Course Code: CS223

Course Name: Digital Design **Section:** 4

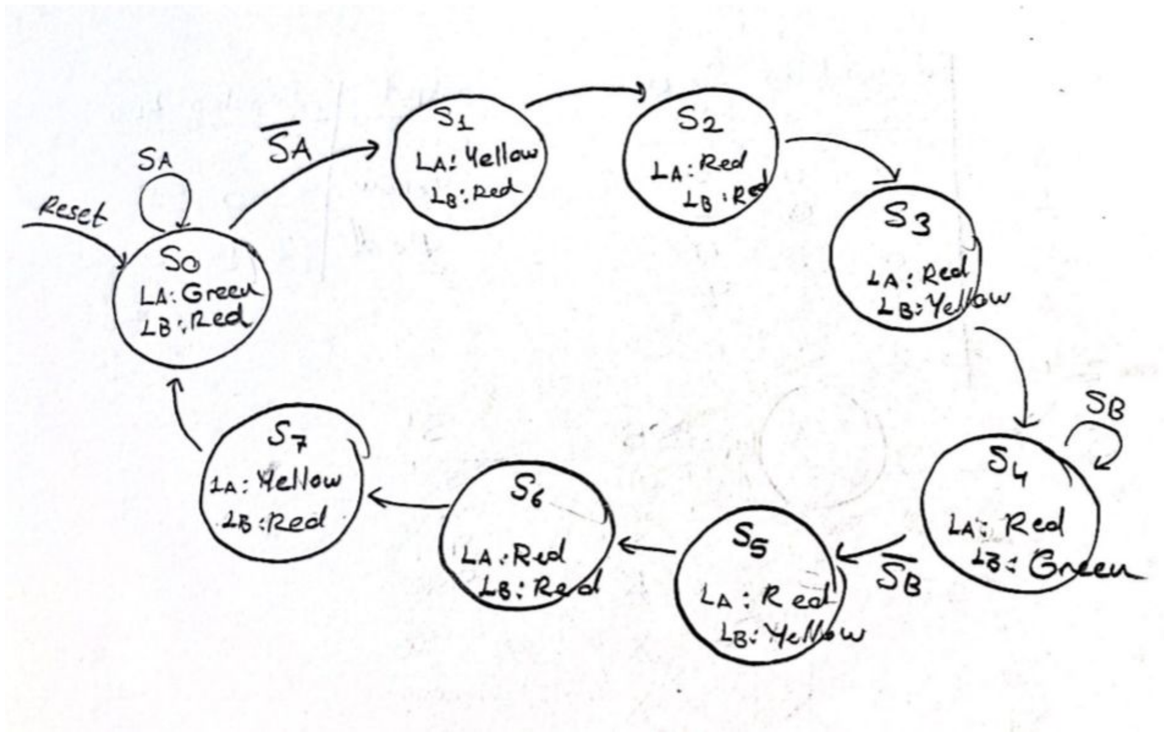
Lab 4

Name: Berk Saltuk Yılmaz **ID:** 21903419

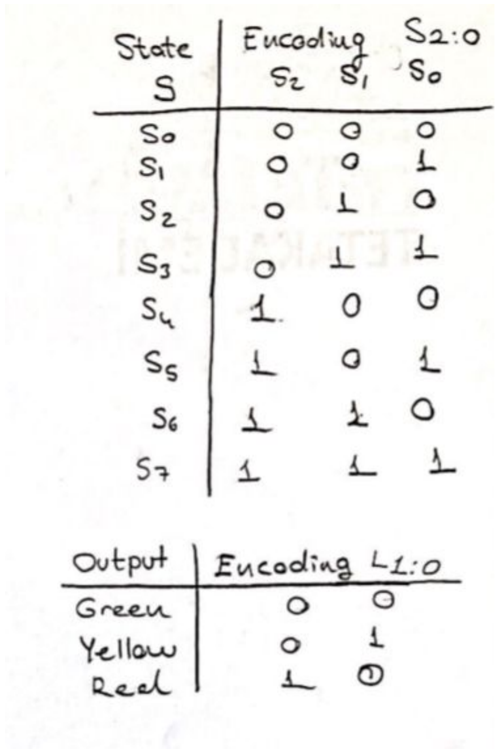
Date: 27.11.2020

Part (a):

Transition Diagram:



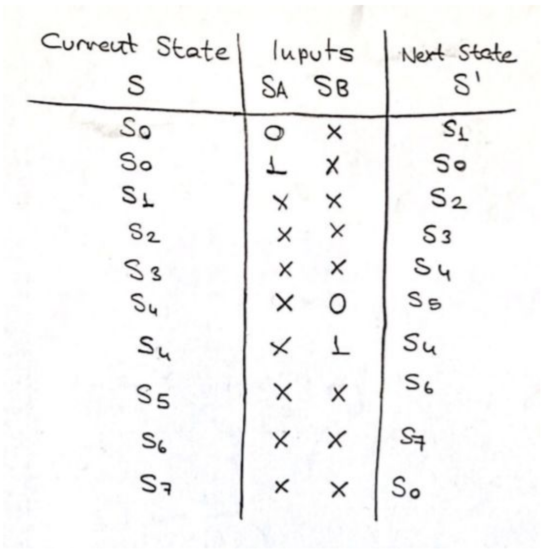
State Encodings:



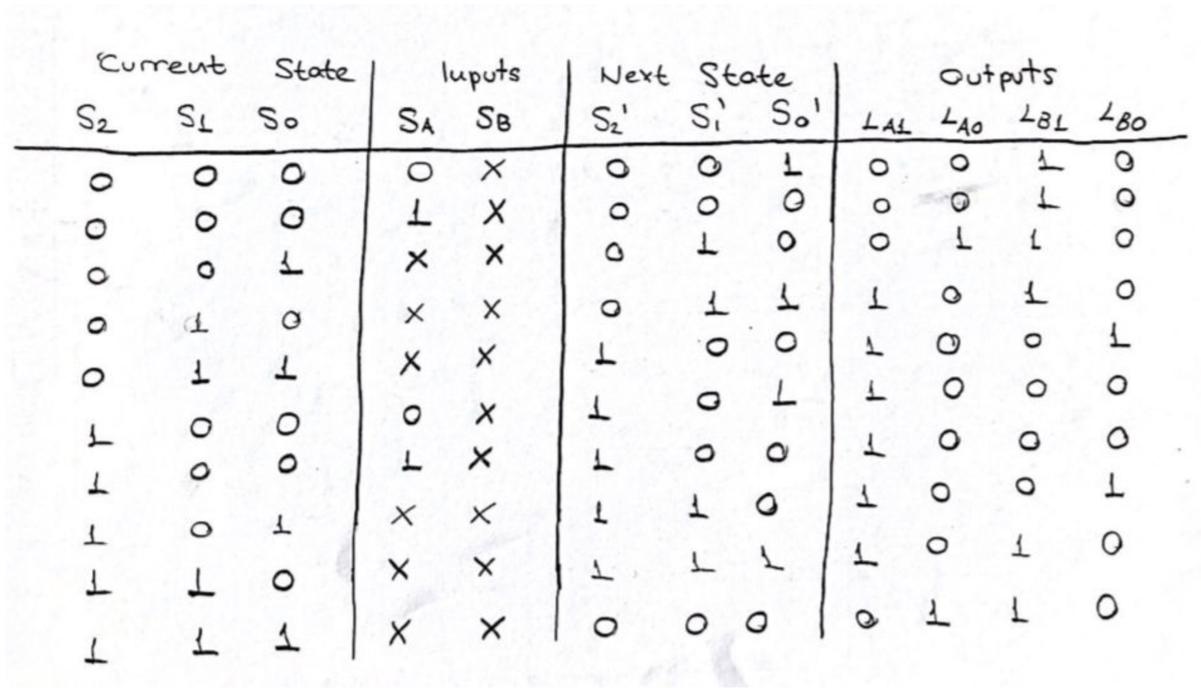
State S	Encoding $S_{2:0}$ S_2	S_1	S_0
S_0	0	0	0
S_1	0	0	1
S_2	0	1	0
S_3	0	1	1
S_4	1	0	0
S_5	1	0	1
S_6	1	1	0
S_7	1	1	1

Output	Encoding $L_{1:0}$	
Green	0	0
Yellow	0	1
Red	1	0

State Transition and Output Tables:



Current State	Inputs		Next State
S	S_A	S_B	S'
S_0	0	X	S_1
S_0	1	X	S_0
S_1	X	X	S_2
S_2	X	X	S_3
S_3	X	X	S_4
S_4	X	0	S_5
S_4	X	1	S_4
S_5	X	X	S_6
S_6	X	X	S_7
S_7	X	X	S_0



Current State			Inputs		Next State			Outputs			
S_2	S_1	S_0	S_A	S_B	S_2'	S_1'	S_0'	L_{A1}	L_{A0}	L_{B1}	L_{B0}
0	0	0	0	X	0	0	1	0	0	1	0
0	0	0	1	X	0	0	0	0	0	1	0
0	0	1	X	X	0	1	0	0	1	1	0
0	1	0	X	X	0	1	1	1	0	1	0
0	1	1	X	X	1	0	0	1	0	0	1
1	0	0	0	X	1	0	1	1	0	0	0
1	0	0	1	X	1	0	0	1	0	0	1
1	0	1	X	X	1	1	0	1	0	1	0
1	1	0	X	X	1	1	1	1	0	1	0
1	1	1	X	X	0	0	0	0	1	1	0

Boolean Equations:

$$L_{A1} = S_2 \cdot \overline{S_1} + S_2 \cdot \overline{S_0} + \overline{S_2} S_1$$

$$L_{A0} = \overline{S_2}\,\overline{S_1} S_0 + S_2 S_1 S_0$$

$$L_{B1} = S_2 S_1 + \overline{S_1}\,\overline{S_2} + \overline{S_2}\,\overline{S_0}$$

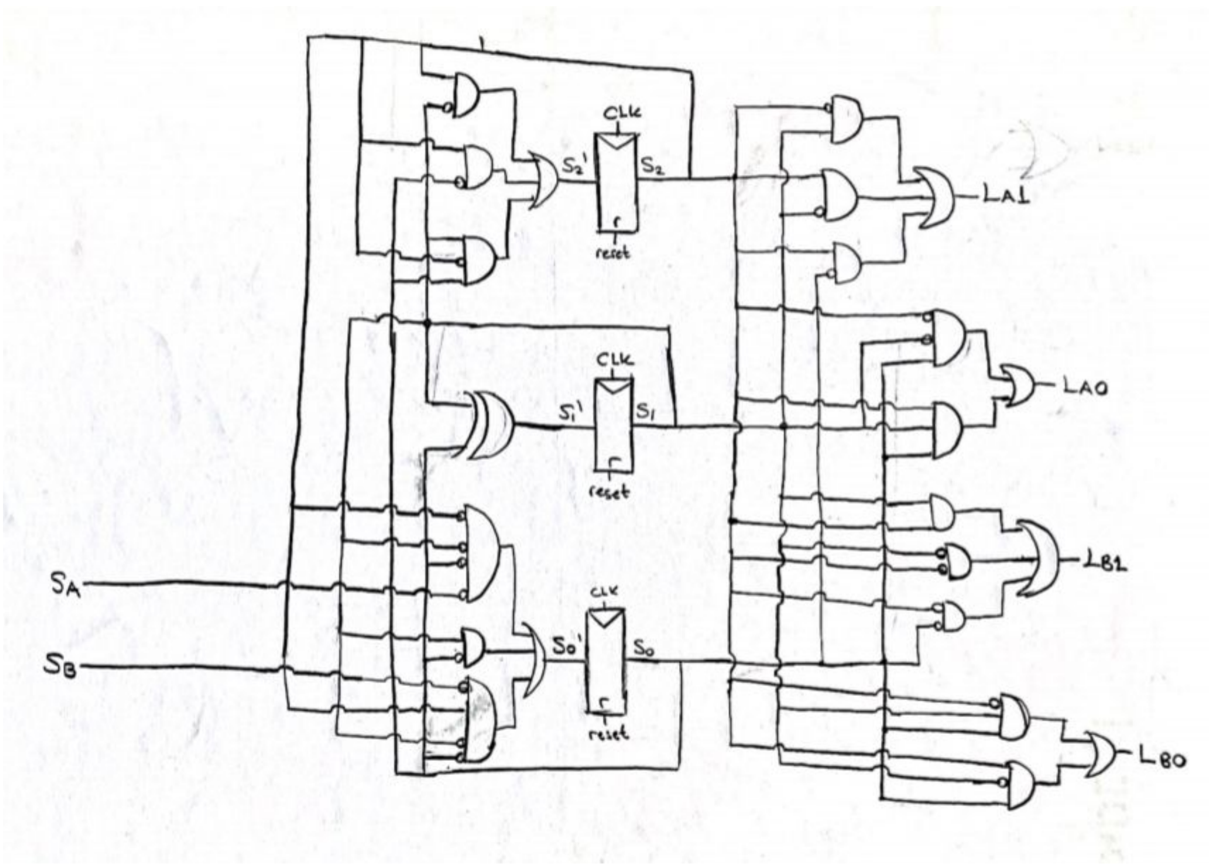
$$L_{B0} = \overline{S_2} S_1 S_0 + S_2 \overline{S_1} S_0$$

$$S_2' = S_2 \cdot \overline{S_1} + S_2 \overline{S_0} + \overline{S_2} S_1 S_0$$

$$S_1' = S_1 \overline{S_0} + \overline{S_1} S_0 = S_1 \oplus S_0$$

$$S_0' = \overline{S_2}\,\overline{S_1}\,\overline{S_0}\,\overline{S_A} + S_1 \overline{S_0} + S_2 \overline{S_1}\,\overline{S_0}\,\overline{S_B}$$

Schematic:



Part (b):

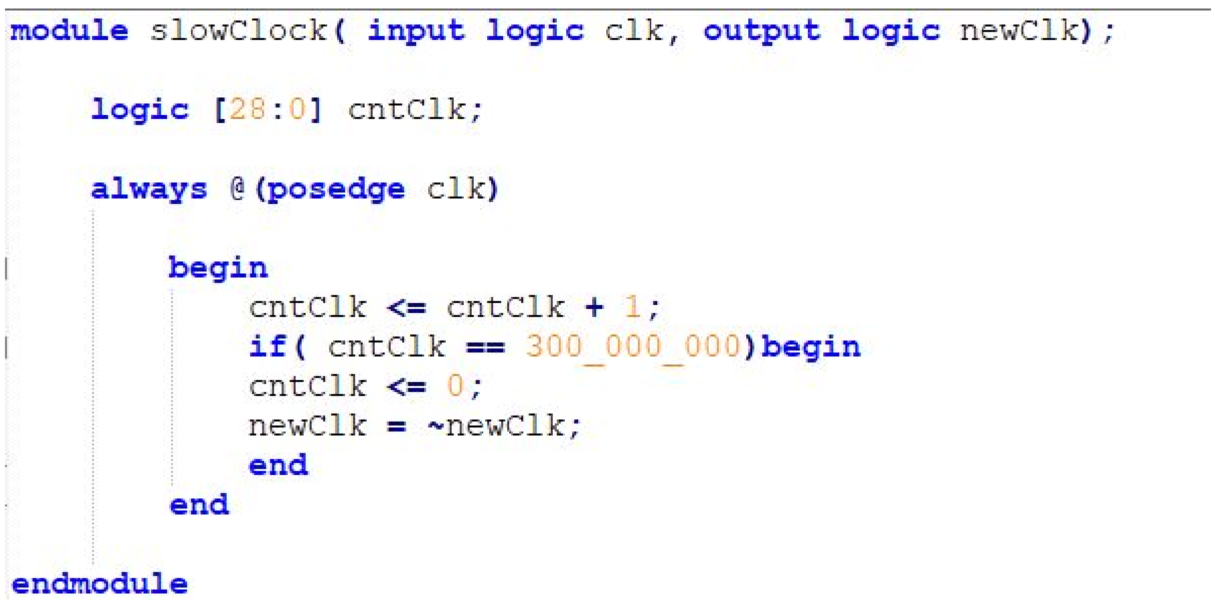
As n flip flops can represent 2^n states and this Finite State Machine consists of 8 states, I will need 3 flip flops to implement this problem.

Part (c):

With quick research, I found that 100MHz = 100000000 Hz. Thus, to have a ⅓ Hz frequency, the clock must be 300000000 times slower. Clock counter must count to the 300000000 before HIGH. I will need 29 bits to store this value as

$$\log_2 300000000 \cong 28.1$$

SystemVerilog for this slow clock:



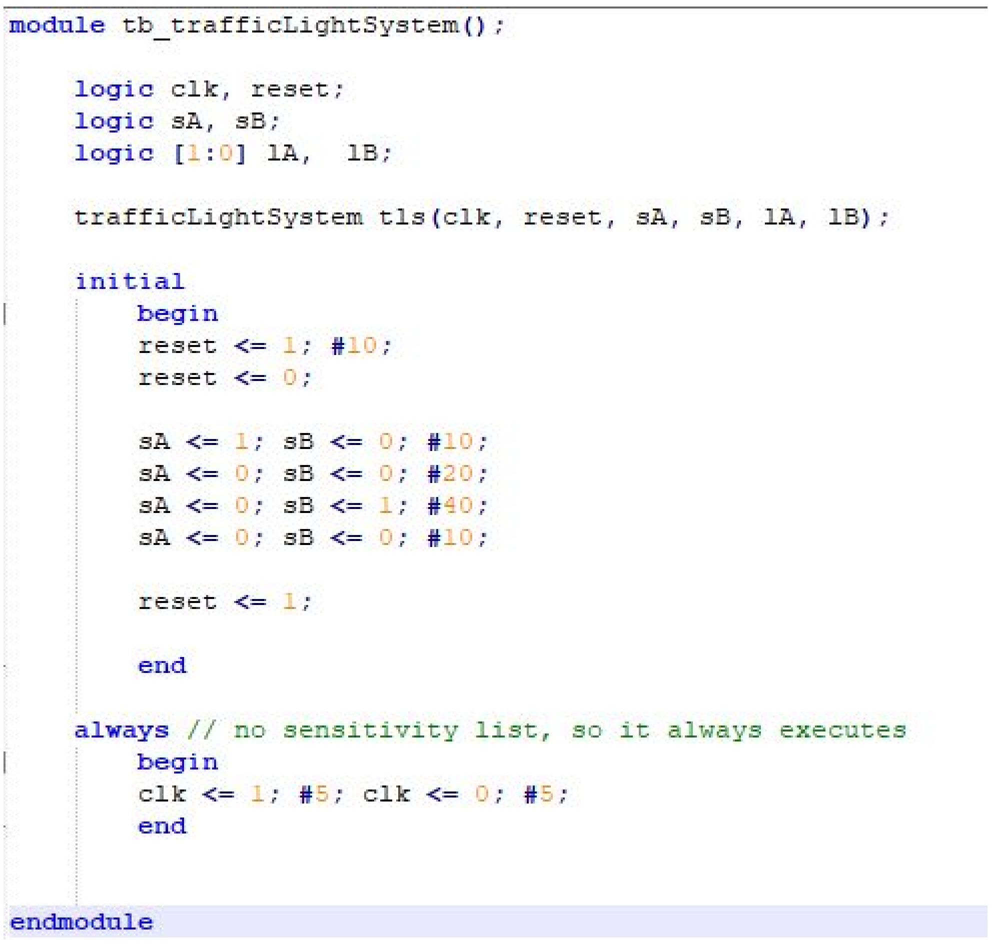
```
module slowClock( input logic clk, output logic newClk);

    logic [28:0] cntClk;

    always @(posedge clk)

        begin
            cntClk <= cntClk + 1;
            if( cntClk == 300_000_000)begin
            cntClk <= 0;
            newClk = ~newClk;
            end
        end

endmodule
```

Part (d):

SystemVerilog code for traffic light system:

```
module trafficLightSystem( input logic clk, reset, sA, sB,
                           output logic [1:0] lA, [1:0] lB);

    typedef enum logic [2:0] {s0, s1, s2, s3, s4, s5, s6, s7} statetype;

    statetype [2:0] state, nextstate;

    parameter green = 2'b00;
    parameter yellow = 2'b01;
    parameter red = 2'b10;

    always_ff @(posedge clk, posedge reset) // register

        if( reset )
            state <= s0;
        else
            state <= nextstate;

    always_comb // next state
    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;
        default: nextstate = s0;
    endcase
    always_comb // output
    case( state)
        s0:
        begin
        lA = green;
        lB = red;
        end

        s1:
        begin
        lA = yellow;
        lB = red;
        end

        s2:
        begin
        lA = red;
        lB = red;
        end

        s3:
        begin
        lA = red;
        lB = yellow;
        end

        s4:
        begin
        lA = red;
        lB = green;
        end

        s5:
        begin
        lA = red;
        lB = yellow;
        end

        s6:
        begin
        lA = red;
        lB = red;
        end

        s7:
        begin
        lA = yellow;
        lB = red;
        end
    endcase
endmodule
```

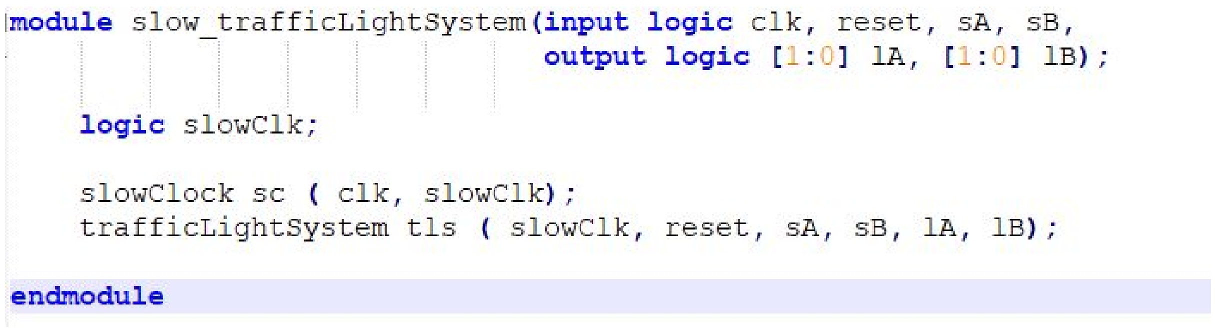
Testbench for it:



```
module tb_trafficLightSystem();

    logic clk, reset;
    logic sA, sB;
    logic [1:0] lA,  lB;

    trafficLightSystem tls(clk, reset, sA, sB, lA, lB);

    initial
        begin
        reset <= 1; #10;
        reset <= 0;

        sA <= 1; sB <= 0; #10;
        sA <= 0; sB <= 0; #20;
        sA <= 0; sB <= 1; #40;
        sA <= 0; sB <= 0; #10;

        reset <= 1;

        end

    always // no sensitivity list, so it always executes
        begin
        clk <= 1; #5; clk <= 0; #5;
        end

endmodule
```

Part (e):

Combined SystemVerilog Code:



```
module slow_trafficLightSystem(input logic clk, reset, sA, sB,
                               output logic [1:0] lA, [1:0] lB);

    logic slowClk;

    slowClock sc ( clk, slowClk);
    trafficLightSystem tls ( slowClk, reset, sA, sB, lA, lB);

endmodule
```