Implementation of Parking System with FSM

Advanced Digital System Design

11/14/2018

Department of Computer Engineering, University of Denver
Submitted By: Siddhesh Suresh Padwal
DU ID (873305527)

Submitted To: Dr. George Edwards

Table of Contents

Abstract

The industrialization of the world, increase in population, slow paced city development and mismanagement of the available parking space has resulted in parking related problems. Due to the proliferation in the number of vehicles on the road, traffic problems are bound to exist. There is a dire need for a secure, intelligent, efficient and reliable system which can be used for guidance towards the parking facility, negotiation of the parking fee, along with the proper management of the parking facility. This is due to the fact that the current transportation infrastructure and car park facility developed are unable to cope with the influx of vehicles on the road. To alleviate the aforementioned problems, the smart parking system has been developed.

This paper reviews Parking System using DE2 Altera used for parking guidance, parking facility management. The discussed system will be able to reduce the problems which are arising due to unavailability of a reliable, efficient and modern parking system. Subsequently, the various sensor systems used in developing the systems in addition to the recent research and commercial system on the market are examined as vehicle detection plays a crucial role in the smart parking system.

Introduction

The industrial growth of the world is reflected by the increase in the number of automobiles on the streets throughout the world, which has caused a lot of parking related problems. The slow paced city planning has increased the problem even more. The search for the parking space is a time consuming process which not only affects the economic activities' efficiency, but also the social interactions and cost. Network companies cannot provide updated information of the parking facilities on the internet as the parking facilities do not cooperate with the companies. Certain big cars are not able to fit into the normally available parking spaces. Hence there is a need for a system; which can take all relevant information into consideration, for finding the parking vacancy.

Human errors are the major source of traffic accidents, therefore building in-car technologies for checking the parking lot, avoiding accidents and guidance to the parking facility is turning out to be an integral area for research. The objective of such technologies is the reduction of the burden on driver, improvement of the traffic capacity, and provision of reliable and secure car functions.

The parking meters which rely on coins or tokens is an inefficient system as it requires man power for management of the parking and exact change for paying the parking charges. Parking control and enforcement systems provide efficient and effective monitoring of meter and it also keeps a check on any violations of the parking lot. This results in best possible use of the parking space for increasing the revenue.

The Parking service, a part of Intelligent Transportation System, gives rise to different parking facilities on the basis of new functions they provide. This service not only manages the internal operations of the parking facility, but it is also designed to work with different aspects related to the parking facility.

The services which the Parking System should provide in the future are

- The parking availability information system and parking reservation system should provide advanced navigation services.
- The mobile electric commerce system and a continuously working gate system should collect the toll charges electrically.

- Provision of strong functions for facilitating administrators and managers in management of the parking facility.

The information related to the availability of unoccupied lot; before the driver enters the facility is provided on the display. An empty parking lot can be reserved by the driver through entering the password. The continuous entry and exit system facilitates a driver by getting rid of time consuming processes such as getting a ticket, and the freedom of selecting any payment method.

Further modifications result in even better systems, such as reserving the parking space online and using a smart card with it will help the driver find the destination quickly, safely and easily. Despite the system requiring no man power, it will still be able to know about the entrance and exit of the vehicles as well as the occupancy rate of the facility. These systems will also decrease the traffic congestion as the number of vehicles parked on the street will decrease. These new systems will boost the parking business by the increase in the number of customers.

The latest advancement in intelligent parking service is the parking space negotiation system which is much different than the parking information system. Parking space negotiation system uses the linking and integration of the parking facilities which results in negotiation and coordinates between the in-vehicle information system and parking facility. This system initializes the negotiation process for the parking charges, the advance reservation of the parking lot, search for the best possible path from the current position to the parking facility and then to the destination. Coordination work is an important task for the negotiation corporation. Negotiation is just like a business where both sellers and buyers decide the terms of business, for getting the best possible deal for both parties.

Working

In the entrance of the parking system, there is a sensor which is activated to detect a vehicle coming. Once the sensor is triggered, a password is requested to open the gate. If the entered password is correct, the gate would open to let the vehicle get in. Otherwise, the gate is still locked. If the current car is getting in the car park being detected by the exit sensor and another car comes, the door will be locked and requires the coming car to enter passwords.

Implementation with Finite State Machine

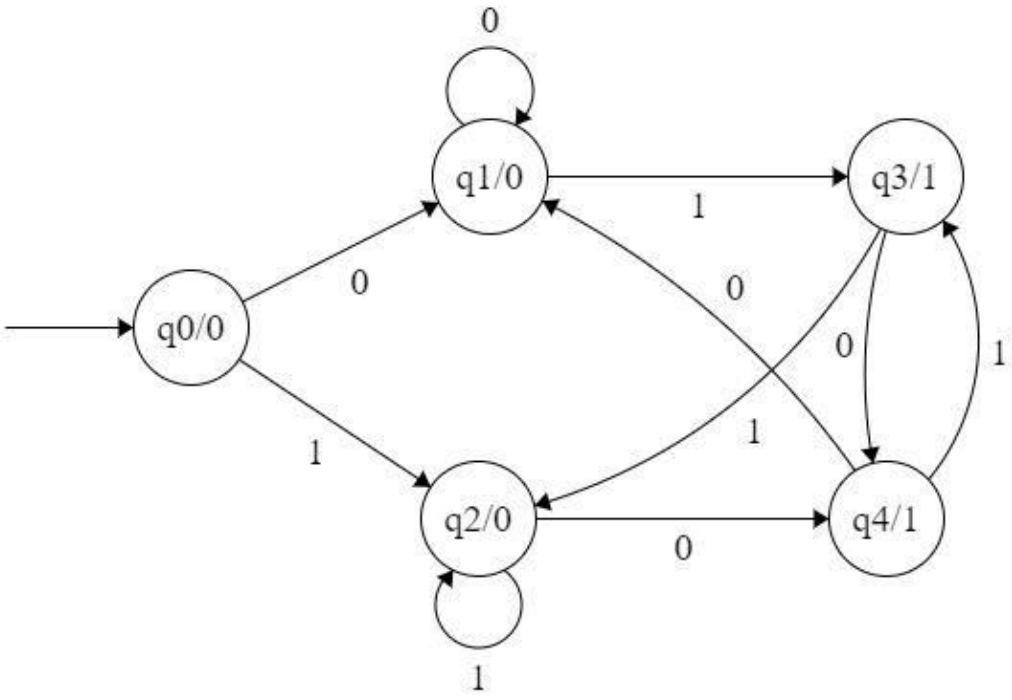
A finite-state machine (FSM) or finite-state automaton, finite automaton, or simply a state machine, is a mathematical model of computation. It is an abstract machine that can be in exactly one of a finite number of states at any given time. The FSM can change from one state to another in response to some external inputs; the change from one state to another is called a transition. An FSM is defined by a list of its states, its initial state, and the conditions for each transition. Finite state machines are of two types - deterministic finite state machines and non-deterministic finite state machines. A deterministic finite-state machine can be constructed equivalent to any non-deterministic one.

Moore Machines:

Moore machines are finite state machines with output value and its output depends only on present state. It can be defined as $(Q, q0, \sum, O, \delta, \lambda)$ where:

- Q is finite set of states.
- q0 is the initial state.
- $\sum$ is the input alphabet.
- O is the output alphabet.

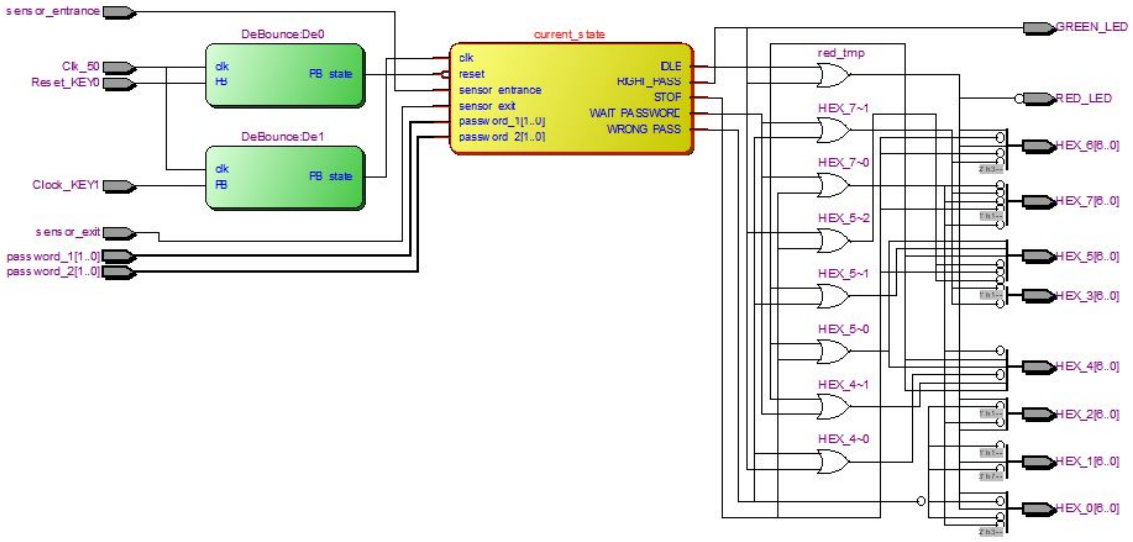
- δ is transition function which maps Q×∑ → Q.
- λ is the output function which maps Q → O.



Example diagram for Moore machine

Circuit Diagram for Parking System using DE2 Altera

Detailed Diagram



Assignments on DE2 Altera:

To implement the parking system we need to have two input sensors, LED's and seven segment display.

Sensors:

1. sensor_entrance (SW[0]): To detect the car is entered in the parking lot.
2. Sensor_exit (SW[1]): To detect the car is passed through the parking system.

LED's:

1. Red LED (LEDR[0]): To display the door is closed and vehicle has to stop.
2. Green LED (LEDG[0]): To display that car can pass through the parking system.

Seven Segment Display (HEX7, HEX6, HEX5, HEX4, HEX3, HEX2, HEX1, HEX0):

To give instructions to the car what needs to be done can be display on screen. (eg. Wait, Enter password, Wrong Password, Go)

PASSWORD (SW[5:2]):

The given password to the driven can be verified at parking system. If it matched with the correct password the door will open or else it will be remain closed.

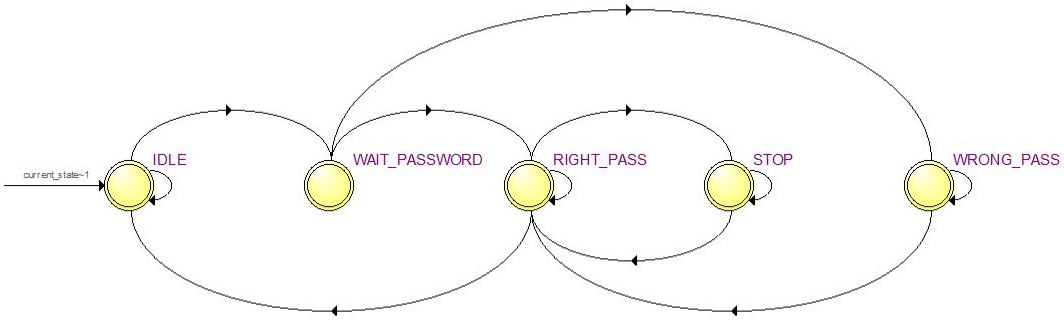
Password can be entered using SW[5:2].

Clock (KEY[1]):

Clock can be handled manually using Key0 to change action of circuit and demonstrate each and each step.

Reset (KEY0): To reset the whole parking system.

FSM diagram for Parking System



Moore machine using Altera

Conditions to alter the states:

States can be altered on changing the following inputs and dependencies shown in the image below:

1. Sensor_entrance
2. Sensor_exit
3. Password_1 & Password_2

	Source State	Destination State	Condition
1	IDLE	WAIT_PASSWORD	(sensor_entrance)
2	IDLE	IDLE	(!sensor_entrance)
3	RIGHT_PASS	STOP	(sensor_entrance).(sensor_exit)
4	RIGHT_PASS	RIGHT_PASS	(!sensor_exit)
5	RIGHT_PASS	IDLE	(sensor_exit).(!sensor_entrance)
6	STOP	STOP	(!password_1[0]) + (password_1[0]).(!password_1[1]).(!password_2[0]).(!password_2[1]) + (password_1[0]).(!password_1[1]).(password_2[0]) + (password_1[0]).(password_1[1])
7	STOP	RIGHT_PASS	(password_1[0]).(!password_1[1]).(!password_2[0]).(password_2[1])
8	WAIT_PASS...	WRONG_PASS	(!password_1[0]) + (password_1[0]).(!password_1[1]).(!password_2[0]).(!password_2[1]) + (password_1[0]).(!password_1[1]).(password_2[0]) + (password_1[0]).(password_1[1])
9	WAIT_PASS...	RIGHT_PASS	(password_1[0]).(!password_1[1]).(!password_2[0]).(password_2[1])
10	WRONG_PASS	WRONG_PASS	(!password_1[0]) + (password_1[0]).(!password_1[1]).(!password_2[0]).(!password_2[1]) + (password_1[0]).(!password_1[1]).(password_2[0]) + (password_1[0]).(password_1[1])
11	WRONG_PASS	RIGHT_PASS	(password_1[0]).(!password_1[1]).(!password_2[0]).(password_2[1])

State explanation with the code and output:

1. IDLE STATE :

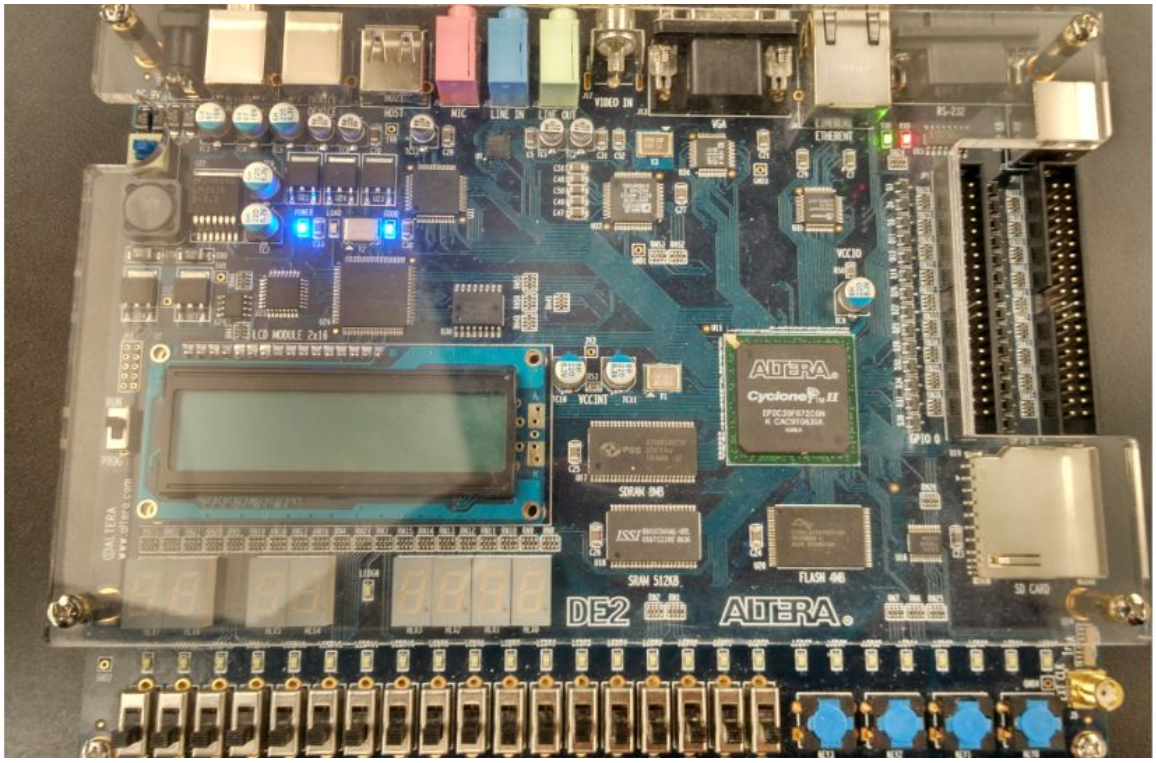


Image1: IDLE state(everything is turned off)

Code:

```
IDLE: begin
if(sensor_entrance == 1)
next_state = WAIT_PASSWORD;
else
next_state = IDLE;
end
```

Explanation:

Idle is the default 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 **WAIT_PASSWORD.**

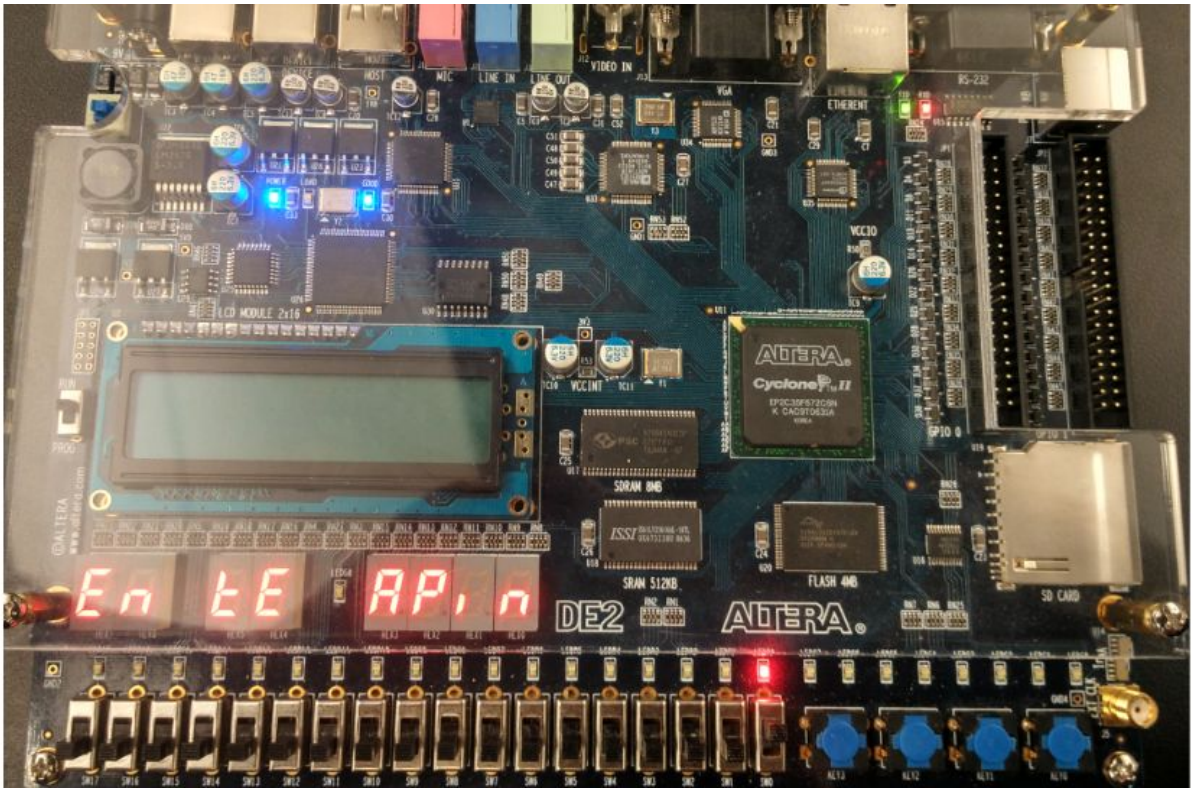


Image2: (entrance_sensor==1) go into wait state

2. WAIT STATE

Code:

```
WAIT_PASSWORD: begin
if((password_1==2'b01)&&(password_2==2'b10))
next_state = RIGHT_PASS;
else
next_state = WRONG_PASS;
end
```

Explanation:

In **wait_password** 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 wrong it will go to the **WRONG_PASS** state and if the password is correct it will go to the **RIGHT_PASS** state.

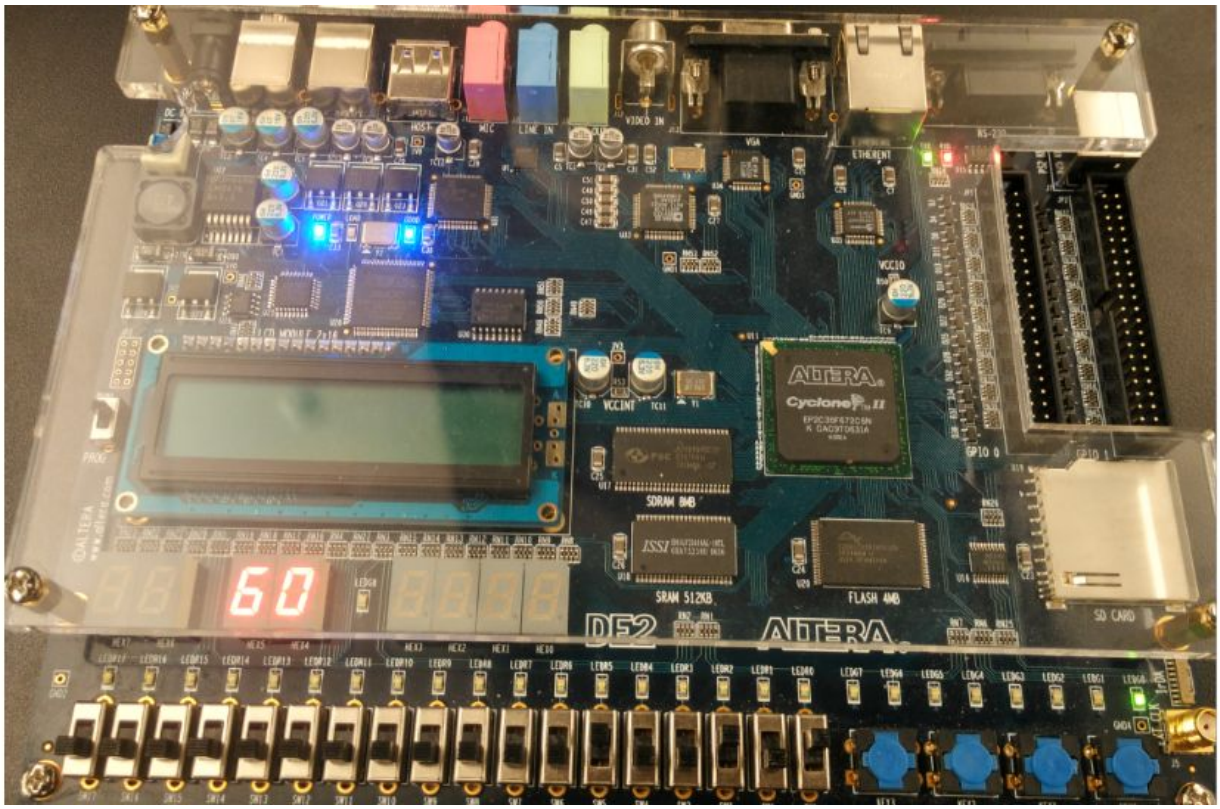


Image3: correct_password(GO) ((password_1==2'b01)&&(password_2==2'b10)

3. Right Password

Code:

```
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
```

Explanation:

If the password entered correctly in wait state, Green light will show in **Right_pass** 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 **idle state** if there is no other car coming.

If another car is coming it will turned on both sensors sensor_entrance==1 && sensor_exit == 1 and so it will go the **STOP state**.

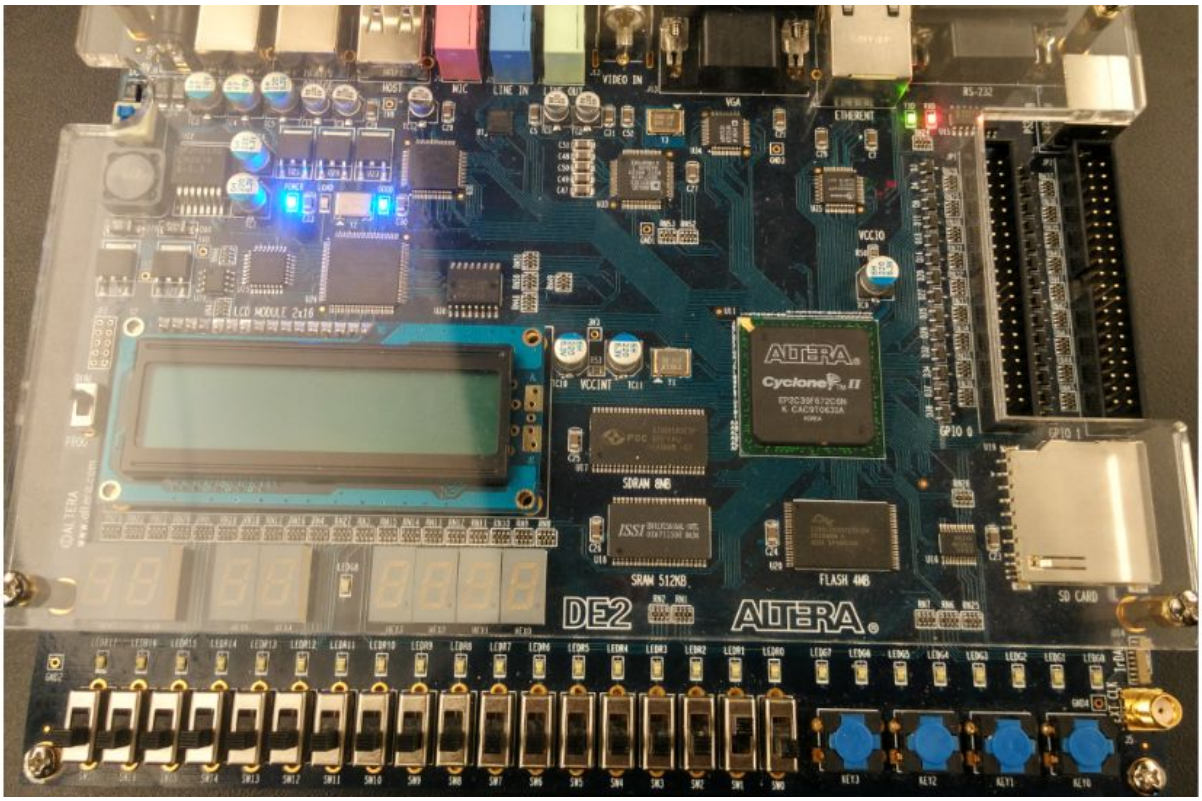


image4(no other car is coming: sensor_exit==1, next state= idle state)

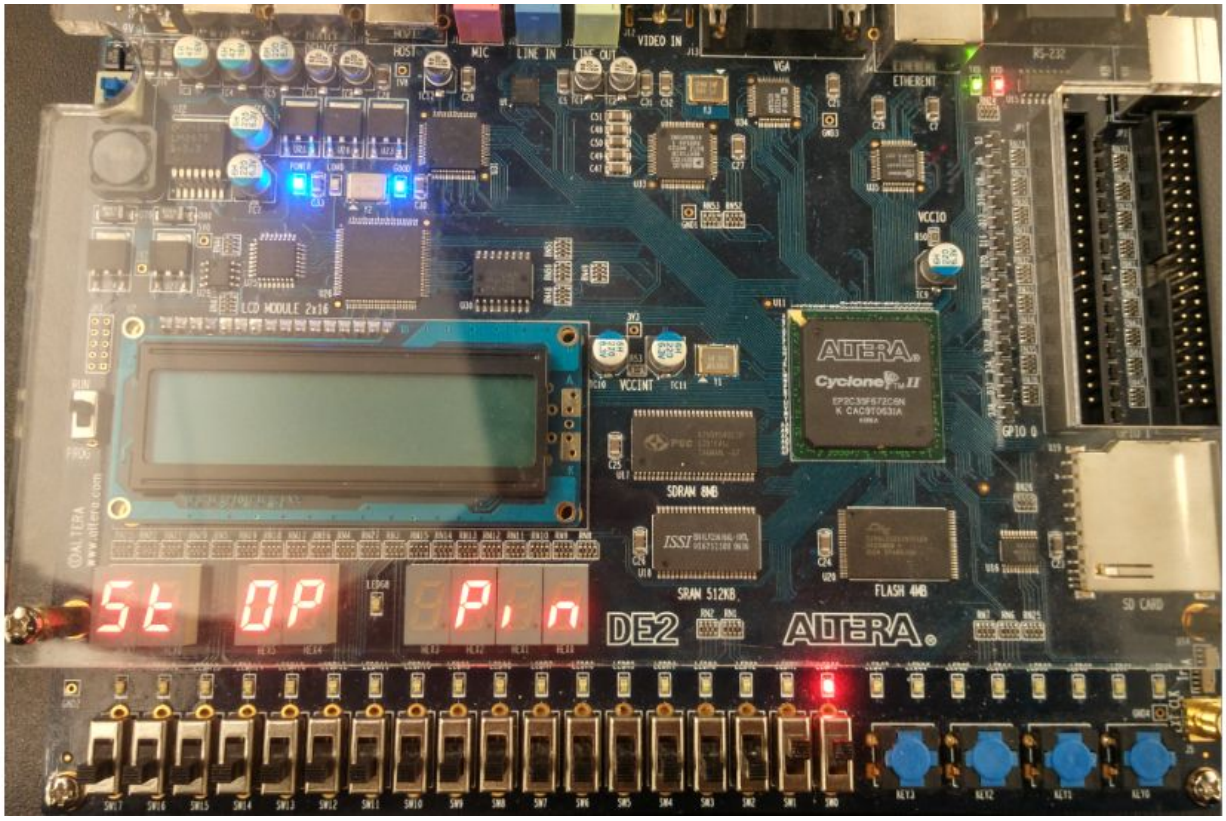


Image5(another car coming both sensors on)
sensor_entrance==1 && sensor_exit == 1

4. **Wrong Password**

Code:

```
WRONG_PASS: begin
 if((password_1==2'b01)&&(password_2==2'b10))
 next_state = RIGHT_PASS;
 else
 next_state = WRONG_PASS;
 end
```

Explanation:

If the password is wrong, **wrong 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 right password after the first failed trial, it will go to the **right password** state, where driver can pass through this system.

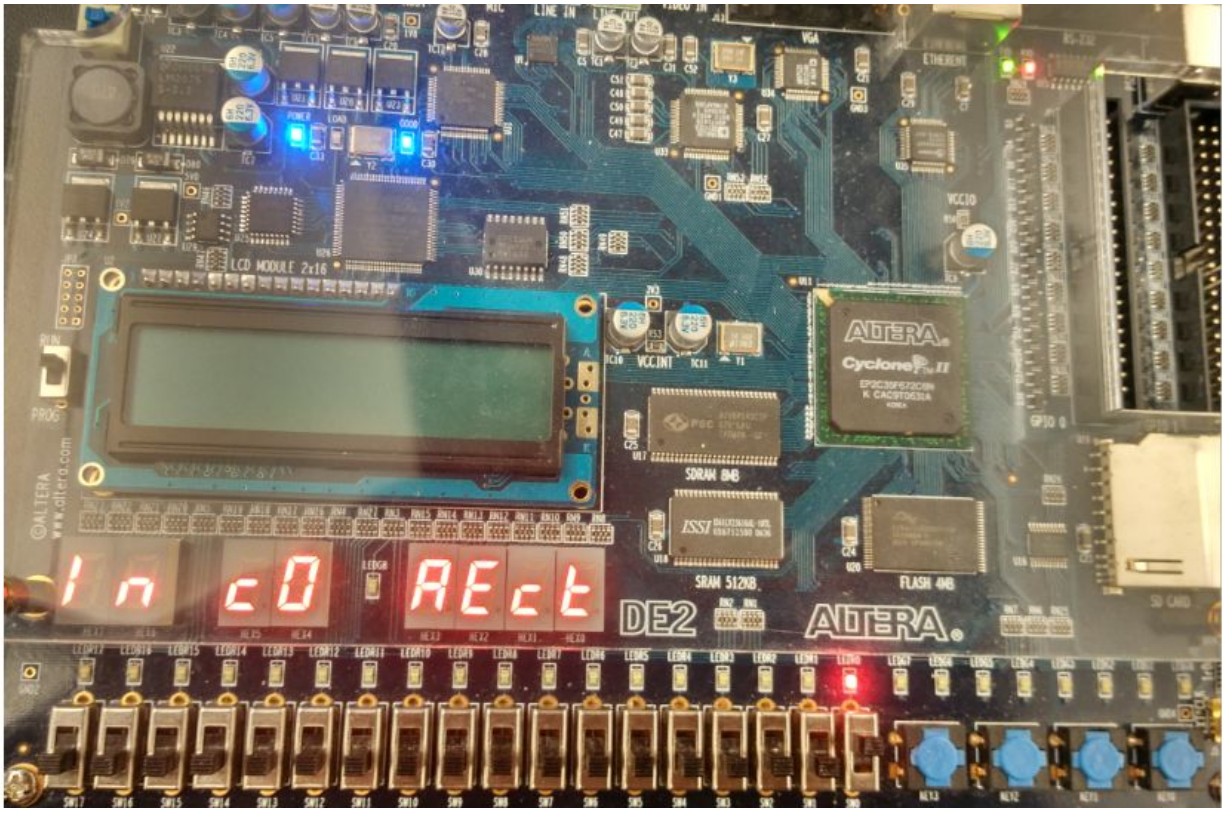


Image6: wrong password(Password entered is wrong)

5. STOP

Code:

```
STOP: begin
 if((password_1==2'b01)&&(password_2==2'b10))
 next_state = RIGHT_PASS;
 else
 next_state = STOP;
 end
 default: next_state = IDLE
 end
```

Explanation:

If another car came after the first car has gone, it will ask to stop and enter the password showing Red light.

If then driver entered the right password it will go to the **right state** else it will stay in the same state and ask for the pin again.

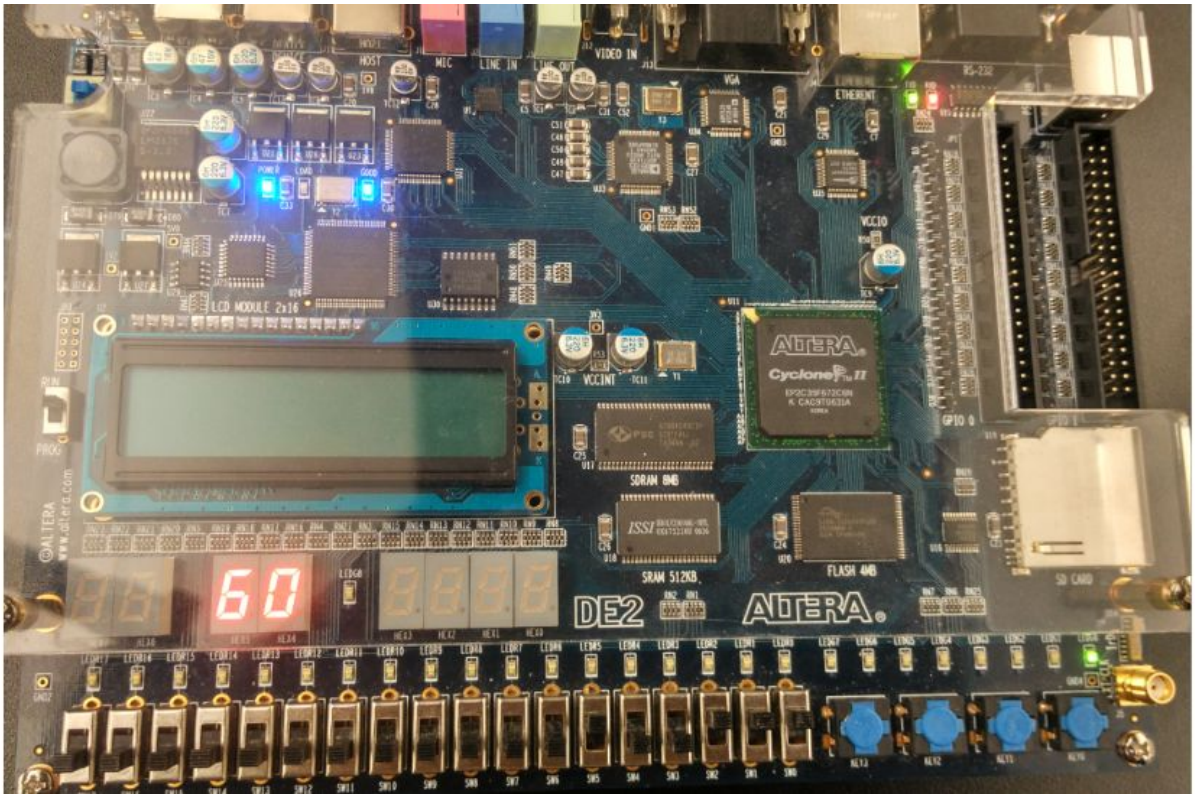


Image7(right password (password_1==2'b01&& password_2==2'b10) entered after stop state)

CONCLUSION

This system can counter the parking problems that arise due to the unavailability of a reliable, efficient and modern Parking system.

Human errors can be avoided by installing this parking system.

There are several advantages of employing a car park system for urban planners, business owners and vehicle drivers. They offer convenience for vehicle users and efficient usage of space for urban-based companies.

Automated car park systems save time, money, space and simplify the often tedious task of parking.

Future work should be done for integrating different technologies together in order to achieve a system which is the most efficient, reliable, secure and inexpensive.

References:

1. Fundamentals_of_Digital_Logic_with_Verilog_Design_(3rd_edition)
2. https://www.geeksforgeeks.org/mealy-and-moore-machines/
3. https://skyline-parking.com/
4. https://cityliftparking.com/solutions/case-studies
5. https://www.westfaliausa.com/products/parking

Appendixes:

The complete code for the project has written below:

Main Module:

```
module simple (SW,
LEDR,HEX0,HEX1,HEX2,HEX3,HEX4,HEX5,HEX6,HEX7,KEY,CLOCK_50, LEDG);
//Main Module

input [5:0] SW; // Toggle switches

input [1:0] KEY;

input CLOCK_50;

output [7:0]LEDR; //Red LEDs

output [7:0]LEDG;

output [6:0] HEX0,HEX1,HEX2,HEX3,HEX4,HEX5,HEX6,HEX7; //LCD 7-Segment


parking_system
instant1(SW[0],SW[1],SW[3:2],SW[5:4],KEY[1],KEY[0],LEDG[0],LEDR[0],HEX7,HEX6,HE
X5,HEX4,HEX3,HEX2,HEX1,HEX0,CLOCK_50);


endmodule
```

Module for FSM:

```
module

parking_system(sensor_entrance,sensor_exit,password_1,password_2,Clock_KEY1,Reset_KEY
0,GREEN_LED,RED_LED,HEX_7,HEX_6,HEX_5,HEX_4,HEX_3,HEX_2, HEX_1, HEX_0,
Clk_50);


input sensor_entrance;

input Clock_KEY1,Reset_KEY0,Clk_50;

input sensor_exit;

input [1:0]password_1, password_2;

output [6:0] HEX_7,HEX_6,HEX_5,HEX_4,HEX_3,HEX_2,HEX_1,HEX_0;

output  GREEN_LED;

output  RED_LED;


//Using Debounce for generate 1 Clock

wire clk;

DeBounce De1 (Clk_50,Clock_KEY1,clk);

//Using Debounce for generate 1 Clock

wire reset_n;

DeBounce De0 (Clk_50,Reset_KEY0,reset_n);


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;
reg green_tmp;
reg [6:0] HEX_7,HEX_6,HEX_5,HEX_4,HEX_3,HEX_2,HEX_1, HEX_0;
 // Next state
 always @(posedge clk or negedge reset_n)
 begin
 if(~reset_n)
 current_state = IDLE;
 else
 current_state = next_state;
 end

 always @(posedge clk or negedge reset_n)
 begin
 case(current_state)
 IDLE: begin
        if(sensor_entrance == 1)
 next_state = WAIT_PASSWORD;
 else
 next_state = IDLE;
 end

 WAIT_PASSWORD: begin
```

```
if((password_1==2'b01)&&(password_2==2'b10))
next_state = RIGHT_PASS;
else
next_state = WRONG_PASS;
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 or negedge reset_n) begin
case(current_state)
IDLE: begin
green_tmp = 1'b0;
red_tmp =   1'b0;
HEX_0 = 7'b1111111; // off
HEX_1 = 7'b1111111; // off
HEX_2 = 7'b1111111;
HEX_3 = 7'b1111111;
HEX_4 = 7'b1111111;
HEX_5 = 7'b1111111;
HEX_6 = 7'b1111111;
HEX_7 = 7'b1111111;

end
WAIT_PASSWORD: begin
green_tmp = 1'b0;
red_tmp =   1'b1;
HEX_7 = 7'b0000110; // E
```

```
HEX_6 = 7'b0101011; // n
HEX_5 = 7'b0000111; //t
HEX_4 = 7'b0000110; //e
HEX_3 = 7'b0001000; //r
HEX_2 = 7'b0001100; //p
HEX_1 = 7'b1101111; //i
HEX_0 = 7'b0101011; //n

end
WRONG_PASS: begin
green_tmp = 1'b0;
red_tmp =   1'b1;
HEX_7 = 7'b1101111; //i
HEX_6 = 7'b0101011; // n
HEX_5 = 7'b0100111; // c
HEX_4 = 7'b1000000; // o
HEX_3 = 7'b0001000; // r
HEX_2 = 7'b0000110; // e
HEX_1 = 7'b0100111; // c
HEX_0 = 7'b0000111; // t
end
RIGHT_PASS: begin
green_tmp = 1'b1;
red_tmp = 1'b0;
HEX_7 = 7'b1111111; //
HEX_6 = 7'b1111111; //
```

```
HEX_5 = 7'b0000010; // 6
HEX_4 = 7'b1000000; // 0
HEX_3 = 7'b1111111; //
HEX_2 = 7'b1111111; //
HEX_1 = 7'b1111111; //
HEX_0 = 7'b1111111; //
end
STOP: begin
green_tmp = 1'b0;
red_tmp =  1'b1;
HEX_2 = 7'b0001100; //p
HEX_1 = 7'b1101111; //i
HEX_0 = 7'b0101011; //n
HEX_3 = 7'b1111111; //
HEX_7 = 7'b0010010; // 5
HEX_6 = 7'b0000111; // t
HEX_5 = 7'b1000000; // o
HEX_4 = 7'b0001100; // P
end
endcase
end
assign RED_LED = red_tmp  ;
assign GREEN_LED = green_tmp;

endmodule
```

CODE for Clock generation using Debounce:

```
module DeBounce(
 input clk, //this is a 50MHz clock provided on FPGA pin PIN_Y2
 input PB, //this is the input to be debounced
 output reg PB_state //this is the debounced switch
);
// Synchronize the switch input to the clock
reg PB_sync_0;
always @(posedge clk)
PB_sync_0 <= PB;
reg PB_sync_1;
always @(posedge clk)
PB_sync_1 <= PB_sync_0;
// Debounce the switch
reg [15:0] PB_cnt;
always @(posedge clk)
if(PB_state==PB_sync_1)
PB_cnt <= 0;
else
```

```
begin
PB_cnt <= PB_cnt + 1'b1;
if(PB_cnt == 16'hffff) PB_state <= ~PB_state;
end
endmodule
```