Lab #4: Finite State Machines (FSMs)

0. GOALS

In this lab, you will design a finite state machine (FSM) in SystemVerilog, simulate it using ModelSim, and then test it in hardware using your DE2-115 board.

You will complete the following steps in this lab:

1. Design an FSM in SystemVerilog
2. Simulate your FSM using a provided testbench in ModelSim
3. Test your FSM in hardware on the FPGA board using Quartus II
4. Submission Instructions

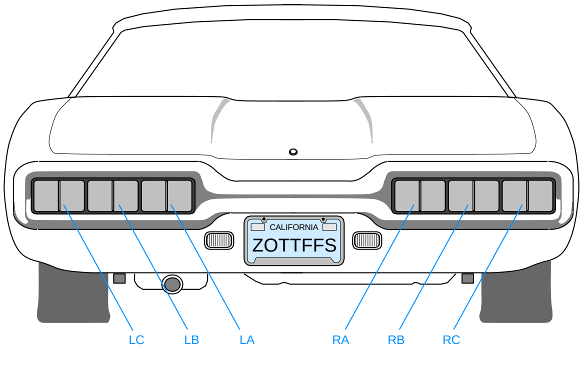
Be sure to read the "What to Turn In" (Section 4) at the end of the lab before beginning the lab.

Important: Most everyone should be currently enrolled in CpE 200. If you are not currently enrolled in CpE 200 (i.e., you took it in a prior semester), you may view the videos, lecture slides, and example SystemVerilog files on CPE 200L's Canvas page.

As always, be sure to **start and finish** the lab early so that you have enough time to work through your issues and/or have time to ask for help.

1. DESIGN AN FSM IN SYSTEMVERILOG

Design a finite state machine (FSM) 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 taillights and Figure 2 shows the flashing sequence for (a) left turns and (b) right turns.



Copyright © 2000 by Prentice Hall, Inc.
Digital Design Principles and Practices, 3/e

Figure 1. Thunderbird Tail Lights

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

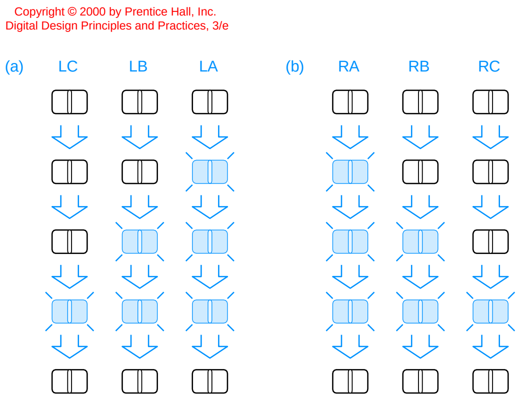

Copyright © 2000 by Prentice Hall, Inc.
Digital Design Principles and Practices, 3/e

Figure 2. Flashing Sequence (shaded lights are illuminated)

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.

Your FSM should be a Moore FSM and should have the following module declaration:

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

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

State Transition Diagram

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*. (For example, in a car, it's physically impossible to have both left and right selected at the same time when selecting for turn signals.)

Sketch a state transition diagram.

At this point, you could write state transition and output tables and then a set of next state and output equations as you've done in CpE 200 homework. Instead, you will design it in SystemVerilog and let the synthesis tool choose the gate-level implementation.

SystemVerilog Entry

Create a new SystemVerilog HDL file and save it as `thunderbird.sv`. Enter SystemVerilog code for your FSM. Remember, you can use any text editor (for example, Notepad++) to write your .sv file.

2. SIMULATE FSM WITH TESTBENCH

Use the testbench_fsm.sv file from Canvas to simulate your FSM in ModelSim. The combinational circuit in Lab 3 used a testbench with a testvector file. This time you will use the testbench itself to set the inputs and toggle the clock; then you will check the outputs manually by viewing the waveform. Open the testbench_fsm.sv file (available on Canvas) and make sure you understand what it does. This testbench toggles the inputs separated by delays in the initial block. You will then manually view the outputs in the waveform to see that they look as expected. Open ModelSim and create a new project. Add existing SystemVerilog files: thunderbird.sv and testbench_fsm.sv. Compile these files as you did in Lab 3. If there are any errors or warnings (see the Transcript pane), fix them and recompile.

When both files compile, simulate the testbench (**not** the thunderbird module). Do this by clicking on Simulate → Start Simulation in the File menu (see Figure 3 below).

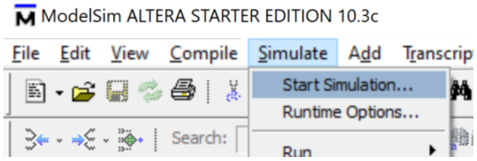
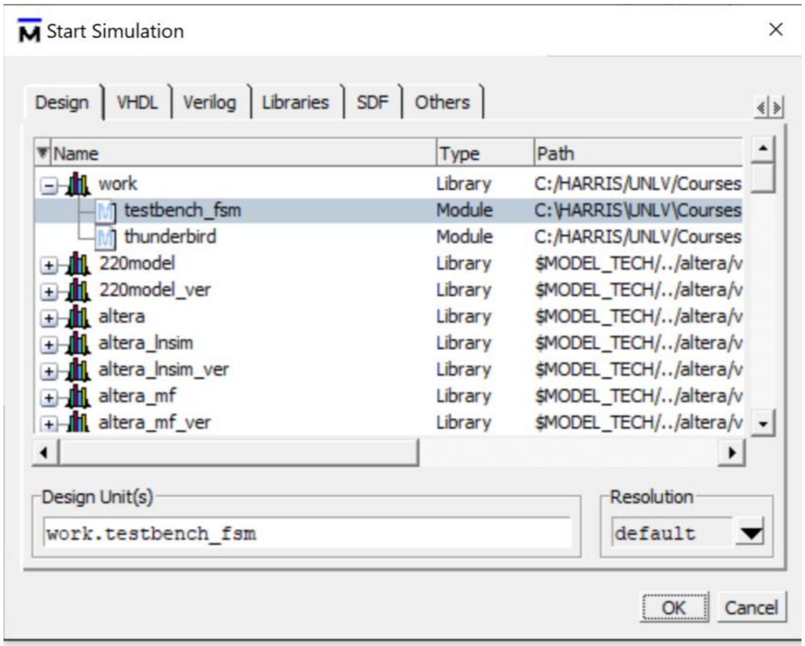


Figure 3. Start simulation in ModelSim

Now expand the work folder. Select (click on) testbench_fsm, and click OK (see Figure 4 below).



Name	Type	Path
work	Library	C:/HARRIS/UNLV/Courses
testbench_fsm	Module	C:\HARRIS\UNLV\Courses
thunderbird	Module	C:/HARRIS/UNLV/Courses
220model	Library	$MODEL_TECH/../altera/v
220model_ver	Library	$MODEL_TECH/../altera/v
altera	Library	$MODEL_TECH/../altera/v
altera_lnsim	Library	$MODEL_TECH/../altera/v
altera_lnsim_ver	Library	$MODEL_TECH/../altera/v
altera_mf	Library	$MODEL_TECH/../altera/v
altera_mf_ver	Library	$MODEL_TECH/../altera/v

Design Unit(s): `work.testbench_fsm`

Resolution: `default`

Figure 4. Select testbench_fsm to simulate

Click on the Wave pane (if you don't see a Wave tab, click on View → Wave from the top File menu). Drag all the signals from the Objects pane into the Wave window (click, shift-click and then drag to the Wave pane). Now type 'run 400' in the Transcript pane. This will run the testbench for 400 simulation ticks. View the waveform and make sure it looks as expected. If it doesn't, revise your FSM in thunderbird.sv, recompile and resimulate. Remember that you can view additional signals (other than the inputs and outputs) by drilling down into the hierarchy and viewing the signals (including internal signals) of lower-level modules. For example, you may want to view the state or nextstate signals.

To view and open your SystemVerilog files after you have started your simulation, click on the Project tab as shown in Figure 5.

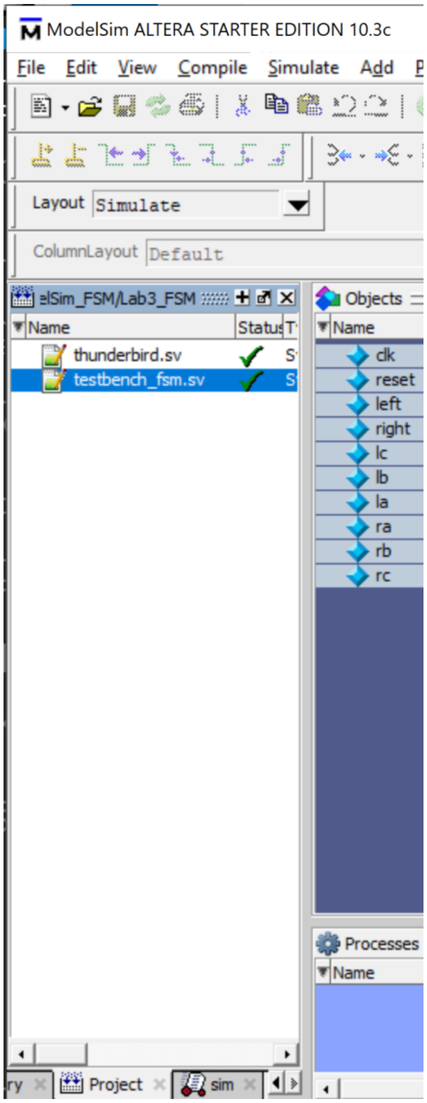
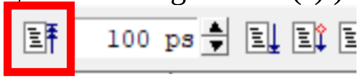
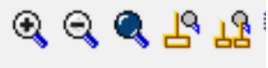


Figure 5. .sv files in Project tab

After you've made changes and recompiled your files, restart the simulation (i.e., reload the updated SystemVerilog module(s)) by clicking on the Restart button at the top of the ModelSim window. 

. You can also use the Zoom buttons to navigate the Wave window and view the waveform:  . Zoom Full allows you to see the entire waveform.

The Toggle Leaf Names <-> Full Names button allows you to shorten the signal names to not show the module name / hierarchy (see Figure 6).

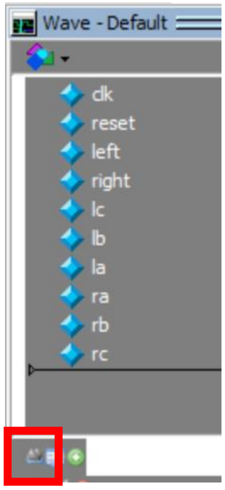


Figure 6. View signal names only

You'll probably have errors in your SystemVerilog file 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!

3. TEST FSM IN HARDWARE

Now you will download your FSM onto the DE2-115 board and test it in hardware. Create a new project in Quartus II (refer to Lab 3 instructions if needed). Be sure to target the correct FPGA. Name the project thunderbird_wrapper. In addition to adding your thunderbird.sv file to the project, also add the thunderbird_wrapper.sv file (available on Canvas).

The thunderbird_wrapper module maps the interface of the Thunderbird FSM to the pushbuttons, switches, and LEDs on the DE2-115 FPGA board. In the last lab, you manually added pin assignments. In this lab, you will use the wrapper module and a Quartus Settings File (.qsf) to import and map the pin assignments. A wrapper module essentially renames a module's interface pins to what is expected in the pin assignment file. In this case, we will map the FSM's interface signals as shown in Table 1 and Figure 7.

Table 1. Thunderbird FSM to DE2-115 board assignments

thunderbird signal name	DE2-115 Board signal name	Description
clk	KEY[0]	Right-most pushbutton
reset	SW[0]	Right-most toggle switch (i.e., toggle switch[0])
right	SW[1]	Toggle switch[1]
left	SW[2]	Toggle switch[2]
lc	LEDR[5]	Red LED[5]
lb	LEDR[4]	Red LED[4]
la	LEDR[3]	Red LED[3]
ra	LEDR[2]	Red LED[2]
rb	LEDR[1]	Red LED[1]
rc	LEDR[0]	Red LED[0]

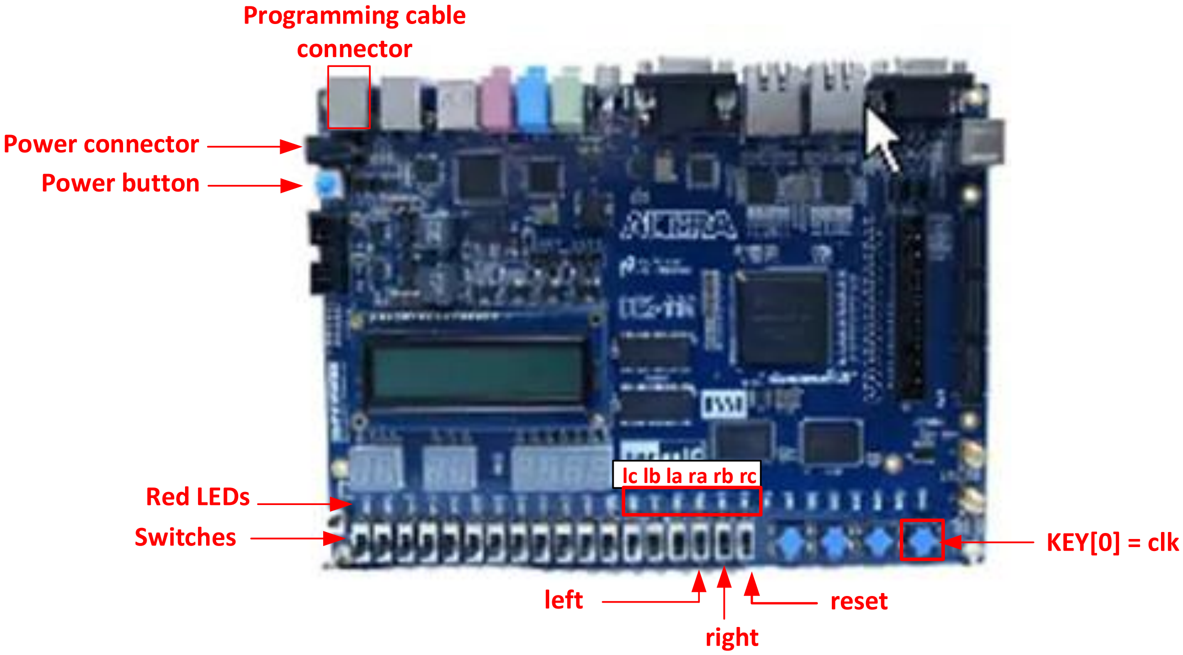


Figure 7. DE2-115 Board Interface to the Thunderbird FSM

After creating the thunderbird_wrapper project, you will map the signal names (KEY[0], SW[0], etc.) to the FPGA pins to which those peripherals (pushbuttons, switches, etc.) are physically connected. You will do this by importing the de2_115.qsf (Quartus Settings File). This file is available and should be downloaded from Canvas. Open that file in any text editor (for example, Notepad++), to view its contents. For example, the following line (found in the middle of the file) assigns PIN G19 of the FPGA to signal LEDR[0]:

```
set_location_assignment PIN_G19 -to LEDR[0]
```

That LED (LEDR[0], i.e., the right-most red LED) is physically connected to PIN G19 on the FPGA using a trace (a wire printed on the printed circuit board: PCB) between the FPGA and the LED.

Earlier in the file, the voltage level of that pin was set using this text:

```
set_instance_assignment -name IO_STANDARD "2.5 V" -to LEDR[0]
```

In this lab, we will use only a subset of the peripherals available on the DE2-115 board. Although we will not be using all the I/O (i.e., peripherals) in this lab, this .qsf file defines the pin connections of all the I/O available on the DE2-115 board for your future use and reference.

Import Pin Assignments

Import the pin assignments from the qsf file by clicking on Assignments → Import Assignments, as shown in Figure 8.

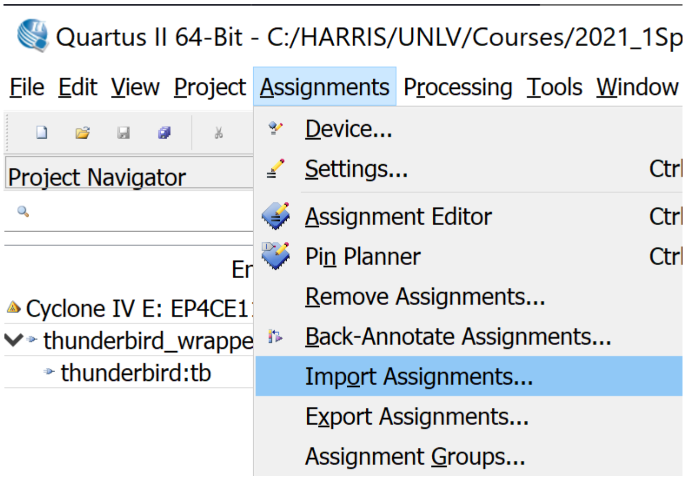
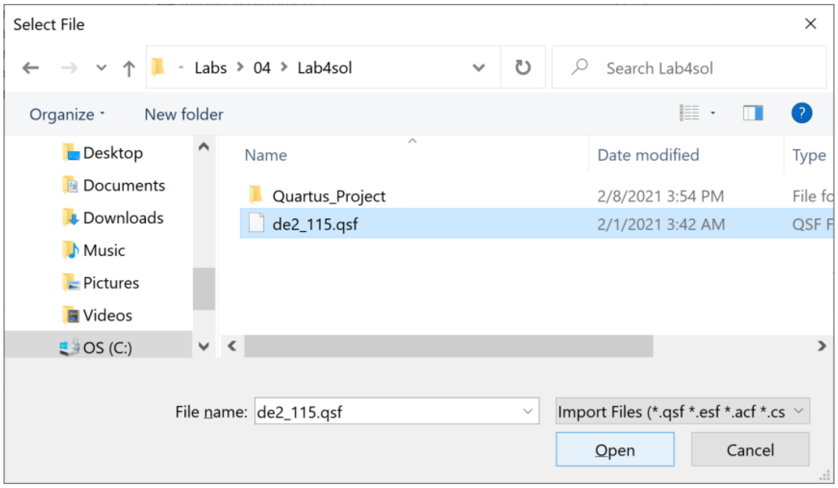


Figure 8. Importing pin assignments

In the next window, browse to where you have placed the de2_115.qsf file (see Figure 9). Select that file (click on it) and click Open.



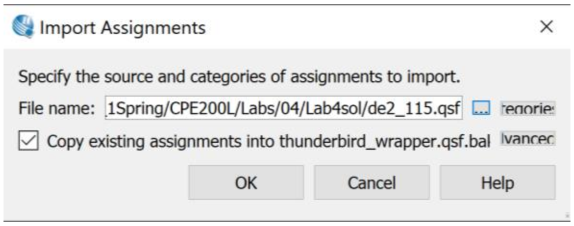
Name	Date modified	Type
Quartus_Project	2/8/2021 3:54 PM	File fo
de2_115.qsf	2/1/2021 3:42 AM	QSF F

File name: de2_115.qsf

Import Files (*.qsf *.esf *.acf *.cs

Figure 9. Importing pin assignments in the de2_115.qsf file

In the next window, you can leave the Copy existing… box selected (see Figure 10). Then click OK.



Specify the source and categories of assignments to import.

File name: 1Spring/CPE200L/Labs/04/Lab4sol/de2_115.qsf

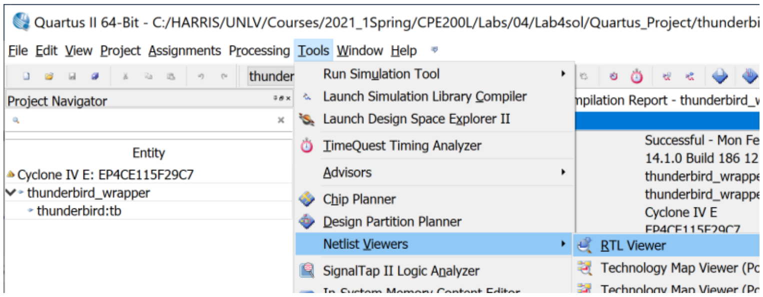
☑ Copy existing assignments into thunderbird_wrapper.qsf.bal

Figure 10. Importing pin assignments – last step

Now you are ready to compile the design (▶). View the messages page on the bottom of the Quartus II window and fix any errors. Note that you will see some warnings that you can ignore: for example, "Timing requirements not met" (there were no timing requirements), "Ignored locations or region assignments to the following nodes" (if you expand that warning, you will see all the pin assignments/signals that were not used in your design but that were in the .qsf file).

View Synthesis Results

View the RTL (register transfer logic) – gates and registers – that your SystemVerilog synthesized to by clicking on Tools → Netlist Viewers → RTL Viewer. A net is a node (signal) and a netlist is a list of nodes and their connections.


File Edit View Project Assignments Processing Tools Window Help
thunder
Project Navigator
Entity
Run Simulation Tool
Launch Simulation Library Compiler
Launch Design Space Explorer II
TimeQuest Timing Analyzer
Advisors
Chip Planner
Design Partition Planner
Netlist Viewers
SignalTap II Logic Analyzer
In-System Memory Content Editor
RTL Viewer
Technology Map Viewer (Pc
Technology Map Viewer (Pc
mpilation Report - thunderbird_w

Quartus II 64-Bit - C:/HARRIS/UNLV/Courses/2021_1Spring/CPE200L/Labs/04/Lab4sol/Quartus_Project/thunderbi

Cyclone IV E: EP4CE115F29C7
thunderbird_wrapper
thunderbird:tb

Successful - Mon Fe
14.1.0 Build 186 12
thunderbird_wrappe
thunderbird_wrappe
Cyclone IV E
EP4CE115F29C7

Figure 11. View RTL created by synthesis tool

You should see the figure shown in Figure 12. You can then click on the module (in this case thunderbird: tb) to view lower levels of the hierarchy. When you click on the FSM itself (i.e., the next state logic and the state register), it will show you a state transition diagram. View what your SystemVerilog module synthesized to and make sure it makes sense. If not, fix any errors and recompile (i.e., resynthesize).

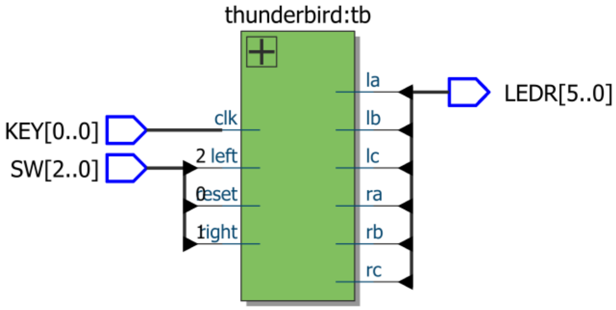


Figure 12. View circuit created by synthesis tool

Under the Flow Summary, look at the resource utilization summary. Check that the number of register and I/O pins matches your expectations. Notice that the number of registers may be more than the number you specified – the synthesis tool may choose a different state encoding than you suggested in your .sv file. If you wanted a specific state encoding, you would need to specifically encode the state register, next state logic, and output logic using gates and registers.

Download Design and Test It in Hardware

Download the design to the DE2-115 board. Check the wrapper, Table 1, and Figure 7 to see which buttons and switches are used for which inputs. Note that a pushbutton switch is used to create a clock. Test your design and watch the LEDs. Remember to reset the system before beginning your testing.

Note: the toggle and pushbutton switches sometimes experience a phenomenon called *bounce*, in which the mechanical contacts bounce as the switch is opening or closing, creating multiple rapid rising and falling pulses rather than a single clock edge. If your lights seem to skip through multiple states at a time, it is probably because of switch bounce on the clock switch. With a bit of practice, you can learn to push the switch in a way that bounces less. It is also possible to build a circuit to "debounce" a switch, but that is beyond the scope of this lab.

6. WHAT TO TURN IN

Total points available: 15

1) **[1 pt]** Please indicate how many hours you spent on this lab. This will be helpful for calibrating the workload for next time the course is taught. Include any suggestions you have for improving the lab. You will get one point for listing your time on each lab.

2) **[14 pts total]** Thunderbird FSM

   a) **[3 pts]** Submit a clear sketch of your Thunderbird FSM state transition diagram.
   b) **[2 pts]** Submit thunderbird.sv (as a separate file from your PDF submission).
   c) **[2 pts]** Submit a screenshot of your Thunderbird FSM state transition diagram timing waveform. Be sure to display the signals in this order (from top to bottom): `clk, reset, left, right, lc, lb, la, ra, rb, rc`.
   d) **[2 pts]** Take a picture of your FSM displaying `ra`, then `ra` and `rb`, then `ra`, `rb`, and `rc` on the right-most LEDs (3 pictures). Clearly also show the "right" switch (SW[1]) being asserted.
   e) **[1 pts]** Take a picture of your FSM displaying `la`, `lb`, and `lc` on LEDR[5:3] (1 picture). Clearly also show the "left" switch (SW[2]) being asserted.
   f) **[2 pts]** Briefly describe how you tested the system on the DE2-115 board and whether it worked according to the specifications. Did you observe switch bounce?
   g) **[2 pts]** Write a brief paragraph about what you learned in this lab.

Please indicate any bugs you found in these lab instructions, or any suggestions you have to improve the lab.