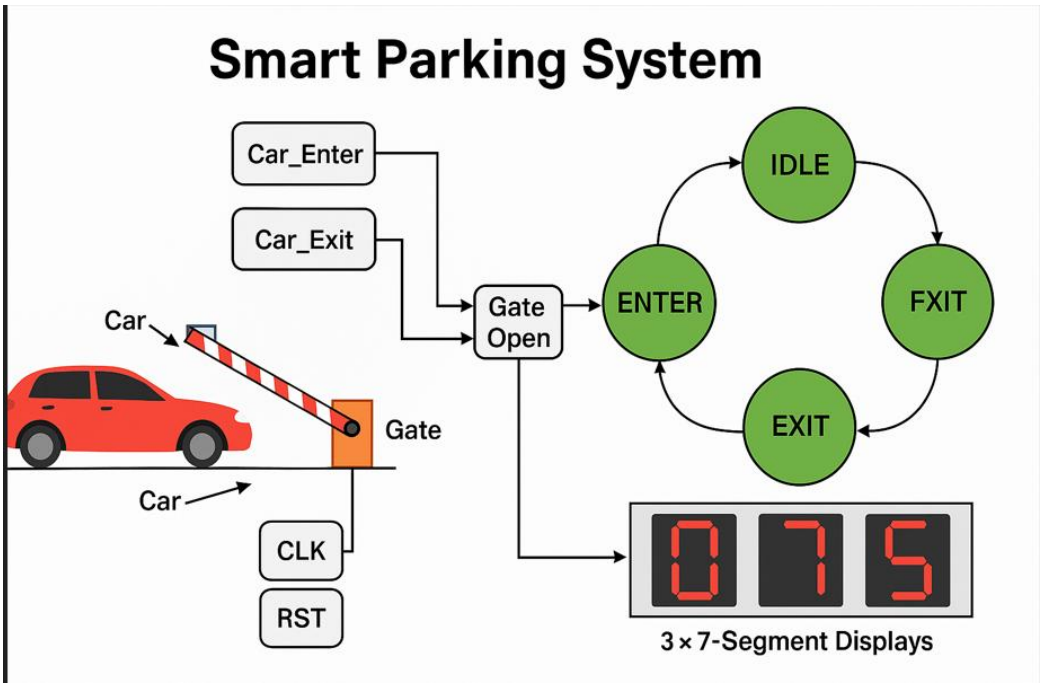
Eng///Hagar Yaser Ellaban

🔷 Smart Parking System – Project Overview

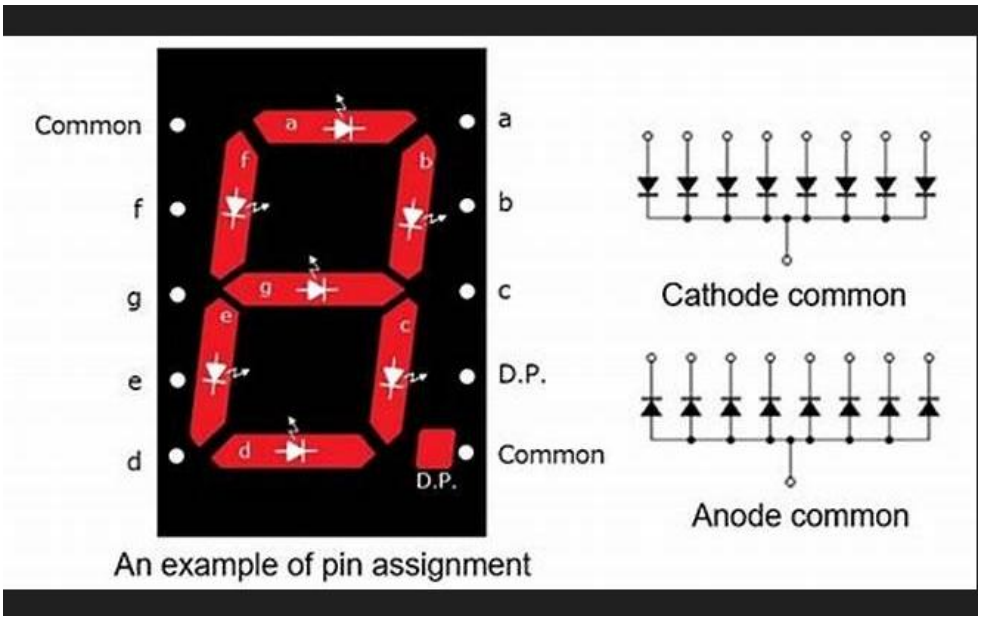
🔹 Objective:

To design a smart parking system using Verilog HDL that automatically counts the number of cars entering and exiting a parking lot and displays the number of available spaces on three 7-segment displays.



🔹 System Description:

- The system tracks **cars entering and exiting** the parking lot.
- A **counter** keeps track of the number of parked cars.
- The **available spaces** are calculated by subtracting the number of cars from the maximum capacity (100).
- The result is displayed in real-time on **three 7-segment displays** (Units, Tens, Hundreds).

- 

- When the parking lot is **full**, the **gate is automatically closed**, and no more cars are allowed to enter.
- When a car **exits**, the system updates the available spaces and opens the gate.

◆ Used Components:

Component	Description
Verilog HDL	Hardware Description Language used for design.
3 × 7-Segment Displays	To show the number of available spaces.
FSM (Finite State Machine)	Used to control system states (IDLE, ENTER, EXIT, FULL).
Up/Down Counter	Counts cars in and out.
Input Signals	Car_Enter, Car_Exit, CLK, RST.
Output Signals	Gate_Open, seg0, seg1, seg2.

◆ FSM Diagram:

States used:

- **IDLE:** Waiting for a car to enter or exit.
- **ENTER:** Car entered, increment counter.
- **EXIT:** Car exited, decrement counter.
- **FULL:** Parking is full; gate remains closed.

🔹 Key Features:

- Real-time car counting.
- Automatic gate control.
- Display of available spaces.
- Efficient logic using FSM and Verilog modules.

- **FSM State Table – Smart Parking System**

Current State	Inputs	Condition	Next State	Actions
IDLE	Car_Enter = 1, Car_Exit = 0	car_counter < MAX_SPACES	ENTER	Increment counter, open gate
IDLE	Car_Enter = 0, Car_Exit = 1	car_counter > 0	EXIT	Decrement counter, open gate
IDLE	Car_Enter = 0, Car_Exit = 0		IDLE	No change
IDLE	Car_Enter = 1, Car_Exit = 0	car_counter == MAX_SPACES	FULL	Do not increment, close gate
ENTER	Car_Enter = 1, Car_Exit = 0		IDLE	Update display
EXIT	Any		IDLE	Update display
FULL	Any		EXIT	Decrement counter
FULL	Car_Exit = 1		FULL	Gate closed

The code

First code for seven segment

```
module seven_segment_decoder #(parameter COMMON_ANODE = 1) (
input [3:0] i_data, // 4-bit binary input
output reg [6:0] o_display // 7-segment output (a, b, c, d, e, f, g)
);
always @(*) begin
case (i_data)
4'b0000: o_display = 7'b0111111; // 0
4'b0001: o_display = 7'b0000110; // 1
4'b0010: o_display = 7'b1011011; // 2
4'b0011: o_display = 7'b1001111; // 3
4'b0100: o_display = 7'b1100110; // 4
4'b0101: o_display = 7'b1101101; // 5
4'b0110: o_display = 7'b1111101; // 6
4'b0111: o_display = 7'b0000111; // 7
4'b1000: o_display = 7'b1111111; // 8
4'b1001: o_display = 7'b1101111; // 9
4'b1010: o_display = 7'b1110111; // A
4'b1011: o_display = 7'b1111100; // B
4'b1100: o_display = 7'b0111001; // C
4'b1101: o_display = 7'b1011110; // D
4'b1110: o_display = 7'b1111001; // E
4'b1111: o_display = 7'b1110001; // F
default: o_display = 7'b0000000; // Blank
endcase
// If it's common anode, invert the output
if (COMMON_ANODE)
o_display = ~o_display;
end
endmodule
```

Second

Code for smart parking system

C:/Users/Eng_Hagar_Ellaban/OneDrive/Documents/my_project_Smart_Parking.v - Default

```
Ln#
  1  module Smart_Parking_System (
  2      input  wire CLK,
  3      input  wire RST,
  4      input  wire Car_Enter,
  5      input  wire Car_Exit,
  6      output reg  Gate_Open,
  7      output  [6:0] seg0, // Units
  8      output  [6:0] seg1, // Tens
  9      output  [6:0] seg2  // Hundreds
 10  );
 11
 12  // ========== Parameters ==========
 13  parameter MAX_SPACES = 7'd100;
 14
 15  // ========== Internal Signals ==========
 16  reg [6:0] car_counter; // Up/Down Counter for number of cars
 17  reg [6:0] available_spaces;
 18  wire [6:0] ones, tens, hundreds;
 19
 20  // ========== FSM States ==========
 21  reg [1:0] state, next_state;
 22  localparam IDLE  = 2'b00,
 23             ENTER = 2'b01,
 24             EXIT  = 2'b10,
 25             FULL  = 2'b11;
 26
 27  // ========== Sequential Logic ==========
 28  always @(posedge CLK or negedge RST) begin
 29      if (!RST) begin
 30          car_counter <= 0;
 31          state <= IDLE;
 32      end else begin
 33          state <= next_state;
 34
 35          // Up/down counter logic
 36          if (Car_Enter && (car_counter < MAX_SPACES))
 37              car_counter <= car_counter + 1;
 38          else if (Car_Exit && (car_counter > 0))
 39              car_counter <= car_counter - 1;
 40      end
```

```
Ln#
41  end
42
43  // ========== Next State Logic ==========
44  always @(*) begin
45      case (state)
46          IDLE: begin
47              if (Car_Enter && (car_counter < MAX_SPACES))
48                  next_state = ENTER;
49              else if (Car_Exit && (car_counter > 0))
50                  next_state = EXIT;
51              else if (car_counter == MAX_SPACES)
52                  next_state = FULL;
53              else
54                  next_state = IDLE;
55          end
56          ENTER: next_state = IDLE;
57          EXIT:  next_state = IDLE;
58          FULL: begin
59              if (Car_Exit)
60                  next_state = EXIT;
61              else
62                  next_state = FULL;
63          end
64          default: next_state = IDLE;
65      endcase
66  end
67
68  // ========== Output Logic ==========
69  always @(*) begin
70      available_spaces = MAX_SPACES - car_counter;
71
72      // Gate control
73      if (available_spaces == 0)
74          Gate_Open = 0;
75      else
76          Gate_Open = 1;
77
78       // seg0 = dec_to_7seg(ones);
79      // seg1 = dec_to_7seg(tens);
80     // seg2 = dec_to_7seg(hundreds);
```

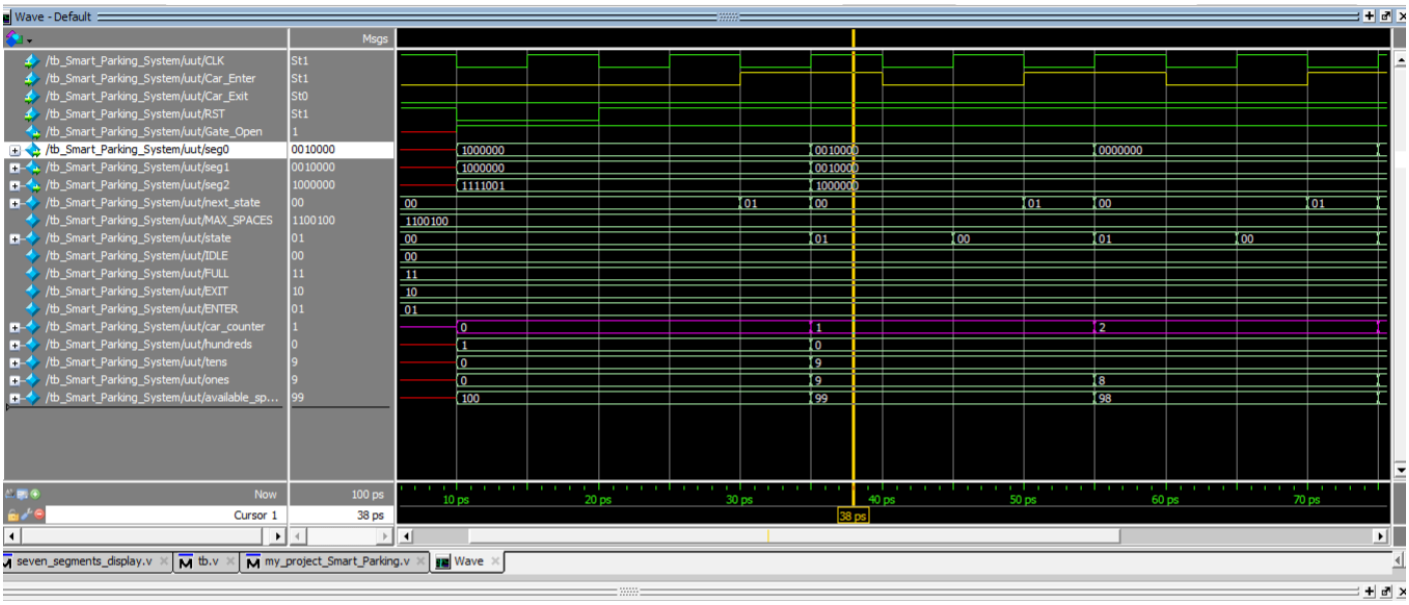
```
81  end
82
83
84      // Extract digits
85     assign hundreds = available_spaces / 100;
86     assign tens     = (available_spaces % 100) / 10;
87     assign ones     = available_spaces % 10;
88
89  seven_segment_decoder #(.COMMON_ANODE(1))  Units_inst (
90  .i_data(ones), // 4-bit binary input
91  .o_display(seg0) // 7-segment output (a, b, c, d, e, f, g)
92  );
93
94
95  seven_segment_decoder #(.COMMON_ANODE(1))  Tens_inst (
96  .i_data(tens), // 4-bit binary input
97  .o_display(seg1) // 7-segment output (a, b, c, d, e, f, g)
98  );
99
100
101 seven_segment_decoder #(.COMMON_ANODE(1))  Hundreds_inst (
102 .i_data(hundreds), // 4-bit binary input
103 .o_display(seg2) // 7-segment output (a, b, c, d, e, f, g)
104 );
105
106 endmodule
107
108
```

Third

The test bench

```
Ln#
 1  module tb_Smart_Parking_System;
 2      reg CLK, RST, Car_Enter, Car_Exit;
 3      wire Gate_Open;
 4      wire [6:0] seg0, seg1, seg2;
 5      Smart_Parking_System uut (
 6          .CLK(CLK),
 7          .RST(RST),
 8          .Car_Enter(Car_Enter),
 9          .Car_Exit(Car_Exit),
10          .Gate_Open(Gate_Open),
11          .seg0(seg0),
12          .seg1(seg1),
13          .seg2(seg2));
14      // Clock generation
15      always #5 CLK = ~CLK;
16      initial begin
17          $monitor("Time=%0t | Enter=%b Exit=%b | Gate=%b | Spaces= %b %b %b",
18                   $time, Car_Enter, Car_Exit, Gate_Open, seg2, seg1, seg0);
19          CLK = 0; RST = 1; Car_Enter = 0; Car_Exit = 0;
20          #10 RST = 0; #10 RST = 1;
21          repeat (100) begin
22              #10 Car_Enter = 1; #10 Car_Enter = 0;
23          end
24          #20;
25          repeat (3) begin
26              #10 Car_Enter = 1; #10 Car_Enter = 0;
27          end
28          repeat (10) begin
29              #10 Car_Exit = 1; #10 Car_Exit = 0;
30          end
31          #50;
32          repeat (2) begin
33              #10 Car_Enter = 1; #10 Car_Enter = 0;
34          end
35          #100 $finish;
36      end
37  endmodule
38
39
```

The wave


Wave - Default
Msgs
/tb_Smart_Parking_System/uut/CLK
/tb_Smart_Parking_System/uut/Car_Enter
/tb_Smart_Parking_System/uut/Car_Exit
/tb_Smart_Parking_System/uut/RST
/tb_Smart_Parking_System/uut/Gate_Open
/tb_Smart_Parking_System/uut/seg0
/tb_Smart_Parking_System/uut/seg1
/tb_Smart_Parking_System/uut/seg2
/tb_Smart_Parking_System/uut/next_state
/tb_Smart_Parking_System/uut/MAX_SPACES
/tb_Smart_Parking_System/uut/state
/tb_Smart_Parking_System/uut/IDLE
/tb_Smart_Parking_System/uut/FULL
/tb_Smart_Parking_System/uut/EXIT
/tb_Smart_Parking_System/uut/ENTER
/tb_Smart_Parking_System/uut/car_counter
/tb_Smart_Parking_System/uut/hundreds
/tb_Smart_Parking_System/uut/tens
/tb_Smart_Parking_System/uut/ones
/tb_Smart_Parking_System/uut/available_sp...
St1
St1
St0
St1
1
0010000
0010000
1000000
00
1100100
01
00
00
11
10
01
1
0
9
9
99
1000000
0010000
1111001
0000000
Now
100 ps
Cursor 1
38 ps
10 ps
20 ps
30 ps
40 ps
50 ps
60 ps
70 ps
seven_segments_display.v
tb.v
my_project_Smart_Parking.v
Wave