ELC 2137 Lab 08: 4-Digit Display

Abigail Joseph

October 22, 2020

Summary

In this lab, we used modules from previous labs to create a 4-digit display that can output either hexadecimal or BCD values. We created some larger muxes and an anode decoder.

Code

Listing 1: Mux 2

```
module mux2 #(
    parameter BITS=4 //default for bits (how large is input and output)
    )(
    input [BITS-1:0] in0,
    input [BITS-1:0] in1,
    input sel,
    output [BITS-1:0] out
    );

    assign out = sel ? in1 : in0 ;

endmodule
```

Listing 2: Mux 2 Testbench

```
module mux2_test();
localparam BITS = 16;

 //declare wires, reg, etc.

reg [BITS-1:0] in0_t;
reg [BITS-1:0] in1_t;
reg sel_t;
wire [BITS-1:0] out_t;


 //instantiate dut

 mux2 #(.BITS(BITS))  dut (
    .in0(in0_t),
    .in1(in1_t),
    .sel(sel_t),
    .out(out_t)
    );
```

```
 //drive the inputs

initial begin
    sel_t=1'b0; in0_t=16'b0010; in1_t=16'b1001; #10
    sel_t=1'b1; in0_t=16'b0010; in1_t=16'b1001; #10
    sel_t=1'b0; in0_t=16'b0001; in1_t=16'b0100; #10
    sel_t=1'b1; in0_t=16'b0001; in1_t=16'b0100; #10

$finish;
end


endmodule
```

Listing 3: Mux 4

```
module mux4 #(parameter BITS = 4) (
    input reg [BITS-1:0] in0,
    input reg [BITS-1:0] in1,
    input reg [BITS-1:0] in2,
    input reg [BITS-1:0] in3,
    input reg [1:0] sel,
    output reg [BITS-1:0] out
    );

    always @*
    case(sel)
    2'b00: out = in0;
    2'b01: out = in1;
    2'b10: out = in2;
    2'b11: out = in3;
    endcase



endmodule
```

Listing 4: Mux 4 Testbench

```
module mux4_test();

localparam BITS = 16;

 //declare wires, reg, etc.

reg [BITS-1:0] in0_t;
reg [BITS-1:0] in1_t;
reg [BITS-1:0] in2_t;
reg [BITS-1:0] in3_t;
reg [1:0] sel_t;
wire [BITS-1:0] out_t;
```

```
 //instantiate dut

 mux4 #(.BITS(BITS))  dut (
    .in0(in0_t),
    .in1(in1_t),
    .in2(in2_t),
    .in3(in3_t),
    .sel(sel_t),
    .out(out_t)
    );


 //drive the inputs

initial begin
    sel_t=2'b00; in0_t=16'b0010; in1_t=16'b1001; in2_t=16'b1110; in3_t=16'
       b1011; #10
    sel_t=2'b01; in0_t=16'b0010; in1_t=16'b1001; in2_t=16'b1110; in3_t=16'
       b1011; #10
    sel_t=2'b00; in0_t=16'b0001; in1_t=16'b0100; in2_t=16'b1101; in3_t=16'
       b1010; #10
    sel_t=2'b01; in0_t=16'b0001; in1_t=16'b0100; in2_t=16'b1101; in3_t=16'
       b1010; #10
    sel_t=2'b10; in0_t=16'b0010; in1_t=16'b1001; in2_t=16'b1110; in3_t=16'
       b1011; #10
    sel_t=2'b11; in0_t=16'b0010; in1_t=16'b1001; in2_t=16'b1110; in3_t=16'
       b1011; #10
    sel_t=2'b10; in0_t=16'b0001; in1_t=16'b0100; in2_t=16'b1101; in3_t=16'
       b1010; #10
    sel_t=2'b11; in0_t=16'b0001; in1_t=16'b0100; in2_t=16'b1101; in3_t=16'
       b1010; #10

$finish;
end


endmodule
```

Listing 5: Anode Decoder

```
module an_decoder (
    input reg [1:0] in,
    output reg [3:0] out
    );

    always @*
    case(in)
    2'b00: out = 4'b1110;
    2'b01: out = 4'b1101;
    2'b10: out = 4'b1011;
    default: out = 4'b0111;
    endcase
```

```
endmodule
```

Listing 6: Anode Decoder Testbench

```
module an_decoder_test();

reg [1:0] in_t;
wire [3:0] out_t;

an_decoder dut(
    .in(in_t),
    .out(out_t)
    );

initial begin

    in_t=2'b00; #10
    in_t=2'b01; #10
    in_t=2'b10; #10
    in_t=2'b11; #10

$finish;
end

endmodule
```

Listing 7: Seven Segment 4

```
module sseg4(
    input [15:0] data,
    input hex_dec,
    input sign,
    input [1:0] digit_sel,
    output [6:0] seg,
    output dp,
    output [3:0] an
    );
    localparam BITS = 16;
    localparam BITS1 = 4;
    localparam BITS2 = 7;
    wire [15:0] BCD_out, mux2_out;
    wire [3:0] mux4_out;
    wire [6:0] ssegdec_out;
    wire sign_sel;
    wire [3:0] an_i;

    BCD11b my_BCD11b(
    .B(data[10:0]),
    .BCD_out(BCD_out)
    );

    mux2 #(.BITS(BITS)) my_mux2 (
    .in0(BCD_out),
    .in1(data),
```

```
    .sel(hex_dec),
    .out(mux2_out)
    );

    mux4 #(.BITS(BITS1)) my_mux4 (
    .in0(mux2_out[3:0]),
    .in1(mux2_out[7:4]),
    .in2(mux2_out[11:8]),
    .in3(mux2_out[15:12]),
    .sel(digit_sel),
    .out(mux4_out)
    );

    sseg_decoder my_sseg_decoder (
    .num(mux4_out),
    .sseg(ssegdec_out)
    );

    an_decoder my_an_decoder (
    .in(digit_sel),
    .out(an_i)
    );

    assign sign_sel = sign & ~an_i[3];
    assign dp = 1'b1;
    assign an = an_i;

    mux2 #(.BITS(BITS2)) my_mux2_1 (
    .in0(ssegdec_out),
    .in1(7'b0111111),
    .sel(sign_sel),
    .out(seg)
    );

endmodule
```

Listing 8: Manual

```
module sseg4_manual(
    input [15:0] sw,
    output [6:0] seg,
    output dp,
    output [3:0] an
    );

    sseg4 my_sseg4(
    .data({4'b0000,sw[11:0]}),
    .hex_dec(sw[15]),
    .sign(sw[14]),
    .digit_sel(sw[13:12]),
    .seg(seg),
    .dp(dp),
   .an(an)
    );
```

```
endmodule
```

Results

in	out
00	1110
01	1101
10	1011
11	0111

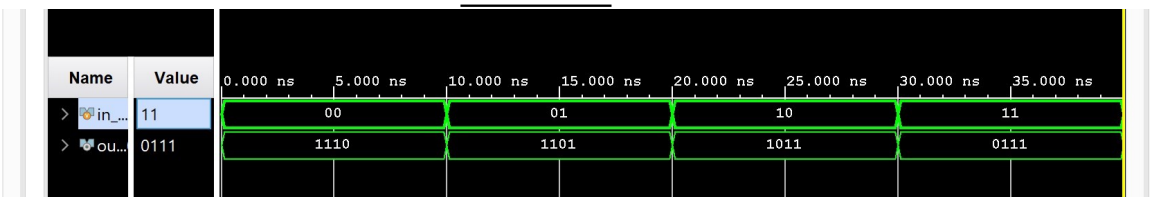

Figure 1: Anode Decoder Simulation

in0	in1	sel	out
0	0010	1001	0010
1	0010	1001	1001
0	0001	0100	0001
1	0001	0100	0100

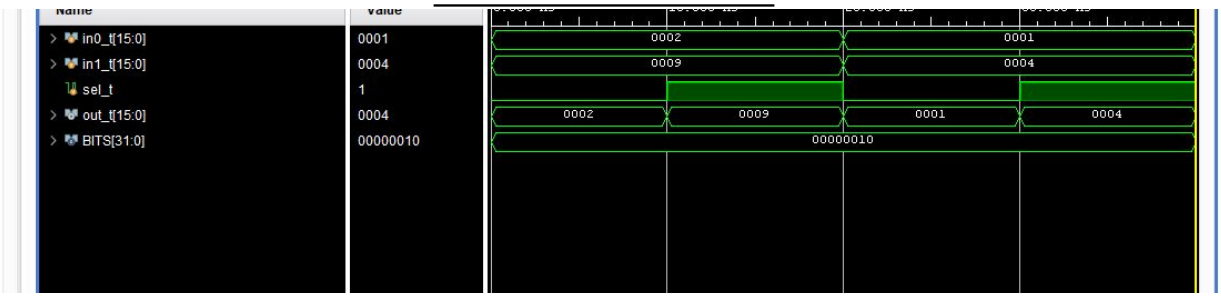

Figure 2: Mux 2 Simulation

in0	in1	in2	in3	sel	out
00	0010	1001	1110	1011	0010
01	0010	1001	1110	1011	1001
00	0001	0100	1101	1010	0001
01	0001	0100	1101	1010	0100
10	0010	1001	1110	1011	1110
11	0010	1001	1110	1011	1011
10	0001	0100	1101	1010	1101
11	0001	0100	1101	1010	1010

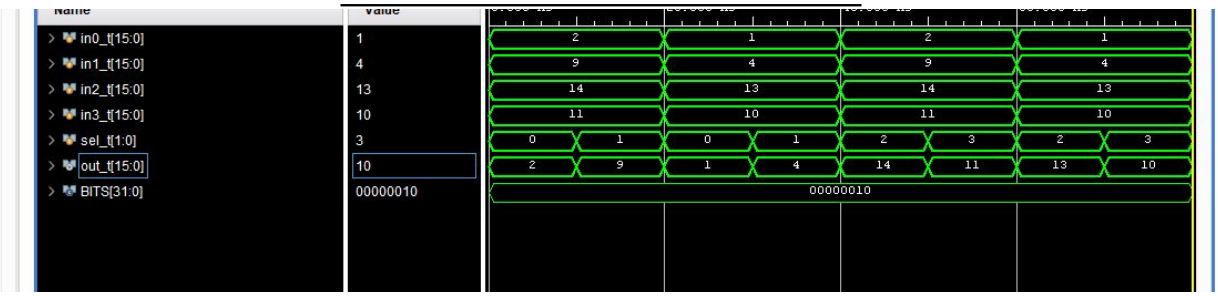

Figure 3: Mux 4 Simulation

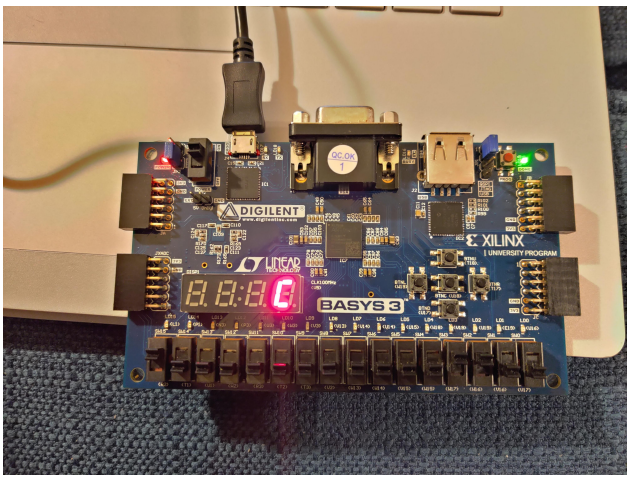

Figure 4: 1. sw15=1, sw3:0=1100, OUTPUT = C

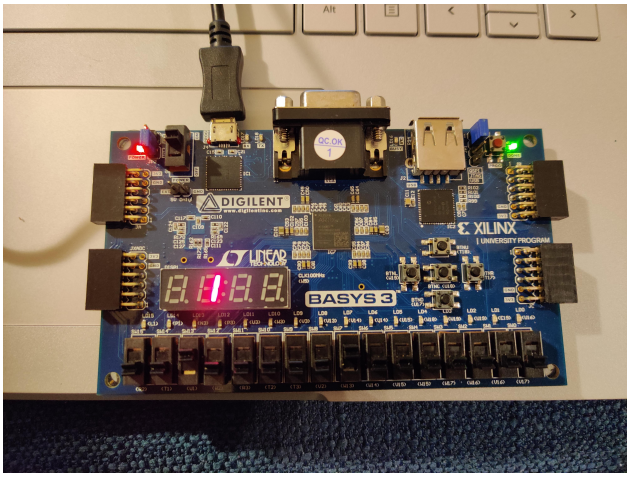

Figure 5: 2. sw13:12=10, sw7=1, OUTPUT = 1

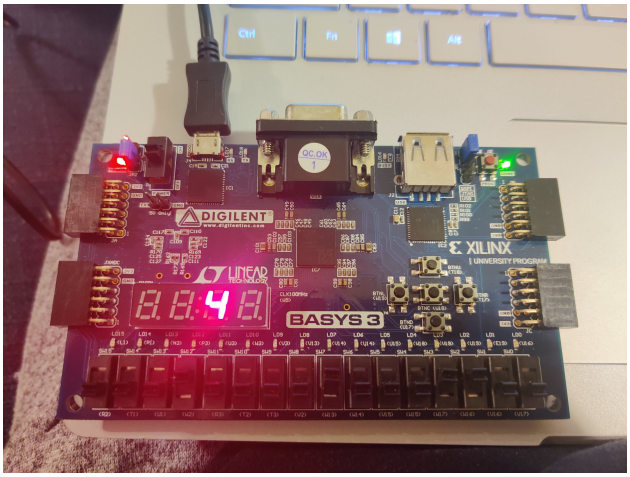

Figure 6: 3. sw13:12=01, sw7=1, sw3:0=1100, OUTPUT = 4

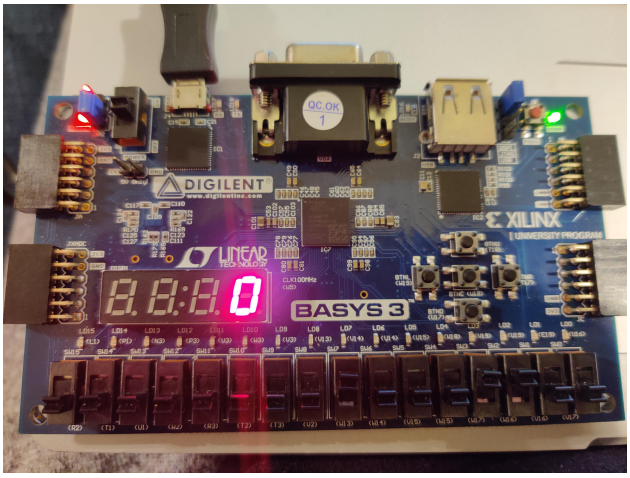

Figure 7: 4. sw13:12=00, sw7=1, sw3:0=1100, OUTPUT = 0

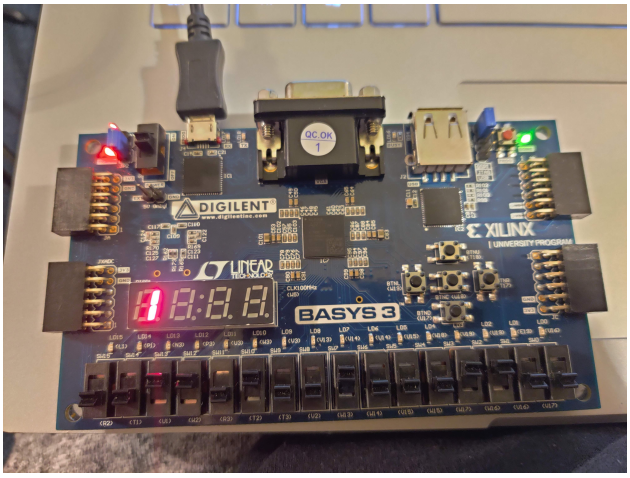

Figure 8: 5. sw13:12=11, sw10=1, sw7=1, sw3:0=1100, OUTPUT = 1

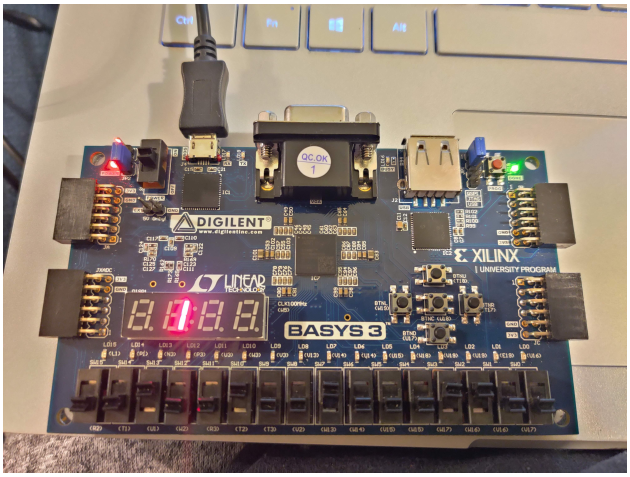

Figure 9: 6. sw13:12=10, sw10=1, sw7=1, sw3:0=1100, OUTPUT = 1

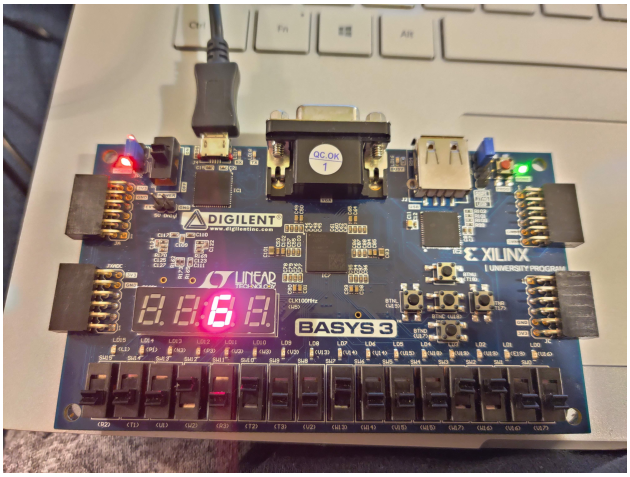

Figure 10: 7. sw13:12=01, sw10=1, sw7=1, sw3:0=1100, OUTPUT = 6

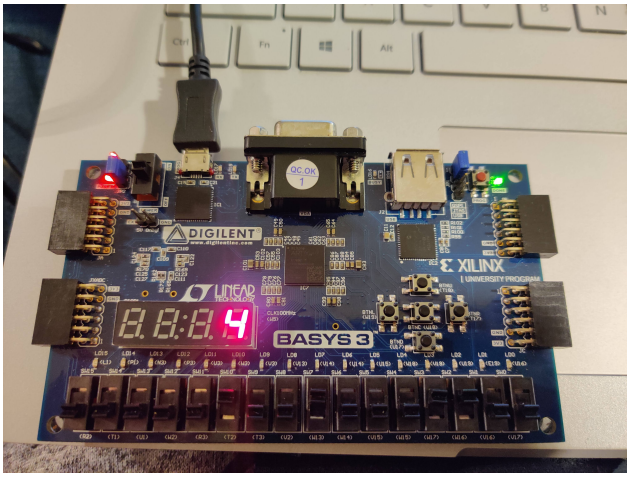

Figure 11: 8. sw13:12=00, sw10=1, sw7=1, sw3:0=1100, OUTPUT = 4

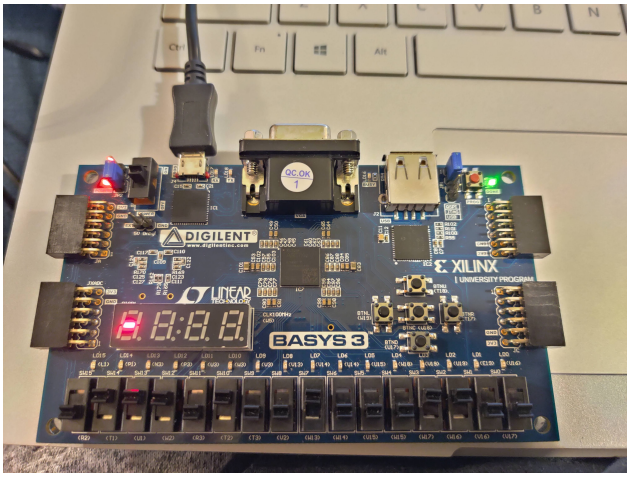

Figure 12: 9. sw14=1, sw13:12=11, sw10=1, sw7=1, sw3:0=1100, OUTPUT = -

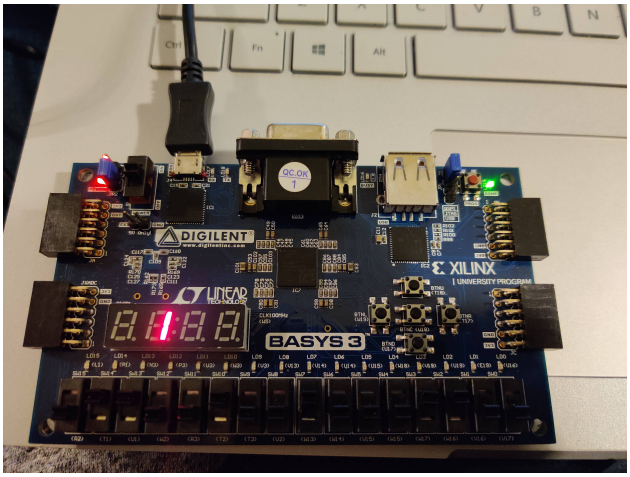

Figure 13: 10. sw14=1 sw13:12=10, sw10=1, sw7=1, sw3:0=1100, OUTPUT = 1

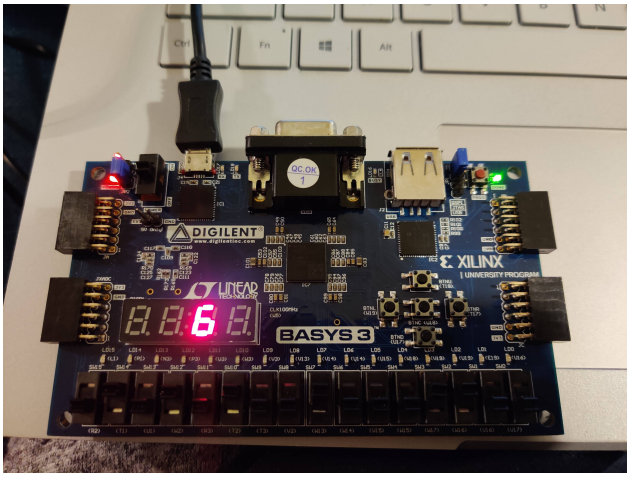

Figure 14: 11. sw14=1 sw13:12=01, sw10=1, sw7=1, sw3:0=1100, OUTPUT = 6

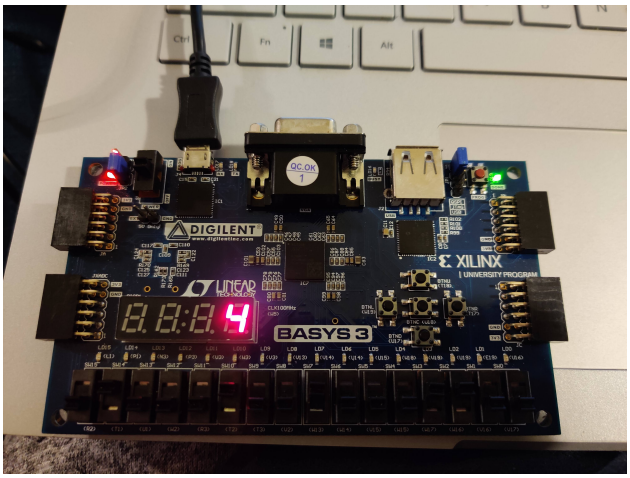

Figure 15: 12. sw14=1 sw13:12=00, sw10=1, sw7=1, sw3:0=1100, OUTPUT = 4