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

Ashlie Lackey

April 15, 2020

Summary

Having previously created a 7-segment display with manual switching between digit displays, we know use a clock to display the digits simultaneously to the eye. By displaying the digit individually, but using a clock to display them at 100MHz, the eye perceives the individual displays lighting up as happening simultaneously to create a 4-digit 7-segment display. In accomplishing this, students gain skills in using synchronous design for sequential circuits, creating a parameterized counter-timer, and use of multiple counters to make a clock-driven 4-digit display.

Q&A

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

   The three main groups are state memory, next-state, and output logic.

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.

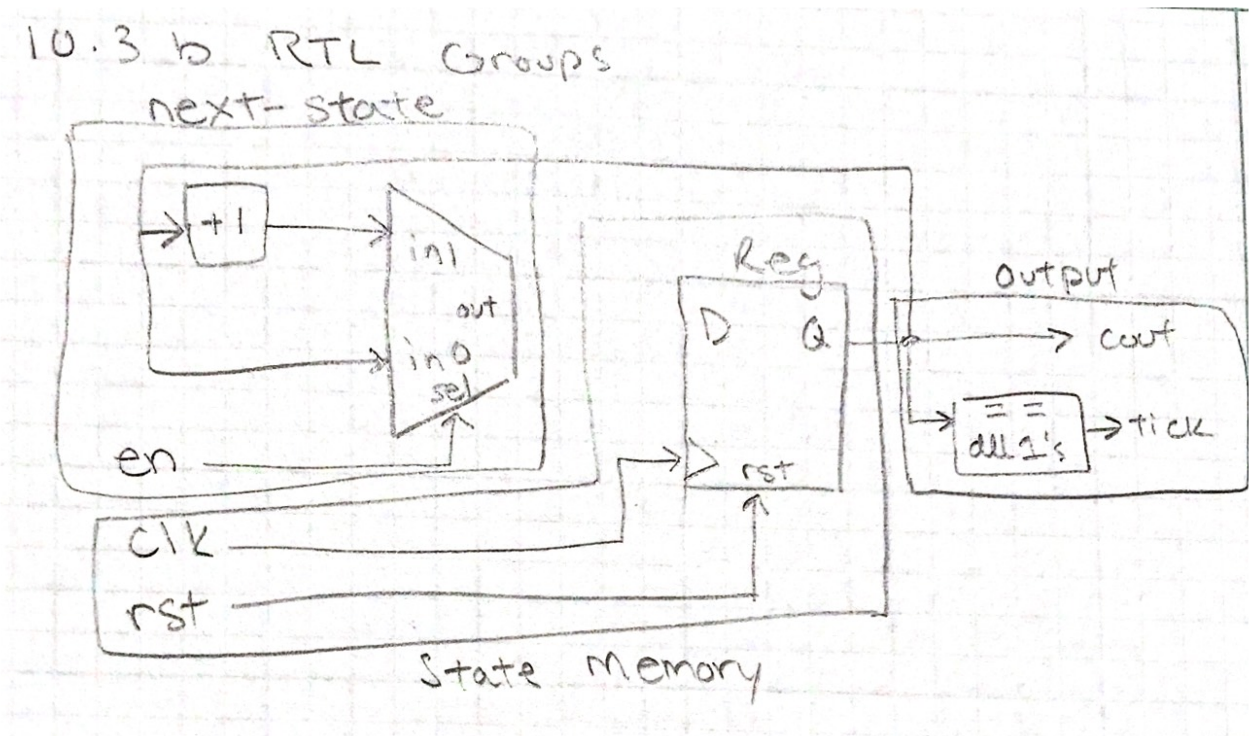

Figure 1: *10.3(b)* RTL Groups Schematic

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 Qnext = /*???*/?

   Qnext = Q_reg - 1'b1

Results

Expected results table, simulation waveforms, and schematic drawings are included in this portion of the report.

NOTE: The sseg4_TDM testbench could not be configured correctly, so the ERT and Waveform are not included

Expected results tables

Table 1: *counter_test* expected results table

Time (ns):	0-5	5-7	7-10	10-15	15-20	20-25	25-30	30-35	35-40	40-45	45-50	...
clk	0	1	1	0	1	0	1	0	1	0	1	...
en	0	0	0	1	0	1	0	1	0	1	0	...
rst	0	0	1	0	0	0	0	0	0	0	0	...
count	X	X	0	0	1	1	2	2	3	3	0	...
tick	X	X	0	0	0	0	0	0	1	1	0	...

Table 2: *counter_test* expected results table

Time (ns):	0-2	2-5	5-10	...	1000005-2000000	2000000-2621435	2621435-3000005
data(hex)	1234	1234 123		...	1234	1234	1234
hex_dec	0	0	0	...	1	0	0
sign	0	0	0	...	0	1	1
reset	0	1	0	...	0	0	0
clock	0	0	5	...			
seg (hex)	X	19	19	...	19	19	30
dp	1	1	1	...	1	1	1
an (hex)	X	e	e	...	e	e	d

Simulation Waveforms

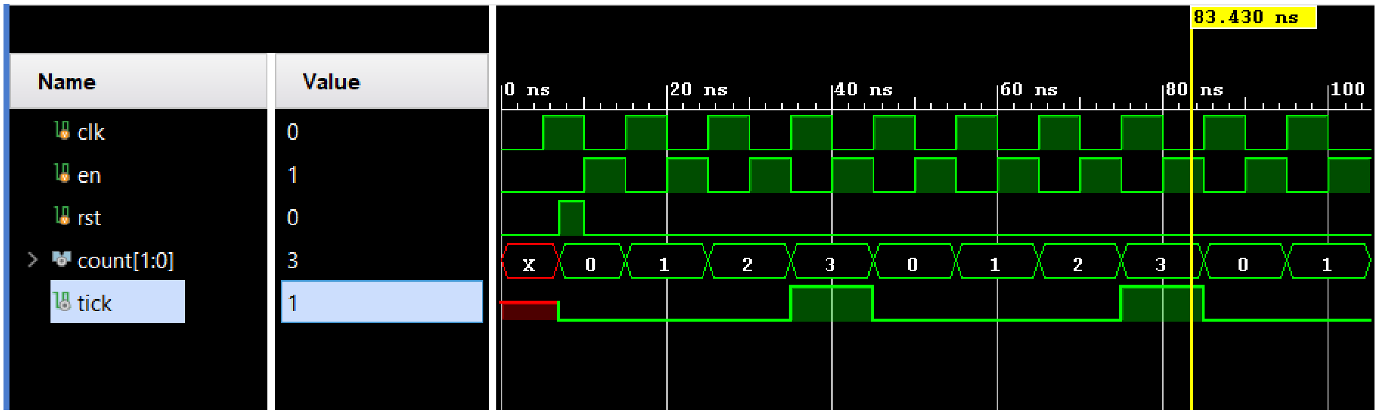

Figure 2: *counter testbench* Simulation Waveform

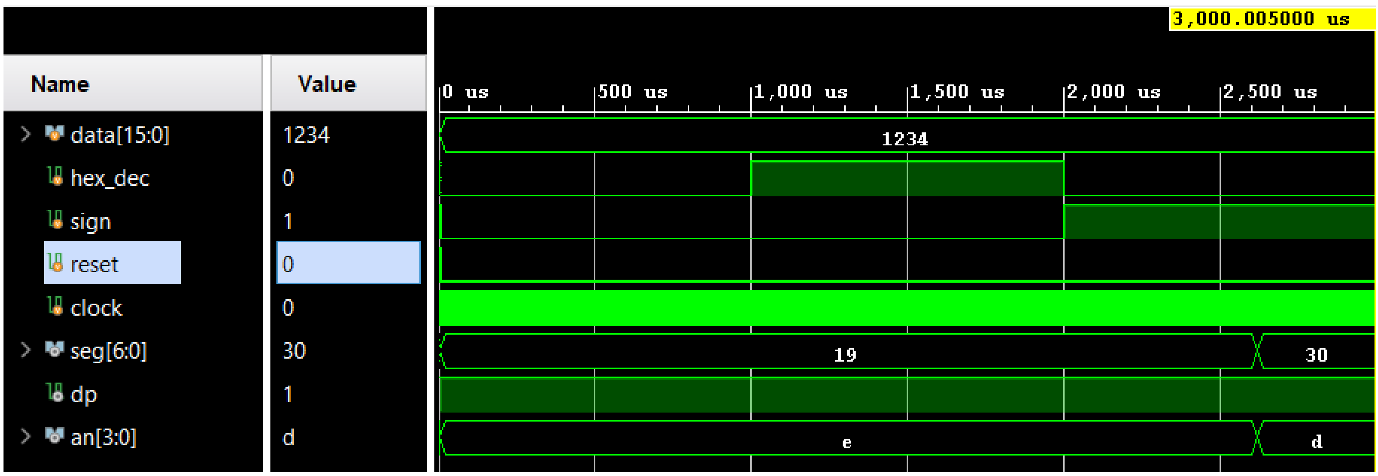

Figure 3: *sseg4_TDM testbench* Simulation Waveform

Schematics

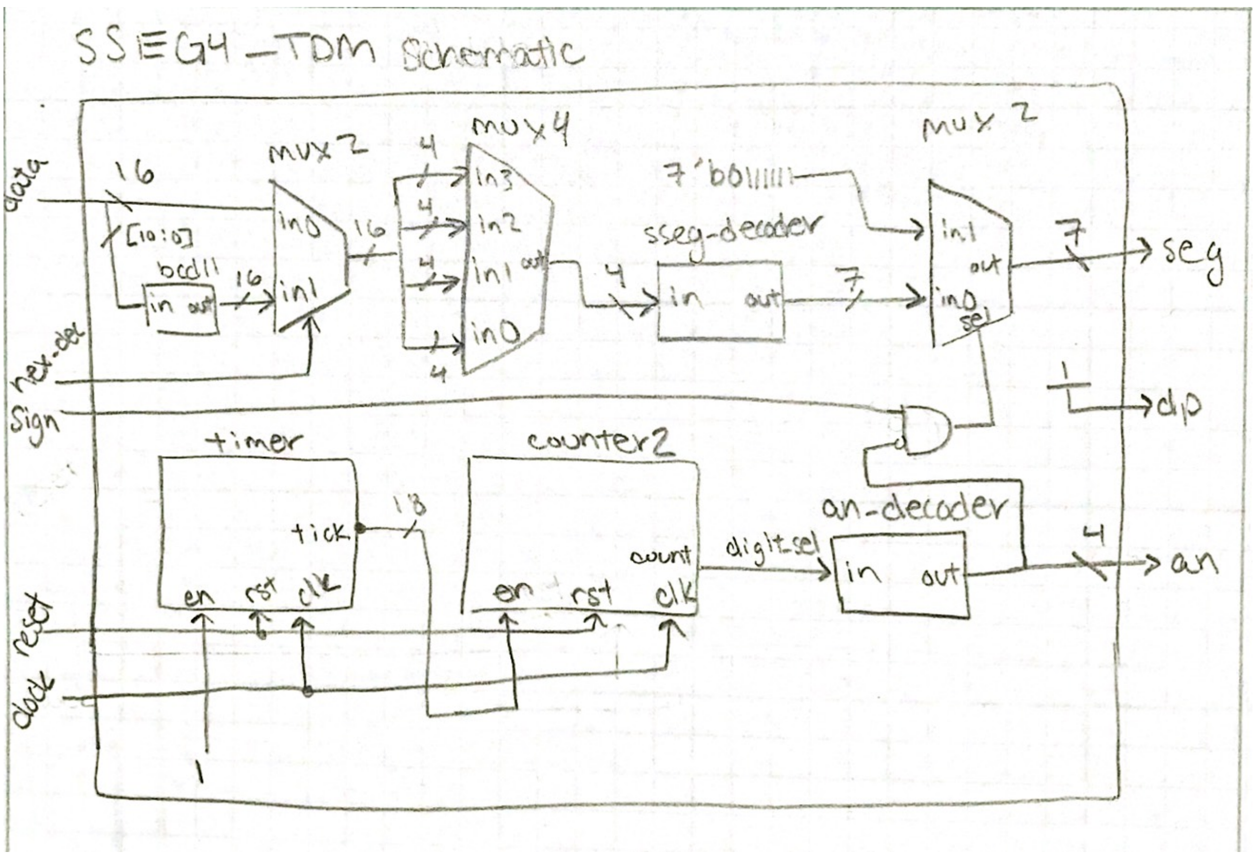

Figure 4: *sseg4_TDM* Module Schematic

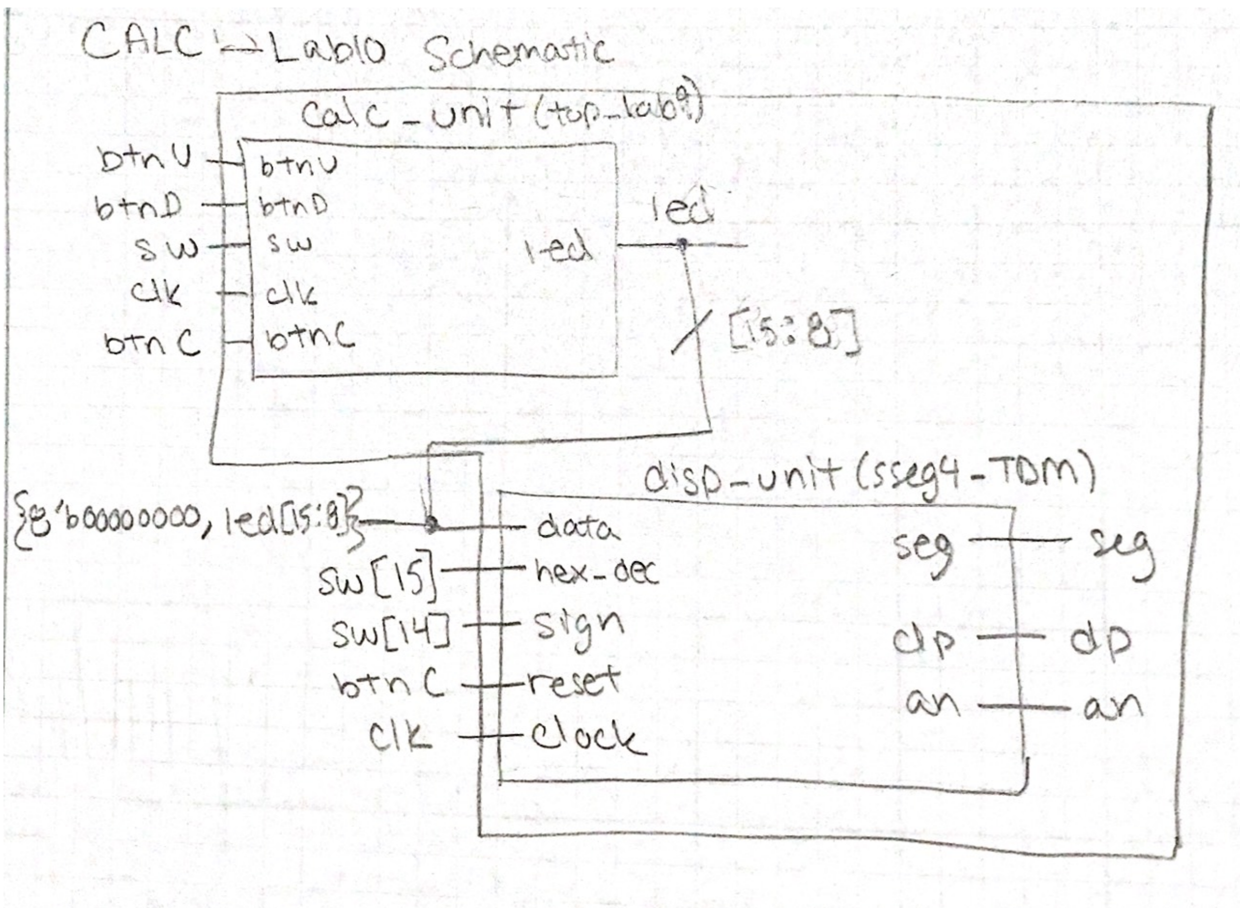

Figure 5: *calc_lab10* Module Schematic

Code

.

Listing 1: Counter Verilog Code

```
‘timescale 1ns / 1ps
// Ashlie Lackey, ELC 2137, 2020-04-08
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’b1; //increase by one
      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: sseg4_TDM Verilog Code

```
‘timescale 1ns / 1ps
// Ashlie Lackey, ELC 2137 , 2020-04-13

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

   wire [17:0] count_dontcare;
   wire tick_out;
   counter #(.N(18)) timer(.clk(clock),.en(1), .rst(reset),
      .count(count_dontcare), .tick(tick_out) );
```

```
   wire [1:0] digit_sel;
   wire tick_dontcare;
   counter #(.N(2)) counter2(.clk(clock),.en(tick_out), .rst(reset),
      .count(digit_sel), .tick(tick_dontcare) );

   wire [15:0] bcd11out ;
   bcd11 TDM_bcd11 (.B(data [10:0]) , .Boutfinal(bcd11out) ) ;

   wire [15:0] mux2_1_out ;
   mux2 #(.N(16)) TDM_mux2_1 (.in0(data [15:0]), .in1(bcd11out), .sel(
      hex_dec), .out(mux2_1_out) );

   wire [3:0] mux4_out;
   mux4 TDM_mux4 (.in0 mux2_1_out [3:0]) , .in1( mux2_1_out [7:4]), .in2 (
      mux2_1_out [11:8]) , .in3( mux2_1_out [15:12]), .sel(digit_sel) , .
         out(mux4_out) ) ;

   wire [6:0] sseg_decoder_out ;
   sseg_decoder TDM_decode (. num ( mux4_out ) , . sseg ( sseg_decoder_out
       ) ) ;

   wire [3:0] decoder_out ;
   an_decode an_decode_TDM (. in ( digit_sel ) , . out ( decoder_out ) ) ;

   wire mux22_in ;
   assign mux22_in = ~ decoder_out [3] & sign ;
   mux2 #(.N(7)) TDM_mux2_2 (.in0( sseg_decoder_out ) , .in1(7'b0111111 )
      , .sel (  mux22_in ) , . out ( seg ) ) ;

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

Listing 3: calc_lab10 Verilog Code

```
`timescale 1ns / 1ps
// Ashlie Lackey, ELC 2137, 2020 -04 -08
module calc_lab10(input btnU, btnD,
   input [15:0] sw,
   input clk, btnC,
   output [15:0] led,
   output dp ,
   output [3:0] an,
   output [6:0] seg);


   top_lab9 calc_unit(.btnU(btnU), .btnD(btnD),.sw(sw),.clk(clk), .btnC(
      btnC),.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),.dp(dp),.an(an));
endmodule
```

Listing 4: counter_test testbench Verilog Code

```
`timescale 1ns / 1ps
// Ashlie Lackey, ELC 2137, 2020-04-08
module counter_test();
   reg clk, en, rst;
   wire [1:0] count;
   wire tick;

   counter #(.N(2)) c(.clk(clk),.en(en), .rst(rst), .count(count), .tick(
      tick) );
   // clock runs continuously
   always begin
      clk = ~clk; #5;
   end
   // this block only runs once
   initial begin
      clk=0; en=0; rst =0; #7;
      rst = 1; #3; // reset
      en = 1; rst = 0; #5;
      en = 0; #5;
      en = 1; #5;
      en = 0; #5;
      en = 1; #5;
      en = 0; #5;
      en = 1; #5;
      en = 0; #5;
      en = 1; #5;
      en = 0; #5;
      en = 1; #5;
      en = 0; #5;
      en = 1; #5;
      en = 0; #5;
      en = 1; #5;
      en = 0; #5;
      en = 1; #5;
      en = 0; #5;
      en = 1; #5;
   $finish;
   end
endmodule
```

Listing 5: sseg4_TDM_test Code

```
`timescale 1ns / 1ps
// Ashlie Lackey, ELC 2137, 2020-04-08
module sseg4_TDM_test();
   reg [15:0] data;
   reg hex_dec, sign, reset, clock;
   wire [6:0] seg;
   wire dp;
   wire [3:0] an;

   sseg4_TDM TDM_test(.data(data),
   .hex_dec(hex_dec), .sign(sign), .reset(reset),
```

```
   .clock(clock),.seg(seg),.dp(dp),.an(an));

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


   initial begin
      data = 16'h1234; clock = 0; reset =0; #2;
      reset = 1; #3; //  reset
      reset = 0; hex_dec = 0; sign = 0; #1000000;
      hex_dec = 1; sign = 0; #1000000;
      hex_dec = 0; sign = 1; #1000000;
   $finish;
   end

endmodule
```