```
// Alan Li
// 01/15/2022
// EE 371
// Lab #1

// DE1_Soc takes 34-bit GPIO_0 and return 1-bit enterIndicator and exitIndicator, 5-bit countIndicator and inputA, inputB
// This serves as the top-level module for the parking lot occupancy counter system

module DE1_SoC(CLOCK_50, GPIO_0, HEX5, HEX4, HEX3, HEX2, HEX1, HEX0);

   // GPIO_0[5] is connected to the reset switch, GPIO_0[6] is connected to the sensorA switch, GPIO_0[7] is connected to the sensorB switch
   inout logic [33:0] GPIO_0;
   input logic CLOCK_50;
   output logic [6:0] HEX5, HEX4, HEX3, HEX2, HEX1, HEX0;
   logic enterIndicator, exitIndicator;
   logic [4:0] countIndicator;
   logic inputA, inputB;


   assign GPIO_0[26] = GPIO_0[6]; // when switch A(sensor A) is enabled/triggered the corresponding LED turns on
   assign GPIO_0[27] = GPIO_0[7]; // when switch B(sensor B) is enabled/triggered the corresponding LED turns on

   // userInput inA and inB takes GPIO[6] and GPIO[7] as input parameters and return Q as inputA and inputB respectively
   userInput inA (.clk(CLOCK_50), .D(GPIO_0[6]), .Q(inputA));
   userInput inB (.clk(CLOCK_50), .D(GPIO_0[7]), .Q(inputB));

   // parkingLotSensors myfsm takes CLOCK_50 as clk, GPIO[5], inputA, input B as parameters to reset, A, B
   // and returns enter and exit as enterIndicator and exitIndicator respectively
   parkingLotSensors myfsm (.clk(CLOCK_50), .reset(GPIO_0[5]), .A(inputA), .B(inputB), .enter(enterIndicator), .exit(exitIndicator));

   // counter mycounter takes CLOCK_50 as clk, GPIO_0[5], enterIndicator and exitIndicator as reset, enter and exit
   // it returns enter, exit and cout as countIndicator[4:0]
   counter mycounter (.clk(CLOCK_50), .reset(GPIO_0[5]), .enter(enterIndicator), .exit(exitIndicator), .cout(countIndicator[4:0]));

   // hexDisplay mydisplay takes CLOCK_50 as clk, countIndicator[4:0] as inputCount
   // and returns HEX5, HEX4, HEX3, HEX2, HEX1, HEX0 as HEX5, HEX4, HEX3, HEX2, HEX1, HEX0 respectively
   hexDisplay mydisplay (.clk(CLOCK_50), .inputCount(countIndicator[4:0]),
                         .HEX5(HEX5), .HEX4(HEX4), .HEX3(HEX3), .HEX2(HEX2), .HEX1(HEX1), .HEX0(HEX0));
endmodule


// DE1_SoC_testbench tests all expected behavior that the parking lot
```

```
occupancy counter system in the lab may encounter
module DE1_SoC_testbench();

   logic clk, reset, A, B;
   logic [6:0] HEX5, HEX4, HEX3, HEX2, HEX1, HEX0;

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

   //Set up the clock.
   parameter CLOCK_PERIOD = 100;

   initial begin
      clk <= 0;
      forever #(CLOCK_PERIOD/2) clk <= ~clk;

   end

   initial begin
      reset <= 1;       @(posedge clk);   // cycle through car entering
      reset <= 0;       @(posedge clk);
                        @(posedge clk);
                        @(posedge clk);
      {A,B} <= 2'b00;   @(posedge clk);
      {A,B} <= 2'b10;   @(posedge clk);
      {A,B} <= 2'b11;   @(posedge clk);
      {A,B} <= 2'b01;   @(posedge clk);
      {A,B} <= 2'b00;   @(posedge clk);

      {A,B} <= 2'b00;   @(posedge clk);
      {A,B} <= 2'b10;   @(posedge clk);
      {A,B} <= 2'b11;   @(posedge clk);
      {A,B} <= 2'b01;   @(posedge clk);
      {A,B} <= 2'b00;   @(posedge clk);

      {A,B} <= 2'b00;   @(posedge clk);
      {A,B} <= 2'b10;   @(posedge clk);
      {A,B} <= 2'b11;   @(posedge clk);
      {A,B} <= 2'b01;   @(posedge clk);
      {A,B} <= 2'b00;   @(posedge clk);

      {A,B} <= 2'b00;   @(posedge clk);
      {A,B} <= 2'b10;   @(posedge clk);
      {A,B} <= 2'b11;   @(posedge clk);
      {A,B} <= 2'b01;   @(posedge clk);
      {A,B} <= 2'b00;   @(posedge clk);   // 4 cars entered

      {A,B} <= 2'b00;   @(posedge clk);
      {A,B} <= 2'b01;   @(posedge clk);
      {A,B} <= 2'b11;   @(posedge clk);
      {A,B} <= 2'b10;   @(posedge clk);
      {A,B} <= 2'b00;   @(posedge clk);

      {A,B} <= 2'b00;   @(posedge clk);
      {A,B} <= 2'b01;   @(posedge clk);
      {A,B} <= 2'b11;   @(posedge clk);
      {A,B} <= 2'b10;   @(posedge clk);
      {A,B} <= 2'b00;   @(posedge clk);

      {A,B} <= 2'b00;   @(posedge clk);
      {A,B} <= 2'b01;   @(posedge clk);
      {A,B} <= 2'b11;   @(posedge clk);
      {A,B} <= 2'b10;   @(posedge clk);
      {A,B} <= 2'b00;   @(posedge clk);   // 3 cars exiting

      reset <= 1;       @(posedge clk);   // cycle through car exiting
      reset <= 0;       @(posedge clk);
                        @(posedge clk);
                        @(posedge clk);
      {A,B} <= 2'b00;   @(posedge clk);
      {A,B} <= 2'b01;   @(posedge clk);
      {A,B} <= 2'b11;   @(posedge clk);
      {A,B} <= 2'b10;   @(posedge clk);
      {A,B} <= 2'b00;   @(posedge clk);
                        @(posedge clk);
      $stop;
   end
endmodule
```