College of Electrical & Mechanical Engineering, NUST

Department of Computer Engineering

EC-410 Digital System Design

Assignment # 1

Course Instructor: Dr Yasin

Student Name: **Salman Mehboob**

Degree/ Syndicate: **42/A**

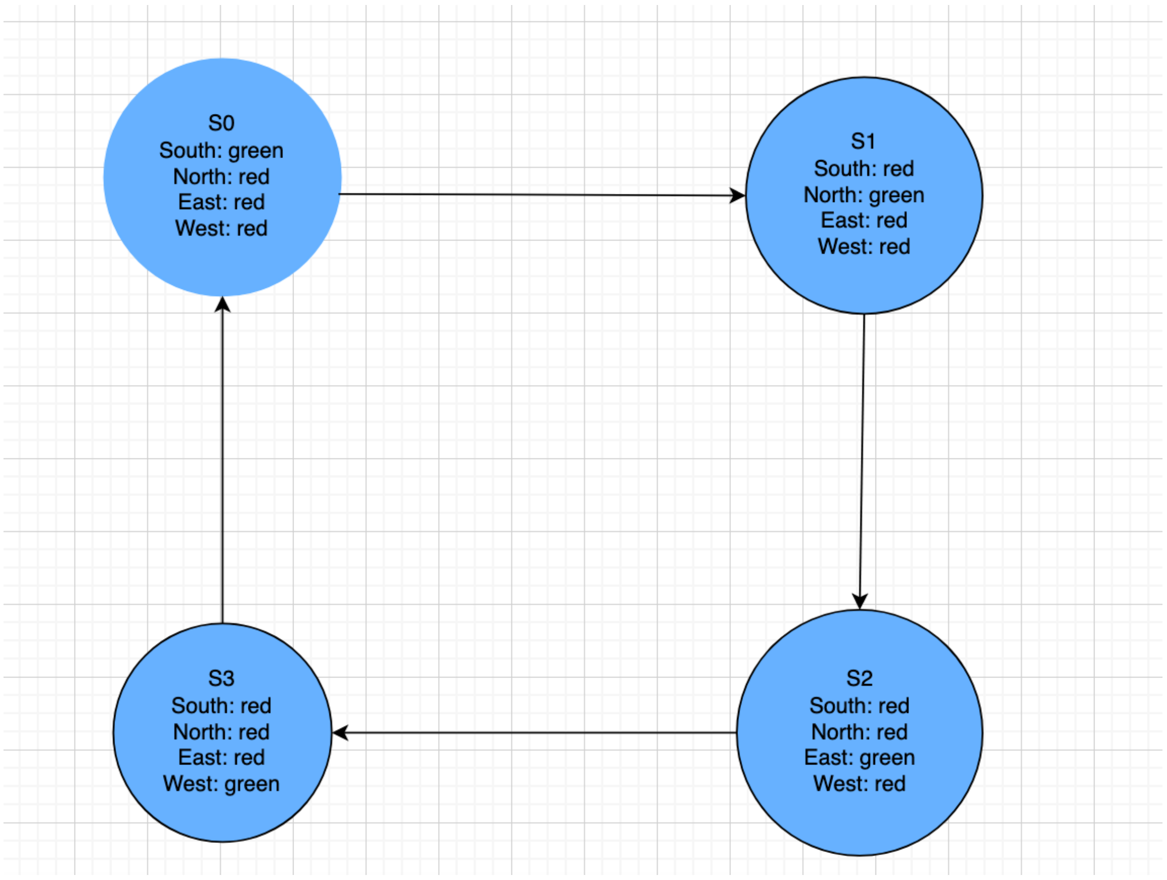
Cms id: 338541

Assignment Task:

1. Visit a traffic light intersection in Pakistan with 4-way traffic, preferably one that has a time display to convey the wait time to drivers. Record two videos documenting complete operation of the traffic lights: the sequence in which lights turn ON or OFF, and the intervals between each ON/OFF operation. (10 marks)

- **Draw the FSM. (10 marks)**
- **Implement the FSM in Verilog and simulate using Xilinx ISE, Modelsim, or EDAplayground.(15-marks)**

Implement the FSM (or a large subset of it) on Nexys 3 FPGA board. (15 marks



Verilog Code:

```
`timescale 1ns / 1ps

module trafficlightpak(
        input clk,
        input reset,
        output reg LED_south, LED_north, LED_east, LED_west,
        output [3:0] anod,
        output [7:0] cath
 );
        reg [1:0] state, next_state;
        reg [1:0] count3 = 2'b00;
        reg rst_counter;
        reg [3:0] anode;
        reg [7:0] cathode;
        reg [26:0] one_sec_clk_count;
        wire clk_enabler;
        reg [3:0] count;
        reg [3:0] count2;
        reg [3:0] counter;
        reg [19:0] refresh_counter;
        wire [1:0] LED_act;

        parameter RED = 1'b0, GREEN = 1'b1;
        parameter S0 = 2'b00, // SOUTH=GREEN ELSE=RED
                                S1 = 2'b01, // NORTH=GREEN ELSE=RED
                                S2 = 2'b10, // WEST=GREEN ELSE=RED
                                S3 = 2'b11; // EAST=GREEN ELSE=RED

        // state register
        always @(posedge clk or posedge reset)
        begin
                if (reset)
                        state <= S0;
                else
                        state <= next_state;
        end

        // next state logic
        always @(*)
        begin
```

```
                next_state = state;
                rst_counter = 0;
        case(state)
        S0: begin
                if (count3==2'b01) begin next_state = S1; rst_counter = 1; end
        end
        S1: begin
                if (count3==2'b10) begin next_state = S2; rst_counter = 1; end
        end
        S2: begin
                if (count3==2'b11) begin next_state = S3; rst_counter = 1; end
        end
        S3: begin
                if (count3==2'b00) begin next_state = S0; rst_counter = 1; end
        end
        endcase
        end

        // output logic
        always @(*)
        begin
        case(state)
        S0: begin LED_south=GREEN; LED_north=RED; LED_west=RED;
LED_east=RED; end
        S1: begin LED_south=RED; LED_north=GREEN; LED_west=RED;
LED_east=RED; end
        S2: begin LED_south=RED; LED_north=RED; LED_west=GREEN;
LED_east=RED; end
        S3: begin LED_south=RED; LED_north=RED; LED_west=RED;
LED_east=GREEN; end
        endcase
        end

        always @(posedge clk or posedge reset)
        begin
                if(reset==1)
                        one_sec_clk_count <= 0;
                else begin
                        if(one_sec_clk_count>=99999999)
                                one_sec_clk_count <= 0;
                        else
                                one_sec_clk_count <= one_sec_clk_count + 1;
                end
        end
        assign clk_enabler = (one_sec_clk_count==99999999)?1:0;

        always @(posedge clk or posedge reset)
        begin
        if(reset==1) begin
                count <= 0;
                count2 <= 0;
        end
        else if(clk_enabler==1) begin
                if (count==4'b0 && count2==4'b0) begin
                        count3 <= count3+1;
                        if (count3==2'b00) begin count2<=4'b0001; end // 19
seconds
                        else if (count3==2'b01) begin count2<=4'b0001; end // 19
seconds
                        else if (count3==2'b10) begin count2<=4'b0001; end // 19
seconds
                        else if (count3==2'b11) begin count2<=4'b0011; end // 39
seconds
                        count<=4'b1001;

                end
                else if (count==4'b0) begin
                        count<=4'b1001;
                        count2 <= count2 - 1;
                end
                else begin
                        count <= count - 1;
                end
        end
        end

        always @(posedge clk or posedge reset)
        begin
                if(reset==1)
                        refresh_counter <= 0;
                else
                        refresh_counter <= refresh_counter + 1;
        end
        assign LED_act = refresh_counter[19:18];

        always @(*)
        begin
        case(LED_act)
```

```
                        2'b00: begin
                                anode = 4'b0111;
                                counter = 4'b0;
                        end
                        2'b01: begin
                                anode = 4'b1011;
                                counter = 4'b0;
                        end
                        2'b10: begin
                                anode = 4'b1101;
                                counter = count2;
                        end
                        2'b11: begin
                                anode = 4'b1110;
                                counter = count;
                        end
        endcase
        end

        always @(*)
        begin
  case(counter)
     4'b0: cathode = 8'b11000000; // "0"
     4'b0001: cathode = 8'b11111001; // "1"
     4'b0010: cathode = 8'b10100100; // "2"
     4'b0011: cathode = 8'b10110000; // "3"
     4'b0100: cathode = 8'b10011001; // "4"
     4'b0101: cathode = 8'b10010010; // "5"
     4'b0110: cathode = 8'b10000010; // "6"
     4'b0111: cathode = 8'b11111000; // "7"
     4'b1000: cathode = 8'b10000000; // "8"
     4'b1001: cathode = 8'b10010000; // "9"
     default: cathode = 8'b00101111; // "0"
  endcase
        end

        assign cath = cathode;
        assign anod = anode;

endmodule
```

Implementation Constraint File:

```
# LED
NET "LED_south" LOC = "U16";
NET "LED_north" LOC = "V16";
NET "LED_west" LOC = "U15";
NET "LED_east" LOC = "V15";


# 7-Segment Display
NET "cath<0>" LOC = "T17";
NET "cath<1>" LOC = "T18";
NET "cath<2>" LOC = "U17";
NET "cath<3>" LOC = "U18";
NET "cath<4>" LOC = "M14";
NET "cath<5>" LOC = "N14";
NET "cath<6>" LOC = "L14";
NET "cath<7>" LOC = "M13";

NET "clk" LOC = "V10";

NET "anod<0>"  LOC = "N16";
NET "anod<1>"  LOC = "N15";
NET "anod<2>"  LOC = "P18";
NET "anod<3>"  LOC = "P17";
```
