EELE 367 – Logic Design
Lab #4 – Binary Character Display

Overview

In this lab you be designing a 4-bit binary character display using the 7-segment displays on the I/O shield. You will first create a lower-level subsystem that fully contains your 2^n clock divider from last lab. You will then create a new 4-bit ripple counter using D-flip-flops in your top.vhd. Each of the four bits of the counter will drive an individual character display, thus creating a 4-bit binary display (i.e., the displays will only show a 0 or a 1). You will also display the 4-bit binary counter, the settings of your selectable clock sub-system, and the divided down clock on the LEDs.

Outcomes

After completing this lab you should be able to:

- Instantiate numerous sub-systems to accomplish tasks in a higher level system.
- Use signal concatenations to effectively drive lower level sub-systems and output ports.
- Use signal concatenations to join fixed logic values with signals.

Deliverables

The deliverable(s) for this lab are as follows:

- Pre-lab: n/a.
- Demonstration of the successful operation of your design (90%).
- Uploading your top.vhd file for this lab to the course DropBox (10%).

Lab Work & Demonstration

System Design & Demonstration (90% + 10%)

A. Create a New Quartus Project

You are going to create a new Quartus II project for this lab. Create a new folder called "Lab04_Binary_Character_Display". Open your Lab #3 project and copy it over to a new project in the directory you just created using *Project – Copy Project* in Quartus II. Open the new project.

B. Make a sub-system for your Selectable 2^n Clock Divider.

In last lab, you implemented a selectable clock divider in your top level by instantiating 32x D-flip-flops and creating a 32-to-1 multiplexer using a process. In this lab, you are going to create a new sub-system called "clock_div_2ton.vhd" that contains all of the functionality for your clock divider. Use the following entity definition:

```
entity clock_div_2ton is
   port (Clock_in  : in  std_logic;
         Reset     : in  std_logic;
         Sel_in    : in  std_logic_vector (4 downto 0);
         Clock_out : out std_logic);
end entity;
```

Move your 32-bit ripple counter and multiplexer into the clock_div_2ton sub-system and modify the signal names as necessary to make the sub-system operational. Instantiate this sub-system in your top.vhd file.

Drive the Clock_in port with the top level Clock. Create a new internal signal called “Clock_div” to connect to the Clock_out port. Drive the Sel_in port with SW2[4:0]. You will use the Clock_div as the main clock for the rest of your logic in top.vhd.

C. Create a new 4-bit ripple counter in your top level.

Instantiate 4x D-flip-flops in your top.vhd file and connect them accordingly to make a new 4-bit ripple counter. Drive the first stage clock of the ripple counter using the Clock_div signal created by your clock_div_2ton sub-system.

D. Create a Binary Character Display

Use each bit of the 4-bit counter to drive a separate 7-segment decoder in order to display either a 0 or 1 on the character displays of the I/O shield. You will need to instantiate 4x of your decoder_7seg_4in.vhd sub-system from prior labs.

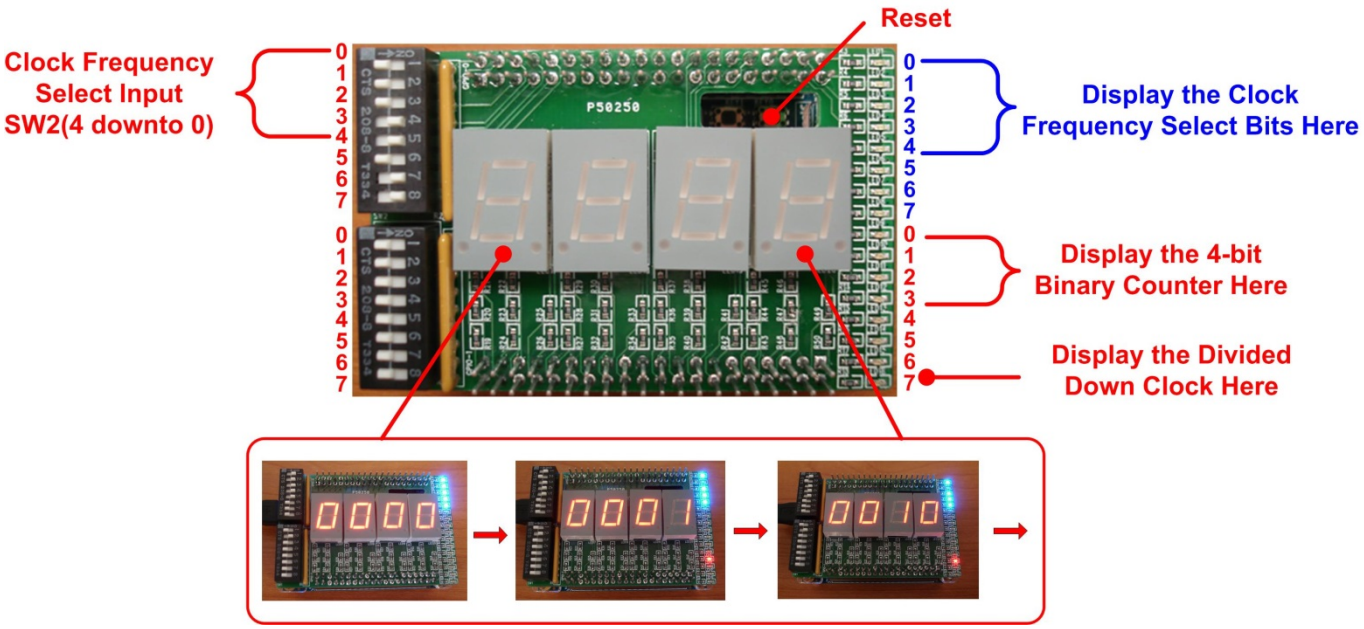
HINT: Consider using signal concatenations to create the 4-bit input vectors to your decoders. What do you need to do to only allow the decoder to drive characters of 0 and 1?

E. Route the 4-bit counter, the clock select inputs, and the divided down clock to the LED_Red and LED_Blue LEDs.

Display the clock select inputs SW2(4:0) on LED_Blue(4:0).

Display the 4-bit binary counter on LED_Red(3:0).

Display the divided down clock (Clock_div) on LED_Red(7).



DEMO

Demonstrate the proper operation of your design (90% of your grade).

F. Upload your top.vhd file to the Lab #4 DropBox (10% of your grade).