CS223-003 (DIGITAL DESIGN)

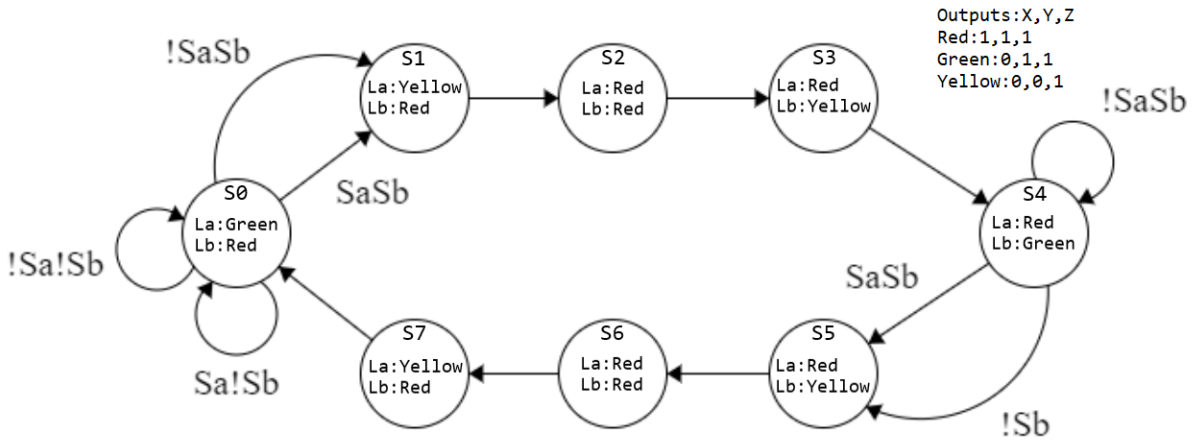
LAB 4

NAME: SERTAÇ DERYA

ID: 22003208

DATE: 28.11.2022

State Transition Diagram



State Encodings

S2	S1	S0	State
0	0	0	S0
0	0	1	S1
0	1	0	S2
0	1	1	S3
1	0	0	S4
1	0	1	S5
1	1	0	S6
1	1	1	S7

Output Table

For La

State	X	Y	Z
S0	0	1	1
S1	0	0	1
S2	1	1	1
S3	1	1	1
S4	1	1	1
S5	1	1	1
S6	1	1	1
S7	0	0	1

For Lb

State	X	Y	Z
S0	1	1	1
S1	1	1	1
S2	1	1	1
S3	0	0	1
S4	0	1	1
S5	0	0	1
S6	1	1	1
S7	1	1	1

State Transition Table

S2	S1	S0	SA	SB	S'2	S'1	S'0
0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	1
0	0	0	1	0	0	0	0
0	0	0	1	1	0	0	1
0	0	1	0	0	0	1	0
0	0	1	0	1	0	1	0
0	0	1	1	0	0	1	0
0	0	1	1	1	0	1	0
0	1	0	0	0	0	1	1
0	1	0	0	1	0	1	1
0	1	0	1	0	0	1	1
0	1	0	1	1	0	1	1
0	1	1	0	0	1	0	0
0	1	1	0	1	1	0	0
0	1	1	1	0	1	0	0
0	1	1	1	1	1	0	0
1	0	0	0	0	1	0	1
1	0	0	0	1	1	0	0
1	0	0	1	0	1	0	1
1	0	0	1	1	1	0	1
1	0	1	0	0	1	1	0
1	0	1	0	1	1	1	0
1	0	1	1	0	1	1	0
1	0	1	1	1	1	1	0
1	1	0	0	0	1	1	1
1	1	0	0	1	1	1	1
1	1	0	1	0	1	1	1
1	1	0	1	1	1	1	1
1	1	1	0	0	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	0	0	0	0
1	1	1	1	1	0	0	0

Equations

Next State:

$$S_2' = \bar{S_2}S_1S_0 + S_2\bar{S_0} + S_2\bar{S_1}$$

$$S_1' = \bar{S_2}\bar{S_1}S_0 + S_1\bar{S_0} + S_2\bar{S_1}S_0$$

$$S_0' = \bar{S_2}\bar{S_1}\bar{S_0}S_b + S_1\bar{S_0} + S_2\bar{S_0}S_a + S_2\bar{S_0}\overline{S_b}$$

Output:

$$xLa = S_2S_1 + S_2\bar{S_1} + S_2\bar{S_0}$$

$$yLa = \bar{S_0} + \bar{S_2}S_1 + S_2\bar{S_1}$$

$$zLa = 1$$

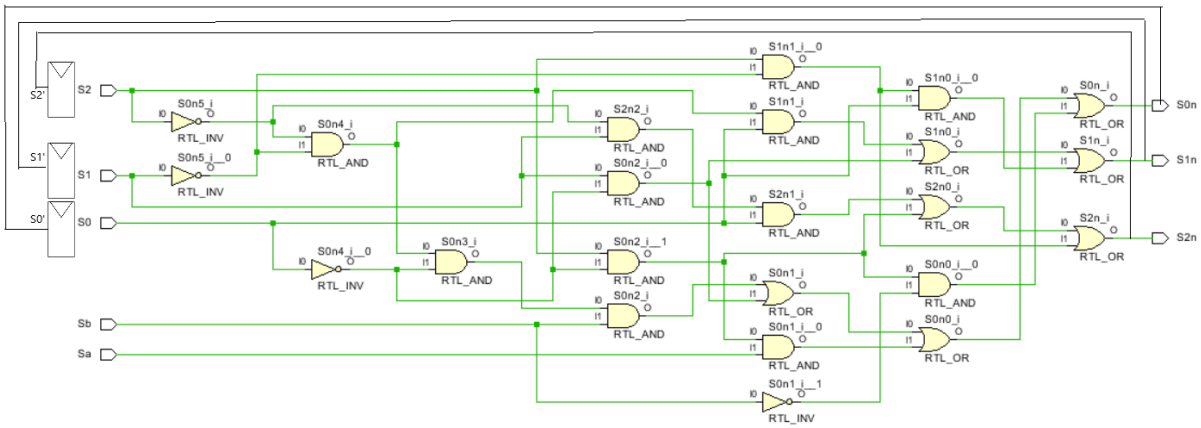
$$xLb = \bar{S_2}\bar{S_1} + S_2S_1 + S_1\bar{S_0}$$

$$yLb = \bar{S_0} + \bar{S_2}\bar{S_1} + S_2S_1$$

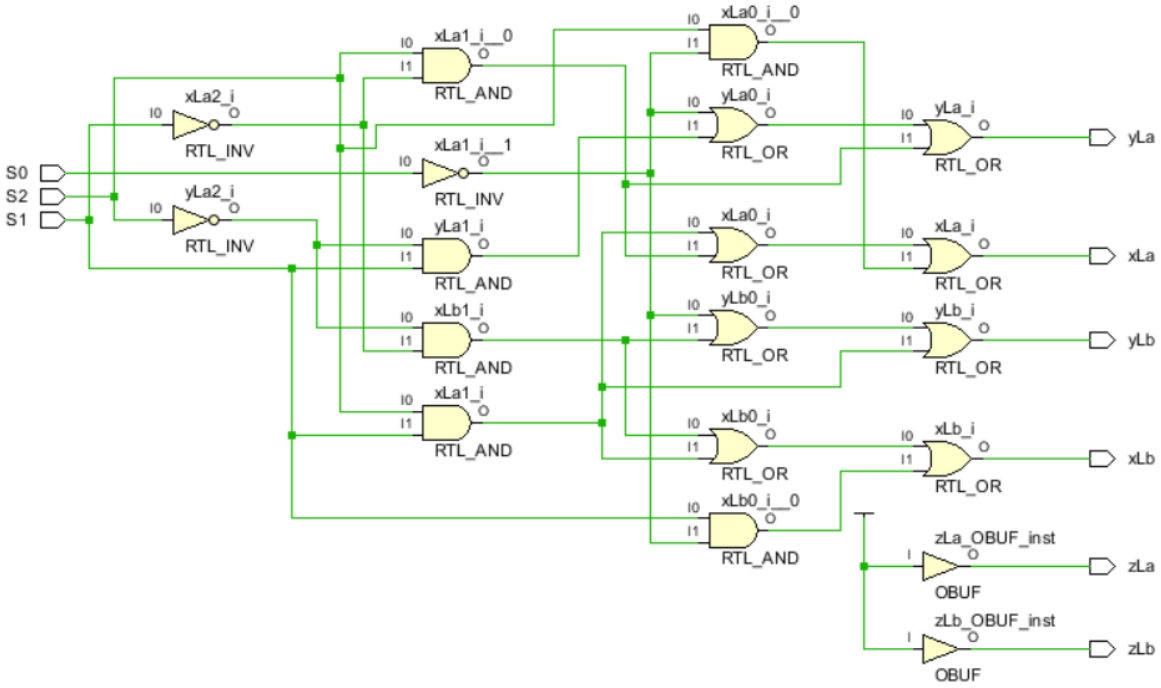
$$zLb = 1$$

b) Schematic for FSM

Next State Schematic:

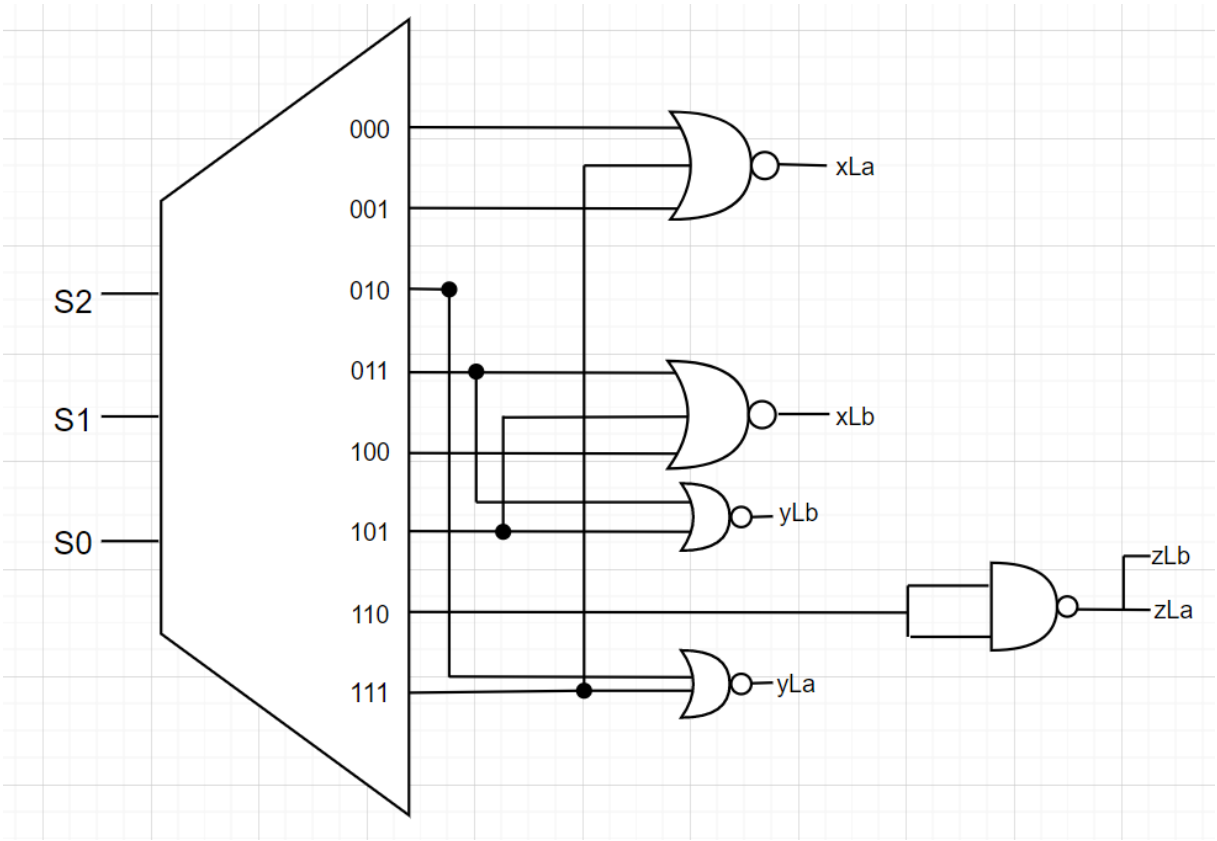


Output Schematic:



c) We need 3 flip flops to design this circuit.

d) Decoder Output Implementation



e) SystemVerilog Files

Module:

```
module TrafficController(
    input logic clk,
    input logic reset,
    input logic Sa,
    input logic Sb,
    output logic xLa,
    output logic yLa,
    output logic zLa,
    output logic xLb,
    output logic yLb,
    output logic zLb);

    typedef enum logic[2:0] {S0,S1,S2,S3,S4,S5,S6,S7} statetype;
    statetype state, nextstate;
    //state register
    always_ff @(posedge clk, posedge reset)
        if(reset) state <= S0;
        else      state <= nextstate;
    //nextstate comb logic
    always_comb
        case(state)
            S0: if(Sb) nextstate=S1;
                else   nextstate=S0;
            S1: nextstate=S2;
            S2: nextstate=S3;
            S3: nextstate=S4;
            S4: if(!Sa&Sb)  nextstate=S4;
                else        nextstate=S5;
            S5: nextstate=S6;
            S6: nextstate=S7;
            S7: nextstate=S0;
            default: nextstate=S0;
        endcase
    //output logic
    assign zLa = 1;
    assign zLb = 1;
    assign yLa =(!state[0] | !state[2] & state[1] | state[2] & !state[1]);
    assign yLb =(!state[0] | !state[2] & !state[1] | state[2] & state[1]);
    assign xLa =(state[2] & state[1] | state[2] & !state[1] | state[2] &
!state[0]);
    assign xLb =(!state[2] & !state[1] | state[2] & state[1] | state[1] &
!state[0]);
endmodule
```

TestBench:

```
module Traffic_Sim();
    logic clk = 0;
    logic reset = 0;
    logic Sa = 0;
    logic Sb = 0;
    logic xLa = 0;
    logic yLa = 0;
    logic zLa = 0;
    logic xLb = 0;
    logic yLb = 0;
    logic zLb = 0;

    TrafficController dut(clk,reset,Sa,Sb,xLa,yLa,zLa,xLb,yLb,zLb);
    always
        begin
            clk = 1;#3; clk = 0;#3;
        end

    initial begin
        Sb = 1; #3; reset = 1; #3; reset = 0; #3;
        Sb = 0; Sa = 1; #3;
        Sb = 1; Sa = 1; #3;
        Sb = 0; Sa = 0; #3;
        Sb = 0; Sa = 1; #3;
        Sb = 1; Sa = 1; #3;
        Sb = 1; Sa = 0; #3;
    end
endmodule
```