CSCE 230 – Lab 8: Finite State Machine

Due: 11:59PM, Oct 31 (Tuesday) (two-week lab)

Objectives

- Learn how to design and build a finite state machine (FSM) using VHDL.

Useful References on Canvas

- Lecture notes for Appendix A and VHDL

Overall design flow:[1]

Big Picture: In this lab, you will design, write, test, and implement in hardware a finite state machine to simulate the taillights of a 1965 Ford Thunderbird.

Background

The 1965 Ford Thunderbird has three lights on each side that operate in sequence to indicate the direction of a turn. Figure 1 shows the tail lights and Figure 2 shows the flashing sequence for (a) left turns and (b) right turns.

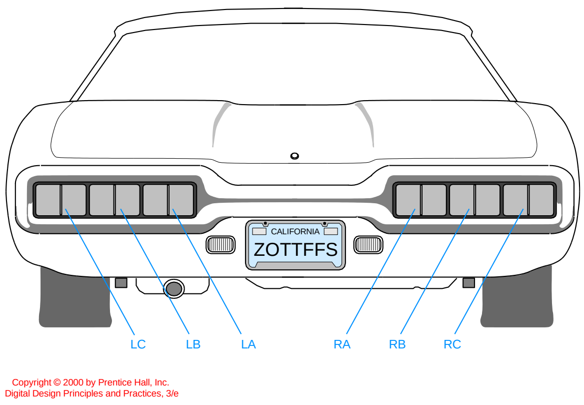

CALIFORNIA
ZOTTFFS
LC
LB
LA
RA
RB
RC
Copyright © 2000 by Prentice Hall, Inc.
Digital Design Principles and Practices, 3/e

Figure 1 – Thunderbird taillights

[1] Modeled after a lab provided with instructor notes for Digital Design and Computer Architecture, David Money Harris & Sarah L. Harris, 2nd Edition

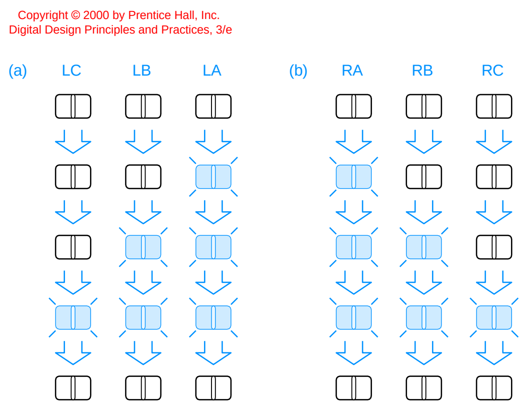

Copyright © 2000 by Prentice Hall, Inc.
Digital Design Principles and Practices, 3/e
(a)
LC
LB
LA
(b)
RA
RB
RC

Figure 2 – Flashing sequence for taillights. The lights shaded light blue are illuminated.

This lab is divided into four parts: design (Prelab), VHDL creation, simulation, and implementation. If you follow the steps of FSM design carefully and ask questions at the beginning if a part is confusing, you will save yourself a great deal of time.

Prelab - FSM Design

Project functionality description

On reset, the FSM should immediately enter a state with all lights off. When you press left, you should see LA, then LA and LB, then LA, LB, and LC, then finally all lights off again. This pattern should occur even if you release left during the sequence. If left is still down when you return to the lights off state, the pattern should repeat. The logic for the right lights is similar. You must decide what you want to do when both left and right switches are on.

Extra Credit: Modify your state machine to make blinking hazard lights when both switches are on.

Prelab Tasks

1. Create a new Quartus Project called **top_tbird**. This will create your top_tbird folder where you can store your files.
2. Sketch a state transition diagram and include it in Canvas (neatly hand drawn is fine for Prelab—**final version for Canvas must be computer-generated**)
    a. Are you designing a Moore or a Mealy machine?
3. Before you create your state tables, take a moment to think about how you want to encode the various states. You can use binary numeric coding, one-hot encoding, or even using a Gray encoding scheme. You are free to choose the scheme of your choice, but keep in mind some schemes may be easier than others when you draw the schematics or the tables or equations.
4. The next step in the design process is **to create the state transition table and the output tables**. These will also be included in Canvas (not hand drawn). On the left side of the state table, you

should have your current state and your inputs (left, right, and reset). On the right side of the table, you will fill in your next state. Make use of "don't care" conditions. If you are most comfortable drawing a state table first with names of states, and then redrawing it with the binary encoding of those states, go ahead and do so. For your output tables; don't forget that **each light is a separate output**.

5. Now that you have your tables, you can write your next state and output equations. Simplify them if possible. Include these in Canvas.
6. Draw a schematic of your design and include it in your Canvas (neatly hand drawn is fine)

In the remainder of the lab, you will be creating your FSM in VHDL, testing it using a testbench, and then implementing your design on the FPGA. Since you are designing a synchronous sequential circuit, **we will be making use of a clock signal** to drive the logic. Finally, while not required for the Prelab, you should already be thinking about what test cases you will use to verify that your design is correct.

Create your Thunderbird FSM

Create a new VHDL module file using the course VHDL template. The file should be called `thunderbird_fsm.vhd`. Implement your FSM in this file using the following entity interface:

```
entity thunderbird_fsm is
      port(
            i_clk,  i_reset : in  std_logic;
            i_left, i_right : in  std_logic;
            o_lights_L      : out std_logic_vector(2 downto 0);
            o_lights_R      : out std_logic_vector(2 downto 0)
      );
end thunderbird_fsm;
```

It is highly recommended that you do not specify state encodings, but rather allow the synthesizer to select the optimal encodings for you (i.e., use enumerated states). Note, this means the synthesized hardware is likely to be different than what you created a schematic of in the Prelab. **View the RTL schematic of the synthesized FSM and include a picture in your Canvas Submission**. Does it look as expected? Do you find anything surprising?

Verify your FSM module via simulation

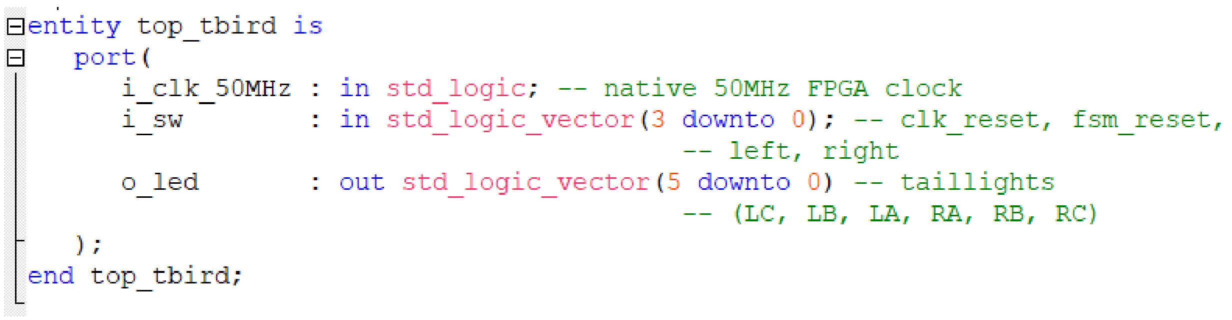
Next, write a simulation script that verifies functionality of your FSM. Make sure you carefully consider what test cases you need to run. For instance, what happens if you change an input in the middle of a blinking pattern? *Explain* the various tests you conducted in plain English and why they are important for verifying your design.

- Make sure the first test you run is to apply a **reset**. This should put your FSM in a known good state.
- Do not forget that you must create a clock input to drive the fsm. **Define the clock period as 10ns** and use this clock throughout your simulation script.

For the simulation waveform, you should include at a minimum the following signals in order: i_clk, i_reset, i_left, i_right, o_lights_L, and o_lights_R. You may also want to include other signals (e.g., signals internal to FSM such as current state) as a part of your **analysis**.

Create top-level module

Once you have verified that your FSM works correctly, use structural modeling (Section 4.3) to connect your FSM signals into a top-level VHDL design file called top_tbird.vhd. You must use the following interface for your top-level entity:



```
entity top_tbird is
    port(
        i_clk_50MHz : in std_logic; -- native 50MHz FPGA clock
        i_sw        : in std_logic_vector(3 downto 0); -- clk_reset, fsm_reset,
                                                       -- left, right
        o_led       : out std_logic_vector(5 downto 0) -- taillights
                                                       -- (LC, LB, LA, RA, RB, RC)
    );
end top_tbird;
```

A VHDL file is provided for you that includes this entity declaration. *Before* you write the architecture, neatly create a diagram (hand drawn is fine) showing your top-level entity and its internal architecture. Upload this drawing in Canvas.

Note, the Altera DE-10 Lite board provides a built-in 50-MHz clock. This is a bit too fast for our application, so you are also provided a VHDL module (clock_divider.vhd) that will take in the fast 50 MHz clock, and produce a slower clock that will make your thunderbird_fsm transitions visible to the human eye. Your top-level design must incorporate a structural connection of the thunderbird_fsm and clock_divider to produce a correct design. Hint: you will need to declare a temporary signal to "wire" the connection between the clock divider's output clock port and the FSM's input clock port.

Implement in Hardware

Based on the pin assignments you've seen from previous labs, ensure you connect the LEDs/switches to your top level design. Below are a couple samples of what should be in your pin assignments. You must add pins as needed to get your design to work properly. It may help to write the physical board locations on your top-level schematic that you previously drew.

Assigning pins

- Now we assign the input and output of top_tbird 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.
    - 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_clk_50MHz	PIN_P11	50 MHz Clock Input
i_sw[0]	PIN_C10	Slider Switch 0 (SW0)
o_led[5]	PIN_C13	LED Red 5
o_led[4]	PIN_D13	LED Red 4

- The complete pin assignment information can be found on pages 26, 17 and 28 of the Altera DE-10 Lite Board User Manual, which is available on Canvas.

Download your design onto the Altera board and test it in hardware. Provide the working product to an instructor for hardware demonstration. Upload a signed cutsheet showing verified functionalities.

Extra Credit

When both the left and right buttons are pressed, all of the lights will blink (hazard lights). To receive full credit, you must adequately test this functionality both in simulation and in hardware, and also provide explanations.

Lab 8 Cut Sheet

Name: ________________________ **Instructor:** ______________________ **Section:** ____________

Item	Points	Out of	Initials	Late Date
Final H/W demo		25		
Hazard lights bonus		5		
Total		**25**		