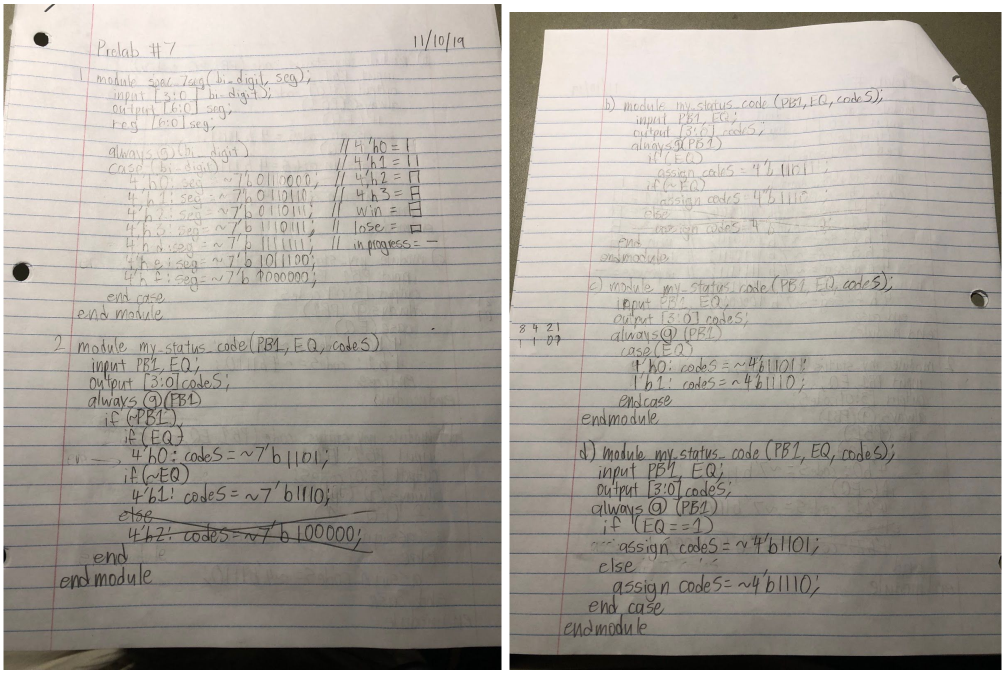
Lab Report #7
Dillon Kanai
Mariela Zuniga

Introduction

In this lab, we will be creating a slot machine with two slots. The slot machine will be using 4 symbols. To randomly create signals that display the symbols, we will be using a clock. The clock's speed can be changed using switches: test (super slow), slow, medium, and fast. On top of the symbols, there will be a status LED that will be connected to a 7 segment LED that will show whether the slot machine is in progress, you win, or you lose.

Prelab Schematic and Verilog



Prelab #7 11/10/19

```
1 module spec_7seg(bi_digit, seg);
    input [3:0] bi_digit);
    output [6:0] seg;
    reg [6:0] seg;

    always @(bi_digit)                          // 4'h0 = |
    case (bi_digit)                             // 4'h1 = ||
      4'h0: seg = ~7'b0110000;                  // 4'h2 = ⊓
      4'h1: seg = ~7'b0110110;                  // 4'h3 = 日
      4'h2: seg = ~7'b0110111;                  // win = 日
      4'h3: seg = ~7'b1110111;                  // lose = ⊓
      4'h d: seg = ~7'b1111111;                 // in progress = —
      4'h e: seg = ~7'b1011100;
      4'h f: seg = ~7'b0000000;
    end case
  end module

2 module my_status_code(PB1, EQ, codeS)
    input PB1, EQ;
    output [3:0] codeS;
    always @(PB1)
      if (~PB1)
        if (EQ)
  →      4'b0: codeS = ~7'b1101;
        if (~EQ)
         4'b1: codeS = ~7'b1110;
        ~~else~~
         ~~4'b2: codeS = ~7'b100000;~~
      end
  endmodule
```

```
b) module my_status_code (PB1, EQ, codeS);
     input PB1, EQ;
     output [3:0] codeS;
     always @(PB1)
       if (EQ)
         assign codeS = 4'b 1101 ;
       if (~EQ)
         assign codeS = 4'b 1110 ;
       ~~else~~
         ~~assign codeS = 4'b ;~~
     end
   endmodule

c) module my_status_code (PB1, EQ, codeS);
     input PB1, EQ,
     output [3:0] codeS;
     always @ (PB1)
       case (EQ)
         4'b0: codeS = ~4'b1101;
         1'b1: codeS = ~4'b1110;
       endcase
   endmodule

d) module my_status_code (PB1, EQ, codeS);
     input PB1, EQ;
     output [3:0] codeS;
     always @ (PB1)
       if (EQ==1)
         assign codeS = ~4'b1101;
       else
         assign codeS = ~4'b1110;
     end case
   endmodule
```

8 4 2 1
1 1 0 1

Figure 1: Prelab Verilog Code

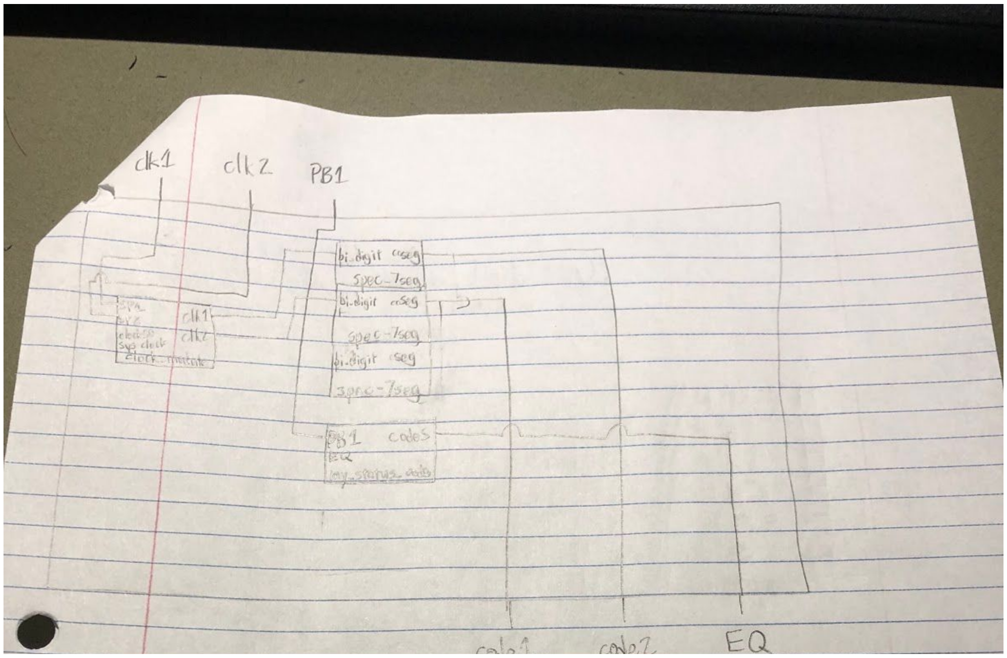
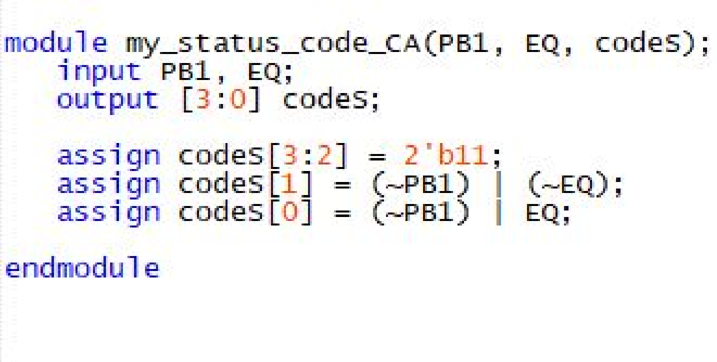


Figure 2: Prelab Schematic

Lab Schematic and Verilog

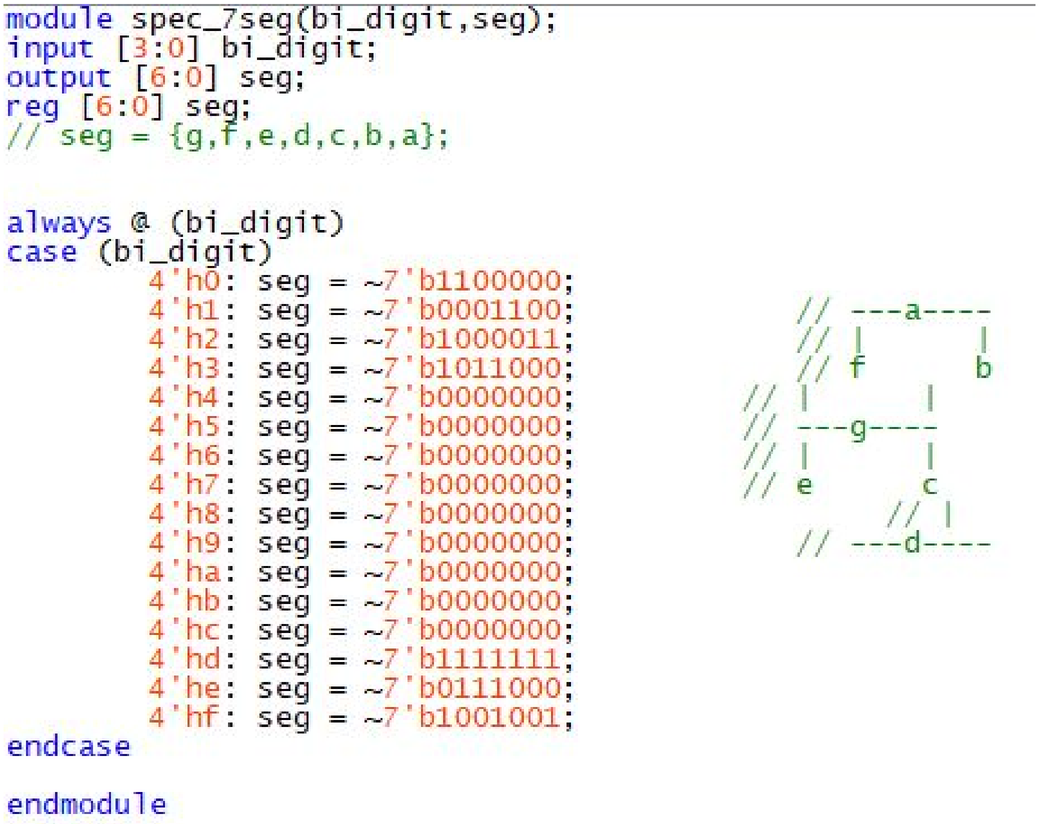


```
module my_status_code_CA(PB1, EQ, codeS);
   input PB1, EQ;
   output [3:0] codeS;

   assign codeS[3:2] = 2'b11;
   assign codeS[1] = (~PB1) | (~EQ);
   assign codeS[0] = (~PB1) | EQ;

endmodule
```

Figure 3: Status Verilog Code

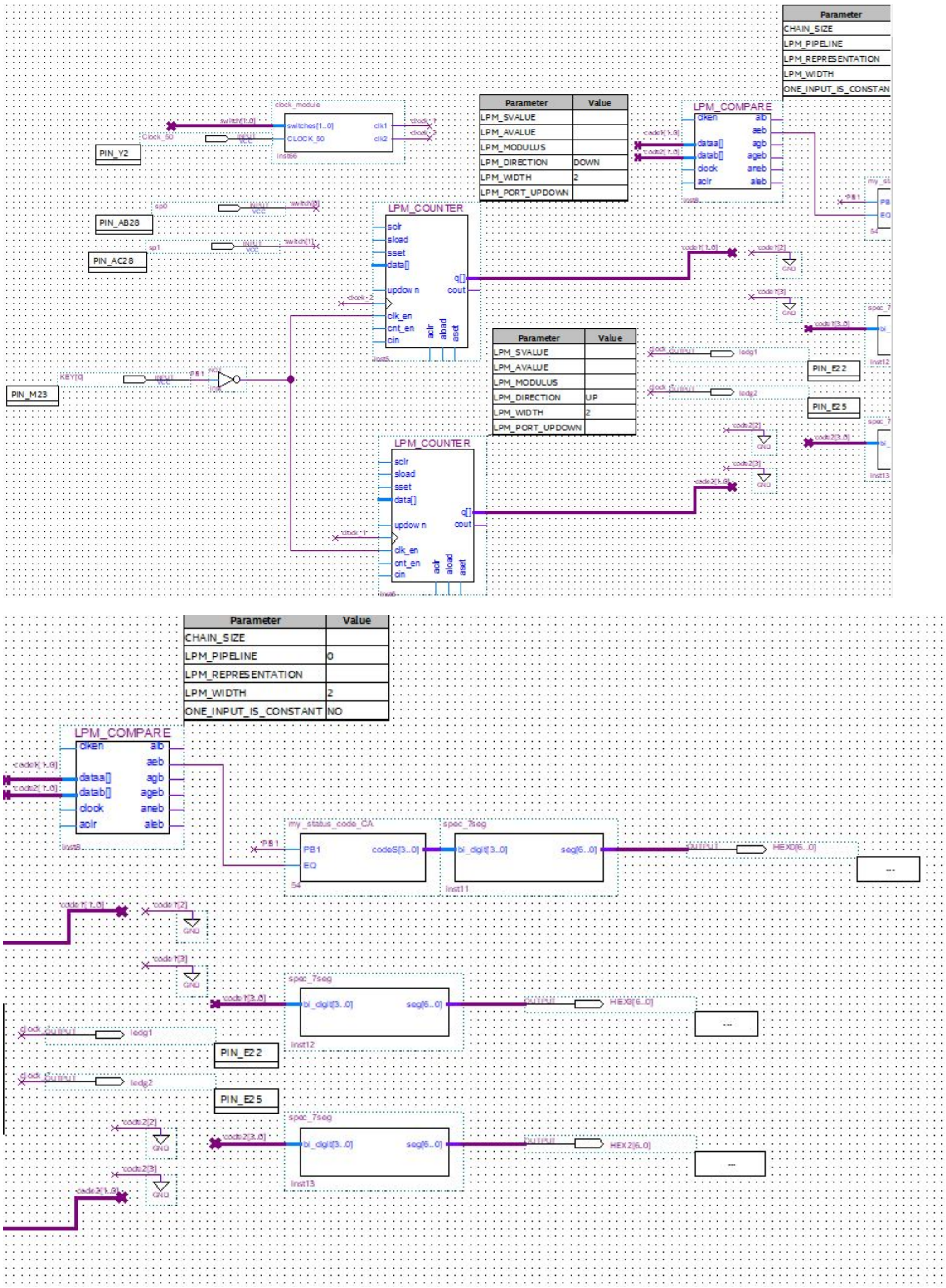


```
module spec_7seg(bi_digit,seg);
input [3:0] bi_digit;
output [6:0] seg;
reg [6:0] seg;
// seg = {g,f,e,d,c,b,a};


always @ (bi_digit)
case (bi_digit)
        4'h0: seg = ~7'b1100000;
        4'h1: seg = ~7'b0001100;            // ---a----
        4'h2: seg = ~7'b1000011;            // |      |
        4'h3: seg = ~7'b1011000;            // f      b
        4'h4: seg = ~7'b0000000;         // |      |
        4'h5: seg = ~7'b0000000;         // ---g----
        4'h6: seg = ~7'b0000000;         // |      |
        4'h7: seg = ~7'b0000000;         // e      c
        4'h8: seg = ~7'b0000000;                 // |
        4'h9: seg = ~7'b0000000;            // ---d----
        4'ha: seg = ~7'b0000000;
        4'hb: seg = ~7'b0000000;
        4'hc: seg = ~7'b0000000;
        4'hd: seg = ~7'b1111111;
        4'he: seg = ~7'b0111000;
        4'hf: seg = ~7'b1001001;
endcase

endmodule
```

Figure 4: Spec 7 Segment Display Verilog



Parameter
CHAIN_SIZE
LPM_PIPELINE
LPM_REPRESENTATION
LPM_WIDTH
ONE_INPUT_IS_CONSTAN

Parameter	Value
LPM_SVALUE	
LPM_AVALUE	
LPM_MODULUS	
LPM_DIRECTION	DOWN
LPM_WIDTH	2
LPM_PORT_UPDOWN	

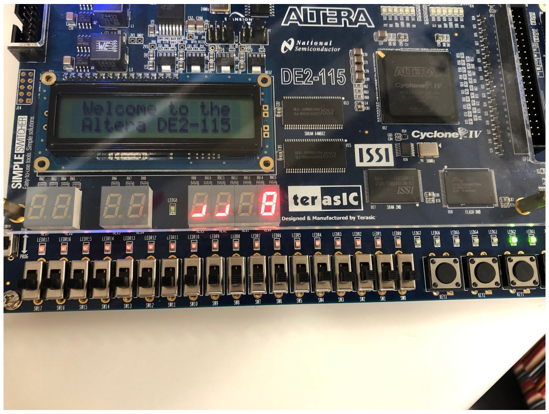
Parameter	Value
LPM_SVALUE	
LPM_AVALUE	
LPM_MODULUS	
LPM_DIRECTION	UP
LPM_WIDTH	2
LPM_PORT_UPDOWN	

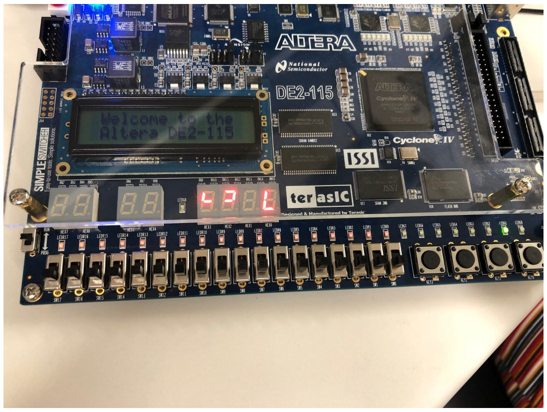
Parameter	Value
CHAIN_SIZE	
LPM_PIPELINE	0
LPM_REPRESENTATION	
LPM_WIDTH	2
ONE_INPUT_IS_CONSTANT	NO

Figure 5: Schematic

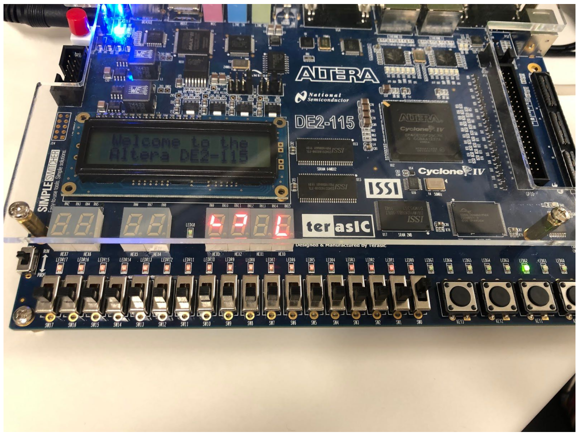
2. How often were you able to win with each mode?

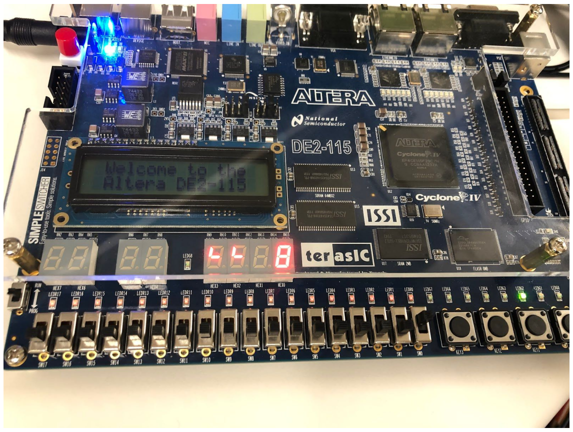
- Fast: won 4 times out of 10 (Figures 6 and 7: Win and Loss)



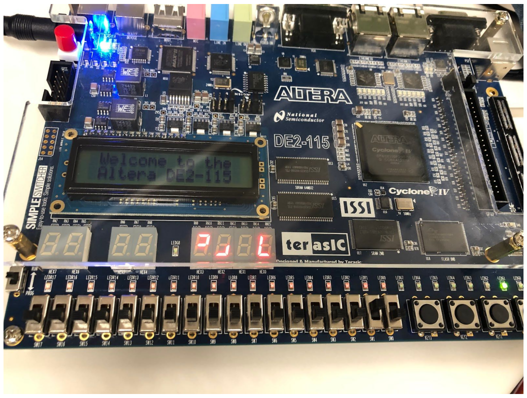


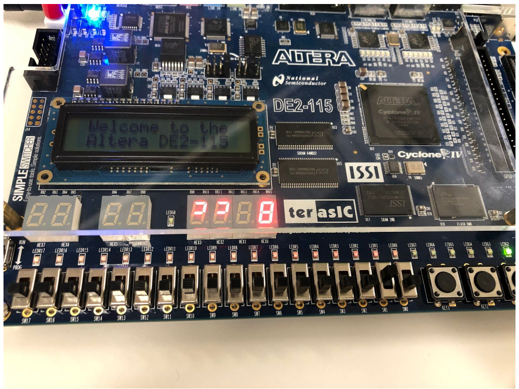
- Medium: won 3 out of 10 (Figures 8 and 9: Win and Loss)



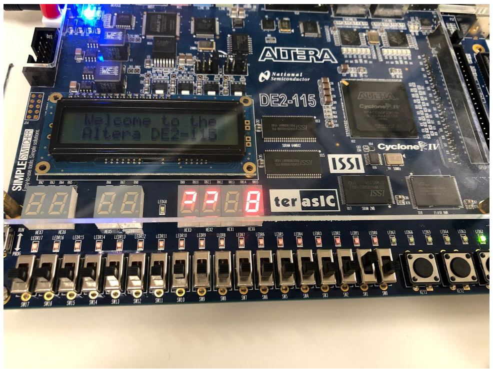


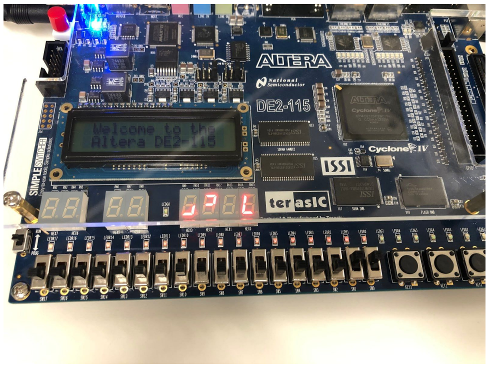
- Slow: won 3 out of 10 times (Figures 10 and 11: Win and Loss)





- Test: won 10 out of 10 times (Figures: Win and Loss)





3. From the prelab, which form of the Verilog module my_status_code was the easiest or the most intuitive for you? Explain briefly.

For this lab, the easiest version to implement was continuous assignment. First, it requires the least typing compared to the others. It also requires less processing time in the worst case scenario because there are no if statements to read. Instead, we get a consistent processing time since the code is guaranteed to trigger the assignments each time. It is also much easier to read.

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

First, the main change we would have to make is to change the comparator component to accept 8 symbols rather than just 4 by changing the <u>width</u> on both the counters and the comparator from 2 to 3. Then we would have to create 4 more signals by editing our *spec_7seg display.*