```
1   // Eugene Ngo
2   // 12/2/2022
3   // EE 271
4   // Lab 5
5
6   // DE1_SoC is the top level module. It takes in a 4-Bit KEY, 1-Bit CLOCK_50, and a 10-Bit
    SW as inputs.
7   // It returns 7-Bit HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, as well as a 10-Bit LEDR. The LEDR
    indicates the
8   // the current position of the "rope" and once it reaches either opposing side, a victor is
    assigned and
9   // displayed on HEX-. Switch 9 is used to reset the game.
10
11  module DE1_SoC (CLOCK_50, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, KEY, LEDR, SW);
12      input logic         CLOCK_50; // 50MHz clock
13      input logic   [3:0] KEY;
14      input logic   [9:0] SW;
15      output logic  [6:0] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
16      output logic  [9:0] LEDR;
17
18      // Generate clk using CLOCK_50 (whichClock determines rate)
19      logic [31:0] clk;
20      parameter whichClock = 15;
21      clock_divider cdiv (CLOCK_50, clk);
22
23      // assign HEX values to be OFF to start game
24      assign HEX4 = 7'b1111111;
25      assign HEX3 = 7'b1111111;
26      assign HEX2 = 7'b1111111;
27      assign HEX1 = 7'b1111111;
28
29      // declare key, key base, and CPU variables
30      logic key0, key3;
31      logic key0_base, key3_base;
32      logic CPU;
33
34      // Instantiates dff modules to process button inputs at clock cycles
35      // doubleFlipFlop module takes in clk[whichClock] to clk, SW[9] to reset, and
    KEY[0]/CPU to button as inputs
36      // and returns out as clocked inputs
37      doubleFlipFlop flipFlop1 (.clk(clk[whichClock]), .reset(SW[9]), .button(~KEY[0]), .out(
    key0_base));
38      doubleFlipFlop flipFlop2 (.clk(clk[whichClock]), .reset(SW[9]), .button(CPU), .out(
    key3_base));
39
40      // Instantiates user input modules with switch zero and three for user inputs
41      // userInput module takes in clk[whichClock] to clk, SW[9] to reset, and and clocked
    inputs to button as inputs
42      // and returns button states to out
43      // One of the user inputs is designated to be the user/human and the other is an
    automated computer opponent
44      userInput human (.clk(clk[whichClock]), .reset(SW[9]), .button(key0_base), .out(key0));
45      userInput computer (.clk(clk[whichClock]), .reset(SW[9]), .button(key3_base), .out(key3
    ));
46
47      // Light instantiations
48      // The light modules takes in clk[whichClock] to clk, SW[9] to reset, key3 to L, key 0
    to R, LEDR[9:0] to NL, LEDR[0:9] to NR, restart to restart
49      // and returns LEDR[0:9] to lightOn.
50      normalLight L1 (.clk(clk[whichClock]), .reset(SW[9]), .L(key3), .R(key0), .NL(LEDR[2]),
    .NR(1'b0), .lightOn(LEDR[1]), .restart(restart));
51      normalLight L2 (.clk(clk[whichClock]), .reset(SW[9]), .L(key3), .R(key0), .NL(LEDR[3]),
    .NR(LEDR[1]), .lightOn(LEDR[2]), .restart(restart));
52      normalLight L3 (.clk(clk[whichClock]), .reset(SW[9]), .L(key3), .R(key0), .NL(LEDR[4]),
    .NR(LEDR[2]), .lightOn(LEDR[3]), .restart(restart));
53      normalLight L4 (.clk(clk[whichClock]), .reset(SW[9]), .L(key3), .R(key0), .NL(LEDR[5]),
    .NR(LEDR[3]), .lightOn(LEDR[4]), .restart(restart));
54
55      // center light is a little different from the normalLight module so that when the game
    is reset or restarted, the LED light is recenetered
56      centerLight L5 (.clk(clk[whichClock]), .reset(SW[9]), .L(key3), .R(key0), .NL(LEDR[6]),
    .NR(LEDR[4]), .lightOn(LEDR[5]), .restart(restart));
57
```

```
58      normalLight L6 (.clk(clk[whichClock]), .reset(SW[9]), .L(key3), .R(key0), .NL(LEDR[7]),
    .NR(LEDR[5]), .lightOn(LEDR[6]), .restart(restart));
59      normalLight L7 (.clk(clk[whichClock]), .reset(SW[9]), .L(key3), .R(key0), .NL(LEDR[8]),
    .NR(LEDR[6]), .lightOn(LEDR[7]), .restart(restart));
60      normalLight L8 (.clk(clk[whichClock]), .reset(SW[9]), .L(key3), .R(key0), .NL(LEDR[9]),
    .NR(LEDR[7]), .lightOn(LEDR[8]), .restart(restart));
61      normalLight L9 (.clk(clk[whichClock]), .reset(SW[9]), .L(key3), .R(key0), .NL(1'b0), .NR
    (LEDR[8]), .lightOn(LEDR[9]), .restart(restart));
62
63      // generate button for CPU
64      logic [9:0] LFSR_out;
65
66      // The LFSR module takes in clk[whichClock] to clk, SW[9] to reset, and returns
    LFSR_OUT to Q
67      // This module is used to simulate computer button presses via a random number generator
68      LFSR random(.clk(clk[whichClock]), .reset(SW[9]), .Q(LFSR_out));
69
70      // The compButton module takes in clk[whichClock] to clk, LFSR_out to Q, SW[8:0] to SW,
    and returns CPU to out
71      // This module formats the switches for the computer opponent's input
72      compButton comp(.clk(clk[whichClock]), .reset(SW[9]), .Q(LFSR_out), .SW(SW[8:0]), .out(
    CPU));
73
74      // Determines who scores points
75      // victory module takes in clk[whichClock] to clk, SW[9] to reset, LEDR[9] to LED9,
    LEDR[1] to LED1, key3 to L, key0 to R,
76      // restart to restart, HEX0 to HEX0 and HEX5 to HEX5.
77      // If the human scores points then it will be displayed and reflects on HEX0 and when
    the computer scores it will be
78      // shown on HEX5
79      winner winnerFound (.clk(clk[whichClock]), .reset(SW[9]), .LED9(LEDR[9]), .LED1(LEDR[1
    ]), .L(key3), .R(key0), .restart(restart), .HEX0(HEX0), .HEX5(HEX5));
80
81  endmodule
82
83  // DE1_SoC_testbench tests all expected, unexpected, and edgecase behaviors
84  module DE1_SoC_testbench();
85     logic        CLOCK_50;
86     logic [6:0] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
87     logic [9:0] LEDR;
88     logic [3:0] KEY;
89     logic [9:0] SW;
90
91     DE1_SoC dut (.CLOCK_50, .HEX0, .HEX1, .HEX2, .HEX3, .HEX4, .HEX5, .KEY, .LEDR, .SW);
92
93     parameter CLOCK_PERIOD = 100;
94     initial begin
95        CLOCK_50 <= 0;
96        forever #(CLOCK_PERIOD / 2)
97        CLOCK_50 <= ~CLOCK_50;
98     end
99
100    initial begin
101                                                    @(posedge CLOCK_50);
102                                                    @(posedge CLOCK_50);
103       SW[9] <= 1;                                  @(posedge CLOCK_50);
104                                                    @(posedge CLOCK_50);
105                                                    @(posedge CLOCK_50);
106                                                    @(posedge CLOCK_50);
107       SW[9] <= 0;                                  @(posedge CLOCK_50);
108                                                    @(posedge CLOCK_50);
109                                                    @(posedge CLOCK_50);
110                                                    @(posedge CLOCK_50);
111       KEY[0] <= 1; SW[8:0] = 9'b000000000;         @(posedge CLOCK_50);
112                                                    @(posedge CLOCK_50);
113       KEY[0] <= 0;                                 @(posedge CLOCK_50);
114                                                    @(posedge CLOCK_50);
115       KEY[0] <= 1;                                 @(posedge CLOCK_50);
116                                                    @(posedge CLOCK_50);
117       KEY[0] <= 0;                                 @(posedge CLOCK_50);
118                                                    @(posedge CLOCK_50);
119       KEY[0] <= 1;                                 @(posedge CLOCK_50);
120                                                    @(posedge CLOCK_50);
```

```
121         KEY[0] <= 0;                                    @(posedge CLOCK_50);
122                                                         @(posedge CLOCK_50);
123         KEY[0] <= 1;                                    @(posedge CLOCK_50);
124                                                         @(posedge CLOCK_50);
125         KEY[0] <= 0;                                    @(posedge CLOCK_50);
126                                                         @(posedge CLOCK_50);
127         KEY[0] <= 1;                                    @(posedge CLOCK_50);
128                                                         @(posedge CLOCK_50);
129         KEY[0] <= 0;                                    @(posedge CLOCK_50);
130                                                         @(posedge CLOCK_50);
131         KEY[0] <= 1;                                    @(posedge CLOCK_50);
132                                                         @(posedge CLOCK_50);
133         KEY[0] <= 0;                                    @(posedge CLOCK_50);
134                                                         @(posedge CLOCK_50);
135         KEY[0] <= 1;                                    @(posedge CLOCK_50);
136                                                         @(posedge CLOCK_50);
137         KEY[0] <= 0;                                    @(posedge CLOCK_50);
138                                                         @(posedge CLOCK_50);
139                                                         @(posedge CLOCK_50);
140                                                         @(posedge CLOCK_50);
141                                                         @(posedge CLOCK_50);
142         SW[9] <= 1;                                     @(posedge CLOCK_50);
143                                                         @(posedge CLOCK_50);
144         SW[9] <= 0;                                     @(posedge CLOCK_50);
145                                                         @(posedge CLOCK_50);
146         KEY[0] <= 0; SW[8:0] <= 9'b0111111110;          @(posedge CLOCK_50);
147                                                         @(posedge CLOCK_50);
148                                                         @(posedge CLOCK_50);
149                                                         @(posedge CLOCK_50);
150                                                         @(posedge CLOCK_50);
151                                                         @(posedge CLOCK_50);
152                                                         @(posedge CLOCK_50);
153                                                         @(posedge CLOCK_50);
154                                                         @(posedge CLOCK_50);
155                                                         @(posedge CLOCK_50);
156         $stop;
157     end
158 endmodule
```