```
1   // Eugene Ngo
2   // 11/18/2022
3   // EE 271
4   // Lab 4
5
6   // DE1_SoC is the top level module. It takes in a 3-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      output logic  [6:0] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
14      output logic  [9:0] LEDR;
15      input logic   [3:0] KEY; // True when not pressed, False when pressed
16      input logic   [9:0] SW;
17
18
19      // logic reset;
20      logic key0, key3;
21      logic key0_Stable, key3_Stable;
22
23      // Assign HEX 5 - 1 to default display
24
25      assign HEX5 = 7'b1111111;
26      assign HEX4 = 7'b1111111;
27      assign HEX3 = 7'b1111111;
28      assign HEX2 = 7'b1111111;
29      assign HEX1 = 7'b1111111;
30
31      // Instantiates dff modules to process button inputs at clock cycles
32      // doubledFlip module takes in CLOCK_50 to clk, SW[9] to reset, and KEY[0 or 3] to
    button as inputs
33      // and returns out as clocked inputs
34
35      doubleFlip ff1 (.clk(CLOCK_50), .reset(SW[9]), .button(~KEY[0]), .out(key0_Stable));
36      doubleFlip ff2 (.clk(CLOCK_50), .reset(SW[9]), .button(~KEY[3]), .out(key3_Stable));
37
38      // Instantiates user input modules with switch zero and three for user inputs
39      // userInput module takes in CLOCK_50 to clk, SW[9] to reset, and and clocked inputs to
    button as inputs
40      // and returns button states to out
41
42      userInput switchZero (.clk(CLOCK_50), .reset(SW[9]), .button(key0_Stable), .out(key0));
43      userInput switchThree (.clk(CLOCK_50), .reset(SW[9]), .button(key3_Stable), .out(key3));
44
45      // Light instantiations
46      // The light modules takes in CLOCK_50 to clk, SW[9] to reset, key3 to L, key 0 to R,
    LEDR[9:0] to NL, LEDR[0:9] to NR
47      // and returns LEDR[0:9] to lightOn.
48
49      normalLight one (.clk(CLOCK_50), .reset(SW[9]), .L(key3), .R(key0), .NL(LEDR[2]), .NR(
    1'b0), .lightOn(LEDR[1]));
50      normalLight two (.clk(CLOCK_50), .reset(SW[9]), .L(key3), .R(key0), .NL(LEDR[3]), .NR(
    LEDR[1]), .lightOn(LEDR[2]));
51      normalLight three (.clk(CLOCK_50), .reset(SW[9]), .L(key3), .R(key0), .NL(LEDR[4]), .NR(
    LEDR[2]), .lightOn(LEDR[3]));
52      normalLight four (.clk(CLOCK_50), .reset(SW[9]), .L(key3), .R(key0), .NL(LEDR[5]), .NR(
    LEDR[3]), .lightOn(LEDR[4]));
53
54      // center light is a little different from the normalLight module so that when the game
    is reset, the LED light is recenetered
55      centerLight five (.clk(CLOCK_50), .reset(SW[9]), .L(key3), .R(key0), .NL(LEDR[6]), .NR(
    LEDR[4]), .lightOn(LEDR[5]));
56
57      normalLight six (.clk(CLOCK_50), .reset(SW[9]), .L(key3), .R(key0), .NL(LEDR[7]), .NR(
    LEDR[5]), .lightOn(LEDR[6]));
58      normalLight seven (.clk(CLOCK_50), .reset(SW[9]), .L(key3), .R(key0), .NL(LEDR[8]), .NR(
    LEDR[6]), .lightOn(LEDR[7]));
59      normalLight eight (.clk(CLOCK_50), .reset(SW[9]), .L(key3), .R(key0), .NL(LEDR[9]), .NR(
```

```
    LEDR[7]), .lightOn(LEDR[8]));
60      normalLight nine (.clk(CLOCK_50), .reset(SW[9]), .L(key3), .R(key0), .NL(1'b0), .NR(LEDR
    [8]), .lightOn(LEDR[9]));
61
62      // Determines who wins
63      // victory module takes in CLOCK_50 to clk, SW[9] to reset, LEDR[9] to LED9, LEDR[1] to
    LED1, key3 to L, key0 to R, and
64      // returns HEX0 to winner
65      victory gameEnds (.clk(CLOCK_50), .reset(SW[9]), .LED9(LEDR[9]), .LED1(LEDR[1]), .L(key3
    ), .R(key0), .winner(HEX0));
66
67
68  endmodule
69
70  //DE1_SoC_testbench tests all expected, unexpected, and edgecase behaviors that the Tug of
    War system implemented in this lab may encounter.
71
72  module DE1_SoC_testbench();
73     logic        CLOCK_50;
74     logic [6:0] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
75     logic [9:0] LEDR;
76     logic [3:0] KEY;
77     logic [9:0] SW;
78
79     DE1_SoC dut (.CLOCK_50, .HEX0, .HEX1, .HEX2, .HEX3, .HEX4, .HEX5, .KEY, .LEDR, .SW);
80
81     parameter CLOCK_PERIOD = 100;
82     initial begin
83        CLOCK_50 <= 0;
84        forever #(CLOCK_PERIOD / 2)
85        CLOCK_50 <= ~CLOCK_50;
86     end
87
88     initial begin
89                                        @(posedge CLOCK_50);
90                                        @(posedge CLOCK_50);
91        SW[9] <= 1;                     @(posedge CLOCK_50);
92                                        @(posedge CLOCK_50);
93                                        @(posedge CLOCK_50);
94                                        @(posedge CLOCK_50);
95        SW[9] <= 0;                     @(posedge CLOCK_50);
96                                        @(posedge CLOCK_50);
97                                        @(posedge CLOCK_50);
98                                        @(posedge CLOCK_50);
99        KEY[0] <= 0; KEY[3] <= 0;       @(posedge CLOCK_50);
100                                       @(posedge CLOCK_50);
101       KEY[0] <= 1;                    @(posedge CLOCK_50);
102                                       @(posedge CLOCK_50);
103       KEY[0] <= 0;                    @(posedge CLOCK_50);
104                                       @(posedge CLOCK_50);
105       KEY[0] <= 1;                    @(posedge CLOCK_50);
106                                       @(posedge CLOCK_50);
107       KEY[0] <= 0;                    @(posedge CLOCK_50);
108                                       @(posedge CLOCK_50);
109       KEY[0] <= 1;                    @(posedge CLOCK_50);
110                                       @(posedge CLOCK_50);
111       KEY[0] <= 0;                    @(posedge CLOCK_50);
112                                       @(posedge CLOCK_50);
113       KEY[0] <= 1;                    @(posedge CLOCK_50);
114                                       @(posedge CLOCK_50);
115       KEY[0] <= 0;                    @(posedge CLOCK_50);
116                                       @(posedge CLOCK_50);
117       KEY[0] <= 1;                    @(posedge CLOCK_50);
118                                       @(posedge CLOCK_50);
119       KEY[0] <= 0;                    @(posedge CLOCK_50);
120                                       @(posedge CLOCK_50);
121       KEY[0] <= 1;                    @(posedge CLOCK_50);
122                                       @(posedge CLOCK_50);
123       KEY[0] <= 0;                    @(posedge CLOCK_50);
124                                       @(posedge CLOCK_50);
125       KEY[0] <= 1;                    @(posedge CLOCK_50);
126                                       @(posedge CLOCK_50);
127                                       @(posedge CLOCK_50);
```

```
                                    @(posedge CLOCK_50);
                                    @(posedge CLOCK_50);
      KEY[0] <= 0;                  @(posedge CLOCK_50);
                                    @(posedge CLOCK_50);
      KEY[0] <= 1;                  @(posedge CLOCK_50);
                                    @(posedge CLOCK_50);
      KEY[0] <= 0; KEY[3] <= 1;     @(posedge CLOCK_50);
                                    @(posedge CLOCK_50);
                   KEY[3] <= 0;     @(posedge CLOCK_50);
                                    @(posedge CLOCK_50);
                   KEY[3] <= 1;     @(posedge CLOCK_50);
                                    @(posedge CLOCK_50);
                   KEY[3] <= 0;     @(posedge CLOCK_50);
                                    @(posedge CLOCK_50);
                   KEY[3] <= 1;     @(posedge CLOCK_50);
                                    @(posedge CLOCK_50);
                   KEY[3] <= 0;     @(posedge CLOCK_50);
                                    @(posedge CLOCK_50);
                   KEY[3] <= 1;     @(posedge CLOCK_50);
                                    @(posedge CLOCK_50);
                   KEY[3] <= 0;     @(posedge CLOCK_50);
                                    @(posedge CLOCK_50);
                   KEY[3] <= 1;     @(posedge CLOCK_50);
                                    @(posedge CLOCK_50);
                   KEY[3] <= 0;     @(posedge CLOCK_50);
                                    @(posedge CLOCK_50);
                   KEY[3] <= 1;     @(posedge CLOCK_50);
                                    @(posedge CLOCK_50);
                   KEY[3] <= 0;     @(posedge CLOCK_50);
                                    @(posedge CLOCK_50);
                   KEY[3] <= 1;     @(posedge CLOCK_50);
                                    @(posedge CLOCK_50);
                   KEY[3] <= 0;     @(posedge CLOCK_50);
                                    @(posedge CLOCK_50);
                   KEY[3] <= 1;     @(posedge CLOCK_50);
                                    @(posedge CLOCK_50);
                   KEY[3] <= 0;     @(posedge CLOCK_50);
                                    @(posedge CLOCK_50);
                   KEY[3] <= 1;     @(posedge CLOCK_50);
                                    @(posedge CLOCK_50);
                   KEY[3] <= 0;     @(posedge CLOCK_50);
                                    @(posedge CLOCK_50);
      $stop;
   end
endmodule
```