MAPUA University

School of Electrical, Electronics and Computer Engineering

INTRODUCTION TO HDL

PROJECT DOCUMENTATION

Moleno, Jethro P.

CPE114L / E01

Engr. Meo Vincent Caya

I. Problem Statement

Initially, the FSM is in IDLE state. If there is a vehicle coming detected by the front sensor, FSM is switched to WAIT_FOR_PASSWORD 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 ACCEPT_PASS state; a Green LED will be blinking. Otherwise, FSM turns to DENIED_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.

- Please provide the FSM model using Mealy or Moore.
- Create a program module in Verilog with corresponding testbench
- The password corresponds to a 4-digit number.
- Include a counter on how many cars already enter the Car parking system

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 getting into the car park.

II. State Table

Inputs:
Fs = front sensor
Bs = back sensor
P4 = 4th digit in password P3 = 3rd digit in password P2 = 2nd digit in password P1 = 1st digit in password
True password: 1234 (0001001000110100)
Wrong password: 0000 (0000000000000000)
Parameters:
carCount = set to 15 maximum cars in parking system

Present State	Input						Output		Next State
	Fs	Bs	P4	P3	P2	P1	gLED	rLED	
IDLE	1	0	0000	0000	0000	0000	0	0	WAIT_FOR_PASSWORD
WAIT_FOR_PASSWORD	1	0	0000	0000	0000	0000	0	1	DENIED_PASS
WAIT_FOR_PASSWORD	1	0	0001	0010	0011	0100	1	0	ACCEPT_PASS
ACCEPT_PASS	1	0	0000	0000	0000	0000	1	0	ACCEPT_PASS
ACCEPT_PASS	0	1	0001	0010	0011	0100	0	0	IDLE
ACCEPT_PASS	1	1	0001	0010	0011	0100	0	1	STOP
DENIED_PASS	1	0	0000	0000	0000	0000	0	1	DENIED_PASS
DENIED_PASS	1	0	0001	0010	0011	0100	1	0	ACCEPT_PASS
STOP	1	0	0000	0000	0000	0000	0	1	STOP
STOP	1	0	0001	0010	0011	0100	1	0	ACCEPT_PASS

Table 1 Car Parking System State Table

II. Mealy Model

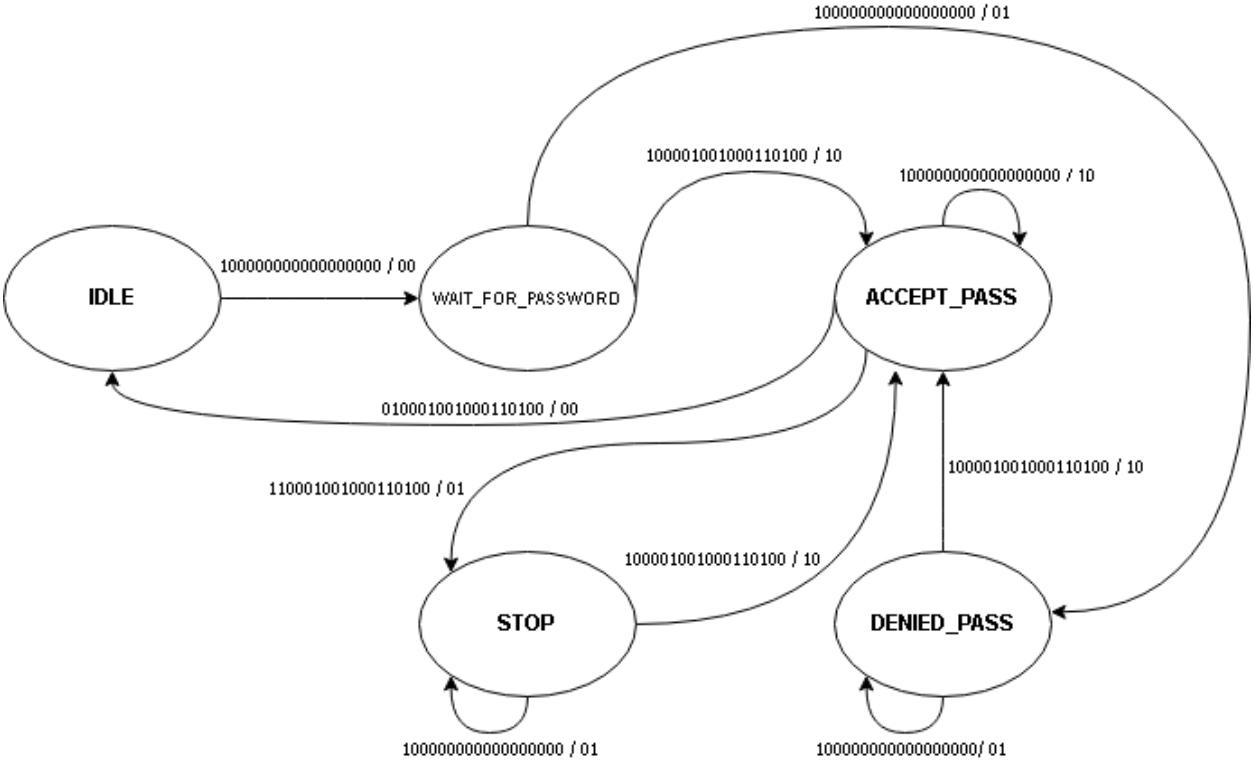

Figure 1 Mealy Model for Car Parking System

III. Testbench Stimulus

1. **Car 1 enters the parking.**
2. **Car 1 enters the right password.**
3. **Car 1 parks.**
4. **Car 2 enters the parking**
5. **Car 2 enters right password.**
6. **Car 2 parks and Car 3 enters simultaneously.**
7. **Car 3 enters wrong password.**
8. **Car 3 enters right password.**
9. **Car 3 parks.**
10. **Car 4 enters the parking.**
11. **Car 4 enters the right password.**
12. **Car 4 parks.**

IV. Verilog Code

Car Parking System Verilog Code

```
`timescale 1ns / 1ps
module car_parking_system(Fs, Bs, P4, P3, P2, P1, clk, reset, gLED, rLED, carCount);
        // Inputs
        input Fs; //front sensor
        input Bs; //back sensor
        input [3:0] P4, P3, P2, P1; // 4-digit password
        input clk; //clock
        input reset; //reset
        output wire gLED, rLED; //green LED & red LED
        output reg [3:0] carCount;
        //states
        parameter IDLE = 3'b000;
        parameter WAIT_FOR_PASSWORD = 3'b001;
        parameter DENIED_PASS = 3'b010;
        parameter ACCEPT_PASS = 3'b011;
        parameter STOP = 3'b100;

        //registers
        reg [2:0] currState, nextState;
        reg gLamp, rLamp;

        //current State -> next State
        always @(posedge clk or negedge reset)
        begin
                if(~reset)
                begin
                     currState = IDLE;
                     carCount = 4'b0000;
                end
                else
                     currState = nextState;
        end

        //car counter
        always @(posedge Bs)
        begin
                carCount = carCount + 1;
        end

        //next state
        always @(*)
        begin
                case(currState)
                     IDLE: begin
                        if(Fs == 1)
                        begin
                             nextState = WAIT_FOR_PASSWORD;
                        end
                        else
                             nextState = IDLE;
                        end
                     WAIT_FOR_PASSWORD: begin
                        if(countWait <= 3)
                             nextState = WAIT_FOR_PASSWORD;
                        else
                        begin
                             if((P4 == 4'b0001) && (P3 == 4'b0010) && (P2 == 4'b0011) && (P1 == 4'b0100))
                             nextState = ACCEPT_PASS;
```

```
                                else
                                nextState = DENIED_PASS;
                                end
                            end
                        DENIED_PASS: begin
                            if((P4 == 4'b0001)&&(P3 == 4'b0010)&&(P2 == 4'b0011)&&(P1 == 4'b0100))
                                nextState = ACCEPT_PASS;
                            else
                                nextState = DENIED_PASS;
                            end
                        ACCEPT_PASS: begin
                            if(Fs == 1 && Bs == 1)
                                nextState = STOP;
                            else if (Bs == 1)
                            begin
                                nextState = IDLE;
                            end
                            else if((P4 == 4'b0001)&&(P3 == 4'b0010)&&(P2 == 4'b0011)&&(P1 == 4'b0100))
                                nextState = ACCEPT_PASS;
                            else
                                nextState = DENIED_PASS;
                            end
                        STOP: begin
                            if((P4 == 4'b0001)&&(P3 == 4'b0010)&&(P2 == 4'b0011)&&(P1 == 4'b0100))
                                nextState = ACCEPT_PASS;
                            else
                                nextState = STOP;
                            end
                        default: nextState = IDLE;
                    endcase
            end

            //LED status dependent on the current state
            always @(posedge clk)
            begin
                    case(currState)
                        IDLE: begin
                            gLamp = 1'b0;
                            rLamp = 1'b0;
                        end
                        WAIT_FOR_PASSWORD: begin
                            gLamp = 1'b0;
                            rLamp = 1'b1;
                        end
                        DENIED_PASS: begin
                            gLamp = 1'b0;
                            rLamp = ~rLamp;
                        end
                        ACCEPT_PASS: begin
                            gLamp = ~gLamp;
                            rLamp = 1'b0;
                        end
                        STOP: begin
                            gLamp = 1'b0;
                            rLamp = ~rLamp;
                        end
                    endcase
            end
            assign gLED = gLamp;
            assign rLED = rLamp;

endmodule
```

Car Parking System Testbench Code

```
`timescale 1ns / 1ps
module test_carparkingsystem;
        //inputs
        reg Fs;
        reg Bs;
        reg [3:0] P4, P3, P2, P1;
        reg clk;
        reg reset;
        wire gLED, rLED;
        wire [3:0]carCount;


        car_parking_system uut(.Fs(Fs), .Bs(Bs), .P4(P4), .P3(P3), .P2(P2), .P1(P1), .clk(clk), .reset(reset), .gLED(gLED),

        //initialize clock
        initial begin
        clk = 0;
        forever #10 clk = ~clk;
        end

        //initialize variables
        initial begin
        reset = 0;
        Fs = 0;
        Bs = 0;
        P4 = 0;
        P3 = 0;
        P2 = 0;
        P1 = 0;
        #100;

        //stimulus
        reset = 1; #20;
        //car 1 enter
        Fs = 1; #100;
        P4 = 4'b0001; P3 = 4'b0010; P2 = 4'b0011; P1 = 4'b0100; #100;
        Fs = 0; Bs = 1; #100;
        //car 1 park
        Bs = 0; #100;
        //car 2 enter
        Fs = 1; #100;
        //car 2 inputs password
        P4 = 4'b0001; P3 = 4'b0010; P2 = 4'b0011; P1 = 4'b0100; #100;
        //car2 park, car 3 triggered front sensor
        Bs = 1; Fs = 1; #100;
        Bs = 0; #100;
        //car3 input pass
        P4 = 4'b0000; P3 = 4'b0010; P2 = 4'b0000; P1 = 4'b0100; #100;
        P4 = 4'b0001; P3 = 4'b0010; P2 = 4'b0011; P1 = 4'b0100; #100;
        Fs = 0; #100;
        Bs = 1; #100;
        //car3 park
        Bs = 0; #100;
        //car 4 enter
        Fs = 1; #100;
        P4 = 4'b0001; P3 = 4'b0010; P2 = 4'b0011; P1 = 4'b0100; #100;
        Bs = 1; #100;
        //car 4 park
        Fs = 0; #100;
        Bs = 0; #100;
        end

endmodule
```

V. Output Waveform

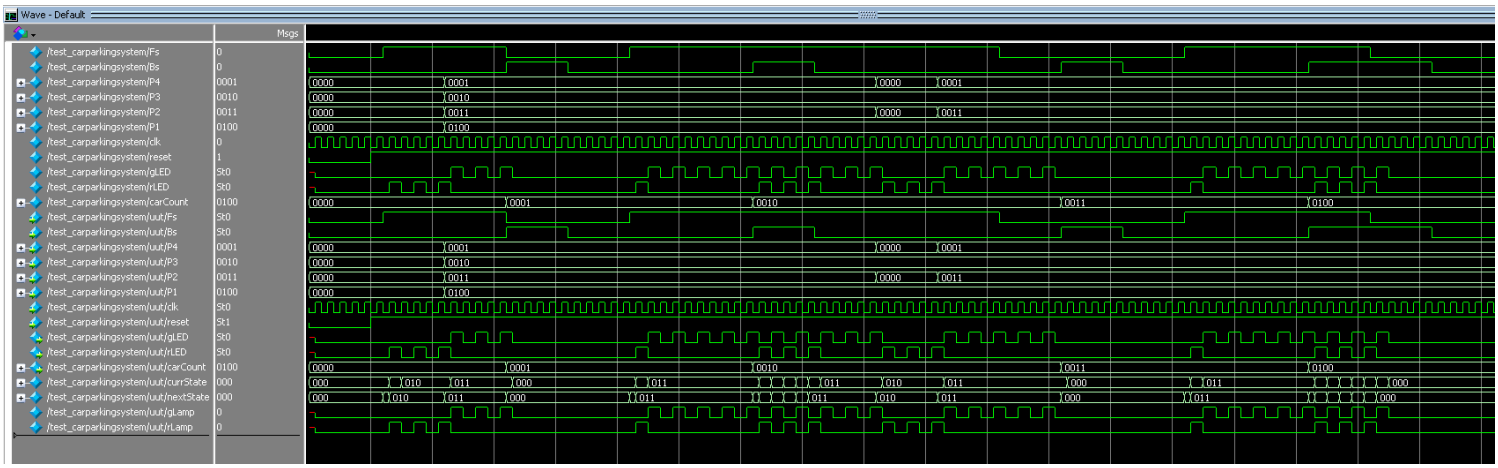

Figure 2 Output Waveform for both Car Parking system and its Testbench