Lab Report 3
Connor Sorrell

Part 1: *Seven Segment Display*

1) BBD

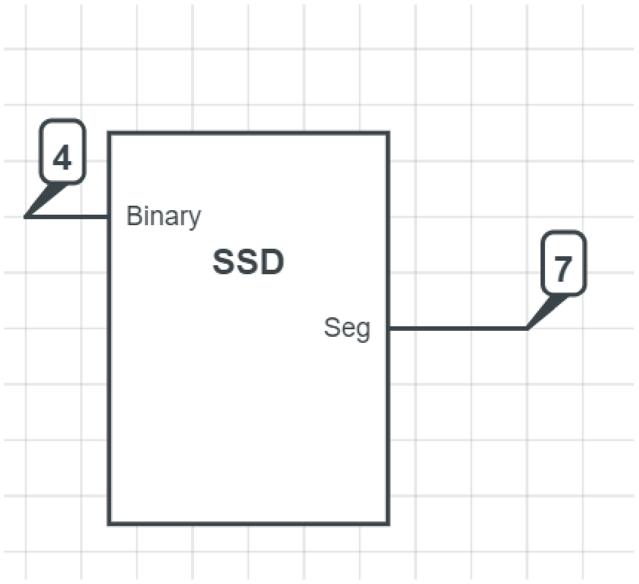

Figure 1.1: BBD for the Seven Segment Display

Decimal	**Binary**	**g seg[6]**	**f seg[5]**	**e seg[4]**	**d seg[3]**	**c seg[2]**	**b seg[1]**	**a seg[0]**	**Display Value**
0	0000	1	0	0	0	0	0	0	0
1	0001	1	1	1	1	0	0	1	1
2	0010	0	1	0	0	1	0	0	2
3	0011	0	1	1	0	0	0	0	3
4	0100	0	0	1	1	0	0	1	4
5	0101	0	0	1	0	0	1	0	5
6	0110	0	0	0	0	0	1	0	6
7	0111	1	1	1	1	0	0	0	7
8	1000	0	0	0	0	0	0	0	8
9	1001	0	0	1	1	0	0	0	9
10	1010	0	0	0	1	0	0	0	A
11	1011	0	0	0	0	0	1	1	b
12	1100	1	0	0	0	1	1	0	C
13	1101	0	1	0	0	0	0	1	d
14	1110	0	0	0	0	1	1	0	E
15	1111	0	0	0	1	1	1	0	F

Figure 1.2: Truth Table for Seven Segment Display

2) Circuit Description using *always_comb* block

```
always_comb
begin
        case(binary)
        4’b0000: seg = 7’b100000;
        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’b111000;
        4’b1000: seg = 7’b0000000;
        4’b1001: seg = 7’b0011000;
        4’b1010: seg = 7’b0001000;
        4’b1011: seg = 7’b0000011;
        4’b1101: seg = 7’b0100001;
        4’b1110: seg = 7’b0000110;
        4’b1111: seg = 7’b0001110;
```

Figure 1.3: always_comb block describing the behavior of the circuit

3) Design and Sim Code

```
module SevenSegDisplayDesign(
   input [3:0] binary,
   output logic [6:0] seg
   );

   always_comb
   begin
      case(binary)
      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'b1101: seg = 7'b0100001;
      4'b1110: seg = 7'b0000110;
      4'b1111: seg = 7'b0001110;
      endcase
   end
endmodule
```

```
module SevenSegDisplaySim();
    logic [3:0] binary;
    logic [6:0] seg;

    SevenSegDisplayDesign SevenSegDisplayDesign_inst(.*);
    initial
    begin
        for (int i = 0; i < 16; i++)
        begin
            binary = i;
            #10;
        end
    end
endmodule
```

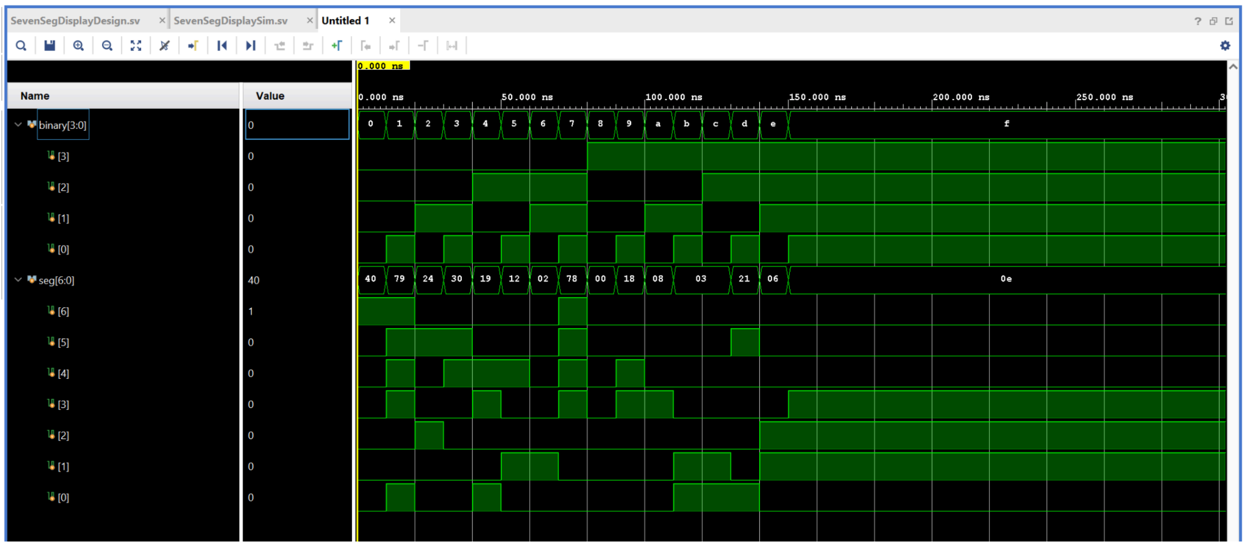

Figure 1.4: Design Code, Sim Code, and Timing Diagram

Part 2: *2x1 MUX*

1) MUX Behavior Model

```
module 2X1MUXDesign # (parameter width = 4) (
        Input [WIDTH -1:0] zero,
        Input [WIDTH -1:0] one,
        Input sel,
        Output logic [WIDTH -1:0] mux_out);
        always_comb
           begin
              if (sel == 1)
                 mux_out = one;
              else
                 mux_out = zero;
           end
```

2) Code and Timing Diagram

- *Design Code*

```
module MUXDesign #(parameter WIDTH=4)(
   input [WIDTH -1:0] one,
   input [WIDTH -1:0] zero,
   input sel,
   output logic [WIDTH -1:0] mux_out
   );

   always_comb
   begin
      if (sel == 1)
         mux_out = one;
      else
         mux_out = zero;
   end

endmodule
```

- *Sim Code*

```
module MUXSim();
   logic [3:0] zero = 4'b1110;
   logic [3:0] one = 4'b1111;
   logic [3:0] mux_out
   logic sel;
   MUXDesign #(4) MUXSim(.*);
   initial begin
      $display ("Begin MUXSim");
      sel = 0;
      #10;

      sel = 1;
      #10;

      end

endmodule
```

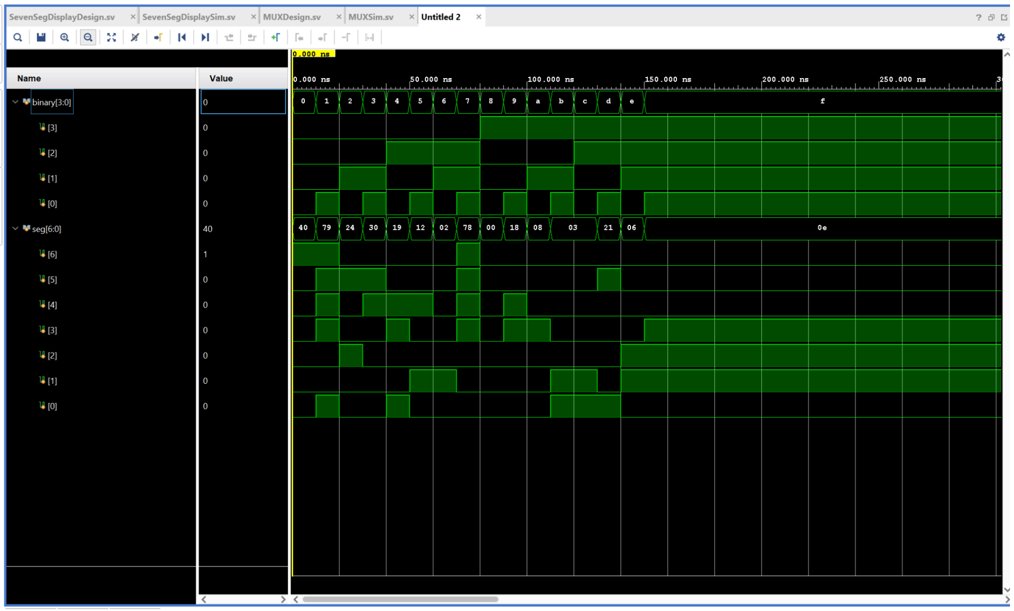

Figure 2.1: Timing Diagram for MUX

Part 3: *Top Level Module*

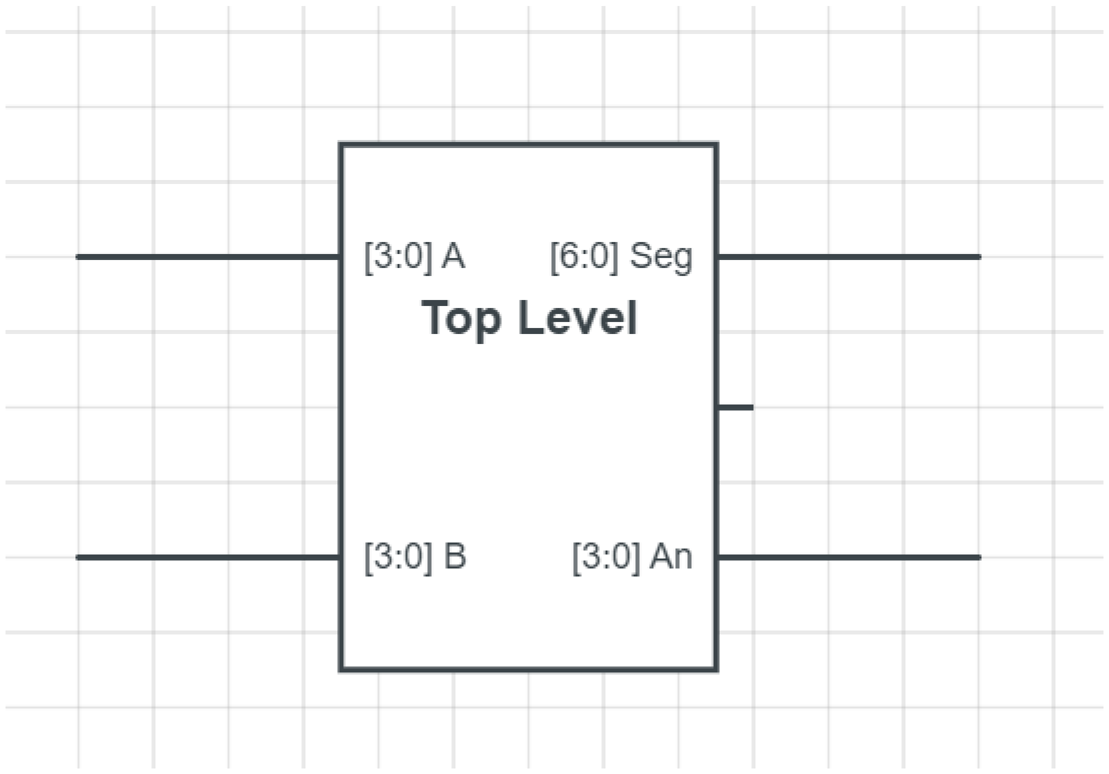

Figure 3.1: BBD For the Top Level Module. Not pictured: A = 4 bits, B = 4 Bits, Seg = 7 bits, An = 4 bits.

1) Code for Top Level Design, Sim, and Constraints (modified section)

```
module TopLevelDesign (
    input [3:0] A,
    input [3:0] B,
    output [6:0] seg,
    output [3:0] an
    );

    logic [3:0] S;
    logic Co;

    RCASource RCASource_inst (.*);

    SevenSegDisplayDesign SS1 (.binary(S),.seg(seg));

    MUXDesign #(4)MUXDesign(.zero(4'b1110),.one(4'b1111),.sel(Co),.mux_out(an));
endmodule
```

```
module TopLevelSim();
```

```
    logic [3:0] A, B;
    logic [3:0] an;
    logic [6:0] seg;
    logic C;

    RCASource RCASource_inst(.*);
    SevenSegDeisplaySim SS1 (.binary(S),.seg(seg));
    mux2#(4)MUXDesign(.zero(4'b1110),.one(4'b1111),.sel(C),.mux_out(an));

    initial
    begin

    //test 1
    A = 5;
    B = 3;
    #10
    if ( S !== 8 && C0 !== 0 && an !== 0) $display ("Error A%h B%h", A, B);

    //test 2
    A = 8;
    B = 9;
    #10
    if ( S !== 1 && C0 !== 1) $display ("Error A%h B%h", A, B);

    $display("Finished");
    end


endmodule
```

#7 segment display

```
set_property PACKAGE_PIN W7 [get_ports {seg[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[0]}]
set_property PACKAGE_PIN W6 [get_ports {seg[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[1]}]
set_property PACKAGE_PIN U8 [get_ports {seg[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[2]}]
set_property PACKAGE_PIN V8 [get_ports {seg[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[3]}]
set_property PACKAGE_PIN U5 [get_ports {seg[4]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[4]}]
set_property PACKAGE_PIN V5 [get_ports {seg[5]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[5]}]
set_property PACKAGE_PIN U7 [get_ports {seg[6]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[6]}]

#set_property PACKAGE_PIN V7 [get_ports dp]
        #set_property IOSTANDARD LVCMOS33 [get_ports dp]

set_property PACKAGE_PIN U2 [get_ports {an[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {an[0]}]
set_property PACKAGE_PIN U4 [get_ports {an[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {an[1]}]
set_property PACKAGE_PIN V4 [get_ports {an[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {an[2]}]
set_property PACKAGE_PIN W4 [get_ports {an[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {an[3]}]
```