```
1   // Alan Li
2   // 01/15/2022
3   // EE 371
4   // Lab #1
5
6   // DE1_Soc takes 34-bit GPIO_0 and return 1-bit enterIndicator and
    exitIndicator, 5-bit countIndicator and inputA, inputB
7   // This serves as the top-level module for the parking lot occupancy
    counter system
8
9   module DE1_SoC(CLOCK_50, GPIO_0, HEX5, HEX4, HEX3, HEX2, HEX1, HEX0);
10
11      // GPIO_0[5] is connected to the reset switch, GPIO_0[6] is
    connected to the sensorA switch, GPIO_0[7] is connected to the sensorB
    switch
12      inout logic [33:0] GPIO_0;
13      input logic CLOCK_50;
14      output logic [6:0] HEX5, HEX4, HEX3, HEX2, HEX1, HEX0;
15      logic enterIndicator, exitIndicator;
16      logic [4:0] countIndicator;
17      logic inputA, inputB;
18
19
20      assign GPIO_0[26] = GPIO_0[6]; // when switch A(sensor A) is
    enabled/triggered the corresponding LED turns on
21      assign GPIO_0[27] = GPIO_0[7]; // when switch B(sensor B) is
    enabled/triggered the corresponding LED turns on
22
23      // userInput inA and inB takes GPIO[6] and GPIO[7] as input
    parameters and return Q as inputA and inputB respectively
24      userInput inA (.clk(CLOCK_50), .D(GPIO_0[6]), .Q(inputA));
25      userInput inB (.clk(CLOCK_50), .D(GPIO_0[7]), .Q(inputB));
26
27      // parkingLotSensors myfsm takes CLOCK_50 as clk, GPIO[5], inputA,
    input B as parameters to reset, A, B
28      // and returns enter and exit as enterIndicator and exitIndicator
    respectively
29      parkingLotSensors myfsm (.clk(CLOCK_50), .reset(GPIO_0[5]), .A(inputA
    ), .B(inputB), .enter(enterIndicator), .exit(exitIndicator));
30
31      // counter mycounter takes CLOCK_50 as clk, GPIO_0[5],
    enterIndicator and exitIndicator as reset, enter and exit
32      // it returns enter, exit and cout as countIndicator[4:0]
33      counter mycounter (.clk(CLOCK_50), .reset(GPIO_0[5]), .enter(
    enterIndicator), .exit(exitIndicator), .cout(countIndicator[4:0]));
34
35      // hexDisplay mydisplay takes CLOCK_50 as clk, countIndicator[4:0]
    as inputCount
36      // and returns HEX5, HEX4, HEX3, HEX2, HEX1, HEX0 as HEX5, HEX4,
    HEX3, HEX2, HEX1, HEX0 respectively
37      hexDisplay mydisplay (.clk(CLOCK_50), .inputCount(countIndicator[4:0
    ]),
38                            .HEX5(HEX5), .HEX4(HEX4), .HEX3(HEX3), .HEX2(
    HEX2), .HEX1(HEX1), .HEX0(HEX0));
39  endmodule
40
41
42  // DE1_SoC_testbench tests all expected behavior that the parking lot
```

```
    occupancy counter system in the lab may encounter
43  module DE1_SoC_testbench();
44
45     logic clk, reset, A, B;
46     logic [6:0] HEX5, HEX4, HEX3, HEX2, HEX1, HEX0;
47
48     DE1_SoC dut (.CLOCK_50(CLOCK_50), .GPIO_0(GPIO_0),
49                             .HEX5(HEX5), .HEX4(HEX4), .HEX3(HEX3), .HEX2(
    HEX2), .HEX1(HEX1), .HEX0(HEX0));
50
51     //Set up the clock.
52     parameter CLOCK_PERIOD = 100;
53
54     initial begin
55        clk <= 0;
56        forever #(CLOCK_PERIOD/2) clk <= ~clk;
57
58     end
59
60     initial begin
61        reset <= 1;       @(posedge clk);   // cycle through car entering
62        reset <= 0;       @(posedge clk);
63                          @(posedge clk);
64                          @(posedge clk);
65        {A,B} <= 2'b00;   @(posedge clk);
66        {A,B} <= 2'b10;   @(posedge clk);
67        {A,B} <= 2'b11;   @(posedge clk);
68        {A,B} <= 2'b01;   @(posedge clk);
69        {A,B} <= 2'b00;   @(posedge clk);
70
71        {A,B} <= 2'b00;   @(posedge clk);
72        {A,B} <= 2'b10;   @(posedge clk);
73        {A,B} <= 2'b11;   @(posedge clk);
74        {A,B} <= 2'b01;   @(posedge clk);
75        {A,B} <= 2'b00;   @(posedge clk);
76
77        {A,B} <= 2'b00;   @(posedge clk);
78        {A,B} <= 2'b10;   @(posedge clk);
79        {A,B} <= 2'b11;   @(posedge clk);
80        {A,B} <= 2'b01;   @(posedge clk);
81        {A,B} <= 2'b00;   @(posedge clk);
82
83        {A,B} <= 2'b00;   @(posedge clk);
84        {A,B} <= 2'b10;   @(posedge clk);
85        {A,B} <= 2'b11;   @(posedge clk);
86        {A,B} <= 2'b01;   @(posedge clk);
87        {A,B} <= 2'b00;   @(posedge clk);   // 4 cars entered
88
89        {A,B} <= 2'b00;   @(posedge clk);
90        {A,B} <= 2'b01;   @(posedge clk);
91        {A,B} <= 2'b11;   @(posedge clk);
92        {A,B} <= 2'b10;   @(posedge clk);
93        {A,B} <= 2'b00;   @(posedge clk);
94
95        {A,B} <= 2'b00;   @(posedge clk);
96        {A,B} <= 2'b01;   @(posedge clk);
97        {A,B} <= 2'b11;   @(posedge clk);
98        {A,B} <= 2'b10;   @(posedge clk);
```

```
 99          {A,B} <= 2'b00;    @(posedge clk);
100
101          {A,B} <= 2'b00;    @(posedge clk);
102          {A,B} <= 2'b01;    @(posedge clk);
103          {A,B} <= 2'b11;    @(posedge clk);
104          {A,B} <= 2'b10;    @(posedge clk);
105          {A,B} <= 2'b00;    @(posedge clk);    // 3 cars exiting
106
107          reset <= 1;        @(posedge clk);    // cycle through car exiting
108          reset <= 0;        @(posedge clk);
109                             @(posedge clk);
110                             @(posedge clk);
111          {A,B} <= 2'b00;    @(posedge clk);
112          {A,B} <= 2'b01;    @(posedge clk);
113          {A,B} <= 2'b11;    @(posedge clk);
114          {A,B} <= 2'b10;    @(posedge clk);
115          {A,B} <= 2'b00;    @(posedge clk);
116                             @(posedge clk);
117          $stop;
118       end
119    endmodule
120
```