Alex Heiler
Francesca Narea
ELEN 21L Friday 2:15
March 10, 2017

Laboratory #7: Slot Machine Inspired Game

Introduction

In this lab, we designed a matching game inspired by slot machines. Further expanding our knowledge of the Altera Quartus II software, we used the design wizard to create a comparator and two counters, one that counted up and one that counted down. We used 7 segment displays to show our custom game symbols and display whether the player won, lost, or was still playing the game. Two switches on the Intel FPGA controlled the difficulty of the game, changing the speed the symbols changed.

Prelab

```
module bin_7seg(bi_digit,seg);
input [3:0] bi_digit;
output [6:0] seg;
reg [6:0] seg;
// seg = {g,f,e,d,c,b,a};


always @ (bi_digit)
case (bi_digit)
        4'h0: seg = ~7'b0100010; //Spec sym 1
        4'h1: seg = ~7'b1001000; //Spec sym 2      // ---a----
        4'h2: seg = ~7'b0010100; //Spec sym 3      // |      |
        4'h3: seg = ~7'b1000001; //Spec sym 4      // f      b
        4'h4: seg = ~7'b       ;                   // |      |
        4'h5: seg = ~7'b       ;                   // ---g----
        4'h6: seg = ~7'b       ;                   // |      |
        4'h7: seg = ~7'b       ;                   // e      c
        4'h8: seg = ~7'b       ;                   // |      |
        4'h9: seg = ~7'b       ;                   // ---d----
        4'ha: seg = ~7'b       ;
        4'hb: seg = ~7'b       ;
        4'hc: seg = ~7'b       ;
        4'hd: seg = ~7'b1100010; //Win
        4'he: seg = ~7'b0111000; //Lose
        4'hf: seg = ~7'01111111; //In progress
endcase

endmodule
```

Figure 1.1: Alex's Prelab pg 1

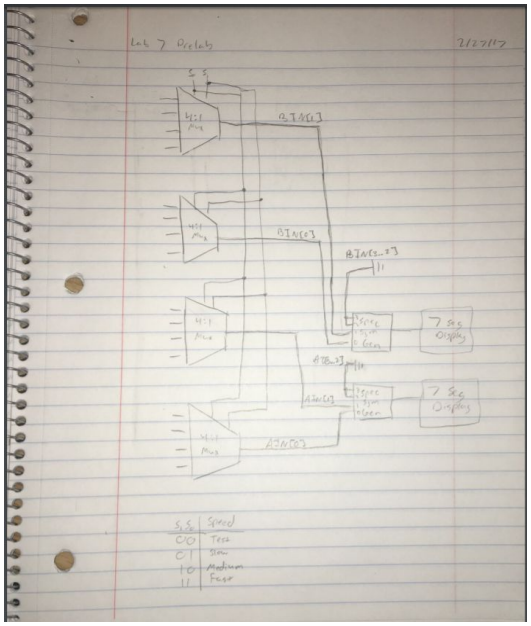

Figure 1.2: Alex's Prelab pg 2

Prelab #7
Francesca Narea

Part 1

Special Symbols:

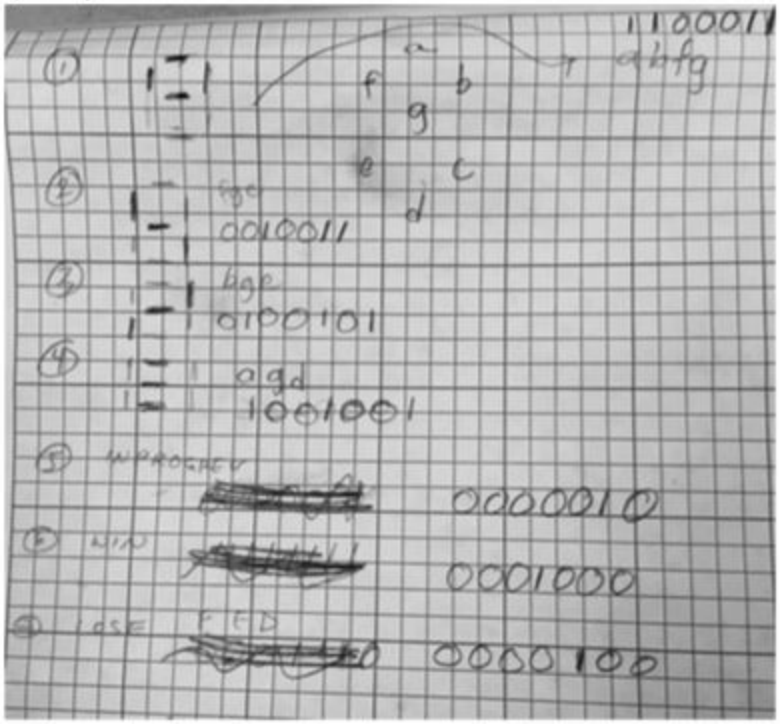

Figure 1.3: Francesca's Prelab (page 1)

Bin Seg File:

```
module bin_7seg(bi_digit,seg);
input [3:0] bi_digit;
output [6:0] seg;
reg [6:0] seg;
// seg = {g,f,e,d,c,b,a};

always @ (bi_digit)
case (bi_digit)
        4'h0: seg = ~7'b1100011;                        // ---a----
        4'h1: seg = ~7'b0010011;                        // |      |
        4'h2: seg = ~7'b0100101;                        // f      b
        4'h3: seg = ~7'b1001001;                        // |      |
        4'h4: seg = ~7'b0001000; //win                  // ---g----
        4'h5: seg = ~7'b0000100; //lose                 // |      |
        4'h6: seg = ~7'b0000010; //in progress          // e      c
        4'h7: seg = ~7'b        ;                       // |      |
        4'h8: seg = ~7'b        ;                       // ---d----
        4'h9: seg = ~7'b        ;
        4'ha: seg = ~7'b        ;
        4'hb: seg = ~7'b        ;
        4'hc: seg = ~7'b        ;
        4'hd: seg = ~7'b        ;
        4'he: seg = ~7'b        ;
        4'hf: seg = ~7'b        ;
endcase

endmodule
```

Part 2
(see next page for schematic)

Figure 1.4: Francesca's Prelab (page 2)

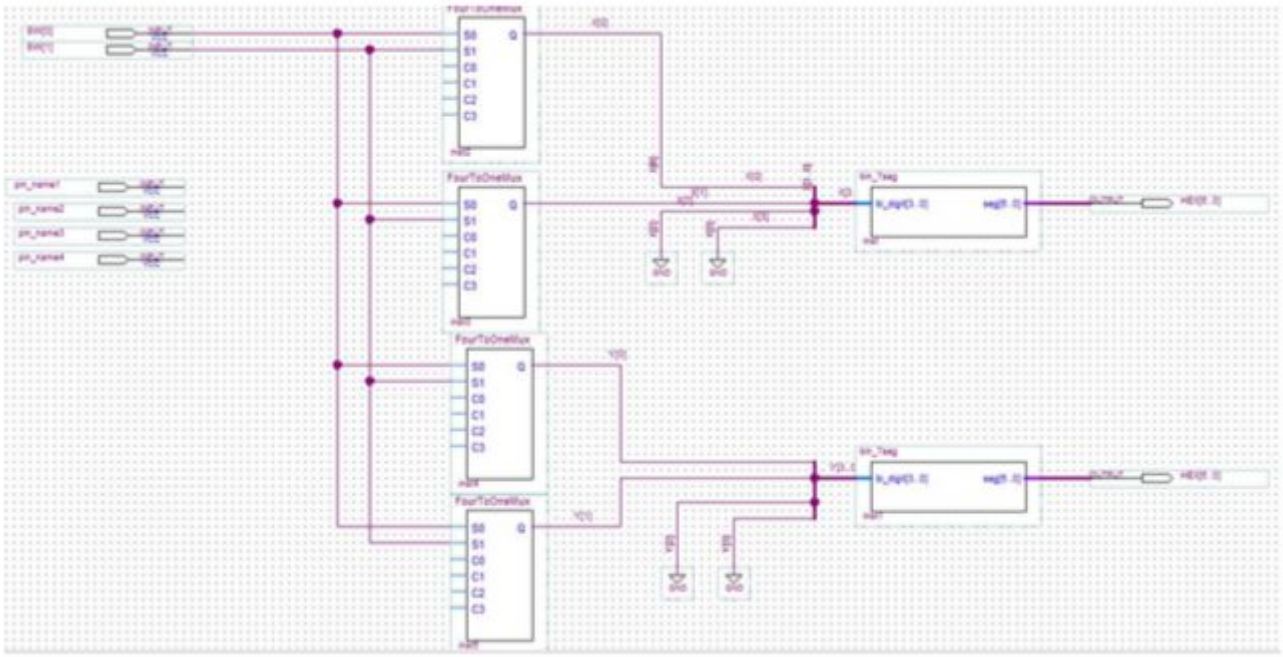

Figure 1.5: Francesca's Prelab (page 3)

Switch settings:
Test: 00
Slow: 01
Medium: 10
High: 11

The two most significant bits of the special symbol generator should be connected to GND since the greatest unsigned number is 0011 (3).

Figure 1.6: Francesca's Prelab (page 4)

Procedures

To start this lab, we first created the comparator, up counter, and down counter using the logic component design wizard. We connected the counters' enables to a push button to start the game and the clock input to the FPGA's clock 50. Various bits of the the clock outputs were connected to the inputs of four 4:1 muxes, allowing the speed of the game to be controlled by two switches. Our special symbol generators converted the mux outputs into the game symbols. The comparator and subsequent logic gates determined whether the player won, lost, or was still playing the game, and displayed the progress symbols on a 7 segment display. The overall circuit design is shown in Fig 3.

After downloading the circuit onto the FPGA, we first tested the game on the "test" setting, which changed the symbols slow enough to allow us to easily check if the game was functioning correctly. We then played on "slow," "medium," and "fast" settings, ensuring the game symbols cycled at increasing speeds.

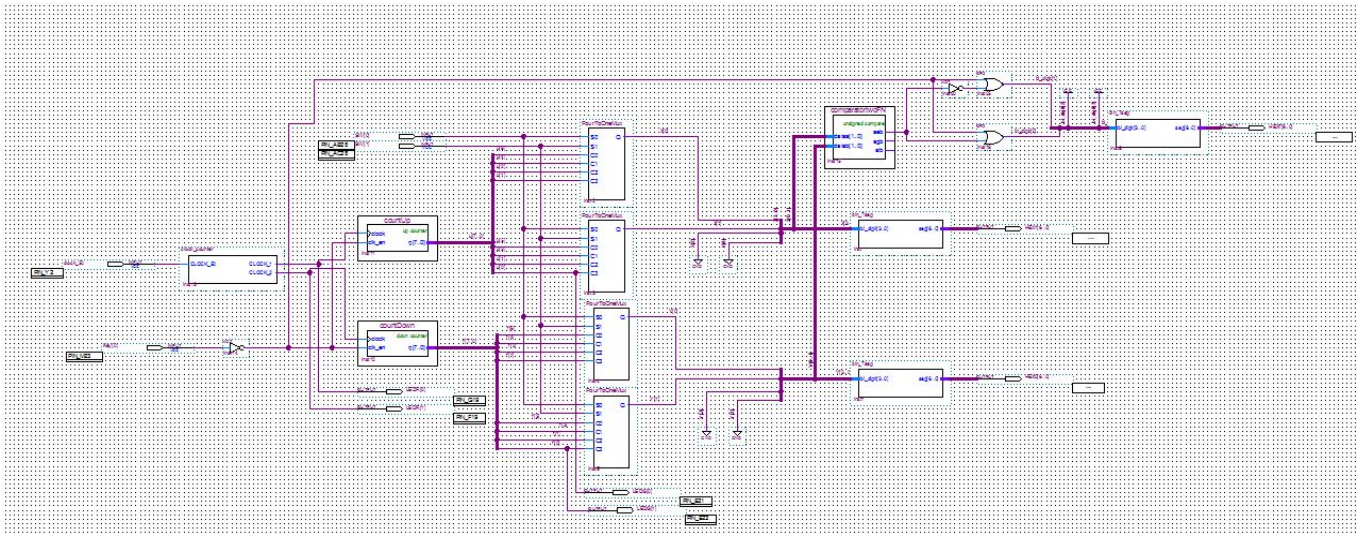

Figure 2: Circuit Schematic

Test Plan

a. Verify that the counters are NOT counting when the pushbutton is NOT depressed.

Test: Do not press pushbutton.
Expected: No symbols displayed.
Result: Verified expected.

b. Verify that both counters are counting when the pushbutton is pressed.

Test: Press pushbutton on 01 mode (slow).
Expected: Each of the 4 special symbols will be displayed.
Result: Verified expected.

c. In the very slow test mode, verify that the symbol display sequences are correct.One should display symbols in the order 0, 1, 2, 3, 0, 1, 2, ... and the other should display symbols in the order 0, 3, 2, 1, 0, 3, 2, …

Test: In 01 mode, look at sequence of symbols to verify that they display in correct order.
Expected: Display 1 → 0, 1, 2, 3, 0, 1, 2… and Display 2 → 0, 3, 2, 1, 0, 3, 2...
Result: Verified expected.

d. Verify that the status display works correctly.

Test: Use pushbutton, play game until win.
Expected: When symbols are moving (pressing push button), in progress symbol is displayed. When symbols land on matched, win symbol is displayed. When symbols are not matched and you release pushbutton, lose button is displayed.
Result: Verified expected.

e. Verify that the speed selects switches work correctly.

Test: Change mode from 01 to 10 to 11.

Expected: Verify that there is a speed change from 01 to 10 (slow to medium) and 10 to 11 (medium to high).

Result: Verified expected.

Conclusion

During this lab, we worked with a real-world problem statement and were able to implement the familiar concept of a slot machine. With our knowledge of hierarchical design, we created a schematic using counters, muxes and a few basic logic gates in order to count, select and output symbols. We incorporated our previous knowledge of verilog code for the symbol design and used the seven segment display on the Altera FPGA to visually present our customized symbols. Compared to our previous labs, this problem statement felt the most applicable to a real-world situation, which made the design process more exciting. Since this was the second to last lab, it definitely felt like all of our previous knowledge applied to this situation, as we were able to seamlessly incorporate numerous types of logic gates and schematics, including those previously created in past labs. Additionally, we enjoyed being able to actually play our game once it was completed, which added to the sense of accomplishment.

Question: If you want to use 8 symbols instead of only 4 to make the game harder, list all the things you would need to change.

If we wanted to use 8 symbols instead of 4, we would need to add on four more four four-to-one muxes since we are adding on four more symbols, so more symbols would be looped through the two displays. We would also need to modify our verilog code to design the four additional symbols, as well as modify our logic gates going into the win/lose/in progress symbol generator.