UNCA 320 Specific Introduction

Sometimes you just ought to use a lab from the textbook. The authors have to put a lot of time on these in order to sell their books so they usually have much better illustrate. However, you might notice that this lab was actually derived from another textbook.

This is the 4th lab in Harvey Mudd's E85, *Digital Electronics and Computer Engineering*, though it is the first lab where they create a design with SystemVerilog. In earlier labs they designed using schematics.

I have done minimal editing of the lab. I have removed sections that discuss facilities we don't have and have changed suggested file names.

I'm not sure this was completed entirely in a formal lab section. It might have been the start of a homework project.

You can work with other students in this lab.

Digital Design and Computer Architecture

Lab 4: Thunderbird Turn Signal

Introduction

In this lab, you will design a finite state machine in SystemVerilog to control the taillights of a 1965 Ford Thunderbird[1]. There are 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.

[1] This lab is derived from an example by John Wakerly from the 3rd Edition of Digital Design.

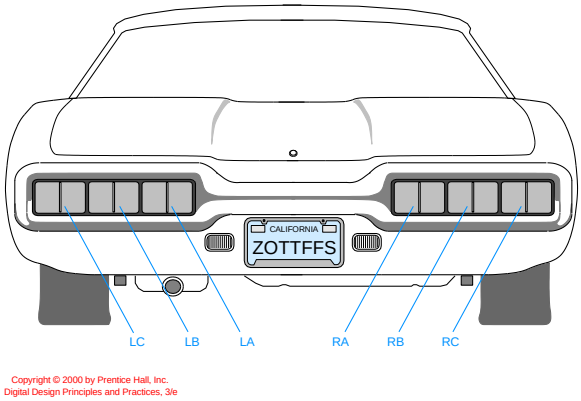

Figure 1. Thunderbird Tail Lights

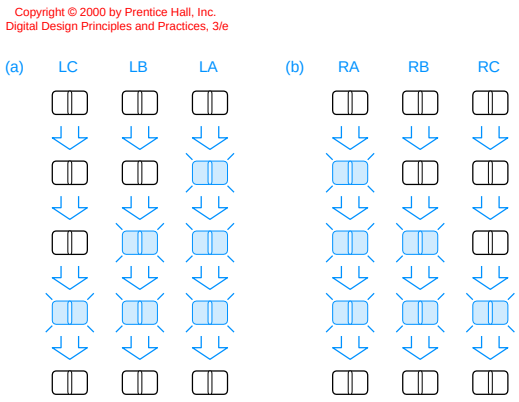

Figure 2. Flashing Sequence (shaded lights are illuminated)

This lab is divided into three parts: design, SystemVerilog entry, and simulation. 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.

1) Design

Your FSM should have the following module declaration:

```
module lab4(input logic clk,
            input logic reset,
            input logic left, right,
            output logic la, lb, lc, ra, rb, rc);
```

You may assume that `clk` runs at the desired speed (e.g. about 1 Hz).

On reset, the FSM should 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. **right** is similar.

It is up to you to decide what to do if the user makes **left** and **right** simultaneously true; make a choice to keep your design *easy*.

Sketch a state transition diagram. This is a very important part of the assignment. The statement transition diagram should only have a few states – may be 7.

2) SystemVerilog Entry

Create a new project named thunderbird, then create a new SystemVerilog HDL file and save it as thunderbird.sv. Enter Verilog code for your FSM. Follow the structures used in the textbook. You one section of Verilog code to generate the next state and another to generate the outputs.

3) Simulation

Create a testbench.sv file that convincingly demonstrates that the FSM performs all functions correctly. Simulate your FSM in ModelSim with your testbench.

You'll probably have errors in your SystemVerilog file or testbench at first. Get used to interpreting the messages from ModelSim and correct any mistakes. In fact, it's good if you have bugs in this lab because it's easier to learn debugging now than later when you are working with a larger system!