Module 6

Test Benches

Test Bench Information

- A test bench is a model used to exercise and verify that your VHDL model is working as you expect
- Most of the debugging will occur as you are testing your VHDL design against the test bench
- It is most ideal to have a separate party develop the test bench

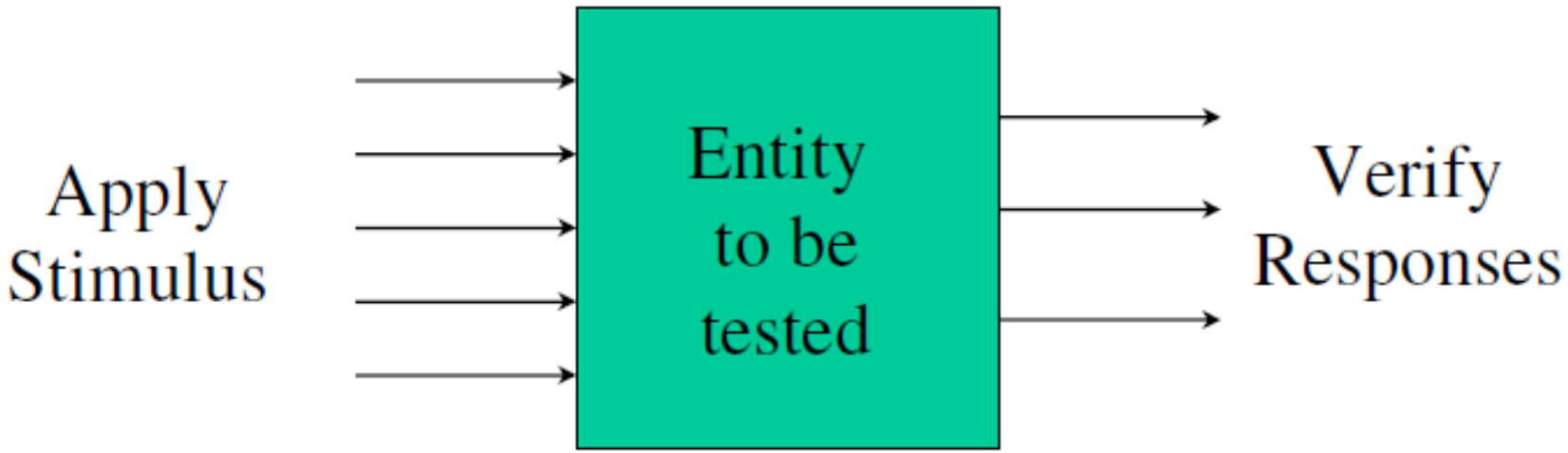
Test Benches

- Test Benches consist of:
  - Entity
    - Contains no ports or generics
  - Architecture
    - Declares, instantiates, and wires together the driver model and the model under test
    - The driver model provides stimulus and verifies the response of the model under test

Test Bench Outline

```vhdl
-- Example Test bench outline
-- Libraries
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;

-- libraries that are only supported
-- test benches (these libraries will not
-- synthesise)
use std.textio.all;
use ieee.std_logic_textio.all;

entity [test_bench_entity] is
end;

architecture test of [test_bench_entity] is
-- Instantiate component of the VHDL entity your
-- testing

component [name_of_entity_tested]
-- Instantiate component of the design
-- your testing here
end component;

-- (To Do) create signals used in your test bench

begin
    -- Sometimes this is called UUT (Unit Under Test)
    dev_to_test:    [name_of_entity_tested] port map (

    -- Define inputs and outputs here

    expected_proc : process(inputs)
        begin
            -- Place code here that gives you the results you want
            -- For example for every input you will specify what the
            -- output should be
        end process expected_proc;

        stimulus : process
            begin
                -- Place your code that goes through and applies
                -- a stimulus to the dev_to_test also known as
                -- UUT (Unit Under Test)
    end process stimulus;
end test;
```

Test Bench Block Diagram



Example – Universal Shift Register

```
1  -- test_USR
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4  use IEEE.numeric_std.all;
5
6  use std.textio.all;
7  use ieee.std_logic_textio.all;
8
9  use work.sim_mem_init.all;
10
11 entity test_USR is
12 end;
13
14 architecture test of test_USR is
15
16 component USR
17 generic (
18     data_width      : integer := 8);
19 port (
20     A               : out std_logic_vector(data_width - 1 downto 0);
21     I               : in std_logic_vector(data_width - 1 downto 0);
22     S               : in std_logic;
23     reset           : in std_logic;
24     clk             : in std_logic);
25 end component;
```

Example – Universal Shift Register

```
27  constant data_width : integer := 8;
28  signal data_in      : std_logic_vector(data_width - 1 downto 0);
29  signal data_out     : std_logic_vector(data_width - 1 downto 0);
30  signal expected     : std_logic_vector(data_width - 1 downto 0);
31  signal sel          : std_logic_vector(1 downto 0);
32  signal clk          : std_logic := '1';
33  signal reset        : std_logic := '1';
34
35  constant in_fname   : string := "input.csv";
36  constant out_fname  : string := "output.csv";
37  file input_file     : text;
38  file output_file    : text;
39
40  begin
41
42      dev_to_test: USR
43          generic map(data_width)
44          port map(data_out, data_in, sel, reset, clk);
45
46      stimulus: process
47
48      variable input_line     : line;
49      variable WriteBuf       : line;
50      variable in_char        : character;
51      variable in_slv         : std_logic_vector(7 downto 0);
52      variable out_slv        : std_logic_vector(7 downto 0);
53
54      variable ErrCnt         : integer := 0;
```

Example – Universal Shift Register

```
56      begin
57          file_open(input_file, in_fname, read_mode);
58          file_open(output_file, out_fname, read_mode);
59
60          while not(endfile(input_file)) loop
61
62              readline(input_file, input_line);
63
64              for i in 0 to 6 loop
65
66                  read(input_line, in_char);
67                  in_slv := std_logic_vector(to_unsigned(character'pos(in_char), 8));
68
69                  -- j is the column number of input file
70                  if(i = 3) then
71                      data_in(7 downto 4) <= ASCII_to_hex(in_slv);
72                  elsif(i = 4) then
73                      data_in(3 downto 0) <= ASCII_to_hex(in_slv);
74                  elsif(i = 6) then
75                      sel <= in_slv(1 downto 0);
76                  end if;
77              end loop;
78
79              readline(output_file, input_line);
80
81              clk <= '0';
82              wait for 10 ns;
83
```

Example – Universal Shift Register

```
84          for i in 0 to 4 loop
85              read(input_line, in_char);
86              out_slv := std_logic_vector(to_unsigned(character'pos(in_char), 8));
87
88              -- j is the column number of the output file
89              if(i = 3) then
90                  expected(7 downto 4) <= ASCII_to_hex(out_slv);
91              elsif(i = 4) then
92                  expected(3 downto 0) <= ASCII_to_hex(out_slv);
93              end if;
94          end loop;
95
96          clk <= '1';
97          wait for 10 ns;
98
99          if (expected /= data_out) then
100             write(WriteBuf, string'("ERROR: USR failed"));
101             write(WriteBuf, string'("expected = "));
102             write(WriteBuf, expected);
103             write(WriteBuf, string'(", data_out = "));
104             write(WriteBuf, data_out);
105
106             writeline(Output, WriteBuf);
107             ErrCnt := ErrCnt + 1;
108         end if;
109     end loop;
```

Example – Universal Shift Register

```
111             file_close(input_file);
112             file_close(output_file);
113
114             if (ErrCnt = 0) then
115                 report "SUCCESS!!! USR Test Completed";
116             else
117                 report "The USR device is broken" severity warning;
118             end if;
119
120         end process stimulus;
121     end test;
```

Testing Purposes Only

- Test benches are used for testing purposes only
- A test bench **cannot** be synthesized and placed onto an FPGA or CPLD
- A test bench can reference from or pull in values from outside files such as text files

Developing Test Benches

- Test benches can be coded completely from scratch
- Xilinx ISE tool set will get you a Test Bench skeleton to work from
- You can code the test benches using Notepad++, Quartus, ModelSIM, or Xilinx ISE
- Look at the test benches located in the reference section for some examples of different styles of test benches

Summary

- Test benches are used to test a VHDL design
- Tools such as Altera's ModelSIM and Xilinx's ISE allow you to run your test bench
- Test benches **cannot** be placed onto a FPGA or CPLD
- Test benches can be written using several different tool sets