Oregon TECH

Computer Systems Engineering Technology
CST 345 – HW/SW Co-Design

Lab 02 – Developing Reusable Hardware
Winter 2015
Instructor: Troy Scevers
Possible Points: 20

Name_____
Due Date: Friday, January 16th @ 5pm

Instructions

This lab covers simple logic design using familiar building blocks. From your previous coursework, you should already be familiar with simple counters, multiplexers, decoders, and seven-segment displays. The goal of this lab is for you to implement a circuit that drives the time-multiplexed quad seven-segment display present on the Digilent Nexys3 board. When you successfully complete this lab, you will have developed a piece of intellectual property that you might be able to re-use in the future.

Now that you are familiar with the tools from Laboratory Assignment #1, you should be able to concern yourself with digital design. Figure 1 shows a symbol of the module you will create. The inputs are shown on the left and the outputs are shown on the right.

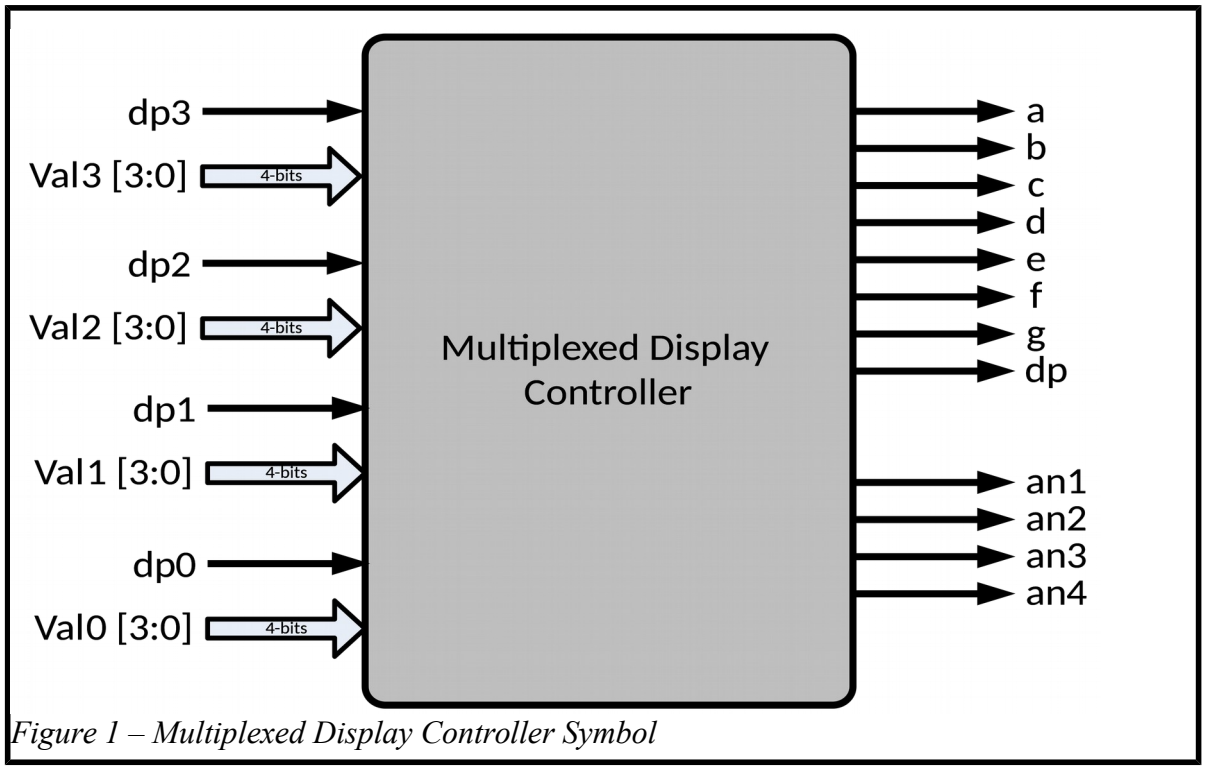

Figure 1 – Multiplexed Display Controller Symbol

The Nexys3 board contains a four-digit common anode seven-segment LED display. Each of the four digits is composed of seven segments arranged in a “figure 8” pattern, with an LED embedded in each segment. Segment LED’s can be individually illuminated, so any one of 128 patterns can be displayed on a digit by illuminating certain LED segments and leaving the others dark. Of these 128 possible patterns, the ten corresponding to the decimal digits are the most useful.

The anodes of the seven LED’s forming each digit are tied together into one “common anode” circuit node, but the LED cathodes remain separate. The common anode signals are available as four “digit enable” input signals to the 4-digit display. The cathodes of similar segments on all four displays are connected into seven circuit nodes labeled CA through CG (so, for example, the four “D” cathodes from the four digits are grouped together into a single circuit node called “CD”). These seven cathode signals are available as inputs to the 4-digit display. This signal connection scheme creates a multiplexed display, where the cathode signals are common to all digits but they can only illuminate the segments of the digit whose corresponding anode signal is asserted.

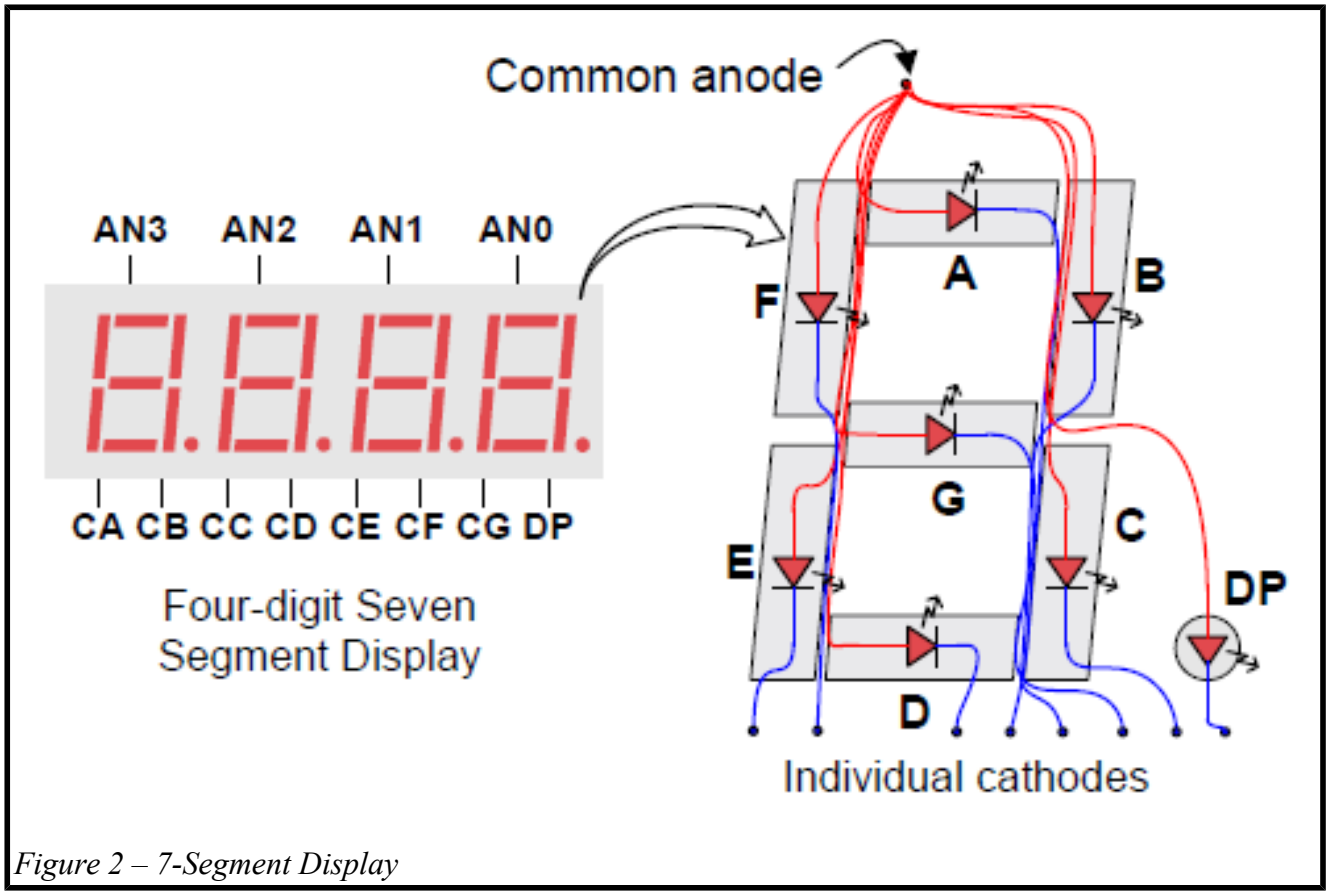

Figure 2 – 7-Segment Display

A scanning display controller circuit can be used to show a four-digit number on this display. This circuit drives the anode signals and corresponding cathode patterns of each digit in a repeating, continuous succession, at an update rate that is faster than the human eye can detect. Each digit is illuminated just one-quarter of the time, but because the eye cannot perceive the darkening of a digit before it is illuminated again, the digit appears continuously illuminated. If the update or “refresh” rate is slowed to around 45 hertz, most people will begin to see the display flicker.

In order for each of the four digits to appear bright and continuously illuminated, all four digits should be driven once every 1 to 16ms, for a refresh frequency of 1KHz to 60Hz. For example, in a 60Hz refresh scheme, the entire display would be refreshed once every 16ms, and each digit would be illuminated for ¼ of the refresh cycle, or 4ms. The controller must drive the cathodes with the correct

pattern when the corresponding anode signal is driven. To illustrate the process, if AN0 is asserted while CB and CC are asserted, then a “1” will be displayed in digit position 1. Then, if AN1 is asserted while CA, CB and CC are asserted, then a “7” will be displayed in digit position 2. If AN0 and CB, CC are driven for 4ms, and then A1 and CA, CB, CC are driven for 4ms in an endless succession, the display will show “17” in the first two digits. An example timing diagram for a four-digit controller is shown in Figure 3.

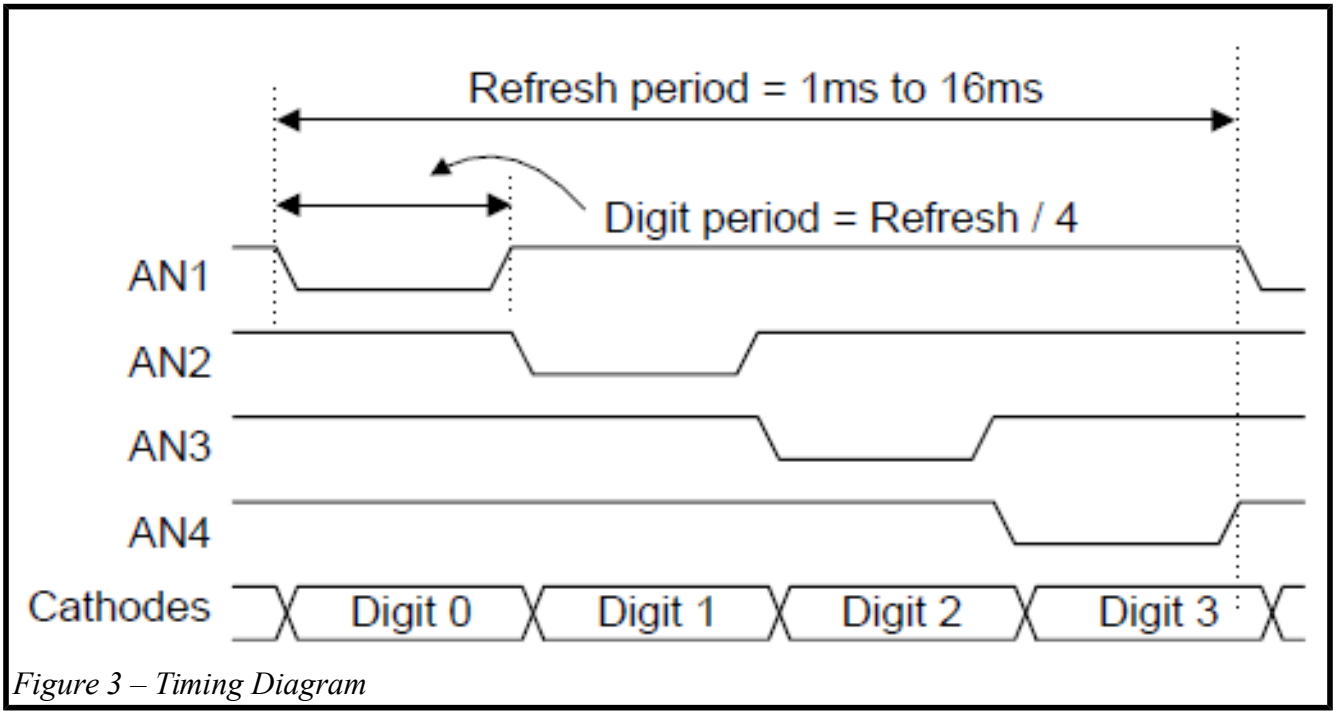

Figure 3 – Timing Diagram

As for the input data we will use the 5 push button switches on the Nexys3 board to determine which of four register will be latched with the data from the lower 5 slide switches. The following table illustrates how this works:

Push-button Switches	Data Register to Load
BTNL	Val0[3:0] gets SW[3:0], DP0 gets SW [4]
BTNR	Val1[3:0] gets SW[3:0], DP1 gets SW [4]
BTNU	Val2[3:0] gets SW[3:0], DP2 gets SW [4]
BTND	Val3[3:0] gets SW[3:0], DP3 gets SW [4]
BTNS	Reset so all LED Segments are off

Procedure

Step 1 – Create a Verilog directory called “z:\CST345\LAB 2” under your home directory (on Z drive). Create Verilog file called “Mux_disp.v” under your design directory.

Step 2 – Develop the hierarchical design for this project and begin writing the various modules in Verilog.

Step 3 – Use Xilinx ISE to compile and place & route your design. Select the device.

Step 4 – Simulate the design using the iSim Simulator.

Step 5 – Demonstrate the various tests for your multiplexed display controller on the Diligent Nexys3 Board.