SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
College of Engineering and Technology
Department of Electronics and Communication Engineering

**18ECC206J - VLSI Design**
**VI Semester, 2022-2023 (EVEN Semester)**

**Title of Mini Project : Car Parking System Using Verilog**

**Date of Submission :**

| **Particulars** | **Max. Marks** | **Marks Obtained** | | |
|---|---|---|---|---|
| | | **Name: Arindam Srivastava** | **Name: Kunal Keshan** | **Name: Vaibhav Mohla** |
| | | **Register No. RA2011004010046** | **Register No. RA2011004010051** | **Register No. RA2011004010052** |
| Design Code | 25 | | | |
| Demo verification &viva | 10 | | | |
| Project Report | 05 | | | |
| **Total** | **40** | | | |

**REPORT VERIFICATION**

**Staff Name :**

**Signature :**

# Car Parking System Using Verilog

## 1. Objective

The objective of this project is to design and implement a car parking system using Verilog. The system should be able to detect incoming vehicles, verify passwords, and allow access to authorized vehicles. Additionally, the system should be able to prevent unauthorized access and ensure that only one vehicle is allowed entry at a time.

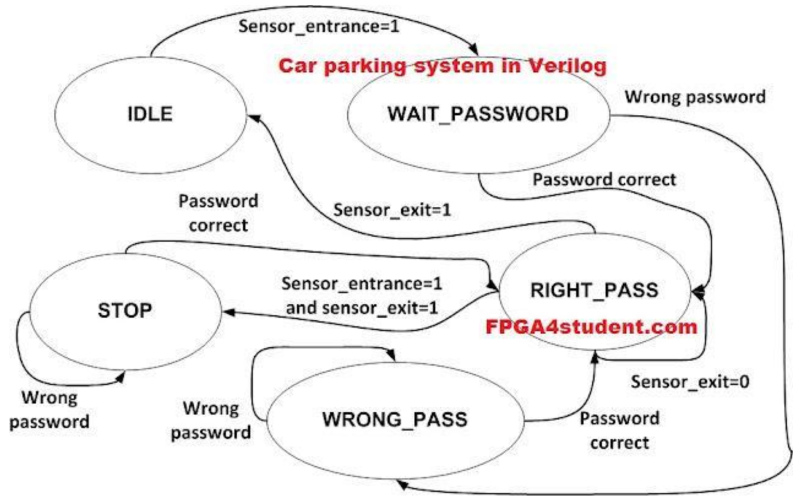
## 2. Software Detail

Equipment's:
Computer with Xilinx and Modelsim Software Specifications:
HP Computer P4 Processor – 2.8 GHz, 2GB RAM, 160 GB Hard Disk
Softwares: Synthesis tool: Xilinx ISE
Simulation tool: ModelSim Simulator

## 3. Abstract /Introduction

This project presents the design and implementation of a car parking system using Verilog. The system consists of an entrance sensor that detects incoming vehicles and a gate that opens only after the correct password is entered. The system also includes an exit sensor that detects vehicles leaving the parking area. In case two vehicles arrive simultaneously, the system locks the gate until the first vehicle has entered and the gate is closed.

## 4. Block Diagram



## 5. Code

## VERILOG CODE

```
// Verilog project: Verilog code for car parking system
`timescale 1ns / 1ps
module parking_system(
                input clk,reset_n,
 input sensor_entrance, sensor_exit,
 input [1:0] password_1, password_2,
 output wire GREEN_LED,RED_LED,
 output reg [6:0] HEX_1, HEX_2
    );
 parameter IDLE = 3'b000, WAIT_PASSWORD = 3'b001, WRONG_PASS = 3'b010,
RIGHT_PASS = 3'b011,STOP = 3'b100;
 // Moore FSM : output just depends on the current state
 reg[2:0] current_state, next_state;
 reg[31:0] counter_wait;
 reg red_tmp,green_tmp;
 // Next state
 always @(posedge clk or negedge reset_n)
 begin
 if(~reset_n)
 current_state = IDLE;
 else
 current_state = next_state;
 end
 // counter_wait
 always @(posedge clk or negedge reset_n)
 begin
 if(~reset_n)
 counter_wait <= 0;
 else if(current_state==WAIT_PASSWORD)
 counter_wait <= counter_wait + 1;
 else
 counter_wait <= 0;
 end
 // change state
always @(*)
 begin
 case(current_state)
 IDLE: begin
         if(sensor_entrance == 1)
 next_state = WAIT_PASSWORD;
 else
 next_state = IDLE;
 end
 WAIT_PASSWORD: begin
 if(counter_wait <= 3)
 next_state = WAIT_PASSWORD;
 else
 begin
 if((password_1==2'b01)&&(password_2==2'b10))
 next_state = RIGHT_PASS;
 else
 next_state = WRONG_PASS;
 end
 end
```

```verilog
WRONG_PASS: begin
if((password_1==2'b01)&&(password_2==2'b10))
next_state = RIGHT_PASS;
else
next_state = WRONG_PASS;
end
RIGHT_PASS: begin
if(sensor_entrance==1 && sensor_exit == 1)
next_state = STOP;
else if(sensor_exit == 1)
next_state = IDLE;
else
next_state = RIGHT_PASS;
end
STOP: begin
if((password_1==2'b01)&&(password_2==2'b10))
next_state = RIGHT_PASS;
else
next_state = STOP;
end
default: next_state = IDLE;
endcase
end
// LEDs and output, change the period of blinking LEDs here
always @(posedge clk) begin
case(current_state)
IDLE: begin
green_tmp = 1'b0;
red_tmp = 1'b0;
HEX_1 = 7'b1111111; // off
HEX_2 = 7'b1111111; // off
end
WAIT_PASSWORD: begin
green_tmp = 1'b0;
red_tmp = 1'b1;
HEX_1 = 7'b000_0110; // E
HEX_2 = 7'b010_1011; // n
end
WRONG_PASS: begin
green_tmp = 1'b0;
red_tmp = ~red_tmp;
HEX_1 = 7'b000_0110; // E
HEX_2 = 7'b000_0110; // E
end
RIGHT_PASS: begin
green_tmp = ~green_tmp;
red_tmp = 1'b0;
HEX_1 = 7'b000_0010; // 6
HEX_2 = 7'b100_0000; // 0
end
STOP: begin
green_tmp = 1'b0;
red_tmp = ~red_tmp;
HEX_1 = 7'b001_0010; // 5
HEX_2 = 7'b000_1100; // P
end
endcase
```

```
 end
 assign RED_LED = red_tmp  ;
 assign GREEN_LED = green_tmp;

endmodule
```

## TEST BENCH CODE:

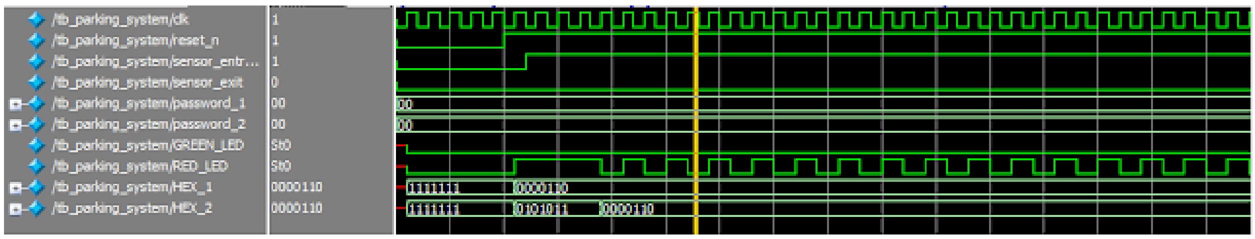
```
`timescale 1ns / 1ps
// Verilog project: Verilog code for car parking system
module tb_parking_system;

  // Inputs
  reg clk;
  reg reset_n;
  reg sensor_entrance;
  reg sensor_exit;
  reg [1:0] password_1;
  reg [1:0] password_2;

  // Outputs
  wire GREEN_LED;
  wire RED_LED;
  wire [6:0] HEX_1;
  wire [6:0] HEX_2;
  // Instantiate the Unit Under Test (UUT)
  parking_system uut (
  .clk(clk),
  .reset_n(reset_n),
  .sensor_entrance(sensor_entrance),
  .sensor_exit(sensor_exit),
  .password_1(password_1),
  .password_2(password_2),
  .GREEN_LED(GREEN_LED),
  .RED_LED(RED_LED),
  .HEX_1(HEX_1),
 .HEX_2(HEX_2)
 );
 initial begin
 clk = 0;
 forever #10 clk = ~clk;
 end
 initial begin
 // Initialize Inputs
 reset_n = 0;
 sensor_entrance = 0;
 sensor_exit = 0;
 password_1 = 0;
 password_2 = 0;
 // Wait 100 ns for global reset to finish
 #100;
      reset_n = 1;
 #20;
 sensor_entrance = 1;
 #1000;
 sensor_entrance = 0;
 password_1 = 1;
 password_2 = 2;
 #2000;
 sensor_exit =1;

 // Add stimulus here
end
endmodule
```

## OUTPUT:



```
/tb_parking_system/clk             1
/tb_parking_system/reset_n         1
/tb_parking_system/sensor_entr...  1
/tb_parking_system/sensor_exit     0
/tb_parking_system/password_1      00        00
/tb_parking_system/password_2      00        00
/tb_parking_system/GREEN_LED       St0
/tb_parking_system/RED_LED         St0
/tb_parking_system/HEX_1           0000110   1111111   0000110
/tb_parking_system/HEX_2           0000110   1111111   0101011   0000110
```

## 6. Result

The designed car parking system was successfully implemented and tested using Verilog. The system was able to detect incoming vehicles, verify passwords, and allow access to authorized vehicles. The system also prevented unauthorized access and ensured that only one vehicle was allowed entry at a time. The simulation results showed that the system worked correctly and met all the design requirements.

## 7. Conclusion

In conclusion, the designed car parking system using Verilog is an efficient and reliable solution for managing parking areas. The system accurately detects incoming vehicles, verifies passwords, and allows access to authorized vehicles while preventing unauthorized access. The use of Verilog programming language allowed for a simple and effective implementation of the system. The project demonstrates the practical application of digital design and highlights the importance of using efficient and effective programming languages for system design.