ECE 270
Practice Lab Practical

STEP	DESCRIPTION	MAX
1	Toggle and display a state element	20
2	Display a shift register	20
3	Display a one-digit decimal counter	20
4	Display a two-digit decimal counter	20
5	Display a three-digit decimal counter	20
6	Post-lab submission: Submit your code.	*
7	Clean up your lab station and log out.	*
	TOTAL	100

*** All lab points are contingent on this step.**

Instructions:

This is a practice lab practical. It would be best if you did no work outside of lab to get this to work and, instead, spent two hours in lab to start and finish it. Your work will not be scored in lab. Instead, all grading will be done after the lab using your submitted work.

Click on the "**ece270-setup**" icon on your desktop (or open a terminal window and run the **~ece270/bin/setup** command). This will create a folder in your ece270 directory called lab13. In it is a lab13.v file. Open this file with kate or gedit as you did in previous lab experiments. Do all of your work in this file.

Your work will be scored based on its operation on an FPGA. You may use the simulator to test and develop your work, but it will not be used for grading. To be clear: You get no credit for code that works on the simulator but not on the FPGA. In an actual lab practical, you will not be given access to a simulator or any other Internet resources.

At any time, you may see a demonstration of expected functionality by opening a terminal window and typing:

```
cd ~/ece270/lab13
make demo
```

Press ‘1’ and ‘Z’ to see what happens. Hold down ‘A’ or ‘D’ on the FPGA system (it will not work on the simulator) to accellerate or decellerate the display rate.

In an actual lab practical, if you encounter strange technical problems such as hardware failures, lights not illuminating as expected, please ask a TA to evaluate it, but you will not be given help with writing or debugging your Verilog code. **For this practice practical, you should feel free to ask for help. That’s why it’s practice.**

Moreover, for this lab exercise, ask yourself the question, “Could I start and finish this entire exercise in two hours?”

Step (1): Toggle and display a state element

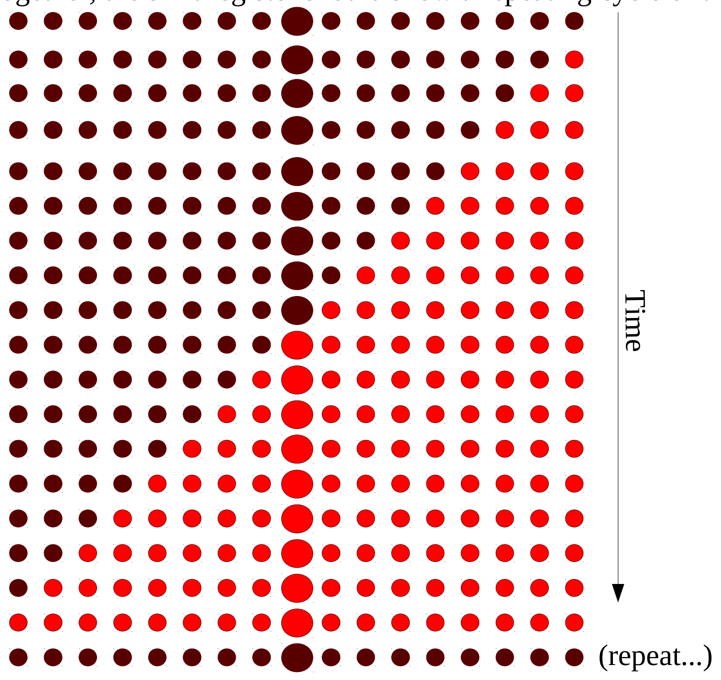
Create a state element named **enable** that is toggled (turned on and off) by pressing the '1' button. It is asynchronously reset back to zero by pressing the 'Z' button which has already been configured to assert the **reset** signal. Use this state element to illuminate the decimal point of the leftmost 7-segment display (ss7[7]). **Your work will be evaluated on the following criteria:**

- Each single press of the '1' button alternately turns the decimal point on or off.
- When the 'Z' button is pressed, the decimal point turns off, regardless of the status of the '1' button.
- While the 'Z' button is held, pressing the '1' button does nothing.

Experiment Step (2): Display a shift register

Create a 17-bit shift register and connect it to the eight right LEDs, the middle red LED, and the left LEDs. It should have the following behavior:

- As long as **reset** is asserted, all bits are asynchronously set to zero. (all LEDs off)
- The shift register should be clocked by the **hz100** signal.
- As long as the **enable** signal, above, is asserted, one new LED at a time should be illuminated, starting from the right, on each positive edge of the **hz100** clock.
- When the all 17 LEDs are lit, the entire shift register will be set back to zero on the next positive clock edge.
- Altogether, the shift register should show a repeating cycle of the following 18 steps:



- If **enable** is deasserted, the pattern will freeze. When enable is reasserted, the pattern will continue.
- Each full cycle of 18 steps will take 18 * 0.01 seconds.
- (You can hold the 'A'/'D' buttons to accellerate/decellerate the 100 Hz clock.)

Your work will be evaluated on the following criteria:

- While **reset** is asserted, all LEDs should be dark.
- Pressing '1' to toggle **enable** will do nothing as long as **reset** is asserted.
- Each LED should be illuminated in sequence from right to left.
- At the end of the 18-step cycle, all the LEDs should be lit.
- The cycle should repeat at 1/0.18 Hz (approximately 5.556 Hz).
- If **enable** is asserted, then deasserted without pressing reset, a contiguous line of between 0 and 17 LEDs should be lit. If enable is pressed again, the pattern will resume where it left off.

Experiment Step (3): Display a one-digit decimal counter

Create a 4-bit synchronous decimal up-counter that is clocked by the leftmost LED of the shift register, is enabled by the **enable** signal, and is asynchronously reset to zero when **reset** is asserted. It should count from 0 – 9 and repeat. Copy your **ssdec** module from a previous lab. (This is an example of a module that you should expect to be given in a practical exam, so you don't have to create it on the spot.) Create an instance of the **ssdec** module to display the counter value on **ss0**. **Your work will be evaluated on the following criteria:**

- As long as **reset** is asserted, **ss0** should display a zero.
- When **enable** is asserted, the digit should increment every time the leftmost LED is illuminated (at a rate of 5.556Hz).
- When **enable** is toggled, the count stops and resumes.
- The counter should repeatedly count from 0 to 9.

Experiment Step (4): Display a two-digit decimal counter

Extend Step (3) to create a two-digit decimal counter so that, when the digit displayed on **ss0** reaches 9, a next-place digit displayed on **ss1** is incremented by 1. When the digit displayed on **ss1** is zero, it should not be displayed. Together, the digits on **ss1,ss0** should display " 0", " 1", " 2", " 3", " 4", " 5", " 6", " 7", " 8", " 9", "10", "11", and so on, up to "99". After that, they should go back to displaying a single zero on **ss0**. As before, **reset** should reset the counter back to zero, and toggling **enable** should stop and resume the count. **Your work will be evaluated on the following criteria:**

- As long as **reset** is asserted, **ss1** should be blank and **ss0** should display a zero.
- When **enable** is asserted, the counter should increment from 0 to 99 and repeat.
- When **enable** is toggled, the count stops and resumes.

Experiment Step (5): Display a three-digit decimal counter

Extend Step (4) to create a three-digit decimal counter shown on **ss2,ss1,ss0** that hides the left two digits when they are zero. (The middle digit should be shown for a value like 103 though.) The counter should count from 0 to 999 and repeat. This step will be evaluated like the previous two.

Experiment Step (6): Upload your work in the post-lab submission

Upload the contents of your completed and tested lab13.v file in the post-lab submission. For this lab, your score will be determined entirely by the submitted file.

Experiment Step (7): Clean up your lab station and log out.

Show your TA that you have logged out of your workstation. He or she will press the key combination <Ctrl> <Alt> <F7> to check if you have created a new session instead of logging out. Also, clean up your lab station. Your TA gets to decide if it is clean enough – not you.

© 2019 by Rick