```
1   /* Name: Eugene Ngo
2      Date: 3/7/2023
3      Class: EE 371
4      Lab 6: Parking Lot 3D Simulation
5   */
6
7   // carCount takes 6 inputs (park1, park2, park3, full, clk, reset.
8   // It sets the state of the number of cars currently left based on
9   // the number of spots available. There is no incrementing or decrementing
10  // of an existing number, this is purely just sequential unit with a moore-like logic
11  // that outputs the number of spots available based on the number
12  // of spots taken. This is derived from the carCount from part 1
13  // therefore, the parameter is kept, but it is not a requirement.
14  `timescale 1 ps / 1 ps
15  module carCount #(parameter MAX=3) (park1, park2, park3, full, clk, reset, out);
16
17     input logic park1, park2, park3, full, clk, reset;
18     output logic [1:0] out;
19
20     // Sequential logic for setting the number of spaces left based on the
21     // parking sensors..
22     always_ff @(posedge clk) begin
23        // If everything is reset, set output to 3 spots left
24        if (reset) begin
25           out <= 2'b11;
26        end
27        // If 0 spots taken, output 3 spots left.
28        if (~park1 & ~park2 & ~park3) begin
29           out <= 2'b11;
30        end
31        // If 1 spot taken, output 2 spots left
32        else if ((park1 & ~park2 & ~park3) | (~park1 & park2 & ~park3) | (~park1 & ~park2 &
    park3)) begin
33           out <= 2'b10;
34        end
35        // If 2 spots taken, output 1 spot left
36        else if ((park1 & park2 & ~park3) | (~park1 & park2 & park3) | (park1 & ~park2 & park3
    )) begin // decrement when not at min
37           out <= 2'b01;
38        end
39        // If 3 spots taken, output 0 spot left (datapath should then convert this to "FULL").
40        else if (full == 1'b1) begin
41           out <= 2'b00;
42        end
43        else
44           out <= out; // hold value otherwise
45     end // always_ff
46
47  endmodule
48
49  // carCount_testbench tests all expected, unexpected, and edgecase behaviors
50  // to ensure the module updates the current number of spots available based on the number
51  // of cars in the parking lot.
52  module carCount_testbench();
53     // Same I/O as carCount()
54     logic park1, park2, park3, full, clk, reset;
55     logic [1:0] out;
56     logic CLOCK_50;
57
58     carCount #(3) dut (.park1, .park2, .park3, .full, .clk(CLOCK_50), .reset, .out);
59
60     // Setting up the clock.
61     parameter CLOCK_PERIOD = 100;
62     initial begin
63        CLOCK_50 <= 0;
64        forever #(CLOCK_PERIOD/2) CLOCK_50 <= ~CLOCK_50; // toggle the clock forever
65     end // initial
66
67     initial begin
68        reset <= 1;                @(posedge CLOCK_50); // reset
69        reset <= 0;                @(posedge CLOCK_50); // inc past max limit
70        park1 <= 0;                @(posedge CLOCK_50); // dec past min limit
71        park1 <= 1;                @(posedge CLOCK_50); // dec past min limit
```

```
72          park2 <= 0;                 @(posedge CLOCK_50); // dec past min limit
73          park2 <= 1;                 @(posedge CLOCK_50); // dec past min limit
74          park3 <= 0;                 @(posedge CLOCK_50); // dec past min limit
75          park3 <= 1;                 @(posedge CLOCK_50); // dec past min limit
76          park3 <= 0;                 @(posedge CLOCK_50); // dec past min limit
77          park2 <= 0;                 @(posedge CLOCK_50); // dec past min limit
78          park3 <= 1;                 @(posedge CLOCK_50); // dec past min limit
79          park1 <= 0;                 @(posedge CLOCK_50); // dec past min limit
80          park3 <= 0;                 @(posedge CLOCK_50); // dec past min limit
81          $stop;
82       end
83    endmodule // counter_testbench
```