```
1   `timescale 1ns / 1ps
2   /////////////////////////////////////////////////////////////////////////////////
3   *
4   * Engineer: Justin Maeder; 015906629
5   * Email:    Justin_maeder@hotmail.com
6   * Filename: pong_graph_st.v
7   * Date:     May 4, 2019
8   * Version:  14.7
9   * Description: This module is implemented to create a graphic generating unit for
10  *              one static object: wall and two animated objects: paddle and ball.
11  *              The wall has left and right boundaries 32 and 35 (horizontal count)
12  *              respectively. The wall top and bottom boundaries range from the top
13  *              of the display to the bottom (vertical count = 0 to 480). The bar has
14  *              left and right boundaries 600 and 603 (horizontal) and the verical
15  *              count of the bar is determined by the users choice in using the
16  *              mechanical onboard buttons with a velocity of +/- 4 pixels. The
17  *              square ball has a size of 8, and on reset the ball starts at the top
18  *              left of the display with a velocity of +/- 2 pixels. The way the
19  *              objects are displayed is when the pixel_x and pixel_y scans are in
20  *              the object boundaries, it will change the RGB signal within the given
21  *              region.
22  *
23  /////////////////////////////////////////////////////////////////////////////////*/
24  module pong_graph_st(clk, reset, video_on, btn_up, btn_down,
25                       pixel_x, pixel_y, graph_rgb);
26
27     // INPUT & OUTPUT declaration
28     input wire clk, reset, video_on, btn_up, btn_down;
29     input wire [9:0] pixel_x, pixel_y;
30     output reg [11:0] graph_rgb;
31
32     // RESET when game over
33     wire game_reset;
34
35     // Reference Tick
36     wire refr_tick;
37
38     // Object output signals
39     wire wall_on, bar_on, sq_ball_on;
40     // RGB signals for objects
41     wire [11:0] wall_rgb, bar_rgb, ball_rgb;
42
43     // BAR top & bottom
44     wire [9:0] bar_y_t, bar_y_b;
45     reg  [9:0] bar_y_reg, bar_y_next;
46
47     // BALL boundaries
48     wire [9:0] ball_y_t, ball_y_b;
49     wire [9:0] ball_x_l, ball_x_r;
50
51     // BALL position
52     reg  [9:0] ball_x_reg,  ball_y_reg;
53     wire [9:0] ball_x_next, ball_y_next;
54
55     //  BALL movement registers
56     reg [9:0] x_delta_reg, x_delta_next;
57     reg [9:0] y_delta_reg, y_delta_next;
```

```
58
59      // constant and signal declaration
60      // x, y coordinates (0,0) to (639,479)
61      localparam MAX_X = 10'd640,
62                 MAX_Y = 10'd480,
63                 //--------------------------
64                 // Vertical stripe as a wall
65                 //--------------------------
66                 // Wall LEFT/RIGHT Boundaries
67                 WALL_X_L = 10'd32,
68                 WALL_X_R = 10'd35,
69                 //--------------------------
70                 // Right Vertical Bar
71                 //--------------------------
72                 BAR_Y_SIZE = 10'd72,
73                 BAR_V = 10'd4,
74                 // Bar LEFT/RIGHT Boundaries
75                 BAR_X_L = 10'd600,
76                 BAR_X_R = 10'd603,
77                 //--------------------------
78                 // Square Ball
79                 //--------------------------
80                 BALL_SIZE = 10'd8,
81                 // Ball speed variables
82                 BALL_V_P = 10'd2,
83                 BALL_V_N = -10'd2;
84
85      // RESET GAME
86      assign game_reset = ball_x_r > MAX_X - 1;
87
88      always @ (posedge clk, posedge reset)
89         if (reset || game_reset) begin
90            if (reset) bar_y_reg <= 10'd0;
91
92            ball_x_reg <= 10'd0;
93            ball_y_reg <= 10'd0;
94
95            x_delta_reg <= 10'h004;
96            y_delta_reg <= 10'h004;
97         end
98
99         else begin
100           bar_y_reg <= bar_y_next;
101
102           ball_x_reg <= ball_x_next;
103           ball_y_reg <= ball_y_next;
104
105           x_delta_reg <= x_delta_next;
106           y_delta_reg <= y_delta_next;
107        end
108
109     assign refr_tick = ( (pixel_y == 10'd481) && (pixel_x == 10'd0) );
110
111     //--------------------------
112     // Wall: LEFT vertical strip
113     //--------------------------
114     // pixel within wall
```

```
115     assign wall_on = (WALL_X_L <= pixel_x) && (pixel_x <= WALL_X_R);
116     assign wall_rgb = 12'h00F; // RED wall
117
118     //---------------------------
119     // RIGHT vertical bar
120     //---------------------------
121     // pixel within bar
122     assign bar_y_t = bar_y_reg;
123     assign bar_y_b = bar_y_t + BAR_Y_SIZE - 10'd1;
124
125     assign bar_on = (BAR_X_L <= pixel_x) && (pixel_x <= BAR_X_R) &&
126                     (bar_y_t <= pixel_y) && (pixel_y <= bar_y_b);
127     assign bar_rgb = 12'hF00; // BLUE bar
128
129     // NEW POSITION = BAR
130     always @ (*) begin
131        bar_y_next = bar_y_reg;
132        if (refr_tick)
133           if ( btn_down && (bar_y_b < MAX_Y - 10'd1 - BAR_V) )
134              bar_y_next = bar_y_reg + BAR_V;
135           else if ( btn_up && (bar_y_t > BAR_V) )
136              bar_y_next = bar_y_reg - BAR_V;
137     end
138
139     //---------------------------
140     // Square ball
141     //---------------------------
142     // pixel within squared ball
143     assign ball_x_l = ball_x_reg;
144     assign ball_x_r = ball_x_l + BALL_SIZE - 10'd1;
145     assign ball_y_t = ball_y_reg;
146     assign ball_y_b = ball_y_t + BALL_SIZE - 10'd1;
147
148     assign sq_ball_on = (ball_x_l <= pixel_x) && (pixel_x <= ball_x_r) &&
149                         (ball_y_t <= pixel_y) && (pixel_y <= ball_y_b);
150     assign ball_rgb = 12'h777; // GREY ball
151
152     // NEXT POSITION = BALL
153     assign ball_x_next = (refr_tick) ? ball_x_reg + x_delta_reg : ball_x_reg;
154     assign ball_y_next = (refr_tick) ? ball_y_reg + y_delta_reg : ball_y_reg;
155
156     // Update ball position change
157     always @ (*) begin
158        x_delta_next = x_delta_reg;
159        y_delta_next = y_delta_reg;
160
161        if(ball_y_t < 1) y_delta_next = BALL_V_P;
162        else if( ball_y_b > (MAX_Y - 1) ) y_delta_next = BALL_V_N;
163
164        // BALL HITS RIGHT BAR
165        else if( (BAR_X_L <= ball_x_r) && (ball_x_r <= BAR_X_R) &&
166                 (bar_y_t <= ball_y_b) && (ball_y_t <= bar_y_b) )
167                x_delta_next = BALL_V_N;// bounce back
168
169        // BALL HITS WALL
170        else if (ball_x_l <= WALL_X_R) x_delta_next = BALL_V_P;   // bounce back
171     end // END balls position change
```

```
172
173
174
175       //-------------------------
176       // RGB multiplexing unit
177       //-------------------------
178       always @(*)
179          if (~video_on)
180             graph_rgb = 12'b0;
181          else
182             if (wall_on) graph_rgb = wall_rgb;
183             else if (bar_on)     graph_rgb = bar_rgb;
184             else if (sq_ball_on) graph_rgb = ball_rgb;
185             else graph_rgb = 12'hFFF; // WHITE background
186    endmodule
```