ELC 2137 Lab 06: MUX and 7-segment decoder

My Nguyen

October 6, 2020

Summary

This lab's purpose is to set up a circuit to display an 8-bit number on two 7-segment displays using Verilog. First, create a multiplexer to switch between the two displays, each display will use 4 bits. Secondly, create a seven-segment decoder to covert the output from the multiplexer into hexadecimal digit for the 7-segment LED display. Then, a top-level module is created to wrap around the multiplexer and decoder proving input and output. Finally, import a constraint file for Basys3 board and create a bitstream file for Basys3 board.

Table and Figure

Code

Listing 1: Multiplexer Implementation

```
module mux2_4b(
    input [3:0] in0, in1,
    input se1,
    output [3:0] out
    );

    assign out = se1 ?  in0 : in1;
endmodule
```

Listing 2: Multiplexer Test Bench

```
module mux2_4b_test();
    reg [3:0] in0, in1;
    reg se1;
    wire [3:0] out;

    mux2_4b dut(
    .in0(in0),
    .in1(in1),
    .se1(se1),
    .out(out)
    );

    initial begin
        in0 <= 4'h8; in1 <= 4'h7; se1 <= 0; #10; se1 <=1; #10;
```

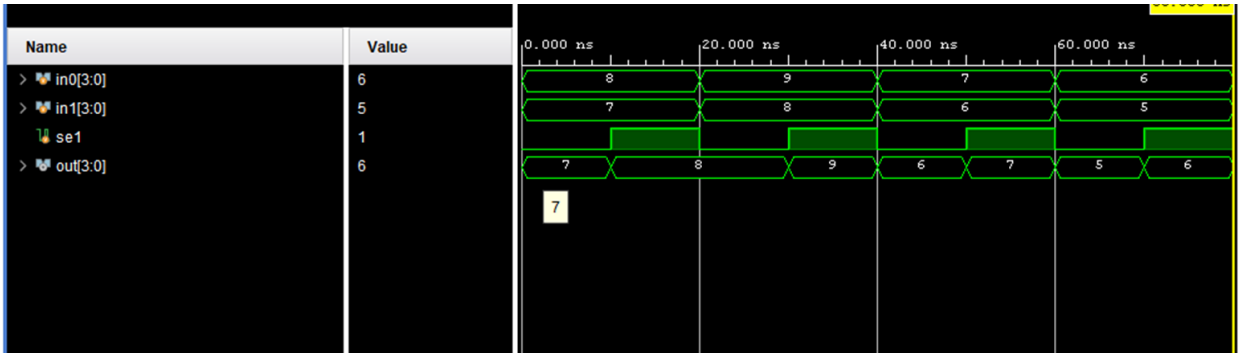

Figure 1: Multiplexer ERT

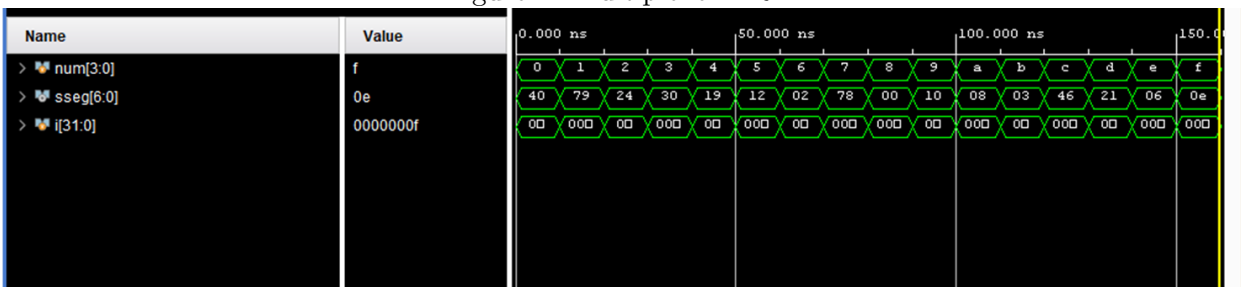

Figure 2: 7-bit Decoder ERT

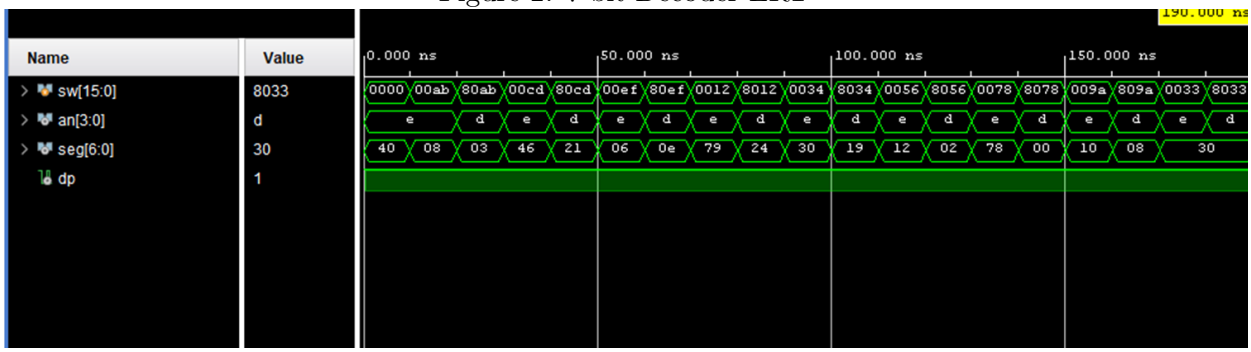

Figure 3: sseg1 ERT

```
        in0 <= 4'h9; in1 <= 4'h8; se1 <= 0; #10; se1 <=1; #10;
        in0 <= 4'h7; in1 <= 4'h6; se1 <= 0; #10; se1 <=1; #10;
        in0 <= 4'h6; in1 <= 4'h5; se1 <= 0; #10; se1 <=1; #10;
        $finish;
    end


endmodule
```

Listing 3: 7-bit Decoder Implementation

```
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'b0010000;
            4'hA: sseg = 7'b0001000;
            4'hB: sseg = 7'b0000011;
            4'hC: sseg = 7'b1000110;
            4'hD: sseg = 7'b0100001;
            4'hE: sseg = 7'b0000110;
            4'hF: sseg = 7'b0001110;
        endcase
endmodule
```

Listing 4: 7-bit Decoder Test Bench

```
module sseg_decoder_test();
    reg [3:0] num;
    wire [6:0] sseg;

    integer i;

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

    initial begin
        for (i=0; i<=8'hF; i=i+1) begin
            num = i; #10;
        end
        $finish;
    end
endmodule
```

Listing 5: Decoder Wrap Implementation

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

    sseg1 main(
        .sw3_0(sw[3:0]),
```

```
        .sw7_4(sw[7:4]),
        .sw15(sw[15]),
        .an(an),
        .seg(seg),
        .dp(dp)
        );
endmodule
```

Listing 6: Decoder Wrap Test Bench

```
module sseg1_test();
    reg [15:0] sw;
    wire [3:0] an;
    wire [6:0] seg;
    wire dp;

    wrap dut(
        .sw(sw),
        .an(an),
        .seg(seg),
        .dp(dp)
        );

    initial begin
        //initalize
        sw = 16'h0000 ; #10;
        // Test case 1
        sw [7:0] = 8'hAB;
        sw [15] = 1'b0; #10;
        sw [15] = 1'b1; #10;

        // Test case 2
        sw [7:0] = 8'hCD;
        sw [15] = 1'b0; #10;
        sw [15] = 1'b1; #10;

        // Test case 3
        sw [7:0] = 8'hEF;
        sw [15] = 1'b0; #10;
        sw [15] = 1'b1; #10;

        // Test case 4
        sw [7:0] = 8'h12;
        sw [15] = 1'b0; #10;
        sw [15] = 1'b1; #10;

        // Test case 5
        sw [7:0] = 8'h34;
        sw [15] = 1'b0; #10;
        sw [15] = 1'b1; #10;

        // Test case 6
        sw [7:0] = 8'h56;
        sw [15] = 1'b0; #10;
```

```
        sw [15] = 1'b1; #10;

        // Test case 7
        sw [7:0] = 8'h78;
        sw [15] = 1'b0; #10;
        sw [15] = 1'b1; #10;

        // Test case 8
        sw [7:0] = 8'h9A;
        sw [15] = 1'b0; #10;
        sw [15] = 1'b1; #10;

        // Test case 9
        sw [7:0] = 8'h33;
        sw [15] = 1'b0; #10;
        sw [15] = 1'b1; #10;
        $finish;
    end
endmodule
```

Screenshot

Questions

3. Most errors found where from typo and misunderstanding the prompt. A particular happens with created a bitstream file, where if you don't get the decoder wrap file as top, it will not causes an error telling you to fix your constrains file.
4. Nine wires are connected to the 7-segment display. If the segments were not all connected together, the decoder output will create a piority network instead of a parallel, which will make it harder to create segment display and harder to debug.

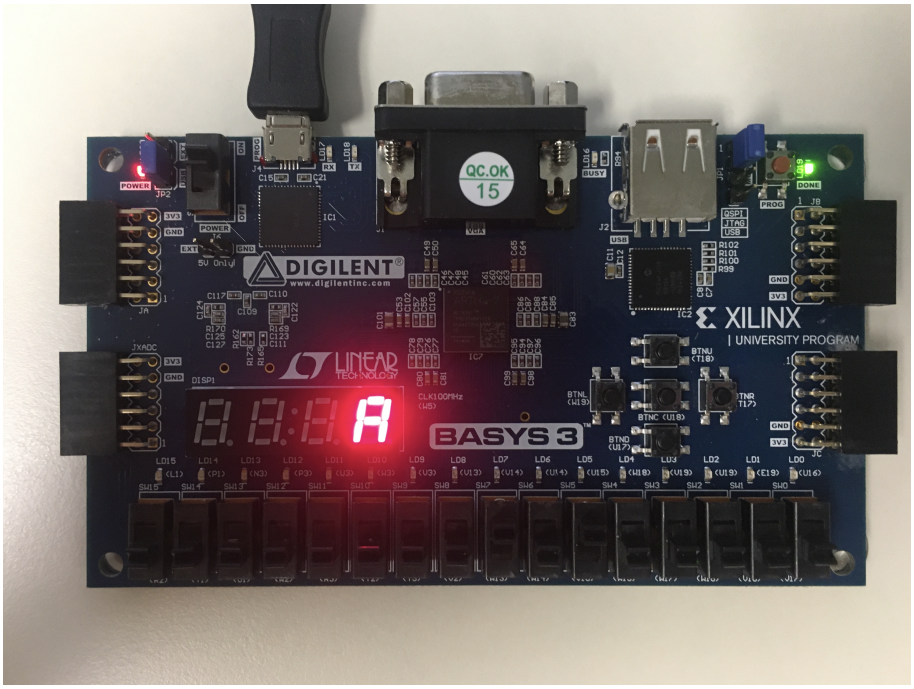

Figure 4: First digit

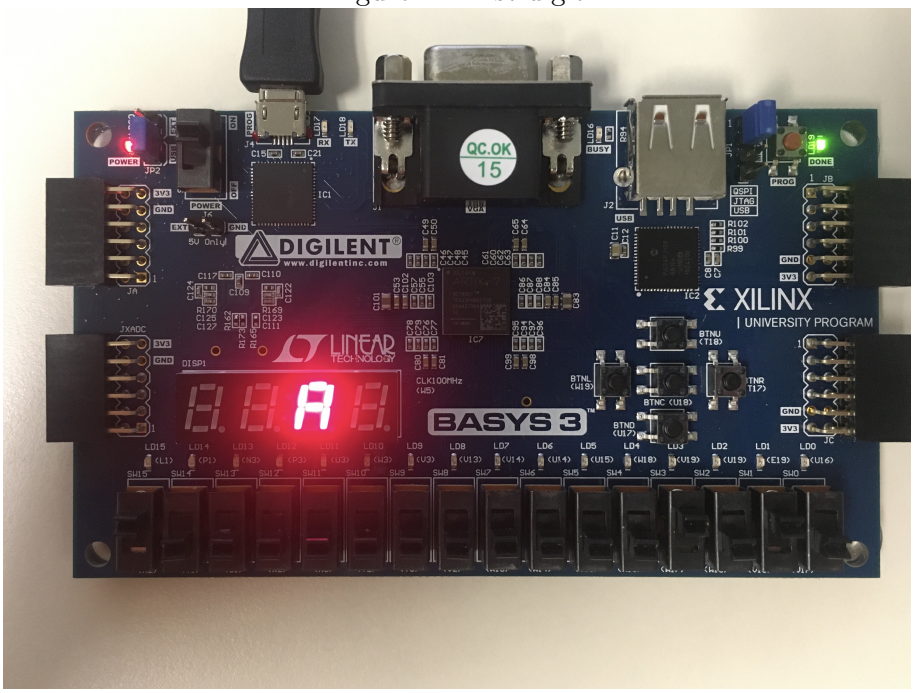

Figure 5: Second digit