Lab 6: Project Proposal, State Machines, LCD, Keyboard, Audio and VGA...and a Partridge in a Pear Tree.

CSC 258, University of Toronto

About this Lab

Learning Objectives

1. To give you practice with FSMs in Verilog.
2. To introduce you to some new I/O options that the DE2 board provides (LCD display, keyboard, speakers).

Marking Scheme

All items in this document labeled **PRELAB TODO:** are to be completed before arriving to the lab, and are worth one of the four marks for the labs. This includes the handin components for Parts I - III. Items marked **INLAB TODO:** are to be completed in lab and will all be marked. There is one mark available for Part II, one mark for completing Part III and IV, and one mark for completing Part V. Note that there is also bonus mark to be earned for completing section VI.

Pre-lab Assignment

Part I - Project Proposal

For Part I of this lab you will be marked on the *completion* of your project proposal. **PRELAB TODO:** You will submit this portion of the prelab online, to be marked for content later in the week. To prove that you have submitted your proposal, print out the confirmation page that appears after you submit your proposal, and present this page to the TAs. Any proposals that are not submitted by the beginning of the lab will get zero on the project proposal mark!

Your proposal is to contain:

1. Your and your project partner's names, student numbers and emails.
2. The title of your project.
3. Your general goal/motivation for the project.
4. Why this project is cool (for both CSC258 students and non-CSC258 students).
5. What your deliverable/milestone will be for the first week.
6. What your deliverable/milestone will be for the second week.
7. What your deliverable/milestone will be for the third week (ie. what your completed project will look like.)
8. How this project builds on and uses the material you have learned in this class.

Handing in your proposal

Submit the elements of your proposal online at `http://tinyurl.com/CSC258Proposal`. Once that is complete, print the confirmation page from your submission and enclose it with the prelab handin, so that the TAs can give you the lab mark for the proposal submission. The contents of the proposal will be verified separately by the instructor for 2% of your final grade (separate from the 4% for this lab), and you will get feedback sent to you by email from the course instructor and/or the head TA if there are any changes that need to be made to your proposal.

Course vs. personal goals of the project

For your project, we will be marking you on whether you meet your personal goals for the project, as well as course-level goals. The course-level goals for the project are:

1. For you to demonstrate your understanding of digital logic, hardware, and Verilog.
2. For you to implement a creative application of these topics.

Project milestones

For your project milestones, we will be marking you as such:

Week 1 (worth 4% of your final grade) is out of 4: one quarter of your mark for attempting to meet your week 1 milestone; the rest for satisfying your milestone.

Week 2 (worth 4% of your final grade) is out of 4: one quarter of your mark for attempting to meet your week 1 milestone; the rest for satisfying your milestone.

Week 3 (worth 4% of your final grade) is out of 4:

- 3 marks for the milestone (one third of your mark for attempting to meet your week 3 milestone; two-thirds for satisfying your milestone.)
- 1 mark for achieving the personal and course-level goals for your project

Final report

Your report is an optional component to the project, which you can submit if you feel that there was work done on the project that didn't come through during the in-lab demos. The report can potentially earn you an extra 4%, which can compensate for any marks you may have lost during your project milestones. However, your total project milestone marks cannot exceed the maximum of 12%.

Should you decide to write a report for your project, it should have a title page, and should be about 3-5 pages in length (not including the title page or appendices). Please include code that you created as an appendix (and cite the source of any code that you used from outside sources). Your report should be single spaced, and marked out of 4:

- 2 marks for clarity
  - 1 mark for structure and readability
  - 1 mark for grammar, spelling, etc.
- 2 marks for convincing us (the teaching staff) that you deserve marks.
  - 1 mark for explaining what you did and why this project is so cool that somebody outside the course would think it is interesting.
  - 1 mark for explaining why it is relevant to the course and builds on the course material, and what you personally learned and got out of doing this project.

Part II - Finite State Machine

We wish to implement a finite state machine (FSM) that recognizes two specific sequences of applied input symbols, namely four consecutive 1s or four consecutive 0s. There is an input w and an output z. Whenever $w = 1$ or $w = 0$ for four consecutive clock pulses the value of z has to be 1; otherwise, $z = 0$. Overlapping sequences are allowed, so that if $w = 1$ for five consecutive clock pulses the output z will be equal to 1 after the fourth and fifth pulses. Figure 1 illustrates the required relationship between w and z.

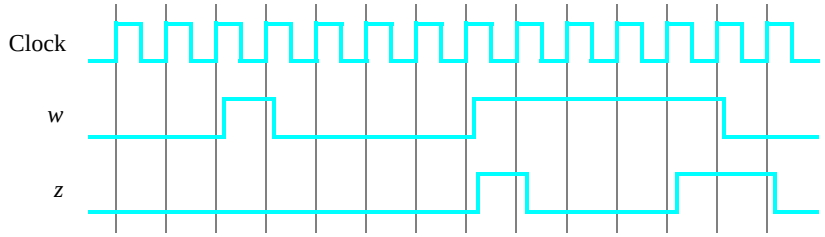

Figure 1: Required timing for the output z.

A state diagram for this FSM is shown in Figure 2. **PRELAB TODO:** For this part you are to manually derive an FSM circuit that implements this state diagram, including the logic expressions that feed each of the state flip-flops. To implement the FSM use nine state flip-flops called $y_8, \ldots, y_0$ and the one-hot state assignment given in Table 1.

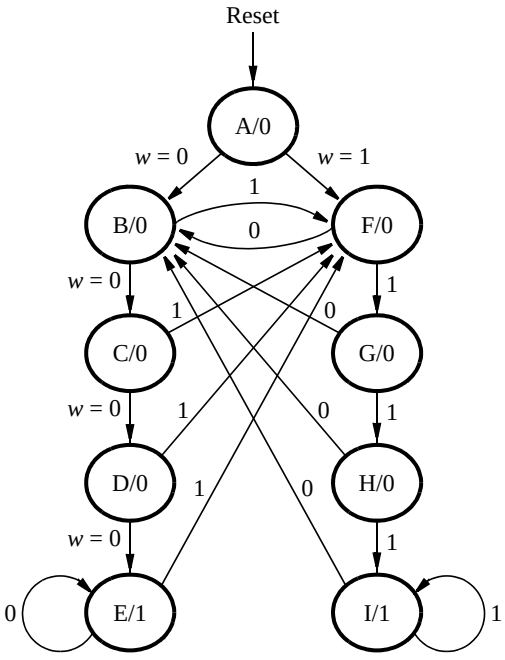

Figure 2: A state diagram for the FSM.

Name	State Code $y_8y_7y_6y_5y_4y_3y_2y_1y_0$
A	000000001
B	000000010
C	000000100
D	000001000
E	000010000
F	000100000
G	001000000
H	010000000
I	100000000

Table 1: One-hot codes for the FSM.

Design and implement your circuit on the DE2-series board as follows:

1. Create a new Quartus II project for the FSM circuit. Select the appropriate target chip that matches the FPGA chip on the Altera DE2-series board.

2. Write a Verilog file that instantiates the nine flip-flops in the circuit and which specifies the logic expressions that drive the flip-flop input ports. Use only simple **assign** statements in your Verilog code to specify the logic feeding the flip-flops. Note that the one-hot code enables you to derive these expressions by inspection.

   Use the toggle switch SW_0 on the DE2-series board as an **active-low synchronous reset** input for the FSM, use SW_1 as the w input, and the pushbutton KEY_0 as the clock input which is applied manually. Use the green light $LEDG_0$ as the output z, and assign the state flip-flop outputs to the red lights $LEDR_8$ to $LEDR_0$.

3. Include the Verilog file in your project, and assign the pins on the FPGA to connect to the switches and the LEDs, as indicated in the User Manual for the DE2-series board. Compile the circuit.

4. Simulate the behaviour of your circuit.

5. Once you are confident that the circuit works properly as a result of your simulation, download the circuit into the FPGA chip. Test the functionality of your design by applying the input sequences and observing the output LEDs. Make sure that the FSM properly transitions between states as displayed on the red LEDs, and that it produces the correct output values on $LEDG_0$.

6. Finally, consider a modification of the one-hot code given in Table 1. When an FSM is going to be implemented in an FPGA, the circuit can often be simplified if all flip-flop outputs are 0 when the FSM is in the reset state. This approach is preferable because the FPGA's flip-flops usually include a *clear* input, which can be conveniently used to realize the reset state, but the flip-flops often do not include a *set* input.

Part III - LCD

For the following sections, don't print all the code that you download, just the relevant sections that you changed for the lab.

For this part we will introduce output to the LCD display. LCD screens are extremely useful for displaying data values, but the code needed to implement them can be complicated. Application Programmer Interfaces (APIs) are created for this reason, to allow developers to work with devices like the LCD without having to research and recreate the implementation from scratch.

Instead of creating the LCD code yourself, take LCD code that already works and modify it to do what you need. To do this, download the code at `http://www.johnloomis.org/digitallab/lcdlab/lcdlab3/lcdlab3.html` and create a Quartus project for it. (You can download all the files together at `http://www.johnloomis.org/digitallab/lcdlab/lcdlab1/lcdlab1.zip`.)

This code does the following: it takes the binary number provided in SW[7:0] and displays it to the LCD display in the following manner:

```
Count=XX
DE2
```

PRELAB TODO: Modify this code to combine it with your code from Part II so that it displays the label of the FSM's current state on the LCD. If the output signal z of the circuit goes high, print "HIGH" on the second line of the LCD display; otherwise print "LOW". Print out the modified code and include it in your prelab handin.

For example, if the circuit has received four 0 inputs in a row, it should produce:

```
State=E
Z=HIGH
```

Note: Since simulating LCDs at home is tricky, we do not expect you to verify your code through simulation as part of the pre-lab.

Part IV - Keyboard

You can find code to provide keyboard input to the DE2 board at `http://www.johnloomis.org/digitallab/ps2lab1/ps2lab1.html`. Each key on the keyboard has a unique 8-bit code – note that, for example, the 2 key at the top-left of your keyboard has a different value than the 2 in the number pad – or that the N key has the same value regardless of if caps lock is on. Also note that to use the keyboard at your workstation for DE2 output, you will need to press the 'Select' button on the KVM switch located above your monitor.

PRELAB TODO: Alter this code to display the key code received from the keyboard on the LCD.

Part V - Audio

PRELAB TODO: Find out how to send output to the speakers. Use the resulting code to send a 392 Hz pitch to the speakers (you may recognize this as the musical note G, used for tuning instruments.)
Hint: how can you slow down the 50 MHz clock of the DE2 board down to 392 Hz? *If you use any premade code you found on the internet, be sure to acknowledge where you found it in your code's comments. We're not expecting you to write this part from scratch!*

Part VI - VGA (Bonus)

PRELAB TODO: Find out how to display a square pattern on the display monitor.

Note: This is the same LCD display attached to the PC on your workstation. You can press the KVM switch to connect the LCD to the DE2 board.

Hint: Look here: `http://www.johnloomis.org/digitallab/vgalab/vgalab1/vgalab1.html` You might need to remove the file *VGA_Controller.v* from the project in order to get the code provided to work.

In lab work

For each part of the pre-lab, compile and download your code to the FPGA. Test the functionality of your design by toggling the switches and observing the displays.

1. **INLAB TODO:** Demonstrate, on the board, your code for Part II to your TA.
2. **INLAB TODO:** Demonstrate, on the board, your code for Part III to your TA.
3. **INLAB TODO:** Demonstrate, on the board, your code for Part IV to your TA.
4. **INLAB TODO:** Demonstrate, on the board, your code for Part V to your TA.
5. **INLAB TODO:** Demonstrate, on the board, your code for Part VI to your TA. (Bonus)

Clean up your station once you are done the lab.