

Lab 5 Assignment- Introduction to Counters and CPLDs

Objectives:

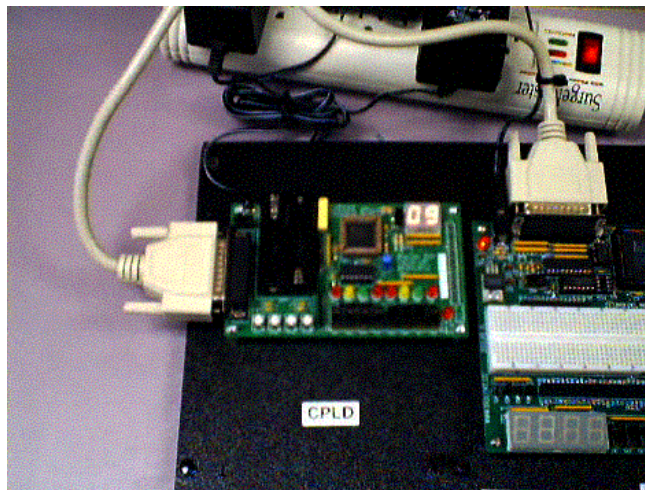
By the time the student completes this lab, and it's associated tutorial, the student should be familiar with the following:

1. How to create multiple-file VHDL designs.
2. How to simulate these designs in ModelSim.
3. How to map these designs to the target device.
4. How to program the design onto the device, and verify it's function.

Note: Before the student performs the lab itself, he or she should have already completed the tutorial.

Part 1: Testing Your Understanding

Here, you will be modifying the design you created in the Lab 5 tutorial. Recall that when we programmed the device, we got the following result:



That is, we had one digit counting from 0xh to Fxh until the end of time, and one digit remaining at zero. Now, we are going to change that so both digits count from 0xh to Fxh (the "h" in xh means hex. To do this, open the XCR_TOP.VHD file. You will need to find the area marked:

-- Component instantiations

Below it, you will notice that we already have one XCR_COUNT component, labeled as C_COUNT. Go ahead and add another one. Do this by copying the first component instantiation and pasting the result right below. Call the second counter anything you want. However, before you can save the design and re-implement it, you will need to make one change to the second component. Recall that in our design, we have a signal called data_bus that has eight individual signals grouped into a bus. In our first counter component C_COUNT, we have the output of the counter mapped like so:

```
data_out_h => data_bus(3 downto 0)
```

You will need to change this in the second counter component. We do not want two counters sharing the same sets of pins. As a result, you will need to change that line in the second counter component from what you see above to what you see below:

```
data_out_h => data_bus(7 downto 4)
```

Now your design will use both digits in the display (there is no need to change the seven-segment driver, it already has support for the second digit built into it.)

Simulate the modified design in ModelSim, and program it onto the XCR board. Demonstrate the resulting design to your TA.

Part 2: Design

The second part of this lab involves designing your own counter. We could give you something as simple as a two-digit hex counter, identical to the one we showed you in the tutorial, except it counts from 00xh to FFxh, (0 - 255 in decimal.) However, doing this one is very simple. All it involves is changing two or three lines in the pre-existing counter. That is, the lines that tells the counter to roll over at 15. In the new design, we would change those to tell the counter to roll over at 255. Furthermore, we would change the output width from four bits to eight bits.

As a result, we will give you a slightly more complex counter design. Here are the specifications:

A. The counter will count from 0 - 99 in decimal. This means that instead of implementing a straight counter, like we did in the tutorial, you will have to force the digit to roll over at nine, instead of fifteen. And, when the digit rolls over, you will have to increment the leftmost digit by one. Here is a simple algorithm describing the basic function:

```
if(right_digit < 9)
    increment right_digit.
if(right_digit = 9)
    right_digit = 0
increment left_digit
```

B. The counter must have a reset function. This reset function should reset the whole system to "00". Note that you are not being asked to implement a pause function; the first part will be complex enough.

You need to do this as a multiple-file VHDL design. Create a top-level file, your counter file, and just include the seven segment driver that was furnished to you in the tutorial.

When you have successfully simulated your design and programmed it to the device, demonstrate the resulting design to your TA. In your report, you need to supply all of your VHDL files, and your ModelSim testbench.