ECE 274 Digital Logic – Spring 2008

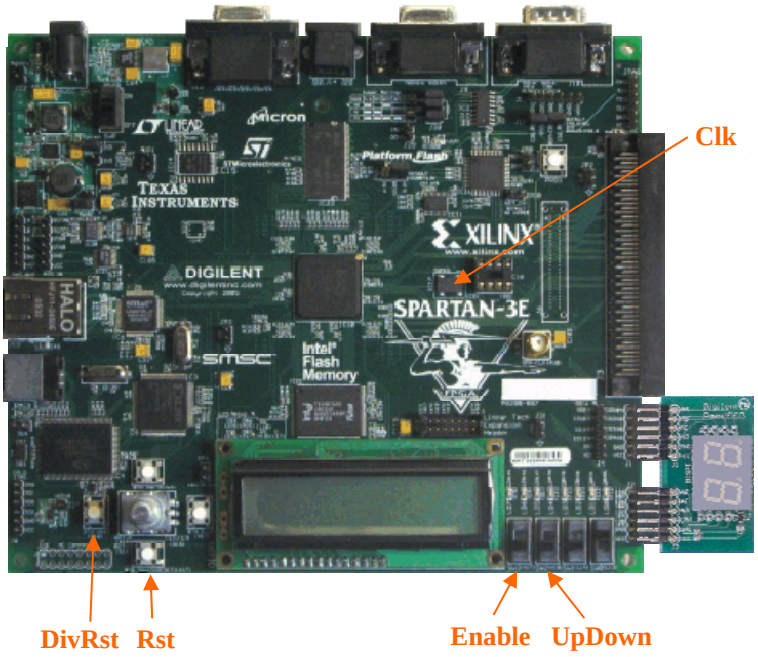
Lab 3: FSM for 4-bit Up/Down Counter

Starts: Feb 25 - Feb 29
Report/Code Due: Mar 14, 11:59PM

Lab Overview:

In this lab, you will build a two implementations for a 4-bit Up/Down Counter and interface the up/down counter with your binary to 7-segment LED decoder designed in Lab 2. The initial 4-bit up/down counter design has four inputs, `Clk`, `Rst`, `Enable`, `UpDown`, and a 4-bit output `Cnt`, represented by the four outputs `Cnt3`, `Cnt2`, `Cnt1`, `Cnt0`. If `Rst` is `1`, the counter should reset its count value to zero (`0000`). Otherwise, if `Enable` is `1`, on every clock cycle the counter should count up by one when `UpDown` is `1` or count down by one when `UpDown` is `0`. Upon reaching the minimum (0) or maximum (15) count, the counter value should wraparound. In other words, when counting up, the counter should wraparound to 0 after 15, and when counting down, the counter should wraparound to 15 after 0. If `Enable` is `0`, the counter should keep its present value.

In order to download your initial 4-bit up/down counter implementation to the Spartan-3E FPGA board, you will need to interface your counter with the binary to 7-segment LED decoder designed in Lab 2. Furthermore, the 50MHz clock integrated within the Spartan-3E FPGA board operates too quickly to be able to view the counter output on the 7-segment LED display. As such, you must also interface with the clock divider component provided with this lab. The clock divider, `ClkDiv`, has two inputs, `Clk` and `Rst`, and one output, `ClkOut`. Given the 50 MHz clock provided by the Spartan-3E FPGA board, the `ClkDiv` component will generate a 1 Hz clock on its output, `ClkOut`. This 1 Hz clock can then be connected to the input of your 4-bit counter. The resulting counter will count up or down once every second if enabled.



The figure to the above right provides an overview of the connections that will be used to implement your 4-bit Up/Down Counter on the Spartan-3E FPGA board. The overall counter implementation will have a single `Clk` input provided by the 50 MHz oscillator integrated on the Spartan-3E FPGA Board, two reset inputs – one for the clock divider (`DivRst`) and one for your 4-bit Up/Down Counter (`Rst`), and `Enable` and `UpDown` inputs connected to your 4-bit Up/Down Counter. The implementation will also interface with your binary to 7-segment LED display using the outputs to the individual segment of the 7-segment LED display. The following is a partial .ucf file providing the connection details for the new inputs and outputs needed for the 4-bit Up/Down Counter *(you will need complete the .ucf file to properly connect to the 7-segment LED display)*:

```
NET "Clk" LOC = "C9";
NET "Rst" LOC = "K17" | PULLDOWN;
NET "DivRst" LOC = "D18" | PULLDOWN;

NET "Enable" LOC = "N17";
NET "UpDown" LOC = "H18";

# Enter additional connections below
```

After completing the initial 4-bit Up/Down Counter design, you will extend the 4-bit Up/Down counter design to incorporate two additional control inputs, `NoWrap` and `OddCnt`. The input `NoWarp` will control whether or not the counter wraps around upon reaching the maximum, or minimum, value. If `NoWrap` is 1, the counter should either maintain the maximum value, when counting up, or maintain the minimum value, when counting down. The `OddCnt` input controls the operation of the counter by determining if the counter should count all numbers or only count odd numbers. If `OddCnt` is `1`, the 4-bit counter should only count odd numbers. Thus, if enabled and counting up, when `OddCnt` is `1`, the counter should count 1, 3, 5, 7, 9, 11, 13, 15, and repeat if the `NoWrap` is `0`. When implementing the modified 4-bit Up/Down Counter, the additional inputs must be mapped to additional switches on the Spartan-3E FPGA board.

Note: Your module names for the initial and modified 4-bit Up/Down Counter implementations should be unique as the represent two different designs. Furthermore, you will need to turn in the Verilog code for both designs and accompanying testbenches. Each module, testbenches included, should be saved as a separate file.

Lab Procedure & Demo

1. Behaviorally design the 4-bit Up/Down Counter as a Finite State Machine (FSM). You FSM design should consist of two always procedures. The first always procedure, should implement the state register. The second always procedure should implement the FSM control logic. In the design of the FSM control logic, you must use a **`case`** statement to describe the combinational behavior associated with each state *(20 points)*.

2. Create a testbench to test your design for correct functionality. Your testbench should be self checking using **`if`** statements and the **`$display`** task to report any errors during simulation. At a minimum, the self checking testbench should test the following cases *(20 points)*:
    a. Check that counter counts up and down correctly *(5 points)*
    b. Check for correct wraparound functionality for counting up and down *(5 points)*
    c. Check for correct reset behavior from non-zero count value *(5 points)*
    d. Check for correct enable behavior *(5 points)*

3. Create a new top-level component that structurally connects your 4-bit Up/Down Counter with the clock divider, `ClkDiv`, and your binary to 7-segment LED decoder. Complete the provided .ucf file and synthesize your 4-bit Up/Down Counter. Download and test your design on the Spartan-3E FPGA board for correct functionality. At a minimum, you should test the same cases as your self checking testbench *(20 points)*.

4. Modify your existing FSM design to incorporate the `NoWrap` and `OddCnt` control inputs and extend your self checking testbench to properly test the modified design. In addition to testing the same cases as the original design, the modified self checking testbench should test the following cases *(15 points)*:
    a. Check for correct non-wraparound functionality when counting up and down *(5 points)*
    b. Check for correct counting by odd number only *(5 points)*
    c. Check for correct enable and up/down behavior when counting by odd numbers *(5 points)*

5. Modify your top-level component to integrate your modify 4-bit Up/Down Counter design with the clock divider, `ClkDiv`, and binary to 7-segment LED decoder. Update the .ucf file to properly connect the `NoWrap` and `OddCnt` inputs to the appropriate switches. Synthesize and download your modified 4-bit Up/Down Counter to the Spartan-3E FPGA board and test your design for correct functionality. At a minimum, you should test the same cases as your self checking testbench *(5 points)*.

Lab Report Requirements *(In addition to the standard lab report format)*

1. State diagrams for the initial and modified 4-bit Up/Down Counter finite state machines.
2. Simulation waveforms demonstrating correct functionality of the initial and modified 4-bit Up/Down Counter designs for the required test cases. Provide annotations on your simulations waveforms to indicate where each test case is evaluated.

Code Turn-In Requirements

Note: Be sure to turn in your code for the initial and modified 4-bit Up/Down Counter designs and testbenches along with the corresponding .ucf files.