ECE 2700 Lab 4

Due the week of February 23, 2015 at the beginning of your registered lab session (200 points)

Objective

To become proficient in designing more complex combinational circuits in Verilog.

1 Preparation

1. Read this document in its entirety.
2. Draw a circuit showing how you might build a 4×1 mux from 2×1 muxes.
3. Read Example 2.23 in the main textbook on pages 72–73 and prepare the truth table for a 4-bit binary number to **hexadecimal** 7-segment display. That is, in addition to 0-9, the display also shows A, b, C, d, E, and F.
4. Read the 2-page reference manual (attached to the end of this document) to better understand the seven-segment display connections.

2 Multiplexer Design

2.1 Building a 4×1 Mux

Inside your `ECE2700` directory, create a new folder called `Lab4`. Start up Xilinx and create a new project called `MuxExp` inside the `Lab4` directory.

Create a new file called `mux4x1.v`. Inside, design a 4×1 mux using the provided 2×1 mux (`mux2x1.v` on Canvas).

Be sure to declare your 4×1 mux as follows: `module mux4x1(I3, I2, I1, I0, S1, S0, D)`, where $I3$, $I2$, $I1$, and $I0$ are data inputs, $S1$ and $S0$ are control inputs, and D is the output. Make sure to design your mux such that when $S0 = 0$, $S1 = 0$, $D = I0$, and when $S0 = 1$, $S1 = 1$, $D = I3$.

Test your design using the bench file `mux4test.v` from Canvas. Show your results to the TA.

2.2 Using the Muxes

Create a new file called `shifter.v`. You are to design a simple shifter **using the 4×1 mux(es) you designed in earlier**. The shifter has four 1-bit data inputs $(i3, i2, i1, i0)$, two 1-bit control inputs $(s1, s0)$, and four 1-bit data outputs $(d3, d2, d1, d0)$. The following table describes the desired behaviors of the shifter. Be sure to declare your shifter as follows: `module shifter(i3, i2, i1, i0, s1, s0, d3, d2, d1, d0)`. Observe that $i3$ is the **leftmost** bit in the input.

$s1$	$s0$	Behavior	Description	Example
0	0	Hold	Output is the same as input.	Input: 1010, Output: 1010
0	1	Shift left	Output is the input shifted left by one bit. The least significant bit is shifted in with a 0.	Input: 0101, Output: 1010 Input: 1111, Output: 1110
1	0	Shift right	Output is the input shifted right by one bit. The most significant bit is shifted in with a 0.	Input: 0101, Output: 0010 Input: 1111, Output: 0111
1	1	Rotate Right	Output is the input rotated right by one bit.	Input: 1011, Output: 1101

Hint: Remember that a mux is simply a circuit that selects the output from one of the inputs. Think about what the data and control inputs need to be for the shifter to work correctly.

Test your design using the bench file `testshifter.v` from Canvas. Show your results to the TA.

3 Simple 7-Segment Display

For this part of the lab you will write code for the 7-segment display and load it onto the Basys board. Create a new project called `SevenSeg`. Add a new source file `seg7.v` and write out your binary number to hexadecimal 7-segment display using the following declaration: `seg7(`w`, `x`, `y`, `z`, `a`, `b`, `c`, `d`, `e`, `f`, `g`)`, where w, x, y, and z are the 4-bit binary input and a, b, c, d, e, f, and g are the outputs. (If you are unsure what to do, remember that you have prepared a truth table. From the truth table, you can convert to Boolean equations and write them directly in Verilog.) Run simulations to make sure your design is correct.

Locate `SW0`, `SW1`, `SW2`, and `SW3` on the Basys board. These switches will represent the 4-bit binary number input. As mentioned earlier, the output (corresponding hex value) will be displayed on the 7-segment display. From reading the reference manual and looking inside `MainBasys.ucf`, you will see that although the 7-segment display contains 4 digits, there are only 7 pin assignments for the 7-segment LEDs, not 28.

```
NET "seg<0>" LOC = "p25"; # Bank = 3, Signal name = CA
NET "seg<1>" LOC = "p16"; # Bank = 3, Signal name = CB
NET "seg<2>" LOC = "p23"; # Bank = 3, Signal name = CC
NET "seg<3>" LOC = "p21"; # Bank = 3, Signal name = CD
NET "seg<4>" LOC = "p20"; # Bank = 3, Signal name = CE
NET "seg<5>" LOC = "p17"; # Bank = 3, Signal name = CF
NET "seg<6>" LOC = "p83"; # Bank = 1, Signal name = CG
```

This is to reduce the number of pins on the board. You can determine which of the 4 digits (or all of them) are displayed by sending appropriate signals to the following anodes.

```
NET "an<3>" LOC = "p26"; # Bank = 3, Signal name = AN3
NET "an<2>" LOC = "p32"; # Bank = 3, Signal name = AN2
NET "an<1>" LOC = "p33"; # Bank = 3, Signal name = AN1
NET "an<0>" LOC = "p34"; # Bank = 3, Signal name = AN0
```

Be Careful! The inputs for the 7-segment LEDs and the anodes are **active low**. In other words, to display an 8 on the rightmost digit, you will set a, b, c, d, e, f, and g all to 0. In addition, you will set the input corresponding to `an<0>` to 0 and set all the other inputs to the anodes to 1.

Add a new file called `SevenSeg.v` with the following declaration: `SevenSeg(`w`, `x`, `y`, `z`, `a`, `b`, `c`, `d`, `e`, `f`, `g`, `$n3$`, `$n2$`, `$n1$`, `$n0$`)`. Here, $n0$ corresponds to the signal sent to anode 0 (`AN0`). Write necessary code so that only the rightmost digit lights up. Recall that you will need to add a `.ucf` file to indicate pin assignments. Generate the corresponding bit file, test your design and show the results to the TA.

4 TA Checkoff

- (20 points) Pre-lab.
- (40 points) 4×1 mux design.
- (60 points) Shifter design.
- (80 points) 7-segment display.

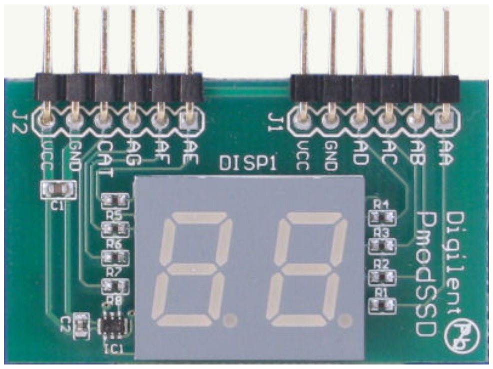
Digilent PmodSSD™ Peripheral Module Board Reference Manual

Revision: March 6, 2007

www.digilentinc.com
215 E Main Suite D | Pullman, WA 99163
(509) 334 6306 Voice and Fax

Overview

The PmodSSD offers a single two-digit seven-segment display device (7sd) that can attach directly to any Digilent system board. The 7sd uses high-bright LEDs that can are easily readable with less than 5mA of current, so they can be driven directly from most system boards.



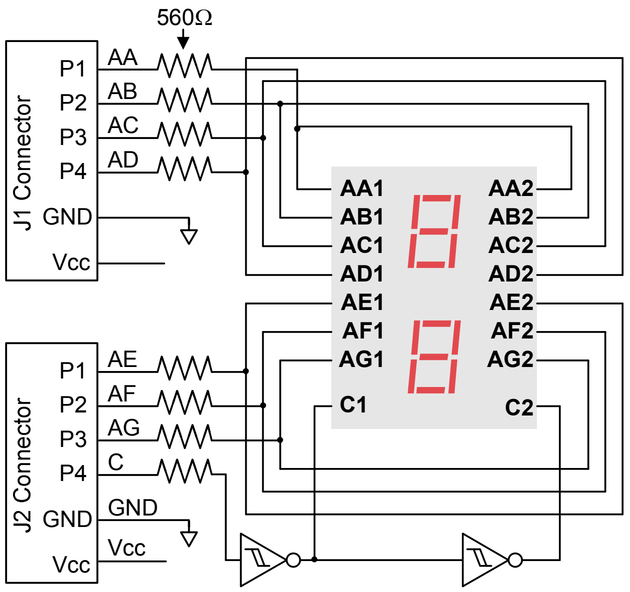
Features include:

- two high bright seven-segment displays
- a 6-pin system connector
- small form factor (0.80" x 0.80").

Functional Description

The two digits on the common cathode seven-segment LED display are each composed of seven segments arranged in a "figure 8" pattern, with an LED embedded in each segment. Segment LEDs 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.

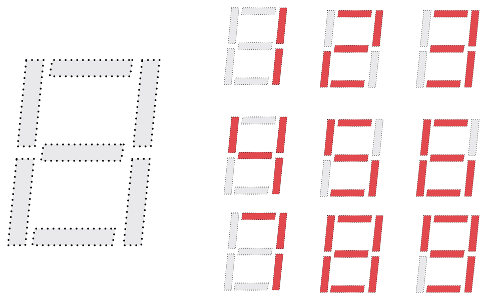
The cathodes of the seven LEDs forming each digit are tied together into one "common cathode" circuit node, but the LED anodes remain separate. The common cathode signals are available as two "digit enable" input signals to the display. The anodes of similar segments on both digits are connected into seven circuit nodes that are available from the Pmod connector pins (so, for example, the two "D" anode signals from the two digits are connected to the P4 pin on J1). These seven anode signals are available as inputs to the 2-digit display. This signal connection scheme



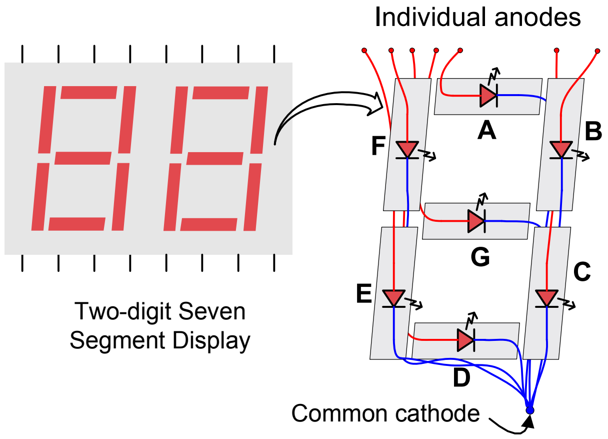
Seven-Segment Display Connection Diagram

creates a multiplexed display, where the anode signals are common to both digits but they can only illuminate the segments of the digit whose corresponding cathode signal is asserted.

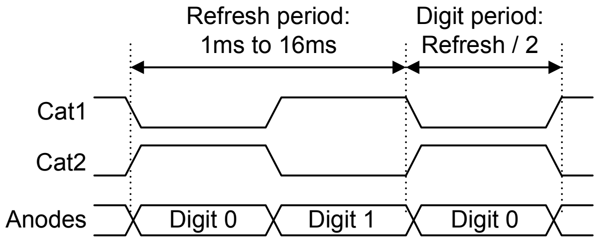
A scanning display controller circuit can be used to show a two-digit number on the display. This circuit drives the anode signals and corresponding cathode patterns of each digit in a repeating, continuous succession, at an update rate that is faster than the human eye can respond. Each digit is illuminated one-half 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 a given point (around 45 hertz), then most people will begin to see the display flicker.


An un-illuminated seven-segment display, and nine illumination patterns corresponding to decimal digits

In order for each of the four digits to appear bright and continuously illuminated, both 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 ½ of the refresh cycle, or 8ms. The controller must assure that the correct anode pattern is present when the corresponding cathode signal is driven. To illustrate the process, if Cat1 is asserted while AB and AC are asserted, then a "1" will be displayed in digit position 1. Then, if Cat2 is asserted while AA, AB and AC are asserted, then a "7" will be displayed in digit position 2. If Cat1 and AB, AC are driven for 8ms, and then Cat2 and AA, AB, AC are driven for 8ms in an endless succession, the display will show "17". An example timing diagram for a two-digit controller is shown below.



Two-digit Seven Segment Display


Refresh period: 1ms to 16ms
Digit period: Refresh / 2
Cat1
Cat2
Anodes
Digit 0
Digit 1
Digit 0