Frank I Gomez
5/11/21
Lab 5, Section D.
ID: 1650550
figomez
CSE 100L

Lab 5 Write Up

1. Description

The purpose of this lab was to create a finite state machine for a small game. The game in question features a timer that's activated on button press. Once the timer reaches 0, LEDs will turn on, and the first player to flip their switch wins (although they can both win if they flip them simultaneously), granting them a point. Flipping the switch early grants the other player a point, unless both players simultaneously flip their switches early, in which case, neither player scores. Finally, flipping the 4th switch displays the countdown, allowing the players to cheat.

2. Design

- Top Level

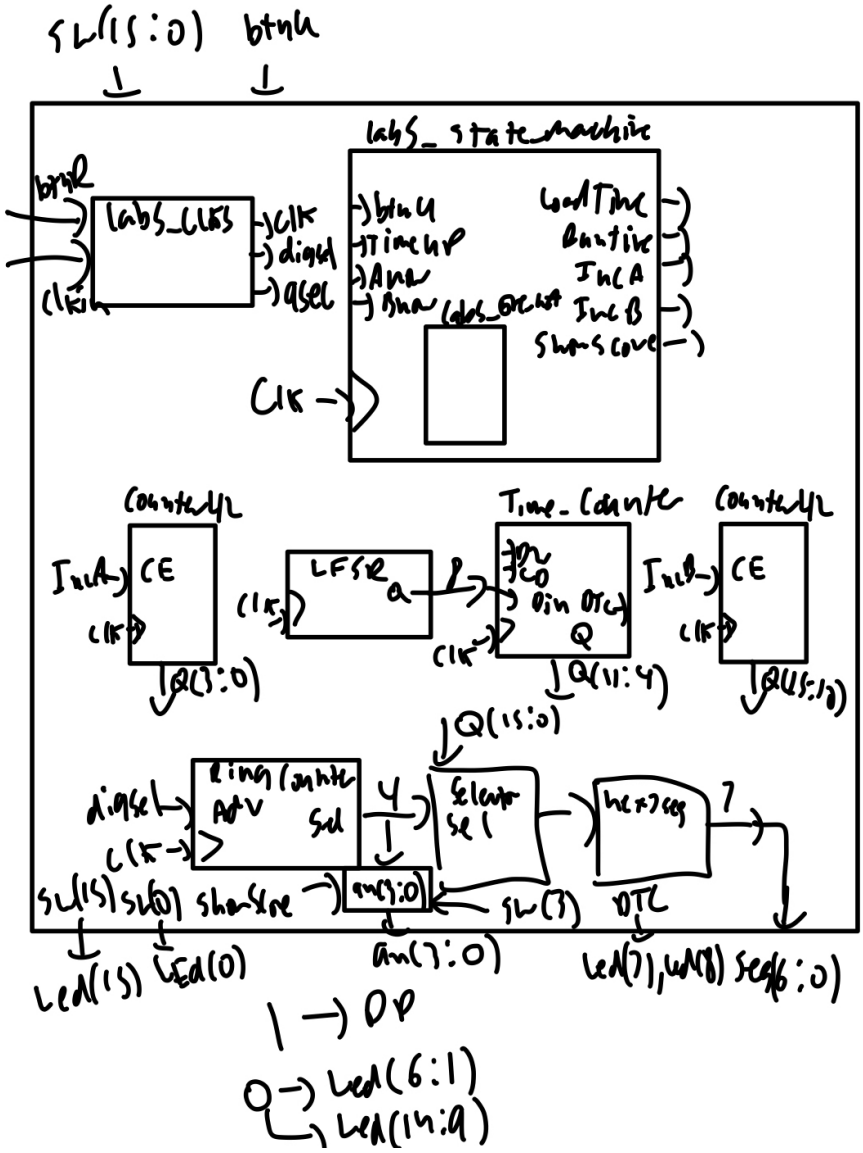


Diagram 1

Top level again, there's simultaneously so much to go over, and yet, so little to actually explain. Like in lab 4, lab5_clks takes in the system clock (clkin), and creates the clk input used everywhere else. Digsel is used for the ring counter as before, and qsec handles Time_Counter's decremental process. Lab5_state_machine handles most of the logic for the lab itself, so it'll be covered on its own in the next section. The only other important things to know, is that switches 0 and 15 enable the respective leds 0 and 15, Time_Counter's DTC controls leds 7 and 8, in addition to being used in the state machine. Ring_Counter, Selector, and hex7seg are the exact same as the prior lab, with Ring_Counter's sel output also being used to change the displayed value of an[3:0], in coordination with switch 3 for an[2:1], and ShowScore for an[0] and an[3].

DP is always high, since it’s not used, and the remaining 12 leds are always off. In addition, Selector’s Q input value is composed of, in the following order, the 4 bits representing Player A’s score, the 8 bits representing the current value of the counter, and the 4 bits representing Player B’s score.

- lab5_state_machine and lab5_one_hot

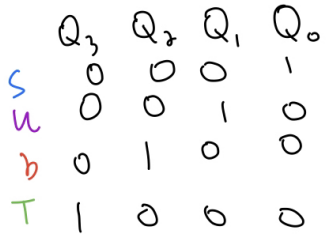


Diagram 2, One Hot Encoding labels for each state

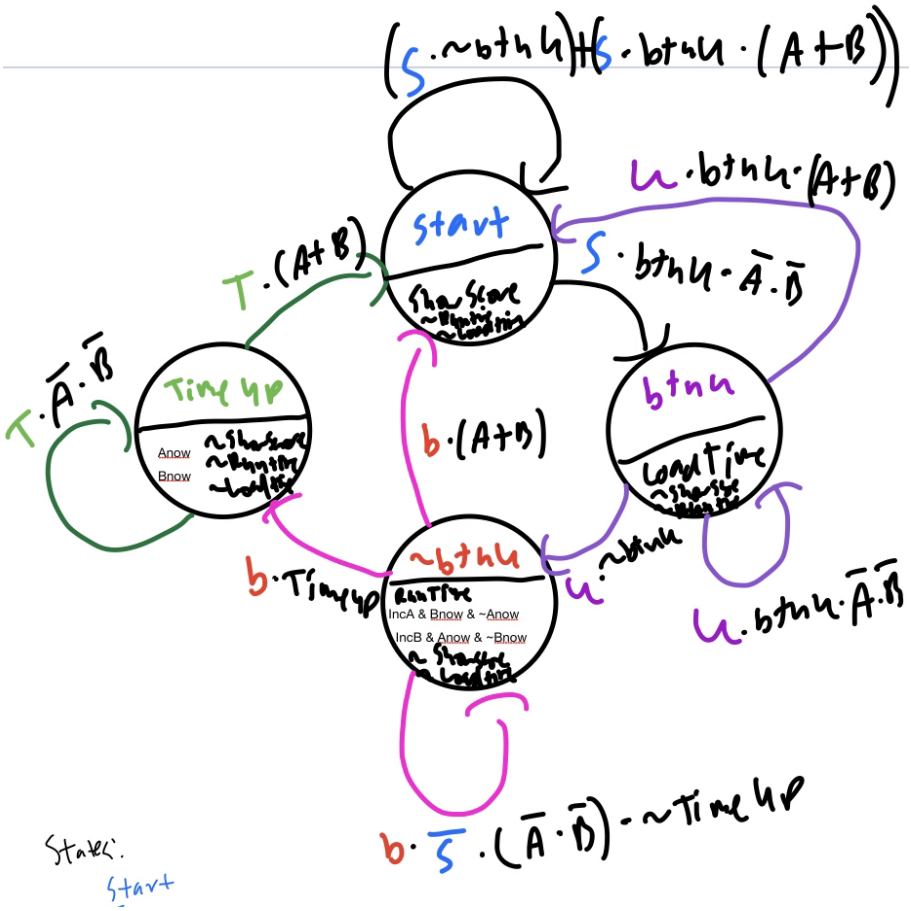


Diagram 3, State Diagram

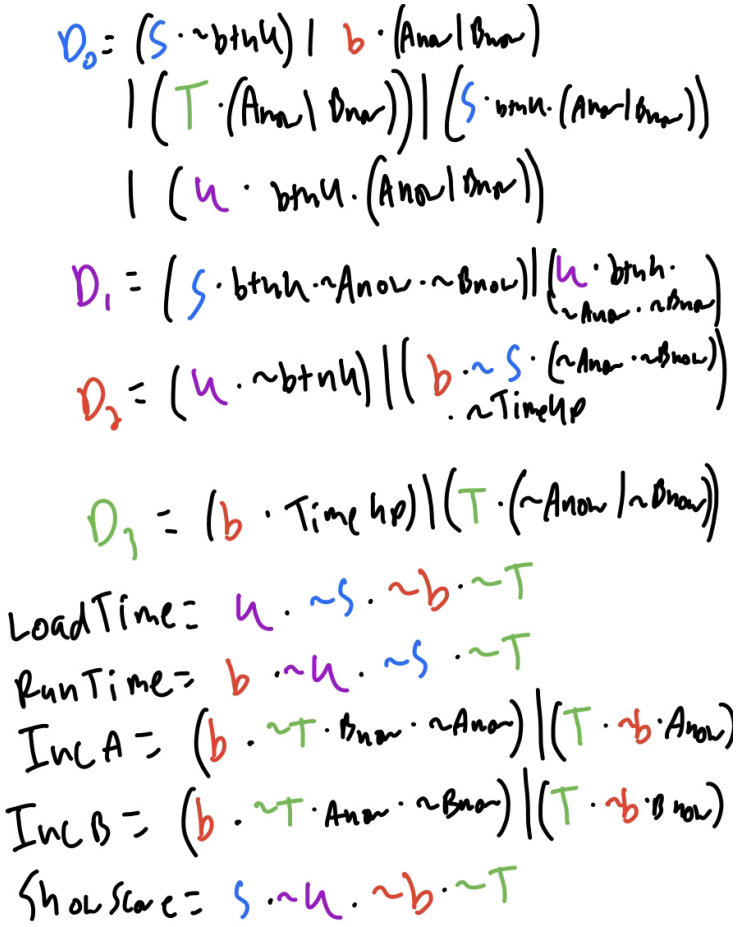


Diagram 4, Next state and output equations

There's a lot going on here, but first, I'm going to preface this section by stating that this section is *technically* two modules, lab5_state_machine and lab5_one_hot. These modules could easily be combined into one, but were split as they make the schematic easier to read. Lab5_state_machine contains **all the output equations**, while lab5_one_hot contains **all the next state equations.** That is the only distinction between the two files, and something that could easily be combined into one. Now, let's get into each state of the machine.

- The machine is in state "Start" when:
    - The machine is currently in "Start" and btnU has not been pressed.
    - The machine is currently in "Start", btnU has been pressed, but inputs (switches) A or B are high.
    - The machine is currently in "Not Up", and inputs A or B are high.
    - The machine is currently in "TimeUp" and inputs A or B are high.
    - The machine is currently in "Up" and inputs A or B are high.
- LEDs 7 and 8 are on in this state. The score is displayed in this state. This state is exited when btnU is high and inputs A and B are low.

- The machine is in state “Up” when:
    - The machine is currently in “Up” and neither A nor B are high.
    - The machine is currently in “Start”, btnU is high, and inputs A and B are low.
- There are no LEDs on in this state. There is no display on the seven segment display in this state, unless switch 3 is high, displaying “88” (all segments enabled from continuous loading). This state loads the value in LFSR into the 8bit counter. This state is exited when btnU is released, assuming inputs A and B are low.

- The machine is in state “Not Up” when:
    - The machine is currently in state “Not Up”, and inputs A, B, and TimeUp are all low.
    - The machine is currently in state “Up” and btnU is low/released.
- There are no LEDs on in this state. There is no display on the seven segment display in this state, unless switch 3 is high, in which case, the current counter display will be shown. Likewise, this state continuously sends a “Down” count to Time_Counter, causing it to decrement at the rate specified by qsec. This state is excited when TimeUp is received from the Time_Counter, or if A or B are high, sending an IncB/IncA signal to score a point for whichever did not flip their switch. If both switches are flipped simultaneously, neither side scores points.

- The machine is in state “TimeUp” when:
    - The machine is currently in state “Time Up” and inputs A and B are low.
    - The machine is currently in state “Not Up” and TimeUp is high.
- LEDs 7 and 8 are enabled in this state. If switch 3 is high, 00 will be displayed on the seven segment display, else nothing will be shown on the display. This state is exited when inputs A or B are high, sending their corresponding IncA/IncB signals to increase their score. If both switches are flipped simultaneously, both sides score a point.

- LFSR

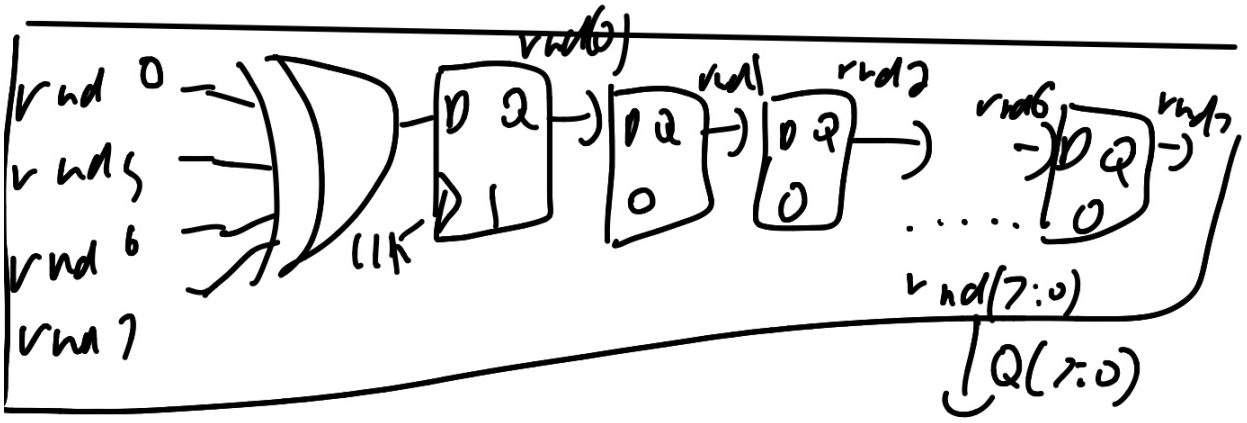


Diagram 5

.

The LFSR contains 8 flip flops, with each output connected to the input of the next. The only exception is the first flip flop, rnd[0], which receives its input from an XOR of rnd[0], rnd[5], rnd[6], and rnd[7]. The first flip flop is also initialized with a value of 1. This combination of flip flops and XOR gates randomly forms an 8bit value, which is then forwarded to the Time_Counter. Only the bottom 6 bits are passed, however, to prevent the timer from taking too long, with the top 2 always being set to 0.

- Time_Counter

Almost identical to countUD16L from lab 4, the only difference is that all references to bits 8-15 have been removed, and all “Up” wires have also been removed. Thus, the counter can only count down now. Its “Dw” input is the AND of qsec and RunTime from the state machine.

- counter4L

Entirely the same as lab 4. It is only included in this report to discuss its unchanged inputs, as Time_Counter needs to use it to count down, while countA and countB only use it to count up. Thus, for Time_Counter, Up is always set to 1'b0, while for countA and countB, Dw is always set to 1'b0.

- lab5_clks

Takes in verilog's clock and breaks it into a cycle of rising and falling edges, to allow the entire file to function on a cycle. Also provides digsel for the ring counter, and qsec for the Time_Counter.

3. Testing and Simulation

After creating the first testbench, which existed solely to test lab5_state_machine, I did notice a bug. Every time an A or B input was received the second and third states, the state machine would briefly, for a single clock cycle, enter two states at once, but still ultimately obtain the expected output. I thought nothing of this, since the machine still worked, and left it as it was. It was only when I programmed the board that I noticed the extent of the bug. Scoring after the timer hit 0 worked as expected, but scoring before was bugged, as it gave the other player 2 points instead of 1. Following this, I made my full testbench for the entire lab, and was able to pick up a few more bugs I had noticed earlier, mainly the case of unset "Up" values in Time_Counter, and the real reason why two states were being entered at once. It turned out that I had accidentally made a logically equivalent absorption statement in my next state equations, which led to the stricter state being absorbed. After fixing this, the state machine worked as expected, and the "double score" bug was fixed.

4. Conclusion

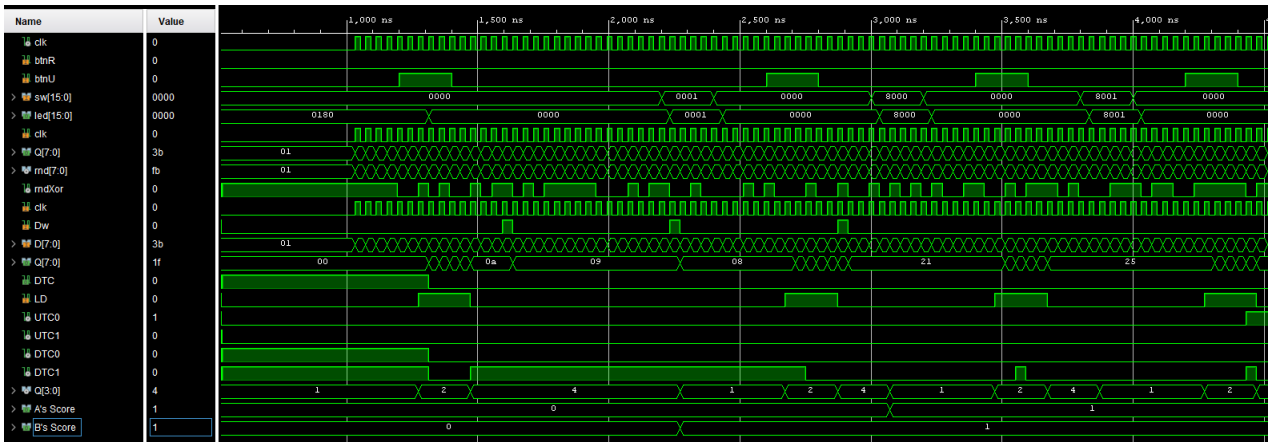
In this lab we learned how to create a finite state machine using flip flops and a state diagram, with the end goal being to create a small, simple game. If I were to do this lab again, I would remind myself to try simplifying my equations, as accidentally logical equivalence ended up being my largest bug. I would also optimize my state machine, as I think it is possible to decrease it to 3 states with enough effort and planning.

5. Appendix

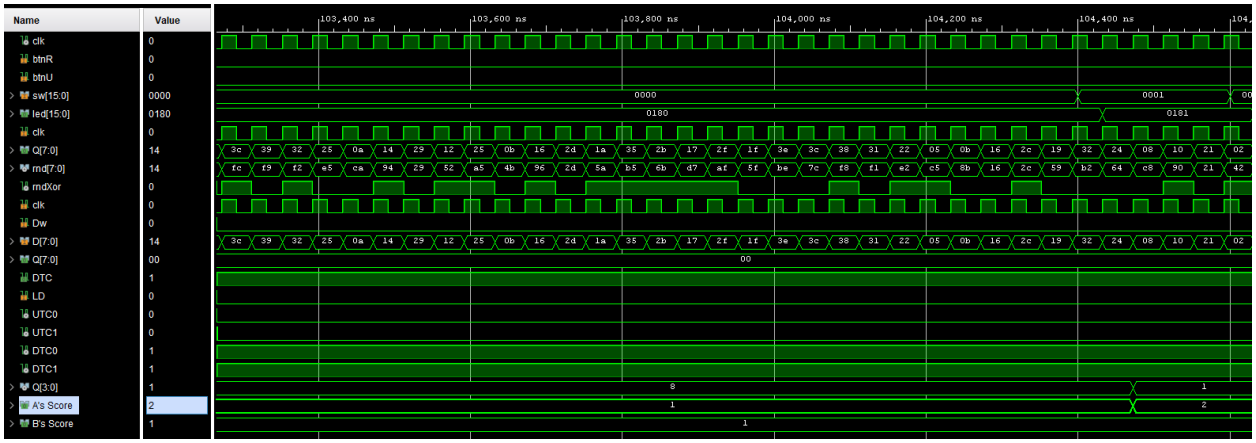
(Note: Waveform Viewer cannot be printed to PDF, and as such, it is added here as a screenshot.)
(Note: hex7seg, Selector, Ring_Counter, and m8_1e are from labs 3 and 4, and as such, will not be included.)

Screenshots of the Waveform Viewer showing simulation results for each of the scenarios below. The following input and output signals should be included: clk, btnR, btnU, sw[], led[], as well as the LFSR contents, the Time Counter, the state bits of the State Machine. Display as buses in Hex.

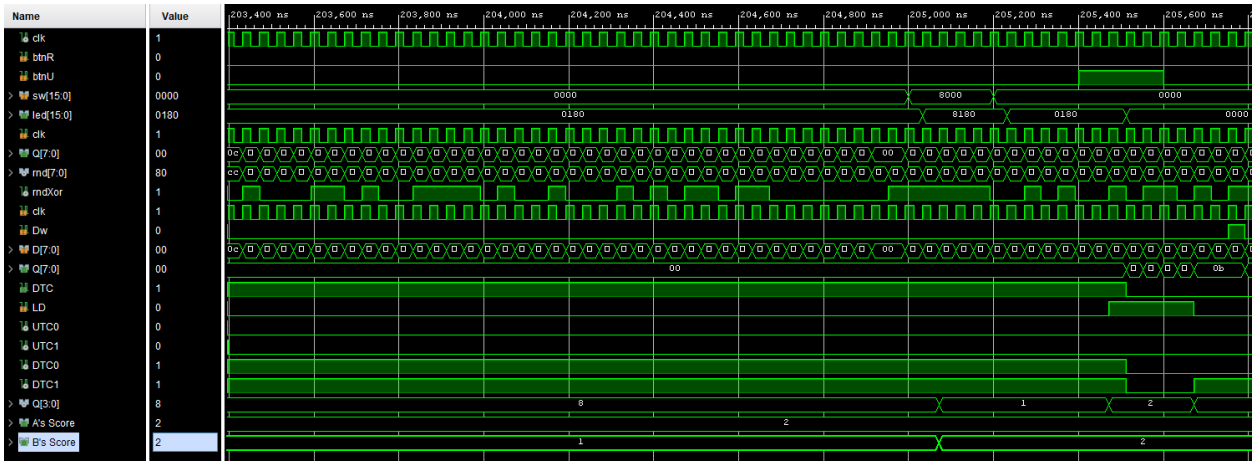
Player B winning by Player A flipping the switch before the green light.
Player A winning by Player B flipping the switch before the green light.
Players A and B both losing by flipping their switches at the same time before the green light.



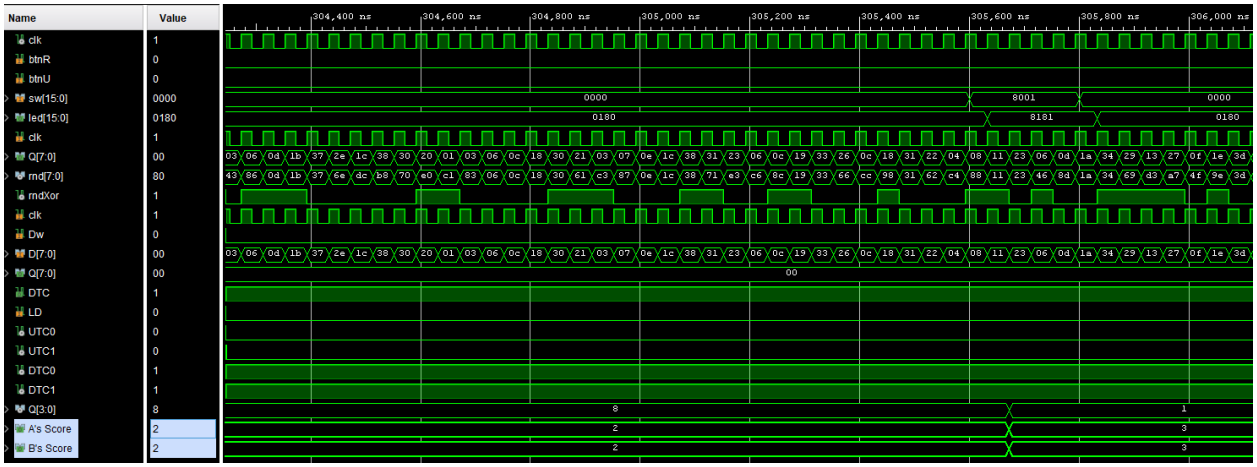
Player A winning by being first after the green light.



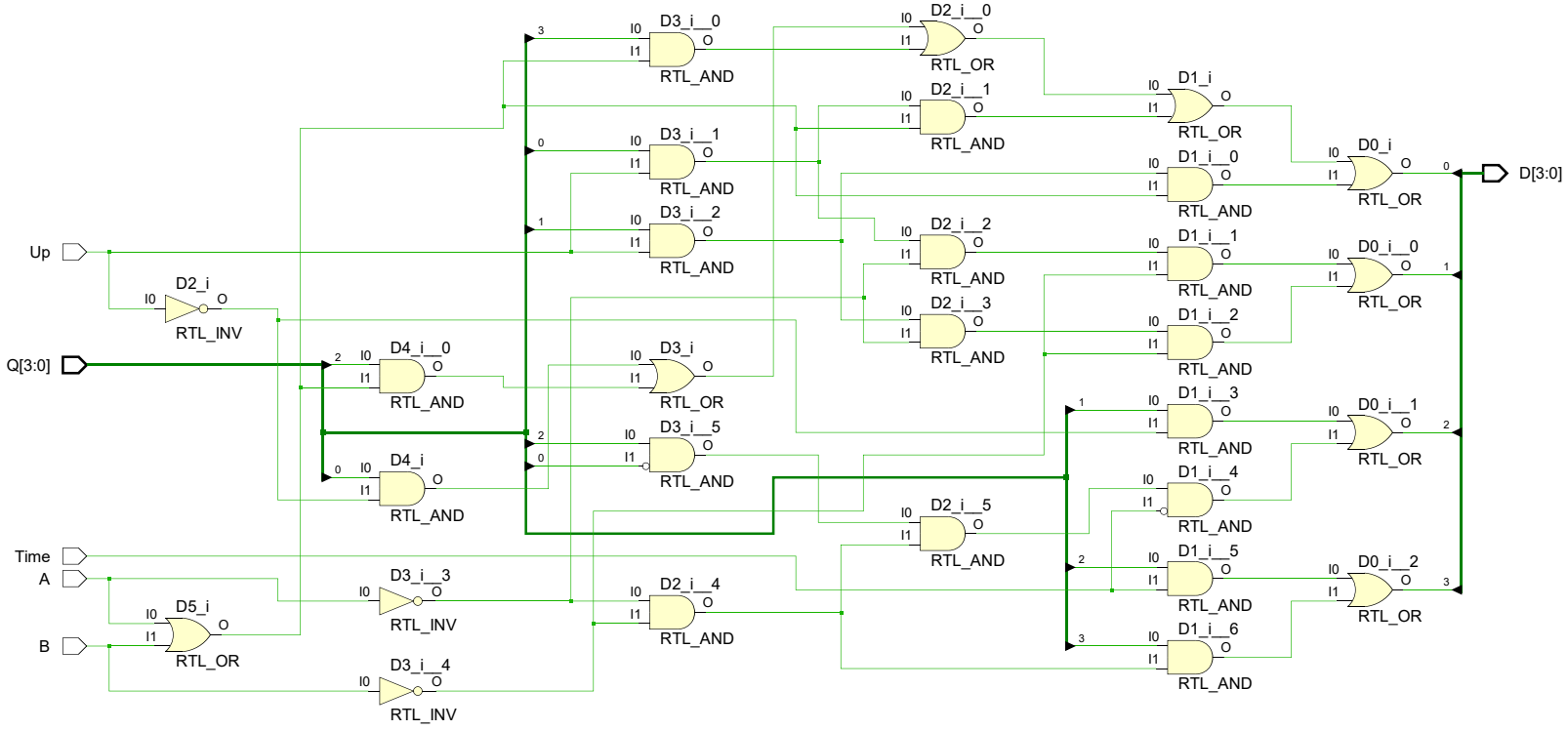
Player B winning by being first after the green light.



Players A and B both winning by flipping their switches at the same time after the green light.

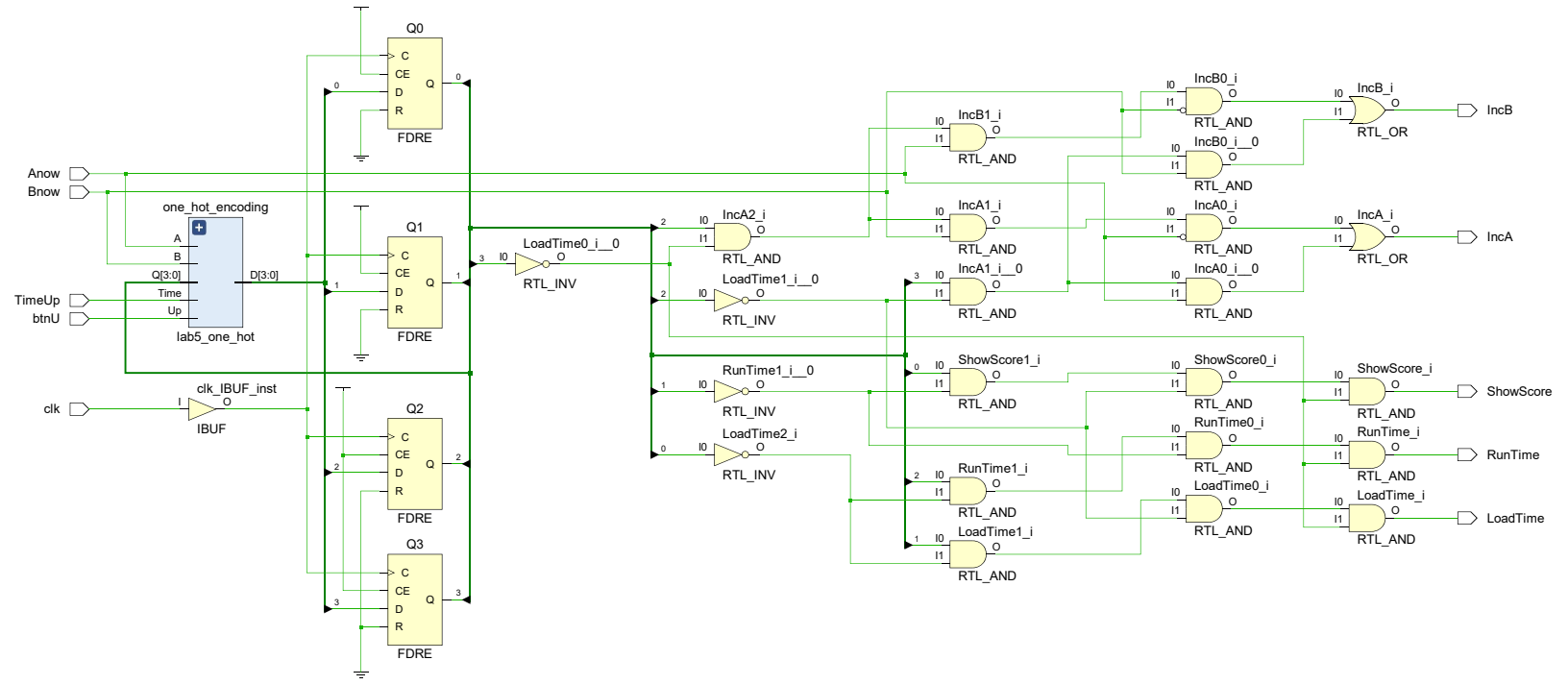


```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 05/10/2021 12:14:27 AM
// Design Name:
// Module Name: lab5_one_hot
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module lab5_one_hot(
    input [3:0] Q,
    input A,
    input B,
    input Up,
    input Time,
    output [3:0] D
    );

    assign D[0] = (Q[0] & ~Up) | (Q[2] & (A | B)) | (Q[3] & (A | B)) | (Q[0] & Up &
(A | B)) | (Q[1] & Up & (A | B));
    assign D[1] = (Q[0] & Up & ~A & ~B) | (Q[1] & Up & ~A & ~B);
    assign D[2] = (Q[1] & ~Up) | (Q[2] & ~Q[0] & (~A & ~B) & ~Time);
    assign D[3] = (Q[2] & Time) | (Q[3] & (~A & ~B));
endmodule
```


Up
Q[3:0]
Time
A
B
D[3:0]
D2_i
RTL_INV
D4_i__0
RTL_AND
D4_i
RTL_AND
D5_i
RTL_OR
D3_i__3
RTL_INV
D3_i__4
RTL_INV
D3_i__0
RTL_AND
D3_i__1
RTL_AND
D3_i__2
RTL_AND
D3_i
RTL_OR
D3_i__5
RTL_AND
D2_i__4
RTL_AND
D2_i__0
RTL_OR
D2_i__1
RTL_AND
D2_i__2
RTL_AND
D2_i__3
RTL_AND
D2_i__5
RTL_AND
D1_i
RTL_OR
D1_i__0
RTL_AND
D1_i__1
RTL_AND
D1_i__2
RTL_AND
D1_i__3
RTL_AND
D1_i__4
RTL_AND
D1_i__5
RTL_AND
D1_i__6
RTL_AND
D0_i
RTL_OR
D0_i__0
RTL_OR
D0_i__1
RTL_OR
D0_i__2
RTL_OR

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 05/09/2021 11:55:16 PM
// Design Name:
// Module Name: lab5_state_machine
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module lab5_state_machine(
    input btnU,
    input TimeUp,
    input Anow,
    input Bnow,
    input clk,
    output LoadTime,
    output RunTime,
    output IncA,
    output IncB,
    output ShowScore
    );
    wire [3:0] Q;
    wire [3:0] D;

    //assign D[0] = (Q[0] & ~btnU) | (Q[2] & (Anow | Bnow)) | (Q[3] & (Anow | Bnow));
    //assign D[1] = (Q[0] & btnU) | (Q[1] & btnU);
    //assign D[2] = (Q[1] & ~btnU) | (Q[2] & ~TimeUp) | (Q[2] & (~Anow & ~Bnow));
    //assign D[3] = (Q[2] & TimeUp) | (Q[3] & (~Anow & ~Bnow));

    lab5_one_hot one_hot_encoding (.Q(Q[3:0]), .A(Anow), .B(Bnow), .Up(btnU),
.Time(TimeUp), .D(D[3:0]));

    FDRE #(.INIT(1'b1) ) Q0 (.C(clk), .CE(1'b1), .D(D[0]), .Q(Q[0]));
```

```
    FDRE #(.INIT(1'b0) ) Q1 (.C(clk), .CE(1'b1), .D(D[1]), .Q(Q[1]));
    FDRE #(.INIT(1'b0) ) Q2 (.C(clk), .CE(1'b1), .D(D[2]), .Q(Q[2]));
    FDRE #(.INIT(1'b0) ) Q3 (.C(clk), .CE(1'b1), .D(D[3]), .Q(Q[3]));

    //FDRE #(.INIT(1'b0) ) Load (.C(clk), .CE(1'b1), .D(Q[1] & ~Q[0] & ~Q[2] &
~Q[3]), .Q(LoadTime));
    //FDRE #(.INIT(1'b0) ) Run (.C(clk), .CE(1'b1), .D(Q[2] & ~Q[0] & ~Q[1] &
~Q[3]), .Q(RunTime));
    //FDRE #(.INIT(1'b0) ) incA (.C(clk), .CE(1'b1), .D((Q[2] & ~Q[3] & Bnow &
~Anow) | (Q[3] & ~Q[2] & Anow)), .Q(IncA));
    //FDRE #(.INIT(1'b0) ) incB (.C(clk), .CE(1'b1), .D((Q[2] & ~Q[3] & Anow &
~Bnow) | (Q[3] & ~Q[2] & Bnow)), .Q(IncB));
    //FDRE #(.INIT(1'b0) ) score (.C(clk), .CE(1'b1), .D(Q[0] & ~Q[1] & ~Q[2] &
~Q[3]), .Q(ShowScore));
    assign LoadTime = Q[1] & ~Q[0] & ~Q[2] & ~Q[3];
    assign RunTime = Q[2] & ~Q[0] & ~Q[1] & ~Q[3];
    assign IncA = (Q[2] & ~Q[3] & Bnow & ~Anow) | (Q[3] & ~Q[2] & Anow);
    assign IncB = (Q[2] & ~Q[3] & Anow & ~Bnow) | (Q[3] & ~Q[2] & Bnow);
    assign ShowScore = Q[0] & ~Q[1] & ~Q[2] & ~Q[3];

endmodule
```


Anow
Bnow
TimeUp
btnU
clk
one_hot_encoding
A
B
Q[3:0]
Time
Up
D[3:0]
lab5_one_hot
clk_IBUF_inst
I
O
IBUF
Q0
Q1
Q2
Q3
C
CE
D
R
Q
FDRE
LoadTime0_i__0
RTL_INV
IncA2_i
LoadTime1_i__0
RunTime1_i__0
LoadTime2_i
IncB1_i
IncA1_i
IncA1_i__0
ShowScore1_i
RunTime1_i
LoadTime1_i
IncB0_i
IncB0_i__0
IncA0_i
IncA0_i__0
ShowScore0_i
RunTime0_i
LoadTime0_i
IncB_i
IncA_i
ShowScore_i
RunTime_i
LoadTime_i
I0
I1
O
RTL_AND
RTL_OR
IncB
IncA
ShowScore
RunTime
LoadTime

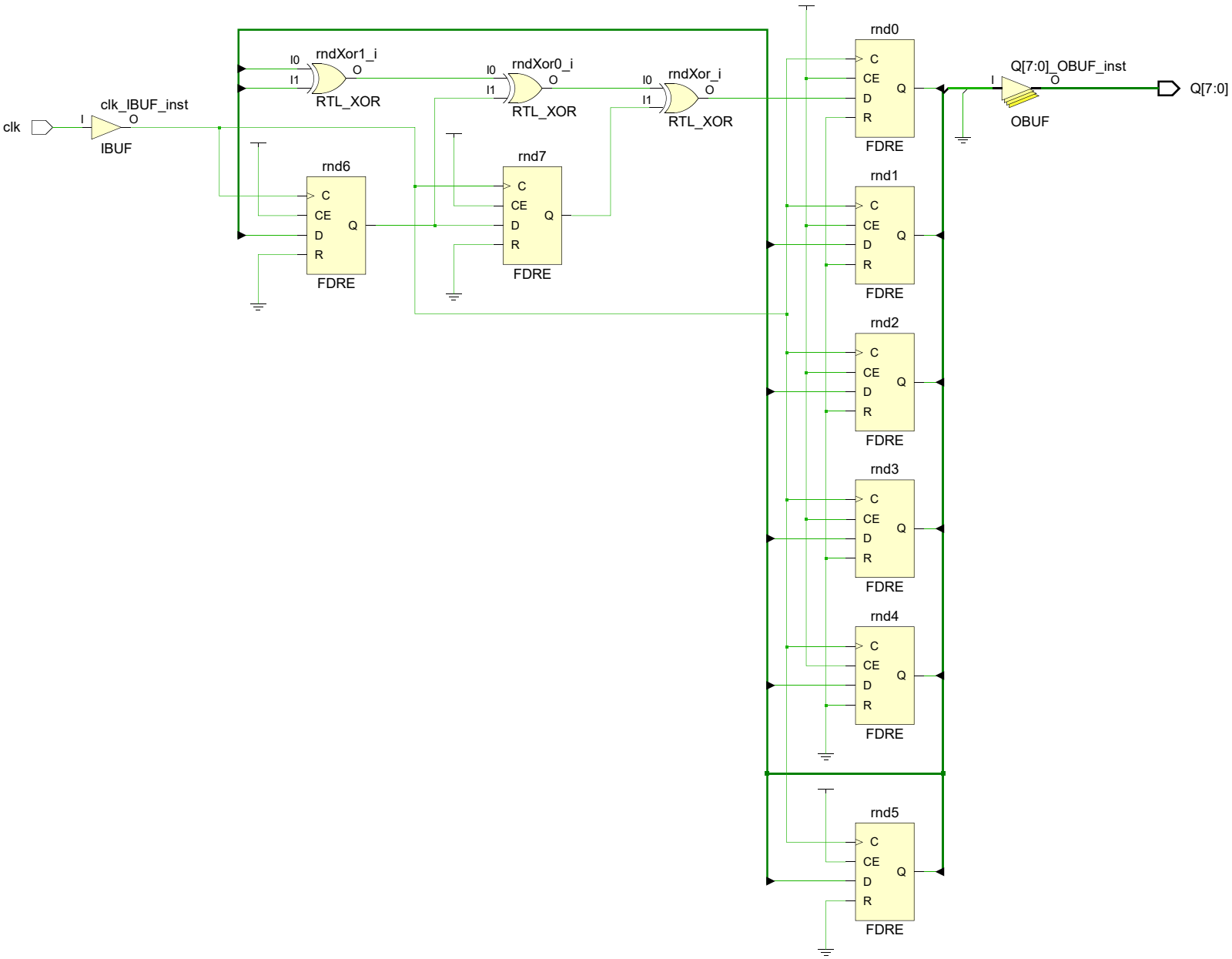
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 05/10/2021 02:03:17 PM
// Design Name:
// Module Name: lab5_fsm_simulation
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module lab5_fsm_simulation();
    reg clkin;
    reg Up = 1'b0;
    reg DTC = 1'b0;
    reg A = 1'b0;
    reg B = 1'b0;
    wire Load, Run, IncA, IncB, Score;
    lab5_state_machine FSM (.btnU(Up), .TimeUp(DTC), .Anow(A), .Bnow(B),
.clk(clkin), .LoadTime(Load), .RunTime(Run), .IncA(IncA), .IncB(IncB),
.ShowScore(Score));



    parameter PERIOD = 10;
    parameter real DUTY_CYCLE = 0.5;
    parameter OFFSET = 2;

    initial    // Clock process for clkin
    begin
        #OFFSET
          clkin = 1'b1;
        forever
        begin
            #(PERIOD-(PERIOD*DUTY_CYCLE)) clkin = ~clkin;
```

```
      end
    end

  initial
  begin
    // add your stimuli here
    // to set signal foo to value 0 use
    // foo = 1'b0;
    // to set signal foo to value 1 use
    // foo = 1'b1;
    //always advance time my multiples of 100ns
    // to advance time by 100ns use the following line
    #1000; //Initial. State 0, new game/game over.
    Up = 1'b1;
    #200; //Should be in state 1 now, telling counter to load.
    Up = 1'b0;
    #200; //Should be in state 2 now, counting down.
    DTC = 1'b1;
    #200; //Should be in state 3 now, game winning possible.
    A = 1'b1;
    #200; //A scored, return to state 0.
    DTC = 1'b0;
    A = 1'b0;
    B = 1'b1;
    Up = 1'b1;
    #200; //A down, B up, btnU up, so it should stay in the same state.
    B = 1'b0;
    #200; //Back to state 1.
    A = 1'b1;
    B = 1'b1;
    #200; //Switches while btnU is pressed! Should return to state 0 since game hasn
technically started yet!
    A = 1'b0;
    B = 1'b0;
    #200; //Alright, now the game should start!
    Up = 1'b0;
    #200; //Entering state 2.
    A = 1'b1;
    #200; //A went high too early! Point should go to B!
    A = 1'b0;
    #200; //We're back in state 0.
    Up = 1'b1;
    #200; //State 1.
    Up = 1'b0;
    #200; //State 2.
    A = 1'b1;
```

```
    B = 1'b1;
    #200; //Uh oh, they both went too early, guess that means they both lose!
    A = 1'b0;
    B = 1'b0;
    #200;
    Up = 1'b1;
    #200; //State 1.
    Up = 1'b0;
    #200; //State 2;
    DTC = 1'b1;
    #200; //State 3, been a while since we've been here.
    A = 1'b1;
    B = 1'b1;
    #200; //Both scored at the same time, should be a win for both of them.
    A = 1'b0;
    B = 1'b0;
    DTC = 1'b0;
   end
endmodule
```

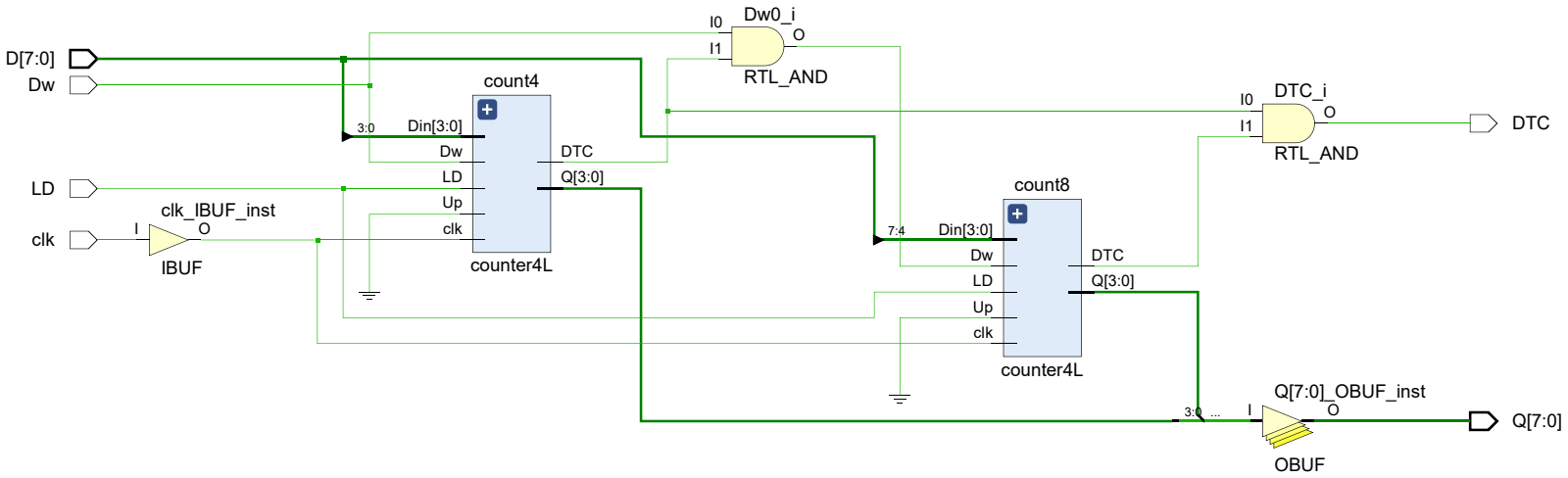
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 05/10/2021 01:00:52 PM
// Design Name:
// Module Name: LFSR
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module LFSR(
    input clk,
    output [7:0] Q
    );
    wire [7:0] rnd;
    wire rndXor;
    assign rndXor = rnd[0] ^ rnd[5] ^ rnd[6] ^ rnd[7];

    FDRE #(.INIT(1'b1) ) rnd0 (.C(clk), .CE(1'b1), .D(rndXor), .Q(rnd[0]));
    FDRE #(.INIT(1'b0) ) rnd1 (.C(clk), .CE(1'b1), .D(rnd[0]), .Q(rnd[1]));
    FDRE #(.INIT(1'b0) ) rnd2 (.C(clk), .CE(1'b1), .D(rnd[1]), .Q(rnd[2]));
    FDRE #(.INIT(1'b0) ) rnd3 (.C(clk), .CE(1'b1), .D(rnd[2]), .Q(rnd[3]));
    FDRE #(.INIT(1'b0) ) rnd4 (.C(clk), .CE(1'b1), .D(rnd[3]), .Q(rnd[4]));
    FDRE #(.INIT(1'b0) ) rnd5 (.C(clk), .CE(1'b1), .D(rnd[4]), .Q(rnd[5]));
    FDRE #(.INIT(1'b0) ) rnd6 (.C(clk), .CE(1'b1), .D(rnd[5]), .Q(rnd[6]));
    FDRE #(.INIT(1'b0) ) rnd7 (.C(clk), .CE(1'b1), .D(rnd[6]), .Q(rnd[7]));

    assign Q[0] = rnd[0];
    assign Q[1] = rnd[1];
    assign Q[2] = rnd[2];
    assign Q[3] = rnd[3];
    assign Q[4] = rnd[4];
    assign Q[5] = rnd[5];
    assign Q[6] = 1'b0;
```

```
    assign Q[7] = 1'b0;
endmodule
```


clk
clk_IBUF_inst
I
O
IBUF
rndXor1_i
I0
I1
O
RTL_XOR
rndXor0_i
I0
I1
O
RTL_XOR
rndXor_i
I0
I1
O
RTL_XOR
rnd6
C
CE
D
R
Q
FDRE
rnd7
C
CE
D
R
Q
FDRE
rnd0
C
CE
D
R
Q
FDRE
rnd1
C
CE
D
R
Q
FDRE
rnd2
C
CE
D
R
Q
FDRE
rnd3
C
CE
D
R
Q
FDRE
rnd4
C
CE
D
R
Q
FDRE
rnd5
C
CE
D
R
Q
FDRE
Q[7:0]_OBUF_inst
I
O
OBUF
Q[7:0]

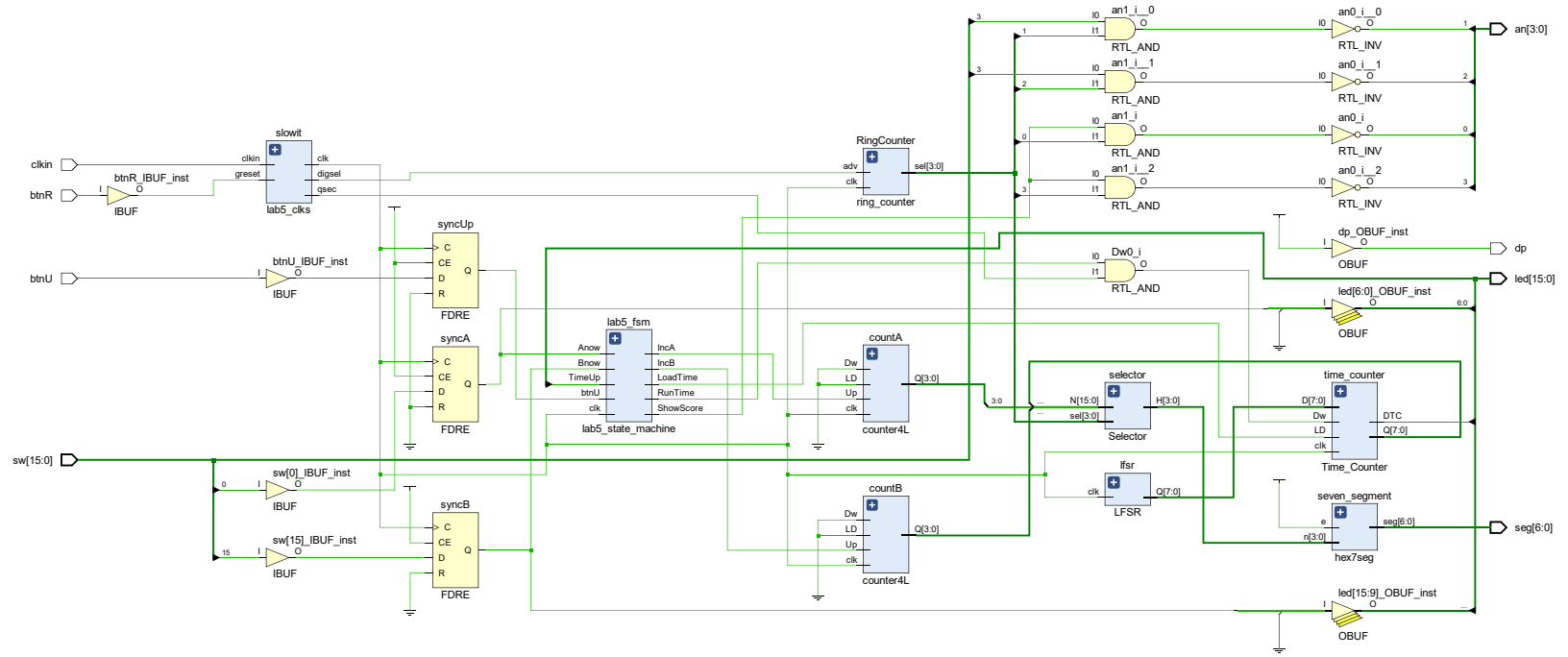
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04/25/2021 09:01:51 PM
// Design Name:
// Module Name: countUD16L
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Time_Counter(
    input clk,
    //input Up,
    input Dw,
    input [7:0] D,
    output [7:0] Q,
    //output UTC,
    output DTC,
    input LD
    );
    wire UTC0, UTC1;//, UTC2, UTC3;
    wire DTC0, DTC1;//, DTC2, DTC3;
    //wire [15:0] R;

    //m2_1x4 sel4(.in0(Q[3:0]), .in1(D[3:0]), .sel(LD), .o(R[3:0]));
    //m2_1x4 sel8(.in0(Q[7:4]), .in1(D[7:4]), .sel(LD), .o(R[7:4]));
    //m2_1x4 sel12(.in0(Q[11:8]), .in1(D[11:8]), .sel(LD), .o(R[11:8]));
    //m2_1x4 sel16(.in0(Q[15:12]), .in1(D[15:12]), .sel(LD), .o(R[15:12]));

    counter4L count4(.clk(clk), .Up(1'b0), .Dw(Dw), .LD(LD), .Din(D[3:0]),
.Q(Q[3:0]), .UTC(UTC0), .DTC(DTC0));
    counter4L count8(.clk(clk), .Up(1'b0), .Dw(Dw & DTC0), .LD(LD), .Din(D[7:4]),
.Q(Q[7:4]), .UTC(UTC1), .DTC(DTC1));
    //countUD4L count12(.clk(clk), .Up(Up & ~Dw & UTC1 & UTC0), .Dw(Dw & ~Up & DTC1
```

```
& DTC0), .LD(LD), .Din(D[11:8]), .Q(Q[11:8]), .UTC(UTC2), .DTC(DTC2));
    //countUD4L count16(.clk(clk), .Up(Up & ~Dw & UTC2 & UTC1 & UTC0), .Dw(Dw & ~Up
& DTC2 & DTC1 & DTC0), .LD(LD), .Din(D[15:12]), .Q(Q[15:12]), .UTC(UTC3), .DTC(DTC3))

    //assign UTC = UTC0 & UTC1 & UTC2 & UTC3;
    assign DTC = DTC0 & DTC1;

endmodule
```


D[7:0]
Dw
LD
clk
clk_IBUF_inst
I
O
IBUF
count4
3:0
Din[3:0]
Dw
LD
Up
clk
DTC
Q[3:0]
counter4L
I0
I1
Dw0_i
O
RTL_AND
count8
7:4
Din[3:0]
Dw
LD
Up
clk
DTC
Q[3:0]
counter4L
I0
I1
DTC_i
O
RTL_AND
DTC
3:0 ...
I
Q[7:0]_OBUF_inst
O
OBUF
Q[7:0]

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 05/09/2021 11:51:40 PM
// Design Name:
// Module Name: lab5_top
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module lab5_top(
    input btnU,
    input [15:0] sw,
    input clkin,
    input btnR,
    output [15:0] led,
    output [3:0] an,
    output dp,
    output [6:0] seg
    );
    wire Up, swA, swB, DTC, IncA, IncB, Load, Run;
    wire digsel, qsec, clk;
    wire Score;
    wire [15:0] Q;
    wire [7:0] Din;
    wire [3:0] sel;
    wire [3:0] n;
    //wire UpA, UpB;
    lab5_clks slowit (.clkin(clkin), .greset(btnR), .clk(clk), .digsel(digsel),
.qsec(qsec));


    FDRE #(.INIT(1'b0) ) syncUp (.C(clk), .CE(1'b1), .D(btnU), .Q(Up));
    FDRE #(.INIT(1'b0) ) syncA (.C(clk), .CE(1'b1), .D(sw[0]), .Q(swA));
```

```
    FDRE #(.INIT(1'b0) ) syncB (.C(clk), .CE(1'b1), .D(sw[15]), .Q(swB));

    lab5_state_machine lab5_fsm (.btnU(Up), .TimeUp(DTC), .Anow(swA), .Bnow(swB),
.clk(clk), .LoadTime(Load), .RunTime(Run), .IncA(IncA), .IncB(IncB),
.ShowScore(Score));

    LFSR lfsr (.clk(clk), .Q(Din));
    Time_Counter time_counter (.clk(clk), .Dw(Run & qsec), .LD(Load), .D(Din),
.DTC(DTC), .Q(Q[11:4]));


    //FDRE #(.INIT(1'b0) ) syncIncA (.C(clk), .CE(1'b1), .D(IncA), .Q(UpA));
    //FDRE #(.INIT(1'b0) ) syncIncB (.C(clk), .CE(1'b1), .D(IncB), .Q(UpB));

    counter4L countA (.clk(clk), .LD(1'b0), .Up(IncA), .Dw(1'b0), .Q(Q[3:0]));
    counter4L countB (.clk(clk), .LD(1'b0), .Up(IncB), .Dw(1'b0), .Q(Q[15:12]));

    ring_counter RingCounter (.clk(clk), .adv(digsel), .sel(sel));
    Selector selector (.sel(sel), .N(Q), .H(n));
    hex7seg seven_segment (.n(n), .e(1'b1), .seg(seg));
    //FDRE #(.INIT(1'b0) ) ff_instance_1095 (.C(clk), .R(reset), .CE(enable),
.D(inD), .Q(outQ));


    //Leds and Switches
    assign an[0] = ~(Score & sel[0]);
    assign an[1] = ~(sw[3] & sel[1]);
    assign an[2] = ~(sw[3] & sel[2]);
    assign an[3] = ~(Score & sel[3]);
    assign led[0] = swA;
    assign led[15] = swB;
    assign led[7] = DTC;
    assign led[8] = DTC;
    assign led[6:1] = 1'b0;
    assign led[14:9] = 1'b0;
    assign dp = 1'b1;
endmodule
```


clkin
btnR
btnU
sw[15:0]
btnR_IBUF_inst
IBUF
slowit
clkin
greset
clk
digsel
qsec
lab5_clks
btnU_IBUF_inst
IBUF
syncUp
C
CE
D
R
Q
FDRE
syncA
FDRE
lab5_fsm
Anow
Bnow
TimeUp
btnU
clk
IncA
IncB
LoadTime
RunTime
ShowScore
lab5_state_machine
sw[0]_IBUF_inst
IBUF
sw[15]_IBUF_inst
IBUF
syncB
FDRE
RingCounter
adv
clk
sel[3:0]
ring_counter
countA
Dw
LD
Up
clk
Q[3:0]
counter4L
countB
counter4L
an1_i__0
an1_i__1
an1_i
an1_i__2
I0
I1
O
RTL_AND
an0_i__0
an0_i__1
an0_i
an0_i__2
RTL_INV
an[3:0]
Dw0_i
RTL_AND
dp_OBUF_inst
OBUF
dp
led[6:0]_OBUF_inst
OBUF
led[15:0]
selector
N[15:0]
sel[3:0]
H[3:0]
Selector
lfsr
clk
Q[7:0]
LFSR
time_counter
D[7:0]
Dw
LD
clk
DTC
Q[7:0]
Time_Counter
seven_segment
e
n[3:0]
seg[6:0]
hex7seg
seg[6:0]
led[15:9]_OBUF_inst
OBUF

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 05/10/2021 02:03:17 PM
// Design Name:
// Module Name: lab5_simulation
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module lab5_simulation();
    reg Up = 1'b0;
    reg btnR = 1'b0;
    reg clkin;
    reg [15:0] sw;
    wire [15:0] led;
    wire [3:0] an;
    wire [6:0] seg;
    wire dp;

    lab5_top UUT (.btnU(Up), .sw(sw), .clkin(clkin), .btnR(btnR), .led(led),
.an(an), .dp(dp), .seg(seg));

    //wire [7:0] D7Seg3,D7Seg2,D7Seg1,D7Seg0; // Change the Radix of these signals
to ASCII
    //show_7segDisplay  showit (.seg(seg),.dp(dp),.an(an),
    // .D7Seg0(D7Seg0),.D7Seg1(D7Seg1),.D7Seg2(D7Seg2),.D7Seg3(D7Seg3));

    parameter PERIOD = 10;
    parameter real DUTY_CYCLE = 0.5;
    parameter OFFSET = 2;

    initial    // Clock process for clkin
    begin
        #OFFSET
          clkin = 1'b1;
```

```
    forever
    begin
        #(PERIOD-(PERIOD*DUTY_CYCLE)) clkin = ~clkin;
    end
 end

initial
begin
  // add your stimuli here
  // to set signal foo to value 0 use
  // foo = 1'b0;
  // to set signal foo to value 1 use
  // foo = 1'b1;
  //always advance time my multiples of 100ns
  // to advance time by 100ns use the following line
  sw = 16'b0000000000000000;
  #1200; //Initial. State 0, new game/game over.
  Up = 1'b1;
  #200; //State 1, something should be loaded into the counter.
  Up = 1'b0;
  #200; //State 2, should be counting down now.
  #600;
  sw[0] = 1'b1;
  #200; //A flips too early, causing B to score!
  sw[0] = 1'b0;
  #200;
  Up = 1'b1;
  #200;
  Up = 1'b0;
  #200;
  sw[15] = 1'b1;
  #200; //B flips too early, causing A to score!
  sw[15] = 1'b0;
  #200;
  Up = 1'b1;
  #200;
  Up = 1'b0;
  #200;
  sw[0] = 1'b1;
  sw[15] = 1'b1;
  #200; //A and B flip too early, causing neither of them to score!
  sw[0] = 1'b0;
  sw[15] = 1'b0;
  #200;
  Up = 1'b1;
  #200;
```

```
    Up = 1'b0;
    #100000;
    sw[0] = 1'b1;
    #200; //A scores!
    sw[0] = 1'b0;
    #200;
    Up = 1'b1;
    #200;
    Up = 1'b0;
    #100000;
    sw[15] = 1'b1;
    #200; //B scores!
    sw[15] = 1'b0;
    #200;
    Up = 1'b1;
    #200;
    Up = 1'b0;
    #100000;
    sw[0] = 1'b1;
    sw[15] = 1'b1;
    #200; //A and B both score!
    sw[0] = 1'b0;
    sw[15] = 1'b0;
    //sw[0] = 1'b1;
    #200; //A scores!
   end


endmodule
```