Implementation Of Smart Car Parking System In Verilog

Why Car Parking System?

Due to urbanization one of the most significant problems faced is car parking, which is mainly due to

- Parking shortage.
- Lack of proper security.

What is the car parking system?

The central idea of the project came from the troubles that the employees face in daily routine. Our approach uses smart car parking system, which works according to Moore FSM. With this technology, we can ensure:

- Safety of parked cars
- No more congested parking space
- Proper management without much investment

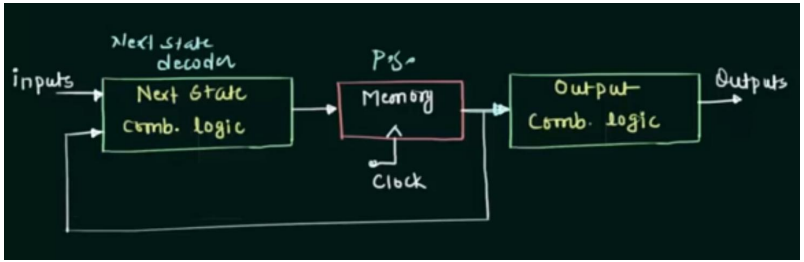
What is Moore FSM?

Moore machines are finite state machines with output value and its output depends only on present state. It can be defined as (Q, q0, Σ, O, δ, λ) where:

- • Q is finite set of states.
- • q0 is the initial state.
- • Σ is the input alphabet.
- • O is the output alphabet.
- • δ is transition function which maps Q×Σ → Q.
- • λ is the output function which maps Q×Σ→ O.

What is Moore FSM?

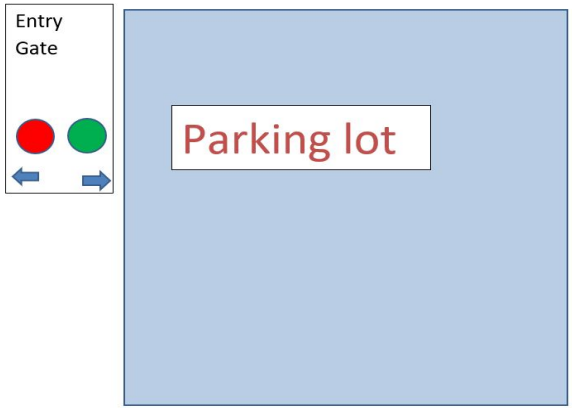
The Moore state machine's block diagram is shown below. The Moore state machine block diagram consists of two parts namely combinational logic as well as memory.



Overview of our Project

This project uses a Finite State Machine to implement an automobile parking system in Verilog (FSM).

- **There is a front-facing sensor (Entrance sensor)which Detect vehicles approaching the auto parking system's gate.**
- **There is another sensor which Detects If any car want to exit the parking lot.**
- **There are Two LED's RED & GREEN**
  - ➢ **Red means STOP**
  - ➢ **Green means GO**



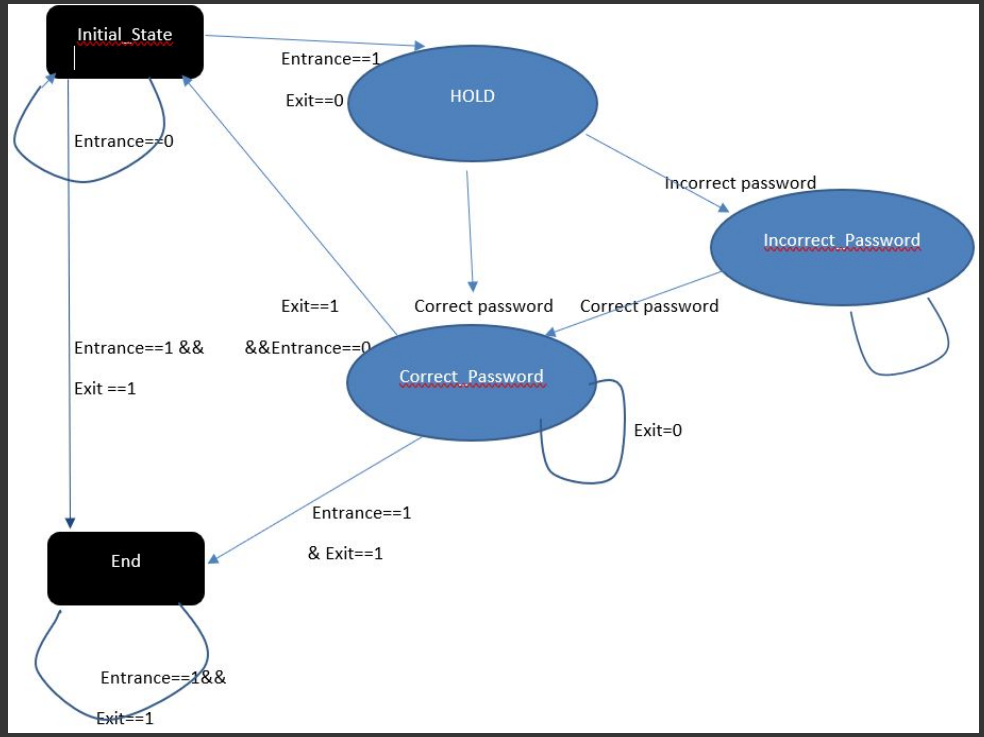
Overview

The FSM is initially in an Initial_State.

- **If Entrance Sensor are 1 it means there is a vehicle coming , FSM is switched to HOLD state.**
- **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 Correct_password state; a Green LED will be blinking. Otherwise, FSM turns to Incorrect_password state; a Red LED will be blinking and it requires the car to enter the password again until the password is correct.**

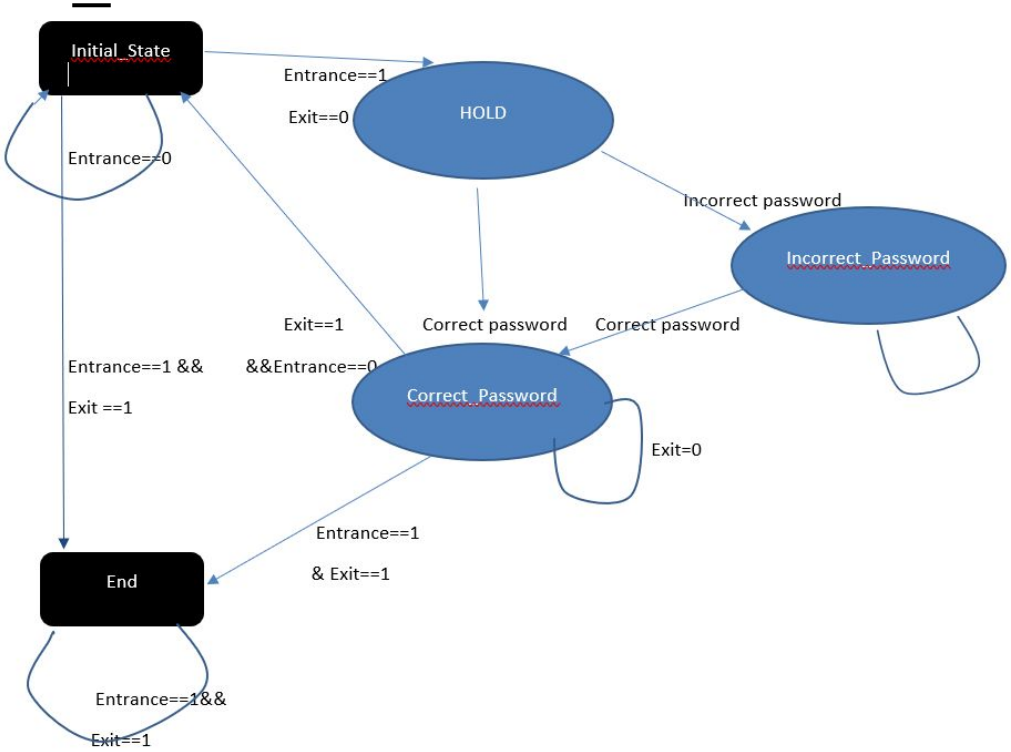
- **If Entrance Sensor are 1 and Exit sensor is also 1 it means a car is going to exit the parking lot also there is the next car coming, the FSM is switched to End state to STOP the incoming car and the Red LED will be blinking so that the next car will be noticed to stop and it first let the exiting car to exit the parking lot . After the car passes the gate and gets into the car park, the FSM returns to HOLD state.**

States Explanation



Note

Moore machine is a finite state machine whose output values are determined only by its current state.



1.Initial State

Initial is the IDLE state, where Entrance sensor detects. If the sensor is off, the system will not perform any actions. When the entrance sensor detects the car has come, state switched to HOLD. If both entrance and exit sensor are true then stops the car and let the exiting car to exit.

2.Hold

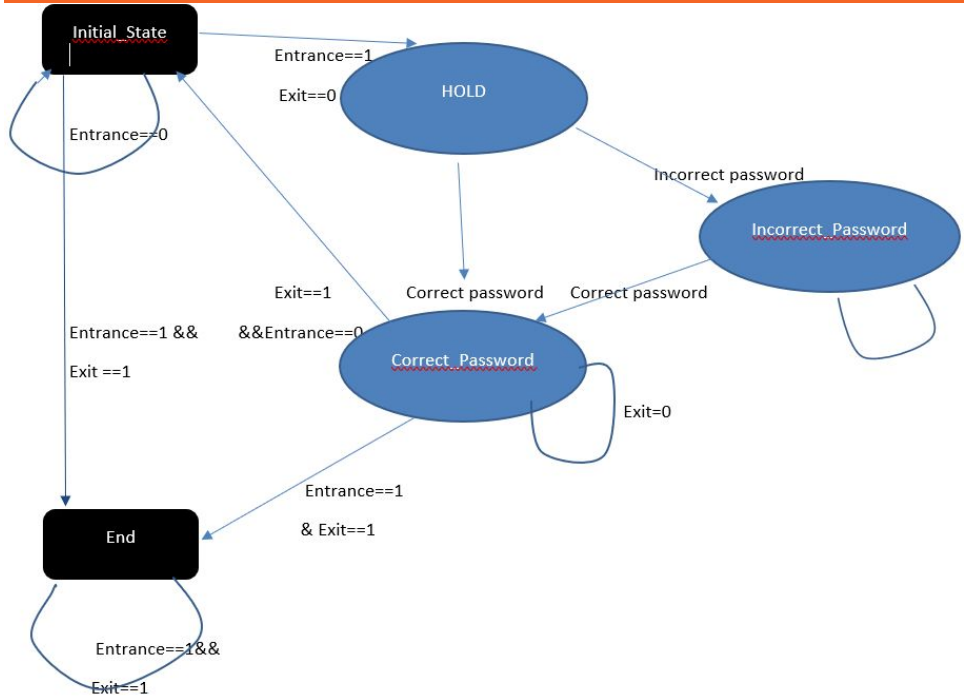
In HOLD state, Red light will turned ON and display will show ‘Enter Password’. Car user need to type the password to get entry through the parking system. If the password is Incorrect it will go to the Incorrect_Password state and if the password is correct it will go to the Correct_Password state

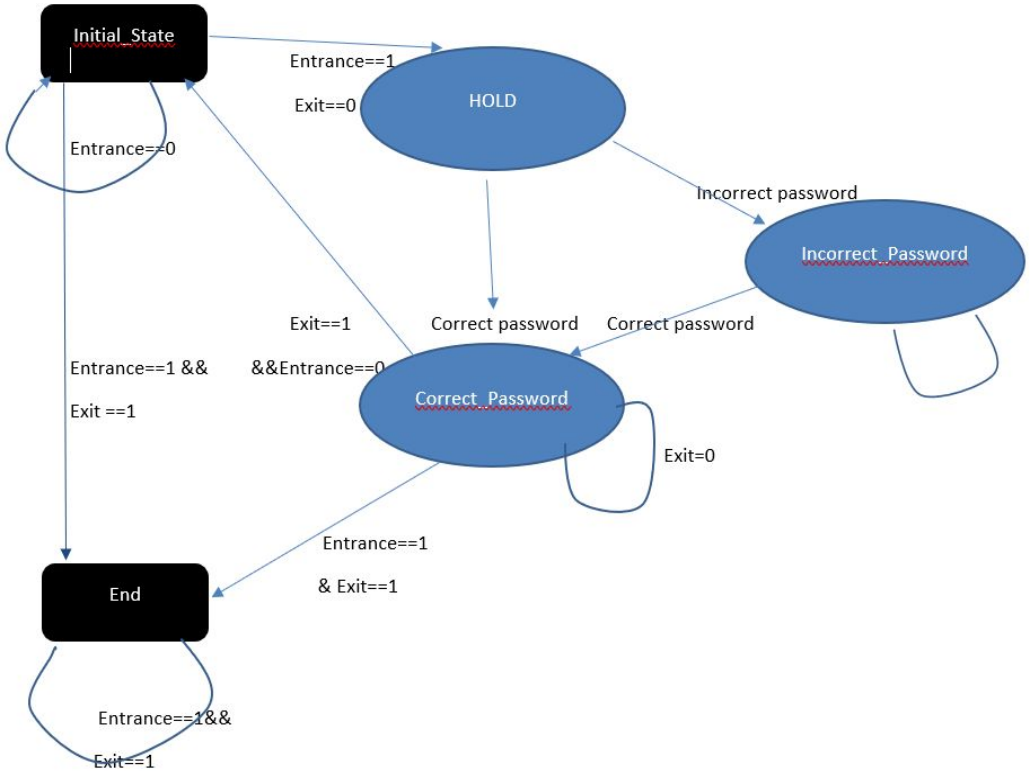
3.Correct_password

If the password entered correctly in hold state, Green light will show in Correct_Password state and it will display ‘GO’ message indicating that you can enter in the parking system. After crossing the car through the parking system, It will go to the Initial state if there is no other car coming. If another car is coming it will turned on both sensors Entrance==1 && Exit == 1 and so it will go the End state

4.Incorrect_password

If the password is Incorrect, Incorrect password state will show Red light and will display that password is incorrect. It will give driver another chance to enter password and will stay in the same state. If driver enter Correct password after the first failed trial, it will go to the Correct password state, where driver can pass through this system.





5.End

In this state Entrance sensor is 1 and exit sensor is also 1. This means a parked car wants to exit parking lot and there is incoming car, to avoid collision it stops incoming car and let the exiting car to exit, then after exit becomes 0 State will change to HOLD

Code Explanation

Modules inputs and outputs

```
`timescale 1ns / 1ps
//Verilog Code for Car parking System
module CAR_Parking_System(
    input RESET,           // 1 bit input for resetting the car parking system
    input CLOCK,           // Clock as continuous input of 1 bit
    input Entrance,        // 1 bit Entrance value asking for entering the system
    input Exit,            // 1 bit Exit value asking for Exiting the system
    input [3:0] PASSWORD, //  Password of 4 bit length to increase security
    output wire Green,
    output wire Red,

    output reg [2:0] INDICATOR // Output variable used to refer states of FSM

);
```

Assigning Parameter

After assigning Initialing CS as Initial state

```
    //Five States
 parameter Correct_password = 3'b011;
 parameter Incorrect_password = 3'b010;
 parameter Initial_State = 3'b000;
 parameter HOLD = 3'b001;
 parameter End = 3'b100;

 // And the code is based on Moore FSM as output only depends on the current state
 reg[2:0] CS;//CS as Current State
 reg[2:0] NS;//NS as Next State


reg RED_tmp,GREEN_tmp;

initial
 begin
    CS=Initial_State;
end
```

```
always @(posedge CLOCK or posedge RESET)
begin
if(RESET)
// When we set reset as 1 It will set current state as initial state
begin
CS = Initial_State;
end
else
CS = NS;
//When Clock changes its value from 0 to 1,It will set current state as Next state
end
```

After this we define interconnections between all states.

Initial_State

- Setting Next state as End if entrance and exit both sensors are 1.
- If Entrance is 1 and Exit is 0, Means there is car. Setting next state on HOLD
- Else State does not Change

```
always @(*)
begin
case(CS)
Initial_State:
 begin
if(Entrance == 1&& Exit== 1 )
NS = End ;
else if(Entrance==1&& Exit== 0 )
NS= HOLD;
else
NS = Initial_State;
end
```

HOLD

- Setting Next state as Correct_password if entered password is correct.
- Setting Next state as Incorrect_password if entered password is Incorrect.

```
HOLD: begin
if(PASSWORD==4'b1011)

NS = Correct_password;

else

NS = Incorrect_password;

end
```

Incorrect_password

- Setting Next state as Correct_password if entered password is correct.
- Setting Next state as Incorrect_password if entered password is Incorrect.

```
Incorrect_password: begin
if(PASSWORD==4'b1011)

NS = Correct_password;

else
NS = Incorrect_password;
end
```

Correct_password

- If Entrance and exit both are 1 then NS=End.
- If Entrance is 0 and exit is 1 then NS=Initial_state.
- Else Setting NS as its CS.

```
Correct_password: begin
if(Entrance==1 && Exit == 1)

NS = End;

else if(Exit ==1)
NS = Initial_State;

else
NS = Correct_password;

end
```

End

- If Entrance and exit both are 1 then NS=End.

- Else Setting NS as its Hold as there It will ask password to the incoming car.

```
End: begin
if(Entrance==1 && Exit == 1)

NS = End;

else
NS=HOLD;

end
default: NS = Initial_State;
endcase
end
```

Description of States

```
always @(posedge CLOCK) begin
  case(CS)
    Initial_State:  begin
      GREEN_tmp = 1'b0;
      RED_tmp = 1'b0;
      INDICATOR = 3'b000;
    end
    HOLD: begin
      GREEN_tmp = 1'b0;
      RED_tmp = 1'b1;
      INDICATOR = 3'b001;
    end
    Incorrect_password:  begin
      GREEN_tmp = 1'b0;
      RED_tmp = 1'b1;
      INDICATOR = 3'b010;
    end
```

```
Correct_password: begin
GREEN_tmp = 1'b1;
RED_tmp = 1'b0;
INDICATOR = 3'b011;
end
End: begin
GREEN_tmp = 1'b0;
RED_tmp = 1'b1;
INDICATOR = 3'b100;
end
endcase
end
assign Red = RED_tmp  ;
assign Green = GREEN_tmp;

endmodule
```

Testbench and its results

```
[Running] testbench1.v
Welcome to our Smart Car Parking System

VCD info: dumpfile CAR_Parking_System.vcd opened for output.
Inputs are as follows:
 Entrance:0          Exit:0
Outputs are as follows:
      Red Led:0      Green Led:0

Current state is: Initial_State
```

```
Inputs are as follows:
 Entrance:1          Exit:0
Outputs are as follows:
      Red Led:1      Green Led:0

Current state is: HOLD
```

```
Inputs are as follows:
          Password  :1000
Entered Password is Incorrect

Outputs are as follows:
      Red Led:1      Green Led:0

Current state is: Incorrect_password
```

```
Inputs are as follows:
          Password  :1100
Entered Password is Incorrect

Outputs are as follows:
      Red Led:1      Green Led:0

Current state is: Incorrect_password
```

```
Inputs are as follows:
          Password :1011
Outputs are as follows:
      Red Led:0      Green Led:1

Entered Password is Correct

Current state is: RIGHT PASSWORD
```

```
Inputs are as follows:
 Entrance:0          Exit:1
Outputs are as follows:
      Red Led:0      Green Led:0

Current state is: Initial_State



Inputs are as follows:
 Entrance:1          Exit:1
Outputs are as follows:
      Red Led:1      Green Led:0

Current state is: End
```

```
Inputs are as follows:
 Entrance:1          Exit:0
Outputs are as follows:
      Red Led:1      Green Led:0

Current state is: HOLD



Inputs are as follows:
            Password :1011
Outputs are as follows:
      Red Led:0      Green Led:1

Entered Password is Correct

Current state is: RIGHT PASSWORD
```

```
Inputs are as follows:
 Entrance:1          Exit:1
Outputs are as follows:
      Red Led:1      Green Led:0

Current state is: End



Inputs are as follows:
 Entrance:1          Exit:1
Outputs are as follows:
      Red Led:1      Green Led:0

Current state is: End



RESETING!!
Current state is: Initial_State
```

Further Scope

- We can also add a count variable to calculate the total number of cars , checked in our parking system.
- Also from the count variable, we can calculate the total income gained from the parking system.

Thank You...

Tanuj Kumar 2020csb1134

Jatin 2020csb1090