ELC 2137 Lab 06: MUX and 7-segment Decoder

Maddie Vorhies

October 7, 2020

Summary

The goal of this lab was to set up a circuit that displayed an 8-bit number on two 7-segment displays. In order to do this I need to create a multiplexer, a seven-segment decoder, a seven segement display (sseg1), and a wrapper. The multiplexer is used to switch between the digits of the display. When the switch is on, the number/letter will be displayed in the second colulmn (from the right) of the screen on the board and when the switch is off, the number/letter will be displyed in the first column (from the right) of the screen on the board. After I put together the multiplexer, I created a test bench to test it. The next design that I put together was the seven-segment decoder. The decoder takes in a four bit binary number and converts it to a 7-bit binary number. To make sure that the decoder worked, I created a test bench. Next, I created the 7-segment display (sseg1) which connected the decoder to the multiplexer. Again, I created one last test bench to make sure that they were properly connected. To clean up the code and make it more organized, I created a wrapper. The wrapper encompasses the sseg1 and connnects some of the wires. The wrapper connected the sw's, an's, dp's, and seg's together. Now I am able to program my board and see if it works. Once I connected my board I tested all of the values to see if it worked. Then I changed switch 15 to on and tested the values again. All of my results were correct.

Q&A

1. How many wires are connected to the 7-segment display?
There are 4 wires connected to the 7-segment display. There is 1 input and 3 outputs.

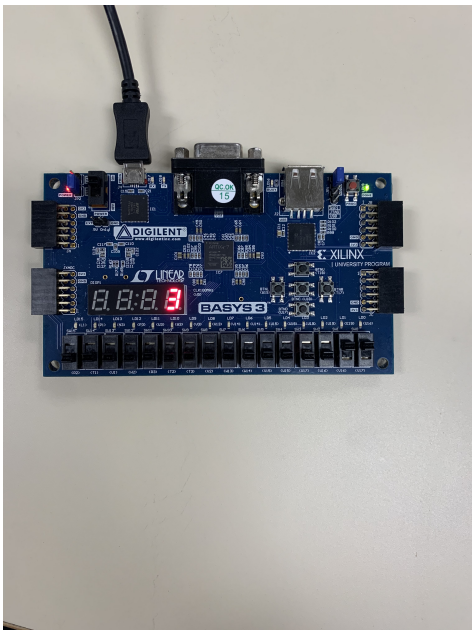
2. If the segements were not all connected together, how many wires would there have to be?
There would be 21 wires connected to the 7-segment display.

3. Why do we prefer the current method vs. separating all of the segments?
The current method is much simpler and much cleaner. The current method helps organized the circuit and make it more readable for the creator and for someone who is looking at your code. It will help in the future when you or someone else looks back on your code to see what you did. Instead of taking the time to firgure out where all 21 wires go, you can quickly see where the 4 groups of wires go.

4. List of Errors:
- Error: procedural assignment to a non-register selt is not permitted, left-hand side shoud be reg/integer/time/genvar.
- Error: HW target shutdown.
- We run simulations so that we can find the errors in our code, if they are present. We run them multiple times throughout writing code so that if there is an error early on you can fix it rather than waiting to fix the error and having to alter all of you're code after that error.

Results

sel = 0 sel = 1



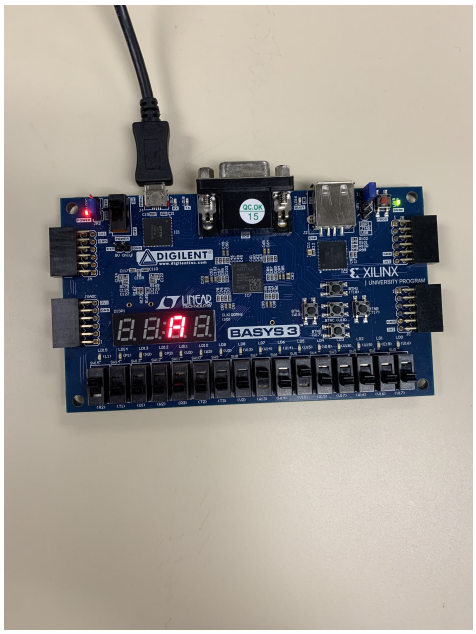


Table 1: Board Pictures

Code

```
‘timescale 1ns / 1ps
//
    //////////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer: Maddie Vorhies
//
// Create Date: 10/01/2020 11:34:33 AM
// Design Name:
// Module Name: mux2_4b
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
```

```
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   //////////////////////////////////////////////////////////////////////////////////



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

    assign out = sel ? in1 : in0;

endmodule
```

```
`timescale 1ns / 1ps
//
   //////////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 10/01/2020 11:47:47 AM
// Design Name:
// Module Name: mux2_4b_test
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   //////////////////////////////////////////////////////////////////////////////////



module mux2_4b_test();

    reg [3:0] in0_t;
    reg [3:0] 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

        in0_t = 4'b0000; in1_t = 4'b0001; sel_t = 0; #10;
        in0_t = 4'b0000; in1_t = 4'b0001; sel_t = 1; #10;
        in0_t = 4'b0011; in1_t = 4'b0000; sel_t = 0; #10;
        in0_t = 4'b0011; in1_t = 4'b0000; sel_t = 1; #10;

        $finish;

    end

endmodule
```

```
`timescale 1ns / 1ps
//
    //////////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 10/01/2020 12:27:07 PM
// Design Name:
// Module Name: sseg_decoder
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
    //////////////////////////////////////////////////////////////////////////////////



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

    always @*
        case(num)
```

```
          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'b0011000;
          4'hA: sseg = 7'b0001000;
          4'hB: sseg = 7'b0000011;
          4'hC: sseg = 7'b0100111;
          4'hD: sseg = 7'b0100001;
          4'hE: sseg = 7'b0000110;
          default: sseg = 7'b0001110;
       endcase

endmodule
```

```
`timescale 1ns / 1ps
//
   //////////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer: Maddie Vorhies
//
// Create Date: 10/01/2020 12:45:12 PM
// Design Name:
// Module Name: sseg_decoder_test
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   //////////////////////////////////////////////////////////////////////////////////



module sseg_decoder_test();

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

    sseg_decoder dut(
       .num(num_t),
```

```
        .sseg(sseg_t)
    );

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



endmodule
```

```
`timescale 1ns / 1ps
//
    //////////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 10/01/2020 01:14:52 PM
// Design Name:
// Module Name: sseg1
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
    //////////////////////////////////////////////////////////////////////////////////



module sseg1(

    input [15:0] switches,
    output [3:0] out,
    output dp,
    output [6:0] sseg
 );

    wire [3:0] num;

    mux2_4b mux (
        .in1(switches[7:4]),
        .in0(switches[3:0]),
```

```
        .sel(switches[15]),
        .out(num)
    );

    sseg_decoder prototype (
        .num(num),
        .sseg(sseg)
    );

    assign out[0] = switches[15];
    assign out[1] = ~switches[15];
    assign dp = 1'b1;
    assign out[3:2] = 2'b11;


endmodule
```

```
`timescale 1ns / 1ps
//
    //////////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 10/07/2020 08:53:21 PM
// Design Name:
// Module Name: sseg1_test
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
    //////////////////////////////////////////////////////////////////////////////////



module sseg1_test();

    reg [15:0] switches;
    wire [3:0] out;
    wire dp;
    wire [6:0] sseg;
    integer i;

    sseg1 dut (
        .switches(switches),
        .out(out),
```

```
        .sseg(sseg),
        .dp(dp)
    );

    initial begin
        switches = 16'h0000; #10;

        for(i = 16'h0000; i < 16'hffff; i = i + 1) begin
            switches[7:0] = i;
            switches[15] = 1'b0; #10;
            switches[15] = 1'b1; #10;
        end

        $finish;
    end


endmodule
```