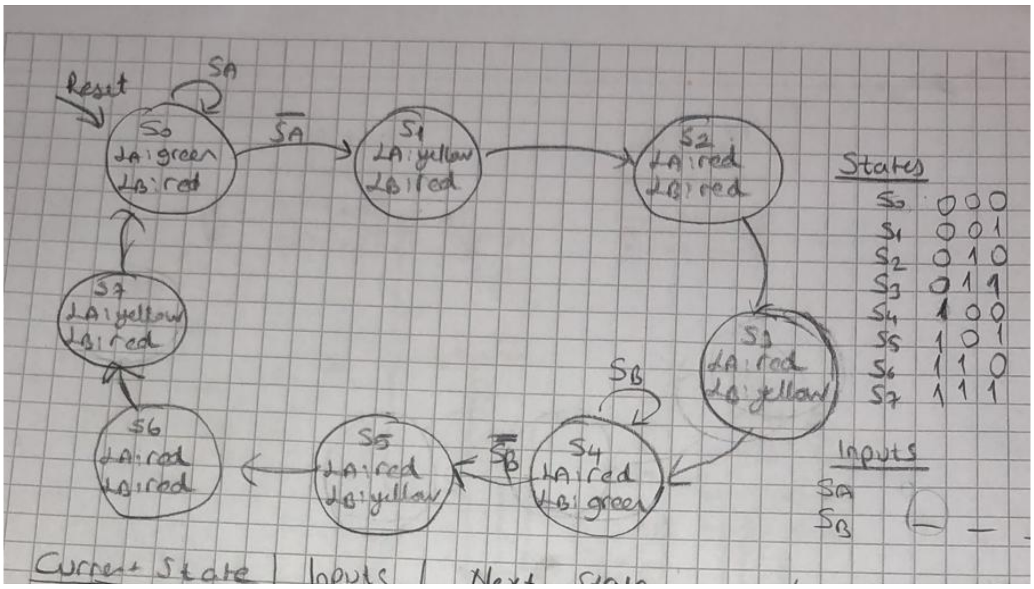
Course: CS-223 Digital Design
Section 01
Lab 04

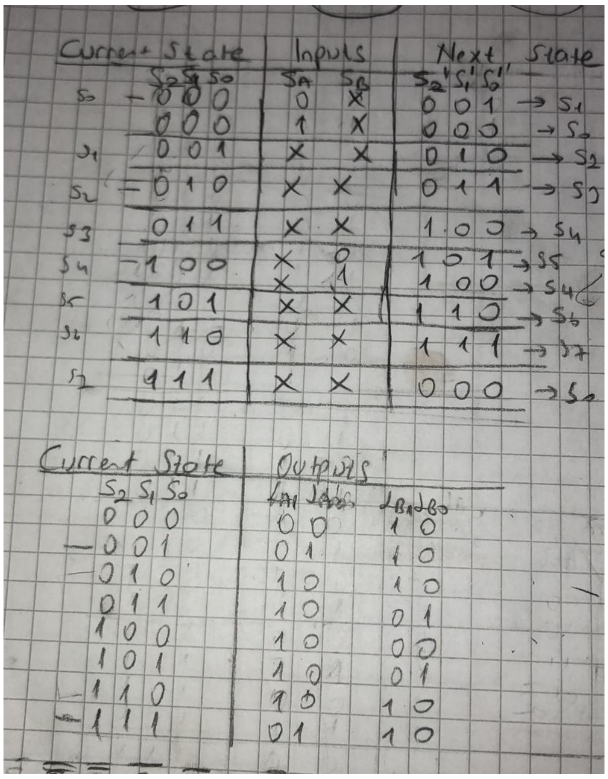
Miray Ayerdem
21901987
06/04/2021

a)

-Moore Finite State Machine (FSM) transition diagram and state encodings

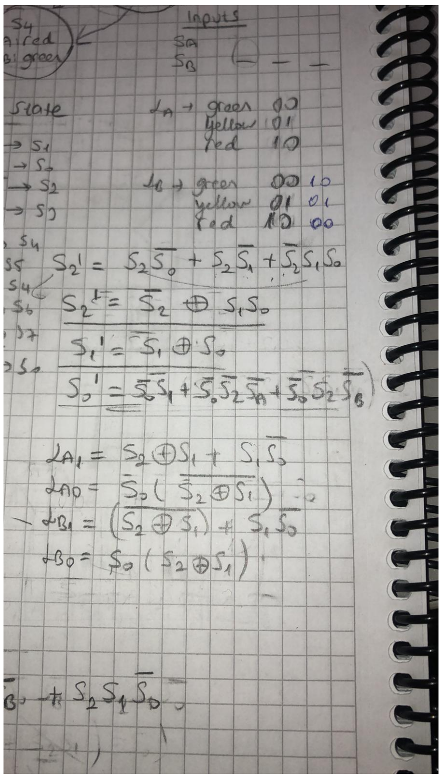


-State transition table, output table, next state, output equations



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

Current State $S_2 S_1 S_0$	Outputs $L_{A1} L_{A0}$	$L_{B1} L_{B0}$
0 0 0	0 0	1 0
0 0 1	0 1	1 0
0 1 0	1 0	1 0
0 1 1	1 0	0 1
1 0 0	1 0	0 0
1 0 1	1 0	0 1
1 1 0	1 0	1 0
1 1 1	0 1	1 0



Inputs
S_A
S_B

L_A →	green	00	
	yellow	01	
	red	10	
L_B →	green	00	10
	yellow	01	01
	red	10	00

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

$$S_2' = \overline{S_2} \oplus S_1S_0$$

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

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

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

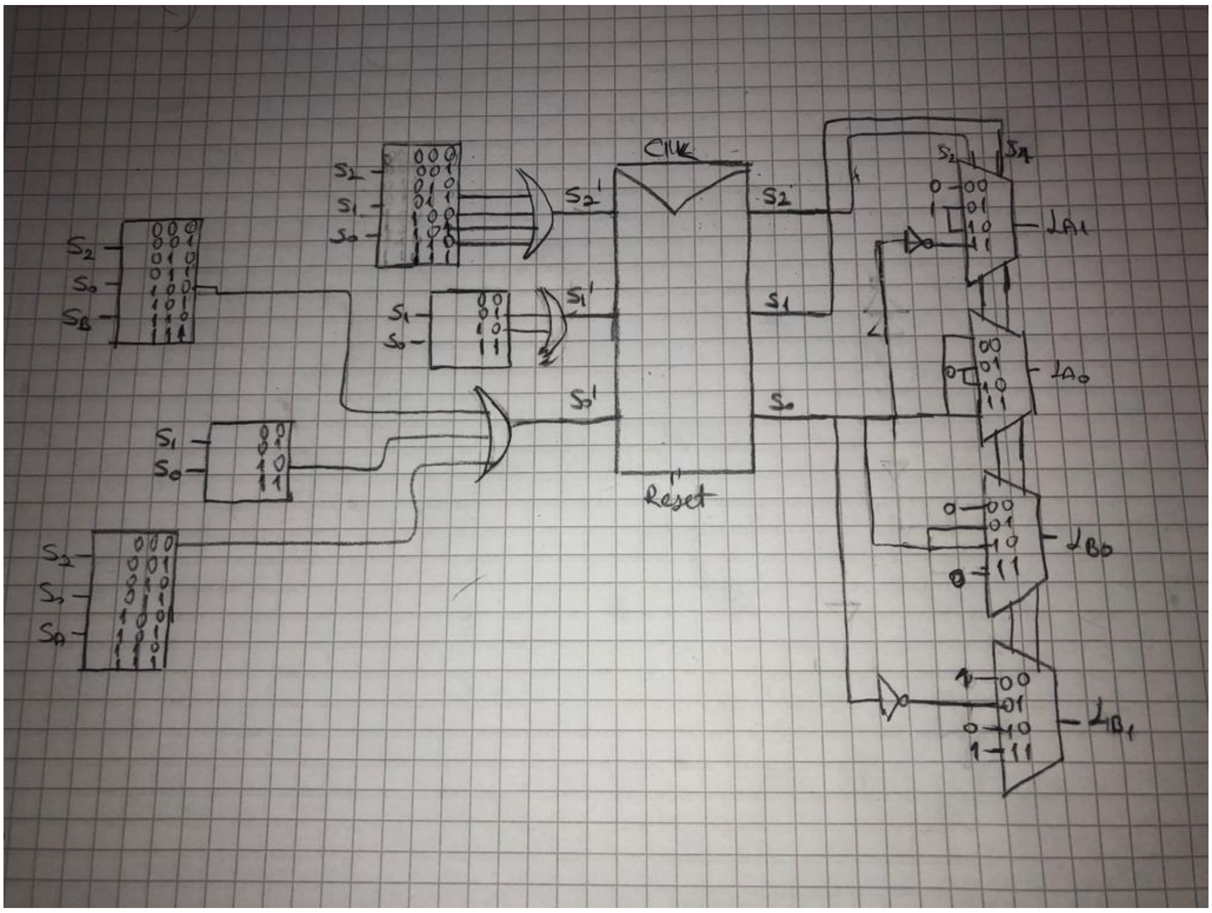
$$L_{A0} = \overline{S_0}(\overline{S_2 \oplus S_1})$$

$$L_{B1} = (\overline{S_2 \oplus S_1}) + S_1\overline{S_0}$$

$$L_{B0} = S_0(S_2 \oplus S_1)$$

$\overline{S}_B + S_2S_1\overline{S_0}$

-FSM’s circuit schematic



b) We need 3 flip-flops because we need 3 bits to show all states.

c)

```
module decoder3_8( input logic in2, in1, in0, output logic [7:0] out1);

assign out1[0] = ~in2 & ~in1 & ~in0;
assign out1[1] = ~in2 & ~in1 & in0;
assign out1[2] = ~in2 & in1 & ~in0;
assign out1[3] = ~in2 & in1 & in0;
assign out1[4] = in2 & ~in1 & ~in0;
assign out1[5] = in2 & ~in1 & in0;
assign out1[6] = in2 & in1 & ~in0;
assign out1[7] = in2 & in1 & in0;

endmodule




module decoder2_4( input logic in1, in0 ,output logic[3:0] out1);

 assign out1[0] = ~in1 & ~in0;
 assign out1[1] = ~in1 & in0;
 assign out1[2] = in1 & ~in0;
 assign out1[3] = in1 & in0;

endmodule



module mux4_1( input logic d0, d1, d2, d3, s0, s1, output logic y);
   assign y = s1 ? ( s0 ? d3: d2 ) : ( s0 ? d1: d0);
endmodule



module Func_Blue(input logic SA, SB, input logic [2:0] state, output logic[2:0] nextstate);
logic [3:0] n3;
logic [7:0]n1, n2, n4;
decoder3_8 s2next(~state[2], state[1], state[0], n4);
decoder2_4 s1next(state[1], state[0], n3);
decoder3_8 s0next(state[2], state[0],SA , n1);
decoder3_8 s0next2(state[2], state[0],SB , n2);
decoder2_4 s0next3(state[1], state[0], n3);
assign nextstate[0] = n1[0] | n2[4] | n3[2];
assign nextstate[1] = n3[1] | n3[2];
assign nextstate[2] = n4[0] |n4[1] | n4[2] | n4[7];

endmodule




module Func_Green(input logic[2:0] current_state, output logic [1:0] LA, LB);
mux4_1 la0module(current_state[0],0,0, current_state[0], current_state[1],current_state[2],
LA[0]);
```

```
mux4_1 la1module(0, 1, 1, ~current_state[0], current_state[1], current_state[2], LA[1]);
mux4_1 lb0module(0,current_state[0], current_state[0], 0, current_state[1], current_state[2],
LB[0]);
mux4_1 lb1module(1, ~current_state[0], 0, 1, current_state[1], current_state[2], LB[1]);


endmodule



// Flip-flop D
module dff(
        input clk, rst, d,
        output logic q);

  always@(posedge clk or posedge rst)
     if (rst)
       q <= 0;
     else
       q <= d;
endmodule


module TrafficLightFSM(input logic reset, clk , SA, SB, output logic  [2:0] current_state,
                        output logic [2:0] next_state, LA3, LB3
                         );

//logic [2:0]  current_state;

logic [1:0] LA, LB;

//output logic

Func_Blue m2(SA, SB, current_state, next_state);
dff dff0(clk, reset, next_state[0], current_state[0]);
dff dff1(clk, reset, next_state[1], current_state[1]);
dff dff2(clk, reset, next_state[2], current_state[2]);


Func_Green m1(current_state, LA, LB);


assign LA3[2] = ~LA[1] & ~LA[0];
assign LA3[1] = ~LA[1];
assign LA3[0] = 1;
assign LB3[2] = ~LB[1] & ~LB[0];
assign LB3[1] = ~LB[1];
assign LB3[0] = 1;
```

```
endmodule
```