ELC 2137 Lab 06: Seven Segment Decoder

Abigail Joseph

October 9, 2020

Summary

In this lab, we created a 7-segment decoder which displays 7-segment hex values on a Baysys3 board.

Q&A

Q: List errors found in simulation. What does this tell you about why we run tests?

A: When simulating, I got the value Z for all my outputs. Upon examination, when calling "ssegdecoder", I had assigned a value to the variable "num" other than the wire I had created in the test file. I didn't have any other errors, only because we did most of the programming all together. We run simulations so that small errors such as mine can be caught before everything is implemented on the hardware, where it takes much longer to identify errors and make and implement corrections.

Q: How many wires are connected to the 7-segment display? If the segments were not all connected together, how many wires would there have to be? Why do we prefer the current method vs. separating all of the segments?

A: There are two 4-bit wires and one single bit wire coming into the display and one 2-bit wire, one 7-bit wire, and 3 single bit wires as outputs for a total of 8. If the segments were all separate wires, there would be 21 wires. The current method makes programming cleaner and less confusing; keeping track of assigning values to specific bits of 8 wires is a lot easier than trying to keep track of 21 wires.

Code

Listing 1: Mux

```
module mux2_4b(
    input [3:0] in0,
    input [3:0] in1,
    input sel,
    output [3:0] out
    );
    assign out = sel ? in1 : in0 ;

endmodule
```

Listing 2: Mux Test

```
module mux2_4b_test();

reg[3:0] in0_t,in1_t;
reg sel_t;

wire[3:0] out_t;

mux2_4b dut(
    .in0(in0_t),
    .in1(in1_t),
    .sel(sel_t),
    .out(out_t)
   );

initial begin
    sel_t=0; in0_t=4'b0010; in1_t=4'b1001; #10
    sel_t=1; in0_t=4'b0010; in1_t=4'b1001; #10
    sel_t=0; in0_t=4'b0001; in1_t=4'b0100; #10
    sel_t=1; in0_t=4'b0001; in1_t=4'b0100; #10

    $finish;
end

endmodule
```

Listing 3: Sseg Decoder

```
module sseg_decoder(
    input [3:0] num,
    output reg [6:0] sseg
    );

 always @*
    case(num) // single giant mux
    4'h0: sseg = 7'b1000000;
    4'h1: sseg = 7'b1111001;
    4'h2: sseg = 7'b0100100;
    4'h3: sseg = 7'b0110000;
    4'h4: sseg = 7'b0011001;
    4'h5: sseg = 7'b0010010;
    4'h6: sseg = 7'b0000010;
    4'h7: sseg = 7'b1111000;
    4'h8: sseg = 7'b0000000;
    4'h9: sseg = 7'b0010000;
    4'hA: sseg = 7'b0001000;
    4'hB: sseg = 7'b0000011;
    4'hC: sseg = 7'b0101011;
    4'hD: sseg = 7'b0100001;
    4'hE: sseg = 7'b0000110;
    default: sseg = 7'b0001110;   // in the case of an open/short/anything
        that's not an option

    endcase
```

```
endmodule
```

Listing 4: Sseg Decoder Test

```
module sseg_decoder_test();

    reg [3:0] num_t;
    wire[6:0] sseg_t;

sseg_decoder dut(
    .num(num_t),
    .sseg(sseg_t)
    );

integer i;

initial begin
    for (i=0; i<=4'b1111; i=i+1) begin
        num_t = i;
        #10;
    end
    $finish;
end


endmodule
```

Listing 5: Sseg1

```
module sseg1(
    input [3:0] A,
    input [3:0] B,
    input sel,
    output [1:0] seg_un,
    output dp,
    output seg_L,
    output seg_R,
    output [6:0] seg
    );

    wire[3:0] num;

    mux2_4b my_mux2_4b(
    .in0(A),
    .in1(B),
    .sel(sel),
    .out(num)
    );

    sseg_decoder my_sseg_decoder(
    .num(num),
    .sseg(seg)
    );
```

```
assign seg_L = ~sel;
assign seg_R= sel;

assign dp = 1;
assign seg_un = 2'b11;


endmodule
```

Listing 6: Sseg1 Test

```
module sseg1_test();



    reg [3:0] A_t; //switches
    reg [3:0] B_t;//switches
    reg sel_t;   //select
    wire [1:0] seg_un_t;
    wire dp_t, seg_L_t, seg_R_t;
    wire [6:0] seg_t;


sseg1 dut(
    .A(A_t),
    .B(B_t),
    .sel(sel_t),
    .seg_un(seg_un_t),
    .dp(dp_t),
    .seg_L(seg_L_t),
    .seg_R(seg_R_t),
    .seg(seg_t)
    );

initial begin

 sel_t = 1'b0; A_t = 4'b0000; B_t = 4'b0000; #10

 //Test case 1
 A_t = 4'b1000; B_t = 4'b0011;
 sel_t = 1'b0; #10
 sel_t = 1'b1; #10

 //Test case 2
 A_t = 4'b0010; B_t = 4'b1000;
 sel_t = 1'b0; #10
 sel_t = 1'b1; #10




$finish;
end
```

```
endmodule
```

Results

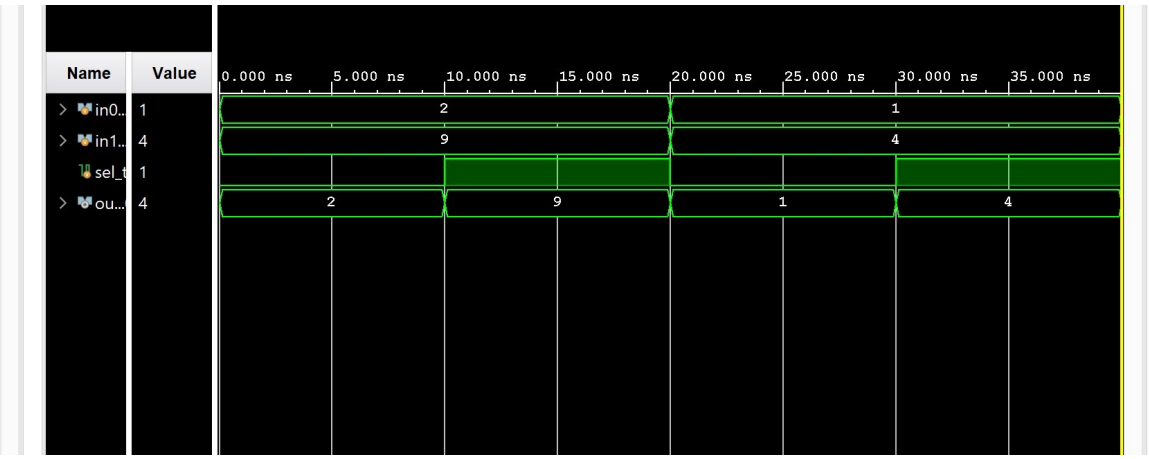

Figure 1: Mux Test

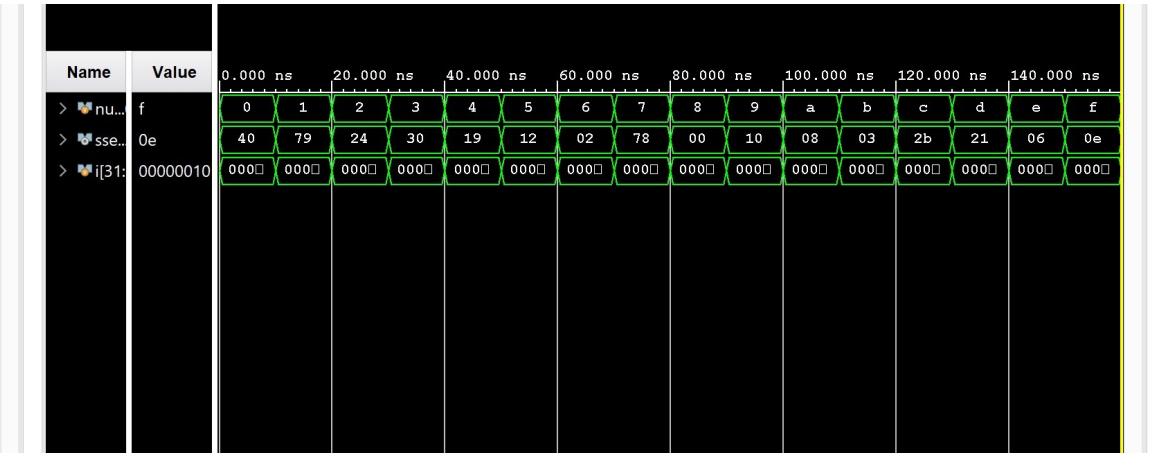

Figure 2: Sseg Decoder Test

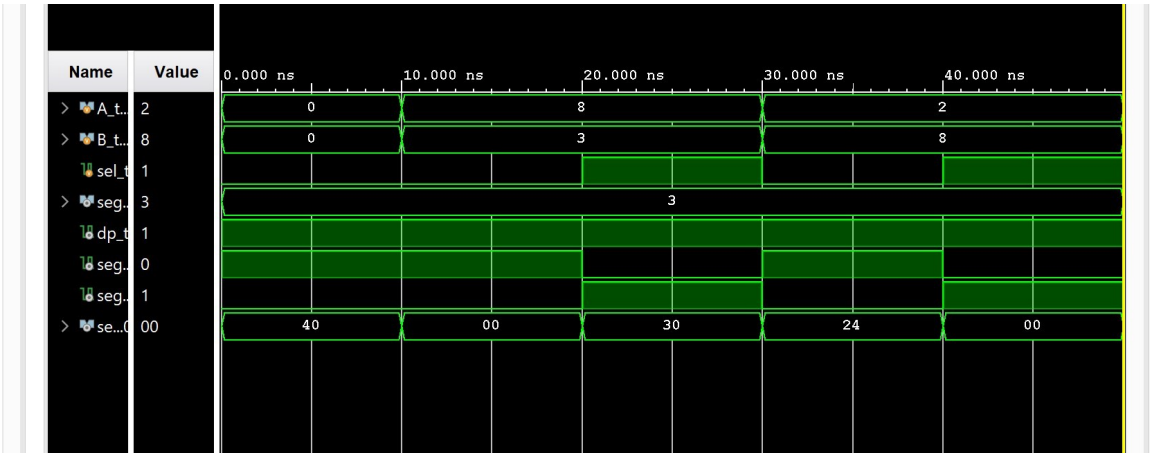

Figure 3: Sseg1 Test

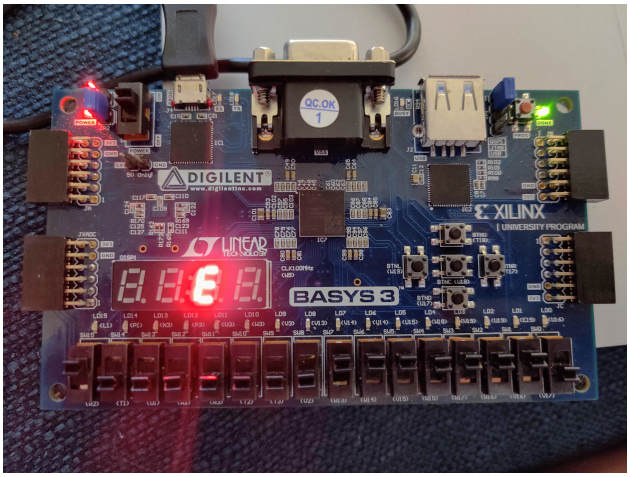

Figure 4: Basys 3 01

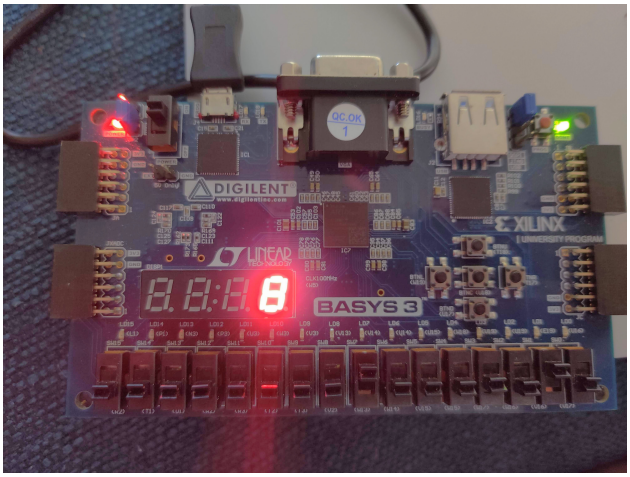

Figure 5: Basys 3 02