```
1   /*
2      Caitlin DeShazo-Couchot and Ashley Guillard
3      EE 271 Au23: Professor Hussein
4      November 9, 2023
5      Lab 4: LED2, LED3, LED4, LED6, LED7, LED8
6   */
7
8   //Top level module for the DE1_SoC Board
9   //I/O Needed for the DE1_DoC as well as CLOCK_50
10  module DE1_SoC (
11      output logic [6:0] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5,
12      output logic [9:0] LEDR,
13      input logic  [3:0] KEY,
14      input logic  [9:0] SW,
15      input logic  CLOCK_50
16  );
17
18      // Default values, turns off the HEX displays
19      assign HEX1 = 7'b1111111;
20      assign HEX2 = 7'b1111111;
21      assign HEX3 = 7'b1111111;
22      assign HEX4 = 7'b1111111;
23      assign HEX5 = 7'b1111111;
24
25      // Logic Variables
26      logic L, R, L_Tug, R_Tug;
27
28      // Flip Flops for Tug-of-War to handle Metastability
29      metaStability L_dff (.CLOCK(CLOCK_50), .Reset(SW[9]), .KEY(~KEY[0]), .out(R_Tug));
30      metaStability R_dff (.CLOCK(CLOCK_50), .Reset(SW[9]), .KEY(~KEY[3]), .out(L_Tug));
31
32      // User Inputs
33      player player1 (.CLOCK(CLOCK_50), .Reset(SW[9]), .KEY(R_Tug), .out(R));
34      player player2 (.CLOCK(CLOCK_50), .Reset(SW[9]), .KEY(L_Tug), .out(L));
35
36      // Instantiate LED FSMs for LED1 to LED4
37      normalLight led1(.CLOCK(CLOCK_50), .Reset(SW[9]), .L(L), .R(R), .NL(LEDR[2]), .NR(1'b0),
        .lightOn(LEDR[1]));
38      normalLight led2(.CLOCK(CLOCK_50), .Reset(SW[9]), .L(L), .R(R), .NL(LEDR[3]), .NR(LEDR[1
    ]), .lightOn(LEDR[2]));
39      normalLight led3(.CLOCK(CLOCK_50), .Reset(SW[9]), .L(L), .R(R), .NL(LEDR[4]), .NR(LEDR[2
    ]), .lightOn(LEDR[3]));
40      normalLight led4(.CLOCK(CLOCK_50), .Reset(SW[9]), .L(L), .R(R), .NL(LEDR[5]), .NR(LEDR[3
    ]), .lightOn(LEDR[4]));
41
42      // Instatiate center LED FSM, LED5
43      centerLight led5(.CLOCK(CLOCK_50), .Reset(SW[9]), .L(L), .R(R), .NL(LEDR[6]), .NR(LEDR[4
    ]), .lightOn(LEDR[5]));
44
45      // Instantiate LED FSMs for LED1 to LED4
46      normalLight led6(.CLOCK(CLOCK_50), .Reset(SW[9]), .L(L), .R(R), .NL(LEDR[7]), .NR(LEDR[5
    ]), .lightOn(LEDR[6]));
47      normalLight led7(.CLOCK(CLOCK_50), .Reset(SW[9]), .L(L), .R(R), .NL(LEDR[8]), .NR(LEDR[6
    ]), .lightOn(LEDR[7]));
48      normalLight led8(.CLOCK(CLOCK_50), .Reset(SW[9]), .L(L), .R(R), .NL(LEDR[9]), .NR(LEDR[7
    ]), .lightOn(LEDR[8]));
49      normalLight led9(.CLOCK(CLOCK_50), .Reset(SW[9]), .L(L), .R(R), .NL(1'b1),    .NR(LEDR[8
    ]), .lightOn(LEDR[9]));
50
51      // Instantiate victory logic to control HEX0
52      victory victory_game(.CLOCK(CLOCK_50), .Reset(SW[9]), .LEDR9(LEDR[9]), .LEDR1(LEDR[1]),
    .L(L), .R(R),  .HEX0(HEX0));
53
54  endmodule
55
56  // Test bench for the testbench module.
57  // Standard I/O for the DE1_SoC
58  module DE1_SoC_testbench();
59       // Test inputs and outputs
60      logic [6:0] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
61      logic [9:0] LEDR;
62      logic  [3:0] KEY;
63      logic  [9:0] SW;
64      logic CLOCK_50;
65
66      // Instantiate the DUT (Device Under Test)
```

```
67        DE1_SoC dut(HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, LEDR, KEY, SW, CLOCK_50);
68
69        // Define Clock period for simulation
70        parameter Clock_PERIOD = 100;
71
72        // Clock generation for simulation
73        initial begin
74            CLOCK_50 <= 0;
75            forever #Clock_PERIOD CLOCK_50 <= ~CLOCK_50;
76        end
77
78           initial begin
79            // Reset the game
80            SW[9] <= 1; @(posedge CLOCK_50)
81            SW[9] <= 0; @(posedge CLOCK_50)
82
83            // Game starts, LEDs should be initialized
84            KEY[0] <= 1; KEY[3] <= 1; @(posedge CLOCK_50)
85
86            // Game where Player1 wins right away
87            for (int i=0; i<6; i++) begin
88               KEY[0] <= 0; @(posedge CLOCK_50);
89               KEY[0] <= 1; @(posedge CLOCK_50);
90            end
91
92            @(posedge CLOCK_50) //To view change in HEX0 in ModelSim
93            @(posedge CLOCK_50)
94            @(posedge CLOCK_50)
95            @(posedge CLOCK_50)
96
97            // Game reset and play again
98            SW[9] <= 1; @(posedge CLOCK_50)
99            SW[9] <= 0; @(posedge CLOCK_50)
100
101           // Game starts, LEDs should be initialized
102           KEY[0] <= 1; KEY[3] <= 1; @(posedge CLOCK_50)
103
104           // Game where Player2 wins right away
105           for (int i=0; i<6; i++) begin
106              KEY[3] <= 0; @(posedge CLOCK_50);
107              KEY[3] <= 1; @(posedge CLOCK_50);
108           end
109
110           @(posedge CLOCK_50) //To view change in HEX0 in ModelSim
111           @(posedge CLOCK_50)
112           @(posedge CLOCK_50)
113           @(posedge CLOCK_50)
114
115           // Game reset and play again
116           SW[9] <= 1; @(posedge CLOCK_50)
117           SW[9] <= 0; @(posedge CLOCK_50)
118
119           // Game starts, LEDs should be initialized
120           KEY[0] <= 1; KEY[3] <= 1; @(posedge CLOCK_50)
121
122           // Game where Players go back and forth, Player2 wins
123           for (int i=0; i<5; i++) begin
124              KEY[3] <= 0; @(posedge CLOCK_50);
125              KEY[3] <= 1; @(posedge CLOCK_50);
126              KEY[0] <= 0; @(posedge CLOCK_50); // Tests long press
127              KEY[3] <= 0; @(posedge CLOCK_50);
128              KEY[3] <= 1; @(posedge CLOCK_50);
129              KEY[0] <= 1; @(posedge CLOCK_50);
130           end
131
132           @(posedge CLOCK_50) //To view change in HEX0 in ModelSim
133           @(posedge CLOCK_50)
134           @(posedge CLOCK_50)
135           @(posedge CLOCK_50)
136
137         $stop;
138       end
139
140   endmodule
```