PROJECT REPORT:

Design and Implementation of a Smart Parking Management System using Verilog HDL

OBJECTIVES

- To design a parking system that can detect vehicle entry/exit.
- To allocate and free slots automatically.
- To count available parking slots and indicate when the parking is FULL.
- To implement the design in Verilog HDL and test using simulation.

MEMBERS

- *DIVYAM*- System Design: Problem statement, block diagram, flowchart.
- *NIKUNJ SINHA*- Verilog Coding: Implements the main smart parking module.
- *NIVEDITHA*- Testbench & Simulation: Writes testbench, simulates design, captures waveforms.
- *KARTHIKEYAN*-Documentation & Presentation: Prepares final report, diagrams, PPT, applications.

PROBLEM STATEMENT

In the urban setting, the increasing number of vehicles has posed a significant challenge to efficient parking space management. Traditional parking lots rely on manual monitoring, which translates to confusion, high waiting times, space mismanagement, and congestion at entry/exit points. Users do not know about available parking spaces in real-time, causing unnecessary movement of vehicles within the parking lot and high fuel consumption.

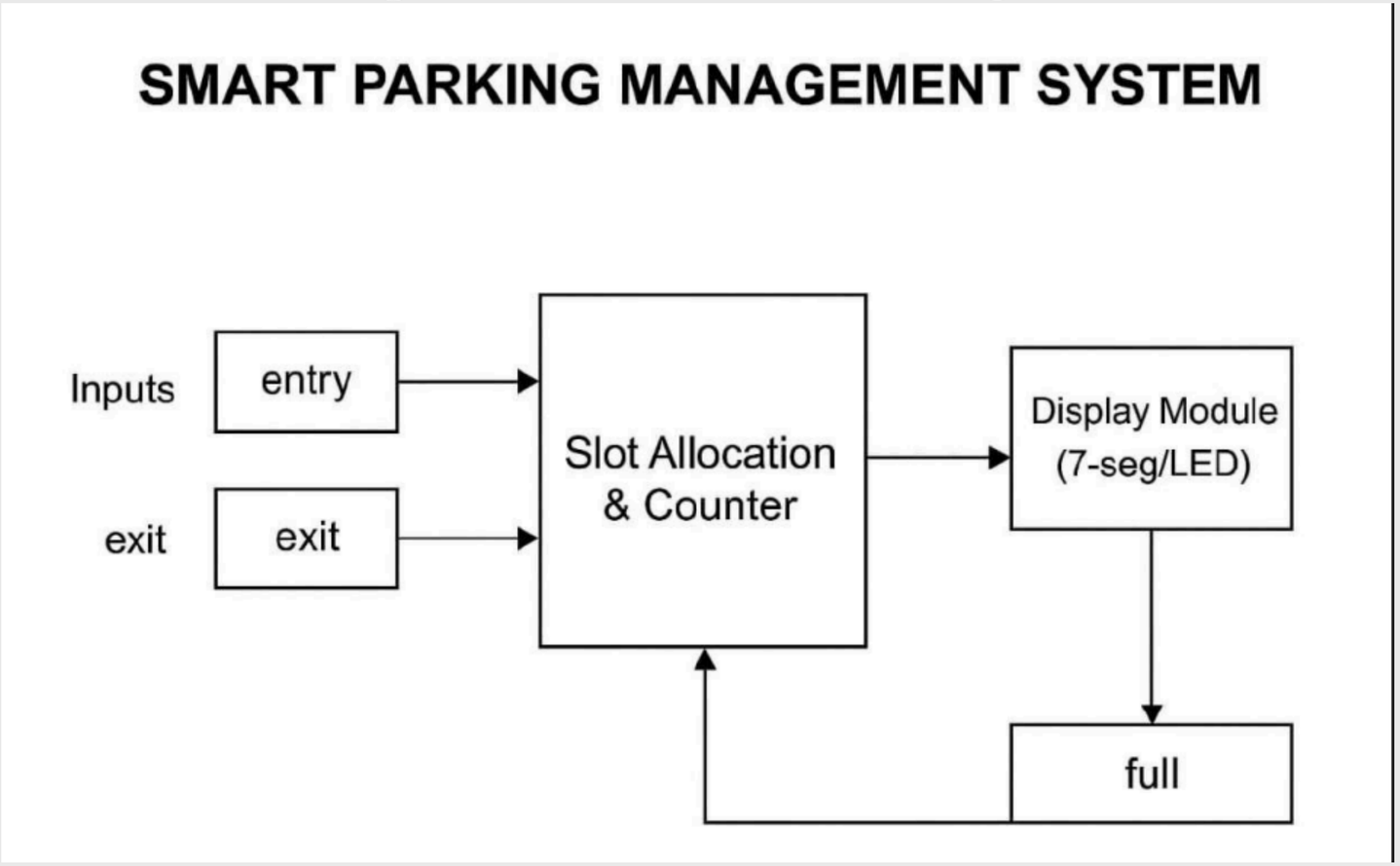
Thus, there is a requirement for a digital Smart Parking System that has the ability to automatically track parking spaces, capture vehicle entry and exit, update available space numbers, and deny entry when the

PROBLEM STATEMENT

parking facility fills up. The system must be simple, reliable, and implementable through the principles of Digital System Design and Verilog HDL. The aim is to automate parking functions, minimize human intervention, and display real-time slot availability data.

BLOCK DIAGRAM

- **Inputs**: clk, rst, entry, exit
- **Outputs**: entry_gate, exit_gate, free_count, full
- **Processing**: Slot allocation logic



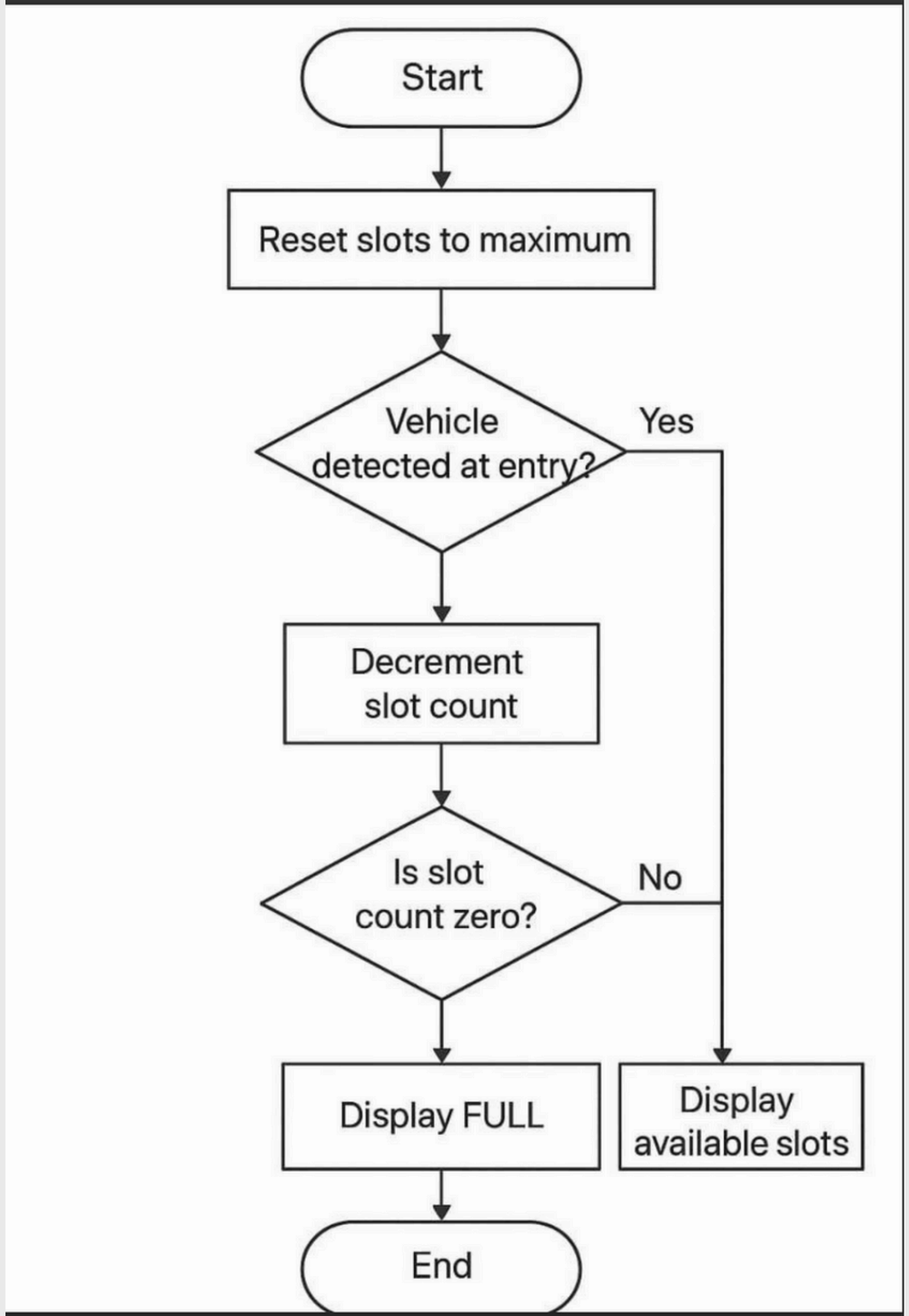
FLOWCHART

1. Reset → All slots free.

2. On **entry**: find free slot, allocate, decrement free_count, open entry gate.

3. On **exit**: find last occupied slot, free it, increment free_count, open exit gate.

4. If free_count = 0 → FULL = 1, else FULL = 0.


Start
Reset slots to maximum
Vehicle detected at entry?
Yes
Decrement slot count
Is slot count zero?
No
Display FULL
Display available slots
End

VERILOG CODE

```
module smart_parking(
    input clk, rst,
    input entry, exit,
    output reg entry_gate, exit_gate,
    output reg [3:0] free_count,
    output reg full
);

    reg [7:0] slots; // 1 = occupied, 0 = free
    integer i;

    always @(posedge clk or posedge rst) begin
        if (rst) begin
            slots <= 8'b00000000;   // all slots free
            free_count <= 8;
            full <= 0;
            entry_gate <= 0;
            exit_gate <= 0;
        end else begin
            entry_gate <= 0;
            exit_gate <= 0;
```

VERILOG CODE

```
// Car Entry
if (entry && free_count > 0) begin
    for (i = 0; i < 8; i = i + 1) begin
        if (slots[i] == 0) begin
            slots[i] <= 1;
            free_count <= free_count - 1;
            entry_gate <= 1;

             i = 8; // exit loop safely
        end
    end
end

// Car Exit
if (exit && free_count < 8) begin
    for (i = 7; i >= 0; i = i - 1) begin
        if (slots[i] == 1) begin
            slots[i] <= 0;
            free_count <= free_count + 1;
            exit_gate <= 1;
            i = -1; // exit loop safely
        end
    end
end
```

VERILOG CODE

```
            end

            // FULL check
            full <= (free_count == 0) ? 1 : 0;
        end
    end
endmodule

module tb_smart_parking;

    reg clk, rst, entry, exit;
    wire entry_gate, exit_gate, full;
    wire [3:0] free_count;

    // Instantiate the Smart Parking module
    smart_parking uut (
        .clk(clk),
        .rst(rst),
        .entry(entry),
        .exit(exit),
        .entry_gate(entry_gate),
        .exit_gate(exit_gate),
        .free_count(free_count),
        .full(full)
```

VERILOG CODE

```
);

// Clock generation (10ns period)
always #5 clk = ~clk;

initial begin
    clk = 0;
    rst = 1;
    entry = 0;
    exit = 0;

    #10 rst = 0;  // Release reset

    // Simulate car entries
    #10 entry = 1; #10 entry = 0;
    #10 entry = 1; #10 entry = 0;
    #10 entry = 1; #10 entry = 0;
    #10 entry = 1; #10 entry = 0;
    #10 entry = 1; #10 entry = 0;
    #10 entry = 1; #10 entry = 0;
    #10 entry = 1; #10 entry = 0;
    #10 entry = 1; #10 entry = 0;
    #10 entry = 1; #10 entry = 0; // Now full

    #20; // Wait to observe full signal
```

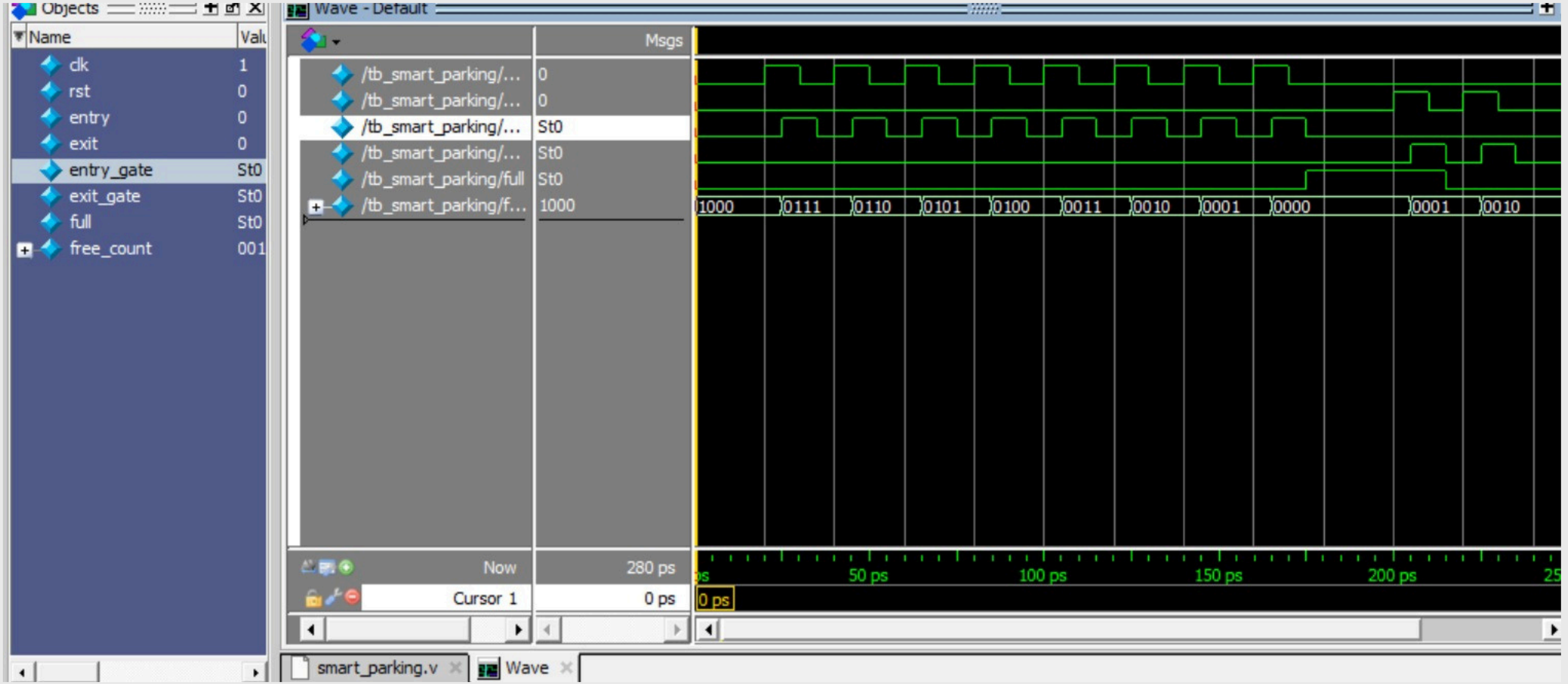
VERILOG CODE

```
        // Simulate car exits
        #10 exit = 1; #10 exit = 0;
        #10 exit = 1; #10 exit = 0;

        #50 $stop; // End simulation
    end
endmodule
```

SIMULATION RESULTS

- **Initially**: free_count = 8, all slots empty.
- Each **entry** → free_count decreases by 1, entry_gate = 1.
- When all slots filled → full = 1.
- Each **exit** → free_count increases, exit_gate = 1, slot freed.
- Waveforms confirm correct functionality.


Objects
Name
Valu
clk
1
rst
0
entry
0
exit
0
entry_gate
St0
exit_gate
St0
full
St0
free_count
001
Wave - Default
Msgs
/tb_smart_parking/...
0
/tb_smart_parking/...
0
/tb_smart_parking/...
St0
/tb_smart_parking/...
St0
/tb_smart_parking/full
St0
/tb_smart_parking/f...
1000
1000
0111
0110
0101
0100
0011
0010
0001
0000
0001
0010
Now
280 ps
Cursor 1
0 ps
50 ps
100 ps
150 ps
200 ps
0 ps
smart_parking.v
Wave

WORKING PRINCIPLE/THEORY

The Smart Parking System is based on digital counters, registers, and control logic implemented using Verilog HDL. The system continuously monitors two main events: car entry and car exit, using digital sensor inputs. The core logic uses:

1. 8-bit Register (slots)

Tracks the status of each parking slot

1 → Slot occupied

0 → Slot empty

WORKING PRINCIPLE/THEORY

2. 4-bit Counter (free_count)

Stores the total number of available parking slots

Updated on every entry and exit event

3. Control Logic

If entry = 1 and free_count > 0 → allocate nearest empty slot and decrement counter

If exit = 1 and free_count < 8 → free the last occupied slot and increment counter

WORKING PRINCIPLE/THEORY

4. Full Indicator (full)

Activated when free_count = 0

Prevents new entries until a car exits

5. Gate Control Signals
entry_gate goes HIGH for one clock cycle during a valid car entry
exit_gate goes HIGH for a valid exit

This system works entirely on clock-based synchronous logic using always @(posedge clk) blocks, ensuring reliable and glitch-free output. The design is simulation-friendly and FPGA synthesizable.

REAL LIFE PROBLEM

- In most urban and semi-urban areas, parking management is still done manually. Vehicle owners drive around inside parking lots searching for empty slots, which leads to:
- Traffic congestion at entry points
- Wasted time and fuel
- Human error in slot counting
- Mismanagement during peak hours
- Overcrowding and confusion inside the parking area

- As the number of vehicles continues to grow, these issues increase parking delays and driver frustration, while also contributing to environmental pollution due to vehicle idling.

PROPOSED SOLUTION

The Smart Parking Management System solves these problems by automating the process of slot monitoring and gate control using digital logic. The system:

- Automatically detects entry and exit of vehicles
- Keeps track of available slots in real-time using counters
- Stops further entry when the parking lot is full
- Reduces the need for manual supervision
- Improves traffic flow and avoids unnecessary vehicle movement
- Provides faster and more organized parking experience

- By using simple, reliable Verilog-based digital logic, the system ensures quick decision-making and error-free parking management, making it suitable for smart city infrastructure.

APPLICATIONS

- The Smart Parking System can be used in:
- Shopping malls & multiplexes
- Airports and railway stations
- Smart city parking zones
- Corporate office and IT parks
- Hospitals and universities
- Residential gated communities

APPLICATIONS

- Underground and multi-level parking structures
- It can also be extended with sensors, IoT communication, and payment systems for real-world smart automation.

ADVANTAGES

- Automates parking management.
- Prevents overfilling.
- Provides real-time slot status.
- Simple, low-cost digital implementation.

FUTURE SCOPE

This project can be extended into a full smart-automation system. Possible future enhancements include:

1. IoT Integration
   - Connect the parking system to the internet
   - Users can check available slots on a mobile app before arriving
2. Automated Billing System
   - Ticket generation at entry
   - Automatic fare deduction at exit based on parking duration
3. ANPR (Automatic Number Plate Recognition)
   - Camera-based vehicle identification
   - Enhanced security and logging of vehicles

FUTURE SCOPE

4. Sensor-Based Slot Detection
   - Use IR/Ultrasonic sensors at each slot
   - Accurately detect vehicle presence instead of loop logic
5. Multi-Level Parking Support
   - Extend counter and slot logic for multi-floor parking complexes
6. Voice / Display Guidance System
   - Guide drivers to the nearest free slot using LEDs or speaker announcements
7. Emergency Override and Priority Parking
   - Reserved slots for VIP, ambulance, or disabled parking

CONCLUSION

The Smart Parking Management System was successfully modeled and simulated with Verilog HDL. The system effectively tracks availability of parking slots, automates the car entry-exit process, and guards against overfilling by tripping a full indicator. Simulation waveforms confirm that counter counts down on each car entry and up on exit, with full tripping at zero availability.

This project illustrates how digital logic, counters, and control circuits are efficiently employed to address actual-world automation issues. The design is intuitive, expandable, and can be applied on FPGA hardware or developed into an IoT-based system for contemporary Smart City implementation. The outcomes confirm that the system is robust, affordable, and able to enhance parking efficiency with minimal manual intervention.

THANK YOU