a la dirección deseada. Esto produce tiempos de acceso mucho más largos que los de las memorias de acceso aleatorio. Un ejemplo de un dispositivo de memoria de acceso secuencial es el respaldo en cinta magnética. Para ilustrar la diferencia entre SAM y RAM, considere la situación en la que se graban 60 minutos de canciones en un cassette de audio. Cuando deseamos escuchar una canción específica, tenemos que rebobinar o adelantar la cinta hasta encontrarla. El proceso es relativamente lento y la cantidad de tiempo requerido depende del lugar en la cinta en el que esté grabada la canción deseada. Esto es igual que la SAM, ya que hay que avanzar en secuencia a través de toda la información hasta encontrar lo que estamos buscando. La contraparte de la RAM para esto sería un CD de audio, en donde se puede seleccionar con rapidez cualquier canción con sólo introducir el código apropiado, y se requiere aproximadamente el mismo tiempo sin importar cuál canción seleccione. Las memorias de acceso secuencial se utilizan cuando los datos a los que se va a acceder siempre vienen en una larga secuencia de palabras sucesivas. Por ejemplo, la memoria de video debe enviar su contenido en el mismo orden una y otra vez, para regenerar la imagen en la pantalla de CRT.

- **Memoria de lectura/escritura (RWM).** Cualquier memoria que se puede leer, o en la que se puede escribir, con la misma facilidad.

- **Memoria de sólo lectura (ROM).** Una amplia variedad de memorias semiconductoras, diseñadas para aplicaciones en las que la proporción de operaciones de lectura en comparación con las de escritura es muy alta. Técnicamente, se puede escribir (programar) en una ROM sólo una vez, y esta operación, por lo general, se realiza en la fábrica. Después de eso, la información sólo se puede leer de la memoria. Otros tipos de ROM son en realidad memorias de casi siempre lectura (RMM), en las cuales se puede escribir más de una vez; pero la operación de escritura es más complicada que la operación de lectura, por lo cual no se realiza con mucha frecuencia. Más adelante hablaremos sobre los diversos tipos de ROM. *Toda la ROM es no volátil* y almacenará los datos aunque se retire la energía eléctrica.

- **Dispositivos de memoria estática.** Los dispositivos de memoria semiconductora en la cual los datos se almacenarán de manera permanente, siempre y cuando se aplique energía, sin necesidad de reescribir los datos en la memoria en forma periódica.

- **Dispositivos de memoria dinámica.** Dispositivos de memoria semiconductora en los cuales los datos *no* se almacenarán en forma permanente, aún y cuando se le aplique energía, a menos que se reescriban en forma periódica en la memoria. A esta última operación se le conoce como operación de *regeneración*.

- **Memoria principal.** También se le conoce como *memoria de trabajo* de la computadora. Almacena instrucciones y datos con los que la CPU está trabajando en un momento dado. Es la de mayor velocidad en la computadora y siempre es semiconductora.

- **Memoria auxiliar.** También se le conoce como *almacenamiento masivo*, ya que guarda cantidades masivas de información externa a la memoria principal. Es más lenta en velocidad que la memoria principal y siempre es no volátil. Los discos magnéticos y los CDs son dispositivos comunes de memoria auxiliar.

PREGUNTAS DE REPASO

1. Defina los siguientes términos.
   (a) *Celda de memoria.*
   (b) *Palabra de memoria.*
   (c) *Dirección.*
   (d) *Byte.*
   (e) *Tiempo de acceso.*
2. Cierta memoria tiene una capacidad de 8K × 16. ¿Cuántos bits hay en cada palabra? ¿Cuántas palabras se almacenan? ¿Cuántas celdas de memoria contiene esta memoria?
3. Explique la diferencia entre las operaciones leer (buscar) y escribir (almacenar).
4. *Verdadero o falso:* una memoria volátil pierde sus datos almacenados cuando se interrumpe la energía eléctrica.
5. Explique la diferencia entre SAM y RAM.
6. Explique la diferencia entre RWM y ROM.
7. *Verdadero o falso:* una memoria dinámica retendrá sus datos mientras se le aplique energía eléctrica.

12.2 OPERACIÓN GENERAL DE LA MEMORIA

Aunque cada tipo de memoria es distinto en su operación interna, ciertos principios de operación básicos son iguales para todos los sistemas de memoria. Comprender estas ideas básicas le ayudará en el estudio de los dispositivos de memoria individuales.

Todo sistema de memoria requiere varios tipos distintos de líneas de entrada y de salida para realizar las siguientes funciones:

1. Seleccionar la dirección en memoria a la que se va a acceder para una operación de lectura o de escritura.
2. Seleccionar una operación de lectura o de escritura a realizar.
3. Suministrar los datos de entrada que se van a almacenar en memoria durante una operación de escritura.
4. Retener los datos de entrada que provienen de la memoria durante una operación de lectura.
5. Habilitar (o deshabilitar) la memoria, de manera que responda (o no) a las entradas de dirección y a la línea de selección de lectura/escritura.

La figura 12-3(a) ilustra estas funciones básicas en un diagrama simplificado de una memoria de 32 × 4, la cual almacena 32 palabras de cuatro bits. Como el tamaño de palabra es de cuatro bits, hay cuatro líneas de entrada de datos (I_0 a I_3) y cuatro líneas de salida de datos (O_0 a O_3). Durante una operación de escritura, los datos que se van a almacenar en memoria deben aplicarse a las líneas de entrada de datos. Durante una operación de lectura, la palabra que se va a leer de memoria aparece en las líneas de salida de datos.

Entradas de dirección

Como esta memoria almacena 32 palabras, tiene 32 distintas ubicaciones de almacenamiento y, por lo tanto, 32 distintas direcciones binarias, las cuales varían de 00000 a 11111 (de 0 a 31 en decimal). Por ende, hay cinco entradas de dirección (A_0 a A_4).

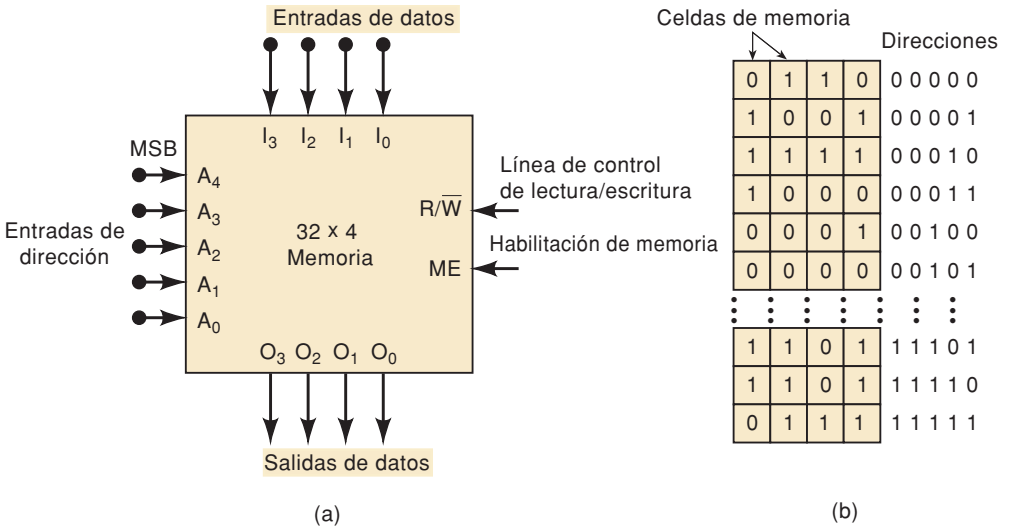


FIGURA 12-3
(a) Diagrama de una memoria de 32 × 4; (b) arreglo virtual de las celdas de memoria en 32 palabras de cuatro bits.

Para acceder a una de las ubicaciones de memoria para una operación de lectura o de escritura se aplica el código de dirección de cinco bits para esa ubicación específica a las entradas de dirección. En general, se requieren N entradas de dirección para una memoria con una capacidad de 2^N palabras.

Podemos visualizar la memoria de la figura 12-3(a) como un arreglo de 32 registros, en donde cada registro almacena una palabra de cuatro bits, como se muestra en la figura 12-3(b). La ubicación de cada dirección que se muestra contiene cuatro celdas de memoria que retienen 1s y 0s, los cuales conforman la palabra de datos que se almacena en esa ubicación. Por ejemplo, la palabra de datos 0110 se almacena en la dirección 00000, la palabra de datos 1001 se almacena en la dirección 00001, y así en lo sucesivo.

La entrada $R/\overline{W}$

Esta entrada controla qué operación de memoria se va a realizar: lectura (R) o escritura (W). La entrada se identifica como $R/\overline{W}$; no hay barra sobre la R, lo cual indica que la operación de lectura ocurre cuando $R/\overline{W} = 1$. La barra sobre la W indica que la operación de escritura se realiza cuando $R/\overline{W} = 0$. A menudo se utilizan otras etiquetas para identificar esta entrada. Dos de las más comunes son $\overline{W}$ (escritura) y $\overline{WE}$ (habilita escritura). De nuevo, la barra indica que la operación de escritura se lleva a cabo cuando la entrada está en BAJO. En estos últimos casos se sobreentiende que la operación de lectura ocurre cuando la entrada está en ALTO.

La figura 12-4 muestra una ilustración simplificada de las operaciones de lectura y de escritura. La figura 12-4(a) muestra cómo se escribe la palabra de datos 0100 en el registro de memoria que se encuentra en la ubicación de dirección 00011. Esta palabra de datos se hubiera aplicado a las líneas de entrada de datos de la memoria, y sustituye los datos que estaban almacenados antes en la dirección 00011. La figura 12-4(b) muestra cómo se lee la palabra de datos 1101 de la dirección 11110. Esta palabra de datos aparecería en las líneas de salida de datos de la memoria. Después de la operación de lectura, la palabra de datos 1101 sigue almacenada en la dirección 11110. En otras palabras, la operación de lectura no modifica los datos almacenados.

Habilitación de memoria

Muchos sistemas de memora tienen ciertos medios para deshabilitar por completo toda o parte de la memoria, de manera que no responda a las demás entradas. Esto se representa en la figura 12-3 como la entrada HABILITACIÓN DE MEMORIA (MEMORY ENABLE), aunque puede tener distintos nombres en los diversos sistemas de memoria, como habilitación de chip (*CE*, acrónimo de CHIP ENABLE)

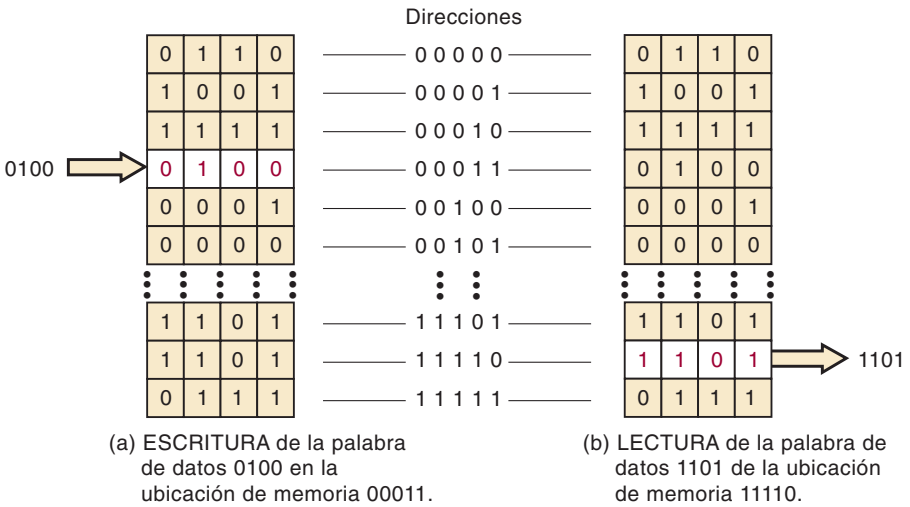


FIGURA 12-4
Ilustración simplificada de las operaciones de lectura y de escritura en la memoria de 32 × 4; (a) escritura de la palabra de datos 0100 en la ubicación de memoria 00011; (b) lectura de la palabra de datos 1101 de la ubicación de memoria 11110.

o selección de chip (*CS*, acrónimo de CHIP SELECT). Aquí se muestra como una entrada activa en ALTO, la cual habilita a la memoria para que opere en forma normal cuando se mantiene en ALTO. Un nivel BAJO en esta entrada deshabilita la memoria, de manera que no pueda responder a las entradas de dirección y $R/\overline{W}$.

Este tipo de entrada es útil cuando se combinan varios módulos de memoria para formar una memoria más grande. Más adelante examinaremos esta idea.

EJEMPLO 12-2

Describa las condiciones en cada entrada y salida cuando se lee el contenido de la ubicación de dirección 00100.

Solución

Entradas de dirección: 00100
Entradas de datos: xxxx (no se utilizan)
$R/\overline{W}$: ALTO
HABILITACIÓN DE MEMORIA: ALTO
Salidas de datos: 0001

EJEMPLO 12-3

Describa las condiciones en cada entrada y salida cuando se escribe la palabra de datos 1110 en la ubicación de dirección 01101.

Solución

Entradas de dirección: 01101
Entradas de datos: 1110
$R/\overline{W}$: BAJO
HABILITACIÓN DE MEMORIA: ALTO
Salidas de datos: xxxx (no se utilizan; por lo general, están en Hi-Z)

EJEMPLO 12-4

Cierta memoria tiene una capacidad de 4K × 8.

(a) ¿Cuántas líneas de entrada y salida de datos tiene?
(b) ¿Cuántas líneas de dirección tiene?
(c) ¿Cuál es su capacidad en bytes?

Solución

(a) Ocho de cada una, ya que el tamaño de palabra es de ocho.

(b) La memoria almacena 4K = 4 × 1024 = 4096 palabras. Por lo tanto, hay 4096 direcciones de memoria. Como $4096 = 2^{12}$, se requiere un código de dirección de 12 bits para especificar una de 4096 direcciones.

(c) Un byte es de ocho bits. Esta memoria tiene una capacidad de 4096 bytes.

La memoria de ejemplo de la figura 12-3 ilustra las funciones de entrada y salida que son comunes para la mayoría de los sistemas de memoria. Desde luego que cada tipo de memoria pude tener otras líneas de entrada y de salida que sean peculiares para esa memoria. Describiremos estas líneas adicionales de entrada y salida cuando veamos los tipos de memoria individuales.

PREGUNTAS DE REPASO

1. ¿Cuántas entradas de dirección, entradas de datos y salidas de datos se requieren para una memoria de 16K × 12?
2. ¿Cuál es la función de la entrada $R/\overline{W}$?
3. ¿Cuál es la función de la entrada HABILITACIÓN DE MEMORIA?

12-3 CONEXIONES ENTRE CPU Y MEMORIA

Una buena parte de este capítulo se dedica al análisis de la memoria semiconductora, que como vimos antes, conforma la memoria principal de la mayoría de las computadoras modernas. Recuerde que esta memoria principal se encuentra en comunicación constante con la unidad central de procesamiento (CPU). No es necesario estar familiarizado con la operación detallada de una CPU en este punto, por lo que el siguiente tratamiento simplificado de la interfaz entre CPU y memoria le proveerá la perspectiva que necesita para que nuestro estudio de los dispositivos de memoria sea más enriquecedor.

La memoria principal de una computadora está compuesta por CIs tipo RAM y ROM, los cuales se conectan a la CPU a través de tres grupos de líneas de señal o buses, los cuales se representan en la figura 12-5 como las líneas de dirección o bus de dirección, las líneas de datos o el bus de datos, y las líneas de control o el bus de control. Cada uno de estos buses consiste de varias líneas (observe que se representan mediante una sola línea con una barra diagonal), y el número de líneas en cada bus varía de una computadora a otra. Los tres buses juegan un papel necesario para permitir que la CPU escriba datos en la memoria y lea datos de ella.

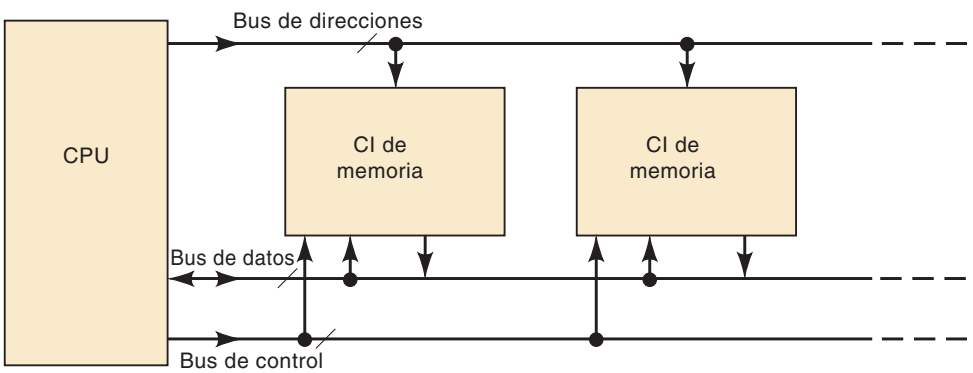


FIGURA 12-5 Tres grupos de líneas (buses) conectan a los CIs de la memoria principal con el CPU.

Cuando una computadora ejecuta un programa de instrucciones, la CPU busca en forma continua (lee) información de esas ubicaciones en memoria que contienen (1) los códigos del programa que representan las operaciones a realizar y (2) los datos sobre los cuáles se va a trabajar. La CPU también almacenará (escribirá) datos en ubicaciones de memoria, según lo dicten las instrucciones del programa. Cada vez que la CPU desea escribir datos a una ubicación de memoria específica, deben realizarse los siguientes pasos:

Operación de escritura

1. La CPU suministra la dirección binaria de la ubicación de memoria en la que se van a almacenar los datos. Coloca esta dirección en las líneas del bus de dirección.
2. La CPU coloca los datos que se van a almacenar en las líneas del bus de datos.
3. La CPU activa las líneas de señal de control apropiadas para la operación de escritura en la memoria.
4. Los CIs de la memoria decodifican la dirección binaria para determinar qué ubicación se está seleccionando para la operación de almacenamiento.
5. Los datos en el bus de datos se transfieren hacia la ubicación de memoria seleccionada.

Cada vez que la CPU desea leer datos de una ubicación específica en memoria, deben realizarse los siguientes pasos:

Operación de lectura

1. La CPU suministra la dirección binaria de la ubicación de memoria de la cual se van a recuperar los datos. Coloca esta dirección en las líneas del bus de dirección.
2. La CPU activa las líneas de señal de control apropiadas para la operación de lectura de la memoria.
3. Los CIs de la memoria decodifican la dirección binaria para determinar cuál ubicación se está seleccionando para la operación de lectura.
4. Los CIs de memoria colocan los datos de la ubicación de memoria seleccionada en el bus de datos, desde donde se van a transferir hacia la CPU.

Los pasos anteriores deben dejar en claro la función de cada uno de los buses del sistema:

- **Bus de dirección.** Este bus *unidireccional* transporta las salidas de la dirección binaria del CPU hacia los CIs de memoria, para seleccionar una ubicación de memoria.
- **Bus de datos.** Este bus *bidireccional* transporta los datos entre el CPU y los CIs de memoria.
- **Bus de control.** Este bus transporta las señales de control (tales como la señal $R/\overline{W}$) desde el CPU hacia los CIs de memoria.

A medida que profundicemos sobre los CIs de memoria reales, examinaremos la actividad de las señales que aparecen en estos buses para las operaciones de lectura y de escritura.

PREGUNTAS DE REPASO

1. Nombre los tres grupos de líneas que conectan a la CPU y con la memoria interna.
2. Describa los pasos que se llevan a cabo cuando la CPU lee de memoria.
3. Describa los pasos que se llevan a cabo cuando la CPU escribe en memoria.

12-4 MEMORIAS DE SÓLO LECTURA

La memoria de sólo lectura es un tipo de memoria semiconductora, diseñada para retener datos que sean permanentes o que no cambien con frecuencia. Durante la operación normal, no pueden escribirse datos en una ROM, pero sí se pueden leer. Para algunas ROMs, los datos almacenados deben integrarse durante el proceso de manufactura; para otras ROMs los datos pueden introducirse mediante electricidad. Al proceso de introducir datos se le llama **programación** o *quemado* de la ROM. Algunas ROMs no pueden cambiar sus datos una vez programadas; otras pueden *borrarse* y reprogramarse tantas veces como se requiera. Más adelante veremos con detalle estos diversos tipos de ROMs. Por ahora supondremos que las ROMs se han programado y que contienen datos.

Las ROMs se utilizan para almacenar datos e información que no va a cambiar durante la operación normal de un sistema. Un uso principal para las ROMs es el almacenamiento de programas en las microcomputadoras. Como todas las ROMs son *no volátiles*, estos programas no se pierden cuando se desconecta la energía eléctrica. Cuando la microcomputadora se enciende, puede comenzar de inmediato a ejecutar el programa almacenado en la ROM. Estas memorias también se utilizan para almacenar programas y datos en equipos controlados por microprocesador, tales como: cajas registradoras, electrodomésticos y sistemas de seguridad.

Diagrama de bloques de la ROM

La figura 12-6(a) muestra un diagrama de bloques típico para una ROM. Tiene tres conjuntos de señales: entradas de dirección, entrada(s) de control y salidas de datos. De nuestros análisis anteriores podemos determinar que esta ROM almacena 16 palabras, ya que tiene $2^4 = 16$ direcciones posibles, y cada palabra contiene ocho bits ya que hay ocho salidas de datos. Es decir, ésta es una ROM de 16×8. Otra manera de describir la capacidad de esta ROM es decir que almacena 16 bytes de datos.

Las salidas de datos de la mayoría de los CIs de ROM son triestado, para permitir la conexión de muchos chips de ROM al mismo bus de datos para expandir la memoria. Los números más comunes de salidas de datos para las ROMs son cuatro, ocho y 16 bits, siendo más comunes las palabras de ocho bits.

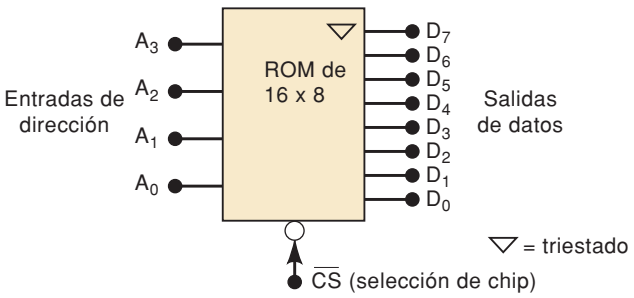
La entrada de control $\overline{CS}$ significa **selección de chip**. En esencia es una entrada de habilitación, para habilitar o deshabilitar las salidas de la ROM. Algunos fabricantes utilizan distintas etiquetas para la entrada de control, como *CE* (habilitación de chip) u *OE* (habilitación de salida). Muchas ROMs tienen dos o más entradas de control que deben estar activas para poder habilitar las salidas de datos, de manera que éstos pueden leerse de la dirección seleccionada. En algunos CIs de ROM, una de las entradas de control (por lo general *CE*) se utiliza para colocar la ROM en un modo de espera de baja energía cuando no está en uso. Esto reduce el consumo de corriente de la fuente de energía del sistema.

La entrada $\overline{CS}$ que se muestra en la figura 12-6(a) es activa en BAJO; por lo tanto, debe encontrarse en el estado BAJO para permitir que aparezcan los datos de la ROM en las salidas de datos. Observe que no hay entrada $R/\overline{W}$ (lectura/escritura) ya que no se puede escribir en la ROM durante su operación normal.

La operación de lectura

Supongamos que la ROM se programó con los datos que se muestran en la tabla de la figura 12-6(b). Se almacenan dieciséis palabras de datos distintas en las 16 ubicaciones de dirección. Por ejemplo, la palabra de datos almacenada en la ubicación 0011 es 10101111. Desde luego que los datos se almacenan en binario dentro de la ROM, pero muy a menudo utilizamos notación hexadecimal para mostrar los datos programados con eficiencia. Esto se hace en la figura 12-6(c).

Para poder leer una palabra de datos de la ROM, necesitamos hacer dos cosas: (1) aplicar las entradas de dirección apropiadas y, después, (2) activar las entradas de control. Por ejemplo, si deseamos leer los datos almacenados en la ubicación



(a)

	Dirección				Datos							
Palabra	A_3	A_2	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	1	1	0	1	1	1	1	0
1	0	0	0	1	0	0	1	1	1	0	1	0
2	0	0	1	0	1	0	0	0	0	1	0	1
3	0	0	1	1	1	0	1	0	1	1	1	1
4	0	1	0	0	0	0	0	1	1	0	0	1
5	0	1	0	1	0	1	1	1	1	0	1	1
6	0	1	1	0	0	0	0	0	0	0	0	0
7	0	1	1	1	1	1	1	0	1	1	0	1
8	1	0	0	0	0	0	1	1	1	1	0	0
9	1	0	0	1	1	1	1	1	1	1	1	1
10	1	0	1	0	1	0	1	1	1	0	0	0
11	1	0	1	1	1	1	0	0	0	1	1	1
12	1	1	0	0	0	0	1	0	0	1	1	1
13	1	1	0	1	0	1	1	0	1	0	1	0
14	1	1	1	0	1	1	0	1	0	0	1	0
15	1	1	1	1	0	1	0	1	1	0	1	1

(b)

	Dirección	Datos
Palabra	A_3 A_2 A_1 A_0	D_7 –D_0
0	0	DE
1	1	3A
2	2	85
3	3	AF
4	4	19
5	5	7B
6	6	00
7	7	ED
8	8	3C
9	9	FF
10	A	B8
11	B	C7
12	C	27
13	D	6A
14	E	D2
15	F	5B

(c)

FIGURA 12-6 (a) Símbolo de bloque típico de la ROM; (b) tabla que muestra los datos binarios en cada una de las ubicaciones de memoria; (c) la misma tabla en hexadecimal.

0111 de la ROM en la figura 12-6, debemos aplicar $A_3A_2A_1A_0 = 0111$ a las entradas de dirección y después aplicar un nivel BAJO a $\overline{CS}$. Las entradas de dirección se decodificarán dentro de la ROM para seleccionar la palabra de datos correcta, 11101101, que debe aparecer en las salidas D_7 a D_0. Si $\overline{CS}$ se mantiene en ALTO, las salidas de la ROM se deshabilitarán y quedarán en el estado Hi-Z.

PREGUNTAS DE REPASO

1. *Verdadero o falso:* todas las ROMs son no volátiles.
2. Describa el procedimiento para leer de ROM.
3. ¿Qué es *programación* o *quemado* en una ROM?

12-5 ARQUITECTURA DE LA ROM

La arquitectura (estructura) interna de un CI de ROM es muy compleja, por lo que no intentaremos familiarizarnos con todos sus detalles. No obstante, por cuestión didáctica sí podemos ver un diagrama simplificado de la arquitectura interna, como el que se muestra en la figura 12-7 para la ROM de 16 × 8. Hay cuatro partes básicas: *arreglo de registros, decodificador de fila, decodificador de columna* y *búferes de salida*.

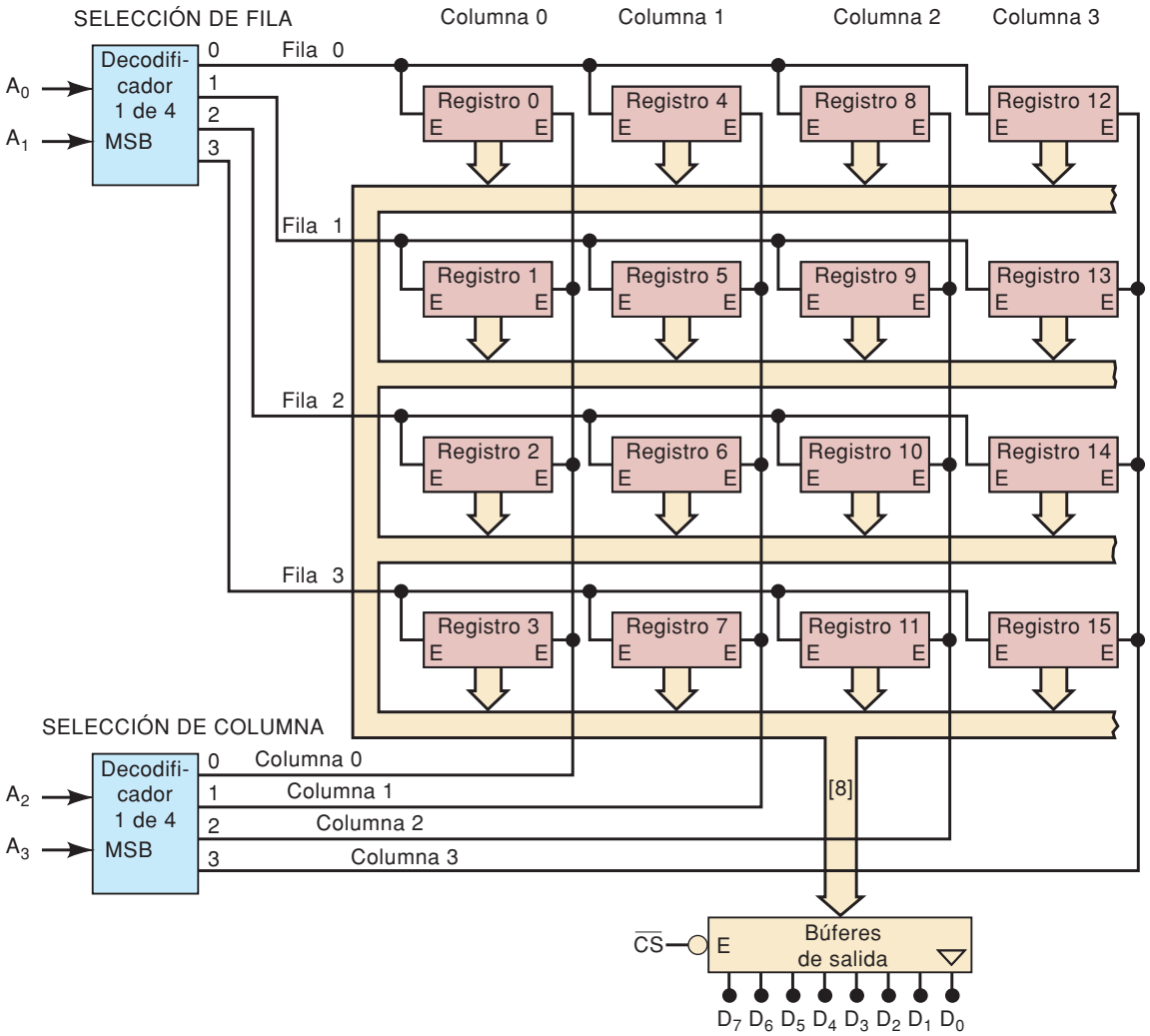


FIGURA 12-7 Arquitectura de una ROM de 16 × 8. Cada registro almacena una palabra de ocho bits

Arreglo de registros

El arreglo de registros almacena los datos que se programan en la ROM. Cada registro contiene varias celdas de memoria que equivalen al tamaño de palabra. En este caso, cada registro almacena una palabra de ocho bits. Los registros se ordenan en un arreglo de matriz cuadrada, común para muchos chips de memoria semiconductora. Podemos especificar que la posición de cada registro está en una fila y una columna específicas. Por ejemplo, el registro 0 está en la fila 0, columna 0, y el registro 9 está en la fila 1, columna 2.

Decodificadores de dirección

El código de dirección $A_3A_2A_1A_0$ que se aplica es el que determina qué registro en el arreglo se habilitará para colocar su palabra de datos de ocho bits en el bus. Los bits de dirección A_1A_0 se alimentan a un decodificador 1 de 4, el cual activa una línea de selección de fila, y los bits de dirección A_3A_2 se alimentan a un segundo decodificador 1 de 4, el cual activa una línea de selección de columna. Sólo habrá un registro

en la fila y la columna seleccionadas por las entradas de dirección, por lo que éste es el que se habilitará.

EJEMPLO 12-5

¿Cuál registro se habilitará a través de la dirección de entrada 1101?

Solución

$A_3A_2 = 11$ hará que el decodificador de columna active la línea de selección de la columna 3, y $A_1A_0 = 01$ hará que el decodificador de fila active la línea de selección de la fila 1. Esto hará que ambas entradas de habilitación del registro 13 cambien a ALTO, para provocar que se coloquen sus salidas de datos en el bus. Observe que los otros registros de la columna 3 tendrán sólo una entrada de habilitación activada; lo mismo se aplica para los registros de la otra fila 1.

EJEMPLO 12-6

¿Qué dirección de entrada habilitará el registro 7?

Solución

Las entradas de habilitación de este registro están conectadas a las líneas de selección de la fila 3 y la columna 1, respectivamente. Para seleccionar la fila 3, las entradas A_1A_0 deben estar en 11 y para seleccionar la columna 1, las entradas A_3A_2 deben estar en 01. Por lo tanto, la dirección requerida será $A_3A_2A_1A_0 = 0111$.

Búferes de salida

El registro que se habilite mediante las entradas de dirección colocará sus datos en el bus de datos. Estos datos se alimentan hacia los búferes de salida, los cuales pasarán esos datos hacia las salidas de datos externas, siempre y cuando $\overline{CS}$ permanezca en BAJO. Si $\overline{CS}$ cambia a ALTO, los búferes de salida se colocarán en el estado Hi-Z y las salidas D_7 a D_0 estarán flotando.

La arquitectura que se muestra en la figura 12-7 es similar a la de muchas ROMs de CI. Dependiendo del número de palabras de datos almacenadas, los registros en algunas ROMs no se ordenarán en un arreglo cuadrado. Por ejemplo, el chip Intel 27C64 es una ROM tipo CMOS que almacena 8192 palabras de ocho bits. Sus 8192 registros están ordenados en un arreglo de 256 filas $\times$ 32 registros. Las capacidades de la ROM varían desde 256 $\times$ 4 hasta 8M $\times$ 8.

EJEMPLO 12-7

Describa la arquitectura interna de una ROM que almacena 4K bytes y utiliza un arreglo de registros cuadrado.

Solución

4K son en realidad 4 $\times$ 1024 = 4096, por lo que esta ROM almacena 4096 palabras de ocho bits. Se puede considerar que cada palabra se almacena en un registro de 8 bits y que hay 4096 registros conectados a un bus de datos común interno del chip. Como $4096 = 64^2$, los registros están ordenados en un arreglo de 64 $\times$ 64; es decir, hay 64 filas y 64 columnas. Esto requiere un decodificador 1 de 64 para decodificar seis entradas de dirección para la selección de fila, y un segundo decodificador 1 de 64 para decodificar otras seis entradas de dirección para la selección de columna. Por lo tanto, se requiere un total de 12 entradas de dirección. Esto tiene sentido, ya que $2^{12} = 4096$ y hay 4096 direcciones distintas.

PREGUNTAS DE REPASO

1. ¿Qué código de dirección de entrada se requiere si queremos leer los datos del registro 9 en la figura 12-7?
2. Describa la función del decodificador de selección de fila, del decodificador de selección de columna y de los búferes de salida en la arquitectura de la ROM.

12-6 SINCRONIZACIÓN DE LA ROM

Durante una operación de lectura se produce un tiempo de propagación entre la aplicación de las entradas de una ROM y la aparición de las salidas de datos. Este tiempo de propagación, conocido como tiempo de acceso (t_{ACC}), es una medida de la velocidad de operación de la ROM. En la figura 12-8 se describe el tiempo en forma gráfica, mediante las formas de onda.

La forma de onda superior representa las entradas de dirección; la forma de onda de en medio es la señal de selección de chip $\overline{CS}$ activa en bajo; y la forma de onda inferior representa las salidas de datos. En el tiempo t_0, todas las entradas de dirección se encuentran en un nivel específico, algunas en ALTO y otras en BAJO. $\overline{CS}$ está en ALTO, por lo que las salidas de datos de la ROM se encuentran en su estado Hi-Z (el cual se representa mediante la línea sombreada).

Justo antes de t_1, las entradas de dirección cambian a una nueva dirección para una nueva operación de lectura. En t_1, la nueva dirección es válida; es decir, cada entrada de dirección se encuentra en un nivel lógico válido. En este punto, el circuito interno de la ROM comienza a decodificar las nuevas entradas de dirección para seleccionar el registro que debe enviar sus datos a los búferes de salida. En t_2 se activa la entrada $\overline{CS}$ para habilitar los búferes de salida. Por último, en t_3 las salidas cambian del estado Hi-Z a los datos válidos que representan los datos almacenados en la dirección especificada.

El retraso de tiempo entre t_1, cuando la nueva dirección se hace válida, y t_3, cuando las salidas de datos se hacen válidas, es el tiempo de acceso t_{ACC}. Las ROMs bipolares ordinarias tendrán tiempos de acceso que varían en el intervalo de 30 a 90 ns; los tiempos de acceso de los dispositivos NMOS variarán de 35 a 500 ns. Las mejoras en la tecnología CMOS han hecho posibles tiempos de acceso en el intervalo de 20 a 60 ns. En consecuencia, los dispositivos bipolares y NMOS se producen raras veces en ROMs más recientes (grandes).

Otro parámetro importante de sincronización es el *tiempo de habilitación de salida* (t_{OE}), que es el retraso entre la entrada $\overline{CS}$ y la salida de datos válida. Los valores

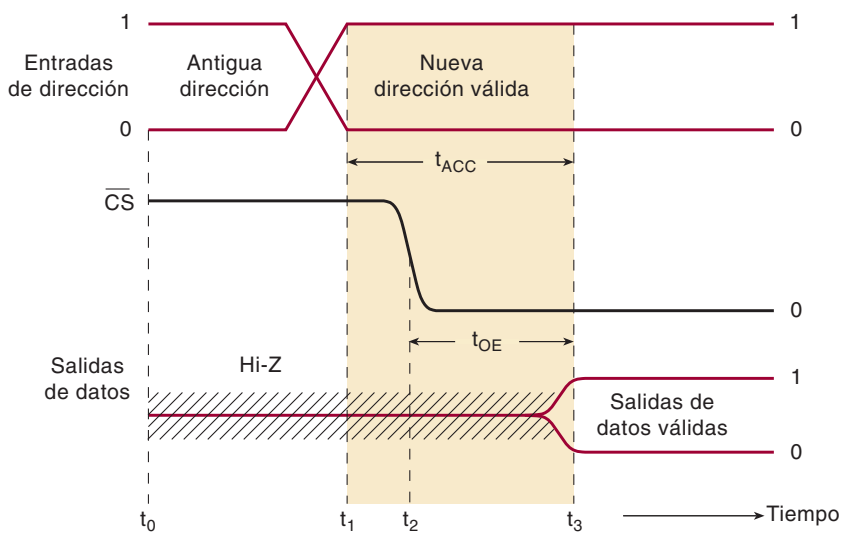


FIGURA 12-8
Sincronización típica para una operación de lectura de la ROM.

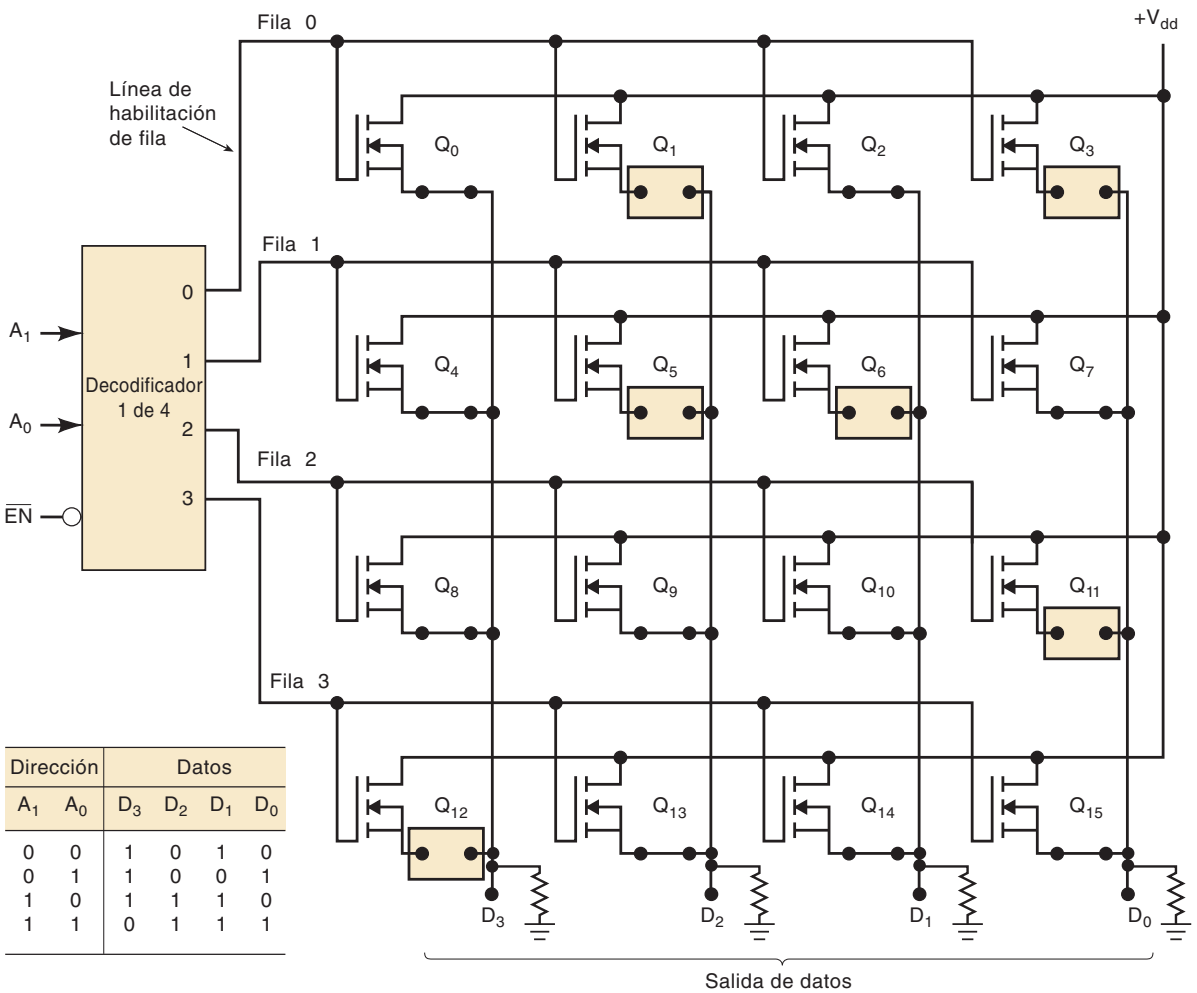
comunes para t_{OE} son de 10 a 20 ns para las ROMs bipolares, de 25 a 100 ns para las ROMs NMOS y de 12 a 50 para las ROMs CMOS. Este parámetro de sincronización es importante en situaciones en las que las entradas de dirección ya se encuentran establecidas en sus nuevos valores, pero las salidas de la ROM todavía no se han habilitado. Cuando *CS* cambie a BAJO para habilitar las salidas, el retraso será t_{OE}.

12-7 TIPOS DE ROMS

Ahora que tenemos una comprensión general sobre la arquitectura interna y la operación externa de los dispositivos de ROM, analizaremos los diversos tipos de ROMs para ver cómo difieren en la manera en que se programan, se borran y se reprograman.

ROM programada por máscara

La ROM programada por máscara (MROM) almacena su información al momento en que se fabrica el circuito integrado. Como puede ver en la figura 12-9, las ROMs están compuestas de un arreglo rectangular de transistores. La información se almacena ya sea mediante la conexión o la desconexión de la fuente de un transistor con



Dirección		Datos			
A_1	A_0	D_3	D_2	D_1	D_0
0	0	1	0	1	0
0	1	1	0	0	1
1	0	1	1	1	0
1	1	0	1	1	1

FIGURA 12-9 La estructura de una MROM tipo MOS muestra que se utiliza un MOSFET para cada celda de memoria. Una conexión de fuente abierta almacena un "0"; una conexión de fuente cerrada almacena un "1".

la columna de salida. El último paso en el proceso de fabricación es formar todas estas rutas conductoras o conexiones. El proceso utiliza una "máscara" para depositar metales en el silicio, los cuales determinan en dónde se forman las conexiones de una manera similar al uso de patrones y pintura en aerosol, pero en una escala mucho más pequeña. La máscara es muy precisa y costosa, y debe fabricarse de manera específica para el cliente, con la información binaria correcta. En consecuencia, este tipo de ROM es económica sólo cuando se van a fabricar muchas ROMs con la misma información exacta.

Por lo general, a las ROMs programadas por máscara se les conoce sólo como ROMs, lo cual puede ser confuso ya que el término ROM en realidad representa a la amplia categoría de dispositivos que, durante la operación normal, sólo se pueden leer. Utilizaremos la abreviación MROM cada vez que hagamos referencia a las ROMs programadas por máscara.

La figura 12-9 muestra la estructura de una pequeña MROM tipo MOS de 16 celdas de memoria ordenadas en cuatro filas o cuatro celdas. Cada celda es un transistor MOSFET de canal N, conectado en la configuración de colector común (entrada en la compuerta, salida en la fuente). La fila superior de celdas (FILA 0) constituye un registro de cuatro bits. Observe que algunos de los transistores en esta fila (Q_0 y Q_2) tienen su fuente conectada a la línea de la columna de salida, mientras que otros (Q_1 y Q_3) no. Lo mismo se aplica a las celdas en cada una de las otras filas. La presencia o ausencia de estas conexiones de fuente determina si una celda almacena un 1 o un 0, respectivamente. La condición de cada conexión de fuente se controla durante la producción mediante la máscara fotográfica, con base en los datos suministrados por el cliente.

Observe que las salidas de datos están conectadas a las líneas de columna. Por ejemplo, al hacer referencia a la salida D_3, cualquier transistor que tenga una conexión de la fuente (como Q_0, Q_4 y Q_8) a la columna de salida puede cambiar a V_{dd} hacia la columna, para que tenga un nivel ALTO lógico. Si V_{dd} no se conecta a la línea de columna, la salida se mantendrá en un nivel lógico BAJO mediante la resistencia descendente. En un momento determinado, se encenderá un máximo de un transistor en una columna debido al decodificador de fila.

El decodificador de 1 a 4 se utiliza para decodificar las entradas de dirección A_1A_0 para seleccionar de cuál fila (registro) se van a leer los datos. Las salidas activas en ALTO del decodificador proporcionan las líneas de habilitación de FILA que son las entradas de compuerta para las diversas filas de celdas. Si la entrada de habilitación $\overline{EN}$ del decodificador se mantiene en ALTO, todas las salidas del decodificador estarán en su estado BAJO inactivo, y todos los transistores en el arreglo estarán apagados debido a la ausencia de voltaje en compuerta. Debido a esta situación, todas las salidas de datos estarán en el estado BAJO.

Cuando $\overline{EN}$ se encuentra en su estado activo en BAJO, las condiciones en las entradas de dirección determinan qué fila (registro) se habilitará para que puedan leerse sus datos en las salidas de datos. Por ejemplo, para leer la FILA 0, las entradas A_1A_0 se establecen en 00. Esto coloca un nivel ALTO en la línea de la FILA 0; todas las demás líneas de fila están en 0 V. Este nivel ALTO en FILA 0 enciende los transistores Q_0, Q_1, Q_2 y Q_3. Cuando todos los transistores en la fila estén conduciendo, V_{dd} se cambiará a cada una de las terminales de la fuente del transistor. Las salidas D_3 y D_1 cambiarán a ALTO debido a que Q_0 y Q_2 están conectados a sus respectivas columnas. D_2 y D_0 permanecerán en BAJO ya que no hay ruta de las terminales de fuente de Q_1 y Q_3 hacia sus columnas. De manera similar, la aplicación de los otros códigos de dirección producirá salidas de datos del registro correspondiente. La tabla en la figura 12-9 muestra los datos para cada dirección. Usted deberá verificar cómo se correlaciona esto con las conexiones de fuente a las diversas celdas.

EJEMPLO 12-8

Las MROMs pueden usarse para almacenar tablas de funciones matemáticas. Demuestre cómo puede usarse la MROM de la figura 12-9 para almacenar la función $y = x^2 + 3$, en donde la dirección de entrada suministra el valor para x y el valor de los datos de salida es y.

TABLA 12-1

x		$y = x^2 + 3$			
A_1	A_0	D_3	D_2	D_1	D_0
0	0	0	0	1	1
0	1	0	1	0	0
1	0	0	1	1	1
1	1	1	1	0	0

Solución

El primer paso es utilizar una tabla que muestre la salida deseada para cada conjunto de entradas. El número binario de entrada x se representa mediante la dirección A_1A_0. El número binario de salida es el valor deseado de y. Por ejemplo, cuando $x = A_1A_0 = 10_2 = 2_{10}$, la salida deberá ser $2^2 + 3 = 7_{10} = 0111_2$. La tabla completa se muestra en la tabla 12-1. Esta tabla se proporciona al fabricante de MROMs para que desarrolle la máscara que realizará las conexiones apropiadas dentro de las celdas de memoria, durante el proceso de fabricación. Por ejemplo, la primera fila en la tabla indica que las conexiones a la fuente de Q_0 y Q_1 se dejarán desconectadas, mientras que las conexiones a Q_2 y Q_3 sí se realizarán.

Como vimos en el capítulo 9, por lo general, las MROMs tienen salidas triestado que les permiten utilizarse en un sistema de bus. En consecuencia, debe haber una entrada de control para habilitar y deshabilitar las salidas triestado. Esta entrada de control se etiqueta por lo común como *OE* (habilitación de salida). Para poder distinguir esta entrada de habilitación triestado de la entrada de habilitación del decodificador de dirección, a esta última, por lo general, se le hace referencia como *CE* (habilitación de chip). La señal de habilitación de chip realiza algo más que sólo habilitar el decodificador de dirección. Cuando se deshabilita *CE*, todas las funciones del chip están deshabilitadas, incluyendo las salidas triestado, y todo el circuito se coloca en un modo de **bajo consumo de energía**, en el cual se consume una cantidad mucho menor de corriente de la fuente de energía. La figura 12-10 muestra una MROM de 32K × 8. Las 15 líneas de dirección (A0-A14) pueden identificar 2^{15} ubicaciones de memoria (32, 767 o 32K). Cada ubicación de memoria guarda un valor de datos de ocho bits, que puede colocarse en las líneas de datos D7-D0 cuando el chip está habilitado y las salidas están habilitadas.

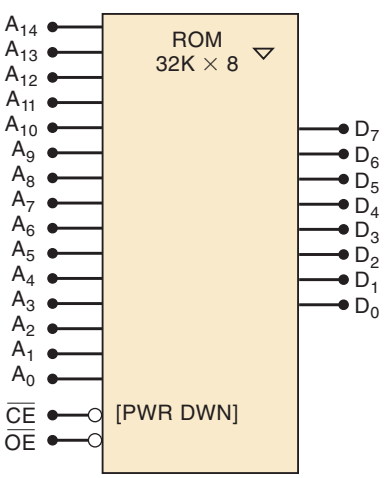


FIGURA 12-10 Símbolo lógico para una MROM de 32K × 8.

ROMs programables (PROMs)

Una ROM programable por máscara es muy costosa y se utiliza sólo en aplicaciones donde se debe fabricar una alta cantidad de memorias idénticas, y así repartir el costo entre muchas unidades. Para las aplicaciones de menor volumen, los fabricantes han desarrollado PROMs con **enlaces de fusible**, las cuales son programables por el usuario; esto es, no se programan durante el proceso de fabricación sino que el usuario las programa según sus necesidades. Sin embargo, una vez programada, una PROM es como una MROM ya que no puede borrarse y reprogramarse. Por lo tanto, si el programa en la PROM tiene fallas o debe modificarse, hay que desechar esa PROM. Por esta razón, a estos dispositivos se les conoce por lo general como ROMs "programables una sola vez" (OTP).

La estructura de una PROM con enlace de fusible es muy similar a la estructura de una MROM, ya que ciertas conexiones se dejan intactas o se abren para poder programar una celda de memoria como 1 o 0, respectivamente. Una PROM viene del fabricante con una conexión delgada de enlace de fusible en la terminal de la fuente de cada transistor. En esta condición, cada transistor almacena un 1. El usuario entonces puede "volar" el fusible de cualquier transistor que necesite almacenar un 0. Por lo común, la programación o "quemado" de datos en una PROM se realiza mediante la selección de una fila al aplicar la dirección deseada en las entradas de dirección, después se colocan los datos deseados en las terminales de datos y luego se aplica un pulso a una terminal especial de programación en el CI. La figura 12-11 muestra el funcionamiento interno de este proceso.

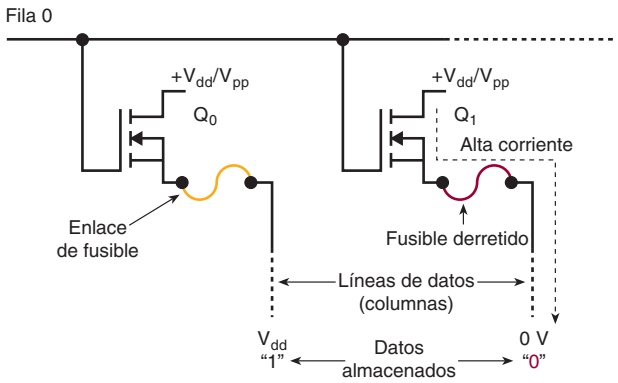


FIGURA 12-11 Las PROMs utilizan enlaces de fusible; el usuario puede abrir estos enlaces en forma selectiva para programar un 0 lógico en una celda.

Todos los transistores en la fila seleccionada (fila 0) se encienden y se aplica V_{pp} en sus terminales de colector. Las columnas (líneas de datos) que tengan un 0 lógico en ellas (por ejemplo, Q1) proveerán una ruta de alta corriente a través del enlace de fusible para quemarlo y dejarlo abierto, con lo cual se almacenará un 0 lógico en forma permanente. Las columnas que tengan un 1 lógico (por ejemplo, Q_0) tendrán V_{pp} en un lado del fusible y V_{dd} en el otro lado, con lo cual consumirán mucha menos corriente y dejarán el fusible intacto. Una vez que se hayan programado todas las ubicaciones de dirección de esta forma, los datos quedarán almacenados en forma permanente en la PROM y podrán leerse una y otra vez, mediante el acceso a la dirección apropiada. Los datos no cambiarán cuando se quite la energía del chip PROM, ya que no hay nada que pueda hacer que un enlace de fusible abierto se vuelva a cerrar.

Una PROM se programa mediante el mismo equipo y proceso que describimos en el capítulo 4 para programar un PLD. La TMS27PC256 es una PROM tipo CMOS muy popular, con una capacidad de 32K $\times$ 8 y una disipación de energía en modo suspendido de sólo 1.4 mW. Está disponible con tiempos de acceso que varían entre 100 y 250 ns.

ROM programable y borrable (EPROM)

Una EPROM puede programarse por el usuario, y también puede *borrarse* y reprogramarse casi todas las veces que se requiera. Una vez programada, la EPROM es

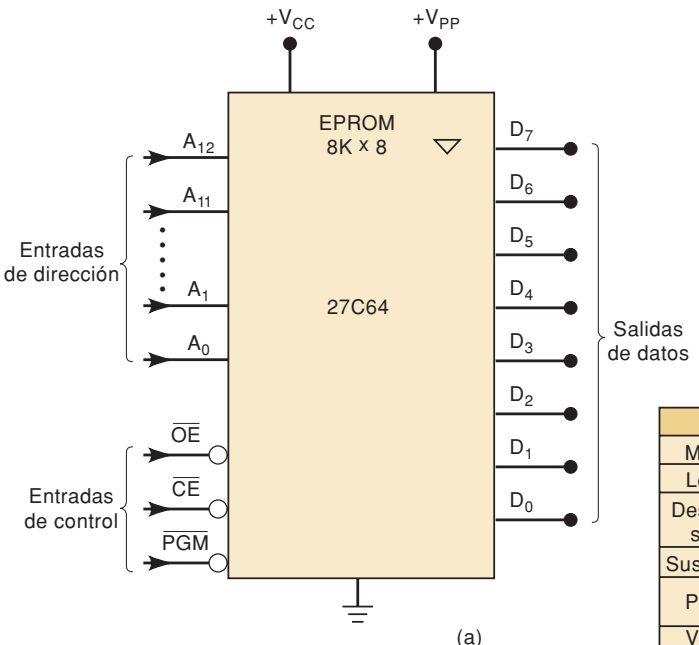
una memoria *no volátil* que almacenará sus datos en forma indefinida. El proceso para programar una EPROM es el mismo que para una PROM.

El elemento de almacenamiento de una EPROM es un transistor MOS con una compuerta de silicio que no tiene conexión eléctrica (es decir, una compuerta flotante) pero está muy cerca de un electrodo. En su estado normal no hay carga almacenada en la compuerta flotante, por lo que el transistor producirá un 1 lógico cada vez que el decodificador de dirección lo seleccione. Para programar un 0 se utiliza un pulso de alto voltaje para dejar una carga neta en la compuerta flotante. Esta carga hace que el transistor produzca como salida un 0 lógico cuando se selecciona. Como la carga está atrapada en la compuerta flotante y no tiene ruta de descarga, el 0 se almacenará hasta que se borre. Para borrar los datos se restauran todas las celdas a un 1 lógico. Para ello se neutraliza la carga en el electrodo flotante al exponer el silicio a una luz ultravioleta (UV) de alta intensidad durante varios minutos.

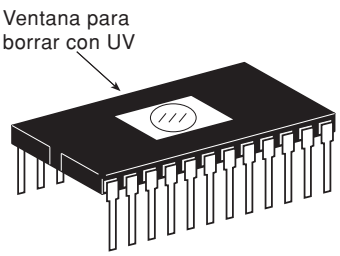
La 27C64 es un ejemplo de un CI de memoria pequeño de 8K × 8K, disponible como PROM "programable sólo una vez" (OTP) o como EPROM borrable con luz UV. La diferencia obvia en los dos CIs es la "ventana" transparente de cuarzo de la EPROM, que se muestra en la figura 12-12(b). Esta ventana permite que la luz UV se refleje sobre el silicio. Ambas versiones operan a partir de una sola fuente de energía de +5 V durante su operación normal.

La figura 12-12(a) es el símbolo lógico para la 27C64. Observe que muestra 13 entradas de dirección (debido a que $2^{13} = 8192$) y ocho salidas de datos. Tiene cuatro entradas de control. $\overline{CE}$ es la entrada de habilitación de chip que se utiliza para colocar el dispositivo en un modo de suspensión, en donde se reduce su consumo de energía. $\overline{OE}$ es la entrada de habilitación de las salidas y se utiliza para controlar los búferes triestado de salida de datos del dispositivo, de manera que éste pueda conectarse al bus de datos de un microprocesador sin contención de bus. V_{PP} es el voltaje de programación especial requerido durante el proceso de programación. $\overline{PGM}$ es la entrada de habilitación de programa que se activa para almacenar datos en la dirección seleccionada.

La 27C64 tiene varios modos de operación que se controlan mediante las terminales $\overline{CE}$, $\overline{OE}$, V_{PP} y PGM, como se presentan en la figura 12-12(c). El modo pro-



(a)



(b)

	Entradas				Salidas
Modo	$\overline{CE}$	$\overline{OE}$	$\overline{PGM}$	VPP	$D_7 - D_0$
Lectura	0	0	1	0–5V	$DATOS_{sal}$
Deshabilita salidas	0	1	1	0–5V	Alta impedancia
Suspensión	1	X	X	X	Alta impedancia
Programa	0	1	0	12.75 V	$DATOS_{ent}$
Verifica PGM	0	0	1	12.75 V	$DATOS_{sal}$

(c)

FIGURA 12-12 (a) Símbolo lógico para la EPROM 27C64; (b) encapsulado típico de una EPROM en el que se muestra una ventana ultravioleta; (c) modos de operación del 27C64.

grama se utiliza para escribir nuevos datos en las celdas de la EPROM. Esto se hace con más frecuencia en una EPROM "limpia", una que haya sido borrada antes con luz UV para que todas las celdas sean 1. El proceso de programación escribe una palabra de ocho bits en una ubicación de memoria en un momento dado, como se indica a continuación: (1) la dirección se aplica a las terminales de dirección; (2) los datos deseados se colocan en las terminales de datos, que funcionan como entradas durante el proceso de programación; (3) se aplica un voltaje de programación más alto de 12.75 V en V_{PP}; (4) $\overline{CE}$ se mantiene en BAJO; (5) se aplica un pulso a $\overline{PGM}$ para que cambie a BAJO durante 100 µs y los datos se leen de vuelta. Si los datos no se almacenaron con éxito, se aplica otro pulso a $\overline{PGM}$. Esto se repite en la misma dirección hasta que los datos se almacenan con éxito.

Una EPROM limpia puede programarse en menos de un minuto, una vez que se hayan introducido, transferido o descargado los datos deseados en el programador de EPROMs. La 27C512 es una EPROM común de 64K × 8 que opera en forma muy parecida a la 27C64, pero ofrece más capacidad de almacenamiento.

Las principales desventajas de las UV-EPROMs son que deben removerse del circuito para programarse y borrarse, la operación borra todo el chip y puede durar hasta 20 minutos.

PROM programable y borrable eléctricamente (EEPROM)

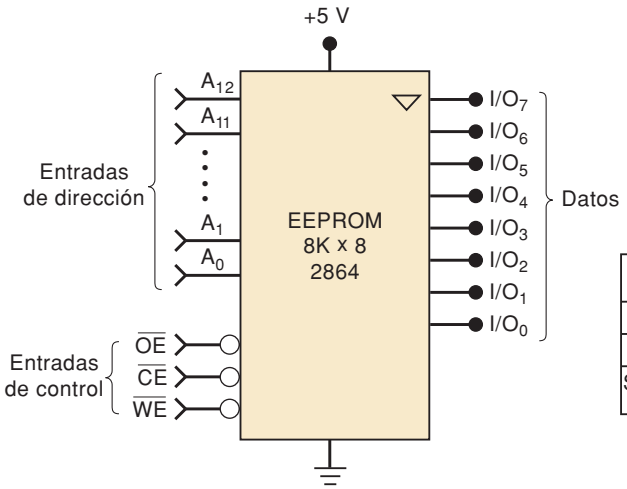
Las desventajas de la EPROM se solucionaron con el desarrollo de la **PROM programable y borrable eléctricamente (EEPROM)** como una mejora sobre la EPROM. La EEPROM retiene la misma estructura de compuerta flotante que la EPROM, pero agrega una región de óxido muy delgada por encima del colector de la celda de memoria de MOSFET. Esta modificación produce la característica principal de la EEPROM: su capacidad de borrarse mediante electricidad. Al aplicar un alto voltaje (21 V) entre la compuerta y el colector del MOSFET, se puede inducir una carga en la compuerta flotante, en donde permanecerá aunque se corte la energía; si se invierte el mismo voltaje se eliminarán las cargas atrapadas de la compuerta flotante y se borrará la celda. Como este mecanismo de transporte de cargas requiere corrientes muy bajas, el borrado y la programación de una EEPROM pueden realizarse *en el circuito* (es decir, sin una fuente de luz UV ni dispositivo especial para programar PROMs).

Otra ventaja de la EEPROM en comparación con la EPROM es la habilidad de borrar y reescribir bytes *individuales* (palabras de ocho bits) en el arreglo de memoria mediante electricidad. Durante una operación de escritura, el circuito interno borra de manera automática todas las celdas en una ubicación de dirección antes de escribir los nuevos datos. Esta capacidad de borrar bytes facilita en forma considerable la realización de modificaciones en los datos que se almacenan en una EEPROM.

Las primeras EEPROMs, como la 2816 de Intel, requerían circuitos externos de soporte apropiados para los chips de memoria. Estos circuitos de soporte incluyen el voltaje de programación de 21 V (V_{PP}), que por lo común se genera a partir de una fuente de +5 V a través de un convertidor de corriente directa a corriente directa, e incluye circuitos para controlar la sincronización y la secuencia de las operaciones de borrar y programar. Los dispositivos más recientes, como el 2864 de Intel, integran estos circuitos de soporte en el mismo chip junto con el arreglo de memoria, por lo que sólo se requiere una sola terminal de energía de +5 V. Esto hace que la EEPROM sea tan fácil de usar como la memoria de lectura/escritura que veremos en breve.

La capacidad de borrar bytes de la EEPROM y su alto nivel de integración incurren en dos faltas: densidad y costo. La complejidad de la celda de memoria y los circuitos de soporte en el chip ponen a las EEPROM en desventaja considerable con las EPROM, en cuanto a la capacidad de bits por milímetro cuadrado de silicio; una EEPROM de 1 Mbit requiere casi el doble de silicio que una EPROM de 1 Mbit. Por lo tanto, y a pesar de su superioridad operacional, las desventajas de la EEPROM en cuanto a densidad y efectividad en costo han evitado que sustituya a la EPROM en aplicaciones en donde la densidad y el costo son factores imperativos.

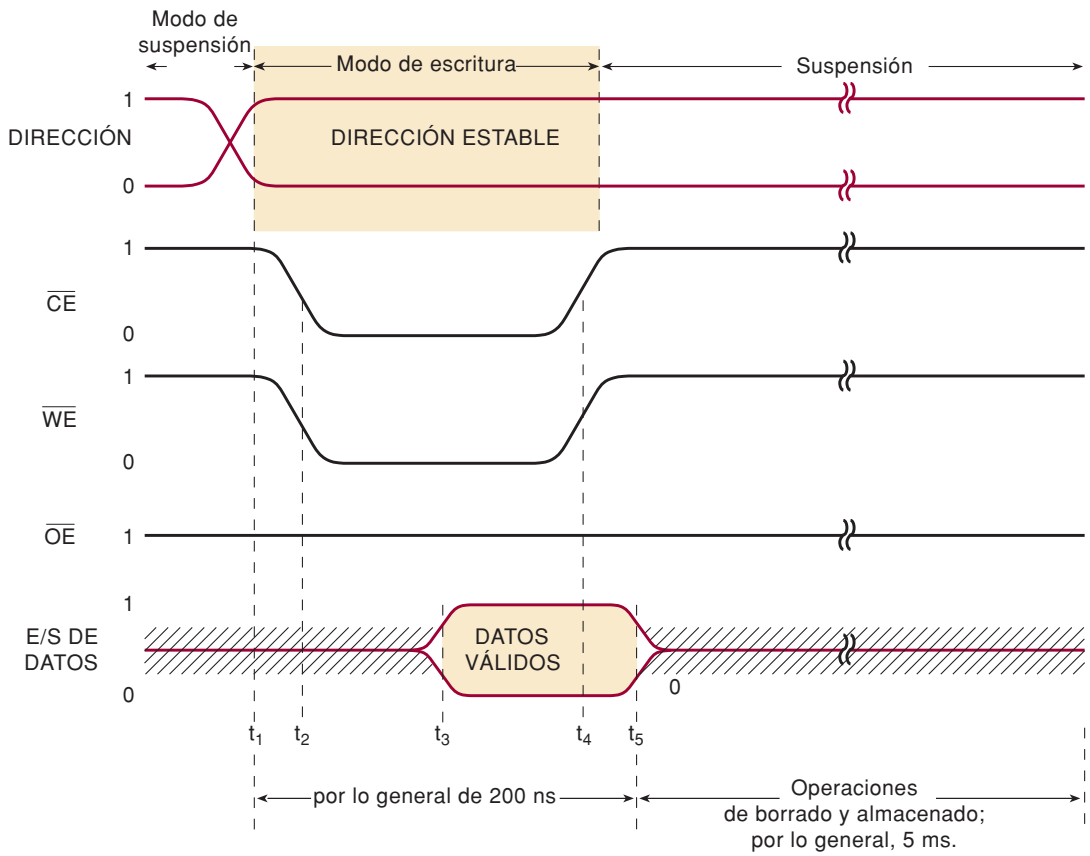
El símbolo lógico para el Intel 2864 se muestra en la figura 12-13(a). Se organiza como un arreglo de 8K × 8, con 13 direcciones de entrada ($2^{13} = 8192$) y ocho terminales de E/S de datos. Tres entradas de control determinan el modo de operación, de acuerdo con la tabla que se proporciona en la figura 12-13(b). Cuando $\overline{CE}$ = ALTO, el chip está en su modo de suspensión de bajo consumo de energía, en el cual no se



(a)

	Entradas			
Modo	$\overline{CE}$	$\overline{OE}$	$\overline{WE}$	Terminales de E/S
LECTURA	BAJO	BAJO	ALTO	$DATOS_{SAL}$
ESCRITURA	BAJO	ALTO	BAJO	$DATOS_{ENT}$
SUSPENSIÓN	ALTO	X	X	Alta impedancia

(b)



(c)

FIGURA 12-13 (a) Símbolo para la EEPROM 2864; (b) modos de operación; (c) sincronización para la operación de escritura.

realizan operaciones en ninguna ubicación de memoria y las terminales de datos se encuentran en el estado Hi-Z.

Para leer el contenido de una ubicación de memoria determinada, se aplica la dirección deseada en las terminales de dirección; $\overline{CE}$ se lleva a BAJO; y la terminal de habilitación de salida $\overline{OE}$ se lleva a BAJO para habilitar los búferes de datos de salida del chip. La terminal de habilitación de salida $\overline{WE}$ se mantiene en ALTO durante una operación de lectura.

Para escribir en (es decir, programar) una ubicación de memoria se deshabilitan los búferes de salida, de manera que los datos que se van a escribir puedan aplicarse como entradas para las terminales de E/S. La figura 12-13(c) muestra un diagrama de la sincronización para la operación de escritura. Antes de t_1, las entradas $\overline{CE}$ y $\overline{WE}$ se llevan a BAJO para comenzar la operación de escritura; $\overline{OE}$ está en ALTO, por lo que las terminales de datos permanecerán en el estado Hi-Z. En t_3 se aplican los datos en las terminales de E/S y en t_4 se escriben en la ubicación de dirección en el flanco positivo de $\overline{WE}$. En t_5 se extraen los datos. En realidad, los datos se fijan primero (en el flanco positivo de $\overline{WE}$) en una memoria de búfer de FF que forma parte de los circuitos del 2864. Los datos se retienen ahí mientras otro circuito en el chip realiza una operación de borrado en la ubicación de dirección seleccionada en el arreglo de la EEPROM, después de lo cual el byte de datos se transfiere del búfer al arreglo de la EEPROM y se almacena en esa ubicación. Esta operación de borrado y almacenado dura aproximadamente 5 ms. Cuando $\overline{CE}$ regresa al nivel ALTO en t_4, el chip está nuevamente en el modo de espera mientras se completan las operaciones internas de borrado y almacenado.

La 2864 tiene un modo de escritura mejorado, en el cual se permite al usuario escribir hasta 16 bytes de datos en la memoria del búfer de FF, en donde se retienen mientras el circuito de la EEPROM borra las ubicaciones de dirección seleccionadas. Después, los 16 bytes de datos se transfieren al arreglo de la EEPROM para almacenarlos en esas ubicaciones. Este proceso dura también alrededor de 5 ms.

Debido a que el proceso interno de almacenar un valor de datos en una EEPROM es bastante lento, la velocidad de la operación de transferencia de datos también puede ser más lenta. En consecuencia, muchos fabricantes ofrecen dispositivos EEPROM en encapsulados de ocho terminales que se integran a un bus *serial* de dos o tres alambres. Esto ahorra espacio físico en el tablero del sistema, lo contrario a utilizar una 2864 en un encapsulado DIP amplio de 28 terminales. También simplifica la interfase de hardware entre la CPU y la EEPROM.

CD-ROM

El disco compacto (CD) es un tipo muy prominente de almacenamiento de sólo lectura que se utiliza en la actualidad en los sistemas computacionales. La tecnología de disco y el hardware necesario para recuperar la información son los mismos que se utilizan en los sistemas de audio. Sólo el formato de los datos es distinto. Los discos se fabrican con una superficie altamente reflectiva. Para almacenar datos en los discos, un rayo láser muy intenso se enfoca en un punto *muy* pequeño del disco. Este rayo quema un pozo difractor de luz en ese punto de la superficie del disco. Los datos digitales (1s y 0s) se almacenan en el disco un bit a la vez, mediante el proceso de quemar o no quemar un pozo en el recubrimiento reflectivo. La información digital se ordena en el disco como una espiral continua de puntos de datos. La precisión del rayo láser permite almacenar muy grandes cantidades de datos (más de 550 Mbytes) en un pequeño disco de 120 mm.

Para poder leer los datos se enfoca un rayo láser mucho menos poderoso, que el utilizado para almacenar, en la superficie del disco. En cualquier punto, la luz reflejada se detecta ya sea como un 1 o como un 0. Este sistema óptico está montado en un transporte mecánico que se mueve hacia delante y hacia atrás a lo largo del radio del disco, siguiendo la espiral de datos a medida que el disco gira. Los datos que se recuperan del sistema óptico llegan un bit a la vez en un flujo de datos en serie. La rotación angular del disco se controla para mantener una velocidad constante de puntos de datos entrantes. Si el disco se va a utilizar para grabación de audio, este flujo de datos se convierte en una forma de onda analógica. Si el disco se va a utilizar como ROM, los datos se decodifican en bytes en paralelo que la computadora

pueda utilizar. La tecnología del reproductor de CD, aunque muy sofisticada, tiene un bajo costo relativo y se está convirtiendo en la manera estándar de cargar grandes cantidades de datos en una computadora personal. Las principales mejoras que están ocurriendo ahora en la tecnología del CD-ROM incluyen un tiempo de acceso más rápido al recuperar los datos.

PREGUNTAS DE REPASO

1. *Verdadero o falso:* una MROM puede programarse por el usuario.
2. ¿Qué diferencia tiene una PROM en comparación con una MROM? ¿Puede borrarse y reprogramarse?
3. *Verdadero o falso:* una PROM almacena un 1 lógico cuando su enlace de fusible está intacto.
4. ¿Cómo se borra una EPROM?
5. *Verdadero o falso:* no hay forma de borrar sólo una porción de una memoria EPROM.
6. ¿Qué función realizan los programadores de PROM y de EPROM?
7. ¿Qué desventajas de las EPROMs solucionan las EEPROMs?
8. ¿Cuáles son las principales desventajas de la EEPROM?
9. ¿Qué tipo de ROM puede borrar un byte a la vez?
10. ¿Cuántos bits se leen de un disco CD-ROM en cualquier punto en el tiempo?

12-8 MEMORIA FLASH

Las EPROMs son no volátiles, ofrecen tiempos de acceso de lectura rápidos (por lo general de 120 ns) y tienen una alta densidad y un bajo costo por bit. No obstante, hay que extraerlas de su circuito/sistema para borrarlas y reprogramarlas. Las EEPROMs son no volátiles, ofrecen un acceso de lectura rápido y permiten borrar y reprogramar bytes individuales con rapidez y dentro del circuito. Sufren de una menor densidad y un costo mucho más alto que las EPROMs.

El reto para los ingenieros de semiconductores era fabricar una memoria no volátil con la capacidad de la EEPROM de borrarse mediante electricidad y dentro del circuito, pero con densidades y costos mucho más cercanos a los de las EPROMs, al tiempo que se mantuviera el acceso de lectura de alta velocidad de ambos tipos de memorias. La respuesta a este reto fue la **memoria Flash**.

En su estructura, la celda de una memoria Flash es parecida a la celda simple de un solo transistor de la EPROM (y distinta de la celda más compleja de dos transistores de la EEPROM), sólo que es un poco más grande. Tiene una capa de óxido más delgada en la compuerta, la cual le permite la capacidad de borrarse mediante electricidad, pero puede construirse con densidades mucho más altas que las EEPROMs. El costo de la memoria Flash es mucho menor que el de la EEPROM. La figura 12-14 ilustra las ventajas y desventajas de las diversas memorias semiconductoras no volátiles. A medida que aumenta la flexibilidad en el borrado/la programación (de la base al vértice del triángulo), también aumentan la complejidad y el costo. Las memorias MROM y PROM son los dispositivos más económicos y simples, pero no pueden borrarse y reprogramarse. La EEPROM es la más compleja y costosa debido a que puede borrarse y reprogramarse en un circuito, byte por byte.

Las memorias Flash se llaman así debido a sus tiempos rápidos de borrado y escritura. La mayoría de los chips Flash utilizan una operación de *borrado en masa*, en el cual todas las celdas en el chip se borran en forma simultánea; este proceso de borrado en masa requiere, por lo general, de cientos de milisegundos, en comparación con los 20 minutos para las EPROMs de UV. Algunas memorias Flash más recientes ofrecen un modo de *borrado por sector*, en donde pueden borrarse sectores específicos del arreglo de memoria (por ejemplo, 512 bytes) uno a la vez. Esto evita tener que borrar y reprogramar todas las celdas cuando sólo se necesita actualizar

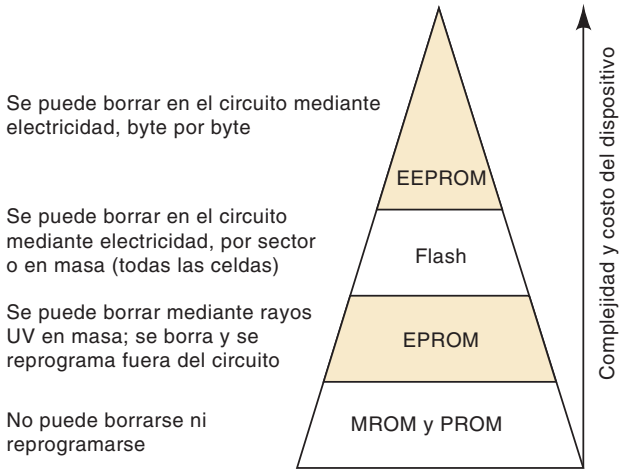
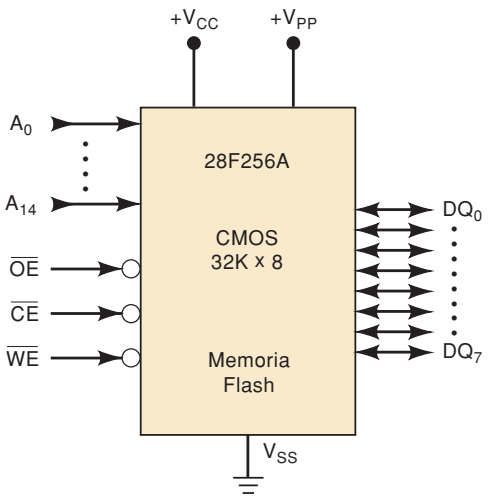


FIGURA 12-14 Las ventajas y desventajas de las memorias semiconductoras no volátiles muestran que la complejidad y el costo aumentan a medida que se incrementa la flexibilidad en el borrado y la programación.

una parte de la memoria. Una memoria Flash común tiene un tiempo de escritura de 10 μs por byte, en comparación con los 100 μs para la EPROM más avanzada y 5 ms para la EEPROM (que incluye el tiempo de borrado de bytes automático).

El CI de memoria Flash CMOS 28F256A

La figura 12-15(a) muestra el símbolo lógico para el circuito de memoria Flash CMOS 28F256A de Intel Corporation, el cual tiene una capacidad de 32K × 8. El diagrama muestra 15 entradas de dirección ($A_0 - A_{14}$) necesarias para seleccionar las distintas direcciones de memoria; es decir, $2^{15} = 32\text{K} = 32{,}768$. Las ocho terminales de entrada/salida de datos ($DQ_0 - DQ_7$) se utilizan como entradas durante las operaciones de escritura de memoria y como salidas durante las operaciones de lectura de memoria. Estas terminales de datos flotan en el estado Hi-Z cuando el chip se deselecciona ($\overline{CE}$ = ALTO) o cuando las salidas están deshabilitadas ($\overline{OE}$ = ALTO). La entrada de habilitación de escritura ($\overline{WE}$) se utiliza para controlar las operaciones de escritura en memoria. Observe que el chip requiere dos voltajes de fuente de energía: V_{CC} es el voltaje estándar de +5 V que se utiliza para los circuitos lógicos; V_{PP} es el voltaje de la fuente de energía para borrar/programar, cuyo valor



(a)

	Entradas			
Modo	$\overline{CE}$	$\overline{OE}$	$\overline{WE}$	Terminales de datos
LECTURA	BAJO	BAJO	ALTO	$DATOS_{SAL}$
SUSPENSIÓN	ALTO	X	X	Alta impedancia
ESCRITURA*	BAJO	ALTO	BAJO	$DATOS_{ENT}$

*Nota: si VPP ≤ 6.5 V, no puede realizarse una operación de escritura

(b)

FIGURA 12-15 (a) Símbolo lógico para el chip de memoria Flash 28F256A; (b) las entradas de control $\overline{CE}$, $\overline{WE}$ y $\overline{OE}$.

nominal es de +12V, el cual se necesita para las operaciones de borrado y programación (escritura). Los chips Flash más nuevos generan el V_{PP} de manera interna y sólo requieren una fuente de energía. Los dispositivos de bajo voltaje más recientes operan sólo con 1.8 V.

Las entradas de control ($\overline{CE}$, $\overline{OE}$ y $\overline{WE}$) controlan lo que ocurre en las terminales de datos, en forma muy parecida a la de la EEPROM 2864, como muestra la tabla de la figura 12-15(b). Por lo general estas terminales de datos se conectan a un bus de datos. Durante una operación de escritura, los datos se transfieren a través del bus (por lo general desde el microprocesador) y hacia el chip. Durante una operación de lectura, los datos en el interior del chip se transfieren a través del bus de datos (por lo general hacia el microprocesador).

Podremos comprender mejor la operación de este chip de memoria Flash si analizamos su estructura interna. La figura 12-16 es un diagrama del 28F256A en el que se muestran sus principales bloques funcionales. Deberá remitirse a este diagrama según lo necesite durante el siguiente análisis. La característica única de esta estructura es el *registro de instrucciones*, el cual se utiliza para administrar todas las funciones del chip. Los códigos de instrucción se escriben en este registro para controlar qué operaciones se deben llevar a cabo dentro del chip (por ejemplo, borrar, borrar-verificar, programar, programar-verificar). Por lo general, estos códigos de instrucción llegan a través del bus de datos desde el microprocesador. La lógica de control de estado examina el contenido del registro de instrucciones y genera señales de lógica y control para que el resto de los circuitos del chip lleve a cabo los pasos en la operación. Algunos ejemplos de los tipos de instrucciones que pueden enviarse a la memoria Flash se muestran aquí para que se dé una idea de por qué son necesarios. Cada instrucción se almacena en el registro mediante el uso del mismo ciclo de escritura que se describió para la EEPROM en la figura 12-13(c).

Instrucción leer. Al escribir un código de 00 hexadecimal en el registro de instrucciones se prepara el CI de memoria para la operación de lectura. Después de esto puede usarse un ciclo normal de lectura para acceder a los datos almacenados en cualquier dirección.

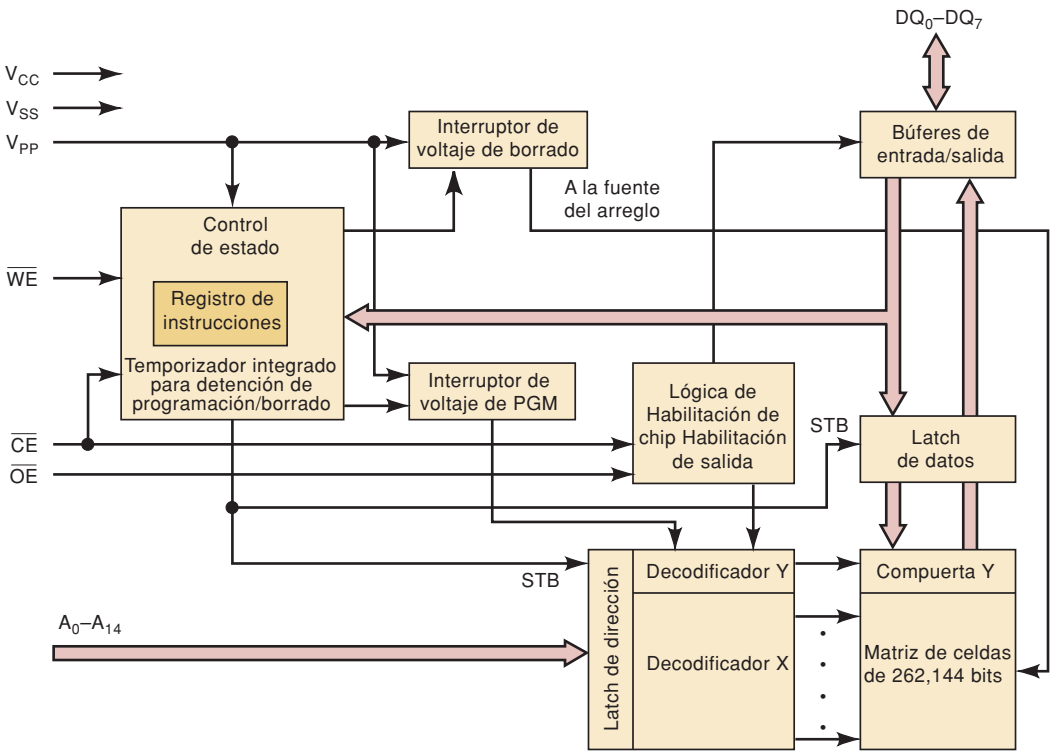


FIGURA 12-16 Diagrama funcional del chip de memoria Flash 28F256A. (Cortesía de Intel Corporation.)

Instrucción establecer-borrar/borrar. Debe escribirse el código de 20 hexadecimal en el registro de instrucciones dos veces en una fila para comenzar la secuencia interna de borrado.

Instrucción verificar borrado. Esta instrucción (FF hexadecimal) hace que el CI de memoria compruebe todas sus ubicaciones de memoria para verificar que todos los bits estén en ALTO.

Instrucción establecer programar/programar. Esta instrucción (40 hexadecimal) coloca el CI de memoria en un modo que permite que los ciclos de escritura subsiguientes almacenen datos en una dirección especificada, un byte a la vez.

Instrucción programar-verificar. Esta instrucción (C0 hexadecimal) se utiliza para verificar que se hayan almacenado los datos correctos en la ROM Flash. Después de escribir este código en el registro de instrucciones, la siguiente operación de lectura producirá el contenido de la última ubicación en la que se escribió, y estos datos pueden compararse con el valor deseado.

Memoria Flash mejorada

La arquitectura básica de la memoria Flash en la actualidad y el conjunto básico de códigos de instrucción son muy similares a los de los dispositivos de primera generación. Los dispositivos Flash más recientes tienen nuevas características y nuevos códigos de instrucción para controlarlas, además de las comunes para los dispositivos anteriores. Desde luego que los dispositivos Flash más recientes tienen mucha más capacidad, operan con mucho menos energía (y a menores voltajes), vienen en encapsulados más pequeños y tienen un costo mucho menor por bit que sus antecesores. También ofrecen características tales como la habilidad de leer/escribir datos mientras que se borra un bloque de memoria. El voltaje de programación V_{PP} se genera en forma interna, lo cual permite el uso de una sola fuente de energía. La velocidad de su operación puede mejorarse mediante el uso de un modo ráfaga. Esto tan sólo significa que se puede acceder a varias direcciones en una fila con mucha rapidez, con lo cual se proporciona una ráfaga de transferencia de datos. Se proporciona una entrada de reloj síncrona para controlar la operación de ráfaga. Una dirección base se fija en la memoria y después el contenido de esta ubicación se transfiere durante el flanco del reloj; también se incrementa la dirección a la siguiente ubicación. De esta forma se puede acceder a varias ubicaciones de memoria secuenciales a una velocidad tan rápida como la velocidad con la que puede oscilar el reloj del sistema, sin la sobrecarga de generar cada dirección. Todas estas características han hecho de la memoria Flash la tecnología predominante de memoria no volátil de estado sólido en uso hoy en día.

PREGUNTAS DE REPASO

1. ¿Cuál es la principal ventaja de la memoria Flash en comparación con las EPROMs?
2. ¿Cuál es la principal ventaja de la memoria Flash en comparación con las EEPROMs?
3. ¿De dónde proviene la palabra *flash*?
4. ¿Para qué se necesita el voltaje V_{PP}?
5. ¿Cuál es la función del registro de instrucciones del 28F256A?
6. ¿Cuál es el propósito de la instrucción borrar-verificar?
7. ¿Cuál es el propósito de la instrucción programar-verificar?

12-9 APLICACIONES DE LA ROM

Con la excepción de MROM y PROM, la mayoría de los dispositivos ROM pueden reprogramarse, por lo que técnicamente no son memorias de *sólo lectura*. Sin embargo,

todavía puede usarse el término *ROM* para incluir EPROMs, EEPROMs y memoria Flash debido a que, durante la operación normal, el contenido almacenado de estos dispositivos no cambia con tanta frecuencia como se lee. Por lo tanto, se considera que las ROMs incluyen a todos los dispositivos de memoria semiconductora no volátil, y que se utilizan en aplicaciones en donde se requiere el almacenamiento no volátil de información, datos o códigos de programa, y en donde los datos almacenados cambian pocas veces o incluso nunca. He aquí algunas de las áreas de aplicación más comunes.

Memoria de programa de microcontrolador embebido

Los microcontroladores prevalecen en la mayoría de los productos electrónicos para el consumidor que hay en el mercado en la actualidad. El sistema de frenos automáticos de su automóvil y el controlador del motor, su teléfono celular, su cámara digital, su horno de microondas y muchos otros productos tienen un microcontrolador como cerebro. Estas pequeñas computadoras tienen sus instrucciones de programa almacenadas en memoria no volátil (en otras palabras, en una ROM). La mayoría de los microcontroladores embebidos en la actualidad tienen una ROM Flash integrada en el mismo CI que la CPU. Muchos también tienen un área de EEPROM que ofrece las características de borrado de bytes y el almacenamiento no volátil.

Transferencia de datos y portabilidad

La necesidad de almacenar y transferir grandes conjuntos de información binaria es un requerimiento de muchos sistemas de bajo consumo operados por baterías hoy en día. Los teléfonos celulares almacenan fotografías y clips de video. Las cámaras digitales almacenan muchas imágenes en medios de memoria removibles. Las unidades Flash se conectan en el puerto USB de una computadora y almacenan gigabytes de información. Su reproductor de MP3 está cargado con música y opera todo el día con baterías. Un PDA (asistente digital personal) almacena información sobre citas, e-mail, direcciones e incluso hasta libros completos. Todos estos aparatos electrónicos personales comunes requieren el almacenamiento de baja energía, bajo costo y alta densidad con una capacidad de escritura en el circuito que está disponible en la memoria Flash.

Memoria de arranque

Muchas microcomputadoras y la mayoría de las computadoras más grandes no tienen sus programas de sistema operativo almacenados en ROM, sino que estos programas se almacenan en la memoria en masa externa, por lo general en disco magnético. ¿Cómo entonces saben estas computadoras lo que deben hacer cuando se encienden? Un programa relativamente pequeño, conocido como **programa de arranque**, se almacena en ROM. Cuando la computadora se enciende, ejecuta las instrucciones que están en este programa de arranque. Por lo general, estas instrucciones provocan que la CPU inicialice el hardware del sistema. Después este programa de arranque carga los programas del sistema operativo desde el dispositivo de almacenamiento masivo (disco) hacia su memoria principal interna. En ese punto la computadora comienza a ejecutar el programa del sistema operativo y está lista para responder a las instrucciones del usuario. A este proceso inicial se le conoce como “arrancar el sistema”.

Muchos de los chips de procesamiento de señales digitales cargan su memoria de programa interna mediante una ROM de arranque externa cuando se les aplica energía. Algunos de los PLDs más avanzados también cargan la información de programación que configura sus circuitos lógicos desde una ROM externa, hacia un área de RAM dentro del PLD. Así, para reprogramar el PLD se cambia la ROM de arranque en vez de cambiar el propio chip PLD.

Tablas de datos

A menudo las ROMs se utilizan para almacenar tablas de datos que no cambian. Algunos ejemplos son las tablas trigonométricas (por ejemplo: seno, coseno, etcéte-

ra) y las tablas de conversión de código. El sistema digital puede utilizar estas tablas de datos para "buscar" el valor correcto. Por ejemplo, una ROM puede usarse para almacenar la función seno para ángulos desde 0° hasta 90°. Podría organizarse como de 128 × 8 con siete entradas de dirección y ocho salidas de datos. Las entradas de dirección representan el ángulo en incrementos de 0.7° aproximadamente. Por ejemplo, la dirección 0000000 es 0°, la dirección 0000001 es 0.7°, la dirección 0000010 es 1.41°, y así en lo sucesivo hasta la dirección 1111111, que es 89.3°. Cuando se aplica una dirección a la ROM, las salidas de datos representan el seno aproximado del ángulo. Por ejemplo, con la dirección de entrada 1000000 (que representa un valor aproximado a 45°) las salidas de datos serán 10110101. Como el seno es menor o igual que 1, estos datos se interpretan como una fracción, es decir, 0.10110101, que cuando se convierte en decimal es igual a 0.707 (el seno de 45°). Es imprescindible que el usuario de esta ROM comprenda el formato en el que se almacenan los datos.

Las ROMs estándar de tabla de búsqueda para funciones tales como estas estuvieron alguna vez disponibles como chips TTL. Sólo hay unas cuantas todavía en producción. Hoy en día, la mayoría de los sistemas que necesitan buscar valores equivalentes involucran a un microprocesador y los datos de la tabla de "búsqueda" se almacenan en la misma ROM que guarda las instrucciones del programa.

Convertidor de datos

El circuito convertidor de datos recibe datos expresados en un tipo de código y produce una salida expresada en otro tipo. Por ejemplo, la conversión de código se necesita cuando una computadora produce como salida datos en código binario directo y queremos convertirlo en BCD para poder visualizarlo en pantallas de LEDs de 7 segmentos.

Uno de los métodos más sencillos de conversión de código utiliza una ROM programada de tal forma que la aplicación de una dirección específica (el código anterior) produzca una salida de datos que represente el equivalente en el nuevo código. La 74185 es una ROM que almacena la conversión de código binario a BCD para una entrada binaria de seis bits. Para ilustrar esto, una entrada de dirección binaria de 100110 (38 decimal) producirá una salida de datos de 00111000, que es el código BCD para el 38 decimal.

Generador de funciones

El generador de funciones es un circuito que produce formas de onda tales como senoidales, dientes de sierra, triangulares y cuadradas. La figura 12-17 muestra cómo se utilizan una tabla de búsqueda de ROM y un DAC para generar una señal de salida de onda senoidal.

La ROM almacena 256 valores distintos de ocho bits, cada uno de los cuales corresponde a un valor distinto de la forma de onda (es decir, un punto de voltaje distinto en la onda senoidal). Al contador de ocho bits se le aplica un pulso continuo mediante una señal de reloj, para proporcionar entradas de dirección secuenciales a la ROM. A medida que el contador avanza en ciclo a través de las 256 direcciones distintas, la ROM produce como salida los 256 puntos de datos y los envía al DAC. La salida del DAC será una forma de onda que pase a través de los 256 valores de voltaje analógicos distintos, correspondientes a los puntos de datos. El filtro pasa-bajas suaviza los intervalos en la salida del DAC para producir una forma de onda uniforme.

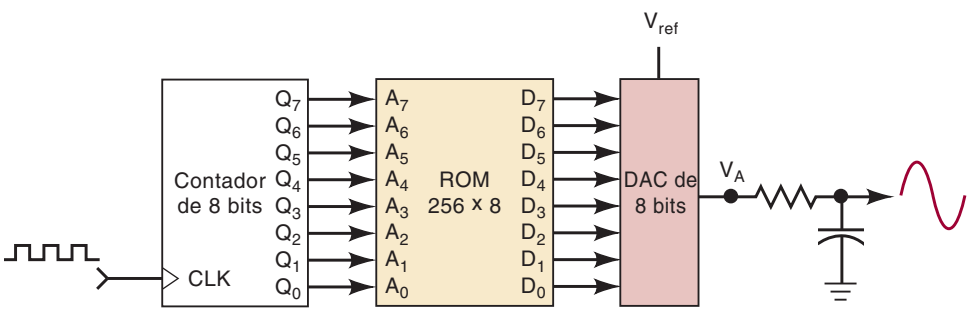


FIGURA 12-17
Generador de funciones que utiliza una ROM y un DAC.

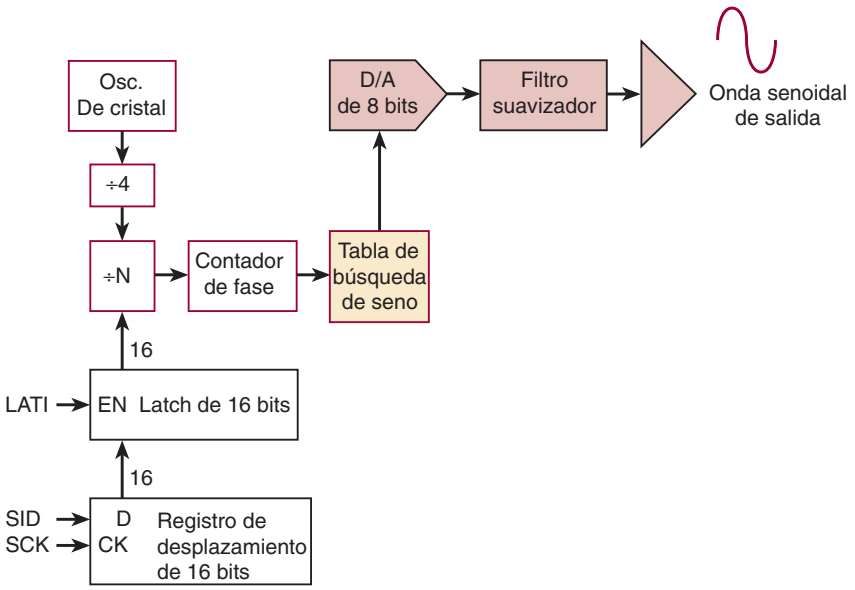


FIGURA 12-18 El generador de ondas senoidales programable ML2035. (Cortesía de MicroLinear Corp.)

Los circuitos como éste se utilizan en ciertos generadores de funciones comerciales. La misma idea se aplica en algunos sintetizadores de voz, en donde los valores de la forma de onda de voz digitalizada se almacenan en la ROM. El ML2035, que se muestra en la figura 12-18, es un chip generador de ondas senoidales programable, el cual incorpora esta estrategia básica para generar una onda senoidal de amplitud fija y una frecuencia que puede seleccionarse desde corriente directa hasta 50 kHz. El número que se desplaza en el registro de desplazamiento de 16 bits se utiliza para determinar la frecuencia de reloj para el contador que controla las entradas de dirección en la tabla de búsqueda de ROM. El ML2035 está diseñado para aplicaciones de telecomunicaciones que requieren la generación de tonos precisos de varias frecuencias.

Almacenamiento auxiliar

Debido a su no volatilidad, alta velocidad, bajos requerimientos de energía y ausencia de piezas móviles, los módulos de memoria Flash se han convertido en alternativas factibles para el almacenamiento en disco magnético. Esto se aplica en especial para las capacidades bajas (5 Mbytes o menos), en donde la memoria Flash puede competir en costo contra el disco magnético. El bajo consumo de energía de la memoria Flash la hace en especial atractiva para las computadoras portátiles que utilizan energía de una batería.

PREGUNTAS DE REPASO

1. Describa cómo utiliza una computadora un programa de arranque.
2. ¿Qué es un convertidor de código?
3. ¿Cuáles son los elementos principales de un generador de funciones?
4. ¿Por qué los módulos de memoria Flash son una alternativa factible para el almacenamiento auxiliar en disco?

12-10 RAM SEMICONDUCTORA

Recuerde que el término *RAM* significa *memoria de acceso aleatorio*, lo cual indica que se puede acceder con la misma facilidad a cualquier ubicación de dirección de memoria. Muchos tipos de memoria pueden clasificarse como de acceso aleatorio, pero

cuando se utiliza el término *RAM* con las memorias semiconductoras, por lo general, se considera como que se refiere a la memoria de lectura/escritura (RWM), a diferencia de la ROM. Como es una práctica común el usar RAM para identificar a la RWM semiconductora, mantendremos esta práctica a lo largo de las siguientes discusiones.

La RAM se utiliza en las computadoras para el almacenamiento *temporal* de programas y datos. Cuando la computadora ejecute un programa, se realizarán operaciones de lectura y de escritura sobre muchas ubicaciones de dirección de la RAM. Para ello se requieren tiempos de ciclo de lectura y de escritura rápidos para la RAM, de manera que no disminuya la velocidad de operación de la computadora.

La principal desventaja de la RAM es que es volátil y perderá toda la información almacenada si se interrumpe o se desconecta la energía. Sin embargo, algunas RAMs tipo CMOS utilizan cantidades tan pequeñas de energía en el modo de suspensión (sin que se realicen operaciones de lectura o de escritura) que pueden operar mediante baterías cuando se interrumpe la energía principal. Desde luego que la principal ventaja de la RAM es que se puede escribir en ella y leer de ella con la misma rapidez y facilidad.

El siguiente análisis sobre la RAM se tratará parte del material que cubrimos en nuestro estudio sobre la ROM, ya que muchos de los conceptos básicos son comunes para ambos tipos de memorias.

12-11 ARQUITECTURA DE LA RAM

Al igual que con la ROM, es útil pensar en la RAM como si consistiera de un número de registros, cada uno de los cuales almacena una sola palabra de datos y tiene una dirección única. Por lo general las RAMs tienen capacidades de palabras de 1K, 4K, 8K, 16K, 64K, 128K, 256K y 1024K, con tamaños de palabra de uno, cuatro u ocho bits. Como veremos más adelante, la capacidad de palabras y el tamaño de palabra pueden expandirse mediante la combinación de chips de memoria.

La figura 12-19 muestra la arquitectura simplificada de una RAM que almacena 64 palabras de cuatro bits cada una (es decir, una memoria de 64 × 4). Estas pala-

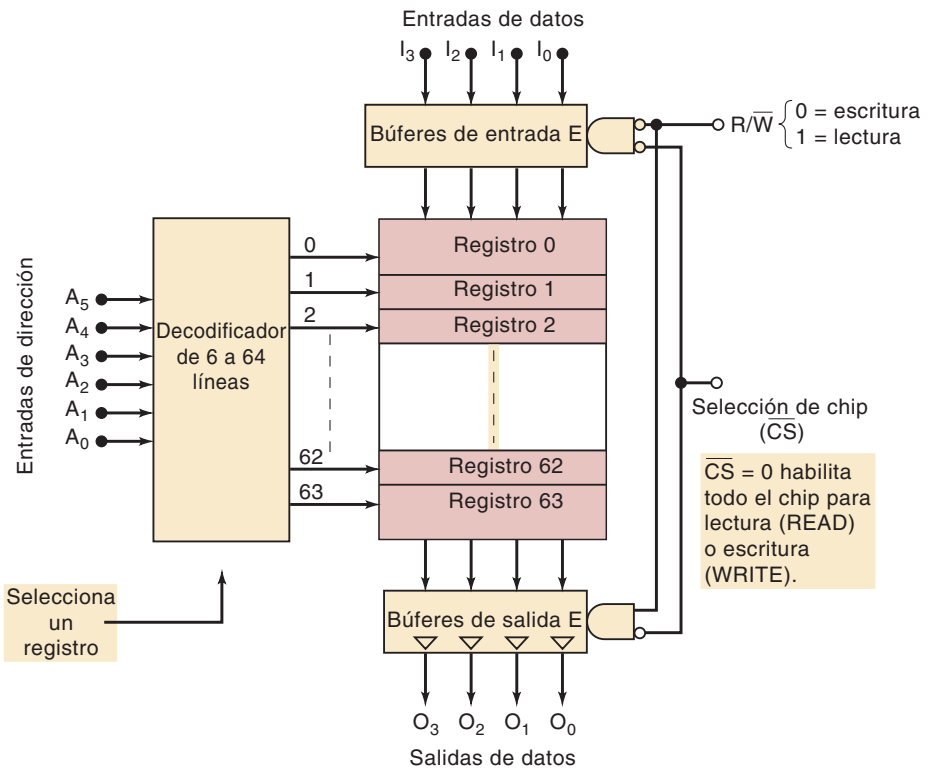


FIGURA 12-19
Organización interna de una RAM de 64 × 4.

bras tienen direcciones que varían de 0 a 63_{10}. Para poder seleccionar una de las 64 ubicaciones de dirección para leer o escribir, se aplica un código de dirección binario a un circuito decodificador. Como $64 = 2^6$, el decodificador requiere un código de entrada de seis bits. Cada código de dirección activa una salida específica del decodificador, que a su vez habilita su registro correspondiente. Por ejemplo, suponga que el código de dirección que se aplica es:

$$A_5A_4A_3A_2A_1A_0 = 011010$$

Como $011010_2 = 26_{10}$, la salida 26 del decodificador cambiará a nivel ALTO y seleccionará el registro 26 para una operación de lectura o de escritura.

Operación de lectura

El código de dirección selecciona un registro en el chip de memoria para leer o escribir. Para poder *leer* el contenido del registro seleccionado, la entrada LEER/ESCRIBIR ($R/\overline{W}$)* debe ser un 1. Además, la entrada SELECCIÓN DE CHIP ($\overline{CS}$) debe activarse (un 0 en este caso). La combinación de $R/\overline{W} = 1$ y $\overline{CS} = 0$ habilita los búferes de salida, de manera que aparecerá el contenido del registro seleccionado en las cuatro salidas de datos. $R/\overline{W} = 1$ también *deshabilita* los búferes de entrada, de manera que las entradas de datos no afecten a la memoria durante una operación de lectura.

Operación de escritura

Para escribir una nueva palabra de cuatro bits en el registro seleccionado se requiere que $R/\overline{W} = 0$ y $\overline{CS} = 0$. Esta combinación *habilita* los búferes de entrada, de manera que la palabra de cuatro bits que se aplica a las entradas de datos se cargue en el registro seleccionado. La condición $R/\overline{W} = 0$ también *deshabilita* los búferes de salida, los cuales son triestado y, por lo tanto, las salidas de datos se encontrarán en su estado Hi-Z durante una operación de escritura. Desde luego que la operación de escritura destruye la palabra que estaba almacenada antes en esa dirección.

Selección de chip

La mayoría de los chips de memoria tienen una o más entradas *CS*, las cuales se utilizan para habilitar todo el chip o deshabilitarlo por completo. En el modo deshabilitado, todas las entradas y salidas de datos están deshabilitadas (Hi-Z), de manera que no puedan llevarse a cabo operaciones de lectura ni de escritura. En este modo, el contenido de la memoria no se ve afectado. La razón de tener entradas *CS* le será clara cuando combinemos chips de memoria para obtener memorias más grandes. Observe que muchos fabricantes llaman a estas entradas HABILITACIÓN DE CHIP (*CE*). Cuando las entradas *CS* o *CE* se encuentran en su estado activo, se dice que el chip de memoria está *seleccionado*; en caso contrario, se dice que está *deseleccionado*. Muchos CIs de memoria están diseñados para consumir mucho menos energía cuando están deseleccionados. En los sistemas de memoria extensos, para una operación de memoria dada se seleccionarán uno o más chips de memoria mientras todos los demás se deseleccionan. Más adelante veremos más detalles sobre este tema.

Terminales comunes de entrada/salida

Para poder conservar terminales en un paquete de CI, por lo general, los fabricantes combinan las funciones de entrada y salida de datos mediante el uso de terminales comunes de entrada/salida. La entrada $R/\overline{W}$ controla la función de estas terminales de E/S. Durante una operación de lectura, las terminales de E/S actúan como salidas de datos que reproducen el contenido de la ubicación de dirección seleccionada.

* Algunos fabricantes utilizan el símbolo $\overline{WE}$ (habilitación de escritura) o $\overline{W}$ en vez de $R/\overline{W}$. En cualquier caso, la operación es la misma.

Durante una operación de escritura, las terminales de E/S actúan como entradas de datos en las que se aplicarán los datos que se van a escribir.

Para ver por qué ocurre esto, considere el chip de la figura 12-19. Con terminales de entrada y salida separadas, se requiere un total de 18 terminales (incluyendo tierra y fuente de energía). Con cuatro terminales comunes de E/S, sólo se requieren 14 terminales. El ahorro de terminales se vuelve más considerable para chips con un tamaño de palabra más grande.

EJEMPLO 12-9

La 2147H es una RAM tipo NMOS, la cual se organiza como de 4K × 1, con entradas separadas de entrada y salida de datos, y una entrada de selección de chip activa en BAJO. Dibuje el símbolo lógico para este chip, en donde se muestren las funciones de todas las terminales.

Solución

El símbolo lógico se muestra en la figura 12-20(a).

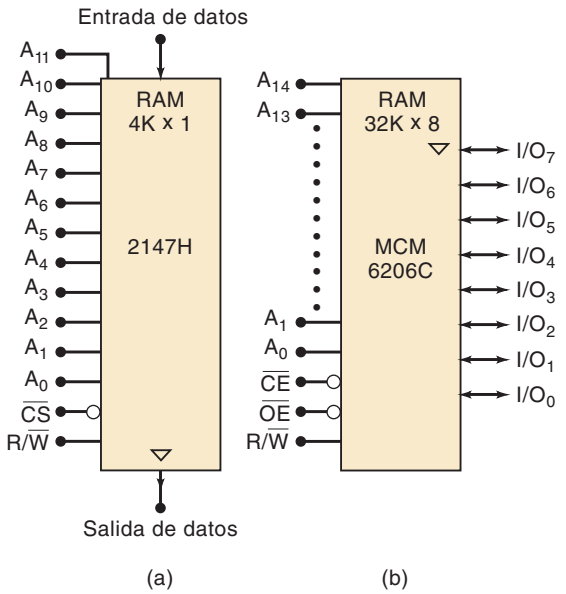


FIGURA 12-20 Símbolos lógicos para (a) el chip de RAM 2147H; (b) la RAM MCM6206C.

EJEMPLO 12-10

El chip MCM6206C es una RAM tipo CMOS con capacidad de 32K × 8, terminales comunes de E/S, una señal de habilitación de chip activa en BAJO y una señal de habilitación de salida activa en BAJO. Dibuje el símbolo lógico.

Solución

El símbolo lógico se muestra en la figura12-20(b).

En la mayoría de las aplicaciones se utilizan dispositivos de memoria con un bus de datos bidireccional, como el que vimos en el capítulo 9. Para este tipo de sistema, aún si el chip de memoria tuviera terminales separadas de entrada y salida, se conectarían entre sí en el mismo bus de datos. Una RAM que tiene terminales separadas de entrada y salida se denomina como RAM de puerto dual. Este tipo de memorias se utiliza en aplicaciones en las que la velocidad es muy importante y los datos entrantes provienen de un dispositivo distinto al que va a recibir los datos. Un buen ejemplo es la RAM de video en su PC. La tarjeta de video debe leer la RAM en forma repetida para regenerar la pantalla y llenarla en forma constante con la nueva información actualizada del bus del sistema.

PREGUNTAS DE REPASO

1. Describa las condiciones de entrada necesarias para leer una palabra desde una ubicación de dirección de RAM específica.
2. ¿Por qué algunos chips de RAM tienen terminales comunes de entrada/salida?
3. ¿Cuántas terminales se requieren para la RAM MCM6208C de 64K × 4, con una entrada *CS* y E/S común?

12-12 RAM ESTÁTICA (SRAM)

La operación de la RAM que hemos visto hasta este punto se aplica a una **RAM estática** (que puede almacenar datos mientras se aplique energía al chip). En esencia, las celdas de memoria de la RAM estática son flip-flops que permanecerán en un estado dado (almacenan un bit) de manera indefinida, siempre y cuando no se interrumpa la energía del circuito. En la sección 12-13 describiremos la **RAM dinámica**, que almacena datos en forma de cargas en los capacitares. Con las RAMs dinámicas, los datos almacenados desaparecerán en forma gradual debido a la descarga del capacitor, por lo que es necesario **regenerar** los datos en forma periódica (es decir, recargar los capacitores).

Las RAMs estáticas (SRAMs) están disponibles en las tecnologías bipolar, MOS y BiCMOS; la mayoría de las aplicaciones utilizan RAMs tipo NMOS o CMOS. Como dijimos antes, los dispositivos bipolares tienen la ventaja en velocidad (aunque CMOS está cada vez más cerca) y los dispositivos MOS tienen mucho mayor capacidad y un bajo consumo de energía. La figura 12-21 muestra para fines de comparación una celda de memoria estática bipolar común y una celda de memoria estática NMOS común. La celda bipolar contiene dos transistores bipolares y dos resistencias, mientras que la celda NMOS contiene cuatro MOSFETs de canal N. La celda bipolar requiere más área en el chip que la celda MOS, ya que un transistor bipolar es más complejo que un MOSFET y porque la celda bipolar requiere resistencias separadas, mientras que la celda MOS utiliza MOSFETs como resistencias (Q_3 y Q_4). Una celda de memoria CMOS sería similar a la celda NMOS, sólo que utilizaría MOSFETs de canal P en vez de Q_3 y Q_4. Esto produce el consumo más bajo de energía pero incrementa la complejidad del chip.

Sincronización de la RAM estática

El uso más común de los CIs de RAM es como memoria interna de una computadora. La CPU (unidad central de proceso) realiza en forma continua operaciones de lectura y de escritura en esta memoria, a una velocidad bastante rápida, la cual se determina con base en las limitaciones de la CPU. Los chips de memoria que se integran

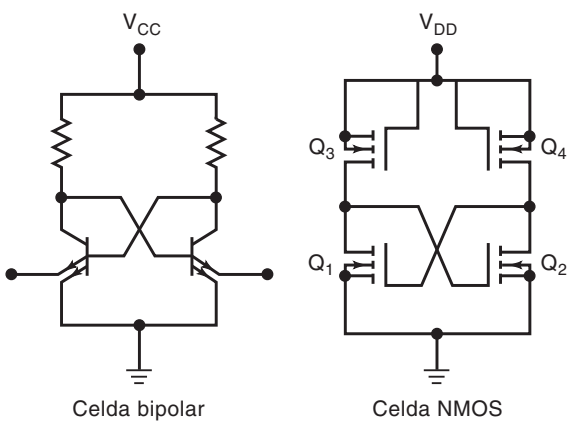


FIGURA 12-21 Celdas bipolares y NMOS de RAM estática típicas.

a la CPU deben ser lo bastante rápidos como para responder a las instrucciones de lectura y escritura de la CPU; un diseñador de computadoras debe tomar en cuenta las diversas características de sincronización de la RAM.

No todas las RAMs tienen las mismas características de sincronización, pero la mayoría de ellas son similares y, por lo tanto, utilizaremos un conjunto típico de características para fines ilustrativos. La nomenclatura para los distintos parámetros de sincronización variará de un fabricante a otro, pero casi siempre es fácil determinar el significado de cada parámetro con base en los diagramas de sincronización de la memoria en las hojas de datos de RAM. La figura 12-22 muestra los diagramas de sincronización para un ciclo completo de lectura y un ciclo completo de escritura en un chip de RAM ordinario.

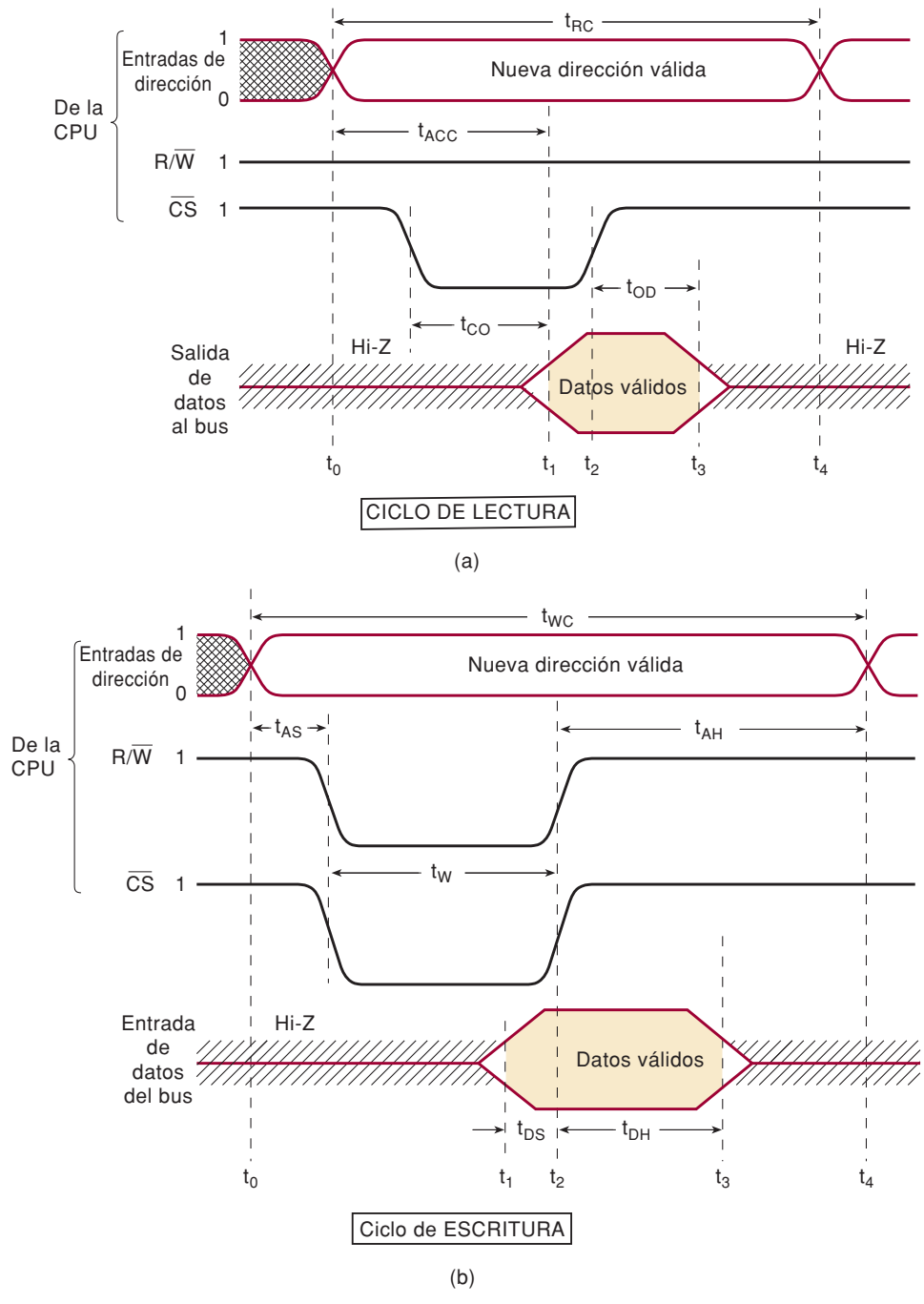


FIGURA 12-22
Sincronización típica para la RAM estática: (a) ciclo de lectura; (b) ciclo de escritura.

Ciclo de lectura

Los diagramas de tiempo de la figura 12-22(a) muestran cómo las entradas de dirección, $R/\overline{W}$ y selección de chip se comportan durante un ciclo de lectura de memoria. Como se indica, la CPU suministra estas señales de entrada a la RAM cuando desea leer datos de una ubicación de dirección de RAM específica. Aunque una RAM puede tener muchas entradas de dirección que provengan del bus de direcciones de la CPU, por cuestión de claridad en el diagrama sólo se muestran dos. También se muestra la salida de datos de la RAM; vamos a suponer que esta RAM en particular tiene una salida de datos. Recuerde que la salida de datos de la RAM se conecta al bus de datos de la CPU (figura 12-5).

El ciclo de lectura comienza en el tiempo t_0. Antes de ese tiempo, las entradas de dirección pueden tener cualquier valor de dirección que esté en el bus de dirección debido a la operación anterior. Como la señal de selección de chip de la RAM no está activa, no responderá a su dirección "antigua". Observe que la línea $R/\overline{W}$ está en ALTO antes de t_0 y permanece en ALTO durante todo el ciclo de lectura. Por lo general, en la mayoría de los sistemas de memoria la línea $R/\overline{W}$ se mantiene en el estado ALTO, sólo cuando se lleva a BAJO durante un ciclo de escritura. La salida de datos de la RAM se encuentra en su estado Hi-Z, debido a que $\overline{CS} = 1$.

En t_0, la CPU aplica una nueva dirección a las entradas de la RAM; ésta es la dirección de la ubicación que se va a leer. Después de dejar pasar un tiempo para que las señales de dirección se estabilicen, se activa la línea $\overline{CS}$. Como respuesta, la RAM coloca los datos de la ubicación direccionada en la línea de salida de datos en t_1. El tiempo entre t_0 y t_1 es el tiempo de acceso de la RAM, t_{ACC}, y representa el tiempo que transcurre entre la aplicación de la nueva dirección y la aparición de los datos de salida válidos. El parámetro de sincronización t_{CO} es el tiempo que tarda la salida de la RAM en cambiar de Hi-Z a un nivel de datos válido, una vez que se activa $\overline{CS}$.

En el tiempo t_2 la señal $\overline{CS}$ regresa al nivel ALTO y la salida de la RAM regresa a su estado Hi-Z después de un intervalo de tiempo, t_{OD}. Así, los datos de la RAM se encontrarán en el bus de datos entre t_1 y t_3. La CPU puede recibir la información que provienen del bus de datos en cualquier punto durante este intervalo. En la mayoría de las computadoras, la CPU utilizará la PGT e la señal $\overline{CS}$ en t_2 para fijar estos datos en uno de sus registros internos.

El tiempo completo del ciclo de lectura (t_{RC}) se extiende desde t_0 a t_4, cuando la CPU cambia las entradas de dirección por una dirección distinta para el siguiente ciclo de lectura o de escritura.

Ciclo de escritura

La figura 12-22(b) muestra la actividad de las señales para un ciclo de escritura que comienza cuando la CPU suministra una nueva dirección a la RAM, en el tiempo t_0. La CPU lleva las líneas $R/\overline{W}$ y $\overline{CS}$ a BAJO después de esperar durante un intervalo de tiempo t_{AS}, al cual se le conoce como *tiempo de establecimiento de dirección*. Esto da tiempo a los decodificadores de dirección de la RAM de responder a la nueva dirección. Las señales $R/\overline{W}$ y $\overline{CS}$ se mantienen en BAJO durante un intervalo de tiempo t_W, conocido como el intervalo de tiempo de escritura.

Durante este intervalo de escritura, en el tiempo t_1 la CPU aplica datos válidos al bus de datos para escribirlos en la RAM. Estos datos deben retenerse en la entrada de la RAM durante por lo menos un intervalo de tiempo t_{DS} antes de, y por lo menos un intervalo de tiempo t_{DF} después de la desactivación de $R/\overline{W}$ y $\overline{CS}$ en t_2. Al intervalo t_{DS} se le conoce como *tiempo de establecimiento de datos* y a t_{DH} se le conoce como *tiempo de retención de datos*. De manera similar, las entradas de dirección deben permanecer estables durante el intervalo de tiempo de retención de la dirección t_{AH}, después de t_2. Si no se cumple con cualquiera de estos requerimientos de tiempo de establecimiento o de tiempo de retención, la operación de escritura no se realizará en forma confiable.

El tiempo del ciclo de escritura completo (t_{WC}) se extiende desde t_0 hasta t_4, cuando la CPU cambia las líneas de dirección a una nueva dirección para el siguiente ciclo de escritura o lectura.

El tiempo del ciclo de lectura (t_{RC}) y el tiempo del ciclo de escritura (t_{WC}) son los que en esencia determinan qué tan rápido puede operar un chip de memoria. Por ejemplo, en una aplicación real la CPU estará leyendo con frecuencia palabras de datos sucesivas de memoria, una después de la otra. Si la memoria tiene un t_{RC} de 50 ns, la CPU puede leer una palabra cada 50 ns, o 20 millones de palabras por segundo; si t_{RC} = 10 ns, la CPU puede leer 100 millones de palabras por segundo. La tabla 12-2 muestra los tiempos mínimos para el ciclo de lectura y el ciclo de escritura de ciertos chips de RAM estática representativos.

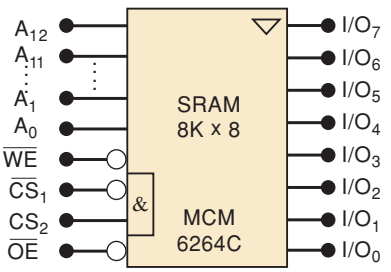
TABLA 12-2

Dispositivo	t_{RC}(mín) (ns)	t_{WC}(mín) (ns)
CMOS MCM6206C, 32K × 8	15	15
NMOS 2147H, 4K × 1	35	35
BiCMOS MCM6708A, 64K × 4	8	8

Chip de SRAM real

La RAM MCM6264C CMOS de 8K × 8 es un ejemplo de un CI de SRAM real, con tiempos de ciclo de lectura y ciclo de escritura de 12 ns y un consumo de energía en suspensión de sólo 100 mW. En la figura 12-23 se muestra el símbolo lógico para este CI. Observe que tiene 13 entradas de dirección, ya que $2^{13} = 8192 = 8K$ y ocho líneas de E/S de datos. Las cuatro entradas de control determinan el modo de operación del dispositivo, de acuerdo con la tabla de modos que lo acompaña.

La entrada $\overline{WE}$ es la misma que la entrada $R/\overline{W}$ que hemos estado utilizando. Un nivel BAJO en $\overline{WE}$ hará que se escriban datos en la RAM, siempre y cuando esté seleccionado el dispositivo (que ambas entradas de selección de chip estén activas). Observe que se utiliza el símbolo "&" para denotar que *ambas* deben estar activas. Un nivel ALTO en $\overline{WE}$ producirá la operación de lectura, siempre y cuando esté seleccionado el dispositivo y los búferes de salida se habiliten mediante $\overline{OE}$ = BAJO. Cuando está deseleccionado, el dispositivo se encuentra en su modo de bajo consumo de energía y ninguna de las otras entradas tiene efecto.



	Entradas				
Modo	$\overline{WE}$	$\overline{CS_1}$	CS_2	$\overline{OE}$	Terminales de E/S
LECTURA	1	0	1	0	$DATOS_{SAL}$
ESCRITURA	0	0	1	X	$DATOS_{ENT}$
Deshabilita salida	1	X	X	1	Alta Imp. (Hi-Z)
No seleccionado (modo suspendido)	X	1	X	X	Alta Imp. (Hi-Z)
	X	X	0	X	

X = no importa

FIGURA 12-23
Símbolo y tabla de modos para el MCM6264C tipo CMOS.

La mayoría de los dispositivos que hemos visto en este capítulo están disponibles de parte de varios fabricantes distintos. Cada fabricante puede ofrecer dispositivos distintos de la misma dimensión (por ejemplo, 32K × 8) pero con distintas especificaciones o características. También hay varios tipos de encapsulados disponibles, como DIP, PLCC y varias formas de ala de gaviota y montaje en superficie.

A medida que analice los diversos dispositivos de memoria que hemos descrito en este capítulo, observará ciertas similitudes. Por ejemplo, analice los chips en la figura 12-24 y tome nota de las asignaciones de las terminales. El hecho de que se asigne la misma función a las mismas terminales en todos estos diversos dispositivos, fabricados

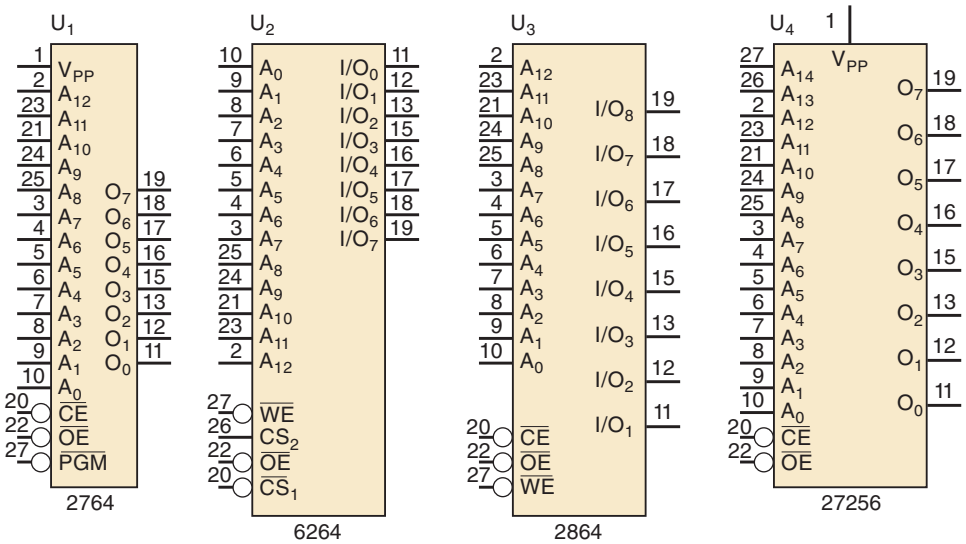


FIGURA 12-24
Encapsulados de memoria estándar de JEDEC.

por distintas compañías, no es coincidencia. Los estándares industriales creados por el Consejo Común de Ingeniería de Dispositivos Electrónicos (**JEDEC**) han originado la creación de dispositivos de memoria compatibles e intercambiables.

EJEMPLO 12-11

Un sistema se configura para un chip ROM de 8K × 8 (2764) y dos chips SRAM de 8K × 8 (6264). Todo el espacio de la ROM de 8K se utiliza para almacenar las instrucciones del microprocesador. Usted desea actualizar el sistema para que tenga cierta capacidad de almacenamiento de lectura/escritura no volátil. ¿Puede modificarse el circuito existente para dar cabida a las nuevas revisiones?

Solución

Sólo basta con sustituir un chip EEPROM 2864 en uno de los zócalos de RAM. La única diferencia funcional está en los requerimientos de un tiempo de ciclo de escritura mucho más largo de la EEPROM. Por lo general, esto se resuelve al modificar el programa de la microcomputadora que utiliza el dispositivo de memoria. Como no hay espacio libre en la ROM para estos cambios, necesitamos una ROM más grande. Una ROM de 32K × 8 (27C256) tiene básicamente la misma distribución de terminales que una 2764. Tan sólo tenemos que conectar dos líneas de dirección más (A_{13} y A_{14}) al zócalo de la ROM y sustituir el chip anterior con un chip 27C256.

Muchos sistemas de memoria aprovechan la versatilidad que proporcionan los estándares de JEDEC. Las terminales que son comunes para todos los dispositivos se cablean de forma fija a los buses del sistema. Las pocas terminales que son distintas entre los diversos dispositivos se conectan a circuitos que pueden modificarse con facilidad para configurar el sistema para el tamaño y tipo adecuado de dispositivo de memoria. Esto permite al usuario reconfigurar el hardware sin necesidad de cortar o soldar en el tablero. El circuito de configuración puede ser tan simple como unos puentes removibles o interruptores DIP configurables por el usuario, o tan complicados como un dispositivo lógico programable en el circuito que la computadora pueda configurar o modificar para cumplir con los requerimientos del sistema.

PREGUNTAS DE REPASO

1. ¿Qué diferencia hay entre la celda de una RAM estática y la celda de una RAM dinámica?
2. ¿Cuál tecnología de memoria utiliza, por lo general, la menor cantidad de energía?
3. ¿Qué dispositivo coloca datos en el bus de datos durante un ciclo de lectura?
4. ¿Qué dispositivo coloca datos en el bus de datos durante un ciclo de escritura?
5. ¿Qué parámetros de sincronización de la RAM determinan su velocidad de operación?
6. *Verdadero o falso:* un nivel BAJO en $\overline{OE}$ habilitará los búferes de salida de un MCM6264C, siempre y cuando ambas entradas de selección de chip estén activas.
7. ¿Qué se debe hacer con la terminal 26 y la terminal 27 si se sustituye un 27256 por un 2764?

12-13 RAM DINÁMICA (DRAM)

Las RAMs dinámicas se fabrican mediante el uso de tecnología MOS y se distinguen por su alta capacidad, bajo requerimiento de energía y velocidad moderada de operación. Como dijimos antes, a diferencia de las RAMs estáticas que almacenan información en FFs, las RAMs dinámicas almacenan 1s y 0s en forma de cargas en un pequeño capacitor MOS (por lo general, de unos cuantos pico-Farads). Debido a la tendencia de estas cargas de fugarse después de un periodo de tiempo, las RAMs dinámicas requieren una recarga periódica de las celdas de memoria; a esto se le conoce como regenerar la RAM dinámica. En los chips DRAM modernos, cada celda de memoria debe regenerarse cada 2, 4 u 8 ms, o se perderán sus datos.

La necesidad de regenerarse es una desventaja de la RAM dinámica en comparación con la RAM estática, debido a que puede requerir de circuitos de soporte externos. Algunos chips de DRAM tienen circuitos de control de regeneración integrados, los cuales no requieren hardware externo adicional pero sí una sincronización especial de las señales de control de entrada del chip. Además y como veremos más adelante, las entradas de dirección para una DRAM deben manejarse de una manera menos directa que con la SRAM. En general, es más complejo diseñar circuitos con DRAM y utilizar este tipo de memoria en un sistema que la SRAM. No obstante, sus capacidades mucho mayores y su consumo de energía mucho menor hacen de las DRAMs la mejor opción para memoria en sistemas en donde las consideraciones de diseño más importantes son mantener un tamaño reducido, un bajo costo y bajo consumo de energía.

Para aplicaciones en las que la velocidad y la reducción en complejidad son más importantes que las consideraciones de costo, espacio y energía, las RAMs estáticas siguen siendo la mejor opción. Por lo general, son más veloces que las RAMs dinámicas y no requieren operación de regeneración. Es más fácil diseñar circuitos con RAMs estáticas, pero no pueden competir con la mayor capacidad y el bajo consumo de energía de las RAMs dinámicas.

Debido a su estructura de celdas simples, las DRAMs tienen cuatro veces la densidad de las SRAMs. Este aumento en densidad permite colocar cuatro veces más capacidad de memoria en una sola tarjeta. El costo por bit del almacenamiento en RAM dinámica es comúnmente de una quinta a una cuarta parte de las RAMs estáticas. Se obtiene un ahorro adicional en costo ya que los requerimientos de bajo consumo de energía de la RAM dinámica, que, por lo general, son de una sexta parte a la mitad de los de una RAM estática, permiten el uso de fuentes de energía más pequeñas y menos costosas.

Las principales aplicaciones de las SRAMs son en áreas en las que sólo se necesitan pequeñas cantidades de memoria o en donde se requiere una alta velocidad. Muchos instrumentos y aparatos controlados por microprocesadores tienen muy pocos requerimientos en cuanto a la capacidad de memoria. Algunos instrumentos

tales como los osciloscopios de almacenamiento digital y los analizadores lógicos, requieren de una memoria de muy alta velocidad. Para aplicaciones tales como estas se utiliza, por lo general, la SRAM.

La memoria interna principal de la mayoría de las microcomputadoras personales (por ejemplo, PCs basadas en Windows o Macs) utiliza DRAM debido a su alta capacidad y bajo consumo de energía. No obstante, estas computadoras utilizan algunas veces pequeñas cantidades de SRAM para las funciones que requieren de una máxima velocidad, como los gráficos de video, las tablas de búsqueda y la memoria caché.

PREGUNTAS DE REPASO

1. ¿Cuáles son las principales desventajas de la RAM dinámica, en comparación con la estática?
2. Liste las ventajas de la RAM dinámica en comparación con la RAM estática.
3. ¿Qué tipo de RAM esperaría encontrar en los módulos de memoria principal de su PC?

12-14 ESTRUCTURA Y OPERACIÓN DE LA RAM DINÁMICA

La arquitectura interna de la RAM dinámica puede visualizarse como un arreglo de celdas de un solo bit, como se muestra en la figura 12-25. Aquí se ordenan 16,384 celdas en un arreglo de 128 × 128. Cada celda ocupa una posición única de fila y columna dentro del arreglo. Se necesitan catorce entradas de dirección para seleccionar una de las celdas (2^{14} = 16,384); los bits de dirección inferiores (A_0-A_6) seleccionan la columna y los bits de mayor orden (A_7-A_{13}) seleccionan la fila. Cada dirección de 14 bits selecciona una celda única en la que se va a escribir, o de la que se va a leer información. La estructura en la figura 12-25 es un chip DRAM de 16K × 1. Hoy en día los chips DRAM están disponibles en diversas configuraciones. Las DRAMs con un tamaño de palabra de cuatro bits (o mayor) tienen una distribución de celdas similar a la de la figura 12-25, sólo que cada posición en el arreglo contiene cuatro celdas y cada dirección que se aplica selecciona un grupo de cuatro celdas para una operación de lectura o de escritura. Como veremos más adelante, también pueden obtenerse tamaños de palabra más grandes si se combinan varios chips en el orden apropiado.

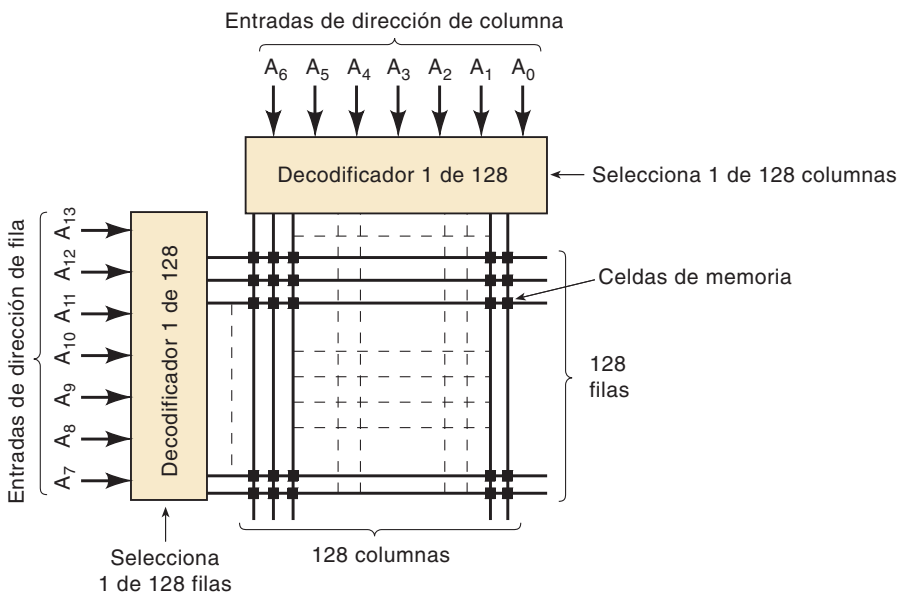


FIGURA 12-25
Distribución de celdas en una RAM dinámica de 16K × 1.

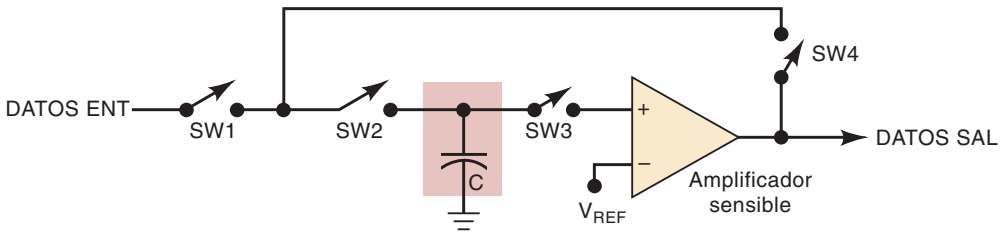


FIGURA 12-26
Representación simbólica de una celda de memoria dinámica. Durante una operación de ESCRITURA se cierran los interruptores semiconductores SW1 y SW2. Durante una operación de lectura se cierran todos los interruptores, excepto SW1.

La figura 12-26 es una representación simbólica de una celda de memoria dinámica y sus circuitos asociados. Muchos de los detalles del circuito no se muestran, pero podemos utilizar este diagrama simplificado para describir las ideas esenciales relacionadas con las operaciones de escritura y lectura en una DRAM. Los interruptores del SW1 al SW4 son en realidad MOSFETs que se controlan mediante varias salidas de un decodificador de direcciones y la señal $R/\overline{W}$. Desde luego que el capacitor es propiamente la celda de almacenamiento. Un amplificador sensible puede dar servicio a una columna completa de celdas de memoria, pero sólo opera en el bit de la fila seleccionada.

Para escribir datos en la celda, las señales de la decodificación de direcciones y la lógica de lectura/escritura cerrarán los interruptores SW1 y SW2, y mantendrán a SW3 y SW4 abiertos. Esto conecta a los datos de entrada con el capacitor *C*. Un 1 lógico en la entrada de datos carga a *C* y un 0 lógico lo descarga. Después los interruptores se abren para que *C* se desconecte del resto del circuito. En teoría, *C* retendría su carga por un tiempo indefinido, pero siempre hay una ruta de fuga a través de los interruptores apagados, de manera que *C* perderá su carga en forma gradual.

Para leer los datos de la celda se cierran los interruptores SW2, SW3 y SW4, y SW1 se mantiene abierto. Esto conecta el voltaje almacenado en el capacitor con el *amplificador sensible*. Este amplificador compara el voltaje con cierto valor de referencia para determinar si es un 0 o un 1 lógico y produce un valor sólido de 0 V o 5 V para la salida de datos. Esta salida de datos se conecta también a *C* (SW2 y SW4 están cerrados) y regenera el voltaje del capacitor al recargarlo o descargarlo. En otras palabras, el bit de datos en una celda de memoria se regenera cada vez que se lee.

Multiplexaje de direcciones

El arreglo de la DRAM de 16K × 1 que se muestra en la figura 12-25 es obsoleto y casi no está disponible en el mercado. Tiene 14 entradas de dirección; un arreglo de DRAM de 64K × 1 tendría 16 entradas de dirección. Una DRAM de 1M × 4 necesita 20 entradas de dirección; una de 4M × 1 necesita 22 entradas de dirección. Los chips de memoria de alta capacidad tales como éstos requerirían muchas terminales si cada entrada de dirección requiriera una terminal separada. Para poder reducir el número de terminales en sus chips DRAM de alta capacidad, los fabricantes utilizan el **multiplexaje de direcciones**, en el cual cada terminal de entrada de dirección puede alojar dos bits de dirección distintos. El ahorro en el número de terminales se traduce en una considerable disminución en el tamaño de los encapsulados de CI. Esto es muy importante en los tableros de memoria de gran capacidad, en donde se desea maximizar la cantidad de memoria que puede acomodarse en un tablero.

En las siguientes discusiones vamos a describir el orden en el cual se lleva a cabo el multiplexaje de direcciones en los chips DRAM. Hay que recalcar que en las DRAMs antiguas de pequeña capacidad la convención era presentar la dirección de menor orden especificando primero la fila, seguida de la dirección de mayor orden que especificaba la columna. Las DRAMs más recientes y los controladores que realizan el multiplexaje utilizan la convención opuesta de aplicar los bits de mayor orden como la dirección de fila y después los bits de menor orden como la dirección de columna. Nosotros describiremos la convención más reciente, pero usted deberá tener en cuenta este cambio cuando estudie los sistemas antiguos.

Utilizaremos la DRAM TMS44100 de 4M × 1 de Texas Instruments para ilustrar la operación de los chips DRAM en la actualidad. El diagrama de bloques funcional

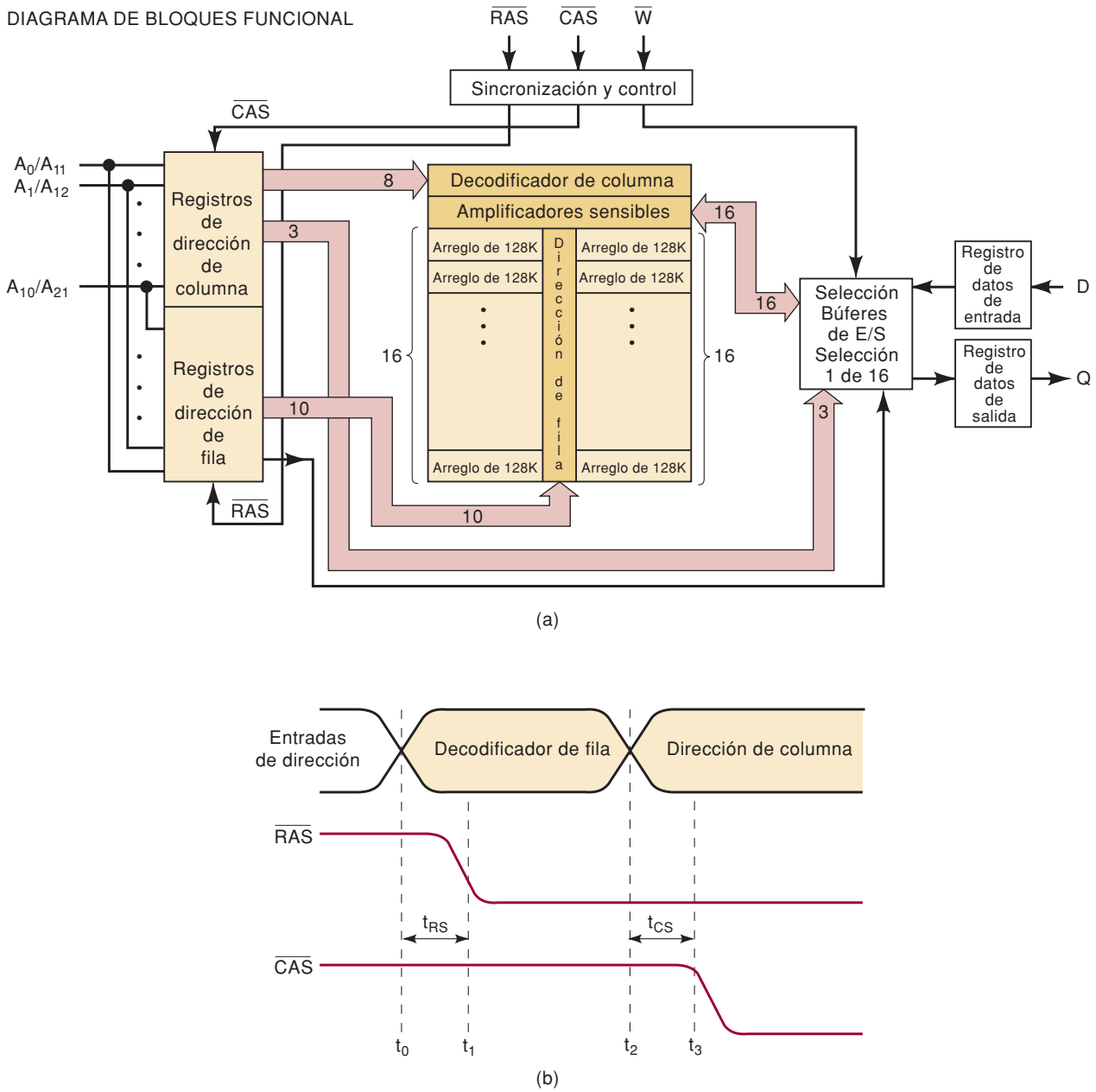


FIGURA 12-27 (a) Arquitectura simplificada de la DRAM TMS44100 de 4M × 1; (b) sincronización de $\overline{RAS}/\overline{CAS}$. (Reimpreso con permiso de Texas Instruments.)

de la arquitectura interna de este chip (que se muestra en la figura 12-27) es común en los diagramas que encontrará en los libros de datos. La distribución del arreglo de memoria en este diagrama puede parecer complicada a primera instancia, pero se puede considerar tan sólo como una versión más grande de la DRAM de 16K × 1 de la figura 12-25. Visto de manera funcional, es un arreglo de celdas distribuidas en forma de 2048 filas por 2048 columnas. El circuito decodificador de dirección selecciona una sola fila; este circuito puede considerarse como un decodificador 1 de 2048. De igual forma, se selecciona una sola columna mediante lo que se conoce como decodificador 1 de 2048. Como las líneas de dirección están multiplexadas, no se puede presentar en forma simultánea toda la dirección completa de 22 bits. Observe que sólo hay 11 líneas de dirección y que van tanto hacia el registro de

dirección de fila como el de columna. Cada uno de los dos registros de dirección almacena la mitad de la dirección de 22 bits. El registro de fila almacena la mitad superior y el registro de columna almacena la mitad inferior. Dos entradas de **estrobo** muy importantes controlan cuando se debe fijar la información de la dirección. La señal **estrobo de dirección de fila (*RAS*)** aplica pulsos de reloj al registro de dirección de fila de 11 bits. La señal **estrobo de dirección de columna (*CAS*)** aplica pulsos de reloj al registro de dirección de columna de 11 bits.

Una dirección de 22 bits se aplica a esta DRAM en dos pasos, mediante el uso de $\overline{RAS}$ y $\overline{CAS}$. En la figura 12-27(b) se muestra la sincronización. Al principio, tanto $\overline{RAS}$ como $\overline{CAS}$ están en ALTO. En el tiempo t_0 se aplica la dirección de fila de 11 bits (A_{11}- A_{21}) a las entradas de dirección. Después de permitir un tiempo para el requerimiento del tiempo de establecimiento (t_{RS}) del registro de dirección de fila, la entrada $\overline{RAS}$ se lleva al nivel BAJO en t_1. Esta NGT carga la dirección de fila en el registro de dirección de fila, por lo que ahora los bits A_{11}- A_{21} aparecen en las entradas del decodificador de fila. El nivel BAJO en $\overline{RAS}$ también habilita este decodificador, para que pueda decodificar la dirección de fila y seleccionar una fila del arreglo.

En el tiempo t_2 se aplica la dirección de columna de 11 bits (A_0-A_{10}) a las entradas de dirección. En t_3 la entrada $\overline{CAS}$ se lleva al nivel BAJO para cargar la dirección de columna en el registro de dirección de columna. $\overline{CAS}$ también habilita el decodificador de columna para que pueda decodificar la dirección de columna y seleccionar una columna del arreglo.

En este punto las dos partes de la dirección están en sus respectivos registros, los decodificadores las han decodificado para seleccionar la celda que corresponde a la dirección de fila y columna, y puede realizarse una operación de lectura o de escritura en esa celda, justo igual que en una RAM estática.

Tal vez haya observado que esta DRAM no tiene una entrada de selección de chip (*CS*). Las señales $\overline{RAS}$ y $\overline{CAS}$ realizan la función de selección de chip, ya que ambas deben estar en BAJO para que los decodificadores puedan seleccionar una celda para leer o escribir.

Como puede ver, hay varias operaciones que deben realizarse antes de que los datos almacenados en la DRAM puedan aparecer en las salidas. El término **latencia** se utiliza a menudo para describir el tiempo requerido para realizar estas operaciones. Cada operación requiere cierta cantidad de tiempo, la cual determina la velocidad máxima a la que podemos acceder a los datos en la memoria.

EJEMPLO 12-12

¿Cuántas terminales se ahorran si utilizamos el multiplexaje de direcciones para una DRAM de 16M × 1?

Solución

Se utilizan doce entradas de dirección en vez de 24; se agregan $\overline{RAS}$ y $\overline{CAS}$; no se requiere CS. Por lo tanto, hay un ahorro neto de once terminales.

En un sistema computacional simple, las entradas de dirección para el sistema de memoria provienen de la unidad central de procesamiento (CPU). Cuando la CPU desea acceder a una ubicación de memoria específica, genera la dirección completa y la coloca en las líneas de dirección que conforman un bus de direcciones. La figura 12-28(a) muestra esto para una pequeña memoria de computadora que tiene una capacidad de 64K palabras y por ello requiere que un bus de direcciones con 16 líneas vaya directamente de la CPU a la memoria.

Este arreglo funciona para la ROM o la RAM estática, pero debe modificarse para la DRAM que utiliza direccionamiento con multiplexaje. Si todos los 64K de la memoria son de DRAM, entonces sólo tendrá ocho entradas de dirección. Esto significa que las 16 líneas de dirección que provienen del bus de direcciones de la CPU deben alimentarse a un circuito multiplexor que transmita ocho bits de dirección a

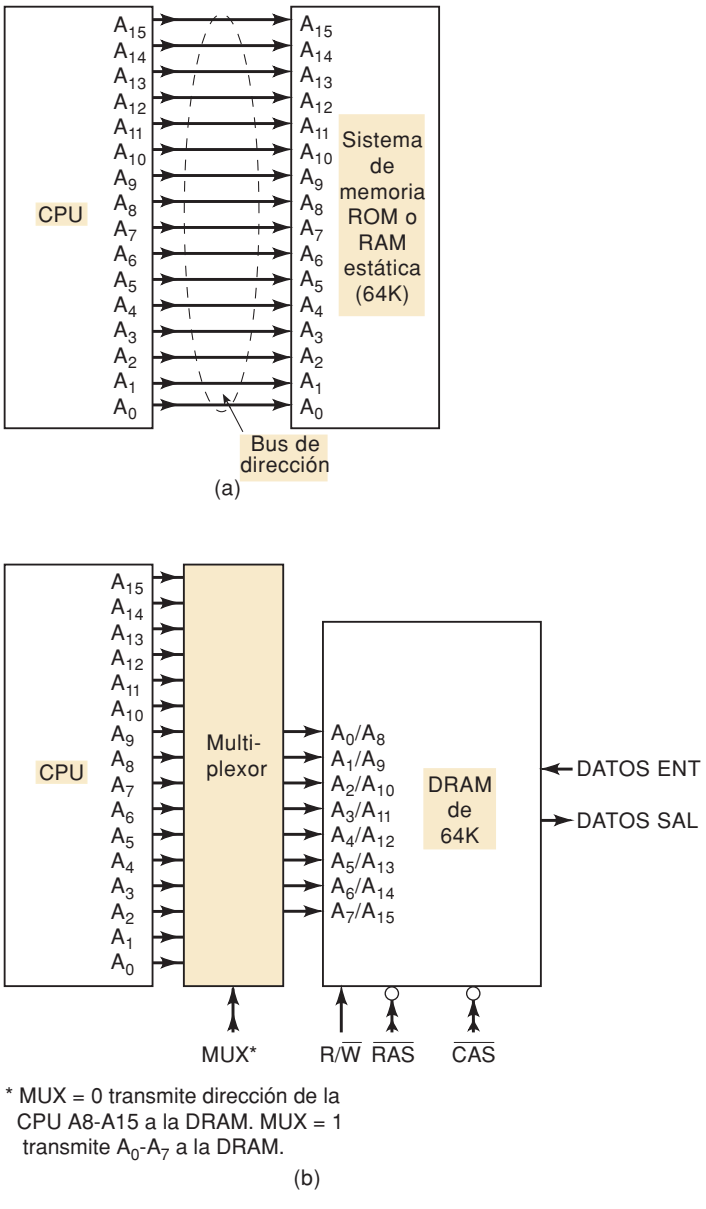


FIGURA 12-28 (a) El bus de direcciones de la CPU que controla la memoria ROM o RAM estática: (b) Direcciones de la CPU que controlan un multiplexor, el cual se utiliza para multiplexar las líneas de dirección de la CPU en la DRAM.

la vez, a las entradas de dirección de la memoria. Esto se muestra en forma simbólica en la figura 12-28(b). La entrada de selección del multiplexor *(MUX)* controla si se van a presentar las líneas de dirección A_0-A_7 de la CPU, o las líneas A_8-A_{15} en las entradas de dirección de la DRAM.

La señal *MUX* debe sincronizarse con las señales *RAS* y *CAS* que aplican los pulsos de reloj para enviar las direcciones a la DRAM. Esto se muestra en la figura 12-29. *MUX* debe estar en BAJO cuando se aplica un pulso a *RAS* para que cambie a BAJO, de manera que las líneas de dirección A_8-A_{15} de la CPU lleguen a las entradas de dirección de la DRAM para cargarse en la NGT de *RAS*. De igual forma, *MUX* debe estar en ALTO cuando se aplica un pulso a *CAS* para que cambie a BAJO, de manera que se presenten las líneas de dirección A_0-A_7 de la CPU en las entradas de la DRAM, para cargarse en la NGT de *CAS*.

No vamos a mostrar aquí el circuito real de multiplexaje y sincronización, sino que lo dejaremos para los problemas de final de capítulo (problemas 12-26 y 12-27).

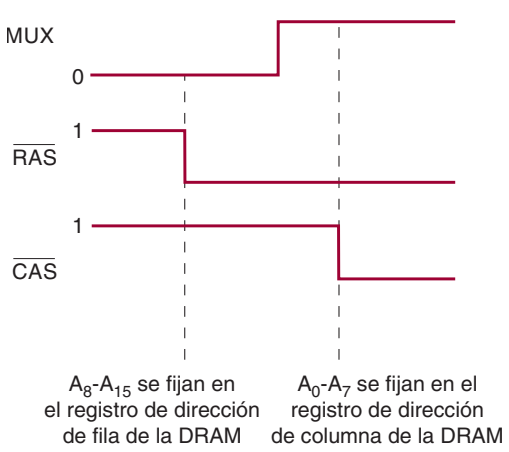


FIGURA 12-29 Sincronización requerida para el multiplexaje de direcciones.

PREGUNTAS DE REPASO

1. Describa la estructura del arreglo de una DRAM de 64K × 1.
2. ¿Cuál es el beneficio del multiplexaje de direcciones?
3. ¿Cuántas entradas de dirección habría en un chip DRAM de 1M × 1?
4. ¿Cuáles son las funciones de las señales $\overline{RAS}$ y $\overline{CAS}$?
5. ¿Cuál es la función de la señal *MUX*?

12-15 CICLOS DE LECTURA/ESCRITURA DE LA DRAM

La sincronización de las operaciones de lectura y escritura de una DRAM es mucho más compleja que para una RAM estática; además hay muchos requerimientos de sincronización críticos que el diseñador de la memoria DRAM debe tener en cuenta. En este punto es probable que una discusión detallada sobre estos requerimientos provoque más confusión que entendimiento, por lo cual vamos a concentrarnos en la secuencia de sincronización básica para las operaciones de lectura y escritura en un pequeño sistema de DRAM, como el de la figura 12-28(b).

Ciclo de lectura de la DRAM

La figura 12-30 muestra la actividad común de las señales durante la operación de lectura. Se supone que *R/W* se encuentra en su estado ALTO a lo largo de toda la operación. La siguiente es una descripción paso a paso de los eventos que ocurren en los tiempos que se indican en el diagrama.

- t_0: *MUX* se lleva al nivel BAJO para aplicar los bits de dirección de fila (A_8-A_{15}) en las entradas de dirección de la DRAM.
- t_1: $\overline{RAS}$ se lleva a BAJO para cargar la dirección de fila en la DRAM.
- t_2: *MUX* cambia a ALTO para colocar la dirección de columna (A_0-A_7) en las entradas de dirección de la DRAM.
- t_3: $\overline{CAS}$ cambia a BAJO para cargar la dirección de columna en la DRAM.
- t_4: la DRAM responde colocando datos válidos provenientes de la celda de memoria seleccionada, en la línea DATOS SAL.
- t_5: *MUX*, $\overline{RAS}$, $\overline{CAS}$ y DATOS SAL regresan a sus estados iniciales.

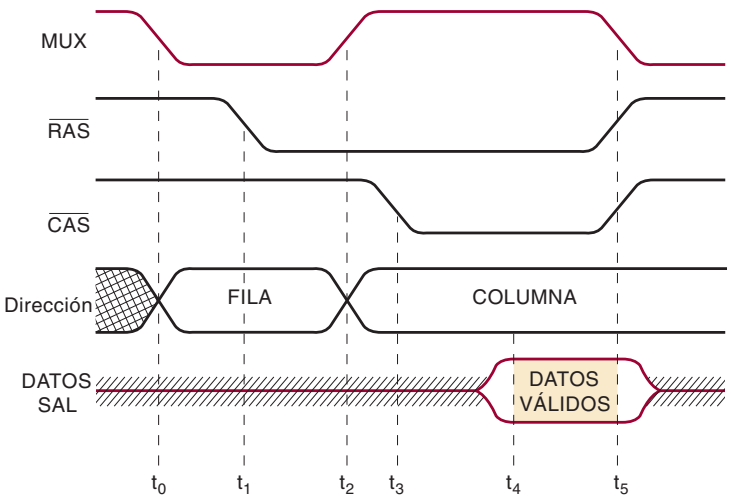


FIGURA 12-30
Actividad de las señales durante una operación de lectura en una RAM dinámica. La entrada $R/\overline{W}$ (no se muestra) se supone en ALTO.

Ciclo de escritura de la DRAM

La figura 12-31 muestra la actividad común de las señales durante una operación de escritura en la DRAM. He aquí una descripción de la secuencia de eventos.

- t_0: el nivel BAJO en *MUX* coloca las direcciones de fila en las entradas de la DRAM.
- t_1: la NGT en $\overline{RAS}$ carga la dirección de fila en la DRAM.
- t_2: *MUX* cambia a ALTO para colocar la dirección de columna en las entradas de la DRAM.
- t_3: la NGT en $\overline{CAS}$ carga la dirección de columna en la DRAM.
- t_4: los datos que se van a escribir se colocan en la línea DATOS ENT.
- t_5: se aplica un pulso a $R/\overline{W}$ para que cambie a BAJO y se escriban los datos en la celda seleccionada.
- t_6: los datos de entrada se extraen de DATOS ENT.
- t_7: *MUX*, $\overline{RAS}$, $\overline{CAS}$ y $R/\overline{W}$ se regresan a sus estados iniciales.

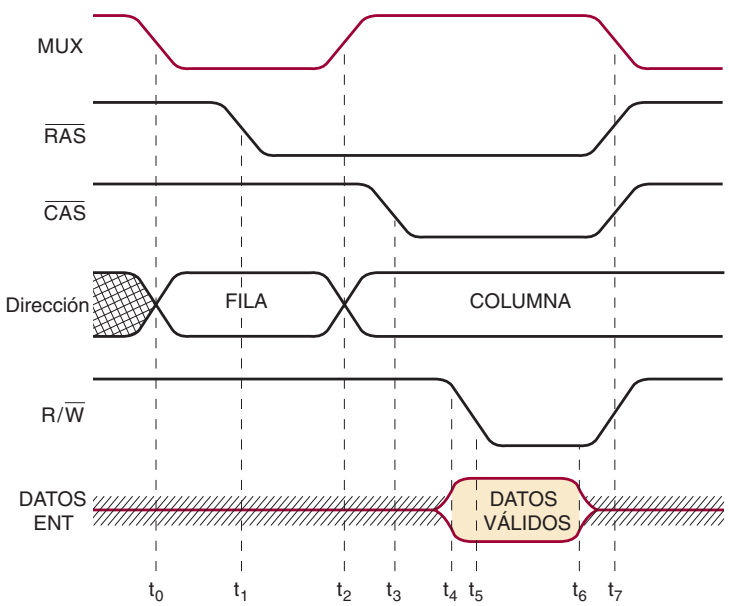


FIGURA 12-31
Actividad de las señales para una operación de escritura en una RAM dinámica.

PREGUNTAS DE REPASO

1. *Verdadero o falso:*
   (a) Durante un ciclo de lectura, la señal $\overline{RAS}$ se activa antes de la señal $\overline{CAS}$.
   (b) Durante una operación de escritura, $\overline{CAS}$ se activa antes de $\overline{RAS}$.
   (c) $R/\overline{W}$ se mantiene en BAJO durante toda la operación de escritura.
   (d) Las entradas de dirección para una DRAM cambiarán dos veces durante una operación de lectura o de escritura.
2. ¿Qué señal de la figura 12-28(b) asegura que aparezca la parte correcta de la dirección completa en las entradas de la DRAM?

12-16 REFRESCO DE LA DRAM

La celda de una DRAM se regenera cada vez que se realiza una operación de lectura sobre esa celda. Cada celda de memoria debe regenerarse por lo general, cada 4 a 16 ms (dependiendo del dispositivo) o se perderán sus datos. Este requerimiento parece en extremo difícil, si no es que imposible, sobre todo para cumplirlo con DRAMs de gran capacidad. Por ejemplo, una DRAM de 1M $\times$ 1 tiene 10^{20} = 1,048,576 celdas. Para asegurar que cada celda se refresque dentro de un lapso no mayor de 4 ms, debe realizar las operaciones de lectura en direcciones sucesivas a la velocidad de una por cada 4 ns (4 ms/1,048,756 $\approx$ 4 ns). Esto es demasiado rápido para cualquier chip DRAM. Por fortuna, los fabricantes han diseñado chips DRAM de manera que

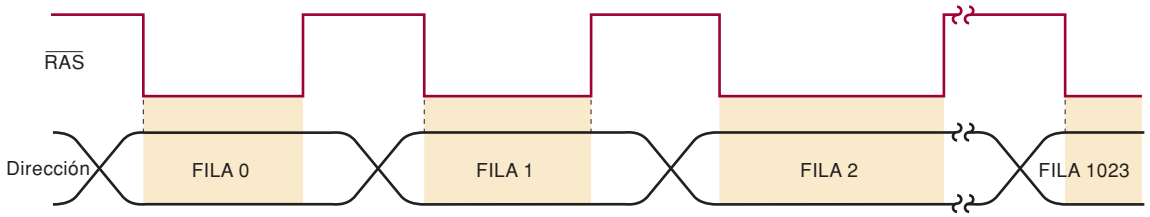
> **cada vez que se realice una operación de lectura en una celda, se regenerarán todas las celdas en esa fila.**

Por lo tanto, es necesario realizar una operación de lectura solo en cada *fila* de un arreglo de DRAM una vez cada 4 ms para garantizar que se refresque cada una de las *celdas* del arreglo. Al referirnos a la DRAM de 4M $\times$ 1 de la figura 12-27(a), si se introduce cualquier dirección en el registro de dirección de fila, se regenerarán de manera automática todas (las 2048) celdas en esa fila.

Es evidente que esta característica facilita la labor de mantener regeneradas todas las celdas de la DRAM. No obstante, durante la operación normal del sistema en el cual funciona una DRAM, es muy poco probable que se realice una operación de lectura en cada fila de la DRAM dentro del límite de tiempo de regeneración requerido. Por lo tanto, se necesita cierto tipo de lógica de control de regeneración, ya sea externa para el chip DRAM o como parte de sus circuitos internos. En cualquier caso hay dos modos de regeneración: en *ráfaga* y *distribuida*.

En el modo de regeneración en ráfaga se suspende la operación normal de la memoria y cada fila de la DRAM se regenera en forma sucesiva hasta que se hayan regenerado todas las filas. En el modo de regeneración distribuida, la regeneración de las filas se entremezcla con las operaciones normales de la memoria.

El método más universal para regenerar una DRAM es la **regeneración sólo de *RAS***. Para ello se aplica mediante estrobo una dirección de fila con *RAS*, mientras que *CAS* y *R/W* permanecen en ALTO. La figura 12-32 ilustra cómo se utiliza la regeneración sólo de *RAS* para una regeneración en ráfaga del TMS44100. Parte de la complejidad del arreglo de memoria en este chip está ahí para facilitar las operaciones de regeneración. Como hay dos bancos alineados en la misma fila, ambos pueden regenerarse al mismo tiempo, que en efecto es lo mismo que si sólo hubiera 1024 filas. Un **contador de regeneración** se utiliza para suministrar direcciones de fila de 10 bits a las entradas de dirección de la DRAM, empezando desde 0000000000 (fila 0). Se aplica un pulso a *RAS* para que cambie a BAJO y se cargue esta dirección en la DRAM, con lo cual se regenera la fila 0 en ambos bancos. El contador se incrementa y el proceso se repite hasta la dirección 1111111111 (fila 1023). Para el TMS44100 puede completarse una regeneración en ráfaga en casi 113 µs y debe repetirse por lo menos cada 16 ms.



* Las líneas R/W y CAS se mantienen en ALTO

FIGURA 12-32 El método de regeneración sólo de $\overline{RAS}$ utiliza únicamente la señal $\overline{RAS}$ para cargar la dirección de fila en la DRAM y regenerar todas las celdas en esa fila. La regeneración sólo de $\overline{RAS}$ puede utilizarse para realizar una regeneración en ráfaga, como se muestra. Un contador de regeneración proporciona las direcciones de fila en secuencia, desde la fila 0 hasta la fila 1023 (para una DRAM de 4M × 1).

Aunque la idea del contador de regeneración parece bastante sencilla, debemos tener en cuenta que las direcciones de fila del contador de regeneración no pueden interferir con las direcciones que vienen de la CPU durante las operaciones normales de lectura/escritura. Por esta razón, las direcciones del contador de regeneración deben multiplexarse con las direcciones de la CPU, de manera que se active la fuente apropiada de direcciones de la DRAM en los tiempos apropiados.

Para poder liberar a la CPU de la computadora de algunas de estas cargas, comúnmente se utiliza un chip especial conocido como **controlador de RAM dinámica (DRAM)**. Como mínimo, este chip realizará el multiplexaje de direcciones y la generación de secuencias de conteo de regeneración, dejando la generación de la sincronización para las señales $\overline{RAS}$, $\overline{CAS}$ y *MUX* para otros circuitos lógicos y la persona que programa la computadora. Otros controladores de DRAM son completamente automáticos. Sus entradas son muy parecidas a las de una RAM estática o una ROM. Generan en forma automática la secuencia de regeneración con la frecuencia suficiente como para mantener la memoria, multiplexar el bus de direcciones, generar las señales $\overline{RAS}$ y $\overline{CAS}$, y supervisar el control de la DRAM entre los ciclos de lectura/escritura de la CPU y las operaciones de regeneración locales. En las computadoras personales actuales, el controlador de DRAM y otros circuitos controladores de alto nivel se integran en un conjunto de circuitos VLSI, a los cuales se les conoce como "conjunto de chips". A medida que se desarrollan nuevas tecnologías de DRAM, se diseñan nuevos conjuntos de chips para aprovechar los últimos avances. En muchos casos, el número de conjuntos de chips existentes (o anticipados) que soportan cierta tecnología en el mercado es lo que determina en qué tecnología de DRAM invertirán los fabricantes.

La mayoría de los chips DRAM en producción hoy en día tienen capacidad de regeneración en el chip, con lo cual se elimina la necesidad de suministrar direcciones de regeneración externas. Uno de estos métodos, que se muestra en la figura 12-33(a), se llama *regeneración* de $\overline{CAS}$ antes que $\overline{RAS}$. En este método, la señal $\overline{CAS}$ se lleva al nivel BAJO primero y se mantiene hasta después de que $\overline{RAS}$ cambia a BAJO. Esta secuencia regenerará una fila del arreglo de memoria e incrementará un contador interno que genera las direcciones de fila. Para realizar una regeneración en ráfaga mediante esta característica, $\overline{CAS}$ puede mantenerse en BAJO mientras que se aplica un pulso a $\overline{RAS}$ para cada fila, hasta que se regeneren todas. Durante este ciclo de regeneración se ignoran todas las direcciones externas. El TMS44100 también ofrece una "regeneración oculta", la cual permite regenerar una fila mientras se retienen datos en la salida. Para ello hay que mantener a $\overline{CAS}$ en BAJO después de un ciclo de lectura y posteriormente aplica un pulso a $\overline{RAS}$, como en la figura 12-33(b).

El modo de regeneración automática de la figura 12-33(c) automatiza el proceso por completo. Al forzar a $\overline{CAS}$ para que cambie a BAJO antes que $\overline{RAS}$, y después mantener ambas señales en BAJO durante por lo menos 100 μs, un oscilador interno aplica pulsos de reloj al contador de dirección de fila hasta que se regeneran todas las celdas. El modo que seleccione un diseñador de sistemas dependerá de qué tan ocupada esté la CPU de la computadora. Si puede ceder 100 μs sin acceder a su

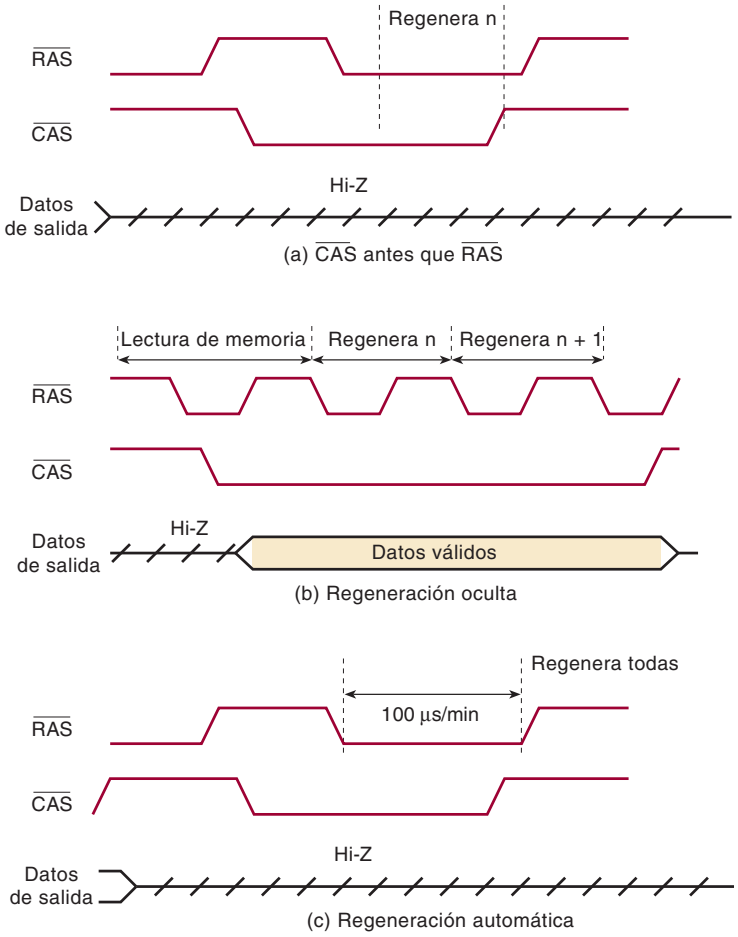


FIGURA 12-33 Modos de regeneración del TMS44100.

memoria, y si puede hacer esto cada 16 ms, lo más conveniente es la regeneración automática.

No obstante, si esto reduce demasiado la velocidad de ejecución del programa, tal vez se requiera cierto tipo de regeneración distribuida, mediante el uso de *CAS* antes que *RAS* o ciclos de regeneración oculta. En cualquier caso, todas las celdas deben regenerarse dentro del tiempo permitido o se perderán los datos.

PREGUNTAS DE REPASO

1. *Verdadero o falso:*

   (a) En la mayoría de las DRAMs, es necesario leer sólo de una celda en cada fila para poder regenerar todas las celdas en esa fila.

   (b) En el modo de regeneración en ráfaga, todo el arreglo se regenera mediante un pulso en $\overline{RAS}$.
2. ¿Cuál es la función de un contador de regeneración?
3. ¿Qué funciones realiza un controlador de DRAM?
4. *Verdadero o falso:*

   (a) En el método de regeneración sólo de $\overline{RAS}$, la señal $\overline{CAS}$ se mantiene en BAJO.

   (b) La regeneración de $\overline{CAS}$ antes que $\overline{RAS}$ sólo la pueden utilizar DRAMs con circuitos de control de generación en el chip.

12-17 TECNOLOGÍA DE LA DRAM*

Al seleccionar un tipo específico de dispositivo de RAM para un sistema, un diseñador se topa con ciertas decisiones difíciles. La capacidad (tan grande como sea posible), la velocidad (tan rápida como sea posible), la energía necesaria (lo menos que sea posible), el costo (lo más bajo que sea posible), y la conveniencia (lo más fácil de cambiar que sea posible) deben mantenerse en un balance razonable, ya que ningún tipo de RAM puede maximizar todas estas características deseadas. El mercado de la RAM semiconductora intenta constantemente producir la mezcla ideal de estas características en sus productos para diversas aplicaciones. En esta sección se explican algunos de los términos actuales que se utilizan en relación con la tecnología de la RAM. Éste es un tema muy dinámico y tal vez algunos de estos términos serán historia en el momento que lea este libro, pero he aquí lo último en la tecnología.

Módulos de memoria

Gracias a que muchas compañías fabrican tarjetas madre para sistemas de computadora personal, se han adoptado conectores de interfase de memoria estándar. Estos conectores reciben una pequeña tarjeta de circuito impreso con puntos de contacto en ambos lados del borde de la tarjeta. Estas tarjetas modulares permiten instalar o sustituir con facilidad los componentes de memoria en una computadora. El módulo de memoria simple en línea (SIMM) es una tarjeta de circuito con 73 contactos equivalentes en funcionalidad, en ambos lados de la tarjeta. Un punto de contacto redundante en cada lado de la tarjeta ofrece cierta seguridad de que se realiza un contacto confiable. Estos módulos utilizan chips DRAM sólo de 5 V, los cuales varían en capacidad desde 1 hasta 16 Mbits en encapsulados tipo ala de gaviota con montaje superficial, o de terminal en J. Los módulos de memoria varían en capacidad, de 1 a 32 Mbytes.

El módulo de memoria dual en línea (DIMM) más reciente tiene 84 contactos únicos en funcionalidad, en cada lado de la tarjeta. Las terminales adicionales son necesarias debido a que los DIMMS se conectan a buses de datos de 64 bits, como los que se utilizan en las PCs modernas. Existen versiones de 3.3 V y de 5 V. También hay versiones con búfer y sin búfer. La capacidad del módulo depende de los chips DRAM que estén montados en él; y a medida que aumente la capacidad de la DRAM, también aumentará la capacidad de los DIMMs. El conjunto de chips y el diseño de la tarjeta madre que se utilicen en un sistema dado es lo que determina el tipo de DIMM que puede utilizarse. Para aplicaciones compactas, como las de computadoras portátiles, hay un módulo de memoria dual en línea de diseño pequeño (SODIMM).

El principal problema en la industria de las computadoras personales es proporcionar un sistema de memoria que sea lo bastante rápido como para mantenerse a la par con las velocidades de reloj del microprocesador que siempre van en aumento, y mantener al mismo tiempo el costo en un nivel aceptable. Se están agregando características especiales a los dispositivos básicos de DRAM para mejorar su ancho de banda total. Acaba de introducirse al mercado un nuevo tipo de encapsulado, conocido como RIMM. RIMM significa Módulo de memoria Rambus en línea. Rambus es una compañía que ha inventado varios nuevos enfoques revolucionarios en relación con la tecnología de la memoria. El RIMM es su encapsulado propietario que almacena sus chips de memoria propietarios, llamados chips de DRAM Rambus directa (DRDRAM). Aunque estos métodos para mejorar el rendimiento cambian en forma constante, en los textos relacionados con la memoria se hace una extensa referencia a las tecnologías que describiremos en las siguientes secciones.

DRAM FPM

El modo de paginación rápida (FPM) permite un acceso más rápido a las ubicaciones de memoria aleatorias dentro de la "página" actual. En esencia, una página es un intervalo de direcciones de memoria que tiene valores idénticos en los bits de

* Puede omitir este tema sin que se vea afectada la continuidad del resto del libro.

dirección superiores. Para poder acceder a los datos en la página actual sólo hay que modificar las líneas de dirección inferiores.

DRAM EDO

Las DRAMs con salida de datos extendida (EDO) ofrecen una mejora en comparación con las DRAMs FPM. Para los accesos en una página dada, se detecta el valor de datos en la ubicación de memoria actual y se fija en las terminales de salida. En las DRAMs FPM, el amplificador sensible controla la salida sin un latch, para lo cual requiere que $\overline{CAS}$ permanezca en el nivel bajo hasta que los valores de datos se vuelvan válidos. Con EDO, mientras estos datos están presentes en las salidas, $\overline{CAS}$ puede completar su ciclo, se puede decodificar una nueva dirección en la página actual y los circuitos de la ruta de datos pueden restablecerse para el siguiente acceso. Esto permite al controlador de memoria enviar como salida la siguiente dirección al mismo tiempo que se lee la palabra actual.

SDRAM

La DRAM síncrona está diseñada para transferir datos en *ráfagas* de disparo rápido de varias ubicaciones de memoria secuenciales. La primera ubicación a la que se accede es la más lenta, debido a la sobrecarga (latencia) del proceso de fijar la dirección de fila y de columna. Después el reloj del sistema de bus aplica pulsos de reloj a los valores de datos (en vez de la línea de control $\overline{CAS}$) en ráfagas de ubicaciones de memoria dentro de la misma página. En su interior, las SDRAMs se organizan en dos bancos. Esto permite leer datos a una velocidad muy rápida, ya que se accede en forma alternativa a cada uno de los dos bancos. Para poder proveer todas las características y la flexibilidad necesaria para que este tipo de DRAM funcione con una amplia variedad de requerimientos de sistema, los circuitos dentro de la SDRAM se han vuelto más complicados. Se necesita una secuencia de instrucciones para indicar a la SDRAM qué opciones son necesarias, como la longitud de la ráfaga, los datos secuenciales o interpaginados y los modos de $\overline{CAS}$ antes que $\overline{RAS}$ o de regeneración automática. El modo de regeneración automática permite al dispositivo de memoria realizar todas las funciones necesarias para mantener sus celdas actualizadas.

DDRSDRAM

La SDRAM de doble velocidad de transferencia de datos ofrece una mejora en comparación con la SDRAM. Para poder agilizar la operación de la SDRAM mientras se opera con base en un reloj de sistema síncrono, esta tecnología transfiere datos en los flancos positivo y negativo del reloj del sistema, duplicando con efectividad la velocidad potencial de la transferencia de datos.

SLDRAM

La DRAM de enlace síncrono es una mejora evolutiva en comparación con la DDR-SDRAM. Puede operar a velocidades de bus de hasta 200 MHz y aplica pulsos de reloj a los datos en forma síncrona, en los flancos positivo y negativo del reloj del sistema. Un consorcio de varios fabricantes de DRAM lo está desarrollando como un estándar abierto. Si se desarrollan conjuntos de chips que puedan aprovechar estos dispositivos de memoria y suficientes diseñadores de sistemas adoptan esta tecnología, es muy probable que se convierta en una forma muy utilizada de DRAM.

DRDRAM

La DRAM Rambus directa es un dispositivo propietario, fabricado y comercializado por Rambus, Inc. Utiliza un nuevo enfoque revolucionario de la arquitectura del sistema de DRAM, con un control más intenso, integrado en el dispositivo de memoria. Esta tecnología aún se encuentra luchando con los demás estándares para encontrar su nicho en el mercado.

PREGUNTAS DE REPASO

1. ¿Pueden intercambiarse los SIMMs y los DIMMs?
2. ¿Qué es una "página" de memoria?
3. ¿Por qué el "modo de paginación" es más rápido?
4. ¿Qué significa *EDO*?
5. ¿Qué término se utiliza para acceder a varias ubicaciones de memoria consecutivas?
6. ¿Con qué se sincroniza una DRAM?

12-18 EXPANSIÓN DE TAMAÑO DE PALABRA Y CAPACIDAD

En muchas aplicaciones de memoria no se puede satisfacer la capacidad o el tamaño de palabra requeridos para una memoria RAM o ROM con un solo chip de memoria. Hay que combinar varios chips de memoria para proporcionar la capacidad y/o el tamaño de palabra. En esta sección veremos cómo se hace esto a través de varios ejemplos en los que se ilustran las ideas importantes que se utilizan cuando se integran chips de memoria con un microprocesador. Los siguientes ejemplos están diseñados para ser instructivos, y los tamaños de los chips de memoria utilizados se eligieron de manera que se conservara espacio. Las técnicas que vamos a presentar pueden extenderse a chips de memoria más grandes.

Expansión del tamaño de palabra

Suponga que necesitamos una memoria que pueda almacenar 16 palabras de ocho bits y todo lo que tenemos son chips de RAM ordenados como memorias de 16 $\times$ 4, con líneas comunes de E/S. Podemos combinar dos de estos chips de 16 $\times$ 4 para producir la memoria deseada. La configuración para hacerlo se muestra en la figura 12-34. Examine este diagrama con cuidado y vea lo que puede averiguar de él antes de seguir leyendo.

Como cada chip puede almacenar 16 palabras de cuatro bits y queremos almacenar 16 palabras de 8 bits, vamos a utilizar cada chip para que almacene la *mitad* de cada palabra. Es decir, la RAM-0 almacena los cuatro bits de *mayor* orden de cada una de las 16 palabras y la RAM-1 almacena los cuatro bits de *menor* orden de cada una de las 16 palabras. De esta manera hay una palabra de ocho bits completa en las salidas de la RAM que se conectan al bus de datos.

Para seleccionar cualquiera de las 16 palabras, se aplica el código de dirección apropiado al *bus de direcciones* de cuatro líneas (A_3, A_2, A_1, A_0). Por lo general, las líneas de dirección se originan en la CPU. Observe que cada línea del bus de direcciones está conectada a la entrada de dirección correspondiente de cada chip. Esto significa que, una vez que se coloca un código de dirección en el bus de direcciones, este mismo código de dirección se aplica a los dos chips, de manera que pueda accederse a la misma ubicación en cada chip al mismo tiempo.

Una vez seleccionada la dirección, podemos leer o escribir en ella bajo el control de la línea común $R/\overline{W}$ y $\overline{CS}$. Para leer, $R/\overline{W}$ debe estar en alto y $\overline{CS}$ debe estar en bajo. Esto hace que las líneas de E/S de la RAM actúen como *salidas*. La RAM-0 coloca su palabra de cuatro bits seleccionada en las cuatro líneas superiores del bus de datos y la RAM-1 coloca su palabra de cuatro bits seleccionada en las cuatro líneas inferiores del bus de datos. Así, el bus de datos contiene la palabra de ocho bits completa que se seleccionó, la cual puede ahora transmitirse hacia algún otro dispositivo (por lo general, hacia un registro en la CPU).

Para escribir, la condición $R/\overline{W} = 0$ y $\overline{CS} = 0$ hace que las líneas de E/S de la RAM actúen como *entradas*. La palabra de ocho bits que se va a escribir se coloca en el bus de datos (por lo general, la CPU se encarga de ello). Los cuatro bits superiores se escribirán en la ubicación seleccionada de la RAM-0, y los cuatro bits inferiores se escribirán en la RAM-1.

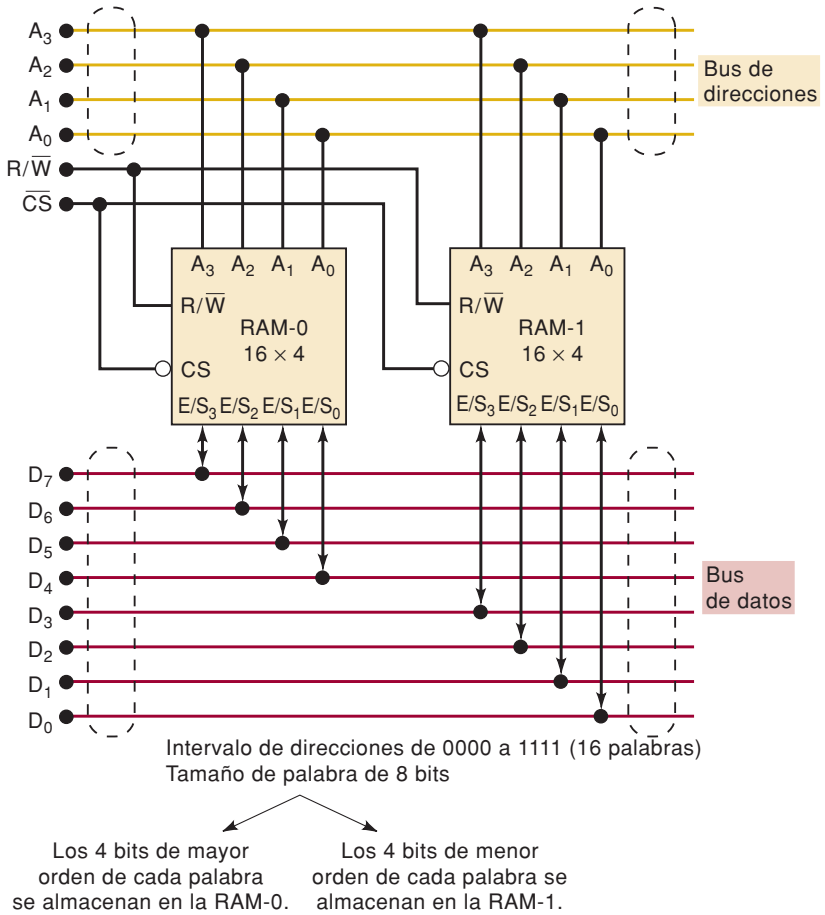


FIGURA 12-34 Combinación de dos RAMs de 16 × 4 para obtener un módulo de 16 × 8.

En esencia, la combinación de los dos chips de RAM actúa como un chip de memoria individual de 16 × 8. A esta combinación se le denomina *módulo de memoria* de 16 × 8.

La misma idea básica para expandir el tamaño de palabra funcionará en distintas situaciones. Lea el siguiente ejemplo y dibuje un diagrama sencillo que ilustre la apariencia del sistema antes de ver la solución.

EJEMPLO 12-13

El 2125A es un CI de RAM estática que tiene una capacidad de 1K × 1, una entrada de selección de chip activa en BAJO y líneas separadas de entrada y salida de datos. Muestre cómo se pueden combinar varios CIs 2125A para formar un módulo de 1K × 8.

Solución

El arreglo se muestra en la figura 12-35, en donde se utilizan ocho chips 2125A para un módulo de 1K × 8. Cada chip almacena uno de los bits de cada una de las 1024 palabras de ocho bits. Observe que todas las entradas $R/\overline{W}$ y $\overline{CS}$ están conectadas entre sí, y que el bus de direcciones de 10 líneas está conectado a las entradas de dirección de cada chip. Observe además que como el 2125A tiene terminales separadas de entrada y salida de datos, ambas terminales de cada chip están conectadas a la misma línea del bus de datos.

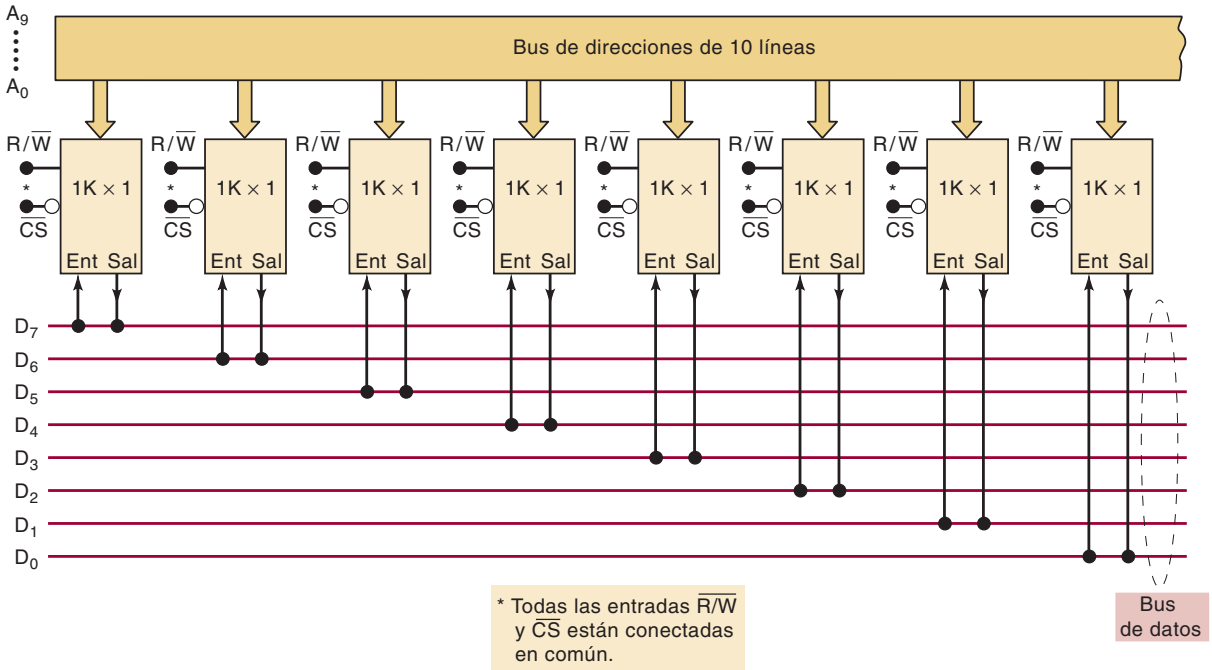


FIGURA 12-35 Ocho chips 2125A de 1K × 1 que forman un arreglo de memoria de 1K × 8.

Expansión de la capacidad

Suponga que necesitamos una memoria que pueda almacenar 32 palabras de cuatro bits y que todo lo que tenemos son los chips de 16 × 4. Si combinamos dos chips de 16 × 4 como se muestra en la figura 12-36, podremos producir la memoria deseada. Una vez más, examine este diagrama y vea qué puede determinar de él antes de seguir leyendo.

Cada RAM se utiliza para almacenar 16 palabras de cuatro bits. Las cuatro terminales de E/S de datos de cada RAM se conectan a un bus de datos común de cuatro líneas. Sólo uno de los chips de RAM puede seleccionarse (habilitarse) en un momento dado, de manera que no haya problemas de colisión de bus. Para asegurar esto se controlan las respectivas entradas $\overline{CS}$ mediante distintas señales lógicas.

La capacidad total de este módulo de memoria es de 32 × 4, por lo que debe haber 32 direcciones distintas. Para esto se requieren *cinco* líneas en el bus de direcciones. La línea de dirección superior A_4 se utiliza para seleccionar una u otra RAM (a través de las entradas $\overline{CS}$), para leer o escribir información en ella. Las otras cuatro líneas de dirección (A_0-A_3) se utilizan para seleccionar una de las 16 ubicaciones de memoria del chip de RAM seleccionado.

Para ilustrar esto, cuando $A_4 = 0$ la señal $\overline{CS}$ de la RAM-0 habilita este chip para lectura o escritura. Después se puede acceder a cualquier ubicación en la RAM-0 a través de las líneas A_3-A_0. Las últimas cuatro líneas de dirección pueden variar de 0000 a 1111 para seleccionar la ubicación deseada. Así, el intervalo de direcciones que representan ubicaciones en la RAM-0 es

$$A_4A_3A_2A_1A_0 = 00000 \text{ a } 01111$$

Observe que cuando $A_4 = 0$ la señal $\overline{CS}$ de la RAM-1 está en alto, por lo cual sus líneas de E/S están deshabilitadas (Hi-Z) y no pueden comunicarse con (dar o recibir datos) el bus de datos.

Hay que dejar en claro que cuando $A_4 = 1$ se invierten los roles de la RAM-0 y la RAM-1. Ahora la RAM-1 está habilitada y las líneas A_3-A_0 seleccionan una de sus ubicaciones. Así, el intervalo de direcciones ubicadas en la RAM-1 es

$$A_4A_3A_2A_1A_0 = 10000 \text{ a } 11111$$

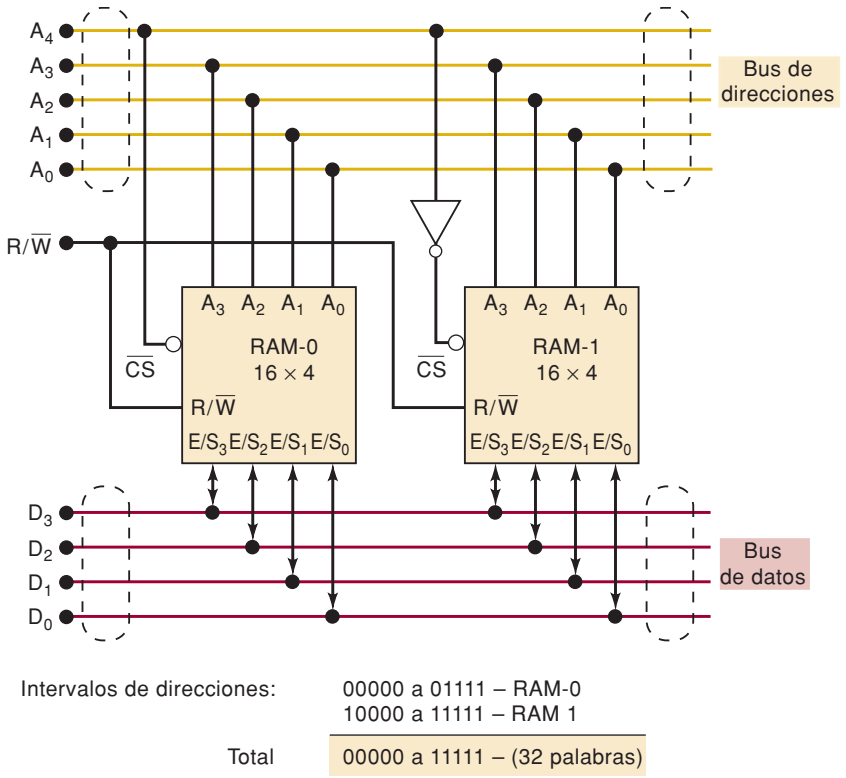


FIGURA 12-36
Combinación de dos chips de 16 × 4 para formar una memoria de 32 × 4.

EJEMPLO 12-14

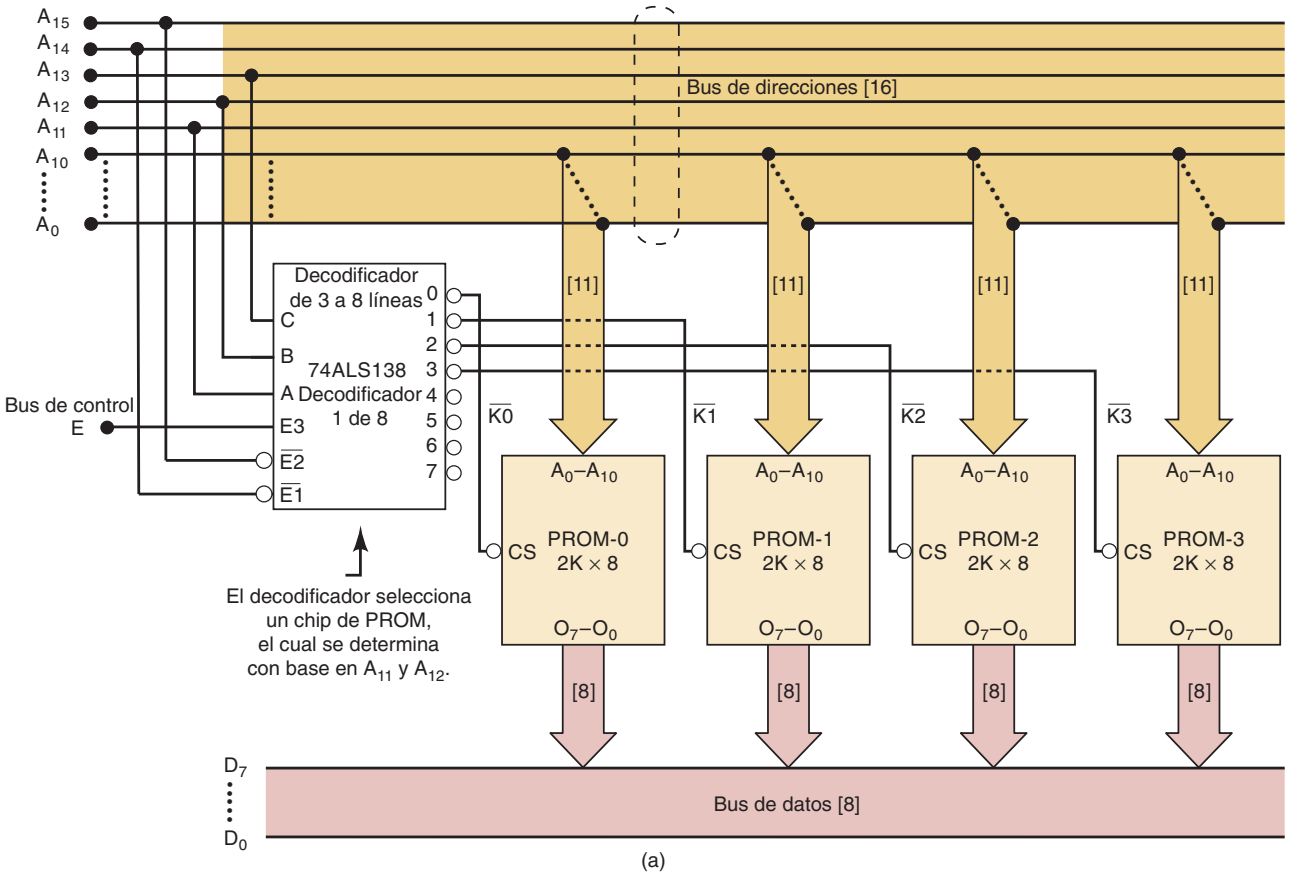
Queremos combinar varias PROMs de 2K × 8 para producir una capacidad total de 8K × 8. ¿Cuántos chips de PROM se necesitan? ¿Cuántas líneas se requieren para el bus de direcciones?

Solución

Se requieren cuatro chips de PROM, cada uno con capacidad de almacenar 2K de las 8K palabras. Como 8K = 8 × 1024 = 8192 = 2^{13}, se requieren *trece* líneas de dirección.

La configuración para la memoria del ejemplo 12-14 es similar a la memoria de 32 × 4 de la figura 12-36. No obstante, es un poco más compleja debido a que requiere un circuito decodificador para generar las señales de entrada $\overline{CS}$. En la figura 12-37(a) se muestra el diagrama completo para esta memoria de 8192 × 8.

La capacidad total del bloque de ROM es de 8192 bytes. Este sistema que contiene el bloque de memoria tiene un bus de direcciones de 16 bits, algo común en un sistema pequeño basado en microcontrolador. El decodificador en este sistema sólo puede habilitarse cuando A_{15} y A_{14} estén en BAJO, y que E esté en ALTO. Esto significa que sólo puede decodificar direcciones menores del valor 4000 hexadecimal (4000H). Es más fácil comprender esto si analizamos el mapa de memoria de la figura 12-37(b). Ahí podemos ver que los dos MSBs superiores (A_{15} y A_{14}) están siempre en BAJO para las direcciones menores que 4000H. Las líneas de dirección A_{13}-A_{11} se conectan a las entradas *C*-*A* del decodificador, respectivamente. Estos tres bits se decodifican y se utilizan para seleccionar uno de los CIs de memoria. Observe en el mapa de bits de la figura 12-37(b) que todas las direcciones dentro de la PROM-0 tienen A_{13}, A_{12}, A_{11} = 0, 0, 0; la PROM-1 se selecciona cuando estos bits tienen un valor de 0, 0, 1; la PROM-2 cuando son 0, 1, 0; y la PROM-3 cuando son 0, 1, 1. Cuando se selecciona cualquier PROM, las líneas de dirección A_{10}-A_0 pueden variar desde sólo 0s hasta sólo 1s. Para sintetizar el esquema de direcciones de este sistema, se



(a)

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Dirección	Mapa del sistema	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000	PROM-0	2K
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	07FF		
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0800	PROM-1	2K
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0FFF		
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1000	PROM-2	2K
0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	17FF		
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1800	PROM-3	2K
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFF		
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000	O_4	
		1	0	1													O_5 Decodificada	8K
		1	1	0													O_6 Expansión	
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFF	O_7	
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000	Disponible	48K
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFF		

(b)

FIGURA 12-37 (a) Cuatro PROMs de 2K × 8 en un arreglo para formar una capacidad total de 8K × 8. (b) Mapa de memoria del sistema completo.

utilizan los dos bits superiores para seleccionar este decodificador, se utilizan los siguientes tres bits (A_{13}-A_{11}) para seleccionar uno de los cuatro chips de PROM y se utilizan las 11 líneas de dirección inferiores para seleccionar una de las 2048 ubicaciones de memoria del tamaño de un byte en la PROM habilitada.

Cuando hay una dirección del sistema de 4000H o mayor en el bus de direcciones, ninguna de las PROMs se habilitará. No obstante, pueden utilizarse las salidas 4-7 del decodificador para habilitar más chips de memoria si deseamos expandir la capacidad del sistema de memoria. El mapa de memoria a la derecha de la figura 12-37(b) muestra un área de 48K del espacio del sistema que no está ocupada por este bloque de memoria. Para poder expandirse a esta área del mapa de memoria, se requeriría más lógica de decodificación.

EJEMPLO 12-15

¿Qué se necesitaría para expandir la memoria de la figura 12-37 a 32K × 8? Describa qué líneas de dirección se utilizan.

Solución

Una capacidad de 32K requiere 16 de los chips PROM de 2K. Ya se muestran cuatro de ellos y pueden conectarse cuatro más a las salidas O_4-O_7 del decodificador. Esto cubre la mitad del sistema. Para seleccionar los otros ocho chips PROM hay que agregar otro decodificador 74ALS138 y habilitarlo sólo cuando $A_{15} = 0$ y $A_{14} = 1$. Para lograrlo se conecta un inversor entre A_{14} y $\overline{E}_1$ mientras se conecta A_{15} en forma directa con E_2. Las otras conexiones son iguales que en el decodificador existente.

Decodificación incompleta de direcciones

En muchos casos es necesario utilizar varios dispositivos de memoria en el mismo sistema de memoria. Por ejemplo, considere los requerimientos de un sistema de tablero de controles digital en un automóvil. Éste se implementa, por lo general, mediante el uso de un microprocesador. En consecuencia, necesitamos cierta ROM no volátil para almacenar las instrucciones del programa. Necesitamos cierta memoria de lectura/escritura para almacenar los dígitos que representan la velocidad, RPM, galones de combustible, etcétera. Hay que almacenar otros valores digitalizados para representar la presión del aceite, la temperatura del motor, voltaje de la batería, etcétera. También necesitamos cierto almacenamiento de lectura/escritura no volátil (EEPROM) para la lectura del odómetro, ya que no sería conveniente que este número se restableciera a 0 o que asumiera un valor aleatorio cada vez que se desconectara la batería del automóvil.

La figura 12-38 muestra un sistema de memoria que podría utilizarse en un sistema de microcomputadora. Observe que la parte correspondiente a la ROM está compuesta de dos dispositivos de 8K × 8 (PROM-0 y PROM-1). La sección de RAM requiere un solo dispositivo de 8K × 8. La EEPROM disponible es sólo un dispositivo de 2K × 8. El sistema de memoria requiere un decodificador para seleccionar sólo un dispositivo a la vez. Este decodificador divide todo el espacio de memoria (suponiendo 16 bits de dirección) en bloques de direcciones de 8K. En otras palabras, la salida de cada decodificador se activa mediante 8192 (8K) direcciones distintas. Observe que las tres líneas de dirección superiores controlan el decodificador. Las 13 líneas de dirección de menor orden se conectan en forma directa a las entradas de dirección en los chips de memoria. La única excepción a esto es la EEPROM, la cual tiene sólo 11 líneas de dirección para su capacidad de 2 Kbytes. Si el intervalo de la dirección (en hexadecimal) de esta EEPROM debe estar entre 6000 y 67FF, responderá a estas direcciones de la manera esperada. No obstante, las dos líneas de dirección A_{11} y A_{12} no están involucradas en el esquema de decodificación para este chip. La salida del decodificador ($K3$) está activa para direcciones de 8K, pero el chip al que está conectado contiene sólo 2K ubicaciones. Como resultado, la EEPROM también responderá a los otros 6K de direcciones en este bloque decodificado de

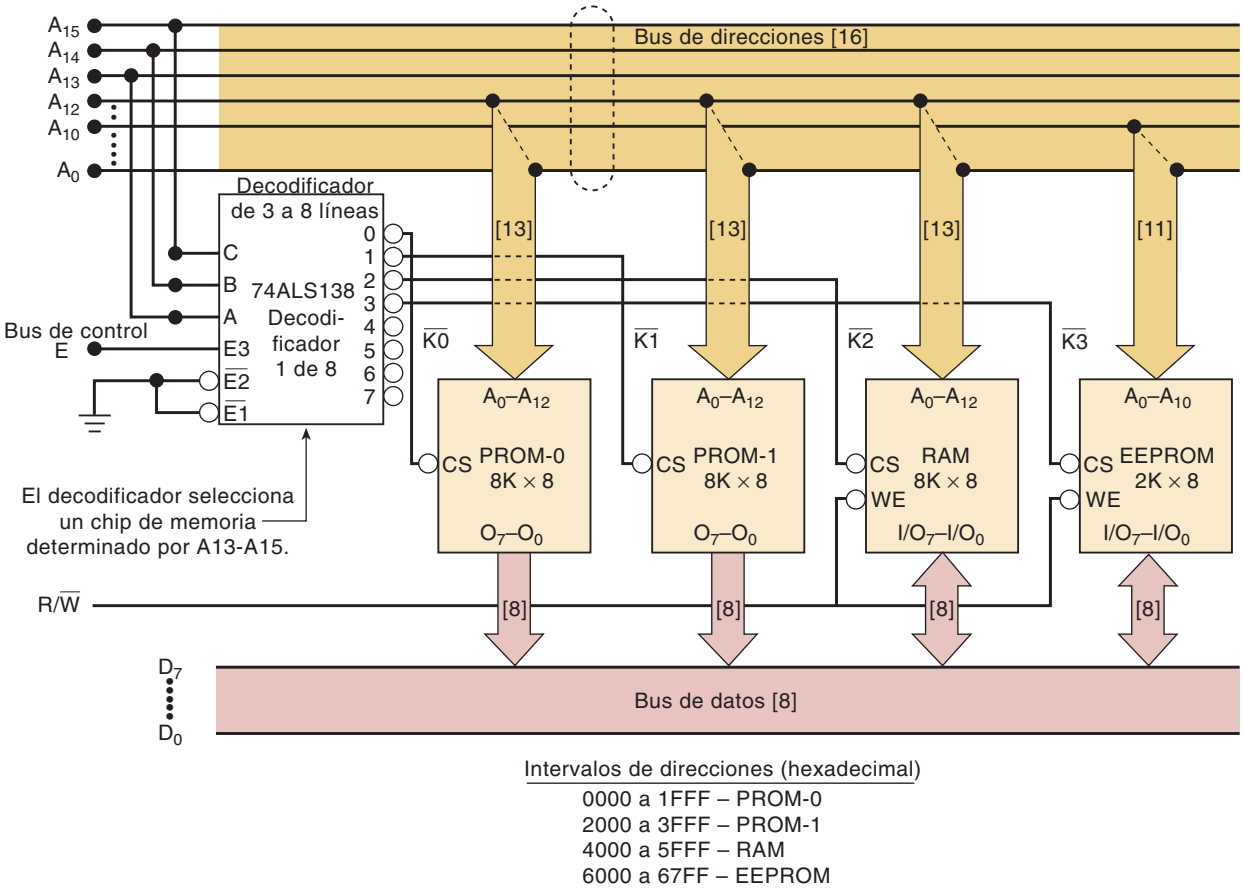


FIGURA 12-38 Un sistema con decodificación incompleta de direcciones.

memoria. El mismo contenido de la EEPROM aparecerá también en las direcciones 6800-6FFF, 7000-77FF y 7800-7FFF. Estas áreas de memoria que se ocupan de manera redundante por un dispositivo debido a la decodificación incompleta de direcciones se conocen como áreas de **empalme de memoria**. Esto ocurre con frecuencia en sistemas en donde hay una abundancia de espacio de direcciones y existe la necesidad de minimizar la lógica de decodificación. Un **mapa de memoria** de este sistema

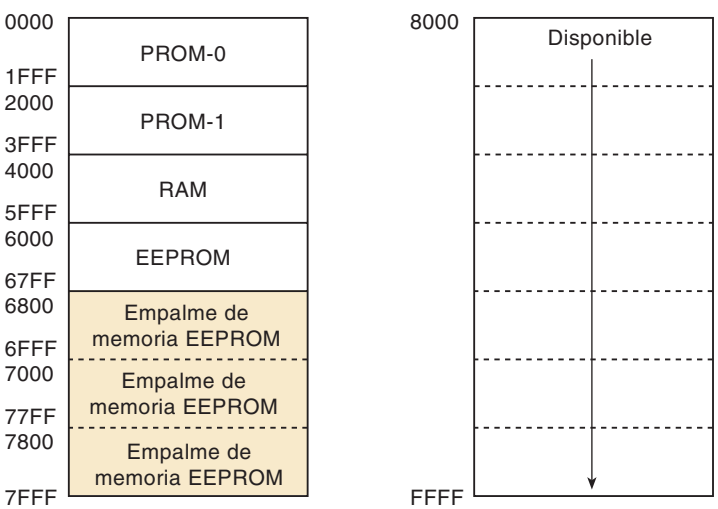


FIGURA 12-39 Un mapa de memoria de un sistema de tablero de controles digital.

(vea la figura 12-39) muestra con claridad las direcciones a las que está asignado cada dispositivo, así como el espacio de memoria disponible para expansión.

Combinación de chips de DRAM

Por lo general, los CIs de DRAM tienen tamaños de palabra de uno o cuatro bits, por lo que es necesario combinar varios de ellos para formar módulos con tamaños de palabra más grandes. La figura 12-40 muestra cómo combinar ocho chips de DRAM TSM44100 para formar un módulo de 4M × 8. Cada chip tiene una capacidad de 4M × 1.

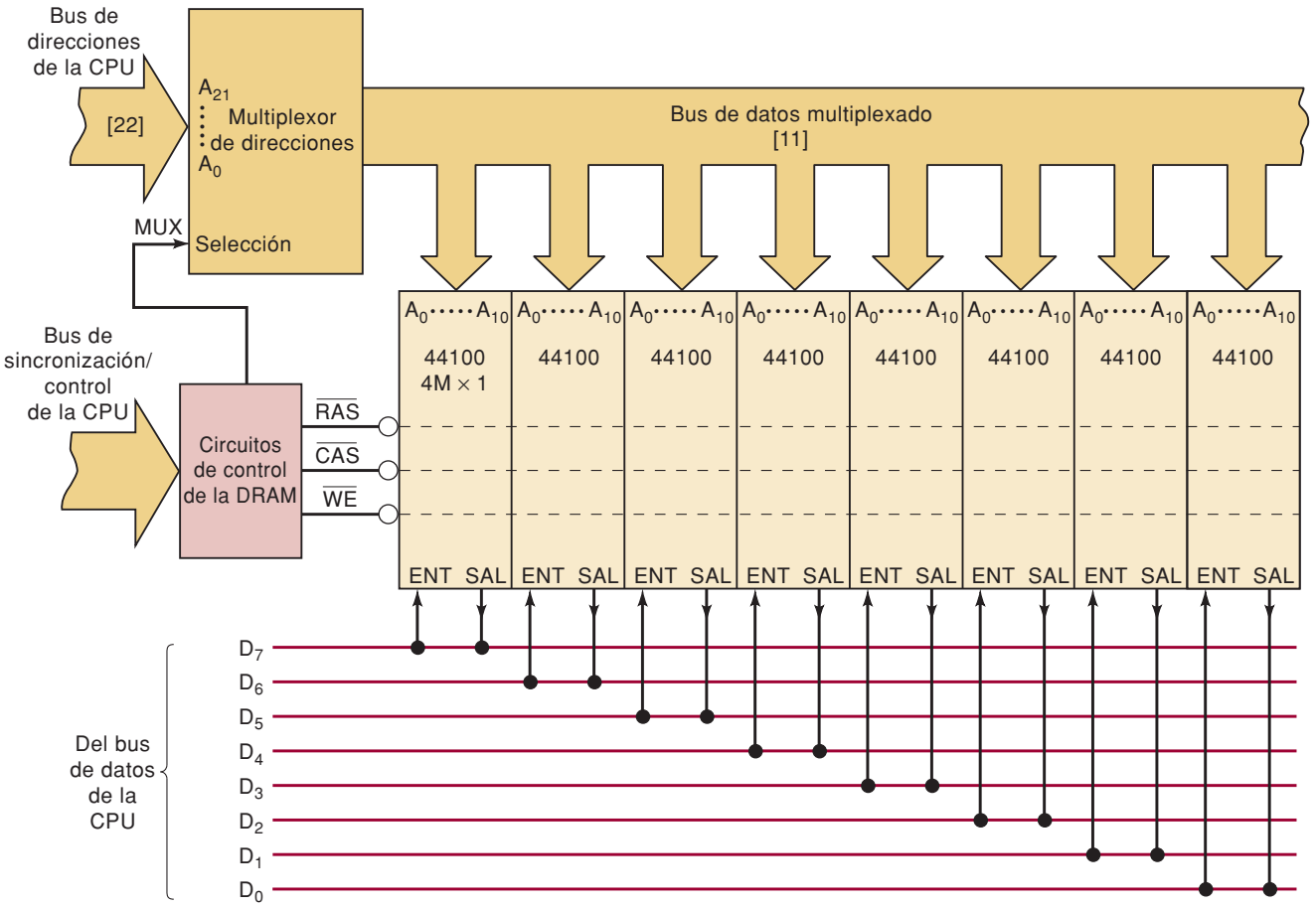


FIGURA 12-40 Ocho chips de DRAM de 4M × 1 combinados para formar un módulo de memoria de 4M × 8.

Hay varios puntos importantes que observar. En primer lugar, como $4M = 2^{22}$ el chip TMS44100 tiene *once* entradas de dirección; recuerde que las DRAMs utilizan entradas de dirección multiplexadas. El multiplexor de direcciones recibe el bus de direcciones de 22 líneas de la CPU y lo cambia por un bus de direcciones de 11 líneas para los chips de DRAM. En segundo lugar, las entradas $\overline{RAS}$, $\overline{CAS}$ y $\overline{WE}$ de los ocho chips se conectan entre sí para que todos los chips se activen de manera simultánea para cada operación de memoria. Por último, recuerde que el TMS44100 cuenta con circuitos de control de regeneración integrados en el chip, por lo que no hay necesidad de un contador de regeneración externo.

PREGUNTAS DE REPASO

1. El MCM6209C es un chip de RAM estática de 64K × 4. ¿Cuántos de estos chips se necesitan para formar un módulo de 1M × 4?
2. ¿Cuántos se necesitan para un módulo de 64K × 16?
3. *Verdadero o falso:* cuando se combinan chips de memoria para formar un módulo con un tamaño de palabra o capacidad más grande, las entradas *CS* de cada chip se conectan siempre juntas.
4. *Verdadero o falso:* cuando se combinan chips de memoria para obtener una capacidad mayor, cada chip se conecta a las mismas líneas del bus de datos.

12-19 FUNCIONES ESPECIALES DE LA MEMORIA

Hemos visto que los dispositivos de RAM y ROM se utilizan como una memoria de computadora interna de alta velocidad, la cual se comunica en forma directa con la CPU (por ejemplo, el microprocesador). En esta sección describiremos con brevedad algunas de las funciones especiales que realizan los dispositivos de memoria semiconductora en las computadoras y en otros equipos y sistemas digitales. Nuestra intención no es proporcionar detalles acerca de cómo se implementan estas funciones, sino introducir las ideas básicas.

Almacenamiento cuando falla la energía

En muchas aplicaciones, la volatilidad de la RAM semiconductora puede indicar la pérdida de datos importantes cuando se desconecta la energía del sistema, ya sea a propósito o como resultado de una interrupción no planeada de energía. A continuación se muestran dos de los muchos ejemplos:

1. Los parámetros críticos de operación para las terminales de gráficos, terminales inteligentes e impresoras. Estos parámetros modificables determinan los modos de operación y los atributos que entrarán en efecto al encender el sistema.
2. Los sistemas de control de procesos industriales que nunca deben "perder su lugar" a mitad de una tarea cuando falla la energía en forma inesperada.

Hay varios métodos para proporcionar el almacenamiento de datos críticos en situaciones de fallo de energía. En uno de estos métodos, todos los datos críticos durante la operación normal del sistema se almacenan en RAM que puede operar a través de baterías de respaldo cuando se interrumpe la energía. Algunos chips de RAM tipo CMOS tienen requerimientos muy bajos de energía en modo suspendido (hasta de 0.5 mW) y se adaptan muy bien para este tipo de tarea. Algunas SRAMs tipo CMOS incluyen una pequeña batería de litio en el chip. Desde luego que aunque tengan un bajo consumo de energía, estas RAMs CMOS drenarán las baterías en un momento dado si se desconecta la energía durante periodos prolongados, y se perderán los datos.

En otro de los métodos se almacenan todos los datos críticos del sistema en memoria Flash no volátil. Este método tiene la ventaja de que no requiere energía mediante batería de respaldo, por lo cual no presenta riesgo de pérdida de datos incluso para fallas de energía de larga duración. Sin embargo, la memoria Flash no puede modificar sus datos con la misma facilidad que la RAM estática. Recuerde que con un chip Flash no podemos borrar y escribir en uno o dos bytes, se debe borrar todo un sector a la vez. Para ello la CPU tiene que reescribir un bloque grande de datos, aún y cuando sólo haya que modificar unos cuantos bytes.

En un tercer método, la CPU almacena todos sus datos en RAM volátil de alta velocidad durante la operación normal del sistema. Cuando se corta la energía, la CPU ejecuta un programa corto para los cortes de energía (desde la ROM), el cual transfiere los datos críticos de la RAM del sistema hacia RAM CMOS con batería de

respaldo o hacia memoria Flash no volátil. Para ello se requiere un circuito especial que detecta la llegada de una interrupción de energía y envía una señal a la CPU para indicarle que comience a ejecutar la secuencia de apagado.

En cualquier caso, cuando regresa la energía la CPU ejecuta un programa de encendido (desde la ROM), el cual transfiere los datos críticos de la memoria de almacenamiento de respaldo hacia la RAM del sistema, de manera que éste puede continuar su operación desde donde se había quedado cuando se interrumpió la energía.

Memoria caché

Las computadoras y demás sistemas digitales pueden tener miles o millones de bytes de memoria interna (RAM y ROM) para almacenar programas y datos que necesita la CPU durante su operación normal. Por lo general, se requiere que toda la memoria interna tenga una velocidad de operación comparable con la de la CPU, para poder lograr la máxima operación del sistema. En muchos sistemas no es económico utilizar dispositivos de memoria de alta velocidad para toda la memoria interna. En vez de ello, los diseñadores utilizan un bloque de memoria **caché** de alta velocidad. Este bloque de memoria caché es el único bloque que se comunica en forma directa con la CPU en alta velocidad; las instrucciones del programa y los datos se transfieren desde la memoria interna que es más lenta y económica, hacia la memoria caché cuando lo requiere la CPU. El éxito de la memoria caché depende de muchos factores complejos, por lo que algunos sistemas no se benefician del uso de la memoria caché.

Las CPUs de las PCs modernas tienen una pequeña caché de memoria interna (8-64 Kbytes) a la cual se le conoce como caché de nivel 1, o L1. El conjunto de chips de la mayoría de los sistemas computacionales también controla un banco externo de RAM estática (SRAM), el cual implementa una caché de nivel 2, o L2 (de 64 Kbytes a 2 Mbytes). La memoria caché se llena con una secuencia de palabras de instrucciones de la memoria del sistema. La CPU (muchas operan a velocidades de reloj de más de 2 GHz) puede acceder al contenido de la caché a una velocidad muy alta. No obstante, cuando la CPU necesita una pieza de información que no se encuentra en ese momento en la caché L1 o L2 (es decir, una omisión en la caché), debe pasar a la DRAM de sistema más lenta para obtenerla. Esta transferencia debe realizarse a la *velocidad de reloj de bus* mucho más lenta, la cual puede ser de 66 MHz a 800 MHz, dependiendo del sistema. Además de la velocidad de reloj más lenta, el tiempo de acceso de la DRAM (latencia) es mucho mayor.

La especificación de 7-2-2-2 o 5-1-1-1 para un sistema de memoria se refiere al número de ciclos de reloj de *bus* necesarios para transferir una ráfaga de cuatro palabras de 64 bits de la DRAM hacia la caché L2. El primer acceso es el que tarda más tiempo debido a la latencia asociada con los ciclos de *RAS*/*CAS*. Los datos subsiguientes se envían mediante pulsos de reloj en una ráfaga que tarda mucho menos tiempo. Por ejemplo, el sistema 7-2-2-2 requeriría 7 ciclos de reloj para obtener la primera palabra de 64 bits, y cada una de las siguientes tres palabras de 64 bits requerirían 2 ciclos cada una. En total se requieren 13 ciclos de reloj para obtener las cuatro palabras de la memoria.

Memoria tipo Primero en entrar, primero en salir (FIFO)

En los sistemas de memoria **FIFO**, los datos que se escriben en el área de almacenamiento de la RAM se leen en el mismo orden en el que se escribieron. En otras palabras, la primera palabra que se escribe en el bloque de memoria es la primera palabra que se lee del bloque de memoria: de aquí que se utilice el nombre FIFO. Esta idea se ilustra en la figura 12-41.

La figura 12-41(a) muestra la secuencia al escribir tres bytes de datos en el bloque de memoria. Observe que a medida que se escribe cada nuevo byte en la ubicación 1, los otros bytes avanzan hacia la siguiente ubicación. La figura 12-41(b) muestra la secuencia al leer los datos del bloque FIFO. El primer byte que se lee es el mismo byte que se escribió primero, y así en lo sucesivo. La operación FIFO se controla mediante *registros apuntadores de direcciones* especiales, los cuales llevan el registro de la ubicación en la que se van a escribir los datos y desde dónde se van a leer.

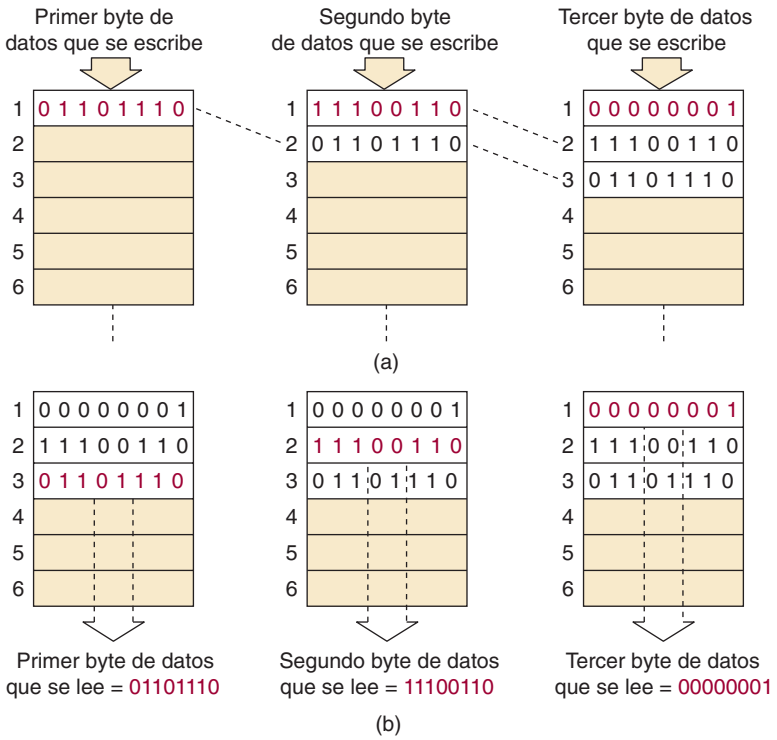


FIGURA 12-41 En FIFO, los valores de los datos se leen de la memoria (b) en el mismo orden en el que se escribieron en la memoria (a).

Una FIFO es útil como **búfer de velocidad (transferencia) de datos** entre sistemas que transfieren datos a muchas velocidades distintas. Un ejemplo es la transferencia de datos de una computadora a una impresora. La computadora envía datos tipo carácter a la impresora, a una velocidad muy alta, por decir un byte cada 10 µs. Estos datos llenan una memoria FIFO en la impresora, la cual después lee los datos de la FIFO a una velocidad mucho menor, por decir un byte cada 5 ms, e imprime los caracteres correspondientes en el mismo orden que los envió la computadora.

Una FIFO también puede utilizarse como búfer de transferencia de datos entre un dispositivo lento, como un teclado, y una computadora de alta velocidad. Aquí la FIFO acepta datos del teclado a una velocidad lenta y asíncrona de los dedos humanos y los almacena. Después la computadora puede leer con mucha rapidez todas las pulsaciones de tecla recientes que estén almacenadas, en un punto conveniente en su programa. De esta manera, la computadora puede realizar otras tareas mientras la FIFO se llena lentamente de datos.

Búferes circulares

Con frecuencia, a los búferes de transferencia de datos (FIFOs) se les denomina **búferes lineales**. En el momento en el que se llenan todas las ubicaciones en el búfer, no se introducen más datos sino hasta que se vacía el búfer. De esta manera no se pierde la información "antigua". Hay un sistema de memoria similar que se conoce como **búfer circular**. Estos sistemas de memoria se utilizan para almacenar los últimos n valores introducidos, en donde n es el número de ubicaciones de memoria en el búfer. Cada vez que se escribe un nuevo valor en un búfer circular, sobrescribe (sustituye) el valor más antiguo. Los búferes circulares se direccionan mediante un contador de direcciones MOD-n. En consecuencia, cuando se llegue a la dirección más alta el contador "dará la vuelta" y la siguiente ubicación será la dirección más baja. Como podrá recordar del capítulo 11, el filtrado digital y otras operaciones de DSP realizan cálculos mediante el uso de un grupo de muestras recientes. El hardware especial que se incluye en un DSP permite implementar con facilidad los búferes circulares en la memoria.

PREGUNTAS DE REPASO

1. ¿Cuáles son las diversas formas de manejar la posible pérdida de datos críticos cuando se interrumpe la energía?
2. ¿Cuál es la razón principal de utilizar una memoria caché?
3. ¿Qué significa *FIFO*?
4. ¿Qué es un búfer de transferencia de datos?
5. ¿Qué diferencia hay entre un búfer circular y un búfer lineal?

12-20 DIAGNÓSTICO DE FALLAS EN SISTEMAS DE RAM

Todas las computadoras utilizan RAM. Muchas computadoras de propósito general y la mayoría de las computadoras de propósito especial (como los controladores basados en microprocesador y las computadoras de control de procesos) utilizan también alguna forma de ROM. Cada CI de RAM y de ROM que forma parte de la memoria interna de una computadora, por lo general, contiene miles de celdas de memoria. Una sola celda de memoria defectuosa puede ocasionar una falla completa del sistema (lo que se conoce comúnmente como "caída del sistema") o cuando menos que el sistema no opere en forma confiable. Los procedimientos de prueba y diagnóstico de fallas en los sistemas de memoria implican el uso de técnicas que no se utilizan con frecuencia en otras partes del sistema digital. Como la memoria consiste de miles de circuitos idénticos que actúan como ubicaciones de almacenamiento, cualquier prueba en cuanto a su operación debe involucrar el proceso de comprobar cuáles ubicaciones exactas están funcionando y cuáles no. Después, si analizamos el patrón de ubicaciones buenas y malas junto con la organización del circuito de memoria, podemos determinar las posibles causas de la falla de la memoria. Por lo general, el problema puede rastrearse hasta llegar a un CI de memoria defectuoso; un CI decodificador defectuoso, compuerta lógica o búfer de señal; o un problema en las conexiones del circuito (es decir, cortos o conexiones abiertas).

Debido a que se debe escribir y leer en la RAM, las pruebas son en general más complejas que las de la ROM. En esta sección veremos algunos procedimientos comunes para realizar pruebas con la porción de RAM de la memoria e interpretar los resultados de la prueba. En la siguiente sección examinaremos las pruebas para la ROM.

Conozca la operación

En nuestros ejemplos utilizaremos el sistema de memoria RAM que muestra la figura 12-42. Como enfatizamos en discusiones anteriores, el diagnóstico de fallas exitoso de un circuito o sistema relativamente complejo comienza con un profundo conocimiento de su operación. Antes de poder hablar sobre cómo probar este sistema de RAM, debemos primero analizarlo con cuidado para poder comprender su operación por completo.

La capacidad total es de 4K $\times$ 8 y está compuesta de cuatro módulos RAM de 1K $\times$ 8. Un módulo puede ser un solo CI o puede consistir de varios CIs (por ejemplo, dos chips de 1K $\times$ 4). Cada módulo se conecta a la CPU mediante los buses de direcciones y de datos, y a través de la línea de control $R/\overline{W}$. Los módulos tienen líneas de datos de E/S comunes. Durante una operación de lectura, estas líneas se convierten en líneas de salida de datos a través de las cuales el módulo seleccionado coloca sus datos en el bus para que la CPU los lea. Durante una operación de escritura, estas líneas actúan como líneas de entrada para que la memoria acepte los datos generados por la CPU a través del bus de datos, para escribirlos en la ubicación seleccionada.

El decodificador 74ALS138 y la compuerta OR de cuatro entradas se combinan para decodificar las seis líneas de dirección de mayor orden para generar las señales de selección de chip $\overline{K0}$, $\overline{K1}$, $\overline{K2}$ y $\overline{K3}$. Estas señales habilitan un módulo específico de RAM para una operación de lectura o de escritura. El INVERSOR se utiliza para invertir la señal de Habilitación (E) generada por la CPU, de manera que el

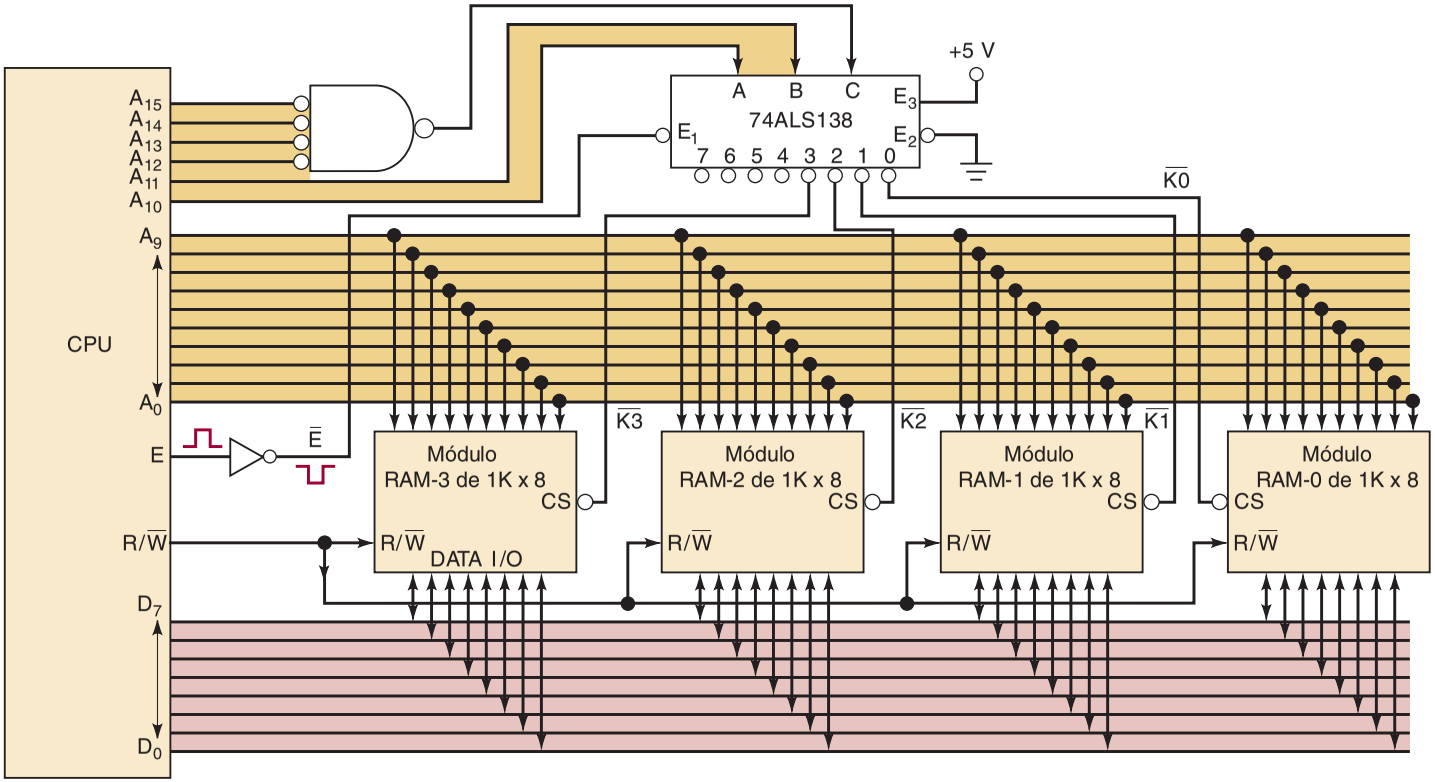


FIGURA 12-42 Memoria RAM de 4K × 8, conectada a una CPU.

decodificador se habilite sólo cuando E se encuentre en ALTO. El pulso en E ocurre sólo después de permitir suficiente tiempo para que las líneas de dirección se estabilicen después de la aplicación de una nueva dirección en el bus de direcciones. E cambiará a BAJO mientras que estén cambiando las líneas de dirección y $R/\overline{W}$; esto evita que ocurran deformaciones en la salida del decodificador que pudieran activar por error un chip de memoria y la posibilidad de almacenar datos no válidos.

Cada módulo de RAM tiene sus entradas de dirección conectadas a las líneas del bus de direcciones A_0-A_9 de la CPU. Las líneas de dirección de mayor orden A_{10}-A_{15} seleccionan uno de los módulos de RAM. El módulo seleccionado decodifica las líneas de dirección A_0-A_9 para encontrar la ubicación de la palabra que se está direccionando. Los siguientes ejemplos mostrarán cómo determinar las direcciones que corresponden a cada uno de los módulos.

EJEMPLO 12-16

Suponga que la CPU va a realizar una operación de lectura a partir de la dirección 06A3 (hexadecimal). ¿De cuál módulo RAM (si es que lo hay) se van a leer los datos?

Solución

Primero escriba la dirección en binario.

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0	1	1	0	1	0	1	0	0	0	1	1

Usted deberá ser capaz de verificar que los niveles de A_{15}-A_9 activen la salida $\overline{K1}$ del decodificador para seleccionar el módulo de RAM-1. Este módulo decodifica en forma interna las líneas de dirección A_9-A_0 para seleccionar la ubicación cuyos datos se van a colocar en el bus de datos.

EJEMPLO 12-17

¿En cuál módulo de RAM se escribirán datos cuando la CPU ejecute una operación de escritura para la dirección 1C65H?

Solución

Si escribimos la dirección en binario, podremos ver que $A_{12} = 1$. Esto produce un nivel ALTO como salida de la compuerta OR y en la entrada C del decodificador. Cuando $A_{11} = A_{10} = 1$ las entradas del decodificador son 111, con lo cual se activa la salida 7. Las salidas $\overline{K0}$ a $\overline{K3}$ estarán inactivas, por lo que ninguno de los módulos de RAM se habilitará. En otras palabras, los datos que la CPU coloque en el bus de datos no los aceptará ninguna de las RAMs.

EJEMPLO 12-18

Determine el intervalo de direcciones para cada módulo en la figura 12-42.

Solución

Cada módulo almacena 1024 palabras de ocho bits. Para determinar las direcciones de las palabras almacenadas en cualquier módulo, debemos empezar por determinar las condiciones del bus de direcciones que activan la entrada de selección de chip de ese módulo. Por ejemplo, el módulo-3 se seleccionará cuando la entrada $\overline{K3}$ del decodificador se encuentre en BAJO (figura 12-43). $\overline{K3}$ estará en BAJO cuando $CBA = 011$. Si regresamos a las líneas de dirección A_{15}- A_{10} de la CPU, podemos ver que el módulo-3 se habilitará cuando se coloque la siguiente dirección en el bus de direcciones:

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0	1	x	x	x	x	x	x	x	x	x	x

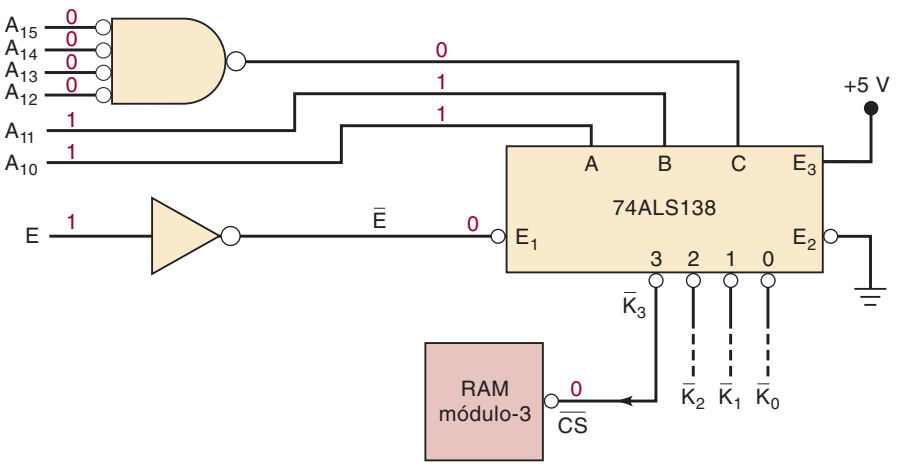


FIGURA 12-43
Ejemplo 12-18, en el que se muestran las condiciones del bus de direcciones necesarias para seleccionar el módulo de RAM-3.

Las "*x*" bajo las entradas A_9-A_0 indican la condición "no importa", ya que el decodificador no utiliza estas líneas de dirección para seleccionar el módulo-3. A_0-A_9 pueden tener cualquier combinación desde 0000000000 hasta 1111111111, dependiendo de cuál sea la palabra en el módulo-3 a la que se esté accediendo. Así, el intervalo completo de direcciones para el módulo-3 se determina mediante el uso de 0s en todas las entradas, y después 1s para todas las "*x*".

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	$\rightarrow 0C00_{16}$
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	$\rightarrow 0FFF_{16}$

Por último, esto nos da el intervalo de direcciones hexadecimales almacenadas en el módulo-3 de 0C00H a 0FFFH. Cuando la CPU coloque en el bus de datos cualquier dirección en este intervalo, sólo el módulo-3 se habilitará para una operación ya sea de lectura o de escritura, dependiendo del estado de $R/\overline{W}$.

Puede utilizarse un análisis similar para determinar los intervalos de direcciones para cada uno de los otros módulos de RAM. Los resultados son los siguientes:

- Módulo-0: 0000H-03FFH
- Módulo-1: 0400H-07FFH
- Módulo-2: 0800H-0BFFH
- Módulo-3: 0C00H-0FFFH

Observe que los cuatro módulos se combinan para un intervalo total de direcciones de 0000H 0FFFH.

Prueba de la lógica de decodificación

En algunas situaciones, la porción del circuito de RAM correspondiente a la lógica de decodificación (figura 12-43) puede probarse mediante el uso de las diversas técnicas que hemos aplicado a los circuitos combinacionales. Puede probarse mediante la aplicación de señales a las seis líneas de dirección más significativas y a *E*, para después monitorear las salidas del decodificador. Para ello, debe ser posible desconectar la CPU con facilidad de estas líneas de señal. Si la CPU es un chip microprocesador en un zócalo, basta con quitarlo de su zócalo.

Una vez que se desconecta la CPU, usted puede suministrar las señales A_{10}-A_{15} y *E* desde un circuito de prueba externo para realizar una prueba estática, mediante el uso de interruptores operados en forma manual para cada señal, o una prueba dinámica mediante el uso de algún tipo de contador para recorrer en ciclo los diversos códigos de dirección. Al aplicar estas señales de prueba, se pueden comprobar las líneas de salida del decodificador para ver si se obtiene la respuesta apropiada.

Pueden utilizarse técnicas estándar de rastreo de señales para aislar cualquier falla en la lógica de decodificación.

Si usted no tiene acceso a las líneas de dirección del sistema, o si no tiene una manera conveniente de generar las señales lógicas estáticas, a menudo es posible forzar el sistema para que genere una secuencia de direcciones. La mayoría de los sistemas computacionales que se utilizan para el desarrollo tienen un programa almacenado en una ROM, el cual permite al usuario visualizar y modificar el contenido de cualquier ubicación de memoria. Cada vez que la computadora accede a una ubicación de memoria debe colocarse la dirección apropiada en el bus, la cual debe hacer que la salida del decodificador cambie al nivel bajo, incluso aunque sea por un tiempo corto. Escriba la siguiente instrucción para la computadora:

```
Display from 0400H to 07FFH
```

Después coloque la sonda lógica en la salida $K1$. La sonda lógica deberá mostrar pulsos durante el tiempo en el que se visualicen los valores de datos.

EJEMPLO 12-19

En la lógica de decodificación de la figura 12-43 se realiza una prueba dinámica; para ello se mantiene $E = 1$ y se conectan las salidas de un contador de seis bits a las entradas de dirección A_{10}-A_{15}. Las salidas del decodificador se monitorean a medida que el contador recorre en forma cíclica y repetitiva todos los códigos de seis bits. Una comprobación con sonda lógica en las salidas del decodificador muestra pulsos en $\overline{K1}$ y $\overline{K3}$, pero muestra que $\overline{K0}$ y $\overline{K2}$ permanecen en ALTO. ¿Cuáles son las fallas más probables?

Solución

Es posible, pero muy poco probable, que $\overline{K0}$ y $\overline{K2}$ pudieran estar atascadas en el nivel ALTO debido a un corto interno o externo con V_{CC}. Una falla más probable sería un circuito abierto entre A_{10} y la entrada A del decodificador, ya que esto actuaría como un nivel lógico ALTO y evitaría que se activara cualquier salida del decodificador con numeración par. También es posible que la entrada A del decodificador se ponga en corto con V_{CC}, pero esto también es muy poco probable ya que de seguro este corto habría afectado la operación del contador que suministra las entradas de dirección.

Prueba del sistema completo de RAM

Los procedimientos de prueba y diagnóstico de fallas en la lógica de decodificación no revelarán problemas con los chips de memoria y sus conexiones con los buses de la CPU. Los métodos más comunes para probar la operación del sistema *completo* de RAM implican la escritura de patrones conocidos de 1s y 0s en cada ubicación de memoria, para luego leerlos y verificar que la ubicación haya almacenado los patrones en forma apropiada. Aunque pueden utilizarse muchos patrones distintos, uno de los más populares es el "patrón de tablero de damas". En este patrón se alternan 1s y 0s, como en 01010101. Una vez que se han probado todas las ubicaciones mediante el uso de este patrón, se invierte (es decir, 10101010) y se prueba cada ubicación otra vez. Hay que recalcar que esta secuencia de pruebas comprobará que cada celda tenga la habilidad de almacenar y leer tanto un 1 como un 0. Debido a que alterna 1s y 0s, el patrón de tablero de damas también detectará cualquier interacción o corto entre celdas adyacentes. Pueden utilizarse muchos otros patrones para detectar varios modos de falla dentro de los chips de RAM.

Ninguna prueba de memoria puede detectar todas las posibles fallas de RAM con un 100 por ciento de precisión, aún y cuando pueda demostrar que cada celda puede almacenar y leer un 0 o un 1. Algunas RAMs defectuosas pueden ser sensibles a los patrones. Por ejemplo, tal vez una RAM pueda almacenar y leer 01010101 y 10101010, pero podría fallar al almacenar 11100011. Incluso para un sistema de

RAM pequeño, se requeriría de un tiempo exageradamente largo para tratar de almacenar y leer todos los patrones posibles en cada ubicación. Por esta razón, si un sistema de RAM pasa la prueba del tablero de damas, puede concluir que *probablemente* está bien; si falla la prueba, entonces *definitivamente* contiene una falla.

El proceso de prueba manual de miles de ubicaciones de RAM mediante el almacenamiento y la lectura de patrones de tablero de damas requeriría cientos de horas, por lo que es obvio que no es factible. La prueba de patrones de RAM se realiza, por lo general, en forma automática, ya sea haciendo que la CPU ejecute un programa de prueba de memoria o conectando un instrumento de prueba especial a los buses del sistema de RAM en lugar de la CPU. De hecho, en muchas computadoras y equipos basados en microprocesador, la CPU ejecuta de manera automática un programa de prueba de memoria cada vez que se enciende; a esto se le conoce como **autoprueba al encendido**. La rutina de autoprueba (la llamaremos AUTOPRUEBA) se almacena en la ROM y se ejecuta cada vez que se enciende el sistema, o cuando el operador la solicita desde el teclado. Cuando el CPU ejecute la AUTOPRUEBA, escribirá los patrones de prueba y leerá esos patrones en/desde cada ubicación de la RAM, y mostrará algún tipo de mensaje al usuario. Puede ser algo tan simple como un LED para indicar una falla en la memoria, o puede ser un mensaje descriptivo impreso en la pantalla o en la impresora. Algunos mensajes típicos podrían ser:

```
RAM module 3 test OK
ALL RAM working properly
Location 027F faulty in bit positions 6 and 7
```

Con mensajes como éstos y un conocimiento de la operación del sistema de RAM, el técnico de diagnóstico de fallas puede determinar qué acción adicional es necesaria para aislar la falla.

PREGUNTAS DE REPASO

1. ¿Cuál es la función de *E* en el circuito de RAM de la figura 12-42?
2. ¿Cuál es la prueba del tablero de damas? ¿Por qué se utiliza?
3. ¿Qué es una autoprueba al encendido?

12-21 PRUEBA DE LA ROM

El circuito de la ROM en una computadora es muy similar al circuito de la RAM (compare las figuras 12-37 y 12-42). La lógica de decodificación de la ROM puede probarse de la misma forma que describimos en la sección anterior para el sistema de RAM. No obstante, los chips de ROM deben probarse en forma distinta a los chips de RAM, ya que no podemos escribir patrones en la ROM y leerlos de vuelta, como en la RAM. Para comprobar el contenido de un CI de ROM se utilizan varios métodos.

En uno de esos métodos, la ROM se coloca en un zócalo de un instrumento especial de prueba que, por lo general, se controla mediante un microprocesador. El instrumento especial de prueba puede programarse para que lea cada una de las ubicaciones en la ROM de prueba e imprima un listado del contenido de cada ubicación. Después el listado puede compararse con lo que se supone debe contener la ROM. Excepto en los chips de ROM de baja capacidad, esta prueba puede requerir de mucho tiempo.

En un método más eficiente, el instrumento de prueba tiene los datos correctos almacenados en su propio chip de ROM de *referencia*. Después, el instrumento de prueba se programa para que lea el contenido de cada ubicación en la ROM de prueba y lo compare con el contenido de la ROM de referencia. Desde luego que este método requiere la disponibilidad de una ROM de referencia preprogramada.

Un tercer método utiliza una **suma de comprobación**, que es un código especial que se coloca en la última o en las dos últimas ubicaciones del chip de ROM al

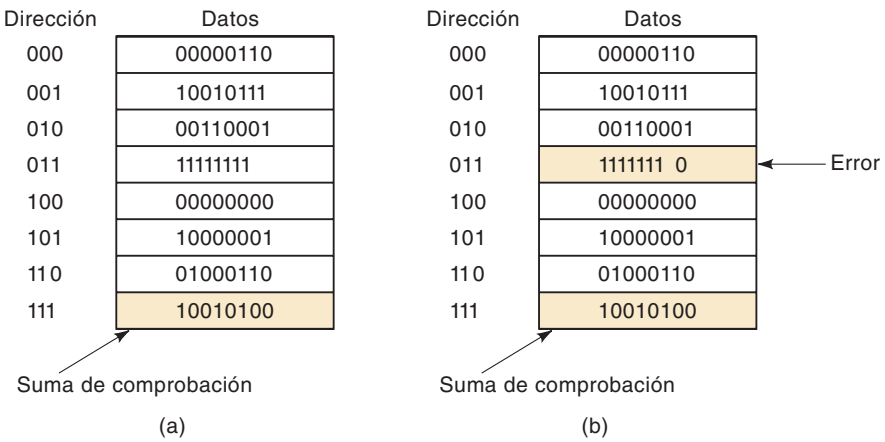


FIGURA 12-44
Método de suma de comprobación para una ROM de 8 × 8: (a) ROM con los datos correctos; (b) ROM con error en sus datos.

momento en que se va a programar. Este código se deriva de la suma de todas las palabras de datos que se van a almacenar en todas las ubicaciones de la ROM (excluyendo las que contienen la suma de comprobación). Cuando el instrumento de prueba lea los datos de cada ubicación de la ROM de prueba, los sumará y desarrollará su propia suma de comprobación. Después comparará la suma de comprobación que calculó con la que está almacenada en las últimas ubicaciones de la ROM, y las dos deberán concordar. De ser así, hay una alta probabilidad de que la ROM esté bien (hay una muy pequeña probabilidad de que una combinación de errores en los datos de la ROM de prueba siga produciendo el mismo valor de la suma de comprobación). Si no concuerdan, hay un problema definitivo en la ROM de prueba.

La idea de la suma de comprobación se ilustra en la figura 12-44(a) para una ROM muy pequeña. La palabra de datos almacenada en la última dirección es la suma de ocho bits de las otras siete palabras de datos (ignorando los acarreos del MSB). Cuando esta ROM se programa, la suma de comprobación se coloca en la última ubicación. La figura 12-44(b) muestra los datos que podrían leerse de una ROM defectuosa que se programó originalmente con los datos de la figura 12-44(a). Observe el error en la palabra de la dirección 011. Cuando el instrumento de prueba lee los datos de cada ubicación de la ROM defectuosa, calcula su propia suma de comprobación con base en esos datos. Debido al error, la suma de comprobación calculada será de 10010011. Cuando el instrumento de prueba compara esta suma con el valor de suma de comprobación almacenado en la ubicación 111 de la ROM, verá que estas dos sumas no concuerdan y se indicará un error en la ROM. Desde luego que no puede determinarse la ubicación exacta del error.

El método de la suma de comprobación también lo puede utilizar una computadora o equipo basado en microprocesador durante una autoprueba automática al encendido, para comprobar el contenido de las ROMs del sistema. De nuevo, como en la autoprueba utilizada para la RAM, la CPU ejecutaría un programa al encenderse, el cual realizaría una prueba de suma de comprobación en cada chip de ROM e imprimiría algún tipo de mensaje de estado. El programa de autoprueba en sí estará ubicado en una ROM, por lo que cualquier error en esa ROM evitaría que se ejecutaran con éxito las pruebas de suma de comparación.

PREGUNTAS DE REPASO

1. ¿Qué es una suma de comprobación? ¿Cuál es su fin?

RESUMEN

1. Todos los dispositivos de memoria almacenan niveles lógicos binarios (1s y 0s) en una estructura tipo arreglo. El tamaño de cada palabra binaria (número de bits) que se almacena varía, dependiendo del dispositivo de memoria. A estos valores binarios se les conoce como *datos*.

2. El lugar (ubicación) en el dispositivo de memoria en donde se almacena cualquier valor de datos se identifica mediante otro número binario, al cual se le conoce como *dirección*. Cada ubicación de memoria tiene una dirección única.
3. Todos los dispositivos de memoria operan de la misma forma general. Para escribir datos en la memoria, la dirección a la cual se va a acceder se coloca en la entrada de dirección, el valor de datos que se va a almacenar se aplica a las entradas de datos y las señales de control se manipulan para almacenar los datos. Para leer datos de la memoria se aplica la dirección, después se manipulan las señales de control y el valor de datos aparece en las terminales de salida.
4. A menudo los dispositivos de memoria se utilizan junto con una CPU de microprocesador que genere las señales de dirección y de control, y que provea los datos que se van a almacenar o que utilice los datos de la memoria. Las operaciones de lectura y de escritura *siempre* se realizan desde la perspectiva de la CPU. Al escribir se colocan datos en la memoria y al leer se obtienen datos de la memoria.
5. La mayoría de las memorias de sólo lectura (ROMs) reciben datos una vez, y de ahí en adelante su contenido no cambia. A este proceso de almacenamiento se le conoce como *programación*. No pierden sus datos cuando se corta la energía del dispositivo. Las MROMs se programan durante el proceso de fabricación. Las PROMs se programan una vez, por medio del usuario. Las EPROMs son justo iguales que las PROMs, sólo que pueden borrarse mediante el uso de luz UV. Las EEPROMs y los dispositivos de memoria Flash pueden borrarse mediante electricidad y alterar su contenido después de la programación. Los CD-ROMs se utilizan para el almacenamiento masivo de información que no necesita cambiar.
6. La memoria de acceso aleatorio (RAM) es un término genérico que se da a los dispositivos que pueden almacenar y recuperar datos con facilidad. Los datos se retienen en un dispositivo de RAM sólo mientras se esté aplicando la energía.
7. La RAM estática (SRAM) utiliza elementos de almacenamiento que son, en escencia, circuitos tipo latch. Una vez que se almacenen los datos, permanecerán sin cambios hasta que se aplique energía al chip. La RAM estática es más fácil de usar, pero más costosa por bit y consume más energía que la RAM dinámica.
8. La RAM dinámica (DRAM) utiliza capacitores para almacenar datos; para ello los carga o los descarga. La simpleza de la celda de almacenamiento permite a las DRAMs almacenar una gran cantidad de datos. Como la carga en los capacitares debe regenerarse con frecuencia, las DRAMs son más complicadas de usar que las SRAMs. A menudo se agregan circuitos extras a los sistemas de DRAM para controlar los ciclos de lectura, de escritura y de regeneración. En muchos de los nuevos dispositivos estas características se están integrando en el mismo chip de DRAM. El objetivo de la tecnología de DRAM es colocar más bits en una pieza de silicio más pequeña, para que consuma menos energía y responda con mayor rapidez.
9. Los sistemas de memoria requieren una amplia variedad de configuraciones distintas. Los chips de memoria pueden combinarse para que implementen cualquier configuración deseada, ya sea que su sistema necesite más bits por ubicación o más capacidad total de palabras. Todos los diversos tipos de ROM y de RAM pueden combinarse dentro del mismo sistema de memoria.

TÉRMINOS IMPORTANTES

apagado
autoprueba al encendido
búfer circular
búfer de transferencia de datos
búfer lineal
bus de control
bus de datos
bus de direcciones
byte
caché
capacidad
celda de memoria
contador de regeneración
controlador de DRAM
densidad
dirección
empalme de memoria

enlace de fusible
estrobo de dirección de columna (*CAS*)
estrobo de dirección de fila (*RAS*)
FIFO
JEDEC
latencia
mapa de memoria
memoria auxiliar
memoria de acceso aleatorio (RAM)
memoria de acceso secuencial (SAM)
memoria de lectura/escritura (RWM)
memoria de sólo lectura (ROM)
memoria Flash
memoria principal
memoria volátil
multiplexaje de direcciones
operación de escritura
operación de lectura
palabra de memoria
programa de arranque
programación
PROM programable y borrable eléctricamente (EEPROM)
RAM dinámica (DRAM)
RAM estática (SRAM)
regeneración
regeneración sólo de *RAS*
selección de chip
suma de comprobación
tiempo de acceso
uso de estrobos

PROBLEMAS

SECCIONES 12-1 A 12-3

B 12-1.* Cierta memoria tiene una capacidad de 16K × 32. ¿Cuántas palabras almacena? ¿Cuál es el número de bits por palabra? ¿Cuántas celdas de memoria contiene?

B 12-2. ¿Cuántas direcciones distintas requiere la memoria del problema 12-1?

B 12-3.* ¿Cuál es la capacidad de una memoria que tiene 16 entradas de dirección, cuatro entradas de datos y cuatro salidas de datos?

B 12-4. Cierta memoria almacena 8K palabras de 16 bits. ¿Cuántas líneas de entrada y de salida de datos tiene? ¿Cuántas líneas de dirección tiene? ¿Cuál es su capacidad en bytes?

PREGUNTAS DE PRÁCTICA

12-5. Defina cada uno de los siguientes términos.

B
(a) RAM.
(b) RWM.
(c) ROM.
(d) Memoria interna.
(e) Memoria auxiliar.
(f) Capacidad.
(g) Volátil.
(h) Densidad.
(i) Lectura.
(j) Escritura.

12-6. (a) ¿Cuáles son los tres buses en un sistema de memoria de computadora?

B
(b) ¿Qué bus utiliza la CPU para seleccionar la ubicación de memoria?
(c) ¿Cuál bus se utiliza para llevar datos de la memoria a la CPU, durante una operación de lectura?
(d) ¿Cuál es el origen de los datos en el bus de datos, durante una operación de escritura?

* Encontrará las respuestas a los problemas marcados con un asterisco al final del libro.

SECCIONES 12-4 Y 12-5

12-7.* Consulte la figura 12-6. Determine las salidas de datos para cada una de las siguientes condiciones de entrada:

B (a) $[A] = 1011$; $CS = 1$

(b) $[A] = 0111$; $CS = 0$

B 12-8. Consulte la figura 12-7.

(a) ¿Cuál registro se habilita mediante la dirección de entrada 1011?

(b) ¿Qué código de dirección de entrada selecciona el registro 4?

B 12-9.* Cierta ROM tiene una capacidad de 16K $\times$ 4 y una estructura interna como la que se muestra en la figura 12-7.

(a) ¿Cuántos registros hay en el arreglo?

(b) ¿Cuántos bits hay por registro?

(c) ¿Qué tamaño de decodificadores requiere?

PREGUNTA DE PRÁCTICA

B 12-10. (a) *Verdadero o falso:* las ROMs no se pueden borrar.

(b) ¿Qué significa *programar* o *quemar* una ROM?

(c) Defina el tiempo de acceso de una ROM.

(d) ¿Cuántas entradas de datos, salidas de datos y entradas de dirección se necesitan para una ROM de 1024 $\times$ 4?

(e) ¿Cuál es la función de los decodificadores en un chip de ROM?

SECCIÓN 12-6

A, D 12-11.* La figura 12-45 muestra cómo pueden transferirse los datos de una ROM hacia un registro externo. La ROM tiene los siguientes parámetros de sincronización: $t_{ACC} = 250$ ns y $t_{OE} = 120$ ns. Suponga que se han aplicado las nuevas entradas de dirección a la ROM 500 ns antes de que ocurra el pulso en TRANSFERIR. Determine la duración mínima del pulso en TRANSFERIR para una transferencia de datos confiable.

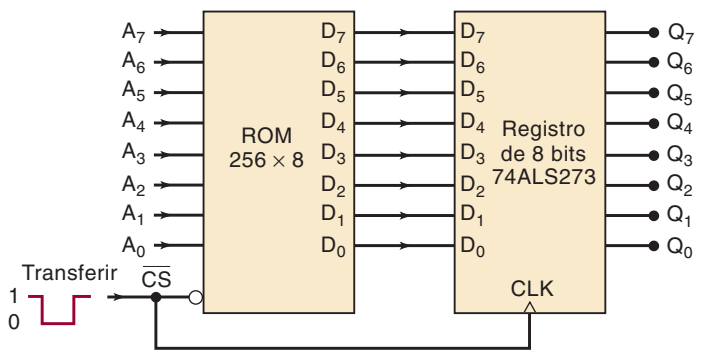


FIGURA 12-45
Problema 12-11.

A, D 12-12. Repita el problema 12-11 si las entradas de dirección se cambian 70 ns antes del pulso en TRANSFERIR.

SECCIONES 12-7 Y 12-8

B 12-13. **PREGUNTA DE PRÁCTICA**

Para cada elemento que se muestra a continuación, indique el tipo de memoria que se está describiendo: MPROM, PROM, EPROM, EEPROM, Flash. Algunos elementos corresponderán a más de un tipo.

(a) El usuario puede programarla pero no se puede borrar.
(b) El fabricante la programa.
(c) Es volátil.
(d) Puede borrarse y reprogramarse una y otra vez.
(e) Pueden borrarse y reescribirse palabras individuales.
(f) Se borra con luz UV.
(g) Se borra mediante electricidad.
(h) Utiliza enlaces de fusible.
(i) Puede borrarse en masa o en sectores de 512 bytes.
(j) No tiene que quitarse del sistema para borrarse y reprogramarse.
(k) Requiere un voltaje de suministro especial para reprogramarse.
(l) El tiempo de borrado es aproximadamente de 15 a 20 min.

B 12-14. ¿Qué transistores en la figura 12-9 estarán conduciendo cuando $A_1 = A_0 = 1$ y $\overline{EN} = 0$?

12-15.* Cambie las conexiones de la MROM en la figura 12-9, de manera que la MROM almacene la función $y = 3x + 5$.

D 12-16. La figura 12-46 muestra un circuito simple para programar en forma manual una EPROM 2732. Cada una de las terminales de datos de la EPROM está conectada a un interruptor que puede establecerse en un nivel de 1 o de 0. Las entradas de dirección se controlan mediante un contador de 12 bits. El pulso de programación de 50 ms proviene de un monoestable cada vez que se oprime el botón PROGRAMAR.

(a) Explique cómo puede utilizarse este circuito para programar las ubicaciones de memoria de la EPROM en forma secuencial con los datos deseados.

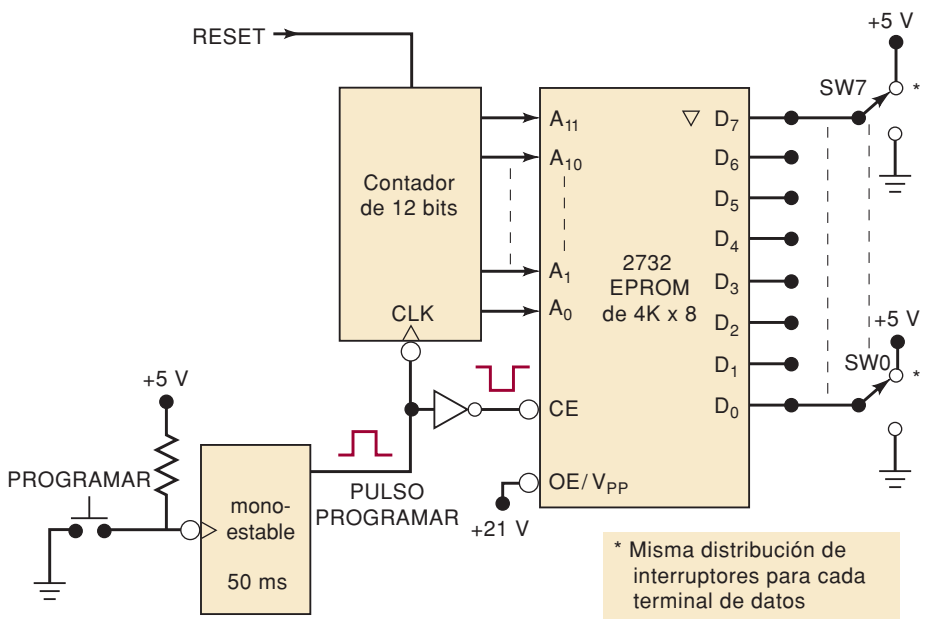


FIGURA 12-46
Problema 12-16.

(b) Muestre cómo pueden utilizarse chips 74293 y un 74121 para implementar este circuito.

(c) ¿Debería el rebote de interruptores tener algún efecto sobre la operación del circuito?

N 12-17.* La figura 12-47 muestra un chip de memoria Flash 28F256A conectado a una CPU a través de un bus de datos y un bus de direcciones. Para escribir en/leer del arreglo de memoria Flash, la CPU envía la dirección de memoria deseada y genera las señales de control apropiadas para el chip [figura 12-15(b)]. La CPU también puede escribir en el registro de instrucciones del chip (figura 12-16) mediante la generación de las señales de control apropiadas y enviando la instrucción deseado a través del bus de datos. Para esta última operación, la CPU no tiene que enviar una dirección de memoria específica al chip; en otras palabras, las líneas de dirección son "no importa".

(a) Considere la siguiente secuencia de operaciones de la CPU. Determine lo que habrá ocurrido a la memoria Flash cuando se complete la secuencia. Suponga que el registro de instrucciones está reteniendo el valor 00_{16}.

1. La CPU coloca 20_{16} en el bus de datos y aplica pulsos a $\overline{CE}$ y $\overline{WE}$ para que cambien a BAJO mientras que $\overline{OE}$ se mantiene en ALTO. El bus de direcciones está en 0000_{16}.
2. La CPU repite el paso 1.

(b) Después de la secuencia anterior se ha ejecutado, la CPU ejecuta la siguiente secuencia. Determine qué es lo que esto provoca en el chip de memoria Flash.

1. La CPU coloca 40_{16} en el bus de datos y aplica pulsos a $\overline{CE}$ y $\overline{WE}$ para que cambien a BAJO mientras que $\overline{OE}$ se mantiene en ALTO. El bus de direcciones está en 0000_{16}.
2. La CPU coloca $3C_{16}$ en el bus de datos y 2003_{16} en el bus de direcciones, y aplica pulsos a $\overline{CE}$ y $\overline{WE}$ para que cambien a BAJO mientras que $\overline{OE}$ se mantiene en ALTO.

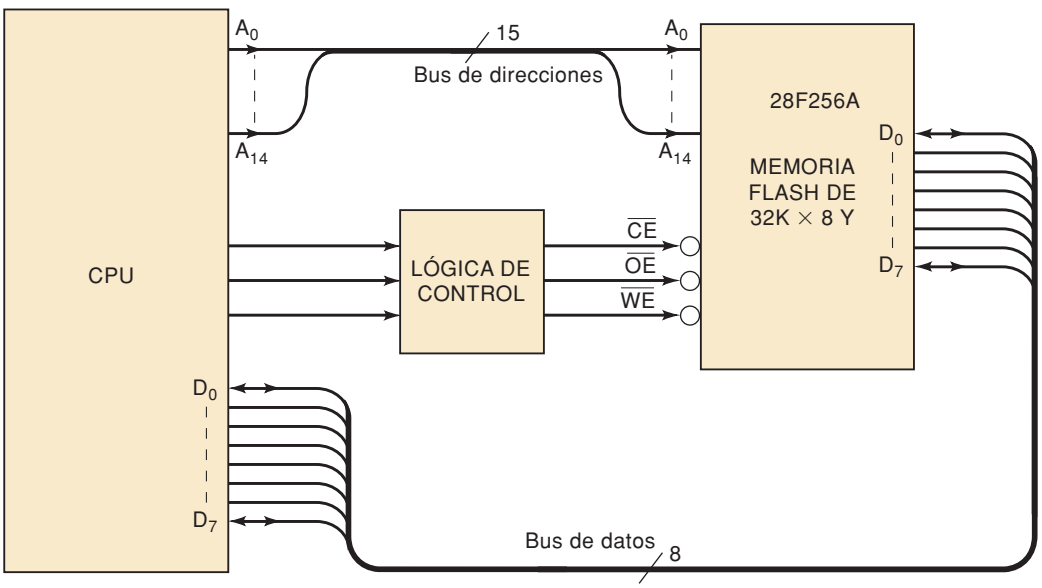
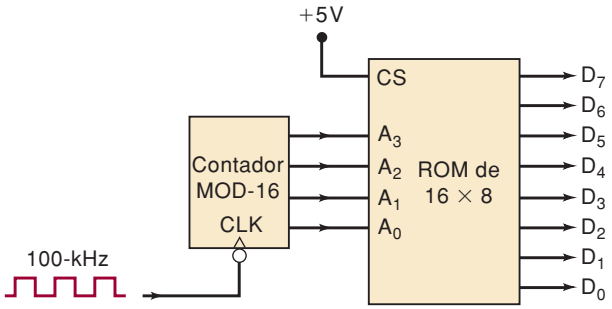


FIGURA 12-47 Problema 12-17.

SECCIÓN 12-9

N 12-18. La generación de señales de sincronización y control es otra de las aplicaciones de la ROM. La figura 12-48 muestra una ROM de 16 × 8 en la que

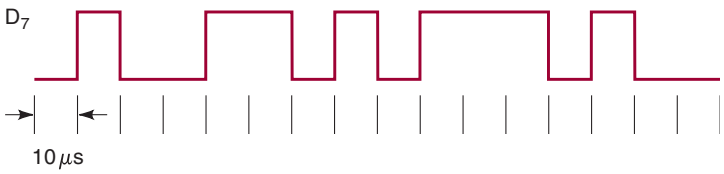
FIGURA 12-48
Problema 12-18



sus entradas de dirección se controlan mediante un contador MOD-16, de manera que se incrementen las direcciones de la ROM en cada pulso de entrada. Suponga que la ROM se programa como en la figura 12-6 y realice un bosquejo de las formas de onda en cada salida de la ROM a medida que se aplican pulsos. Ignore los tiempos de retraso de la ROM. Suponga que el contador inicia en 0000.

D 12-19.* Modifique el programa almacenado en la ROM del problema 12-18 para que genere la forma de onda de D_7 de la figura 12-49.

FIGURA 12-49
Problema 12-19.



D 12-20.* Consulte el generador de funciones de la figura 12-17.

(a) ¿Qué frecuencia de reloj producirá una onda senoidal de 100 Hz en la salida?

(b) ¿Qué método podría utilizarse para variar la amplitud de pico a pico de la onda senoidal?

A 12-21. Para el ML2035 de la figura 12-18, suponga que un valor de 038E (hexadecimal) en latch producirá la frecuencia deseada. Dibuje el diagrama de sincronización para las entradas *LATI*, *SID* y *SCK* y suponga que primero se desplaza el LSB.

N, A 12-22.* El sistema que se muestra en la figura 12-50 es un generador de formas de funciones. Utiliza cuatro tablas de búsqueda de 256 puntos en una ROM de 1 Kbyte, para almacenar en cada una un solo ciclo de una onda senoidal (dirección 000-0FF), una rampa con pendiente positiva (dirección 100-1FF), una rampa con pendiente negativa (200-2FF) y una onda triangular (300-3FF). La relación de fase entre los tres canales de salida se controla mediante los valores que se cargan al principio en los tres contadores. Los parámetros críticos de sincronización son $t_{pd(ck\text{-}Q\ y\ OE\text{-}Q\ máx)}$, contadores = 10 ns, latches = 5 ns y t_{ACC} de ROM = 20 ns. Estudie el diagrama hasta que comprenda cómo funciona y después responda a lo siguiente:

(a) Si al principio el contador A se carga con 0, ¿qué valores deben cargarse en los contadores B y C, de forma que A se retrase con respecto a B por 90° y que A se retrase con respecto a C por 180°?

(b) Si al principio el contador A se carga con 0, ¿qué valores deben cargarse en los contadores B y C para generar una onda senoidal de tres fases, con un desplazamiento de 120° entre cada salida?

(c) ¿Cuál debe ser la frecuencia de pulsos en DAC_SAL para poder generar una salida de onda senoidal de 60 Hz?

(d) ¿Cuál es la frecuencia máxima de la entrada CLK?

(e) ¿Cuál es la frecuencia máxima de las formas de onda de salida?

(f) ¿Cuál es el propósito del contador de selección de función?

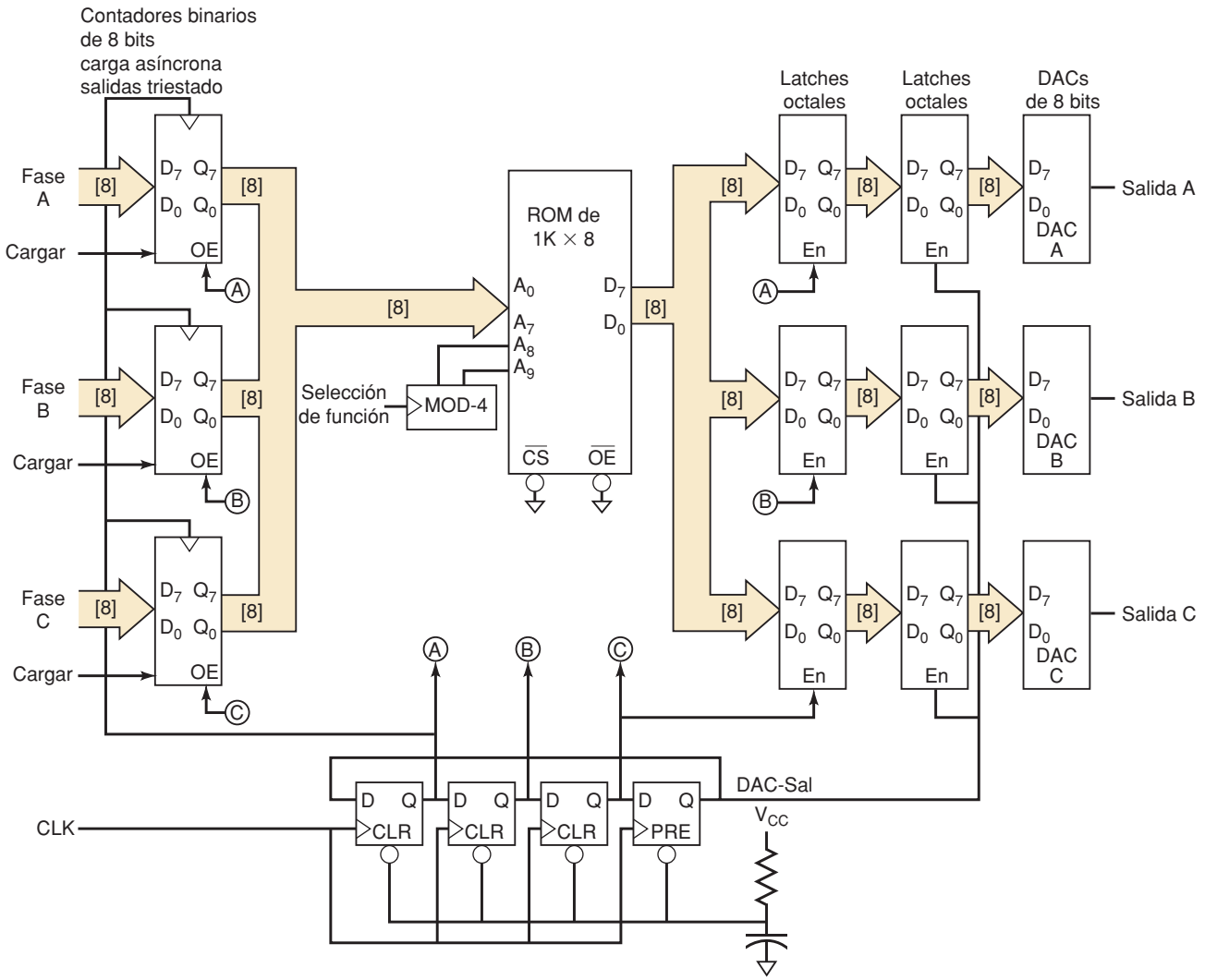


FIGURA 12-50 Problema 12-22.

SECCIÓN 12-11

12-23. (a) Dibuje el símbolo lógico para un MCM101514, una RAM estática tipo CMOS organizada como memoria de 256K × 4 con entradas y salidas de datos separadas, y una señal de habilitación de chip activa en BAJO.

(b) Dibuje el símbolo lógico para un MCM6249, una RAM estática tipo CMOS organizada como memoria de 1M × 4 con E/S común, señal de habilitación de chip activa en BAJO y señal de habilitación de salida activa en BAJO.

SECCIÓN 12-12

12-24.* Cierta RAM estática tiene los siguientes parámetros de sincronización (en nanosegundos):

$t_{RC} = 100$	$t_{AS} = 20$
$t_{ACC} = 100$	t_{AH} = no se proporciona
$t_{CO} = 70$	$t_W = 40$
$t_{OD} = 30$	$t_{DS} = 10$
$t_{WC} = 100$	$t_{DH} = 20$

(a) ¿Cuánto tiempo después de que se estabilicen las líneas de dirección aparecerán datos válidos en las salidas, durante un ciclo de lectura?

(b) ¿Cuánto tiempo permanecerán válidos los datos después de que $\overline{CS}$ regrese al nivel ALTO?

(c) ¿Cuántas operaciones de lectura pueden realizarse por segundo?

(d) ¿Cuánto tiempo deberán mantenerse $R/\overline{W}$ y $\overline{CS}$ en ALTO después de que se estabilice la nueva dirección, durante un ciclo de escritura?

(e) ¿Cuál es el mínimo de tiempo que deben permanecer válidos los datos de entrada para que ocurra una operación de escritura confiable?

(f) ¿Cuánto tiempo deben permanecer estables las entradas de dirección después de que *R/W* y *CS* regresen al nivel ALTO?

(g) ¿Cuántas operaciones de escritura pueden realizarse por segundo?

SECCIONES 12-13 A 12-17

12-25. Dibuje el símbolo lógico para el TMS4256, un chip de DRAM de 256K ×1. ¿Cuántas terminales podemos ahorrar si utilizamos el multiplexaje de direcciones para esta DRAM?

D 12-26. La figura 12-51(a) muestra un circuito que genera las señales $\overline{RAS}$, $\overline{CAS}$ y *MUX* necesarias para que el circuito de la figura 12-28(b) opere en forma apropiada. La señal de reloj maestro de 10 MHz proporciona la sincronización básica para la computadora. La CPU genera la señal de petición de memoria (*MEMR*) en sincronía con el reloj maestro, como se muestra en la parte (b) de la figura. Por lo general *MEMR* se encuentra en BAJO y se lleva a ALTO cada vez que la CPU quiere acceder a la memoria para una operación de lectura o de escritura. Determine las formas de onda en Q_0, $\overline{Q}_1$ y Q_2, y compárelas con las formas de onda deseadas de la figura 12-29.

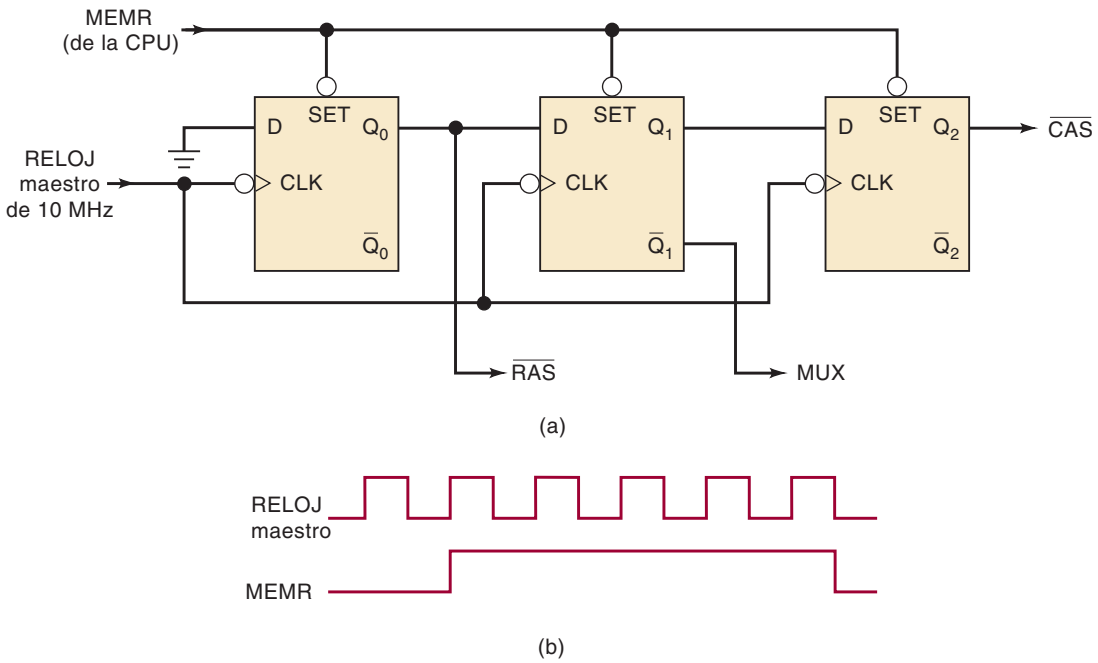


FIGURA 12-51 Problema 12-26.

D 12-27. Muestre cómo conectar dos multiplexores 74157 (sección 9-6) para proporcionar la función de multiplexaje requerida en la figura 12-28(b).

12-28. Consulte las señales en la figura 12-30. Describa lo que ocurre en cada uno de los puntos de tiempo etiquetados.

12-29. Repita el problema 12-28 para la figura 12-31.

A 12-30.* El chip 21256 es una DRAM de 256K × 1 que consiste de un arreglo de 512 × 512 celdas. Estas celdas deben regenerarse dentro de un intervalo no mayor de 4 ms para que puedan retenerse los datos. La figura 12-33(a) muestra las señales que se utilizan para ejecutar un ciclo de regeneración de $\overline{CAS}$ antes que $\overline{RAS}$. Cada vez que ocurre un ciclo como éste, los circuitos de regeneración integrados en el chip regenerarán una fila del arreglo en la dirección de fila especificada por un contador de regeneración. El contador se incrementa después de cada regeneración. ¿Con qué frecuencia deben aplicarse los ciclos de $\overline{CAS}$ antes que $\overline{RAS}$ para poder retener todos los datos?

12-31.* Estudie el diagrama de bloques funcional de la DRAM TMS44100 en la figura 12-27.

(a) ¿Cuáles son las dimensiones reales del arreglo de celdas de DRAM?

(b) Si el arreglo de celdas fuera cuadrado, ¿cuántas filas habría?

(c) ¿Cómo afectaría esto al tiempo de regeneración?

SECCIÓN 12-18

D 12-32. Muestre cómo combinar dos chips de RAM 6206 (figura 12-20) para producir un módulo de 32K × 16.

D 12-33. Muestre cómo conectar dos de los chips de RAM 6264 que se simbolizan en la figura 12-23, para producir un módulo de RAM de 16K × 8. El circuito no debe requerir lógica adicional. Dibuje un mapa de memoria que muestre el intervalo de direcciones de cada chip de RAM.

D 12-34.* Describa cómo modificar el circuito de la figura 12-37 para que tenga una capacidad total de 16K × 8. Use el mismo tipo de chips de PROM.

D 12-35. Modifique el circuito decodificador de la figura 12-37 para que opere a partir de un bus de direcciones de 16 líneas (es decir, agregue A_{13}, A_{14} y A_{15}). Las cuatro PROMs deberán mantener los mismos intervalos de direcciones hexadecimales.

A 12-36. Para el sistema de memoria de la figura 12-38, suponga que la CPU va a almacenar un byte de datos en la dirección 4000H del sistema.

(a) ¿En cuál chip está almacenado el byte?

(b) ¿Hay alguna otra dirección en este sistema que pueda acceder a este byte de datos?

(c) Responda a las partes (a) y (b) suponiendo que la CPU almacenó un byte en la dirección 6007H. (*Sugerencia:* recuerde que la EEPROM no está decodificada por completo.)

(d) Suponga que el programa va a almacenar una secuencia de bytes de datos en la EEPROM y que acaba de completar el byte 2048 en la dirección 67FFH. Si el programador le permite almacenar un byte más en la dirección 6800H, ¿cuál será el efecto sobre los primeros 2048 bytes?

D 12-37. Dibuje el diagrama completo para una memoria de 256K × 8 que utiliza chips de RAM con las siguientes especificaciones: capacidad de 64K × 4, línea común de entrada/salida y dos entradas de selección de chip activas en BAJO. [*Sugerencia:* el circuito puede diseñarse utilizando sólo dos inversores (más los chips de memoria).]

SECCIÓN 12-20

12-38.* Modifique el circuito de RAM de la figura 12-42 de la siguiente manera: cambie la compuerta OR por una compuerta AND y desconecte su salida de *C*;

conecte la salida de la compuerta AND a E_3; conecte C a tierra. Determine el intervalo de direcciones para cada módulo de RAM.

A, D 12-39. Muestre cómo expandir el sistema de la figura 12-42 a una memoria de 8K × 8, con un intervalo de direcciones de 0000H a 1FFFH. (*Sugerencia*: puede agregar los módulos de memoria necesarios y modificar la lógica de decodificación existente.)

F 12-40.* En la lógica de decodificación de la figura 12-42 se realiza una prueba dinámica; para ello se mantiene $E = 1$ y se conectan las salidas de un contador de seis bits a las entradas de dirección A_{10}-A_{15}. Las salidas del decodificador se monitorean con un osciloscopio (o un analizador lógico), a medida que se aplican pulsos al contador en forma continua mediante un reloj de 1 MHz. La figura 12-52(a) muestra las señales visualizadas. ¿Cuáles son las fallas más probables?

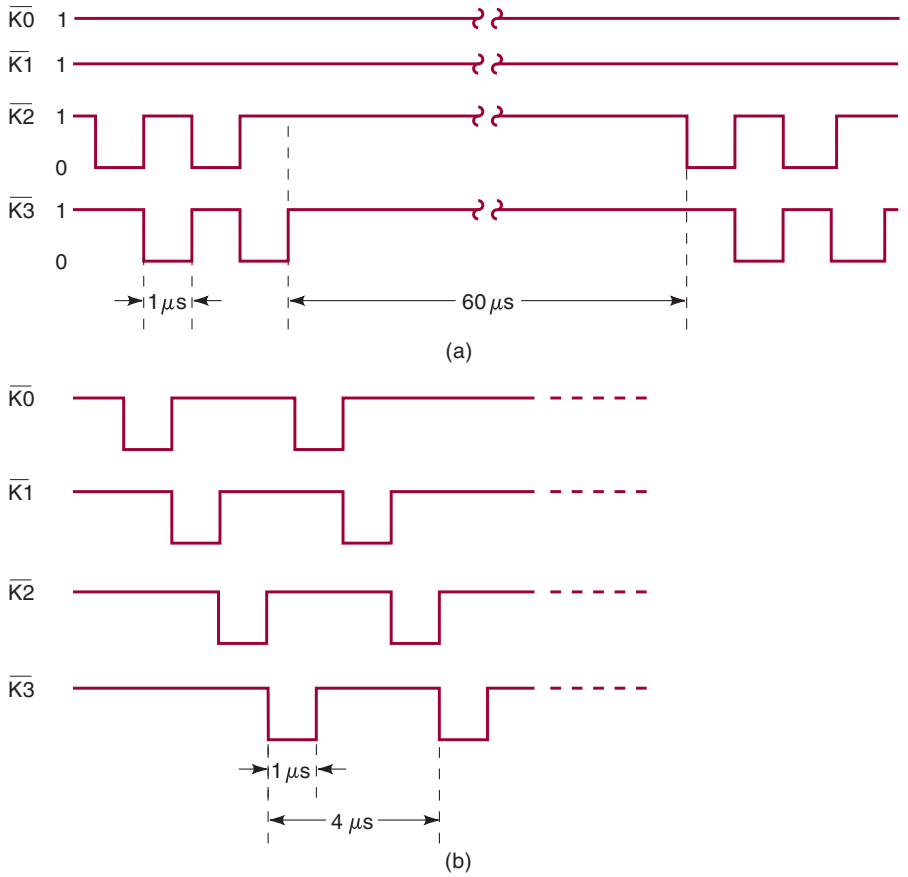


FIGURA 12-52 Problemas 12-40 y 12-41.

A, F 12-41. Repita el problema 12-40 para las salidas del decodificador que se muestran en la figura 12-52(b).

A, D 12-42.* Considere el sistema de RAM de la figura 12-42. La prueba del patrón de tablero de damas no podrá detectar ciertos tipos de fallas. Por ejemplo, suponga que hay una interrupción en la conexión que va a la entrada A del decodificador. Si se realiza una AUTOPRUEBA de patrón de tablero de damas en este circuito, los mensajes visualizados indicarán que la memoria está bien.

(a) Explique por qué no se detectó la falla en el circuito.

(b) ¿Cómo modificaría la AUTOPRUEBA, de manera que puedan detectarse fallas como ésta?

F 12-43.* Suponga que los módulos de 1K × 8 que se utilizan en la figura 12-42 están formados por dos chips de RAM de 1K × 4. Los siguientes mensajes se imprimen cuando se realiza la autoprueba al encendido en este sistema de RAM:

```
module-0 test OK
module-1 test OK
address 0800 faulty at bits 4-7
address 0801 faulty at bits 4-7
address 0802 faulty at bits 4-7
   .  .  .    .     .     .   .
   .  .  .    .     .     .   .
   .  .  .    .     .     .   .
address 0BFE faulty at bits 4-7
address 0BFF faulty at bits 4-7
module-3 test OK
```

Examine estos mensajes y liste las posibles fallas.

F 12-44.* Los siguientes mensajes se imprimen cuando se realiza la autoprueba al encendido en el sistema de RAM de la figura 12-42.

```
module-0 test OK
module-1 test OK
module-2 test OK
address 0C00 faulty at bit 7
address 0C01 faulty at bit 7
address 0C02 faulty at bit 7
  .     .  .     .    .
  .     .  .     .    .
  .     .  .     .    .
address 0FFE faulty at bit 7
address 0FFF faulty at bit 7
```

Examine estos mensajes y liste las posibles fallas.

F 12-45. ¿Qué mensajes se imprimirían si se realizara una autoprueba al encendido en el sistema de RAM de la figura 12-42, si hubiera un corto entre las salidas $\overline{K2}$ y $\overline{K3}$ del decodificador?

SECCIÓN 12-21

F 12-46.* Considere la ROM de 16 × 8 en la figura 12-6. Sustituya la palabra de datos almacenada en la ubicación de dirección 1111 con una suma de comprobación calculada con base en las otras 15 palabras de datos.

RESPUESTAS A LAS PREGUNTAS DE REPASO DE SECCIÓN

SECCIÓN 12-1

1. Vea el texto. 2. 16 bits por palabra; 8192 palabras; 131,072 bits o celdas
3. En una operación de lectura, se obtiene una palabra de una ubicación de memoria y se transfiere hacia otro dispositivo. En una operación de escritura, se coloca una nueva palabra en una ubicación de memoria y se sustituye la que estaba almacenada ahí.
4. Verdadero 5. SAM: el tiempo de acceso no es constante, sino que depende de la ubicación física de la palabra a la que se esté accediendo. RAM: el tiempo de acceso es el mismo para cualquier ubicación de dirección. 6. RWM es memoria en la que se puede leer o escribir con igual facilidad. ROM es memoria que, por lo general, se lee y casi nunca se escribe. 7. Falso; sus datos deben regenerarse en forma periódica.

SECCIÓN 12-2

1. 14, 12, 12 2. Indica a la memoria que debe realizar una operación de lectura o de escritura. 3. Cuando se encuentra en su estado activo, esta entrada habilita a la memoria para que realice la operación de lectura o de escritura seleccionada por la entrada $R/\overline{W}$. Cuando se encuentra en su estado inactivo, esta entrada deshabilita a la memoria para que no pueda realizar la función de lectura o la de escritura.

SECCIÓN 12-3

1. Líneas de dirección, líneas de datos, líneas de control 2. Vea el texto. 3. Vea el texto.

SECCIÓN 12-4

1. Verdadero 2. Aplicar las entradas de dirección deseadas; activar la(s) entrada(s) de control; los datos aparecen en las salidas de datos. 3. El proceso de introducir datos en la ROM.

SECCIÓN 12-5

1. $A_3A_2A_1A_0 = 1001$ 2. El decodificador de selección de fila activa una de las entradas de habilitación de todos los registros en la fila seleccionada. El decodificador de selección de columna activa una de las entradas de habilitación de todos los registros en la columna seleccionada. Los búferes de salida pasan los datos del bus de datos interno hacia las terminales de salida de la ROM cuando la terminal *CS* está activada.

SECCIÓN 12-7

1. Falso; por el fabricante. 2. Una PROM puede programarse una vez por el usuario. No puede borrarse y reprogramarse. 3. Verdadero 4. Mediante la exposición a la luz UV. 5. Verdadero. 6. Programan los datos de manera automática en las celdas de memoria, una dirección a la vez. 7. Una EEPROM puede borrarse y reprogramarse mediante electricidad sin tener que quitarla de su circuito, y puede borrarse byte por byte. 8. Baja densidad; alto costo. 9. EEPROM. 10. Uno.

SECCIÓN 12-8

1. Se puede borrar y programar mediante electricidad sin quitarla del circuito.
2. Mayor densidad; menor costo. 3. Tiempos de borrado y programación cortos.
4. Para las operaciones de borrado y programación. 5. El contenido de este registro controla todas las funciones internas del chip. 6. Para confirmar que se ha borrado una dirección de memoria con éxito (es decir, datos = todos 1s). 7. Para confirmar que se ha programado una dirección de memoria con los datos correctos.

SECCIÓN 12-9

1. Al encendido, la computadora ejecuta un pequeño programa de arranque desde la ROM para inicializar el hardware del sistema y cargar el sistema operativo desde la unidad de almacenamiento masivo (disco). 2. Circuito que recibe datos representados en un tipo de código y los convierte en otro tipo de código. 3. Contador, ROM, DAC, filtro paso bajo. 4. Son no volátiles, rápidos, confiables, pequeños y consumen poca energía.

SECCIÓN 12-11

1. Se aplica la dirección deseada a las entradas de dirección; $R/\overline{W} = 1$; se activa *CS* o *CE*. 2. Para reducir el conteo de terminales. 3. 24, incluyendo V_{CC} y tierra

SECCIÓN 12-12

1. Las celdas de SRAM son flip-flops; las celdas de DRAM utilizan capacitores.
2. CMOS 3. Memoria. 4. CPU. 5. Los tiempos de los ciclos de lectura y escritura. 6. Falso; cuando $\overline{WE}$ está en BAJO, las terminales de E/S actúan como entradas de datos, sin importar el estado de $\overline{OE}$ (segunda entrada en la tabla de modos). 7. A_{13} puede permanecer conectada a la terminal 26. A_{14} debe quitarse y la terminal 27 debe conectarse a +5 V.

SECCIÓN 12-13

1. Una velocidad, por lo general, más lenta; necesidad de regenerarse. 2. Bajo consumo de energía; alta capacidad; menor costo por bit. 3. DRAM.

SECCIÓN 12-14

1. 256 filas × 256 columnas. 2. Ahorra terminales en el chip. 3. 1M = 1024K = 1024 × 1024. Por lo tanto, hay 1024 filas por 1024 columnas. Como $1024 = 2^{10}$, el chip necesita 10 entradas de dirección. 4. $\overline{RAS}$ se utiliza para fijar la dirección de fila en el registro de dirección de fila de la DRAM. $\overline{CAS}$ se utiliza para fijar la dirección de columna en el registro de dirección de columna. 5. *MUX* multiplexa la dirección completa en las direcciones de fila y de columna para introducirlas a la RAM.

SECCIÓN 12-15

1. (a) Verdadero (b) falso (c) falso (d) verdadero 2. *MUX*

SECCIÓN 12-16

1. (a) Verdadero (b) falso. 2. Proporciona las direcciones de fila a la DRAM durante los ciclos de regeneración. 3. Multiplexaje de direcciones y la operación de regeneración. 4. (a) Falso (b) verdadero.

SECCIÓN 12-17

1. No 2. Ubicaciones de memoria con la misma dirección superior (misma fila).
3. Sólo debe fijarse la dirección de columna. 4. Salida de datos extendida.
5. *Ráfaga* 6. El reloj del sistema.

SECCIÓN 12-18

1. Dieciséis 2. Cuatro 3. Falso; cuando se expande la capacidad de la memoria, cada chip se selecciona mediante una salida distinta del decodificador (vea la figura 12-43). 4. Verdadero

SECCIÓN 12-19

1. Respaldo de batería para la RAM tipo CMOS; memoria Flash. 2. La economía
3. Los datos se leen de la memoria en el mismo orden en el que se escribieron.
4. Una FIFO que se utiliza para transferir datos entre dispositivos con velocidades de operación muy distintas. 5. Los búferes circulares "dan la vuelta" de la dirección más alta hasta la dirección más baja, y los datos más recientes siempre sobrescriben a los datos más antiguos.

SECCIÓN 12-20

1. Evita las deformaciones en la decodificación al deshabilitar el decodificador mientras cambian las líneas de dirección. 2. Una manera de probar la RAM mediante la escritura de un patrón de tablero de damas (primero 01010101, después 10101010) en cada ubicación de memoria, para después leerlo. Se utiliza porque detectará cualquier corto o interacción entre las celdas adyacentes. 3. Una prueba automática de la RAM, la cual se lleva a cabo mediante la computadora al momento del encendido.

SECCIÓN 12-21

1. Un código que se coloca en la última o en las últimas dos ubicaciones de la ROM, el cual representa la suma de los datos esperados en la ROM para todas las demás ubicaciones. Se utiliza como un medio para probar si hay errores en una o más ubicaciones de la ROM.

CAPÍTULO 13

ARQUITECTURAS DE LOS DISPOSITIVOS LÓGICOS PROGRAMABLES*†

CONTENIDO

13-1 Árbol familiar de los sistemas digitales
13-2 Fundamentos de los circuitos de un PLD
13-3 Arquitecturas de PLDs
13-4 El GAL 16V8 (matriz lógica genérica)
13-5 El CPLD EPM7128S de Altera
13-6 La familia FLEX10K de Altera
13-7 La familia Cyclone de Altera

* Hemos reproducido los diagramas del dispositivo GAL 16V8 que presentamos en este capítulo gracias a la cortesía de Lattice Semiconductor Corporation, Hillsboro, Oregon.

† Hemos reproducido los diagramas de los dispositivos de las familias MAX7000S y FLEX10K que presentamos en este capítulo gracias a la cortesía de Altera Corporation, San José, California.

OBJETIVOS

Al terminar este capítulo, usted podrá:

- Describir las distintas categorías de dispositivos de sistemas digitales.
- Describir los distintos tipos de PLDs.
- Interpretar la información de los libros de datos de PLDs.
- Definir la terminología de un PLD.
- Comparar las distintas tecnologías de programación utilizadas en los PLDs.
- Comparar las arquitecturas de los distintos tipos de PLDs.
- Comparar las características de las familias MAX7000S y FLEX10K de PLDs de Altera.

INTRODUCCIÓN

A lo largo de los capítulos de este libro le hemos presentado una amplia variedad de circuitos digitales. Ahora sabe cómo funcionan los bloques fundamentales de los sistemas digitales y puede combinarlos para resolver una amplia variedad de problemas digitales. También hemos descrito los sistemas digitales más complicados, como las microcomputadoras y los procesadores de señales digitales. La diferencia definitiva entre los sistemas de microcomputadora/DSP y los otros sistemas digitales es que los primeros siguen una secuencia programada de instrucciones que especifica el diseñador. Muchas aplicaciones requieren una respuesta más rápida de la que puede dar una arquitectura de microcomputadora/DSP, por lo que en estos casos debe utilizarse un circuito digital convencional. En el mercado de la tecnología actual que avanza con gran rapidez, la mayoría de los sistemas digitales no se están implementando con chips de dispositivos lógicos estándar que contienen sólo compuertas simples o funciones tipo MSI, sino que se utilizan dispositivos lógicos programables, los cuales contienen los circuitos necesarios para crear funciones lógicas. Estos dispositivos no se programan con una lista de instrucciones, como una computadora o un DSP, sino que su hardware interno se configura mediante puntos que se conectan y se desconectan de manera electrónica en el circuito.

¿Por qué los PLDs han acaparado una gran parte del mercado? Con los dispositivos programables puede obtenerse la misma funcionalidad con un CI que si se utilizan varios chips lógicos individuales. Esta característica implica menos espacio para los tableros, menos consumo de energía, una mayor confiabilidad, menos inventario y una reducción en el costo total de fabricación.

En los capítulos anteriores usted se familiarizó con el proceso de programar un PLD mediante el uso de AHDL o VHDL. Al mismo tiempo, aprendió acerca de todos los bloques fundamentales de los sistemas digitales. Hasta ahora, hemos presentado las implementaciones en PLD de circuitos digitales como una "caja negra". No nos preocupamos por lo que pasaba dentro del PLD para que pudiera funcionar. Ahora que comprende el funcionamiento de todos los circuitos dentro

de la caja negra, es tiempo de encender las luces ahí dentro y ver cómo funciona. Esto le permitirá tomar las mejores decisiones a la hora de seleccionar y aplicar un PLD para resolver un problema. En este capítulo daremos un vistazo a los diversos tipos de hardware disponible para diseñar sistemas digitales. Después le presentaremos las arquitecturas de varias familias de PLDs.

13-1 ÁRBOL FAMILIAR DE LOS SISTEMAS DIGITALES

Aunque el principal objetivo de este capítulo es investigar las arquitecturas de los PLDs, también es conveniente analizar las diversas opciones de hardware disponibles para los diseñadores de sistemas digitales, ya que podremos obtener una mejor percepción de las alternativas de hardware digital de la actualidad. Por lo general, se puede lograr la funcionalidad deseada de un circuito mediante el uso de varios tipos distintos de hardware digital. A lo largo de este libro hemos descrito los dispositivos lógicos estándar y también la manera en que pueden utilizarse los dispositivos lógicos programables para crear los mismos bloques funcionales. Las microcomputadoras y los sistemas de DSP también pueden aplicarse a menudo con la secuencia necesaria de instrucciones (es decir, el programa de la aplicación) para producir la función deseada de un circuito. Las decisiones de ingeniería de diseño deben tomar en cuenta muchos factores, incluyendo la velocidad de operación necesaria para el circuito, el costo de fabricación, el consumo de energía del sistema, su tamaño, la cantidad de tiempo disponible para diseñar el producto, etcétera. De hecho, la mayoría de los diseños digitales complejos incluyen una mezcla de distintas categorías de hardware. Hay que tomar en cuenta muchas concesiones entre los diversos tipos de hardware para diseñar un sistema digital.

Un árbol familiar de los sistemas digitales (vea la figura 13-1) en el que se muestre la mayoría de opciones de hardware disponibles en la actualidad puede ser útil para ordenar las diversas categorías de dispositivos digitales. La representación gráfica en la figura no muestra todos los detalles; algunos de los tipos de dispositivos más complejos tienen muchas subcategorías adicionales y se omitieron los tipos de dispositivos obsoletos por cuestión de claridad. Las principales categorías de sistemas digitales incluyen la lógica estándar, los circuitos integrados de aplicación específica (ASIC) y los dispositivos de microprocesador/procesador digital de señales (DSP).

La primera categoría de dispositivos de **lógica estándar** se refiere a los componentes digitales con funcionalidad básica (compuertas, flip-flops, decodificadores, multiplexores, registros, contadores, etc.) que están disponibles en chips SSI y MSI.

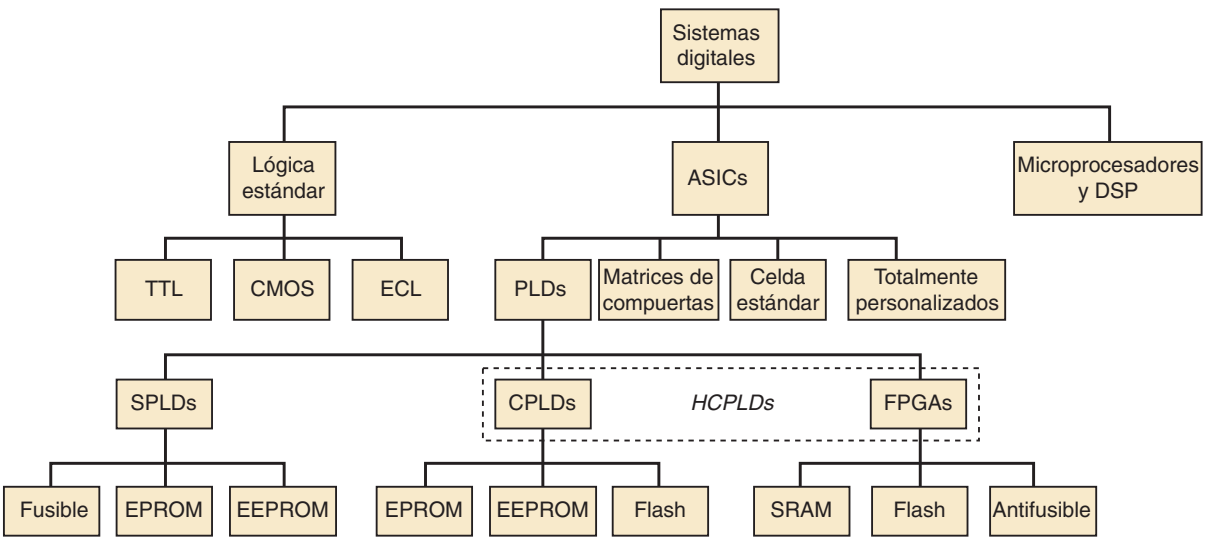


FIGURA 13-1 Árbol familiar de los sistemas digitales.

Estos dispositivos se han utilizado por muchos años (algunos por más de 30) para diseñar sistemas digitales complejos. Una desventaja obvia es que el sistema puede consistir literalmente de cientos de dichos chips. Estos dispositivos económicos aún pueden ser útiles si nuestro diseño no es muy complejo. Como vimos en el capítulo 8, hay tres familias principales de dispositivos lógicos estándar: TTL, CMOS y ECL. TTL es una tecnología madura, la cual consiste de numerosas subfamilias que se han desarrollado a lo largo de muchos años de uso. En la actualidad muy pocos diseños aplican lógica TTL, pero muchos, muchos sistemas digitales aún contienen dispositivos TTL. CMOS es la familia de dispositivos lógicos estándar más popular actualmente, en gran parte debido a su bajo consumo de energía. La tecnología ECL, desde luego, se aplica en los diseños de mayor velocidad. Los dispositivos lógicos estándar aún están disponibles para el diseñador digital, pero si la aplicación es muy compleja se necesitarán muchos chips SSI/MSI. Esa solución no es muy atractiva para nuestras necesidades de diseño actuales.

La categoría de **microprocesadores/procesadores digitales de señales (DSP)** es un enfoque mucho muy distinto para el diseño de sistemas digitales. Estos dispositivos contienen los diversos tipos de bloques funcionales que hemos visto a lo largo de este libro. Con los sistemas de microcomputadora/DSP, los dispositivos pueden controlarse en forma electrónica y los datos pueden manipularse mediante la ejecución de un programa de instrucciones que se haya escrito para esa aplicación. Puede lograrse una gran flexibilidad con los sistemas de microcomputadora/DSP, ya que todo lo que hay que hacer es cambiar el programa. La principal desventaja de esta categoría de sistemas digitales es la velocidad. *Siempre será más rápido utilizar una solución de hardware para su diseño de sistemas digitales que una solución de software.*

La tercera categoría principal de sistemas digitales es la de los **circuitos integrados de aplicación específica (ASIC)**. Esta amplia categoría representa la solución moderna de diseño de hardware para los sistemas digitales. Como su acrónimo lo indica, un circuito integrado está diseñado para implementar una aplicación específica deseada. Hay cuatro subcategorías de dispositivos ASIC disponibles para crear sistemas digitales: dispositivos lógicos programables, matrices de compuertas, de celda estándar y totalmente personalizados.

Los **dispositivos lógicos programables (PLDs)**, que algunas veces se denominan dispositivos lógicos programables en campo (FPLDs), pueden configurarse a la medida para crear cualquier circuito digital deseado, desde simples compuertas lógicas hasta sistemas digitales complejos. En capítulos anteriores vimos muchos ejemplos de diseños con PLDs. Esta opción de la categoría ASIC para el diseñador es muy distinta de las otras tres subcategorías. Con una inversión de capital relativamente pequeña, cualquier compañía puede comprar el software y hardware de desarrollo necesario para programar PLDs para sus diseños digitales. Por otro lado, para obtener un ASIC de matriz de compuertas, de celda estándar o totalmente personalizado se requiere que la mayoría de las compañías firmen contrato con una fundición de CIs para fabricar el chip de CI deseado. Esta opción puede ser extremadamente costosa y, por lo general, se requiere que la compañía compre un gran volumen de piezas para que sea efectiva en costo.

Las **matrices de compuertas** son circuitos ULSI que ofrecen cientos o miles de compuertas. Las funciones lógicas deseadas se crean mediante las interconexiones de estas compuertas prefabricadas. Una máscara de diseño personalizado para la aplicación específica determina las interconexiones de las compuertas, en forma muy parecida a los datos almacenados en una ROM programable por máscara. Por esta razón, es muy común que se les conozca como matrices de compuertas programadas por máscara (MPGAs). En forma individual, estos dispositivos son menos costosos que los PLDs con una cuenta de compuertas comparable, pero el proceso de programación personalizado por el fabricante del chip es muy costoso y requiere un plazo de producción muy largo.

Los **ASICs de celda estándar** utilizan bloques fundamentales de funciones lógicas predefinidas (conocidos como celdas) para crear el sistema digital deseado. La distribución del CI de cada celda se diseña previamente y una biblioteca de celdas disponibles se almacena en una base de datos de computadora. Las celdas necesarias se distribuyen para la aplicación deseada y se determinan las interconexiones

entre las celdas. Los costos de diseño para los ASICs de celda estándar son incluso mayores que para las MPGAs, debido a que todas las máscaras de fabricación de CIs que definen los componentes y las interconexiones deben diseñarse a la medida. También se requiere un plazo de producción mayor para la creación de las máscaras adicionales. Las celdas estándar tienen una ventaja considerable en comparación con las matrices de compuertas. Las funciones basadas en la celda se diseñan de manera que sean más pequeñas que sus funciones equivalentes en las matrices de compuertas, lo cual permite una operación de mayor velocidad y un menor costo de fabricación.

Las **ASICs totalmente personalizadas** se consideran como la última opción de ASIC. Como su nombre lo indica, todos los componentes (transistores, resistencias y capacitores) y las interconexiones entre ellos se elaboran a la medida por el diseñador de CIs. Esto origina CIs que pueden operar a la mayor velocidad posible con la menor área de pastilla (chip de CI individual). Gracias a los tamaños menores de pastilla de CI se pueden acomodar más pastillas en un disco de silicio, lo cual reduce en forma considerable el costo de fabricación para cada CI.

Más sobre PLDs

Como este capítulo está dedicado en gran parte a los PLDs, vamos a profundizar un poco más sobre esa rama del árbol familiar. El desarrollo de la tecnología de PLDs ha estado avanzando en forma continua desde que aparecieron los primeros PLDs hace más de 30 años. Los primeros dispositivos contenían el equivalente de unos cuantos cientos de compuertas, y ahora tenemos piezas que contienen varios millones. Los dispositivos antiguos podían manejar unas cuantas entradas y salidas con capacidades lógicas limitadas. Ahora hay PLDs que pueden manejar cientos de entradas y salidas. Los dispositivos originales podían programarse sólo una vez, y si el diseño cambiaba había que quitar el PLD antiguo del circuito e insertar en su lugar uno nuevo, programado con el diseño actualizado. Con los dispositivos más recientes, el diseño lógico interno puede cambiarse al instante, mientras el chip sigue conectado a un tablero de circuito impreso en un sistema electrónico.

Por lo general, los PLDs pueden clasificarse en tres tipos distintos: **dispositivos lógicos programables simples (SPLDs)**, **dispositivos lógicos programables complejos (CPLDs)** o **matrices de compuertas programables en campo (FPGAs)**. Hay varios fabricantes con muchas familias distintas de dispositivos PLD, por lo que hay muchas variaciones en la arquitectura. Trataremos de hablar sobre las características generales de cada uno de estos tipos, pero debemos hacer una advertencia: las diferencias no son siempre claras y definidas. A menudo la distinción entre CPLDs y FPGAs está un poco borrosa, pues los fabricantes diseñan en forma constante nuevas arquitecturas mejoradas y con frecuencia crean confusión para fines de comercialización. En conjunto, a los CPLDs y FPGAs se les conoce como **dispositivos lógicos programables de alta capacidad (HCPLDs)**. Las tecnologías de programación para los dispositivos PLD se basan en los diversos tipos de memoria semiconductora. A medida que se han desarrollado nuevos tipos de memoria, la misma tecnología se ha aplicado a la creación de nuevos tipos de dispositivos PLD.

La cantidad de recursos lógicos disponibles es la principal característica de diferenciación entre los SPLDs y los HCPLDs. Hoy en día, los SPLDs son dispositivos que, por lo general, contienen el equivalente de 600 compuertas o menos, mientras que los HCPLDs contienen miles y cientos de miles de compuertas disponibles. Los recursos de interconexión de señales programables internas están mucho más limitados con los SPLDs; además, estos dispositivos, por lo general, son menos complicados y mucho más económicos que los HCPLDs. Muchas aplicaciones digitales pequeñas sólo necesitan los recursos de un SPLD. Por otro lado, los HCPLDs son capaces de proporcionar los recursos de circuito para sistemas digitales complejos completos, y cada año se diseñan dispositivos HCPLD más grandes y sofisticados.

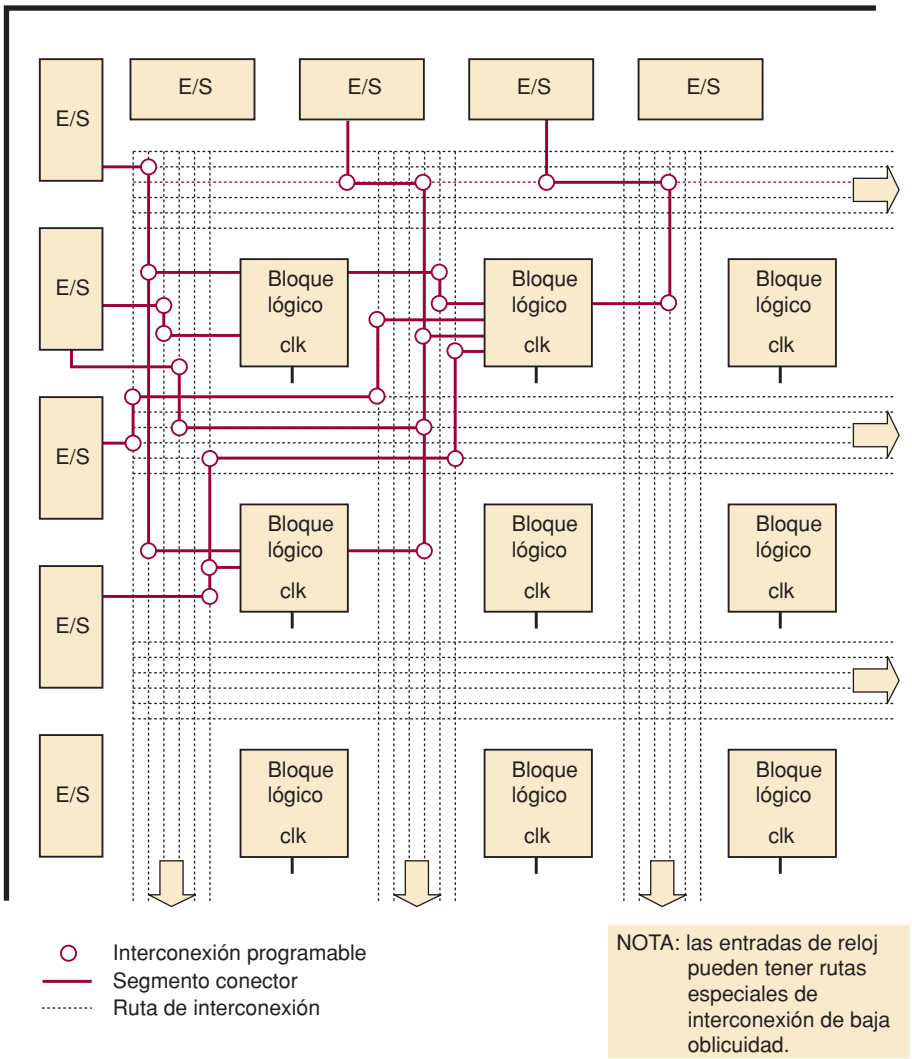
La clasificación de SPLDs incluye los primeros dispositivos PLD. La cantidad de recursos lógicos contenida en los primeros PLDs puede ser pequeña en comparación con los estándares de hoy, pero representaban un considerable avance tecnológico en cuanto a su habilidad de crear un CI personalizado que pudiera sustituir

varios dispositivos lógicos estándar. A través de los años, numerosos avances en los semiconductores han creado distintos tipos de SPLDs. El primer tipo de PLD en ganarse el interés de los diseñadores de circuitos se programaba literalmente quemando fusibles seleccionados en la matriz de programación, para que se abrieran. Los fusibles que se dejaban intactos en estos dispositivos **programables una sola vez (OTP)** proporcionaban las conexiones eléctricas para que los circuitos AND/OR produjeran las funciones deseadas. Este dispositivo lógico se basaba en los enlaces de fusible de la tecnología de memoria PROM (vea la sección 12-7) y se le conocía como arreglo lógico programable (PLA). En realidad los PLDs no obtuvieron una amplia aceptación con los diseñadores digitales sino hasta finales de la década de 1970, cuando se introdujo un dispositivo conocido como **matriz lógica programable (PAL)**. Los enlaces de fusible programables en una PAL se utilizan para determinar las conexiones de entrada para un conjunto de compuertas AND que se conectan a compuertas OR fijas. Con el desarrollo de la PROM borrable mediante luz ultravioleta llegaron los PLDs basados en EPROM a mediados de la década de 1980, y después de ellos llegaron los PLDs que utilizaban tecnología de borrado mediante electricidad (EEPROM).

Los CPLDs son dispositivos que, por lo general, combinan una matriz de dispositivos tipo PAL en el mismo chip. Los bloques lógicos en sí son circuitos lógicos programables con compuertas AND/OR fijas, los cuales tienen menos términos de productos disponibles que la mayoría de los dispositivos PAL. Por lo común, cada bloque lógico (más conocido como **macrocelda**) puede manejar muchas variables de entrada, y los recursos de enrutamiento de señales lógicas programables internos tienden a ser muy uniformes a lo largo del chip, con lo cual se producen retrasos consistentes en las señales. Cuando se necesitan más términos de productos, pueden compartirse las compuertas entre los bloques lógicos, o varios bloques lógicos pueden combinarse para implementar la expresión. A menudo, el flip-flop que se utiliza para implementar el registro en la macrocelda puede configurarse para la operación en D, JK, T (conmutación) o SR. Las terminales de entrada y salida para algunas arquitecturas CPLD se asocian con una macrocelda específica y es común que las macroceldas adicionales se oculten (es decir, que no se conecten a una terminal). Otras arquitecturas CPLD pueden tener bloques de E/S independientes con registros integrados, que pueden utilizarse para fijar los datos entrantes o salientes. Todas las tecnologías de programación que se utilizan en los dispositivos CPLD son no volátiles e incluyen a EPROM, EEPROM y Flash, de las cuales EEPROM es la más común. Las tres tecnologías pueden borrarse y reprogramarse.

Las FPGAs también tienen algunas características fundamentales que se comparten. Por lo general, consisten de muchos módulos lógicos programables relativamente pequeños e independientes, los cuales pueden interconectarse para crear funciones más grandes. Es común que cada módulo pueda manejar sólo hasta cuatro o cinco variables de entrada. La mayoría de los módulos lógicos FPGA utiliza el método de **tabla de búsqueda (LUT)** para crear las funciones lógicas deseadas. Una tabla de búsqueda funciona justo igual que una tabla de verdad, en la cual la salida puede programarse para crear la función combinacional deseada, almacenando el 0 o 1 apropiado para cada combinación de entrada. Los recursos de enrutamiento de la señal programable dentro del chip tienden a ser muy variados, con muchas longitudes de ruta distintas disponibles. Los retrasos de la señal producidos por un diseño dependen del enrutamiento actual de la señal seleccionado por el software de programación. Los módulos lógicos también contienen registros programables y no están asociados con ninguna terminal de E/S, sino que cada terminal de E/S se conecta a un bloque de entrada/salida programable que, a su vez, se conecta a los módulos lógicos con las líneas de enrutamiento seleccionadas. Los bloques de E/S pueden configurarse para proporcionar capacidad de entrada, de salida o bidireccional, y pueden utilizarse los registros integrados para fijar los datos entrantes o salientes. En la figura 13-2 se muestra una arquitectura general de las FPGAs. Todos los bloques lógicos y los de entrada/salida pueden programarse para implementar casi cualquier circuito lógico. Las interconexiones programables se realizan mediante líneas que pasan a través de las filas y columnas en los canales entre los bloques lógicos. Algunas FPGAs incluyen bloques extensos de memoria RAM; otras no.

FIGURA 13-2
Arquitectura de FPGA.



Las tecnologías de programación que se utilizan en los dispositivos FPGA incluyen SRAM, Flash y antifusible, de las cuales SRAM es la más común. Los dispositivos basados en SRAM son volátiles y, por lo tanto, requieren que el dispositivo FPGA se reconfigure (programe) cuando se enciende. La información de programación que define la manera en que funciona cada bloque lógico, cuáles bloques de E/S son entradas y salidas, y cómo se interconectan los bloques, se almacena en cierto tipo de memoria externa que se descarga en la FPGA basada en SRAM cuando se aplica la energía. Los dispositivos antifusibles son programables una sola vez y, por lo tanto, son no volátiles. La tecnología de la memoria antifusible no se utiliza en la actualidad para los dispositivos de memoria pero, como su nombre lo implica, es el opuesto de la tecnología de fusible. En lugar de abrir un enlace de fusible para evitar una conexión de señal, se crea un corto eléctrico en una capa aislante entre las interconexiones para producir una conexión de señal. Los dispositivos antifusible se programan mediante un programador de dispositivos, ya sea por el usuario final o por la fábrica o distribuidor.

Las diferencias en la arquitectura entre CPLDs y FPGAs, entre los distintos fabricantes de HCPLDs y entre las distintas familias de dispositivos de un solo fabricante pueden afectar a la eficiencia de la implementación de diseño para una aplicación específica. Usted podría preguntar, "¿La arquitectura de esta familia de PLDs es la que mejor se adapta a mi aplicación?". No obstante, es muy difícil predecir qué arquitectura podría ser la mejor opción a utilizar para un sistema digital complejo. Sólo puede utilizarse una parte de las compuertas disponibles. ¿Quién puede

saber cuántas compuertas equivalentes se necesitarán para un diseño extenso? La idea básica de los recursos de enrutamiento de señales puede afectar a la cantidad de recursos lógicos del PLD que puedan utilizarse. Las interconexiones segmentadas que se encuentran con frecuencia en las FPGAs pueden producir retrasos más cortos entre los bloques lógicos adyacentes, pero también pueden producir retrasos más largos entre los bloques que se encuentran más separados de los que se producirían debido al tipo continuo de interconexión que se encuentra en la mayoría de los CPLDs. No existe una respuesta fácil para su pregunta, pero cada uno de los fabricantes de HCPLDs le darán una respuesta de todas formas: ¡su producto es el mejor!

Como puede ver, el campo de los PLDs es bastante amplio y cambia constantemente. Ahora deberá tener un conocimiento básico acerca de los diversos tipos y tecnologías necesarias para interpretar las hojas técnicas de los PLDs y aprender más acerca de ellos.

PREGUNTAS DE REPASO

1. ¿Cuáles son las tres categorías principales de los sistemas digitales?
2. ¿Cuál es la mayor desventaja de un diseño de microprocesador/DSP?
3. ¿Qué significa ASIC?
4. ¿Cuáles son los cuatro tipos de ASICs?
5. ¿Qué son los HCPLDs?
6. ¿Cuáles son las dos principales diferencias entre CPLDs y FPGAs?
7. ¿A qué se refiere la volatilidad?

13-2 FUNDAMENTOS DE LOS CIRCUITOS DE UN PLD

En la figura 13-3 se muestra un dispositivo PLD simple. Cada una de las cuatro compuertas OR pueden producir una salida que sea una función de las dos variables de entrada, A y B. Cada función de salida se programa con los fusibles ubicados entre las compuertas AND y cada una de las compuertas OR.

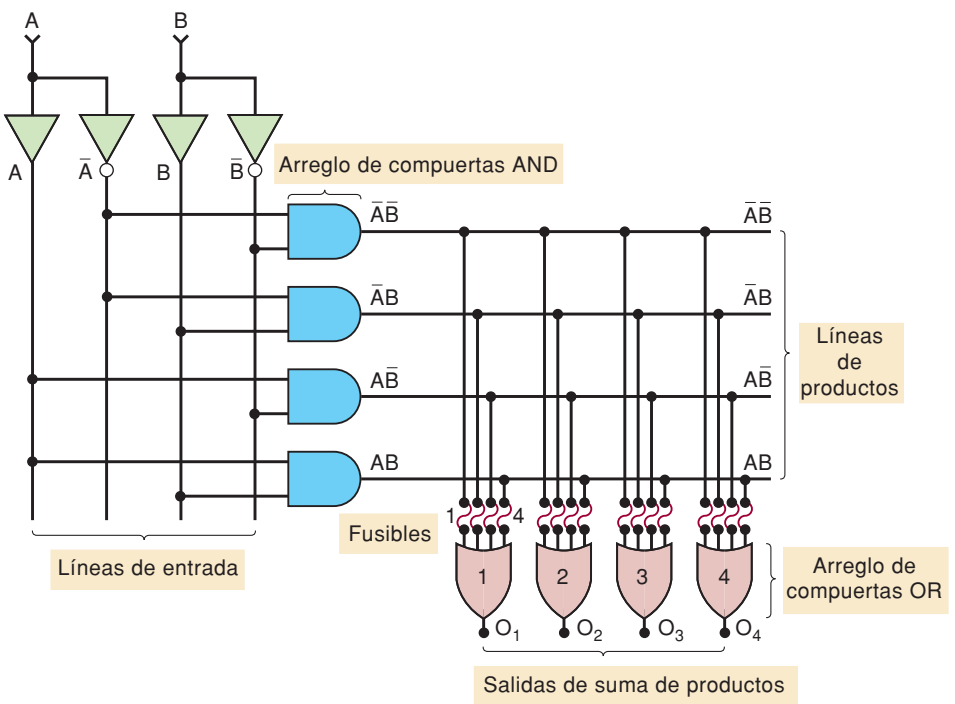


FIGURA 13-13
Ejemplo de un dispositivo lógico programable.

Cada una de las entradas A y B alimenta a un búfer no inversor y a un búfer inversor para producir las formas verdadera e invertida de cada variable. Éstas son las *líneas de entrada* para el arreglo de compuertas AND. Cada compuerta AND se conecta a dos líneas distintas de entrada para generar un producto único de las variables de entrada. A las salidas de las compuertas AND se les llama *líneas de productos*.

Cada una de las líneas de productos se conecta a una de las cuatro entradas de cada compuerta OR, a través de un enlace de fusible. Cuando al principio todos los enlaces están intactos, la salida de cada compuerta OR será un 1 constante. He aquí la prueba:

$$\begin{aligned} O_1 &= \overline{A}\,\overline{B} + \overline{A}B + A\overline{B} + AB \\ &= \overline{A}(\overline{B} + B) + A(\overline{B} + B) \\ &= \overline{A} + A = 1 \end{aligned}$$

Cada una de las cuatro salidas O_1, O_2, O_3 y O_4 pueden *programarse* para ser cualquier función de A y B, para ello se queman en forma selectiva los fusibles apropiados. Los PLDs están diseñados de manera que una entrada OR quemada actúe como un 0 lógico. Por ejemplo, si quemamos los fusibles 1 y 4 en la compuerta OR 1, la salida O_1 se convierte en

$$O_1 = 0 + \overline{A}B + A\overline{B} + 0 = \overline{A}B + A\overline{B}$$

Podemos programar cada una de las salidas OR con cualquier función deseada en forma similar. Una vez programadas todas las salidas, el dispositivo generará en forma permanente las funciones de salida seleccionadas.

Simbología de PLDs

El ejemplo en la figura 13-3 sólo tiene dos variables de entrada y el diagrama del circuito esta un tanto saturado de líneas y conexiones. Imagine lo complejo que sería el diagrama para PLDs con muchas más entradas. Por esta razón, los fabricantes de PLDs han adoptado una representación simbólica de los circuitos internos de estos dispositivos.

La figura 13-4 muestra el mismo circuito de PLD que la figura 13-3, sólo que utiliza los símbolos simplificados. En primer lugar, observe que los búferes de entrada se representan como un búfer individual con dos salidas, una invertida y otra no invertida. A continuación observe que se muestra una *línea individual* que va hacia la compuerta AND para representar las cuatro entradas. Cada vez que la línea de las filas cruza una columna, representa una entrada separada para la compuerta AND. Las conexiones que van desde las líneas de las variables de entrada hasta las entradas de las compuertas AND se indican como puntos. Un punto significa que esa conexión con la entrada de la compuerta AND es fija y no puede cambiarse. A primera instancia parece como si las variables de entrada estuvieran conectadas unas con otras. Es importante tener en cuenta que éste *no* es el caso, ya que la línea de fila individual representa *múltiples* entradas para la compuerta AND.

Las entradas para cada una de las compuertas OR también se designan mediante una línea individual que representa a las cuatro entradas. Una X representa un fusible intacto que conecta una línea de producto con una entrada de la compuerta OR. La ausencia de una X (o un punto) en cualquier intersección representa un fusible quemado. Para las entradas de las compuertas OR se asume que los fusibles quemados (entradas no conectadas) están en BAJO, y para las compuertas AND los fusibles quemados están en ALTO. En este ejemplo las salidas se programan como

$$\begin{aligned} O_1 &= \overline{A}B + A\overline{B} \\ O_2 &= AB \\ O_3 &= 0 \\ O_4 &= 1 \end{aligned}$$

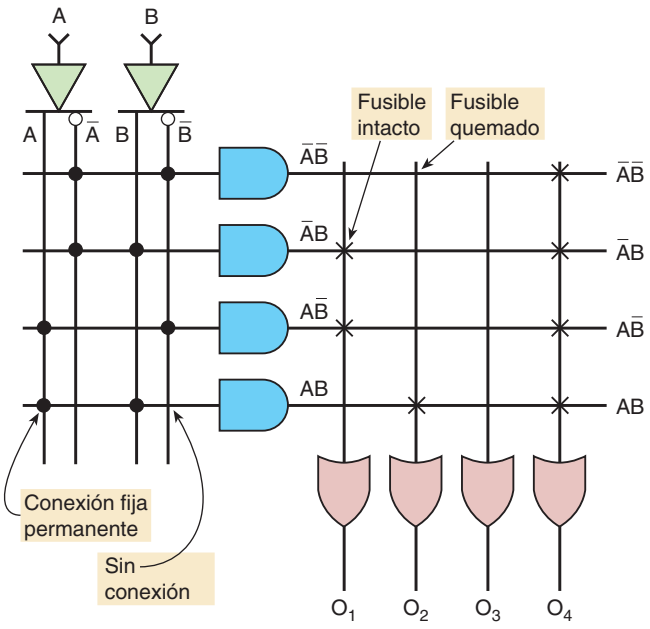


FIGURA 13-4
Simbología simplificada de un PLD.

PREGUNTAS DE REPASO

1. ¿Qué es un PLD?
2. ¿Cuál sería la salida O_1 en la figura 13-3 si se quemaran los fusibles 1 y 2?
3. ¿Qué representa una X en un diagrama de PLD?
4. ¿Qué representa un punto en un diagrama de PLD?

13-3 ARQUITECTURAS DE PLDS

El concepto de los PLDs ha originado muchos diseños arquitectónicos distintos de los circuitos internos de estos dispositivos. En esta sección exploraremos algunas de las diferencias básicas en la arquitectura.

PROMs

La arquitectura de los circuitos programables de la sección anterior implica la programación de las conexiones con la compuerta OR. Las compuertas AND se utilizan para decodificar todas las posibles combinaciones de las variables de entrada, como se muestra en la figura 13-5(a). Para cualquier combinación de entrada dada se activa la correspondiente fila (cambia a ALTO). Si la entrada OR se conecta a esa fila, aparece un nivel ALTO en la salida OR. Si la entrada no se conecta, aparece un nivel BAJO en la salida OR. ¿Le suena esto familiar? Consulte la figura 12-9. Si considera las variables de entrada como entradas de dirección y los fusibles intactos/quemados como 1s y 0s almacenados, deberá reconocer la arquitectura de una PROM.

La figura 13-5(b) muestra cómo se programaría la PROM para generar cuatro funciones lógicas especificadas. Sigamos el procedimiento para la salida $O_3 = AB + C\,D$. El primer paso es construir una tabla de verdad que muestre el nivel deseado de la salida O_3 para todas las posibles combinaciones de entrada (tabla 13-1).

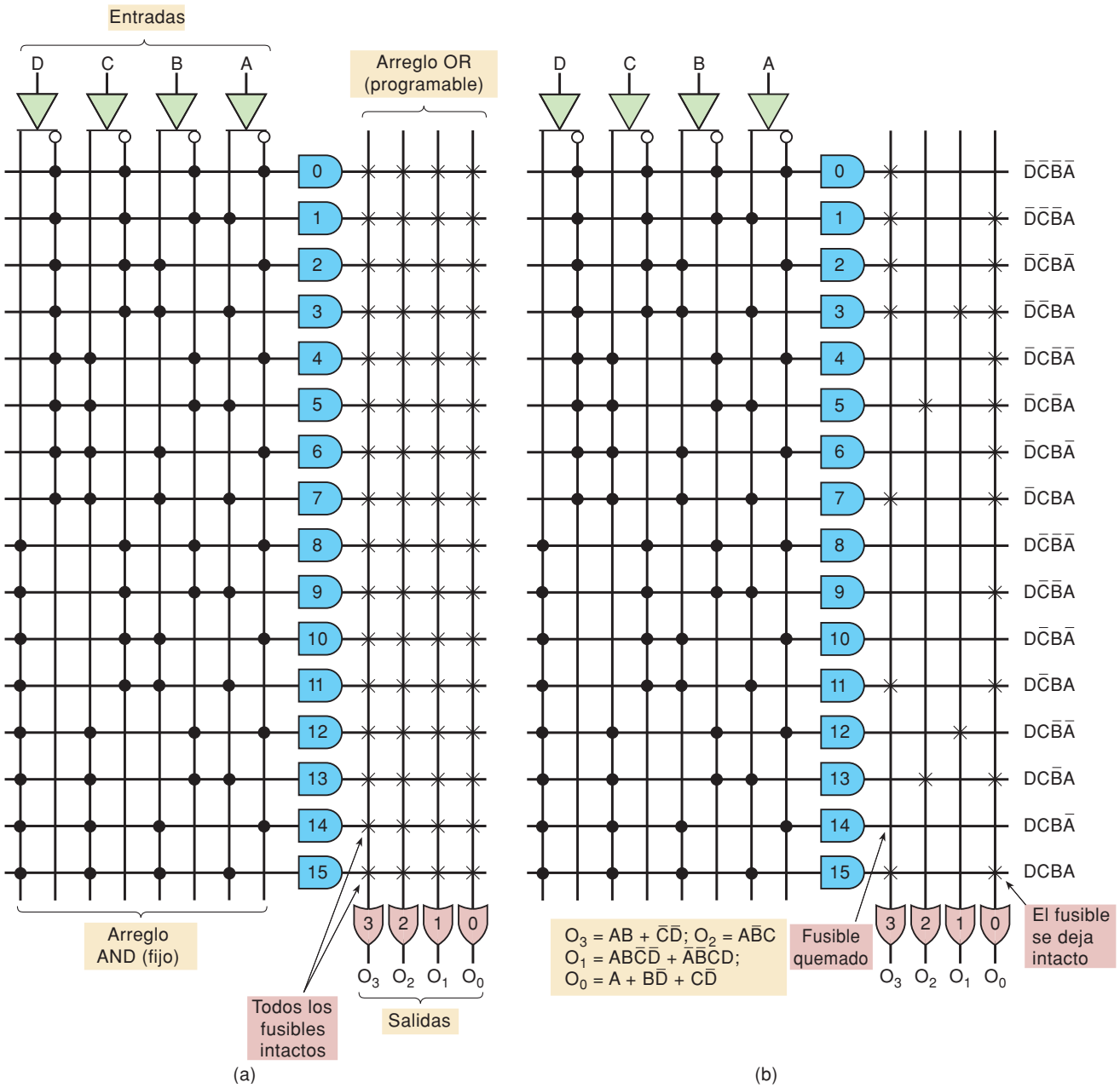


FIGURA 13-5 (a) La arquitectura de la PROM la hace adecuada para los PLDs; (b) los fusibles se queman para programar las salidas para las funciones dadas.

A continuación se anotan los productos AND para los casos en los que la salida debe ser un 1. La salida O_3 debe ser la suma OR de estos productos. Por ende, sólo los fusibles que conectan estos términos de productos con las entradas de la compuerta OR 3 deben dejarse intactos. Todos los demás se deben quemar, como se indica en la figura 13-5(b). Este mismo procedimiento se sigue para determinar el estado de los fusibles en las demás entradas de las compuertas OR.

La PROM puede generar cualquier función lógica posible de las variables de entrada, ya que genera todos los términos de productos AND posibles. En general, cualquier aplicación que requiera que todas las combinaciones de entrada estén disponibles es un buen candidato para una PROM. No obstante, las PROMs se vuelven imprácticas cuando debe acomodarse un gran número de variables de entrada,

TABLA 13-1

D	C	B	A	O_3		
0	0	0	0	1	→	$\overline{D}\,\overline{C}\,\overline{B}\,\overline{A}$
0	0	0	1	1	→	$\overline{D}\,\overline{C}\,\overline{B}A$
0	0	1	0	1	→	$\overline{D}\,\overline{C}B\,\overline{A}$
0	0	1	1	1	→	$\overline{D}\,\overline{C}BA$
0	1	0	0	0		
0	1	0	1	0		
0	1	1	0	0		
0	1	1	1	1	→	$\overline{D}CBA$
1	0	0	0	0		
1	0	0	1	0		
1	0	1	0	0		
1	0	1	1	1	→	$D\overline{C}BA$
1	1	0	0	0		
1	1	0	1	0		
1	1	1	0	0		
1	1	1	1	1	→	$DCBA$

debido a que el número de fusibles se duplica para cada variable de entrada que se agregue.

Decir que una PROM es un PLD es sólo cuestión de semántica. Usted ya sabía que una PROM es programable y que es un dispositivo lógico. Ésta es sólo una manera de utilizar una PROM y de pensar en que su propósito es implementar expresiones lógicas de SOP en vez de almacenar valores de datos en ubicaciones de memoria. El verdadero problema es traducir las ecuaciones lógicas en el mapa de fusibles para una PROM dada. Un compilador lógico de propósito general diseñado para programar SPLDs tiene una lista de dispositivos PROM que puede soportar. Si usted opta por utilizar cualquier EPROM vieja y rebuscada como un PLD, tal vez necesite generar su propio mapa de bits (como solían hacerlo antes), lo cual es muy tedioso.

Matriz lógica programable (PAL)

La arquitectura de la PROM se adapta muy bien a las aplicaciones en las que se requieren todas las posibles combinaciones de entradas para generar las funciones de salida. Algunos ejemplos son los convertidores de código y las tablas de almacenamiento de datos (de búsqueda) que examinamos en el capítulo 12. No obstante, al implementar expresiones de SOP, no hacen un uso muy eficiente de los circuitos. Cada combinación de entradas de dirección debe decodificarse por completo y cada término de producto expandido tiene un fusible asociado que se utiliza para aplicar una operación OR entre todos los términos. Por ejemplo, observe cuántos fusibles se requirieron en la figura 13-5 para programar las expresiones simples de SOP y cuántos términos de productos no se utilizan a menudo. Esto ha originado el desarrollo de una clase de PLDs llamada lógica de matriz programable (PAL). La arquitectura de una PAL difiere un poco de la de una PROM, como se muestra en la figura 13-6(a).

La PAL tiene una estructura de compuertas AND y OR similar a una PROM, pero en la PAL las entradas de las compuertas AND son programables, mientras que las entradas de las compuertas OR son fijas. Esto significa que toda compuerta AND puede programarse para generar cualquier producto deseado de las cuatro variables de entrada y sus complementos. Cada compuerta OR se conecta en forma permanente a sólo cuatro salidas AND. Esto limita cada función de salida a cuatro términos de productos. Si una función requiere más de cuatro términos de productos,

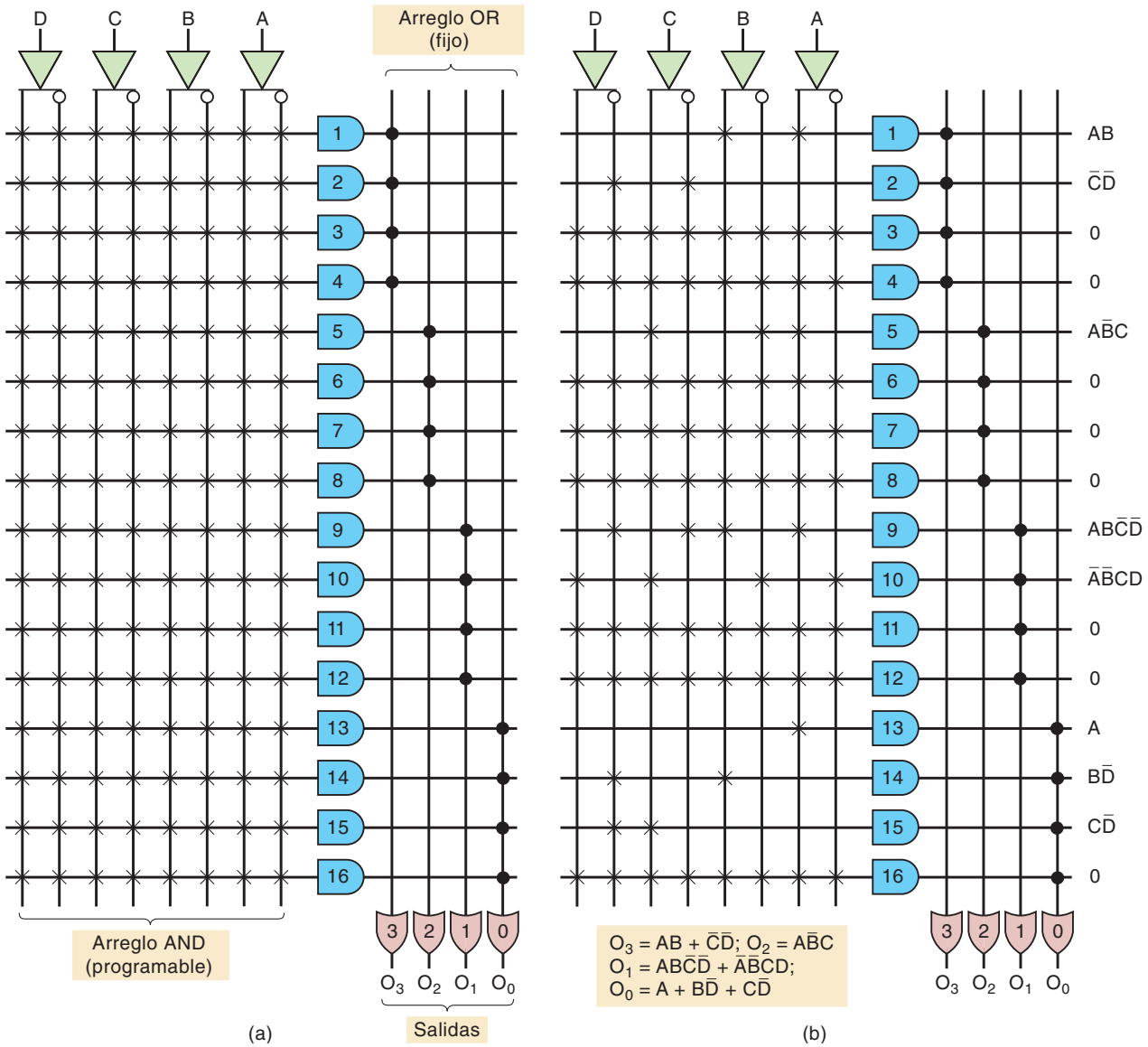


FIGURA 13-6 (a) Arquitectura de una PAL típica; (b) la misma PAL programada para las funciones dadas.

no puede implementarse con esta PAL; tendría que utilizarse una que tuviera más entradas OR. Si se requieren menos de cuatro términos de productos, los que no se necesiten pueden hacerse 0.

La figura 13-6(b) muestra cómo se programa esta PAL para generar cuatro funciones lógicas específicas. Sigamos el procedimiento para la salida $O_3 = AB + \overline{C}\,\overline{D}$. Primero debemos expresar esta salida como la suma OR de cuatro términos, ya que las compuertas OR tienen cuatro entradas. Para ello debemos colocar 0s. Por ende, tenemos que

$$O_3 = AB + \overline{C}\,\overline{D} + 0 + 0$$

A continuación debemos determinar cómo programar las entradas para las compuertas AND 1, 2, 3 y 4 de manera que proporcionen los términos de productos correctos para la compuerta OR 3. Debemos hacer esto término por término. El primer término AB se obtiene dejando intactos los fusibles que conectan las entradas A y B con la compuerta AND 1 y quemando todos los demás fusibles en esa línea.

De igual forma, el segundo término $\overline{C}\,\overline{D}$ se obtiene dejando intactos sólo los fusibles que conectan las entradas $\overline{C}$ y $\overline{D}$ con la compuerta AND 2. El tercer término es 0. Para producir un 0 constante en la salida de la compuerta AND 3 se dejan todos sus fusibles de entrada intactos. Esto produciría una salida de $A\overline{A}B\overline{B}C\overline{C}D\overline{D}$ que, como sabemos, es 0. El cuarto término también es 0, por lo que los fusibles de entrada para la compuerta AND 4 también se dejan intactos.

Las entradas para las otras compuertas AND se programan de manera similar para generar las otras funciones de salida. En especial observe que muchas de las compuertas AND tienen todos sus fusibles de entrada intactos, ya que necesitan generar 0s.

Un ejemplo de un circuito integrado PAL real es el PAL16L8, el cual tiene 10 entradas lógicas y ocho funciones de salida. La salida de cada compuerta OR se conecta en forma fija a siete salidas de compuerta AND, por lo que puede generar funciones que incluyan hasta siete términos. Una característica adicional de esta PAL específica es que seis de las ocho salidas se retroalimentan al arreglo AND, en donde pueden conectarse como entradas para cualquier compuerta AND. Esto la hace muy útil para generar todo tipo de lógica combinacional.

La familia PAL también contiene dispositivos con variaciones de los circuitos de SOP básicos que hemos descrito. Por ejemplo, la mayoría de los dispositivos PAL tienen un búfer triestado que controla la terminal de salida. Otros canalizan el circuito lógico de SOP hacia una entrada de FF D y utilizan una de las terminales como entrada de reloj para aplicar pulsos de reloj a todos los flip-flops de salida en forma síncrona. Estos dispositivos se denominan *PLDs registrados*, ya que las salidas pasan a través de un registro. Un ejemplo es el PAL16R8, el cual tiene hasta seis salidas registradas (que también pueden servir como entradas) más ocho entradas dedicadas.

Matriz lógica programable en campo (FPLA)

La matriz lógica programable en campo (FPLA) se desarrolló a mediados de la década de 1970 como el primer dispositivo lógico programable sin memoria. Se utilizaba como arreglo AND programable y también como arreglo OR programable. Aunque la FPLA es más flexible que la arquitectura PAL, no ha sido tan ampliamente aceptada por los ingenieros. Las FPLAs se utilizan principalmente en el diseño de máquinas de estado, en donde se necesita un gran número de términos de productos en cada expresión de SOP.

PREGUNTAS DE REPASO

1. Verifique que se quemen los fusibles correctos para las funciones O_2, O_1 y O_0 en la figura 13-5(b).
2. Una PAL tiene un arreglo _____ fijo y un arreglo _____ programable.
3. Una PROM tiene un arreglo _____ fijo y un arreglo _____ programable.
4. ¿Cómo cambiaría la ecuación para la salida de O_1 en la figura 13-5(b) si todos los fusibles de la compuerta AND 14 se dejaran intactos?

13-4 EL GAL 16V8 (MATRIZ LÓGICA GENÉRICA)

El GAL 16V8, introducido por Lattice Semiconductor, tiene una arquitectura muy similar a los dispositivos PAL que describimos en la sección anterior. Las PAL estándar de baja densidad son programables una sola vez. Por otro lado, el chip GAL utiliza una matriz EEPROM (ubicada en las intersecciones de fila y columna de la figura 13-7) para controlar las conexiones programables con la matriz AND, lo cual les permite borrarse y reprogramarse por lo menos 100 veces. Además de las compuertas AND y OR utilizadas para producir las funciones de suma de productos, el GAL 16V8 contiene flip-flops opcionales para aplicaciones de registro y contador,

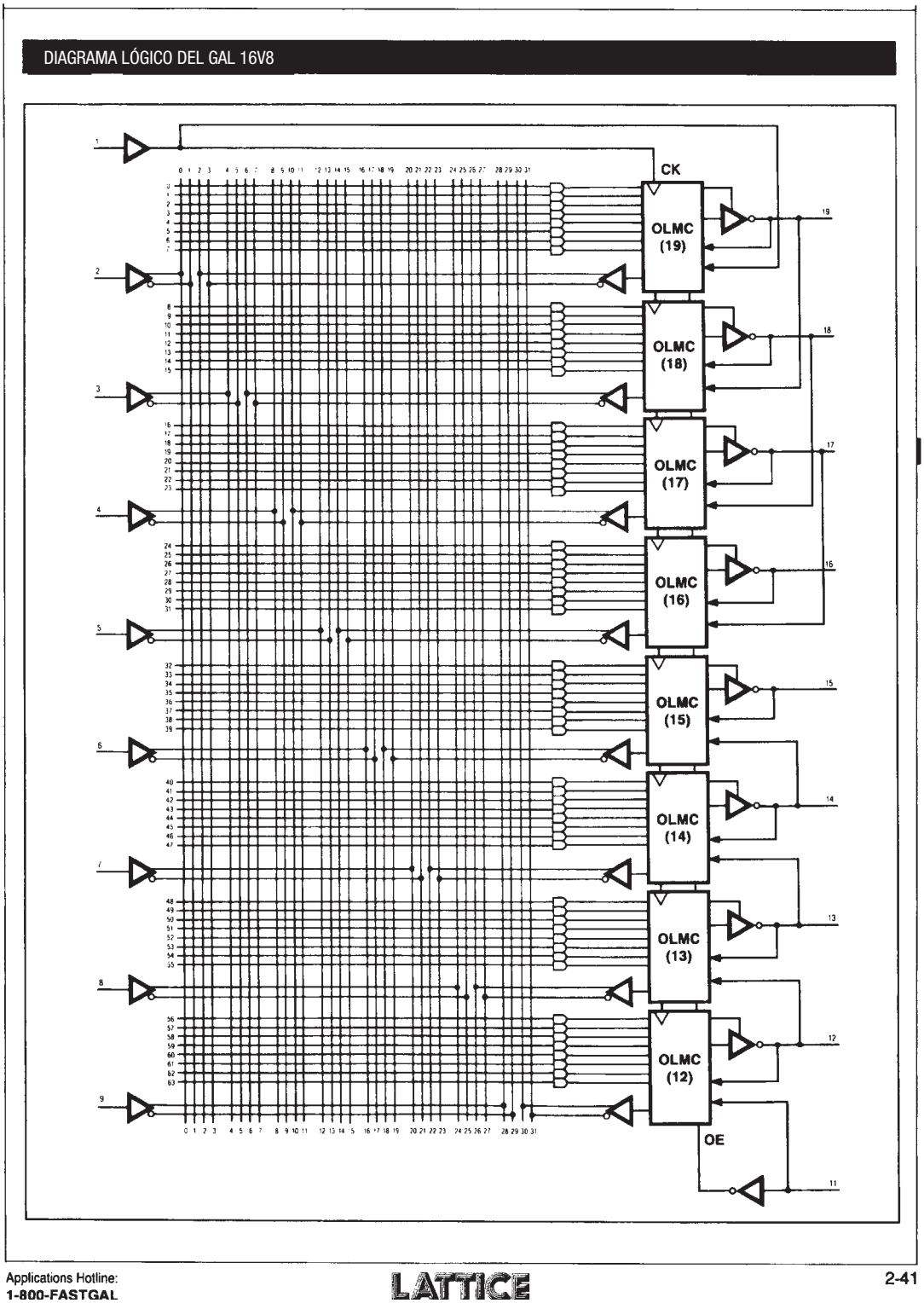


FIGURA 13-7 Diagrama lógico del GAL 16V8. (Reimpreso con permiso de Lattice Semiconductor.)

búferes triestado para las salidas y multiplexadores de control utilizados para seleccionar los diversos modos de operación. En consecuencia, puede utilizarse como un reemplazo genérico con terminales compatibles para la mayoría de los dispositivos PAL. En el arreglo de memoria se designan ubicaciones específicas para controlar las diversas conexiones programables en el chip. Por fortuna no es necesario buscar con detalle en las direcciones de cada ubicación de bit en la matriz. El software de programación se encarga de estos detalles en una forma amigable para el usuario.

En la figura 13-7 se muestra el diagrama lógico completo del GAL 16V8. Este dispositivo tiene ocho terminales de entrada dedicadas (terminales 2–9), dos entradas de función especiales (terminales 1 y 11) y ocho terminales (12–19) que pueden utilizarse como entradas o salidas. Los principales componentes de los dispositivos GAL son la matriz de términos de entrada; las compuertas AND, que generan los productos de los términos de entrada; y las macroceldas de lógica de salida (OLMCs). Observe que cada una de las ocho entradas (terminales 2–9) se conecta en forma directa con una columna de la matriz de términos de entrada. El complemento de cada una de estas entradas se conecta también a una columna de la matriz. Estas terminales deben especificarse siempre como entradas cuando se programa el 16V8. También se retroalimentan un nivel lógico y su complemento desde cada OLMC hacia una columna de la matriz de entrada. Esto explica las 32 variables de entrada (columnas en la matriz de entrada) que pueden programarse como conexiones para las 64 compuertas AND de múltiples entradas.

La flexibilidad del GAL 16V8 está en su macrocelda de lógica de salida programable. Ocho productos distintos (salidas de compuertas AND) se aplican como entradas a cada una de las ocho macroceldas de lógica de salida. Dentro de cada OLMC se aplica una operación OR a todos los productos en conjunto para generar la suma de productos (SOP). Recuerde, del capítulo 4, que cualquier función lógica puede expresarse en forma de SOP. Dentro de la OLMC, la salida de SOP puede enlutarse hacia la terminal de salida para implementar un circuito combinacional, o puede aplicarse mediante pulsos de reloj a un flip-flop D para implementar un circuito de salida registrado.

Para comprender la operación detallada de la OLMC, consulte la figura 13-8. La figura muestra la estructura de la OLMC(n), en donde n es un número del 12 al 19.

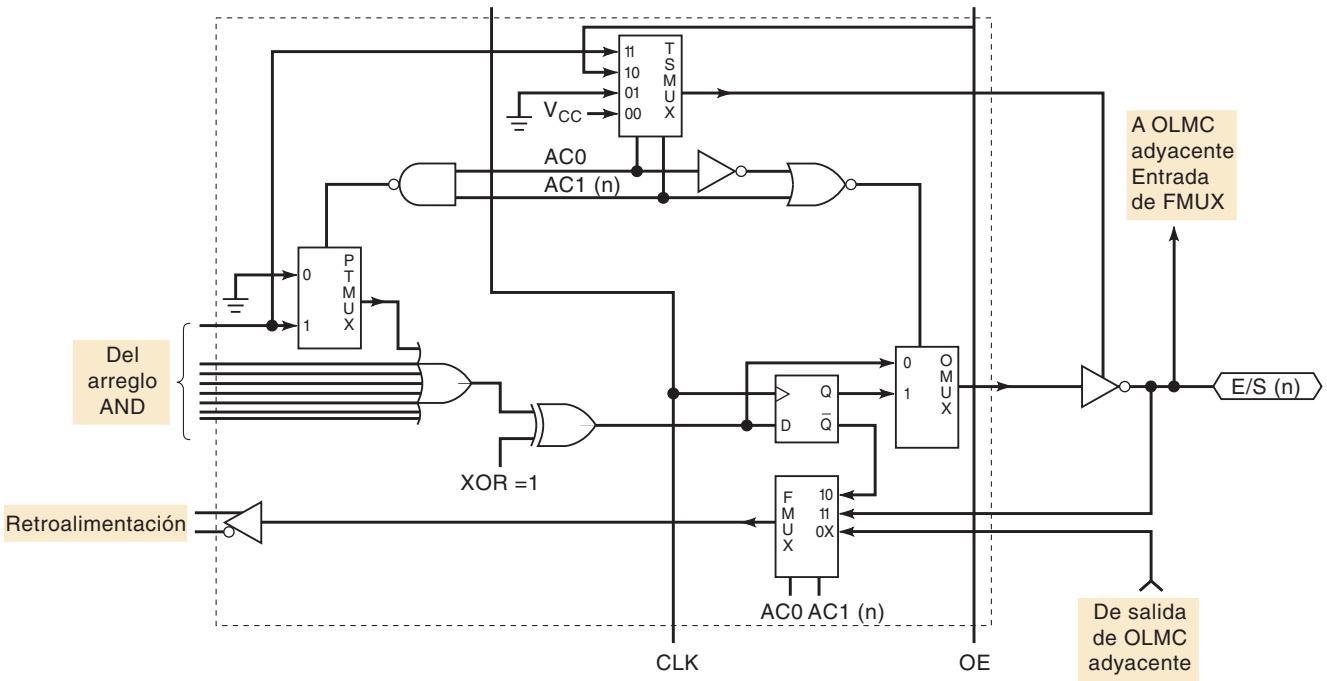
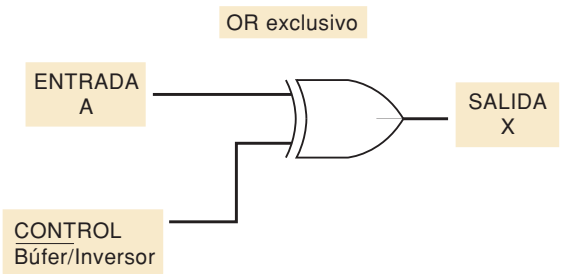


FIGURA 13-8 Macrocelda de lógica de salida para el GAL 16V8. (Reimpreso con permiso de Lattice Semiconductor.)



CONTROL	ENTRADA A	SALIDA X	
0	0	0	No invertida (con búfer)
0	1	1	
1	0	1	Invertida
1	1	0	

FIGURA 13-9 Uso de XOR para complementar en forma selectiva.A

Observe que siete de los productos están conectados en forma incondicional a las entradas de la compuerta OR. El octavo término de producto se conecta a un multiplexor de términos de productos de dos entradas (PTMUX), el cual dirige la octava entrada hacia la compuerta OR. El octavo término de producto también se conecta a una entrada de un mutiplexor de cuatro entradas (TSMUX). La salida del TSMUX habilita el inversor triestado que controla la terminal de salida [E/S(n)]. El multiplexor de salida (OMUX) es un MUX de dos entradas que selecciona entre la salida combinacional (compuerta OR) y la salida registrada (el flip-flop D). Un cuarto MUX selecciona la señal lógica que se retroalimenta hacia la matriz de entrada. A éste se le conoce como multiplexor de *retroalimentación* (FMUX).

Cada uno de estos multiplexores se controla mediante bits programables (AC1 y AC0) en la matriz de EEPROM. Ésta es la forma en que el programador puede alterar la configuración de la OLMC. La entrada a la compuerta XOR es otro bit programable. Esta entrada proporciona la característica de polaridad de salida programable. Recuerde que una compuerta XOR puede utilizarse para complementar una señal lógica en forma selectiva, como se muestra en la figura 13-9. Cuando la línea de control es un 0 lógico, el XOR pasa el nivel lógico en la entrada A sin inversión. Cuando el bit de control es un 1 lógico, el XOR invierte la señal de manera que $X = A$. En la figura 13-8 el bit programable (etiquetado como XOR) es un 1 lógico bajo condiciones de lógica positiva normales. Esto invierte la salida de la compuerta OR, la cual se invierte de nuevo cuando pasa a través del búfer inversor triestado en la salida.

Podemos comprender las diversas opciones de configuración si estudiamos las posibles entradas para cada multiplexor. El TSMUX controla la entrada de habilitación del búfer triestado. Si se selecciona la entrada V_{CC} la salida siempre está habilitada, como una compuerta lógica combinacional estándar. Si se selecciona la entrada a tierra, la salida triestado del inversor siempre se encuentra en su estado de alta impedancia (y permite utilizar la terminal de E/S como una entrada). Otra entrada para el MUX que puede seleccionarse proviene de la entrada *OE;* es decir, la terminal 11. Esto permite habilitar o deshabilitar la salida mediante una señal lógica externa que se aplica en la terminal 11. La última selección de entrada posible es un término de producto para la octava compuerta AND, lo que permite una combinación AND de términos de la matriz de entrada, para habilitar o deshabilitar la salida.

El FMUX selecciona la señal que se retroalimenta hacia la matriz de entrada. En este caso hay tres posibles selecciones. Si se selecciona la entrada del *MUX* que se conecte a una etapa adyacente, o la entrada del *MUX* que está conectada a su propia terminal de E/S de la OLMC se podrá retroalimentar una etapa de salida existente hacia la matriz de entrada en algunos de los modos de operación. Esta característica otorga al GAL 16V8 la habilidad de implementar circuitos secuenciales tales como el circuito de latch de compuerta NAND con acoplamiento cruzado que describimos en el capítulo 5. Esta opción de retroalimentación también permite utilizar una terminal de E/S como entrada dedicada, en vez de salida. Se elije una de estas dos rutas de retroalimentación, dependiendo del MODO en el que se programa el chip. La tercera opción, seleccionar la salida del flip-flop D, permite retroalimentar el estado presente del flip-flop (que puede utilizarse para determinar el siguiente estado) a la

matriz de entrada. Esto permite implementar circuitos secuenciales síncronos, como contadores y registros de desplazamiento.

Con todas estas opciones parecería que debe haber una larga lista de configuraciones posibles. En la práctica todas estas decisiones de configuraciones las realiza el software. En realidad, el GAL 16V8 sólo tiene tres modos distintos: (1) *modo simple*, que se utiliza para implementar lógica combinacional de SOP simple sin salidas triestado; (2) *modo complejo*, que implementa lógica combinacional de SOP con salidas triestado que se habilitan mediante una expresión de productos AND; y (3) *modo registrado*, el cual permite que OLMCs individuales operen en una configuración combinacional con salidas triestado (similar al modo complejo) o en modo síncrono con FFs sincronizados por reloj en D a una señal de reloj común.

El GAL 16V8 es un chip PLD económico y versátil, pero ¿qué pasa si el diseño requiere más recursos de hardware de los que contiene el 16V8? Es posible dividir el diseño en bloques más pequeños que puedan implementarse en varios chips 16V8. Por fortuna hay otros miembros de la familia GAL a elegir. Otro PLD popular de propósito general es el GAL 22V10. Este dispositivo tiene 10 terminales de salida y 12 terminales de entrada en una arquitectura similar, pero no idéntica, al GAL 16V8. Los grupos de términos de productos se suman en forma lógica con una compuerta OR, la cual alimenta a un OLMC. No obstante y a diferencia del 16V8, cada compuerta OR en el 22V10 no combina el mismo número de términos de productos, el cual varía desde ocho hasta dieciséis. Para sacar ventaja de los términos adicionales, debemos asignar las expresiones Booleanas más largas a la terminal correcta de salida. Los flip-flops D contenidos en las OLMCs también tienen capacidades de restablecimiento asíncrono y prestablecimiento síncrono. Ahora hay disponible una versión más reciente del 22V10 (el ispGAL 22V10). Se dice que este dispositivo es *programable en sistema* (ISP). En vez de requerir un programador como en los chips PAL y GAL estándar, se conecta un cable de la PC en forma directa a un conjunto especial de terminales en el dispositivo ISP para realizar la programación.

PREGUNTAS DE REPASO

1. Nombre dos ventajas de los dispositivos GAL en comparación con los dispositivos PAL.
2. Nombre los tres modos de operación para un GAL 16V8.

13-5 EL CPLD EPM7128S DE ALTERA

Vamos a investigar la arquitectura del EPM7128S, un dispositivo basado en EEPROM de la familia de CPLDs MAX7000S de Altera. Este dispositivo se encuentra en varios tableros de desarrollo académico, incluyendo Altera UP2, DeVry eSOC y RSR PLDT-2. La figura 13-10 muestra el diagrama de bloques para esta familia. Las principales estructuras en la familia MAX7000S son los **bloques de matrices lógicas (LABs)** y la **matriz de interconexión programable (PIA)**. Un LAB contiene un conjunto de 16 macroceldas y tiene una apariencia muy similar a la de un dispositivo SPLD individual. Cada macrocelda consiste de un circuito AND/OR programable y de un registro programable (flip-flop). Las macroceldas en un solo LAB pueden compartir recursos lógicos tales como términos de productos comunes o compuertas AND sin usar. El número de macroceldas contenidas en uno de los dispositivos de la familia MAX7000S depende del número de pieza. Como se muestra en la tabla 13-2, el EPM7128S tiene 128 macroceldas distribuidas en ocho LABs.

Las señales lógicas se enrutan entre los LABs a través de la PIA, la cual es un bus global que conecta cualquier fuente de señal a cualquier destino dentro del dispositivo. Todas las entradas para el dispositivo MAX7000S y todas las salidas de las macroceldas alimentan a la PIA. Hasta 36 señales pueden alimentar a cada LAB desde la PIA. Sólo las señales necesarias para producir las funciones requeridas para cualquier LAB se alimentan en ese LAB.

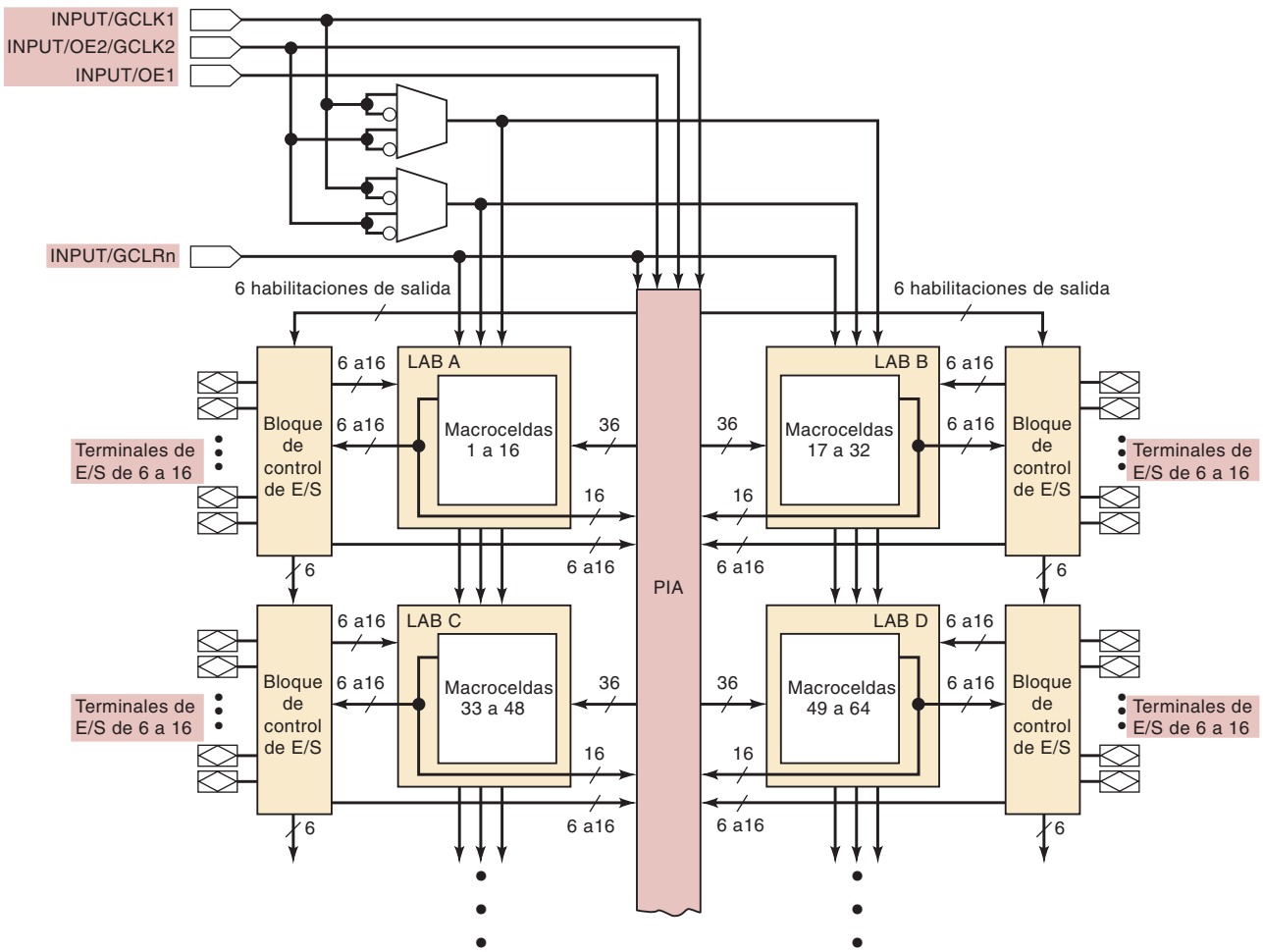


FIGURA 13-10 Diagrama de bloques de la familia MAX7000S. (Cortesía de Altera Corporation.)

Las terminales de E/S en la familia MAX7000S se conectan a macroceldas específicas. El número de terminales de E/S disponibles para el usuario depende del encapsulado del dispositivo. Un EPM7128S en un encapsulado PQFP de 160 terminales tiene 12 terminales de E/S por LAB, más cuatro terminales adicionales de sólo entrada, para un total de 100 terminales. Por otro lado, en un encapsulado PLCC de 84 terminales, el cual se incluye en los tableros de desarrollo antes mencionados, hay ocho terminales de E/S por LAB más las cuatro adicionales, para un total de 68 terminales de E/S. El EPM7128S es un dispositivo programable en el sistema (ISP). La característica ISP utiliza una interfase del grupo de acción de pruebas conjuntas (JTAG), la cual requiere que se dediquen cuatro terminales específicas a la interface de programación y, por lo tanto, no están disponibles para la E/S de usuario

TABLA 13-2 Características de dispositivos de la familia MAX7000S de Altera.

Característica	EPM7032S	EPM7064S	EPM7128S	EPM7160S	EPM7192S	EPM7256S
Compuertas utilizables	600	1250	2500	3200	3750	5000
Macroceldas	32	64	128	160	192	256
LABs	2	4	8	10	12	16
Máximo número de terminales de E/S de usuario	36	68	100	104	124	164

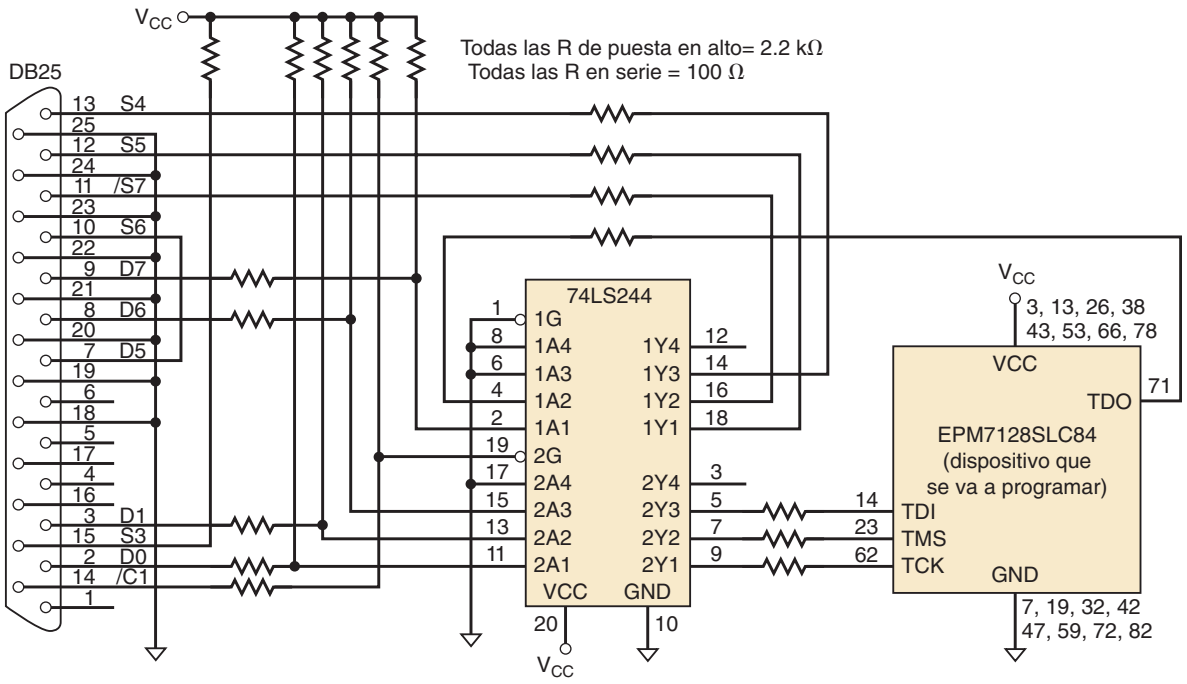


FIGURA 13-11 Interfase de JTAG entre el puerto paralelo de la PC y el EPM7128SLC84.

general. El PLD de destino puede programarse en el sistema a través de las terminales JTAG; para ello se conectan al puerto paralelo de una PC con compuertas controladores, como se muestra en la figura 13-11. Las señales JTAG se nombran TDI (datos de prueba de entrada), TDO (datos de prueba de salida), TMS (selección de modo de prueba) y TCK (reloj de prueba). Con esto se reduce el total de terminales de E/S de usuario para un EPM7128SLC84 (un EPM7128S en un encapsulado PLCC de 84 terminales) a 64. No obstante, pueden utilizarse todas las 68 terminales para la E/S de usuario si el EPM7128SLC84 se programa en un programador de PLDs, en lugar de programarse en el sistema. Al compilar el diseño, debemos indicar si el dispositivo utilizará o no una interface JTAG. En cualquier caso podemos ver que algunas macroceldas no se conectarán en forma directa a las terminales de E/S de usuario. El compilador puede utilizar estas macroceldas para la lógica interna (oculta).

Las cuatro terminales de sólo entrada que se encuentran en los dispositivos de la familia MAX7000S pueden configurarse como señales de control de alta velocidad específicas o como entradas generales de usuario. GCLK1 es la entrada de reloj global primaria para todas las macroceldas en el dispositivo. Se utiliza para aplicar pulsos de reloj a todos los registros en forma síncrona en un diseño. Se localiza en la terminal 83 en un EPM7128SLC84 (vea la figura 13-12). La terminal 2 en este dispositivo es GCLK2 (reloj global secundario). Como alternativa, esta terminal se puede utilizar como habilitación de salida global secundaria (OE2) para cualquier macrocelda designada para tener una salida triestado. La señal de habilitación triestado primaria (OE1) se localiza en la terminal 84. La última de las cuatro señales de control globales es GLCRn en la terminal 1. Esta entrada activa en BAJO puede controlar la señal de borrar asíncrona en cualquier registro de macrocelda. La forma en que se van a utilizar estas terminales para una aplicación específica se asigna en MAX+PLUS II o en Quartus II ya sea en forma automática por el compilador o en forma manual por el diseñador durante el proceso de diseño.

Los bloques de control de E/S (vea la figura 13-10) configuran cada terminal de E/S para la operación de entrada, salida o bidireccional. Todas las terminales de E/S en la familia MAX7000S tienen un búfer de salida triestado que (1) está habilitado o deshabilitado en forma permanente, (2) se controla mediante una de las dos terminales de habilitación de salida globales, o (3) se controla mediante otras entradas o funciones generadas por otras macroceldas. Cuando una terminal de E/S se configura como entrada, la macrocelda asociada se puede utilizar para la lógica oculta.

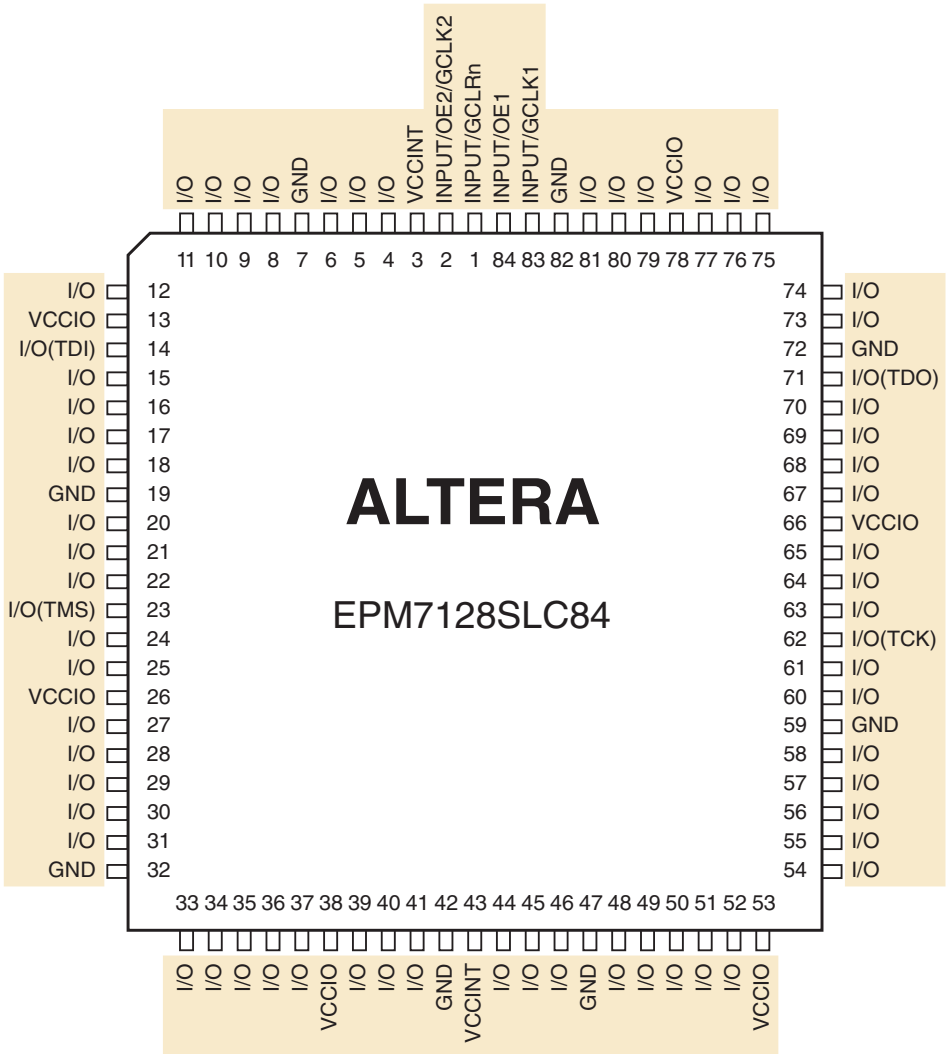


FIGURA 13-12
Diagrama de terminales para el EPM7128SLC84.

Durante la programación en el sistema, las terminales de E/S se harán triestado y se elevarán en forma interna para eliminar conflictos en el tablero.

La figura 13-13 muestra el diagrama de bloques para una macrocelda MAX7000S. Cada macrocelda puede producir una salida combinacional o registrada. El registro (flip-flop) contenido en una macrocelda se desvía para producir una salida combinacional. El circuito de suma de productos programable tiene una apariencia muy parecida a la de un chip GAL. Cada macrocelda puede producir cinco términos de productos. Aunque esto es menor de lo que se encuentra en los chips GAL más simples que vimos antes, a menudo es suficiente para la mayoría de las funciones lógicas. Si se necesitan más términos de productos, el compilador programará en forma automática una macrocelda para pedir prestados hasta cinco términos de productos de cada una de las tres macroceldas adyacentes en el mismo LAB. Esta opción de expansor lógico en paralelo puede proporcionar un total de 20 términos de productos. Las compuertas prestadas ya no las puede utilizar la macrocelda que las prestó. Hay otra opción de expansión disponible en cada LAB, llamada expansores lógicos compartidos. En lugar de agregar más términos de productos, esta opción permite producir un término de producto común una vez y después utilizarlo por varias macroceldas dentro del LAB. Sólo puede utilizarse un término de producto por macrocelda de esta forma, pero con 16 macroceldas por LAB puede haber hasta 16 términos de productos comunes disponibles. El compilador optimiza de manera automática la asignación de los términos de productos disponibles dentro de un LAB, de acuerdo con los requerimientos lógicos del diseño. El uso de cualquiera de las dos opciones de expansor incurre en una pequeña cantidad de retraso de propagación adicional.

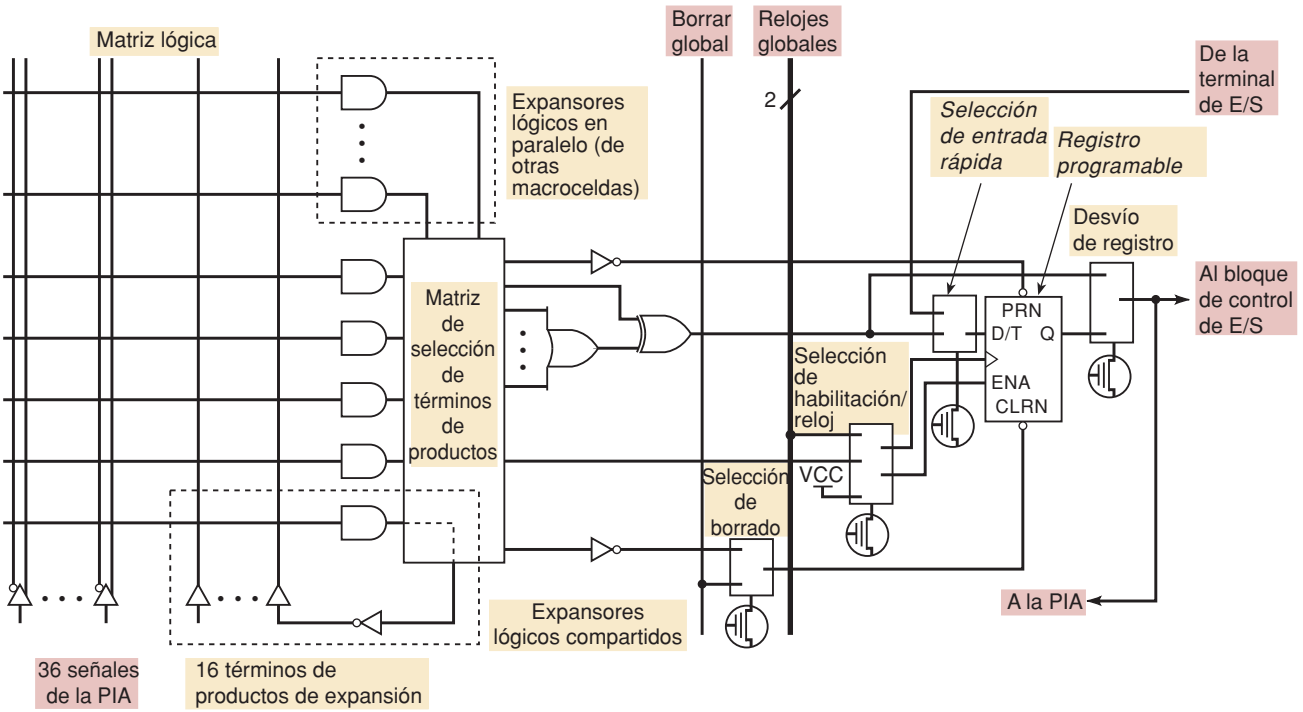


FIGURA 13-13 Macrocelda de la familia MAX7000S. (Cortesía de Altera Corporation.)

Para las funciones registradas, cada flip-flop de macrocelda puede programarse de manera individual para implementar la operación en D, T, JK o SR. Cada registro programable puede sincronizarse en tres modos distintos: (1) con una señal de reloj global, (2) con una señal de reloj global cuando se habilita el flip-flop, o (3) con una señal de reloj de matriz producida por una macrocelda oculta o una terminal de entrada (no global). En el EPM7128S puede utilizarse cualquiera de las terminales de reloj globales (GCLK1 o GCLK2) para producir el rendimiento de reloj a Q más rápido. Cualquier flanco del reloj puede programarse para disparar los flip-flops. Cada registro puede prestablecerse en forma asíncrona o borrarse con un término de producto activo en ALTO o activo en BAJO. Cada registro puede borrarse también con la terminal de borrar global activa en BAJO (GCLRn). También hay disponible una ruta de entrada de datos rápida desde una terminal de E/S hacia los registros, con desvío de la PIA. Todos los registros en el dispositivo se restablecerán en forma automática al encendido.

Los dispositivos MAX7000S tienen una opción de ahorro de energía que permite al diseñador programar cada macrocelda individual para la operación en alta velocidad (el bit turbo encendido) o bajo consumo de energía (el bit turbo apagado). Debido a que la mayoría de las aplicaciones lógicas requiere que sólo una pequeña fracción de las compuertas opere a la frecuencia máxima, esta característica puede producir un ahorro considerable en el consumo total de energía del sistema. Las rutas sensibles a la velocidad en el diseño pueden trabajar a la máxima velocidad, mientras que las rutas de señal restantes pueden operar con una energía reducida.

PREGUNTAS DE REPASO

1. ¿Qué es una macrocelda?
2. ¿Qué es un dispositivo ISP?
3. ¿Qué funciones de control especiales se proporcionan con las cuatro terminales de sólo entrada en un dispositivo MAX7000S?
4. ¿Qué ventaja del sistema se logra al reducir la velocidad de las macroceldas seleccionadas en un dispositivo MAX7000S?

13-6 LA FAMILIA FLEX10K DE ALTERA

La familia de dispositivos lógicos programables FLEX10K de Altera tiene una arquitectura muy distinta. En vez de la matriz programable de compuertas AND/OR fijas que se utiliza en los dispositivos MAX7000S, esta familia se basa en una arquitectura de tabla de búsqueda (LUT). Para producir funciones lógicas, la tabla de búsqueda almacena los resultados de salida de la función en una memoria basada en SRAM. En esencia, funciona igual que la tabla de verdad para la función lógica. La tecnología SRAM para PLDs programa con mucha más velocidad que los dispositivos basados en EEPROM, y también produce una densidad muy alta de celdas de almacenamiento que se utilizan para programar los PLDs más grandes. Sin embargo y a diferencia de la mayoría de las FPGAs, Altera ha optado por utilizar un diseño de enrutamiento de señales programable para la familia FLEX10K, que tiene una apariencia más parecida a una versión mejorada de la PIA que se encuentra en la familia MAX7000S de CPLD. Como resultado, la familia FLEX10K tiene características de arquitectura que son una combinación de las dos clasificaciones de HCPLDs. Con base en la arquitectura de alta densidad de las celdas lógicas, los dispositivos FLEX10K se clasifican generalmente como FPGAs.

Vamos a examinar el concepto de una tabla de búsqueda. La LUT es la porción del bloque de lógica programable que produce una función combinacional (vea la figura 13-14). Esta función puede utilizarse como la salida del bloque lógico o puede registrarse (controlada por el MUX interno). La tabla de búsqueda en sí consiste de un conjunto de flip-flops que almacenan la tabla de verdad deseada para nuestra función. Por lo general, las LUTs son pequeñas, y manejan comúnmente cuatro variables de entrada, por lo que nuestra tabla de verdad tendría un total de 16 combinaciones. Necesitaremos un flip-flop para almacenar cada uno de los 16 valores de funciones (vea la figura 13-15). Hasta cuatro variables de entrada en nuestra LUT de ejemplo estarán conectadas a las entradas de datos en el bloque decodificador, mediante el uso de interconexiones programables. La combinación de entrada que se aplique determinará cuál de los 16 flip-flops se seleccionará para alimentar la salida por medio de los búferes triestado. En esencia, la tabla de búsqueda es un bloque de memoria SRAM de 16 $\times$ 1. Todo lo que tenemos que hacer para crear cualquier función deseada (de hasta cuatro variables de entrada) es almacenar el conjunto apropiado de 0s y 1s en los flip-flops de la LUT. Esto es lo que se hace básicamente para programar este tipo de PLD. Como los flip-flops son volátiles (son SRAM), necesitamos cargar en la memoria de la LUT las funciones deseadas cada vez que el PLD se enciende. A este proceso se le llama configurar el PLD. Otras porciones del dispositivo también se programan en la misma forma, utilizando otros bits de la memoria SRAM para almacenar la información de programación. Ésta es la técnica de programación básica para los bloques lógicos, llamada **elementos lógicos (LEs)**, que se encuentra en los dispositivos FLEX10K.

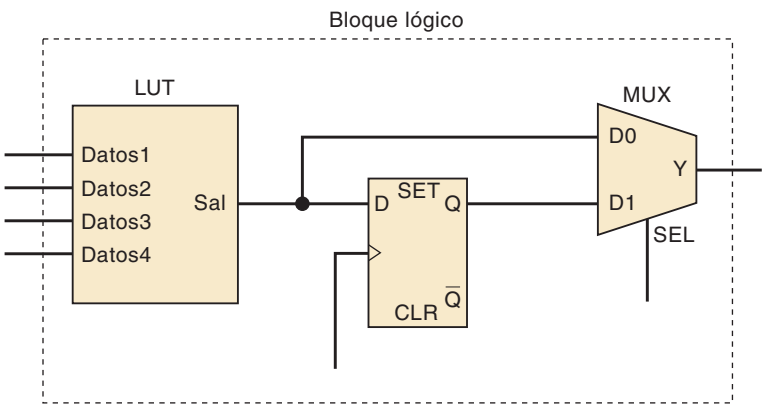


FIGURA 13-14
Diagrama de bloques lógicos simplificado para el dispositivo FLEX10K.

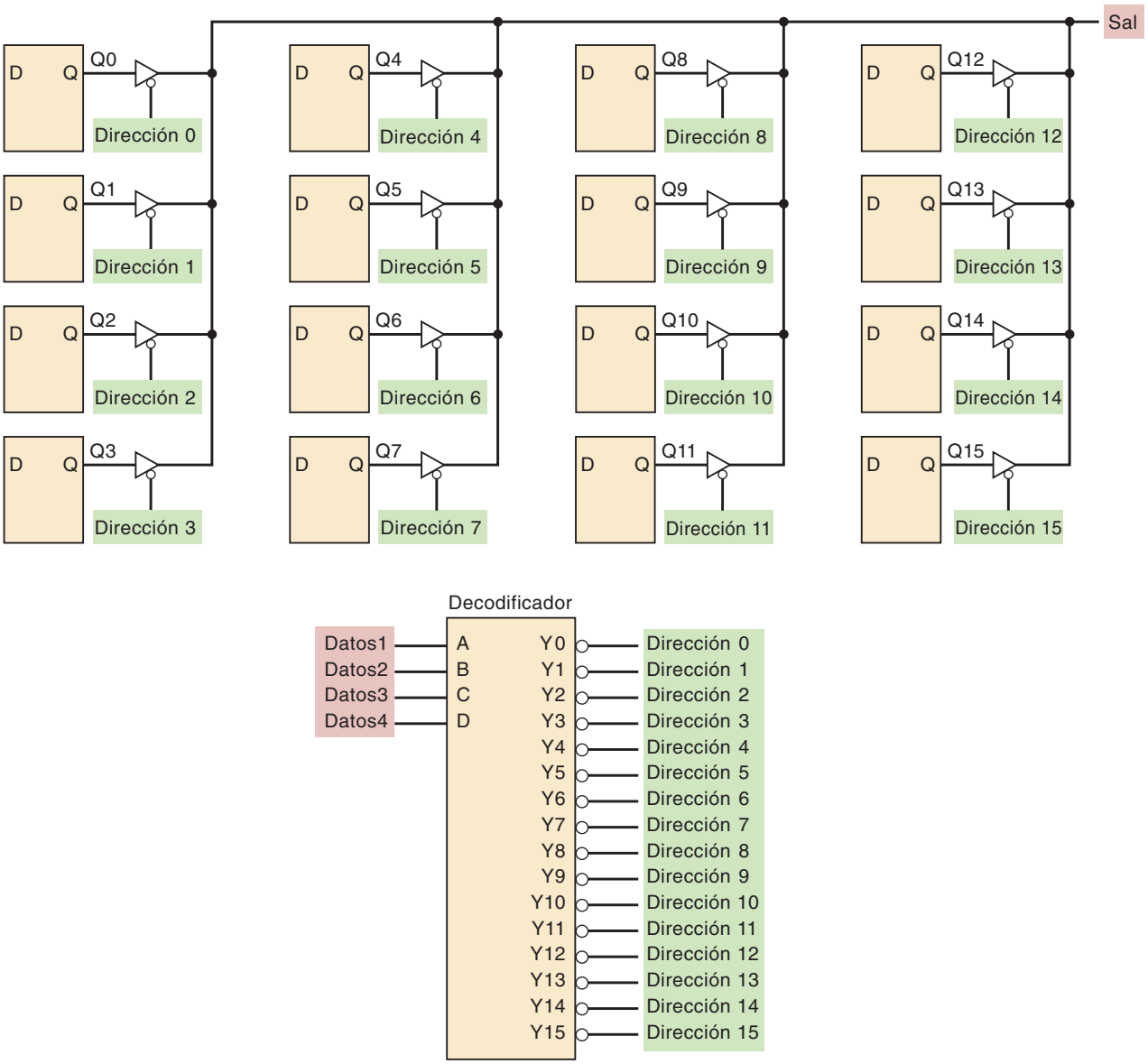


FIGURA 13-15 Diagrama de bloques funcional para una LUT.

La figura 13-16 muestra el diagrama de bloques para un elemento lógico FLEX10K. Contiene la LUT y el registro programable, así como circuitos de expansión en cascada y de acarreo, funciones de control programables e interconexiones de bus locales y globales. El flip-flop programable puede configurarse para la operación en D, T, JK o SR y se desviará para las funciones combinacionales. Las señales de control del flip-flop (clock, clear y preset) pueden controlarse en forma selectiva mediante entradas globales, terminales de E/S de propósito general o cualquier función creada en forma interna. El LE puede producir dos salidas para controlar interconexiones locales (LAB) y globales (FastTrack) en el chip. Esto permite a la LUT y al registro en un LE utilizarse para funciones no relacionadas. Dos tipos de rutas de datos de alta velocidad (cadenas en cascada y cadenas de acarreo) conectan los LEs adyacentes sin utilizar interconexiones locales. La expansión de cadena en cascada permite a la arquitectura FLEX10K crear funciones con más de cuatro variables de entrada. Las LUTs adyacentes pueden conectarse entre sí en paralelo, en donde cada LUT adicional proporciona cuatro variables más de entrada. La cadena de acarreo proporciona una función de acarreo de avance rápido entre los LEs, lo cual permite

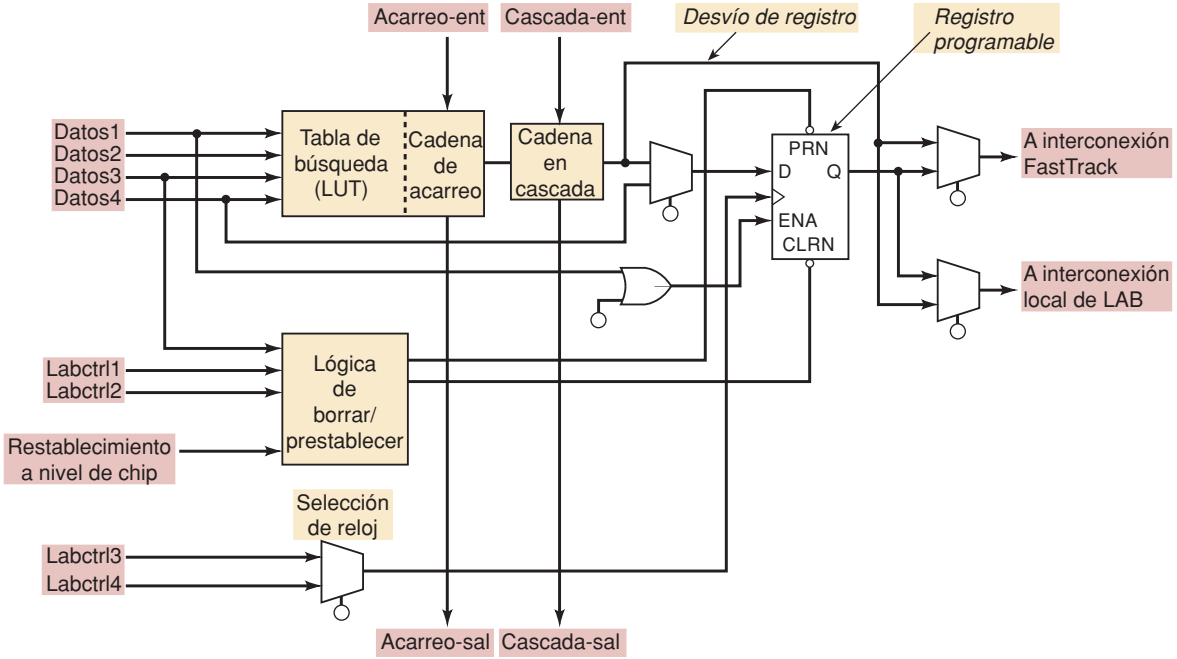


FIGURA 13-16 Elemento lógico FLEX10K. (Cortesía de Altera Corporation.)

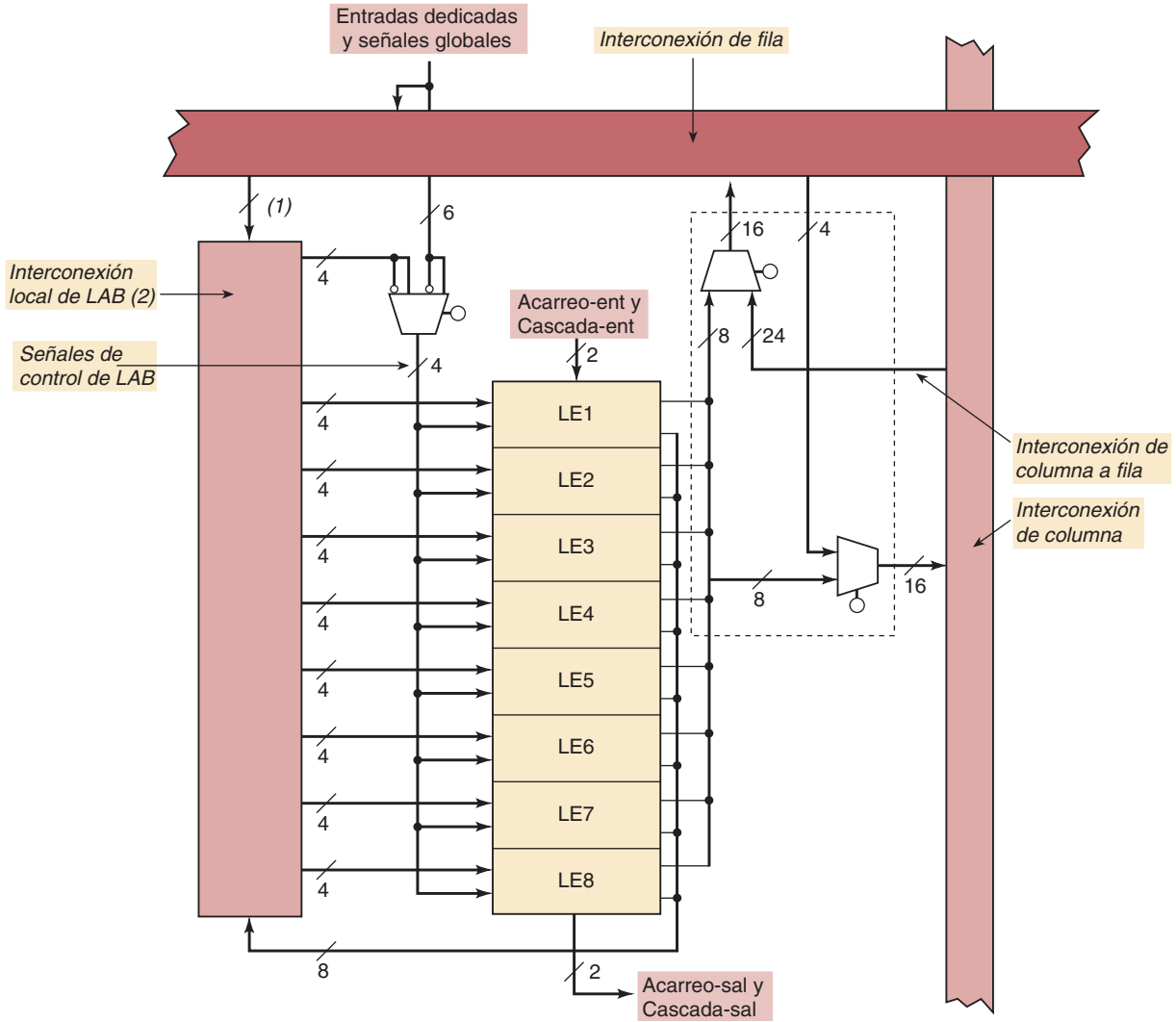


FIGURA 13-17 Bloque de matriz lógica FLEX10K. (Cortesía de Altera Corporation.)

implementar con eficiencia funciones que se basan en otras funciones, como las que se encuentran en los contadores, sumadores y comparadores. En estas funciones, los bits superiores dependen de los bits inferiores. Sin una característica de expansión como la cadena de acarreo, los tiempos de propagación pueden volverse bastante largos para los circuitos más grandes. La lógica de cadena en cascada y cadena de acarreo puede crearse en forma automática mediante el software compilador, o en forma manual por el diseñador durante la introducción del diseño. Los tiempos de propagación se incrementarán por una pequeña cantidad cuando se utilicen las opciones de expansión. El software MAX+PLUS II o Quartus II Timing Analyzer calculan estos retrasos adicionales para un diseño dado. El uso intensivo de las cadenas de acarreo y en cascada puede reducir la flexibilidad de enrutamiento, por lo que deberá limitarse a las porciones del diseño sensibles a la velocidad.

El bloque de matriz lógica para la familia FLEX10K contiene ocho elementos lógicos y la interconexión local para ese LAB (vea la figura 13-17). Las señales de un LE a otro dentro de un LAB se enlutan con la interconexión local. Las interconexiones de fila y columna, que Altera denominó interconexiones FastTrack, proporcionan las rutas de señales entre LABs. Cada LAB tiene cuatro señales de control disponibles para los ocho LEs. Dos pueden utilizarse para los relojes de registro y las otras dos para las señales prestablecer o borrar.

En la figura 13-18 se muestra el diagrama de bloques general para un dispositivo FLEX10K. Además de los bloques de matrices lógicas y las interconexiones *FastTrack* que ya hemos descrito, los dispositivos contienen elementos de E/S (IOEs) y bloques de matrices embebidos (EABs). Cada uno de los IOEs contiene un búfer de E/S bidireccional y un registro, los cuales pueden utilizarse para el almacenamiento de datos de entrada o de salida. Cada EAB proporciona un bloque flexible de 2048

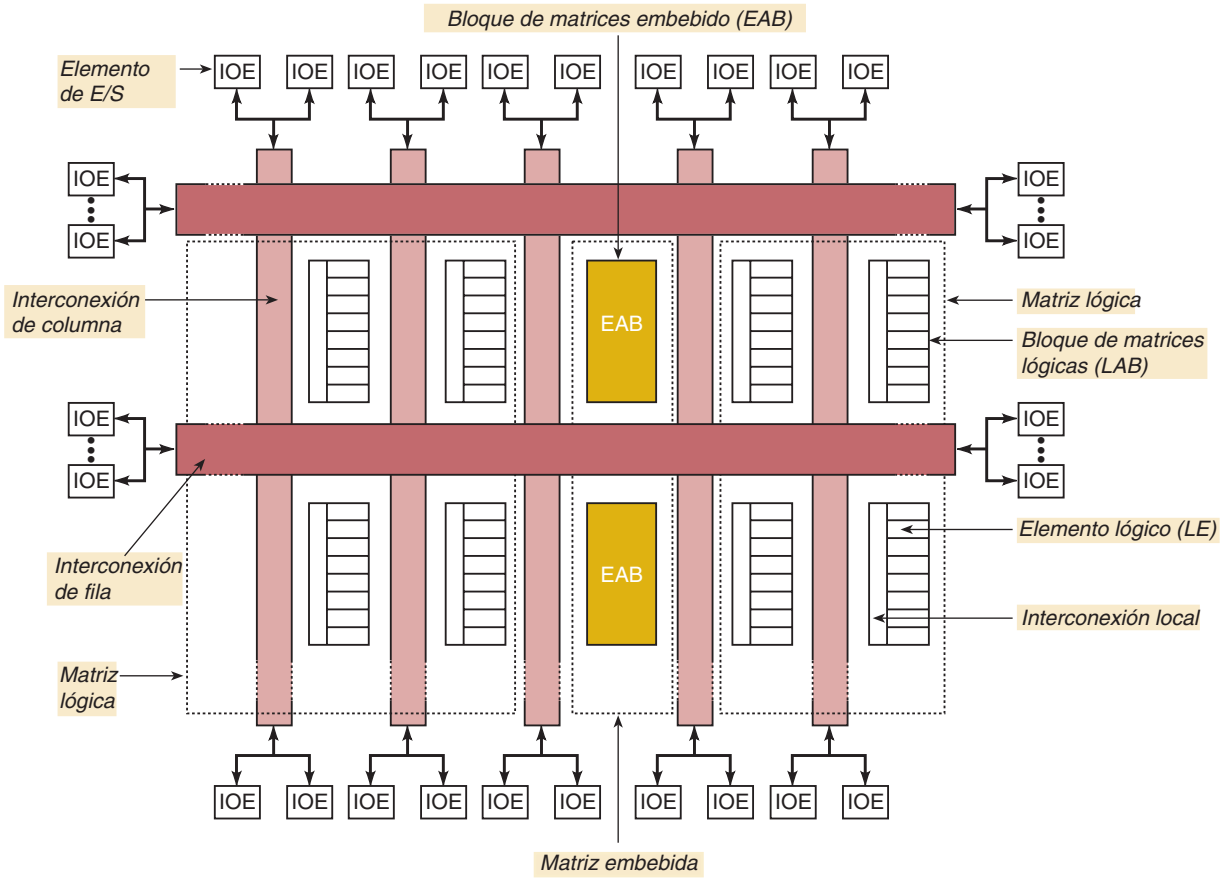


FIGURA 13-18 Diagrama de bloques de un dispositivo FLEX10K. (Cortesía de Altera Corporation.)

TABLA 13-3 Características de los dispositivos de la familia FLEX10K de Altera.

Característica	EPF10K10	EPF10K20	EPF10K30	EPF10K40	EPF10K50	EPF10K70	EPF10K100	EPF10K120	EPF10K250
Número común de compuertas	10,000	20,000	30,000	40,000	50,000	70,000	100,000	120,000	250,000
Número máximo de compuertas	31,000	63,000	69,000	93,000	116,000	118,000	158,000	211,000	310,000
LEs	576	1,152	1,728	2,304	2,880	3,744	4,992	6,656	12,160
LABs	72	144	216	288	360	468	624	832	1,520
EABs	3	6	6	8	10	9	12	16	20
Número máximo de terminales de E/S	150	189	246	189	310	358	406	470	470

bits de almacenamiento en RAM para varias aplicaciones de memoria internas. Si se combinan varios EABs en un chip se pueden crear bloques más grandes de RAM. Un EAB también puede utilizarse para crear funciones combinacionales extensas mediante la implementación de una LUT.

La familia FLEX10K tiene varios tamaños de dispositivos, como se muestra en la tabla 13-3. La tarjeta de desarrollo académico UP2 de Altera también contiene un dispositivo EPF10K70 en un encapsulado de 240 terminales. Como puede ver en la tabla, este dispositivo tiene muchos recursos lógicos disponibles.

PREGUNTAS DE REPASO

1. ¿Qué es una tabla de búsqueda?
2. ¿Qué ventaja tiene la tecnología de programación de SRAM en comparación con EEPROM?
3. ¿Qué desventaja tiene la tecnología de programación de SRAM en comparación con EEPROM?
4. ¿Qué son los EABs? ¿Para qué pueden utilizarse?

13-7 LA FAMILIA CYCLONE DE ALTERA

Continuamente se desarrollan nuevas familias de dispositivos HCPLD. Las arquitecturas de estas nuevas familias proporcionan varias combinaciones de mejoras en los recursos lógicos y de enrutamiento de señales, en la densidad (mayor número de elementos lógicos), en la cantidad de memoria embebida, en el número de terminales de E/S disponibles para el usuario, en mayores velocidades y menores costos. Otra familia de Altera que podría ser de nuestro interés es la familia Cyclone. La tarjeta de desarrollo académico UP3 de Altera contiene un dispositivo Cyclone EP1C6. En un dispositivo Cyclone, las funciones lógicas se implementan en LEs (elementos lógicos) que contienen una LUT (tabla de búsqueda) de cuatro entradas y un registro programable (flip-flop D), de manera similar a los que se encuentran en los dispositivos FLEX10K. El LE Cyclone contiene características avanzadas para proporcionar un uso lógico más eficiente que con el FLEX10K. Por ejemplo, el LE Cyclone se ha mejorado para crear con más eficiencia aplicaciones digitales que utilizan sumadores/restadores, carga asíncrona del registro programable y registros de desplazamiento. Los bloques de matrices lógicas en los dispositivos Cyclone consisten de 10 LEs y una interconexión local. Esta familia también contiene bloques

TABLA 13-4
Características de los dispositivos de la familia Cyclone de Altera.

Característica	EP1C3	EP1C4	EP1C6	EP1C12	EP1C20
LEs	2,910	4,000	5,980	12,060	20,060
Bloques de RAM M4K	13	17	20	52	64
Bits de RAM totales	59,904	78,336	92,160	239,616	294,912
PLLs	1	2	2	2	2
Número máximo de terminales de E/S	104	301	185	249	301

de 4K bits de memoria RAM, los cuales pueden configurarse como memoria de puerto dual o sencillo con palabras de hasta 36 bits de ancho. Una red de relojes globales con ocho líneas de reloj global proporciona relojes para todos los elementos de E/S, los LEs y los bloques de memoria. Los ciclos de bloqueo de fase internos (PLLs) proporcionan las operaciones de multiplicación y división de la frecuencia del reloj y el desplazamiento de fase de la señal de reloj. En la tabla 13-4 se comparan las características de los dispositivos de la familia Cyclone. Estos dispositivos tienen la capacidad de integrarse con otros circuitos digitales mediante el uso de múltiples estándares de E/S, pero no soportan la E/S de 5 V. El software de diseño MAX+PLUS II no soporta los dispositivos de la familia Cyclone.

RESUMEN

1. Los dispositivos lógicos programables (PLDs) son la tecnología clave en el futuro de los sistemas digitales.
2. Los PLDs pueden reducir el inventario de piezas, simplificar los circuitos de prototipos, acortar el ciclo de desarrollo, reducir el tamaño y los requerimientos de energía del producto, y permitir que el hardware de un circuito se actualice con facilidad.
3. Las principales categorías de los sistemas digitales son: lógica estándar, circuitos integrados de aplicación específica (ASICs) y dispositivos de microprocesador/procesadores digitales de señales (DSP).
4. Los dispositivos ASIC pueden ser dispositivos lógicos programables (PLDs), matrices de compuertas, celdas estándar o dispositivos totalmente personalizados.
5. Los PLDs son el tipo de ASIC menos costoso de desarrollar.
6. Los PLDs simples (SPLDs) contienen el equivalente de 600 o menos compuertas y se programan mediante las tecnologías de fusible, EPROM o EEPROM.
7. Los PLDs de alta capacidad (HCPLDs) tienen dos categorías de arquitectura principales: dispositivos lógicos programables complejos (CPLDs) y matrices de compuertas programables en campo (FPGAs).
8. Las tecnologías de programación de CPLDs más comunes son EEPROM y Flash, ambas no volátiles.
9. La tecnología de programación de CPLDs más común es SRAM, la cual es volátil.
10. El GAL 16V8 es uno de los PLDs más simples disponibles, pero aún se utiliza mucho y demuestra los principios básicos detrás de todos los PLDs.
11. El CPLD EPM7128S de Altera contiene 128 macroceldas, cada una de las cuales contiene un circuito AND/OR programable y un registro programable.
12. El EPM7128SLC84 puede tener hasta 68 entradas y salidas.

13. La familia MAX7000S de CPLDs es programable en sistema (ISP).
14. Las familias de dispositivos FLEX10K y Cyclone de Altera utilizan una arquitectura de tabla de búsqueda (LUT) en una tecnología de SRAM.
15. La tecnología de programación de SRAM es volátil, lo cual significa que los dispositivos deben reconfigurarse al encenderlos.

TÉRMINOS IMPORTANTES

ASIC de celda estándar
ASIC totalmente personalizado
bloque de matrices lógicas (LAB)
circuito integrado de aplicación específica (ASIC)
dispositivo lógico programable (PLD)
elemento lógico (LE)
lógica estándar
macrocelda
matriz de compuertas
matriz de compuertas programables en campo (FPGA)
matriz de interconexiones programables (PIA)
matriz lógica programable (PAL)
microprocesador
PLD complejo (CPLD)
PLD de alta capacidad (HCPLD)
PLD simple (SPLD)
procesador digital de señales (DSP)
programable una sola vez (OTP)
tabla de búsqueda (LUT)

PROBLEMAS

SECCIÓN 13-1

13-1. Describa cada una de las siguientes categorías principales de los sistemas digitales:
   (a) Lógica estándar.
   (b) ASICs.
   (c) Microprocesador/DSP.

13-2.* Nombre tres factores que, por lo general, se consideran al hacer decisiones de ingeniería de diseño.

13-3. ¿Por qué a un sistema de microprocesador/DSP se le llama solución de software para un diseño?

13-4.* ¿Cuál es la principal ventaja que tiene una solución de diseño de hardware en comparación con una solución de software?

13-5. Describa cada una de las cuatro siguientes subcategorías de ASIC:
   (a) PLDs.
   (b) Matrices de compuertas.
   (d) Celda estándar.
   (e) Totalmente personalizado.

13-6.* ¿Cuáles son las principales ventajas y desventajas de un ASIC totalmente personalizado?

13-7. Nombre las seis tecnologías de programación de PLDs. ¿Cuál de ellas es programable sólo una vez? ¿Cuál es volátil?

13-8.* ¿Qué diferencia tiene la programación de PLDs basados en SRAM, en comparación con las otras tecnologías de programación?

SECCIÓN 13-5

13-9. Describa las funciones de cada una de las siguientes estructuras de arquitectura que se encuentran en la familia MAX7000S de Altera:

* Encontrará las respuestas a los problemas marcados con un asterisco al final del libro.

(a) LAB.

(b) PIA.

(c) Macrocelda.

13-10.* ¿Cuáles son las dos formas que pueden usarse para programar los dispositivos de la familia MAX7000S?

13-11. ¿Qué interface de dispositivo estándar se utiliza para la programación en sistema en la familia MAX7000S?

13-12.* ¿Cuáles son las cuatro terminales de sólo entrada en el EPM7128SLC84 (por número de terminal y función)?

13-13. ¿Cuál es la ventaja de utilizar una de las entradas de reloj globales para la operación registrada?

SECCIÓN 13-6

13-14.* ¿Cuál es la diferencia fundamental de arquitectura entre las familias MAX7000S y FLEX10K? ¿Cuál es la tecnología de programación que se utiliza para cada familia? ¿Cuál familia es no volátil? ¿Cuál familia contiene más recursos lógicos?

RESPUESTAS A LAS PREGUNTAS DE REPASO DE SECCIÓN

SECCIÓN 13-1

1. Lógica estándar, ASICs, microprocesador. 2. La velocidad 3. Circuito integrado de aplicación específica. 4. Dispositivos lógicos programables, arreglos de compuertas, celdas estándar, totalmente personalizados. 5. Dispositivo lógico programable de alta capacidad. 6. (1) Bloques lógicos: CPLD de AND/OR fijo programable en comparación con FPGA de tabla de búsqueda (2) Recursos de enrutamiento de señales: CPLD uniforme en comparación con FPGA variado. 7. La volatilidad se refiere al caso en el que un PLD (o dispositivo de memoria) pierde la información almacenada cuando se desconecta la energía.

SECCIÓN 13-2

1. Un CI que contiene un gran número de compuertas, cuyas interconexiones pueden modificarse por el usuario para realizar una función específica. 2. $O_1 = A$
3. Un fusible intacto. 4. Una conexión fija permanente.

SECCIÓN 13-3

2. OR fija permanente; AND programable. 3. AND fija permanente; OR programable. 4. $O_1 = AB\overline{CD} + \overline{AB}CD + ABCD = \overline{A}B\overline{CD} + \overline{A}CD$

SECCIÓN 13-4

1. Pueden borrarse y reprogramarse; tienen una OLMC. 2. Simple, complejo y registrado.

SECCIÓN 13-5

1. Una macrocelda es el bloque lógico programable en los CPLDs MAX7000S, que consiste de un circuito AND/OR programable y un registro programable (flip-flop). 2. Un dispositivo PLD ISP se programa en sistema, lo cual significa que puede programarse mientras está conectado en el circuito. 3. Relojes globales, habilitaciones de salida triestado, borrar asíncrona. 4. El consumo de energía puede reducirse al disminuir la velocidad de las macroceldas.

SECCIÓN 13-6

1. Por lo general, una tabla de búsqueda es un arreglo de SRAM de 16 palabras por 1 bit, que se utiliza para almacenar los niveles lógicos de salida deseados para una función lógica simple. 2. La SRAM se programa más rápido y tiene una mayor densidad de celdas lógicas que la EEPROM. 3. La SRAM es volátil y debe reconfigurarse cuando se enciende el dispositivo. 4. Los bloques de matrices embebidos proporcionan almacenamiento tipo RAM en el PLD.



GLOSARIO

& Cuando se utiliza dentro de un símbolo de IEEE/ANSI, indica una compuerta AND o una función AND.

Acarreo Dígito o bit que se genera cuando se suman dos números y el resultado es mayor que la base para el sistema numérico que se esté utilizando.

Acarreo anticipado Habilidad de algunos sumadores en paralelo de predecir, sin tener que esperar a que el acarreo se propague a través de los sumadores completos, si se generará o no un bit de acarreo (C_{OUT}) como resultado de la suma, con lo cual se reducen los retrasos de propagación en general.

Actuador Dispositivo controlado eléctricamente que a su vez controla una variable física.

Acumulador Registro principal de una unidad aritmética-lógica (ALU).

ADC ascendente/descendente de rampa digital Tipo de convertidor analógico-digital que utiliza un contador ascendente/descendente para aumentar o disminuir el voltaje de un convertidor digital-analógico hasta que intersecta a la entrada analógica.

ADC de aproximaciones sucesivas Tipo de convertidor analógico-digital en el cual se utilizan un registro paralelo interno y la lógica compleja de control para realizar la conversión. El tiempo de conversión para este tipo de convertidor analógico-digital es siempre el mismo, sin importar el valor de la señal analógica de entrada.

ADC de rampa Tipo de convertidor analógico-digital en el cual se genera una forma de onda interna tipo escalera y se utiliza para lograr la conversión. El tiempo de conversión para este tipo de convertidor analógico-digital varía dependiendo del valor de la señal analógica de entrada.

ADC de voltaje a frecuencia Tipo de convertidor analógico-digital que convierte el voltaje analógico en un tren de pulsos que luego se cuenta para producir una salida digital.

ADC Flash Tipo de convertidor analógico-digital que tiene la mayor velocidad de operación disponible.

Adquisición de datos Proceso mediante el cual una computadora obtiene datos analógicos digitalizados.

Agrupamiento Combinación de cuadros adyacentes en un mapa de Karnaugh que contiene 1s para fines de simplificar una expresión de suma de productos.

Álgebra Booleana Proceso algebraico que se utiliza como herramienta en el análisis y diseño de sistemas digitales. En el álgebra Booleana sólo hay dos valores posibles, 0 y 1.

Alias Una señal digital que se produce por el muestreo de una señal entrante a una tasa menor que el doble de la frecuencia más alta contenida en la señal entrante.

Almacenamiento masivo Almacenamiento de grandes cantidades de datos; no forma parte de la memoria interna de una computadora.

AND alambrado Término que se utiliza para describir la función lógica que se crea cuando las salidas de colector abierto se conectan entre sí.

Anidar Incrustar una estructura de control dentro de otra estructura de control.

Ánodo común Pantalla de LED en la cual los ánodos de todos los LEDs están unidos.

ARCHITECTURE Palabra clave en VHDL utilizada para comenzar una sección de código que define la operación de un bloque de circuito (ENTITY).

Arreglo de bits Una manera de representar un grupo de bits, al darle un nombre y asignar un número de elemento a la posición de cada bit. A esta misma estructura algunas veces se le da el nombre de vector de bits.

Arreglo de compuertas Circuito integrado de aplicación específica (ASIC) fabricado a partir de cientos de miles de compuertas básicas prefabricadas, las cuales pueden interconectarse en forma personalizada en las últimas

etapas de fabricación, para formar el circuito digital deseado.

Arreglo de Compuertas Programable en Campo (FPGA) Clase de PLDs que contiene una matriz de celdas lógicas más complejas, las cuales pueden interconectarse en forma muy flexible para implementar circuitos lógicos de alto nivel.

Arreglo de Interconexión Programable (PIA) Un término utilizado por Altera Corporation para describir los recursos que se utilizan para conectar los LABs entre sí y también con los módulos de entrada/salida.

Arreglo de Lógica Programable (PLA) Clase de dispositivos lógicos programables. Sus matrices AND y OR son programables. También se le conoce como *arreglo de lógica programable en campo (FPLA)*.

Arreglo Lógico Programable en Campo (FPLA) Un PLD que utiliza una matriz AND programable y una matriz OR programable.

Arriba hacia abajo Método de diseño que empieza en el nivel del sistema en general y después define una jerarquía de módulos.

Asignación de señal condicional Una construcción concurrente en VHDL que evalúa una serie de condiciones en forma secuencial, para determinar el valor apropiado que debe asignarse a una señal. La primera condición verdadera que se evalúa es la que determina el valor asignado.

Asignación de señal selecta Instrucción en VHDL que permite que a un objeto de datos se le asigne un valor de una de varias fuentes de señal, dependiendo del valor de una expresión.

Asignado Término que se utiliza para describir el estado de una señal lógica; es sinónimo de "activo".

Autoprueba al encendido Programa almacenado en ROM que la CPU ejecuta al momento de encender el equipo para probar las porciones correspondientes a la RAM y/o la ROM del circuito de la computadora.

Bajo consumo de energía Modo de operación en el cual un chip se deshabilita y consume mucho menos energía que cuando está habilitado por completo.

Biblioteca de Módulos Parametrizados (LPM) Conjunto de funciones de biblioteca genéricas diseñadas para ser muy flexibles, en cuanto a que permiten al usuario especificar el número de bits, número mod, opciones de control, etc.

Bibliotecas Una colección de descripciones de circuitos de hardware de uso común, los cuales pueden usarse como módulos en un archivo de diseño.

Bit Dígito en el sistema binario.

BIT En VHDL, el tipo de objeto de datos que representa a un dígito binario (bit).

Bit de paridad Bit adicional que se agrega en cada palabra, de manera que el número total de 1s que se transmita siempre sea par (o siempre impar).

Bit de signo Bit binario que se agrega a la posición más a la izquierda de un número binario para indicar si ese número representa una cantidad positiva o negativa.

Bit más significativo (MSB) Bit binario de más a la izquierda con ponderación mayor de una cantidad expresada en binario.

Bit menos significativo (LSB) El bit más a la derecha y con menor ponderación en una cantidad expresada en binario.

BIT_VECTOR En VHDL, el tipo de objeto de datos que representa a un arreglo de bits. *Vea también* Arreglo de bits.

Bloque de Arreglos Lógicos (LAB) Término que utiliza Altera Corporation para describir los bloques de construcción de sus CPLDs. Cada LAB es similar en complejidad con un SPLD.

Bloque de control común Símbolo utilizado por el estándar IEEE/ANSI para describir cuando una o más entradas son comunes para más de uno de los circuitos en un CI.

Búfer circular Un sistema de memoria que siempre contiene los últimos *n* datos que se han escrito. Siempre que se almacena un nuevo dato, éste sobrescribe al valor más antiguo en el búfer.

Búfer de velocidad (transferencia de) datos Aplicación de FIFOs en donde los datos secuenciales se escriben en la FIFO a cierta velocidad y leen a una velocidad distinta.

Búfer lineal Sistema de memoria tipo FIFO (primero en entrar, primero en salir) que llena a cierta velocidad y vacía a otra. Una vez que está lleno, no podrán almacenarse datos sino hasta que se lean los datos que están en el búfer. *Vea también* Memoria FIFO. (Primero en entrar, primero en salir.)

Búfer/Reforzador Circuito diseñado para tener una mayor capacidad de corriente y/o voltaje de salida, en comparación con un circuito lógico ordinario.

Burbujas Pequeños círculos en las líneas de entrada o de salida en los símbolos de circuitos lógicos, los cuales representan el inverso de una determinada señal. Si hay una burbuja presente, se dice que la salida o entrada es activa en BAJO.

Bus Grupo de cables que transportan bits de información relacionados.

Bus de control Conjunto de líneas digitales que se utilizan para sincronizar las actividades de la CPU y los elementos separados de la microcomputadora.

Bus de datos Líneas bidireccionales que transportan datos entre la CPU y la memoria, o entre la CPU y los dispositivos de E/S.

Bus de direcciones Líneas unidireccionales que llevan el código de dirección del CPU a la memoria y/o los dispositivos de E/S.

Bus flotante Cuando todas las salidas conectadas a un bus de datos se encuentran en el estado de alta impedancia (Hi-Z).

Byte Grupo de ocho bits.

Caché Un sistema de memoria de alta velocidad que puede cargarse desde la DRAM del sistema, la cual es más lenta, para que el CPU de alta velocidad pueda acceder a ella con rapidez.

Capacidad Cantidad de espacio de almacenamiento en una memoria; expresada como el número de bits o el número de palabras.

Capacidad de carga Número máximo de entradas lógicas estándar que puede excitar la salida de un circuito digital sin pérdida de confiabilidad.

Captura esquemática Programa de computadora que puede interpretar símbolos gráficos y conexiones de señales, para traducirlas en relaciones lógicas.

Carga en paralelo *Vea* Transferencia de datos en paralelo.

CAS (Estrobo de Dirección de Columna) Señal que se utiliza para fijar la dirección de columna en una DRAM.

CAS antes que RAS Método para refrescar memorias DRAM que tienen contadores de regeneración integrados. Cuando la entrada *CAS* pone en BAJO y se mantiene ahí mientras *RAS* se pone también en BAJO, el contador de regeneración integrado en el chip realiza una operación de regeneración interna en la dirección de fila especificada.

Cascada Conexión de circuitos lógicos en serie, en donde la salida de un circuito se conecta a la entrada del siguiente, y así sucesivamente.

CASE Estructura de control que, con base en el valor de un dato, selecciona una de varias opciones al describir la operación de un circuito.

Cátodo común Pantalla de LED en la cual todos los cátodos de los LEDs están unidos.

Celda de memoria Dispositivo que almacena un solo bit.

Celda estándar Circuito integrado de aplicación específica (ASIC) fabricado de bloques lógicos prediseñados que provienen de una biblioteca de diseños de celda estándar, los cuales se interconectan durante la etapa de diseño del sistema y después se fabrican en un solo circuito integrado.

Ciclo FOR *Vea* Ciclo iterativo.

Ciclo iterativo Estructura de control que implica una operación repetitiva y un número establecido de iteraciones.

Circuito de conducción de pulso Un circuito lógico que puede usarse para seleccionar el destino de un pulso de entrada, dependiendo de los niveles lógicos presentes en las entradas del circuito.

Circuito de muestreo y retención Tipo de circuito que utiliza un amplificador de búfer de ganancia unitaria en conjunto con un capacitor para mantener la entrada estable durante un proceso de conversión analógica-digital.

Circuito detector de flancos Circuito que produce un pico o impulso positivo breve que ocurre coincidente con la transición activa de un pulso de entrada de reloj.

Circuito Integrado de Aplicación Específica (ASIC) Un CI diseñado específicamente para cumplir con los requerimientos de una aplicación. Las subcategorías incluyen: PLDs, matrices de compuertas, celdas estándar y circuitos integrados personalizados.

Circuito lógico Cualquier circuito que se comporta de acuerdo con un conjunto de reglas lógicas.

Circuito NOR exclusivo (XNOR) Circuito lógico de dos entradas que produce una salida en ALTO sólo cuando las entradas son iguales.

Circuito NOT *Vea* INVERSOR.

Circuito OR exclusivo (XOR) Circuito lógico de dos entradas que produce una salida en ALTO sólo cuando las entradas son distintas.

Circuito secuencial Un circuito lógico cuyas salidas pueden cambiar de estados en sincronía con una señal de reloj periódica. El nuevo estado de una salida puede depender de su estado actual, así como de los estados actuales de otras salidas.

Circuitos de inhibición Circuitos lógicos que controlan el paso de una señal de entrada hacia la salida.

Circuitos integrados bipolares Circuitos integrados digitales en los cuales los elementos principales del circuito son transistores NPN y PNP.

Circuitos integrados digitales Circuitos digitales autocontenidos que se fabrican mediante el uso de uno de varias tecnologías de fabricación de circuitos integrados.

Circuitos integrados unipolares Circuitos digitales integrados en los cuales los transistores de efecto de campo (MOSFET) unipolares son los elementos principales del circuito.

Circuitos lógicos combinacionales Circuitos compuestos de compuertas lógicas, sin retroalimentación de las salidas hacia las entradas.

CLEAR Una entrada a un latch o FF que se utiliza para hacer que $Q = 0$.

CMOS (Metal-Óxido-Semiconductor Complementario) Tecnología de circuitos integrados que usa transistores MOSFET como elemento principal del circuito. Esta familia lógica pertenece a la categoría de circuitos digitales unipolares.

Codificación Uso de un grupo de símbolos para representar números, letras o palabras.

Codificación binaria directa Representación de un número decimal en base a su número binario equivalente.

Codificador Circuito digital que produce un código de salida, dependiendo de cuál de sus entradas se activa.

Codificador de prioridad Tipo especial de codificador que detecta cuando se activan dos o más entradas en forma simultánea y después genera un código que corresponde a la entrada de mayor numeración.

Código ASCII (American Standard Code for Information Interchange — Código estándar estadounidense para el intercambio de información) Código alfanumérico de siete bits, utilizado por la mayoría de los fabricantes de computadoras.

Código Decimal Codificado en Binario (BCD) Código de cuatro bits que se utiliza para representar los dígitos de un número decimal mediante su equivalente binario de cuatro bits.

Código Gray Código en el que nunca cambia más de un bit cuando se pasa de un estado a otro.

Códigos alfanuméricos Códigos que representan números, letras, signos de puntuación y caracteres especiales.

Colisión de bus Situación en la cual las salidas de dos o más dispositivos activos se colocan en la misma línea de bus, al mismo tiempo.

Colisiones Dos (o más) señales de salida conectadas entre sí, las cuales tratan de excitar un punto común a niveles de voltaje distintos. *Vea también* Colisión de bus.

Comandos del preprocesador Comandos de compilador que se procesan antes que el código del programa principal, para poder controlar la forma en que se debe interpretar el código.

Comentarios Texto que se agrega a cualquier archivo de diseño o programa de computadora de HDL para describir el propósito y la operación del código en general, o de instrucciones individuales en el código. En los comentarios también puede incluirse documentación relacionada con el autor, la fecha, la revisión, etcétera.

Comparador de magnitud Circuito digital que compara dos cantidades binarias de entrada y genera salidas para indicar si las entradas son iguales o, en caso contrario, indica cuál es mayor.

Comparador de voltaje analógico Circuito que compara la magnitud de dos voltajes de entrada analógicos y produce una salida que indica cuál entrada es mayor.

Compatibilidad de terminales Cuando las correspondientes terminales en dos circuitos integrados distintos tienen las mismas funciones.

Compatible eléctricamente Cuando dos circuitos integrados de distintas series lógicas pueden conectarse en forma directa, sin necesidad de tomar medidas especiales para asegurar su correcta operación.

Compilador Un programa que traduce un archivo de texto escrito en un lenguaje de alto nivel en un archivo binario que puede cargarse en un dispositivo programable, tal como un PLD o en la memoria de una computadora.

Complemento *Vea* Inversión.

Complemento a 1 Resultado que se obtiene cuando se complementa cada uno de los bits de un número binario.

COMPONENT Palabra clave de VHDL que se utiliza en la parte superior de un archivo de diseño para proporcionar información sobre un componente de biblioteca.

Comprobador de paridad Circuito que toma un conjunto de bits de datos (incluyendo el bit de paridad) y comprueba que tenga la paridad correcta.

Compuerta AND Circuito digital que implementa la operación AND. La salida de este circuito es nivel ALTO (nivel lógico 1) si y sólo si todas sus entradas son ALTAS.

Compuerta de transmisión *Vea* Interruptor bilateral.

Compuerta NAND Circuito lógico que opera como una compuerta AND seguida de un INVERSOR. La salida de una compuerta NAND está en BAJO (nivel 0 lógico) sólo si todas las entradas están en ALTO (nivel 1 lógico).

Compuerta NOR Circuito lógico que opera como una compuerta OR seguida de un INVERSOR. La salida de una compuerta NOR está en BAJO (nivel 0 lógico) cuando al menos una de las entradas se encuentra en ALTO (nivel 1 lógico).

Compuerta OR Circuito digital que implementa la operación OR. La salida de este circuito está en ALTO (nivel 1 lógico) si al menos una de sus entradas está en ALTO.

Computadora digital Sistema de hardware que realiza operaciones lógicas y aritméticas, manipula datos y realiza decisiones.

Concatenar Un término que se utiliza para describir el arreglo o la unión de dos o más objetos de datos en conjuntos ordenados.

Concurrentes Eventos que ocurren en forma simultánea (al mismo tiempo). En HDL, los circuitos generados por instrucciones concurrentes no se ven afectados por el orden o la secuencia de las instrucciones en el código.

Conjunto Agrupación de variables o señales concatenadas.

Conmutación Proceso de cambio de un estado binario al otro.

Constantes Nombres simbólicos que pueden usarse para representar valores numéricos (escalares) fijos.

Contador ascendente Contador que cuenta en forma ascendente desde 0 hasta una cuenta máxima.

Contador ascendente/descendente Contador que puede contar en forma ascendente o descendente, dependiendo de la manera en que se activen sus entradas.

Contador asíncrono Tipo de contador en el cual la salida de cada flip-flop sirve como la señal de reloj para el siguiente flip-flop en la cadena.

Contador BCD Contador binario que cuenta desde 0000_2 hasta 1001_2 antes de reiniciarse.

Contador binario Grupo de flip-flops conectados en un arreglo especial, en el cual los estados de los flip-flops representan el número binario equivalente del número de pulsos que han ocurrido en la entrada del contador.

Contador con autocorrección Un contador que siempre progresa a su secuencia destinada, sin importar su estado inicial.

Contador con capacidad de precarga Contador que puede preestablecerse a cualquier conteo inicial, ya sea en forma síncrona o asíncrona.

Contador de anillo Registro de desplazamiento en el cual la salida del último flip-flop se conecta a la entrada del primer flip-flop.

Contador de décadas Cualquier contador capaz de pasar por 10 estados lógicos distintos.

Contador de frecuencia Circuito que puede medir y mostrar la frecuencia de una señal.

Contador de ondulación (rizos) *Vea* Contador asíncrono.

Contador de regeneración Contador que lleva el registro de las direcciones de fila durante una operación de regeneración de DRAM.

Contador descendente Contador que cuenta desde una cuenta máxima hasta cero.

Contador en paralelo *Vea* Contador síncrono.

Contador Johnson Registro de desplazamiento en el cual la salida invertida del último flip-flop se conecta a la entrada del primer flip-flop.

Contador múltietapas Contador en el cual se conectan varias etapas de contadores, de manera que la salida de una etapa sirve como entrada de reloj de la siguiente etapa para lograr un mayor alcance de conteo o división de frecuencia.

Contador síncrono Contador en el cual todos los flip-flops se sincronizan con el reloj y trabajan en forma simultánea.

Controlador de DRAM CI que se utiliza para manejar las operaciones de regeneración y multiplexaje de direcciones que necesitan los sistemas de DRAM.

Conversión de paralelo a serial Proceso mediante el cual todos los bits de datos se presentan de manera simultánea a la entrada de un circuito y después se transmiten un bit a la vez hacia su salida.

Convertidor Analógico-Digital (ADC) Circuito que convierte una entrada analógica en su correspondiente salida digital.

Convertidor analógico-digital de doble pendiente Tipo de convertidor analógico-digital que carga un capacitor en forma lineal, a partir de una corriente proporcional a V_A durante un intervalo fijo de tiempo y después incrementa un contador a medida que el capacitor se descarga en forma lineal hasta llegar a 0.

Convertidor Digital-Analógico (DAC) Circuito que convierte una entrada digital en su correspondiente salida analógica.

DAC bipolar Convertidor digital-analógico que acepta números binarios con signo como entrada y produce el valor de salida analógico, positivo o negativo, correspondiente.

DAC de escalera R/2R Tipo de convertidor analógico-digital cuyos valores de las resistencias varían sólo en un factor de 2 a 1.

Datos Representaciones binarias de valores numéricos o de información no numérica en un sistema digital. Los datos se utilizan y a menudo se modifican mediante un programa de computadora.

Decodificador Circuito digital que convierte un código binario de entrada en una sola salida activa correspondiente al valor presentado en el código.

Decodificador 1 de 10 *Vea* Decodificador de BCD a decimal.

Decodificador activo en ALTO (BAJO) Decodificador que produce un nivel lógico ALTO (BAJO) en su salida.

Decodificador de 4 a 10 *Vea* Decodificador de BCD a decimal.

Decodificador de BCD a decimal Decodificador que convierte una entrada en BCD en una salida decimal equivalente.

Decodificador/reforzador de BCD a 7 segmentos Circuito digital que toma una entrada BCD de cuatro bits y activa las salidas requeridas para visualizar el dígito decimal equivalente en una pantalla de 7 segmentos.

Decodificar Acción de identificar una combinación binaria específica para poder mostrar su valor o reconocer su presencia.

Defasamiento de reloj Llegada de una señal de reloj en las entradas de reloj de distintos flip-flops, en tiempos diferentes como resultado de retrasos por tiempos de propagación.

DEFAULTS Palabra clave de AHDL que se utiliza para establecer un valor predeterminado para una señal combinacional, en caso de que el código no especifique un valor de manera explícita.

Demultiplexor (DEMUX) Circuito lógico que, dependiendo del estado de sus entradas de selección, canaliza su entrada de datos hacia una de varias salidas de datos.

Densidad Una medida relativa de capacidad para almacenar bits en una cantidad de espacio dada.

Desacoplamiento de la fuente de energía Conexión de un pequeño capacitor RF entre tierra y V_{CC} cerca de cada circuito integrado TTL en una tarjeta de circuitos.

Desbordamiento Cuando en el proceso de sumar números binarios con signo, se genera un acarreo de 1 desde la posición MSB del número hacia la posición del bit de signo.

Descarga Proceso de transferir archivos de salida hacia un dispositivo de programación.

Descarga Electrostática (ESD) El acto, con frecuencia perjudicial, de transferencia de una carga electrostática de una superficie a otra. Este impulso de corriente puede destruir dispositivos electrónicos.

Deshabilitar Acción en la cual se evita que un circuito realice su función normal, como pasar una señal de entrada hacia su salida.

Desplazador de barril Registro que puede desplazar de forma muy eficiente un número binario a la izquierda o a la derecha por cualquier número de posiciones de bits.

Diagrama de tiempos Descripción de los niveles lógicos en relación con el tiempo.

Diagrama de transición de estado Representación gráfica de la operación de un circuito binario secuencial que muestra la secuencia de estados de FF individuales y las condiciones necesarias para las transiciones de un estado al siguiente.

Digitalización Proceso por el cual una señal analógica se convierte en datos digitales.

Dígito binario Bit.

Dígito más significativo (MSD) Dígito que transmite la ponderación mayor en un número determinado.

Dígito menos significativo (LSD) Dígito que lleva la ponderación menor en un número determinado.

Dirección (Código de) Número que identifica en forma única la ubicación de una palabra en memoria.

Disco duro Disco magnético de metal rígido que se utiliza como medio de almacenamiento masivo.

Diseño jerárquico Método en el que para diseñar un proyecto, éste se descompone en los módulos que lo conforman, cada uno de los cuales puede a su vez descomponerse en módulos más simples.

Disparado por flanco Manera en la cual un flip-flop se activa mediante una transición de señal. Un flip-flop se puede disparar por flanco positivo o negativo.

Disparador de Schmitt Circuito digital que acepta una señal de entrada con cambios lentos y produce en la salida una transición rápida, libre de oscilaciones.

Disparo Señal de entrada de un flip-flop o de un circuito monoestable que hace que la salida cambie de estados, dependiendo de las condiciones de las señales de control.

Dispositivo Lógico Programable (PLD) CI que contiene un extenso número de funciones lógicas interconectadas. El usuario puede programar el CI para una función específica al interrumpir de manera selectiva las interconexiones apropiadas.

Distancia entre terminales La distancia entre los centros de terminales adyacentes en un CI.

Distribuidores de datos *Vea* Demultiplexor.

Dividir y conquistar Técnica de detección y solución de problemas en la cual se llevan a cabo pruebas que eliminan la mitad de las posibles causas de falla.

División de frecuencia El uso de circuitos flip-flop para producir una forma de onda de salida cuya frecuencia es igual a la frecuencia del reloj de entrada dividida por cierto valor entero.

ECL Lógica de emisor acoplado; también se le conoce como *lógica en modo de corriente*.

Efecto de cierre Condición de corriente peligrosamente alta en un circuito integrado CMOS, la cual se produce debido a picos de alto voltaje o sobreoscilación en las terminales de entrada y salida del dispositivo.

Elementos lógicos Término utilizado por Altera Corporation para describir los bloques de construcción de su familia FLEX10K de PLDs. Los elementos lógicos se programan como una tabla de referencia en una memoria RAM.

ELSE Estructura de control que se utiliza en conjunto con IF/THEN para realizar una acción alternativa, en caso de que la condición sea falsa. Una instrucción IF/THEN/ELSE siempre realiza una de dos acciones.

ELSIF Estructura de control que puede utilizarse varias veces después de una instrucción IF para seleccionar una de varias opciones al describir la operación de un circuito, con base en si las expresiones asociadas son verdaderas o falsas.

Empalme de memoria Habilitación redundante de un dispositivo de memoria en más de un intervalo de direcciones, como resultado de una decodificación incompleta de la dirección.

Enlace de fusible Material conductor que puede hacerse no conductor (es decir, abierto) si se pasa demasiada corriente a través del mismo.

ENTITY Palabra clave de VHDL que se utiliza para definir la estructura básica de bloques de un circuito. A esta palabra le siguen un nombre para el bloque y las definiciones de sus puertos de entrada/salida.

Entrada diferencial Método para conectar una señal analógica a las entradas + y – de un circuito analógico de las cuales ninguna es tierra, de tal forma que el circuito analógico actúe con base en la diferencia de voltaje entre las dos entradas.

Entrada en serie-Salida en paralelo Tipo de registro que puede cargarse con datos en serie y tiene salidas en paralelo disponibles.

Entrada en serie-Salida en serie Tipo de registro que puede cargarse con datos en serie y que sólo tiene una salida en serie.

Entrada flotante Señal de entrada que se deja desconectada en un circuito lógico.

Entradas asíncronas Entradas del flip-flop que pueden afectar la operación del flip-flop de forma independiente a las entradas síncronas y del reloj.

Entradas de control Señales de entrada sincronizadas con la transición activa de reloj que determinan el estado de la salida de un flip-flop.

Entradas de control síncronas *Vea* Entradas de control.

Entradas predominantes Sinónimo de "entradas asíncronas".

Equivalente funcional Cuando las funciones lógicas realizadas por dos circuitos integrados distintos son idénticas.

Error a escala completa Término utilizado por ciertos fabricantes de convertidores digitales-analógicos para especificar la precisión de un convertidor digital-analógico. Se define como la máxima desviación de la salida de un convertidor digital-analógico de su valor ideal esperado.

Error de cuantización Error producido por la resolución distinta de cero de un convertidor analógico-digital. Es un error inherente del dispositivo.

Error de desplazamiento Desviación del valor ideal de 0 V en la salida de un convertidor digital-analógico cuando la entrada consta sólo de 0s. En realidad existe un voltaje de salida muy pequeño para esta situación.

Error de linealidad Término utilizado por algunos fabricantes de convertidores digital-analógico para especificar la precisión de un dispositivo. Se define como la máxima desviación en el tamaño del escalón, con respecto al tamaño ideal de escalón.

ESCRITURA Término utilizado para describir la condición cuando la CPU envía datos a otro elemento.

Estado casi estable Estado en el cual se dispara un circuito de un solo disparo en forma temporal (por lo general $Q = 1$), antes de regresar a su estado estable (por lo general $Q = 0$).

Estado CLEAR El estado de un flip-flop en el que $Q = 0$.

Estado RESET El estado de un flip-flop en el que $Q = 0$.

Estado SET El estado de un flip-flop cuando $Q = 1$.

Estroboscopio Otro nombre para una entrada de habilitación que, por lo general, se utiliza para fijar un valor en un registro.

Estructuras de control de decisiones Las instrucciones y la sintaxis que permiten elegir entre dos o más opciones en el código.

EVENT Palabra clave de VHDL que se utiliza como un atributo que se adjunta a una señal para detectar una transición de esa señal. Por lo general, un evento indica un estado de modificación de señal.

Filtrado por interpolación Otro nombre para sobremuestreo. La interpolación hace referencia a los valores intermedios que se insertan en la señal digital, para suavizar la forma de onda.

Firmware Programas de computadora que se almacenan en ROM.

Flip-flop Dispositivo de memoria capaz de almacenar un nivel lógico.

Flip-flop D *Vea* Flip-flop sincronizado por reloj en D.

Flip-flop sincronizado por reloj en D Tipo de flip-flop en el cual la entrada *D* (datos) es la entrada síncrona.

Flip-flop sincronizado por reloj en J-K Tipo de flip-flop en el cual las entradas *J* y *K* son las entradas síncronas.

Flip-flop sincronizado por reloj en S-R Tipo de flip-flop en el cual las entradas S y R (set y reset) son las entradas síncronas.

Flip-flops sincronizados por reloj Flip-flops que tienen una entrada de reloj.

Forma de complemento a 2 Resultado que se obtiene cuando se suma un 1 a la posición del bit menos significativo de un número binario en la forma de complemento a 1.

Forma de onda tipo escalera Tipo de forma de onda que se genera en la salida de un convertidor digital-analógico, a medida que su señal de entrada digital cambia en forma incremental.

Forma de producto de sumas Expresión lógica que consiste de dos o más términos OR (sumas) a los que se aplica una operación AND. Maxitérminos.

Forma de suma de productos Expresión lógica que consiste de dos o más términos AND (productos) a los que se aplica la operación OR. Minitérminos

Frecuencia El número de ciclos por unidad de tiempo de una forma de onda periódica.

Frecuencia de muestreo La velocidad a la cual se digitaliza una señal analógica, dado en muestras por segundo.

Frecuencia máxima de reloj ($f_{MÁX}$) Frecuencia más alta que puede aplicarse a la entrada de reloj de un flip-flop y conservar la certeza de disparo.

Generación de función lógica Implementación de una función lógica en forma directa a partir de una tabla de verdad, por medio de un CI digital tal como un multiplexor.

Generador de funciones Circuito que produce distintas formas de onda. Puede construirse a partir de una memoria ROM, un DAC y un contador.

Generador de paridad Circuito que toma un conjunto de bits de datos y produce el bit de paridad correcto para esos datos.

Generador de reloj controlado por cristal Circuito que utiliza un cristal de cuarzo para generar una señal de reloj a una frecuencia precisa.

GENERATE Palabra clave de VHDL que se utiliza con la instrucción FOR para definir en forma iterativa varios componentes similares e interconectarlos.

GSI Integración de giga-escala (1,000,000 de compuertas o más).

Habilitación Acción en la cual se permite a un circuito realizar su función normal, como pasar una señal de entrada hacia su salida.

Habilitación de conteo Una entrada en un contador síncrono que controla si las salidas deben responder a una transición activa de reloj o ignorarla.

Hecho a la medida Un circuito integrado de aplicación específica (ASIC) que está diseñado y fabricado en su totalidad a partir de elementos fundamentales de dispositivos electrónicos, tales como transistores, diodos, resistencias y capacitores.

IEEE/ANSI Instituto de Ingenieros Eléctricos y Electrónicos/Instituto Estadounidense de Estándares Nacionales, ambas organizaciones profesionales que establecen estándares.

IF/THEN Estructura de control que evalúa una condición y realiza una acción si la condición es verdadera o ignora la acción y continúa si la condición es falsa.

Indeterminado De un nivel de voltaje lógico, fuera del intervalo requerido de voltajes para un 0 lógico y para un 1 lógico.

Índice Otro nombre para el número de elemento de cualquier bit dado en un arreglo de bits.

Inmunidad al ruido Habilidad de un circuito para tolerar los voltajes de ruido en sus entradas.

Instrucción de asignación concurrente Una instrucción en AHDL o VHDL que describe un circuito que trabaja en forma concurrente con todos los demás circuitos descritos por instrucciones concurrentes.

Instrucciones Códigos binarios que indican a una computadora qué operación debe realizar. Un programa está compuesto de una secuencia ordenada de instrucciones.

INTEGER En VHDL es el tipo de objeto de datos que representa un valor numérico entero.

Interfase Unión de dispositivos distintos de tal forma que son capaces de funcionar de una manera compatible y coordinada; conexión de la salida de un sistema a la entrada de un sistema distinto con diferentes características eléctricas.

Interruptor bilateral Circuito CMOS que actúa como un interruptor de un polo – un tiro (SPST), controlado mediante un nivel lógico de entrada.

Intervalo de muestreo Ventana de tiempo durante la cual un contador de frecuencia realiza muestreos y en base a ello determina la frecuencia desconocida de una señal.

Inversión Hacer que un nivel lógico cambie al estado opuesto.

INVERSOR También se le conoce como circuito NOT; circuito lógico que implementa la operación NOT. Un INVERSOR sólo tiene una entrada y su nivel lógico de salida es siempre el opuesto del nivel lógico de esa entrada.

JEDEC Consejo Común de Ingeniería de Dispositivos Electrónicos, que estableció estándares para las asignaciones de las terminales de circuitos integrados y el formato de archivos de PLD.

Jerarquía Grupo de tareas ordenadas por orden de magnitud, importancia o complejidad.

JTAG Grupo Común de Acción de Pruebas, el cual creó una interfaz estándar que permite el acceso al funcionamiento interno de un CI para fines de prueba, control y programación.

Latch Tipo de flip-flop; también, la acción mediante la cual la salida de un circuito lógico captura y almacena el valor de una entrada.

Latch D Circuito que contiene un latch de compuerta NAND y dos compuertas NAND de dirección.

Latch de compuerta NAND Flip-flop construido a partir de dos compuertas NAND con retroalimentación cruzada.

Latch de compuerta NOR Flip-flop construido a partir de dos compuertas NOR con retroalimentación cruzado.

Latencia El retraso inherente asociado con la lectura de datos de una memoria DRAM. Se produce debido a los requerimientos de sincronización relacionados con el proceso de proporcionar las direcciones de fila y de columna, y el tiempo necesario para que las salidas de datos se estabilicen.

LCD Pantalla de Cristal Líquido.

Lectura Término utilizado para describir la condición cuando la CPU recibe datos de otro elemento.

LED Diodo Emisor de Luz.

Lenguaje de Descripción de Hardware (HDL) Método basado en texto para describir el hardware digital que sigue una determinada sintaxis para representar objetos de datos y estructuras de control.

Lenguaje de Descripción de Hardware (VHDL) de Circuito Integrado de muy Alta Velocidad (VHSIC) Lenguaje de descripción de hardware desarrollado por el Departamento de Defensa de Estados Unidos para documentar, simular y sintetizar sistemas digitales complejos.

Lenguaje de Descripción de Hardware de Altera (AHDL) Un HDL propietario desarrollado por Altera Corporation para programar sus dispositivos lógicos programables.

Línea de datos bidireccional Término que se utiliza cuando una línea de datos funciona ya sea como línea de entrada o de salida, dependiendo del estado de las entradas de habilitación.

Lista de sensibilidad La lista de señales utilizadas para invocar la secuencia de instrucciones en un proceso (PROCESS).

Literales En VHDL, un valor escalar o patrón de bits que se va a asignar a un objeto de datos.

Lógica de Arreglo Programable (PAL) Clase de dispositivos lógicos programables. Su matriz AND es programable, mientras que su matriz OR es fija.

Lógica de drenado de corriente Familia lógica en la cual la salida de un circuito lógico consume la corriente de la entrada del circuito lógico que está excitando.

Lógica de emisor acoplado *Vea* ECL.

Lógica de suministro de corriente Familia lógica en la cual la salida de un circuito lógico suministra la corriente a la entrada del circuito lógico que está excitando.

Lógica estándar La extensa variedad de componentes de CI digitales básicos, disponibles en diversas tecnologías como chips MSI, SSI.

Longitud de palabra Número de bits en las palabras binarias con las que opera un sistema digital.

LSI Integración de gran escala (de 100 a 9999 compuertas).

MAC Abreviación de Unidad de Multiplicación y Acumulación, la sección de hardware de un DSP que mul-

tiplica una muestra con un coeficiente y después acumula estos productos.

MACHINE Palabra clave de AHDL utilizada para crear una máquina de estado en un archivo de diseño.

Macrocelda Un circuito formado de un grupo de componentes digitales básicos, tales como compuertas AND, compuertas OR, registros y circuitos de control de tres estados que pueden interconectarse dentro de un PLD por medio de un programa.

Macrocelda de Lógica de Salida (OLMC) Grupo de elementos lógicos (compuertas, multiplexores, flip-flops, búferes) en un PLD que pueden configurarse de varias formas.

Macrofunciones Término utilizado por Altera Corporation para describir las descripciones predefinidas de hardware en sus bibliotecas, las cuales representan componentes estándar de circuitos integrados.

Mapa de Karnaugh (Mapa K) Forma bidimensional de una tabla de verdad que se utiliza para simplificar una expresión de suma de productos.

Mapa de memoria Diagrama de un sistema de memoria que muestra la capacidad disponible de todos los dispositivos de memoria existentes, así como el espacio de memoria disponible para la expansión.

Máquinas de estado Circuito secuencial que progresa a través de varios estados definidos.

Margen de ruido Medida cuantitativa de inmunidad al ruido.

Matriz de términos de entrada Parte de un dispositivo lógico programable que permite conectar o desconectar de manera selectiva las entradas de los circuitos lógicos internos.

Medio sumador Circuito lógico con dos entradas y dos salidas. Las entradas son un bit del primer sumando y un bit del sumado. Las salidas son el bit de resultado producido por la adición de los sumandos y el bit de acarreo resultante (C_{OUT}), lo cual se sumará a la siguiente etapa.

Megafunciones Un bloque de construcción complejo o de alto nivel, disponible en la biblioteca de Altera.

Memoria Habilidad de la salida de un circuito de permanecer en un estado, aún y cuando se haya removido la condición de entrada que produjo ese estado.

Memoria auxiliar La parte de la memoria de una computadora que está separada de su memoria principal de trabajo. Por lo general, tiene alta densidad y alta capacidad, como un disco magnético.

Memoria de Acceso Aleatorio (RAM) Memoria en la cual el tiempo de acceso es el mismo para cualquier ubicación.

Memoria de Acceso Secuencial (SAM) Memoria en la cual el tiempo de acceso varía dependiendo de la ubicación de almacenamiento de los datos.

Memoria de Lectura/Escritura (RWM) Cualquier memoria de la que se puedan leer o escribir datos con igual facilidad.

Memoria de Sólo Lectura (ROM) Dispositivo de memoria diseñado para aplicaciones en donde la relación de operaciones de lectura con las operaciones de escritura es muy alta.

Memoria en cinta magnética Memoria de almacenamiento masivo que almacena datos en forma de puntos magnetizados en una cinta de plástico con recubrimiento magnético.

Memoria en disco magnético Memoria de almacenamiento masivo que almacena datos en forma de puntos magnetizados en una superficie de disco giratoria y plana.

Memoria en disco óptico Clase de dispositivos de memoria masivos que utilizan un rayo láser para escribir y leer en un disco con recubrimiento especial.

Memoria FIFO (Primero en Entrar, Primero en Salir) Memoria semiconductora de acceso secuencial en la cual las palabras de datos se leen en el mismo orden en el que se escribieron.

Memoria flash CI de memoria no volátil que tiene el acceso de alta velocidad y la capacidad de borrarse integrada de la memoria EEPROM, pero con mayores densidades y a un menor costo.

Memoria no volátil Memoria que conserva la información almacenada sin necesidad de energía eléctrica.

Memoria principal Parte de alta velocidad en la memoria de una computadora que almacena el programa y los datos con los que ésta se encuentra trabajando. También se le conoce como *memoria de trabajo*.

Memoria volátil Memoria que requiere energía eléctrica para mantener almacenada la información.

Método de paridad Esquema utilizado para la detección de errores durante la transmisión de datos.

Microcomputadora Miembro más reciente de la familia de computadoras, que consiste de un chip microprocesador, chips de memoria y chips de interfaz de E/S. En algunos casos, todo lo antes mencionado se encuentra dentro de un solo CI.

Microcontrolador Pequeña microcomputadora utilizada como controlador dedicado para una máquina, una pieza de equipo o un proceso.

Microcontrolador embebido Un microcontrolador que está implantado en un producto comercial, como un reproductor de DVD o un electrodoméstico.

Microprocesador (MPU) Chip LSI que contiene la unidad central de procesamiento (CPU).

Minuendo Número a partir del cual se va a restar el sustraendo.

Modelo de Mealy Modelo de máquina de estados en el cual las señales de salida se controlan mediante entradas combinacionales, así como el estado del circuito secuencial.

Modelo de Moore Modelo de máquina de estados en el cual las señales de salida se controlan sólo mediante las salidas de circuitos secuenciales.

Modo El atributo de un puerto en un circuito digital que lo define como entrada, salida o bidireccional.

Modo de conmutación Modo en el cual un flip-flop cambia de estados para cada pulso del reloj.

Modulación Sigma/Delta Método en el que se muestrea una señal analógica y se convierten sus puntos de datos en un flujo de bits de datos en serie.

Monoestable (One-Shot) Circuito que pertenece a la familia de flip-flops, pero que sólo tiene un estado estable (por lo general, cuando $Q = 0$).

Monoestable digital Un circuito monoestable que para generar el pulso utiliza un contador y un reloj, en vez de un circuito RC como base de tiempo.

Monoestable no redisparable Tipo de circuito de un solo disparo que no responde a una señal de entrada de disparo mientras se encuentra en su estado casi estable.

Monoestable redisparable Tipo de circuito de un solo disparo (One-Shot) que responde a una señal de entrada de disparo mientras se encuentra en su estado casi estable.

Monotonicidad Propiedad mediante la cual la salida de un convertidor digital-analógico aumenta a medida que se incrementa la entrada binaria.

Montaje superficial Método de fabricación de tarjetas de circuitos en las que los circuitos integrados están soldados a bases conductoras en la superficie de la tarjeta.

MOSFET Transistor de efecto de campo de metal-óxido-semiconductor.

MSI Integración de mediana escala (de 12 a 99 compuertas).

Muestreo El proceso de adquirir y digitalizar un punto de datos de una señal analógica, en un momento de tiempo dado.

Muestreo estroboscópico Técnica que se utiliza a menudo para eliminar picos en la decodificación.

Multiplexaje Proceso de selección de uno de varios orígenes de datos de entrada y transmisión los datos seleccionados hacia un solo canal de salida.

Multiplexaje de direcciones Multiplexaje que se utiliza en memorias RAM dinámicas para reducir el número de terminales en un CI. Implica el proceso de separar las dos mitades de una dirección completa en el CI, y presentarlas en forma secuencial.

Multiplexor (MUX) Circuito lógico que, dependiendo del estado de sus entradas de selección, canalizará una de varias entradas de datos hacia su salida.

Multivibrador estable Circuito digital que oscila entre dos estados de salida inestables.

Multivibrador monoestable *Vea* Monoestable.

Negación Operación de conversión de un número positivo en su equivalente negativo, o viceversa. Un número binario con signo se niega mediante la operación de complemento a 2.

Nibble Un grupo de cuatro bits.

Nivel de abstracción de comportamiento Una técnica para describir un circuito digital que se enfocada a la forma en que el circuito reacciona a sus entradas.

Nivel de abstracción estructural Técnica para describir un circuito digital que se enfoca en conectar puertos de módulos con señales.

Nivel lógico Estado de una variable de voltaje. Los estados 1 (ALTO) y 0 (BAJO) corresponden a los dos intervalos de voltaje utilizables de un dispositivo digital.

Nivel lógico activo Nivel lógico en el que un circuito se considera activo. Si el símbolo para el circuito incluye un pequeño círculo, también llamado Burbuja, significa que el circuito es activo en BAJO. Por otro lado, si tiene la línea directa entonces es activo en ALTO.

N-MOS (Metal-Óxido-Semiconductor de canal N) Tecnología de circuitos integrados que utiliza MOSFETs de canal N como el elemento principal de un circuito.

No asignado Término que se utiliza para describir el estado de una señal lógica; sinónimo de "inactivo".

"No importa" (X) Situación en la cual el nivel de salida de un circuito para un conjunto dado de condiciones de entrada puede asignarse como un 1 o un 0.

NODE Palabra clave de AHDL utilizada para declarar una variable intermedia (objeto de datos) que es local para ese subdiseño.

Nodo oculto Un punto definido en un circuito al cual no se puede acceder desde el exterior del circuito.

Notación de dependencia Método que se utiliza para representar en forma simbólica la relación entre las entradas y las salidas de circuitos lógicos. Este método emplea el uso de símbolos calificadores incrustados cerca del centro superior o centro geométrico de un elemento tipo símbolo.

Número MOD Número de distintos estados en la secuencia de un contador; la relación de división de frecuencia del contador.

Objetos Varias maneras de representar datos en el código de cualquier HDL.

Observación/análisis Proceso utilizado para detectar fallas en circuitos o sistemas y poder diagnosticar las posibles fallas antes de emplear un instrumento de detección de fallas. Cuando se utiliza este proceso, el técnico detector de fallas debe comprender la operación del circuito, observar los síntomas de la falla y después razonar toda la operación.

Octetos Grupos de ocho 1s adyacentes unos con otros dentro de un mapa de Karnaugh.

Ondulación de acarreo *Vea* Propagación de acarreo.

Operación AND Operación de álgebra booleana en la que el símbolo se utiliza para indicar la operación AND lógica entre dos o más variables lógicas. El resultado de la operación AND será ALTO (nivel lógico 1) si y sólo si todas las variables son ALTAS.

Operación de carga Transferencia de datos hacia un flip-flop, un registro, un contador o una ubicación de memoria.

Operación de escritura Operación en la cual se coloca una nueva palabra en una ubicación específica de memoria.

Operación de lectura Operación en la cual se detecta una palabra en una ubicación específica de memoria, y es posible que se transfiera a otro dispositivo.

Operación NOT Operación de álgebra booleana en la cual se utiliza la barra horizontal superior ($\bar{}$) o el símbolo de primo ($'$) para indicar la inversión de una o más variables lógicas.

Operación OR Operación de álgebra booleana en la cual se utiliza el símbolo + para indicar la operación con OR de dos o más variables lógicas. El resultado de la operación OR será ALTO (nivel 1 lógico) si una o más variables están en ALTO.

Oscilador Controlado por Voltaje (VCO) Circuito que produce una señal de salida con una frecuencia proporcional al voltaje que se aplica en su entrada.

Osciloscopio de almacenamiento digital Instrumento que muestrea, digitaliza, almacena y muestra formas de onda de voltaje analógico.

PACKAGE Palabra clave en VHDL utilizada para definir un conjunto de elementos globales que están disponibles para otros módulos.

Palabra Grupo de bits que representa una cierta unidad de información.

Palabra de computadora Grupo de bits que forman la unidad primaria de información en una computadora.

Palabra de memoria Grupo de bits en memoria que representan instrucciones o datos de cierto tipo.

Paquete dual en línea (DIP) Un paquete de CI muy común, en el cual dos filas paralelas de terminales se insertan en

un zócalo o a través de hoyos perforados en una tarjeta de circuito impreso.

Periódico Un ciclo que se repite con regularidad, en tiempo y forma.

Periodo La cantidad de tiempo requerido para un ciclo completo de un evento periódico o forma de onda.

Pico *Vea* Transitorio.

Píxel Pequeños puntos que conforman una imagen gráfica en una pantalla.

Plano posterior Conexión eléctrica común para todos los segmentos de un LCD.

PLD Complejo (CPLD) Clase de PLDs que contiene una matriz de bloques tipo PAL, los cuales pueden interconectarse.

PLD de Alta Capacidad (HCPLD) Un PLD con miles de compuertas lógicas y muchos recursos de macroceldas programables, junto con recursos de interconexión muy flexibles.

PLD Simple (SPLD) Un PLD con unos cuantos cientos de compuertas lógicas y tal vez unas cuantas macroceldas programables disponibles.

P-MOS (Metal-Óxido-Semiconductor de canal P) Tecnología de circuitos integrados que utiliza MOSFETs de canal P como el elemento principal de un circuito.

Polaridad de salida programable Característica de muchos PLDs en donde una compuerta XOR con fusible de polaridad le proporciona al diseñador la opción de invertir o no invertir la salida de un dispositivo.

PORT MAP Palabra clave de VHDL que va antes de la lista de conexiones especificadas entre componentes.

Preescalador Un circuito contador que toma la frecuencia de referencia base y le aplica una escala al dividir la frecuencia y reducirla a una proporción requerida por el sistema.

PRESET Entrada asíncrona utilizada para establecer $Q = 1$ de manera inmediata.

Primer sumando Número al cual se le sumará un sumando.

Primitiva lógica Descripción de un componente fundamental de un circuito que está integrado en el sistema de bibliotecas de MAX+PLUS II.

Procesamiento digital de señales (DSP) Método para realizar cálculos repetitivos sobre un flujo entrante de datos digitales para realizar cierto tipo de condicionamiento de señal. Los datos, por lo general, son muestras digitalizadas de una señal analógica.

PROCESS Palabra clave de VHDL que define el principio de un bloque de código que describe un circuito que debe responder cada vez que ciertas señales (en la lista de sensibilidad) cambian de estado. Todas las instrucciones secuenciales deben ocurrir dentro de un proceso.

Programa Secuencia de instrucciones codificadas en binario diseñadas para que una computadora realice una tarea específica.

Programa de arranque Programa almacenado en ROM que ejecuta una computadora al encenderla.

Programable Sólo una Vez (OTP) Una amplia categoría de componentes programables, cuya programación se lleva a cabo mediante la alteración permanente de las conexiones (por ejemplo, se funde un elemento de fusible).

Programación El acto de almacenar 1s y 0s en un dispositivo lógico programable para configurar las características de su comportamiento.

Programador Dispositivo utilizado para aplicar los voltajes apropiados para los chips PLD y PROM, a fin de programarlos.

Promedio ponderado Un cálculo promedio de un grupo de muestras que asigna un peso diferente (entre 0.0 y 1.0) a cada muestra.

Propagación de acarreo Retraso intrínseco en el circuito de algunos sumadores en paralelo que impide que el bit de acarreo (C_{OUT}) y el resultado de la suma aparezcan en la salida al mismo tiempo.

Prototipo de función Descripción de texto que contiene todos los atributos esenciales de definición de una función o módulo de biblioteca.

Prueba de escalera Proceso mediante el cual se incrementa la entrada digital de un convertidor digital-analógico y se vigila su salida para determinar si exhibe o no un formato de escalera.

Prueba de precisión estática Prueba en la cual se aplica un valor binario fijo a la entrada de un convertidor digital-analógico y la salida analógica se mide con precisión. El resultado medido debe encontrarse dentro del intervalo esperado, especificado por el fabricante del convertidor digital-analógico.

Pulsador lógico Herramienta de prueba que genera un pulso de corta duración cuando se opera en forma manual.

Pulso Un cambio momentáneo del estado lógico que representa un evento para un sistema digital.

Punto binario Marca que separa la parte entera de la parte fraccionaria de una cantidad binaria.

RAM Dinámica (DRAM) Tipo de memoria semiconductora que almacena datos en forma de cargas de capacitor que necesitan regenerarse en forma periódica.

RAM estática (SRAM) RAM semiconductora que almacena información en celdas de flip-flops que no tienen que regenerarse en forma periódica.

RAM no volátil Combinación de un arreglo de RAM y una memoria EEPROM o flash en el mismo CI. La EEPROM sirve como respaldo no volátil para la RAM.

RAS (Estrobo de Dirección de Fila) Señal que se utiliza para fijar la dirección de fila en un chip DRAM.

Rebotes de contacto La tendencia que tienen todos los interruptores mecánicos de vibrar cuando se les fuerza a tomar una nueva posición. Las vibraciones hacen que el circuito haga contacto e interrumpa el contacto en forma repetida, hasta que todas las vibraciones se estabilicen.

Reforzador Término técnico que algunas veces se agrega a la descripción de un CI para indicar que las salidas del circuito pueden operar con límites de corriente y/o voltaje más altos que los de un CI estándar.

Reforzadores de bus Circuitos que refuerzan las salidas de los dispositivos conectados a un bus común; se utilizan cuando un gran número de dispositivos comparten un bus común.

Regeneración Proceso mediante el cual se recargan las celdas de una memoria dinámica.

Regeneración sólo de RAS Método para refrescar la DRAM, en el cual sólo las direcciones de fila se fijan en la DRAM, mediante el uso de la entrada *RAS*.

Registro Grupo de flip-flops capaces de almacenar datos.

Registro de búfer Registro que retiene datos digitales en forma temporal.

Registro de desplazamiento Circuito digital que acepta datos binarios de cierta fuente de entrada y después desplaza estos datos a través de una cadena de flip-flops, un bit a la vez.

Registro de desplazamiento circulante Registro de desplazamiento en el cual una de las salidas del último flip-flop se conecta a la entrada del primer flip-flop.

Registro de entrada en paralelo-salida en paralelo Tipo de registro que puede cargarse con datos en paralelo y que tiene salidas en paralelo.

Registro de entrada en paralelo-salida en serie Tipo de registro que puede cargarse con datos en paralelo pero que sólo tiene salida en serie.

Reloj Señal digital en forma de un tren de pulsos rectangulares o una onda cuadrada.

Representación analógica Representación de una cantidad que varía a través de un intervalo continuo de valores.

Representación digital Representación de una cantidad que varía en pasos discretos, a través de un intervalo de valores.

RESET Término que es sinónimo de "CLEAR".

Resolución En un convertidor digital-analógico, el cambio más pequeño que puede producirse en la salida para un cambio en la entrada digital; también se le conoce como *tamaño de escalón*. En un convertidor analógico-digital, la mínima cantidad mediante la cual debe cambiar la entrada analógica para producir un cambio en la salida digital.

Resolución porcentual Relación del tamaño de escalón con el valor de escala completa de un convertidor digital-analógico. La resolución porcentual también puede definirse como el recíproco del número máximo de escalones de un convertidor digital-analógico.

ROM Programable (PROM) ROM que el usuario puede programar mediante electricidad. No puede borrarse y reprogramarse.

ROM programable y borrable (EPROM) ROM que puede programarse por el usuario mediante electricidad. Por lo general, puede borrarse con luz ultravioleta y reprogramarse casi todas las veces que se desee.

ROM Programable y Borrable Eléctricamente (EEPROM) ROM que puede programarse, borrarse y reprogramarse mediante electricidad.

ROM Programada con Máscara (MROM) ROM que programa el fabricante en base a las especificaciones del cliente. No puede borrarse ni reprogramarse.

Ruido Fluctuaciones espurias de voltaje que pueden presentarse en el entorno y provocar que los circuitos digitales fallen.

Salida a escala completa El máximo valor posible de un convertidor digital-analógico.

Salida de colector abierto Tipo de estructura de salida de algunos circuitos TTL, en los que sólo se utiliza solo un transistor con el colector flotante.

Salida en forma de tótem Término que se utiliza para describir la manera en que se configuran dos transistores bipolares en la salida de la mayoría de los circuitos TTL.

SBD Diodo de barrera Schottky, utilizado en todas las series Schottky TTL.

Schottky de Baja Potencia TTL (LS-TTL) Subfamilia TTL que usa el circuito Schottky TTL idéntico, pero con valores de resistencia mayores.

Schottky TTL Subfamilia TTL que utiliza el circuito TTL estándar básico, sólo que utiliza un diodo de barrera Schottky (SBD) conectado entre la base y el colector de cada transistor, para una conmutación más rápida.

Secuencial Se lleva a cabo un elemento a la vez, en cierto orden. En HDL, los circuitos que se generan mediante instrucciones secuenciales se comportan de manera distinta, dependiendo del orden de las instrucciones en el código.

Selección de chip Entrada para un dispositivo digital que controla si éste realizará o no su función. También se le conoce como *habilitación de chip*.

Selectores de datos *Vea* Multiplexor.

Señal local *Vea* Nodo oculto.

Señalización Diferencial de Bajo Voltaje (LVDS) Tecnología para excitar líneas de datos de alta velocidad en sistemas de bajo voltaje, en la cual se utilizan dos conductores y se invierte la polaridad para diferenciar entre ALTO y BAJO.

SET Entrada a un latch o FF que se utiliza para hacer $Q = 1$.

Sigma (Σ) Letra griega que representa la adición y a menudo se utiliza para identificar los bits de salida de un sumador en paralelo.

Símbolo lógico alterno Un símbolo equivalente que indica el nivel lógico activo de las entradas y las salidas.

Simulador Programa de computadora que calcula los estados de salida correctos de un circuito lógico con base en una descripción del circuito lógico y las entradas actuales.

Sintaxis Las reglas que definen las palabras clave y su distribución, uso, puntuación y formato para un lenguaje dado.

Sistema analógico Combinación de dispositivos diseñados para manipular cantidades físicas que se representan en forma analógica.

Sistema binario Sistema numérico en el cual sólo hay dos valores posibles para un dígito: 0 y 1.

Sistema de magnitud con signo Sistema para representar números binarios con signo, en el cual el bit más significativo representa el signo del número y los bits restantes representan la magnitud del valor binario.

Sistema de valor posicional Sistema en el cual el valor de un dígito depende de su posición relativa.

Sistema decimal Sistema numérico que utiliza 10 dígitos o símbolos diferentes para representar una cantidad.

Sistema digital Combinación de dispositivos diseñados para manipular cantidades físicas representadas en forma digital.

Sistema híbrido Sistema que emplea técnicas tanto analógicas como digitales.

Sistema numérico hexadecimal Sistema numérico que tiene base 16. Para expresar un número hexadecimal se utilizan los dígitos del 0 al 9 más las letras de la A a la F.

Sistema numérico octal Sistema numérico de base 8; para expresar un número octal se utilizan los dígitos del 0 al 7.

Sistemas síncronos Sistemas en los cuales las salidas del circuito pueden cambiar de estado sólo en las transiciones de un reloj.

Sobremuestreo Inserción de puntos de datos entre datos muestreados en una señal digital para facilitar el filtrado de los bordes con cambios bruscos de la forma de onda que sale del DAC.

Sonda lógica Herramienta de diagnóstico digital que detecta e indica el nivel lógico en un punto específico de un circuito.

SSI Integración de pequeña escala (menos de 12 compuertas).

STD_LOGIC En VHDL, un tipo de datos definido como estándar IEEE. Es similar al tipo BIT, sólo que además del 1 o el 0, ofrece más valores posibles.

STD_LOGIC_VECTOR En VHDL, un tipo de datos definido como estándar IEEE. Es similar al tipo BIT_VECTOR, sólo que ofrece más valores posibles además del 1 o el 0 para cada elemento.

SUBDESIGN Palabra clave en AHDL que se utiliza para comenzar la descripción de un circuito.

Submuestreo Adquisición de muestras de una señal a una velocidad menor que el doble de la frecuencia más alta contenida en la señal.

Substrato Pieza de material semiconductor que forma parte del bloque de construcción de cualquier CI digital.

Suma de comprobación Palabra de datos especial que se almacena en la última ubicación de la ROM. Se deriva de la suma de todas las demás palabras de datos en la ROM y se utiliza para fines de comprobación de errores.

Sumador completo Circuito lógico con tres entradas y dos salidas. Las entradas son un bit de acarreo (C_{ENT}) de una etapa anterior, un bit del sumando principal y un bit del sumando, respectivamente. Las salidas son el bit de resultado y el bit de acarreo (C_{SAL}) que se produce por la suma ambos sumandos y C_{ENT}.

Sumador en paralelo Circuito digital fabricado a partir de sumadores completos y que se utiliza para sumar todos los bits de de ambos sumandos al mismo tiempo.

Sumador/Restador Un circuito sumador que puede restar mediante el complemento (negación) de uno de los operandos. *Vea también* Sumador en paralelo.

Sumando Número que se sumará a otro.

Supercomputadoras Computadoras con la mayor velocidad y poder de cómputo.

Sustraendo Número que va a restarse de un minuendo.

Tabla de estado Tabla cuyas entradas representan la secuencia de estados de FF individuales (es decir, 0 o 1) para un circuito secuencial.

Tabla de estado actual-estado siguiente Una tabla que lista cada uno de los posibles estados presentes de un circuito secuencial e identifica el siguiente estado que corresponda.

Tabla de excitación de circuito Tabla que muestra las posibles transiciones de estado PRESENTE a SIGUIENTE de un circuito y los niveles *J* y *K* requeridos en cada flip-flop.

Tabla de excitación J-K Tabla que muestra las condiciones de entrada *J* y *K* requeridas para cada posible transición de estado de un flip-flop J-K.

Tabla de Referencia (LUT) Una manera de implementar una sola función lógica al almacenar el estado lógico de salida correcto en una ubicación de memoria que corresponde a cada combinación específica de variables de entrada.

Tabla de verdad Tabla lógica que describe la respuesta de la salida de un circuito con respecto a las diversas combinaciones de los niveles lógicos en sus entradas.

Tamaño de escalón *Vea* Resolución.

Tecnología de bajo voltaje Línea de dispositivos lógicos que operan con un voltaje de suministro nominal de 3.3V o menor.

Temporizador 555 CI compatible con TTL que puede cablearse para operar en varios modos distintos, como un circuito de un monoestable o como un multivibrador astable.

Teoremas Booleanos Reglas que pueden aplicarse al álgebra Booleana para simplificar las expresiones lógicas.

Teoremas de DeMorgan (1) Teorema que establece que el complemento de una suma (operación OR) es igual al producto (operación AND) de los complementos, y (2) teorema que establece que el complemento de un producto (operación AND) es igual a la suma (operación OR) de los complementos.

Tiempo de acceso El tiempo que transcurre desde que un circuito de memoria recibe una nueva dirección de entrada hasta que los datos están disponibles y estables a la salida para una operación de lectura.

Tiempo de adquisición Tiempo requerido para que un circuito de muestreo y retención capture el valor analógico presente en su entrada.

Tiempo de estabilización Cantidad de tiempo que tarda la salida de un convertidor digital-análogo en pasar de 0 al valor final con una variación de la mitad del paso de resolución a plena escala cuando la entrada cambia de todos 0s a todos 1s.

Tiempo de establecimiento (t_S) Intervalo de tiempo que sigue justo después de la transición activa de la señal de reloj, durante el cual la entrada de control se debe mantener en el nivel apropiado.

Tiempo de retención (t_H) Intervalo de tiempo que sigue inmediatamente después de la transición activa de la señal de reloj, durante el cual la entrada de control debe mantenerse en el nivel apropiado.

Tiempos de propagación (t_{PLH}/t_{PHL}) Retraso a partir del tiempo en que se aplica una señal, hasta el momento en el que la salida realiza su cambio.

Tiempos de transición de reloj Tiempos mínimos de elevación y caída para las transiciones de la señal de reloj utilizada por un CI dado, especificado por el fabricante del CI.

Tipo El atributo de una variable en un lenguaje basado en computadora, el cual define su tamaño y cómo puede usarse.

Tipo enumerado Un tipo definido por el usuario en VHDL para una señal o variable.

Traductor de nivel de voltaje Circuito que toma un conjunto de niveles de voltaje de entrada y lo traduce en un conjunto distinto de niveles de salida.

Transductor Dispositivo que convierte una variable física en una variable eléctrica (por ejemplo, una fotocelda o un termopar).

Transferencia asíncrona Transferencia de datos que se realiza sin ayuda del reloj.

Transferencia atascada *Vea* Transferencia asíncrona.

Transferencia de datos *Vea* Transferencia de datos en paralelo o Transferencia de datos en serie.

Transferencia de datos en paralelo Operación mediante la cual varios bits de datos se transfieren de manera simultánea hacia un contador o registro.

Transferencia de datos en serie Transferencia de datos de un lugar a otro, un bit a la vez.

Transferencia síncrona Transferencia de datos que se realiza mediante el uso de las entradas síncronas y de reloj de un flip-flop.

Transición con Pendiente Negativa (NGT) Cuando un reloj pasa de 1 a 0.

Transición con Pendiente Positiva (PGT) Cuando una señal de reloj cambia de un 0 lógico a un 1 lógico.

Transistor ascendente *Vea* Transistor de suministro de corriente.

Transistor de suministro de corriente Nombre que se da al transistor de salida (Q_3) de la mayoría de los circuitos TTL. Este transistor conduce cuando el nivel lógico de salida es ALTO.

Transistor descendente *Vea* Transistor drenador de corriente.

Transistor drenador de corriente Nombre que se da al transistor de salida (Q_4) de un circuito TTL. Este transistor conduce cuando el nivel lógico de salida es BAJO.

Transitorio Cambio de voltaje momentáneo, estrecho, espurio y muy definido.

Transitorios de corriente Picos de corriente generados por el circuito de salida en configuración de poste de un circuito TTL, los cuales se producen cuando ambos transistores se encienden simultáneamente.

Transmisión en paralelo Transferencia simultánea de todos los bits de un número binario, de un lugar a otro.

Transmisión en serie Transferencia de información binaria de un lugar a otro, un bit a la vez.

Transparente De un latch *D*, operando de manera que la salida *Q* refleje de inmediato a la entrada *D*.

Triestado Tipo de estructura de control que permite tres tipos de estados de salida: ALTO, BAJO y de alta impedancia (Hi-Z).

TTL (Lógica de Transistor/Transistor) Tecnología de circuitos integrados que utiliza el transistor bipolar como elemento principal del circuito.

ULSI Integración a ultra gran escala (100,000 o más compuertas).

Unidad aritmética-lógica (ALU) Circuito digital utilizado en las computadoras para realizar diversas operaciones aritméticas y lógicas.

Unidad Central de Procesamiento (CPU) Parte de una computadora que consta de la unidad aritmética-lógica (ALU) y de la unidad de control.

Unidad de control Parte de una computadora que proporciona la decodificación de las instrucciones de un programa, junto con las señales de sincronización y control necesarias para la ejecución de dichas instrucciones.

Unidad de entrada Parte de una computadora que facilita la alimentación de información a la unidad de memoria o a la ALU de la computadora.

Unidad de memoria Parte de una computadora que almacena las instrucciones y los datos que se reciben de la unidad de entrada, así como los resultados provenientes de la unidad aritmético-lógica.

Unidad de salida Parte de una computadora que recibe datos de la unidad de memoria o ALU y los presenta al mundo exterior.

VARIABLE Palabra clave en AHDL que se utiliza para comenzar una sección del código que define los nombres y los tipos de los objetos de datos y las primitivas de biblioteca. Una palabra que se utiliza en VHDL para declarar un objeto de datos local dentro de un proceso (PROCESS).

Vector de prueba Conjuntos de entradas que se utilizan para evaluar el diseño de un PLD antes de programarlo.

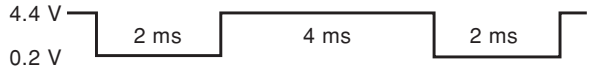
VLSI Integración a muy grande escala (de 10,000 a 99,999 compuertas).

ZIF Zócalo de circuito integrado con cero esfuerzo de inserción.

RESPUESTAS A LOS PROBLEMAS SELECCIONADOS

CAPÍTULO 1

1-1. (a) y (e) son digitales; (b), (c) y (d) son analógicas
1-3. (a) 25 (b) 9.5625 (c) 1241.6875
1-5. 000, 001, 010, 011, 100, 101, 110, 111
1-7. 1023
1-9. Nueve bits
1-11.



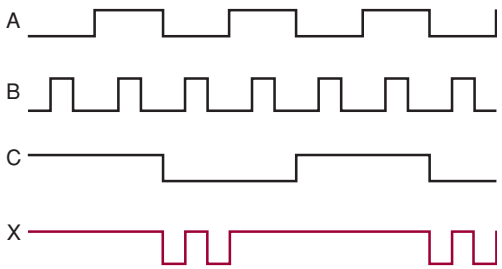
1-13. (a) $2^N - 1 = 15$ y $N = 4$; por lo tanto, se requieren cuatro líneas para la transmisión en paralelo. (b) Sólo se requiere una línea para la transmisión en serie.

CAPÍTULO 2

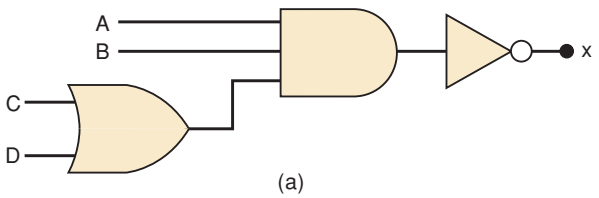
2-1. (a) 22 (c) 2313 (e) 255 (g) 983
2-2. (a) 100101 (c) 10111101 (e) 1001101 (g) 11001101 (i) 111111111
2-3. (a) 255
2-4. (a) 1859 (c) 14333 (e) 357 (g) 2047
2-5. (a) 3B (c) 397 (e) 303 (g) 10000
2-6. (a) 11101000011 (c) 11011111111101 (e) 101100101 (g) 011111111111
2-7. (a) 16 (c) 909 (e) FF (g) 3D7
2-9. $2133_{10} = 855_{16} = 100001010101_2$
2-11. (a) 146 (c) 14,333 (e) 15 (g) 704
2-12. (a) 4B (c) 800 (e) 1C4D (g) 6413
2-15. (a) 16 (c) 909 (e) FF (g) 3D7
2-16. (a) 10010010 (c) 0011011111111101 (e) 1111 (g) 1011000000
2-17. 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 28A, 28B, 28C, 28D, 28E, 28F, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 29A, 29B, 29C, 29D, 29E, 29F, 2A0
2-19. (a) 01000111 (c) 000110000111 (e) 00010011 (g) 10001001011000100111
2-21. (a) 9752 (c) 695 (e) 492
2-22. (a) 64 (b) FFFFFFFF (c) 999,999
2-25. 78, A0, BD, A0, 33, AA, F9
2-26. (a) BEN SMITH
2-27. (a) 101110100 (bit de paridad a la izquierda) (c) 11000100010000100 (e) 0000101100101
2-28. (a) No hay error de un solo bit (b) Error de un solo bit (c) Error doble (d) No hay error de un solo bit
2-30. (a) 10110001001 (b) 11111111 (c) 209 (d) 59,943 (e) 9C1 (f) 010100010001 (g) 565 (h) 10DC (i) 1961 (j) 15,900 (k) 640 (l) 952B (m) 100001100101 (n) 947 (o) 10001100101 (p) 101100110100 (q) 1001010 (r) 01011000 (BCD)
2-31. (a) 100101 (b) 00110111 (c) 25 (d) 0110011 0110111 (e) 45
2-32. (a) Hexadecimal (b) 2 (c) Dígito (d) De Gray (e) Paridad; errores de un solo bit (f) ASCII (g) Hexadecimal (h) byte
2-33. (a) 1000
2-34. (a) 0110
2-35. (a) 777A (c) 1000 (e) A00
2-36. (a) 7778 (c) OFFE (e) 9FE
2-37. (a) 1,048,576 (b) Cinco (c) 000FF
2-39. Ocho

CAPÍTULO 3

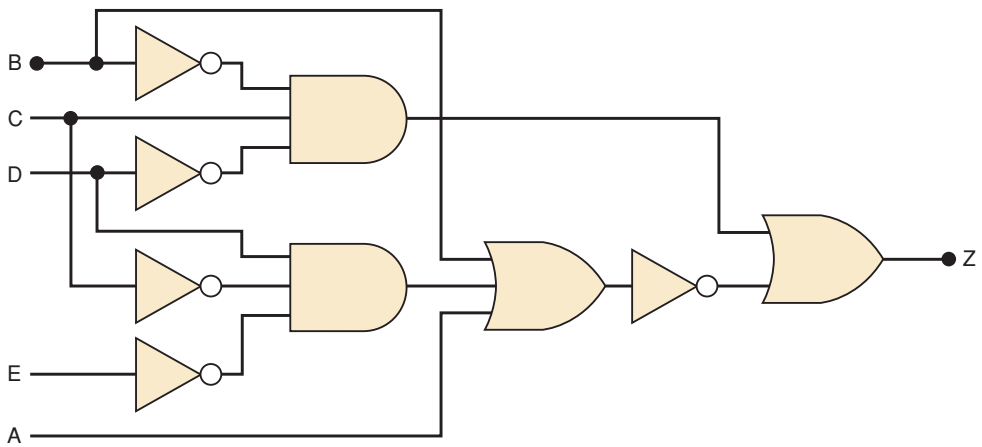
3-1.



3-3. x estará en nivel ALTO constante.
3-6. (a) x está en ALTO sólo cuando A, B y C están en ALTO.
3-7. Se cambia la compuerta OR por una compuerta AND.
3-8. SAL siempre está en BAJO.
3-12. (a) $x = (\overline{\overline{A} + \overline{B}})BC$. x está en ALTO sólo cuando ABC = 111
3-13. X está en ALTO para todos los casos en los que $E = 1$, excepto para $EDCBA$ = 10101, 10110 y 10111.
3-14. (a) $x = \mathrm{D} \cdot (\overline{AB + C}) + E$
3-16.

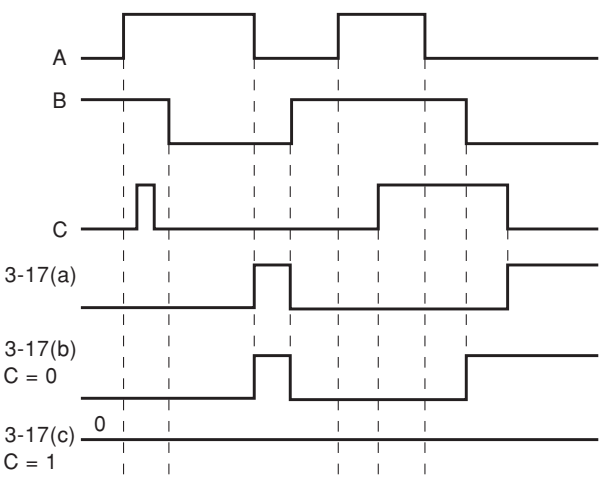


(a)

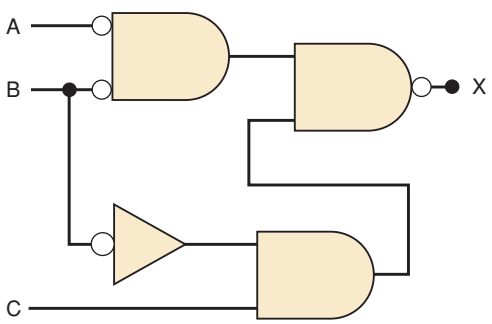


(b)

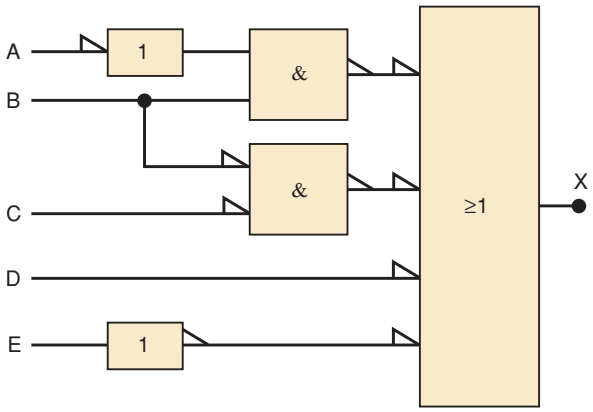
3-17–3-18.



3-19. $x = \overline{(\overline{A + B}) \cdot (\overline{B + C})}$
$x = 0$ sólo cuando $A = B = 0, C = 1$.
3-23. (a) 1 (b) A (c) 0 (d) C (e) 0 (f) D (g) D (h) 1 (i) G (j) y
3-24. (a) $MP\overline{N} + \overline{M}PN$
3-26. (a) $A + \overline{B} + C$ (c) $\overline{A} + B + CD$ (e) $A + B$ (g) $\overline{A} + B + \overline{C} + \overline{D}$
3-27. $A + B + \overline{C}$
3-32. (a) $W = 1$ cuando $T = 1$ y $P = 1$ o $R = 0$.
3-33. (a) NOR (b) AND (c) NAND
3-35. (a)



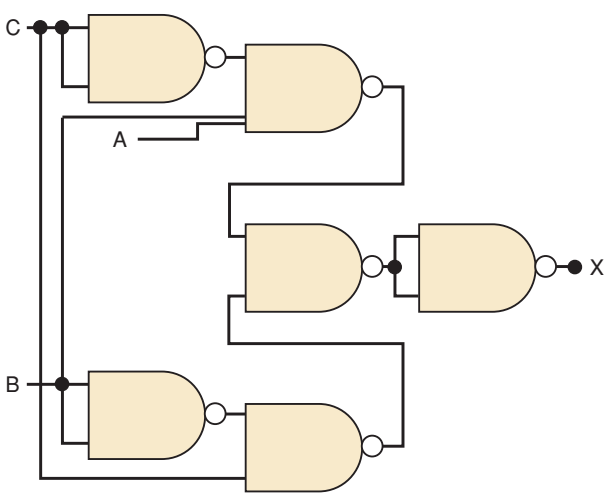
3-38. X cambiará a ALTO cuando $E = 1$, o $D = 0$, o $B = C = 0$, o cuando $B = 1$ y $A = 0$.
3-39. (a) ALTO (b) BAJO
3-41. $\overline{LUZ} = 0$ cuando $A = B = 0$ o $A = B = 1$.
3-42 (a)



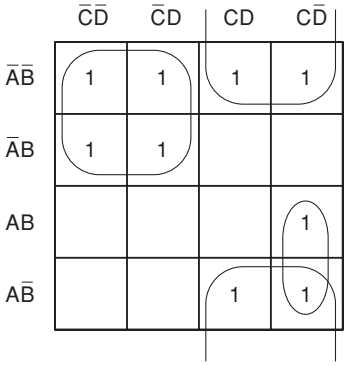
3-43. (a) Falso (b) Verdadero (c) Falso
(d) Verdadero
(e) Falso (f) Falso (g) Verdadero (h) Falso
(i) Verdadero (j) Verdadero
3-45. Las soluciones en AHDL y VHDL vienen en el CD incluido.
3-47. Se colocan INVERSORES en las entradas A_7, A_5, A_4, A_2 para el 74HC30.
3-49. Se requieren seis compuertas NAND de 2 entradas.

CAPÍTULO 4

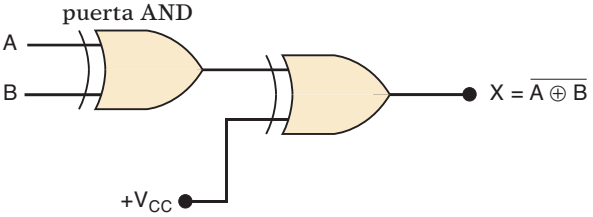
4-1. (a) $C\overline{A} + CB$ (b) $\overline{Q}R + Q\overline{R}$ (c) $C + \overline{A}$ (d) $\overline{R}\,\overline{S}\,\overline{T}$
(e) $BC + \overline{B}(\overline{C} + A)$
(f) $BC + \overline{B}(\overline{C} + A)$ o $BC + \overline{B}\,\overline{C} + AC$
(g) $\overline{D} + A\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C$
(h) $x = ABC + AB\overline{D} + \overline{A}BD + \overline{B}\,\overline{C}\,\overline{D}$
4-3. $MN + Q$
4-4. Una solución: $\overline{x} = \overline{B}C + AB\overline{C}$. Otra: $x = \overline{A}B + \overline{B}C = BC$. Otra: $BC + \overline{B}\,\overline{C} + \overline{A}\,\overline{C}$
4-7. $x = \overline{A}_3(A_2 + A_1A_0)$
4-9.



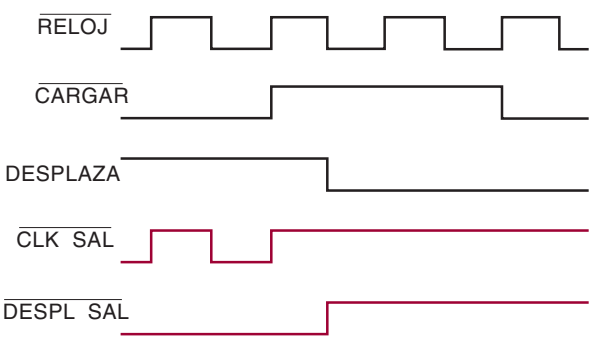
4-11. (a) $x = \overline{A}\,\overline{C} + \overline{B}C + AC\overline{D}$



4-14. (a) $x = BC + \overline{B}\,\overline{C} + AC$; o $x = BC + \overline{B}\,\overline{C} + A\overline{B}$
Un posible agrupamiento:
$x = \overline{A}BD + ABC + AB\overline{D} + \overline{B}\,\overline{C}\,\overline{D}$; otro sería:
$x = ABC + \overline{A}BD + A\overline{C}\,\overline{D} + \overline{B}\,\overline{C}\,\overline{D}$
4-15. $x = \overline{A}_3A_2 + \overline{A}_3A_1A_0$
4-16. (a) Mejor solución: $x + B\,\overline{C} + A\,D$
4-17. $x = \overline{S}_1\overline{S}_2 + \overline{S}_1\overline{S}_3 + \overline{S}_3\overline{S}_4 + \overline{S}_2\overline{S}_3 + \overline{S}_2\overline{S}_4$
4-18. $z = \overline{B}C + \overline{A}B\overline{D}$
4-21. $A = 0, B = C = 1$
4-23. Una posibilidad se muestra a continuación.
4-24. Cuatro compuertas XNOR alimentando una compuerta AND



4-26. Cuatro salidas, en donde z_3 es el MSB
$z_3 = y_1y_0x_1x_0$
$z_2 = y_1x_1(\overline{y}_0 + \overline{x}_0)$
$z_1 = y_0x_1(\overline{y}_1 + \overline{x}_0) + y_1x_0(\overline{y}_0 + \overline{x}_1)$
$z_0 = y_0x_0$
4-28. $x = AB(\overline{C \oplus D})$
4-30. N−S $= \overline{C}\,\overline{D}(A + B) + AB(\overline{C} + \overline{D})$; E−W $= \overline{N}-\overline{S}$
4-33. (a) No (b) No
4-35. $x = A + BCD$
4-38. $z = x_1x_0y_1y_0 + x_1\overline{x}_0y_1\overline{y}_0 + \overline{x}_1x_0\overline{y}_1y_0 + \overline{x}_1\overline{x}_0\overline{y}_1\overline{y}_0$
No hay pares, cuádruples ni octetos
4-40. (a) Indeterminado (b) 1.4-1.8 V (c) Vea a continuación.

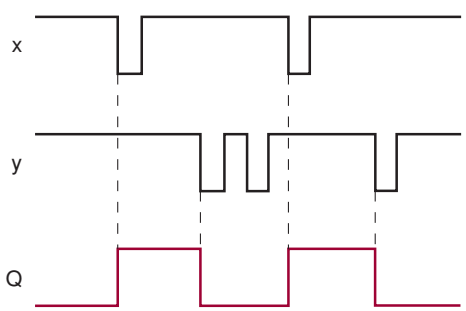


4-43. Posibles fallas: V_{CC} o tierra defectuosos en Z2; Z2-1 o Z2-2 en circuito abierto interno o externo; Z2-3 en circuito abierto interno

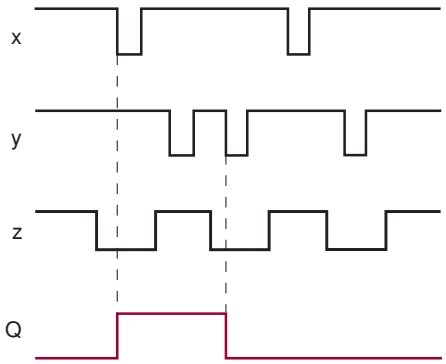
4-44. Sí: (c), (e), (f). No: (a), (b), (d), (g).
4-46. Z2-6 y Z2-11 en corto
4-48. Fallas más probables: tierra o V_{CC} defectuosos en Z2; Z1 conectado al revés; Z1 con daño interno
4-49. Posibles fallas:
Z2-13 en corto con V_{CC};
Z2-8 en corto con V_{CC};
conexión abierta a Z2-13;
Z2-3, Z2-6, Z2-9 o Z2-10 en corto a tierra
4-50. (a) V, (b) V, (c) F, (d) F, (e) V
4-54. Ecuación Booleana; tabla de verdad; diagrama esquemático
4-56. (a) AHDL: aparatos[7..0] :OUTPUT
VHDL: aparatos :OUT BIT_VECTOR (7 DOWNTO 0);
4-57. (a) AHDL: H"98" B"10011000" 152
VHDL: X"98" B"10011000" 152
4-58.
```
AHDL: bitssal[3]  =   bitsent[1];
      bitssal[2]  =   bitsent[3];
      bitssal[1]  =   bitsent[0];
      bitssal[0]  =   bitsent[2];
VHDL: bitssal(3)  <=  bitsent(1);
      bitssal(2)  <=  bitsent(3);
      bitssal(1)  <=  bitsent(0);
      bitssal(0)  <=  bitsent(2);
```
4-60.
```
BEGIN
      IF valor digital[ ] < 10 THEN
           z 5 VCC;  -- produce un 1 de salida
      ELSE z 5 GND;  -- produce un 0 de salida
      END IF;
END;
```
4.62
```
PROCESS (valor digital)
       BEGIN
              IF (valor digital < 10) THEN
                     z < 5  '1';
              ELSE
                     Z < 5  '0';
              END IF;
END PROCESS
```
65-65. S=!P#Q&R
4-68. (a) De 00 a EF

CAPÍTULO 5

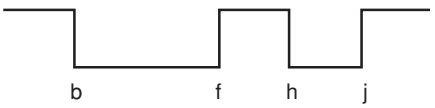
5-1.



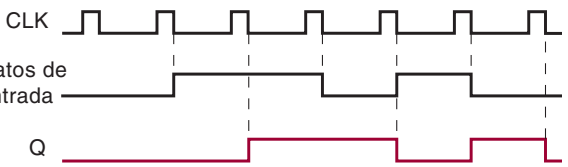
5-3.



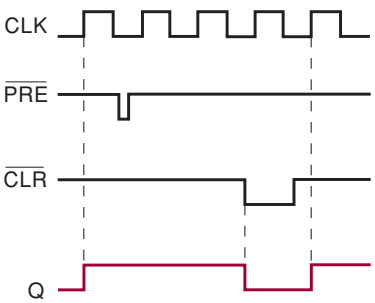
5-6. Z1-4 permanece atascada en ALTO.
5-9. Se supone que al principio $Q = 0$.
Para un FF con PGT: Q cambiará a ALTO en la primera PGT de CLK.
Para un FF con NGT: Q cambiará a ALTO en la primera NGT de CLK, a BAJO en la segunda NGT y de vuelta al nivel ALTO en la cuarta NGT.
5-11.



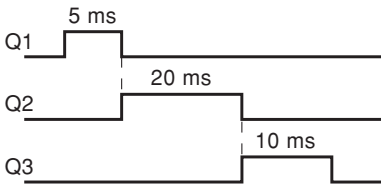
5-12. (a) Onda cuadrada de 5 kHz.
5-14.



5-16. Onda cuadrada de 500 Hz.
5-21.



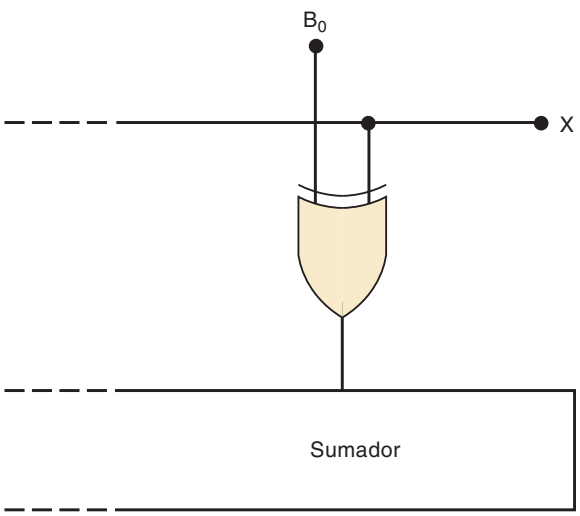
5-23. (a) 200 ns (b) 747; 74C74
5-25. Se conecta A con J y $\overline{A}$ con K.
5-27. (a) Se conecta X con J, $\overline{X}$ con K. (b) Use el arreglo de la figura 5-41.
5-29. Se conecta X_0 a la entrada D de X_2.
5-30. (a) 101;011;000
5-33. (a) 10 (b) 1953 Hz (c) 1024 (d) 12
5-36. Se colocan inversores en A_8, A_{11} y A_{14}.
5-41.



5-43. (a) A_1 o A_2 deben permanecer en BAJO cuando ocurra una PGT en B.
5-45. Una posibilidad es que $R = 1$ k y $C = 80$ nF.
5-50. (a) No (b) Sí
5-51. (a) Sí
5-53. (a) No (b) No
5-55. (a) No (b) No (c) Sí
5-56 (a) Latch NAND y NOR (b) J-K (c) latch D (d) Flip-flop D
5-59. Vea los archivos Prob5_59.tdf y prob5_59.vhd en el CD incluido.
5-61. Vea los archivos Prob5_61.tdf y prob5_61.vhd en el CD incluido.
5-66. (a) Vea el archivos Prob5_66a.tdf en el CD incluido. (b) Vea el archivos Prob5_66b.vhd en el CD incluido.

CAPÍTULO 6

6-1. (a) 10101 (b) 10010 (c) 1111.0101
6-2. (a) 00100000 (incluyendo el bit de signo) (b) 11110010 (c) 00111111 (d) 10011000 (e) 01111111 (f) 10000001 (g) 01011001 (h) 11001001
6-3. (a) +13 (b) −3 (c) +123 (d) −103 (e) +127
6-5. De -16_{10} a 15_{10}
6-6. (a) 01001001, 10110111 (b) 11110100, 00001100
6-7. De 0 a 1023; de -512 a +511
6-9. (a) 00001111 (b) 11111101 (c) 11111011 (d) 10000000 (e) 00000001
6-11. (a) 100011 (b) 1111001
6-12. (a) 11 (b) 111
6-13. (a) 10010111 (BCD) (b) 10010101 (BCD) (c) 010100100111 (BCD)
6-14. (a) 6E24 (b) 100D (c) 18AB
6-15. (a) 0EFE (b) 229 (c) 02A6
6-17. (a) 119 (b) +119
6-19. SUMA = $A \oplus B$; ACARREO = AB
6-21. $[A] = 1111$, o $[A] = 000$ (si $C_0 = 1$)
6-25. $C_3 = A_2B_2 + (A_2 + B_2)\{A_1B_1 + (A_1 + B_1)[A_0B_0 + A_0C_0 + B_0C_0]\}$
6-27. (a) SUMA = 0111
6-32



6-33.

	[F]	C_{N+4}	OVR
(a)	1001	0	1

6-35. (a) 00001100
6-37. (a) 0001 (b) 1010
6-39. (a) 1111 (b) ALTO (c) Sin cambio (d) ALTO
6-41. (a) 00000100 (b) 10111111
6-43. (a) 0 (b) 1 (c) 0010110
6-44. **AHDL**
z[6..0] = a[7..1];
z[7] = a[0];
VHDL
z(6..0) < = a(7..1);
z(7) < = a(0);
6-47. AHDL: ovr <= c[4] $ c[3)];
VHDL: ovr <= c(4) XOR c(3);
6-48. Vea los archivos Prob6_48.tdf y Prob6-48.vhd en el CD incluido.
6-53. Use flip-flops D. Conecte ($S_3 + S_2 + S_1 + S_0$) a la entrada D del FF 0; C_4 a la entrada D del FF de acarreo; y S_3 a la entrada D del FF de signo.
6-54. 0000000001001001; 1111111110101110

CAPÍTULO 7

Nota: las soluciones a ciertos problemas en el capítulo 7 se proporcionan en un archivo de documento (*Soluciones capitulo 7.doc*) en el CD incluido. Consulte este archivo según se indique a continuación.

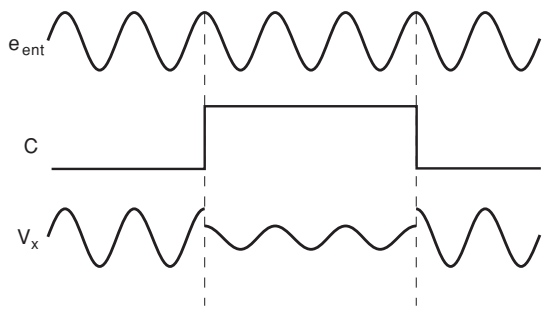
7-1. (a) 250 kHz; 50% (b) Igual que (a) (c) 1 MHz (d) 32
7-3. 10000_2
7-5. Los estados 1000 y 0000 nunca ocurrirán
7-7. (a) Vea el diagrama esquemático en el CD. (b) 33 MHz
7-9. La frecuencia en $D = 100$ Hz (vea el diagrama en el CD)
7-11. Sustituya la compuerta NAND de cuatro entradas con una compuerta NAND de tres entradas que controle todas las señales CLR de los FFs cuyas entradas sean Q5, Q4 y Q1
7-13. Vea el diagrama en el CD.
7-15. El contador cambia de estado entre 000 y 111 en cada pulso de reloj
7-17. Vea el diagrama de sincronización en el CD.
7-19. Vea el diagrama de sincronización en el CD.
7-21. (a) 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011 y se repite (b) MOD-12 (c) La frecuencia en QD (MSB) es 1/12 de la frecuencia de CLKD (d) 33.3%
7-23. (a) Vea el diagrama de sincronización en el CD (b) MOD-10 (c) De 10 hasta 1 (d) Se puede producir MOD-10, pero no la misma secuencia
7-25. (a), (b) Vea los diagramas en el CD.
7-27. Vea los diagramas en el CD.
7-29.

Salida:	QA	QB	QC	QD	RCO
Frecuencia:	3 MHz	1.5 MHz	750 kHz	375 kHz	375 kHz
Ciclo de trabajo:	50%	50%	50%	50%	6.25%

7-31. Frecuencia en f_{sal1} = 500 kHz, en f_{sal2} = 100 kHz
7-33. 12M/8 = 1.5M 1.5M/10 = 150k 1.5M/15 = 100k Vea el diagrama en el CD
7-35. Vea los símbolos de las compuertas en el CD.
7-37. Vea la simulación en el CD.
7-39. Vea la simulación en el CD.
7-41. Vea el diagrama en el CD.
7-43. (a) $J_A = B\,\overline{C}$, $K_A = 1$, $J_B = C\,A + \overline{C}\,\overline{A}$, $K_B = 1$, $J_C = B\,\overline{A}$, $K_C = B + \overline{A}$
(b) $J_A = B\,\overline{C}$, $K_A = 1$, $J_B = K_B = 1$, $J_C = K_C = B$
7-45. $J_A = K_A = 1$, $J_B = C\,\overline{A} + D\,\overline{A}$, $K_B = \overline{A}$, $J_C = D\,\overline{A}$, $K_C = B\,A$, $J_D = C\,B\,A$, $K_D = A$
7-47. $D_A = \overline{A}$, $D_B = B\,A + \overline{B}\,\overline{A}$, $D_C = C\,A + C\,B + \overline{C}\,\overline{B}\,\overline{A}$
7-49. Vea los archivos de HDL en el CD. `mod13 ahdl mod13 vhdl`
7-51. Vea los archivos de HDL en el CD. `gray ahdl gray vhdl`
7-53. Vea los archivos de HDL en el CD. `divide por50 ahdl divide por50 vhdl`
7-55. Vea los archivos de HDL en el CD. `mod256 ahdl mod256 vhdl`
7-57. Vea los archivos de HDL en el CD. `mod16 ahdl mod16 vhdl`
7-59. Vea el diagrama en el CD.
7-61. Vea los archivos de HDL en el CD. `mod10 ahdl mod5 ahdl mod50 vhdl mod10 vhdl mod5 vhdl`
7-63. Vea los archivos de HDL en el CD. `wash mach delux wash mach delux`
7-65. Vea la tabla en el CD.
7-67. Se necesitan ocho pulsos de reloj para cargar en serie un 74166, ya que hay ocho FFs en el chip.
7-69. Vea el diagrama de sincronización en el CD.
7-71. Vea la respuesta en el CD.
7-73. Vea el diagrama en el CD.
7-75. Vea el diagrama en el CD.
7-77. La salida del AND de 3 entradas o las entradas J, K del FF D en corto a tierra, la salida del FF D en corto a tierra, la entrada CLK en el FF D abierta, la entrada B de la NAND abierta
7-79. Vea los archivos de HDL en el CD. `siso8 ahdl siso8 vhdl`
7-81. Vea los archivos de HDL en el CD. `piso8 ahdl piso8 vhdl`
7-83. Vea la simulación en el CD.
7-85. Vea los archivos de HDL en el CD. `johnson ahdl johnson vhdl`
7-87. Vea la simulación en el CD.
7-89. (a) En paralelo (b) Binario (c) MOD-8 descendente (d) MOD-10, BCD, de décadas (e) Asíncrono, de rizo (f) De anillo (g) Johnson (h) Todos (i) Prestablecible (j) Ascendente/descendente (k) Asíncrono, de rizo (l) MOD-10, BCD, de décadas (m) Síncrono, en paralelo

CAPÍTULO 8

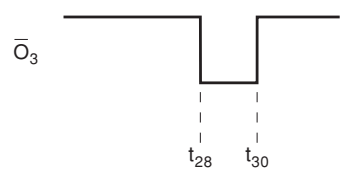
8-1. (a) A; B (b) A (c) A
8-2. (a) 39.4 mW, 18.5 ns (b) 65.6 mW, 7.0 ns
8-3. (a) 0.9 V
8-4. (a) I_H (b) I_{CCL} (c) t_{PHL} (d) V_{NH} (e) Montaje superficial (f) Drenado de corriente (g) Factor de carga (h) En forma de tótem (i) Transistor de drenado (j) De 4.75 a 5.25 V (k) 2.5 V; 2.0 V (l) 0.8 V; 0.5 V (m) Suministro
8-5. (a) 0.7 V; 0.3 V (b) 0.5 V; 0.4 V (c) 0.5 V; 0.3 V
8-6. (b) AND, NAND (c) Entradas sin conectar
8-7. (a) 40 (b) 33
8-8. (a) 20 μA/0.4 mA
8-9. (a) 30/15 (b) 24 mA
8-11. El factor de carga no se excede en ningún caso.
8-13. 60 ns; 38 ns
8-14. (a) 2 kΩ
8-15. (b) La resistencia de 4-7 kΩ es demasiado grande.
8-19. *a*, *c*, *e*, *f*, *g*, *h*
8-21. 12.6 mW
8-27. $AB + CD + FG$
8-29. (a) 5 V (b) R_S = 110 Ω para una corriente de LED de 20 mA
8-30. (a) 12 V (b) 40 mA
8-33. Contador de anillo
8-36. 1.22 V; 0 V
8-37.



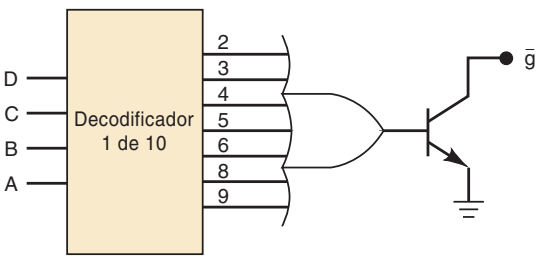
8-38. −1 y −2
8-39. (a) 74HCT (b) Convierte los voltajes lógicos (c) CMOS no puede drenar corriente TTL. (d) Falso
8-41. (a) Ninguna
8-44. El factor de carga del 74HC00 se excede; desconecte la terminal 3 del 7402 y conéctela a tierra.
8-46. R_2 = 1.5 kΩ, R_1 = 18 kΩ
8-49. (b) es una posible falla.
8-50. De 0 V a −11.25 V y de vuelta a −6 V.

CAPÍTULO 9

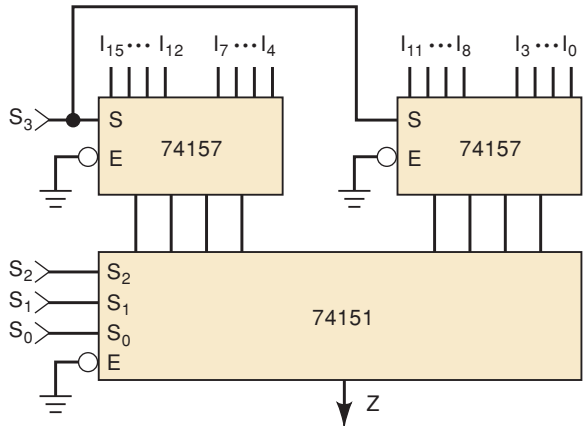
9-1. (a) Todas en ALTO (b) O_0 = BAJO
9-2. Seis entradas, 64 salidas
9-3. (a) $E_3E_2E_1$ = 100; $[A]$ = 110 (b) $E_3E_2E_1$ = 100; $[A]$ = 011
9-5.



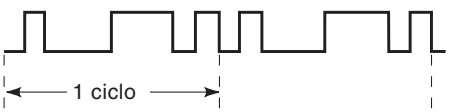
9-7. Se habilita cuando D = 0
9-10. Las resistencias son de 250 Ω.
9-12.



9-13. (a), (b) Codificador (c), (d), (e) Decodificador
9-17. La cuarta pulsación de tecla se introduciría en el registro del MSD.
9-18. La opción (b)
9-20. (a) Sí (b) No (c) No
9-21. La línea A_2 del bus está abierta entre Z2 y Z3.
9-23. El segmento *g* o el transistor de salida del decodificador se quemarían.
9-25. Salidas del decodificador: *a* y *b* están en corto.
9-26. La conexión "f" del decodificador/controlador a la compuerta XOR está abierta.
9-29. Un MUX de 4 a 1
9-31.



9-32. (b) El número total de conexiones en el circuito que utiliza multiplexores es de 63, sin incluir V_{CC} y GND, y sin incluir las conexiones a las entradas del reloj del contador. El número total para el circuito, utilizando decodificador/controladores separados es de 66.
9-33.



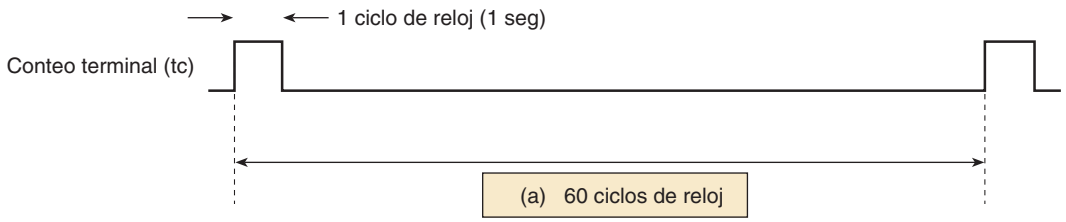
9-35.

A	B	C	
0	0	0	$0 \Rightarrow l_0$
0	0	1	$0 \Rightarrow l_1$
0	1	0	$0 \Rightarrow l_2$
0	1	1	$1 \Rightarrow l_3$
1	0	0	$0 \Rightarrow l_4$
1	0	1	$1 \Rightarrow l_5$
1	1	0	$1 \Rightarrow l_6$
1	1	1	$1 \Rightarrow l_7$

9-37. Z = ALTO para *DCBA* = 0010, 0100, 1001, 1010.
9-39. (a) Codificador, MUX (b) MUX, DEMUX (c) MUX (d) Codificador (e) Decodificador, DEMUS (f) DEMUX (g) MUX
9-41. Cada salida del DEMUX cambia a BAJO, una a la vez y en secuencia.
9-43. Cinco líneas
9-46. (a) La secuencia se detiene después de que se activa el actuador 3.
9-47. La probable falla es un corto a tierra, en el MSB del MUX de las décadas.
9-48. Es probable que Q_0 y Q_1 estén invertidas.
9-49. Es probable que las entradas 6 y 7 del MUX estén en corto.
9-50. S_1 se atascó en BAJO.
9-53. Use tres chips 74HC85.
9-55. Es probable que A_0 y B_0 estén invertidas.
9-57. $\overline{OE}_C = 0$, $\overline{IE}_C = 1$; $\overline{OE}_B = \overline{OE}_A = 1$; $\overline{IE}_B = \overline{IE}_A = 0$; aplicar un pulso de reloj.
9-61. (a) En t_3 cada registro contiene 1001.
9-63. (a) 57FA (b) De 5000 a 57FF (c) De 9000 a 97FF (d) no
9-65. Vea los archivos Prob9_65.tdf y Prob9-65.vhd en el CD incluido.

CAPÍTULO 10

10-1. (d) 20 Hz (e) Sólo un LED se encenderá en un momento dado.
10-2. 24
10-3. Cuatro estados = cuatro intervalos * 15°/intervalo = 60 ° de rotación.
10-5. Tres transiciones de estado * 15°/intervalo = 45° de rotación.
10-10. 1111
10-12. (a) 1011
10-13. No
10-15. Los datos desaparecen (Hi-Z) antes de que DAV cambie a BAJO. El estado Hi-Z está fijo.
10-16.



10-17. 60 ciclos/seg * 60 seg/min * 60 min/hr * 24 hrs/día = 5,184,000 ciclos/día. Se requiere una gran cantidad de tiempo para generar un archivo de simulación.
10-18. Cuando la entrada set está activa, se desvía el preescalador y se alimenta la señal de reloj de 60 Hz en forma directa al contador de unidades de segundos.
10-22. Vea los archivos Prob10_22.tdf y Prob10_22.vhd en el CD incluido.

CAPÍTULO 11

11-1. (f), (g) Falso
11-3. LSB = 20 mV
11-5. Aproximadamente 5 mV
11-7. 14.3 por ciento, 0.286 V
11-9. 250.06 rpm
11-11. Los ocho MSBs: PUERTO[7..0] DAC[9..2]
11-13. 800 Ω; no
11-15. Utiliza menos valores distintos para *R*
11-17. (a) Siete
11-19. El valor de 242.5 mV no está dentro de las especificaciones.
11-21. El bit 1 del DAC está abierto o atascado en ALTO.
11-22. Los bits 0 y 1 están invertidos.
11-24. (a) 10010111
11-27. (a) 1.2 mV (b) 2.7 mV
11-28. (a) 0111110110
11-31. La frecuencia de la forma de onda reconstruida es de 3.33 kHz.
11-32. (a) 5 kHz (b) 9.9 kHz
11-33. Rampa digital: *a*, *d*, *e*, *f*, *h*. SAC: *b*, *c*, *d*, *e*, *g*, *h*
11-36. 80 μs
11-38. 2.276 V
11.40. (a) 00000000 (b) 500 mV (c) 510 mV (d) 255 mV (e) 01101110 (f) 0.2°F; 2 mV
11-45. El interruptor se quedó atascado en la posición cerrada; el interruptor se quedó atascado en la posición abierta o el capacitor está en corto.
11-47. (a) La dirección es EA*xx*.
11-52. Falso: *a*, *e*, *g*; Verdadero: *b*, *c*, *d*, *f*, *h*

CAPÍTULO 12

12-1. 16,384; 32; 524,288
12-3. 64k X 4
12-7. (a) Hi-Z (b) 11101101
12-9. (a) 16,3848 (b) Cuatro (c) Dos decodificadores 1 de 128
12-11. 120 ns
12-15. Los siguientes transistores tendrán conexiones de colector abierto: Q_0, Q_2, Q_5, Q_6, Q_7, Q_9, Q_{15}.
12-17. (a) Borra todas las ubicaciones de memoria para retener el valor FF_{16}
(b) Escribe el valor $3C_{16}$ en la dirección 2300_{16}
12-19. Datos hexadecimales: 5E, BA, 05, 2F, 99, FB, 00, ED, 3C, FF, B8, C7, 27, EA, 52, 5B
12-20. (a) 25.6 kHz (b) Ajustar V_{ref}.
12-22. (a) [*B*] = 40 (hex); [*C*] = 80 (hex) (b) [*B*] = 55 (hex); [*C*] = AA (hex) (c) 15,360 Hz (d) 28.6 MHz (e) 27.9 kHz
12-24. (a) 100 ns (b) 30 ns (c) 10 millones (d) 20 ns (e) 30 ns (f) 40 ns (g) 10 millones
12-30. Cada 7.8 μs.
12-31. (a) 4096 columnas, 1024 filas (b) 2048 (c) Se duplicaría.
12-34. Agregue cuatro PROMs más (de la PROM-4 a la PROM-7) al circuito. Conecte sus salidas de datos y entradas de datos a los buses de datos y de direcciones, en forma respectiva. Conecte AB_{13} a la entrada *C* del decodificador y conecte las salidas de la 4 a la 7 del decodificador a las entradas *CS* de las PROMs de la 4 a la 7, en forma respectiva.
12-38. F000–F3FF; F400–F7FF; F800–FBFF; FC00–FFFF
12-40. La entrada *B* del decodificador está abierta o se atascó en ALTO.
12-42. Sólo los módulos 1 a 3 de RAM se están probando.
12-43. El chip de RAM con las salidas de datos de la 4 a la 7 en el módulo 2 no está funcionando en forma apropiada.
12-44. La salida 7 del módulo 3 de la RAM está abierta o se atascó en ALTO.
12-46. Suma de comprobación = 11101010.

CAPÍTULO 13

13-2. La velocidad necesaria de operación para el circuito, el costo de fabricación, el consumo de energía del sistema, el tamaño del sistema, la cantidad de tiempo disponible para diseñar el producto, etc.
13-4. La velocidad de operación
13-6. Ventajas: la velocidad máxima y el área de pastilla más pequeña; desventajas: tiempo de diseño/desarrollo y gastos
13-8. Los PLDs basados en SRAM deben configurarse (programarse) al encendido.
13-10. En un programador de PLDs o en el sistema (mediante la interfase JTAG)
13-12. terminal 1—GCLRn (Borrar global)
terminal 2—OE2/GCLK2 (Habilitación de salida 2/Reloj global 2)
terminal 83—GCLK1 (Reloj global 1)
13-14. La celda lógica en la familia MAX7000S es el circuito AND/OR, en comparación con la tabla de búsqueda en FLEX10K; EEPROM (MAX7000S) y SRAM (FLEX10K); MAX7000S es no volátil; FLEX10K tiene mayores recursos lógicos.



ÍNDICE DE CIs

7400 Compuertas NAND cuádruples de 2 entradas, 569
7402 Compuertas NOR cuádruples de 2 entradas, 569
7404 Inversores hexadecimales, 156
7406 Búfer/controlador de inversores hexadecimales (con salida de alto voltaje de colector abierto), 536-537, 552, 565, 670
7407 Búfer/controlador hexadecimal (con salida de alto voltaje de colector abierto), 537
7413 Compuertas NAND duales de 4 entradas con entradas tipo disparador de Schmitt, 256
7414 Inversores tipo disparador de Schmitt hexadecimales, 256, 261
7442 Decodificador de BCD a decimal, 581-582, 659
7445 Decodificador/controlador de BCD a decimal, 582-583
7446 Decodificador/controlador de BCD a siete segmentos, 586
7447 Decodificador/controlador de BCD a siete segmentos, 586, 605, 642
7474 Flip-flop D dual de disparo por flanco, 240-241
7483 Sumador completo de 4 bits, 326, 653
7486 Compuertas EX-OR cuádruples de 2 entradas, 569
7485 Comparador de magnitud de 4 bits, 621, 652
74121 Monoestable no redisparable sencillo, 259-260, 569
74122 Monoestable redisparable sencillo, 259
74123 Monoestable redisparable dual, 259
74138 Decodificador/demultiplexor 1 de 8, 638-639
74147 Codificador con prioridad de decimal a BCD, 592-593, 645
74148 Codificador con prioridad de octal a binario, 592
74160 Contador BCD, 694-695
74173 Registros tipo D de 4 bits con salidas de 3 estados, 635
74185 Convertidor de código de binario a BCD de 6 bits, 813
74191 Contador ascendente/descendente síncrono de 4 bits, 451
74194 Registro de desplazamiento universal bidireccional de 4 bits, 444
74221 Monoestable no redisparable dual, 259
74373 Latch octal con salidas de 3 estados, 444
74374 Flip-flop D octal con salidas de 3 estados, 444
74382 ALU, 337-338, 343
74AC02 Compuertas NOR cuádruples de 2 entradas, 157
74AC11004 Inversores hexadecimales, 524
74ACT02 Compuertas NOR cuádruples de 2 entradas, 157
74ACT11293 Contador binario de 4 bits, 524
74AHC74 Flip-flop D dual de disparo por flanco, 537
74AHC126 Búferes triestado cuádruples no inversores, 539-540
74ALS00 Compuertas NAND cuádruples de 2 entradas, 450, 504, 510-511, 513
74ALS04 Inversores hexadecimales, 156
74ALS14 Inversores hexadecimales tipo disparador de Schmitt, 447
74ALS138 Decodificador/demultiplexor 1 de 8, 579-581, 610-611, 6714, 840, 842, 848, 850
74ALS151 Multiplexor de 8 entradas, 601-602
74ALS157 Selectores multiplexores de datos cuádruples de 2 a 1 líneas, 603-605
74ALS160 Contador de décadas síncrono, 369, 380-384, 396

74ALS161 Contador MOD-16 asíncrono, 369, 380-384, 396, 450, 472, 474
74ALS162 Contador de décadas síncrono, 369, 380-384, 396
74ALS163 Contador MOD-16 síncrono, 369, 380-384, 387-388, 396, 472, 583
74ALS164 Registro de desplazamiento de entrada en serie/salida en paralelo de 8 bits, 443-445
74ALS165 Registro de desplazamiento de entrada en paralelo/salida en serie de 8 bits, 441-443
74ALS166 Registro de desplazamiento de entrada en serie/salida en serie de 8 bits, 439-441, 482
74ALS173 Registros tipo D de 4 bits con salidas de 3 estados, 629-631
74ALS174 Registro de entrada en paralelo/salida en paralelo de 6 bits, 437-439, 479
74ALS190 Contador ascendente/descendente síncrono de 4 bits, 380, 384-389, 396
74ALS191 Contador ascendente/descendente síncrono de 4 bits, 380, 384-389, 396
74ALS192 Contador ascendente/descendente síncrono de 4 bits, 380, 396
74ALS193 Contador ascendente/descendente síncrono de 4 bits, 380, 396
74ALS194 Registro de desplazamiento universal bidireccional de 4 bits, 444
74ALS273 Registro de 8 bits, 856
74ALS299 Registro de 8 bits con líneas de E/S comunes, 637
74ALS373 Latch octal con salidas de 3 estados, 444
74ALS374 Flip-flop D octal con salidas de 3 estados, 444
74AS04 Inversores hexadecimales, 156, 513
74AS20 Compuertas NAND positivas duales de 4 entradas, 511-512
74AS74 Flip-flop D dual con disparo por flanco, 513
74AUC08 Compuerta AND cuádruple de 2 entradas, 554
74AVC08 Compuerta AND cuádruple de 2 entradas, 553
74AVC1T45 Traductor de nivel de suministro dual, 553
74C02 Compuertas NOR cuádruples de 2 entradas, 157
74C74 Flip-flop D dual con disparo por flanco, 240
74C86 Compuertas EX-OR cuádruples de 2 entradas, 145
74C266 EX-NOR cuádruple, 146
74F04 Inversores hexadecimales, 508
74HC00 Compuertas NAND cuádruples de 2 entradas, 132, 159, 552, 567-570
74HC02 Compuerta NOR cuádruple de 2 entradas, 157, 167
74HC04 Inversores hexadecimales, 159, 524, 548
74HC05 Inversores hexadecimales con colector abierto, 535, 537, 570
74HC08 Compuerta AND cuádruple de 2 entradas, 167, 553-554
74HC13 Compuertas NAND duales de 4 entradas con entradas tipo disparador de Schmitt, 256
74HC14 Inversores hexadecimales tipo disparador de Schmitt, 256, 261
74HC42 Decodificador de BCD a decimal, 581
74HC83 Sumador completo de 4 bits, 626
74HC85 Comparador de magnitud de 4 bits, 621-624
74HC86 Compuertas EX-OR cuádruples de 2 entradas, 145, 588
74HC112 Flip-flop J-K dual con disparo por flanco, 237, 240-241
74HC123 Monoestable redisparable dual, 259
74HC125 Búferes triestado cuádruples no inversores, 552, 670
74HC126 Búferes triestado cuádruples no inversores, 628
74HC138 Decodificador/demultiplexor 1 de 8, 608, 612, 666
74HC139 Decodificador 1 de 4 dual con habilitación activa en BAJO, 670
74HC147 Codificador de decimal a BCD con prioridad, 592
74HC148 Codificador de octal a binario con prioridad, 592
74HC151 Multiplexor de 8 entradas, 601-603, 608, 612, 663-664
74HC157 Selectores/multiplexores de datos cuádruples de 2 a 1 línea, 603-609
74HC160 Contador de décadas síncrono, 369, 380-384
74HC161 Contador MOD-16 síncrono, 369, 380-384
74HC162 Contador de décadas síncrono, 369, 380-384
74HC163 Contador MOD-16 síncrono, 369, 380-384
74HC164 Registro de desplazamiento de entrada en serie/salida en paralelo de 8 bits, 443-445
74HC165 Registro de desplazamiento de entrada en paralelo/salida en serie de 8 bits, 441-443
74HC166 Registro de desplazamiento de entrada en serie/salida en serie de 8 bits, 439-441
74HC173 Registros tipo D de 4 bits con salidas de 3 estados, 629-632
74HC174 Registro de entrada en paralelo/salida en paralelo de 6 bits, 437-439, 670
74HC175 Flip-flop D cuádruple, 237-238
74HC181 ALU (Unidad aritmética-lógica), 335
74HC190 Contador ascendente/descendente síncrono de 4 bits, 377, 380, 387-389, 472
74HC191 Contador ascendente/descendente síncrono de 4 bits, 377, 380, 387-389, 472
74HC192 Contador ascendente/descendente síncrono de 4 bits, 380
74HC193 Contador ascendente/descendente síncrono de 4 bits, 380
74HC194 Registro de desplazamiento universal bidireccional de 4 bits, 444
74HC221 Monoestable dual no redisparable, 259
74HC266 EX-NOR cuádruple, 146
74HC283 Sumador completo de 4 bits, 326-327
74HC373 Latch octal con salidas de 3 estados, 444
74HC374 Flip-flop D octal con salidas de 3 estados, 444
74HC382 ALU (Unidad aritmética-lógica), 332-334
74HC541 Controlador de bus octal, 635-637
74HC881 ALU (Unidad aritmética-lógica), 335

74HC4016 Interruptor bilateral cuádruple, 547-548, 567, 572
74HC4017 Contador Johnson, 449
74HC4022 Contador Johnson, 449
74HC4316 Interruptor bilateral cuádruple, 548
74HC4511 Decodificador/controlador de BCD a 7 segmentos, 588-589
74HC4543 Decodificador/controlador de pantalla numérica de LCD, 589
74HCT02 Compuertas NOR cuádruples de 2 entradas, 157
74HCT04 Inversores hexadecimales, 524
74HCT74 Flip-flop D dual con disparo por flanco, 537
74HCT293 Contador binario de 4 bits, 524
74HCT373 Latch octal con salidas de 3 estados, 444
74HCT374 Flip-flop D octal con salidas de 3 estados, 444
74LS00 Compuertas NAND cuádruples de 2 entradas, 85-86, 164, 168, 202, 265, 291
74LS01 Compuertas NAND cuádruples de 2 entradas de colector abierto, 537, 565
74LS04 Inversores hexadecimales, 156, 202, 524
74LS05 Inversores hexadecimales con colector abierto, 535
74LS08 Compuerta AND cuádruple de 2 entradas, 85-86
74LS13 Compuertas NAND duales de 4 entradas con entradas tipo disparador de Schmitt, 256
74LS14 Inversores hexadecimales tipo disparador de Schmitt, 256, 261, 563, 598
74LS20 Compuertas NAND positivas duales de 4 entradas, 562
74LS32 Compuerta OR cuádruple de 2 entradas, 85
74LS37 Compuertas NAND cuádruples de 2 entradas (búfer), 561
74LS42 Decodificador de BCD a decimal, 581
74LS74 Flip-flop D dual con disparo por flanco, 265
74LS83A Sumador completo de 4 bits, 326
74LS85 Comparador de magnitud de 4 bits, 621
74LS86 Compuertas EX-OR cuádruples de 2 entradas, 145, 168, 562
74LS112 Flip-flop J-K dual con disparo por flanco, 236-237, 240-241, 536, 561, 565, 568
74LS114 Flip-flop J-K dual con disparo por flanco, 291
74LS122 Monoestable redisparable sencillo, 259
74LS123 Monoestable redisparable dual, 259
74LS125 Búferes triestado no inversores cuádruples, 539, 541, 566
74LS126 Búferes triestado no inversores cuádruples, 539, 541
74LS138 Decodificador/demultiplexor 1 de 8, 658, 779
74LS147 Codificador de decimal a BCD con prioridad, 592-594
74LS148 Codificador de octal a binario con prioridad, 592, 596
74LS181 ALU (Unidad aritmética-lógica), 335
74LS193 Contador ascendente/descendente síncrono de 4 bits, 562
74LS221 Monoestable redisparable dual, 259
74LS244 Búferes octales y controladores de línea con salidas triestado, 887
74LS266 EX-NOR cuádruple, 146
74LS283 Sumador completo de 4 bits, 326, 329, 331, 336
74LS374 Registro de FF tipo D octal, 540
74LS382 ALU (Unidad aritmética-lógica), 332-334
74LS881 ALU (Unidad aritmética-lógica), 335
74LVC07 Búfer/controlador hexadecimal (con colector abierto), 553-554
74S00 Compuertas NAND cuádruples de 2 entradas, 506-507
74S04 Inversores hexadecimales, 156
74S112 Flip-flop J-K dual con disparo por flanco, 569

OTROS CIs

555 Temporizador, 261-262
2125A SRAM de 1K × 1, 837-838
2147H RAM NMOS de 4K × 1, 817, 821
27C64 ROM MOS de 8K × 8, 798, 804-805
2732 EPROM de 8K × 1, 857
2764 ROM MOS de 8K × 8, 822
21256 DRAM de 256K × 1, 862
27256 ROM de 32K × 8, 822
27C256 ROM de 32K × 8, 822
27C512 EPROM de 64K × 8, 805
28F256A Memoria Flash CMOS de 32K × 8, 809-811, 858
2816 EEPROM de 2K × 8, 805
2864 EEPROM de 8K × 8, 805-806, 810, 822
4001B Compuertas NOR cuádruples de 2 entradas, 157, 552, 569
4016 Interruptor bilateral cuádruple, 547-548
4049B Inversores hexadecimales, 568
4316 Interruptor bilateral cuádruple, 548
6264 SRAM de 8K × 8, 822
AD1154 Circuito integrado de muestreo y retención, 762
AD7524 DAC de 8 bits, 736
AD9020 ADC Flash de 10 bits, 757
ADC0804 ADC de aproximaciones sucesivas, 751-754
ADC0808 ADC de aproximaciones sucesivas, 763
EMP7128S CPLD de ALTERA, 885-889
EMP7128SLC84 PLD de ALTERA, 155, 887-888
FLEX10K Familia de CPLDs de ALTERA, 890-894
GAL16V8A Lógica de matriz genérica, 881-883
GAL22V10 Lógica de matriz genérica, 885
LM34 Dispositivo de medición de temperatura, 555
LM339 Comparador de voltaje analógico cuádruple, 555
MAX7000S Familia de CPLDs de ALTERA, 270, 885-889
MCM101514 SRAM CMOS de 256K × 4, 860
MCM6206C RAM CMOS de 32K × 8, 817, 821
MCM6208C RAM CMOS de 64K × 4, 818, 821
MCM6209C SRAM de 64K × 4, 844
MCM6249 SRAM CMOS de 1M × 4, 860
MCM6264C SRAM CMOS de 8K × 8, 821
MCM6708A SRAM BiCMOS de 64K × 4, 821
ML2035 Generador de ondas senoidales programable, 814
TMS27PC256 PROM CMOS de 32K × 8, 803
TMS44100 DRAM de 4M × 1, 825-826, 831-833, 843



ÍNDICE

A

Acarreo, 298, 318
  anticipado, 326
  bit de, 319
  de rizo, 325
  propagación, 325-326
Activa/activo. (*Vea también* Niveles asignados)
  en ALTO, decodificación, 390-391
  en BAJO, decodificación, 391
  niveles lógicos, 88-89
Actuador, 721
Administración de pequeños proyectos (mediante el uso de HDL), 678-679
  definición, 678
  integración y prueba del sistema, 679
  planeación estratégica, 678
  síntesis y prueba, 681
Agrupamiento, 135-138
  cuádruples, 136-137
  método de, 637
  octetos, 137-138
  pares, 135-136
AHDL, 98-99, 409-410
  archivo de diseño, 182
  BEGIN, 103
  CASE, 190, 417, 458-459, 639, 649
  Codificador, 646-647
  comentarios, 105-106
  comparador, 652
    de magnitud, 652
  CONSTANT, 344
  contador, 277-278, 409-410, 459-461
    BCD MOD-100, 421-423
    con todas las características, 412-414
    de anillo, 460
    de rizo (MOD-8), 277-278
    MOD-5, 406-407, 418
    MOD-6, 689-699, 707
    MOD-8, 682
    MOD-10, 421-422, 700, 707
    MOD-10, símbolos de bloque gráficos, 705
    MOD-12, 702-703
    MOD-60, 707
contadores BCD en cascada, 421-423
controlador de motor paso a paso, 684
  prueba de simulación, 686
controlador de semáforo, 430-432
convertidor, 654-655
  de código, 654-655
  de código BCD a binario, 654-655
declaraciones de arreglos de bits, 178-179
decodificador(es), 639-641
  controlador, 642-643
  secuencia de paso completo, 683
decodificar el contador MOD-5, 417-418
DEFAULTS, 639, 650
demultiplexores, 649-650
descripción Booleana mediante el uso de, 103
descripción del comportamiento de un contador en, 410
descripciones de estado en, 406-407
elementos esenciales en, 103
ELSE, 413
ELSIF, 189, 413
END, 103
flip-flops, 272-273
flip-flop JK, 273
identificadores de puertos primitivos, 272
IF/THEN/ELSE, 186, 646-647, 702
INCLUDE, 707
INPUT, 103
instrucción de asignación concurrente, 103
integración de módulos, 707-708
latch D, 271
latch NAND, 270
literales, 181
MACHINE, 426-427
máquinas de estado simples 426-427
monoestable no redisparable, 462
monoestables, simple, 462
multiplexores, 646-650
NODE, AHDL, 106, 339
nodos ocultos, 105-106
OUTPUT, 103

prototipo de función, 338
proyecto
de contador de frecuencia, 710-714 (*vea también* HDL)
de controlador de motor paso a paso, 679-686 (*vea también* HDL)
de reloj digital (HDL), 693-710 (*vea también* HDL)
proyecto de codificador de teclado numérico, 687-693 (*vea también* HDL)
simulación, 693
solución, 689-691
redisparable, monoestable con disparo por flanco, 465
registro PISO, 455-456
registro SISO, 453-454
SUBDESIGN, 103, 105-106, 178, 273, 341, 406, 413, 417-418
sumador, 342
de cuatro bits, 342
sumador/restador, 344-345
tablas de verdad, 181-182
TABLE, 640, 642-643, 646
VARIABLE, 105, 272, 339, 406, 689
variables intermedias en, 107
Alambrado mal instalado, 167
Alias, 747-748
Almacenamiento
auxiliar, 814
temporal, RAM, 815
Alta capacidad, dispositivos programables de (HPCLDs), 872
Alta velocidad
CMOS, 74HC/HCT, 524
comparación lógica, 546
lógica de interfase de bus, 541-542
ALTERA
archivo de descripción gráfica de una ALU de 8 bits, 338
bloques de matrices lógicas (LABs), 885
CPLD EPM7128S, 885-889
elementos lógicos (LEs), 890
familia
Cyclone, 894-895
FLEX10K, 890-894
identificadores de puertos primitivos, 272
lenguaje de descripción de hardware, 98-99
macrofunción, 337
matriz de interconexiones programable (PIA), 885
MAX+PLUS II, 99
MAX7000S, 885-889
prototipo de función, 338
uso de funciones de la biblioteca TTL con, 337-338
Alterna, 12
ALU, circuitos integrados, 317-318, 331-335, 767
expansión de la ALU, 334
operaciones
AND, 333
borrar, 332
EX-OR, 333
OR, 333
PRESET, 333
resta, 332
suma, 332
otras ALUs, 335
Análisis de contadores síncronos, 393-396
Analógico(a)
cantidad, 719
representación, 4
sistemas, 5-6
Análogo-digital (ADC)
adquisición de datos, 745-748
aproximaciones sucesivas, 749-755
CI, aproximaciones sucesivas de 8 bits (ADC0804), 751-755
Clk Ent, 753
Clk Sal, 753
entradas diferenciales, 751
Interrupción (INTR), 753
READ (RD), 753
Selección de chip (CS), 752
una aplicación, 754
$V_{ref}/2$, 753
WRITE (WR), 753
circuito de muestreo y retención, 761-762
control de amplitud digital, 737
conversión, 737, 739-740
convertidor (ADC), 7, 720
de voltaje a frecuencia, 758
error de cuantización, 743
Flash, 755-757
modulación sigma/delta, 758-761
multiplexaje, 762, 764
otros métodos de conversión, 757-761
pendiente dual, 757-758
precisión, 742-744
rampa digital, 740-745
ascendente/descendente, 757
rastreo, 757
resolución, 742-744
tiempo de conversión, 744, 750-751
Anchura de pulso activo asíncrono, 239
AND
alambrado, conexión de, 535-536
compuerta, 62-65 (*vea también* Circuitos lógicos combinacionales)
decodificación de contadores, 389-393
definición, 63
descripción Booleana, 62
implementación a partir de expresiones Booleanas, 71-73
qué representación utilizar, 89-95
representación de compuerta lógica alternativa, 86-89
resumen de operación, 63
símbolo, 63
teoremas Booleanos, 76-80
operación, 57, 62-65
resumen, 63
Anillo
contador de, 445-447, 594-596
comenzar un, 447
diagrama de estado de, 446, 689
en el circuito, 595
torcido, contadores, 447
Anticipado, acarreo, 326
Aproximaciones sucesivas, ADC de, 749-755
ARCHITECTURE, 104, 179
Arranque
memoria, 812
programa, 812
Arriba-abajo, diseño jerárquico (reloj digital mediante el uso de HDL), 696-0698
Ascendente, transistor TTL, 501
Ascendente/descendente, ADC de rampa digital, 757
ASCII, código, 39-41
ASICs, 870-871, ASICs, 872
Atascada, transferencia, 246, 380
Auxiliar
almacenamiento, 814
memoria, 786
Avanzado
BiCMOS de bajo voltaje (74ALVT/ALB), 531-532
CMOS
74AC/ACT, 524

de alta velocidad, 74AHC, 525
de bajo voltaje (74ALVC), 531
de muy bajo voltaje (AVC), 531
de ultra bajo voltaje (74AUC), 531
TTL Schottky
de baja potencia, Serie 74ALS (ALS-TTL), 507
Serie 74AS (AS-TTL), 507
ultra baja potencia (74AUP), 531

B

B, registro, 318
Baja potencia, Schottky TTL de, serie 74LS (LS-TTL), 506-507
Bajo voltaje (74LV), 531
características de la serie, 532
CMOS (74LVC), 530-531
tecnología
BiCMOS (74LVT), 531
de voltaje, 530-532
Barril, desplazador de, 767
Base 10, sistema, 10
BCD (decimal codificado en binario), código, 33-35
a 7 segmentos, decodificador/controlador, 584-587
a decimal
decodificador, 581-582
decodificador/controlador, 582
códigos prohibidos, 34
comparación con binario, 35
contadores, 375-376
decodificación, 391-392
visualizar dos dígitos múltiples, 605
resta, 314
suma, 312-314
igual a 9 o menos, 312
mayor que 9, 313-314
ventaja, 35
Bibliotecas de módulos parametrizados, 347-348
BiCMOS, lógica de 5 volts, 525, 531-532
Bidireccionales
buses, 637-638
líneas de datos, 637
Biestables, multivibradores, 211, 260-263 (*vea también* Flip-flops)
Bilateral, interruptor, 546-548
Binario(a)
BCD, 33-35
cantidades, representación de, 13-15
círculos aritméticos y numéricos, 309
contador, 251
decimal codificado en (*vea también* BCD, código)
dígito, 12
división, 311-312
método de paridad para detección de errores, 41-44
multiplicación, 310-311
punto, 12
secuencia de conteo, 13
sistema, 11-13
conversión de binario a decimal, 26
conversión de binario a Gray, 36
conversión de binario a hexadecimal, 31
conversión de decimal a binario, 26-29
conversión de Gray a binario, 36
conversión de hexadecimal a binario, 31
conversiones, resumen, 33
negación, 303
números con signo, representación, 299-306
representación de cantidades, 13-15
transmisión en paralelo y en serie, 17-18
suma, 289-299
Bipolares
CIs digitales, 155-156
ECL, 543-546
DACs, 728
Bit, 12
arreglos, 177-178, 344
de acarreo, 319
más significativo (MSB), 12
menos significativo (LSB), 12
vectores de, 177-178
Booleana
álgebra, 57
compuerta NAND, 73-76
compuerta NOR, 73-76
constantes y variables, 57
descripción de circuitos lógicos, 66-68
evaluación de las salidas de circuitos lógicos, 68-71
implementación de circuitos a partir de expresiones, 71-73
operación AND, 62-65
operación NOT, 57, 65-66
resumen, 66
operación OR, 57-62
resumen, 60
qué representación utilizar, 89-95
representación de compuerta lógica alternativa, 86-89
simplificación de circuitos lógicos, 121
tablas de verdad, 57-58
teoremas, 76-80
de DeMorgan, 80-83
Borrables eléctricamente, PROMs (EEPROMs), 805-807
Borrado
en masa, 809
estado, 213
Borrar
comando, 811
verificar comando, 811
Búfer(es)
circulares, 846
controlador, 536-537
de colector abierto, 536-537
inversores, 539
lineales, 846
no inversores, 539
salida, ROM, 769-797
triestado, 539-540
Burbujas, 88-89
colocación de, 91
Bus
ciclos de reloj, 845
colisión, 540
controladores, 635
de control, 794
de datos, 794
de direcciones, 794
expansión, 634-635
lógica de interfase de alta velocidad, 541-542
representación simplificada, 635-637
señales, 633-634
técnicas de terminación, 542
velocidad de reloj, 845
Buses bidireccionales, 637-638, 794
Búsqueda, tabla de (LUT), 873
Byte, 37-39, 787

C

Caché, memoria, 845
Capacidad de memoria
definición, 787
expansión, 838-841
Características básicas de los CIs digitales, 153-160

Carga
factor de, 491
TTL, 509-514
CASE mediante el uso de
AHDL, 190-191
VHDL, 191-192
Casi estable, estado, 256
Cero
conteo, 12
zócalo de cero fuerza de inserción (ZIF), 172
Chip, 153
Circuito integrado
contadores de registro de desplazamiento de, 449
de contorno pequeño (SOIC), 497
registros de, 437
entrada en paralelo/salida en paralelo; el 74ALS174/74HC174, 437-439
entrada en paralelo/salida en serie; el 74ALS165/74HC165, 441-443
entrada en serie/salida en paralelo; el 74ALS164/74HC164, 443-445
entrada en serie/salida en serie; el 74ALS166/74HC166, 439-441
sumador en paralelo de, 326-328
Circuitos
aritméticos, 317-318
digitales, 15-17 (*vea también* Lógicos, circuitos)
generador de reloj, 263
habilitar/deshabilitar, 151-153
integrados de aplicación específica (ASICs), 870-871
tabla de excitación de, 399
Circuitos integrados, encapsulados, 495-497
comunes, 496
de contorno pequeño (SOIC), 497
distancia entre terminales, 496
duales en línea, 495-496, 497
encapsulado delgado de contorno muy pequeño (TVSOP), 497
encapsulado plano cuádruple (QFP), 497
encapsulado plano cuádruple delgado (TQFP), 497
encapsulado reducido de contorno pequeño (SSOP), 497
encapsulado reducido delgado de contorno pequeño (TSSOP), 497
matriz de rejillas de bolas finas de bajo perfil (LFBGA), 497
portador de chip de plástico con terminales (PLCC), 497
tecnología de montaje superficial, 496
terminales tipo J, 497
tipo ala de gaviota, 497
Circuitos integrados, familias lógicas de, 488-574
ALU(s), 331-335
expansión, 334
operación AND, 333
operación borrar, 332
operación de resta, 332
operación de suma, 332
operación EX-OR, 333
operación OR, 333
operación PRESET, 333
otros, 335
bipolares, 155-156, 543-546
características básicas, 153-160
definición, 489-490
ECL, 543-546
interfases, 548-52
MOS (*vea también* MOS, familia lógica)
resumen, 558
terminología, 490-497
unipolares, 155-156
Circuitos lógicos
análisis, 92-94
mediante el uso de una tabla, 69-71
aritmética, 317-318
circuito formador de pulsos, 365
definición, 16
descripción, 54-117
algebraica, 66-68
deshabilitados, 151-153
diagramas de conexiones, 158-160
dirección de pulsos, 153
evaluación de las salidas, 68-71
habilitados, 151-153
implementación
a partir de expresiones Booleanas, 71-73
con PLDs, 100-101
interfase, 548-552
Circuitos lógicos combinacionales, 118-207
diseño, 127-133
forma de suma de productos, 120-121
generador y comprobador de paridad, 149-151
método del mapa de Karnaugh, 133-144, 322
NOR exclusivo, 144-149
OR exclusivo, 144-149
procedimiento completo de diseño, 128-133
proceso completo de simplificación, 138-141
resumen, 193
simplificación algebraica, 121-126
simplificación, 121-126
sumas de productos, 120-121
Circulante, registro de desplazamiento, 445
Circulares, búferes, 846
Círculos numéricos y aritmética binaria, 309
CLEAR, 234-235
CML (lógica de modo actual), 543
CMOS, familia lógica, 16, 155-158, 521-530
74ALB, 532
avanzado de bajo voltaje, 531
BiCMOS de 5 Volts, 525
BiCMOS de bajo voltaje, 531
características, 523-530
de las series, 523-530
circuito INVERSOR, 155, 521-522
compuerta
de transmisión, 546-548
NAND, 522
NOR, 522-523
control de TTL, 551
en el estado ALTO, 551
en el estado BAJO, 551-552
descarga electrostática (ESD), 529
disipación de energía, 526
efecto de cierre, 529
eléctricamente compatible, 156, 524
entradas
no conectadas, 157-158
sin usar, 528
equivalente funcional, 524
factor de carga, 527-528
FF SET-RESET, 523
interruptor
bilateral, 546-548
TS, 531
márgenes de ruido, 526
memoria Flash (28F256A), 809-811
niveles
de bajo voltaje, 525-526
de voltaje, 525-526
P_D se incrementa con la frecuencia;
rangos de voltajes de niveles lógicos, 157

salidas
de colector abierto, 533-538
en corto, 533
triestado, 538-541
sensibilidad estática, 528-529
serie 74AC, 156-157, 524
serie 74ACT, 156, 524
serie 74AHC, 525
serie 74AHCT, 525
serie 74ALVC, 531
serie 74ALVT, 531
serie 74AUP, 531
serie 74AVC, 531
serie 74C, 156
serie 74HCT, 156, 524
serie 74LV, 531
serie 74LVC, 530-531
serie 74LVT, 531
serie 74VME, 532
series 4000/14000, 156-157, 524
terminales compatibles, 524
tierra, 157
velocidad de conmutación, 528
voltaje(s)
de entrada, 526
de salida, 526
de suministro de energía, 157, 525
Codificación, 591
directa en binario, 33
Codificadores, 591-597
con prioridad, 592-593
de 8 a 3 líneas, 591
de decimal a BCD con prioridad, 592-593
de octal a binario, 591
interruptor, 593-596
Código
alfanumérico, 39-41
BCD, 33-35
definición, 33
Gray, 35-36
juntándolo todo, 37
Código estándar estadounidense para el intercambio de información (ASCII), 39-41
Códigos alfanuméricos, 39-41
Colector abierto
búfer/controladores de, 536-537
salidas de, 233-238, 533-538
Combinación de chips de DRAM, 843
Comparadores de voltaje analógico, 554-556
Complementación, 65 (*vea también* NOT, operación)
Complemento a 1, forma, 300
Complemento a 2
forma, 300
representación de caso especial, 304-305
resta, 307-310
sistema, 299, 328-331
multiplicación, 310-311
resta, 328-331
suma y resta, combinación, 330-331
suma, 328
suma, 306-307
Complemento lógico o inversión (operación NOT), 65-66
Comportamiento
descripción, 409
nivel de abstracción, 409
Comprobador de paridad, 149-151
Compuerta(s)
AND, 63
arreglos, 871
NAND, 73-76
NOR, 73-76
OR, 5862
qué representación utilizar, 89-95
Compuertas lógicas, 57-117
AND, 63
circuito NOT (INVERSOR), 65-66
evaluación de las salidas, 68-71
NAND, 73-76
NOR, 73-76
OR, 58-62
qué representación utilizar, 89-95
representación alternativa, 86-89
representaciones IEEE/ANSI, 95-96
resumen de métodos para describir, 96-98
tablas de verdad, 57-58
teoremas Booleanos, 76-80
teoremas de DeMorgan, 80-83
Computadora
controlador embebido, 21
dedicada, 21
diagrama funcional de, 19
digital, 19-21, 721
lenguajes de programación, 59
microcomputadora, 20
microcontrolador, 21
partes principales de, 19-21
proceso de decisión de un programa, 100
sistema de adquisición de datos, 746
tipos de, 20-21
Común
bloque de control, 237
terminales de entrada/salida (en RAM), 816-817
Concatenación, 182-183, 453
Concesiones (para las memorias no volátiles), 809
Conjuntos, 339
Conmutación, modo de, 227
Conmutativas, leyes, 78
Constantes, 344
Construcción de los bloques desde cero (reloj digital mediante el uso de HDL), 698
Contadores, 360-486
arreglo de múltiples etapas, 388-389
asíncronos (de rizo), 362-365
número MOD, 363-364
tiempo de propagación, 365-367
autocorregibles, 394
básicos mediante el uso de HDL, 405-411
BCD, decodificación, 391-392
con números MOD $<2^N$, 370-377
con retroalimentación, 445
de anillo, 445-447, 594-596
de décadas, 375-376
de rizo, 277-280, 362-365
decodificación de, 381, 389-393, 448-449
deformaciones, 367,372
diagnóstico de fallas, 450-452
diseño síncrono, 396-404
con FFs D, 402-404
en cascada, 388-389
en HDL, características, 412-417
entrada
en paralelo/salida en paralelo;
el 74ALS174/74HC174, 437-439
en serie/salida en paralelo;
el 74ALS164/74HC164, 443-445
estado
PRESENTE, 393-404
SIGUIENTE, 393-404
estados indeseables, 397
HDL básico, 405-411
idea básica, 396
Johnson, 447-449
las series

74ALS160-163/74HC160-163, 380-384
74ALS190-191/74HC190-191, 384-388
pico, 372
prestablecibles, 379-380
procedimiento de diseño, 397-400
reciclar, 363
registro
de desplazamiento, 445-449
de circuitos integrados, 437
resumen, 436-437, 468
síncrono
análisis, 393-396
en paralelo, 367-370, 380-389
en paralelo, descendente y ascendente/descendente, 377-379
síncronos de CI, 380-389
tabla de excitación J-K, 397, 399
visualización de estados, 372
y registros, 360-486
Conteo
binario, 12-13
decimal, 10-11
hexadecimal, 31-32
operación, 251-252
Control
bus de, 794
entradas de, 223-233
síncronas, 223
unidad de, 20
Controlador, decodificador, 582
Convertidor de datos, 813
Convertidores de código, 624-627, 653-655
idea básica, 624-625
implementación del circuito, 626-627
otras implementaciones, 627
proceso de conversión, 625-626
Corriente
acción de drenado, TTL, 495-500
acción de suministro, 495, 500-501
lógica de modo de, 543
parámetros para CIs digitales, 490-491
transientes, TTL, 516-517
transistor
de drenado, TTL, 500
de suministro, TTL, 501
CPU (unidad central de proceso), 20
Cristal
líquido, pantallas de (LCDs), 587-591
controlar una, 588-589
plano posterior, 587
tipos, 589-590
generadores de reloj controlados por, 263
Cuádruple
agrupamiento, 136-137
encapsulado plano (QFP), 497
multiplexores de dos entradas, 603-604
Cuantización, error de, 743
Cuarzo
cristal de, 263
reloj de, 251
Cuatro entradas, multiplexores de, 601
Cuenta, habilitación de, 381

D

D, latch (*vea* Flip-flops)
latch transparente, 232-233, 271
DAC (*vea también* Digital-analógico, convertidor)
Datos
adquisición de, 745-748
almacenamiento y transferencia, 245-247
búfer de transferencia, 846
bus de, 628, 794
definición, 628
flotante, 629
método de agrupamiento, 637
operación, 632-638
convertidor, 813
distribuidores, 610-617
enrutamiento mediante MUXs, 604-606
líneas, 255
muestreo, 745
palabra, 635
selectores, 599-604
tablas, 812-813
tiempo
de establecimiento, 820
de retención, 820
Datos, transferencia, 245-247
asíncrona, 245
buses de datos, 628-629
comparación entre transferencia en paralelo y en serie, 250
economía y simpleza de, 250
velocidad, 250
demultiplexores, 610-617
en paralelo, 231, 246-247
en serie, 247-250
operación, 633
registros de desplazamiento, 247-250
registros, entre, 633
requerimiento de tiempo de retención, 248
simultánea, 247
síncrona, 245
y almacenamiento, 245-247
y portabilidad, 812
Décadas, contadores de, 375-376
Decimal
a BCD, codificador con prioridad (74147), 592-593
conteo, 10-11
punto, 10
sistema, 10-11
conversión de binario a decimal, 26
conversión de decimal a binario, 22-29
conversión de decimal a binario, rango de conteo, 28
conversión de decimal a hexadecimal, 30-31
conversión de hexadecimal a decimal, 29-30
conversiones, resumen, 33
Decisiones, estructuras de control en HDL, 184-192
Decodificación
contadores, 389-393
Johnson, contador, 448-449
Decodificador de 3 líneas a 8 líneas, 578-580
Decodificadores, 577-584
1 de 10, 581-582
1 de 8, 578-580
4 de 10, 581-582
aplicaciones, 582-584
controladores de BCD a 7 segmentos, 584-587
de 3 a 8 líneas, 578-580
de BCD a decimal, 581-582
de binario a octal, 578-580
de columna, 796-797
de dirección, 797-798
demultiplexor, 610-617
entradas de HABILITACIÓN, 578
fila, 796-797
pantallas de cristal líquido (LCD), 587-591
simulación (HDL), 683
uso de HDL, 638-641
Deformaciones, 367-372
DeMorgan, teoremas, 80-83
implicaciones de, 81-83

Demultiplexores (DEMUXs), 610-617, 648-651
de 1 a 8 líneas, 610, 612
sistema de monitoreo de seguridad, aplicaciones, 612-613
Dependencia, notación de, 95-96
&, 95-96, 631
Σ (Sigma), 334
◊, 537
, 541
⟩, 95
≥, 95-96
1, 95-96
C, 236-237
R, 236-237
S, 236-237
Depleción, MOSFET de, 518-519
Desacoplamiento, fuente de alimentación TTL, 517
Desarrollo
sistema de (para programar PLDs), 172
software de (para PLDs), 172-174
Desbordamiento
aritmético, 308-309
bit, 323
Descarga electrostática (ESD), 529
Descendente, transistor TTL, 500
Descripción de circuitos lógicos, 54-117
Desplazamiento
contadores de registros de, 445-449
registros de, 247-250
a la izquierda, 250
bidireccionales universales, 444
entrada en paralelo/salida en paralelo; el 74ALS174/74HC174, 437-439
entrada en serie/salida en serie; el 74ALS166/74HC166, 439-441
octales (8 bits), 444
Detección
de errores, método de paridad para, 41-44
de una secuencia de entrada, 244-245
Detector de flancos, circuito, 226-227
Diagnóstico de fallas
buscar nodos en corto, 557
circuitos de flip-flops, 264-268
entradas abiertas, 264-265
salidas en corto, 265-266
contadores, 450-452
convertidores digitales-analógicos, 738-739
decodificadores, circuito con, 597-599
diagrama de árbol, 620
dividir y conquistar, 597
ejemplo práctico
compuertas, 168-170
sumador/restador binario en paralelo, 335-337
falla
aislamiento, 160
CI externo, 166-168
corrección, 160
detección, 160
fallas internas en CIs, 162-166
herramientas utilizadas en, 161, 556-557
observación/análisis, 597
pasos básicos, 160-162
sistema
de monitoreo de seguridad, 612-613
de transmisión síncrona de datos, 613-617
sistemas de RAM, 847-852
conocer la operación, 847-850
probar el sistema completo, 851-852
probar la lógica de decodificación, 850-851
sistemas digitales, 160-162, 556-557 (*vea también* Digitales, sistemas)
sistemas lógicos secuenciales, 450
sumador/restador binario en paralelo, 335-337
Diagrama de bloques (reloj digital mediante el uso de HDL), 694
Diagramas
de conexiones de circuitos lógicos, 158-160
de sincronización, 394
de bus simplificado, 634
de transición de estado, 252-253, 372-373, 394
Diferenciales, entradas, 751
Digital(es)
ADC de rampa, 740-745
ascendente/descendente, 757
cantidad, 719
circuitos, 15-17
integrados, 16
computadoras, 19-21 (*vea también* Microcomputadora)
control de amplitud, 737
monoestables, HDL, 461-467
multiplexor, 599
osciloscopio de almacenamiento (DSO), 767-765
aplicaciones relacionadas, 765
proyecto de reloj (HDL), 693-710 (*vea también* HDL)
pulsos, 220-221
representación, 4-5
sistemas numéricos, 10-13
Digital, aritmética, 296-358
circuitos, 317-318
y operaciones, 296-358
círculos numéricos y aritmética binaria, 309
división binaria, 311-312
multiplicación binaria, 310-311
operaciones y circuitos, 296-358
propagación de acarreo, 325-326
representación
de números con signo, 299-306
hexadecimal de números con signo, 316-317
resta hexadecimal, 315-316
resumen, 349-350
sistema de complementos a 2
multiplicación, 311
resta, 307-310, 328-331
suma, 306-307, 328
suma
BCD, 312-314
binaria, 298-299
hexadecimal, 314-315
sumador
binario en paralelo, 318-320
completo, 319
en paralelo de circuito integrado, 326-328
Digital-analógico, convertidor (DAC) 7, 720-721
aplicaciones, 736-738
bipolar, 728
circuito integrado (AD7524), 735-736
circuitos, 728-733
control, utilizado en, 736
conversión, 721-728
analógica-digital, utilizado en, 740-745
diagnóstico de fallas, 738-739
digitalización de una señal, 736
en serie, 737-738
error de desplazamiento, 734-735
escalera, 724
escalera R/2R, 732-733
especificaciones, 733-735
forma de onda de salida, 724
monotonicidad, 735
ponderaciones de entrada, 723
precisión, 734
de la conversión, 730-731
prueba
de escalera, 738
de precisión estática, 738

reconstrucción de señales, 736-737
resolución, 724
porcentual, 725-726
qué significa, 726
revisión de comparación entre, 719-721
salida
a escala completa, 721, 724
analógica, 723
de corriente, con, 731-732
tamaño de intervalo, 724
tiempo de establecimiento, 735
Digitales, sistemas, 5-6
árbol familiar, 870-875
asignaciones típicas de voltaje, 15
carga de salida, 168
conceptos introductorios, 2-23
resumen, 21
corto entre dos terminales, 165-166
diagnóstico de fallas, 160-162, 556-557, 597-599, 671-620
diagrama de árbol, 620
ejemplo práctico, 168-170
entrada
en corto interno con tierra o con la fuente, 162-163
o salida con circuito abierto, 163-165
falla en los circuitos internos, 162
fuente de alimentación defectuosa, 167
líneas
de señal abiertas, 166-167
de señal en corto, 167
salida en corto interno con tierra o con la fuente, 163
señal típica, 15
y analógicos, sistemas, 5-9
Digitales, técnicas
limitaciones, 6-9
ventajas, 6
Digitalizar
reconstrucción de una señal, 746-748
señal, 736-737, 745-748
Dígito más significativo (MSD), 10
Dígito menos significativo (LSD), 10
Dígitos, 4, 10
DIMM (módulo de memoria dual en línea), 834
Dinámica, RAM (DRAM), 823-824
combinación de chips, 843
contador de regeneración, 831
controlador, 832
DDRSDRAM, 835
DIMM, 834
DRDRAM, 835
EDO, 835
estructura y operación, 824-829
FPM (Modo de paginación rápida), 834-835
lectura/escritura, ciclos de, DRAM, 829-831
escritura, ciclo de, 830
lectura, ciclo de, 829-830
módulos de memoria, 834
multiplexaje de direcciones, 825-829
regeneración, 823, 831-833
métodos,
$\overline{\text{CAS}}$ antes que $\overline{\text{RAS}}$, 832, 835
distribuida, 831
en ráfaga, 831
regeneración sólo de $\overline{\text{RAS}}$, 831
SDRAM (DRAM síncrona), 835
SIMM, 834
SLDRAM, 835
SODIMM, 834
tecnología, 834-835
Diodo de barrera Schottky (SBD), 506
Diodos emisores de luz (LEDs), 586-587
ánodo común y cátodo común, comparación, 586
DIP (encapsulado dual en línea), 153
Dirección, 788
bus de, 794, 836
código de, 254
decodificación incompleta de, 841-843
decodificadores de, ROM, 797-798
entradas de, 599, 790-791
multiplexaje de (en DRAM), 825-829
registros apuntadores, 845
tiempo de establecimiento, 820
unidireccional, 794
Discretos, intervalos, 4
Diseño de circuitos lógicos combinacionales, 127-133
Disparador de Schmitt
dispositivos, 256
oscilador, 260-261
respuesta a una entrada lenta y ruidosa, 257
Disparo, entrada de, 225
Dispositivos lógicos programables complejos (CPLDs), 872
Dispositivos lógicos programables simples (SPLDs), 872
Distributiva, ley, 78
Dividendo, 311
Dividir y conquistar, proceso de diagnóstico de fallas, 597
División binaria, 311-312
Divisor, 311
Doble velocidad de datos, SDRAM (DDRSDRAM), 835
Dos entradas, multiplexor básico de, 600-601
DRAM Rambus directa (DRDRAM), 835
DSP (Procesamiento de señales digitales), 765-768
Dual
en línea, encapsulado (DIP), 153, 497
pendiente, ADC, 757-758

E

ECL, familia de circuitos integrados, 16, 543-546 (*vea también* Emisor acoplado, lógica de)
EDO (Salida extendida de datos) DRAM, 835
EEPROMs (PROMs borrables eléctricamente), 805-807
Efecto de cierre, 529
ELSIF, 187-189
uso de AHDL, 189
uso de VHDL, 189
Embebido
controlador, 21
memoria de programa de microcontrolador, 812
Emisor acoplado, lógica de (ECL), 16, 543-546
características, 543-546
circuito básico, 543-544
compuerta OR/NOR, 543, 545
En cascada, sumadores paralelos, 326-328
Encapsulado
delgado de contorno muy pequeño (TVSOP), 497
plano cuádruple delgado (TQFP), 497
reducido delgado de contorno pequeño (TSSOP), 497
Encapsulado reducido de contorno pequeño (SSOP), 497
Energía
almacenamiento al corte de, 844-845
auto prueba al encendido, RAM, 852
corte (en MROM), 802
desacoplamiento de la alimentación, TTL, 517
requerimientos para los CIs digitales, 492-493
Enlace de fusible, PROMs, 803
Enriquecimiento, MOSFET de, 518-519
Entrada
corrientes para dispositivos estándar, 550
detección de secuencia de, 244-245
unidad de, 20
Entradas
asíncronas, 233-236
designaciones para, 234-235
conectadas entre sí, TTL, 515

EPROMs (ROMs programables y borrables), 803-805
Error
  de desplazamiento, 734-735
  de linealidad (de un DAC), 734
Escala completa
  error (de un DAC), 734
  salida (de un DAC), 721, 724
Escalera
  forma de onda, de un DAC, 724
  prueba, de un DAC, 738
  R/2R, 732-733
Escritura, ciclo de, 255
  tiempo, 820
    de establecimiento de datos, 820
    de establecimiento de dirección, 820
    de retención de datos, 820
Escritura, operación de
  CPU, 794
  definición, 788
  RAM, 816
Establecer
  -borrar, 811
  el flip-flop
    latch, 212
    y restablecer en forma simultánea, 213
  programar/control de programa, 811
Estado
  bajo, margen de ruido (V_{NL}), 493
  descripciones
    en AHDL, 406-407
    en VHDL, 407-408
  diagrama de transiciones, 252-253, 372-373
    contador MOD-6, 373
    contador síncrono, 398
  máquinas de, 425-437
    controlador de semáforo, 429-435
  métodos de descripción de transiciones, 405
  PRESENTE, 393-404
  SIGUIENTE, 393-404
  tabla, 252
Estándar
  ASICs de celda, 871-872
  lógica, 871
Estática, RAM (SRAM), 818-822
  ciclo de escritura, 820-821
  ciclo de lectura, 820
  chip real (MCM6264C), 821-822
  sincronización, 818-819
Estrobo
  de dirección de columna (CAS), 827
  de dirección de fila (RAS), 827
  entradas tipo (en DRAM), 827
Etiquetado
  señales activas en BAJO, 94
  señales biestado, 94
Evento, 272
Exclusivo
  circuito NOR, 145-147
  circuito OR, 144-145
Expansión de la capacidad de bits de un circuito, 343-348
Extensión, signo, 302

F

Factor de carga, 491
  CMOS, 527-528
  determinar, 510-513
  TTL, 509-514
Fallas externas, 166-168
Flanco, dispositivos disparados por, 272-277
  evento, 272
  primitiva lógica, 272
Flancos de una señal de reloj, 221-222
Flash
  ADC, 755-757
  memoria, 808-811
  tiempo de conversión de, 756-757
Flip-flops, 19, 228-295
  aplicaciones, 243
  borrar, 211
  características de memoria, 211
  CIs reales, 240
  consideraciones de sincronización, 238-241
  D (datos), 230-231
    implementación de, 231
  definición, 210-211
  detección de la secuencia de entrada, 244-245
  diagnóstico de fallas en circuitos, 264-268
  disparados por flanco, 222
  dispositivos relacionados, 208-295
  división y conteo de frecuencia, 250-254
  entradas asíncronas, 223-226
  estado al encendido, 218
  ignorar entradas, 234
  latch D (latch transparente), 232-233, 271
  latch de compuerta NAND, 211-216, 226
    ejemplo práctico de diagnóstico de fallas, 219-220
    representaciones alternativas, 214
    resumen de, 213
    uso de AHDL, 270
  latch de compuerta NOR, 216-218
  latches, 19
  multivibrador biestable, 211
  problemas de sincronización, 241-242
  registros de desplazamiento, 247-250
  restablecer, 211
  resumen, 280-281
  salida ambigua, 227
  señales de reloj, 221-224
  sincronización, 243-244
  sincronizados, 221-224
    D, 230-231
    entradas asíncronas, 233-236
    J-K, 227-229
    latch D (latch transparente) , 232-233, 271
    S-R, 224-227
  sincronizados por reloj, 221-224
    en D, 230-231
    en D, implementación, 231
    en J-K, 227-229
    en S-R, 224-227
  terminología, 214-215
  tiempos
    de establecimiento y retención, 223-224
    de propagación, 23-239
Flotante
  bus, 629
  compuerta, EEPROMs, 804
  entradas, 157-158, 514
Frecuencia, 222
  división de, 250-254
    y conteo, 250-254
  proyecto de contador (HDL), 710-714 (*vea también* HDL)
Funciones especiales de memoria, 844-846
Funciones, generador de, 813-814
Futuro digital, 8

G

GAL16V8 (Lógica de matriz genérica), 881-885
  macroceldas de salida lógica (OLMCs), 883-885
  modo complejo, 895
  modo registrado, 885

modo simple, 885
multiplexor de retroalimentación (FMUX), 884
multiplexor de salida (OMUX), 884
multiplexor de términos de productos (PTMUX), 884
GAL22V10 (Lógica de matriz genérica), 885
Generador
de funciones, 813-814
de paridad, 149-151
Giga escala, integración a (GSI), 154-155
Glosario, 898-910
Gran escala, integración a (LSI), 154-155, 489
Gray, código de, 35-36

H

HABILITACIÓN, entradas de decodificadores, 578
Habilitar/deshabilitar circuitos, 151-153
HDL, 98, 173
administración de pequeños proyectos, 678-679
definición, 678
integración y prueba de sistemas, 679
planeación estratégica, 678
síntesis y prueba, 678-679
alambrado de módulos entre sí, 417
anidamiento, 686
arreglos de bits, 177-178
CASE, 459, 639, 683, 701
circuitos con múltiples componentes, 277-280
circuitos secuenciales mediante el uso de, 268-271
latch D, 271
latch NAND, 270
codificadores, 645-648
combinación de bloques utilizando sólo, 706-707
comparador, 652-653
de magnitud, 652-653
concatenación, 453
contadores
básicos mediante el uso de, 405-411
con todas las características en, 412-417
de anillo, 459-461
convertidores de código, 653-655
decodificador/controlador de 7 segmentos, 642-645
decodificadores mediante el uso de, 638-641
demultiplexores, 648-651
descripción del comportamiento, 409
diagrama esquemático, 102
diseño
de sistemas numéricos, 177
jerárquico, 417
escalares, 177
estructuras de control de decisiones, 184-192
concurrentes, 184
secuenciales, 184
formato, 102-104
IF/ELSE, 184-185, 460
IF/ELSIF, 701
IF/THEN, 185
índice, 178
literales, 177
métodos de descripción de transiciones de estado, 405
MOD-12, diseño de, 701-702
MOD-60, símbolo de bloque gráfico, 706
modo, 102
monoestables (one-shots), 461-467
multiplexores, 648-651
nivel
de abstracción de comportamiento, 409
estructural de abstracción, 280
PRESENTE, 459
prototipo de función, 338
proyecto de codificador de teclado numérico, 687-693
análisis del problema, 687
diagrama de bloques, 687
planeación estratégica, 689
simulación, 693
proyecto de contador de frecuencia, 710-714
bloque de sincronización y control, 713
diagrama de bloques, 711
diagrama de sincronización, 712
intervalo de muestreo, 710
proyecto de controlador de motor paso a paso, 679-686
enunciado del problema, 680-681
planeación estratégica, 681
secuencia de control de onda, 680
secuencia de medio paso, 680
secuencia de paso completo, 679
síntesis y prueba, 681
proyecto de reloj digital, 639-710
circuito de sección de horas, 695
combinación de bloques en forma gráfica, 705-706
construcción de los bloques desde cero, 698
diagrama de bloques, 694
diseño jerárquico de arriba-abajo, 696-698
jerarquía completa del proyecto, 697
preescalador, 697
sección MOD-60, 697
simulación de contador MOD-6, 700
proyectos que utilicen, 676-717
redisparables, monoestables disparados por flanco en, 464-465
registros, 452-459
representación de datos, 177-181
SIGUIENTE, 459
simulación
de contador básico, 411
de contador con todas las características, 416
de sincronización, 175
sintaxis, 102-104
sumadores, 340-343
tablas de verdad, 181-184
TABLE, 639
tipo, 102
vectores de bits, 177-178
Hertz, 222
Hexadecimal
aritmética, 314-317
representación de números con signo, 316-317
resta, 315-316
sistema numérico, 29-33
suma, 314-315
Híbridos, sistemas, 8
Horas, circuito de sección (reloj digital mediante el uso de HDL), 695

I

IEEE/ANSI, símbolos lógicos estándar, 95-96, 236-237, 537-541
AND, 95
bloque de control común, 237
definición, 95-96
flip-flop D, 237
flip-flop J-K, 237
flip-flops, 236-238
inversor, 95
monoestable (one-shot), 256-260
multivibrador monoestable, 256-260
NAND, 95
NOR, 95
NOR exclusivo, 145-147
notación de dependencia (*vea también* dependencia, notación de)
OR, 95
OR exclusivo, 144-145
para CIs de compuertas lógicas, 95

salida de colector abierto, 537
salidas triestado, 541
tradicionales, 95-96
IF/ELSE, 184-185
IF/THEN, 185
IF/THEN/ELSE mediante el uso de AHDL, 186
Ignorar entradas, 234
Implementación de circuitos lógicos con PLDs, 100-101
Implicaciones de los teoremas de DeMorgan, 81-83
Incompleta, decodificación de dirección, 841-843
Indeterminado(s)
nivel lógico, 161
voltajes, 157
Inhibición, circuitos de, 64
Instrucción de asignación de señal condicional, 647
Integración
a pequeña escala, (SSI), 154-155, 489
y prueba del sistema (mediante el uso de HDL), 679
Interfaces
circuito integrado, 548-552
CIs lógicos, 549
con el mundo analógico, 718-182
resumen, 769-770
no necesarias, 549
requeridas, 549
salidas de alto voltaje que controlan cargas de bajo voltaje, 554
salidas de bajo voltaje que controlan cargas de alto voltaje, 553
TTL y CMOS de 5 V, 550
voltaje mixto, 553-554
Intermedias, señales, 105-108
Interpolación, filtrado por, 767
Interruptor
bilateral, 546-548
codificadores, 593-596
eliminación de rebotes, 215
Inválidos, niveles de voltaje, 494-495
Inversión (*vea también* NOT, operación), 57, 65-66
Inversor, 65-66
búfer triestado, 539
circuitos que contienen, 67-68
controlado, 147
inversor controlado, 147
respuesta a un inversor lento y ruidoso, 257
Invertida, salida de flip-flop, 210

J

JEDEC
encapsulado de memoria estándar (JEDEC), 822
estándar, 172
Jerarquía, 696
completa del proyecto (reloj digital mediante el uso de HDL), 697
Jerárquico, diseño, 173
J-K, tabla de excitación, 397-399
Johnson, contador, 447-449
decodificación, 448-449

K

Karnaugh, mapa de
agrupamiento, 135-138
condiciones "no importa", 142-143
formato, 134-135
llenar a partir de una expresión de salida, 141
método, 133-144
proceso completo de simplificación, 138-141
resumen, 143-144
simplificación, 322

L

Latches, 18, 211-218, 232-233, 268-271 (*vea también* Flip-flops)
establecimiento, 212
restablecimiento, 212
Latencia, 827
LCDs con luz posterior, 587
Lectura, operación de
CPU, 794
definición, 788
RAM, 816
Lectura/escritura
entrada (R/W), 791
memoria (RWM), 789
Leer, comando, 810
Lenguaje de descripción de hardware (HDL), 98, 173
Lenguajes de descripción y lenguajes de programación, comparación, 98-100
Leyes asociativas, 78
Limitaciones de las técnicas digitales, 6-9
Llenado de mapas K a partir de la expresión de salida, 141
Locales, señales, 105
Lógica de transceptor de disparo Plus, 542
Plus (74GTLP1394), 531
Lógico(a)
ciclo de vida de producto, 532
diagrama mediante el uso de captura esquemática, 160
estándar, 871
generación de funciones, 607-609
nivel, 57
primitiva, 272
pulsador, cómo utilizar, 161, 556-557
sonda, cómo utilizar, 161, 556-557
LPMs (biblioteca de módulos parametrizados), 348
LUT (tabla de búsqueda), 873
diagrama de bloques funcional, 891
LVDS (señalización diferencial de bajo voltaje), 542

M

Macrofunción, 337
Magnitud
comparador, 621-624, 652-653
aplicaciones, 623
entradas de datos, 621
entradas en cascada, 622-623
salidas, 623
de números binarios, 299
Máquinas de estado, 425-437
controlador de semáforo, 429-435
modelo de Mealy, 426
modelo de Moore, 426
Margen de ruido, 493
CMOS, 526
DC, 493
de estado alto (V_{NH}), 493
Máscara, programados por
dispositivos de almacenamiento, 786
matrices de compuerta (MPGAs), 871
ROM (MROM), 800-802
Matrices de compuertas programables en el campo (FPGA), 872
Matriz de registros, 796-797
Máxima frecuencia de reloj ($f_{MÁX}$), 239
Mealy, modelo de, 426
Media escala, integración a (MSI), 154-155, 489
Medio paso, secuencia (Motor paso a paso en HDL), 98, 173
Medio sumador (HA), 322-323
Megafunciones, 348
Mejorada, memoria Flash, 811
Memoria, 18-19, 784-866
arranque, 812

auxiliar, 786-789
capacidad, 787
CD-ROM, 807-808
celda, 787
celda de RAM estática bipolar, 818
celda RAM estática NMOS, 818
conexiones, CPU, 793-794
de acceso aleatorio, 788-789
de acceso secuencial, 789
de lectura/escritura, 789
de sólo lectura, 789
de trabajo, 786, 789
densidad, 788
dinámicos, dispositivos, 789
disco compacto, 786
dispositivos, 784-866
empalme, 842
estática, dispositivos de, 789
expansión
de la capacidad, 838-841
del tamaño de palabra, 836-838
Flash, 808-811
borrado de sectores, 808
borrado en agrupamiento, 808
CI (28F256A), 809-811
comando de borrado, 811
comando de verificación de borrado, 811
comando Leer, 810
comando programar, 811
comando programar-verificar, 811
concesiones, 809
establecer programar, 811
establecer-borrar, 811
mejorada, 811
registro de comandos, 810
funciones especiales, 844-846
almacenamiento en corte de energía, 844-845
memoria caché, 845
primero en entrar, primero en salir, 845-846
habilitar, 791-792
mapa, 842
masiva, 789
módulos, 834, 837
no volátil, 788-789, 795
operación
de almacenamiento, 788
de búsqueda. 788
general, 790-793
palabra de, 787
primero en entrar, primero en salir (FIFO), 845-846
principal, 786, 789
programa de microcontrolador embebido, 812
resumen, 853-854
terminología, 786-789
unidad de, 20
volátil, 788
Microcomputadora
aplicación, 254-255
definición, 20
unidad de entrada, 20
unidad de memoria, 20
unidad de salida, 20
Microcontrolador, 21
Microprocesador, 20
ESCRIBIR, operación, 734
LEER, operación, 794
procesamiento de señales digitales (DSP), 871
Minuendo, 308
MOD, número, 253, 363-364
contador de anillo, 445-447
contador Johnson, 447-449
modificación, 374
procedimiento general, 374-375
Monoestable, multivibrador, 256-260.
Monotonicidad (de un DAC), 735
Moore, modelo de, 426
Mordentado incompleto, 167
MOS
descarga electrostática (ESD), 529
familia lógica, 518-521
FETs, 518-521
NMOS, 519-520
sensibilidad estática, 528-529
tecnología, 518-521
MOSFET, 16, 518-521
circuitos digitales, 519-520
CMOS, 521-523
interruptor básico, 519-521
Motor paso a paso
control, 401-402
proyecto de controlador (HDL), 679-686 (*vea también* HDL)
universal, circuito de interfase, 682
MSI, circuitos lógicos, 576-563
buses de datos, 628-629
codificadores, 591-597
decodificador de BCD a decimal, 581-582
decodificador/controladores de BCD a 7 segmentos, 584-587
decodificadores, 577-584
demultiplexores (DEMUXs), 610-617
multiplexores (MUX), 599-604
pantallas de cristal líquido (LCDs), 587-591
registros triestado, 629-631
resumen, 656-657
Muestreo, 745
de frecuencia, 748
y retención, circuitos de, 761-762
Múltiple emisor, transistor con entrada de, 498
Multiplexaje, 599
ADC, 762-764
de direcciones (en DRAM), 825-829
Multiplexores (MUX), 599-604, 648-651
aplicaciones, 604-609, 612-617, 828
cuádruples de dos entradas, 603-604
de cuatro entradas, 601
de dos entradas, básicos, 600-601
de ocho entradas, 601-602
secuencia de control, siete pasos, 608
secuencia de operaciones mediante el uso de, 607
Multiplicación
AND, 62
de números binarios, 310-311
en el sistema de complementos a 2, 311
Multivibrador de funcionamiento libre, 260-263
Multivibradores astables, 260-262
uso de un temporizador 555 como, 261-263

N

NAND, compuerta, 73-76
circuitos internos del FF J-K disparado por flanco, 229
circuitos internos del FF S-R disparado por flanco, 226-227
CMOS, 522
decodificación de contadores, 389-393
definición, 75
flip-flop latch, 211-216
resumen de, 213
qué representación utilizar, 89-95
representación alternativa, 86-89
TTL, 498
universalidad de, 83-86
Negación, 303
Nibble, 37-39

Niveles asignados, 94
N-MOS, 519
  celda RAM estática, 818
  familia lógica, 16
    circuitos lógicos, 519
  P-MOS, 520
No asignados, niveles, 94
No conectadas, entradas
  CMOS, 157-158, 528
  TTL, 157-158, 514
No importa, condiciones, 142-143
No redisparable, monoestable, 258
No volátil, memoria, 788-789, 795, 803
NOR, compuerta, 73-76
  CMOS, 522-523
  definición, 73
  ECL, 543, 545
  latch, 216-218
  qué representación de compuerta utilizar, 89-95
  representación alternativa, 86-89
  universalidad de, 83-86
Normal, salida de flip-flop, 210
NOT
  circuito (INVERSOR), 65-66
    circuitos que contienen, 67-68
    definición, 66
    implementación a partir de expresiones Booleanas, 71-73
    inversor controlado, 147
    NMOS, 519
    qué representación utilizar, 89-95
    representación alternativa, 86-89
    símbolo, 65
    teoremas de DeMorgan, 80-83
  operación, 57, 65-66
Numérica, representación 4-5
Numéricos, sistemas 10-13
  aplicaciones, 44-46
  binario, 11-12
  decimal, 10
  digital, 10-13
  hexadecimal, 29-33
  juntándolo todo, 37
  resumen, 46
  y códigos, 24-52
Números con signo
  en forma de magnitud-signo, 299
  representación, 299-306

O

Oblicuidad, reloj, 266-268
Observación/análisis, proceso de diagnóstico de fallas, 597
Ocho entradas, multiplexores de, 601-602
Octal a binario, decodificadores de, 591
Octetos, agrupamiento, 137-139
Ocultos, nodos en AHDL, 105-106
One-shot (multivibrador monoestable), 256-260
  AHDL, 462, 465
  dispositivos reales, 259
  HDL, 461-467
  redisparable, disparado por flanco en HDL, 464-65
  VHDL, 462-464, 466-467
Operación
  buscar, 788
  de transferencia de datos, 245
  regenerar, 789
Operacional, amplificador (en un DAC), 728
Operaciones lógicas, 57
  sobrearreglos de bits, 338-340
OR
  compuerta, 58-62
  definición, 59
  ECL, 543, 545
  implementación a partir de expresiones Booleanas, 71-73
  representación de compuerta lógica alternativa, 86-89
  símbolo, 59
  teoremas Booleanos, 76-80
  operación, 57-62
  qué representación utilizar, 89-95
  resumen, 60
Organizacional, diagrama jerárquico, 175
Oscilador, disparador de Schmitt, 260-261
OTP (ROM programable una sola vez), 803, 873

P

Paginación rápida, modo de (FPM), DRAM, 834-835
Palabra, 37-39
  tamaño de, 39
Pantallas
  LCD, 587-591
    con luz posterior, 587
    Nemática Trenzada (TN), 590
    panel de matriz pasiva, 590
    reflectiva, 587
    Super Trenzada Nemática (STN), 590
    TFT (Transistor de película delgada), 590
  LED, 586-587
    ánodo común, 586
    cátodo común, 586
Paralelo
  carga, 379
  conversión de paralelo a serial, 606-607
  entrada en paralelo/salida en paralelo; el 74ALS165/74HC165, 441-443
  entrada en/salida en paralelo; el 74ALS174/74HC174, 437-439
  sumador binario, 318-320
    circuitos integrados, 326-328
    completo, con registros, 323-325
    ejemplo práctico de diagnóstico de fallas, 335-337
    propagación del acarreo, 325-326
    sistema de complemento a 2, 328-331
  transferencia de datos en, 231, 246-247
    comparación con transferencia en serie, 250
  transmisión, 17-18
  y en serie, transmisión, 17-18
    concesiones entre, 18
Parásito, 529
Pares, agrupamiento, 135-136
Paridad
  bit, 42-44
  comprobación, 150
  comprobador, 149-151
  comprobar la, 43
  errores de
    de dos bits, 43
    de un solo bit, 43
  generador de, 149-151
  impar, método, 42
  método para detección de errores, 41-44
  par, método de, 42
Partes principales de una computadora, 19-21
Paso completo, secuencia (motor paso a paso en HDL), 679
Pendiente positiva
  transición (PGT), 222
  voltaje de umbral (V_{T+}), 256
Periodo, 222
Pico, 372
PIPO (entrada en paralelo/salida en paralelo), 437

PISO (entrada en paralelo/salida en serie), 437
registro en AHDL, 455-456
registro en VHDL, 456
Píxeles, 589
Planeación estratégica (mediante el uso de HDL), 678-681
Plano posterior, LCD, 587
Ponderación binaria, 730
Portador de chip de plástico con terminales (PLCC), 497
Precisión, fuente de referencia, 731
Preescalador (reloj digital mediante el uso de HDL), 697
PRESET, 234
Prestablecibles, contadores, 379-380
Primer sumando, 306, 318
Primero en entrar, primero en salir, memoria (FIFO), 845-846
Prioridad, codificadores con, 592-593, 756
Procesamiento de señales digitales (DSP), 764-768
arquitectura, 767
desplazador de barril, 767
filtrado, 766
filtrado por interpolación, 767
promedio ponderado, 766
sección de multiplicación y acumulación (MAC), 767
sobremuestreo, 767
unidad aritmética-lógica (ALU), 767
Producto de sumas, 120-121
Programables
y borrables, ROMs (EPROMs), 803-805
ROMs (PROMs), 803
Programable(s), dispositivo(s) lógico(s) (PLDs), 99, 170-176, 871
arquitectura(s), 868-918, 877-881
FPGA (matriz de compuertas programable en el campo), 874
FPLA (matriz lógica programable en el campo), 881
lógica de matriz programable (PAL), 873, 879-881
PROMs, 877-879
resumen, 895-896
CPLD, 872
diagrama de flujo del ciclo de desarrollo, 176
diagrama jerárquico organizacional, 175
diseño jerárquico, 173
encapsulado de memoria JEDEC estándar, 822
FPGA, 872
fundamentos de los circuitos de PLDs, 875-877
hardware, 170-171
HCPLD, 872
lógica de matriz genérica (GAL16V8), 881-885
lógica de matriz programable (PAL), 873
macrocelda, 873
más acerca de, 872-875
matrices de compuertas programadas por máscara (MPGAs), 871
proceso de diseño y desarrollo, 174-175
arriba-abajo, 174
vectores de prueba, 175
programables una sola vez (OTP), 873
programación, 171-172
estándar JEDEC, 172
JTAG, 172
sistema de desarrollo, 172
zócalo de cero fuerza de inserción (ZIF), 172
programador, 172
universal, 172
simbología, 876-877
software de desarrollo, 173-174
AHDL, 173
compiladores, 101
simulación de sincronización, 175
VHDL, 173
SPLD, 872
tabla de búsqueda (LUT), 873
Programación, lenguajes de, 99
Programador, 172
Programar
comando, 811
comando verificar, 811
definición, 6, 19
PROMs (ROMs programables), 803
Propagación, tiempos de (t_{PLH}/t_{PHL}),
circuitos integrados, 491
compuerta NAND TTL, 505
en contadores asíncronos, 365-367
flip-flop, 238-241
Prototipo de funciones, ALTERA, 338
Proyectos de sistemas digitales mediante el uso de HDL, 676-717
resumen, 714
Proyectos mediante el uso de HDL, 676-717
administración, 678-679
codificador de teclado numérico, 687-693
contador de frecuencia, 710-714
controlador de motor paso a paso, 679-686
reloj digital, 693-710
Prueba automática de circuitos (mediante el uso de DACs), 736
Prueba de precisión estática, de un DAC, 738
Pulso(s)
circuito de dirección de, 153, 226
circuito formador de, 365
flanco anterior, 221
flanco posterior, 221
negativo, 220-221
positivo, 220-221

R

R/2R, convertidores digitales-analógicos tipo escalera, 732-733
RAMs (memorias de acceso aleatorio),
arquitectura, 815-817
autoprueba al encendido, 852
definición, 788-789
diagnóstico de fallas, 847-852
conocer la operación, 847-850
probar el sistema completo, 851-852
probar la lógica de decodificación, 850-851
dispositivos dinámicos, 789
estática (SRAM), 818-822
expansión
de la capacidad, 838-841
del tamaño de palabra, 836-838
semiconductoras, 814-815
Rastreo, ADC, 757
Rebote de contactos, 215
Reconstrucción de una señal digitalizada, 746-748
Redisparable, monoestable (one-shot), 258-259
Reflectivas, LCDs, 587
Regeneración
contador de, 831
de DRAM, 789, 818, 823, 831-833
Registro
acumulador, 318
de comandos, 810
Registros, 245, 360-486
acumulador, 318
apuntador de dirección, 845
arreglo de, 797
B, 318
en HDL, 452-459
notación, 323-325
operación de desplazamiento a la izquierda, 250
secuencia de operaciones, 325
sumador completo en paralelo con, 323-325
triestado (74ALS173/HC173), 629-631
y contadores, 360-486

Reloj
circuitos generadores, 260-263
controlado por cristal, 263
definición, 221
flancos, 222
frecuencia, 222
oblicuidad, 266-268
periodo, 222
pulso
en ALTO en t_W(H), 239
en BAJO en t_W(L), 239
señales, 221-224
tiempos de transición, 239
Repetida, método de división, 27-29
Representación
alternativa de compuertas lógicas, 86-89
de cantidades binarias, 13-15
de datos en HDL, 177-181
de números con signo, 299-306
uso del complemento a 2, 300-306
Reproductor de CD, diagrama de bloques, 174
RESET, 235
Resolución
ADC, 742-744
DAC, 724, 733-734
porcentual, 725-726
qué significa, 726
Respuestas a los problemas seleccionados, 911-918
Resta
BCD, 315-316
hexadecimal, 315-316
sistema de complemento a 2, 307-310, 328-331
Restablecer un flip-flop
definición, 212
latch, 212
y establecerlo en forma simultánea, 213
Retroalimentación, multiplexor (FMUX), 884
ROM (memoria de sólo lectura), 795-796
aplicaciones, 811-814
arquitectura, 796-798
búferes de salida, 798
borrado de, 795
búfer de salida, 796-797
CD, 807-808
Decodificador
de columna, 796-797
de fila, 796-797
definición, 789
diagrama de bloques, 795
LECTURA, operación, 795-796
programable una sola vez (OTP), 803
programación, 795
por máscara, 800-802
prueba, 852-853
quemar, 795
sincronización, 799-800
tipos de, 800-808
Ruido, 6, 264
inmunidad al, 493
Ruido eléctrico, 41

S

S, 236-237
Salida
búferes, ROM, 798
carga, 168
corrientes para dispositivos estándar, 550
tiempo de habilitación (t_{OE}), 799
unidad, 20
SAM (memoria de acceso secuencial), 789
SBD (Diodo de barrera Schottky), 506
Schottky
diodo de barrera (SBD), 506
TTL, serie 74S, 506
SDRA (DRAM asíncrona), 835
Sector, borrado de, 808
Secuencial, memoria de acceso (SAM), 789
Secuenciales
circuitos, 243
diseño, 396-404
mediante el uso de HDL, 268-271
sistemas lógicos, diagnóstico de fallas, 4540
Selección
de chip, 795-816
entradas de (en MUXs), 599-600
Sensible, amplificador (en DRAM), 825
Señal
alias, 748
contención, 165
flujo, 363
Señales
digitales y diagramas de sincronización, 15
lógicas
etiquetado de activas en BAJO, 94
etiquetado de biestado, 94
Serial
entrada/salida en serie; el 74ALS166/74HC166, 439-441
transferencia de datos, 247-250
entre registros, 248-249
transmisión, 17-18
Serie
74AC, 156-157, 524
74ACT, 156-157, 524
74ALB, 532
74ALS TTL, 507-508
74ALVC, 531
74ALVT, 531
74AS TTL, 507-508
74AUC, 531
74AVC, 531
74C, 156-157
74CBT, 531
74CBTLV, 531
74F-Rápida TTL, 508
74GTLP, 531
74HC, 156-157, 524, 525
74HCT, 156-157, 524, 525
74LS TTL, 156, 506-508
74LV, 531
74LVC, 530-531
74LVT, 531
74S TTL, 506, 508
74SSTV, 531
74TTL, 156, 506, 508
74TVC, 531
Sigma (Σ), 326
modulación delta (ADC), 758-761
Signo
bit de, 299
extensión de, 302
sistema de magnitud, 299
Símbolos lógicos, interpretación de, 88-89
resumen, 89
SIMM (módulo de memoria sencillo en línea), 834
Sin utilizar, entradas
CMOS, 157-159, 528
TTL, 157-158, 514-515
Sincronización
de flip-flops, 243-244
diagramas de, 15, 394
bus simplificado, 634
problemas en circuitos de Flip-flops, 241-242

Síncrono(a)
diseño de contadores con FF D, 403
entradas de control, 223, 233
prestablecimiento, 380
sistemas, 221
transferencia, 245
Síncronos (en paralelo), contadores, 367-370
CIs reales, 369-370
descendente y ascendente/descendente, 377-379
diseño, 396-404
control de motor paso a paso, 402
operación, 369
prestablecible, 379-380
ventajas sobre los asíncronos, 369
Síntesis y prueba (mediante el uso de HDL), 681
SIPO (registro de entrada en serie/salida en paralelo), 437
SISO (registro de entrada en serie/salida en serie), 437
registro en AHDL, 453-454
Sistema
de monitoreo de seguridad, 612-613
de transmisión síncrona de datos, 613-617
operación del receptor, 614-615
operación del transmisor, 614
sincronización del sistema, 615-616
Sistemas asíncronos, 221
SODIMM (módulo de memoria dual en línea de contorno pequeño), 834
Soldadura, puentes de, 167
SPLDs (dispositivos lógicos programables simples), 872
SUBDESIGN, 103, 178
Submuestreo, 748
Subpíxeles, 589
Suma
bit de, 319
de comprobación, 852
de productos, forma, 120-121
Suma en
BCD, 312-314
binario, 298-299
hexadecimal, 314-315
OR, 58-62
sistema de complemento a 2, 306-307, 328
dos números negativos, 307
dos números positivos, 306
número positivo y número negativo más grande, 307
número positivo y número negativo más pequeño, 306
números iguales y opuestos, 307
Sumador completo, 319
diseño de, 320-323
simplificación de mapa K, 322
Sumadores
completos, 319
en paralelo, 318-320
Sumando, 306, 318
Sustraendo, 308
Sustrato, 153

T

Tabla
análisis mediante el uso de, 69
de búsqueda (LUT), 873
de estado, 252
de excitación
de circuitos, 397, 399
J-K, 397, 399
Tablas de verdad, 57-58
mediante el uso de AHDL, 181-182
mediante el uso de HDL, 181-184
mediante el uso de VHDL, 182-183
Tamaño de intervalo, 724
Teclado numérico, proyecto de codificador de (HDL), 687-693 (*vea también* HDL)
Tecnología transversal (74CBT), 531
de bajo voltaje (74CBLTLV), 531
Temporizador 555 utilizado como un multivibrador astable, 261-263
Teoremas
Booleanos, 76-80
de DeMorgan, 80-83
multivariables, 77-78
TI, interruptor de señal (interruptor TS), 531
Tiempo
de acceso
definición, 788
ROM, 799
de adquisición, circuitos de muestreo y retención, 762
de conversión, ADC, 744, 750-751
de estabilización de un DAC, 735
de establecimiento (t_S), 223-224, 238
de retención (t_H), 223-224, 238, 248
Tipos de computadoras, 20-21
controlador embebido, 21
dedicadas, 21
microcomputadora, 20
microcontrolador, 21
microprocesador, 20
Tipos de LCDs, 589-590
Tótem, circuito de salida en forma de, 501
Tradicional o IEEE/ANSI, 96
Transductor, 720
Transferencia asíncrona, 245
Transición con pendiente negativa (NGT), 222
Transiciones de estado, diagrama de, 252-253
Transistor de película delgada (TFT), LCD, 590
Transmisión, compuerta de, CMOS, 546-548
Transparente, latch (latch D), 232-233. (*vea también* Flip-flops)
Triestado
búferes, 539-540
bus de datos, 540
CIs, 540
registros (74ALS173/HC173), 629-631
conectados al bus de datos, 632
salidas, 538-541
TTL, 538-541
búferes, 539-540
CIs, 540
ventajas de, 538-539
TTL rápido (74F), 508
TTL, familia lógica, 155-156, 498-502
acción
de drenado de corriente, TTL, 500
de elevación activa, 501
características, 506-509
de las series, 506-509
carga, 509-514
circuito de salida en forma de tótem, 498, 501
circuito INVERSOR, 155
clasificaciones
actuales, 512
máximas de voltaje, 504-505
comparación de características de las series, 508
compuerta
NAND básica, 499
NOR básica, 502
definición, 16
disipación de energía, 505
energía, 157
entradas
conectadas entre sí, 515
sin conectar (flotantes), 157-158, 514
sin utilizar, 514-515

estándar, serie 74, 506
factor de carga, 509-514
hojas de datos, 502-506
interfase con CMOS, 550
niveles de voltaje, 503-505
operación del circuito
en estado ALTO, 500
en estado BAJO, 498-500
otras características, 514-518
polarización de entradas en BAJO, 516
rango(s)
de temperatura, 503
de voltaje de niveles lógicos, 157
resumen, 502
salidas de colector abierto, 533-538
Schottky
de baja potencia, serie 74LS (LS-TTL), 506-507
serie 74S, 506
serie ALS, 156
serie AS, 156
serie LS, 156
serie rápida (74F), 508
serie S, 156
subfamilias, 156, 506-509
tiempos de propagación, 505
tierra, 157
transientes de corriente, TTL, 516-517
triestado, 538-541
voltaje de alimentación (energía), 157, 503
voltajes
de entrada, 526
de salida, 526

U

Ultra gran escala, integración de (ULSI), 154-155
Una sola vez programable, ROM (OTP), 803, 873
Unidad aritmética-lógica (ALU), 20, 317-318, 331-335
partes funcionales de, 318
Unidad central de proceso (CPU), 20 (*vea también* Microprocesador)
Unipolares, CIs digitales, 155-156. *Vea también* CMOS, familia lógica
Universales, programadores, 172
Universalidad de compuertas NAND y NOR, 83-86
Uso de funciones de la biblioteca TTL con ALTERA, 337-338
Utilidad de hexadecimal y octal, 32
UV, luz, EPROMs, 804

V

Valor posicional, sistema de, 10
Vectores de prueba, 175
Ventajas de las técnicas digitales, 6
VERSA Eurocard, Módulo (74VME), 532
VHDL (lenguaje de descripción de hardware para circuitos integrados de muy alta velocidad), 98-99, 410-411
AND, 342
ARCHITECTURE, 104, 179, 420, 424
archivo de diseño, 183
BEGIN, 104, 408, 411
bibliotecas, 180
de módulos parametrizados, 347-348
BIT, 104, 180, 274
BIT_VECTOR, declaraciones, 179-181, 184, 339, 411, 420
BUFFER, 420
CASE, 190-192, 407-408, 419, 428, 432, 435, 458-459, 691
ciclo FOR, 347
iterativo, 347
codificador, 647-648
comentarios, 653
comparador, 653
de magnitud, 653
COMPONENT, 273-274, 420, 425, 432, 708-709
biblioteca, 273-275
circuitos en HDL con varios, 277-280
declaración, 274
representación gráfica mediante el uso de, 274
CONSTANT, 345
Contador
BCD MOD-100, 424
con todas las características, 414-415
de anillo, 460-461
de rizo ascendente (MOD-8), 278-279
MOD-5, 408
MOD-10, 700-701, 708
MOD-60, 708
contador MOD-6, 699-708
símbolos de bloque gráficos, 705
simulación, 700
contador MOD-8, 682
simulación, 682
contador MOD-12, 703-705
símbolos de bloque gráficos, 705
simulación, 705
contadores BCD en cascada, 423-425
controlador de motor paso a paso, 685
prueba de simulación, 686
controlador de semáforo, 432-435
convertidor, 655
de código, 655
de código BCD a binario, 655
decodificación del contador MOD-5, 419
decodificador(es), 641
controlador, 643-644
secuencia de paso completo, 683
demultiplexores, 650-651
descripción
Booleana mediante el uso de, 104
del comportamiento de un contador en, 410
descripciones de estado en, 407-408
DOWNTO, 339, 342 691
elementos esenciales en, 104
ELSIF, 187-190
END, 104, 108
ENTITY, 104, 179, 183, 278, 341, 407-408, 419-420 423-424
EVENT, 276, 278, 408
flip-flops, 275-277
contador de rizo MOD-8, 278-279
simulación, 276
flip-flop JK, 274-275
simulación, 276
IF/THEN/ELSE 187, 428, 435, 644, 653
IN, 179
Instrucción
de asignación concurrente, 104
de asignación de señal condicional, 647
GENERATE, 346-348
INTEGER, 180, 187, 411, 415, 643-644
integración de módulos, 708-710
latch D, 271
latch NAND, 271
LIBRARY, 647-648
componentes, 273-275
lista de sensibilidad, 187
LPMs, 348
macrofunciones, 180
máquina de estado simple, 428-429
megafunciones, 348
monoestable (one-shot) redisparable, 463
monoestables (one-shots), 462-464
simulación, 464, 467
multiplexores, 650-651
objetos, 180

OR, 342
PACKAGE, 345
PORT, 104
  MAP, 275, 280, 420, 425, 432, 710
PROCESS, 187, 275-276, 278, 407-408, 411, 414-415, 419, 435, 459, 466, 643-644, 653, 691, 703-704
proyecto de codificación de teclado numérico (HDL), 687-693
  simulación, 693
  solución, 691-693
proyecto
  de contador de frecuencia, 710-714
  de controlador de motor paso a paso, 679-686
  de reloj digital, 693-710
RANGE, 187, 411
redisparable, monoestable disparado por flanco, 466-467
registro
  PISO, 456
  SISO, 454-455
reloj completo, 709
SELECT, 182
señales
intermedias en, 117
  locales, 106-107
SIGNAL, 106, 184, 276, 279, 339, 407-408, 643-644, 704
simulación de contador con todas las características, 415
STD_LOGIC, 180, 274
  valores, 181
STD_LOGIC_VECTOR, 180
sumador, 343, 347
  completo de un solo bit, 347
  de cuatro bits, 342, 343
  sumador/restador, 345-346
  tablas de verdad, 182-184
    asignaciones de señal selecta, 182-184
    concatenación, 182-183
  tipo enumerado, 428
  tipos de datos comunes, 180
  TYPE, 428
  VARIABLE, 275-276, 407-408, 466, 643-644, 703-704
  WHEN, 641, 647-648
  WITH, 182
Visualización de las etapas del contador, 372
VLSI (integración a muy gran escala), 154-155, 489
Volátil, memoria, 788
Voltaje
  a ADC de frecuencia, 758
  comparadores, 554-556
  de umbral con pendiente negativa (V_T), 256
  mixto, interfaz de, 553-554
    salidas de alto voltaje que controlan cargas de bajo voltaje, 554
    salidas de bajo voltaje que controlan cargas de alto voltaje, 553
    traductor de nivel de voltaje, 553
  niveles inválidos de, 494-495
  oscilador controlado por, lineal (VCO), 758
  parámetros para CIs digitales, 490-491
  traductor de nivel de, 553

W

Control de onda, secuencia de (motor paso a paso en HDL), 680

TEOREMAS BOOLEANOS

1. $x \cdot 0 = 0$
2. $x \cdot 1 = x$
3. $x \cdot x = x$
4. $x \cdot \bar{x} = 0$
5. $x + 0 = x$
6. $x + 1 = 1$
7. $x + x = x$
8. $x + \bar{x} = 1$
9. $x + y = y + x$
10. $x \cdot y = y \cdot x$
11. $x + (y + z) = (x + y) + z = x + y + z$
12. $x(yz) = (xy)z = xyz$

13a. $x(y + z) = xy + xz$

13b. $(w + x)(y + z) = wy + xy + wz + xz$

14. $x + xy = x$

15a. $x + \bar{x}y = x + y$

15b. $\bar{x} + xy = \bar{x} + y$

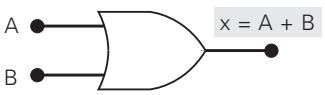
16. $\overline{x + y} = \bar{x}\,\bar{y}$

17. $\overline{xy} = \bar{x} + \bar{y}$

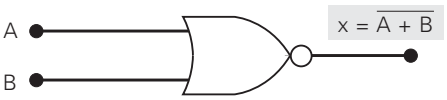
TABLAS DE VERDAD DE LAS COMPUERTAS LÓGICAS

A	B	OR $A + B$	NOR $\overline{A + B}$	AND $A \cdot B$	NAND $\overline{A \cdot B}$	XOR $A \oplus B$	XNOR $\overline{A \oplus B}$
0	0	0	1	0	1	0	1
0	1	1	0	0	1	1	0
1	0	1	0	0	1	1	0
1	1	1	0	1	0	0	1

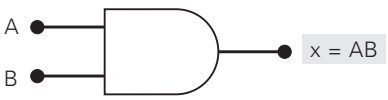
SÍMBOLOS DE COMPUERTAS LÓGICAS



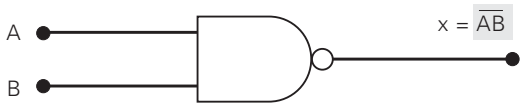
Compuerta OR



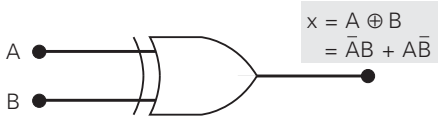
Compuerta NOR



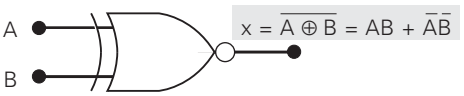
Compuerta AND



Compuerta NAND



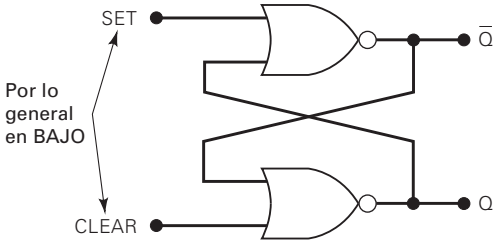
XOR

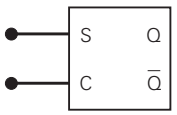


XNOR

FLIP-FLOPS

Latch NOR

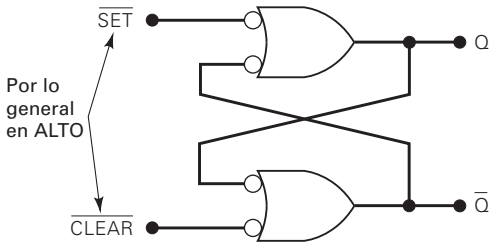


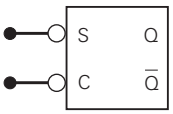


(Símbolo alternativo)

S	C	Q
0	0	Sin cambio
1	0	Q = 1
0	1	Q = 0
1	1	Inválido

Latch NAND

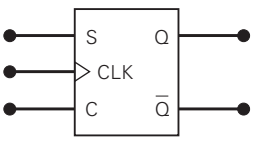




(Símbolo alternativo)

S	C	Q
0	0	Inválido
1	0	Q = 0
0	1	Q = 1
1	1	Sin cambio

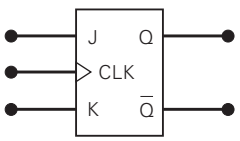
Sincronizado por reloj en S-C



S	C	CLK	Q
0	0	↑	Q_0 Sin cambio
1	0	↑	1
0	1	↑	0
1	1	↑	Ambiguo

↓ de CLK no tiene efecto sobre Q

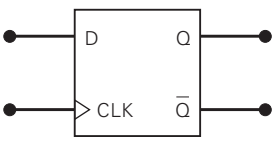
Sincronizado por reloj en J-K



J	K	CLK	Q
0	0	↑	Q_0 Sin cambio
1	0	↑	1
0	1	↑	0
1	1	↑	$\overline{Q_0}$ (conmuta)

↓ de CLK no tiene efecto sobre Q

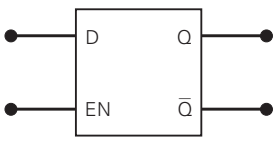
Sincronizado por reloj en D



D	CLK	Q
0	↑	0
1	↑	1

↓ de CLK no tiene efecto sobre Q

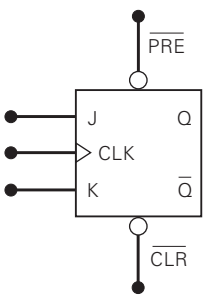
Latch D



EN	D	Q*
0	X	Sin cambio
1	0	0
1	1	1

*Q sigue la entrada D cuando EN está en ALTO

Entradas asíncronas



$\overline{PRE}$	$\overline{CLR}$	Q*
1	1	Sin efecto; el FF puede responder a J, K y CLK
1	0	Q = 0 sin importar J, K, CLK
0	1	Q = 1 1 sin importar J, K, CLK
0	0	Ambigua (no se utiliza)

*CLK puede encontrarse en cualquier estado