ECE 274 Digital Logic – Spring 2008

Lab 4: Binary to BCD Converter and Multiplexed BCD Display Driver

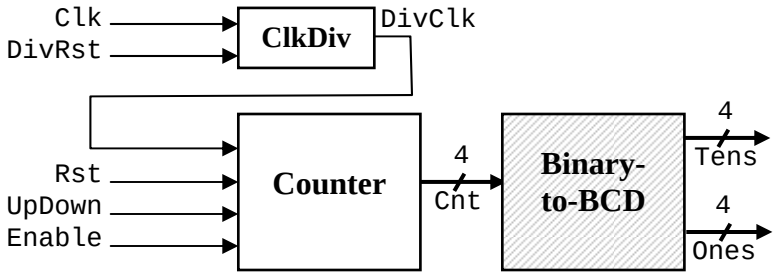
Starts: Mar 24 - Mar 28
Report/Code Due: Apr 11, 11:59PM

Lab Overview:

In your previous labs, you utilized a binary to 7-segment LED decoder to display a 4-bit binary number as a hexadecimal number. However, as the general public is not likely to be familiar with hexadecimal numbers, we may instead want to display the 4-bit number as a two digit decimal numbers (00 thru 15) using the two 7-segment LED displays. In this lab, you will design a *Binary to Binary Coded Decimal (BCD) Converter* and a *Multiplexed BCD Display Driver* to display the 4-bit output of your *Up/Down Counter* on the two 7-segment displays connected to the Spartan-3E FPGA board.

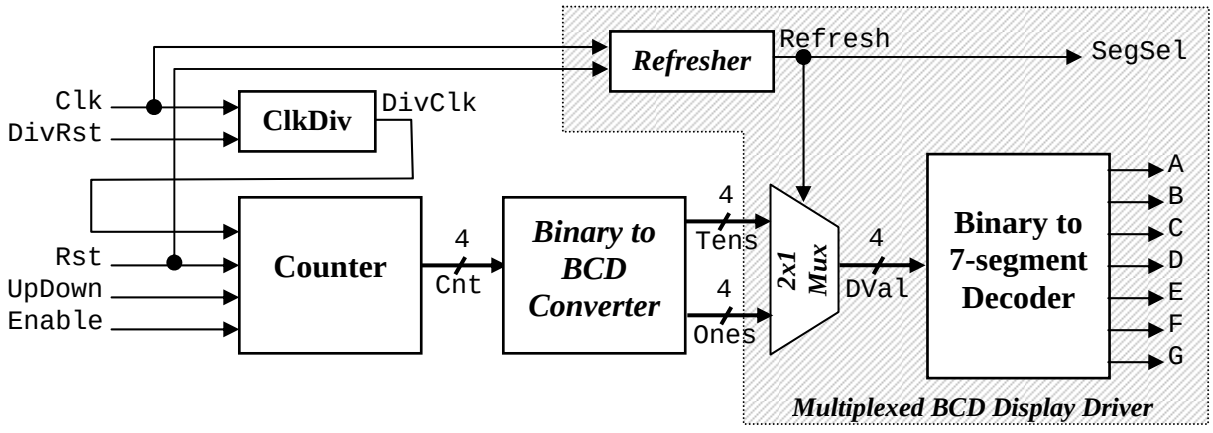
Binary-to-BCD Converter

In order to display the 4-bit binary number in decimal, you will first design a *Binary to BCD Converter*. A BCD number is a number that uses a 4-bit binary number to represent each decimal digit. For example, the binary number `1111` *(15 in decimal)*, can be encoded as BCD number as `0001 0101`. As such, the *Binary to BCD Converter* has a 4-bit input, `Cnt`, and two 4-bit outputs, `Tens` and `Ones`, corresponding to the binary representation of the tens and ones digit of the decimal equivalent, as illustrated in the following figure.

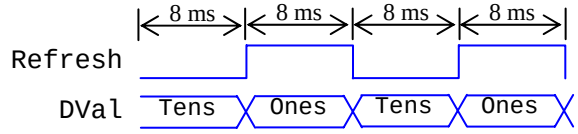


Multiplexed 2-digit BCD Display Controller

In designing the binary to 7-segment LED decoder in Lab 2, the `SegSel` output was used to control which 7-segment LED display would be utilized to display the 4-bit binary number. As such, we cannot simultaneously display a digit on both 7-segment LED displays. Instead, by repeatedly and continuously display a digit on each display faster than the human eye can respond, both displays will appear to be illuminated at the same time. In this lab, you will also design and build a *Multiplexed BCD Display Driver* to display the `Tens` and `Ones` outputs of the *Binary to BCD Converter* on the corresponding 7-segment LED displays. The *Multiplexed BCD Display Driver* builds upon your binary to 7-segment decoder by adding a refresher circuit to control when each 7-segment display will be illuminated and a multiplexer to select between the `Tens` and `Ones` output of the *Binary to BCD Converter*. The following provides an overview of the multiplexed BCD to 7-segment display driver.



For the 7-segment LED display connected to the Spartan-3E FPGA board, each 7-segment LED display should be illuminated for 8 ms, as illustrated in the following timing diagram. Given the 50 MHz clock provided by the Spartan-3E FPGA board, the *Refresher* circuit should generate an oscillating output, `Refresh`, that remains `1` for 8 ms, `0` for 8 ms, and repeats. The `Refresh` output will be used to control the `SegSel` output and as the select line to the multiplexer for selecting which BCD digit will be the input to your *Binary to 7-segment LED Decoder*. Please note that you will need to modify your *Binary to 7-segment LED Decoder* to remove the `SegSel` output, as the `SegSel` signal will be connected to the `Refresh` output of the *Refresher* circuit.



The final implementation will utilize one of the implementation of the *Up/Down Counter* designed in Lab 3, but replaces the *Binary to 7-segment LED Decoder* from your previous lab with the *Binary to BCD Converter* and *Multiplexed BCD Display Driver* components built in this lab. As such, when implemented on Spartan-3E FPGA Board, the overall top-level design will have the same inputs and output as the counter design from Lab 3 and can use the same User Constraint File (.ucf).

Lab Procedure & Demo

1. Structurally design the *Binary to BCD Converter* using any of the following datapath components: adders, subtractors, incrementers, decrementers, multipliers, comparators, shifters, registers, multiplexers, decoders, encoders, and logic gates *(only when necessary)*. Each datapath component used ***must*** be modeled behaviorally as a separate module, and the *Binary to BCD Converter* must be implemented as a structural connection of those datapath components. *Note that you do* ***not*** *need to utilize all components listed above, but rather you are restricted to those components (30 points).*

    a. Create a testbench to test the *Binary to BCD Converter* for correct functionality. No specific requirements are needed for the testbench, but you must be able to demonstrate the correctness of your design to your TA *(5 points).*

2. Structurally design the *Multiplexed BCD Display Driver* and *Refresher* sub-component using any of the above listed datapath components. Again, each datapath component used ***must*** be modeled behaviorally as a separate module, and the *Multiplexed BCD Display Driver* and *Refresher* must be implemented as a structural connection of those datapath components *(30 points)*.

    a. Create a testbench to test the *Multiplexed BCD Display Driver* for correct functionality for one full refresh period. Due to the memory limitations of the computers within the ECE 274 Laboratory, you should test your design assuming a refresh period is 16 us (instead of 16 ms). *Be sure to correct your design before synthesizing the circuit to the Spartan-3E FPGA board (5 points).*

3. Create a top-level component that structurally connects your *4-bit Up/Down Counter*, the previously provided clock divider (*ClkDiv*), the *Binary to BCD Converter*, and the *Multiplexed BCD Display Driver*. Synthesize and download your modified design to the Spartan-3E FPGA board and test your design for correct functionality *(20 points)*.

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

1. Block diagram for the *Binary to BCD Converter*, *Multiplexed BCD Display Driver*, and *Refresher* components clearly indicating the various datapath components utilized and their interconnections.
2. Simulation waveforms demonstrating correct functionality of the *Binary to BCD Converter*.

Code Turn-In Requirements

Note: Be sure to turn in all code needed to implement your final design, including the *Up/Down Counter* design, along with the corresponding .ucf file.