```
/* Name: Eugene Ngo
   Date: 1/13/2023
   Class: EE 371
   Lab 1: Parking Lot Occupancy Counter*/

// DE1_SoC is the top-level module that defines the I/Os for the DE-1 SoC board.
// DE1_SoC takes three switches from the GPIO as inputs, and outputs to 2 LEDs on the breadboard through GPIO and 6 7-bit
// hex displays (HEX0-HEX5). It displays "full" or "clear" messages when the parking lot is either full or clear, and it
// displays the decimal value of the number of cars in the lot accordingly.

module DE1_SoC #(parameter MAX=25) (HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, GPIO_0, CLOCK_50);
    output logic [6:0]  HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
    inout logic [33:0] GPIO_0;
    input logic CLOCK_50;

   // Assigning and clk to CLOCK_50
   logic clk;
   assign clk = CLOCK_50;

   logic enter, exit, full, clear;
   logic [4:0] counter_out;

   // Outputting a and b to breadboard
   assign GPIO_0[26] = GPIO_0[5];
   assign GPIO_0[27] = GPIO_0[7];

   // carSensor parkCheck takes two switches from the breadboard as the input of the two parking sensors,
   // and outputs to enter and exit when an entering or exiting vehicle is detected.
   carSensor parkCheck (.a(GPIO_0[7]), .b(GPIO_0[5]), .enter, .exit, .clk, .reset(GPIO_0[9]));

   // carCount counter takes enter and exit from sensors, and outputs the car count to counter_out.
   // it als outputs full and clear status to full and clear.
   carCount #(MAX) counter (.inc(enter), .dec(exit), .out(counter_out), .full, .clear, .clk, .reset(GPIO_0[9]));

   // seg7 display takes counter_out, full, and clear from ct, and outputs decimal values or status messages to hex displays.
   seg7 display (.in(counter_out), .full, .clear, .hexout0(HEX0), .hexout1(HEX1), .hexout2(HEX2), .hexout3(HEX3), .hexout4(HEX4), .hexout5(HEX5));

endmodule // DE1_SoC

// DE1_SoC_testbench tests all expected, unexpected, and edgecase behaviors
module DE1_SoC_testbench();
   logic [6:0]  HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
   wire [33:0] GPIO_0;
   logic CLOCK_50;

   DE1_SoC #(5) dut (.HEX0, .HEX1, .HEX2, .HEX3, .HEX4, .HEX5, .GPIO_0, .CLOCK_50);

   // Setting up a simulated clock.
   parameter CLOCK_PERIOD = 100;
   initial begin
      CLOCK_50 <= 0;
      forever #(CLOCK_PERIOD/2) CLOCK_50 <= ~CLOCK_50; // Forever toggle the clock
   end

   // Assiging logic to wire
   logic reset, a, b;
   assign GPIO_0[9] = reset;
   assign GPIO_0[7] = a;
   assign GPIO_0[5] = b;

   // Testing the module
   initial begin
      // reset
      reset <= 1;                      repeat(3) @(posedge CLOCK_50);

      //enters until full
      reset <= 0;      a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50);
                       a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
                       a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 1st car enters
                       a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
                       a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 2nd car enters
                       a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
                       a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 3rd car enters
                       a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
                       a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 4th car enters
                       a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
                       a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 5th car enters, full

   // exits until clear
                       a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50);
                       a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
                       a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 1st car exits
                       a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
                       a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 2nd car exits
                       a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
                       a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 3rd car exits
                       a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
                       a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 4th car exits
                       a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
                       a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 5th car exits, clear

      // direction changes while entering
                       a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50);
                       a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
                       a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
                       a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50);
      // direction changes while exiting
                       a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50);
                       a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
                       a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
                       a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
                       a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50);

      $stop;
   end
endmodule // DE1_SoC_testbench
```