```
1   module DE1_SoC_2 (CLOCK_50, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, KEY, LEDR, SW);
2       input logic CLOCK_50; // 50MHz clock.
3       output logic [6:0] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
4       output logic [9:0] LEDR;
5       input logic [3:0] KEY; // True when not pressed, False when pressed
6       input logic [9:0] SW;
7
8       // Generate clk off of CLOCK_50, whichClock picks rate.
9
10      logic reset;
11      logic [31:0] div_clk;
12
13  //  assign reset = SW[9];
14  //  parameter whichClock = 0; // 25MHz clock
15      clock_divider cdiv (.clock(CLOCK_50),
16                                  .reset(reset),
17                                  .divided_clocks(div_clk));
18
19      // Clock selection; allows for easy switching between simulation and board
20      // clocks
21      logic key0, key3, input0, input3;
22
23
24      assign HEX1 = 7'b1111111;
25      assign HEX2 = 7'b1111111;
26      assign HEX3 = 7'b1111111;
27      assign HEX4 = 7'b1111111;
28      assign HEX5 = 7'b1111111;
29
30      DFlipFlop Flip0 (.clk(CLOCK_50), .reset(SW[9]), .key(~KEY[0]), .out(input0));
31      DFlipFlop Flip3 (.clk(CLOCK_50), .reset(SW[9]), .key(~KEY[3]), .out(input3));
32
33      UserInput switch0 (.clk(CLOCK_50), .reset(SW[9]), .key(input0), .out(key0));
34      UserInput switch3 (.clk(CLOCK_50), .reset(SW[9]), .key(input3), .out(key3));
35
36      normalLight L1 (.clk(CLOCK_50), .reset(SW[9]), .L(key3), .R(key0), .NL(LEDR[2]), .NR(
        1'b0), .lightOn(LEDR[1]));
37      normalLight L2 (.clk(CLOCK_50), .reset(SW[9]), .L(key3), .R(key0), .NL(LEDR[3]), .NR(
        LEDR[1]), .lightOn(LEDR[2]));
38      normalLight L3 (.clk(CLOCK_50), .reset(SW[9]), .L(key3), .R(key0), .NL(LEDR[4]), .NR(
        LEDR[2]), .lightOn(LEDR[3]));
39      normalLight L4 (.clk(CLOCK_50), .reset(SW[9]), .L(key3), .R(key0), .NL(LEDR[5]), .NR(
        LEDR[3]), .lightOn(LEDR[4]));
40      centerLight L5 (.clk(CLOCK_50), .reset(SW[9]), .L(key3), .R(key0), .NL(LEDR[6]), .NR(
        LEDR[4]), .lightOn(LEDR[5]));
41      normalLight L6 (.clk(CLOCK_50), .reset(SW[9]), .L(key3), .R(key0), .NL(LEDR[7]), .NR(
        LEDR[5]), .lightOn(LEDR[6]));
42      normalLight L7 (.clk(CLOCK_50), .reset(SW[9]), .L(key3), .R(key0), .NL(LEDR[8]), .NR(
        LEDR[6]), .lightOn(LEDR[7]));
43      normalLight L8 (.clk(CLOCK_50), .reset(SW[9]), .L(key3), .R(key0), .NL(LEDR[9]), .NR(
        LEDR[7]), .lightOn(LEDR[8]));
44      normalLight L9 (.clk(CLOCK_50), .reset(SW[9]), .L(key3), .R(key0), .NL(1'b0), .NR(
        LEDR[8]), .lightOn(LEDR[9]));
45
46      winner displayWinner(.clk(CLOCK_50), .reset(SW[9]), .LED9(LEDR[9]), .LED1(LEDR[1]), .
        L(key3), .R(key0), .user(HEX0));
47
48
49  endmodule
50
51  module DE1_SoC_2_testbench();
52      logic CLOCK_50;
53      logic [6:0] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
54      logic [9:0] LEDR;
55      logic [3:0] KEY;
56      logic [9:0] SW;
57
58      DE1_SoC_2 dut (.CLOCK_50, .HEX0, .HEX1, .HEX2, .HEX3, .HEX4, .HEX5, .KEY, .LEDR, .SW
        );
```

```
59
60      parameter CLOCK_PERIOD = 100;
61      initial begin
62          CLOCK_50 <= 0;
63          forever #(CLOCK_PERIOD / 2)
64          CLOCK_50 <= ~CLOCK_50;
65      end
66
67      initial begin
68          SW[9] <= 1; @(posedge CLOCK_50);
69          SW[9] <= 2; @(posedge CLOCK_50);
70          KEY[0] <= 0; KEY[3] <= 0;       repeat(2) @(posedge CLOCK_50);
71          KEY[0] <= 1;                        repeat(2) @(posedge CLOCK_50);
72          KEY[0] <= 0;                        repeat(2) @(posedge CLOCK_50);
73          KEY[0] <= 1;                        repeat(2) @(posedge CLOCK_50);
74          KEY[0] <= 0;                        repeat(2) @(posedge CLOCK_50);
75          KEY[0] <= 1;                        repeat(2) @(posedge CLOCK_50);
76          KEY[0] <= 0;                        repeat(2) @(posedge CLOCK_50);
77          KEY[0] <= 1;                        repeat(2) @(posedge CLOCK_50);
78          KEY[0] <= 0;                        repeat(2) @(posedge CLOCK_50);
79          KEY[0] <= 1;                        repeat(2) @(posedge CLOCK_50);
80          KEY[0] <= 0;                        repeat(2) @(posedge CLOCK_50);
81          KEY[0] <= 1;                        repeat(2) @(posedge CLOCK_50);
82          KEY[0] <= 0;                        repeat(2) @(posedge CLOCK_50);
83          KEY[0] <= 1;                        repeat(2) @(posedge CLOCK_50);
84          KEY[0] <= 0;                        repeat(2) @(posedge CLOCK_50);
85          KEY[0] <= 1;                        repeat(2) @(posedge CLOCK_50);
86          KEY[0] <= 0; KEY[3] <= 1;       repeat(2) @(posedge CLOCK_50);
87          KEY[3] <= 1;                        repeat(2) @(posedge CLOCK_50);
88          KEY[3] <= 0;                        repeat(2) @(posedge CLOCK_50);
89          KEY[3] <= 1;                        repeat(2) @(posedge CLOCK_50);
90          KEY[3] <= 0;                        repeat(2) @(posedge CLOCK_50);
91          KEY[3] <= 1;                        repeat(2) @(posedge CLOCK_50);
92          KEY[3] <= 0;                        repeat(2) @(posedge CLOCK_50);
93          KEY[3] <= 1;                        repeat(2) @(posedge CLOCK_50);
94          KEY[3] <= 0;                        repeat(2) @(posedge CLOCK_50);
95          KEY[3] <= 1;                        repeat(2) @(posedge CLOCK_50);
96          KEY[3] <= 0;                        repeat(2) @(posedge CLOCK_50);
97          KEY[3] <= 1;                        repeat(2) @(posedge CLOCK_50);
98          KEY[3] <= 0;                        repeat(2) @(posedge CLOCK_50);
99          KEY[3] <= 1;                        repeat(2) @(posedge CLOCK_50);
100         KEY[3] <= 0;                        repeat(2) @(posedge CLOCK_50);
101         KEY[3] <= 1;                        repeat(2) @(posedge CLOCK_50);
102         KEY[3] <= 0;                        repeat(2) @(posedge CLOCK_50);
103
104         $stop;
105     end
106 endmodule
```