Smart Car Parking System

November 24, 2021

1.Jyoti Dhiman - 2020csb1092
2.Sanyam Walia - 2020csb1122
3.Rohit Kinha - 2020csb1118

Instructor:
Dr. Neeraj Goel

Summary:

Motor vehicle use is increasing every day, resulting in increased noise, traffic congestion, and parking space concerns, with finding a vacant parking place becoming increasingly difficult.
In this project, we are proposing a smart car parking system using Verilog code.
This system of parking cars will handle all aspects of vehicle management, including parking and security.
In addition through this system, we can calculate the fare of a car for the time it was parked.

---

1. Introduction

When it comes to a parking system, the problems mostly we face are:

A. Congestion at the entrance and exit gate which may even sometimes lead to a collision between cars.
B. The other problem we face is the security of parked cars.
C. Lastly, another problem that we encounter is how to charge a particular car, whether it should be fixed for every car irrespective of the time it is parked or it should be the other way around.

So, solving all these problems, we propose a smart car parking system where these three problems are solved very conveniently.

Here is a small description of our project and how it is working.
Firstly, whenever a car is detected by the entrance sensor, we turn on the red light, asking the person to stop and enter a password to get into the parking slot. If he enters the wrong password, ask him/her to enter the password. Until he/she doesn't enter the right password he is stopped at the gate this way we can ensure the security of the other cars parked and eliminate the unwanted cars to block the parking slots.
After he enters the right password, we will park the coming car and increase the number of cars parked by 1.
At the same time, we will start the fare counter for that car.
Now suppose a parked car wants to go out, the exit sensor will be triggered and the red light will be turned on and a password is asked from the user. If he enters the right password,
And no other car is coming in at the same time we will allow the car to go out and charge it for its fare, else we will stop the coming car for some time till the exit sensor gets off means the car has exited from the gate.

---

2. About Moore State Machine

Moore machine is a finite-state machine whose output values depend only upon the present state and not on the input values. The value of the output function is a function of the current state and the changes at the clock edges, whenever state changes occur.

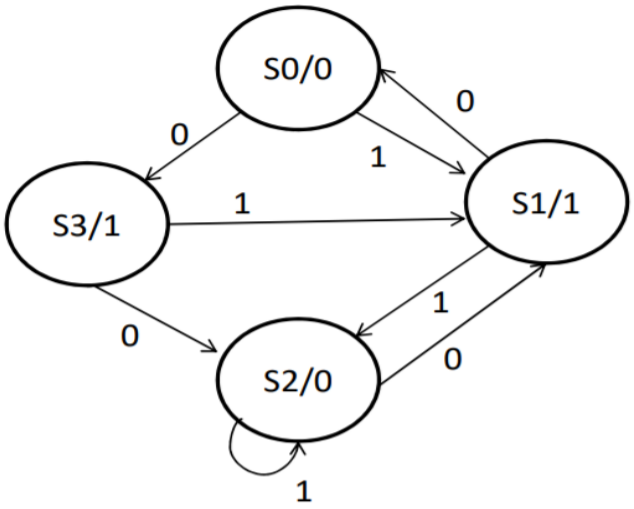
A Moore machine can be described by a 6 tuple (Q, ∑, O, δ, X, q_0) where −

- Q is a finite set of states.
- ∑ is a finite set of symbols called the input alphabet.
- O is a finite set of symbols called the output alphabet.
- δ is the input transition function where δ: Q × ∑ → Q.
- X is the output transition function where X: Q → O.
- q_0 is the initial state from where any input is processed (q_0 ∈ Q).

The state table of a Moore Machine is shown below −

Present State	Next State		Output
	Input = 0	Input = 1	
a	b	c	X_2
b	b	d	X_1
c	c	d	X_2
d	d	d	X_3

The state diagram of the above Moore Machine is −



Moore State machine

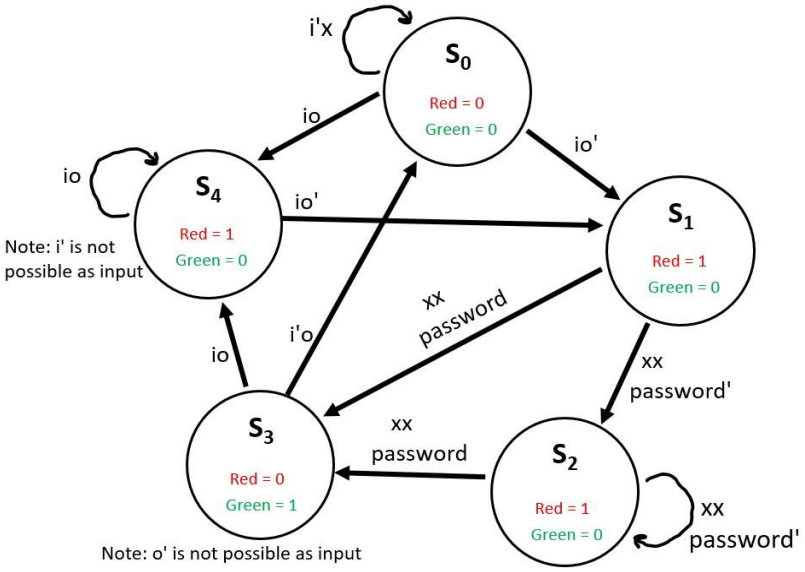
3. Project Details

3.1. Concept

This project presents a car parking system in VHDL using a Finite State Machine (FSM). VHDL code and testbench for the car parking system are fully provided. The car parking system is shown in the following figure. There is a front sensor to detect vehicles going to the gate of the car parking system. Another back sensor is to detect if the coming vehicle passed the gate and got into the car park.

Initially, the FSM is in IDLE state. If there is a vehicle detected by the front sensor, FSM is switched to WAIT_PASSWORD state for 4 cycles. The car will input the password in this state; if the password is correct, the gate is opened to let the car get in the car park and FSM turns to RIGHT_PASS state; a Green LED will be blinking. Otherwise, FSM turns to WRONG_PASS state; a Red LED will be blinking and it requires the car to enter the password again until the password is correct. When the current car gets into the car park detected by the back sensor and there is the next car coming, the FSM is switched to STOP state and the Red LED will be blinking so that the next car will be noticed to stop and enter the password. After the car passes the gate and gets into the car park, the FSM returns to IDLE state.

3.2. State Diagram



3.3.State Table

Present State	Next State								Present Output	
	i=0 o=0 p=0	i=0 o=0 p=1	i=0 o=1 p=0	i=0 o=1 p=1	i=1 o=0 p=0	i=1 o=0 p=1	i=1 o=1 p=0	i=1 o=1 p=1	Red	Green
S0	S0	S0	S0	S0	S1	S1	S4	S4	0	0
S1	S2	S3	S2	S3	S2	S3	S2	S3	1	0
S2	S2	S3	S2	S3	S2	S3	S2	S3	1	0
S3	Z	Z	S0	S0	Z	Z	S4	S4	0	1
S4	Z	Z	Z	Z	S1	S1	S4	S4	1	0

4. Verilog Implementation

4.1 Design.v

Our main module is parking_system. In this module, we take some inputs and define our output variables, given below:

```
module parking_system(
 input clk, reset_n,
 input sensor_entrance, sensor_exit,
 input[1:0] password,
 output wire GREEN_LED, RED_LED,
 output reg[2:0] state,
 output reg[3:0] number,
 output reg[9:0] fare
 );
```

After assigning some parameters, we move into our main function.
Here we first begin with the IDLE state.
After checking through the reset condition we move into our state interconnection stage where we implement the concept shown in the below figure to connect and move through different stages.

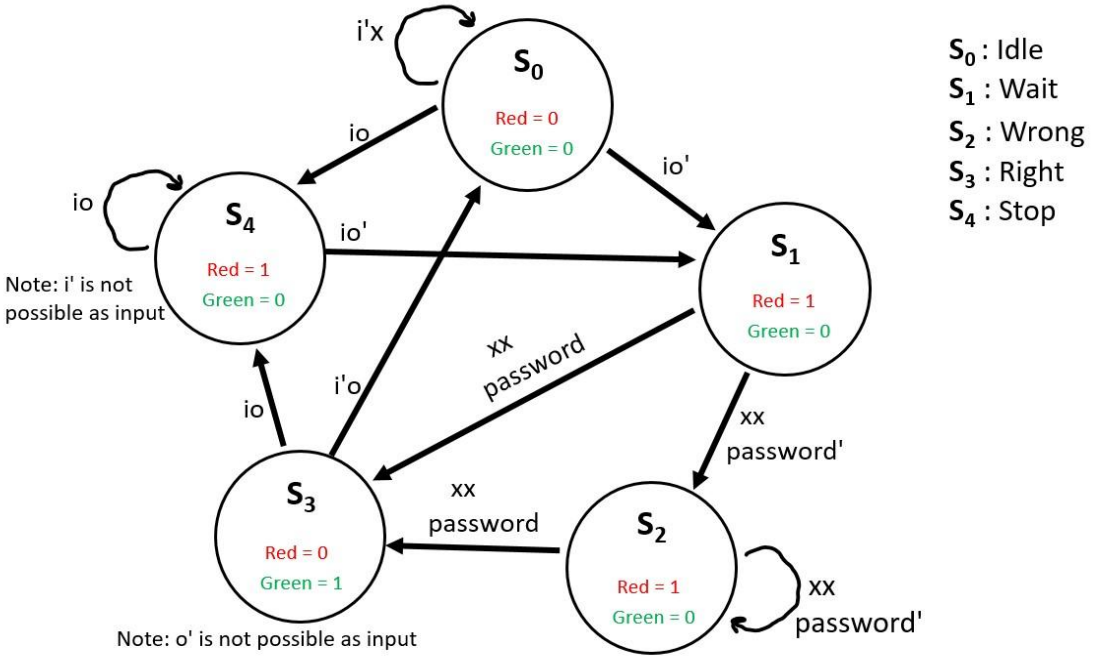


Figure 1 - Diagram showing the interconnection of different states

The Verilog code showing the same interconnection of these states is:

```
//State Interconnections...
always @(*)
  begin
   case(current_state)
    IDLE:
      begin
       if(sensor_entrance==1 && sensor_exit==1)
    next_state = STOP;
     else if(sensor_entrance==1)
    next_state = WAIT_PASSWORD;
     else
      next_state = IDLE;
    end
    WAIT_PASSWORD:
      begin

        if((password==2'b11))
      next_state = RIGHT_PASS;
     else
      next_state = WRONG_PASS;
    end
```

```
WRONG_PASS:
    begin
     if(password==2'b11)
  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(sensor_entrance==1 && sensor_exit==1)
   next_state = STOP;
  else
   next_state = WAIT_PASSWORD;
  end
  default: next_state = IDLE;
 endcase
end
```

At last in our Verilog code, we gave the description of different states we used in our project:

```
//Description of States
always @(posedge clk)
  begin
    case(current_state)
  IDLE:
    begin
   green_tmp = 1'b0;
   red_tmp = 1'b0;
   state = 3'b000;
      fare = fare + (number*(4'b1010));
    end
    WAIT_PASSWORD:
    begin
   green_tmp = 1'b0;
   red_tmp = 1'b1;
   state = 3'b001;
      fare = fare + (number*(4'b1010));
    end
  WRONG_PASS:
    begin
   green_tmp = 1'b0;
   red_tmp = ~red_tmp;
   state = 3'b010;
      fare = fare + (number*(4'b1010));
  end
```

```
  RIGHT_PASS:
    begin
   green_tmp = 1'b1;
   red_tmp = 1'b0;
   state = 3'b011;
     number = number+4'b0001;
      fare = fare + (number*(4'b1010));
  end
    STOP:
    begin
   green_tmp = 1'b0;
   red_tmp = 1'b1;
   state = 3'b100;
     fare = fare + (number*(4'b1010));
    end
  endcase
end
assign RED_LED = red_tmp;
assign GREEN_LED = green_tmp;
```

This way our main and only module ends.
To check our design we will check a testcase which we will consider in next section.

4.2 Testcase.v and Results

Here, we took some possible conditions that we may encounter in our parking system.

A. In this case, the inputs we take are:
   i. Reset = 0
   ii. Entrance sensor = 0
   iii. Exit sensor = 0
   iv. Password = 00

The results we get are:

```
INPUTS
Sensor-entrance: 0, Sensor-exit: 0    //Idle state
Password: 00  (Correct password=11)
OUTPUTS
Red Light: 0, Green Light: 0
Number: 0000(Binary),  0 (Decimal) | Fare:    0(Decimal)
State: IDLE
```

B. In this case, the inputs we take are:
    i. Reset = 0
    ii. Entrance sensor = 1
    iii. Exit sensor = 0
    iv. Password = 00

    The results we get are:

```
INPUTS
Sensor-entrance: 1, Sensor-exit: 0    //Car incoming. Red Light and ask for Password
Password: 00  (Correct password=11)
OUTPUTS
Red Light: 1, Green Light: 0
Number: 0000(Binary),  0 (Decimal) | Fare:    0(Decimal)
State: WAIT PASSWORD
```

C. In this case, the inputs we take are:
    i. Reset = 0
    ii. Entrance sensor = 1
    iii. Exit sensor = 0
    iv. Password = 00

    The results we get are:

```
INPUTS
Sensor-entrance: 1, Sensor-exit: 0    //Wrong password. Ask again
Password: 00  (Correct password=11)
OUTPUTS
Red Light: 0, Green Light: 0
Number: 0000(Binary),  0 (Decimal) | Fare:    0(Decimal)
State: WRONG PASSWORD
```

D. In this case, the inputs we take are:
    i. Reset = 0
    ii. Entrance sensor = 1
    iii. Exit sensor = 0

iv. Password = 11

The results we get are:

```
INPUTS
Sensor-entrance: 1, Sensor-exit: 0    //Correct password. Green light and Number of cars increased.
Password: 11  (Correct password=11)
OUTPUTS
Red Light: 0, Green Light: 1
Number: 0001(Binary),  1 (Decimal) | Fare:   10(Decimal)
State: RIGHT PASSWORD
```

E. In this case, the inputs we take are:
   i. Reset = 0
   ii. Entrance sensor = 0
   iii. Exit sensor = 1
   iv. Password = 11

The results we get are:

```
INPUTS
Sensor-entrance: 0, Sensor-exit: 1
Password: 11  (Correct password=11)
OUTPUTS
Red Light: 0, Green Light: 0
Number: 0001(Binary),  1 (Decimal) | Fare:   20(Decimal)
State: IDLE
```

F. In this case, the inputs we take are:
   i. Reset = 0
   ii. Entrance sensor = 1
   iii. Exit sensor = 0
   iv. Password = 00

The results we get are:

```
INPUTS
Sensor-entrance: 1, Sensor-exit: 0    //New Car incoming. Red light and ask for password.
Password: 00  (Correct password=11)
OUTPUTS
Red Light: 1, Green Light: 0
Number: 0001(Binary),  1 (Decimal) | Fare:   30(Decimal)
State: WAIT PASSWORD
```

G. In this case, the inputs we take are:
   i. Reset = 0
   ii. Entrance sensor = 1
   iii. Exit sensor = 0
   iv. Password = 11

The results we get are:

```
INPUTS
Sensor-entrance: 1, Sensor-exit: 0    //Correct Password. Green Light and Parked
Password: 11  (Correct password=11)
OUTPUTS
Red Light: 0, Green Light: 1
Number: 0010(Binary),  2 (Decimal) | Fare:   50(Decimal)
State: RIGHT PASSWORD
```

H. In this case, the inputs we take are:
   i. Reset = 0
   ii. Entrance sensor = 1
   iii. Exit sensor = 1
   iv. Password = 00

The results we get are:

```
INPUTS
Sensor-entrance: 1, Sensor-exit: 1    //New car comes simultaneously. Stop it to avoid collision.
Password: 00  (Correct password=11)
OUTPUTS
Red Light: 1, Green Light: 0
Number: 0010(Binary),  2 (Decimal) | Fare:   70(Decimal)
State: STOP
```

I. In this case, the inputs we take are:
   i. Reset = 0
   ii. Entrance sensor = 1
   iii. Exit sensor = 1
   iv. Password = 00

The results we get are:

```
INPUTS
Sensor-entrance: 1, Sensor-exit: 1    //Still both sensors are 1 So New car still Stopped.
Password: 00  (Correct password=11)
OUTPUTS
Red Light: 1, Green Light: 0
Number: 0010(Binary),  2 (Decimal) | Fare:   90(Decimal)
State: STOP
```

J. In this case, the inputs we take are:
   i. Reset = 0
   ii. Entrance sensor = 1
   iii. Exit sensor = 0
   iv. Password = 00

The results we get are:

```
INPUTS
Sensor-entrance: 1, Sensor-exit: 0    //Finally, exit sensor=0 so ask new car for password.
Password: 00  (Correct password=11)
OUTPUTS
Red Light: 1, Green Light: 0
Number: 0010(Binary),  2 (Decimal) | Fare:  110(Decimal)
State: WAIT PASSWORD
```

K. In this case, the inputs we take are:
   i. Reset = 0
   ii. Entrance sensor = 1
   iii. Exit sensor = 0
   iv. Password = 11

The results we get are:

```
INPUTS
Sensor-entrance: 1, Sensor-exit: 0    //Correct password. Green light. Parked.
Password: 11  (Correct password=11)
OUTPUTS
Red Light: 0, Green Light: 1
Number: 0011(Binary),  3(Decimal) | Fare:  140(Decimal)
State: RIGHT PASSWORD
```

L. In this case, the inputs we take are:
   i. Reset = 1
   ii. Entrance sensor = 1
   iii. Exit sensor = 0
   iv. Password = 00

The results we get are:

```
RESETTING

INPUTS
Sensor-entrance: 0, Sensor-exit: 0
Password: 00  (Correct password=11)
OUTPUTS
Red Light: 0, Green Light: 0
Number: 0000(Binary),  0(Decimal) | Fare:    0(Decimal)
State: IDLE

Reset successfull. Do visit again :)
```

After going through these situations, we found that the results are the same that we get through our understanding of the project.
So our design successfully passed the testcase.

---

5. Conclusion

The goal of this project was to develop an effective smart car parking system where we could avoid various problems like collisions due to congestion at the gate, security issues, and inappropriate fare charges.
The system built can be used for many applications, and can easily increase the number of slot choices and increase parking protection. Through using the above-implemented program parking becomes simple. The car is correctly identified and parked safely.
This type of system is very useful as it avoids wastage of time of people in unproductive work. Also, it eliminates the tedious work of finding vacant parking at crowded places such as Malls, Huge Organizations by loitering here and there and wasting our time doing so.

Acknowledgments

We would like to express special thanks to our professor **Dr. Neeraj Goel** who gave us this opportunity to work on this project **Smart Car Parking System** using Verilog.
Through this project, we learned a lot of new things and also helped us in doing some good research about this topic and implementing the same.
Secondly, I would like to thank my other team members for completing this project before the deadline.
Jyoti Dhiman - 2020csb1092
Sanyam Walia - 2020csb1122
Rohit Kinha - 2020csb1118

References

[1] Charles H.Roth, Jr. Larry L. Kinney *Fundamentals of Logic Design*, USA, 2010

Appendix A

Charles H.Roth, Jr. Larry L. Kinney *Fundamentals of Logic Design*, USA, 2010

Thank You!