CMPE 100L: Lab6 Write Up

Yanliang Li

March 2, 2017

Contents

1 Description

The purpose of this lab was to create a turkey counter. What a turkey counter does is that it counts how many turkey passes by the two sensors. The turkey counter would be able to count turkey passing by from both direction. We assume that one turkey passing from left to right is adding one so passing from right to less is a minus one. The segment display shows the number of the turkey. Additionally, if a turkey is sticking in the middle, which means intersecting the sensors, the display will tell how long is the turkey sticking. The led light will indicate which sensor is being used. In this lab, we used two buttons' input to represent the turkey passing by as well as the sensors. We would be implementing many modules for this lab, including a turkey counter, a time counter, and a sensor inputs counter. Most importantly, we would also need to implement a state machine as well as a top level design by ourselves.

2 Methods

2.1 Turkey Counter

A turkey counter was to count how many turkey passing by. There is an additional input called "Up", that is for the counter to count up or down. Therefore, when Up goes high, the turkey counter will count up. When the Uo goes low, the turkey counter will count down.

2.1.1 Counter

The counter needed to be able to count up and count down because when a turkey passes from left sensor to right sensor the counter counts up one and when a turkey passes from right sensor to left sensor the counter counts down one. This counter is used to implement turkey counter.

2.2 Time Counter

A time counter was to count up to F. In this module, we used 4 bits out of 8 bits for counting the time.

2.3 Sensor Inputs Counter

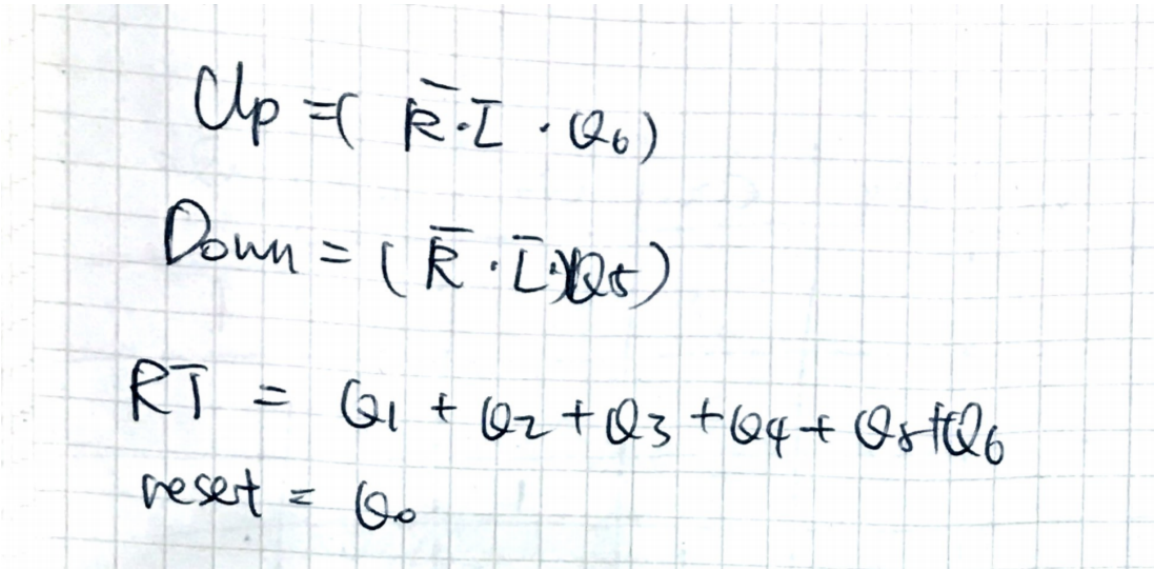
The sensor inputs counter was to indicate which button(s) is(are) being pressed, or which sensor(s) is(are) being used.

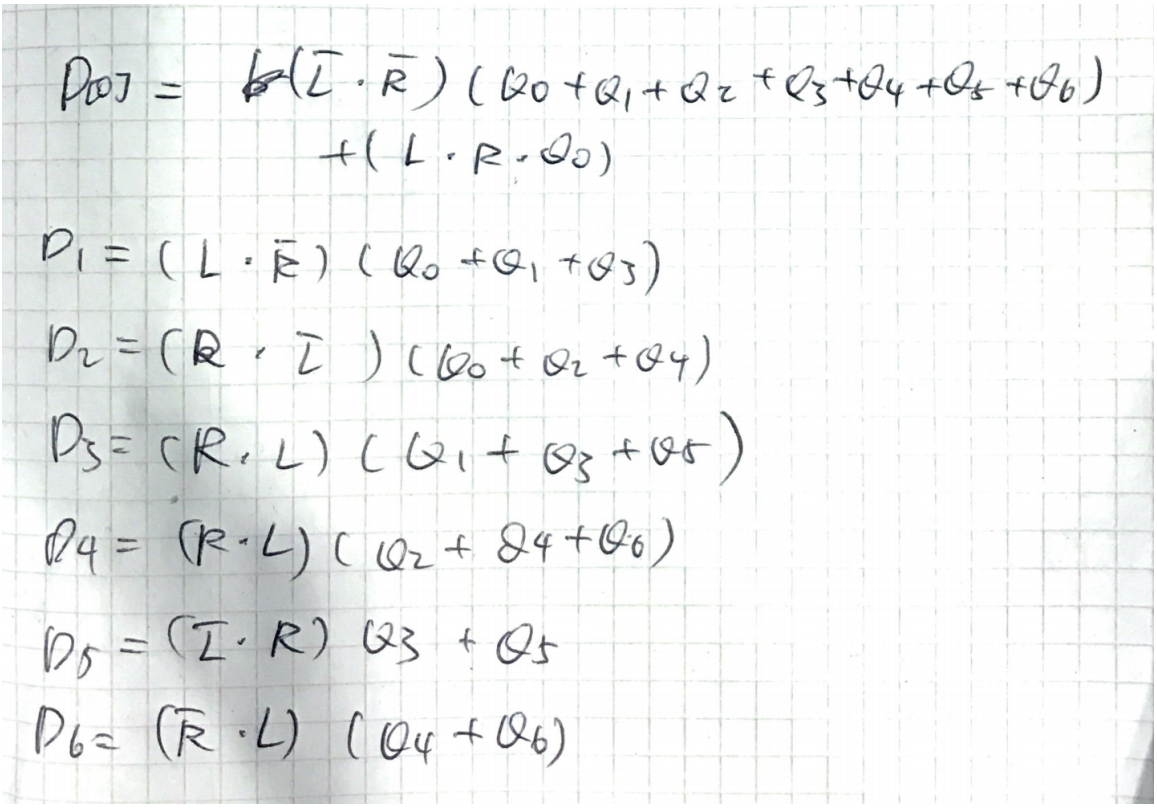
2.4 State Machine

I have designed a state machine for this lab. There are 7 states for the state machine.

3 Results

3.1 State Machine





State	Q6	Q5	Q4	Q3	Q2	Q1	Q0
IDLE	0	0	0	0	.	1	1
btnL 1	0	0	0	0		1	
btnR 1	0	0	0	0	1		
both L→R	0	0	0	1	1	.	.
both R→L	0	0	1		-	..	
btnR 2	0	1			0	e	
btnL 2	1						

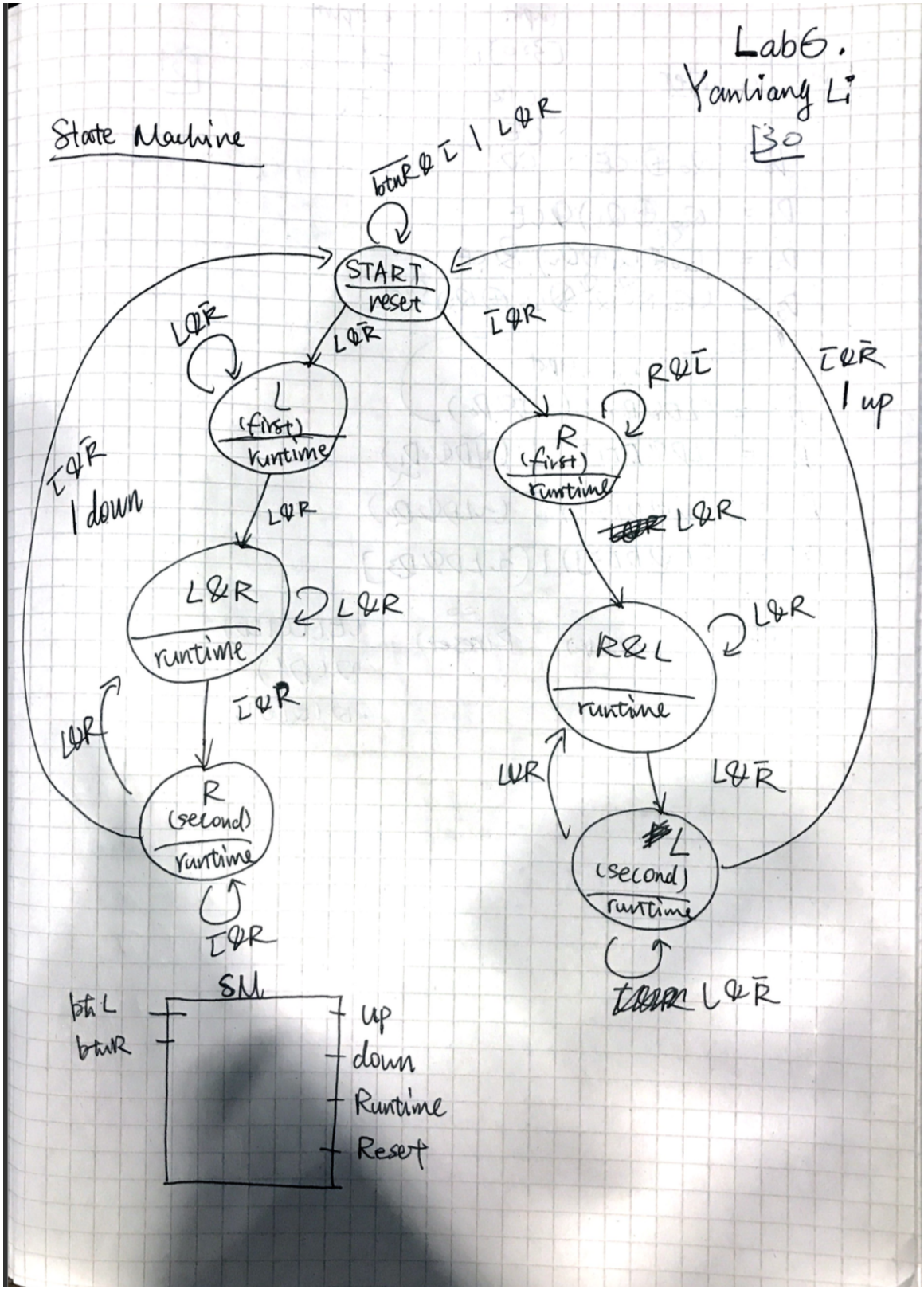
3.2 Top Module

The top level design is attached in the appendixes.

4 Conclusion

This lab was interesting to me. We had more space to design our own stuffs, like state machine and top level. I find myself can learn more from some previous related experience. For example, we could refer to lab 5 for help design the state machine and top level. The process was the same while the content was different. Also, the lab was more like the realistic problem that an engineer can be facing in the future. One of the most difficult tasks for me was to create the state machine. Once I figured that out, I was one huge step closer from finishing. Even though we have reference from the lab 5, this was my first time implement my own top level design. It was fun. One last thing was debugging, which is always tedious, and yet the most difficult part. Thank you for reading my report.

5 Notes


Lab6.
Yanliang Li
B30
State Machine
START
reset
L
(first)
runtime
R
(first)
runtime
L&R
runtime
R
(second)
runtime
R&L
runtime
L
(second)
runtime
SM
btnL
btnR
up
down
Runtime
Reset

time_counter

input
[3:0] D
CLK
CE
LD

output
[3:0] Q
TC

B1

$D_0 = Q_0 \oplus CE$

$D_1 = (Q_0 \oplus Q_1) \& CE$

$D_2 = (Q_0 \oplus Q_1 \oplus Q_2) \& CE$

$D_3 = (Q_0 \oplus Q_1 \oplus Q_2 \oplus Q_3) \& CE$

From input

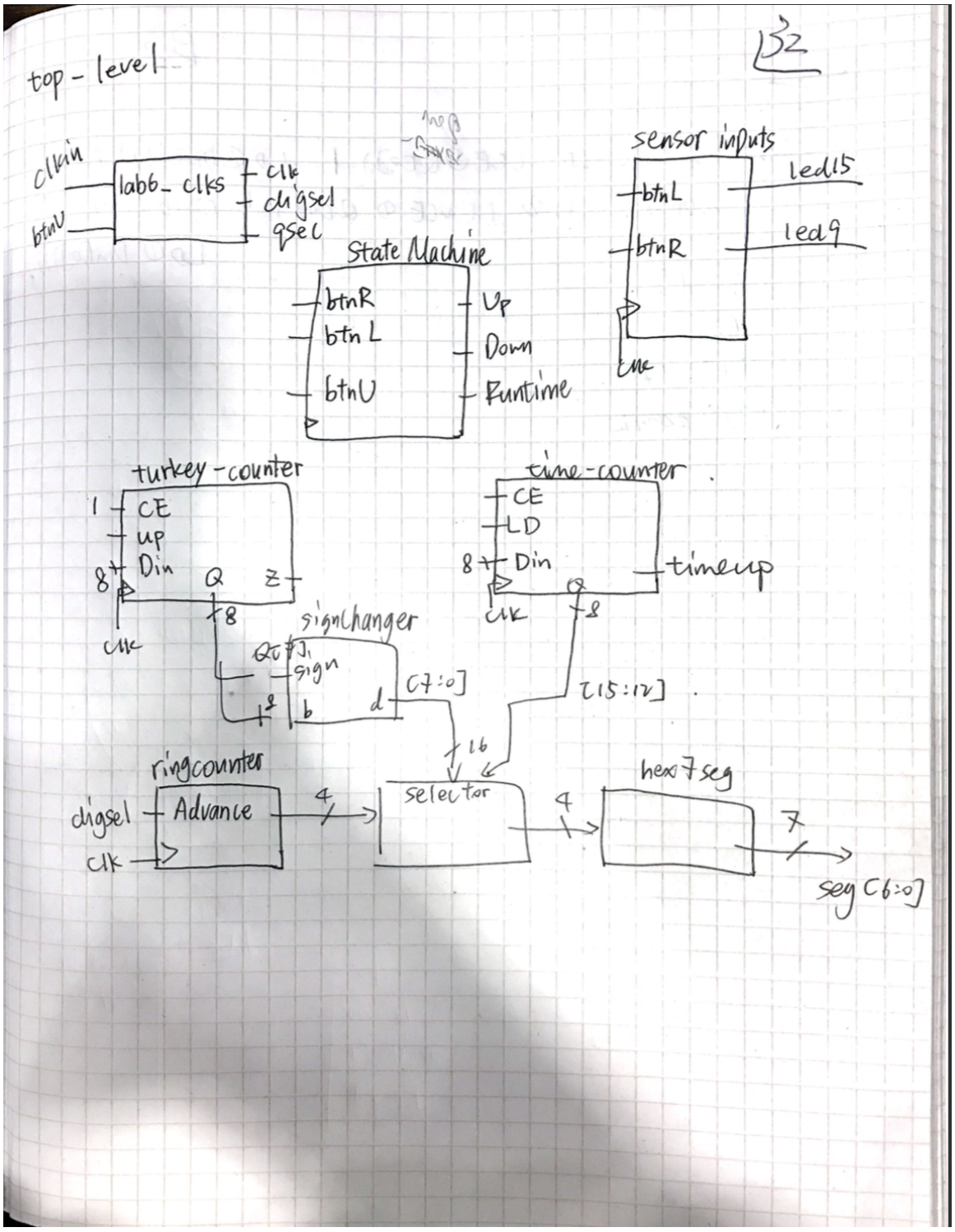
$D_0' = (LD \& D[0]) \mid (\sim LD \& D_0)$

$D_1' = (LD \& D[1]) \mid (\sim LD \& D_1)$

$D_2' = (LD \& D[2]) \mid (\sim LD \& D_2)$

$D_3' = (LD \& D[3]) \mid (\sim LD \& D_3)$

FF .C(CLK) .R(reset) = 0, .CE(CE|LD) .D(D') .Q(Q[0]).


top-level
clkin
btnU
labb_clks
clk
digsel
qsec
sensor inputs
btnL
btnR
led15
led9
clk
State Machine
btnR
btn L
btnU
Up
Down
Runtime
turkey-counter
1
CE
up
8
Din
Q
Z
clk
8
time-counter
CE
LD
8
Din
timeup
clk
8
signchanger
Q
sign
8
b
d
[7:0]
[15:12]
16
ringcounter
digsel
Advance
clk
4
selector
4
hex7seg
7
seg[6:0]

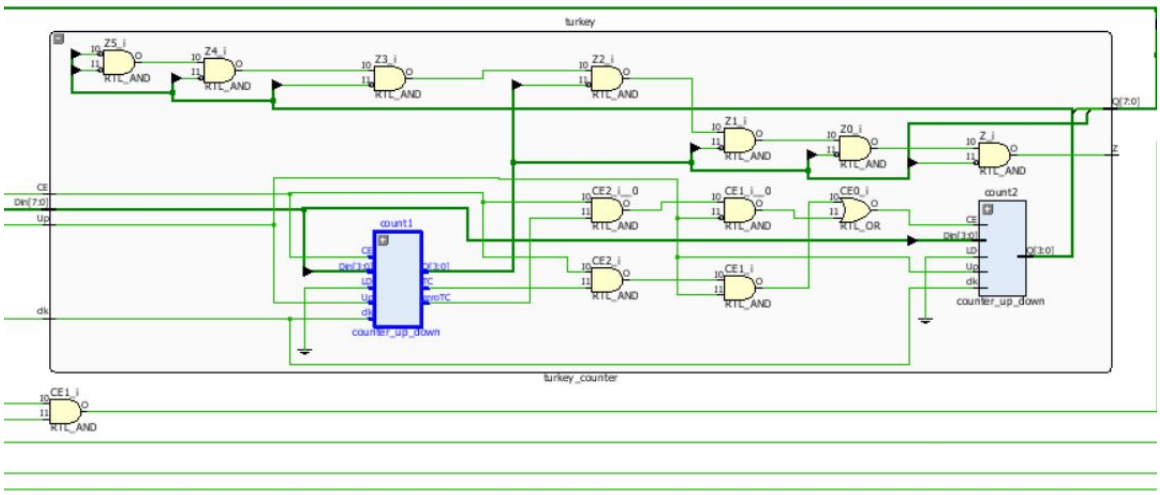
B3

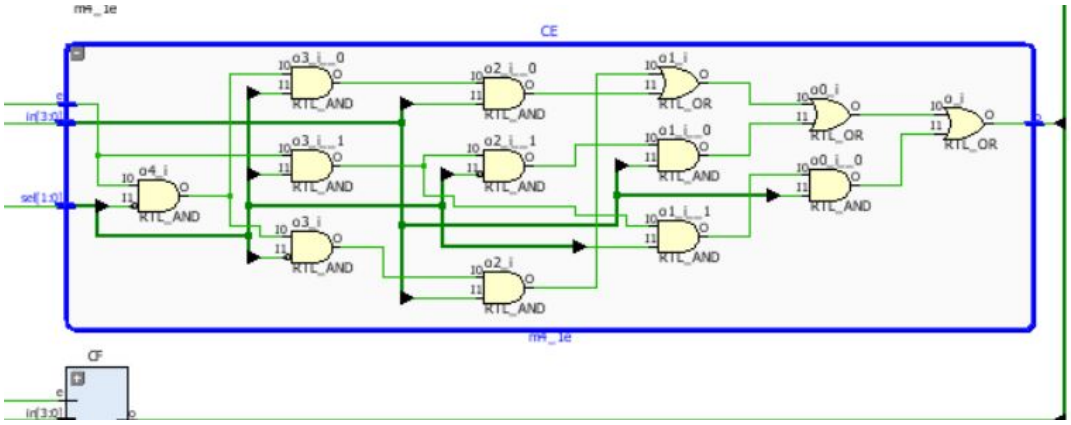
counter

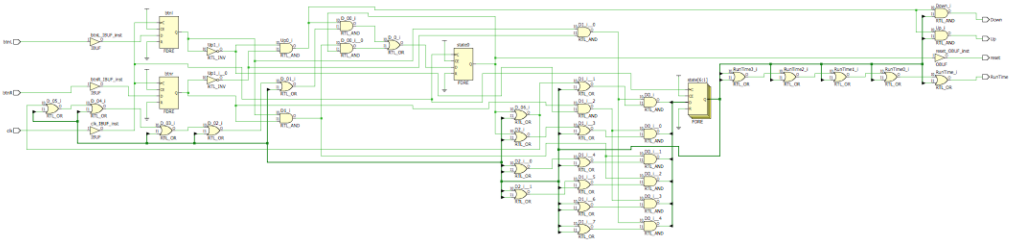
D[0] = Up & (~LD & (CE & Q[0]) | LD & Din[0]) | |
~Up & (~LD & ((~CE & Q[0]) | CE & ~Q[0] |
LD & Din[0]));

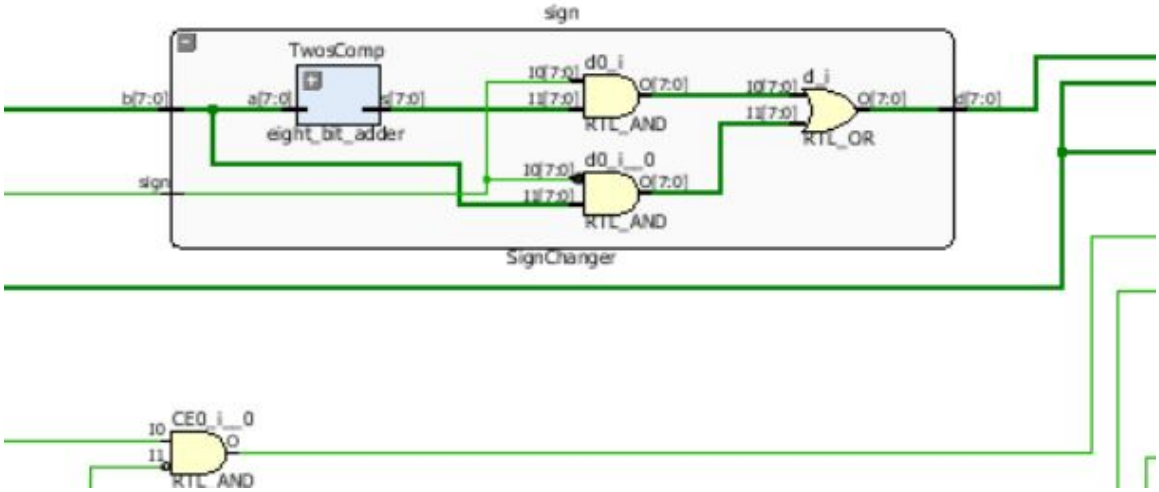
6345
10:20
2123117
Roger Kerr

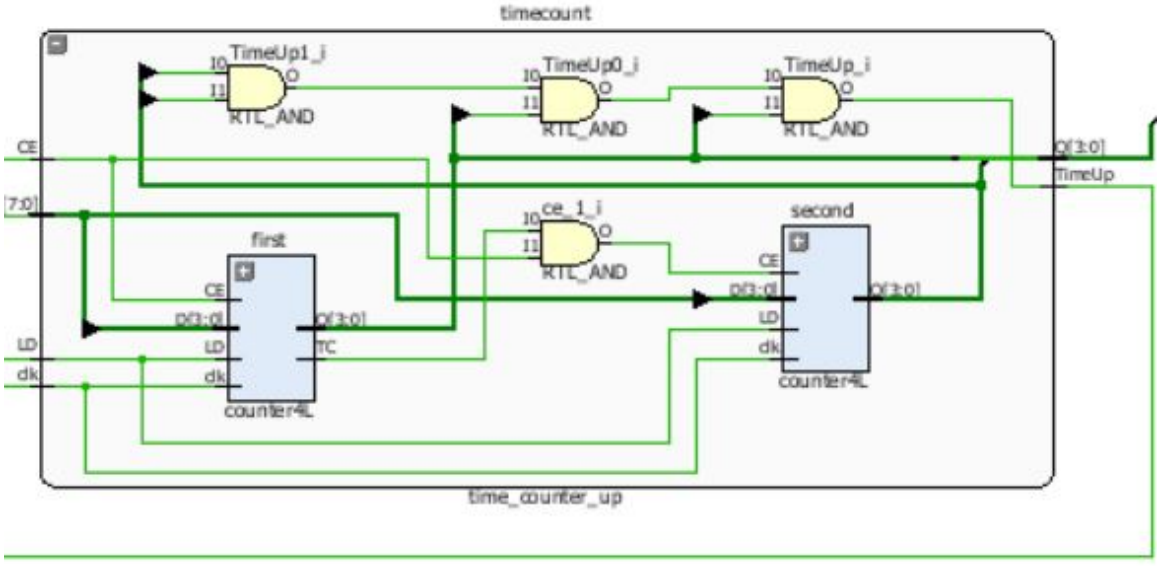
6 Schematic

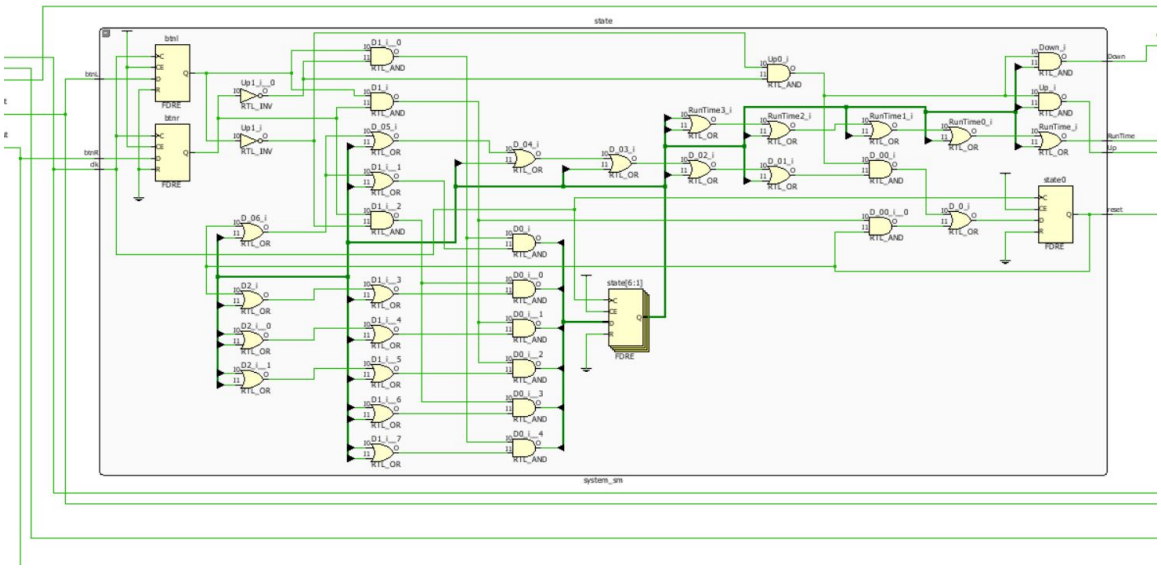

turkey
Z5_i
Z4_i
Z3_i
Z2_i
Z1_i
Z0_i
Z_i
RTL_AND
RTL_OR
CE2_i__0
CE1_i__0
CE0_i
CE2_i
CE1_i
count1
count2
counter_up_down
turkey_counter
CE
Dn[7:0]
Up
clk
Q[7:0]
Z

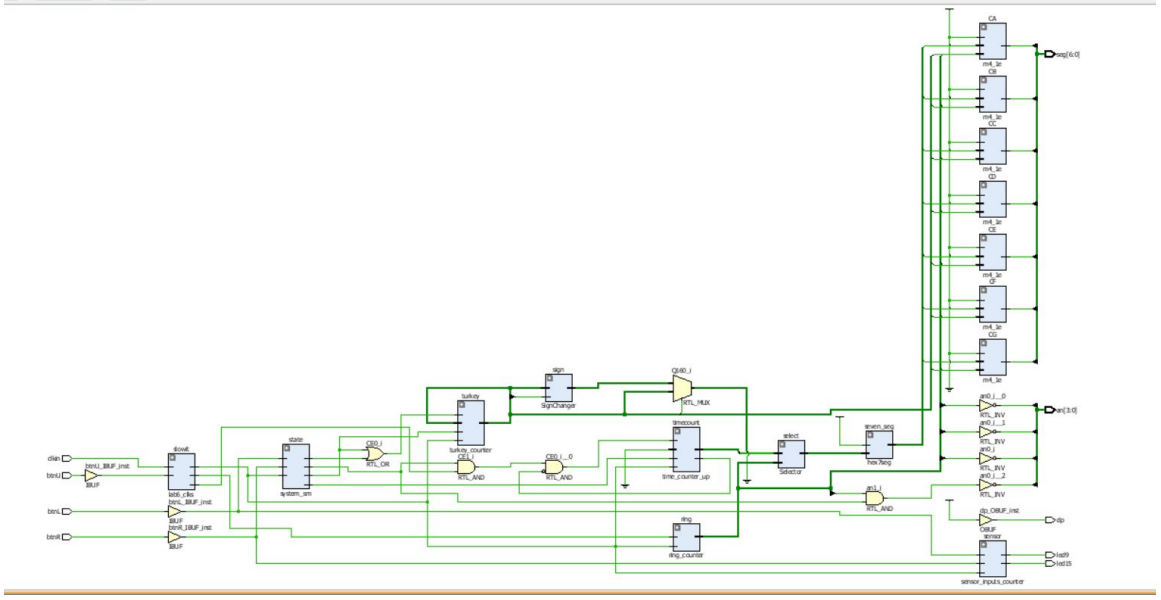

mux_1e
CE
o3_i__0
o3_i__1
o3_i
o4_i
o2_i__0
o2_i__1
o2_i
o1_i
o1_i__0
o1_i__1
o0_i
o0_i__0
RTL_AND
RTL_OR
in[3:0]
sel[1:0]
CF

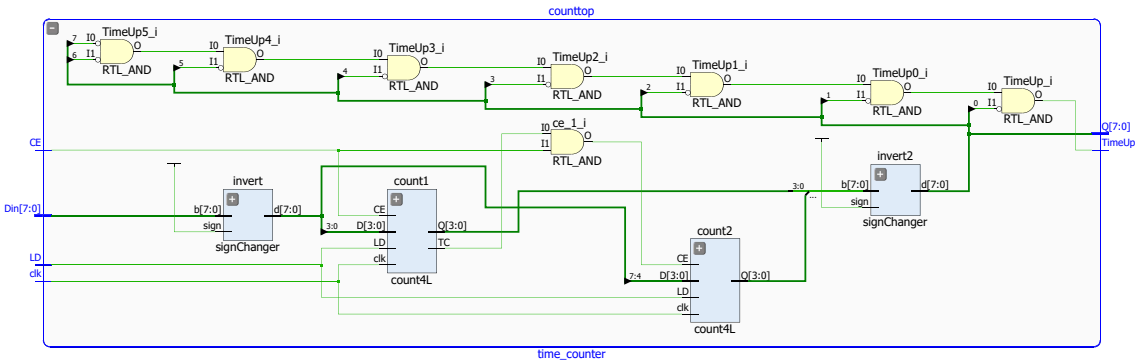



sign
TwosComp
b[7:0]
a[7:0]
s[7:0]
eight_bit_adder
sign
d0_i
I0[7:0]
I1[7:0]
O[7:0]
RTL_AND
d_i
RTL_OR
d0_i__0
SignChanger
CE0_i__0
I0
I1
O
RTL_AND


timecount
TimeUp1_i
TimeUp0_i
TimeUp_i
RTL_AND
CE
7:0]
first
D[3:0]
LD
dk
Q[3:0]
TC
counter4L
ce_1_i
second
TimeUp
time_counter_up


state
btnl
btnr
clk
FDRE
RTL_INV
RTL_AND
RTL_OR
D1_i
D0_i
D2_i
D_05_i
D_06_i
D_04_i
D_03_i
D_02_i
D_01_i
D_00_i
RunTime3_i
RunTime2_i
RunTime1_i
RunTime0_i
Up0_i
Down_i
Up_i
RunTime_i
state0
state[6:1]
reset
Down
RunTime
Up
system_sm


clkin
btnU
btnL
btnR
dpowit
state
system_sm
turkey
turkey_counter
sign
SignChange
time_counter_up
timecount
selector
RTL_MUX
RTL_AND
RTL_OR
RTL_INV
IBUF
OBUF
CA
CB
CC
CD
CE
CF
CG
sensor
sensor_inputs_counter


counttop
TimeUp5_i
TimeUp4_i
TimeUp3_i
TimeUp2_i
TimeUp1_i
TimeUp0_i
TimeUp_i
RTL_AND
ce_1_i
CE
Din[7:0]
LD
clk
invert
signChanger
count1
count4L
count2
invert2
b[7:0]
d[7:0]
sign
D[3:0]
Q[3:0]
TC
Q[7:0]
TimeUp
time_counter

7 Verilog

```
module ring(
    input Advance,
    input clk,
    output [3:0] Q
);
//////////////ring
FDRE #( .INIT(1'b1)) ring0 (.C(clk), .CE(Advance), .D(Q[3]), .Q(Q[0]));
FDRE #( .INIT(1'b0)) ring1 (.C(clk), .CE(Advance), .D(Q[0]), .Q(Q[1]));
FDRE #( .INIT(1'b0)) ring2 (.C(clk), .CE(Advance), .D(Q[1]), .Q(Q[2]));
FDRE #( .INIT(1'b0)) ring3 (.C(clk), .CE(Advance), .D(Q[2]), .Q(Q[3]));

endmodule
```

```
module sensorInputsCounter(
    input btnL,
 input btnR,
 input clk,
 output led15,
 output led9,
 output [1:0] Q
 );
 wire btnL1, btnR1, one;
 assign one = 1'b1;
 FDRE #( .INIT(1'b0)) inbtnL (.C(clk), .CE(one), .D(btnL), .Q(Q[0]));
 FDRE #( .INIT(1'b0)) inbtnR (.C(clk), .CE(one), .D(btnR), .Q(Q[1]));

 assign led15 = ~Q[0];
 assign led9 = ~Q[1];



endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 02/13/2017 05:05:35 PM
// Design Name:
// Module Name: signChanger
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module signChanger(
   input sign,
   input [7:0] b,
   output [7:0] d
   );
   wire [7:0] new;
   wire [7:0] sign0;
   assign sing0={8{sign}};

   eight_bit_adder TwosComp ( .s(new[7:0]), .a(b[7:0]) );
   assign d = (sign0 & new) | (~(sign0) & b);

endmodule
```

```
module state_machine(
  input clk,
  input btnL,
  input btnR,
  output Up,
  output Down,
  output RunTime,
  output reset
  );

  wire [6:0] D, Q;
  wire one;
  assign one = 1'b1;
  wire btnR1, btnL1;
  FDRE #( .INIT(1'b0)) btnr (.C(clk), .CE(one), .D(btnR), .Q(btnR1));
  FDRE #( .INIT(1'b0)) btnl (.C(clk), .CE(one), .D(btnL), .Q(btnL1));

  assign D[0] =  ((~btnL1 & ~btnR1) & (Q[0] | Q[1] | Q[2] | Q[3] | Q[4] | Q[5]| Q[6]) )| (btnL1 & btnR1 & Q[0]) ;
  assign D[1] = (btnL1 & ~btnR1) & (Q[0] | Q[1] | Q[3]);
  assign D[2] = (btnR1 & ~btnL1) & (Q[0] | Q[2] | Q[4]);
  assign D[3] = (btnR1 & btnL1) & (Q[1] | Q[3] | Q[5]);
  assign D[4] = (btnR1 & btnL1) & (Q[2] | Q[4] | Q[6]);
  assign D[5] = (~btnL1 & btnR1) & (Q[3] | Q[5]);
  assign D[6] = (~btnR1 & btnL1) & (Q[4] | Q[6]);

  assign Up = (~btnR1 & ~btnL1 & Q[6]);
  assign Down = (~btnR1 & ~btnL1 & Q[5]);
  assign RunTime = Q[1] | Q[2] | Q[3] | Q[4] | Q[5] | Q[6];
  assign reset = Q[0];

  FDRE #( .INIT(1'b1)) state0 (.C(clk), .CE(one), .D(D[0]), .Q(Q[0]));
  FDRE #( .INIT(1'b0)) statesixtoone (.C({6{clk}}), .CE({6{one}}), .D(D[6:1]), .Q(Q[6:1]));

endmodule
```

```
module top(input btnR,
    input btnL,
    input btnU,
    output led15,
    output led9,
    output [3:0] an,
    output dp,
    output [6:0] seg);
    wire one, zero;
    wire digsel, clk, qsec, turkey_z, TU;
    wire up, down, runtime, reset;
    wire [3:0] inSEL, outSEL;
    wire [7:0] turkey_out, in_turkey, time_out, twosComp;
    wire [15:0] out16;
    wire [7:0] time_in;
    wire [1:0] ledwire;
    wire [6:0] segwire;
    assign one = 1'b1;
    assign zero = 1'b0;
    assign in_turkey = 8'b00000000;
    lab6_clks slowit (.clkin(clkin), .greset(btnU), .clk(clk), .digsel(digsel), .qsec(qsec));
    state_machine state (.clk(clk), .btnR(btnR), .btnL(btnL), .Up(up), .Down(down), .RunTime(runtime), .reset(reset));
    assign one = 1'b1;
    sensorInputsCounter sensor (.clk(clk), .btnL(btnL), .btnR(btnR), .led15(led15), .led9(led9), .Q(ledwire[1:0]));
    turkey_counter turkey (.clk(clk), .CE(one & (up|down)), .Up(up), .Din(turkey_out), .Z(turkey_z), .Q(turkey_out[7:0]));
    sign signtop (.sign(turkey_out[7]), .b(turkey_out[7:0]), .d(twosComp[7:0]));
    assign out16[7:0] =  ({8{turkey_out[7]}} & twosComp[7:0]) | (~{8{turkey_out[7]}} & turkey_out[7:0]);
    time_counter timecounttop (.clk(clk), .CE(runtime & qsec & ~TU), .LD(reset), .Din(time_in[7:0]), .Q(out16[15:12]), .TimeUp(TU));
    ring ringtop (.clk(clk), .Advance(digsel), .Q(inSEL[3:0]));
    selector selecttop (.sel(inSEL[3:0]), .N(out16[15:0]), .H(outSEL[3:0]));
    hex7seg segtop (.n(outSEL[3:0]), .e(one), .seg(segwire[6:0]));
    m4_1e CG ( .in({zero, one, segwire[6], segwire[6]}), .sel({inSEL[2], turkey_out[7]}), .e(one), .o(seg[6]));
    m4_1e CF ( .in({one, one, segwire[5], segwire[5]}), .sel({inSEL[2], turkey_out[7]}), .e(one), .o(seg[5]));
    m4_1e CE ( .in({one, one, segwire[4], segwire[4]}), .sel({inSEL[2], turkey_out[7]}), .e(one), .o(seg[4]));
    m4_1e CD ( .in({one, one, segwire[3], segwire[3]}), .sel({inSEL[2], turkey_out[7]}), .e(one), .o(seg[3]));
    m4_1e CC ( .in({one, one, segwire[2], segwire[2]}), .sel({inSEL[2], turkey_out[7]}), .e(one), .o(seg[2]));
    m4_1e CB ( .in({one, one, segwire[1], segwire[1]}), .sel({inSEL[2], turkey_out[7]}), .e(one), .o(seg[1]));
    m4_1e CA ( .in({one, one, segwire[0], segwire[0]}), .sel({inSEL[2], turkey_out[7]}), .e(one), .o(seg[0]));
    assign dp = 1;assign an[3] = ~(inSEL[3] & (runtime));assign an[2] = ~(inSEL[2]);
    assign an[1] = ~(inSEL[1] & one);
    assign an[0] = ~(inSEL[0] & one);
endmodule
```

```
module turkey_counter(
     input [7:0] Din,
  input CE,
  input Up,
  input clk,
  output [7:0] Q,
  output Z
  );

  wire [1:0] ce, tc, zerotc;
  wire [7:0] loadQ;
  wire zero;
  assign zero = 1'b0;
  assign ce[0] = CE;

  counter count1 (.clk(clk), .Din(Din[3:0]), .CE(ce[0]), .LD(zero), .Up(Up), .Q(Q[3:0]), .TC(tc[0]), .zeroTC(zerotc[0]));
    assign ce[1] = (CE & tc[0] & Up) | (CE & zerotc[0] & ~Up);
  counter count2 (.clk(clk), .Din(Din[7:4]), .CE(ce[1] ), .LD(zero), .Up(Up), .Q(Q[7:4]), .TC(tc[1]), .zeroTC(zerotc[1]));

  assign Z = ~Q[7] & ~Q[6] & ~Q[5] & ~Q[4] & ~Q[3] & ~Q[2] & ~Q[1] & ~Q[0];


endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 02/13/2017 05:14:01 PM
// Design Name:
// Module Name: count4L
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////



module count4L(
    input clk,
   input CE,
   input LD,
   input [3:0] D,
   output [3:0] Q,
   output TC
   );

   wire [3:0] inD;

   assign inD[3] = ~LD & CE & (Q[3]^Q[2] & Q[1] & Q[0]) | LD & D[3];
   assign inD[2] = ~LD & CE & (Q[2]^Q[1] & Q[0]) | LD & D[2];
   assign inD[1] = ~LD & CE & (Q[1]^Q[0]) | LD & D[1];
   assign inD[0] = ~LD & (CE ^ Q[0]) | LD & D[0] ;

   assign TC = Q[3] & Q[2] & Q[1] & Q[0];

   FDRE #( .INIT(1'b0)) zero (.C(clk), .CE(LD|CE), .D(inD[0]), .Q(Q[0]));
   FDRE #( .INIT(1'b0)) one (.C(clk), .CE(LD|CE), .D(inD[1]), .Q(Q[1]));
   FDRE #( .INIT(1'b0)) two (.C(clk), .CE(LD|CE), .D(inD[2]), .Q(Q[2]));
   FDRE #( .INIT(1'b0)) three (.C(clk), .CE(LD|CE), .D(inD[3]), .Q(Q[3]));
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 02/13/2017 05:28:38 PM
// Design Name:
// Module Name: FullAdder
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module FullAdder(
  input cin,
  input a,b,
  output s,
  output cout
  );
  wire e;
  assign e = 1;
  wire zero = 0;

  wire [3:0] x;
  wire [1:0] y;
  wire [3:0] w;
  assign x = {cin,~cin,~cin,cin};
  assign y = {a, b};

  assign w = {e, cin, cin, zero};

  m4_1e sum (.o(s), .in(x[3:0]), .sel(y[1:0]), .e(e) );
  m4_1e carry (.o(cout), .in(w[3:0]), .sel(y[1:0]), .e(e));


endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 02/06/2017 04:42:50 PM
// Design Name:
// Module Name: hex7seg
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module hex7seg(
    input [3:0] n,
    input e,
    output [6:0] seg
    );

    wire t0, t1;
    assign t0 = 0;
    assign t1 = 1;

    wire [2:0] sel;
    assign sel = n[3:1];

    wire [7:0] CA, CB, CC, CD, CE, CF, CG;

    assign CA = {t0, n[0], n[0], t0, t0, ~n[0], t0, n[0]};
    assign CB = {t1, ~n[0], n[0], t0, ~n[0], n[0], t0, t0};
    assign CC = {t1, ~n[0], t0, t0, t0, t0, ~n[0], t0};
    assign CD = {n[0], t0, ~n[0], n[0], n[0], ~n[0], t0, n[0]};
    assign CE = {t0, t0, t0, n[0], n[0], t1, n[0], n[0]};
    assign CF = {t0, n[0], t0, t0, n[0], t0, t1, n[0]};
    assign CG = {t0, ~n[0], t0, t0, n[0], t0, t0, t1};

    m8_1e seg0 (.in(CA[7:0]), .sel(sel), .e(e), .o(seg[0]) );
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 02/13/2017 05:30:22 PM
// Design Name:
// Module Name: m4_1e
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module m4_1e(
  input [3:0] in,
  input [1:0] sel,
  input e,
  output o
  );

  assign o = (e & ~sel[1] & ~sel[0] & in[0]) | (e & ~sel[1] & sel[0] & in[1]) | (e &
sel[1] & ~sel[0] & in[2]) | (e & sel[1] & sel[0] & in[3]);

endmodule
```

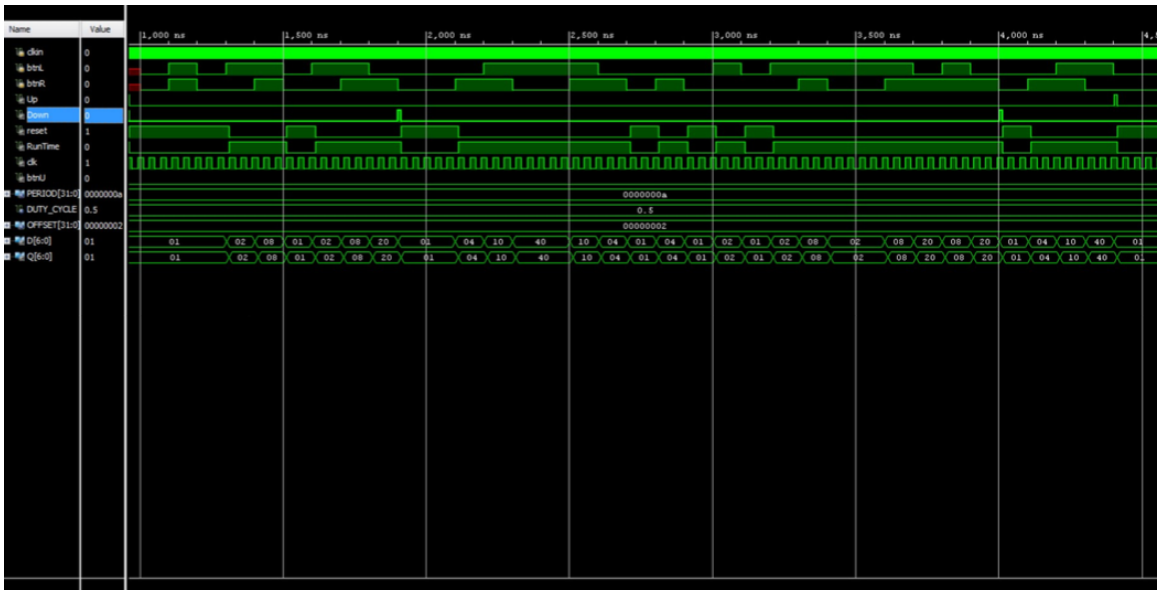
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 02/06/2017 07:43:49 PM
// Design Name:
// Module Name: m8_1e
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////



module m8_1e(
    input [7:0] in,
    input [2:0] sel,
    input e,
    output o
    );

    assign o = ( e & (~sel[2] & ~sel[1] & ~sel[0]) & in[0]) | ( e & (~sel[2] &
~sel[1] & sel[0]) & in[1]) | ( e & (~sel[2] & sel[1] & ~sel[0]) & in[2])
                | ( e & (~sel[2] & sel[1] & sel[0]) & in[3]) | ( e & (sel[2] &
~sel[1] & ~sel[0]) & in[4]) | ( e & (sel[2] & ~sel[1] & sel[0]) & in[5])
                | ( e & (sel[2] & sel[1] & ~sel[0]) & in[6]) | ( e & (sel[2] & sel[1]
& sel[0]) & in[7]);
endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 02/16/2017 10:22:01 AM
// Design Name:
// Module Name: selector
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module selector(
 input [3:0] sel,
   input [15:0] in,
   output [3:0] out
   );
   wire [3:0]s;

   assign s[0] = ~sel[3] & ~sel[2] & ~sel[1] & sel[0];
   assign s[1] = ~sel[3] & ~sel[2] & sel[1] & ~sel[0];
   assign s[2] = ~sel[3] & sel[2] & ~sel[1] & ~sel[0];
   assign s[3] = sel[3] & ~sel[2] & ~sel[1] & ~sel[0];



   assign out = ({4{s[0]}} & in[3:0]) | ({4{s[1]}} & in[7:4]) | ({4{s[2]}} &
in[11:8]) | ({4{s[3]}} & in[15:12]);

endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 02/13/2017 05:05:35 PM
// Design Name:
// Module Name: signChanger
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module signChanger(
   input sign,
   input [7:0] b,
   output [7:0] d
   );
   wire [7:0] new;
   wire [7:0] sign0;
   assign sing0={8{sign}};

   eight_bit_adder TwosComp ( .s(new[7:0]), .a(b[7:0]) );
   assign d = (sign0 & new) | (~(sign0) & b);

endmodule
```

```
module state_machine(
   input clk,
   input btnL,
   input btnR,
   output Up,
   output Down,
   output RunTime,
   output reset
   );

   wire [6:0] D, Q;
   wire one;
   assign one = 1'b1;
   wire btnR1, btnL1;
   FDRE #( .INIT(1'b0)) btnr (.C(clk), .CE(one), .D(btnR), .Q(btnR1));
   FDRE #( .INIT(1'b0)) btnl (.C(clk), .CE(one), .D(btnL), .Q(btnL1));

   assign D[0] =  ((~btnL1 & ~btnR1) & (Q[0] | Q[1] | Q[2] | Q[3] | Q[4] | Q[5]| Q[6]) )| (btnL1 & btnR1 & Q[0]) ;
   assign D[1] = (btnL1 & ~btnR1) & (Q[0] | Q[1] | Q[3]);
   assign D[2] = (btnR1 & ~btnL1) & (Q[0] | Q[2] | Q[4]);
   assign D[3] = (btnR1 & btnL1) & (Q[1] | Q[3] | Q[5]);
   assign D[4] = (btnR1 & btnL1) & (Q[2] | Q[4] | Q[6]);
   assign D[5] = (~btnL1 & btnR1) & (Q[3] | Q[5]);
   assign D[6] = (~btnR1 & btnL1) & (Q[4] | Q[6]);

   assign Up = (~btnR1 & ~btnL1 & Q[6]);
   assign Down = (~btnR1 & ~btnL1 & Q[5]);
   assign RunTime = Q[1] | Q[2] | Q[3] | Q[4] | Q[5] | Q[6];
   assign reset = Q[0];

   FDRE #( .INIT(1'b1)) state0 (.C(clk), .CE(one), .D(D[0]), .Q(Q[0]));
   FDRE #( .INIT(1'b0)) statesixtoone (.C({6{clk}}), .CE({6{one}}), .D(D[6:1]), .Q(Q[6:1]));

endmodule
```

8 Simulation



References

[1] M. Schlag. CMPE 100 Winter 2017.