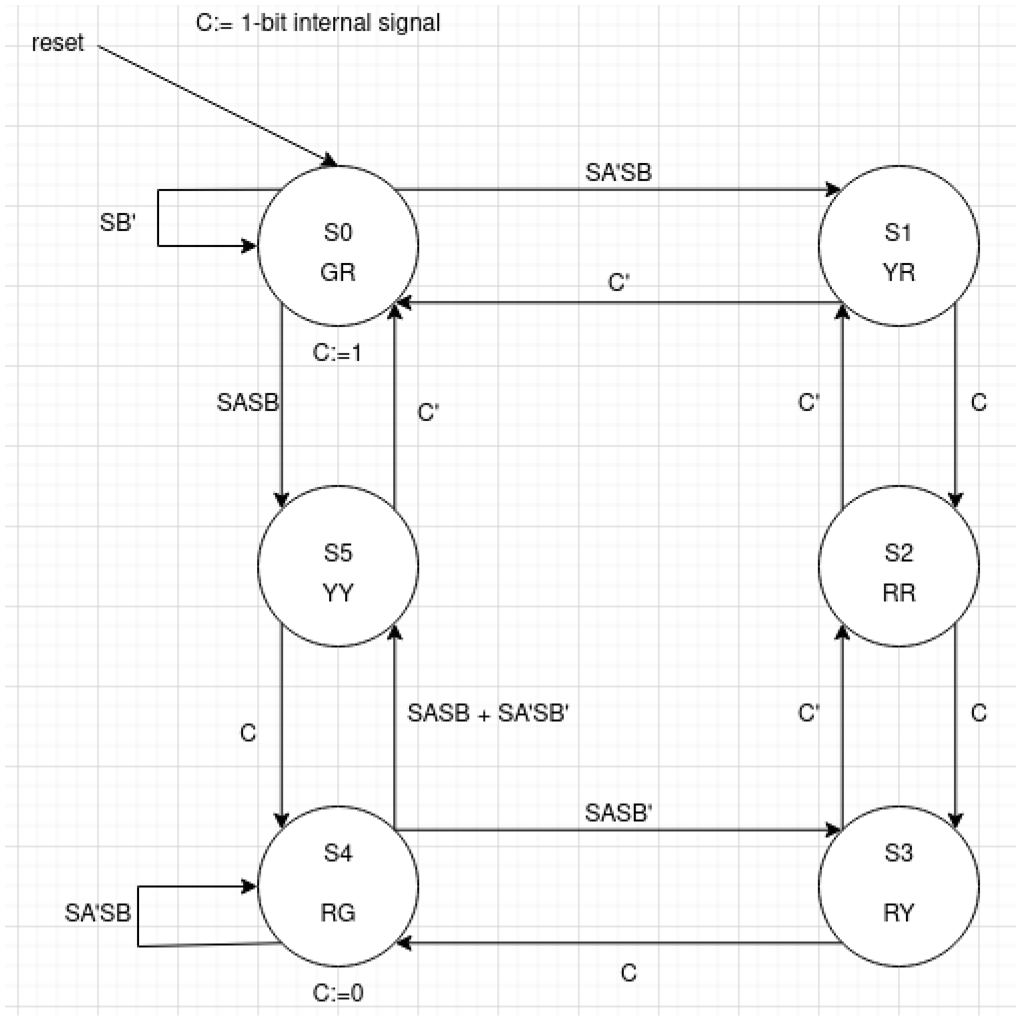
Digital Design
CS223
Lab 4

Erkin Aydın
ID:22002956

27/11/2022

Part a):

<u>Improved Moore Machine State Transition Diagram:</u>



I have introduced an internal signal C. This is neither an output nor an input and not a state bit. The purpose of this signal is to track whether the previous green light state was S0 or S4. If it was S0, C is set to 1. If it was S4, C is set to 0. This way, we can change the state, not being affected by the traffic fluctuations in the middle of light changing. This signal is added as a simplifier to the machine.

State Encodings:

State Bits			
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**

State Transition Table:

Current State			Inputs		Internal Signal		Next State		
S2	**S1**	**S0**	**SA**	**SB**	**C**	**C’(next)**	**S2’**	**S1’**	**S0’**
0	0	0	0	0	x	1	0	0	0
0	0	0	0	1	x	1	0	0	1
0	0	0	1	0	x	1	0	0	0
0	0	0	1	1	x	1	1	0	1
0	0	1	x	x	0	0	0	0	0
0	0	1	x	x	1	1	0	1	0
0	1	0	x	x	0	0	0	0	1
0	1	0	x	x	1	1	0	1	1
0	1	1	x	x	0	0	0	1	0
0	1	1	x	x	1	1	1	0	0
1	0	0	0	0	x	0	1	0	1
1	0	0	0	1	x	0	0	1	1
1	0	0	1	0	x	0	0	1	1
1	0	0	1	1	x	0	1	0	1
1	0	1	x	x	0	0	0	0	0
1	0	1	x	x	1	1	1	0	0

Output Table:

Current State			Outputs					
S2	**S1**	**S0**	**LA1**	**LA1**	**LA0**	**LB2**	**LB1**	**LB0**
0	0	0	0	0	1	1	1	1
0	0	1	0	1	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	0	1
1	0	1	0	1	1	0	1	1

Next State Equations:

On the right-hand side, “‘“ means complement. For example, S2’ means the complement of S2 on the right-hand side. On the left-hand side, it means “next.” For example, S2’ means the next value of S2 on the left-hand side.

S2’ = S2S1’S0’SA’SB + S2’S1S0C + S2S1’S0’SASB + S2S1’S0’SA’SB’ + S2’S1’S0’SASB

S1’ = S2S1’S0’SASB’ + S2’S1S0’C + S2’S1S0C’ + S2’S1’S0C

S0’ = S2S1’S0’SASB + S2S1’S0’SA’SB’ + S2’S1’S0’SASB + S2’S1’S0’SA’SB + S2’S1S0’C’ + S2S1’S0’SASB’ + S2’S1S0’C

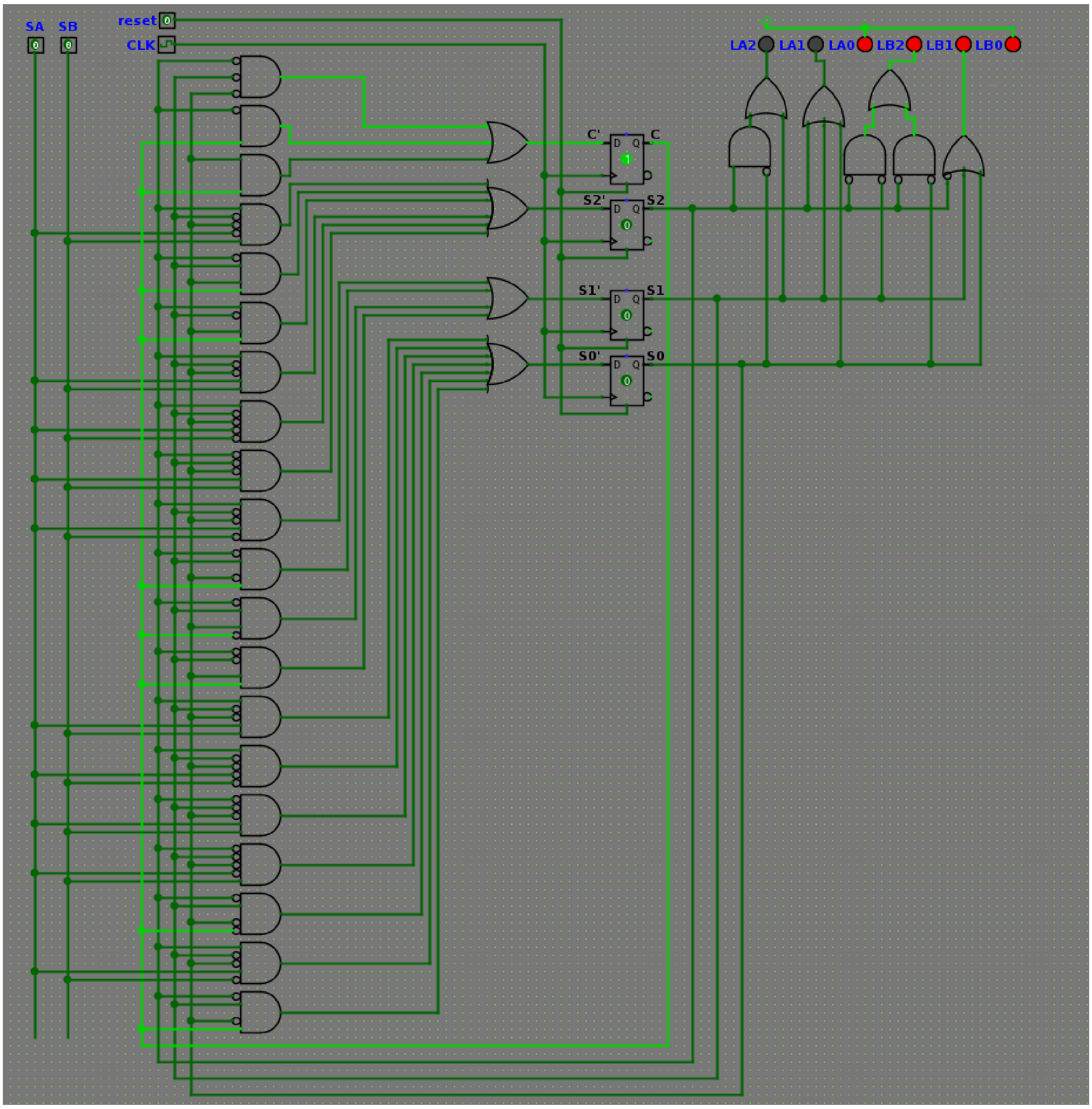
The Equation of Internal Signal C = S2’S1’S0’ + S2’C + S0C

Output Equations:

On the right-hand side, “‘“ means complement. For example, S2’ means the complement of S2 on the right-hand side. On the left-hand side, it means “next.” For example, S2’ means the next value of S2 on the left-hand side.

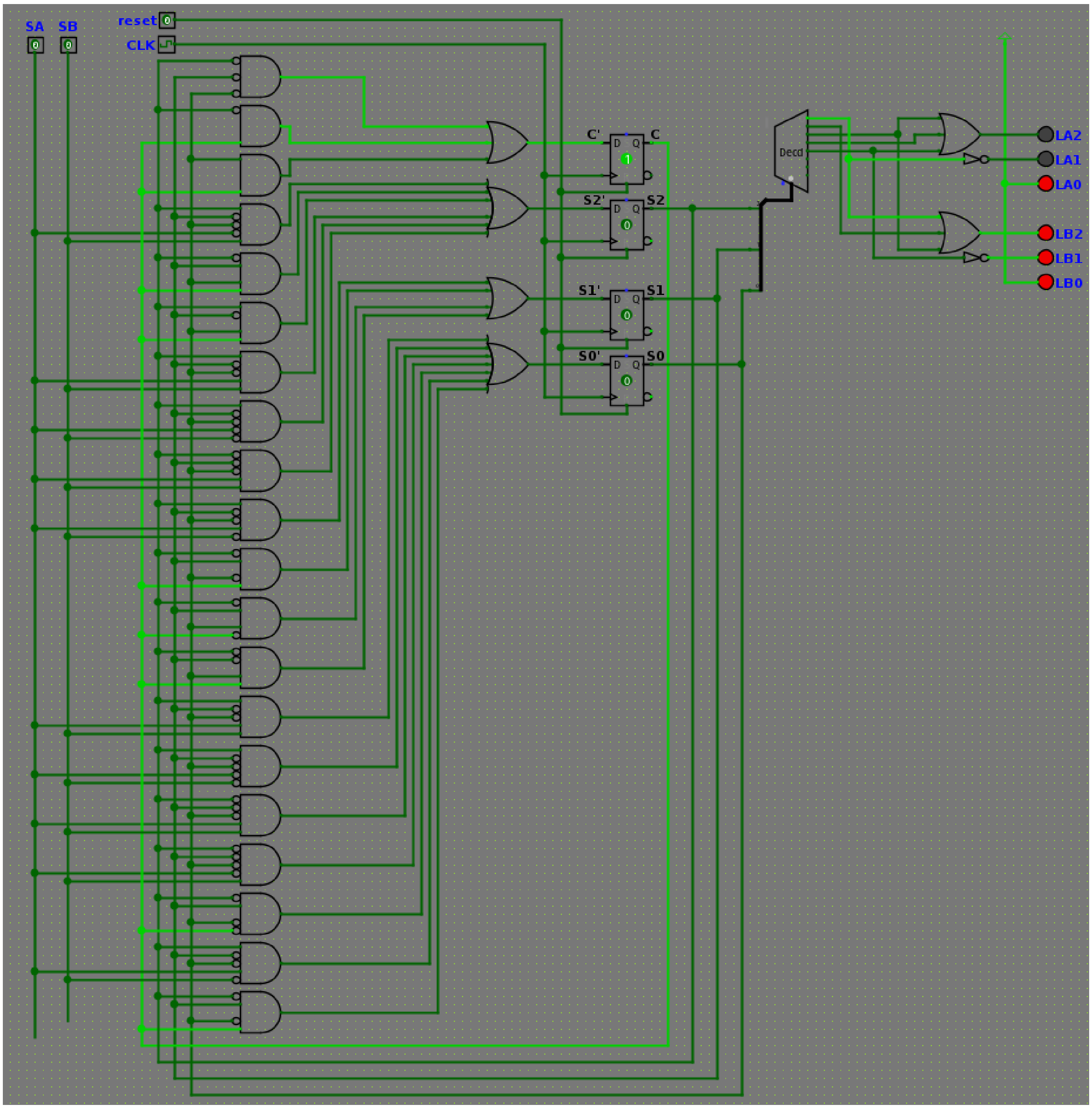
LA2 = S1 + S2S0’
LA1 = S2 + S1 + S0
LA0 = 1
LB2 = S2’S1’ + S2’S0’
LB1 = S2’ + S1 + S0
LB0 = 1

Part b):



Part c): 4 flip-flops are required.

Part d):



Part e):

<u>Design Code:</u>

```
`timescale 1ms / 1ms

module state_machine(
        input Sa, Sb, reset,
        output La2, La1, La0, Lb2, Lb1, Lb0);

        logic clk;

        // Clock Procedure
        always begin
        clk <= 0;
        #1500;
        clk <= 1;
        #1500;
        end
        // Note: Procedure repeats

        logic c = 0;
        logic s2 = 0;
        logic s1 = 0;
        logic s0 = 0;
        logic c_next, s2_next, s1_next, s0_next;

        logic c_1, c_2, c_3;
        logic s2_1, s2_2, s2_3, s2_4, s2_5, s2_6;
        logic s1_1, s1_2, s1_3, s1_4;
        logic s0_1, s0_2, s0_3, s0_4, s0_5, s0_6, s0_7;

        logic l1, l2, l3;

        DFF d_c(c, c_next, clk);
        DFF d_s2(s2, s2_next, clk);
        DFF d_s1(s1, s1_next, clk);
        DFF c_s0(s0, s0_next, clk);

        and ac1(c_1, ~s2, ~s1, ~s0);
        and ac2(c_2, ~s2, c);
        and ac3(c_3, s0, c);

        or o1(c_next, c_1, c_2, c_3);
        and a1(s2_1, s2, ~s1, ~s0, ~Sa, Sb);
        and a2(s2_2, ~s2, s1, s0, c);
        and a3(s2_3, s2, ~s1, s0, c);
        and a4(s2_4, s2, ~s1, ~s0, Sa, Sb);
        and a5(s2_5, s2, ~s1, ~s0, ~Sa, ~Sb);
        and a6(s2_6, ~s2, ~s1, ~s0, Sa, Sb);

        or o2(s2_next, s2_1, s2_2, s2_3, s2_4, s2_5, s2_6);

        and a7(s1_1, s2, ~s1, ~s0, Sa, ~Sb);
        and a8(s1_2, ~s2, s1, ~s0, c);
        and a9(s1_3, ~s2, s1, s0, ~c);
        and a10(s1_4, ~s2, ~s1, s0, c);

        or o3(s1_next, s1_1, s1_2, s1_3, s1_4);

        and a11(s0_1, s2, ~s1, ~s0, Sa, Sb);
        and a12(s0_2, s2, ~s1, ~s0, ~Sa, ~Sb);
        and a13(s0_3, ~s2, ~s1, ~s0, Sa, Sb);
        and a14(s0_4, ~s2, ~s1, ~s0, ~Sa, Sb);
        and a15(s0_5, ~s2, s1, ~s0, ~c);
        and a16(s0_6, s2, ~s1, ~s0, Sa, ~Sb);
        and a17(s0_7, ~s2, s1, ~s0, c);

        or o4(s0_next, s0_1, s0_2, s0_3, s0_4, s0_5, s0_6, s0_7);

        and aout_1(l1, s2, ~s0);
        or oout_1(La2, l1, s1);

        or oout_2(La1, s0, s1, s2);

        assign La0 = 1;

        and aout_2(l2, ~s2, ~s1);
        and aout_3(l3, ~s2, ~s0);
        or oout_3(Lb2, l2, l3);

        or oout_4(Lb1, ~s2, s1, s0);

        assign Lb0 = 1;

endmodule

module DFF(Q,D,Clk);
```

```
        input D,Clk;
        output Q;
        reg Q;

        always @(posedge Clk)
        Q<=D;
endmodule
```

<u>Testbench:</u>

```
`timescale 1s / 1s

module tb_state_machine();
        logic Sa, Sb, reset;
        logic La2, La1, La0, Lb2, Lb1, Lb0;

        state_machine states(Sa, Sb, reset, La2, La1, La0, Lb2, Lb1, Lb0);

        initial begin
        assign reset = 0;

        assign Sa = 0;
        assign Sb = 0; #5;

        assign Sb = 1; #18;
        assign Sa = 1; #21;

        assign Sa = 1;
        assign Sb = 0;
        end

endmodule
```