DIGITAL DESIGN

CS223

Section: 001

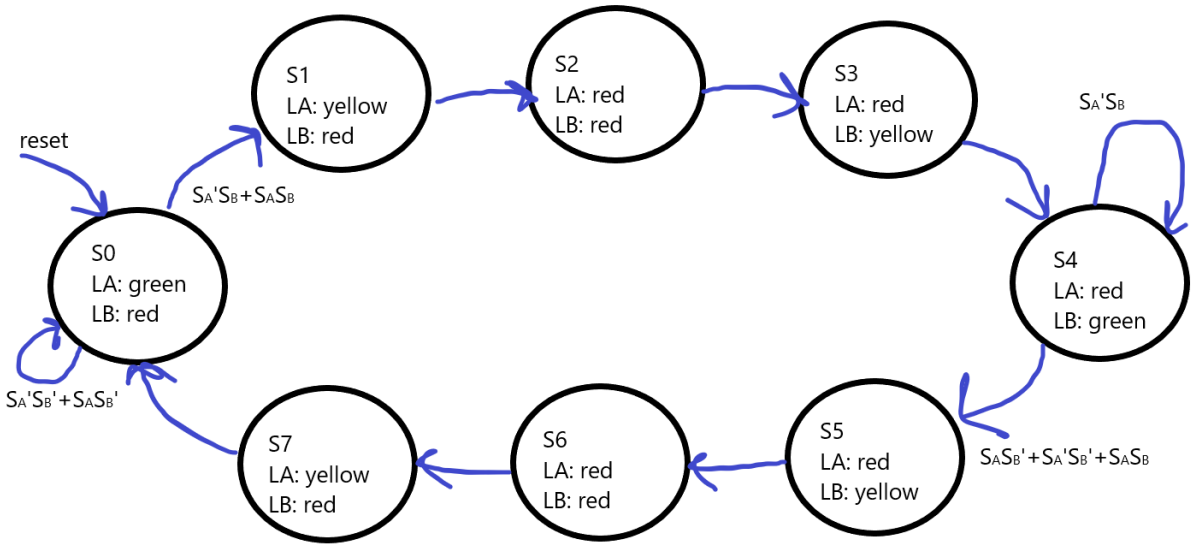
Lab3 Report

Name: Tolga Han Arslan

ID: 22003061

Date: 28.11.2022

a) Your improved Moore machine state transition diagram, state encodings, state transition table, output table, next state, and output equations.



State and Output Encodings:

State	Encoding $S_{2:0}$
S0	000
S1	001
S2	010
S3	011
S4	100
S5	101
S6	110
S7	111

Output	Encoding $L_{2:0}$
Red	111
Green	011
Yellow	001

State Transition Table:

Current State			**Inputs**		**Next State**		
S_2	S_1	S_0	S_A	S_B	S_2’	S_1’	S_0’
0	0	0	X	0	0	0	0
0	0	0	X	1	0	0	1
0	0	1	X	X	0	1	0
0	1	0	X	X	0	1	1
0	1	1	X	X	1	0	0
1	0	0	0	1	1	0	0
1	0	0	X	0	1	0	1
1	0	0	1	1	1	0	1
1	0	1	X	X	1	1	0
1	1	0	X	X	1	1	1
1	1	1	X	X	0	0	0

Output table:

Current State			**Output**					
S_2	S_1	S_0	L_{A2}	L_{A1}	L_{A0}	L_{B2}	L_{B1}	L_{B0}
0	0	0	0	1	1	1	1	1
0	0	1	0	0	1	1	1	1
0	1	0	1	1	1	1	1	1
0	1	1	1	1	1	0	1	1
1	0	0	1	1	1	0	1	1
1	0	1	1	1	1	0	0	1
1	1	0	1	1	1	1	1	1
1	1	1	0	0	1	1	1	1

Next State: (On the right hand side " ' " means not)

$S_2' = S_2S_1' + S_2'S_1S_0 + S_2S_1S_0'$

$S_1' = S_1 \oplus S_0$

$S_0' = S_1S_0' + S_1'S_0'(S_2 \oplus S_B + (S_2S_AS_B))$

Output:

$L_{A0} = 1$

$L_{A1} = (S_2 \oplus S_1) + S_2S_1S_0' + S_2'S_1'S_0'$

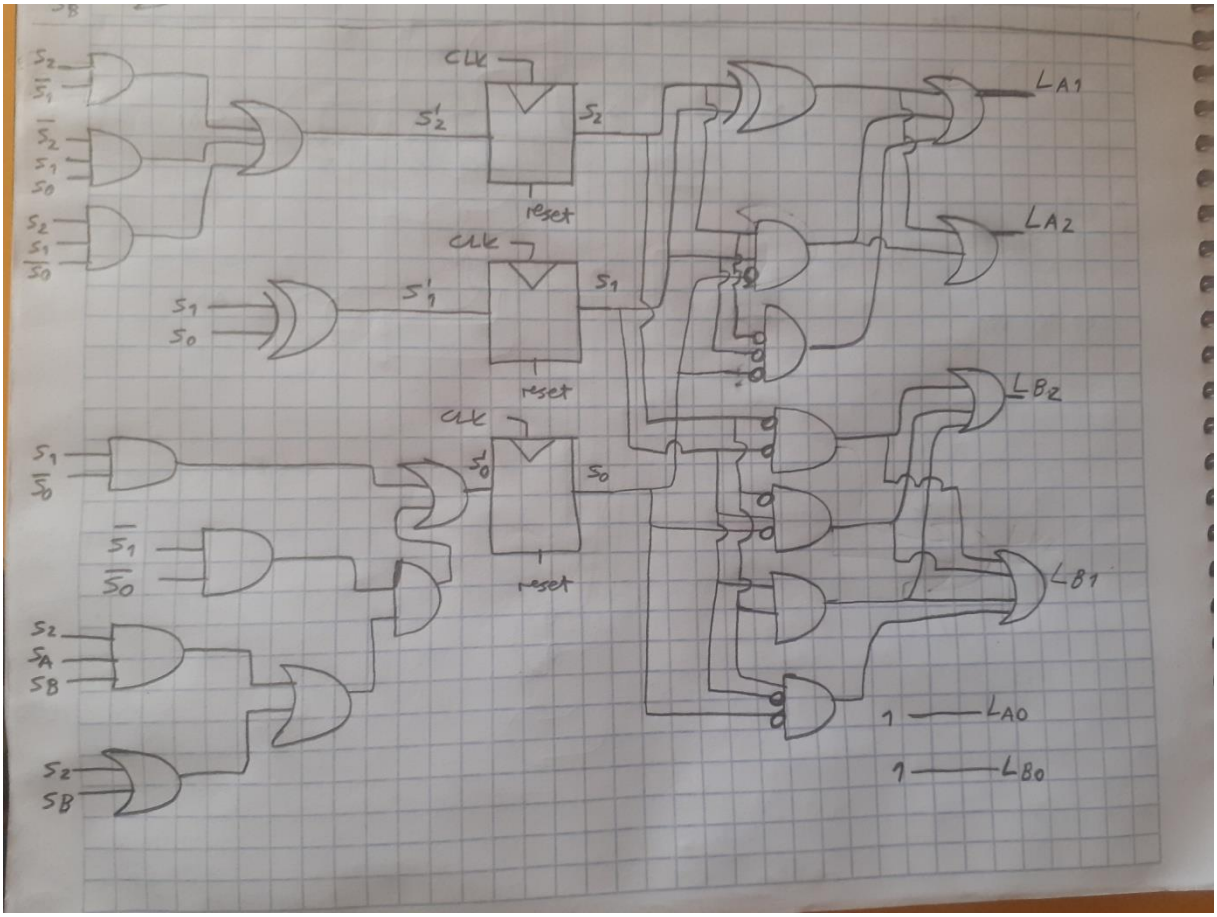
$L_{A2} = (S_2 \oplus S_1) + S_2S_1S_0'$

$L_{B0} = 1$

$L_{B1} = S_2'S_1' + S_2'S_1S_0' + S_2S_1'S_0' + S_2S_1$

$L_{B2} = S_2'S_1' + S_2'S_1S_0' + S_2S_1$

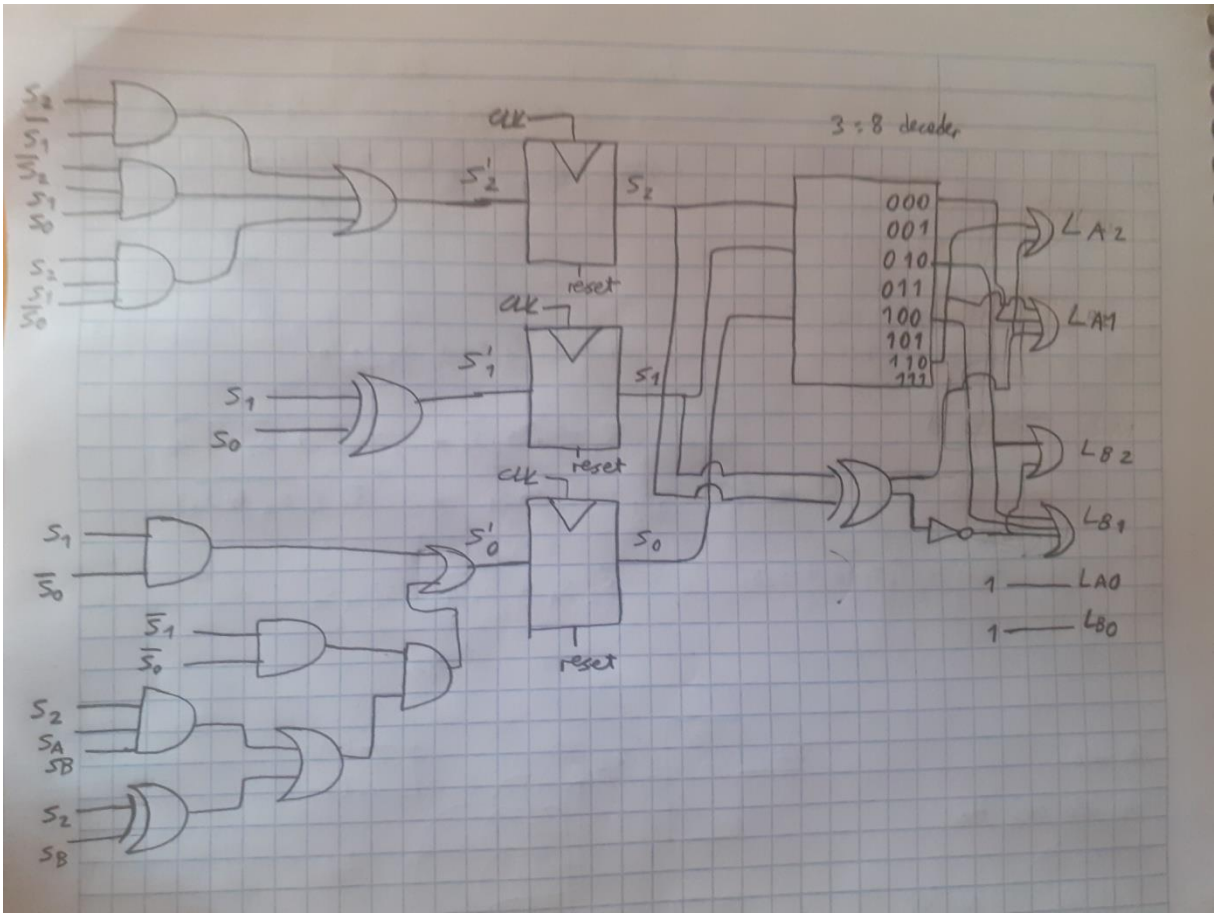
b) Your Finite State Machine schematic.



c) How many flip-flops do you need to implement this problem?

3 flip-flops

d) Redesign of your outputs using decoders.



e) Your System Verilog design code and testbench.

```
module fsm(
    input logic clk,
    input logic reset,
    input logic SA, SB,
    output logic [2:0] LA, [2:0] LB
    );
    typedef enum logic [2:0]{S0,S1,S2,S3,S4,S5,S6,S7} statetype;
```

```
    statetype state, nextstate; // dont put [2:0] here
//state register
    always_ff @(posedge clk, posedge reset)
        if(reset) state <= S0;
        else state <= nextstate;
    //next state logic
    always_comb
        case(state)
        S0: if(~SA&SB|SA&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;
        endcase
assign LA[0] = 1;
     assign LA[1] = (state[2]^state[1]) | (state[2]&state[1]&~state[0]) |
(~state[2]&~state[1]&~state[0]);
     assign LA[2] = (state[2]^state[1]) | (state[2]&state[1]&~state[0]);
     assign LB[0] = 1;
```

```
    assign LB[1] = (~state[2]&~state[1]) | (~state[2]&state[1]&~state[0]) |
(state[2]&~state[1]&~state[0]) | (state[2]&state[1]);

    assign LB[2] = (~state[2]&~state[1]) | (~state[2]&state[1]&~state[0]) |
(state[2]&state[1]);

endmodule

module fsm_tb();

logic clk, reset, SA, SB;

logic [2:0] LA, LB;

fsm dut(clk, reset, SA, SB, LA,LB);

   initial

      begin

      reset <= 1; #100;

      reset <= 0;

      SA = 0; SB = 0; #100;

      SA = 0; SB = 1; #100;

      SA = 0; SB = 0; #100;

      SA = 1; SB = 0; #100;

      SA = 0; SB = 0; #100;

      SA = 1; SB = 1; #100;

      reset <=1;

      SA = 0; SB = 0; #100;

      SA = 0; SB = 1; #100;

      SA = 0; SB = 0; #100;

      SA = 1; SB = 0; #100;

      SA = 0; SB = 0; #100;
```

```
        SA = 1; SB = 1; #100;
       end

        always
        begin
        clk<=1; #10;
        clk<=0; #10;
        end

endmodule
```