

```

1  // Eugene Ngo
2  // 11/28/2022
3  // EE 271
4  // Lab 5
5
6  // winner module takes in 1-bit clk, 1-bit reset, 1-bit LED9, 1-bit LED1, 1-bit L, and
  // 1-bit R as inputs
7  // and returns 7-bit HEX0, 7-bit HEX5, and 1-bit restart. This module determines who the
  // winner of Cyber War
8  // is by checking to see whether the human or the computer scores and then increments the
  // displayed score
9  // appropriately: if human wins then the score displayed on HEX0 is incremented and if the
  // computer wins then
10 // the score on HEX5 is incremented. Once a competitor, either human or computer, wins, the
   // game is restarted.
11
12 module winner (restart, HEX0, HEX5, clk, reset, LED9, LED1, L, R);
13     input logic clk, reset;
14     input logic LED9, LED1, L, R;
15     output logic [6:0] HEX0, HEX5;
16     output logic restart;
17
18     logic [2:0] Lcount;
19     logic [2:0] Rcount;
20
21     enum {off, P1, P2} ps, ns;
22
23     always_comb begin
24         case(ps)
25             off:   if(LED1 & ~L & R) ns = P1;
26
27                   else if(LED9 & ~R & L) ns = P2;
28
29                   else ns = off;
30
31             P1: ns = P1; // Human
32             P2: ns = P2; // Computer
33
34         endcase
35     end
36
37     // counter system
38     always_comb begin
39
40
41
42         // Human
43         if(Rcount == 3'b000) // 0
44             HEX0 = 7'b1000000;
45         else if(Rcount == 3'b001) // 1
46             HEX0 = 7'b1111001;
47         else if(Rcount == 3'b010) // 2
48             HEX0 = 7'b0100100;
49         else if(Rcount == 3'b011) // 3
50             HEX0 = 7'b0110000;
51         else if(Rcount == 3'b100) // 4
52             HEX0 = 7'b0011001;
53         else if(Rcount == 3'b101) // 5
54             HEX0 = 7'b0010010;
55         else if(Rcount == 3'b110) // 6
56             HEX0 = 7'b0000010;
57         else begin // 7
58             HEX0 = 7'b1111000;
59         end
60
61         // Computer
62         if(Lcount == 3'b000) // 0
63             HEX5 = 7'b1000000;
64         else if(Lcount == 3'b001) // 1
65             HEX5 = 7'b1111001;
66         else if(Lcount == 3'b010) // 2
67             HEX5 = 7'b0100100;
68         else if(Lcount == 3'b011) // 3

```

```

69         HEX5 = 7'b0110000;
70     else if(Lcount == 3'b100) // 4
71         HEX5 = 7'b0011001;
72     else if(Lcount == 3'b101) // 5
73         HEX5 = 7'b0010010;
74     else if(Lcount == 3'b110) // 6
75         HEX5 = 7'b0000010;
76     else begin // 7
77         HEX5 = 7'b1111000;
78     end
79
80 end
81
82 always_ff @(posedge clk) begin
83     if(ps == off & ns == P1) begin
84
85         Rcount <= Rcount + 1;
86
87     end
88
89     else if(ps == off & ns == P2) begin
90
91         Lcount <= Lcount + 1;
92     end
93
94     else begin
95         Rcount <= Rcount;
96         Lcount <= Lcount;
97     end
98
99
100    if(reset) begin
101        Lcount <= 3'b000;
102        Rcount <= 3'b000;
103        ps <= off;
104        restart <= 0;
105    end
106
107    if(Lcount == 3'b111) begin
108        Lcount <= 3'b000;
109        Rcount <= 3'b000;
110        ps <= off;
111        restart <= 0;
112    end
113
114    if(Rcount == 3'b111) begin
115        Lcount <= 3'b000;
116        Rcount <= 3'b000;
117        ps <= off;
118        restart <= 0;
119    end
120
121    else if(restart) begin
122        ps <= off;
123        restart <= 0;
124    end
125
126    else
127        ps <= ns;
128
129    if(ps == P1 | ps == P2)
130        restart <= 1;
131    else
132        restart <= 0;
133 end
134
135 endmodule
136
137 // winner_testbench tests all expected, unexpected, and edgecase behaviors
138 module winner_testbench();
139     logic clk, reset;
140     logic LED9, LED1, L, R;
141     logic restart;

```

```
142 logic [6:0] HEX0, HEX5;
143
144 winner dut (.restart, .HEX0(HEX0), .HEX5(HEX5), .clk, .reset, .LED9, .LED1, .L, .R);
145
146 parameter CLOCK_PERIOD = 100;
147 initial begin
148     clk <= 0;
149     forever #(CLOCK_PERIOD / 2)
150         clk <= ~clk;
151 end
152
153 initial begin
154     reset <= 1; @ (posedge clk);
155                 @ (posedge clk);
156     reset <= 0; @ (posedge clk);
157                 @ (posedge clk);
158     LED9 <= 1; LED1 <= 0; L <= 1; R <= 0; @ (posedge clk);
159                 @ (posedge clk);
160     LED9 <= 0; LED1 <= 1; @ (posedge clk);
161                 @ (posedge clk);
162     LED9 <= 1; LED1 <= 1; @ (posedge clk);
163                 @ (posedge clk);
164                 LED1 <= 0; R <= 1; @ (posedge clk);
165                 @ (posedge clk);
166     reset <= 1; @ (posedge clk);
167                 @ (posedge clk);
168     reset <= 0; @ (posedge clk);
169                 @ (posedge clk);
170     LED9 <= 0; LED1 <= 1; L <= 0; R <= 1; @ (posedge clk);
171                 @ (posedge clk);
172     LED9 <= 1; @ (posedge clk);
173                 @ (posedge clk);
174     LED9 <= 0; L <= 1; @ (posedge clk);
175                 @ (posedge clk);
176     $stop;
177 end
178 endmodule
```