CSCE 230 – Lab 7: Mux and Finite State Machine (FSM)

Due: 11:59PM, Oct 17 (Tuesday) (two-week lab)

Objectives

- Altera Quartus
  - Learn how to connect the logic circuit signals with the DE10 Lite board devices
  - Learn how to burn FPGA to implement logic circuits
  - Learn how to add debugging signals to VHDL design and simulation script
- VHDL
  - Learn how to use the concurrent component statement
  - Learn basic combinational and sequential circuits
- Learn how to build a finite state machine (FSM) using VHDL.

Useful References on Canvas

- Lecture notes for Appendix A and VHDL
- Altera DE-10 Lite board
  - DE-10 Lite board user manual (pin assignment information)
- Altera Software
  - Quartus Introduction
  - ModelSim Introduction
  - ModelSim Commands
- VHDL
  - VHDL tutorial
  - VHDL notes

Lab Task 1: Multiplexer

In this task, we will use a multiplexer as an example to study the concurrent component statements and study how to load a VHDL design to a DE-10 Lite board.

Step 1 – Creating a Quartus project

- Please create a Quartus project by following the same steps as described in the last lab.
  - Top-level entity is "**mux2input4bit"**
  - Device family is "Max 10"
  - Available device is "10M50DAF484C7G"

Step 2 – Writing two VHDL design files

- VHDL file 1
  - Create a new VHDL file "mux2input1bit.vhd" using the following entity and implementation.

- When saving this file, make sure to check "Add file to current project".

```
library ieee;
use ieee.std_logic_1164.all;

entity mux2input1bit is
      port(
            s      : in  std_logic;
            a,b    : in  std_logic;
            output : out std_logic);
end mux2input1bit;

architecture implementation of mux2input1bit is
 signal term1, term2 : std_logic;
begin
      term1 <= not s and a;
      term2 <= s and b;
      output <= term1 or term2;
end implementation;
```

➢ VHDL file 2

- Create a new VHDL file "mux2input4bit.vhd" using the following entity definition
- **Please write an implementation for this circuit using concurrent component statements**. Your implementation should have exactly four component statements.
- When saving this file, make sure to check "Add file to current project".

```
library ieee;
use ieee.std_logic_1164.all;

entity mux2input4bit is
      port(
            s      : in  std_logic;
            a,b    : in  std_logic_vector(3 downto 0);
            output : out std_logic_vector(3 downto 0));
end mux2input4bit;

architecture implementation of mux2input4bit is
  component mux2input1bit is
      port(
            s      : in  std_logic;
            a,b    : in  std_logic;
            output : out std_logic);
  end component;
begin

   -- Exactly four component statements below


end implementation;
```

Step 3 – Compiling your project:

- Please compile your project, fix all compilation errors (if any), and ignore the warnings.

Step 4 – Assigning pins

- Now we assign the input and output of multiplexer mux2input4bit to the devices on the DE-10 Lite board according to the following table.
  - Select menu "Assignments", then "Pin Planner" to open the pin planner window.
  - In the bottom table of the pin planner window, click on the empty cells of the "Location" column to type the following pin location for each node in the "Node Name" column.
  - **NOTE: You should also appropriately set the I/O Standard for each pin.**
  - Once you are done, click menu "File", select "close" to close the pin planner window, and then **recompile** your project.

Node Name	Location	Corresponding devices on DE-10 Lite	I/O Standard
S	PIN_A7	Push button 1 (KEY1)	3.3 v SCHMITT TRIGGER
a[3]	PIN_A14	Slider Switch 7 (SW7)	3.3-V LVTTL
a[2]	PIN_A13	Slider Switch 6 (SW6)	3.3-V LVTTL
a[1]	PIN_B12	Slider Switch 5 (SW5)	3.3-V LVTTL
a[0]	PIN_A12	Slider Switch 4 (SW4)	3.3-V LVTTL
b[3]	PIN_C12	Slider Switch 3 (SW3)	3.3-V LVTTL
b[2]	PIN_D12	Slider Switch 2 (SW2)	3.3-V LVTTL
b[1]	PIN_C11	Slider Switch 1 (SW1)	3.3-V LVTTL
b[0]	PIN_C10	Slider Switch 0 (SW0)	3.3-V LVTTL
output[3]	PIN_B10	LED Red 3	3.3-V LVTTL
output[2]	PIN_A10	LED Red 2	3.3-V LVTTL
output[1]	PIN_A9	LED Red 1	3.3-V LVTTL
output[0]	PIN_A8	LED Red 0	3.3-V LVTTL

- The above pin assignment
  - connects slider switches SW7, SW6, SW5, and SW4 to input a of mux2input4bit
  - connects slider switches SW3, SW2, SW1, and SW0 to input b of mux2input4bit
  - connects pushbutton KEY1 to input s of mux2input4bit
  - and connects four red LEDs to output of mux2input4bit.
- If you are interested, the complete pin assignment information can be found on pages 25, 26, and 27 of the Altera DE-10 Lite Board User Manual, which is available on Canvas.

Step 5 – Burning FPGA

- Now we can load our design onto the DE-10 Lite board to see the design in action.
- Select menu "Tools", then click "Programmer" to open the programmer window.

- The design file "mux2input4bit.sof" should be listed in the programmer window.
- Click the "Start" button to burn the FPGA chip on your DE-10 Lite board to implement mux2input4bit.
- If the "Start" button is not enabled (i.e., grey color), please check "Hardware Setup" to select "USB-Blaster" from the list of currently selected hardware.
- Wait for a few seconds until the "progress" bar on the top-right corner shows "100% (Successful)".

➢ If you are interested, more detailed information about burning FPGA can be found in Section 9 of the Altera Quartus Introduction, which is available on Canvas.

Step 6 – Testing on DE1 Board

➢ Now we test your mux2input4bit on the DE-10 Lite board

➢ Set the slider switches to the following positions. That is, mux2input4bit input port signal a="1111" and signal b="1100".

- SW7: top
- SW6: top
- SW5: top
- SW4: top
- SW3: top
- SW2: top
- SW1: bottom
- SW0: bottom

➢ Correctness check:

- When you are pressing press KEY1, all four red LED lights LEDR3, LEDR2, LEDR1, and LEDR0 should be turned on
- When you do not press KEY1, LEDR3 and LEDR2 should be turned on, and LEDR1 and LEDR0 should be turned off.

➢ Canvas questions:

- **Question 1 (Difficult question)**: What is the value (exactly 1 bit) of input port signal s of mux2input4bit, when you are pressing KEY1? This question can be answered by just thinking about the results of the above correctness check.
- **Questions 2**: Please upload a photo of your DE1 board to clearly show the red LEDs and the slider switches, when the slider switches are in the positions specified above and when you are pressing KEY1.
- **Question 3**: Please copy-and-paste the final VHDL design file for mux2input4bit with an updated code header.

Lab Task 2: Finite State Machine

In this lab task, we will design a mod-8 counter, which increases by 1 or 2 depending on input signal *increment* (SW0) at each rising edge of input signal *clock* (KEY1), and displays the current state on the first seven-segment display (HEX0). In addition, the counter is reset to zero if input signal *reset* (SW1) becomes 1.

- State
  - 3-bit binary encoding of decimal numbers 0, 1, 2, 3, 4, 5, 6, 7
- Input
  - 1-bit *increment*:
    - 0: the counter increases the current state by 1
    - 1: the counter increases the current state by 2
  - 1-bit *reset*
    - 0: does not change the current state
    - 1: resets the current state to 0 immediately.
  - 1-bit *clock*
- Output:
  - 7 bits to turn on/off the seven segments of HEX0 according to the current state
  - Note: **Different from turning on a segment by setting it to 1 using assembly instructions in Monitor, we turn on a segment by setting it to 0 using VHDL in Quartus. For example, to display 0, we set HEX0 to 1000000 in VHDL.**
- State transition table:

current state n	next state		output
	increment=0	increment=1	
000	001	010	1000000
001	010	011	1111001
010	011	100	0100100
011	100	101	0110000
100	101	110	0011001
101	110	111	0010010
110	111	000	0000010
111	000	001	1111000

Step 1 – VHDL design

- Please create a new Quartus project.
  - Top-level entity is “counter”
  - Device family is “Max 10”
  - Available device is “10M50DAF484C7G”
- Please write a VHDL design file to implement the above FSM.
- You must use the VHDL template below, which has three processes corresponding to the current state logic, the next state logic, and the output logic, respectively.
- The current state logic is a sequential circuit triggered by the rising edge of clock.
- Both the next state logic and the output logic are combinational circuits.

```
library ieee;
use ieee.std_logic_1164.all;

entity counter is
      port(
            clock     : in  std_logic;
            reset     : in  std_logic;
            increment : in  std_logic;
            output    : out std_logic_vector(6 downto 0);
            debug_current_state : out std_logic_vector(2 downto 0));
end counter;

architecture implementation of counter is
      signal current_state: std_logic_vector(2 downto 0);
      signal next_state   : std_logic_vector(2 downto 0);
begin
      debug_current_state <= current_state;

      -- current state logic defined below
      process(clock, reset)
      begin

      ....

      end process;

      -- next state logic defined below
      process(current_state, increment)
      begin

      ....

      end process;

      -- Output logic defined below
      process(current_state)
      begin

      ....

      end process;

end implementation;
```

Step 2 – DE-10 Lite board testing

- Please do the pin assignment according to the following table.
  - **NOTE: It is very important to appropriately set the I/O Standard or it won't work!**

Node Name	Location	Corresponding devices on DE-10 Lite Board	I/O Standard
clock	PIN_A7	Push button 1 (KEY1)	3.3V Schmitt Trigger
reset	PIN_C11	Slider Switch 1 (SW1)	3.3-V LVTTL

increment	PIN_C10	Slider Switch 0 (SW0)	3.3-V LVTTL
debug_current_state[2]	PIN_A10	Red LED light LEDR2	3.3-V LVTTL
debug_current_state[1]	PIN_A9	Red LED light LEDR1	3.3-V LVTTL
debug_current_state[0]	PIN_A8	Red LED light LEDR0	3.3-V LVTTL
output[6]	PIN_C17	Segment 6 of HEX0	3.3-V LVTTL
output[5]	PIN_D17	Segment 5 of HEX0	3.3-V LVTTL
output[4]	PIN_E16	Segment 4 of HEX0	3.3-V LVTTL
output[3]	PIN_C16	Segment 3 of HEX0	3.3-V LVTTL
output[2]	PIN_C15	Segment 2 of HEX0	3.3-V LVTTL
output[1]	PIN_E15	Segment 1 of HEX0	3.3-V LVTTL
output[0]	PIN_C14	Segment 0 of HEX0	3.3-V LVTTL

- Re-compile your project, and then burn FPGA on DE-10 Lite board to implement your circuit. Maintain a copy of your counter.sof (will be in the Output folder) file to submit to Canvas.
- Please test your circuit according to the following steps
  - Set *reset* to 1 by moving SW1 to the top position. HEX0 should now display 0.
  - Set *reset* back to 0 by moving SW1 back to the bottom position.
  - Set *increment* to 0 by moving SW0 to the bottom position.
  - Push KEY1 and then release. HEX0 should display 1.
  - Push KEY1 and then release. HEX0 should display 2.
  - Push KEY1 and then release. HEX0 should display 3.
  - Set *increment* to 1 by moving SW0 to the top position.
  - Push KEY1 and then release. HEX0 should display 5.
  - Push KEY1 and then release. HEX0 should display 7. At this step, please take and upload a photo of your DE1 board clearly showing the slider switches and HEX0.
  - Push KEY1 and then release. HEX0 should display 1.
  - Set *reset* to 1 by moving SW1 to the top position. HEX0 should display 0.
  - Push KEY1 and then release. HEX0 should still display 0.

Step 3 – ModelSim simulation

- Please create a simulation script file using the following script.

```
#the name must be the same as your vhdl filename
vsim counter

#view waveform
view wave

#view input signals
add wave clock
add wave reset
add wave increment

#view output signals
add wave debug_current_state
add wave output

#set input signal values
force clock 1 0, 0 5 -repeat 10
force reset 1 0, 0 5, 1 80, 0 85, 1 140
force increment 0 0, 1 90

#run simulation
run 150
```

- Run ModelSim simulation
- Answer the questions on Canvas, upload a screenshot of the waveform, and upload a copy of the counter.sof file to Canvas.