ELC 2137 Lab 9: ALU

CJ Jones

April 23, 2020

Summary

This lab explores the skill of creating a clock that drives the sequential logic to do the switching for us. The name given to this technique is time-division multiplexing (TDM). Overall, this lab demonstrated how to utilize software and programmable logic to produce a hardware output.

Q&A

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

   The three main groups of the RTL definition of sequential logic 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.

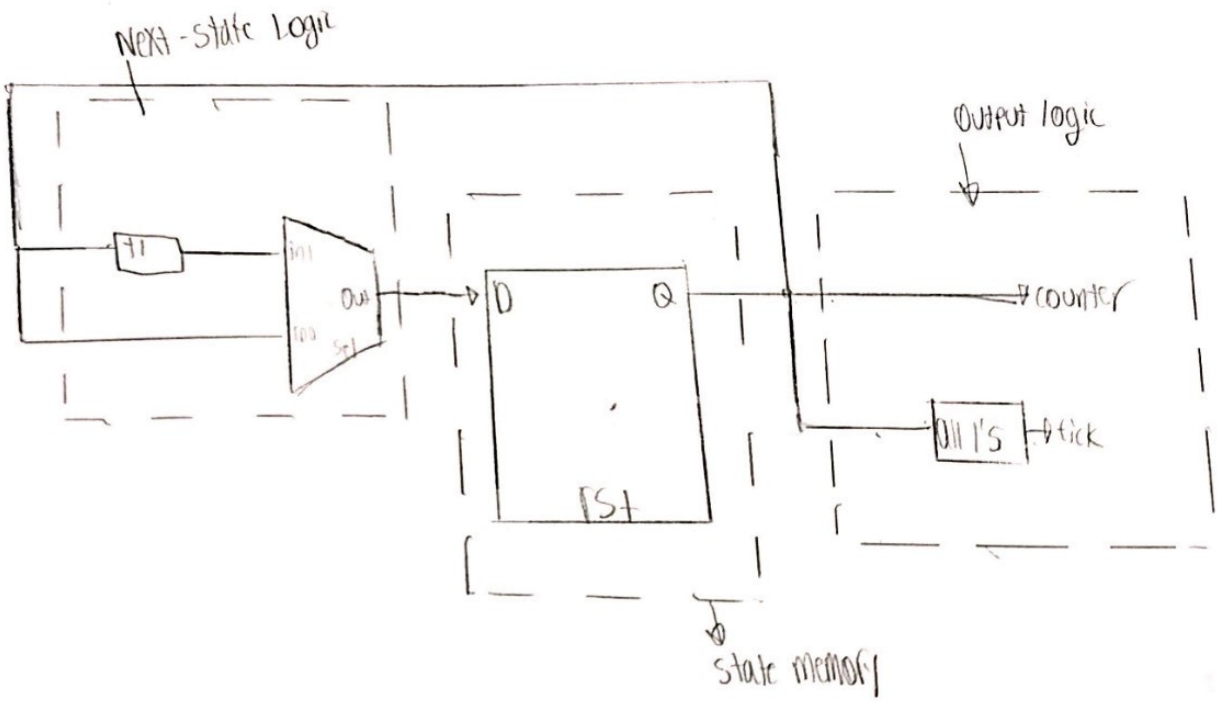

Figure 1: Figure 10.3 with boxes
=

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

Time (ns):	0-5	5-10	15-20	20-25	25-30	30-35	35-40	40-45	45-50	50-55
clk	0	1	0	1	0	1	0	1	0	1
en	1	1	1	1	1	1	1	1	1	1
rst	1	1	1	1	1	1	1	1	1	1
count	0	0	0	0	0	0	0	0	0	0
tick	0	0	0	0	0	0	0	0	0	0

Time (ns):	55-60	60-65	65-70	70-75	75-80	80-85	85-90	90-95	95-100	100-105
clk	0	1	0	1	0	1	0	1	0	1
en	0	0	0	0	0	0	0	0	0	0
rst	0	0	0	0	0	0	0	0	0	0
count	0	0	0	0	0	0	0	0	0	0
tick	0	0	0	0	0	0	0	0	0	0

Time (ns):	105-110	110-115	115-120	120-125	125-130	130-135	135-140	140-145	145-150
clk	1	0	1	0	1	0	0	1	0
en	1	1	1	1	1	1	1	1	1
rst	0	0	0	0	0	0	0	0	0
count	1	1	2	2	3	3	0	1	1
tick	0	0	0	1	1	0	0	0	0

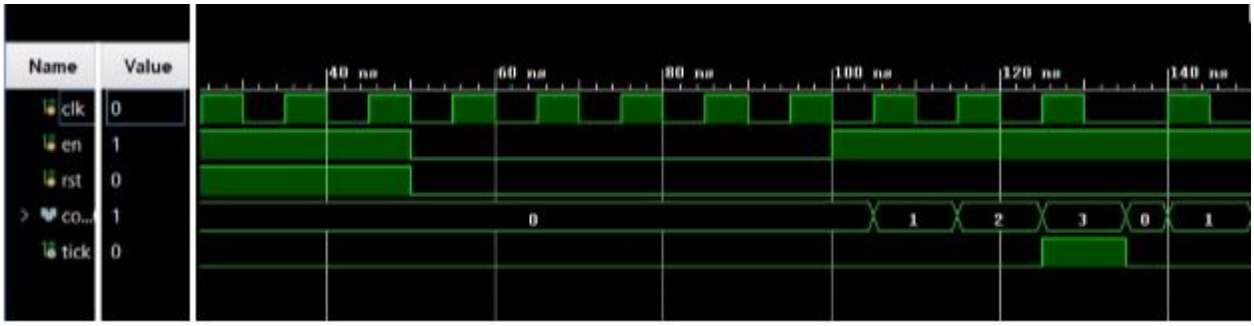

Figure 2: Counterexpected results table and simulation

=

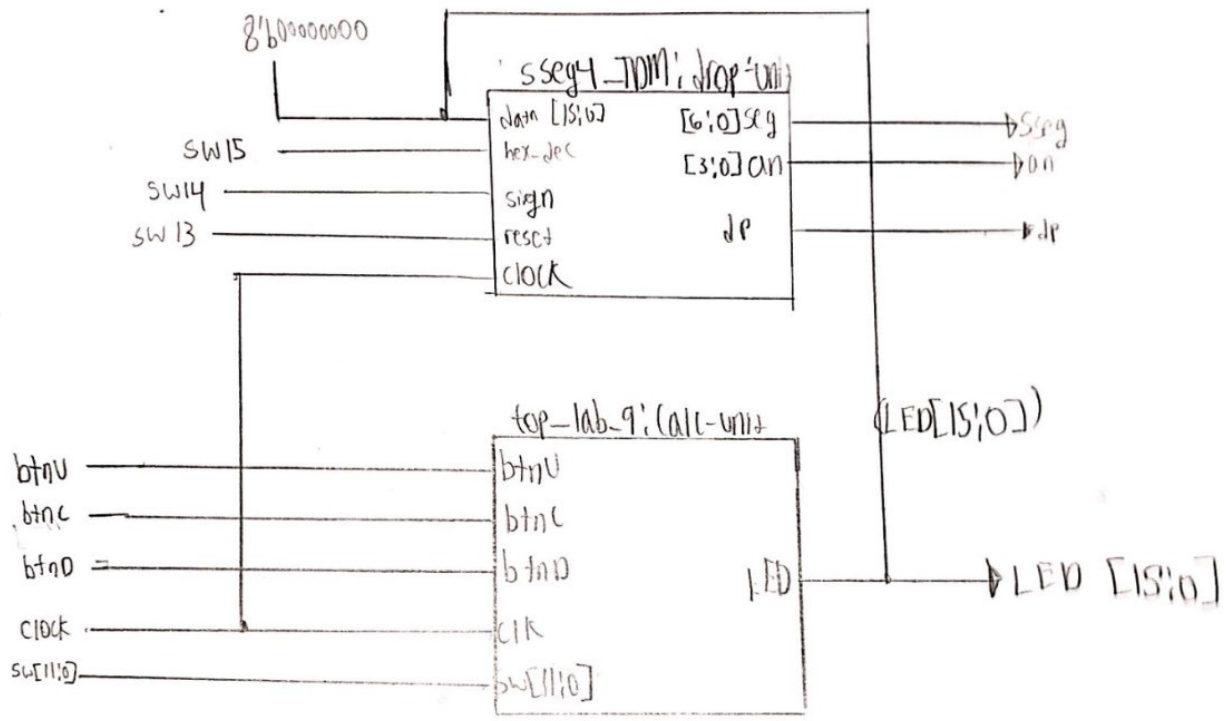

Figure 3: Figure 10.3 with boxes

=

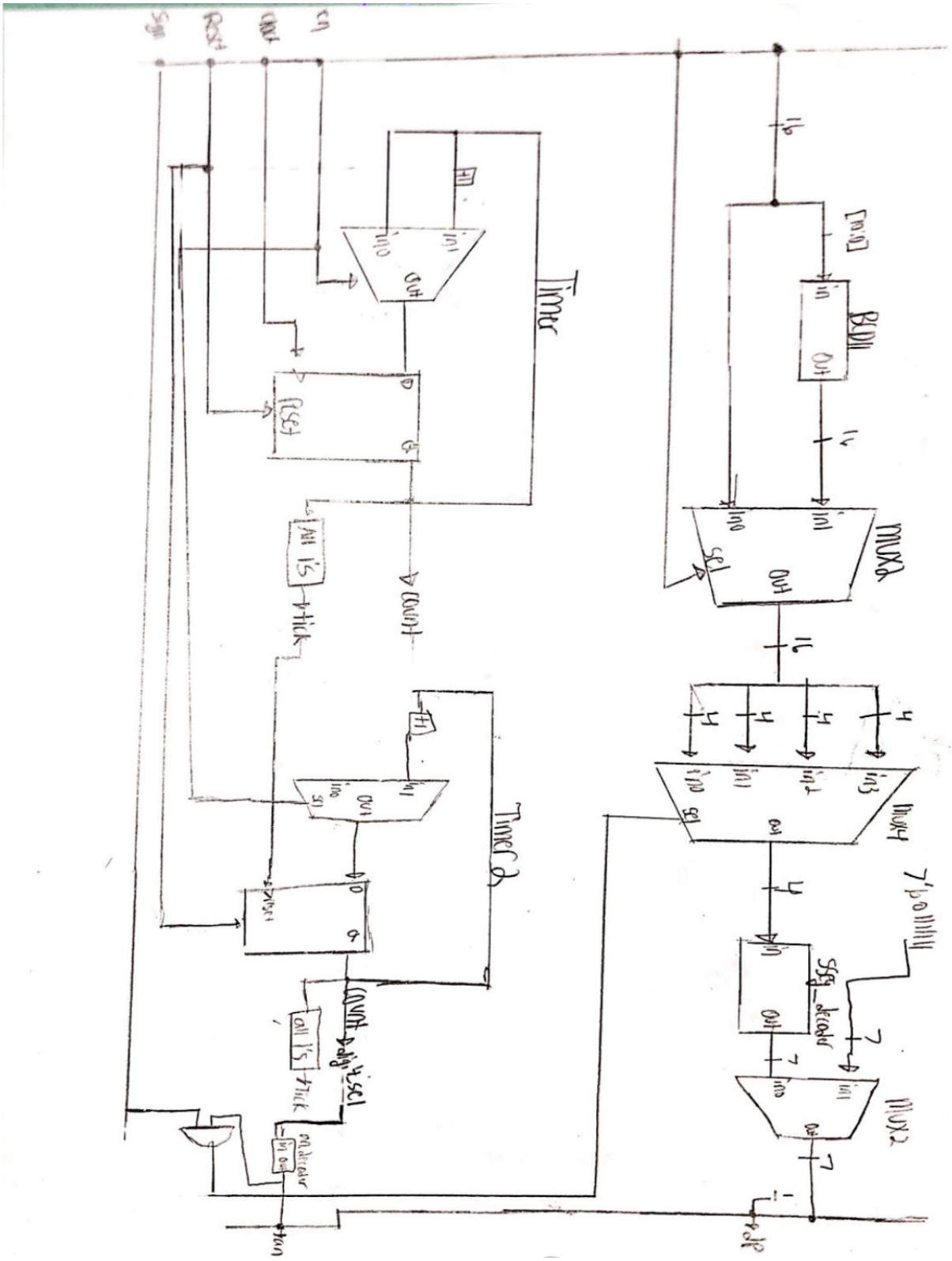

Figure 4: Figure 10.3 with boxes

=

Code

Listing 1: Counter Code

```
// Chris Jones , ELC 2137, 2020 -3-30-20

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 Bench

```
// Chris Jones , ELC 2137, 2020 -3-30-20

module counter_test();

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

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

always begin
clk = ~clk;
#5;
end
```

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

Listing 3: sseg4 TDM code

```
// Chris Jones , ELC 2137, 2020 -3-30-20

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

wire [15:0] ebout, mux2out;
wire [3:0] mux4out;
wire [6:0] decout;
wire andecout;
wire m2sel;
```

```
wire [1:0] digit_sel;
wire tickout;

counter #(.N(18)) c1(.clk(clock), .rst(reset), .en(1), .tick(tickout));
counter #(.N(18)) c2(.clk(clock), .on(tickout), .count(digit_sel), .rst(
   reset));
BCD11 e0(.in(data[10:0]), .out(ebout));
mux2 #(.N(16)) mux2A(.in0(data), .in1(ebout), .sel(hex_dec), .out(mux2out)
   );
mux4 #(.N(4)) mux4A(.in0(mux2out[3:0]), .in1(mux2out[7:4]), .in2(mux2out
   [11:8]), .in3(mux2out[15:12]), .sel(digit_sel), .out(mux4out));
sseg_decoder s1(.num(mux4out), .sseg(decout));
an_decoder an1(.in(digit_sel), .out(an));
assign m2sel = ~an[3];
and agate1(andecout, sign, m2sel);
mux2 #(.N(7)) mux2B(.in0(decout), .in1(7'b01111111), .sel(andecout), .out(
   seg));
assign dp = 1;
endmodule
```

Listing 4: sseg4 TDM test bench code

```
// Chris Jones , ELC 2137, 2020 -3-30-20

module  sseg_TDM_test();

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

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

always begin
clock = ~clock; #5;
end

initial begin
clock = 0;
data = 16'b0000000000000010;
hex_dec = 0;
sign = 0;
reset = 0;
#20;
data = 16'b0000000000000001;
hex_dec=1;
sign = 0;
reset = 0;
#40;
```

```
data = 16'b0000000000000010;
hex_dec=0;
sign= 0;
reset = 0;
#80;
data = 16'b0000000000000010;
hex_dec = 1;
sign = 0;
reset = 0;
#160
data = 16'b0000000000000011;
hex_dec = 0;
sign = 0;
reset = 1;
#160;
data = 16'b0000000000000011;
hex_dec= 1;
sign = 0;
reset = 0;
#320;
data = 16'b0000000000000100;
hex_dec = 0;
sign = 0;
reset = 0;
#640;
data = 16'b0000000000000100;
hex_dec = 1;
sign = 0;
reset = 0;
#1280;
data= 16'b0000000000000101;
hex_dec = 0;
sign = 0;
reset = 0;
#2560;
data = 16'b0000000000000101;
hex_dec = 1;
sign = 0;
reset = 0;
#5120;
data = 16'b0000000000000110;
hex_dec = 0;
sign = 0;
reset = 0;
#10240
data = 16'b0000000000000110;
hex_dec = 1;
sign = 0;
reset = 0;
#20480;
data = 16'b0000000000000111;
hex_dec = 0;
reset = 0;
#40960;
```

```
$finish;
end
endmodule
```