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

Makenna Meyers

April 14, 2020

Summary

This lab uses Time Division Mulitplexing alongside a 7-segment display driver to simplify circuits. The code used can be found in Listings 1, 2, 3, 4, and 5. Schematics for the 7-seg driver and calculator modules can be found in Figures 3 and 4, respectively.

Q/A

1. What are the three main "groups" of the RTL definition of sequential logic?

   state memory, next-state, and output logic

2. Include an annotated figure 10.3b below.

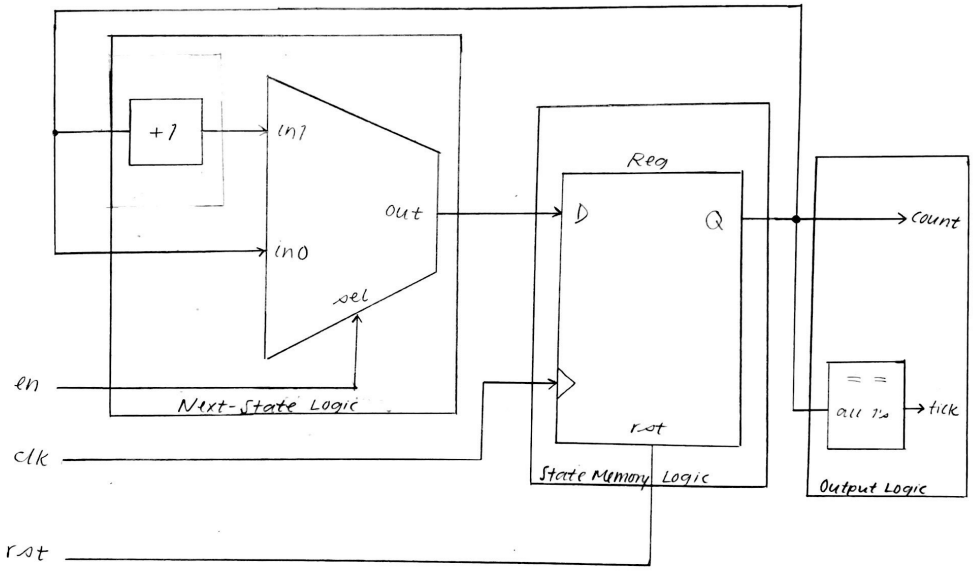

Figure 1: Annotated Figure 10.3b

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 $Q_next = /*?? * /?$

   $Q_next = Q_reg - 1$

Results

Table 1: *counter* expected results table for time 0 through 45 (ns)

Time (ns):	0-5	5-10	10-15	15-20	20-25	25-30	30-35	35-40	40-45
clk	0	1	0	1	0	1	0	1	0
en	1	1	1	1	1	1	1	1	1
rst	1	1	1	1	1	0	1	1	1
count	0	0	0	0	0	0	0	0	0
tick	0	0	0	0	0	0	0	0	0

Table 2: *counter* expected results table for time 45 through 90 (ns)

Time (ns):	45-50	50-55	55-60	60-65	65-70	70-75	75-80	80-85	85-90
clk	1	0	1	0	1	0	1	0	1
en	1	0	0	0	0	0	0	1	1
rst	1	0	0	0	0	0	0	0	0
count	0	0	0	0	0	0	0	0	0
tick	0	0	0	0	0	0	0	0	0

Table 3: *counter* expected results table for time 90 through 135 (ns)

Time (ns):	90-95	95-100	100-105	105-110	110-115	115-120	120-125	125-130	130-135
clk	0	1	0	1	0	1	0	1	0
en	0	0	1	1	1	1	1	1	1
rst	0	0	0	0	0	0	0	0	0
count	0	0	0	1	1	2	2	3	3
tick	0	0	0	0	0	0	0	1	1

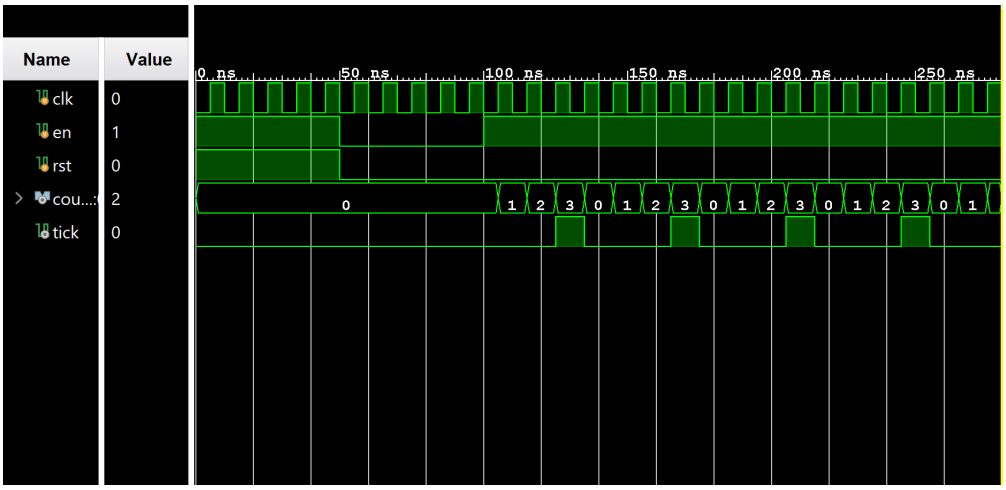

Figure 2: Counter Simulation Waveform

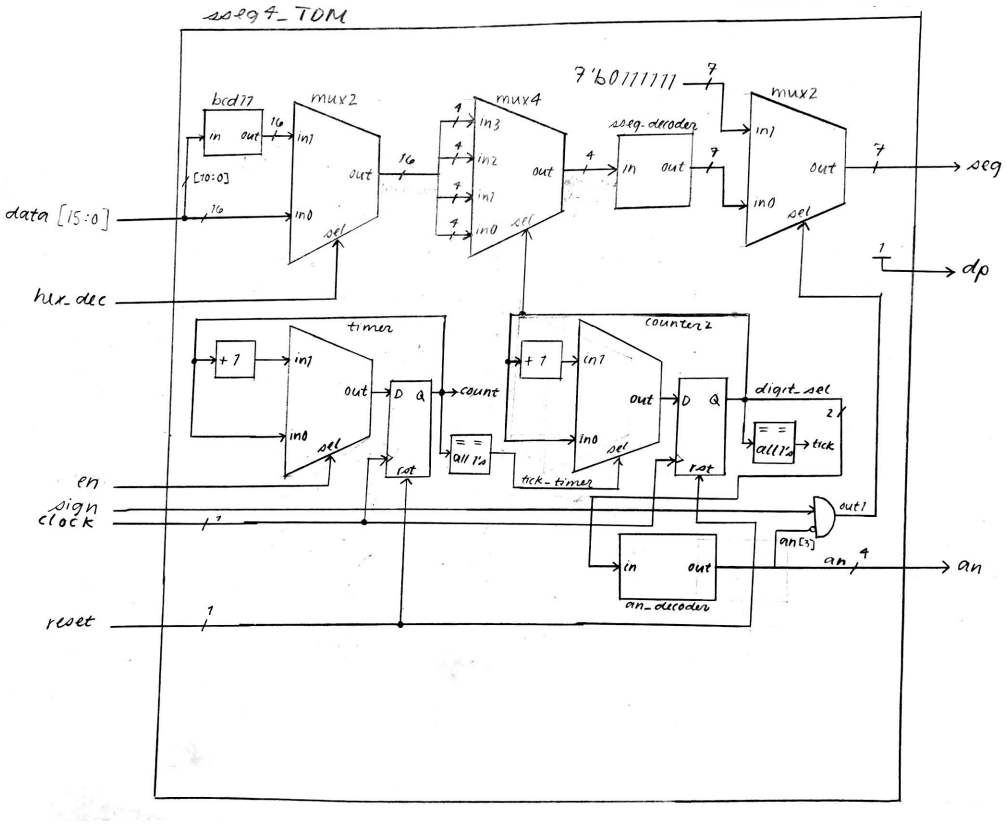

Figure 3: 7-seg Driver Schematic

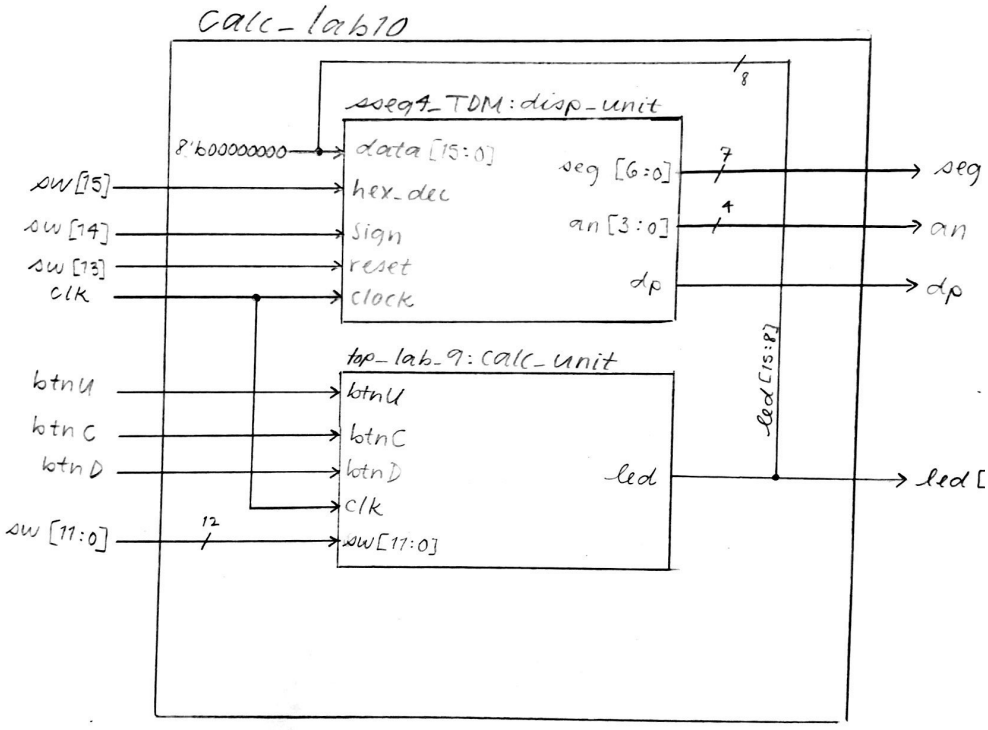

Figure 4: Calculator Schematic

Code

Listing 1: Counter Module

```
`timescale 1ns / 1ps
// Makenna Meyers , ELC 2137 , 2020-4-09


module counter #( parameter N =1)
(
    input clk , rst , en ,
    output [N -1:0] count ,
    output tick
) ;

// internal signals
    reg [N -1:0] Q_reg , Q_next ;

// register ( state memory )
    always @ ( posedge clk , posedge rst )
    begin
    if ( rst )
      Q_reg <= 0;
    else
      Q_reg <= Q_next ;
    end

// next - state logic
    always @ *
    begin
    if ( en )
      Q_next = Q_reg + 1;
    else
      Q_next = Q_reg ; // no change
    end

// output logic
    assign count = Q_reg ;
    assign tick = ( Q_reg =={ N {1'b1}}) ? 1'b1 : 1'b0 ;

endmodule // counter
```

Listing 2: Counter Test Bench

```
`timescale 1ns / 1ps
// Makenna Meyers , ELC 2137 , 2020-4-09


module counter_test ();

    reg clk , en , rst ;
    wire [1:0] count;
    wire tick;

    counter #(.N(2)) r(.count(count), .clk(clk),
```

```
    .en(en), .rst(rst), .tick(tick) );

// clock runs continuously
always begin
    clk = ~clk ; #5;
end

// this block only runs once
initial begin
clk = 0; en = 1; rst = 1; #5;
clk = 1; en = 1; rst = 1; #5;
clk = 0; en = 1; rst = 1; #5;
clk = 1; en = 1; rst = 1; #5;
clk = 0; en = 1; rst = 1; #5;
clk = 1; en = 1; rst = 1; #5;
clk = 0; en = 1; rst = 1; #5;
clk = 1; en = 1; rst = 1; #5;
clk = 0; en = 1; rst = 1; #5;
clk = 1; en = 1; rst = 1; #5;
clk = 0; en = 0; rst = 0; #5;
clk = 1; en = 0; rst = 0; #5;
clk = 0; en = 0; rst = 0; #5;
clk = 1; en = 0; rst = 0; #5;
clk = 0; en = 0; rst = 0; #5;
clk = 1; en = 0; rst = 0; #5;
clk = 0; en = 0; rst = 0; #5;
clk = 1; en = 0; rst = 0; #5;
clk = 0; en = 0; rst = 0; #5;
clk = 1; en = 0; rst = 0; #5;
clk = 0; en = 1; rst = 0; #5;
clk = 1; en = 1; rst = 0; #5;
clk = 0; en = 1; rst = 0; #5;
clk = 1; en = 1; rst = 0; #5;
clk = 0; en = 1; rst = 0; #5;
clk = 1; en = 1; rst = 0; #5;
clk = 0; en = 1; rst = 0; #5;
clk = 1; en = 1; rst = 0; #5;
clk = 0; en = 1; rst = 0; #5;
clk = 1; en = 1; rst = 0; #5;
clk = 0; en = 1; rst = 0; #5;
clk = 1; en = 1; rst = 0; #5;
clk = 0; en = 1; rst = 0; #5;
clk = 1; en = 1; rst = 0; #5;
clk = 0; en = 1; rst = 0; #5;
clk = 1; en = 1; rst = 0; #5;
clk = 0; en = 1; rst = 0; #5;
clk = 1; en = 1; rst = 0; #5;
clk = 0; en = 1; rst = 0; #5;
clk = 1; en = 1; rst = 0; #5;
clk = 0; en = 1; rst = 0; #5;
clk = 1; en = 1; rst = 0; #5;
clk = 0; en = 1; rst = 0; #5;
clk = 1; en = 1; rst = 0; #5;
clk = 0; en = 1; rst = 0; #5;
```

```
    clk = 1; en = 1; rst = 0; #5;
    clk = 0; en = 1; rst = 0; #5;
    clk = 1; en = 1; rst = 0; #5;
    clk = 0; en = 1; rst = 0; #5;
    clk = 1; en = 1; rst = 0; #5;
    clk = 0; en = 1; rst = 0; #5;
    clk = 1; en = 1; rst = 0; #5;
    clk = 0; en = 1; rst = 0; #5;
    clk = 1; en = 1; rst = 0; #5;
    clk = 0; en = 1; rst = 0; #5;
    clk = 1; en = 1; rst = 0; #5;
    $finish ;
    end
endmodule
```

Listing 3: 7-seg Driver Module

```
`timescale 1ns / 1ps
// Makenna Meyers , ELC 2137 , 2020-04-09


module sseg4_TDM(

input [15:0]data,
input hex_dec, sign, reset, clock, en,

output [6:0]seg,
output [3:0]an,
output dp

);

wire [15:0]bcd_out;
wire [15:0]hex_dec_out;
wire [3:0]dig_sel_out;
wire [6:0]sseg_decode_out;
wire out1;
wire [1:0]count_timer;
wire [1:0]tick_timer;
wire [1:0]digit_sel;
wire [1:0]tick_counter2;

BCDmod11 bcd(.in1(data[10:0]),.out11(bcd_out));
mux2 #(.N(16)) mux_hexdec(.in0(data), .in1(bcd_out), .out(hex_dec_out), .
   sel(hex_dec));
mux4 #(.N(4)) mux_digit_sel(.in3(hex_dec_out[15:12]), .in2(hex_dec_out
   [11:8]), .in1(hex_dec_out[7:4]),
            .in0(hex_dec_out[3:0]), .out(dig_sel_out), .sel(digit_sel));
sseg_decoder s(.num(dig_sel_out),.sseg(sseg_decode_out));
mux2 #(.N(7)) mux_sseg(.in0(sseg_decode_out), .in1(7'b0111111), .out(seg),
    .sel(out1));
an_decode ad(.digit_sel(digit_sel), .an(an));
and a1(out1, sign, ~an[3]);
counter #(.N(18)) timer(.clk(clock), .en(en), .rst(reset), .count(
```

```
   count_timer), .tick(tick_timer));
counter #(.N(2)) counter2(.clk(clock), .en(tick_timer), .rst(reset), .
   count(digit_sel), .tick(tick_counter2));

endmodule
```

Listing 4: 7-seg Driver Test Bench

```
`timescale 1ns / 1ps
// Makenna Meyers , ELC 2137 , 2020-4-09


module sseg4_TDM_test ();

    reg clk , en, rst, sign, hex_dec;
    reg [15:0]data;

    wire [6:0] seg;
    wire[3:0] an;
    wire dp;

    sseg4_TDM s4test(.data(data), .hex_dec(hex_dec), .sign(sign), .en(en),
        .reset(rst), .clock(clk), .seg(seg), .an(an), .dp(dp));

    always begin
        clk = ~clk; #1;
    end

    // this block only runs once
    initial begin
        clk = 0; rst = 0; #5
        rst = 1; #5
        rst = 0; sign = 0; hex_dec = 1;
        data = 16'b0000000000000; #1000000;
        data = 16'b0000000000001; #1000000;
        data = 16'b0000000000010; #1000000;
        data = 16'b0000000000011; #1000000;
        data = 16'b0000000000100; #1000000;
        data = 16'b0000000000101; #1000000;
        data = 16'b0000000000110; #1000000;
        data = 16'b0000000000111; #1000000;
        data = 16'b0000000001000; #1000000;
        data = 16'b0000000001001; #1000000;
        data = 16'b0000000001010; #1000000;
        data = 16'b0000000001011; #1000000;
        data = 16'b0000000001100; #1000000;
        data = 16'b0000000001101; #1000000;
        data = 16'b0000000001110; #1000000;
        data = 16'b0000000001111; #1000000;
        data = 16'b0000000010000; #1000000;
        data = 16'b0000000010001; #1000000;
        data = 16'b0000000010010; #1000000;
        data = 16'b0000000010011; #1000000;
        data = 16'b0000000010100; #1000000;
```

```
        data = 16'b000000010101; #1000000;
        data = 16'b000000010110; #1000000;
        data = 16'b000000010111; #1000000;
        data = 16'b000000011000; #1000000;
        data = 16'b000000011001; #1000000;
        data = 16'b000000011010; #1000000;
        data = 16'b000000011011; #1000000;
        data = 16'b000000011100; #1000000;
        data = 16'b000000011101; #1000000;
        data = 16'b000000011110; #1000000;
        data = 16'b000000011111; #1000000;
        $finish;
    end

endmodule
```

Listing 5: Calculator Module

```
`timescale 1ns / 1ps
// Makenna Meyers , ELC 2137 , 2020-4-09


module calc_lab10(

    input btnU, btnD, btnC, clk, en,
    input [15:0]sw,

    output [6:0]seg,
    output [3:0]an,
    output [15:0]led,
    output dp
);

    top_lab_9 calc_unit(.btnU(btnU), .btnC(btnC), .btnD(btnD), .clk(clk),
       .sw(sw[11:0]), .led(led));
    sseg4_TDM disp_unit(.data({8'b00000000, led[15:8]}), .hex_dec(sw[15]),
        .sign(sw[14]), .reset(sw[13]), .clock(clk), .en(1), .seg(seg), .an
       (an), .dp(1));
endmodule
```