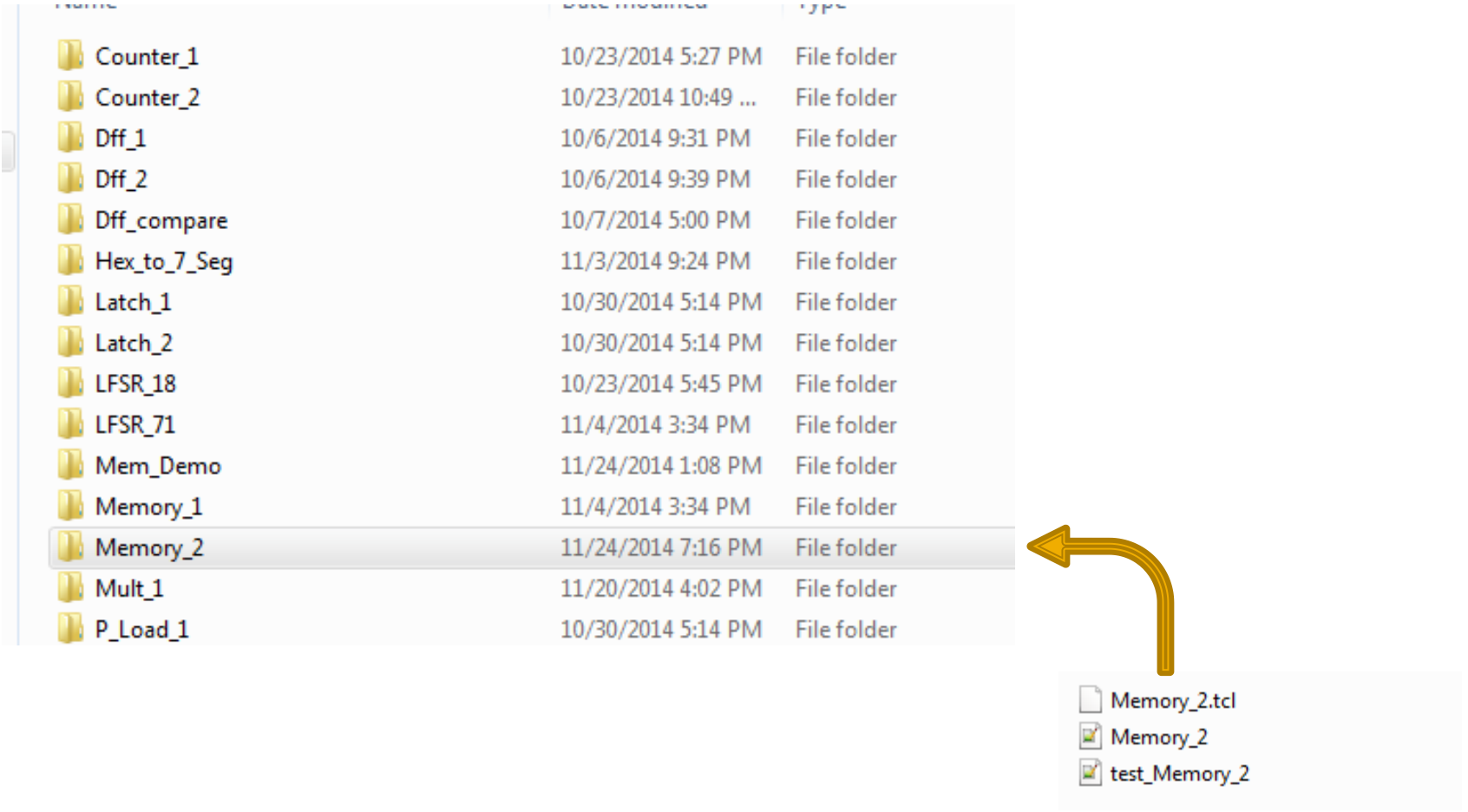
Module 5

VHDL Coding Styles

Organization



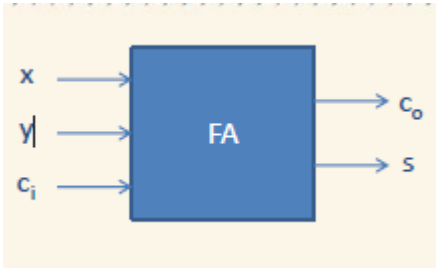
Types of Coding styles

- Structural
  - Uses Component Instantiation
- Behavioral
  - Uses Process Statements
- Dataflow
  - Uses Boolean logic and concurrent signal assignments
- Mixed
  - Uses all three

Example - Full Adder

TRUTH TABLE (Full Adder - 1bit)

X	Y	Cin	Cout	S
0	0	0	**0**	**0**
0	0	1	**0**	**1**
0	1	0	**0**	**1**
0	1	1	**1**	**0**
1	0	0	**0**	**1**
1	0	1	**1**	**0**
1	1	0	**1**	**0**
1	1	1	**1**	**1**



Full Adder Behavioral

- A behavioral style architecture does not imply a specific hardware implementation.
- A behavioral style architecture simply tells the circuit how to behave.
- By using a behavioral style it allows the compiler to determine the specific hardware implementation.

Full Adder Behavioral

```vhdl
----------------------------------------------------------
-- Full Adder
-- Created by:    Jordan Christman
----------------------------------------------------------
-- Library's
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Entity declaration
entity Full_Adder is
    port (
        S       : out std_logic;
        C_out   : out std_logic;
        x       : in std_logic;
        y       : in std_logic;
        C_in    : in std_logic);
end Full_Adder;

-- Behavioural architecture (Easier to read)
architecture behavior of Full_Adder is

signal inputs   : std_logic_vector(2 downto 0);
signal outputs  : std_logic_vector(1 downto 0);

begin

-- Combine inputs & outputs into std_logic_vector
inputs <= C_in & x & y;
C_out <= outputs(1);
S <= outputs(0);

adder_proc : process(inputs)
begin
    case inputs is
        when "000" =>
            outputs <= "00";
        when "001" =>
            outputs <= "01";
        when "010" =>
            outputs <= "01";
        when "011" =>
            outputs <= "10";
        when "100" =>
            outputs <= "01";
        when "101" =>
            outputs <= "10";
        when "110" =>
            outputs <= "10";
        when "111" =>
            outputs <= "11";
        when others =>
            outputs <= (others => 'X');
    end case;
end process adder_proc;
end behavior;
```

Full Adder Structural

```
16  architecture structure of adder_1 is
17
18      component full_adder_1 is
19          Port (
20              s       : out std_logic;
21              c_out   : out std_logic;
22              x       : in std_logic;
23              y       : in std_logic;
24              c_in    : in std_logic
25          );
26      end component;
27
28      signal carry    : std_logic_vector(bit_depth-1 downto 0);
29      begin
30
31      sum_0 : full_adder_1 port map(s(0), carry(0), x(0), y(0), c_in);
32
33      full_adders:
34      for i in 1 to bit_depth-1 generate
35          sum_x : full_adder_1 port map(s(i), carry(i), x(i), y(i), carry(i-1));
36      end generate full_adders;
37
38      c_out <= carry(bit_depth-1);
39
40  end structure;
```

Full Adder Dataflow

- Data flow architecture tells the compiler how to implement the VHDL file by specifying the hardware at a gate level.
- Data Flow architecture describes how data flows through a system

```
14  architecture dataflow of Full_Adder_1 is
15      begin
16
17      C_out <= (C_in and (x xor y)) or (x and y);
18      S <= x xor y xor C_in;
19
20  end dataflow;
```

State Machines

- A state machine is a somewhat complex counter that can count out of order based on other control signals.
- The count of a state machine is usually referred to as the state.
- See an example of a state machine used in a change dispensing system, located in the references section

Summary

- When design VHDL files make sure to stay organized
- There are multiple different ways you can design a circuits architecture
- When you are coding a VHDL architecture make sure that you try to make it as readable as possible