ELC 2137 Lab 10: 7-segment Display with Time-Division Multiplexing

Jake Simmons

April 23, 2020

Summary

The purpose of this lab was to learn how to use the technique called Time-Division Multiplexing. We are using Time-Division Multiplexing to slow down the signals going to the 4 digit display. This is so there will be enough time for a human to process and see what is actually being displayed. First, a counter module was made and succesfully simulated. Secondly, the sseg4 from a previous lab was modified and the counter was added to the module. A schematic of the new, sseg4TDM, is provided. After testing sseg4TDM, the top level module, calclab10, was created. The top level module toplab9 was used in this module which was also created in a previous lab.

Q&A

1. What are the three main “groups” of the RTL definition of sequential logic?
   (a) The three main groups of the RTL definitions of sequential logic is event driven, clock driven and pulse driven.
2. Copy Figure 10.3b onto your own paper (or do it electronically) and draw three boxes around the components that belong to each group. Include your annotated figure in your report.

Figure 1: 10.3B Labeling

3. If instead of a counter, you wanted to make a shift register that moved the input bits from right to left (low to high). What would you put on the line Q next = /*???*/
   (a) To make a shift register that moved bits from right to left, you would put on the line Q next = Q reg - 1;

Results

Time (ns):	0	50	100	150	200	250	300	350	400	450	500	550	600	650	700	750
clk	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
rst	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
in	0	1	1	0	1	1	1	1	1	0	1	1	1	0	0	0
out	X	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0
tick	X	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
i	X	0	1	4	6	9	a	a	a	a	1	4	6	9	a	a

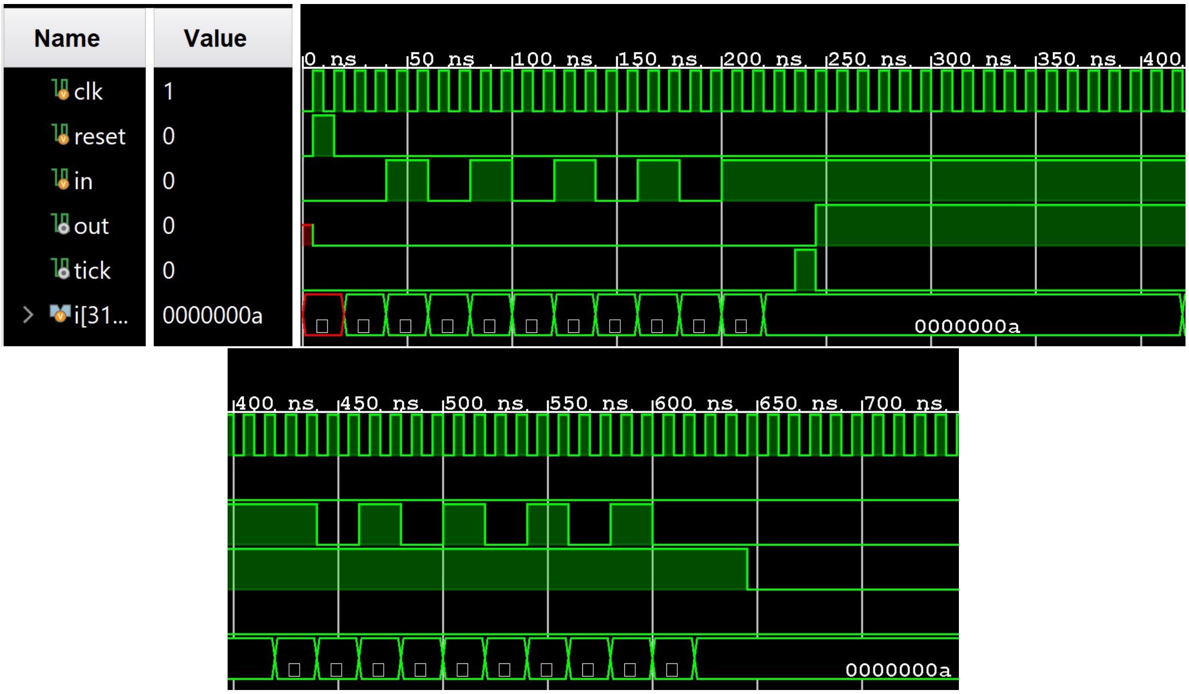
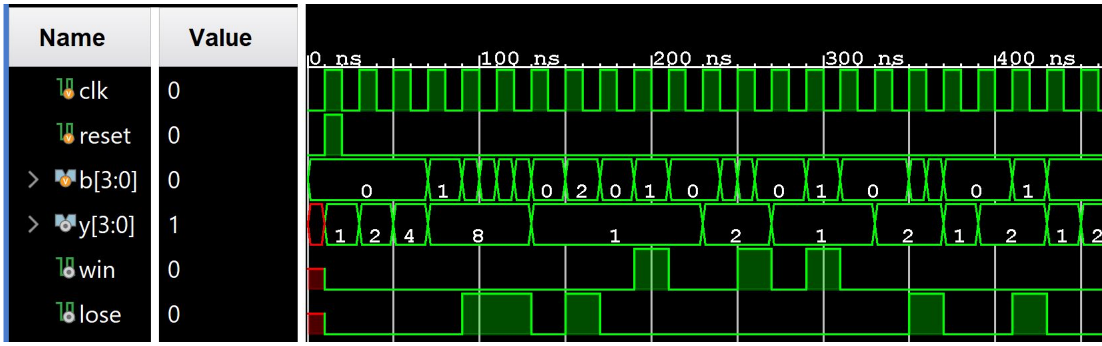


Figure 2: debounce simulation waveform and ERT

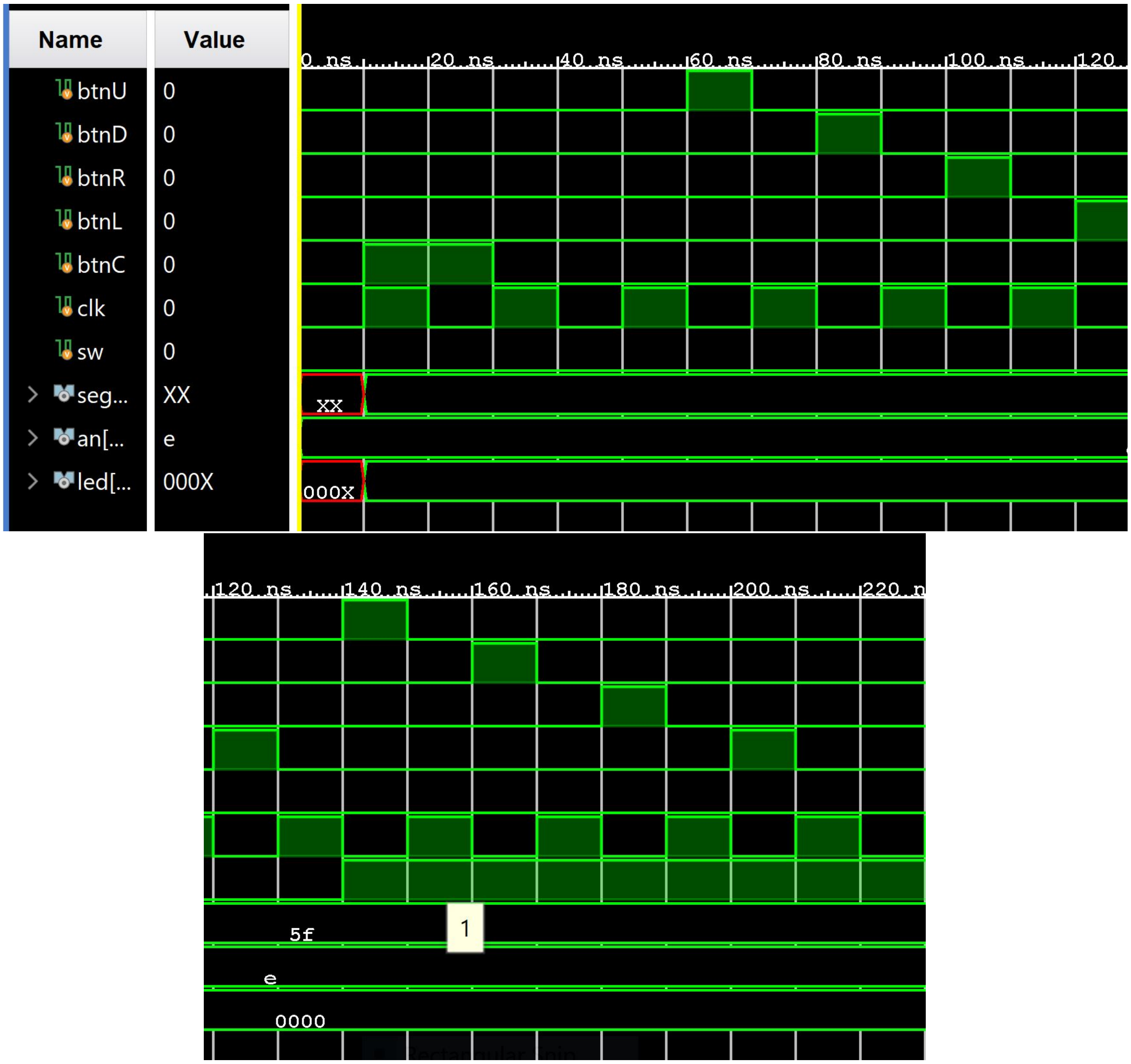
Time (ms):	0	25	50	75	100	125	150	175	200	225	250	275	300	325	350	375	400
clk	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
rst	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b	0	0	0	1	2	1	0	0	1	0	4	0	1	8	b	0	0
y	X	1	2	4	8	8	1	1	1	1	2	2	1	1	2	2	1
win	X	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0
lose	X	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	1



Name	Value
clk	0
reset	0
b[3:0]	0
y[3:0]	1
win	0
lose	0

Figure 3: guess FSM simulation waveform and ERT

Time (ms):	0	20	40	60	80	100	120	140	160	180	200	220
btnU	0	0	0	1	0	0	0	1	0	0	0	0
btnD	0	0	0	0	1	0	0	0	1	0	0	0
btnR	0	0	0	0	0	1	0	0	0	1	0	0
btnL	0	0	0	0	0	0	1	0	0	0	1	0
btnC	0	1	0	0	0	0	0	0	0	0	0	0
clk	1	1	1	1	1	1	1	1	1	1	1	1
sw	0	0	0	0	0	0	0	1	1	1	1	1
seg	X	5f	5f	5f	5f	5f	5f	5f	5f	5f	5f	5f
an	e	e	e	e	e	e	e	e	e	e	e	e
led	X	0	0	0	0	0	0	0	0	0	0	0



Name	Value
btnU	0
btnD	0
btnR	0
btnL	0
btnC	0
clk	0
sw	0
seg...	XX
an[...	e
led[...	000X

Figure 4: guess FSM simulation waveform and ERT

Figure 5: sseg4TDM Schematic

Figure 6: calclab10 Schematic

Code

Listing 1: guess FSM Module

```
‘timescale 1ns / 1ps
//ELC 2137, Jake Simmons, 2020-04-20

module guess_FSM (
   input [3:0]b,
   input reset,
   input clk,
   output reg [3:0]y,
   output reg win,
   output reg lose
   );

   //states
   localparam [2:0]
      S0 = 3’b000,
      S1 = 3’b001,
      S2 = 3’b010,
      S3 = 3’b011,
      SWIN = 3’b100,
      SLOSE = 3’b101;

   //internal signals
   reg[2:0] nState, State;

   always_ff @(posedge clk or posedge reset)
      if(reset == 1) begin
         State <= S0;
      end
      else begin
         State <= nState;
      end


   always_comb begin
      case(State)
      S0: begin
         y[0] = 1;
         y[3:1] = 0;
         lose = 1’b0;
         win = 1’b0;
         if(b==1)
            nState = SWIN;
         else if(b==0)
            nState = S1;
         else
            nState = SLOSE;
         end

      S1: begin
         y[1] = 1;
         y[0] = 0;
         y[3:2] = 0;
         if(b==2)
            nState = SWIN;
         else if(b==0)
            nState = S2;
         else
            nState = SLOSE;
         end

      S2: begin
         y[2] = 1;
         y[3] = 0;
         y[1:0] = 0;
         if(b==4)
            nState = SWIN;
         else if(b==0)
            nState = S3;
         else
            nState = SLOSE;
         end

         S3: begin
            y[3] = 1;
            y[2:0] = 0;
            if(b==8)
               nState = SWIN;
            else if(b==0)
               nState = S0;
            else
               nState = SLOSE;
            end

            SWIN: begin
               win = 1'b1;
               lose = 1'b0;
               if(b==0)
                  nState = S0;
               else
                  nState = SWIN;
               end

            SLOSE: begin
               lose = 1'b1;
               win = 1'b0;
               if(b==0)
                  nState = S0;
               else
                  nState = SLOSE;
               end
            endcase
         end
endmodule
```

Listing 2: guess FSM Test Bench

```
‘timescale 1ns / 1 ps
//ELC 2137, Jake Simmons, 2020-04-21
module guess_FSM_test();

   reg clk, reset;
   reg [3:0] b;
   wire [3:0] y;
   wire win, lose;

   guess_FSM gs( .clk(clk), .b(b), .reset(reset), .y(y), .win(win),
   .lose(lose) );

   always begin
      clk = ~clk; #10;
   end

   initial begin
      clk = 0; reset = 0; b = 0; #10;
      reset = 1; #10;
      reset = 0; #10;

      b = 0; #10; //S1
      b = 0; #10; //S2
      b = 0; #10; //S3

      b = 0; #10; //S0
      b = 1; #20; //SWIN
      b = 2; #10; //SWIN

      b = 0; #10 //S0;
      b = 2; #10; //SLOSE
      b = 1; #10; //SLOSE

      b = 0; #10; //S0
      b = 0; #10; //S1
      b = 2; #10; //SWIN
      b = 2; #10; //SWIN

      b = 0; #10; //S0
      b = 0; #10; //S1
      b = 1; #10; //SLOSE
      b = 1; #10; //SLOSE

      b = 0; #10; //S0
      b = 0; #10; //S1
      b = 0; #10; //S2
      b = 4; #10; //SWIN
      b = 2; #10; //SWIN
      b = 0; #10; //S0
      b = 0; #10; //S1
      b = 0; #10; //S2
      b = 1; #10; //SLOSE
      b = 1; #10; //SLOSE
      b = 0; #10; //S0

      b = 0; #10; //S1
      b = 0; #10; //S2
      b = 0; #10; //S3
      b = 8; #10; //SWIN
      b = 2; #10; //SWIN
      b = 0; #10; //S0
      b = 0; #10; //S1
      b = 0; #10; //S2
      b = 0; #10; //S3
      b = 1; #10; //SLOSE
      b = 1; #10; //SLOSE
      b = 0; #10; //S0


   end
endmodule
```

Listing 3: sseg4TDM Module

```
‘timescale 1ns / 1ps
//ELC 2137 2020-4-7

module sseg4_TDM(
   input clock,
   input reset,
   input [15:0] data,
   input hex_dec,
   input sign,
   output [6:0] seg,
   output dp,
   output [3:0] an
   );

   wire [1:0] digit_sel;
   wire [15:0] W1 ;
   wire [15:0] W2 ;
   wire [3:0] W3;
   wire [6:0] W4;
   wire [3:0] W5;
   wire W6;
   wire [1:0] W7;

   Counter #(.N(18)) timer( .clk(clock), .en(1’b1), .tick(W7), .rst(reset)
      );

   Counter #(.N(2)) counter2( .clk(clock),.en(W7),.count(digit_sel), .rst(
      reset) );

   BCD11_2 B1( .in11(data[10:0]), .out11(W1));

   mux2 #(.N(16)) B2( .in0(data), .in1(W1), .sel(hex_dec), .out(W2));
   mux4 B3( .in0(W2[3:0]), .in1(W2[7:4]), .in2(W2[11:8]), .in3(W2[15:12]),
       .sel(digit_sel), .out(W3));

   sseg_decoder B5( .num(W3), .sseg(W4));

   mux2 #(.N(7)) B6( .in0(W4), .in1(7'b0111111), .out(seg) , .sel(W6));

   and G2( W6, sign, ~W5[3]);

   annode_decoder B7( .in(digit_sel), .out(W5));

   assign dp = 1;
   assign an = W5;
endmodule
```

Listing 4: sseg4TDM Test Bench

```
`timescale 1ns / 1ps
//ELC 2137 2020-7-4

module sseg4_TDM_test();
   reg clock;
   reg reset;
   reg [15:0] data;
   reg hex_dec;
   reg sign;
   wire [6:0] seg;
   wire dp;
   wire [3:0] an;

   sseg4_TDM sseg4( .clock(clock), .reset(reset), .data(data), .hex_dec(
      hex_dec),
   .sign(sign), .seg(seg), .dp(dp), .an(an));



   // clock runs continuously

   always begin

      clock = ~clock; #10;

   end

   // this block only runs once

   initial begin
      data = 0; hex_dec = 0; reset = 1;  clock = 0; sign = 0; #2621440;
      data = 1; hex_dec = 1; reset = 0;  sign = 0; #2621440;
      data = 2; hex_dec = 1; reset = 0;  sign = 0;  #2621440;
      data = 3; hex_dec = 1; reset = 0;  sign = 0;  #2621440;
      data = 4; hex_dec = 1; reset = 0;  sign = 0;  #2621440;
      data = 5; hex_dec = 1; reset = 0;  sign = 0;  #2621440;
      data = 6; hex_dec = 1; reset = 0;  sign = 0;  #2621440;
      data = 7; hex_dec = 1; reset = 0;  sign = 0;  #2621440;
      data = 8; hex_dec = 0; reset = 0;  sign = 1;  #2621440;
      data = 9; hex_dec = 0; reset = 0;  sign = 1;  #2621440;
      data = 10; hex_dec = 0; reset = 0;  sign = 1;  #2621440;
      data = 11; hex_dec = 0; reset = 0;  sign = 1;  #2621440;
      data = 12; hex_dec = 1; reset = 0;  sign = 1;  #2621440;
      data = 13; hex_dec = 1; reset = 0;  sign = 1;  #2621440;
      data = 14; hex_dec = 0; reset = 0;  sign = 0;  #2621440;
      data = 15; hex_dec = 0; reset = 0;  sign = 0;  #2621440;

   $finish;

   end
endmodule
```

Listing 5: calclab10 Module

```
‘timescale 1ns / 1ps
//ELC 2137 Jake Simmons 2020-4-8

module calc_lab10(
   input clk,
   input btnU,
   input btnD,
   input [11:0] sw,
   input btnC,
   output [15:0] led,
   output [6:0] seg,
   output dp,
   output [3:0] an
   );
   wire [7:0] W1;

   sseg4_TDM disp_unit( .data({8'b00000000, W1}), .hex_dec(sw[15]),
   .reset(btnC), .clock(clock), .sign(sw[14]),
   .seg(seg), .dp(dp), .an(an));

   top_lab9 calc_unit( .btnU(btnU), .btnD(btnD), .sw(sw[11:0]),
   .clk(clk), .btnC(btnC), .led(led) );

   assign w1 = led[15:8];
endmodule
```