```
1   /*=============================================================================
2    * This document contains information proprietary to the CSULB student that
3      created the file - any reuse without adequate approval and documentation is
4      prohibited.
5    * Class:      CECS 360 - Integrated Circuit Design Software
6    * File Name:  vga_controller_top_tf.v
7    * Project:    VGA Controller
8    * Designer:   Michael Rios - Copyright © 2017. All rights reserved.
9    * Email:      riosmichael28@ymail.com
10   * Rev. Date:  October 20, 2017
11   *
12   * Description: This module is a text fixture for the vga_controller module.
13                  This module is a self-checking test bench, meaning that
14   *
15   * In submitting this file for class work at CSULB, I am confirming that this
16     is my work and the work of no one else.
17   *
18   * In the event other code sources are utilized I will document which portion
19     of the code and who is the author.
20   *
21   * In submitting this code I acknowledge that plagiarism in student project
22     work is subject to dismissal from the class.
23   *=============================================================================*/
24  `timescale 1ns / 1ps
25
26  module vga_controller_top_tf;
27
28     // Inputs
29     reg clk;
30     reg rst;
31
32     // Outputs
33     wire hsync;
34     wire vsync;
35     wire [11:0] rgb;
36
37
38
39     reg flag = 1'b0; // flag will be used to determine success
40
41     // ========================================================
42     // Generate the 25MHz pixel rate from the 100Mhz board clk
43     // ========================================================
44
45     reg [1:0] pxl_clk;
46     wire tick;
47     assign tick = pxl_clk == 2'b11;
48
49     always @ (posedge clk, posedge rst)
50
51        if (rst)       pxl_clk <= 2'b0;
52        else if (tick) pxl_clk <= 2'b0;
53        else           pxl_clk <= pxl_clk + 2'b1;
54
55     // ================================
56     // Horizontal Sync
57     // Horizontal count 0 ... 799
```

```
58      // Update at 25MHz
59      // ================================
60
61      reg [9:0] hcount;
62      wire endh;
63      assign endh = (hcount == 10'd799);
64
65      always @ (posedge clk, posedge rst)
66         if (rst) hcount <= 10'b0;
67         else if (tick)
68            if (endh)   hcount <= 10'b0;
69            else        hcount <= hcount + 10'b1;
70
71            assign pixel_x = hcount;
72
73      // ================================
74      // Vertical Sync
75      // Vertical count 0 ... 524
76      // Update at 25MHz
77      // ================================
78
79      reg [9:0] vcount;
80      wire endv;
81      assign endv = (vcount == 10'd524);
82
83      always @ (posedge clk, posedge rst)
84         if (rst) vcount <= 10'b0;
85         else if (tick)
86            if (endh)
87               if (endv)
88                  begin
89                     vcount <= 10'b0;
90                     //check the flag register which will
91                     // be asserted if there is an error
92                     if (flag == 1'b1) $display("Error");
93                     else              $display("Success");
94
95                     // finish simulation after display has
96                     // been scanned once
97                     $finish;
98                  end
99               else        vcount <= vcount + 10'b1;
100
101
102      assign pixel_y = vcount;
103
104
105     // Instantiate the Unit Under Test (UUT)
106     vga_controller_top uut (
107        .clk(clk),
108        .rst(rst),
109        .hsync(hsync),
110        .vsync(vsync),
111        .rgb(rgb)
112     );
113
114     // generate 10ns clk
```

```
115      always
116         #10 clk = ~clk;
117
118      always @ (posedge clk, posedge rst)
119      begin
120
121             // Verify the placement of the wall
122             if (pixel_x >= 32 && pixel_x <= 35 && !(rgb == 12'h00f))
123                  flag = 1'b1;
124
125             // Verify the placement of the ball
126             else if ((pixel_x >= 580) && (pixel_x <= 588) && (pixel_y >= 238)
127                       && (pixel_y <= 246) && !(rgb == 12'hf00))
128                  flag = 1'b1;
129
130             // Verify the placement of the paddlea
131             else if ((pixel_x >= 600) && (pixel_x <= 603) && (pixel_y >= 204)
132                       && (pixel_y <= 276) && !(rgb == 12'h0f0))
133                  flag = 1'b1;
134
135             // Verify that no pixel displays when display should be off
136             else if (pixel_x > 640 && !(rgb == 12'h000))
137                  flag = 1'b1;
138
139             // Verify that no pixel displays when display should be off
140             else if (pixel_y > 480 && !(rgb == 12'h000))
141                  flag = 1'b1;
142
143             // Display "Success" if every case is satisfied
144             else
145                  flag = 1'b0;
146
147      end
148
149      initial begin
150         // Initialize Inputs
151         clk = 0;
152         rst = 1;
153
154         // Wait 100 ns for global reset to finish
155         #100;
156         rst = 0;
157
158       end
159
160
161   endmodule
162
163
```