```
1   /* Name: Eugene Ngo
2      Date: 1/13/2023
3      Class: EE 371
4      Lab 1: Parking Lot Occupancy Counter*/
5
6   // DE1_SoC is the top-level module that defines the I/Os for the DE-1 SoC board.
7   // DE1_SoC takes three switches from the GPIO as inputs, and outputs to 2 LEDs on the
    breadboard through GPIO and 6 7-bit
8   // hex displays (HEX0-HEX5). It displays "full" or "clear" messages when the parking lot is
    either full or clear, and it
9   // displays the decimal value of the number of cars in the lot accordingly.
10
11  module DE1_SoC #(parameter MAX=25) (HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, GPIO_0, CLOCK_50);
12      output logic [6:0]  HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
13      inout logic [33:0] GPIO_0;
14      input logic CLOCK_50;
15
16     // Assigning and clk to CLOCK_50
17     logic clk;
18     assign clk = CLOCK_50;
19
20     logic enter, exit, full, clear;
21     logic [4:0] counter_out;
22
23     // Outputting a and b to breadboard
24     assign GPIO_0[26] = GPIO_0[5];
25     assign GPIO_0[27] = GPIO_0[7];
26
27     // carSensor parkCheck takes two switches from the breadboard as the input of the two
    parking sensors,
28     // and outputs to enter and exit when an entering or exiting vehicle is detected.
29     carSensor parkCheck (.a(GPIO_0[7]), .b(GPIO_0[5]), .enter, .exit, .clk, .reset(GPIO_0[9
    ]));
30
31     // carCount counter takes enter and exit from sensors, and outputs the car count to
    counter_out.
32     // it als outputs full and clear status to full and clear.
33     carCount #(MAX) counter (.inc(enter), .dec(exit), .out(counter_out), .full, .clear, .clk,
     .reset(GPIO_0[9]));
34
35     // seg7 display takes counter_out, full, and clear from ct, and outputs decimal values
    or status messages to hex displays.
36     seg7 display (.in(counter_out), .full, .clear, .hexout0(HEX0), .hexout1(HEX1), .hexout2(
    HEX2), .hexout3(HEX3), .hexout4(HEX4), .hexout5(HEX5));
37
38  endmodule // DE1_SoC
39
40  // DE1_SoC_testbench tests all expected, unexpected, and edgecase behaviors
41  module DE1_SoC_testbench();
42     logic [6:0]  HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
43     wire [33:0] GPIO_0;
44     logic CLOCK_50;
45
46     DE1_SoC #(5) dut (.HEX0, .HEX1, .HEX2, .HEX3, .HEX4, .HEX5, .GPIO_0, .CLOCK_50);
47
48     // Setting up a simulated clock.
49     parameter CLOCK_PERIOD = 100;
50     initial begin
51        CLOCK_50 <= 0;
52        forever #(CLOCK_PERIOD/2) CLOCK_50 <= ~CLOCK_50; // Forever toggle the clock
53     end
54
55     // Assiging logic to wire
56     logic reset, a, b;
57     assign GPIO_0[9] = reset;
58     assign GPIO_0[7] = a;
59     assign GPIO_0[5] = b;
60
61     // Testing the module
62     initial begin
63        // reset
64        reset <= 1;                    repeat(3) @(posedge CLOCK_50);
65
```

```
66          //enters until full
67          reset <= 0;       a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50);
68                            a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
69                            a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
70                            a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
71                            a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 1st car enters
72                            a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
73                            a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
74                            a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
75                            a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 2nd car enters
76                            a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
77                            a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
78                            a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
79                            a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 3rd car enters
80                            a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
81                            a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
82                            a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
83                            a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 4th car enters
84                            a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
85                            a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
86                            a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
87                            a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 5th car enters, full
88
89      // exits until clear
90                            a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50);
91                            a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
92                            a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
93                            a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
94                            a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 1st car exits
95                            a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
96                            a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
97                            a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
98                            a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 2nd car exits
99                            a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
100                           a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
101                           a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
102                           a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 3rd car exits
103                           a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
104                           a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
105                           a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
106                           a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 4th car exits
107                           a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
108                           a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
109                           a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
110                           a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 5th car exits, clear
111
112         // direction changes while entering
113                           a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50);
114                           a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
115                           a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
116                           a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
117                           a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
118                           a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
119                           a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
120                           a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
121                           a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50);
122         // direction changes while exiting
123                           a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50);
124                           a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
125                           a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
126                           a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
127                           a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
128                           a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
129                           a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
130                           a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
131                           a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50);
132
133         $stop;
134     end
135 endmodule // DE1_SoC_testbench
```