Aadhav Sivakumar
June 3, 2022
Section 01A
Lab 7

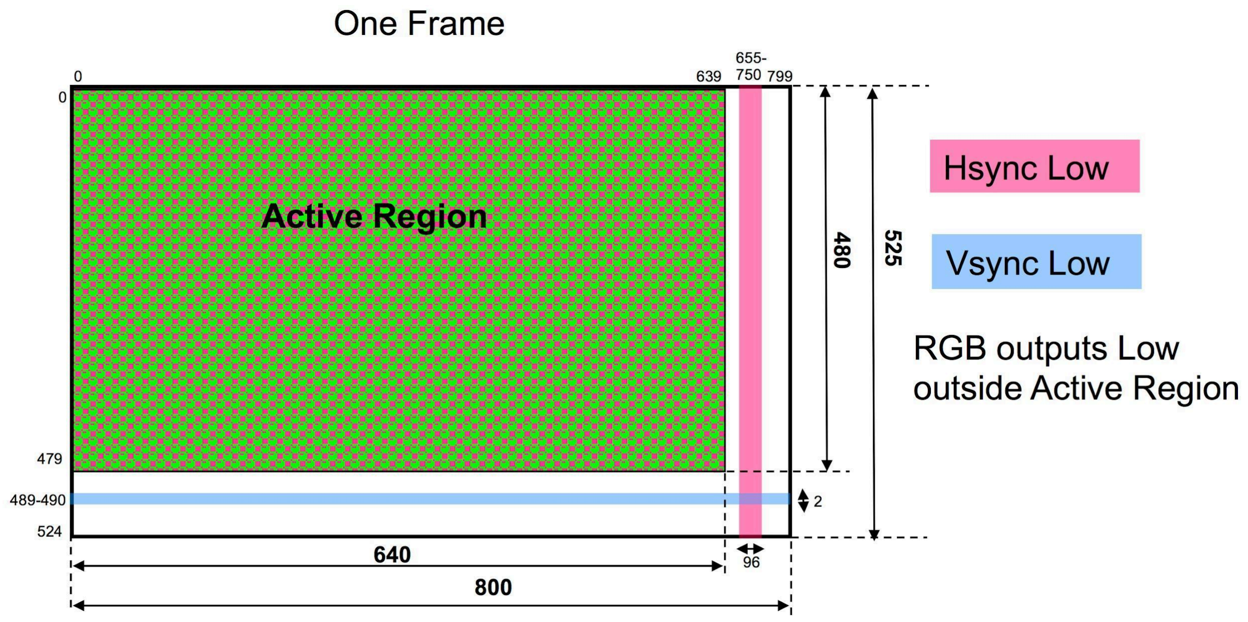
CSE100 Lab 7 write up

Description:

In this lab, the goal was to create a game by using the vga monitor provided in the lab room. The vga protocol was important, and everything just contributed to drawing the correct frame at the right time. The game being made is pretty similar to either flappy bird or frogger. The objective is to move a frog up and down using the input buttons on the BASYS board, and avoid plants that are constantly moving to the left. These plants' positions are randomized, so the player has to make a choice whether to go up or down whenever a plant is encountered. When the frog hits a plant, it should keep on blinking while staying in place until the player hits the restart button, at which point the game starts over. When the start button is hit, it should also make the frog blink 2 times to give the player time to react before the plants start moving.

Methods:

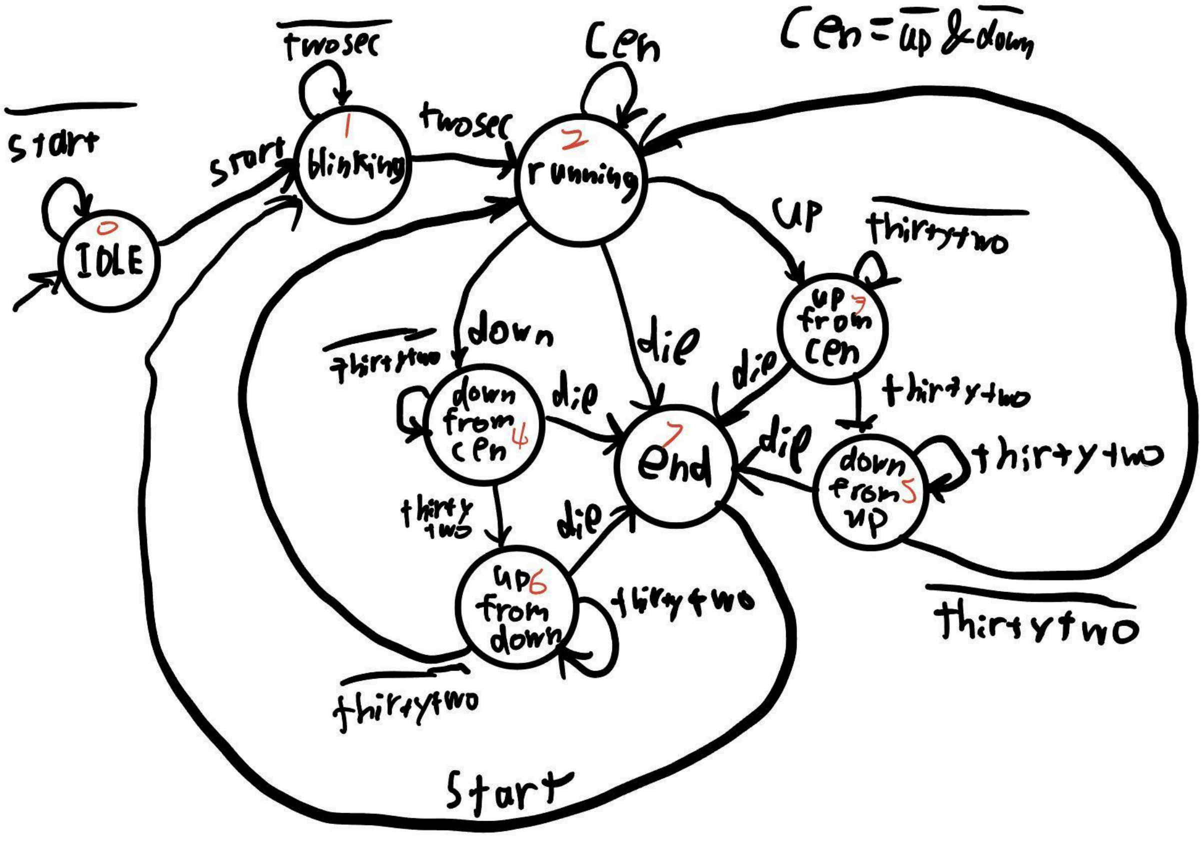
The first part of this lab was making sure that the vga controller worked properly. It works by going 1 pixel to the right every single clock cycle, and setting the specified color for that unique pixel. Once the Hsync signal goes low, the controller starts at the next line and continues to move to the right again. This continues like this until the Vsync signal goes low, which resets the vga controller to the first line, starting the process over again. Below is a visual representation of that:



In the case of this lab, we are using a 640x480 screen, but the actual region is 800x525 pixels, going past the bounds of the screen. Then, a frog was drawn on the screen as a box of white pixels, 16 wide by 16 tall. Once that was drawn, the next thing to be drawn was the water and the plants. The water was based on a gradient, becoming darker and darker as you go further down the screen. The plants start with 2 on the screen, with the third plant going just offscreen.

Now that the basic visuals are in place, it was time to control them. The top edge of the frog was parameterized, and the right edge of each plant was as well. This could then be controlled later by setting that edge to a different value when necessary, making it move.

When they moved is decided by a state machine, and a single one was used for this project. Here is the diagram:



This was implemented in a state machine module, and most of the inputs passed in were from particular buttons. There were also counters that kept track of some of the inputs, such as thirtytwo which goes high after 32 frames and goes back low after 64 frames, controlling the up/down movement of the frog. The die input comes from the combination of the frog and the plant activators, going to the end state regardless of the current state.

To control the actual movement of the plants and the frog, counters were used. Once counter resets after 240 frames (the amount of time it takes for all 3 plants to move across the screen), whereas the frog utilizes the same counter purposed for timing exactly 32 frames.

The blinking was implemented in the same way, using a counter that goes to 64 and resets, hiding the frog when the value is less than 32 and showing it when it's greater than 32. This was incorporated with the activators, and it turns of the color for that thing (in this case, the frog) if the conditional above is true.

After that the main thing was to randomize the y position of the plants. It was done by parameterizing the top edge of each plant and connecting it to the LFSR, a pseudo randomizer. This made the plants appear at random y coordinates while not leaving the water, making the game more challenging and decision based.

Results:

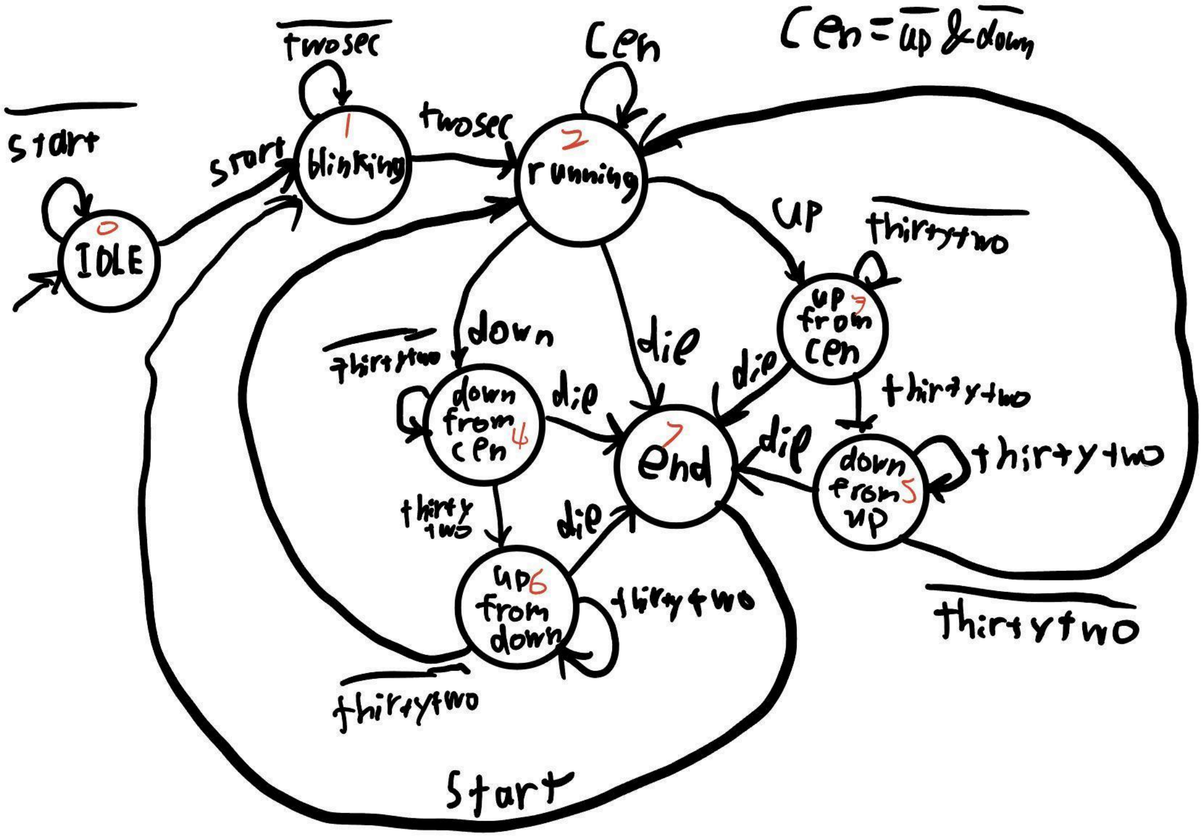
Design:

Old modules:
The modules Ring_counter, selector, and hex7seg are from a previous lab, used in tandem to create a readable hex display 7 segment displays.
The LFSR is from a previous lab, used to generate a random number that can be used to control the vertical heights of each of the plants
Counters:
The counters used in this design are as follows:

- count490
    - Counts to 524 then resets to control the vertical line the vga is currently on, named incorrectly
- count799
    - Counts to 799 then resets to control the current horizontal position of the vga
- frametimecounter
    - Counts every time a frame passes, then resets at 64to control how long the frogs move up and down. Another instance of this module exists to control the blinking after the frog dies or after the start game button is pressed.
- superframetimecounter
    - Counts every time a frame passes, then resets at 240 to control the horizontal position of the plants
- Scorecounter
    - Counts the score every time the frog successfully jumps over another plant
- UD16Lcounter
    - An extra counter to base new counters' design off of

Vnegedge:

This module is a simple thing that detects when there is a negative edge on the Vsync signal. This is used to control when a frame should be counted for things like movement.
GameStateMachine:
This is the state machine that controls when the frog moves, which direction the frog moves, and when the plants move. Its state diagram can be seen here:



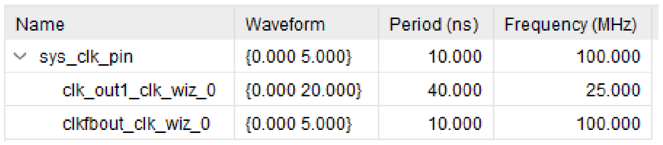
Top Module:

The top module is where most of the code resides, acting as both a connector for all the other modules and acting as the vga controller. It controls the vga by setting the red green and blue values of the pixel at certain points. It does this by having activators for the water, the frog, and the plants. If there are no activators in a single pixel, it draws no colors and the pixel ends up just being black. The top module also controls the collision logic by anding together the or of all the different plant activators. Other than that, it makes sure all of the other modules work well and their inputs/outputs go to where they are necessary.

Simulation & Testing:

The testing started with the clock and vsync/hsync. They both required counters to control them, counters that counted up from 0 to 799 horizontally and 0 to 524 vertically. The provided simulation had a way of testing this, giving errors when the vsync and hsync went low at the wrong times, and also when the colors at each point were wrong as well.

Lab Questions:



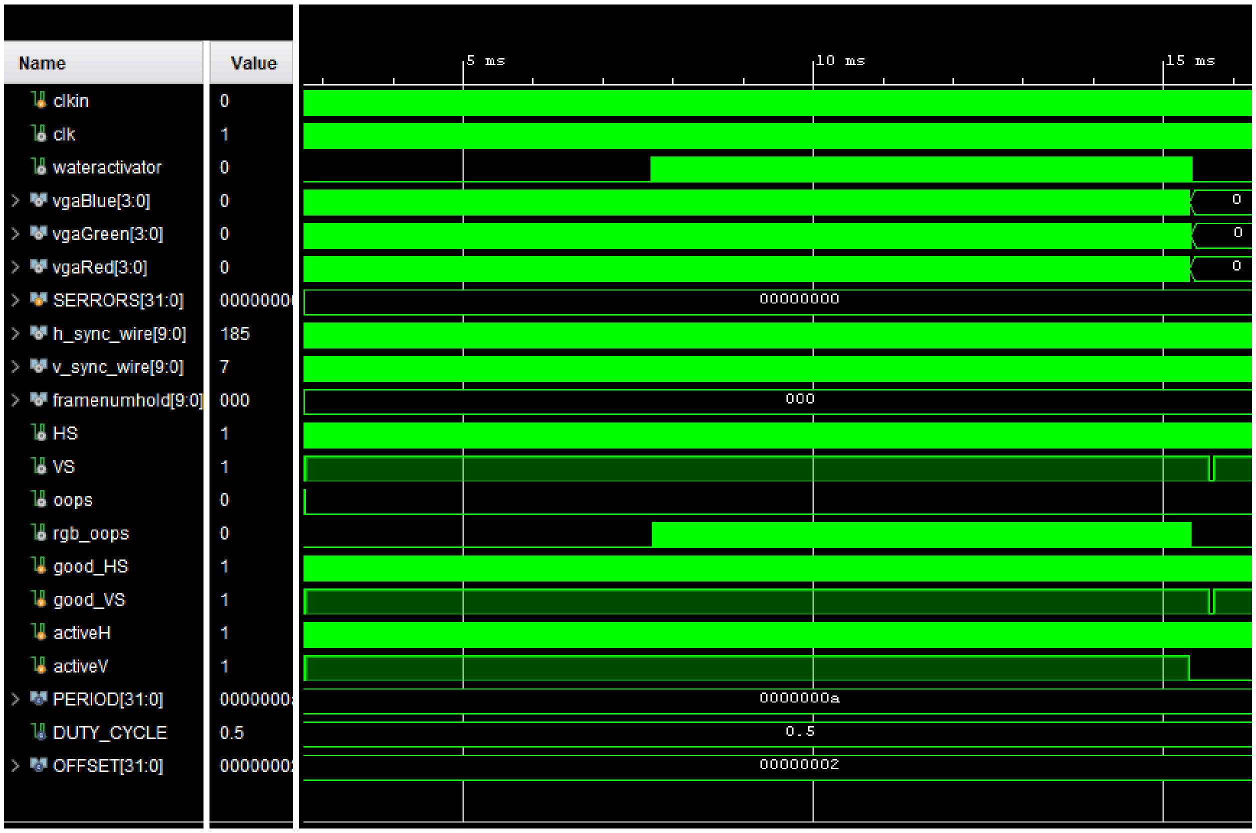
Name	Waveform	Period (ns)	Frequency (MHz)
sys_clk_pin	{0.000 5.000}	10.000	100.000
clk_out1_clk_wiz_0	{0.000 20.000}	40.000	25.000
clkfbout_clk_wiz_0	{0.000 5.000}	10.000	100.000

Conclusion:

While doing this lab, I learned how to program a vga cable to get a desired output on a vga-controlled screen. That was the main takeaway, since the rest of the lab was pretty reliant on what we had done before. It used counters and state machines, something I've had a lot of experience with in this class. The main difficulty was controlling some of the counters, since they constantly reset and when they do, it has an effect on what's happening in the game. Since the game runs constantly, there is no grace period and the plants need to seem like they're moving fluidly, which requires some extra logic when the counters reset. If I did this lab again, I would definitely optimize how the plants move, using 3 separate counters instead of using 1 and having a lot of problems where the only solution was extra logic.

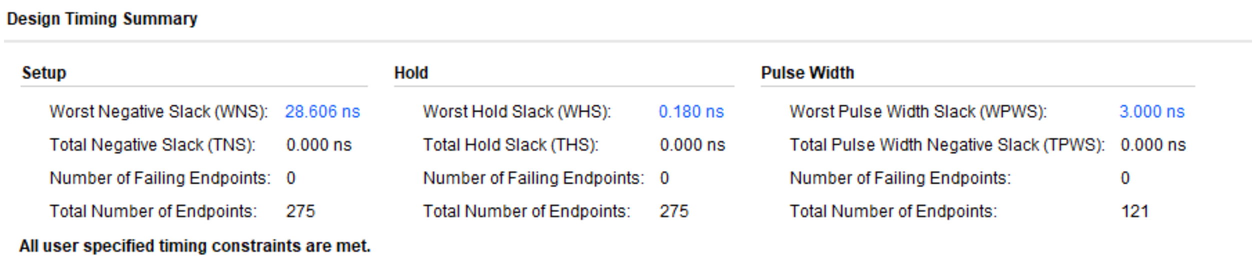
Appendix:

Simulation:



Name	Value
clkin	0
clk	1
wateractivator	0
vgaBlue[3:0]	0
vgaGreen[3:0]	0
vgaRed[3:0]	0
SERRORS[31:0]	0000000
h_sync_wire[9:0]	185
v_sync_wire[9:0]	7
framenumhold[9:0]	000
HS	1
VS	1
oops	0
rgb_oops	0
good_HS	1
good_VS	1
activeH	1
activeV	1
PERIOD[31:0]	000000
DUTY_CYCLE	0.5
OFFSET[31:0]	0000000

Timing Summary Report:



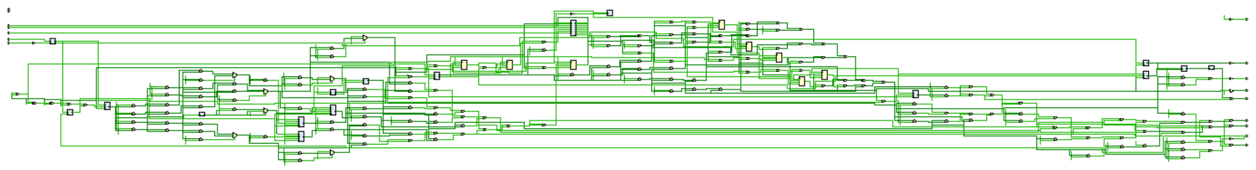
Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	28.606 ns	Worst Hold Slack (WHS):	0.180 ns	Worst Pulse Width Slack (WPWS):	3.000 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	275	Total Number of Endpoints:	275	Total Number of Endpoints:	121

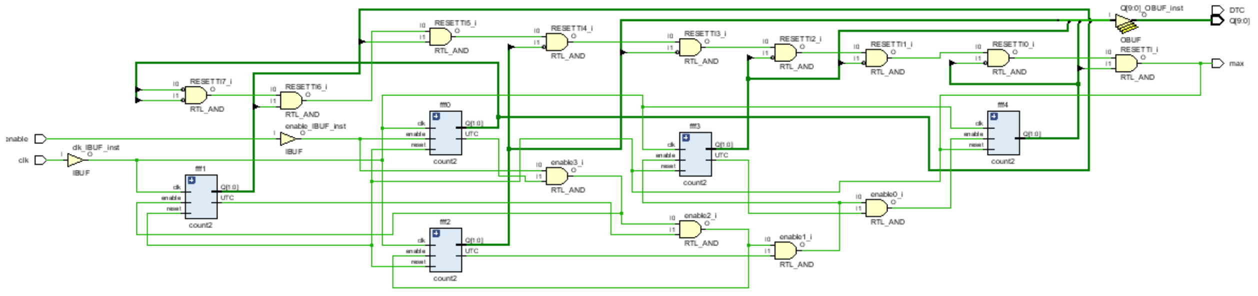
All user specified timing constraints are met.

Schematics:

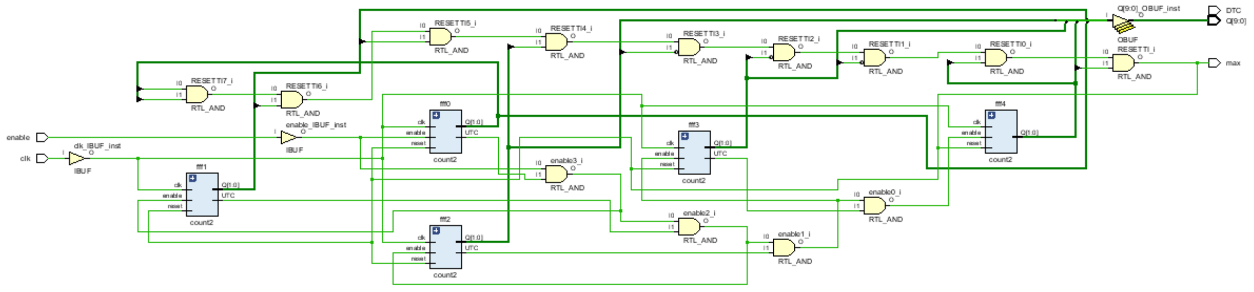
Top Module



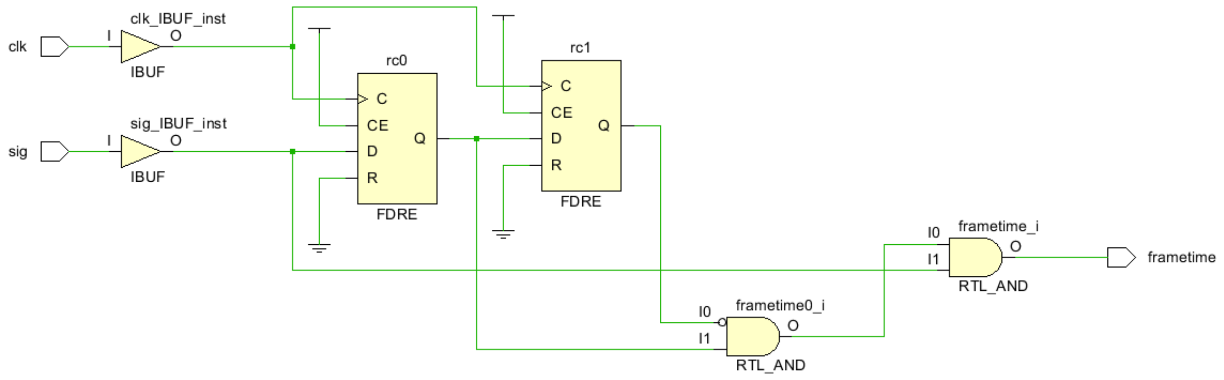
count490



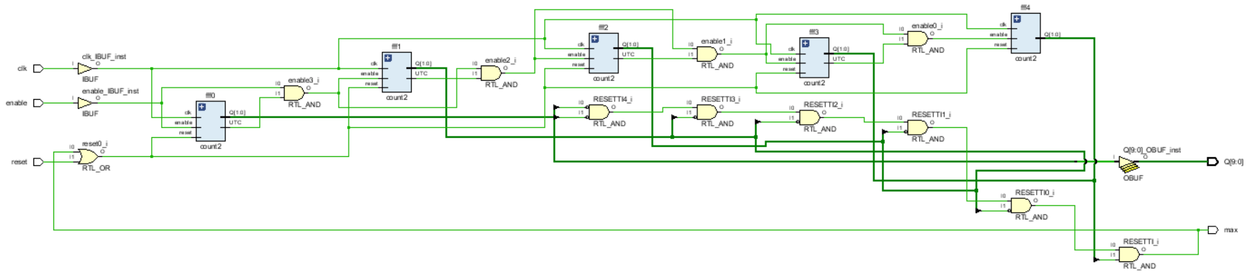
Count799



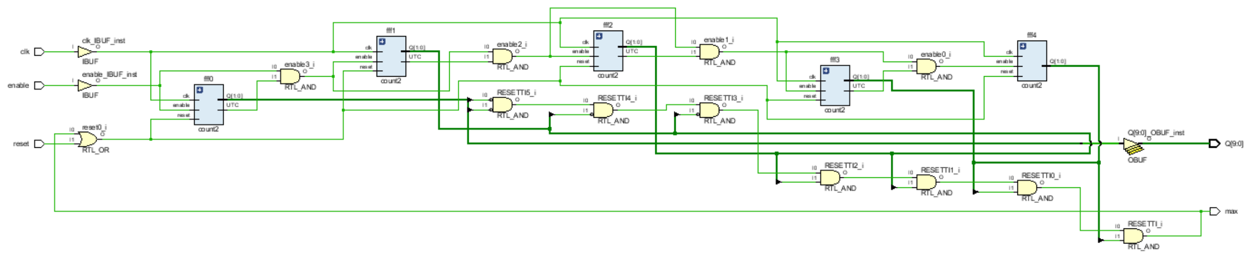
Vnegedge



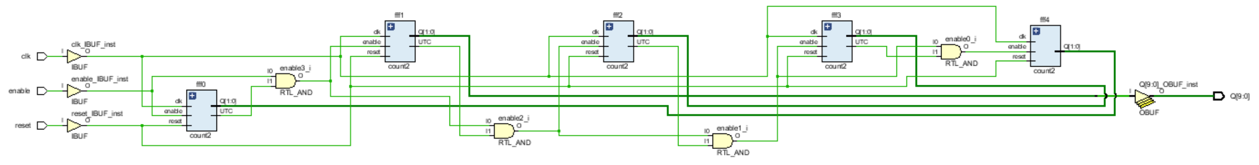
Frametimecounter



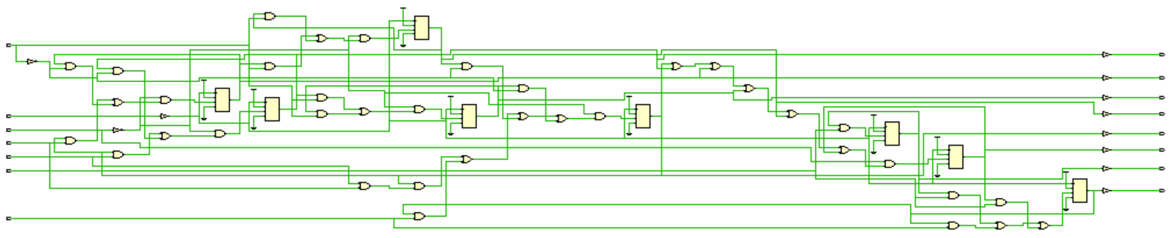
Superframetimecounter



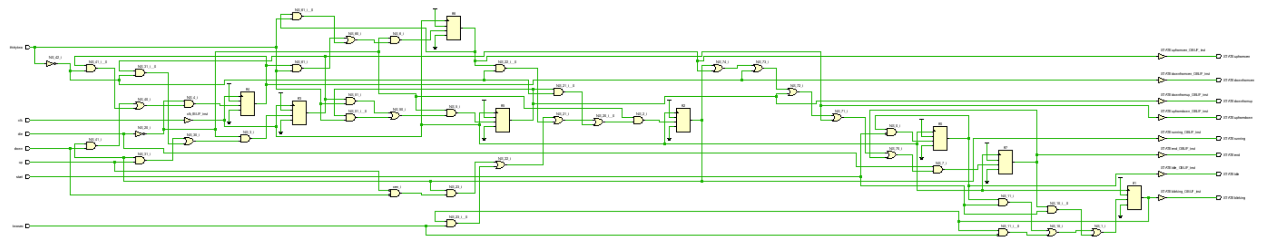
Scorecounter



GameStateMachine



Ring counter



Code:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 05/19/2022 01:49:20 AM
// Design Name:
// Module Name: TopMod
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module TopMod(
    input clkin,
    input btnR,
    input btnC,
    input btnU,
    input btnD,
    input [15:0]sw,
    input btnL,
    output [3:0] vgaRed,
    output [3:0] vgaGreen,
    output [3:0] vgaBlue,
    output Hsync,
    output Vsync,
    output [6:0]seg,
    output [3:0]an,
    output dp,
    output [15:0]led
    );

    wire [3:0]rcout;
    wire [15:0]full16segdisp;
    wire [3:0]H;
    wire [6:0]singledisp;

    wire clk;
    wire digsel;
    wire [9:0] h_sync_wire;
    wire [9:0] v_sync_wire;
    wire max524;
    wire max799;

    wire frogactivator;
    wire wateractivator;
    wire background;
    wire boxcolor;

    wire frametimeholder;
    wire superframer;

    wire testa;
    wire testb;
    wire testc;

    wire [9:0]frogvert;
    wire [9:0]watervert;
    wire frogcontrol;
    wire [9:0]waterpos;

    wire [9:0]framenumhold;

    wire [7:0]stateholder;

    wire btnedge;

    wire framer;

    wire [9:0]superframehold;
```

```
    wire [1:0]twosecwirer;

    wire [9:0]plant1top;
    wire [9:0]plant1left;
    wire [9:0]plant1righttemp;
    wire [9:0]plant1right;
    wire [9:0]plant2top;
    wire [9:0]plant2left;
    wire [9:0]plant2righttemp;
    wire [9:0]plant2right;
    wire [9:0]plant3top;
    wire [9:0]plant3left;
    wire [9:0]plant3righttemp;
    wire [9:0]plant3right;
    wire plant1activator;
    wire plant2activator;
    wire plant3activator;

    wire tempwire;

    wire hadroncollider;

    wire plantmove;
    assign plantmove=stateholder[2]|stateholder[3]|stateholder[4]|stateholder[5]|stateholder[6];



    assign testa=(h_sync_wire>=10'd655)&(h_sync_wire<=10'd750);
    assign testb=(v_sync_wire>=10'd489)&(v_sync_wire<=10'd490);
    lab7_clks not_so_slow (.clkin(clkin), .greset(btnR), .clk(clk), .digsel(digsel));

    count490 c524 (.clk(clk), .enable(max799), .Q(v_sync_wire), .max(max524));

    count799 c799 (.clk(clk), .enable(1'b1), .Q(h_sync_wire), .max(max799));

    vnegedge vn1 (.clk(clk), .sig(~testb), .frametime(frametimeholder));

    vnegedge vn2 (.clk(clk), .sig(~btnC), .frametime(btnedge));

    FDRE #(.INIT(1'b0) ) ff0 (.C(clk), .R(stateholder[0]|stateholder[7]), .CE(framer), .D(1'b1),
.Q(twosecwirer[0]));
    FDRE #(.INIT(1'b0) ) ff1 (.C(clk), .R(stateholder[0]|stateholder[7]),
.CE(twosecwirer[0]&framer), .D(1'b1), .Q(twosecwirer[1]));
```

```
   wire framerendblink;
   wire [9:0]framenumholdendblink;

   frametimecounter ftc (.clk(clk), .reset(stateholder[0]|stateholder[2]),
.enable(~stateholder[7]&frametimeholder), .Q(framenumhold), .max(framer));
   frametimecounter ftcc (.clk(clk), .reset(stateholder[0]|stateholder[2]),
.enable(stateholder[7]&frametimeholder), .Q(framenumholdendblink), .max(framerendblink));
   superframetimecounter sftc (.clk(clk), .reset(stateholder[1]),
.enable(plantmove&frametimeholder), .Q(superframehold), .max(superframer));
   wire [9:0]scoreholder;
   scorecounter scer (.clk(clk), .reset(stateholder[1]), .enable(framer), .Q(scoreholder));

   assign led[0]=stateholder[0];
   assign led[1]=stateholder[1];
   assign led[2]=stateholder[2];
   assign led[3]=stateholder[3];
   assign led[4]=stateholder[4];
   assign led[5]=stateholder[5];
   assign led[6]=stateholder[6];
   assign led[7]=stateholder[7];
   assign led[15]=tempwire;


   assign led[14:8]=7'b0000000;

   GameStateMachine gsm (.clk(clk),
    .start(btnC),
    .twosec(twosecwirer[0]&twosecwirer[1]),
    .thirtytwo(framenumhold>=10'd32),
    .up(btnU),
    .down(btnD),
    .die(tempwire),
   .STATEidle(stateholder[0]),
   .STATEblinking(stateholder[1]),
   .STATErunning(stateholder[2]),
   .STATEupfromcen(stateholder[3]),
   .STATEdownfromcen(stateholder[4]),
   .STATEdownfromup(stateholder[5]),
   .STATEupfromdown(stateholder[6]),
   .STATEend(stateholder[7]));

   Ring_Counter rcc (.clk(clk),.Advance(digsel),.ringout(rcout));
```

```
    Selector seller ( .sel(rcout),.N(scoreholder),.H(H));

    hex7seg h7 (.n(H),.seg(singledisp));

    wire [4:0]endstateholder;

    assign full16segdisp=framenumhold;
    assign dp=1'b1;
    assign an=~rcout;
    assign seg=singledisp|{7{(stateholder[7]&(framenumholdendblink<=10'd32))}};
    assign watervert=10'd240;
    assign frogvert=
    (({10{stateholder[0]}})&(10'd232))|
    (({10{stateholder[1]}})&(10'd232))|
    (({10{stateholder[2]}})&(10'd232))|
    (({10{stateholder[3]}})&(10'd232-framenumhold-framenumhold-framenumhold))|
    (({10{stateholder[4]}})&(10'd232+framenumhold+framenumhold+framenumhold))|
    (({10{stateholder[5]}})&(10'd040+framenumhold+framenumhold+framenumhold))|
    (({10{stateholder[6]}})&(10'd424-framenumhold-framenumhold-framenumhold))|
    (({10{stateholder[7]}})&({10{endstateholder[0]}})&(10'd232))|

(({10{stateholder[7]}})&({10{endstateholder[1]}})&(10'd232-framenumhold-framenumhold-framen
umhold))|

(({10{stateholder[7]}})&({10{endstateholder[2]}})&(10'd232+framenumhold+framenumhold+fram
enumhold))|

(({10{stateholder[7]}})&({10{endstateholder[3]}})&(10'd040+framenumhold+framenumhold+fram
enumhold))|

(({10{stateholder[7]}})&({10{endstateholder[4]}})&(10'd424-framenumhold-framenumhold-framen
umhold));

    FDRE #(.INIT(1'b0) ) adsf1 (.C(clk),
.R(stateholder[3]|stateholder[4]|stateholder[5]|stateholder[6]), .CE(stateholder[2]),
.D(stateholder[2]), .Q(endstateholder[0]));
    FDRE #(.INIT(1'b0) ) adsf2 (.C(clk),
.R(stateholder[2]|stateholder[4]|stateholder[5]|stateholder[6]), .CE(stateholder[3]),
.D(stateholder[3]), .Q(endstateholder[1]));
    FDRE #(.INIT(1'b0) ) adsf3 (.C(clk),
.R(stateholder[3]|stateholder[2]|stateholder[5]|stateholder[6]), .CE(stateholder[4]),
.D(stateholder[4]), .Q(endstateholder[2]));
```

```
   FDRE #(.INIT(1'b0) ) adsf4 (.C(clk),
.R(stateholder[3]|stateholder[4]|stateholder[2]|stateholder[6]), .CE(stateholder[5]),
.D(stateholder[5]), .Q(endstateholder[3]));
   FDRE #(.INIT(1'b0) ) adsf5 (.C(clk),
.R(stateholder[3]|stateholder[4]|stateholder[5]|stateholder[2]), .CE(stateholder[6]),
.D(stateholder[6]), .Q(endstateholder[4]));

   wire [3:0]rander;

   LFSR lf (.clk(clk), .rnd(rander));

   wire [5:0]vertdet;
   assign vertdet={rander[2:0],2'b00};
   wire [5:0]finalvertchange;
   assign finalvertchange=(rander[3])?(5'b00000-vertdet):(5'b00000+vertdet);

   wire [16:0]plant1topstore;
   wire [16:0]plant2topstore;
   wire [16:0]plant3topstore;


   UD16Lcounter pl1t ( .clk(clk),
   .Up(1'b0),
   .Dw(1'b0),
   .LD(plant1right>10'd681),
   .Din(finalvertchange),
   .Q(plant1topstore)
   );
   UD16Lcounter pl2t ( .clk(clk),
   .Up(1'b0),
   .Dw(1'b0),
   .LD(plant2right>10'd681),
   .Din(finalvertchange),
   .Q(plant2topstore)
   );
   UD16Lcounter pl3t ( .clk(clk),
   .Up(1'b0),
   .Dw(1'b0),
   .LD(plant3right>10'd681),
   .Din(finalvertchange),
   .Q(plant3topstore)
   );

   assign plant1top=10'd178+plant1topstore;
   assign
plant1right=((superframehold)>10'd80)?(10'd960-superframehold-superframehold-superframeho
ld):(10'd240-superframehold-superframehold-superframehold);
   assign plant1left=(plant1right<10'd040)?10'd000:(plant1right-10'd40);
   assign plant2top=10'd178+plant2topstore;
   assign
plant2right=((superframehold)>10'd160)?(11'd1200-superframehold-superframehold-superframe
hold):(10'd480-superframehold-superframehold-superframehold);
   assign plant2left=(plant2right<10'd040)?10'd000:(plant2right-10'd40);
   assign plant3top=10'd178+plant3topstore;
   assign
plant3right=((superframehold)>10'd240)?(11'd1440-superframehold-superframehold-superframe
hold):(10'd720-superframehold-superframehold-superframehold);
   assign plant3left=(plant3right<10'd040)?10'd000:(plant3right-10'd40);

   FDRE #(.INIT(1'b0) ) ff7 (.C(clk), .R(stateholder[0]|stateholder[1]), .CE(hadroncollider),
.D(1'b1), .Q(tempwire));

   assign hadroncollider=frogactivator&(plant1activator|plant2activator|plant3activator);

   assign
frogcontrol=~((stateholder[1]&(framenumhold<=10'd32))|(stateholder[7]&(framenumholdendblin
k<=10'd32)));

   wire activeregion;
   assign activeregion=(h_sync_wire<10'd641)&(v_sync_wire<10'd481);

   assign frogactivator=
(frogcontrol)&((h_sync_wire<=10'd136)&(h_sync_wire>=10'd120)&(v_sync_wire<=(frogvert+10'
d16))&(v_sync_wire>=frogvert));
   assign
wateractivator=(~frogactivator&~plant1activator&~plant2activator&~plant3activator)&((h_sync_
wire<=10'd640)&(h_sync_wire>=10'd0)&(v_sync_wire<=(watervert+10'd240))&(v_sync_wire>=
watervert));
   assign plant1activator=
((h_sync_wire<=(plant1right))&(h_sync_wire>=plant1left)&(v_sync_wire<=(plant1top+10'd96))&(
v_sync_wire>=plant1top));
   assign plant2activator=
((h_sync_wire<=(plant2right))&(h_sync_wire>=plant2left)&(v_sync_wire<=(plant2top+10'd96))&(
v_sync_wire>=plant2top));
   assign plant3activator=
((h_sync_wire<=(plant3right))&(h_sync_wire>=plant3left)&(v_sync_wire<=(plant3top+10'd96))&(
v_sync_wire>=plant3top));
```

```
    assign waterpos=v_sync_wire-10'd240;
    assign vgaRed=
({4{activeregion}})&(({4{frogactivator}}&4'hF)|({4{wateractivator}}&4'h0)|({4{plant1activator}}&4'h
0)|({4{plant2activator}}&4'h0)|({4{plant3activator}}&4'h0));
    assign vgaGreen=
({4{activeregion}})&(({4{frogactivator}}&4'hF)|({4{wateractivator}}&4'h0)|({4{plant1activator}}&4'h
F)|({4{plant2activator}}&4'hF)|({4{plant3activator}}&4'hF));
    assign vgaBlue=
({4{activeregion}})&(({4{frogactivator}}&4'hF)|({4{wateractivator}}&(4'hF-(waterpos[7:4]))));

    assign Hsync=~testa;
    assign Vsync=~testb;

endmodule

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 05/18/2022 11:55:23 PM
// Design Name:
// Module Name: count10
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module count490(
        input clk,
        input enable,
        output [9:0]Q,
        output DTC,
        output max
    );
    wire [4:0]UTCWIRE;
    wire RESETTI=~Q[0]&~Q[1]&Q[2]&Q[3]&~Q[4]&~Q[5]&~Q[6]&~Q[7]&~Q[8]&Q[9];
    assign max=RESETTI;
    count2 fff0 (.clk(clk), .reset(RESETTI), .enable(enable), .Q(Q[1:0]), .UTC(UTCWIRE[0]));
    count2 fff1 (.clk(clk), .reset(RESETTI), .enable(enable&UTCWIRE[0]), .Q(Q[3:2]),
.UTC(UTCWIRE[1]));
    count2 fff2 (.clk(clk), .reset(RESETTI),.enable(enable&UTCWIRE[0]&UTCWIRE[1]),
.Q(Q[5:4]), .UTC(UTCWIRE[2]));
    count2 fff3 (.clk(clk),
.reset(RESETTI),.enable(enable&UTCWIRE[0]&UTCWIRE[1]&UTCWIRE[2]), .Q(Q[7:6]),
.UTC(UTCWIRE[3]));
    count2 fff4 (.clk(clk),
.reset(RESETTI),.enable(enable&UTCWIRE[0]&UTCWIRE[1]&UTCWIRE[2]&UTCWIRE[3]),
.Q(Q[9:8]), .UTC(UTCWIRE[4]));

endmodule

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 05/18/2022 11:55:23 PM
// Design Name:
// Module Name: count10
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module count799(
        input clk,
        input enable,
        output [9:0]Q,
        output DTC,
        output max
    );

    wire [4:0]UTCWIRE;
    wire RESETTI=Q[0]&Q[1]&Q[2]&Q[3]&Q[4]&~Q[5]&~Q[6]&~Q[7]&Q[8]&Q[9];

    assign max=RESETTI;

    count2 fff0 (.clk(clk), .reset(RESETTI), .enable(enable), .Q(Q[1:0]), .UTC(UTCWIRE[0]));
    count2 fff1 (.clk(clk), .reset(RESETTI), .enable(enable&UTCWIRE[0]), .Q(Q[3:2]),
.UTC(UTCWIRE[1]));
    count2 fff2 (.clk(clk), .reset(RESETTI), .enable(enable&UTCWIRE[0]&UTCWIRE[1]),
.Q(Q[5:4]), .UTC(UTCWIRE[2]));
    count2 fff3 (.clk(clk), .reset(RESETTI),
.enable(enable&UTCWIRE[0]&UTCWIRE[1]&UTCWIRE[2]), .Q(Q[7:6]), .UTC(UTCWIRE[3]));
    count2 fff4 (.clk(clk), .reset(RESETTI),
.enable(enable&UTCWIRE[0]&UTCWIRE[1]&UTCWIRE[2]&UTCWIRE[3]), .Q(Q[9:8]),
.UTC(UTCWIRE[4]));
endmodule

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 05/21/2022 11:05:59 PM
// Design Name:
// Module Name: vnegedge
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module vnegedge(
        input sig,
        input clk,
        output frametime
    );
    wire [1:0]hold;


    FDRE #(.INIT(1'b0) ) rc0 (.C(clk), .CE(1'b1), .D(sig), .Q(hold[0]));
    FDRE #(.INIT(1'b0) ) rc1 (.C(clk), .CE(1'b1), .D(hold[0]), .Q(hold[1]));
    assign frametime=~hold[1]&hold[0]&sig;


endmodule

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 05/19/2022 09:55:44 PM
// Design Name:
// Module Name: frametimecounter
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module frametimecounter(
    input clk,
    input enable,
    input reset,
    output [9:0] Q,
    output max
    );

    wire [9:0]UTCWIRE;
    wire RESETTI;
    assign RESETTI=~Q[0]&~Q[1]&~Q[2]&~Q[3]&~Q[4]&~Q[5]&Q[6];
    assign max=RESETTI;
    count2 fff0 (.clk(clk), .reset(RESETTI|reset), .enable(enable), .Q(Q[1:0]),
.UTC(UTCWIRE[0]));
    count2 fff1 (.clk(clk), .reset(RESETTI|reset), .enable(enable&UTCWIRE[0]), .Q(Q[3:2]),
.UTC(UTCWIRE[1]));
    count2 fff2 (.clk(clk), .reset(RESETTI|reset), .enable(enable&UTCWIRE[0]&UTCWIRE[1]),
.Q(Q[5:4]), .UTC(UTCWIRE[2]));
    count2 fff3 (.clk(clk), .reset(RESETTI|reset),
.enable(enable&UTCWIRE[0]&UTCWIRE[1]&UTCWIRE[2]), .Q(Q[7:6]), .UTC(UTCWIRE[3]));
    count2 fff4 (.clk(clk), .reset(RESETTI|reset),
.enable(enable&UTCWIRE[0]&UTCWIRE[1]&UTCWIRE[2]&UTCWIRE[3]), .Q(Q[9:8]),
.UTC(UTCWIRE[4]));


endmodule

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 05/19/2022 09:55:44 PM
// Design Name:
// Module Name: frametimecounter
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module superframetimecounter(
    input clk,
    input enable,
    input reset,
    output [9:0] Q,
    output max
    );

    wire [9:0]UTCWIRE;
    wire RESETTI;

    assign RESETTI=~Q[0]&~Q[1]&~Q[2]&~Q[3]&Q[4]&Q[5]&Q[6]&Q[7];
    assign max=RESETTI;
    count2 fff0 (.clk(clk), .reset(RESETTI|reset), .enable(enable), .Q(Q[1:0]),
.UTC(UTCWIRE[0]));
    count2 fff1 (.clk(clk), .reset(RESETTI|reset), .enable(enable&UTCWIRE[0]), .Q(Q[3:2]),
.UTC(UTCWIRE[1]));
    count2 fff2 (.clk(clk), .reset(RESETTI|reset), .enable(enable&UTCWIRE[0]&UTCWIRE[1]),
.Q(Q[5:4]), .UTC(UTCWIRE[2]));
    count2 fff3 (.clk(clk), .reset(RESETTI|reset),
.enable(enable&UTCWIRE[0]&UTCWIRE[1]&UTCWIRE[2]), .Q(Q[7:6]), .UTC(UTCWIRE[3]));
    count2 fff4 (.clk(clk), .reset(RESETTI|reset),
.enable(enable&UTCWIRE[0]&UTCWIRE[1]&UTCWIRE[2]&UTCWIRE[3]), .Q(Q[9:8]),
.UTC(UTCWIRE[4]));









endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 05/26/2022 11:19:02 AM
// Design Name:
// Module Name: scorecounter
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module scorecounter(
    input clk,
    input enable,
    input reset,
    output [9:0] Q
    );
    assign RESETTI=1'b0;
    wire [4:0]UTCWIRE;
    count2 fff0 (.clk(clk), .reset(RESETTI|reset), .enable(enable), .Q(Q[1:0]),
.UTC(UTCWIRE[0]));
    count2 fff1 (.clk(clk), .reset(RESETTI|reset), .enable(enable&UTCWIRE[0]), .Q(Q[3:2]),
.UTC(UTCWIRE[1]));
    count2 fff2 (.clk(clk), .reset(RESETTI|reset), .enable(enable&UTCWIRE[0]&UTCWIRE[1]),
.Q(Q[5:4]), .UTC(UTCWIRE[2]));
    count2 fff3 (.clk(clk), .reset(RESETTI|reset),
.enable(enable&UTCWIRE[0]&UTCWIRE[1]&UTCWIRE[2]), .Q(Q[7:6]), .UTC(UTCWIRE[3]));
    count2 fff4 (.clk(clk), .reset(RESETTI|reset),
.enable(enable&UTCWIRE[0]&UTCWIRE[1]&UTCWIRE[2]&UTCWIRE[3]), .Q(Q[9:8]),
.UTC(UTCWIRE[4]));
endmodule

`timescale 1ns / 1ps
```

```
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 05/21/2022 11:57:27 PM
// Design Name:
// Module Name: GameStateMachine
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module GameStateMachine(
        input clk,
        input start,
        input twosec,
        input thirtytwo,
        input up,
        input down,
        input die,
        output STATEidle,
        output STATEblinking,
        output STATErunning,
        output STATEupfromcen,
        output STATEdownfromcen,
        output STATEdownfromup,
        output STATEupfromdown,
        output STATEend
    );
    wire [7:0]NS;
    //states:IDLE,blinking,running
    wire cen;
    assign cen=~up&~down;
    FDRE #(.INIT(1'b1) ) ff0 (.C(clk),  .CE(1'b1), .D(NS[0]), .Q(STATEidle));
    FDRE #(.INIT(1'b0) ) ff1 (.C(clk),  .CE(1'b1), .D(NS[1]), .Q(STATEblinking));
    FDRE #(.INIT(1'b0) ) ff2 (.C(clk),  .CE(1'b1), .D(NS[2]), .Q(STATErunning));
    FDRE #(.INIT(1'b0) ) ff3 (.C(clk),  .CE(1'b1), .D(NS[3]), .Q(STATEupfromcen));
    FDRE #(.INIT(1'b0) ) ff4 (.C(clk),  .CE(1'b1), .D(NS[4]), .Q(STATEdownfromcen));
    FDRE #(.INIT(1'b0) ) ff5 (.C(clk),  .CE(1'b1), .D(NS[5]), .Q(STATEdownfromup));
    FDRE #(.INIT(1'b0) ) ff6 (.C(clk),  .CE(1'b1), .D(NS[6]), .Q(STATEupfromdown));
    FDRE #(.INIT(1'b0) ) ff7 (.C(clk),  .CE(1'b1), .D(NS[7]), .Q(STATEend));

    assign NS[0]=(STATEidle&~start);
    assign NS[1]=(STATEidle&start)|(STATEblinking&~twosec)|(STATEend&start);
    assign
NS[2]=~die&((STATErunning&cen)|(STATEblinking&twosec)|(STATEupfromdown&~thirtytwo)|(S
TATEdownfromup&~thirtytwo));
    assign NS[3]=~die&((STATErunning&up)|(STATEupfromcen&~thirtytwo));
    assign NS[4]=~die&((STATErunning&down)|(STATEdownfromcen&~thirtytwo));
    assign NS[5]=~die&((STATEupfromcen&thirtytwo)|(STATEdownfromup&thirtytwo));
    assign NS[6]=~die&((STATEdownfromcen&thirtytwo)|(STATEupfromdown&thirtytwo));
    assign
NS[7]=(die&(STATErunning|STATEupfromcen|STATEdownfromcen|STATEdownfromup|STATEu
pfromdown|STATEend));

endmodule

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04/30/2022 06:17:15 PM
// Design Name:
// Module Name: RingCounter
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
```

```
module Ring_Counter(
    input clk,

    input Advance,
    output [3:0] ringout
    );



    FDRE #(.INIT(1'b1) ) rc0 (.C(clk), .CE(Advance), .D(ringout[3]), .Q(ringout[0]));
    FDRE #(.INIT(1'b0) ) rc1 (.C(clk), .CE(Advance), .D(ringout[0]), .Q(ringout[1]));
    FDRE #(.INIT(1'b0) ) rc2 (.C(clk), .CE(Advance), .D(ringout[1]), .Q(ringout[2]));
    FDRE #(.INIT(1'b0) ) rc3 (.C(clk), .CE(Advance), .D(ringout[2]), .Q(ringout[3]));


endmodule

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04/20/2022 09:47:56 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]=(N[0]&sel[0])|(N[4]&sel[1])|(N[8]&sel[2])|(N[12]&sel[3]);
    assign H[1]=(N[1]&sel[0])|(N[5]&sel[1])|(N[9]&sel[2])|(N[13]&sel[3]);
    assign H[2]=(N[2]&sel[0])|(N[6]&sel[1])|(N[10]&sel[2])|(N[14]&sel[3]);
    assign H[3]=(N[3]&sel[0])|(N[7]&sel[1])|(N[11]&sel[2])|(N[15]&sel[3]);
endmodule

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04/30/2022 06:17:36 PM
// Design Name:
// Module Name: hex7seg
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module m8_1(
    input [7:0] in,
    input [2:0] sel,
    output o
    );
    assign o=
     (~sel[0]&~sel[1]&~sel[2]&in[0])|
     (sel[0]&~sel[1]&~sel[2]&in[1])|
     (~sel[0]&sel[1]&~sel[2]&in[2])|
     (sel[0]&sel[1]&~sel[2]&in[3])|
     (~sel[0]&~sel[1]&sel[2]&in[4])|
     (sel[0]&~sel[1]&sel[2]&in[5])|
     (~sel[0]&sel[1]&sel[2]&in[6])|
     (sel[0]&sel[1]&sel[2]&in[7]);
endmodule
```

```
module hex7seg(
   input [3:0] n,
   output [6:0] seg
   );
   m8_1 seglog0 (.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]));
   m8_1 seglog1 (.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]));
   m8_1 seglog2 (.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]));
   m8_1 seglog3 (.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]));
   m8_1 seglog4 (.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]));
   m8_1 seglog5 (.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]));
   m8_1 seglog6 (.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]));
endmodule
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04/30/2022 06:17:36 PM
// Design Name:
// Module Name: hex7seg
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module m8_1(
   input [7:0] in,
    input [2:0] sel,
    output o
    );
    assign o=
     (~sel[0]&~sel[1]&~sel[2]&in[0])|
     (sel[0]&~sel[1]&~sel[2]&in[1])|
     (~sel[0]&sel[1]&~sel[2]&in[2])|
     (sel[0]&sel[1]&~sel[2]&in[3])|
     (~sel[0]&~sel[1]&sel[2]&in[4])|
     (sel[0]&~sel[1]&sel[2]&in[5])|
     (~sel[0]&sel[1]&sel[2]&in[6])|
     (sel[0]&sel[1]&sel[2]&in[7]);
endmodule




module hex7seg(
    input [3:0] n,
    output [6:0] seg
    );
    m8_1 seglog0 (.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]));
    m8_1 seglog1 (.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]));
    m8_1 seglog2 (.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]));
    m8_1 seglog3 (.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]));
    m8_1 seglog4 (.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]));
    m8_1 seglog5 (.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]));
    m8_1 seglog6 (.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]));
endmodule

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04/30/2022 07:02:13 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 [3:0]rnd
   );
   wire [3:0] interand;
   wire xorin;
   assign xorin=interand[0]^interand[2]^interand[3];
   FDRE #(.INIT(1'b1) ) LFSRFF0 (.C(clk), .CE(1'b1), .D(xorin), .Q(interand[0]));
   FDRE #(.INIT(1'b0) ) LFSRFF1 (.C(clk), .CE(1'b1), .D(interand[0]), .Q(interand[1]));
   FDRE #(.INIT(1'b0) ) LFSRFF2 (.C(clk), .CE(1'b1), .D(interand[1]), .Q(interand[2]));
   FDRE #(.INIT(1'b0) ) LFSRFF3 (.C(clk), .CE(1'b1), .D(interand[2]), .Q(interand[3]));


   assign rnd=interand;

endmodule

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04/19/2022 08:19:56 PM
// Design Name:
// Module Name: counterUD16L
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
```

```
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module UD16Lcounter(
   input clk,
   input Up,
   input Dw,
   input LD,
   input [15:0] Din,
   output [15:0] Q,
   output UTC,
   output DTC
   );

   wire [3:0] smallUTC;
   wire [3:0] smallDTC;

   countUD4L counter0 (.Din(Din[3:0]), .Up(Up), .Dw(Dw), .clk(clk), .LD(LD),
.Q(Q[3:0]),.UTC(smallUTC[0]),.DTC(smallDTC[0]));
   countUD4L counter1 (.Din(Din[7:4]), .Up(Up&smallUTC[0]), .Dw(Dw&smallDTC[0]), .clk(clk),
.LD(LD), .Q(Q[7:4]),.UTC(smallUTC[1]),.DTC(smallDTC[1]));
   countUD4L counter2 (.Din(Din[11:8]), .Up(Up&smallUTC[1]&smallUTC[0]),
.Dw(Dw&smallDTC[1]&smallDTC[0]), .clk(clk), .LD(LD),
.Q(Q[11:8]),.UTC(smallUTC[2]),.DTC(smallDTC[2]));
   countUD4L counter3 (.Din(Din[15:12]), .Up(Up&smallUTC[2]&smallUTC[1]&smallUTC[0]),
.Dw(Dw&smallDTC[2]&smallDTC[1]&smallDTC[0]), .clk(clk), .LD(LD),
.Q(Q[15:12]),.UTC(smallUTC[3]),.DTC(smallDTC[3]));

   assign UTC=smallUTC[0]&smallUTC[1]&smallUTC[2]&smallUTC[3];
   assign DTC=smallDTC[0]&smallDTC[1]&smallDTC[2]&smallDTC[3];

endmodule

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
```

```
//
// Create Date: 04/19/2022 10:55:25 AM
// Design Name:
// Module Name: countUD4L
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module countUD4L(
    input clk,
    input Up,
    input Dw,
    input LD,
    input [3:0] Din,
    output [3:0] Q,
    output UTC,
    output DTC
    );
    wire [3:0] T;
    wire [3:0] FFLogic;
    wire enabler;
    assign enabler=LD|(Up^Dw);

    assign FFLogic[0]=(LD&Din[0])|(~LD&Up&(~T[0]))|(~LD&Dw&(~T[0]));
    assign FFLogic[1]=(LD&Din[1])|(~LD&Up&(T[0]^T[1]))|(~LD&Dw&~(T[0]^T[1]));
    assign
FFLogic[2]=(LD&Din[2])|(~LD&Up&((~T[1]&T[2])|(~T[0]&T[2])|(T[0]&T[1]&~T[2])))|(~LD&Dw&((T[
1]&T[2])|(T[0]&T[2])|(~T[0]&~T[1]&~T[2])));
    assign
FFLogic[3]=(LD&Din[3])|(~LD&Up&((T[0]&T[1]&T[2]&~T[3])|(~T[0]&T[3])|(~T[1]&T[3])|(~T[2]&T[3
])))|(~LD&Dw&((~T[0]&~T[1]&~T[2]&~T[3])|(T[2]&T[3])|(T[0]&T[3])|(T[1]&T[3])));

    FDRE #(.INIT(1'b0) ) ff0 (.C(clk), .CE(enabler), .D(FFLogic[0]), .Q(T[0]));
    FDRE #(.INIT(1'b0) ) ff1 (.C(clk), .CE(enabler), .D(FFLogic[1]), .Q(T[1]));
    FDRE #(.INIT(1'b0) ) ff2 (.C(clk), .CE(enabler), .D(FFLogic[2]), .Q(T[2]));
    FDRE #(.INIT(1'b0) ) ff3 (.C(clk), .CE(enabler), .D(FFLogic[3]), .Q(T[3]));

    assign Q[0]=T[0];
    assign Q[1]=T[1];
    assign Q[2]=T[2];
    assign Q[3]=T[3];

    assign UTC = Q[0]&Q[1]&Q[2]&Q[3];
    assign DTC = ~Q[0]&~Q[1]&~Q[2]&~Q[3];

endmodule
```