CS 223

Digital Design

Section 5 Lab 5

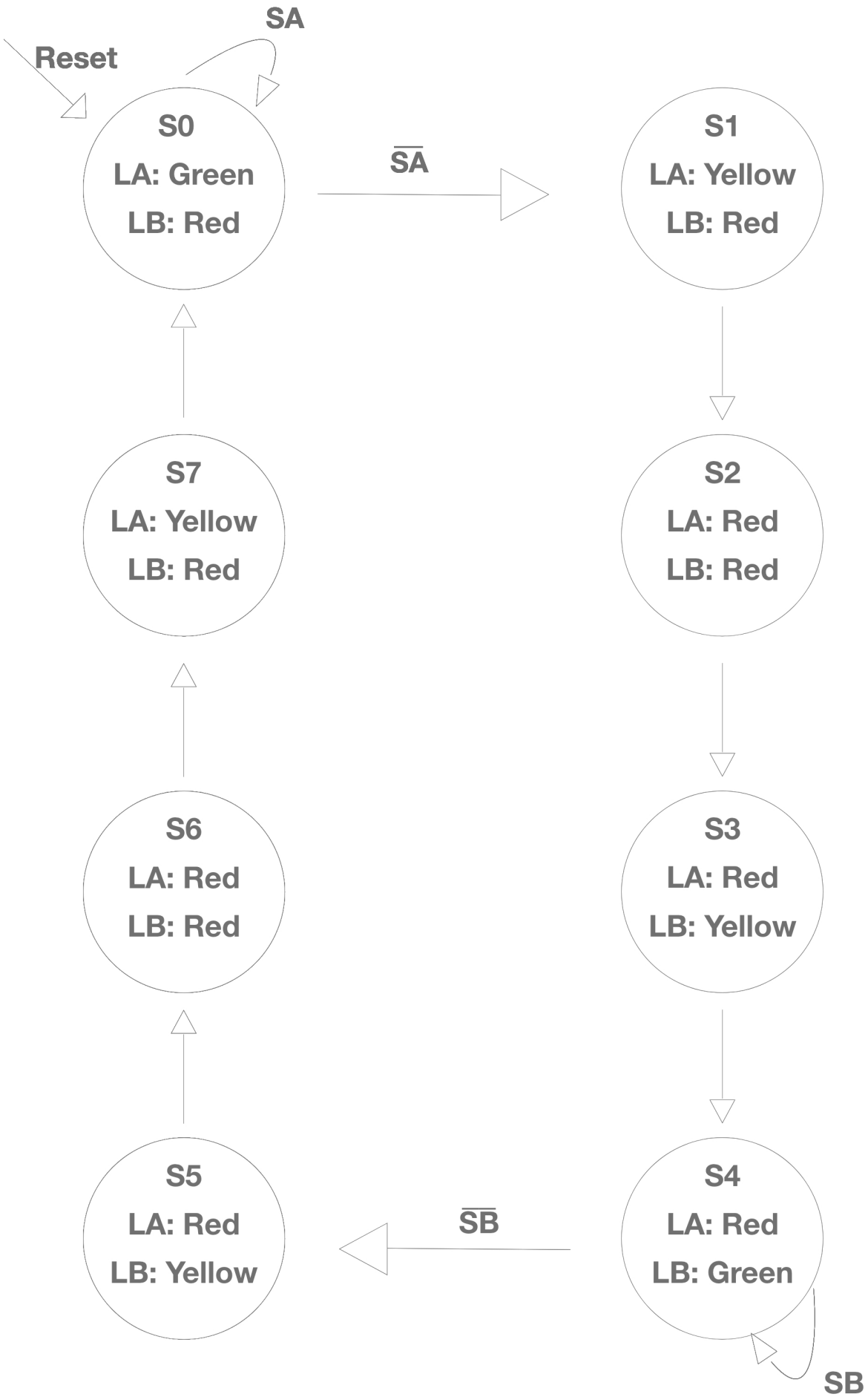
Ömer Oktay Gültekin

21901413

29/11/2021

Part A

STATE TRANSITION DIAGRAM



STATE ENCODING

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

STATE TRANSITION TABLE

CURRENT STATE	INPUTS		NEXT STATE
S	*SA*	*SB*	S'
S0	0	X	S1
S0	1	X	S0
S1	X	X	S2
S2	X	X	S3
S3	X	X	S4
S4	X	0	S5
S4	X	1	S4
S5	X	X	S6
S6	X	X	S7
S7	X	X	S0

OUTPUT ENCODING

OUTPUT	ENCODING $L_{1:0}$
Green	10
Yellow	01
Red	11

STATE TRANSITION TABLE WITH BINARY ENCODINGS

CURRENT STATE			INPUTS		NEXT STATE		
S2	*S1*	*S0*	*SA*	*SB*	*S2'*	*S1'*	*S0'*
0	0	0	0	X	0	0	1
0	0	0	1	X	0	0	0
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	X	0	1	0	1
1	0	0	X	1	1	0	0
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			OUTPUTS			
S2	*S1*	*S0*	*LA1*	*LA0*	*LB1*	*LB0*
0	0	0	1	0	1	1
0	0	1	0	1	1	1
0	1	0	1	1	1	1
0	1	1	1	1	0	1
1	0	0	1	1	1	0
1	0	1	1	1	0	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1

NEXT STATE AND OUTPUT BINARY EQUATIONS

$$S0' = \overline{S2} \cdot \overline{S1} \cdot \overline{S0} \cdot \overline{SA} + S1 \cdot \overline{S0} + S2 \cdot \overline{S1} \cdot \overline{S0} \cdot \overline{SB}$$

$$S1' = S1 \oplus S0$$

$$S2' = \overline{S2} \cdot S1 \cdot S0 + S2 \cdot \overline{S1} + S2 \cdot (S1 \oplus S0)$$

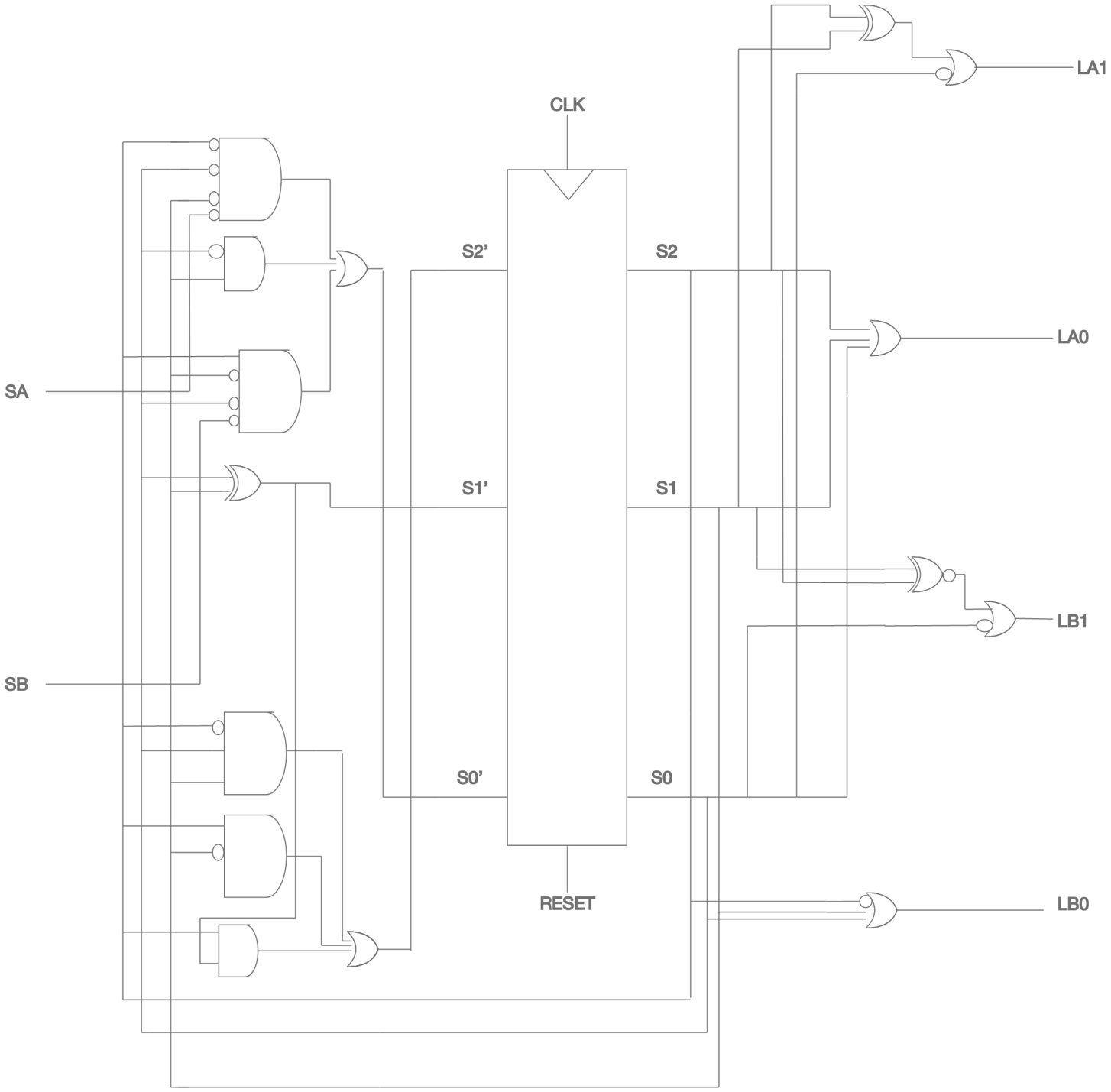
$$LA0 = S0 + S1 + S2$$

$$LA1 = \overline{S0} + (S1 \oplus S2)$$

$$LB0 = S0 + S1 + \overline{S2}$$

$$LB1 = \overline{S0} + \overline{(S1 \oplus S2)}$$

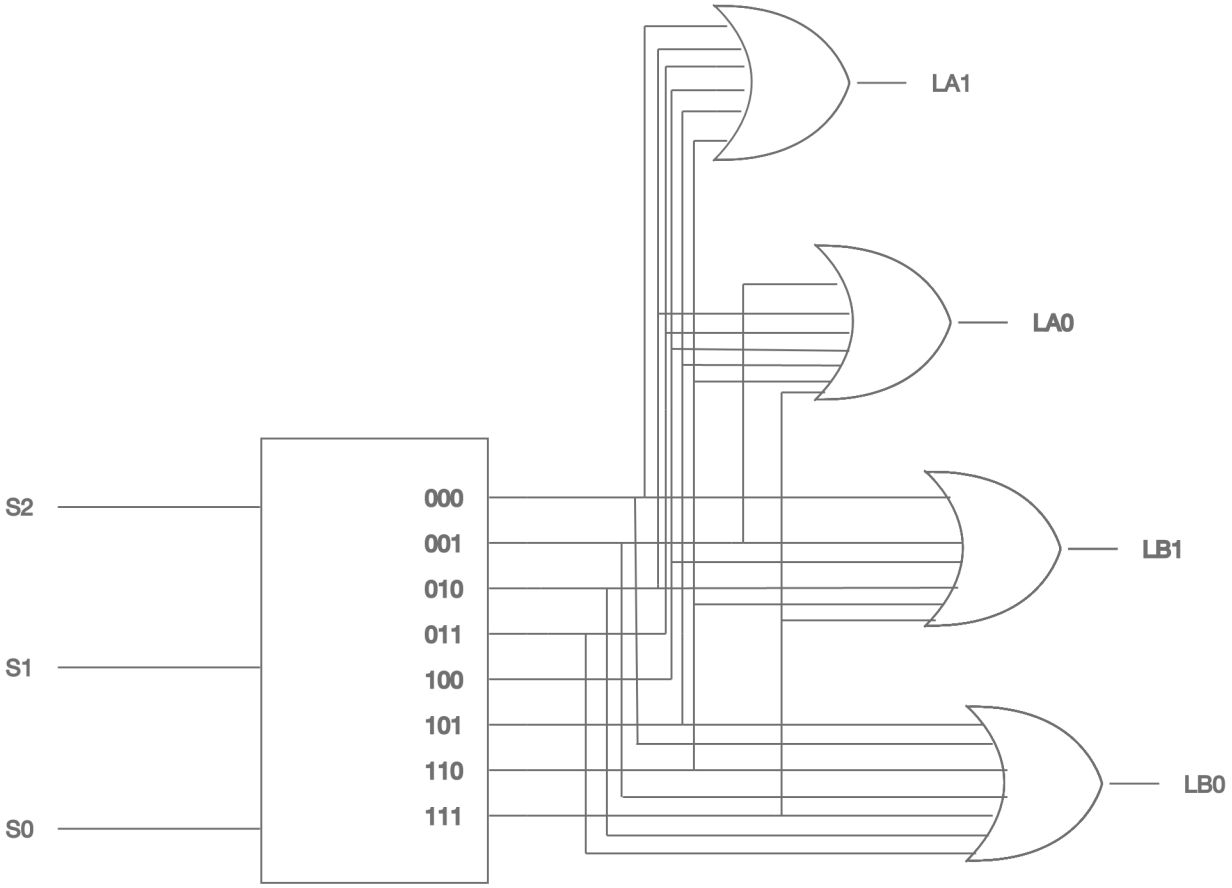
FINITE STATE MACHINE SCHEMATİC



Part B

3 Flip-Flops is needed to implement this problem.

Part C



SystemVerilog Design Code

```
`timescale 1ns / 1ps

module clockdivider( input logic clk,
                     output logic clk_div);
    // this module is for FPGA's that has 100 Mhz clock speed
    logic [31:0] cnt;
    always@(posedge clk)
    begin
       if(cnt == 149999999) // 1.5 s is past for 100Mhz clock
           cnt <= 32'b0;
       else
           cnt <= cnt + 1;
    end

    always@(posedge clk)
    begin
       if(cnt == 149999999) // at every 1.5 s the new clock will change the edge
           clk_div <= ~clk_div;
       else
           clk_div <= clk_div;
    end
endmodule
```

```
`timescale 1ns / 1ps

module trafficlightsFSM(
        input logic clk,
        input logic reset,
        input logic SA,
        input logic SB,
        output logic [2:0] LA,
        output logic [2:0] LB
    );

    typedef enum logic [2:0] {S0,S1, S2, S3, S4, S5, S6, S7} statetype;
    statetype state, nextstate;
    logic clk_div;

    // if simulation will be used comment the next line and uncomment the
    // next one, otherwise use clockdivider module to create new clk
    //clockdivider clk_divider(clk, clk_div);
    assign clk_div = clk;

    //state register
    always_ff@(posedge clk_div)
        if(reset) state <= S0;
        else      state <= nextstate;
```

```
    //next state logic
    always_comb
        case(state)
            S0: if(SA) nextstate <= S0;
                else  nextstate <= S1;
            S1: nextstate <= S2;
            S2: nextstate <= S3;
            S3: nextstate <= S4;
            S4: if(SB) nextstate <= S4;
                else  nextstate <= S5;
            S5: nextstate <= S6;
            S6: nextstate <= S7;
            S7: nextstate <= S0;
        endcase

    //output logic
    always_comb
        case(state)
            S0: begin
                LA <= 3'b110;
                LB <= 3'b111;
            end
            S1: begin
                LA <= 3'b100;
                LB <= 3'b111;
            end
            S2: begin
                LA <= 3'b111;
                LB <= 3'b111;
            end
            S3: begin
                LA <= 3'b111;
                LB <= 3'b100;
            end
            S4: begin
                LA <= 3'b111;
                LB <= 3'b110;
            end
            S5: begin
                LA <= 3'b111;
                LB <= 3'b100;
            end
            S6: begin
                LA = 3'b111;
                LB = 3'b111;
            end
            S7: begin
                LA = 3'b100;
                LB = 3'b111;
            end
        endcase
endmodule
```

SystemVerilog TestBench

```
`timescale 1ns / 1ps

module trafficlights_testbench();
    logic clk, reset, SA, SB;
    logic [2:0] LA;
    logic [2:0] LB;
    trafficlightsFSM tlFSM(clk, reset, SA, SB, LA, LB);
    initial begin
        clk <= 0;
        reset = 1; SA = 1; SB = 1; #4; // 4 nanosecond is period of the clock
        reset = 0; #4;
        // do the following 4 times to see the outputs of the input independent
        // state transitions with every input combinations and prove that
        // the transitions are really input independent
        for(int i = 0; i < 4; i++)begin
            SB = 0; #4;
            SB = 1; SA = 0; #4;
            {SA, SB} = i; #8;
            SB = 1; #4;
            SA = ~SA; #4;
            SB = 0; #4;
            {SA, SB} = i; #8;
            SA = 1; SB = 1; #4;
        end
         //do everything again with reset active too
         reset = 1; #4;
         for(int i = 0; i < 4; i++)begin
             SB = 0; #4;
             SB = 1; SA = 0; #4;
             {SA, SB} = i; #8;
             SB = 1; #4;
             SA = ~SA; #4;
             SB = 0; #4;
             {SA, SB} = i; #8;
             SA = 1; SB = 1; #4;
         end
    end
    //2ns clock
    always #2 clk <= ~clk;
endmodule
```