Andrew Scott
E155
Lab 2
September 21, 2015

# Lab 2: Multiplexed Display

**Introduction**

In this lab, I extended the functionality of the circuit I built last week to take input from an additional 4 bit DIP switch and to then display the hexadecimal numbers from each switch on the dual digit seven segment display. I did this while using only one seven segment display module in my Verilog code and one set of pins and resistors to control the seven segment display, which was made possible by switching off back and forth between which display was being controlled at a speed too fast for the human eye to detect.

**Design and Testing Methodology**

The main software decision to be made in this lab was how often to switch between controlling the two displays. From a quick Google search I was able to determine that the human eye sees at a rate of about 60 frames per second. With a 40 MHz clock cycle, we would need to switch every 666,666 cycles to be right at the edge of what a human could see. I decided to play it safe and chose to switch every $2^{17}$ cycles, which is every 131,072 cycles. I chose $2^{17}$ because in doing so I could have an 18 bit wide counter and switch based on the most significant bit. I tested that my time multiplexing design was working properly by simulating in Verilog with a testbench. After forcing my counter to be 0, I then forced the two switch inputs to be different values. I then checked that the output controlling the seven segment display changed after my counter got large enough, and that it changed at the same time as my signals controlling which digit of the display was lit up. Finally, I ran the simulation longer to make sure that all the signals changed back after the counter got large enough.

On the hardware side, I decided to use a single set of resistors and pins from the utility board to control both of the seven segment displays. Originally I was thinking of having a pair of flops to hold which segments to light up for each of the digits in the display, then having each display with its own set of resistors and pins on the utility board, but after some thought I realized that this was unnecessary. Since I am controlling which digit of the display has power at all times and only one digit is on at a time, I can always have the pins from the utility board driving the proper values for the digit that is lit up. This realization allowed me to save both 14 registers in the Verilog code and 7 resistors on the breadboard. I tested my circuit by running it and making sure that everything worked as expected. I also experimented with changing the length of time between when I switched which digit I was controlling. When I made the time too small, both of the digits displayed the output from one switch, and when I made the time too long the display flickered. This reinforced my belief that I had picked a good clock cycle.

**Results and Discussion**

I was able to finish this entire lab, and everything appears to be working as it is supposed to. The one thing I wish I did differently was planning a little more before putting together my implementation on the breadboard. I had already placed wires to control each digit on the display with separate pins from the utility board before I realized I could make a more efficient design.

**Conclusion**

I spent about 7 hours on this lab. Probably the most difficult part for me was understanding how the PNP transistors work that I had to use. The explanation in class of NPN transistors confused me somewhat, particularly since it didn't occur to me that I was using PNP as opposed to NPN.

**Breadboard Schematic**

## Verilog Code

```
/*
Andrew Scott
ascott@hmc.edu
Created on September 17, 2015

This is the top level module for e155 lab 2. It takes input from two sets
of switches and then controls two seven segment displays from those swiches. It
also output the sum of the two switches, which can be displayed on 5 LEDs.

*/
module lab2_as(input logic clk,
            input logic [3:0] s0, s1,
            output logic [6:0] seg,
            output logic [4:0] sum,
            output logic seg0On, seg1On);
logic [3:0] switchIn;
logic [17:0] count;
logic switchSelect;
always_ff @(posedge clk)
begin
    count <= count + 1'b1;
end

assign switchSelect = count[17];
assign sum = s0 + s1;
assign switchIn = switchSelect ? s1 : s0;
assign seg0On = switchSelect;
assign seg1On = ~switchSelect;
seven_seg_disp display(switchIn, seg);

endmodule

/*
Andrew Scott
ascott@hmc.edu
Created on September 14, 2015

This module contains the logic for controlling a common anode seven segment
display with a four bit input. The module uses a truth table  to determine when
to light up each segment of the display.
*/
module seven_seg_disp(input logic [3:0] s,
                    output logic [6:0] seg);

always_comb
case(s)
    4'b0000 : seg = 7'b1000000;
    4'b0001 : seg = 7'b1111001;
    4'b0010 : seg = 7'b0100100;
    4'b0011 : seg = 7'b0110000;
    4'b0100 : seg = 7'b0011001;
    4'b0101 : seg = 7'b0010010;
    4'b0110 : seg = 7'b0000010;
    4'b0111 : seg = 7'b1111000;
    4'b1000 : seg = 7'b0000000;
    4'b1001 : seg = 7'b0011000;
    4'b1010 : seg = 7'b0001000;
    4'b1011 : seg = 7'b0000011;
    4'b1100 : seg = 7'b0100111;
    4'b1101 : seg = 7'b0100001;
    4'b1110 : seg = 7'b0000110;
    4'b1111 : seg = 7'b0001110;
    default : seg = 7'b0000000;
endcase
endmodule
```