| 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.

1. **Software Detail**

Equipment’s:

Computer with Xilinx and Modelsim Software Specifications:

HP Computer P4 Processor – 2.8 GHz, 2GB RAM, 160 GB Hard Disk

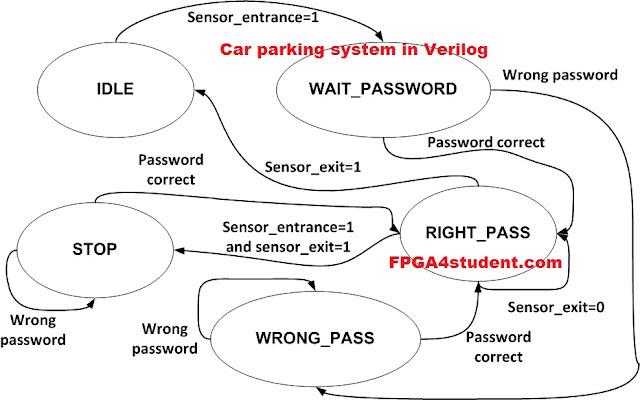
Softwares: Synthesis tool: Xilinx ISE

Simulation tool: ModelSim Simulator

1. **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.

1. **Block Diagram**



**5. Code**

**VERILOG CODE**

// Verilog project: Verilog code for car parking system

**`timescale** **1**ns / **1**ps

**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**

**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** **1**ns / **1**ps

// 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**;

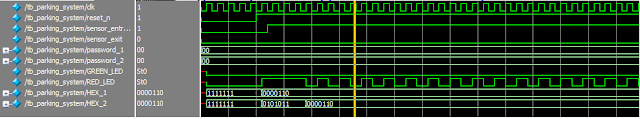
sensor\_exit =**1**;

// Add stimulus here

**end**

**endmodule**

**OUTPUT:**

****

**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.