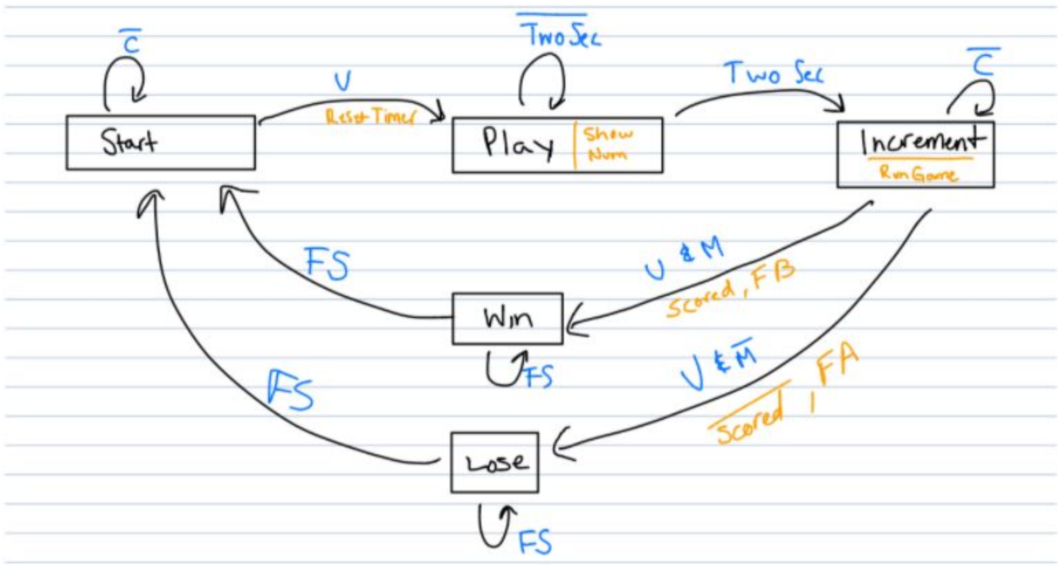
Scott Fischer
November 16, 2020
Lab 5
Section 1D

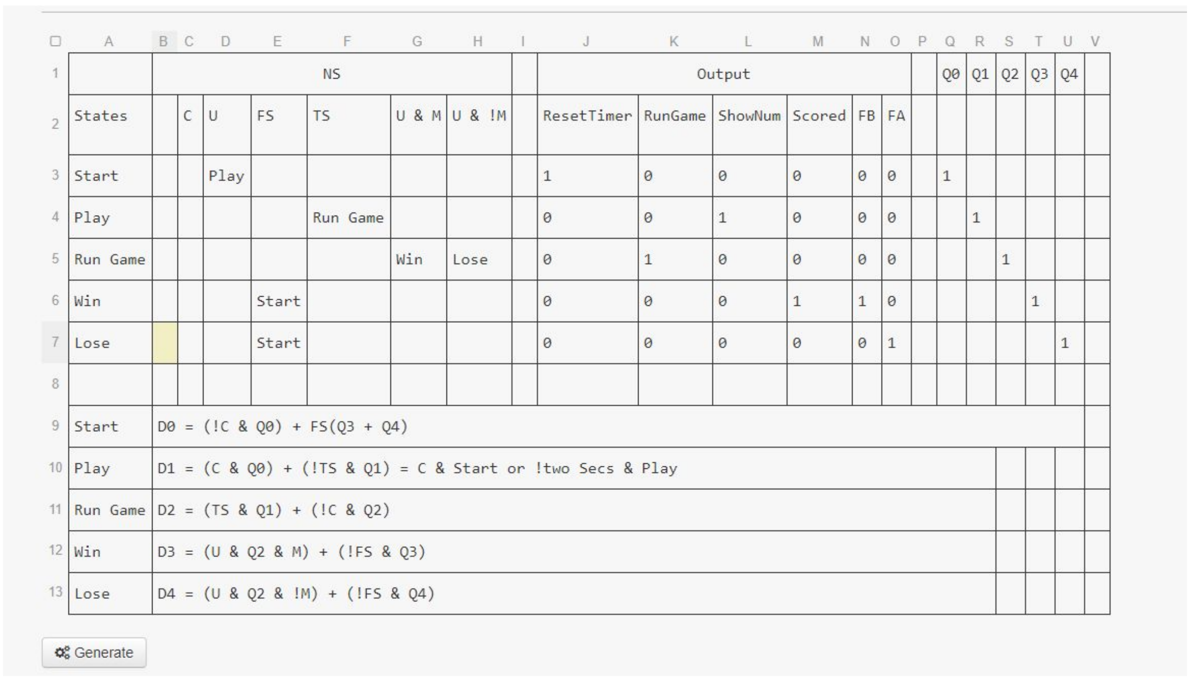
# Lab 5 Writeup

Introduction: In this lab we were tasked with creating a game we could play on the Baysus board. The game works by starting once button C is pressed a random number is generated and displayed on the left two bits of the display. After two seconds a counter begins counting up which is displayed on the two rightmost bits of the display. The game counter goes from 0 to 3F in hex and then rolls over infinitely until button U is pressed. If the number when button U is pressed matched with the random number then both numbers flash simultaneously. If they do not match then they flash in an alternating pattern. The flashing lasts for four seconds, and then button C needs to be pressed to start another round. Each win makes one more led light up.

For this lab instead of starting at the lowest level I started on the most central part of the lab, the state machine. I started by creating a state transition diagram which took a couple of times to conceptually grasp and translate to paper.



Once I had this I used it to create a state transition table that I could then use to turn into the logic that I would need to code my state machine module in Verilog. I used one hot encoding to create the states and the table.



| | NS | | | | | | Output | | | | | | Q0 | Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| States | C | U | FS | TS | U & M | U & !M | ResetTimer | RunGame | ShowNum | Scored | FB | FA | | | | | |
| Start | | Play | | | | | 1 | 0 | 0 | 0 | 0 | 0 | 1 | | | | |
| Play | | | | Run Game | | | 0 | 0 | 1 | 0 | 0 | 0 | | 1 | | | |
| Run Game | | | | | Win | Lose | 0 | 1 | 0 | 0 | 0 | 0 | | | 1 | | |
| Win | | | Start | | | | 0 | 0 | 0 | 1 | 1 | 0 | | | | 1 | |
| Lose | | | Start | | | | 0 | 0 | 0 | 0 | 0 | 1 | | | | | 1 |
| | | | | | | | | | | | | | | | | | |
| Start | D0 = (!C & Q0) + FS(Q3 + Q4) | | | | | | | | | | | | | | | | |
| Play | D1 = (C & Q0) + (!TS & Q1) = C & Start or !two Secs & Play | | | | | | | | | | | | | | | | |
| Run Game | D2 = (TS & Q1) + (!C & Q2) | | | | | | | | | | | | | | | | |
| Win | D3 = (U & Q2 & M) + (!FS & Q3) | | | | | | | | | | | | | | | | |
| Lose | D4 = (U & Q2 & !M) + (!FS & Q4) | | | | | | | | | | | | | | | | |

I used this to create a module with 5 D flip flops that each represented one state of the state machine. So if you are in the start state Q0 would be high, and all other Q values would be low. This made it easy to create the logic for both the inputs of each of the flip flops and the outputs of the module as whole. Once I had this all coded in Verilog I created a testbench and made sure that each state would transition properly given the correct inputs, and that each output was correct given the state that it was in.

After I was happy with my state machine I created both the time counter and the game counter. I copied the logic from lab4 in order to create counter modules. For the game counter I simply created a 6 bit up counter and hard coded the last two bits so that my counter would only go up to 3F in hex and then roll back over. For the time counter I kept it as an 8 bit counter same as lab4. For the random number generator I copied the diagram from the lab manual and simply coded it into verilog. For the LED shifter I created a counter then changed it so that when each flip flop became high it would stay high. After that I simply copied all the hex7seg, ringcounter, and selector modules from lab4. After that I created all the logic for the anodes and connected all the modules together in the proper way and tested it through simulations until I was satisfied.

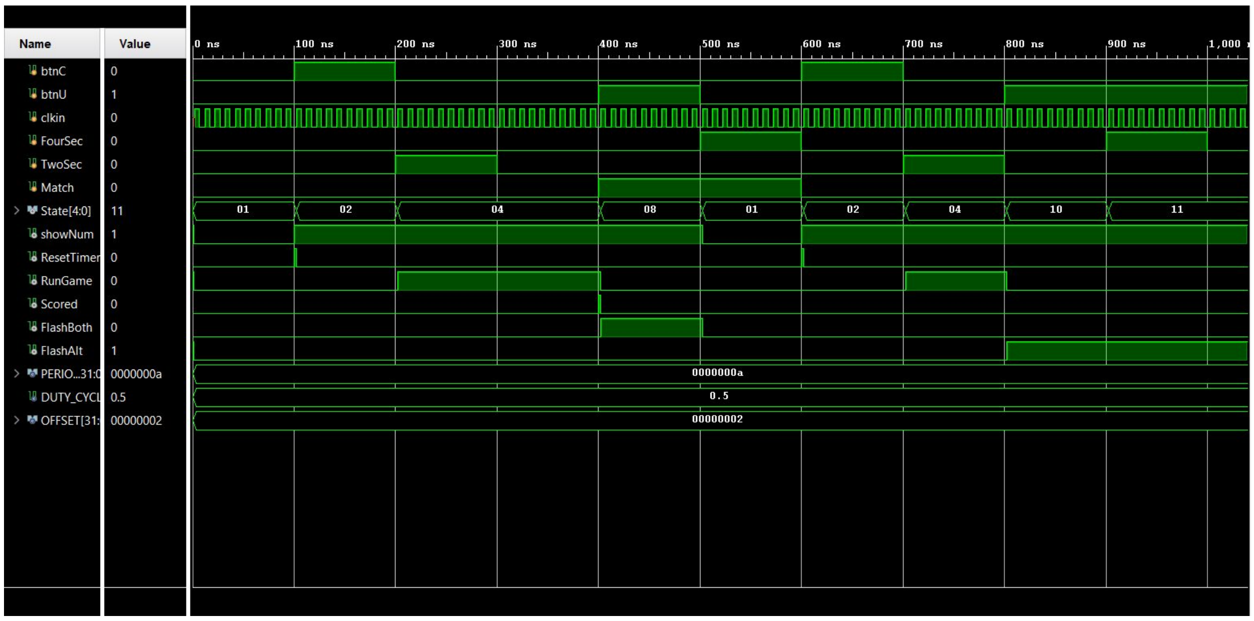
Overall, the lab mostly taught me how to create a state diagram from a concept of inputs and outputs and turn that diagram into logic that can be programmed in Verilog. This seems like it is very useful practice and skill to have improved for logic design.

## Lab Questions:

My Start state is the in initial state the machine starts in, and the one it sits in at the end while it idles. The Play state occurs once you hit the C button and is the state that sets the display to show the random number. This state exists on its own and goes into the Run Game state. The Run Game state is where it idles with the game counter continually running until button U is pushed. Once this happens there is a fork and the state machine either goes to the win or lose state. In these states we either increment the wins or do nothing, but in both states we reset the timers and have the random number not show anymore. After both of these states we return to the Start state and idle.

## Supplementary Materials:

### -Waveform:



This is the printout of my waveform for my state machine that shows how the buttons affect whether you win or lose.

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
///////////////
// Company:
// Engineer:
//
// Create Date: 11/11/2020 03:43:17 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
    );
    assign H[0] = (sel[0] & ~sel[1] & ~sel[2] & ~sel[3] & N[0]) |
(~sel[0] & sel[1] & ~sel[2] & ~sel[3] & N[4])| (~sel[0] & ~sel[1] &
sel[2] & ~sel[3] & N[8]) | (~sel[0] & ~sel[1] & ~sel[2] & sel[3] &
N[12]);
    assign H[1] = (sel[0] & ~sel[1] & ~sel[2] & ~sel[3] & N[1]) |
(~sel[0] & sel[1] & ~sel[2] & ~sel[3] & N[5])| (~sel[0] & ~sel[1] &
sel[2] & ~sel[3] & N[9]) | (~sel[0] & ~sel[1] & ~sel[2] & sel[3] &
N[13]);
```

```
    assign H[2] = (sel[0] & ~sel[1] & ~sel[2] & ~sel[3] & N[2]) |
(~sel[0] & sel[1] & ~sel[2] & ~sel[3] & N[6])| (~sel[0] & ~sel[1] &
sel[2] & ~sel[3] & N[10]) | (~sel[0] & ~sel[1] & ~sel[2] & sel[3] &
N[14]);
    assign H[3] = (sel[0] & ~sel[1] & ~sel[2] & ~sel[3] & N[3]) |
(~sel[0] & sel[1] & ~sel[2] & ~sel[3] & N[7])| (~sel[0] & ~sel[1] &
sel[2] & ~sel[3] & N[11]) | (~sel[0] & ~sel[1] & ~sel[2] & sel[3] &
N[15]);
endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
//////////////
// Company:
// Engineer:
//
// Create Date: 11/11/2020 11:29:04 AM
// Design Name:
// Module Name: state_machine_module
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////
//////////////


module state_machine_module(
    input btnC,
    input btnU,
    input clk,
    input FourSec,
    input TwoSec,
    input Match,

    output showNum,
    output ResetTimer,
    output RunGame,
    output Scored,
    output FlashBoth,
    output FlashAlt,
    output Play,
    output [4:0] State
    );

    wire [4:0] D;
    wire [4:0] Q;
    assign D[0] = (~btnC & Q[0]) | (FourSec & (Q[3] | Q[4]));
    //Start State
    assign D[1] = (btnC & Q[0]) | (~TwoSec & Q[1]);
    //Play State
    assign D[2] = (TwoSec & Q[1]) | (~btnU & Q[2]) | (Q[2] & btnC);
                    //RunGame State
    assign D[3] = (btnU & Match) | (~FourSec & Q[3]);
//Win State
    assign D[4] = (btnU & ~Match) | (~FourSec & Q[4]);
//Lose State



    FDRE #(.INIT(1'b1)) Q0_FF (.C(clk), .CE(1'b1), .D(D[0]),
.Q(Q[0]));
    FDRE #(.INIT(1'b0)) Q1_FF (.C(clk), .CE(1'b1), .D(D[1]),
.Q(Q[1]));
    FDRE #(.INIT(1'b0)) Q2_FF (.C(clk), .CE(1'b1), .D(D[2]),
.Q(Q[2]));
    FDRE #(.INIT(1'b0)) Q3_FF (.C(clk), .CE(1'b1), .D(D[3]),
.Q(Q[3]));
    FDRE #(.INIT(1'b0)) Q4_FF (.C(clk), .CE(1'b1), .D(D[4]),
.Q(Q[4]));
    assign State[4:0] = Q[4:0];
    assign showNum      = (Q[0]&btnC) | Q[1] | Q[2] | Q[3] | Q[4];
     //Shitfing the Number
    assign ResetTimer   = Q[0] & btnC;
    assign RunGame      = Q[2];
    assign Scored       = Match & btnU & Q[2];
    assign FlashBoth    = Q[3] & Match;
    assign FlashAlt     = Q[4] & ~Match;
    assign Play          = Q[0] & btnC;
endmodule
```

```verilog
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
///////////////
// Company:
// Engineer:
//
// Create Date: 11/11/2020 10:24:32 PM
// Design Name:
// Module Name: store_module
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////
///////////////


module store_module(
    input CE,
    input clk,
    input [7:0] in,
    output [7:0] Q

    );
    //wire [7:0] D;
    //assign D[0] = CE & in[0];
    FDRE #(.INIT(1'b0)) Q0_FF (.C(clk), .CE(CE), .D(in[0]),
.Q(Q[0]));
    FDRE #(.INIT(1'b0)) Q1_FF (.C(clk), .CE(CE), .D(in[1]),
.Q(Q[1]));
```

```
    FDRE #(.INIT(1'b0)) Q2_FF (.C(clk), .CE(CE), .D(in[2]), 
.Q(Q[2]));
    FDRE #(.INIT(1'b0)) Q3_FF (.C(clk), .CE(CE), .D(in[3]), 
.Q(Q[3]));
    FDRE #(.INIT(1'b0)) Q4_FF (.C(clk), .CE(CE), .D(in[4]), 
.Q(Q[4]));
    FDRE #(.INIT(1'b0)) Q5_FF (.C(clk), .CE(CE), .D(in[5]), 
.Q(Q[5]));
    FDRE #(.INIT(1'b0)) Q6_FF (.C(clk), .CE(CE), .D(in[6]), 
.Q(Q[6]));
    FDRE #(.INIT(1'b0)) Q7_FF (.C(clk), .CE(CE), .D(in[7]), 
.Q(Q[7]));


endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
///////////////
// Company:
// Engineer:
//
// Create Date: 11/11/2020 03:36:02 PM
// Design Name:
// Module Name: time_counter_module
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////
///////////////


module time_counter_module(
    input clk,
    input R,
    input CE,       //this is set as qsec
    output [7:0] Q
    );
    wire [7:0] D;
    FDRE #(.INIT(1'b0)) Q0_FF (.C(clk), .R(R), .CE(CE), .D(D[0]),
.Q(Q[0]));
    FDRE #(.INIT(1'b0)) Q1_FF (.C(clk), .R(R), .CE(CE), .D(D[1]),
.Q(Q[1]));
    FDRE #(.INIT(1'b0)) Q2_FF (.C(clk), .R(R), .CE(CE), .D(D[2]),
.Q(Q[2]));
```

```
    FDRE #(.INIT(1'b0)) Q3_FF (.C(clk), .R(R), .CE(CE), .D(D[3]),
.Q(Q[3]));
    FDRE #(.INIT(1'b0)) Q4_FF (.C(clk), .R(R), .CE(CE), .D(D[4]),
.Q(Q[4]));
    FDRE #(.INIT(1'b0)) Q5_FF (.C(clk), .R(R), .CE(CE), .D(D[5]),
.Q(Q[5]));
    FDRE #(.INIT(1'b0)) Q6_FF (.C(clk), .R(R), .CE(CE), .D(D[5]),
.Q(Q[6]));
    FDRE #(.INIT(1'b0)) Q7_FF (.C(clk), .R(R), .CE(CE), .D(D[5]),
.Q(Q[7]));
    assign D[0] = Q[0] ^ CE;
    assign D[1] = Q[1] ^ (CE&Q[0]);
    assign D[2] = Q[2] ^ (CE&Q[0]&Q[1]);
    assign D[3] = Q[3] ^ (CE&Q[0]&Q[1]&Q[2]);
    assign D[4] = Q[4] ^ (CE&Q[0]&Q[1]&Q[2]&Q[3]);
    assign D[5] = Q[5] ^ (CE&Q[0]&Q[1]&Q[2]&Q[3]&Q[4]);
    assign D[6] = Q[6] ^ (CE&Q[0]&Q[1]&Q[2]&Q[3]&Q[4]&Q[5]);
    assign D[7] = Q[7] ^ (CE&Q[0]&Q[1]&Q[2]&Q[3]&Q[4]&Q[5]&Q[6]);
    //assign Q[6] = 1'b0;
    //assign Q[7] = 1'b0;
endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
///////////////
// Company:
// Engineer:
//
// Create Date: 11/11/2020 05:09:19 PM
// Design Name:
// Module Name: top_module
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////
///////////////


module top_module(
    input btnC,
    input btnU,
    input btnR,
    input clkin,

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

    );
    wire Flash;
    wire Go,Stop,FourSecs,TwoSecs,Match,Play;
    wire ShowNum,ResetTimer,RunGame,Scored,FlashBoth,FlashAlt;
    wire qsec,clk,digsel;
    wire [7:0] rngNum;
    wire [7:0] rngO;
    wire [7:0] gameNum;
    wire [7:0] timer;
    wire [3:0] ringCount;
    wire [15:0] selectBits;
    wire [3:0] selOut;

    assign selectBits[15:8] = rngNum[7:0];
    assign selectBits[7:0] = gameNum[7:0];

    assign Match = ((rngNum[7:0] & gameNum[7:0]) | (~rngNum[7:0] &
~gameNum[7:0]));


    assign TwoSecs = ~timer[7] & ~timer[6] & ~timer[5] & ~timer[4]
& timer[3] & ~timer[2] & ~timer[1] & ~timer[0];
    assign FourSecs = ~timer[7] & ~timer[6] & ~timer[5] & timer[4]
& ~timer[3] & ~timer[2] & ~timer[1] & ~timer[0];
    assign dp = 1'b1;


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

    state_machine_module states(.clk(clk), .btnC(btnC),
.btnU(btnU), .TwoSec(TwoSecs),.FourSec(FourSecs),
.Match(Match),.Play(Play),
    .showNum(ShowNum), .ResetTimer(ResetTimer), .RunGame(RunGame),
.Scored(Scored), .FlashBoth(FlashBoth), .FlashAlt(FlashAlt)
//,.State(States[4:0])
    );

    rng_module randNum(.clk(clk), .rnd(rngO));
    store_module stores(.clk(clk), .CE(btnC&Play), .Q(rngNum[7:0]),
.in(rngO[7:0]));
    //Counters
    game_counter_module gameCount(.clk(clk),
.CE(RunGame&qsec),.R(ResetTimer), .Q(gameNum));
    time_counter_module timeCount(.clk(clk), .CE(qsec),
.R(ResetTimer), .Q(timer));

    led_shift_module ledMod(.clk(clk), .CE(Scored),.in(1'b1),
.Q(led[15:0]));
    ring_counter_module RCMod(.clk(clk), .Advance(digsel),
.o(ringCount[3:0]));
    selector sel(.sel(ringCount[3:0]), .N(selectBits[15:0]),
.H(selOut));
    hex7seg hex(.n(selOut[3:0]),.seg(seg[6:0]));



    flash_module flash(.clk(clk), .CE(qsec), .Flash(Flash));
    assign an[0] = ~(ringCount[0] & ~ringCount[1]  & ~ringCount[2]
& ~ringCount[3] ) | ((FlashAlt|FlashBoth) & Flash);
    assign an[1] = ~(~ringCount[0] & ringCount[1]  & ~ringCount[2]
& ~ringCount[3] ) | ((FlashAlt|FlashBoth) & Flash);
    assign an[2] = ~(~ringCount[0] & ~ringCount[1] & (ringCount[2]&
ShowNum)  & ~ringCount[3]) | ((FlashBoth & Flash) | (FlashAlt &
~Flash));
    assign an[3] = ~(~ringCount[0] & ~ringCount[1] & ~ringCount[2]
& (ringCount[3]& ShowNum)) | ((FlashBoth & Flash) |  (FlashAlt &
~Flash));
endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
////////////////
// Company:
// Engineer:
//
// Create Date: 11/12/2020 02:06:23 AM
// Design Name:
// Module Name: flash_module
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////
////////////////


module flash_module(
    input CE,
    input clk,
    output Flash
    );
    wire a;
    assign a = ~Flash;
    FDRE #(.INIT(1'b0)) Q0_FF (.C(clk), .CE(CE), .D(a), .Q(Flash));
endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
///////////////
// Company:
// Engineer:
//
// Create Date: 11/02/2020 11:08:30 AM
// Design Name:
// Module Name: countUD4L
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////
///////////////


module game_counter_module(
    //input [7:0] x,
    //input w,
    input clk,
    input R,
    input CE,
    output [7:0] Q
    //,output y
    );

    wire [5:0] D;
    FDRE #(.INIT(1'b0)) Q0_FF (.C(clk), .R(R), .CE(CE), .D(D[0]),
.Q(Q[0]));
```

```
    FDRE #(.INIT(1'b0)) Q1_FF (.C(clk), .R(R), .CE(CE), .D(D[1]),
.Q(Q[1]));
    FDRE #(.INIT(1'b0)) Q2_FF (.C(clk), .R(R), .CE(CE), .D(D[2]),
.Q(Q[2]));
    FDRE #(.INIT(1'b0)) Q3_FF (.C(clk), .R(R), .CE(CE), .D(D[3]),
.Q(Q[3]));
    FDRE #(.INIT(1'b0)) Q4_FF (.C(clk), .R(R), .CE(CE), .D(D[4]),
.Q(Q[4]));
    FDRE #(.INIT(1'b0)) Q5_FF (.C(clk), .R(R), .CE(CE), .D(D[5]),
.Q(Q[5]));
    assign D[0] = Q[0] ^ CE;
    assign D[1] = Q[1] ^ (CE&Q[0]);
    assign D[2] = Q[2] ^ (CE&Q[0]&Q[1]);
    assign D[3] = Q[3] ^ (CE&Q[0]&Q[1]&Q[2]);
    assign D[4] = Q[4] ^ (CE&Q[0]&Q[1]&Q[2]&Q[3]);
    assign D[5] = Q[5] ^ (CE&Q[0]&Q[1]&Q[2]&Q[3]&Q[4]);
    assign Q[7] = 1'b0;
    assign Q[6] = 1'b0;
    //assign y = ((x[7:0] & Q[7:0]) | (~x[7:0] & ~Q[7:0]));
endmodule
```

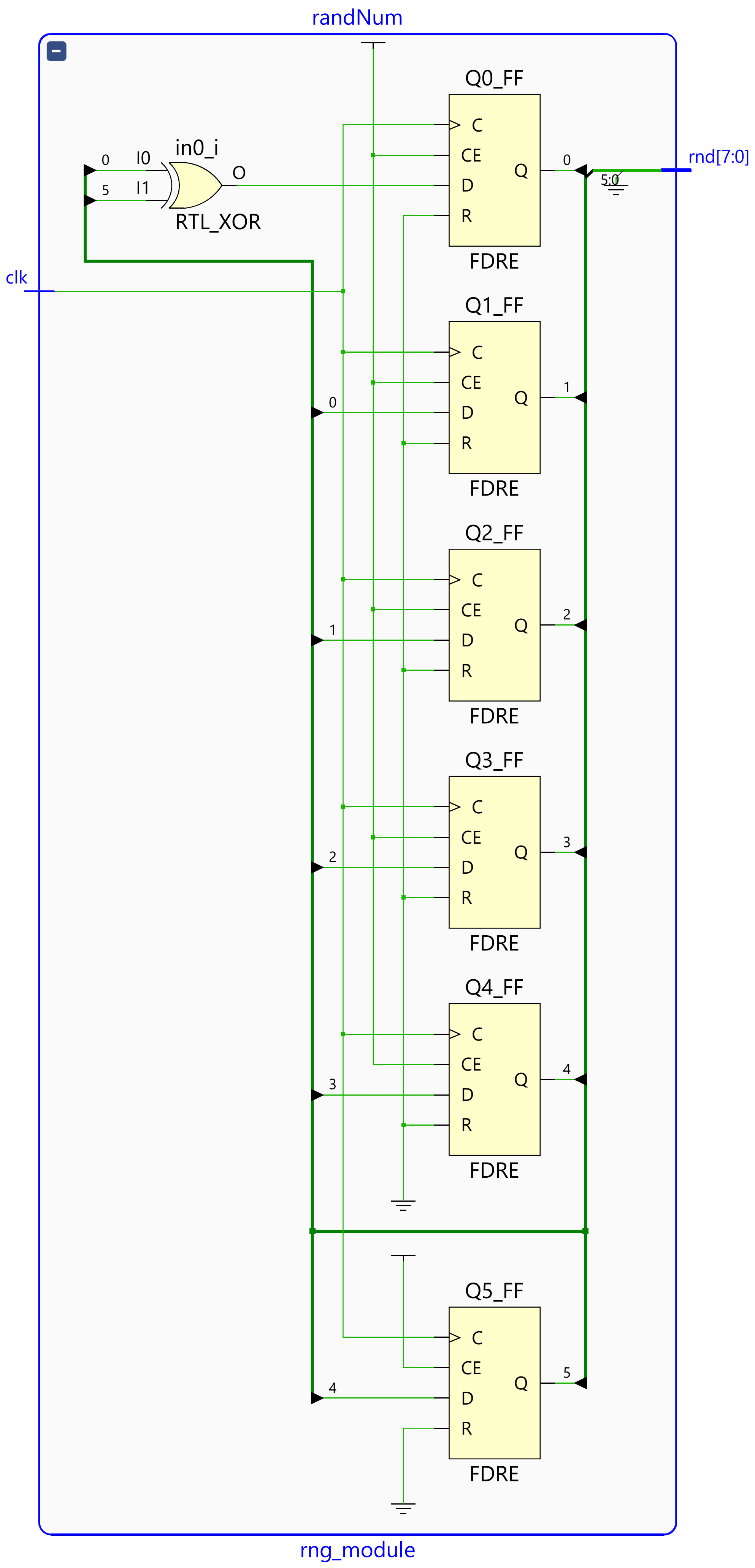
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
////////////////
// Company:
// Engineer:
//
// Create Date: 11/11/2020 03:54:26 PM
// Design Name:
// Module Name: hex7seg
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////
////////////////




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

    m8_1 mSeg0(.in({1'b0,n[0],n[0],1'b0,1'b0,~n[0],1'b0,n[0]}),
.sel({n[3],n[2],n[1]}), .o(seg[0]));             //controls segment
0 for 7 segment display
    m8_1 mSeg1(.in({1'b1,~n[0],n[0],1'b0,~n[0],n[0],1'b0,1'b0}),
.sel({n[3],n[2],n[1]}), .o(seg[1]));            //controls segment
1 for 7 segment display
    m8_1 mSeg2(.in({1'b1,~n[0],1'b0,1'b0,1'b0,1'b0,~n[0],1'b0}),
.sel({n[3],n[2],n[1]}), .o(seg[2]));                   //controls
segment 2 for 7 segment display
    m8_1 mSeg3(.in({n[0],1'b0,~n[0],n[0],n[0],~n[0],1'b0,n[0]}),
.sel({n[3],n[2],n[1]}), .o(seg[3]));        //controls segment 3
for 7 segment display
    m8_1 mSeg4(.in({1'b0,1'b0,1'b0,n[0],n[0],1'b1,n[0],n[0]}),
.sel({n[3],n[2],n[1]}), .o(seg[4]));              //controls
segment 4 for 7 segment display
    m8_1 mSeg5(.in({1'b0,n[0],1'b0,1'b0,n[0],1'b0,1'b1,n[0]}),
.sel({n[3],n[2],n[1]}), .o(seg[5]));                 //controls
segment 5 for 7 segment display
    m8_1 mSeg6(.in({1'b0,~n[0],1'b0,1'b0,n[0],1'b0,1'b0,1'b1}),
.sel({n[3],n[2],n[1]}), .o(seg[6]));                  //controls
segment 6 for 7 segment display

endmodule
```

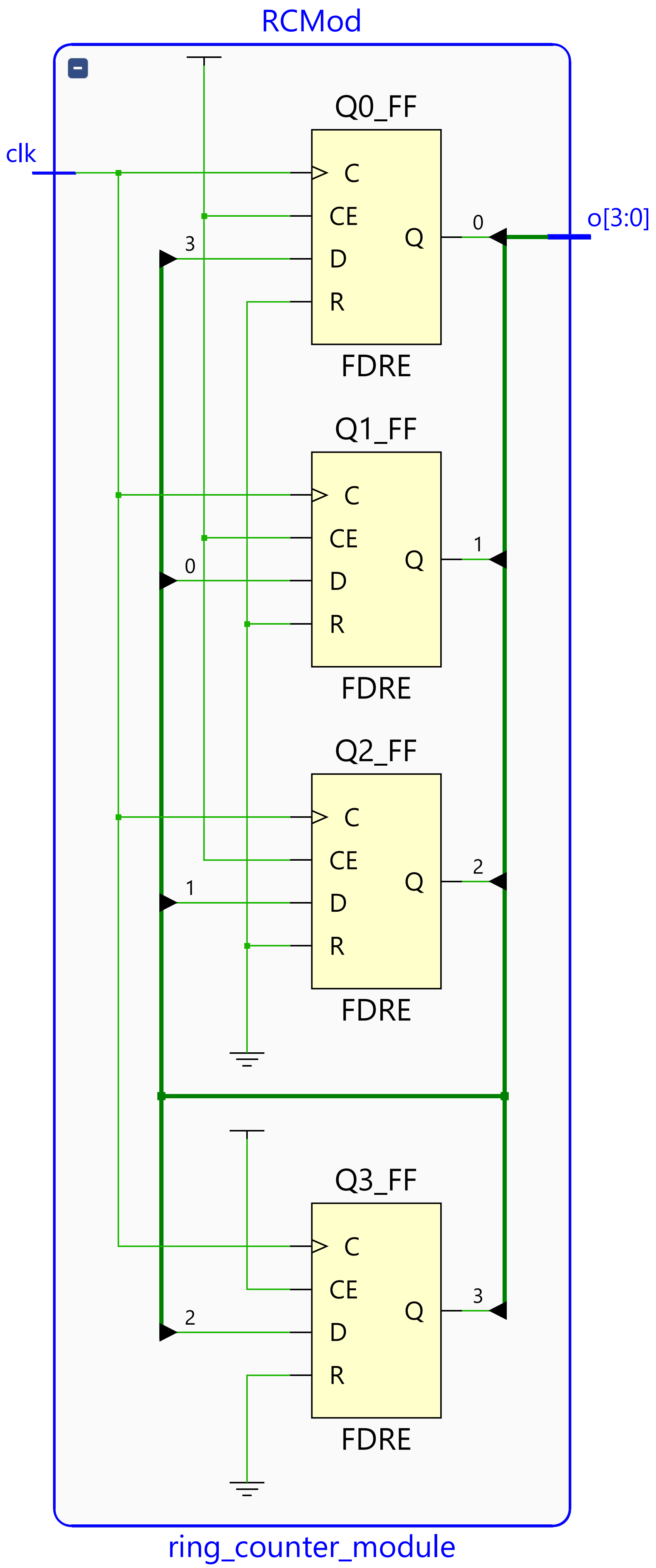
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
////////////////
// Company:
// Engineer:
//
// Create Date: 11/11/2020 03:56:23 PM
// Design Name:
// Module Name: led_shift_module
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////
////////////////


module led_shift_module(
    input clk,
    input in,
    input CE,
    output [15:0] Q
    );
    wire [15:0] out;
    FDRE #(.INIT(1'b0)) Q0_FF (.C(clk), .CE(CE), .D(in),
.Q(out[0]));
    FDRE #(.INIT(1'b0)) Q1_FF (.C(clk), .CE(CE), .D(out[0]),
.Q(out[1]));
    FDRE #(.INIT(1'b0)) Q2_FF (.C(clk), .CE(CE), .D(out[1]),
.Q(out[2]));
```

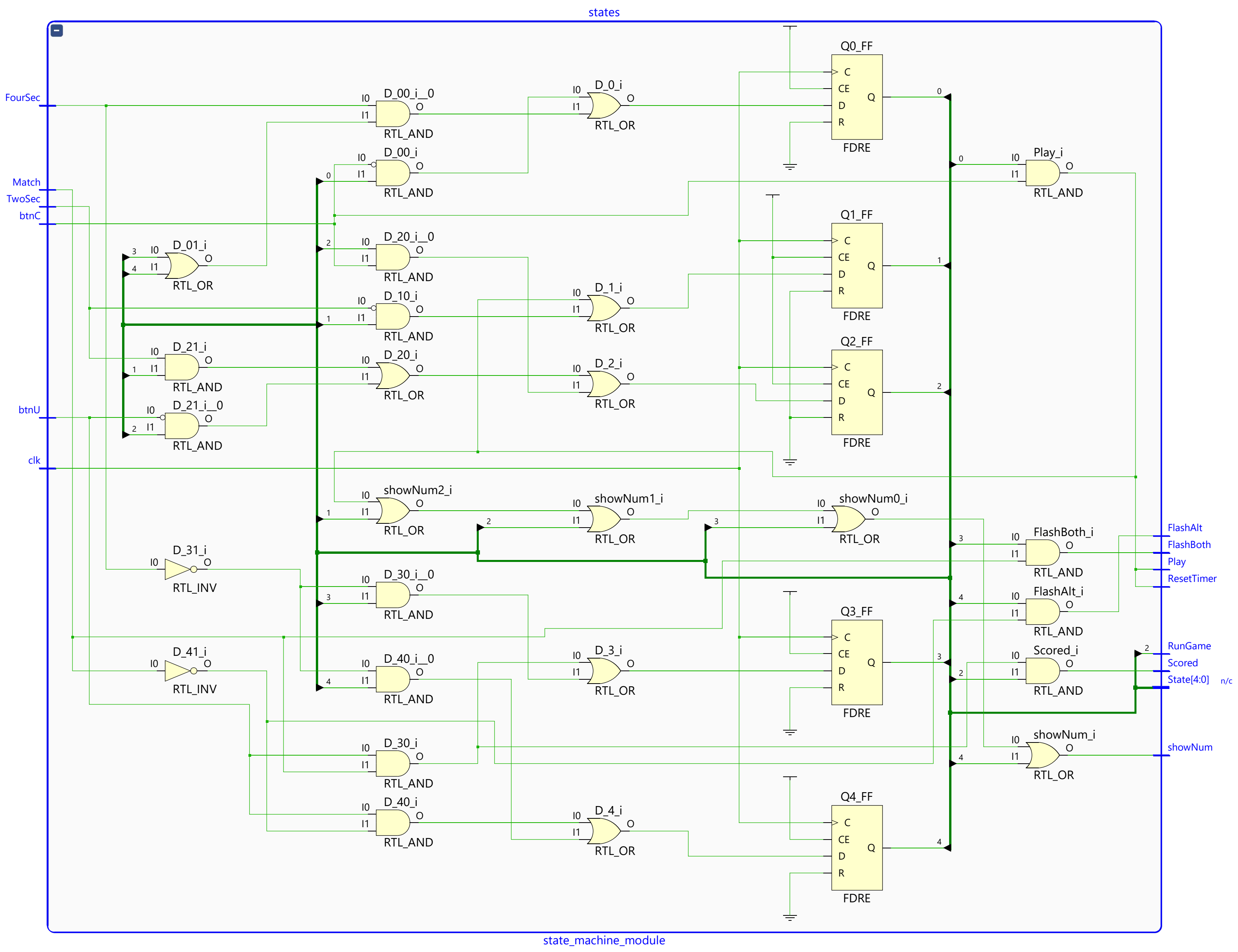
```
    FDRE #(.INIT(1'b0)) Q3_FF (.C(clk), .CE(CE), .D(out[2]),
.Q(out[3]));
    FDRE #(.INIT(1'b0)) Q4_FF (.C(clk), .CE(CE), .D(out[3]),
.Q(out[4]));
    FDRE #(.INIT(1'b0)) Q5_FF (.C(clk), .CE(CE), .D(out[4]),
.Q(out[5]));
    FDRE #(.INIT(1'b0)) Q6_FF (.C(clk), .CE(CE), .D(out[5]),
.Q(out[6]));
    FDRE #(.INIT(1'b0)) Q7_FF (.C(clk), .CE(CE), .D(out[6]),
.Q(out[7]));
    FDRE #(.INIT(1'b0)) Q8_FF (.C(clk), .CE(CE), .D(out[7]),
.Q(out[8]));
    FDRE #(.INIT(1'b0)) Q9_FF (.C(clk), .CE(CE), .D(out[8]),
.Q(out[9]));
    FDRE #(.INIT(1'b0)) Q10_FF (.C(clk), .CE(CE), .D(out[9]),
.Q(out[10]));
    FDRE #(.INIT(1'b0)) Q11_FF (.C(clk), .CE(CE), .D(out[10]),
.Q(out[11]));
    FDRE #(.INIT(1'b0)) Q12_FF (.C(clk), .CE(CE), .D(out[11]),
.Q(out[12]));
    FDRE #(.INIT(1'b0)) Q13_FF (.C(clk), .CE(CE), .D(out[12]),
.Q(out[13]));
    FDRE #(.INIT(1'b0)) Q14_FF (.C(clk), .CE(CE), .D(out[13]),
.Q(out[14]));
    FDRE #(.INIT(1'b0)) Q15_FF (.C(clk), .CE(CE), .D(out[14]),
.Q(out[15]));
    assign Q = out;
endmodule
```

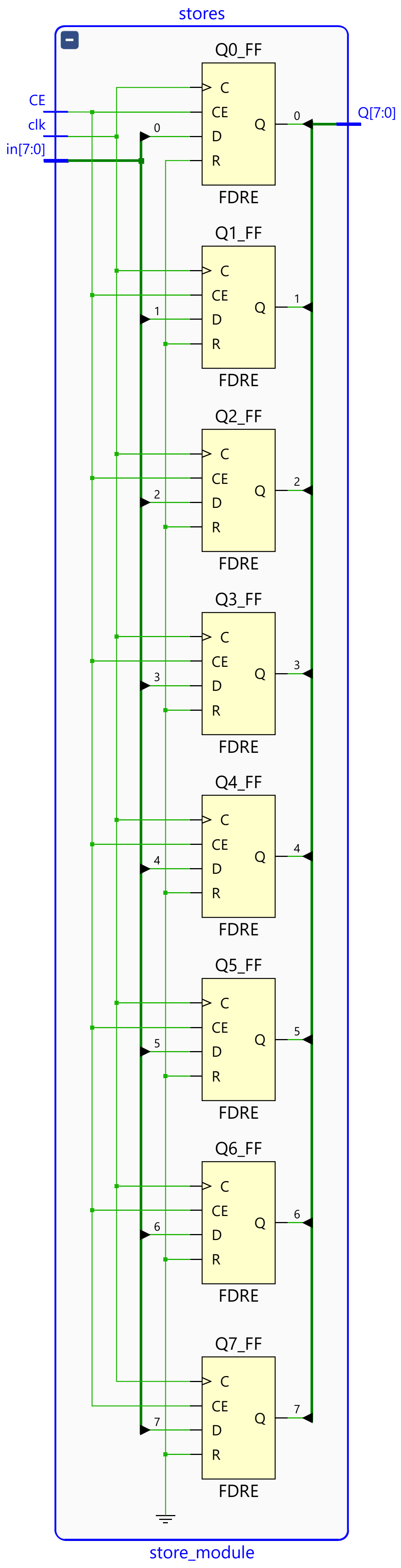
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
////////////////
// Company:
// Engineer:
//
// Create Date: 11/11/2020 03:41:08 PM
// Design Name:
// Module Name: ring_counter_module
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////
////////////////


module ring_counter_module(
    input clk,
    input Advance,
    output [3:0] o
    );
    wire [3:0] out;
    FDRE #(.INIT(1'b1)) Q0_FF (.C(clk), .CE(1'b1), .D(out[3]),
.Q(out[0]));
    FDRE #(.INIT(1'b0)) Q1_FF (.C(clk), .CE(1'b1), .D(out[0]),
.Q(out[1]));
    FDRE #(.INIT(1'b0)) Q2_FF (.C(clk), .CE(1'b1), .D(out[1]),
.Q(out[2]));
    FDRE #(.INIT(1'b0)) Q3_FF (.C(clk), .CE(1'b1), .D(out[2]),
.Q(out[3]));

    assign o = out;
endmodule
```

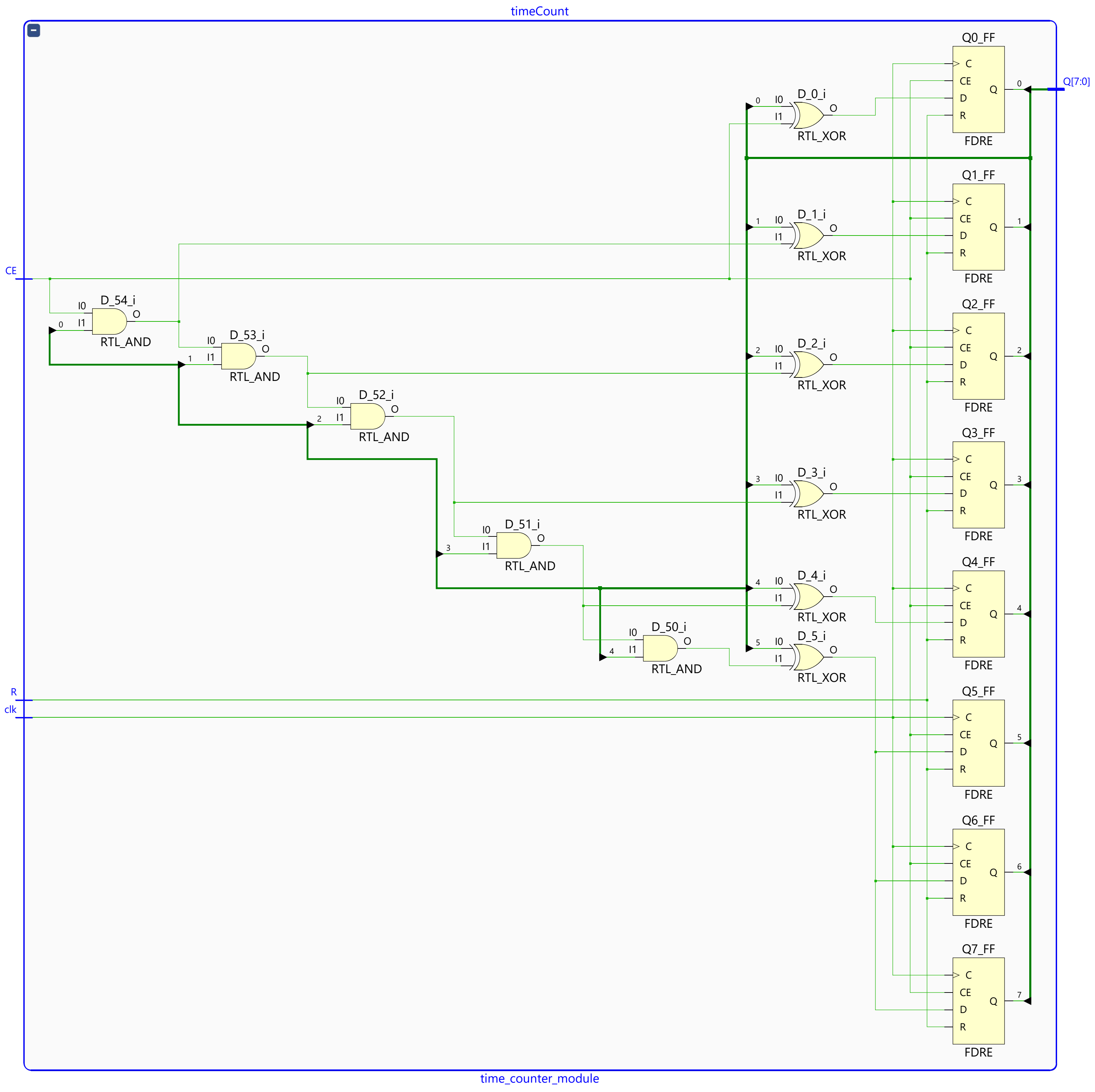
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
///////////////
// Company:
// Engineer:
//
// Create Date: 11/10/2020 09:34:27 PM
// Design Name:
// Module Name: rng_module
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////
///////////////


module rng_module(
    input clk,
    output [7:0] rnd
    );

    wire [7:0] num;
    wire in;
    assign in = num[0] ^ num[5] ^ num[6] ^ num[7];
    FDRE #(.INIT(1'b1)) Q0_FF (.C(clk), .CE(1'b1), .D(in),
.Q(num[0]));
    FDRE #(.INIT(1'b0)) Q1_FF (.C(clk), .CE(1'b1), .D(num[0]),
.Q(num[1]));
    FDRE #(.INIT(1'b0)) Q2_FF (.C(clk), .CE(1'b1), .D(num[1]),
.Q(num[2]));
    FDRE #(.INIT(1'b0)) Q3_FF (.C(clk), .CE(1'b1), .D(num[2]),
.Q(num[3]));
    FDRE #(.INIT(1'b0)) Q4_FF (.C(clk), .CE(1'b1), .D(num[3]),
.Q(num[4]));
    FDRE #(.INIT(1'b0)) Q5_FF (.C(clk), .CE(1'b1), .D(num[4]),
.Q(num[5]));
    //FDRE #(.INIT(1'b0)) Q6_FF (.C(clk), .CE(1'b1), .D(num[5]),
.Q(num[6]));
    //FDRE #(.INIT(1'b0)) Q7_FF (.C(clk), .CE(1'b1), .D(num[6]),
.Q(num[7]));
    assign num[6] = 1'b0;
    assign num[7] = 1'b0;
    assign rnd[7:0] = num[7:0];
endmodule
```


randNum
Q0_FF
in0_i
0
I0
5
I1
O
RTL_XOR
C
CE
D
R
Q
0
FDRE
rnd[7:0]
5:0
clk
Q1_FF
1
FDRE
Q2_FF
2
FDRE
Q3_FF
3
FDRE
Q4_FF
4
FDRE
Q5_FF
5
FDRE
rng_module

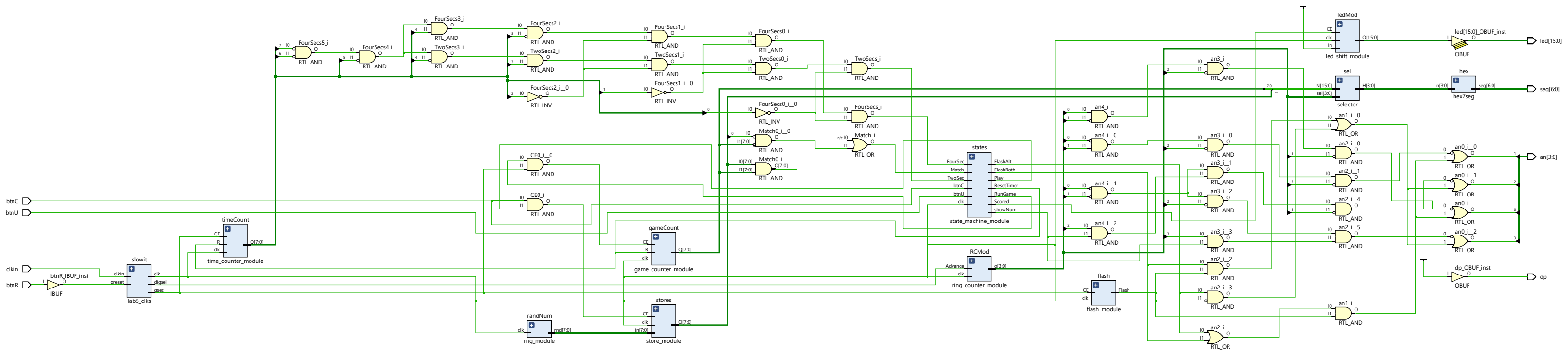

RCMod
clk
Q0_FF
C
CE
D
R
Q
FDRE
o[3:0]
0
3
Q1_FF
C
CE
D
R
Q
FDRE
1
0
Q2_FF
C
CE
D
R
Q
FDRE
2
1
Q3_FF
C
CE
D
R
Q
FDRE
3
2
ring_counter_module


states
FourSec
Match
TwoSec
btnC
btnU
clk
D_00_i__0
D_00_i
D_0_i
D_01_i
D_20_i__0
D_10_i
D_1_i
D_21_i
D_20_i
D_2_i
D_21_i__0
showNum2_i
showNum1_i
showNum0_i
D_31_i
D_30_i__0
D_41_i
D_40_i__0
D_3_i
D_30_i
D_40_i
D_4_i
Q0_FF
Q1_FF
Q2_FF
Q3_FF
Q4_FF
FDRE
Play_i
FlashBoth_i
FlashAlt_i
Scored_i
showNum_i
RTL_AND
RTL_OR
RTL_INV
FlashAlt
FlashBoth
Play
ResetTimer
RunGame
Scored
State[4:0]
n/c
showNum
state_machine_module

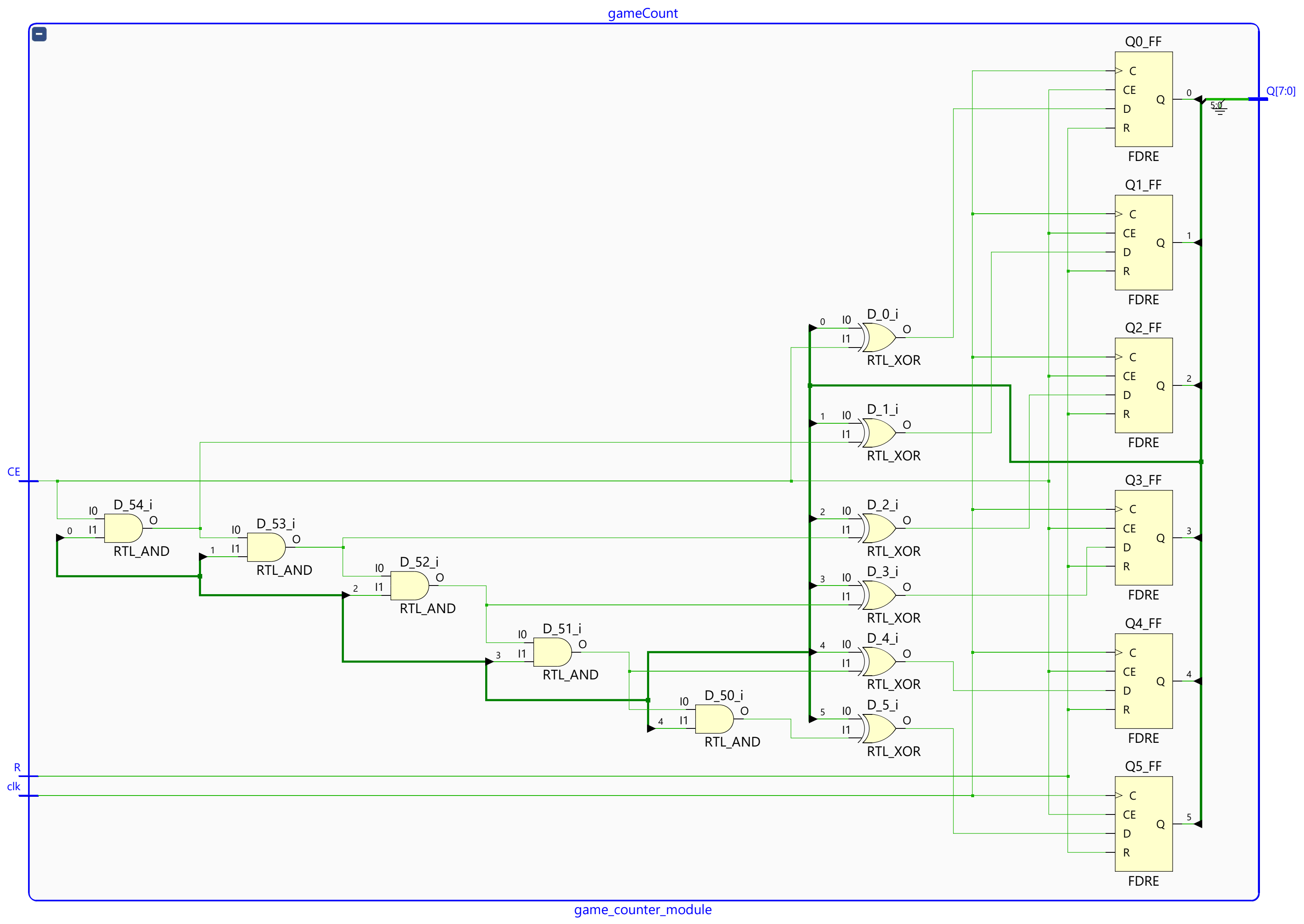

stores
Q0_FF
C
CE
Q
D
R
FDRE
Q1_FF
Q2_FF
Q3_FF
Q4_FF
Q5_FF
Q6_FF
Q7_FF
CE
clk
in[7:0]
Q[7:0]
0
1
2
3
4
5
6
7
store_module

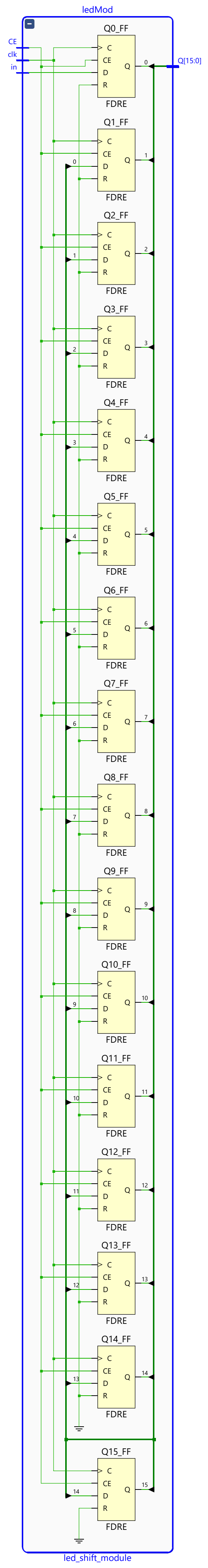


timeCount
CE
R
clk
Q[7:0]
D_54_i
D_53_i
D_52_i
D_51_i
D_50_i
RTL_AND
D_0_i
D_1_i
D_2_i
D_3_i
D_4_i
D_5_i
RTL_XOR
I0
I1
O
Q0_FF
Q1_FF
Q2_FF
Q3_FF
Q4_FF
Q5_FF
Q6_FF
Q7_FF
C
CE
D
R
Q
FDRE
time_counter_module



btnC
btnU
clkin
btnR
btnR_IBUF_inst
IBUF
slowit
lab5_clks
timeCount
time_counter_module
FourSecs5_i
FourSecs4_i
FourSecs3_i
TwoSecs3_i
FourSecs2_i
TwoSecs2_i
FourSecs2_i__0
FourSecs1_i
TwoSecs1_i
FourSecs1_i__0
FourSecs0_i
TwoSecs0_i
FourSecs0_i__0
TwoSecs_i
FourSecs_i
Match0_i__0
Match0_i
Match_i
CE0_i__0
CE0_i
RTL_AND
RTL_INV
RTL_OR
gameCount
game_counter_module
randNum
rng_module
stores
store_module
states
state_machine_module
FourSec
Match
TwoSec
btnC
btnU
clk
FlashAlt
FlashBoth
Play
ResetTimer
RunGame
Scored
showNum
RCMod
ring_counter_module
flash
flash_module
an4_i
an4_i__0
an4_i__1
an4_i__2
an3_i
an3_i__0
an3_i__1
an3_i__2
an3_i__3
an2_i
an2_i__0
an2_i__1
an2_i__2
an2_i__3
an2_i__4
an2_i__5
an1_i
an1_i__0
an0_i
an0_i__0
an0_i__1
an0_i__2
ledMod
led_shift_module
led[15:0]_OBUF_inst
OBUF
sel
selector
hex
hex7seg
dp_OBUF_inst
led[15:0]
seg[6:0]
an[3:0]
dp


gameCount
Q0_FF
Q1_FF
Q2_FF
Q3_FF
Q4_FF
Q5_FF
C
CE
D
R
Q
FDRE
Q[7:0]
5:0
D_0_i
D_1_i
D_2_i
D_3_i
D_4_i
D_5_i
RTL_XOR
D_54_i
D_53_i
D_52_i
D_51_i
D_50_i
RTL_AND
I0
I1
O
CE
R
clk
game_counter_module



ledMod
CE
clk
in
Q[15:0]
Q0_FF
Q1_FF
Q2_FF
Q3_FF
Q4_FF
Q5_FF
Q6_FF
Q7_FF
Q8_FF
Q9_FF
Q10_FF
Q11_FF
Q12_FF
Q13_FF
Q14_FF
Q15_FF
C
CE
D
R
Q
FDRE
led_shift_module