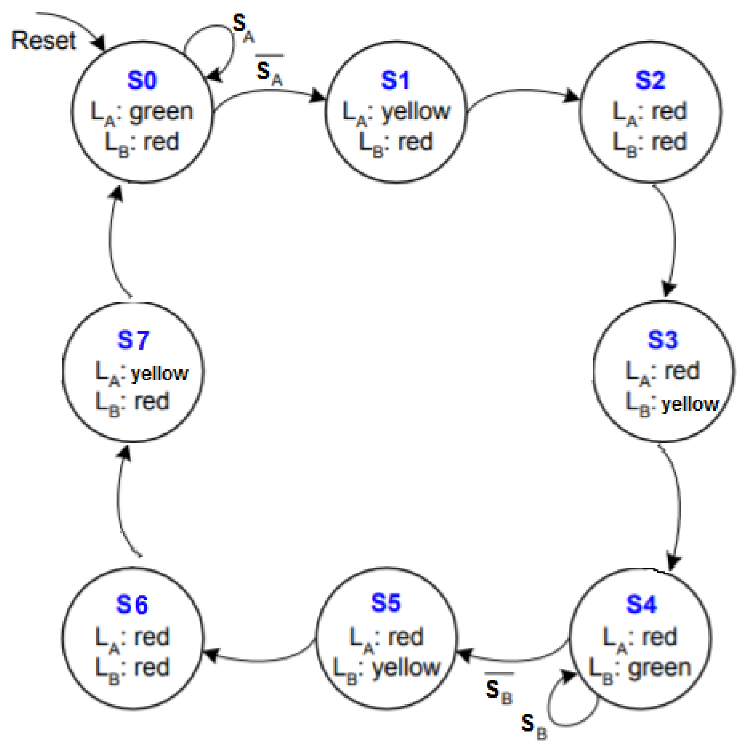
CS 223
Digital Design
Section 01
Laboratory Assignment 4
Traffic Light System

Arda Önal

21903350

28.11.2020

a)



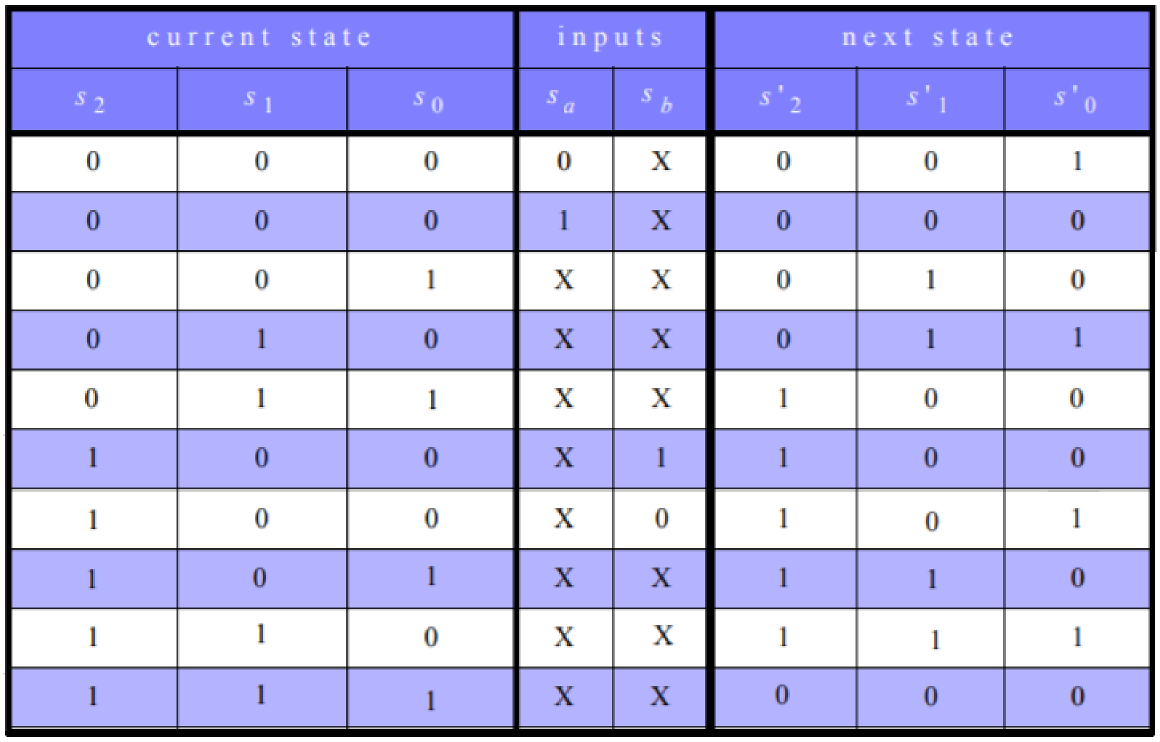
Moore state machine transition diagram

state	encoding $s_{1:0}$
S0	000
S1	001
S2	010
S3	011
S4	100
S5	101
S6	110
S7	111

State encodings

Output	Encoding $L_{2:0}$
green	011
yellow	001
red	111

Output encodings



current state			inputs		next state		
s_2	s_1	s_0	s_a	s_b	s'_2	s'_1	s'_0
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	1	1	0	0
1	0	0	X	0	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

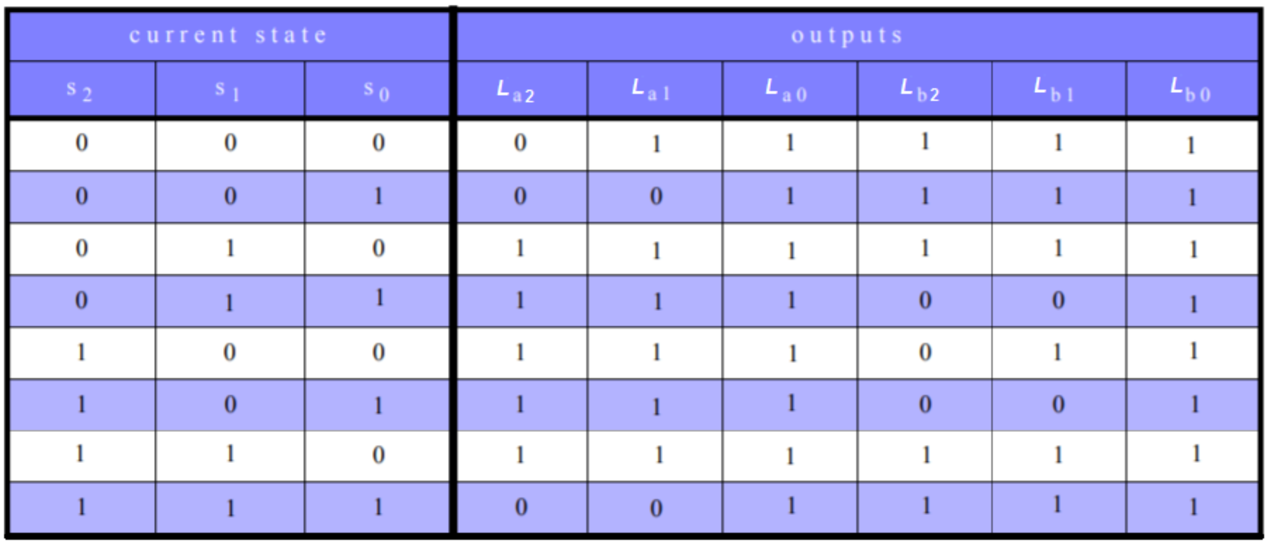
State transition table

Next state equations:

$S'_2 = \overline{S_2}\ S_1\ S_0 + S_2(\overline{S_1}+\overline{S_0})$

$S'_1 = S_1 \oplus S_0$

$S'_0 = \overline{S_0}\ (S_1 + \overline{S_2}\,\overline{S_A} + S_2\overline{S_B})$



current state			outputs					
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	0	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

Output table

Output equations:

$L_{a2} = (S_1 \oplus S_2) + S_1\overline{S_0}$

$L_{a1} = (S_1 \oplus S_2) + \overline{S_0}$

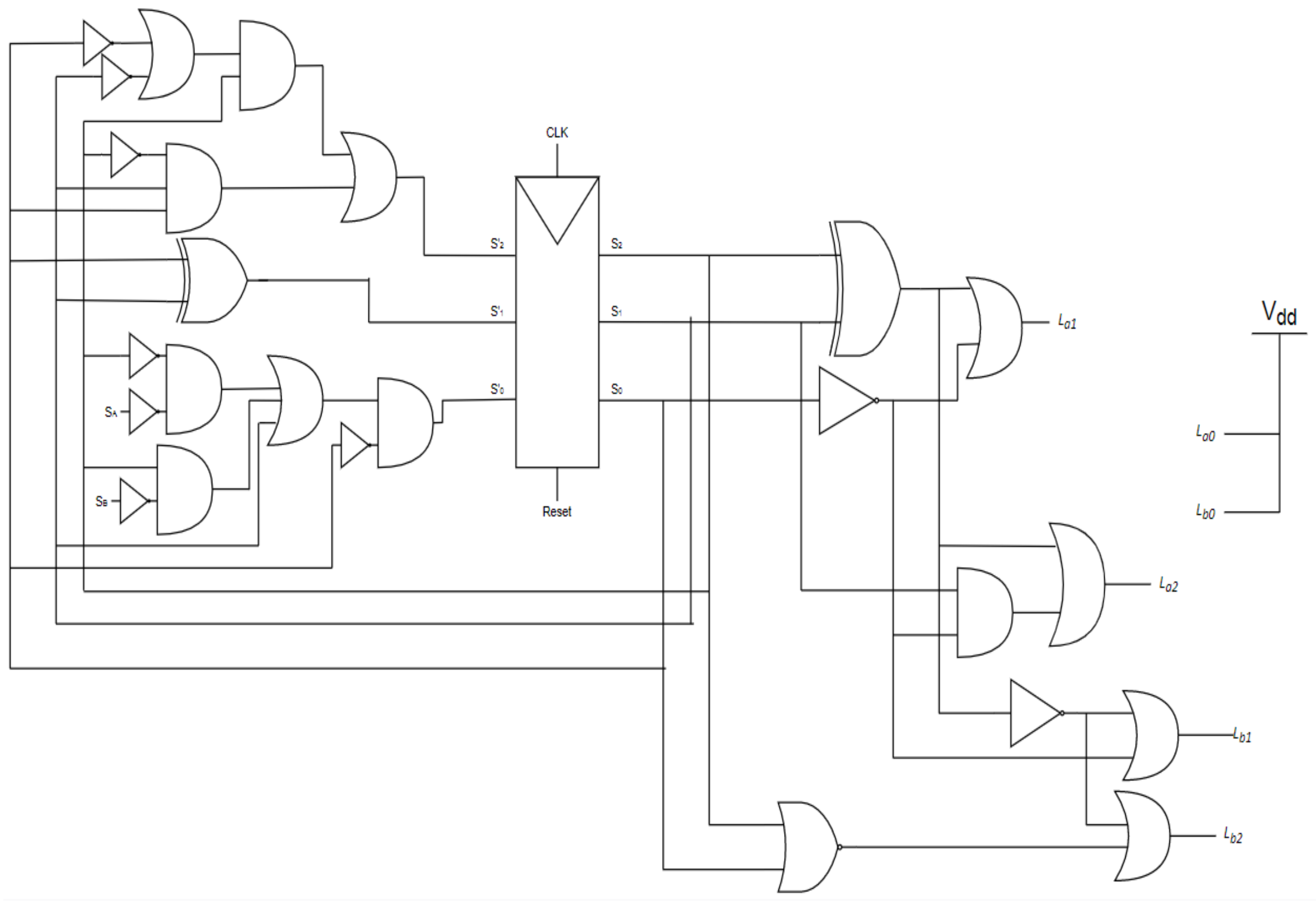
$L_{a0} = 1$

$L_{b2} = (\overline{S_2 \oplus S_1}) + \overline{S_2}\ \overline{S_0}$

$L_{b1} = (\overline{S_2 \oplus S_1}) + \overline{S_0}$

$L_{b0} = 1$

Finite State Machine schematic



b) We need three flip-flops to implement this problem. This because our states have three bit encodings which means that our register will need 3 flip-flops to work.

c) We need to count 15 x 10^7 rising edges to obtain 1/3 Hz frequency. This is because BASYS3 has a clock speed of 100MHz and obtaining 1/3 Hz frequency means counting 3 x 10^8 transitions. Since half of these transitions are rising edges, our answer is 3 x 10^8 / 2 = 15 x 10^7.

SystemVerilog code that will count original clock cycles output HIGH in every 3 seconds:

```
module clock_counter(input clock, input reset, output logic new_clock);


    localparam risingEdgeNumber = 150000000; // The number we found in partC of lab04


    logic [31:0] count;
    always @ (posedge(clock), posedge(reset))
    begin
        if (reset == 1'b1)
            count <= 32'b0;
        else if (count == risingEdgeNumber - 1)
            count <= 32'b0;
        else
            count <= count + 1;
    end

    always @ (posedge(clock), posedge(reset))
    begin
        if (reset == 1'b1)
            new_clock <= 1'b0;
        else if (count == risingEdgeNumber - 1)
            new_clock <= ~new_clock;
        else
            new_clock <= new_clock;
    end
endmodule
```

d) SystemVerilog code for the traffic light system:

```
module traffic_light_system(input logic clk,reset,sa,sb,output logic [2:0] La, Lb);
    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;
    // 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;
        default: nextstate = S0;
    endcase

    // output logic
    logic n0;
    logic n1;
    xor(n0,state[1],state[2]);
    xnor(n1,state[1],state[2]);

    assign La[2] = n0 | (state[1] & ~state[0]);
    assign La[1] = n0 | ~state[0];
    assign La[0] = 1;

    assign Lb[2] = n1 | (~state[2] & ~state[0]);
    assign Lb[1] = n1 | ~state[0];
    assign Lb[0] = 1;
endmodule
```

Testbench for the traffic light system:

```
module testbench_traffic();
   logic clk,reset,sa,sb;
   logic [2:0] La, Lb;

   // instantiate device under test
   traffic_light_system dut(clk,reset,sa,sb,La,Lb);

   // clock
   always
      begin
      clk = 1; #5;
      clk = 0; #5;
      end

   // apply inputs one at a time
   initial begin
      reset = 1; sa= 1; sb = 0; #10;
      reset = 0; #100;
      sb = 1; #100;
      sa = 0; #100;
      sa = 0; sb = 0; #100;
   end
endmodule
```

e) SystemVerilog code for combined new clock generator in part c and traffic light system in part d:

```
module combined_traffic_light(input logic clk,reset,sa,sb,output logic [2:0] La, Lb);
   logic new_clock;
   clock_counter newclock(clk,reset,new_clock);
   traffic_light_system improved_fsm(new_clock,reset,sa,sb,La,Lb);
endmodule
```