Aidan Smith

11/16/19

Section 01D

# Lab 5 Report

## Description:

For this lab we had to create a game! It is a take on a chicken game, where two people compete to see who does something first. For this game, two players must compete to hit their button at the last possible second before some a random timer counts down to zero. The hard part is that you cannot see the timer! To make this we had to create our first state machine. The stages of the game can be modeled into different states on our state machine. Such as win someone scores, there is a state where the scores are shown, and whoever scored has their score flash. The state machine is very important and is the main part of the lab.

## Methods:

As previously mentioned, the state machine was the center of the lab. This represented the different states in the game. We create other modules around our state machine to work with the top module and complete the game. We had to create a Linear Feedback Shift Register that creates a random number as our countdown number. We had to reuse a lot of old modules from labs like: N-bit counters, ring counter, selector, hex7 display, etc. Most of the lab was reused from previous ones with some slight changes to work for this lab.

When I made the lab, I started by completing everything before the state machine. This included making the time counter, score counter, LSFR, ring counter, selector, hex display, etc. Most of it was a copy from last lab with some tweaks so it was pretty easy. I then created my state machine and refined it until I was ready to test it. I put it all together on my top module and with some fixes it worked!

Results:

Breakdown of sections

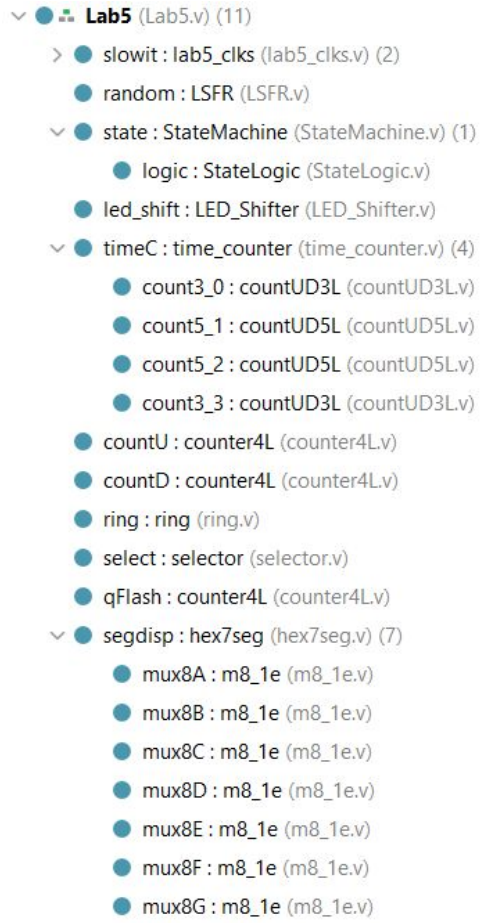
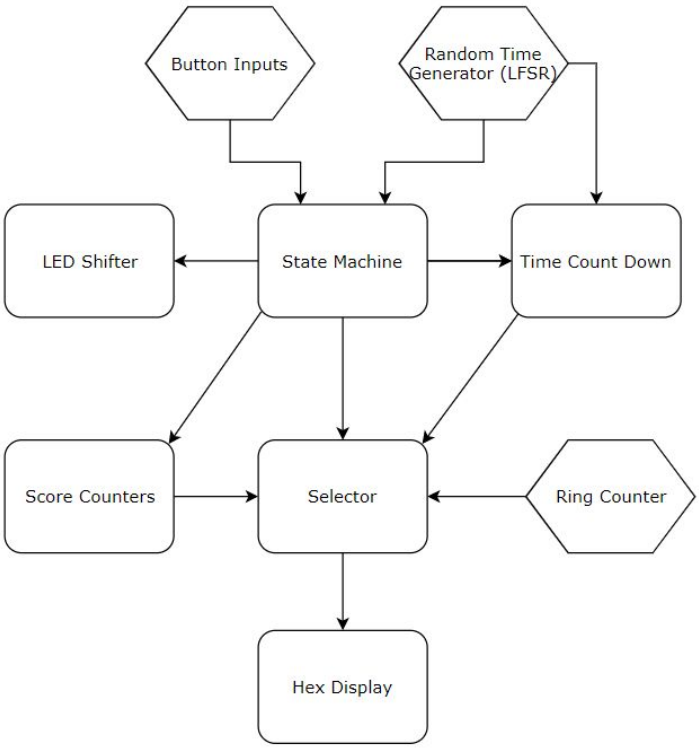



Diagram of top level



- How did you test your design to make sure it worked?

I tested it slightly with my test bench. Once I felt comfortable with how it looked, I moved on to testing it on the board, as testing it on the simulation is a bit more confusing, and I didn't run into many issues so I felt ready to test it on the board.

- Why did you choose those inputs?

I chose the inputs that were specified to us. I don't think there was any other inputs to pick without not meeting the specifications of the lab.

- Were there corner cases you needed to consider?

The weird cases that I had to make sure that were covered were around pressing the buttons U and D at the same time and around the time being up. If they are both pressed before time is up then they both get a point. But if it is before or after time is up then neither of them get a point.

- Discuss problems that you discovered during testing

I had a problem where my counter would keep going even after a player scored. I also had a problem with my states, where if both players scored at the same time, they would both get 3 points. This is due to looser restrictions on which state could be entered next. Once fixing this it worked properly.

- Extra Lab Questions:

Describe each state of your machine in words.

**Before:** This is the machine's initial state. In this state the score is shown. The machine stays in this state until button C is pressed.

**Ready:** This is the machines next state. The machine enters this state after button C is pressed from the Before state. When the machine first enters this state the time is loaded and is shown. The machine stays in this state until the last LED is lit up.

**Go:** This is the machines next state. The machine runs the time, starting in this state. The machine stays in this state until button U or button D are pressed, or until the time is up. If time is up then it goes back to the Before state.

**Ufirst:** This state is entered if button U is pressed before time is up or button D is pressed. This means player U has pressed their button first. The machine exits this state if time is up or if button D is pressed before time is up.

Dfirst: This state is entered if button D is pressed before time is up or button U is pressed. This means player D has pressed their button first. The machine exits this state if time is up or if button U is pressed before time is up.

**Both:** This state is entered if both button D and button U are pressed at the same time, and before time is up. The scores will be shown and both scores will flash as they both get a point. When button C is pressed it returns to the Ready state.

**Uwin:** This state is entered if either time is up before player D presses their button, or if player U presses their button after player D but before time is up. The score is shown and player U gains a point and therefore has their score flash. Once button C is pressed it goes back to the Ready state.

**Dwin:** This state is entered if either time is up before player U presses their button, or if player D presses their button after player U but before time is up. The score is shown and player

D gains a point and therefore has their score flash. Once button C is pressed it goes back to the Ready state.

**OTHER PARTS OF STATE DIAGRAM, NEXT STATE, AND OUTPUTS ARE AFTER THIS PAGE (PICTURES OF MY NOTES)**

Conclusion:

- What did you learn from the lab?

  I learned how to make a state machine! This was super tricky but once I learned how to do it I got it built and made it work properly.

- Did you have any difficulties?

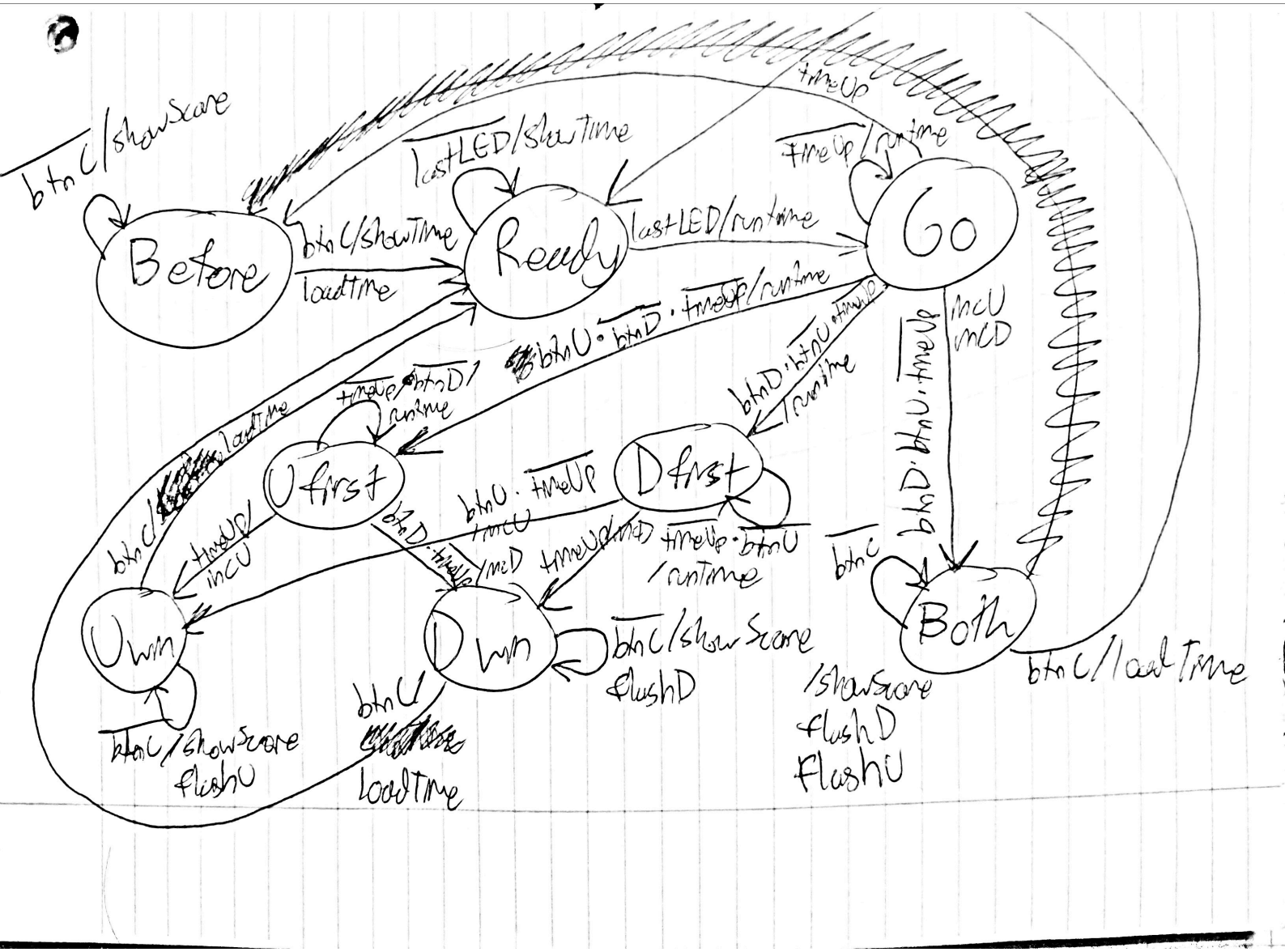
  The hardest part was making the state machine. I wasn't sure what type of state machine I wanted (Mealy or Moore), and I wasn't sure how many states I wanted and how I was to make it.

- If you could do this lab again, what would you do differently?

  I would make the state machine sooner to be able to test it.

- Are there components you would optimize?

  My state machine seems fully optimized. I would however clean up my functions, removing unneeded parts and combine modules that are almost the same. For example: I had a 3-bit, 4-bit, 5-bit, and 16-bit counter. I would've liked to try to minimize the logic as I only needed 4-bit counters and the 16-bit counter.


Before
Ready
Go
Ufirst
Dfirst
Uwin
Dwin
Both
btnC/showScore
btnC/showTime
loadTime
lastLED/showTime
lastLED/runTime
timeUp/runTime
btnU · btnD · timeUp/runTime
btnD · btnU · timeUp /runTime
timeUp/btnD/runTime
btnC/loadTime
timeUp/incU
btnD · timeUp
btnU · timeUp incU
incD
timeUp · btnU /runTime
timeUp · btnD
incU incD
btnC
btnC/showScore flushU
btnC/showScore flushD
btnC/loadTime
/showScore flushD flushU

Next State

$Before = Before \cdot \overline{btnC} + Go \cdot timeUp$

$Ready = btnC \cdot (Before + Uwin + Dwin + both) + Ready \cdot \overline{lastLED}$

$Go = Ready \cdot lastLED + Go \cdot \overline{timeUp} \cdot \overline{btnU} \cdot \overline{btnD}$

$Dfirst = Go \cdot btnD \cdot \overline{btnU} \cdot \overline{timeUp} + Dfirst \cdot \overline{btnU} \cdot \overline{timeUp}$

$Ufirst = Go \cdot \overline{btnD} \cdot btnU \cdot timeUp + Ufirst \cdot \overline{btnD} \cdot \overline{timeUp}$

$Both = Go \cdot btnD \cdot btnU \cdot timeUp + Both \cdot \overline{btnC}$

$Dwin = Dfirst \cdot timeUp + Ufirst \cdot btnD \cdot \overline{timeUp} + Dwin \cdot \overline{btnC}$

$Uwin = Ufirst \cdot timeUp + Dfirst \cdot btnU \cdot \overline{timeUp} + Uwin \cdot \overline{btnC}$

Outputs

$showTime = Ready$

$loadTime = btnC \cdot (Before + both + Uwin + Dwin)$

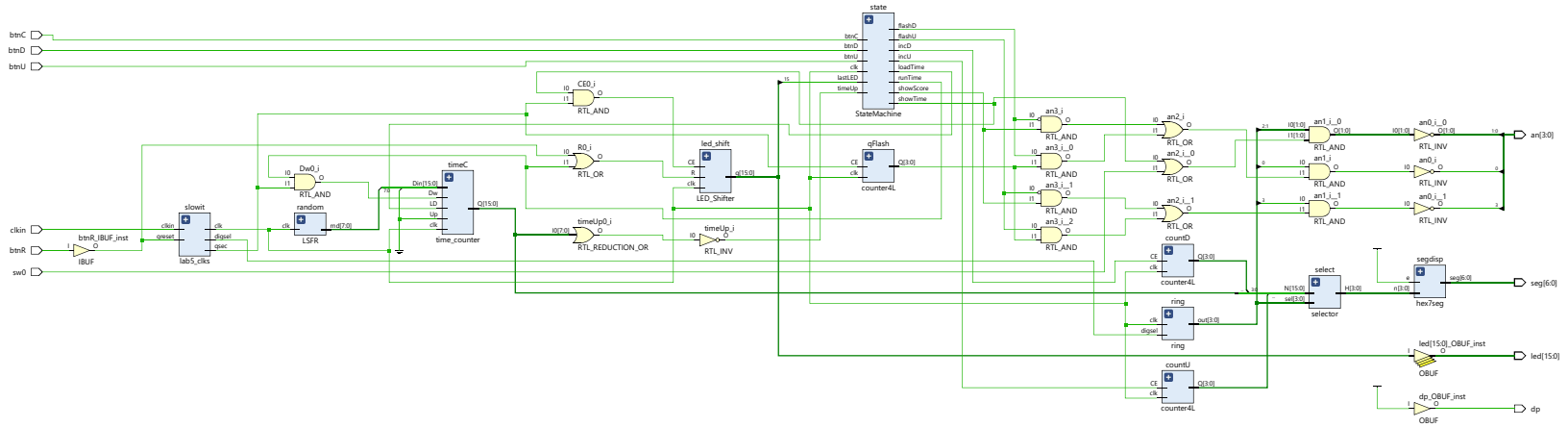
$runTime = Go + Dfirst + Ufirst$

$incU = Ufirst \cdot timeUp + Dfirst \cdot btnU \cdot \overline{timeUp} + Go \cdot btnU \cdot \overline{btnD} \cdot \overline{timeUp}$

$incD = Dfirst \cdot timeUp + Ufirst \cdot btnD \cdot \overline{timeUp} + Go \cdot btnU \cdot btnD \cdot \overline{timeUp}$

$showScore = Before + Both + Uwin + Dwin$

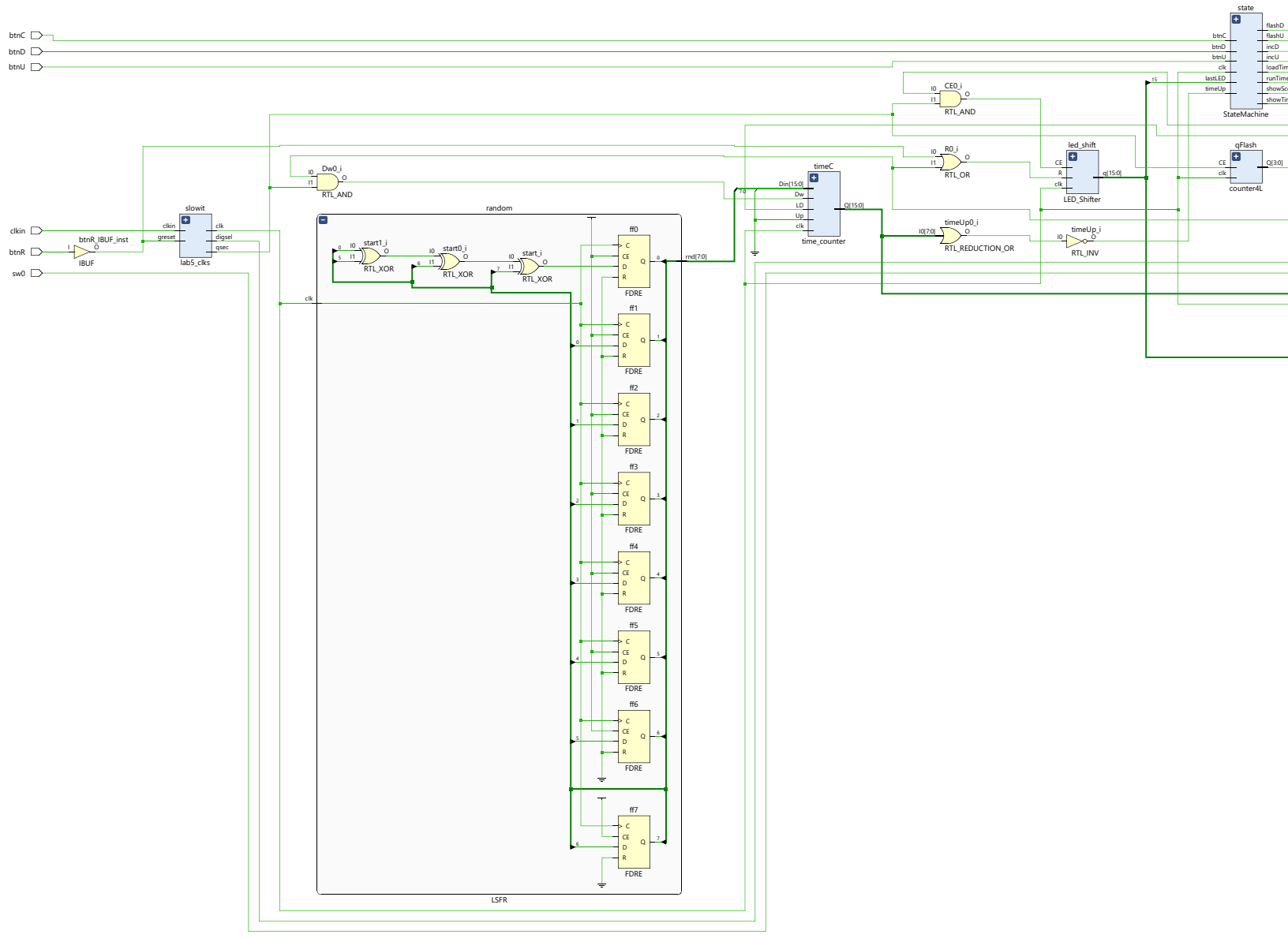
$flashU = both + Uwin \qquad flashD = both + Dwin$


btnC
btnD
btnU
clkin
btnR
sw0
btnR_IBUF_inst
IBUF
slowit
clkin
qreset
clk
digsel
qsec
labS_clks
Dw0_i
RTL_AND
random
clk
LSFR
rnd[7:0]
timeC
Din[15:0]
Dw
LD
Up
clk
Q[15:0]
time_counter
CE0_i
RTL_AND
R0_i
RTL_OR
timeUp0_i
I0[7:0]
RTL_REDUCTION_OR
led_shift
CE
R
clk
q[15:0]
LED_Shifter
timeUp_i
RTL_INV
state
btnC
btnD
btnU
clk
lastLED
timeUp
flashD
flashU
incD
incU
loadTime
runTime
showScore
showTime
StateMachine
qFlash
CE
clk
Q[3:0]
counter4L
an2_i
an3_i__0
an3_i__1
an3_i__2
an2_i__0
an2_i__1
an1_i__0
an1_i__1
an1_i
an0_i__0
an0_i__1
an0_i
countD
countU
ring
out[3:0]
select
N[15:0]
sel[1:0]
H[3:0]
selector
segdisp
seg[6:0]
hex7seg
led[15:0]_OBUF_inst
OBUF
dp_OBUF_inst
OBUF
an[3:0]
seg[6:0]
led[15:0]
dp

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 11/05/2019 10:55:21 AM
// Design Name:
// Module Name: Lab5
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Lab5(
    input clkin,
    input btnR,
    input btnU,
    input btnC,
    input btnD,
    input sw0,

    output [15:0] led,
    output [3:0] an,
    output dp,
    output [6:0] seg
    );
```

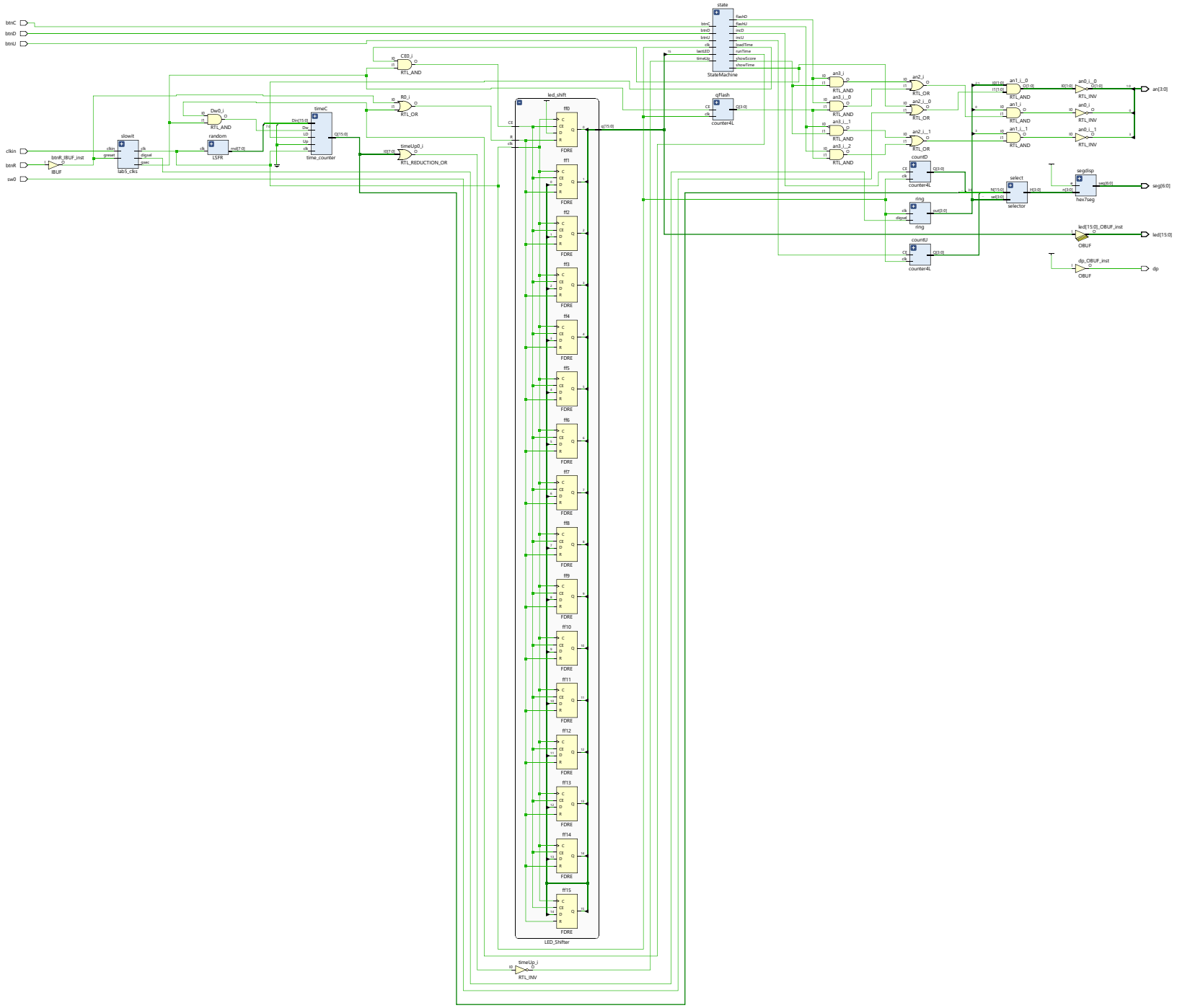
```
    wire [15:0] disp; // hex 7 disp values

    wire clk, digsel, qsec; // timers

    lab5_clks slowit (.clkin(clkin), .greset(btnR), .clk(clk),
.digsel(digsel), .qsec(qsec));

    wire timeUp; // LSFR variables
    wire [7:0] rnd;

    LSFR random( .clk(clk), .rnd(rnd) );

    wire lastLED; // LED_shifter value

    wire showTime, loadTime, runTime, incU, incD, showScore,
flashU, flashD; // state machine outputs

    StateMachine state( .clk(clk), .btnC(btnC), .btnU(btnU),
.btnD(btnD), .timeUp(timeUp), .lastLED(lastLED),
.showTime(showTime), .loadTime(loadTime), .runTime(runTime),
.incU(incU), .incD(incD), .showScore(showScore), .flashU(flashU),
.flashD(flashD) );

    LED_Shifter led_shift( .CE(showTime & qsec), .clk(clk), .R(btnR
| runTime), .q(led) );

    assign lastLED = led[15]; // if last LED shown

    wire [7:0] temp;

    time_counter timeC( .clk(clk), .Up(1'b0), .Dw(runTime & qsec),
.LD(loadTime), .Din(rnd), .Q( {temp, disp[11:4]}) );

    assign timeUp = ~(|disp[11:4]); // if time = 0000 0000, then
timeUp = 1
```

```
    counter4L countU( .clk(clk), .CE(incU), .Q(disp[15:12]) );
    counter4L countD( .clk(clk), .CE(incD), .Q(disp[3:0]) );

    wire [3:0] sel, n; // selector bits and n (the 4-bit bus that
is selected)

    ring ring( .digsel(digsel), .clk(clk), .out(sel) );

    selector select( .sel(sel), .N(disp), .H(n) );

    wire [3:0] qF;

    counter4L qFlash( .clk(clk), .CE(qsec), .Q(qF) );

    assign an[0] = ~( sel[0] & ( (~flashD & showScore) | (flashD &
qF[1]) ) ); // player D score
    assign an[3] = ~( sel[3] & ( (~flashU & showScore) | (flashU &
qF[1]) ) ); // player U score
    assign an[2:1] = ~(sel[2:1] & {2{ (showTime | sw0) }} ); //
timer
    assign dp = 1'b1; // turn off decimal point

    hex7seg segdisp( .n(n), .e(1'b1), .seg(seg) );

endmodule
```


btnC
btnD
btnU
clkin
btnR
sw0
IBUF
btnR_IBUF_inst
slowit
clkin
greset
clk
digsel
qsec
lab5_clks
Dw0_i
RTL_AND
random
start1_i
start0_i
start_i
RTL_XOR
clk
ff0
ff1
ff2
ff3
ff4
ff5
ff6
ff7
FDRE
rnd[7:0]
LSFR
timeC
Din[15:0]
Dw
LD
Up
clk
Q[15:0]
time_counter
CE0_i
RTL_AND
R0_i
RTL_OR
timeUp0_i
I0[7:0]
RTL_REDUCTION_OR
timeUp_i
RTL_INV
led_shift
CE
R
clk
q[15:0]
LED_Shifter
state
btnC
btnD
btnU
clk
lastLED
timeUp
flashD
flashU
incD
incU
loadTim
runTime
showSc
showTir
StateMachine
qFlash
CE
clk
Q[3:0]
counter4L

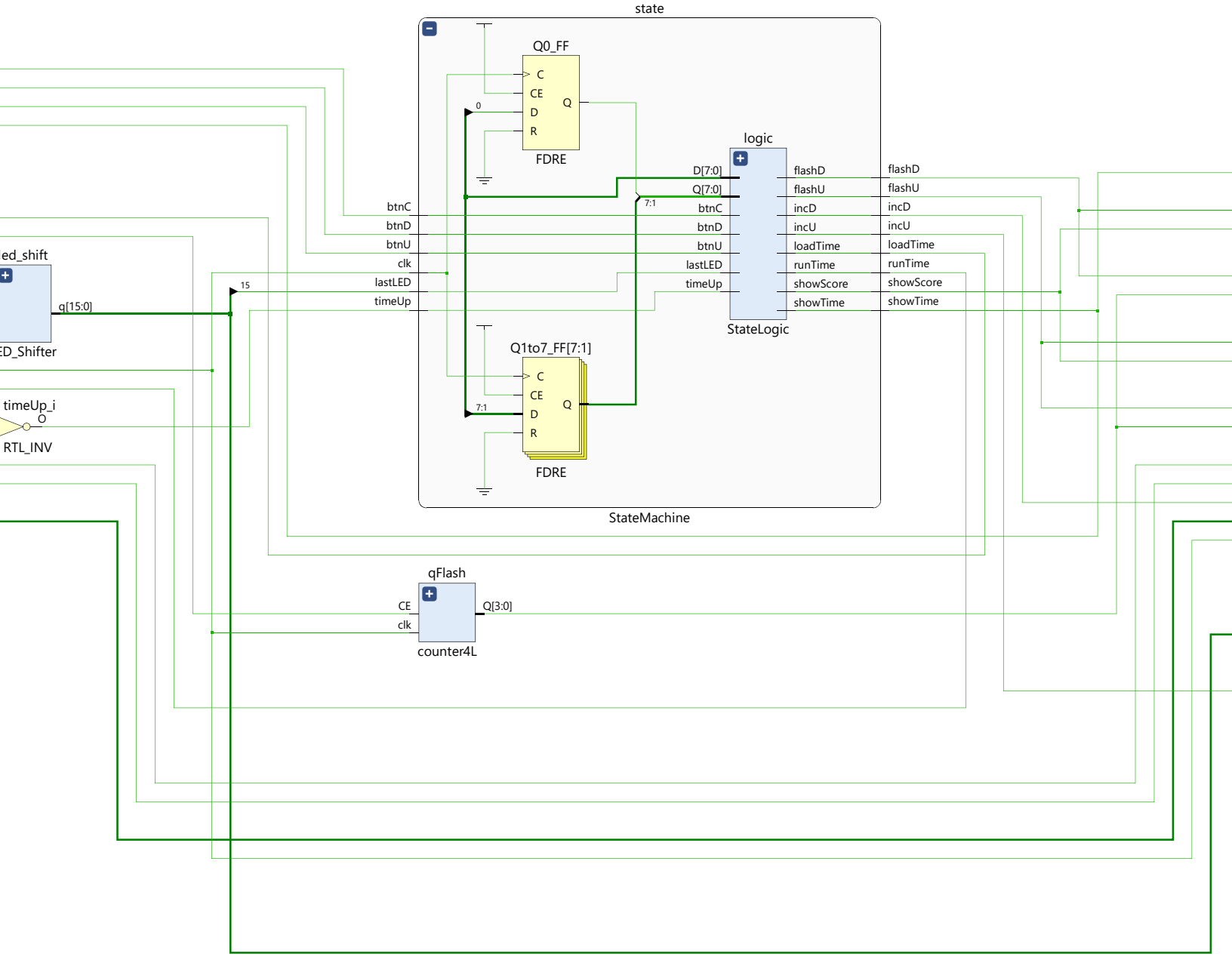
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 11/05/2019 11:13:11 AM
// Design Name:
// Module Name: LSFR
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module LSFR(
    input clk,

    output [7:0] rnd
    );

    wire start;

    assign start = rnd[0] ^ rnd[5] ^ rnd[6] ^ rnd[7];

    FDRE #(.INIT(1'b0) ) ff0 (.C(clk), .R(1'b0), .CE(1'b1),
.D(start), .Q(rnd[0]));
    FDRE #(.INIT(1'b0) ) ff1 (.C(clk), .R(1'b0), .CE(1'b1),
```

```
.D(rnd[0]), .Q(rnd[1]));
    FDRE #(.INIT(1'b0) ) ff2 (.C(clk), .R(1'b0), .CE(1'b1),
.D(rnd[1]), .Q(rnd[2]));
    FDRE #(.INIT(1'b0) ) ff3 (.C(clk), .R(1'b0), .CE(1'b1),
.D(rnd[2]), .Q(rnd[3]));
    FDRE #(.INIT(1'b0) ) ff4 (.C(clk), .R(1'b0), .CE(1'b1),
.D(rnd[3]), .Q(rnd[4]));
    FDRE #(.INIT(1'b0) ) ff5 (.C(clk), .R(1'b0), .CE(1'b1),
.D(rnd[4]), .Q(rnd[5]));
    FDRE #(.INIT(1'b0) ) ff6 (.C(clk), .R(1'b0), .CE(1'b1),
.D(rnd[5]), .Q(rnd[6]));
    FDRE #(.INIT(1'b1) ) ff7 (.C(clk), .R(1'b0), .CE(1'b1),
.D(rnd[6]), .Q(rnd[7])); // initially 1

endmodule
```


btnC
btnD
btnU
clkin
btnR
sw0
CE0_i
RTL_AND
R0_i
RTL_OR
Dw0_i
RTL_AND
random
LSFR
timeC
time_counter
timeUp0_i
RTL_REDUCTION_OR
slowit
lab5_clks
btnR_IBUF_inst
IBUF
led_shift
ff0
ff1
ff2
ff3
ff4
ff5
ff6
ff7
ff8
ff9
ff10
ff11
ff12
ff13
ff14
ff15
FDRE
LED_Shifter
timeUp_i
RTL_INV
state
StateMachine
qFlash
counter4L
an3_i
an3_i_0
an3_i_1
an3_i_2
an2_i
an2_i_0
an2_i_1
an1_i_0
an1_i
an1_i_1
an0_i_0
an0_i
an0_i_1
RTL_INV
an[3:0]
countD
countU
ring
select
selector
segdisp
hex7seg
seg[6:0]
led[15:0]_OBUF_inst
OBUF
led[15:0]
dp_OBUF_inst
dp

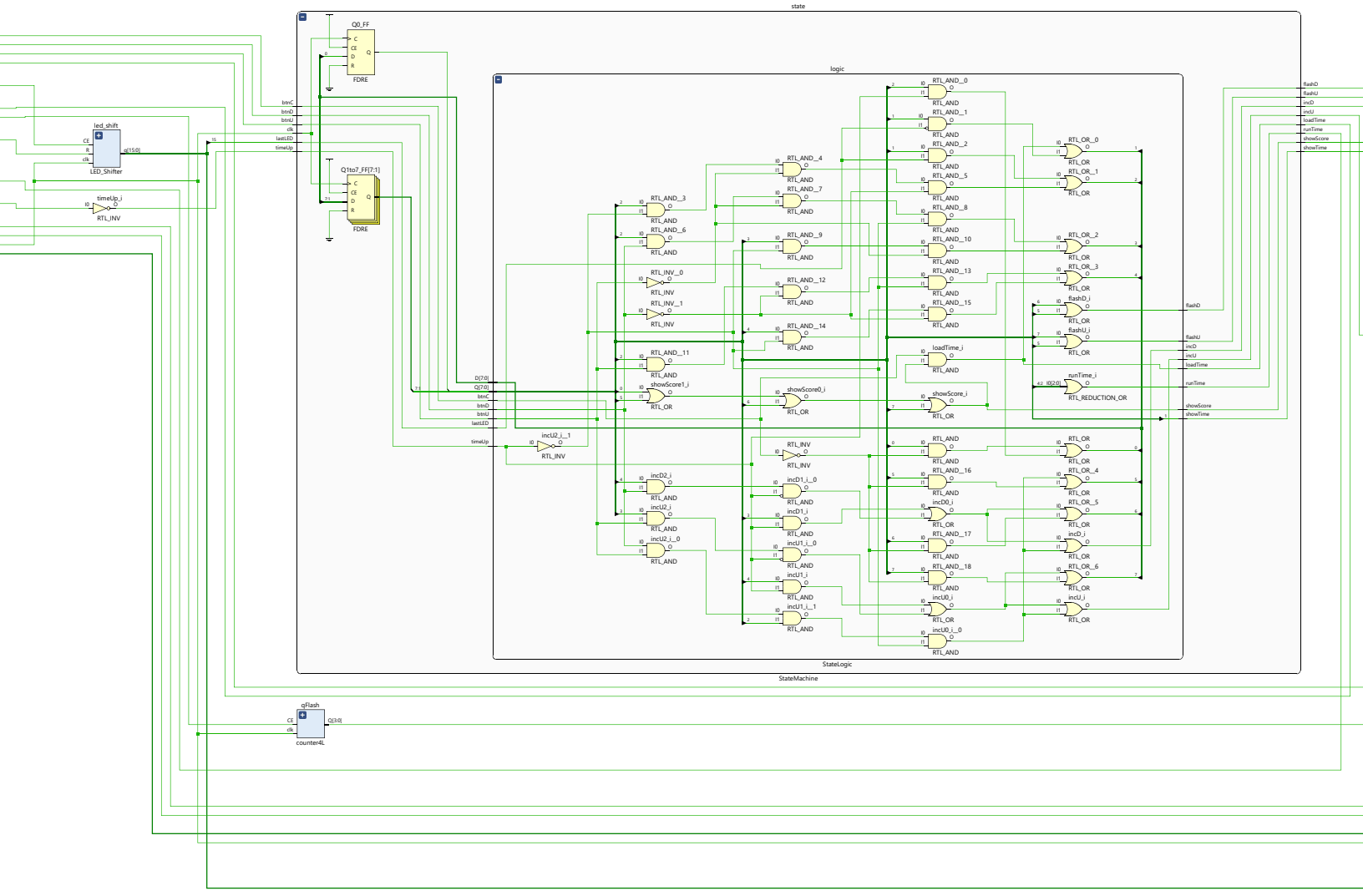
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 10/24/2019 04:06:57 PM
// Design Name:
// Module Name: ring
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module LED_Shifter(
    input CE,
    input clk,
    input R, // reset

    output [15:0] q
    );

    wire [15:0] rcount;

    FDRE #(.INIT(1'b0) ) ff0 (.C(clk), .R(R), .CE(CE), .D(1'b1),
.Q(rcount[0])); // initalizied to 0, takes 1 as its first value
    FDRE #(.INIT(1'b0) ) ff1 (.C(clk), .R(R), .CE(CE),
```

```
.D(rcount[0]), .Q(rcount[1]));
    FDRE #(.INIT(1'b0) ) ff2 (.C(clk), .R(R), .CE(CE),
.D(rcount[1]), .Q(rcount[2]));
    FDRE #(.INIT(1'b0) ) ff3 (.C(clk), .R(R), .CE(CE),
.D(rcount[2]), .Q(rcount[3]));
    FDRE #(.INIT(1'b0) ) ff4 (.C(clk), .R(R), .CE(CE),
.D(rcount[3]), .Q(rcount[4]));
    FDRE #(.INIT(1'b0) ) ff5 (.C(clk), .R(R), .CE(CE),
.D(rcount[4]), .Q(rcount[5]));
    FDRE #(.INIT(1'b0) ) ff6 (.C(clk), .R(R), .CE(CE),
.D(rcount[5]), .Q(rcount[6]));
    FDRE #(.INIT(1'b0) ) ff7 (.C(clk), .R(R), .CE(CE),
.D(rcount[6]), .Q(rcount[7]));
    FDRE #(.INIT(1'b0) ) ff8 (.C(clk), .R(R), .CE(CE),
.D(rcount[7]), .Q(rcount[8]));
    FDRE #(.INIT(1'b0) ) ff9 (.C(clk), .R(R), .CE(CE),
.D(rcount[8]), .Q(rcount[9]));
    FDRE #(.INIT(1'b0) ) ff10 (.C(clk), .R(R), .CE(CE),
.D(rcount[9]), .Q(rcount[10]));
    FDRE #(.INIT(1'b0) ) ff11 (.C(clk), .R(R), .CE(CE),
.D(rcount[10]), .Q(rcount[11]));
    FDRE #(.INIT(1'b0) ) ff12 (.C(clk), .R(R), .CE(CE),
.D(rcount[11]), .Q(rcount[12]));
    FDRE #(.INIT(1'b0) ) ff13 (.C(clk), .R(R), .CE(CE),
.D(rcount[12]), .Q(rcount[13]));
    FDRE #(.INIT(1'b0) ) ff14 (.C(clk), .R(R), .CE(CE),
.D(rcount[13]), .Q(rcount[14]));
    FDRE #(.INIT(1'b0) ) ff15 (.C(clk), .R(R), .CE(CE),
.D(rcount[14]), .Q(rcount[15]));

    assign q = rcount;

endmodule
```


state
Q0_FF
C
CE
D
R
Q
FDRE
0
logic
D[7:0]
Q[7:0]
7:1
btnC
btnD
btnU
clk
lastLED
timeUp
flashD
flashU
incD
incU
loadTime
runTime
showScore
showTime
StateLogic
Q1to7_FF[7:1]
7:1
FDRE
StateMachine
15
q[15:0]
ED_Shifter
timeUp_i
O
RTL_INV
qFlash
CE
clk
Q[3:0]
counter4L

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 11/05/2019 04:33:07 PM
// Design Name:
// Module Name: StateMachine
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module StateMachine(
    input clk,
    input btnC,
    input btnU,
    input btnD,
    input timeUp,
    input lastLED,

    output showTime,
    output loadTime,
    output runTime,
    output incU,
    output incD,
```

```
    output showScore,
    output flashU,
    output flashD
    );

    wire [7:0] Q;
    wire [7:0] D;

    StateLogic  logic( .btnC(btnC), .btnU(btnU), .btnD(btnD),
.timeUp(timeUp), .lastLED(lastLED), .Q(Q), .showTime(showTime),
.loadTime(loadTime), .runTime(runTime), .incU(incU), .incD(incD),
.showScore(showScore), .flashU(flashU), .flashD(flashD), .D(D) );

    FDRE #(.INIT(1'b1)) Q0_FF (.C(clk), .CE(1'b1), .D(D[0]),
.Q(Q[0])); // greset gives 0000 0001
    FDRE #(.INIT(1'b0)) Q1to7_FF[7:1] (.C({7{clk}}),
.CE({7{1'b1}}), .D(D[7:1]), .Q(Q[7:1]));

endmodule
```


state
logic
Q0_FF
FDRE
Q1to7_FF[7:1]
FDRE
led_shift
LED_Shifter
timeUp_i
RTL_INV
btnC
btnD
btnU
clk
lastLED
timeUp
D[7:0]
Q[7:0]
RTL_AND__0
RTL_AND__1
RTL_AND__2
RTL_AND__3
RTL_AND__4
RTL_AND__5
RTL_AND__6
RTL_AND__7
RTL_AND__8
RTL_AND__9
RTL_AND__10
RTL_AND__11
RTL_AND__12
RTL_AND__13
RTL_AND__14
RTL_AND__15
RTL_AND__16
RTL_AND__17
RTL_AND__18
RTL_INV__0
RTL_INV__1
RTL_OR__0
RTL_OR__1
RTL_OR__2
RTL_OR__3
RTL_OR__4
RTL_OR__5
RTL_OR__6
loadTime_i
showScore0_i
showScore1_i
showScore_i
runTime_i
flashD_i
flashU_i
incD0_i
incD1_i
incD1_i__0
incD2_i
incD_i
incU0_i
incU0_i__0
incU1_i
incU1_i__1
incU1_i__0
incU2_i
incU2_i__0
incU2_i__1
incU_i
RTL_REDUCTION_OR
StateLogic
StateMachine
qFlash
counter4L

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 11/05/2019 04:57:33 PM
// Design Name:
// Module Name: StateLogic
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module StateLogic(
    input btnC,
    input btnU,
    input btnD,
    input timeUp,
    input lastLED,
    input [7:0] Q,

    output showTime,
    output loadTime,
    output runTime,
    output incU,
    output incD,
```

```
    output showScore,
    output flashU,
    output flashD,
    input [7:0] D
    );

    // NEXT STATE
    assign D[7] = (Q[4] & timeUp) | (Q[3] & btnU & ~timeUp) | (Q[7]
& ~btnC); // Uwin
    assign D[6] = (Q[3] & timeUp) | (Q[4] & btnD & ~timeUp) | (Q[6]
& ~btnC); // Dwin
    assign D[5] = (btnD & btnU & Q[2] & ~timeUp) | (Q[5] & ~btnC);
// both
    assign D[4] = (Q[2] & btnU & ~btnD & ~timeUp) | (Q[4] & ~timeUp
& ~btnD); // Ufirst
    assign D[3] = (Q[2] & btnD & ~btnU & ~timeUp) | (Q[3] & ~timeUp
& ~btnU); // Dfirst
    assign D[2] = (Q[1] & lastLED) | (Q[2] & ~timeUp & ~btnU &
~btnD); // Go
    assign D[1] = (btnC & (Q[0] | Q[5] | Q[6] | Q[7])) | (Q[1] &
~lastLED); // ready
    assign D[0] = (Q[0] & ~btnC) | (Q[2] & timeUp); // before


    // OUTPUTS
    assign showTime = Q[1];
    assign loadTime = btnC & (Q[0] | Q[5] | Q[6] | Q[7]);
    assign runTime = |Q[4:2];
    assign incU = (Q[4] & timeUp) | (Q[3] & btnU & ~timeUp) | (btnD
& btnU & Q[2] & ~timeUp);
    assign incD = (Q[3] & timeUp) | (Q[4] & btnD & ~timeUp) | (btnD
& btnU & Q[2] & ~timeUp);
    assign showScore = Q[0] | Q[5] | Q[6] | Q[7];
    assign flashU = (Q[7] | Q[5]);
    assign flashD = (Q[6] | Q[5]);

endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 10/24/2019 03:05:17 PM
// Design Name:
// Module Name: countUD16L
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module time_counter(
    input clk,
    input Up,
    input Dw,
    input LD,
    input [15:0] Din, // only for load enabled

    output [15:0] Q
    );

    wire [3:0] outUTC, outDTC;
    wire [15:0] tempQ;
```

```
    countUD3L count3_0(.clk(clk), .Up(Up), .Dw(Dw), .LD(LD),
.Din(Din[2:0]), .Q(tempQ[2:0]), .UTC(outUTC[0]), .DTC(outDTC[0]));
    countUD5L count5_1(.clk(clk), .Up(outUTC[0]&Up&~LD),
.Dw(outDTC[0]&Dw), .LD(LD), .Din(Din[7:3]), .Q(tempQ[7:3]),
.UTC(outUTC[1]), .DTC(outDTC[1]));
    countUD5L count5_2(.clk(clk), .Up((&outUTC[1:0])&Up&~LD),
.Dw((&outDTC[1:0])&Dw&~LD), .LD(LD), .Din(Din[12:8]),
.Q(tempQ[12:8]), .UTC(outUTC[2]), .DTC(outDTC[2]));
    countUD3L count3_3(.clk(clk), .Up((&outUTC[2:0])&Up&~LD),
.Dw((&outDTC[2:0])&Dw&~LD), .LD(LD), .Din(Din[15:13]),
.Q(tempQ[15:13]), .UTC(outUTC[3]), .DTC(outDTC[3]));

    assign Q = tempQ;

endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 10/22/2019 03:44:43 PM
// Design Name:
// Module Name: countUD3L
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module countUD3L(
    input clk,
    input Up,
    input Dw,
    input LD,
    input [2:0] Din, // only for load enabled

    output [2:0] Q,
    output UTC,
    output DTC
    );

    wire enable;
```

```
    assign enable = Up | Dw | LD; // if all on then load overrides.
up/dw cancel out
    // ~enable = ~Up & ~Dw & ~LD;

    wire [2:0] D;

    FDRE #(.INIT(1'b0) ) ff1 (.C(clk), .R(1'b0), .CE(enable),
.D(D[0]), .Q(Q[0]));
    FDRE #(.INIT(1'b0) ) ff2 (.C(clk), .R(1'b0), .CE(enable),
.D(D[1]), .Q(Q[1]));
    FDRE #(.INIT(1'b0) ) ff3 (.C(clk), .R(1'b0), .CE(enable),
.D(D[2]), .Q(Q[2]));

    assign D[0] = (~Q[0] & (Up ^ Dw) & ~LD) | (Din[0] & LD) | (Q[0]
& ~enable);
    assign D[1] = ((Q[1] ^ (Up & Q[0])) & ~Dw & ~LD) | ((Q[1] ^ (Dw
& ~Q[0])) & ~Up & ~LD) | (Din[1] & LD) | (Q[1] & ~enable);
    assign D[2] = ((Q[2] ^ (Up & Q[1] & Q[0])) & ~Dw & ~LD) |
((Q[2] ^ (Dw & ~Q[1] & ~Q[0])) & ~Up & ~LD) | (Din[2] & LD) | (Q[2]
& ~enable);

    assign UTC = &Q;
    assign DTC = &(~Q);

endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 10/22/2019 03:45:06 PM
// Design Name:
// Module Name: countUD5L
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module countUD5L(
    input clk,
    input Up,
    input Dw,
    input LD,
    input [4:0] Din, // only for load enabled

    output [4:0] Q,
    output UTC,
    output DTC
    );

    wire enable;
```

```
    assign enable = Up | Dw | LD; // if all on then load overrides.
up/dw cancel out
    // ~enable = ~Up & ~Dw & ~LD;

    wire [4:0] D;

    FDRE #(.INIT(1'b0) ) ff0 (.C(clk), .R(1'b0), .CE(enable),
.D(D[0]), .Q(Q[0]));
    FDRE #(.INIT(1'b0) ) ff1 (.C(clk), .R(1'b0), .CE(enable),
.D(D[1]), .Q(Q[1]));
    FDRE #(.INIT(1'b0) ) ff2 (.C(clk), .R(1'b0), .CE(enable),
.D(D[2]), .Q(Q[2]));
    FDRE #(.INIT(1'b0) ) ff3 (.C(clk), .R(1'b0), .CE(enable),
.D(D[3]), .Q(Q[3]));
    FDRE #(.INIT(1'b0) ) ff4 (.C(clk), .R(1'b0), .CE(enable),
.D(D[4]), .Q(Q[4]));

    assign D[0] = (~Q[0] & (Up ^ Dw) & ~LD) | (Din[0] & LD) | (Q[0]
& ~enable);
    assign D[1] = ((Q[1] ^ (Up & Q[0])) & ~Dw & ~LD) | ((Q[1] ^ (Dw
& ~Q[0])) & ~Up & ~LD) | (Din[1] & LD) | (Q[1] & ~enable);
    assign D[2] = ((Q[2] ^ (Up & Q[1] & Q[0])) & ~Dw & ~LD) |
((Q[2] ^ (Dw & ~Q[1] & ~Q[0])) & ~Up & ~LD) | (Din[2] & LD) | (Q[2]
& ~enable);
    assign D[3] = ((Q[3] ^ (Up & Q[2] & Q[1] & Q[0])) & ~Dw & ~LD)
| ((Q[3] ^ (Dw & ~Q[2] & ~Q[1] & ~Q[0])) & ~Up & ~LD) | (Din[3] &
LD) | (Q[3] & ~enable);
    assign D[4] = ((Q[4] ^ (Up & Q[3] & Q[2] & Q[1] & Q[0])) & ~Dw
& ~LD) | ((Q[4] ^ (Dw & ~Q[3] & ~Q[2] & ~Q[1] & ~Q[0])) & ~Up &
~LD) | (Din[4] & LD) | (Q[4] & ~enable);

    assign UTC = &Q;
    assign DTC = &(~Q);

endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 10/22/2019 03:45:06 PM
// Design Name:
// Module Name: countUD5L
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module counter4L(
    input clk,
    input CE,

    output [3:0] Q
    );

    wire [3:0] D;

    FDRE #(.INIT(1'b0) ) ff0 (.C(clk), .R(1'b0), .CE(CE), .D(D[0]),
.Q(Q[0]));
    FDRE #(.INIT(1'b0) ) ff1 (.C(clk), .R(1'b0), .CE(CE), .D(D[1]),
.Q(Q[1]));
```

```
    FDRE #(.INIT(1'b0) ) ff2 (.C(clk), .R(1'b0), .CE(CE), .D(D[2]),
.Q(Q[2]));
    FDRE #(.INIT(1'b0) ) ff3 (.C(clk), .R(1'b0), .CE(CE), .D(D[3]),
.Q(Q[3]));

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

endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 10/24/2019 04:06:57 PM
// Design Name:
// Module Name: ring
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module ring(
    input digsel,
    input clk,

    output [3:0] out
    );

    wire [3:0] rcount;

    FDRE #(.INIT(1'b1) ) ff1 (.C(clk), .R(1'b0), .CE(digsel),
.D(rcount[3]), .Q(rcount[0])); // initalizied to 1
    FDRE #(.INIT(1'b0) ) ff2 (.C(clk), .R(1'b0), .CE(digsel),
.D(rcount[0]), .Q(rcount[1]));
```

```
    FDRE #(.INIT(1'b0) ) ff3 (.C(clk), .R(1'b0), .CE(digsel),
.D(rcount[1]), .Q(rcount[2]));
    FDRE #(.INIT(1'b0) ) ff4 (.C(clk), .R(1'b0), .CE(digsel),
.D(rcount[2]), .Q(rcount[3]));

    assign out = rcount;

endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 10/24/2019 03:40:36 PM
// 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] N,

    output [3:0] H
    );
    //H is x[15:12] when sel=(1000)
    //H is x[11:8] when sel=(0100)
    //H is x[7:4] when sel=(0010)
    //H is x[3:0] when sel=(0001)

    wire fail;
    assign fail = (sel[0] & (|sel[3:1])) | (sel[1] & (|sel[3:2])) |
```

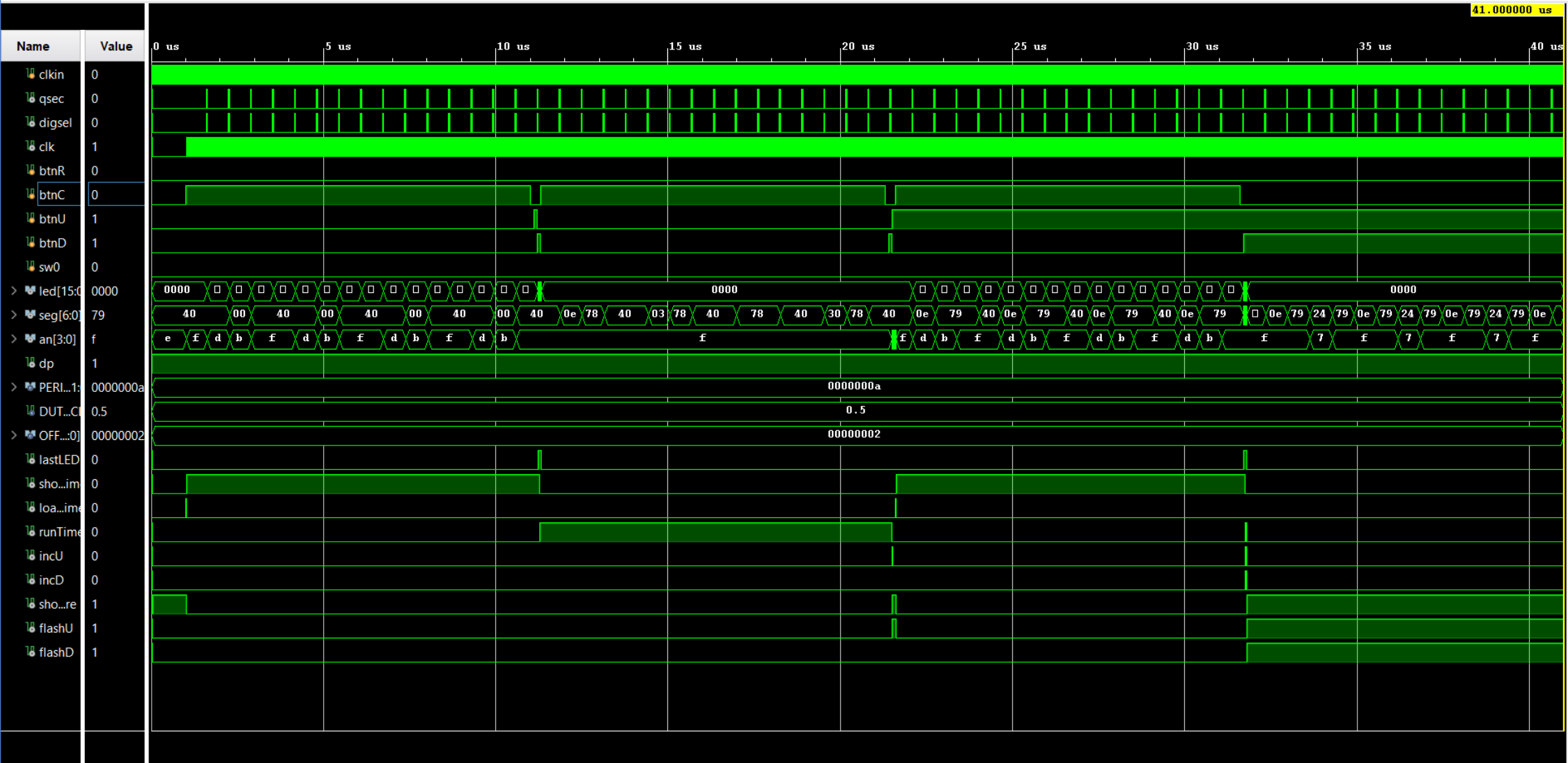
```
(sel[2] & sel[3]); // more than one is on

    assign H = (N[15:12] & {4{sel[3] & ~fail}}) | (N[11:8] &
{4{sel[2] & ~fail}}) | (N[7:4] & {4{sel[1] & ~fail}}) | (N[3:0] &
{4{sel[0] & ~fail}});

endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 10/16/2019 06:41:25 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, // enable

    output [6:0] seg
    );

    m8_1e mux8A( .in({1'b0, n[0], n[0], 1'b0, 1'b0, ~n[0], 1'b0,
n[0]}), .sel({n[3], n[2], n[1]}), .e(e), .o(seg[0])); // A
    m8_1e mux8B( .in({1'b1, ~n[0], n[0], 1'b0, ~n[0], n[0], 1'b0,
1'b0}), .sel({n[3], n[2], n[1]}), .e(e), .o(seg[1])); // B
    m8_1e mux8C( .in({1'b1, ~n[0], 1'b0, 1'b0, 1'b0, 1'b0, ~n[0],
1'b0}), .sel({n[3], n[2], n[1]}), .e(e), .o(seg[2])); // C
```

```
    m8_1e mux8D( .in({n[0], 1'b0, ~n[0], n[0], n[0], ~n[0], 1'b0,
n[0]}), .sel({n[3], n[2], n[1]}), .e(e), .o(seg[3])); // D
    m8_1e mux8E( .in({1'b0, 1'b0, 1'b0, n[0], n[0], 1'b1, n[0],
n[0]}), .sel({n[3], n[2], n[1]}), .e(e), .o(seg[4])); // E
    m8_1e mux8F( .in({1'b0, n[0], 1'b0, 1'b0, n[0], 1'b0, 1'b1,
n[0]}), .sel({n[3], n[2], n[1]}), .e(e), .o(seg[5])); // F
    m8_1e mux8G( .in({1'b0, ~n[0], 1'b0, 1'b0, n[0], 1'b0, 1'b0,
1'b1}), .sel({n[3], n[2], n[1]}), .e(e), .o(seg[6])); // G

endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 10/15/2019 03:33:43 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, // enable

    output o
    );

    wire [7:0] temp;

    assign temp[0] = in[0] & ~sel[2] & ~sel[1] & ~sel[0]; // 000
    assign temp[1] = in[1] & ~sel[2] & ~sel[1] & sel[0];  // 001
    assign temp[2] = in[2] & ~sel[2] & sel[1] & ~sel[0];  // 010
```

```
    assign temp[3] = in[3] & ~sel[2] & sel[1] & sel[0];   // 011
    assign temp[4] = in[4] & sel[2] & ~sel[1] & ~sel[0];  // 100
    assign temp[5] = in[5] & sel[2] & ~sel[1] & sel[0];   // 101
    assign temp[6] = in[6] & sel[2] & sel[1] & ~sel[0];   // 110
    assign temp[7] = in[7] & sel[2] & sel[1] & sel[0];    // 111

    assign o = (|temp) & e;

endmodule
```


41.000000 us
Name
Value
0 us
5 us
10 us
15 us
20 us
25 us
30 us
35 us
40 us
clkin 0
qsec 0
digsel 0
clk 1
btnR 0
btnC 0
btnU 1
btnD 1
sw0 0
led[15:0] 0000
seg[6:0] 79
an[3:0] f
dp 1
PERI...1: 0000000a
DUT...CI 0.5
OFF...:0] 00000002
lastLED 0
sho...im 0
loa...ime 0
runTime 0
incU 0
incD 0
sho...re 1
flashU 1
flashD 1
0000
0000
0000
0000000a
0.5
00000002

Lab 5

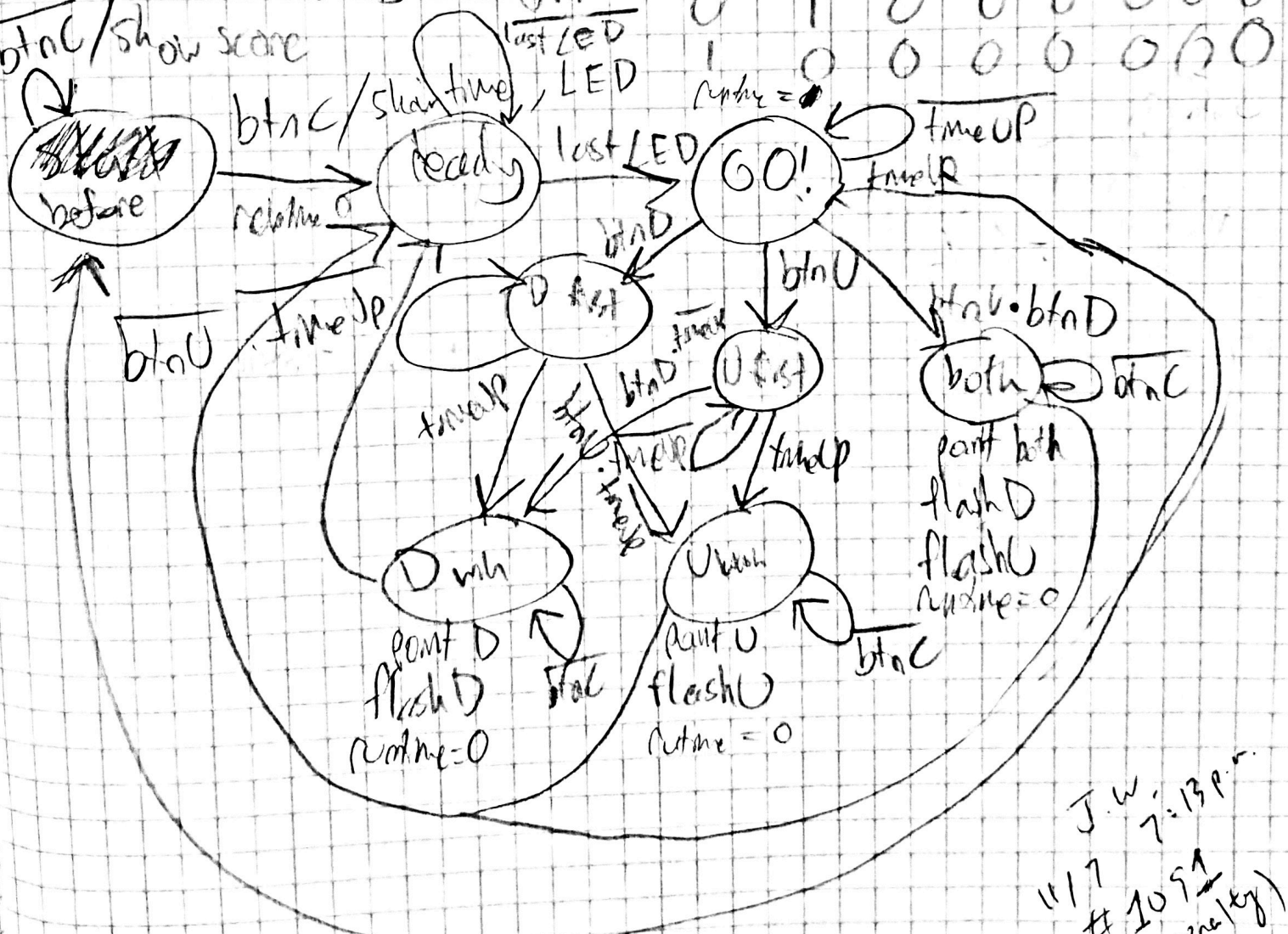
11/5/14
3:20PM

Inputs
- btnC
- btnU
- btnD
- timeUp
- last LED

Outputs
- showTime
- load time
- run time
- inCU
- inCD
- showScore
- flashU
- flashD

GG

| | $Q_7$ | $Q_6$ | $Q_5$ | $Q_4$ | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
|---|---|---|---|---|---|---|---|---|
| reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| ready | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| go | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Dtest | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Utest | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| both | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| Dwin | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Uwin | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| lastLED | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |




J.W. 7:13P.M.
11/7
#10 91
(2% penalty)

# Lab 5 - States

0 before = before · $\overline{btnC}$ + go · timeUp

1 ready = btnC(before + Dwin + Uwin + both) + ready · $\overline{lastLED}$

2 go = ready · lastLED + go · $\overline{timeUp}$

3 Dfirst = go · btnD + Dfirst · $\overline{timeUp}$ · $\overline{btnU}$

4 Ufirst = go · btnU + Ufirst · $\overline{timeUp}$ · $\overline{btnD}$

5 both = btnU · btnD · go + both · $\overline{btnC}$

6 Dwin = Dfirst · timeUp + Ufirst · btnD + Dwin · $\overline{btnC}$

7 Uwin = Ufirst · timeup + Dfirst · btnU + Uwin · $\overline{btnC}$

Stop counter when scored

fix scoring.