Sherman Lam

E155

September 22, 2014

Lab 2 Report: Time-Multiplexed 7-Segment Displays

1 Introduction

In this lab I displayed two 7-segment displays while only using one 7-segment decoder. This is was done by time-multiplexing the two displays. With this technique, one switch is read, decoded, and shown on its corresponding 7-segment display. Then, then the second switch is read, decoded, and shown on its corresponding 7-segment display. Only one 7-segment display is on at a time. The FPGA rapidly switches between the two sets of switches and displays at a rate fast enough to appear steady to the human eye. In addition, the sum of the two switches is displayed on the LED bar on the μMudd.

Figure 1, shows the result of the lab. The DIP switches on the breadboard control the left 7-segment display (currently set to 4) and the DIP switches on the μMudd control the right 7-segment display (currently set to 2). The sum 6 is displayed on the LED bar using the 5 bottom LEDs. The least significant bit is at the bottom.
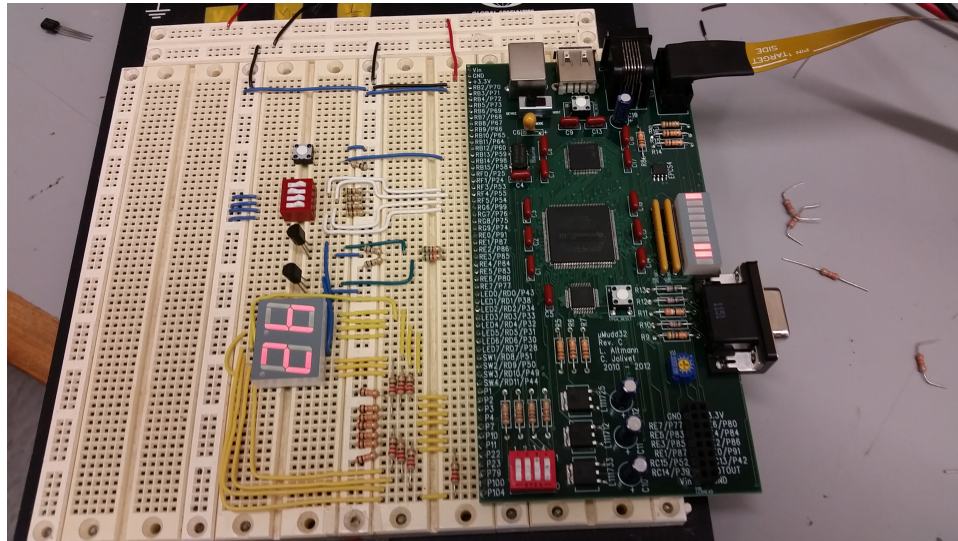

Figure 1: Assembled board.

2 Design and Testing Methodology

2.1 Hardware

A PNP transistor (2N3906) was chosen for toggling the displays because the switch was acting on the anode. The general circuit for using a PNP transistor for switching purposes in shown in Figure 2. If the transistor's based voltage (V_B) is brought close to GND, the PNP will turn on. If V_B is brought close to VCC, the transistor will turn off. However,

since BJT transistors are current controlled, a resistor (R_B) must be placed between V_B and its signal pin (V_{signal}). To select a value for this resistor, I first considered the largest current draw expected (I_C).

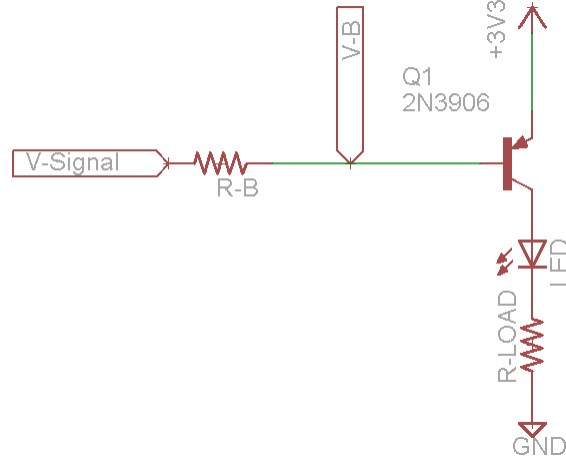Figure 2: General use of a PNP transistor for switching a load

$$I_C = (\#segments) * (I_{1segment})$$
$$I_C = (\#segments) * (\frac{V_{CC} - V_{led}}{R})$$
$$I_C = (\#segments) * (\frac{3.3V - 1.7V}{220\Omega})$$
$$I_C = 7 * (7.3mA)$$
$$I_C = 51mA$$

Then, I considered how much base current (I_B) I need in order to supply I_C. The transistor DC current gain (denoted β or h_{FE}) was approximated from Figure 3 to be 120. So, I_B was found to be:

$$I_B = \frac{I_C}{\beta}$$
$$I_B = \frac{51mA}{120}$$
$$I_B = 0.4mA$$

We can then find the resistor need to supply this voltage when the signal pin is pulled LOW. Because this is a silicon transistor, $V_{EB} = 0.7V$. Note that R was rounded to the nearest resistor value available in the lab. This is okay because the the value of R is not critical. It simply needs to be high enough to prevent the transistor from burning out (if V_EB is too high) and low enough to allow sufficient current to light up the LEDs. If the value of the R is lower than ideal, the transistor will saturate but still function.

$$R = \frac{Vcc - V_{EB}}{I_B}$$
$$R = \frac{3.3 - 0.7}{0.4mA}$$
$$R = 6.1k\Omega$$
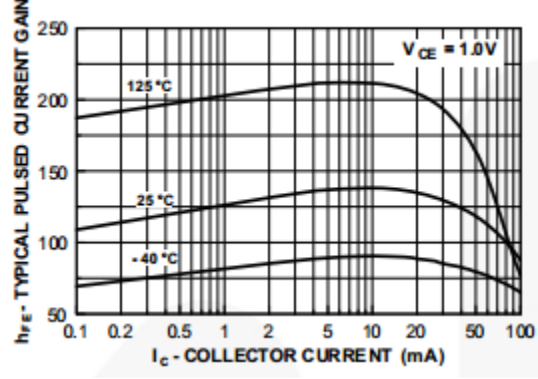$$R \approx 5.6k\Omega$$

Figure 3: DC gain of 2N3906 transistor. Image obtained from 2N3906 datasheet.

PNP transistors were also used because MOSFETs were not available.

2.2 Software

Table 1 shows how the switch inputs map to the 7-segment outputs.

Figure 4 shows the logic flow of the program. on1 is the control signal that decides whether or not the first 7-segment display is on (the first display is set to the left display). Note, because the hardware requires on1 to be pulled LOW in order for the PNP transistor to turn on, s1[3:0] is selected when on1 is 0. Since only one display is on at at time, on2 (the control signal for the second display) is the bitwise NOT of on1. s3 is the value on the selected switch and is passed through the 7-segment decoder to obtain the state of each segment in the display. Even though similar cathodes of both displays share the same state, only the segments in one display will be turned on (as determined by the states of on1 and on2). The sum of both switches are simply passed through an adder to obtain a 5-bit sum. This sum is directly written to the LED bar. See Tables 2, 3, 4, 5, and 6 for the pin mappings of each signal.

To flash a display, the display needs to be toggled twice: once to turn on, and once to turn off. Thus, in order for the program to flash the segments at 60Hz, each segment needs to be toggled at 120Hz. The on-board clock oscillates at 40MHz so the LED displays need to be toggled every $\frac{40MHz}{120Hz} \approx 333333$ clock cycles. The 60Hz flashing rate was experimentally determined to be successful at deceiving the human eye.

7-Segment Display Truth Table												
Inputs					Ouputs							
s[3]	s[2]	s[1]	s[0]	(hex)	G	F	E	D	C	B	A	(hex)
0	0	0	0	0x0	1	0	0	0	0	0	0	0x40
0	0	0	1	0x1	1	1	1	1	0	0	1	0x79
0	0	1	0	0x2	0	1	0	0	1	0	0	0x24
0	0	1	1	0x3	0	1	1	0	0	0	0	0x30
0	1	0	0	0x4	0	0	1	1	0	0	1	0x19
0	1	0	1	0x5	0	0	1	0	0	1	0	0x12
0	1	1	0	0x6	0	0	0	0	0	1	0	0x02
0	1	1	1	0x7	1	1	1	1	0	0	0	0x78
1	0	0	0	0x8	0	0	0	0	0	0	0	0x00
1	0	0	1	0x9	0	0	1	1	0	0	0	0x18
1	0	1	0	0xA	0	0	0	1	0	0	0	0x08
1	0	1	1	0xB	0	0	0	0	0	1	1	0x03
1	1	0	0	0xC	0	1	0	0	1	1	1	0x27
1	1	0	1	0xD	0	1	0	0	0	0	1	0x21
1	1	1	0	0xE	0	0	0	0	1	1	0	0x06
1	1	1	1	0xF	0	0	0	1	1	1	0	0x0E

Table 1: Truth table for 7-Segment LED decoder
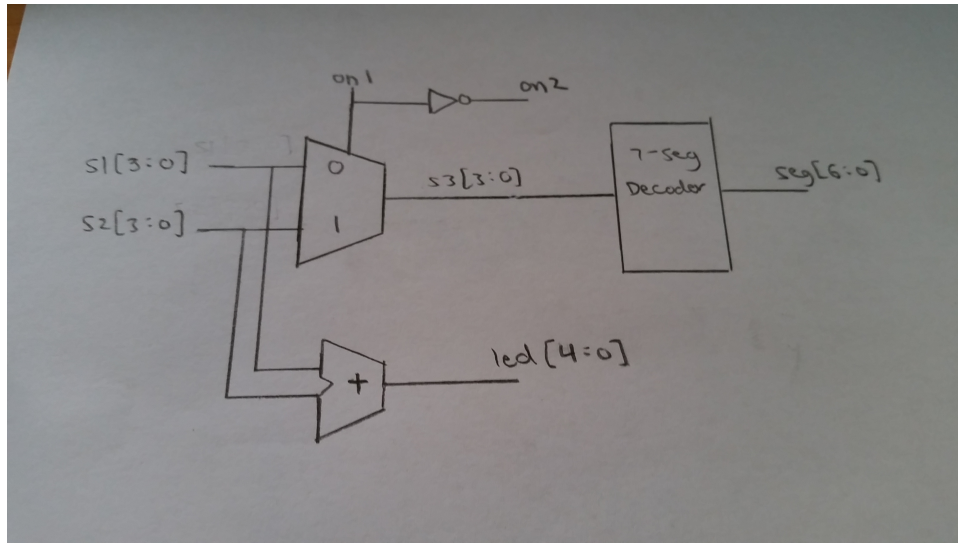
Figure 4: Logic flow of time multiplexer

LED Bar Pin Mapping				
led[4]	led[3]	led[2]	led[1]	led[0]
P33	P32	P31	P30	P28

Table 2: Pin mapping of LED bar

7 - Segment Display Pin Mapping						
seg[0]/A	seg[1]/B	seg[3]/C	seg[4]/D	seg[5]/E	seg[6]/F	seg[7]/G
P2	P1	P4	P10	P11	P3	P7

Table 3: Pin mapping of 7-segment display

Control Signals Pin Mapping			
on1	on2	clk	reset
P87	P86	P88	P60

Table 4: Pin mapping of control signals

Switch 1 Pin Mapping			
s1[3]	s1[2]	s1[1]	s1[0]
P24	P53	P55	P54

Table 5: Pin mapping for DIP switch set 1

Switch 2 Pin Mapping			
s2[3]	s2[2]	s2[1]	s2[0]
P51	P50	P49	P44

Table 6: Pin mapping for DIP switch set 2

2.2.1 Simulation

The code's logic was tested in ModelSim-Altera. The following show the results of the wave simulations that were run. The program was simulated in sections to ease debugging and wave simulation viewing.
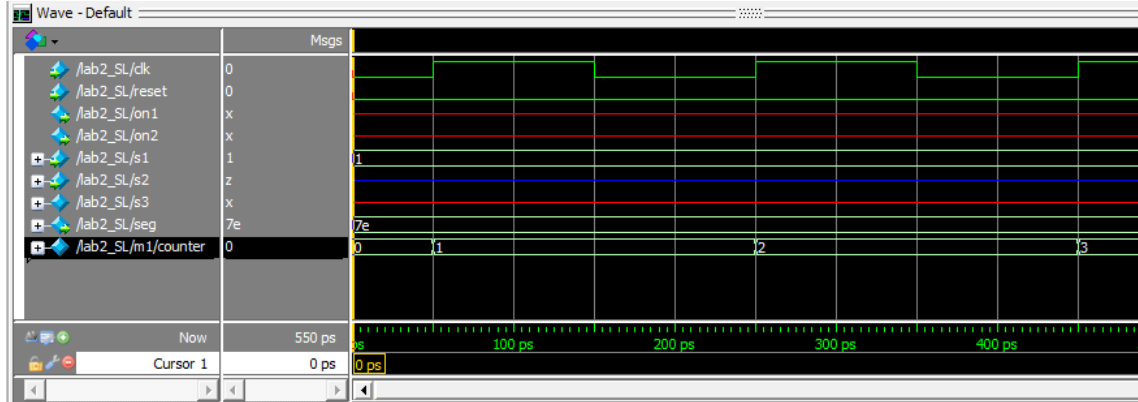

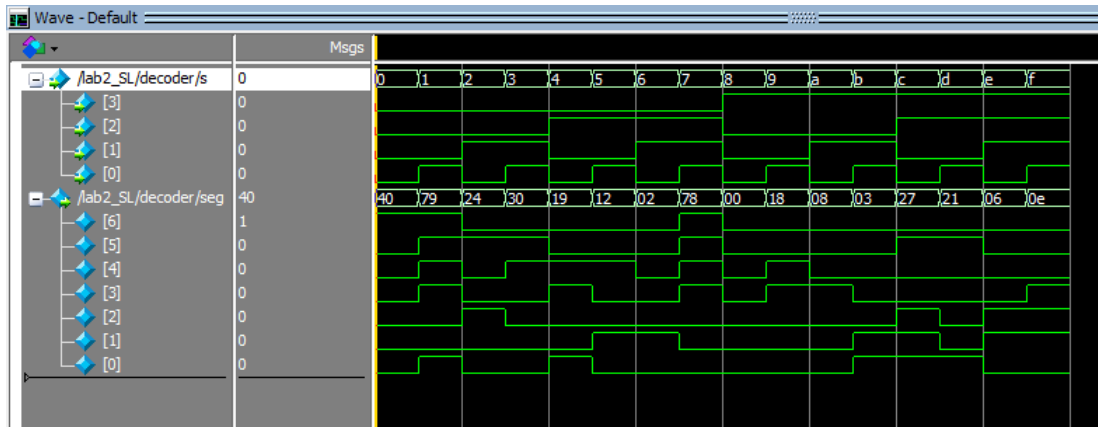Figure 5: The counter increments at every clock cycle.
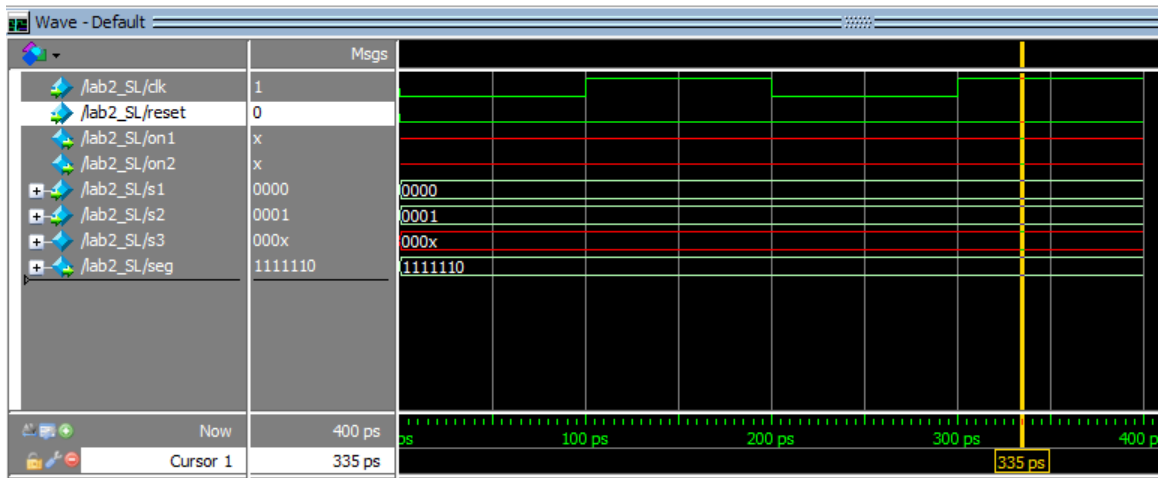

Figure 6: Logic for 7-segment decoder.

Figure 7: If the reset signal is never HIGH, the control signals for toggling the 7-segment displays (on1 and on2) will not resolve to a logic level.
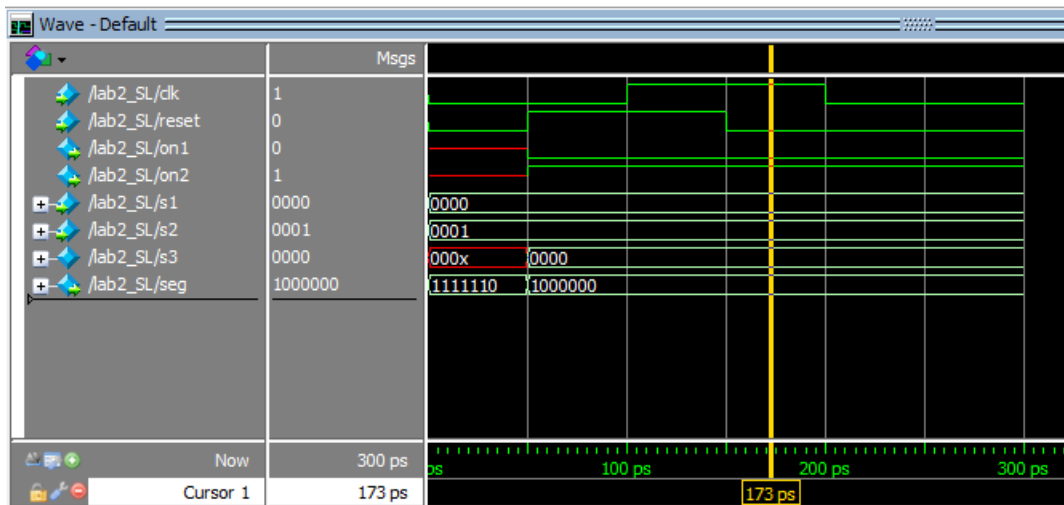

Figure 8: Once the reset signal is HIGH, the control signals for toggling the 7-segment displays (on1 and on2) are initialized to hard-coded starting values.
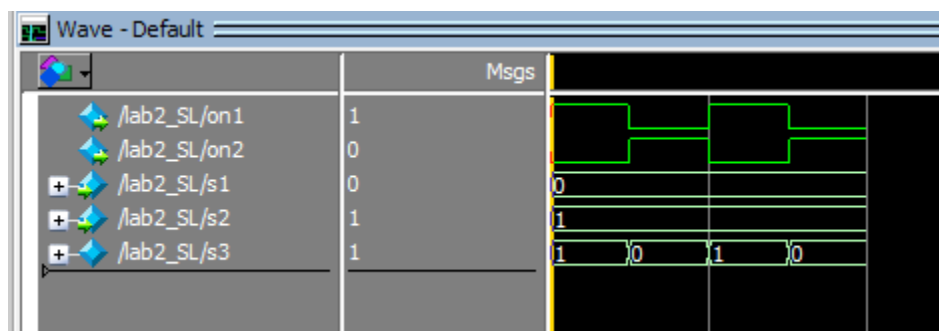

Figure 9: When on1 and on2 will swap states, s3 (the signal used for the 7-segment decoding) changes from s1 to s2 or vice versa.

7

Figure 10: The summation of the two input switch values (s1 and s2) is sent to the led bar (led[4:0]).

3   Technical Documentation

The following section shows schematics for breadboard circuit that was built. The source code is also provided.
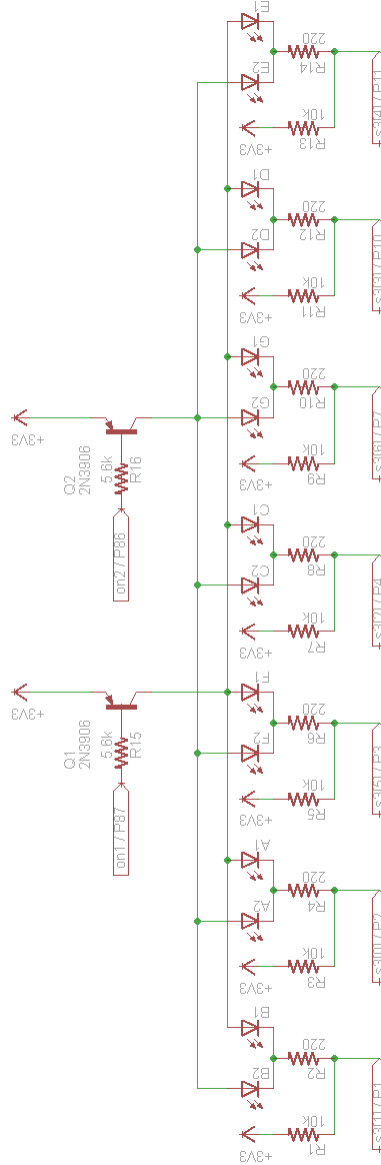
3.1   7-segment Displays Schematic

Figure 11: Full schematic for dual 7-segment display. Note that on1 and on2 toggle the two displays on and off. Only one or the other is on at any given time.
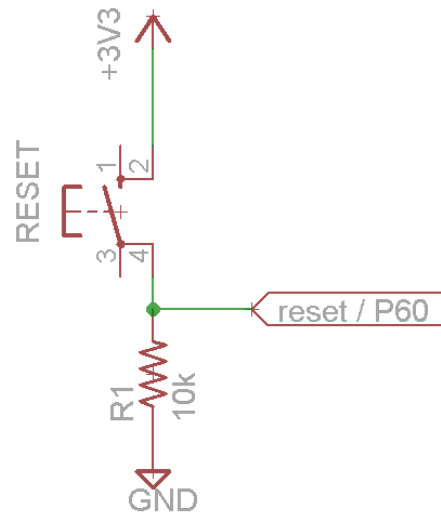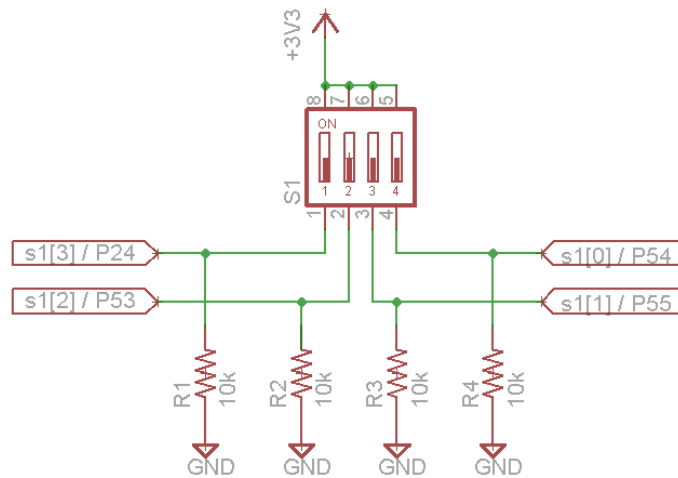
Figure 12: Schematic for reset button.

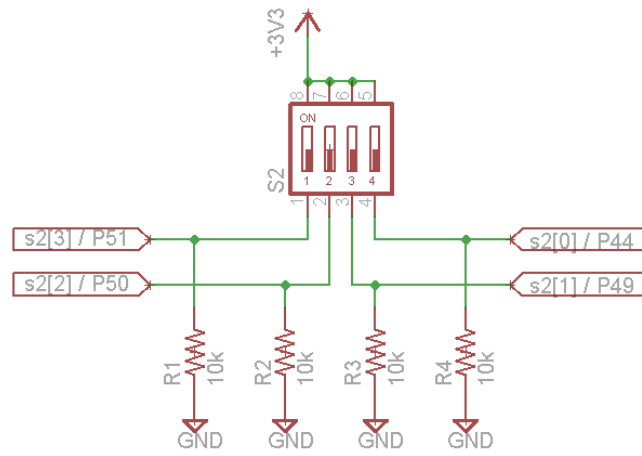Figure 13: Schematic for DIP switch 1. This controls display segment 1 (left).

Figure 14: Schematic for DIP switch 2. This controls display segment 2 (right).

3.2 System Verilog Code

```
1
2   /* This is the main module. It selects which set of switch
3      outputs to use and then decodes the number of the selected
4      switch. This also sets the clock that time−multiplexes the
5      two 7 segment outputs.
6
7      Author: Sherman Lam
8      Email: slam@g.hmc.edu
9      Date: Sep 17, 2014
10  */
11  module lab2_SL(input logic clk, reset,
12                  input logic [3:0] s1,s2, //DIP switches
13                  output logic on1, on2,    //if on1 is pulled LOW, LED set 1 is on.
14                  output logic [6:0] seg,
15                  output logic [4:0] led); //segment states
16
17      // time multiplexing
18      multiplexer m1(.clk(clk), .on1(on1), .reset(reset));
19
20      // the segments always have opposite states.
21      assign on2 = ˜on1;
22
23      // select the right set of switches.
24      // on1 −> s1 is used. on2 −> s2 is used
25      // if on1 is pulled LOW, LED set 1 is on.
26      logic [3:0] s3;
27      assign s3 = on1? s2 : s1;
28
29      // 7 segment decoder
30      led7Decoder decoder(.s(s3), .seg(seg));
31
32      // sum the outputs and write to LED bar
33      assign led = s1 + s2;
34
35
36  endmodule
37
38
39  /* This module time multiplexes by toggling on1.
40
41      Author: Sherman Lam
42      Email: slam@g.hmc.edu
43      Date: Sep 17, 2014
44  */
45  module multiplexer(  input logic clk, reset,
46                       output logic on1);
47      // time multiplexer for switching bewteen displays
48      logic [18:0] hPeriod = 19'd333333;  // 120Hz toggling
49      logic [18:0] counter = 'b0;
50
51      always_ff @(posedge clk, posedge reset) begin
52          if (reset)
53              on1 = 1'b0;
54          else begin
55              if (counter >= hPeriod) begin
56                  counter = 'b0;
57                  on1 = ˜on1;
```

```
58              end
59            else
60                //on1 = on1;
61                counter <= counter + 1'b1;
62          end
63       end
64
65   endmodule
66
67
68   /* This module decodes the switch inputs into an output for the
69      7 segment display on the development board.
70      s[3:0] = [sw3, ... ,sw1]
71      seg[6:0] = [g,f, ... ,b,a]
72
73      Author: Sherman
74      Email: slam@g.hmc.edu
75      Date: Sep 9, 2014
76   */
77   module led7Decoder(  input  logic [3:0] s,        //4 DIP switches
78                        output logic [6:0] seg);    //segments in 7-seg display
79
80       always_comb begin
81          //lookup table for s-seg relationship
82          case(s)
83              4'h0: seg = 7'b100_0000;        // 0x0
84              4'h1: seg = 7'b111_1001;        // 0x1
85              4'h2: seg = 7'b010_0100;        // 0x2
86              4'h3: seg = 7'b011_0000;        // 0x3
87              4'h4: seg = 7'b001_1001;        // 0x4
88              4'h5: seg = 7'b001_0010;        // 0x5
89              4'h6: seg = 7'b000_0010;        // 0x6
90              4'h7: seg = 7'b111_1000;        // 0x7
91              4'h8: seg = 7'b000_0000;        // 0x8
92              4'h9: seg = 7'b001_1000;        // 0x9
93              4'ha: seg = 7'b000_1000;        // 0xA
94              4'hb: seg = 7'b000_0011;        // 0xB
95              4'hc: seg = 7'b010_0111;        // 0xC
96              4'hd: seg = 7'b010_0001;        // 0xD
97              4'he: seg = 7'b000_0110;        // 0xE
98              4'hf: seg = 7'b000_1110;        // 0xF
99              default: seg = 7'b111_1110;      // default to a dash
100         endcase
101
102      end
103  endmodule
```

4   Results and Discussion

The system works as expected. The number set by the DIP switches on the breadboard is displayed on the left 7-segment display. The number set by the DIP switches on the μMudd is displayed on the right 7-segment display. The FPGA flashes both displays at 60Hz. This flashing is unnoticeable when viewed by the human eye. Since each display is operating at 50% duty cycle, the intensity of the display is about half that of a display being held on (100% duty cycle). In addition, when the reset button is pressed, display 1 (left) turns on and display 2 (right) turns off. This states is held until the reset button is released. The sum of the values on the two switches is correctly displayed on the LED bar.

5   Conclusion

5.1   Time Spent

Programming, Simulating 2.5hrs

Breadboarding 2hrs

Writing Report 5hrs

Total Time Spent 9.5hrs

5.2   Suggestions for lab

The current lab asks the student to use the DIP switches on the μMudd in addition to 4 wires to control the two 7-segment displays. However, many people were asking if 4 wires meant an extra set of DIP switch. The instructions seem confusing in regards to this detail. Unless the lab's goal is to purposefully give semi-ambiguous instructions in order to force students to think (channeling E80, eh?), I would suggest explicitly give instructions to use a second set of DIP switches.

Also, using MOSFETs may making the switching much easier. Maybe you could consider stocking the E155 Lab with some?