

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

Justin Woods

April 15, 2020

Summary

In this lab, we used synchronous design methodology to design a calculator. This was done by creating a counter module which was tested. This was done using a similar method as creating a register, but differed in that it has no input. A form of sseg4 was created to combine the counter and a display driver, also tested. This 4-digit display was then combined with the top level calculator form Lab 9 and implemented into the Basys3 board.

Questions

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

State memory , next-state, output logic

2. Annotated Counter-timer

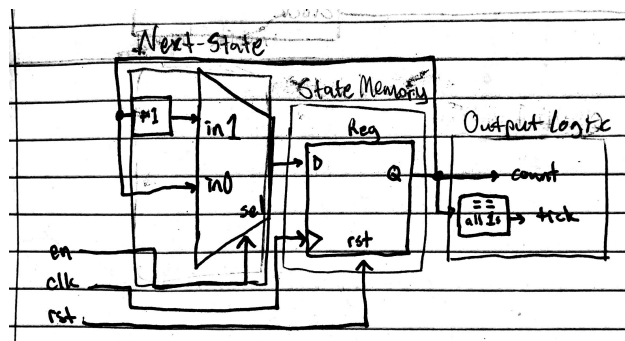


Figure 1: Annotated counter timer

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 $Q_{next} = /*???*/?$

$Q_{reg} - 1;$

Results

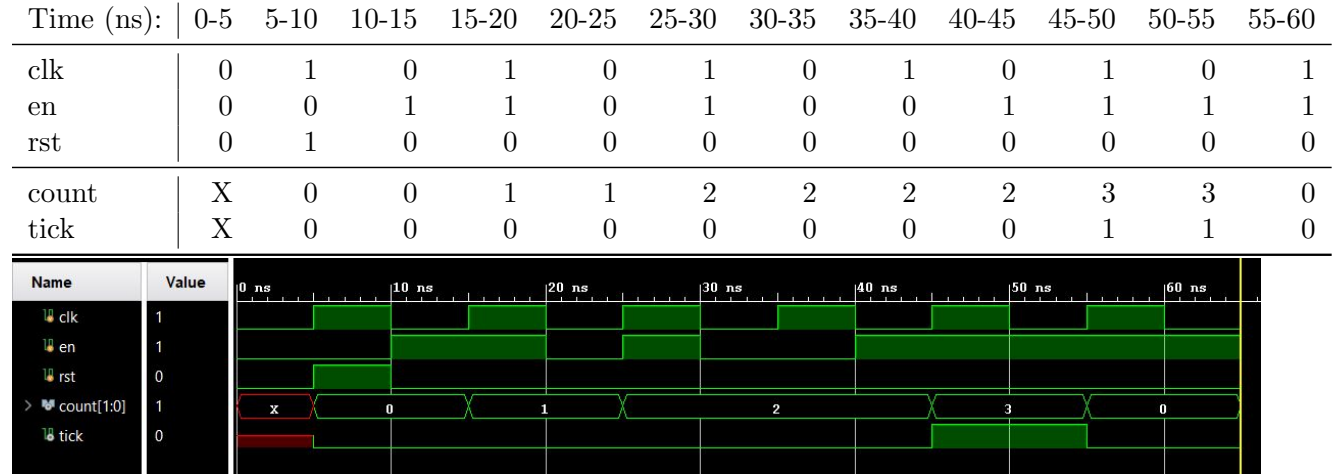


Figure 2: Counter Simulation Waveform and ERT

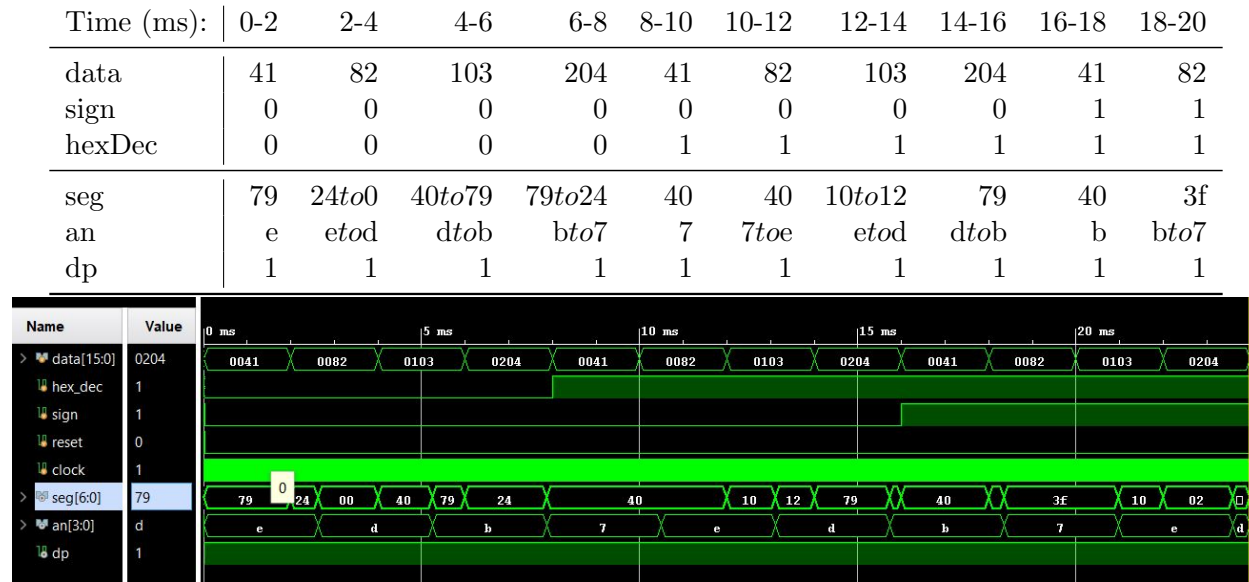


Figure 3: sseg4TDM Simulation Waveform and ERT

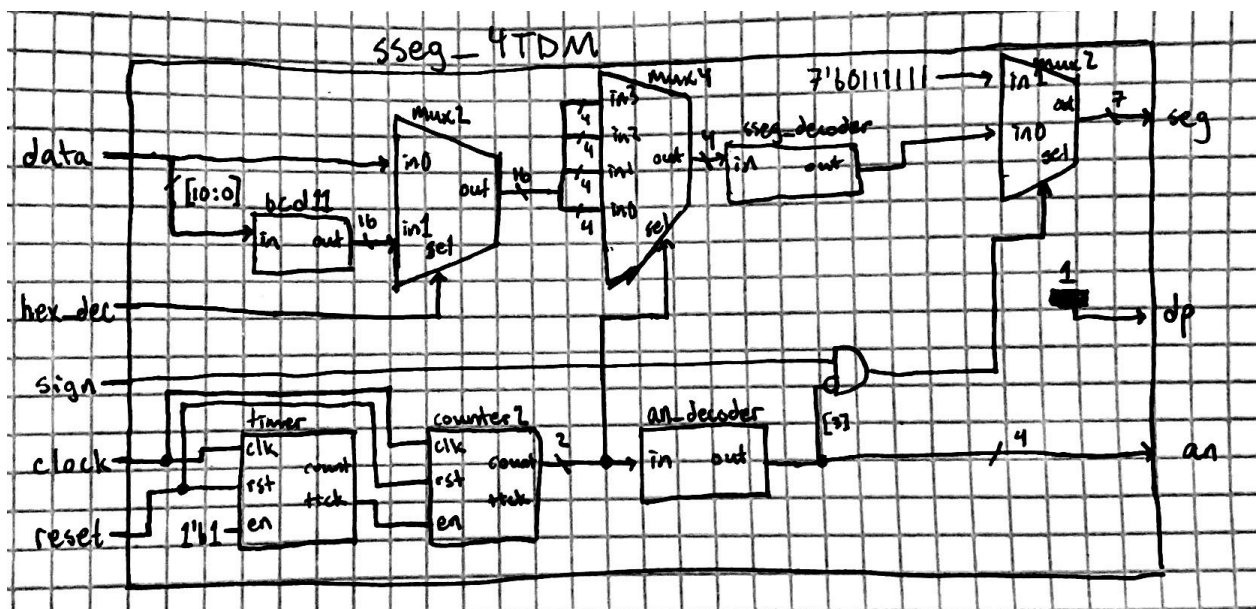


Figure 4: sseg4tdm Schematic

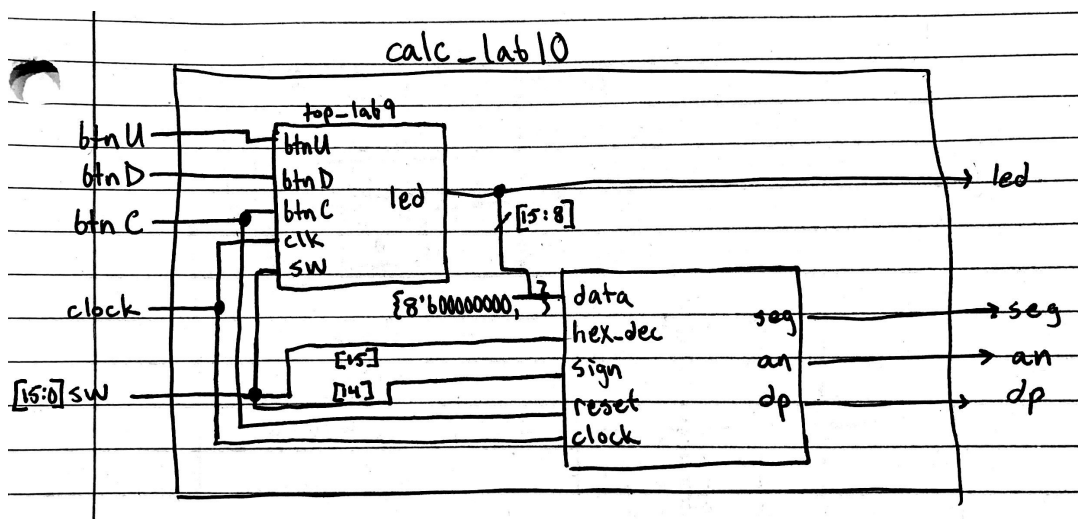


Figure 5: calclab10 Schematic

Code

Listing 1: Counter Source Code

```
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
```

Listing 2: Counter Test

```

module counterTest();

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

    counter #(.N(2)) c(.count(count), .clk(clk),
        .en(en), .rst(rst),
        .tick(tick) );

    always begin
        clk = ~clk; #5;
    end

    // this block only runs once
    initial begin
        clk=0; en=0; rst=0; ; #5;
        rst = 1; #5;
        // reset
        en = 1; rst = 0; #10;
        en = 0;      #5;
        en = 1;      #5;
        en = 0;      #10;
        en = 1;      #5;
        $finish;
    end
endmodule

```

Listing 3: Sseg4TDM Source Code

```

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

    wire [15:0] mux2mux4, bcd2mux2;
    wire [6:0] ssegmux2;
    wire [3:0] mux4sseg, anan;
    wire [1:0] digit_sel;
    wire andmux2;
    wire timer2counter;

    bcd11 bcd11in(
        .in(data[10:0]),
        .out(bcd2mux2)
    );

    mux2 #(.N(16)) mux2num1(
        .in0(data), .in1(bcd2mux2), .sel(hex_dec),
        .out(mux2mux4)
    );

```

```

    );
    mux4 mux4num1(
        .in3(mux2mux4[15:12]), .in2(mux2mux4[11:8]),
        .in1(mux2mux4[7:4]), .in0(mux2mux4[3:0]),
        .sel(digit_sel),
        .out(mux4sseg)
    );
    sseg_decoder ssegdecode(
        .in(mux4sseg),
        .out(ssegmux2)
    );
    mux2 #(N(7)) mux2num2(
        .in1(7'b0111111), .in0(ssegmux2), .sel(andmux2),
        .out(seg[6:0])
    );
    and A1(
        andmux2,
        sign, ~anan[3]
    );
    an_decoder anddecode(
        .in(digit_sel),
        .out(anan)
    );

    counter #(N(18)) timer(
        .en(1'b1), .clk(clock), .rst(reset),
        .count(), .tick(timer2counter)
    );

    counter #(N(2)) counter2(
        .en(1'b1), .clk(timer2counter), .rst(reset),
        .count(digit_sel), .tick()
    );

    assign an = anan;
    assign dp = 1;

endmodule

```

Listing 4: Sseg4TDM Test

```

module sseg4_TDMTEST();

    reg [15:0] data;
    reg hex_dec, sign;
    reg reset, clock;
    wire [6:0] seg;
    wire [3:0] an;
    wire dp;

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

    always begin
        clock = ~clock; #5;
    end

    // this block only runs once
    initial begin
        clock = 0; reset=0; #5;
        reset = 1; #5;

        // reset
        reset = 0; sign = 0; hex_dec = 0;
        data = 16'b0000000001000001; #2000000;
        data = 16'b0000000010000010; #2000000;
        data = 16'b0000000100000011; #2000000;
        data = 16'b0000001000000100; #2000000;
        reset = 0; sign = 0; hex_dec = 1;
        data = 16'b0000000001000001; #2000000;
        data = 16'b0000000010000010; #2000000;
        data = 16'b0000000100000011; #2000000;
        data = 16'b0000001000000100; #2000000;
        reset = 0; sign = 1; hex_dec = 1;
        data = 16'b0000000001000001; #2000000;
        data = 16'b0000000010000010; #2000000;
        data = 16'b0000000100000011; #2000000;
        data = 16'b0000001000000100; #2000000;

        $finish;
    end
endmodule

```

Listing 5: CalcLab10

```
module calc_lab10(  
    input btnU, btnD, clk, btnC,  
    input [15:0] sw,  
    output [6:0] seg,  
    output [3:0] an,  
    output [15:0] led,  
    output dp  
);  
  
    top_lab9 calc_unit(  
        .btnU(btnU), .btnD(btnD), .clk(clk), .btnC(btnC),  
        .sw(sw),  
        .led(led)  
    );  
  
    sseg4_TDM disp_unit(  
        .data({8'b00000000, led[15:8]}), .hex_dec(sw[15]),  
        .sign(sw[14]), .reset(btnC), .clock(clk),  
        .seg(seg), .an(an), .dp(dp)  
    );  
  
endmodule
```
