Anh Nguyen

017623292

12/05/2019

CECS 361 Fall 2019

Project : Completed Pong Game

Table of Contents

Project Description: 3

Top Level Block Diagram 5

Top Level Block Diagram (detail) 6

Debounce State Machine 7

Source code 8

Project Description:

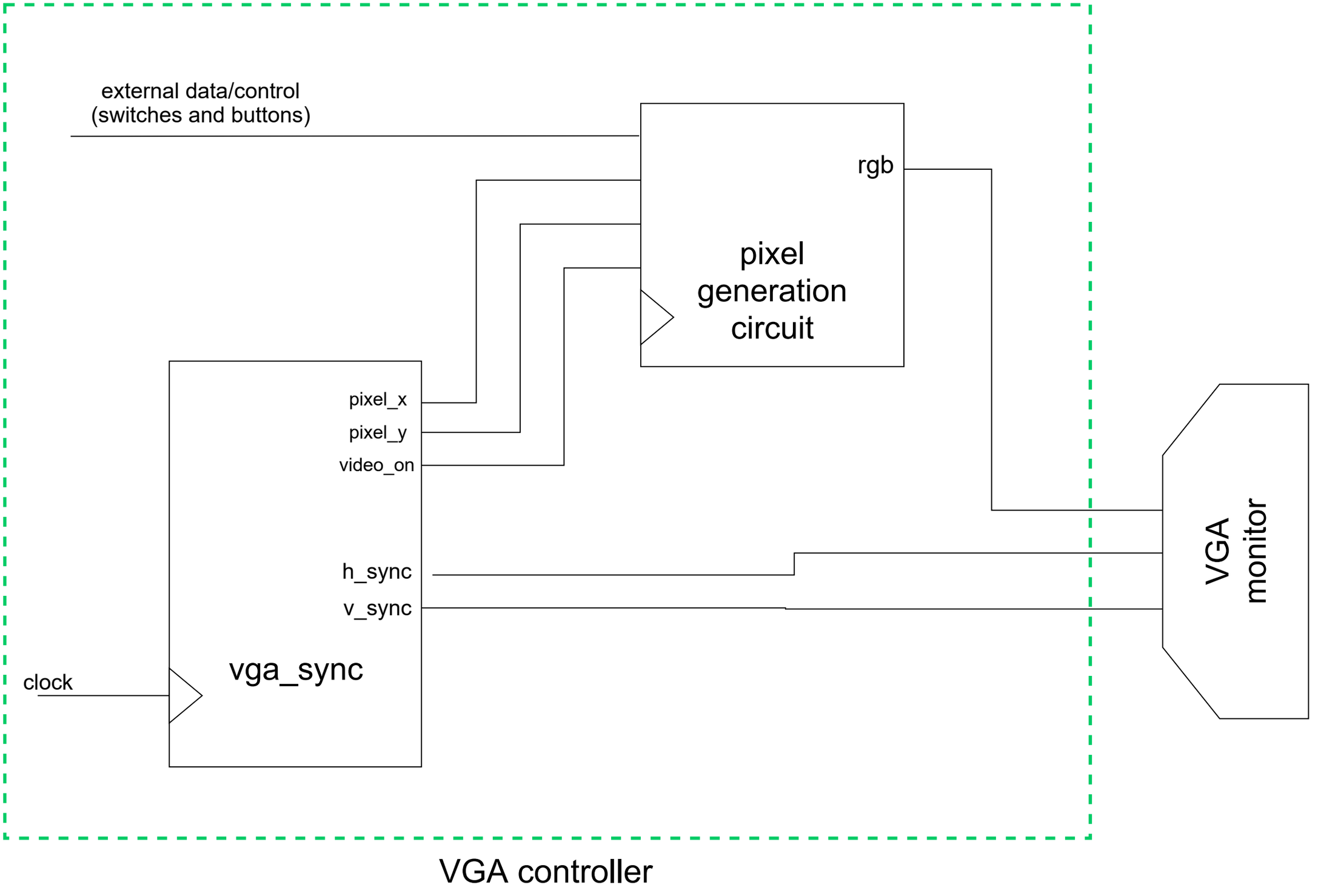
The purpose of this lab was to finalize all the previous labs and create a complete Pong game with the animation of the ball and paddles. The circuit used fixed pixel generator to create a ball, top and bottom border as well as two paddles so that two players can play the game at the same time.

The lab combined parts and modules from the previous lab. For example, VGA synchronization module. The VGA synchronization block includes the vga_sync block and the pixel generation block, one Asynchronous In Synchronous Out module to produce one reset signal for all other modules. A clock divider was also created to act as an enable for the h_sync and v_sync block. The h_sync and v_sync blocks then function as two counters which scan through the entire screen and produce video signal which later be processed via a pixel generation circuit. The circuit then creates RGB colors and projects to the screen using input from switches. User can play with the switches to make a unique combination that creates a specific color of the wall borders, the paddles as well as the ball.

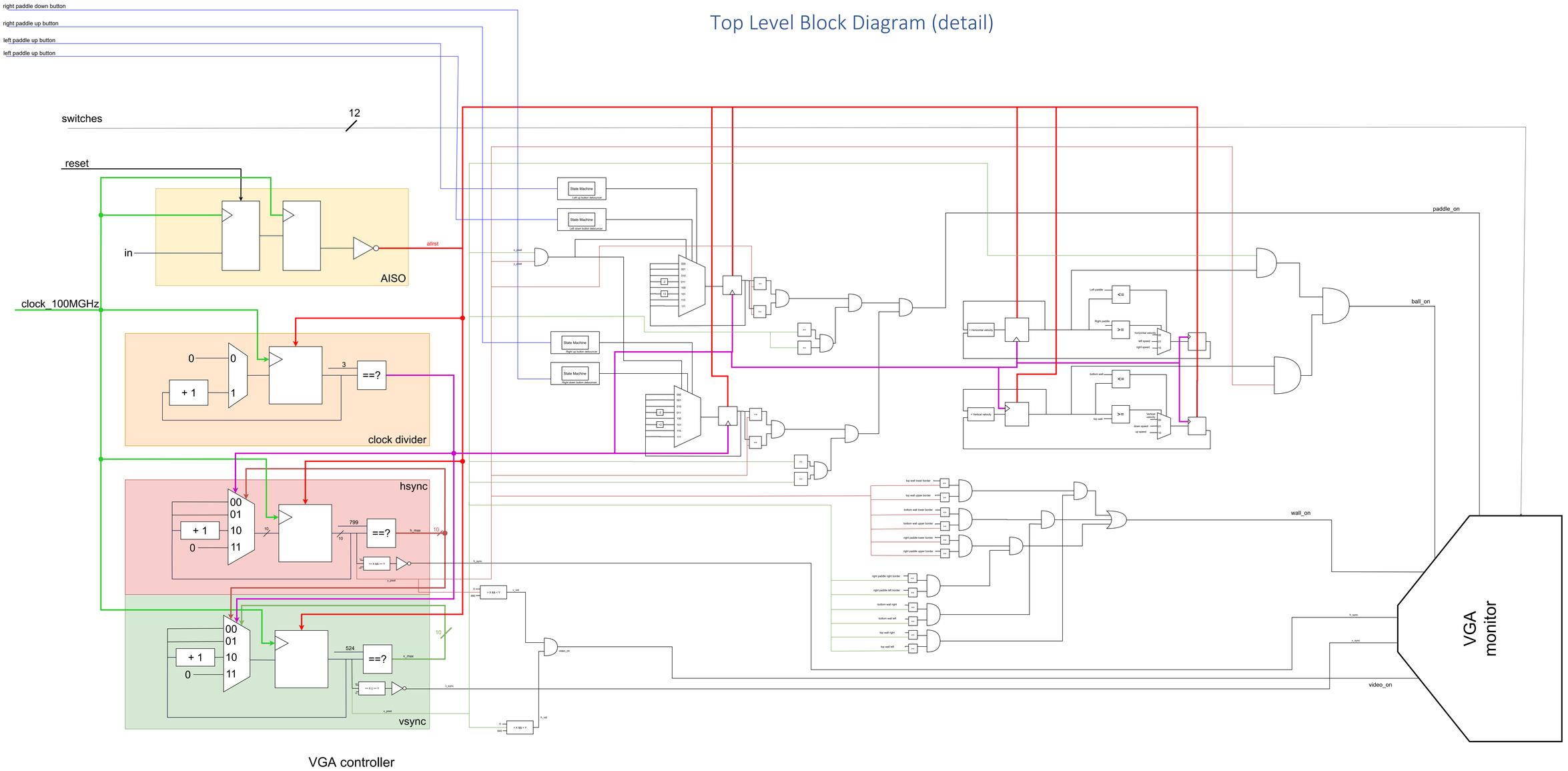
The movement of the paddles is created by first debouncing the buttons that used to control the paddles. After that, while scanning the screen at the same refresh rate as the monitor, it updates the location of the paddles and projects it onto the screen which is controlled by the players. By using some predefined values such as the position of the edges of the ball, the borders, the paddles' edges, the animation that when ever the ball hits the paddles and the borders, it bounces back to the opposite direction, was created. The ball's speed and directions of movement can be adjusted to create all kinds of variation for the game. In addition, to finish the logic of the game, the ball is put in the middle of the screen as a player fails to hit the ball and let it touch the side walls. A signal defined as "rescan" keeps track of the debounce button signal and check if it's active. If any buttons' signal is active, it moves by 2 pixels/scan in the corresponding direction.

To sum up, all the modules is connected to a top level module. The module instantiates lower module and outputs the signals: rgb, h_sync and v_sync. h_sync and v_sync told the board at what rate to refresh. RGB indicated when and where to grab color signals from. User can use the switch to change the color of the wall, paddles and the ball.

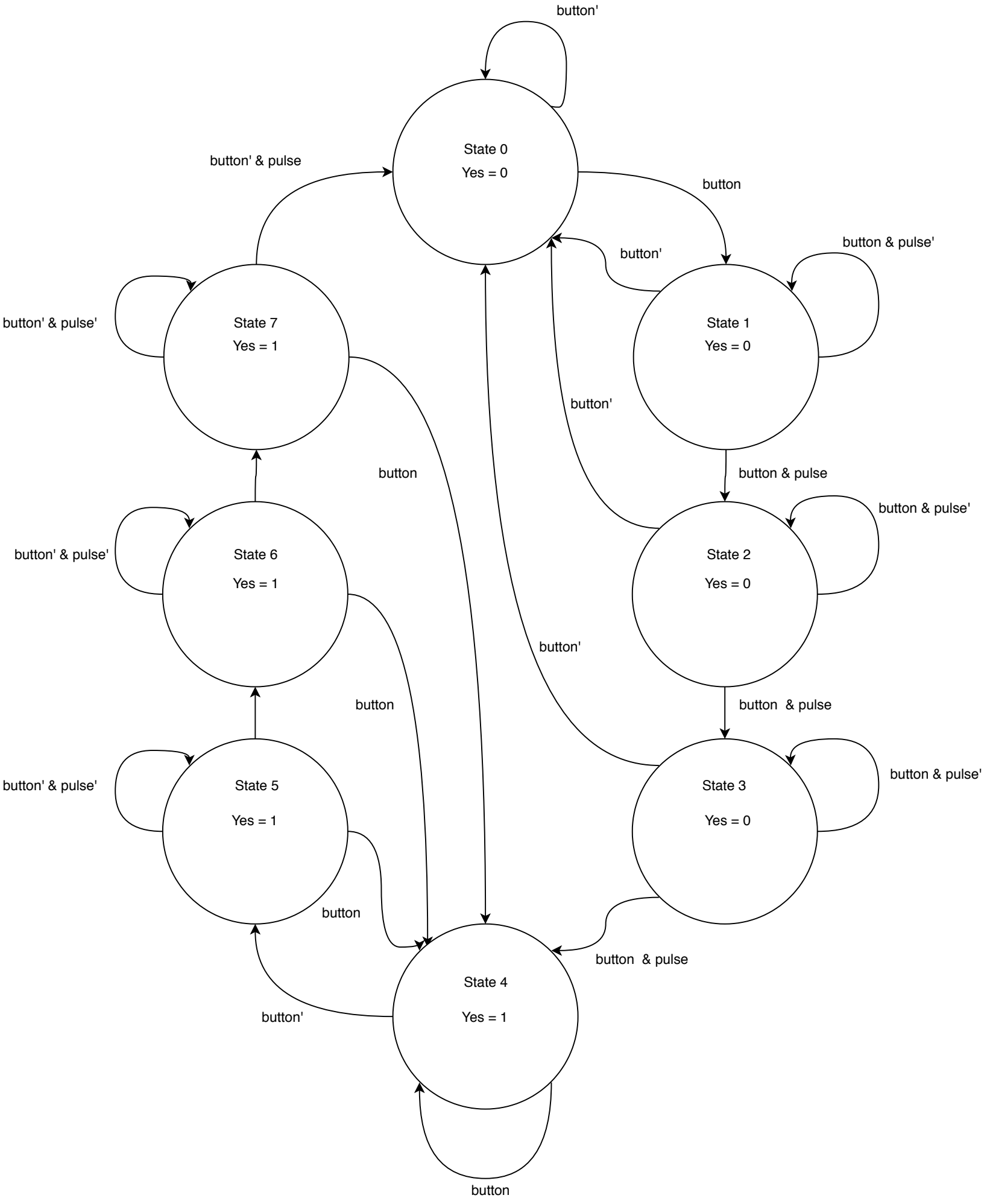
Top Level Block Diagram



Top Level Block Diagram (detail)



Debounce State Machine



```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////////////////////
3   //    File name: topLevelModule                                                 //
4   //                                                                              //
5   //    Created by: Anh Nguyen on 10/17/19                                        //
6   //    Copyright @2019 Anh Nguyen. All rights reserved.                          //
7   //                                                                              //
8   //    In submitting this file for class work at CSULB                           //
9   //    I am confirming that this is my work and the work                         //
10  //    of no one else. In submitting this code I acknowledge                     //
11  //    that plagiarism in student project work is subject to                     //
12  //    dismissal from the class.                                                 //
13  //                                                                              //
14  //////////////////////////////////////////////////////////////////////////////////
15  module topLevelMod(clk,rst,sw,hscan,vscan,vgaR, vgaG, vgaB,upButton, downButton,
    rightUpButton,rightDownButton);
16     input          clk,rst,upButton, downButton, rightUpButton, rightDownButton;
17     input [11:0]   sw;
18     wire           allclk,allrst;
19     output wire       hscan, vscan;
20     output wire [3:0]    vgaR, vgaG, vgaB;
21     //wire [9:0] pixel_x,pixel_y;
22
23     //AISO
24     AISO resetall     (.rst(rst), .clk(clk), .reset(allrst));
25
26     //clock divider
27     clkdiv divider    (.clk(clk), .rst(allrst), .clk_25MGHz(allclk));
28
29     //vga_top
30     vga_top vga       (.clk(clk), .rst(allrst), .select(allclk), .sw(sw[11:0]), .hsync(
    hscan),
31                         .vsync(vscan), .rgb({vgaR[3:0],vgaG[3:0],vgaB[3:0]}), .upButton(
    upButton), .downButton(downButton),.rightUpButton(rightUpButton), .rightDownButton(
    rightDownButton));
32
33  endmodule
34
```

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////////////////////
3   //    File name: Asynchronous In/ Synchronous Out Reset                         //
4   //                                                                              //
5   //    Created by: Anh Nguyen on 10/17/19                                        //
6   //    Copyright @2019 Anh Nguyen. All rights reserved.                          //
7   //                                                                              //
8   //    In submitting this file for class work at CSULB                           //
9   //    I am confirming that this is my work and the work                         //
10  //    of no one else. In submitting this code I acknowledge                     //
11  //    that plagiarism in student project work is subject to                     //
12  //    dismissal from the class.                                                 //
13  //                                                                              //
14  //////////////////////////////////////////////////////////////////////////////////
15
16  module AISO(clk, rst, reset);
17     input wire        clk, rst;
18     output wire       reset;
19     reg               q1,q2;
20
21     always @(posedge clk, posedge rst)
22        if (rst)
23           q1 <= 1'b0;
24        else
25           {q1,q2} <= {1'b1,q1};
26
27     assign reset = ~q2;
28
29  endmodule
30
31
```

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////////////////////
3   //    File name: clock divider                                                  //
4   //                                                                              //
5   //    Created by: Anh Nguyen on 10/17/19                                        //
6   //    Copyright @2019 Anh Nguyen. All rights reserved.                          //
7   //                                                                              //
8   //    In submitting this file for class work at CSULB                           //
9   //    I am confirming that this is my work and the work                         //
10  //    of no one else. In submitting this code I acknowledge                     //
11  //    that plagiarism in student project work is subject to                     //
12  //    dismissal from the class.                                                 //
13  //                                                                              //
14  //////////////////////////////////////////////////////////////////////////////////
15  module clkdiv(clk, rst, clk_25MGHz);
16     input wire        clk, rst;
17     reg     [1:0]     count,ncount;
18     output wire       clk_25MGHz;
19
20     always @(posedge clk, posedge rst)
21        if (rst)
22           count <= 2'b0;
23        else
24           count <= count + 2'b1;
25
26     assign clk_25MGHz =(count == 2'b11);
27
28  endmodule
29
```

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////////////////////
3   //    File name: VGA controller                                                 //
4   //                                                                              //
5   //    Created by: Anh Nguyen on 10/17/19                                        //
6   //    Copyright @2019 Anh Nguyen. All rights reserved.                          //
7   //                                                                              //
8   //    In submitting this file for class work at CSULB                           //
9   //    I am confirming that this is my work and the work                         //
10  //    of no one else. In submitting this code I acknowledge                     //
11  //    that plagiarism in student project work is subject to                     //
12  //    dismissal from the class.                                                 //
13  //                                                                              //
14  //////////////////////////////////////////////////////////////////////////////////
15  module vga_top(clk,rst,select,hsync,vsync,sw,rgb,upButton,downButton,rightUpButton,
    rightDownButton);
16     input                clk,rst,select,upButton,downButton,rightUpButton,
    rightDownButton;
17     input       [11:0]   sw;
18     output reg  [11:0]   rgb;
19     output wire          hsync,vsync;
20     wire [9:0]  x_pixel, y_pixel;
21     wire on;
22
23     reg         [11:0]   rgbr;
24
25     allsync hv (.clk(clk), .rst(rst), .select(select), .hcount(x_pixel), .vcount(
    y_pixel), .h_sync(hsync), .v_sync(vsync), .video_on(on));
26
27     always@ (posedge clk, posedge rst)
28        if (rst)
29           rgbr <= 12'b0;
30        else
31           rgbr <= sw;
32
33     // Paddle position register
34     reg [9:0] topOfPaddle, nTopOfPaddle, topOfRightPaddle, ntopOfRightPaddle;
35
36     // Debounced buttons
37     wire upDetected, downDetected, rightUpDetected, rightDownDetected;
38
39     // Update pixels based on this signal
40     wire rescan;
41
42     // Predefined paddle max and min position
43     localparam paddleBottomEdge = 475;
44     localparam paddleTopEdge = 9;
45
46     // Paddles'length
47     localparam paddleHeight = 70;
48
49     // Paddles'sides position
50     // Left
51     localparam paddleLeftSide = 10;
52     localparam paddleRightSide = 16;
53     // Right
54     localparam rightPaddleLeftSide = 624;
```

```
55      localparam rightPaddleRightSide = 630;
56
57      // Top border
58      localparam topWallUpperBorder = 3;
59      localparam topWallLowerBorder = 9;
60
61      // Bottom border
62      localparam bottomWallUpperBorder = 472;
63      localparam bottomWallLowerBorder = 478;
64
65      // Right border
66      localparam rightWallLeftBorder = 624;
67      localparam rightWallRightBorder = 630;
68
69      // Detect ball's border
70      wire [9:0] ballLeftBorder;
71      wire [9:0] ballRightBorder;
72      wire [9:0] ballUpperBorder;
73      wire [9:0] ballLowerBorder;
74
75      // Ball's dimension
76      localparam ballHeight = 6;
77      localparam ballWidth  = 6;
78
79      // Register to calculate ball's location
80      reg  [9:0] ballUpperReg;
81      wire [9:0] ballUpperNext;
82      reg  [9:0] ballRightReg;
83      wire [9:0] ballRightNext;
84
85      // Ball goes sideway
86      reg [9:0] ballHVeloReg;
87      reg [9:0] ballHVeloNext;
88
89      // Ball goes up or down
90      reg [9:0] ballVVeloReg;
91      reg [9:0] ballVVeloNext;
92
93      // Predefined speed
94      localparam ballSpeedLeft  = -0.5;
95      localparam ballSpeedRight =  0.5;
96      localparam ballSpeedUp    = -0.5;
97      localparam ballSpeedDown  =  0.5;
98
99      // Signal That Indicates When The Pixel is On The Ball Pixel Values
100     wire ballon;
101
102     // Detecting the upper/lower border of the ball
103     assign ballUpperBorder = ballUpperReg;
104     assign ballLowerBorder = ballUpperBorder + ballHeight;
105
106     // Detecting the side border of the ball
107     assign ballRightBorder = ballRightReg;
108     assign ballLeftBorder = ballRightReg - ballWidth;
109
110     // The pixels match the ball pixels
111     assign ballon = (x_pixel >= ballLeftBorder && x_pixel <= ballRightBorder) && (
```

```
      y_pixel <= ballLowerBorder && y_pixel >= ballUpperBorder);
112
113      // Rescan scanning start position
114      assign rescan = (y_pixel == 481) && (x_pixel == 0);
115
116      // Debounce the buttons on the board
117      // Left paddle
118      upDebounce up (.clk(clk),.rst(rst),.pulse(select),.button(upButton), .yes(
      upDetected));
119      downDebounce down (.clk(clk),.rst(rst),.pulse(select),.button(downButton), .yes(
      downDetected));
120
121      // Right paddle
122      upDebounce rightUp (.clk(clk),.rst(rst),.pulse(select),.button(rightUpButton), .yes
      (rightUpDetected));
123      downDebounce rightDown (.clk(clk),.rst(rst),.pulse(select),.button(rightDownButton
      ), .yes(rightDownDetected));
124
125
126      // If reset is off, the left paddle get its next value
127      always @(posedge clk , posedge rst)
128         if (rst)
129            topOfPaddle <= 10'd249;
130         else
131            topOfPaddle <= nTopOfPaddle;
132
133      // If reset is off, the right paddle get its next value
134      always @(posedge clk , posedge rst)
135         if (rst)
136            topOfRightPaddle <= 10'd249;
137         else
138            topOfRightPaddle <= ntopOfRightPaddle;
139
140      // Left paddle movement
141      always@(*)
142         case({upDetected,downDetected,rescan})
143
144            3'b000: nTopOfPaddle = topOfPaddle;       // If rescan is inactive, paddle
      stays still
145            3'b001: nTopOfPaddle = topOfPaddle;       // If rescan is active and no
      button is pressed, paddle stays still
146            3'b010: nTopOfPaddle = topOfPaddle;       // If rescan is inactive and down
      is pressed, paddle stays still
147            3'b100: nTopOfPaddle = topOfPaddle;       // If rescan is inactive and up is
      pressed, paddle stays still
148            3'b110: nTopOfPaddle = topOfPaddle;       // If rescan is inactive and both
      buttons are pressed, paddle stays still
149            3'b111: nTopOfPaddle = topOfPaddle;       // If rescan is active and both
      buttons are pressed paddle stays still
150
151            // If rescan is active and down is pressed, paddle moves down
152            3'b011:
153               begin
154                  nTopOfPaddle = topOfPaddle + 10'd2;
155
156                  // Paddle cannot go beyond bottom wall
157                  if(topOfPaddle + paddleHeight >= paddleBottomEdge)
```

```
158                     nTopOfPaddle = paddleBottomEdge - paddleHeight;
159                end
160
161             // If rescan is active and up is pressed, paddle moves up
162             3'b101:
163                begin
164                   nTopOfPaddle = topOfPaddle - 10'd2;
165
166                   // Paddle cannot go beyond top wall
167                   if (topOfPaddle <= paddleTopEdge)
168                      nTopOfPaddle = paddleTopEdge;
169                end
170             default: nTopOfPaddle = topOfPaddle;
171          endcase
172
173       // Right paddle movement
174       always@(*)
175          case({rightUpDetected,rightDownDetected,rescan})
176             3'b000: ntopOfRightPaddle = topOfRightPaddle;         // If rescan is
      inactive, paddle stays still
177             3'b001: ntopOfRightPaddle = topOfRightPaddle;         // If rescan is active
      and no buttons are pressed, paddle stays still
178             3'b010: ntopOfRightPaddle = topOfRightPaddle;         // If rescan is
      inactive and down is pressed, paddle stays still
179             3'b100: ntopOfRightPaddle = topOfRightPaddle;         // If rescan is
      inactive and up is pressed, paddle stays still
180             3'b110: ntopOfRightPaddle = topOfRightPaddle;         // If rescan is
      inactive and both buttons are pressed, paddle stays still
181             3'b111: ntopOfRightPaddle = topOfRightPaddle;         // If rescan is active
      and both buttons are pressed paddle stays still
182
183             // If rescan is active and down is pressed, paddle moves down
184             3'b011:
185                begin
186                   ntopOfRightPaddle = topOfRightPaddle + 10'd2;
187
188                   //Paddle Cannot Go Lower Than Bottom Wall
189                   if(topOfRightPaddle + paddleHeight >= paddleBottomEdge)
190                      ntopOfRightPaddle = paddleBottomEdge - paddleHeight;
191                end
192
193             // If rescan Active and Up Pressed, Paddle Goes Up
194             3'b101:
195                begin
196                   ntopOfRightPaddle = topOfRightPaddle - 10'd2;
197
198                   // Paddle Cannot Go Higher Than Top Wall
199                   if (topOfRightPaddle <= paddleTopEdge)
200                      ntopOfRightPaddle = paddleTopEdge;
201                end
202             default: ntopOfRightPaddle = topOfRightPaddle;
203          endcase
204
205       // Flop to update registers at every posedge
206       always @(posedge clk, posedge rst)
207          // If reset is on or the paddles does not hit the ball, reset the position and
      speed
```

```
208         if((rst) || (ballLeftBorder < paddleLeftSide))
209            begin
210               ballRightReg <= 317;
211               ballUpperReg <= 237;
212               ballHVeloReg <= 1;
213               ballVVeloReg <= -1;
214            end
215
216         // Continue the game otherwise
217         else
218            begin
219               ballRightReg <= ballRightNext;
220               ballUpperReg <= ballUpperNext;
221               ballHVeloReg <= ballHVeloNext;
222               ballVVeloReg <= ballVVeloNext;
223            end
224
225      // Next horizontal position = previous position + velocity
226      assign ballRightNext = (rescan) ? ballRightReg + ballHVeloReg : ballRightReg;
227
228      // Next vertical position = previous position + velocity
229      assign ballUpperNext = (rescan) ? ballUpperReg + ballVVeloReg : ballUpperReg;
230
231      // Direction detection
232      always@(*)
233         begin
234            ballHVeloNext = ballHVeloReg;
235            ballVVeloNext = ballVVeloReg;
236
237            // Ball goes down when hitting top wall
238            if(ballUpperBorder <= topWallLowerBorder)
239               ballVVeloNext = ballSpeedDown;
240
241            // Ball goes up when hitting bottom wall
242            else if(ballLowerBorder >= bottomWallUpperBorder)
243               ballVVeloNext = ballSpeedUp;
244
245            // Ball goes left when hitting right paddle
246            else if(((ballLeftBorder) >= rightPaddleLeftSide) && (ballLeftBorder <=
      rightPaddleRightSide)   && ((ballUpperBorder)>= topOfRightPaddle) && ((ballUpperBorder
       <= topOfRightPaddle + paddleHeight)))
247               begin
248                  ballHVeloNext = ballSpeedLeft;
249               end
250
251            // Ball goes right when hitting left paddle
252            else if(((ballLeftBorder) <= paddleRightSide) && (ballLeftBorder >=
      paddleLeftSide) && ((ballUpperBorder)>= topOfPaddle) && ((ballUpperBorder <=
      topOfPaddle + paddleHeight)))
253               begin
254                  ballHVeloNext = ballSpeedRight;
255               end
256         end
257
258      always@ (*)
259         //top
260         if ((x_pixel >= 2 && x_pixel <= rightWallRightBorder) && (y_pixel >=
```

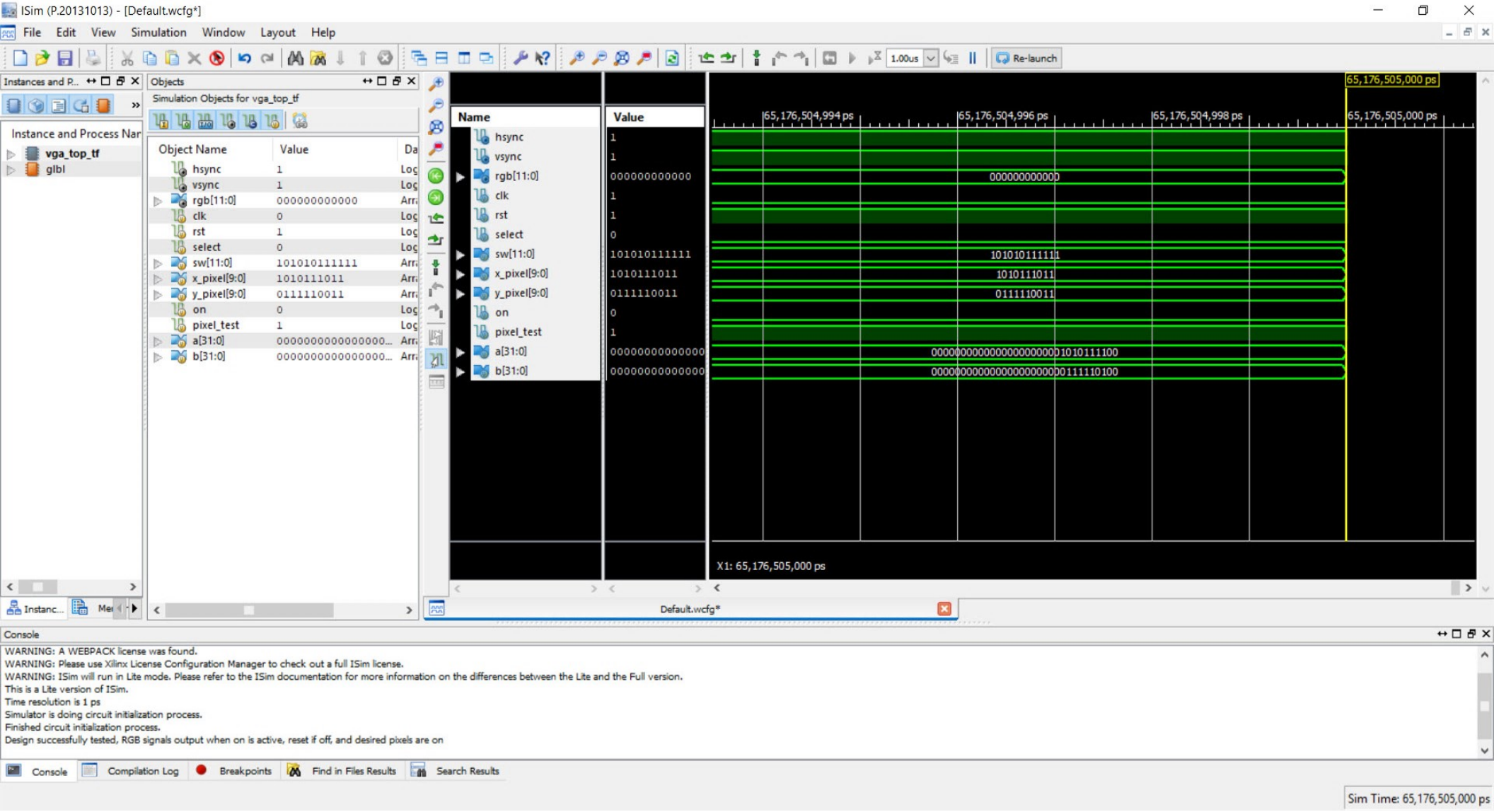
```
      topWallUpperBorder && y_pixel <= topWallLowerBorder))
261            rgb = rgbr;
262         //bottom
263         else if ((x_pixel >= 2 && x_pixel <= rightWallRightBorder) && (y_pixel >=
      bottomWallUpperBorder && y_pixel <= bottomWallLowerBorder))
264            rgb = rgbr;
265         //left
266         else if ((x_pixel >= paddleLeftSide && x_pixel <= paddleRightSide) && (y_pixel
      >= topOfPaddle) && (y_pixel <= topOfPaddle + paddleHeight))
267            rgb = rgbr;
268         //right
269         else if ((x_pixel >= rightPaddleLeftSide && x_pixel <= rightPaddleRightSide) &&
      (y_pixel >= topOfRightPaddle) && (y_pixel <= topOfRightPaddle + paddleHeight))
270            rgb = rgbr;
271         //ball
272         else if ((ballon) && (on))
273            rgb = rgbr;
274         else if(on)
275            rgb = 12'h0FF;
276         else
277            rgb = 12'h000;
278   endmodule
279
```

VGA Teset Fixture

```
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
//  File name: Test fixture for VGA controller                                //
//  Created by: Anh Nguyen on 11/07/19                                        //
//  Copyright @2019 Anh Nguyen. All rights reserved.                          //
//  In submitting this file for class work at CSULB                           //
//  I am confirming that this is my work and the work                         //
//  of no one else. In submitting this code I acknowledge                     //
//  that plagiarism in student project work is subject to                     //
//  dismissal from the class.                                                 //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////

 module vga_top_tf;

 // Inputs
 reg clk;
 reg rst;
 reg select;
 reg [11:0] sw;
 reg [9:0] x_pixel;
 reg [9:0] y_pixel;
 reg on;

 // Outputs
 wire hsync;
 wire vsync;
 wire [11:0] rgb;

 // Instantiate the Unit Under Test (UUT)
 vga_top uut (
     .clk(clk),
     .rst(rst),
     .select(select),
     .hsync(hsync),
     .vsync(vsync),
     .sw(sw),
     .rgb(rgb),
     .vOn(on)
 );

 reg pixel_test;
 integer a,b;
```

```
    //100MHz clock
    always #5 clk = ~clk;

    initial begin
        // Initialize Inputs
        clk = 0;
        rst = 1;
        select = 0;
        sw = 12'habf;
        x_pixel = 0;
        y_pixel = 0;
        on = 0;
        // Wait 10 ns for global reset to finish
        #10;

        rst = 0;
        on = 1;

        // Keep videoOn On for 100ns
        #100;

for(a = 0; a < 700; a = a + 1)
    begin
    x_pixel = a;
    #2;
    for(b = 0; b< 500; b = b + 1)
        begin
        y_pixel = b;
        #2;

        // Top border
        if((x_pixel >=2 && x_pixel <= 638) && (y_pixel >= 3 && y_pixel <= 5) && (on)
         && (rgb ==0))
            $display("Top border failed to display. Please check again");

        // Bottom border
        else if((x_pixel >=2 && x_pixel <= 638) && (y_pixel >= 475 && y_pixel <= 477)
       && (on) && (rgb ==0))
       $display("Bottom border failed to display. Please check again");

        // Left paddle
        else if((x_pixel >= 10 && x_pixel <= 16) && (y_pixel >= 231 && y_pixel <=
       249) && (on) && (rgb ==0))
            $display("Left paddle failed to display. Please check again");
```

```
        // Right paddle
        else if((x_pixel >= 624 && x_pixel <= 630) && (y_pixel >= 231 && y_pixel <=
        249) && (on) && (rgb ==0))
            $display("Right paddle failed to display. Please check again");

        // Ball
        else if((x_pixel >= 317 && x_pixel <= 323) && (y_pixel >= 237 && y_pixel <=
        243) && (on) && (rgb ==0))
            $display("Ball failed to display. Please check again");

        // Background
         else if((on) && (rgb != 12'h0FF))
            $display ("Background color failed to display. Please check again");

        end
    end

    // Turn on reset and set video on to 0
    rst = 1;
    on = 0;

    // Wait 100ns
    #100;

    // Make sure rgb is 0
    if(rgb == 12'b0 && on == 0)
        pixel_test = 1'b1;

    // Wait 100ns
    #100;

    // Display result
    if(pixel_test)
        $display("Design successfully tested, RGB signals output when on is active,
    reset if off, and desired pixels are on");
  end

endmodule
```


ISim (P.20131013) - [Default.wcfg*]
File Edit View Simulation Window Layout Help
Re-launch
Instances and P...
Instance and Process Nar
vga_top_tf
glbl
Objects
Simulation Objects for vga_top_tf
Object Name
Value
hsync 1
vsync 1
rgb[11:0] 000000000000
clk 0
rst 1
select 0
sw[11:0] 101010111111
x_pixel[9:0] 1010111011
y_pixel[9:0] 0111110011
on 0
pixel_test 1
a[31:0]
b[31:0]
Name
Value
65,176,504,994 ps
65,176,504,996 ps
65,176,504,998 ps
65,176,505,000 ps
X1: 65,176,505,000 ps
Default.wcfg*
Console
WARNING: A WEBPACK license was found.
WARNING: Please use Xilinx License Configuration Manager to check out a full ISim license.
WARNING: ISim will run in Lite mode. Please refer to the ISim documentation for more information on the differences between the Lite and the Full version.
This is a Lite version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
Design successfully tested, RGB signals output when on is active, reset if off, and desired pixels are on
Console
Compilation Log
Breakpoints
Find in Files Results
Search Results
Sim Time: 65,176,505,000 ps

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////////////////////
3   //    File name: Horizontal Scan and Vertical Scan                              //
4   //                                                                              //
5   //    Created by: Anh Nguyen on 10/15/19                                        //
6   //    Copyright @2019 Anh Nguyen. All rights reserved.                          //
7   //                                                                              //
8   //    In submitting this file for class work at CSULB                           //
9   //    I am confirming that this is my work and the work                         //
10  //    of no one else. In submitting this code I acknowledge                     //
11  //    that plagiarism in student project work is subject to                     //
12  //    dismissal from the class.                                                 //
13  //                                                                              //
14  //////////////////////////////////////////////////////////////////////////////////
15
16  module allsync(clk,rst,select,hcount,vcount,h_sync,v_sync,video_on);
17     input          clk,rst,select;
18     output reg     h_sync,v_sync;
19     output reg [9:0] hcount,vcount;
20     reg            hSyncReg,vSyncReg;
21     reg [9:0]      hncount,vncount;
22     output wire    video_on;
23     wire           h_count,v_count,h_vid,v_vid;
24
25     //Horizontal scan
26     //count from 0 to 799
27     assign h_count =(hcount == 10'd799);
28
29     always@ (posedge clk, posedge rst)
30        if (rst)
31           hcount <= 10'b0;
32        else
33           hcount <= hncount;
34
35     //mux 4:1
36     always@ (*)
37        case({select,h_count})
38           2'b00: hncount = hcount;
39           2'b01: hncount = hcount;
40           2'b10: hncount = hcount + 10'b1;
41           2'b11: hncount = 10'b0;
42        endcase
43
44     //horizontal sync signal is low active from 656-751
45     always@ (posedge clk)
46        h_sync = ~((hcount >= 10'd656) && (hcount <= 10'd751));
47
48     //horizontal video signal is high active from 0-639
49     assign h_vid = (hcount < 10'd640);
50
51     //Vertical Scan
52     //count from 0 to 524
53     assign v_count =(vcount == 10'd524);
54
55     always@ (posedge clk, posedge rst)
56        if (rst)
57           vcount <= 10'b0;
```

```
58          else
59             vcount <= vncount;
60
61       //mux 4:1
62       always@ (*)
63          case({{select && h_count},v_count})
64             2'b00: vncount = vcount;
65             2'b01: vncount = vcount;
66             2'b10: vncount = vcount + 10'b1;
67             2'b11: vncount = 10'b0;
68          endcase
69
70       //vertical sync signal is low active from 490 to 491
71       always@ (posedge clk)
72          v_sync = ~((vcount == 10'd490) || (vcount == 10'd491));
73
74       //vertical video on signal is high active from 0-479
75       assign v_vid = (vcount < 10'd480);
76
77       //Video ON/OFF
78       //video on signal is high active when both horizontal and vertical video signal
     are active
79       assign video_on = (h_vid && v_vid);
80
81    endmodule
82
```

```
1   `timescale 1ns / 1ps
2   ////////////////////////////////////////////////////////////////////////////////
3   //    File name: Paddle Up Button Debouncer                                   //
4   //                                                                            //
5   //    Created by: Anh Nguyen on 12/03/19                                      //
6   //    Copyright @2019 Anh Nguyen. All rights reserved.                        //
7   //                                                                            //
8   //    In submitting this file for class work at CSULB                         //
9   //    I am confirming that this is my work and the work                       //
10  //    of no one else. In submitting this code I acknowledge                   //
11  //    that plagiarism in student project work is subject to                   //
12  //    dismissal from the class.                                               //
13  //                                                                            //
14  ////////////////////////////////////////////////////////////////////////////////
15  module upDebounce(clk,rst,pulse,button,yes);
16     input clk,rst,pulse,button;
17     output reg yes;
18     reg [2:0] state, nState;
19     reg nYes;
20
21     always@(posedge clk, posedge rst)
22        if(rst)
23           {state,yes} <= 4'b0;
24        else
25           {state,yes} <= {nState, nYes};
26
27     always@(*)
28        case({state,pulse,button})
29           5'b000_00: {nState, nYes} = 4'b000_0;
30           5'b000_01: {nState, nYes} = 4'b001_0;
31           5'b000_10: {nState, nYes} = 4'b000_0;
32           5'b000_11: {nState, nYes} = 4'b001_0;
33           5'b001_00: {nState, nYes} = 4'b000_0;
34           5'b001_01: {nState, nYes} = 4'b001_0;
35           5'b001_10: {nState, nYes} = 4'b000_0;
36           5'b001_11: {nState, nYes} = 4'b010_0;
37           5'b010_00: {nState, nYes} = 4'b000_0;
38           5'b010_01: {nState, nYes} = 4'b010_0;
39           5'b010_10: {nState, nYes} = 4'b000_0;
40           5'b010_11: {nState, nYes} = 4'b011_0;
41           5'b011_00: {nState, nYes} = 4'b000_0;
42           5'b011_01: {nState, nYes} = 4'b011_0;
43           5'b011_10: {nState, nYes} = 4'b000_0;
44           5'b011_11: {nState, nYes} = 4'b100_1;
45           5'b100_00: {nState, nYes} = 4'b101_1;
46           5'b100_01: {nState, nYes} = 4'b100_1;
47           5'b100_10: {nState, nYes} = 4'b101_1;
48           5'b100_11: {nState, nYes} = 4'b100_1;
49           5'b101_00: {nState, nYes} = 4'b101_1;
50           5'b101_01: {nState, nYes} = 4'b100_1;
51           5'b101_10: {nState, nYes} = 4'b110_1;
52           5'b101_11: {nState, nYes} = 4'b100_1;
53           5'b110_00: {nState, nYes} = 4'b110_1;
54           5'b110_01: {nState, nYes} = 4'b100_1;
55           5'b110_10: {nState, nYes} = 4'b111_1;
56           5'b110_11: {nState, nYes} = 4'b100_1;
57           5'b111_00: {nState, nYes} = 4'b111_1;
```

```
58              5'b111_01: {nState, nYes} = 4'b100_1;
59              5'b111_10: {nState, nYes} = 4'b000_0;
60              5'b111_11: {nState, nYes} = 4'b100_1;
61              default: {nState, nYes} = 4'b000_0;
62          endcase
63
64
65
66
67
68
69      endmodule
70
71
```

```
1   `timescale 1ns / 1ps
2   ////////////////////////////////////////////////////////////////////////////////
3   //    File name:  Paddle Down Button Debouncer                                //
4   //                                                                            //
5   //    Created by: Anh Nguyen on 12/03/19                                      //
6   //    Copyright @2019 Anh Nguyen. All rights reserved.                        //
7   //                                                                            //
8   //    In submitting this file for class work at CSULB                         //
9   //    I am confirming that this is my work and the work                       //
10  //    of no one else. In submitting this code I acknowledge                   //
11  //    that plagiarism in student project work is subject to                   //
12  //    dismissal from the class.                                               //
13  //                                                                            //
14  ////////////////////////////////////////////////////////////////////////////////
15  module downDebounce(clk,rst,pulse,button,yes);
16     input clk,rst,pulse,button;
17     output reg yes;
18     reg [2:0] state, nState;
19     reg nYes;
20
21     always@(posedge clk, posedge rst)
22        if(rst)
23           {state,yes} <= 4'b0;
24        else
25           {state,yes} <= {nState, nYes};
26
27     always@(*)
28        case({state,pulse,button})
29           5'b000_00: {nState, nYes} = 4'b000_0;
30           5'b000_01: {nState, nYes} = 4'b001_0;
31           5'b000_10: {nState, nYes} = 4'b000_0;
32           5'b000_11: {nState, nYes} = 4'b001_0;
33           5'b001_00: {nState, nYes} = 4'b000_0;
34           5'b001_01: {nState, nYes} = 4'b001_0;
35           5'b001_10: {nState, nYes} = 4'b000_0;
36           5'b001_11: {nState, nYes} = 4'b010_0;
37           5'b010_00: {nState, nYes} = 4'b000_0;
38           5'b010_01: {nState, nYes} = 4'b010_0;
39           5'b010_10: {nState, nYes} = 4'b000_0;
40           5'b010_11: {nState, nYes} = 4'b011_0;
41           5'b011_00: {nState, nYes} = 4'b000_0;
42           5'b011_01: {nState, nYes} = 4'b011_0;
43           5'b011_10: {nState, nYes} = 4'b000_0;
44           5'b011_11: {nState, nYes} = 4'b100_1;
45           5'b100_00: {nState, nYes} = 4'b101_1;
46           5'b100_01: {nState, nYes} = 4'b100_1;
47           5'b100_10: {nState, nYes} = 4'b101_1;
48           5'b100_11: {nState, nYes} = 4'b100_1;
49           5'b101_00: {nState, nYes} = 4'b101_1;
50           5'b101_01: {nState, nYes} = 4'b100_1;
51           5'b101_10: {nState, nYes} = 4'b110_1;
52           5'b101_11: {nState, nYes} = 4'b100_1;
53           5'b110_00: {nState, nYes} = 4'b110_1;
54           5'b110_01: {nState, nYes} = 4'b100_1;
55           5'b110_10: {nState, nYes} = 4'b111_1;
56           5'b110_11: {nState, nYes} = 4'b100_1;
57           5'b111_00: {nState, nYes} = 4'b111_1;
```

```
58              5'b111_01: {nState, nYes} = 4'b100_1;
59              5'b111_10: {nState, nYes} = 4'b000_0;
60              5'b111_11: {nState, nYes} = 4'b100_1;
61              default: {nState, nYes} = 4'b000_0;
62          endcase
63
64
65
66
67
68
69      endmodule
70
```

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////////////////////
3   //    File name: Paddle Up Button Debouncer                                     //
4   //                                                                              //
5   //    Created by: Anh Nguyen on 12/03/19                                        //
6   //    Copyright @2019 Anh Nguyen. All rights reserved.                          //
7   //                                                                              //
8   //    In submitting this file for class work at CSULB                           //
9   //    I am confirming that this is my work and the work                         //
10  //    of no one else. In submitting this code I acknowledge                     //
11  //    that plagiarism in student project work is subject to                     //
12  //    dismissal from the class.                                                 //
13  //                                                                              //
14  //////////////////////////////////////////////////////////////////////////////////
15  module upDebounce(clk,rst,pulse,button,yes);
16     input clk,rst,pulse,button;
17     output reg yes;
18     reg [2:0] state, nState;
19     reg nYes;
20
21     always@(posedge clk, posedge rst)
22        if(rst)
23           {state,yes} <= 4'b0;
24        else
25           {state,yes} <= {nState, nYes};
26
27     always@(*)
28        case({state,pulse,button})
29           5'b000_00: {nState, nYes} = 4'b000_0;
30           5'b000_01: {nState, nYes} = 4'b001_0;
31           5'b000_10: {nState, nYes} = 4'b000_0;
32           5'b000_11: {nState, nYes} = 4'b001_0;
33           5'b001_00: {nState, nYes} = 4'b000_0;
34           5'b001_01: {nState, nYes} = 4'b001_0;
35           5'b001_10: {nState, nYes} = 4'b000_0;
36           5'b001_11: {nState, nYes} = 4'b010_0;
37           5'b010_00: {nState, nYes} = 4'b000_0;
38           5'b010_01: {nState, nYes} = 4'b010_0;
39           5'b010_10: {nState, nYes} = 4'b000_0;
40           5'b010_11: {nState, nYes} = 4'b011_0;
41           5'b011_00: {nState, nYes} = 4'b000_0;
42           5'b011_01: {nState, nYes} = 4'b011_0;
43           5'b011_10: {nState, nYes} = 4'b000_0;
44           5'b011_11: {nState, nYes} = 4'b100_1;
45           5'b100_00: {nState, nYes} = 4'b101_1;
46           5'b100_01: {nState, nYes} = 4'b100_1;
47           5'b100_10: {nState, nYes} = 4'b101_1;
48           5'b100_11: {nState, nYes} = 4'b100_1;
49           5'b101_00: {nState, nYes} = 4'b101_1;
50           5'b101_01: {nState, nYes} = 4'b100_1;
51           5'b101_10: {nState, nYes} = 4'b110_1;
52           5'b101_11: {nState, nYes} = 4'b100_1;
53           5'b110_00: {nState, nYes} = 4'b110_1;
54           5'b110_01: {nState, nYes} = 4'b100_1;
55           5'b110_10: {nState, nYes} = 4'b111_1;
56           5'b110_11: {nState, nYes} = 4'b100_1;
57           5'b111_00: {nState, nYes} = 4'b111_1;
```

```
58              5'b111_01: {nState, nYes} = 4'b100_1;
59              5'b111_10: {nState, nYes} = 4'b000_0;
60              5'b111_11: {nState, nYes} = 4'b100_1;
61              default: {nState, nYes} = 4'b000_0;
62          endcase
63
64
65
66
67
68
69      endmodule
70
71
```

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////////////////////
3   //    File name:  Paddle Down Button Debouncer                                  //
4   //                                                                              //
5   //    Created by: Anh Nguyen on 12/03/19                                        //
6   //    Copyright @2019 Anh Nguyen. All rights reserved.                          //
7   //                                                                              //
8   //    In submitting this file for class work at CSULB                           //
9   //    I am confirming that this is my work and the work                         //
10  //    of no one else. In submitting this code I acknowledge                     //
11  //    that plagiarism in student project work is subject to                     //
12  //    dismissal from the class.                                                 //
13  //                                                                              //
14  //////////////////////////////////////////////////////////////////////////////////
15  module downDebounce(clk,rst,pulse,button,yes);
16     input clk,rst,pulse,button;
17     output reg yes;
18     reg [2:0] state, nState;
19     reg nYes;
20
21     always@(posedge clk, posedge rst)
22        if(rst)
23           {state,yes} <= 4'b0;
24        else
25           {state,yes} <= {nState, nYes};
26
27     always@(*)
28        case({state,pulse,button})
29           5'b000_00: {nState, nYes} = 4'b000_0;
30           5'b000_01: {nState, nYes} = 4'b001_0;
31           5'b000_10: {nState, nYes} = 4'b000_0;
32           5'b000_11: {nState, nYes} = 4'b001_0;
33           5'b001_00: {nState, nYes} = 4'b000_0;
34           5'b001_01: {nState, nYes} = 4'b001_0;
35           5'b001_10: {nState, nYes} = 4'b000_0;
36           5'b001_11: {nState, nYes} = 4'b010_0;
37           5'b010_00: {nState, nYes} = 4'b000_0;
38           5'b010_01: {nState, nYes} = 4'b010_0;
39           5'b010_10: {nState, nYes} = 4'b000_0;
40           5'b010_11: {nState, nYes} = 4'b011_0;
41           5'b011_00: {nState, nYes} = 4'b000_0;
42           5'b011_01: {nState, nYes} = 4'b011_0;
43           5'b011_10: {nState, nYes} = 4'b000_0;
44           5'b011_11: {nState, nYes} = 4'b100_1;
45           5'b100_00: {nState, nYes} = 4'b101_1;
46           5'b100_01: {nState, nYes} = 4'b100_1;
47           5'b100_10: {nState, nYes} = 4'b101_1;
48           5'b100_11: {nState, nYes} = 4'b100_1;
49           5'b101_00: {nState, nYes} = 4'b101_1;
50           5'b101_01: {nState, nYes} = 4'b100_1;
51           5'b101_10: {nState, nYes} = 4'b110_1;
52           5'b101_11: {nState, nYes} = 4'b100_1;
53           5'b110_00: {nState, nYes} = 4'b110_1;
54           5'b110_01: {nState, nYes} = 4'b100_1;
55           5'b110_10: {nState, nYes} = 4'b111_1;
56           5'b110_11: {nState, nYes} = 4'b100_1;
57           5'b111_00: {nState, nYes} = 4'b111_1;
```

```
58              5'b111_01: {nState, nYes} = 4'b100_1;
59              5'b111_10: {nState, nYes} = 4'b000_0;
60              5'b111_11: {nState, nYes} = 4'b100_1;
61              default: {nState, nYes} = 4'b000_0;
62          endcase
63
64
65
66
67
68
69      endmodule
70
```

```
## This file is a general .ucf for the Nexys4 DDR Rev C board
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used signals according to the project

## Clock signal
NET "clk"   LOC = "E3"  | IOSTANDARD = "LVCMOS33";             #Bank = 35, Pin name =
#IO_L12P_T1_MRCC_35,              Sch name = clk100mhz
#NET "clk100mhz" TNM_NET = sys_clk_pin;
#TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 100 MHz HIGH 50%;


## Switches
NET "sw<0>"          LOC=J15 | IOSTANDARD=LVCMOS33; #IO_L24N_T3_RS0_15
NET "sw<1>"          LOC=L16 | IOSTANDARD=LVCMOS33; #IO_L3N_T0_DQS_EMCCLK_14
NET "sw<2>"          LOC=M13 | IOSTANDARD=LVCMOS33; #IO_L6N_T0_D08_VREF_14
NET "sw<3>"          LOC=R15 | IOSTANDARD=LVCMOS33; #IO_L13N_T2_MRCC_14
NET "sw<4>"          LOC=R17 | IOSTANDARD=LVCMOS33; #IO_L12N_T1_MRCC_14
NET "sw<5>"          LOC=T18 | IOSTANDARD=LVCMOS33; #IO_L7N_T1_D10_14
NET "sw<6>"          LOC=U18 | IOSTANDARD=LVCMOS33; #IO_L17N_T2_A13_D29_14
NET "sw<7>"          LOC=R13 | IOSTANDARD=LVCMOS33; #IO_L5N_T0_D07_14
NET "sw<8>"          LOC=T8  | IOSTANDARD=LVCMOS33; #IO_L24N_T3_34
NET "sw<9>"          LOC=U8  | IOSTANDARD=LVCMOS33; #IO_25_34
NET "sw<10>"         LOC=R16 | IOSTANDARD=LVCMOS33; #IO_L15P_T2_DQS_RDWR_B_14
NET "sw<11>"         LOC=T13 | IOSTANDARD=LVCMOS33; #IO_L23P_T3_A03_D19_14
#NET "sw<12>"        LOC=H6  | IOSTANDARD=LVCMOS33; #IO_L24P_T3_35
#NET "sw<13>"        LOC=U12 | IOSTANDARD=LVCMOS33; #IO_L20P_T3_A08_D24_14
#NET "sw<14>"        LOC=U11 | IOSTANDARD=LVCMOS33; #IO_L19N_T3_A09_D25_VREF_14
#NET "sw<15>"        LOC=V10 | IOSTANDARD=LVCMOS33; #IO_L21P_T3_DQS_14


## Buttons
#NET "cpu_resetn"     LOC=C12 | IOSTANDARD=LVCMOS33; #IO_L3P_T0_DQS_AD1P_15

NET "rst"           LOC=N17 | IOSTANDARD=LVCMOS33; #IO_L9P_T1_DQS_14
NET "rightDownButton"            LOC=P18 |  IOSTANDARD=LVCMOS33; #IO_L9N_T1_DQS_D13_14
NET "downButton"           LOC=P17 | IOSTANDARD=LVCMOS33; #IO_L12P_T1_MRCC_14
NET "rightUpButton"           LOC=M17 | IOSTANDARD=LVCMOS33; #IO_L10N_T1_D15_14
NET "upButton"             LOC=M18 | IOSTANDARD=LVCMOS33; #IO_L4N_T0_D05_14


## LEDs
#NET "Q<0>"          LOC=H17 | IOSTANDARD=LVCMOS33; #IO_L18P_T2_A24_15
#NET "Q<1>"          LOC=K15 | IOSTANDARD=LVCMOS33; #IO_L24P_T3_RS1_15
#NET "Q<2>"          LOC=J13 | IOSTANDARD=LVCMOS33; #IO_L17N_T2_A25_15
#NET "Q<3>"          LOC=N14 | IOSTANDARD=LVCMOS33; #IO_L8P_T1_D11_14
#NET "led<4>"        LOC=R18 | IOSTANDARD=LVCMOS33; #IO_L7P_T1_D09_14
#NET "led<5>"        LOC=V17 | IOSTANDARD=LVCMOS33; #IO_L18N_T2_A11_D27_14
#NET "led<6>"        LOC=U17 | IOSTANDARD=LVCMOS33; #IO_L17P_T2_A14_D30_14
#NET "led<7>"        LOC=U16 | IOSTANDARD=LVCMOS33; #IO_L18P_T2_A12_D28_14
#NET "led<8>"        LOC=V16 | IOSTANDARD=LVCMOS33; #IO_L16N_T2_A15_D31_14
#NET "led<9>"        LOC=T15 | IOSTANDARD=LVCMOS33; #IO_L14N_T2_SRCC_14
#NET "led<10>"       LOC=U14 | IOSTANDARD=LVCMOS33; #IO_L22P_T3_A05_D21_14
#NET "led<11>"       LOC=T16 | IOSTANDARD=LVCMOS33; #IO_L15N_T2_DQS_DOUT_CSO_B_14
#NET "led<12>"       LOC=V15 | IOSTANDARD=LVCMOS33; #IO_L16P_T2_CSI_B_14
#NET "led<13>"       LOC=V14 | IOSTANDARD=LVCMOS33; #IO_L22N_T3_A04_D20_14
#NET "led<14>"       LOC=V12 | IOSTANDARD=LVCMOS33; #IO_L20N_T3_A07_D23_14
```

```
57  #NET "led<15>"      LOC=V11 | IOSTANDARD=LVCMOS33; #IO_L21N_T3_DQS_A06_D22_14
58
59
60  ##LEDs_RGB
61  #NET "led16_b"        LOC=R12 | IOSTANDARD=LVCMOS33; #IO_L5P_T0_D06_14
62  #NET "led16_g"        LOC=M16 | IOSTANDARD=LVCMOS33; #IO_L10P_T1_D14_14
63  #NET "led16_r"        LOC=N15 | IOSTANDARD=LVCMOS33; #IO_L11P_T1_SRCC_14
64  #NET "led17_b"        LOC=G14 | IOSTANDARD=LVCMOS33; #IO_L15N_T2_DQS_ADV_B_15
65  #NET "led17_g"        LOC=R11 | IOSTANDARD=LVCMOS33; #IO_0_14
66  #NET "led17_r"        LOC=N16 | IOSTANDARD=LVCMOS33; #IO_L11N_T1_SRCC_14
67
68
69  ## 7 segment display
70  #NET "cathodes<6>"             LOC=T10 | IOSTANDARD=LVCMOS33; #IO_L24N_T3_A00_D16_14
71  #NET "cathodes<5>"             LOC=R10 | IOSTANDARD=LVCMOS33; #IO_25_14
72  #NET "cathodes<4>"             LOC=K16 | IOSTANDARD=LVCMOS33; #IO_25_15
73  #NET "cathodes<3>"             LOC=K13 | IOSTANDARD=LVCMOS33; #IO_L17P_T2_A26_15
74  #NET "cathodes<2>"             LOC=P15 | IOSTANDARD=LVCMOS33; #IO_L13P_T2_MRCC_14
75  #NET "cathodes<1>"             LOC=T11 | IOSTANDARD=LVCMOS33; #IO_L19P_T3_A10_D26_14
76  #NET "cathodes<0>"             LOC=L18 | IOSTANDARD=LVCMOS33; #IO_L4P_T0_D04_14
77  #NET "cathodes<7>"             LOC=H15 | IOSTANDARD=LVCMOS33; #IO_L19N_T3_A21_VREF_15
78
79  #NET "anodes<0>"          LOC=J17 | IOSTANDARD=LVCMOS33; #IO_L23P_T3_FOE_B_15
80  #NET "anodes<1>"          LOC=J18 | IOSTANDARD=LVCMOS33; #IO_L23N_T3_FWE_B_15
81  #NET "anodes<2>"          LOC=T9  | IOSTANDARD=LVCMOS33; #IO_L24P_T3_A01_D17_14
82  #NET "anodes<3>"          LOC=J14 | IOSTANDARD=LVCMOS33; #IO_L19P_T3_A22_15
83  #NET "anodes<4>"          LOC=P14 | IOSTANDARD=LVCMOS33; #IO_L8N_T1_D12_14
84  #NET "anodes<5>"          LOC=T14 | IOSTANDARD=LVCMOS33; #IO_L14P_T2_SRCC_14
85  #NET "anodes<6>"          LOC=K2  | IOSTANDARD=LVCMOS33; #IO_L23P_T3_35
86  #NET "anodes<7>"          LOC=U13 | IOSTANDARD=LVCMOS33; #IO_L23N_T3_A02_D18_14
87
88
89  ## Pmod Header JA
90  #NET "ja<1>"          LOC=C17 | IOSTANDARD=LVCMOS33; #IO_L20N_T3_A19_15
91  #NET "ja<2>"          LOC=D18 | IOSTANDARD=LVCMOS33; #IO_L21N_T3_DQS_A18_15
92  #NET "ja<3>"          LOC=E18 | IOSTANDARD=LVCMOS33; #IO_L21P_T3_DQS_15
93  #NET "ja<4>"          LOC=G17 | IOSTANDARD=LVCMOS33; #IO_L18N_T2_A23_15
94  #NET "ja<7>"          LOC=D17 | IOSTANDARD=LVCMOS33; #IO_L16N_T2_A27_15
95  #NET "ja<8>"          LOC=E17 | IOSTANDARD=LVCMOS33; #IO_L16P_T2_A28_15
96  #NET "ja<9>"          LOC=F18 | IOSTANDARD=LVCMOS33; #IO_L22N_T3_A16_15
97  #NET "ja<10>"         LOC=G18 | IOSTANDARD=LVCMOS33; #IO_L22P_T3_A17_15
98
99  ## Pmod Header JB
100 #NET "jb<1>"          LOC=D14 | IOSTANDARD=LVCMOS33; #IO_L1P_T0_AD0P_15
101 #NET "jb<2>"          LOC=F16 | IOSTANDARD=LVCMOS33; #IO_L14N_T2_SRCC_15
102 #NET "jb<3>"          LOC=G16 | IOSTANDARD=LVCMOS33; #IO_L13N_T2_MRCC_15
103 #NET "jb<4>"          LOC=H14 | IOSTANDARD=LVCMOS33; #IO_L15P_T2_DQS_15
104 #NET "jb<7>"          LOC=E16 | IOSTANDARD=LVCMOS33; #IO_L11N_T1_SRCC_15
105 #NET "jb<8>"          LOC=F13 | IOSTANDARD=LVCMOS33; #IO_L5P_T0_AD9P_15
106 #NET "jb<9>"          LOC=G13 | IOSTANDARD=LVCMOS33; #IO_0_15
107 #NET "jb<10>"         LOC=H16 | IOSTANDARD=LVCMOS33; #IO_L13P_T2_MRCC_15
108
109 ## Pmod Header JC
110 #NET "jc<1>"          LOC=K1 | IOSTANDARD=LVCMOS33; #IO_L23N_T3_35
111 #NET "jc<2>"          LOC=F6 | IOSTANDARD=LVCMOS33; #IO_L19N_T3_VREF_35
112 #NET "jc<3>"          LOC=J2 | IOSTANDARD=LVCMOS33; #IO_L22N_T3_35
113 #NET "jc<4>"          LOC=G6 | IOSTANDARD=LVCMOS33; #IO_L19P_T3_35
```

```
114   #NET "jc<7>"          LOC=E7 | IOSTANDARD=LVCMOS33; #IO_L6P_T0_35
115   #NET "jc<8>"          LOC=J3 | IOSTANDARD=LVCMOS33; #IO_L22P_T3_35
116   #NET "jc<9>"          LOC=J4 | IOSTANDARD=LVCMOS33; #IO_L21P_T3_DQS_35
117   #NET "jc<10>"         LOC=E6 | IOSTANDARD=LVCMOS33; #IO_L5P_T0_AD13P_35
118
119   ## Pmod Header JD
120   #NET "jd<1>"          LOC=H4 | IOSTANDARD=LVCMOS33; #IO_L21N_T3_DQS_35
121   #NET "jd<2>"          LOC=H1 | IOSTANDARD=LVCMOS33; #IO_L17P_T2_35
122   #NET "jd<3>"          LOC=G1 | IOSTANDARD=LVCMOS33; #IO_L17N_T2_35
123   #NET "jd<4>"          LOC=G3 | IOSTANDARD=LVCMOS33; #IO_L20N_T3_35
124   #NET "jd<7>"          LOC=H2 | IOSTANDARD=LVCMOS33; #IO_L15P_T2_DQS_35
125   #NET "jd<8>"          LOC=G4 | IOSTANDARD=LVCMOS33; #IO_L20P_T3_35
126   #NET "jd<9>"          LOC=G2 | IOSTANDARD=LVCMOS33; #IO_L15N_T2_DQS_35
127   #NET "jd<10>"         LOC=F3 | IOSTANDARD=LVCMOS33; #IO_L13N_T2_MRCC_35
128
129   ##Pmod Header JXADC
130   #NET "xa_n<1>"        LOC=A14 | IOSTANDARD=LVDS; #IO_L9N_T1_DQS_AD3N_15
131   #NET "xa_p<1>"        LOC=A13 | IOSTANDARD=LVDS; #IO_L9P_T1_DQS_AD3P_15
132   #NET "xa_n<2>"        LOC=A16 | IOSTANDARD=LVDS; #IO_L8N_T1_AD10N_15
133   #NET "xa_p<2>"        LOC=A15 | IOSTANDARD=LVDS; #IO_L8P_T1_AD10P_15
134   #NET "xa_n<3>"        LOC=B17 | IOSTANDARD=LVDS; #IO_L7N_T1_AD2N_15
135   #NET "xa_p<3>"        LOC=B16 | IOSTANDARD=LVDS; #IO_L7P_T1_AD2P_15
136   #NET "xa_n<4>"        LOC=A18 | IOSTANDARD=LVDS; #IO_L10N_T1_AD11N_15
137   #NET "xa_p<4>"        LOC=B18 | IOSTANDARD=LVDS; #IO_L10P_T1_AD11P_15
138
139
140   ##VGA Connector
141   NET "vgaR<0>"       LOC=A3 | IOSTANDARD=LVCMOS33; #IO_L8N_T1_AD14N_35
142   NET "vgaR<1>"       LOC=B4 | IOSTANDARD=LVCMOS33; #IO_L7N_T1_AD6N_35
143   NET "vgaR<2>"       LOC=C5 | IOSTANDARD=LVCMOS33; #IO_L1N_T0_AD4N_35
144   NET "vgaR<3>"       LOC=A4 | IOSTANDARD=LVCMOS33; #IO_L8P_T1_AD14P_35
145
146   NET "vgaG<0>"       LOC=C6 | IOSTANDARD=LVCMOS33; #IO_L1P_T0_AD4P_35
147   NET "vgaG<1>"       LOC=A5 | IOSTANDARD=LVCMOS33; #IO_L3N_T0_DQS_AD5N_35
148   NET "vgaG<2>"       LOC=B6 | IOSTANDARD=LVCMOS33; #IO_L2N_T0_AD12N_35
149   NET "vgaG<3>"       LOC=A6 | IOSTANDARD=LVCMOS33; #IO_L3P_T0_DQS_AD5P_35
150
151   NET "vgaB<0>"       LOC=B7 | IOSTANDARD=LVCMOS33; #IO_L2P_T0_AD12P_35
152   NET "vgaB<1>"       LOC=C7 | IOSTANDARD=LVCMOS33; #IO_L4N_T0_35
153   NET "vgaB<2>"       LOC=D7 | IOSTANDARD=LVCMOS33; #IO_L6N_T0_VREF_35
154   NET "vgaB<3>"       LOC=D8 | IOSTANDARD=LVCMOS33; #IO_L4P_T0_35
155
156   NET "hscan"         LOC=B11 | IOSTANDARD=LVCMOS33; #IO_L4P_T0_15
157   NET "vscan"         LOC=B12 | IOSTANDARD=LVCMOS33; #IO_L3N_T0_DQS_AD1N_15
158
159
160   ##Micro SD Connector
161   #NET "sd_sck"         LOC=B1 | IOSTANDARD=LVCMOS33; #IO_L9P_T1_DQS_AD7P_35
162   #NET "sd_reset"       LOC=E2 | IOSTANDARD=LVCMOS33; #IO_L14P_T2_SRCC_35
163   #NET "sd_cd"          LOC=A1 | IOSTANDARD=LVCMOS33; #IO_L9N_T1_DQS_AD7N_35
164   #NET "sd_cmd"         LOC=C1 | IOSTANDARD=LVCMOS33; #IO_L16N_T2_35
165   #NET "sd_dat<0>"      LOC=C2 | IOSTANDARD=LVCMOS33; #IO_L16P_T2_35
166   #NET "sd_dat<1>"      LOC=E1 | IOSTANDARD=LVCMOS33; #IO_L18N_T2_35
167   #NET "sd_dat<2>"      LOC=F1 | IOSTANDARD=LVCMOS33; #IO_L18P_T2_35
168   #NET "sd_dat<3>"      LOC=D2 | IOSTANDARD=LVCMOS33; #IO_L14N_T2_SRCC_35
169
170
```

```
171  ##PWM Audio Amplifier
172  #NET "aud_pwm"         LOC=A11 | IOSTANDARD=LVCMOS33; #IO_L4N_T0_15
173  #NET "aud_sd"          LOC=D12 | IOSTANDARD=LVCMOS33; #IO_L6P_T0_15
174
175
176  ##Accelerometer
177  #NET "acl_miso"        LOC=E15 | IOSTANDARD=LVCMOS33; #IO_L11P_T1_SRCC_15
178  #NET "acl_mosi"        LOC=F14 | IOSTANDARD=LVCMOS33; #IO_L5N_T0_AD9N_15
179  #NET "acl_sclk"        LOC=F15 | IOSTANDARD=LVCMOS33; #IO_L14P_T2_SRCC_15
180  #NET "acl_csn"         LOC=D15 | IOSTANDARD=LVCMOS33; #IO_L12P_T1_MRCC_15
181  #NET "acl_int<1>"      LOC=B13 | IOSTANDARD=LVCMOS33; #IO_L2P_T0_AD8P_15
182  #NET "acl_int<2>"      LOC=C16 | IOSTANDARD=LVCMOS33; #IO_L20P_T3_A20_15
183
184
185  ##Temperature Sensor
186  #NET "tmp_ct"          LOC=B14 | IOSTANDARD=LVCMOS33; #IO_L2N_T0_AD8N_15
187  #NET "tmp_int"         LOC=D13 | IOSTANDARD=LVCMOS33; #IO_L6N_T0_VREF_15
188  #NET "tmp_scl"         LOC=C14 | IOSTANDARD=LVCMOS33; #IO_L1N_T0_AD0N_15
189  #NET "tmp_sda"         LOC=C15 | IOSTANDARD=LVCMOS33; #IO_L12N_T1_MRCC_15
190
191
192  ##USB-RS232 Interface
193  #NET "uart_cts"        LOC=D3 | IOSTANDARD=LVCMOS33; #IO_L12N_T1_MRCC_35
194  #NET "uart_rts"        LOC=E5 | IOSTANDARD=LVCMOS33; #IO_L5N_T0_AD13N_35
195  #NET "uart_rxd_out"    LOC=D4 | IOSTANDARD=LVCMOS33; #IO_L11N_T1_SRCC_35
196  #NET "uart_txd_in"     LOC=C4 | IOSTANDARD=LVCMOS33; #IO_L7P_T1_AD6P_35
197
198
199  ##Omnidirectional Microphone
200  #NET "m_clk"           LOC=J5 | IOSTANDARD=LVCMOS33; #IO_25_35
201  #NET "m_data"          LOC=H5 | IOSTANDARD=LVCMOS33; #IO_L24N_T3_35
202  #NET "m_lrsel"         LOC=F5 | IOSTANDARD=LVCMOS33; #IO_0_35
203
204
205  ##USB HID (PS/2)
206  #NET "ps2_clk"         LOC=F4 | IOSTANDARD=LVCMOS33; #IO_L13P_T2_MRCC_35
207  #NET "ps2_data"        LOC=B2 | IOSTANDARD=LVCMOS33; #IO_L10N_T1_AD15N_35
208
209
210  ##Quad SPI Flash
211  #NET "qspi_csn"        LOC=L13 | IOSTANDARD=LVCMOS33; #IO_L6P_T0_FCS_B_14
212  #NET "qspi_dq<0>"      LOC=K17 | IOSTANDARD=LVCMOS33; #IO_L1P_T0_D00_MOSI_14
213  #NET "qspi_dq<1>"      LOC=K18 | IOSTANDARD=LVCMOS33; #IO_L1N_T0_D01_DIN_14
214  #NET "qspi_dq<2>"      LOC=L14 | IOSTANDARD=LVCMOS33; #IO_L2P_T0_D02_14
215  #NET "qspi_dq<3>"      LOC=M14 | IOSTANDARD=LVCMOS33; #IO_L2N_T0_D03_14
216
217
218  ##SMSC Ethernet PHY
219  #NET "eth_rxd<0>"      LOC=C11 | IOSTANDARD=LVCMOS33; #IO_L13P_T2_MRCC_16
220  #NET "eth_rxd<1>"      LOC=D10 | IOSTANDARD=LVCMOS33; #IO_L19N_T3_VREF_16
221  #NET "eth_txd<0>"      LOC=A10 | IOSTANDARD=LVCMOS33; #IO_L14P_T2_SRCC_16
222  #NET "eth_txd<1>"      LOC=A8 | IOSTANDARD=LVCMOS33; #IO_L12N_T1_MRCC_16
223  #NET "eth_crsdv"       LOC=D9 | IOSTANDARD=LVCMOS33; #IO_L6N_T0_VREF_16
224  #NET "eth_intn"        LOC=B8 | IOSTANDARD=LVCMOS33; #IO_L12P_T1_MRCC_16
225  #NET "eth_mdc"         LOC=C9 | IOSTANDARD=LVCMOS33; #IO_L11P_T1_SRCC_16
226  #NET "eth_mdio"        LOC=A9 | IOSTANDARD=LVCMOS33; #IO_L14N_T2_SRCC_16
227  #NET "eth_refclk"      LOC=D5 | IOSTANDARD=LVCMOS33; #IO_L11P_T1_SRCC_35
```

```
228   #NET "eth_rstn"        LOC=B3 | IOSTANDARD=LVCMOS33; #IO_L10P_T1_AD15P_35
229   #NET "eth_txen"        LOC=B9 | IOSTANDARD=LVCMOS33; #IO_L11N_T1_SRCC_16
230   #NET "eth_rxerr"       LOC=C10 | IOSTANDARD=LVCMOS33; #IO_L13N_T2_MRCC_16
231
```