CS223 - DIGITAL DESIGN

SECTION 3 - LAB 5

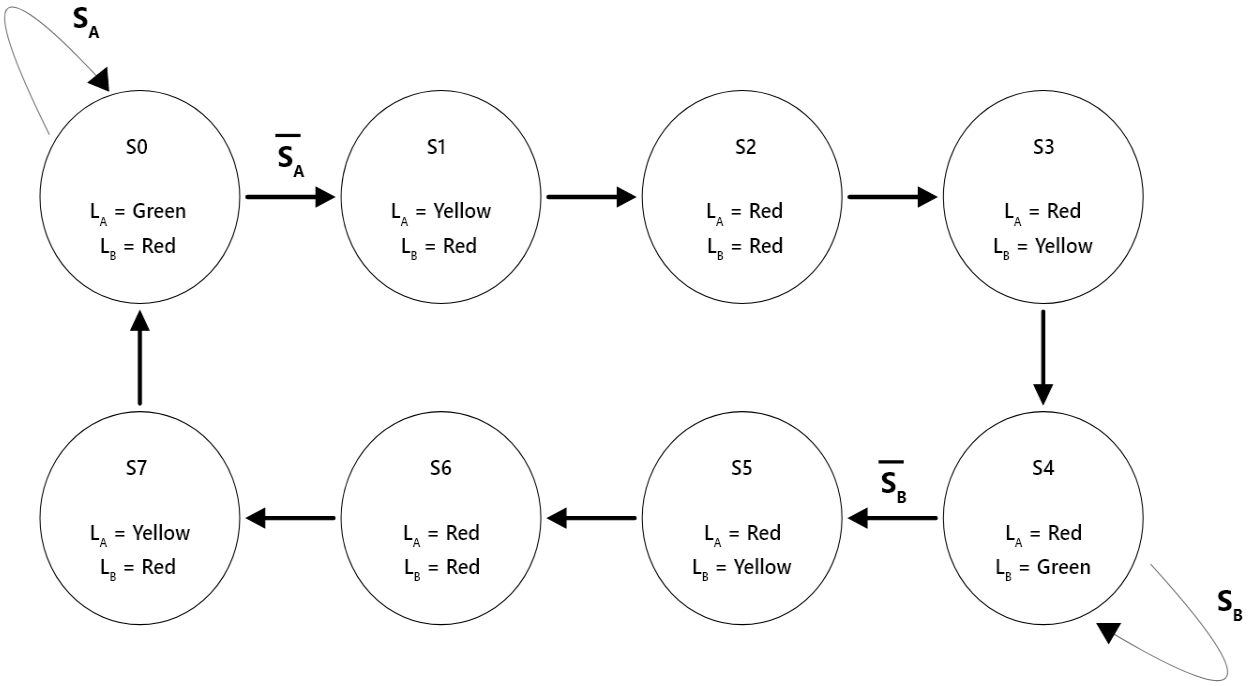
ARDA TAVUSBAY

21902722

28.11.2021

Part A)

i. State Transition Diagram



ii. State Encodings, State Transition Table, Output Table, Next State and Output Equations

State Encoding Table

State	Encoding
S0	000
S1	001
S2	010
S3	011
S4	100
S5	101
S6	110
S7	111

Output Encoding Table

Output	Encoding
Red	00
Yellow	01
Green	10

Output Table

S_2	S_1	S_0	L_{B1}	L_{B0}	L_{A1}	L_{A0}
0	0	0	1	0	0	0
0	0	1	1	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	1	0
1	0	0	0	0	1	0
1	0	1	0	1	1	0
1	1	0	1	0	1	0
1	1	1	1	0	0	1

State Transition Diagram

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	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

Next State & Output Equations

$$S_2^{'} = S_2\overline{S_1} + S_0\overline{S_2} + \overline{S_2}S_1S_0$$

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

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

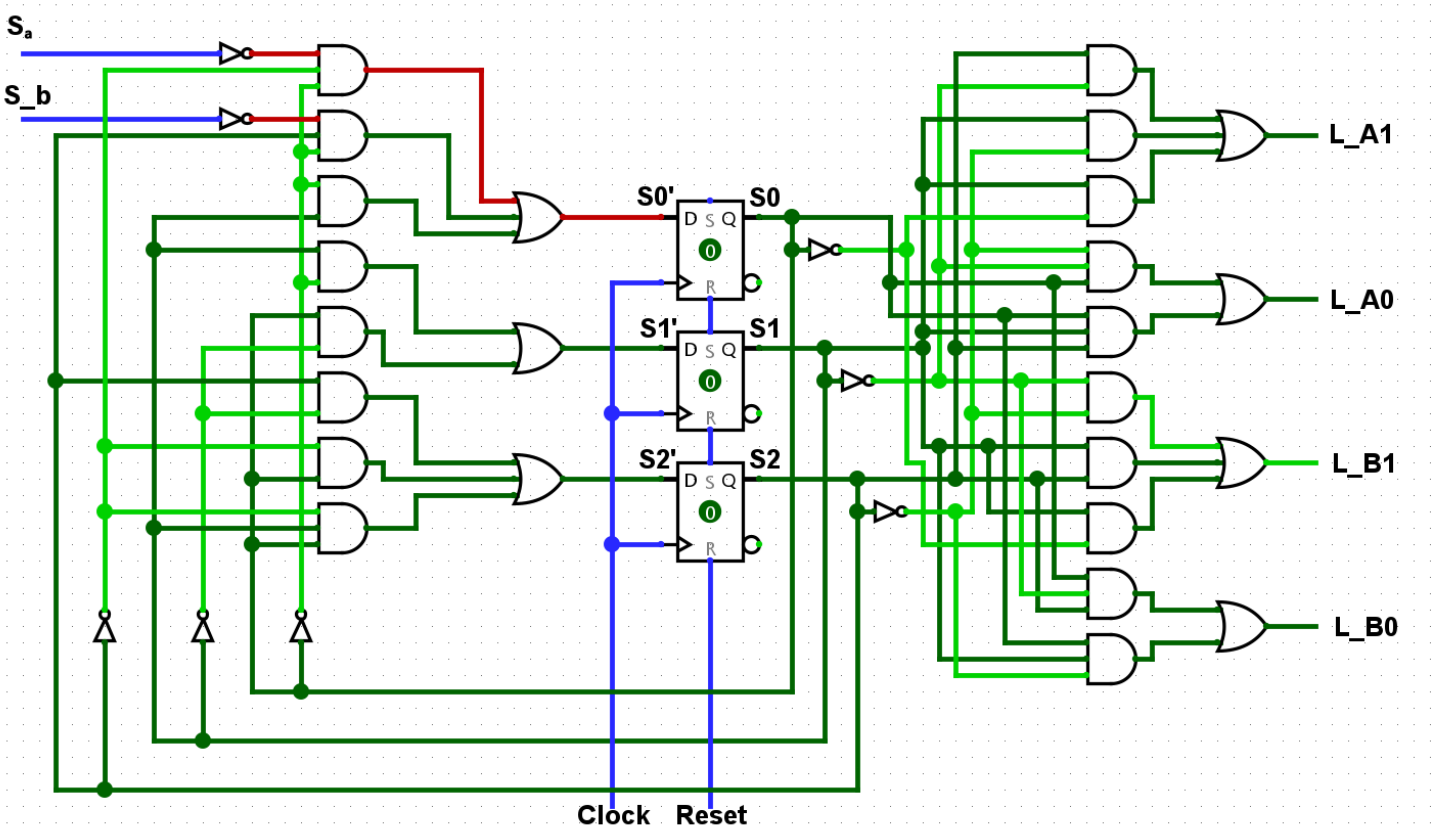
$$L_{A1} = S_2\overline{S_1} + S_1\overline{S_2} + \overline{S_0}S_1$$

$$L_{A0} = \overline{S_2 S_1}S_0 + S_2S_1S_0$$

$$L_{B1} = \overline{S_2 S_1} + S_2S_1 + \overline{S_0}S_1$$

$$L_{B0} = \overline{S_2}S_1S_0 + S_2\overline{S_1}S_0$$

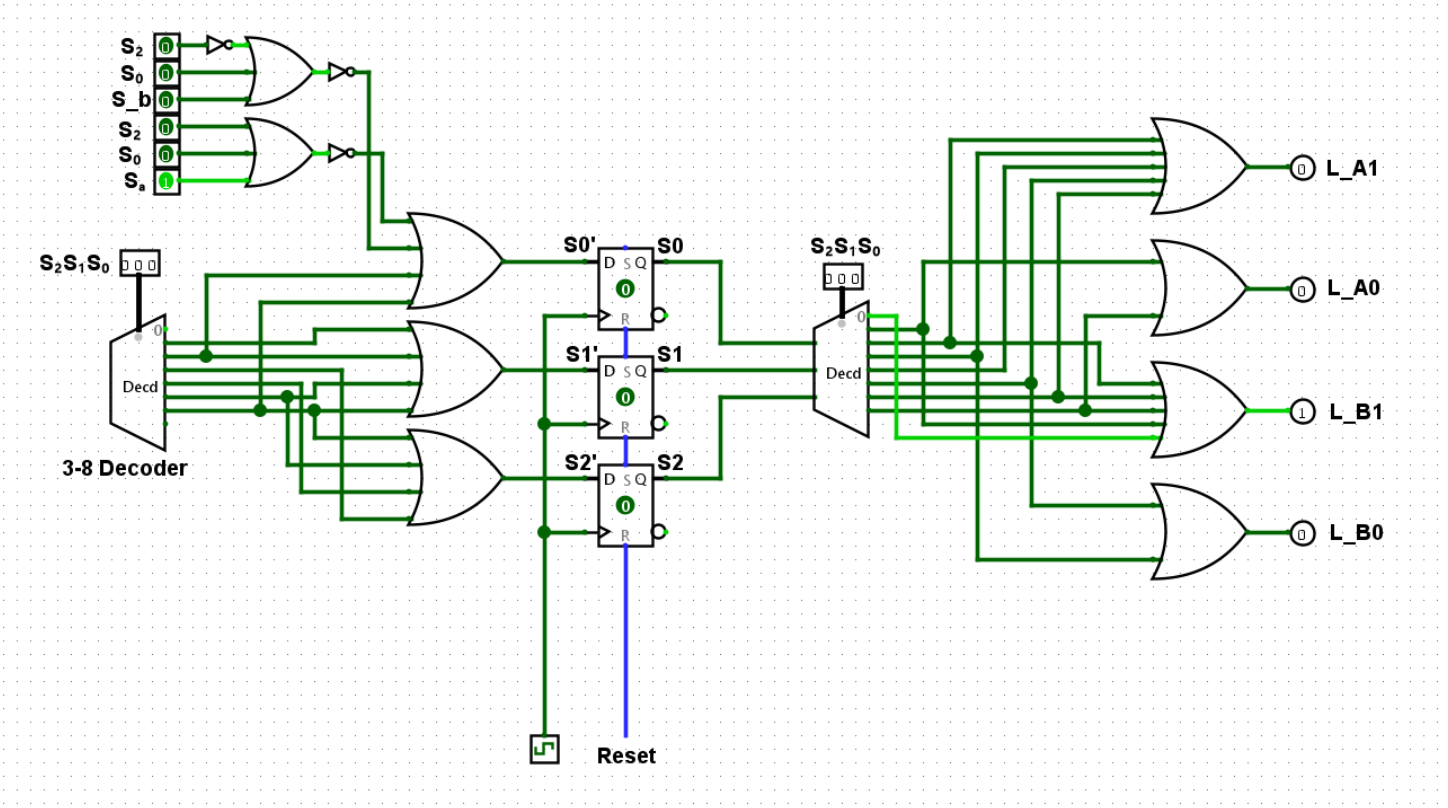
Finite State Machine Schematic



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

We need to use three flip-flops since we have eight states which corresponds to three state variables. Thus, each flip-flop will hold one state variable.

Part C) Redesign your outputs using decoders.



(When every input besides S_A traffic sensor is 0, output L_A is 00 which is green, and L_B is 10 which is red as seen in the figure.)

Design Code & Testbench Part - Design of Structural Traffic Light System using Decoders in System Verilog and a Testbench for it.

```
`timescale 1ns / 1ps

module TrafficLightSystem( input logic clk, res, sensor_a, sensor_b, output logic [2:0] Led1,
Led2 );

logic S2, S1, S0; //Current States
logic S2n, S1n, S0n; //Next States
logic [7:0] out1, out2; //Decoder Outputs
logic LA1, LA0, LB1, LB0; //Light Output

//Decoder for determining current states
Decoder3to8 dec1(S2, S1, S0, out1);

//Decoder for determining outputs
Decoder3to8 dec2(S2, S1, S0, out2);

//Wiring Outputs to Decoder
assign LA1 = out2[2] | out2[3] | out2[4] | out2[5] | out2[6];
assign LA0 = out2[1] | out2[7];
assign LB1 = out2[0] | out2[1] | out2[2] | out2[6] | out2[7];
assign LB0 = out2[3] | out2[5];

always_ff @ (posedge clk, posedge res)
if (res)
begin
   //Reset States, Back to S0 (000)
   S0 <= 0;
   S1 <= 0;
   S2 <= 0;
end
else
begin
   //Current State <= Next State
   S0 <= S0n;
```

```
    S1 <= S1n;
    S2 <= S2n;
end

always@*
begin
//Determining Next State
S2n <= out1[3] | out1[4] | out1[5] | out1[6];
S1n <= out1[1] | out1[2] | out1[5] | out1[6];
S0n <= out1[2] | out1[6] | ~(~S2 | S0 | sensor_b) | ~(S2 | S0 | sensor_a);

//001 => Red light, 011 => Yellow Light, 111 => Green Light
Led1[0] <= 1;
Led1[1] <= (~LA1 & LA0) | (~LA1 & ~LA0);
Led1[2] <= (~LA1 & ~LA0);

Led2[0] <= 1;
Led2[1] <= (~LB1 & LB0) | (~LB1 & ~LB0);
Led2[2] <= (~LB1 & ~LB0);
end

endmodule

`timescale 1ns / 1ps

module Decoder3to8( input logic i2, i1, i0, output logic [7:0] out );

assign out[0] = ~i2 & ~i1 & ~i0;
assign out[1] = ~i2 & ~i1 & i0;
assign out[2] = ~i2 & i1 & ~i0;
assign out[3] = ~i2 & i1 & i0;
assign out[4] = i2 & ~i1 & ~i0;
assign out[5] = i2 & ~i1 & i0;
assign out[6] = i2 & i1 & ~i0;
assign out[7] = i2 & i1 & i0;

endmodule
```

Testbench for the Structural Traffic Light System:

```
`timescale 1ns / 1ps

module TB_TrafficLightSystem();

logic clk, res, sensor_a, sensor_b;
logic [2:0] Led1, Led2;

TrafficLightSystem dut(clk, res, sensor_a, sensor_b, Led1, Led2);

always
begin
	clk = 0; #25;
	clk = 1; #25;
end
initial begin
	clk = 0; res = 1;
	sensor_a = 0;
	sensor_b = 0; #100;

	res = 0; #100;
	sensor_a = 0;
	sensor_b = 1; #100;
	sensor_a = 1;
	sensor_b = 0; #100;

	res = 1; #100;
	sensor_a = 0;
	sensor_b = 1; #100;
	sensor_a = 1;
	sensor_b = 0; #100;

	res = 0; #100;
	sensor_a = 0;
	sensor_b = 1; #100;
end
endmodule
```