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

Alexander Noll

April 14, 2020

Summary

Building upon previous labs, we utilized registers and sequential logic to design timers. Using these timers, the calculator and 7-segment display driver were combined to create a 4 digit calculator with display.

Q&A

1. The three main groups of the RTL definition of sequential logic include state memory, next-state, and output logic. 2. 3. Qnext=Qreg[N-2],1.

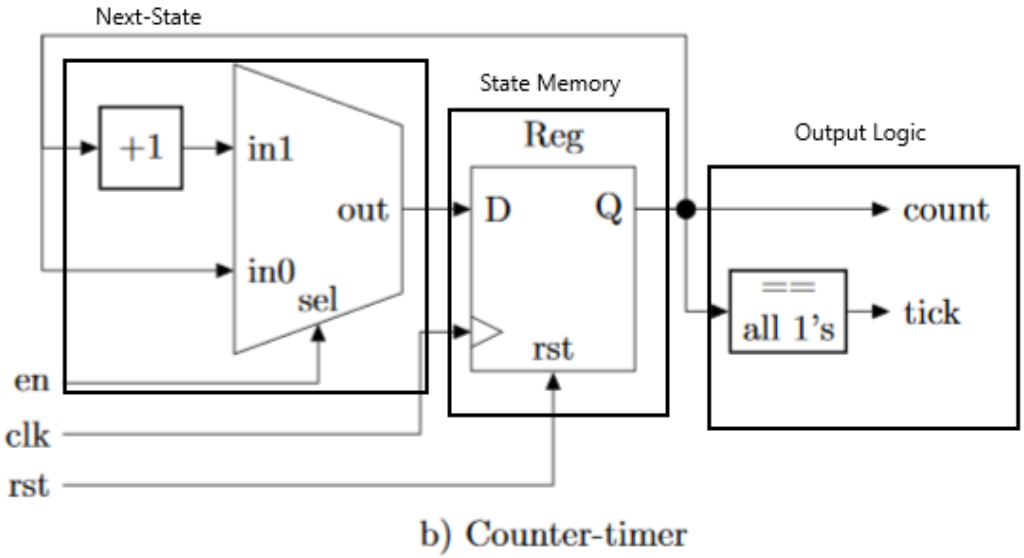

Figure 1: RTL Groups in a Counter Timer

Results

Table 1: *register* expected results table

Time (ns):	0-1	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	9-10	10-11	11-12	12-13	13-14	14-15
clk	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
en	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0
rst	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0
Q (dec)	X	X	X	0	0	0	0	1	1	2	2	3	3	3	3
tick	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

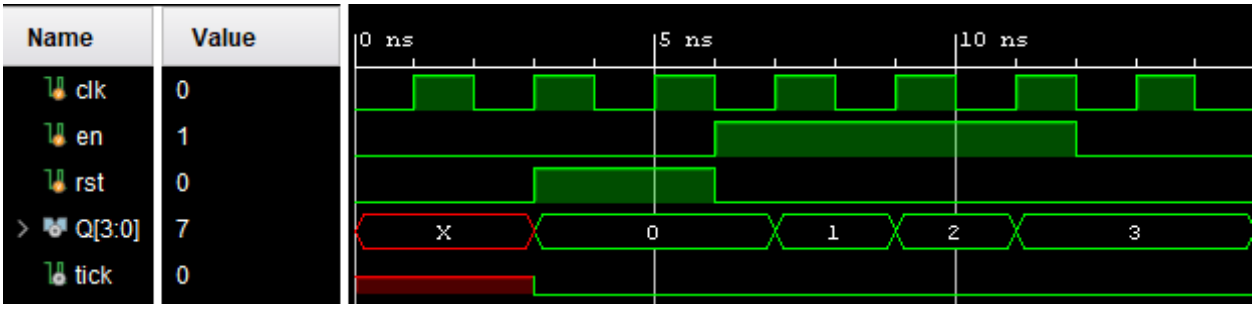

Figure 2: Counter Test Waveform

Table 2: *alu* expected results table skeleton

Clock Cycles:	0-1	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	9-10	10-11	11-12	12-13	13-14	14-15
Input	49	49	49	49	49	49	49	49	49	49	49	49	49	49	49
Reset	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
Seg	40	40	40	40	40	30	30	30	30	30	78	40	40	30	78
an	X	X	X	X	X	e	e	e	e	e	d	b	7	e	d
dp	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

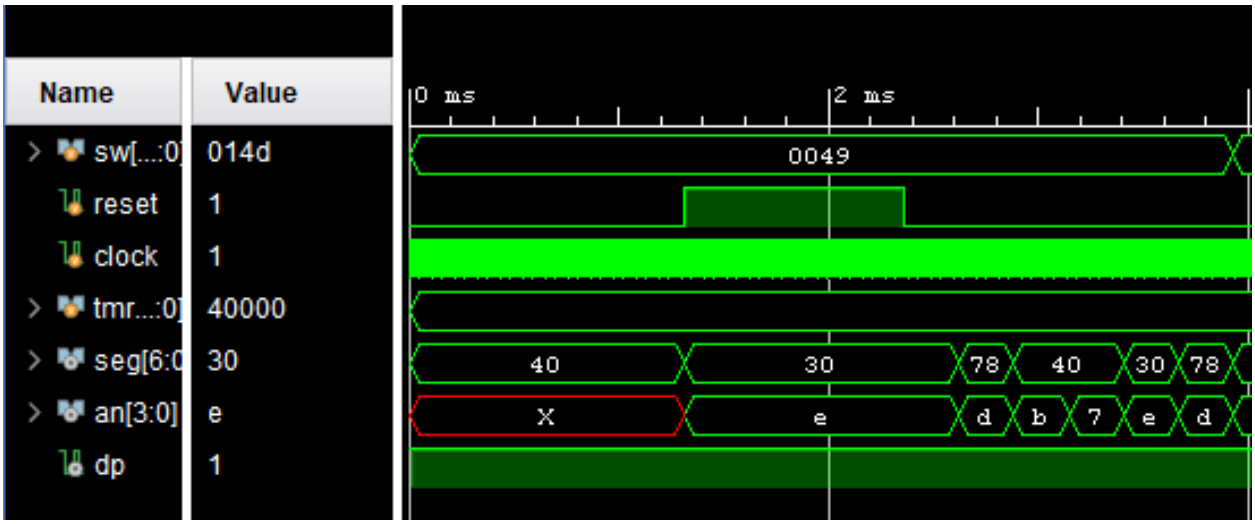

Figure 3: 7-Segment Driver Test Waveform

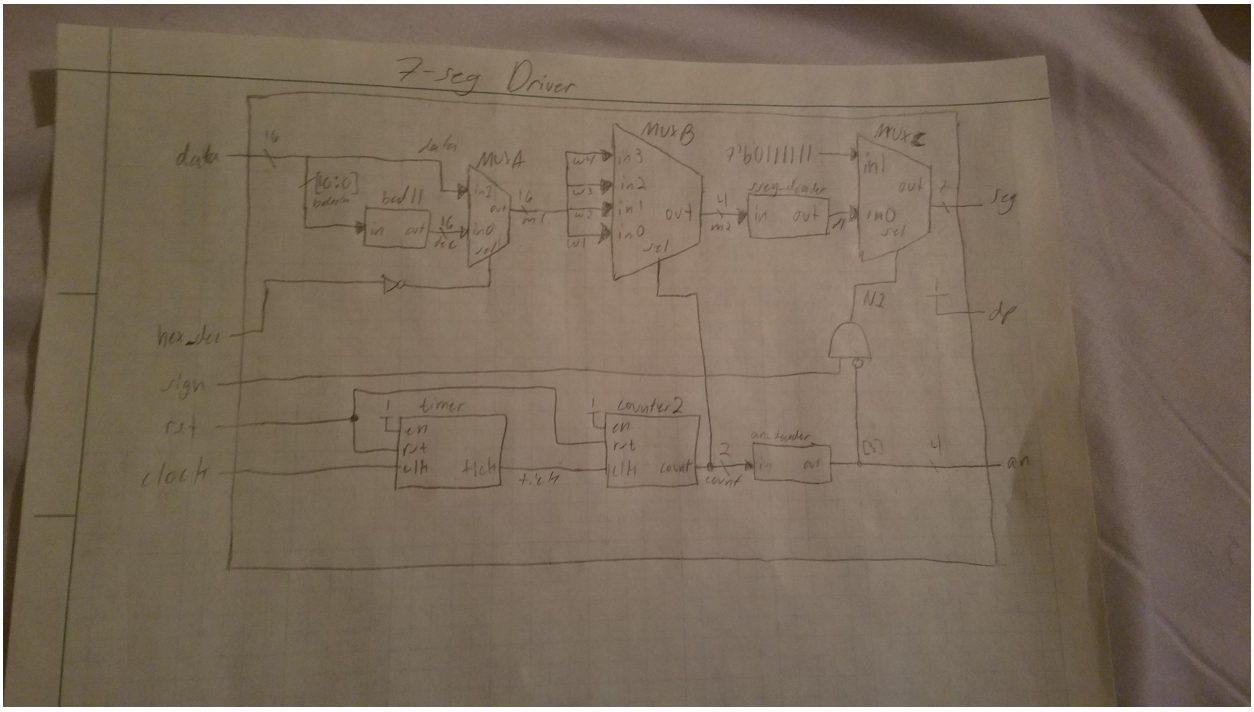

Figure 4: Sseg4 TDM Schematic Drawing

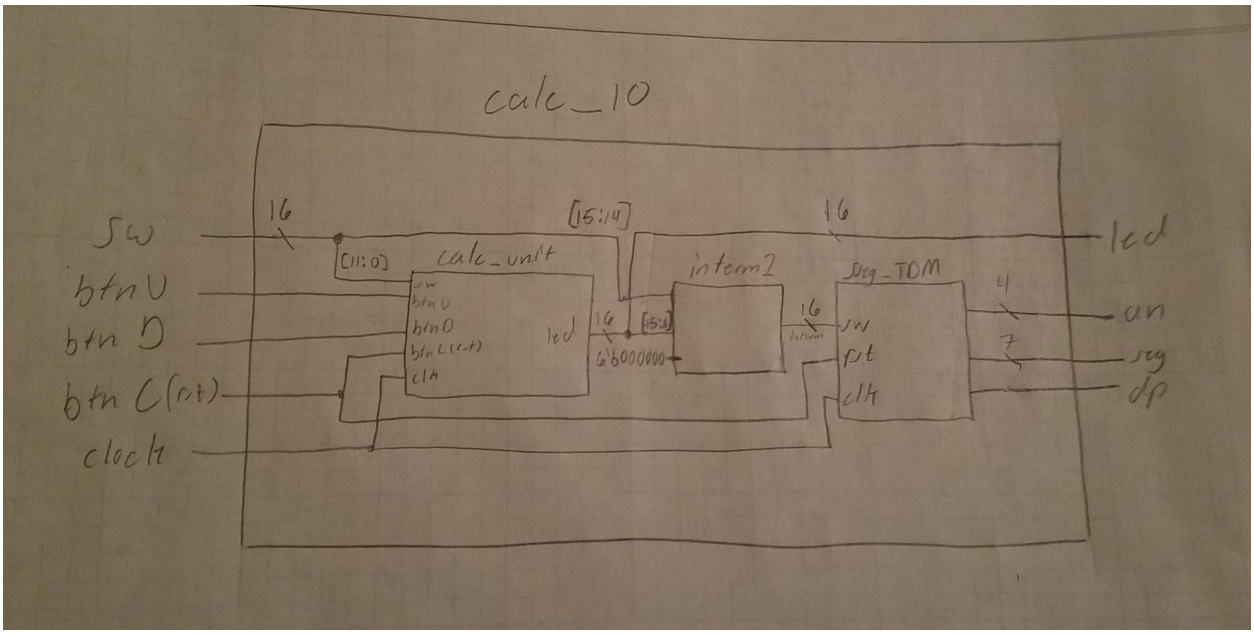

Figure 5: Calc Lab10 Schematic Drawing

Code

Listing 1: Verilog Counter Source

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

Listing 2: Verilog Sseg Source

```
module sseg_TDM(
    input [15:0] sw,
    input reset, clock,
    output [3:0] an,
    output [6:0] seg,
    output dp);
    reg [15:0] data;
    reg hex_dec,sign;
    reg [10:0] bdata;
    wire [15:0] dec;
    wire [15:0] m1;
    wire [3:0] m2;
    wire [6:0] d1;
    wire N1;
    wire [3:0] w1, w2, w3, w4;
    wire [1:0] count;
    wire [17:0] Q;
    wire t,tick;

    assign data = {4'b0000, sw[11:0]};
```

```
    assign w1 = m1[3:0];
    assign w2 = m1[7:4];
    assign w3 = m1[11:8];
    assign w4 = m1[15:12];
    assign N1 = {sign & (~an[3])};
    assign dp = 1;
    assign bdata = sw[10:0];
    assign sign = sw[14];
    assign hex_dec = sw[15];

    BCD11 BCD(.number(bdata), .nout(dec));

    mux2 #(.N(16)) MUXA(.in0(data), .in1(dec), .sel(hex_dec), .out(m1));

    counter  #(.N(19)) timer (.clk(clock), .en(dp), .rst(reset), .count(Q)
       , .tick(tick));

    counter  #(.N(2)) counter2 (.clk(tick), .en(dp), .rst(reset), .count(
       count), .tick(t));

    mux4 #(.N(4)) MUXB(.in0(w1), .in1(w2),.in2(w3), .in3(w4), .dig_sel(
       count), .out(m2));

    an_decode AN1(.dig_sel(count),.an(an));

    sseg_decoder SSEG1(.num(m2), .sseg(d1));

    mux2 #(.N(7)) MUXC(.in0(d1), .in1(7'b0111111), .sel(N1), .out(seg));



endmodule
```

Listing 3: Verilog Top Lab 10 Source

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

    wire [15:0] interm1;

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

    assign interm1={sw[15:14],6'b000000,led[15:8]};

sseg_TDM disp_unit (.sw(interm1),.reset(btnC),.clock(clk),.an(an),.seg(seg
   ),.dp(dp));
```

```
endmodule
```

Listing 4: Verilog Sseg Test Bench

```
module sseg_TDM_test();
reg [15:0] sw;
reg reset, clock;
reg [18:0] tmrcyc;
wire [6:0] seg;
wire [3:0] an;
wire dp;

assign tmrcyc= 19'b1000000000000000000;
sseg_TDM Disp (.sw(sw),.reset(reset),.clock(clock),.seg(seg),.an(an),.dp(
   dp));


    always  begin
    clock = ~clock; #(0.5);
    end

    initial  begin
        clock=0; sw=16'b0000000001001001; reset =0; #(5*tmrcyc);
        reset = 1; #(4*tmrcyc);
        reset = 0; #(6*tmrcyc);
        sw=16'b0000000001011001;     #(4*tmrcyc);
        sw=16'b0000000011001101;     #(6*tmrcyc);
        sw=16'b0000000001100111;     #(4*tmrcyc);
        reset = 0; #(6*tmrcyc);
        reset = 1; #(4*tmrcyc);
        sw=16'b0000000101001101;     #(4*tmrcyc);
        $finish;
    end




endmodule
```

Listing 5: Verilog Counter Test Bench

```
module counter_test();
    reg clk , en , rst;
    wire  [17:0] Q;
    wire tick;
    counter  #(.N(18)) r (.clk(clk),
    .en(en), .rst(rst), .count(Q), .tick(tick));

    always  begin
    clk = ~clk; #(0.5);
    end

    initial  begin
        clk=0; en=0;  rst =0; #3;
        rst = 1; #3;
        en = 1; rst = 0; #6;
        en = 0;       #3;
        en = 1;       #6;
        en = 0;       #3;
        en = 1;       #(19'b1000000000000000001);
        $finish;
    end
endmodule
```