```
1   // An Nguyen, Darren Do
2   // 3/7/24
3   // EE 371
4   // LAB 6, task#2
5   //
6   // Inputs:
7   // clk: Timer to indicate the functionality of our module.
8   // reset: Resets our counter back to '0'.
9   // in: Indicates that a car has entered our parking lot.
10  // full: Indicates that no parking spots are open, nothing should be counted in this case
11  //
12  // Outputs:
13  // 16-bit car_in: Outputs the total number of cars in our parking lot beginning at hour
14  //                '0' to hour '7'.
15  //
16  // Logic:
17  // 16-bit car_in_internal: Keeps track of the number of cars in our parking lot beginning
    at hour
18  //                '0' to hour '7'.
19  module car_Tracker(clk, reset, in, full, car_in);
20     input logic clk, reset, in, full;
21     output logic [15:0] car_in;
22     logic [15:0] car_in_internal;
23
24     // if reset, the car count goes to '0'.
25     // else, we keep couting when a car enters.
26     always_ff @(posedge clk) begin
27        if (reset)
28           car_in_internal <= 0;
29        else begin
30           if (in && ~full)
31              car_in_internal <=  car_in_internal + 1;
32        end
33     end
34
35     assign car_in = car_in_internal;
36  endmodule
37
38
39  // This testbench tests for 2 different cases:
40  // 1.) Cars are coming in and the parking lot is not full.
41  //     In this case, the counter should go up.
42  // 2.) Cars are coming in and the parking lot is full.
43  //     In this case, the counter should NOT go up.
44  module car_Tracker_testbench();
45     logic clk, reset, in, full;
46     logic [15:0] car_in;
47
48     car_Tracker dut1(.*);
49
50     parameter clock_period = 100;
51     initial begin
52        clk <= 0;
53        forever #(clock_period /2) clk <= ~clk;
54
55     end
56
57     initial begin
58        reset <= 1; in <= 0; full <= 0; @(posedge clk);
59        reset <= 1; @(posedge clk);
60        reset <= 0; @(posedge clk);
61        // full is off, it should count
62        for (int i = 0; i < 5; i++) begin
63           in <= 1; @(posedge clk);
64        end
65        // full is off, it should NOT count
66        full <=  1; @(posedge clk);
67        for (int i = 0; i < 5; i++) begin
68           in <= 1; @(posedge clk);
69        end
70        $stop;
71     end
72
73  endmodule
```

74
75
76