```
1   // An Nguyen, Darren Do
2   // 3/7/24
3   // EE 371
4   // LAB6, Task#2
5   //
6   // Inputs:
7   // car1: Signals whether parking spot 1 is taken.
8   // car2: Signals whether parking spot 2 is taken.
9   // car3: Signals whether parking spot 3 is taken.
10  //
11  // Output:
12  // 2-bit spot_left: Returns how many spots are available in the parking lot.
13  //
14  // Logic:
15  // 2-bit current_occ: intermediate logic to keep track of how many cars are present.
16  // Summary: Module 'count_spots' is designed to keep track of how many spots are currently
17  //          available in our 3D parking lot.
18  module count_spots(car1, car2, car3, spot_left);
19     input logic car1, car2, car3;
20     output logic [1:0] spot_left;
21     logic [1:0] current_occ;
22
23     // logic to determine how many cars are present
24     // '0' represents open spot
25     // '1' represents parked spot
26     always_comb begin
27          case({car1, car2, car3})
28              3'b111: current_occ = 0;
29              3'b101: current_occ = 1;
30              3'b000: current_occ = 3;
31              3'b010: current_occ = 2;
32              3'b011: current_occ = 1;
33              3'b110: current_occ = 1;
34              3'b001: current_occ = 2;
35              3'b100: current_occ = 2;
36          endcase
37     end
38
39     // Assign intermediate logic to final output logic
40     assign spot_left = current_occ;
41
42  endmodule
43
44  // This Testbench is testing every single combination of car parking pattern
45  // to ensure that our counter can account for all possible cases in terms of cars
46  // entering the parking lot to determine how many parking spots are left.
47  module count_spots_testbench();
48     logic clk;
49     logic car1, car2, car3;
50     logic [1:0] spot_left;
51
52     count_spots dut1(.*);
53
54     parameter clock_period = 100;
55     initial begin
56        clk <= 0;
57        forever #(clock_period /2) clk <= ~clk;
58
59     end
60
61     // testing all combinations of car entry
62     // 1, 2, 3 spots available in different configurations
63     initial begin
64        car1 <= 0; car2 <=0; car3 <=0; @(posedge clk);
65        car1 <= 0; car2 <=0; car3 <=1; @(posedge clk);
66        car1 <= 0; car2 <=1; car3 <=0; @(posedge clk);
67        car1 <= 0; car2 <=1; car3 <=1; @(posedge clk);
68        car1 <= 1; car2 <=0; car3 <=0; @(posedge clk);
69        car1 <= 1; car2 <=0; car3 <=1; @(posedge clk);
70        car1 <= 1; car2 <=1; car3 <=0; @(posedge clk);
71        car1 <= 1; car2 <=1; car3 <=1; @(posedge clk);
72        $stop;
73     end
74  endmodule
```