Course Code: CS223

Course Name: Digital Design

Section: 1

Lab 5

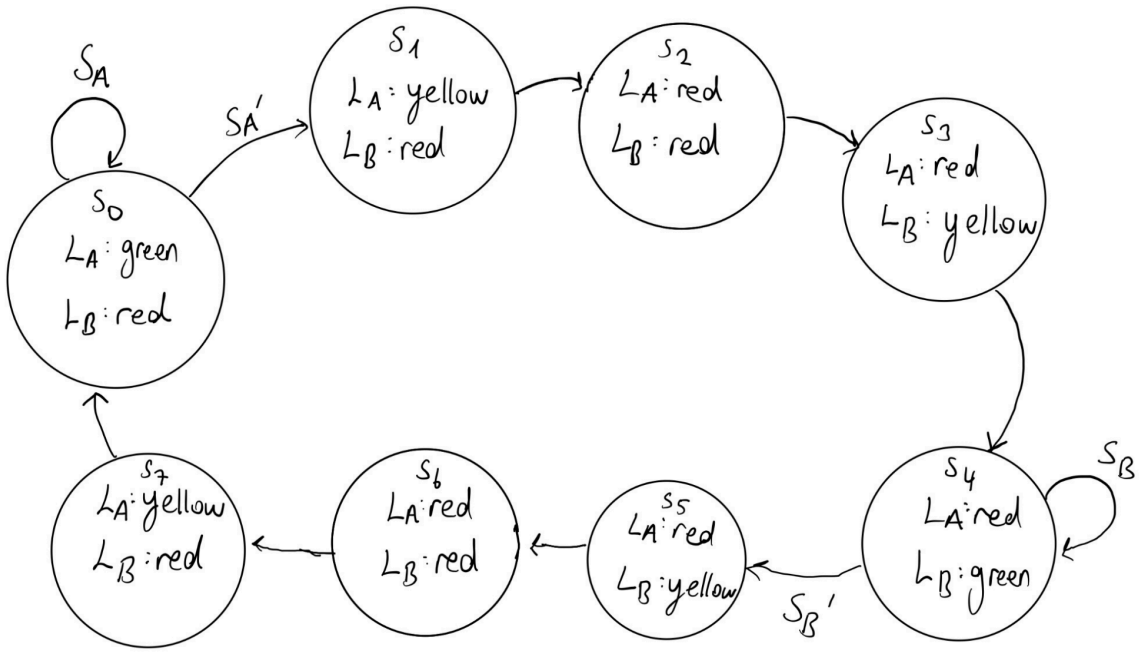
Mustafa Mert Gülhan

22201895

Date: 27.04.2024

1. Traffic Light System

Moore State Transition Diagram:



State Encodings:

State	Encoding
S0	000.
S1	001.
S2	010.
S3	011.
S4	100.
S5	101.
S6	110.
S7	111.

State Transition Table

Current State	Sa	Sb	Next State
S0	1.	X	S0
S0	0.	X	S1
S1	X	X	S2
S2	X	X	S3
S3	X	X	S4
S4	X	1.	S4
S4	X	0.	S5
S5	X	X	S6
S6	X	X	S7
S7	X	X	S0

Output Table:

Current State	La	Lb
S0	110	111
S1	100	111
S2	111	111
S3	111	100
S4	111	110
S5	111	100
S6	111	111
S7	111	100

Output Encodings:

Output	Encoding
Red	111.
Yellow	100.
Green	110.

Next State and Output Equations:

$L_A[2] = 1$

$L_A[1] = S[2]+S[1]+S[0]'$

$L_A[0] = (S[2]+S[1]+S[0])(S[2]+S[1]+S[0]')$

$L_B[2] = 1$

$L_B[1] = (S[2]+S[1]'+S[0]')$

$(S[2]'+S[1]+S[0]')(S[2]'+S[1]'+S[0]')$

$L_B[0] = S[3]'S[2]'S[1]' + S[3]'S[2]'S[1] +$

$S[3]'S[2]S[1]' + S[3]S[2]S[1]'$

$S2' = S[2]'S[1]S[0] + S[2]S[1]'S[0]'Sb +$
$S[2]S[1]'S[0]'Sb' + S[2]S[1]'S[0] +$
$S[2]S[1]S[0]'$

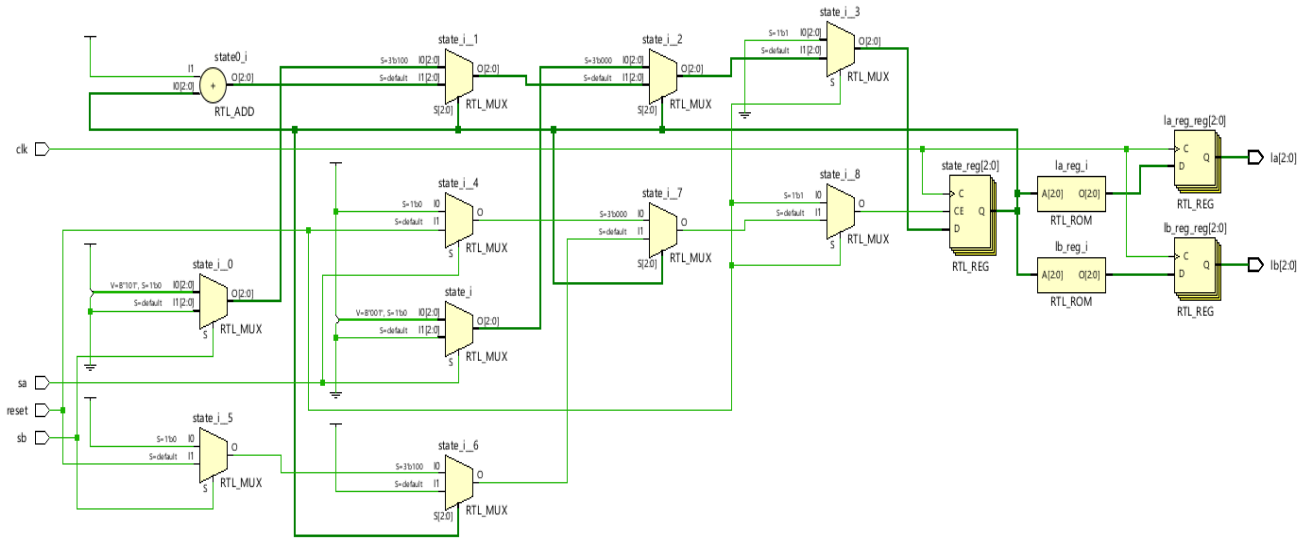
S1’ = S[2]’S[1]’S[0] + S[2]’S[1]S[0]’ + S[2]S[1]’S[0] + S[2]S[1]S[0]’

S0’ = S[2]’S[1]’S[0]’Sa’ + S[2]’S[1]S[0]’ + S[2]S[1]’S[0]’Sb’ + S[2]S[1]S[0]’

How Many Flip-Flops?

Because of 8 states $2^3 = 8$. 3 Flip Flops are enough.

FSM Schematic



SystemVerilog module

```
module tls(input logic clk, reset, sa, sb,
output logic [2:0]la,[2:0]lb );
```

```
parameter green = 3'b110;
parameter yellow = 3'b100;
parameter red = 3'b111;
logic [2:0] state = 3'b000;
logic [2:0]la_reg = green;
logic [2:0]lb_reg = red;

always @(posedge clk) begin
    if(reset) begin
        state = 3'b000;
    end
    else begin
        if(state == 3'b000) begin
            if(!sa)
                state <= state + 1'b1;
        end
        else if(state == 3'b100) begin
            if(!sb)
                state <= state + 1'b1;
        end
```

```
        else begin
            state <= state + 1'b1;
        end
    end
end

always @(posedge clk) begin
    case(state)
    3'b000: begin
        la_reg <= green;
        lb_reg <= red;
    end
    3'b001: begin
        la_reg <= yellow;
        lb_reg <= red;
    end
    3'b010: begin
        la_reg <= red;
        lb_reg <= red;
    end
    3'b011: begin
```

```
        la_reg <= red;
        lb_reg <= yellow;
    end
    3'b100: begin
        la_reg <= red;
        lb_reg <= green;
    end
    3'b101: begin
        la_reg <= red;
        lb_reg <= yellow;
    end
    3'b110: begin
        la_reg <= red;
        lb_reg <= red;
    end
    3'b111: begin
        la_reg <= yellow;
        lb_reg <= red;
    end
    default: begin
        la_reg <= green;
```

```
            lb_reg <= red;
        end
        endcase
    end

    assign la = la_reg;
    assign lb = lb_reg;

endmodule
```

2. Binary-Coded-Decimal Counter

Single Digit up and BCD SystemVerilog module

```
module singlebcd(input logic clk, en,
ld,reset, [3:0]d, output logic [3:0]q);
    reg [3:0] q_reg = 4'b0000;
    always @(posedge clk) begin
        if(reset) begin
            q_reg <= 0;
        end
        else if(ld) begin
```

```
            q_reg <= d;
        end
        else if(en) begin
            if(q_reg == 4'b1001) begin
                q_reg <= 4'b0000;
            end
            else begin
                q_reg <= q_reg + 1'b1;
            end
        end
    end
    assign q = q_reg;
endmodule
```

Single Digit up and BCD SystemVerilog Testbench

```
module singlebcd_tb();
    logic clk = 0;
    logic en, ld, reset;
    logic [3:0]d;
    logic [3:0]q;
```

```
    singlebcd dut(clk,en,ld,reset,d,q);

    always begin
        clk = ~clk; #20;
    end

    initial begin
        en = 0; ld = 0; #100;
        en = 1; d = 4'b0100; #100;
        ld = 1; #40 ld = 0;

    end
endmodule
```

Two Digit BCD counter SystemVerilog Module

```
module twodigitbcd(input logic clk, output
logic [7:0]q);
    logic enable2 = 0;
    logic [7:0] q_reg = 8'b0;
    singlebcd
last(clk,1,0,0,4'b0000,q_reg[3:0]);
```

```
    singlebcd
first(clk,enable2,0,0,4'b0000,q_reg[7:4]);
    always @(posedge clk) begin
        if(q_reg[3:0] == 4'b1000) begin
            enable2 <= 1;
        end
        else begin
            enable2 <= 0;
        end
    end

    assign q = q_reg;

endmodule
```

Two Digit BCD counter SystemVerilog Testbench

```
module twodigitbcd_tb();
    logic clk = 0;
    logic [7:0] q;
    twodigitbcd dut(clk,q);
```

```
    always begin
        clk = ~clk; #2;
    end
endmodule
```