ELC 2137 Lab 6: MUX and 7-segment Decoder

Spencer Stinson

October 9, 2020

Summary

In this experiment, we learned the design and code for a multiplexer, decoder, and how to put these both into one more complex design. We also learned to use this design with a physical board. We used new syntax that we had previously not had use for, for example; always@*, for, and case. To apply this design to the FPGA, we also had to learn how to use constraints and how to run synthesis and implementations, as well as to generate a bitstream.

Q&A

- List the errors found during simulation. What does this tell you about why we run simulations?
  One of my main errors was in naming. I consistently had an input or output name slightly different than it was meant to be. This caused me many problems and also caused my Basys board not to run correctly originally. There were also errors with the connections of some of the wires, which I only found by examining the test benches.
- 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. seperating all of the segments?
  There are 5 wires connected to the 7-segment display. If the segments were not all connected together, there would be 29 wires. We prefer the current method because it is much more efficient and it allows us to do all of this in less steps, as we are doing the same thing to each bit, so doing it all individually would be repetitive and cause more probability for error.

Code

Listing 1: mutiplexer code

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

// Company:
// Engineer:
//
// Create Date: 10/07/2020 08:33:48 PM
// Design Name:
// Module Name: mux4_2b
```

```
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   //////////////////////////////////////////////////////////////////////////////////



module mux4_2b(
    input [3:0] A,
    input [3:0] B,
    input sel,
    output [3:0] out
    );

    assign out = sel ? B : A ;
endmodule
```

Listing 2: multiplexer test bench code

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

// Company:
// Engineer:
//
// Create Date: 10/07/2020 08:35:19 PM
// Design Name:
// Module Name: mux4_2b_testbench
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   //////////////////////////////////////////////////////////////////////////////////
```

```
module mux4_2b_testbench();
    reg [3:0] A_t;
    reg [3:0] B_t;
    reg sel_t;
    wire [3:0] out_t;

mux4_2b dut(
  .A(A_t), .B(B_t), .sel(sel_t), .out(out_t)
    );
  initial begin
 A_t<=4'h8; B_t<= 4'h7; sel_t<=0; #10; sel_t<=1; #10;
 A_t<=4'h6; B_t<= 4'h5; sel_t<=0; #10; sel_t<=1; #10;
 A_t<=4'h4; B_t<= 4'h3; sel_t<=0; #10; sel_t<=1; #10;
 A_t<=4'h2; B_t<= 4'h1; sel_t<=0; #10; sel_t<=1; #10;


  $finish;
  end
endmodule
```

Listing 3: 7 segment decoder code

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

// Company:
// Engineer:
//
// Create Date: 10/07/2020 08:54:50 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'b0110000;
    4'h3: sseg = 7'b0100100;
    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;
    4'hF: sseg = 7'b0001110;
    endcase
endmodule
```

Listing 4: 7 segment decoder testbench code

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

// Company:
// Engineer:
//
// Create Date: 10/07/2020 09:01:30 PM
// Design Name:
// Module Name: sseg_decoder_testbench
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
    //////////////////////////////////////////////////////////////////////////////////



module sseg_decoder_testbench( );

integer i;

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

```
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
```

Listing 5: 7 segment code

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

// Company:
// Engineer:
//
// Create Date: 10/08/2020 04:01:49 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 [3:0] A,
    input [3:0] B,
    input sel,
    output [1:0] seg_un,
    output dp,
    output [6:0] sseg,
    output seg_l,
    output seg_r
    );

    wire [3:0] num;
```

```
    mux4_2b m1(
    .A(A), .B(B),
    .sel(sel), .out(num)
    );

    sseg_decoder s1(
    .num(num), .sseg(sseg)
    );

    assign seg_r = sel;
    assign seg_l = ~sel;
    assign dp=1'b1 ;
    assign seg_un = 2'b11;
endmodule
```

Listing 6: 7 segment wrapper code

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

// Company:
// Engineer:
//
// Create Date: 10/08/2020 02:46:25 PM
// Design Name:
// Module Name: sseg1_wrapper
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   //////////////////////////////////////////////////////////////////////////////////



module sseg1_wrapper(
    input [15:0] sw,
    input clk,
    output [3:0] an,
    output dp,
    output [6:0] seg
    );

     sseg1_ s1(
    .A( sw[7:4]),
```

```
    .B(sw[3:0]),
    .sel(sw[15]),
    .seg_l(an[1]),
    .seg_r(an[0]),
    .sseg(seg),
    .seg_un(an[3:2]),
    .dp(dp)
    );


endmodule
```

Listing 7: 7 segment testbench code

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

// Company:
// Engineer:
//
// Create Date: 10/07/2020 08:35:19 PM
// Design Name:
// Module Name: mux4_2b_testbench
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   //////////////////////////////////////////////////////////////////////////////////



module sseg_1_testbench();
    integer i;
    reg [15:0] sw_t;
    reg clk_t;
    wire [3:0] an_t;
    wire dp_t;
    wire [6:0] sseg_t;

sseg1_wrapper dut(
.sw(sw_t),
.clk(clk_t),
.an(an_t),
.dp(dp_t),
.sseg(sseg_t)
```

```
);

initial begin
sw_t= 16'h0000;
for (i= 16'h0000; i<=16'hffff; i=i+1) begin
    sw_t[7:0]= i;
    sw_t[15]= 1'b1; #10;
    sw_t[15]= 1'b0; #10;
end
$finish ;
end
endmodule
```

Results

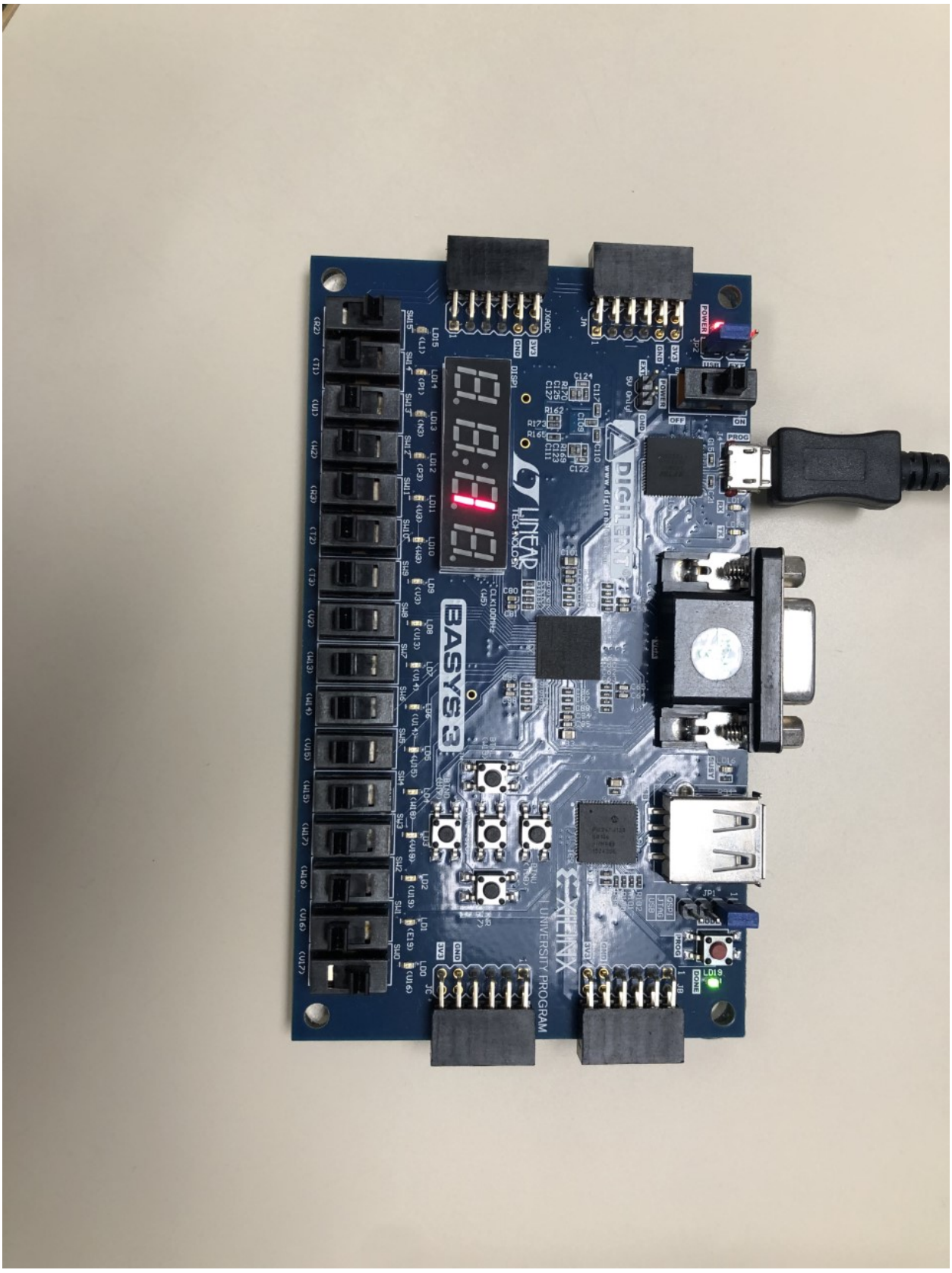

Figure 1: Basys board running 7 segment on second display

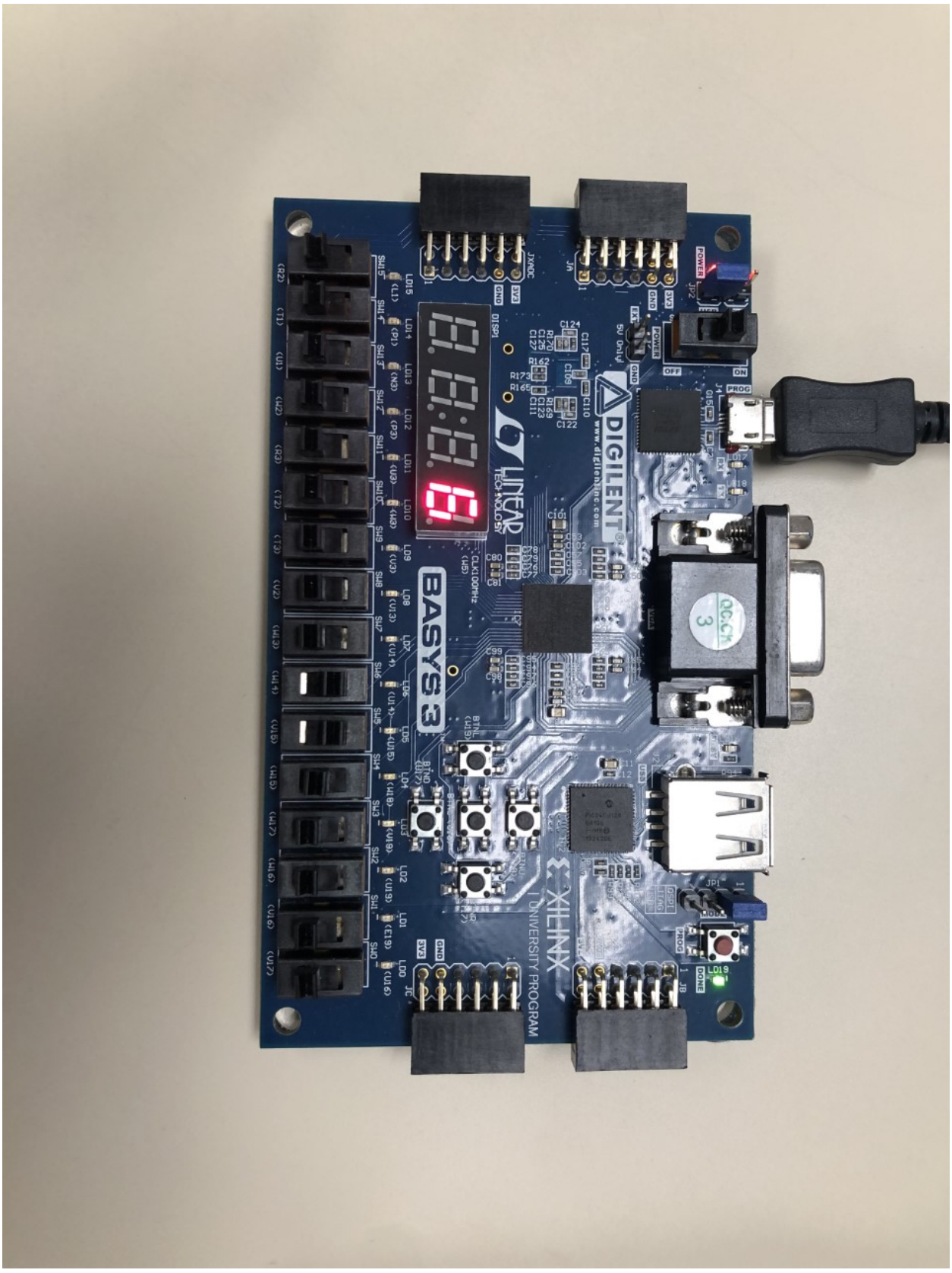

Figure 2: Basys board running 7 segment