**Digital System Design Lab CSE-308L**

**Semester:6th**



**Lab Report # 9**

**Traffic light controller using FSM**

**Submitted By:** *Zainab Khalid*

**Registration No:***19PWCSE1743*

**Section: A**

**Submitted to:** *Mam Madeeha Sher*

**Department of Computer Systems Engineering**

**University of Engineering and Technology Peshawar**

**LAB 09**

**TASK:**

**Implement a Traffic Light controller using FSM.**

**INTRODUCTION:**

The lab this week will continue the introduction to FSMs with a simple Traffic Light Controller design project. In the lab, you are required to create a state diagram of a Mealy machine, which implements a traffic light controller, based on provided guidelines. You will then use this state diagram to write the behavioral Verilog description of the traffic light controller. The testing of your traffic light controller will take place during the lab session using the ISE development tools and the Spartan 6 board. As with the previous lab, we will make use of the character LEDs to display the outputs of our digital circuit.

**THE TRAFFIC LIGHT CONTROLLER:**

This week you will design a traffic light controller for the highway and farm road intersection shown in Figure 1. For simplicity, we will assume that the traffic lights on opposite ends of the intersection are the same; in other words, if the light for the North bound traffic is green, then the light for the South bound traffic is also green. The traffic light controller outputs two 2-bit signals, highwaySignal and farmSignal, for the highway road and the farm road, respectively.

Diagram

Description automatically generated

Figure 1: Example Illustration

The following encoding is used for both signals:

***001 : Green***

***010 : Yellow***

***100 : Red***

Obviously, to avoid accidents, the highway lights and farm road lights must not be green at the same time. Furthermore, the transitions from green on each road must be followed by 3 seconds of yellow. Because the traffic on the highway is much greater than that of the farm road, the traffic light must remain green on the highway longer than on the farm road. For this design, the highway light must remain green unless there is a vehicle on the farm road, while the farm road light must remain green for only 10 seconds.

The brain of the traffic light controller is the FSM. For the initial design, Vehicle, Clock and Rst constitute the only input to the FSM. The output of the FSM includes the highwaySignal and farmSignal. The operation of the FSM came be described by using a state diagram as shown in figure 2.

HG\_FR

HR\_FY

HR\_FG

HY\_FR

H=001 F=100

H=100 F=010

H=100 F=001

H=010 F=100

V = 0

V = 1

D < 3 sec

D = 3 sec

D < 10 sec

D = 10 sec

D < 3sec

D = 3 sec

Figure 2: State Transition Diagram

**CODE:**

module D\_FF (q, d, clock, reset);

output reg q;

input d, clock, reset;

always @(negedge clock or posedge reset)

begin

if (reset)

q = 1'b0;

else

q = d;

end

endmodule

module syncronizer(

input clk, rst, in,

output out

);

wire w1;

D\_FF DF01 (w1, in, clk, rst);

D\_FF DF02 (out, w1, clk, rst);

endmodule

module clk\_divider(clk, clk1, rst);

integer cnt;

input clk, rst;

output reg clk1;

always @(posedge clk)

begin

if(rst)

begin

cnt = 0;

clk1 = 0;

end

else

begin

cnt = cnt + 1;

if(cnt == 100000000)

begin

clk1 = ~clk1;

cnt = 0;

end

end

end

endmodule

module L2P(

input clk1, rst, in,

output reg out

);

syncronizer sync (clk1, rst, in, synin);

parameter s0 = 0;

parameter s1 = 1;

reg state;

reg next\_state;

always @(posedge clk1)

if(rst)

begin

state = s0;

end

else

state = next\_state;

always @(\*) begin

case (state)

s0:

begin

if(synin == 0)

begin

out = 0;

next\_state = s0;

end

else if(synin== 1)

begin

out = 1;

next\_state = s1;

end

end

s1:

begin

if(synin== 0)

begin

out = 0;

next\_state = s0;

end

else if(synin == 1)

begin

out = 0;

next\_state = s1;

end

end

endcase

end

endmodule

//module lock (on, off, reset,clk,onpulse,offpulse, out);

module traffic\_controller(reset, clk, VEHICAL\_IN, HighWayLight, FarmLight,state7seg,enable);

input reset, VEHICAL\_IN, clk;

output reg [2:0] HighWayLight, FarmLight;

output [6:0] state7seg;

output enable;

reg [1:0] state, next\_state;

assign enable =0;

clk\_divider clk\_Div (clk, clk1, reset);

L2P l1(clk1, reset, VEHICAL\_IN, VEHICAL);

seven\_seg seg1(state, state7seg);

parameter RED = 3'b100;

parameter GREEN = 3'b001;

parameter YELLOW = 3'b010;

parameter FR\_HG=0;

parameter FY\_HR=1;

parameter FG\_HR=2;

parameter FR\_HY=3;

always @(posedge clk1)

if(reset)

begin

state = FR\_HG;

end

else

state = next\_state;

always @(\*) begin

case(state)

FR\_HG:

begin

if(VEHICAL == 1)

begin

next\_state = FY\_HR;

FarmLight = YELLOW;

HighWayLight = RED;

end

else

begin

next\_state = FR\_HG;

HighWayLight = GREEN;

FarmLight = RED;

end

end

FY\_HR:

begin

FarmLight = YELLOW;

HighWayLight = RED;

next\_state = FG\_HR;

end

FG\_HR:

begin

FarmLight = GREEN;

HighWayLight = RED;

next\_state = FR\_HY;

end

FR\_HY:

begin

FarmLight = RED;

HighWayLight = YELLOW;

next\_state = FR\_HG;

end

endcase

end

endmodule

module seven\_seg(in, out);

input [1:0] in;

output [6:0] out;

assign out = (in == 0) ? 7'b1000000:

(in == 1) ? 7'b1111001:

(in == 2) ? 7'b0100100:

(in == 3) ? 7'b0110000: 7'b1111111;

endmodule

// module seven\_seg(in, out);

// input [1:0] in;

// output [6:0] out;

// assign out = (in == 0) ? 7'd0:

// (in == 1) ? 7'b1111001:

// (in == 2) ? 7'b0100100:

// (in == 3) ? 7'b0110000:

// (in == 4) ? 7'b0011001:

// (in == 5) ? 7'b0010010:

// (in == 6) ? 7'b0000010:

// (in == 7) ? 7'b1111000:

// (in == 8) ? 7'b0000000:

// (in == 9) ? 7'b0011000: 7'b1111111;

// endmodule

**UCF FILE:**

//reset, clk, VEHICAL\_IN, HighWayLight, FarmLight,state7seg

NET "reset" LOC = C17 | PULLUP;

NET "clk" LOC = V10 | PULLUP;

NET "VEHICAL\_IN" LOC = F17 | PULLUP;

NET "HighWayLight[0]" LOC = P15;

NET "HighWayLight[1]" LOC = P16;

NET "HighWayLight[2]" LOC = N15;

NET "FarmLight[0]" LOC = N16;

NET "FarmLight[1]" LOC = U17;

NET "FarmLight[2]" LOC = U18;

NET "state7seg[0]" LOC = A3;

NET "state7seg[1]" LOC = B4;

NET "state7seg[2]" LOC = A4;

NET "state7seg[3]" LOC = C4;

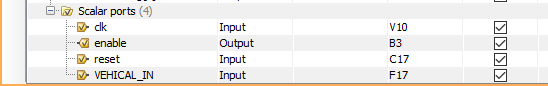
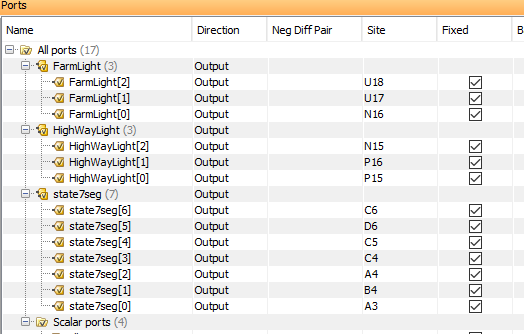
NET "state7seg[4]" LOC = C5;

NET "state7seg[5]" LOC = D6;

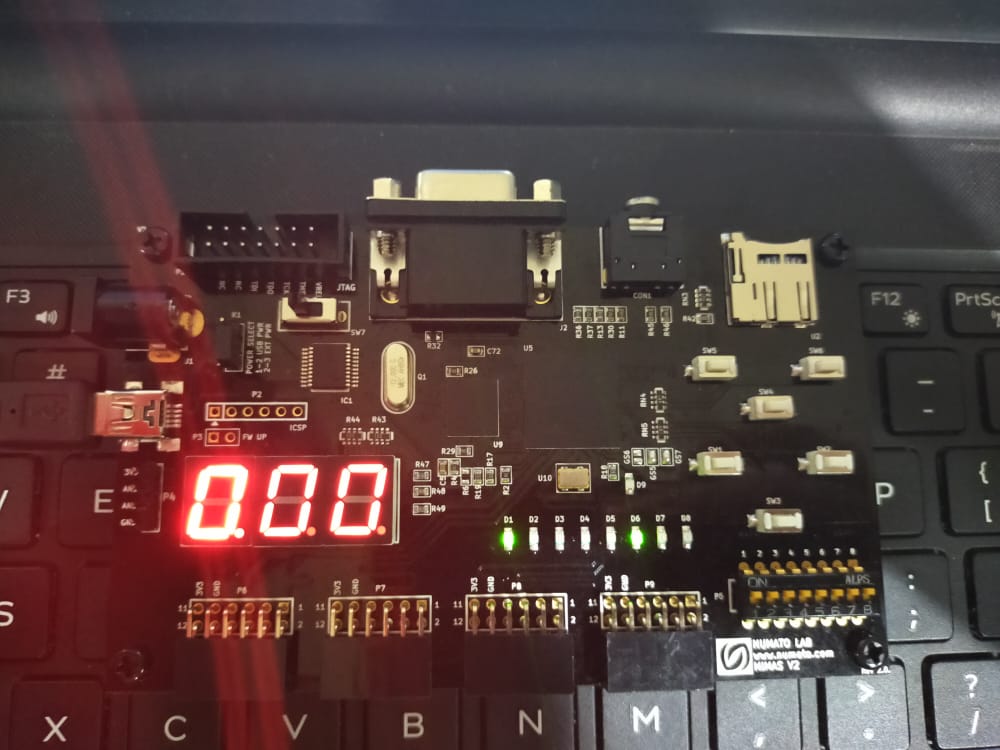
NET "state7seg[6]" LOC = C6;

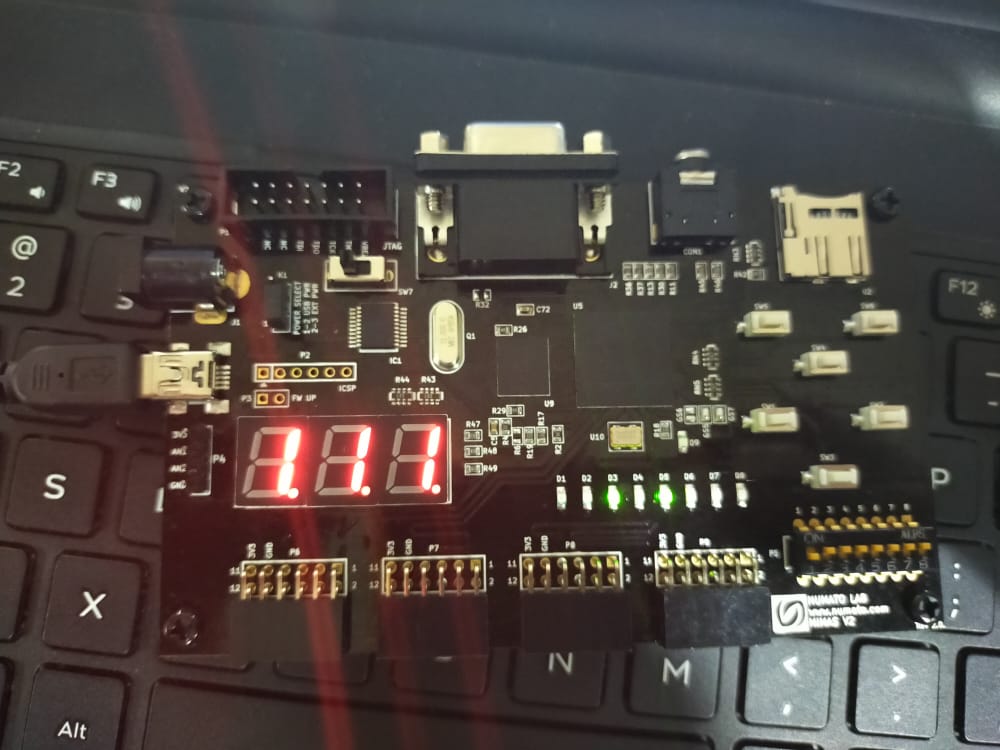
NET "enable" LOC = B3;

**I/O PIN PLANNING:**

****

**OUTPUT:**

****

****

****

****