CS 223 - Digital Design

Laboratory Assignment 4

Traffic Light System

Arda İçöz

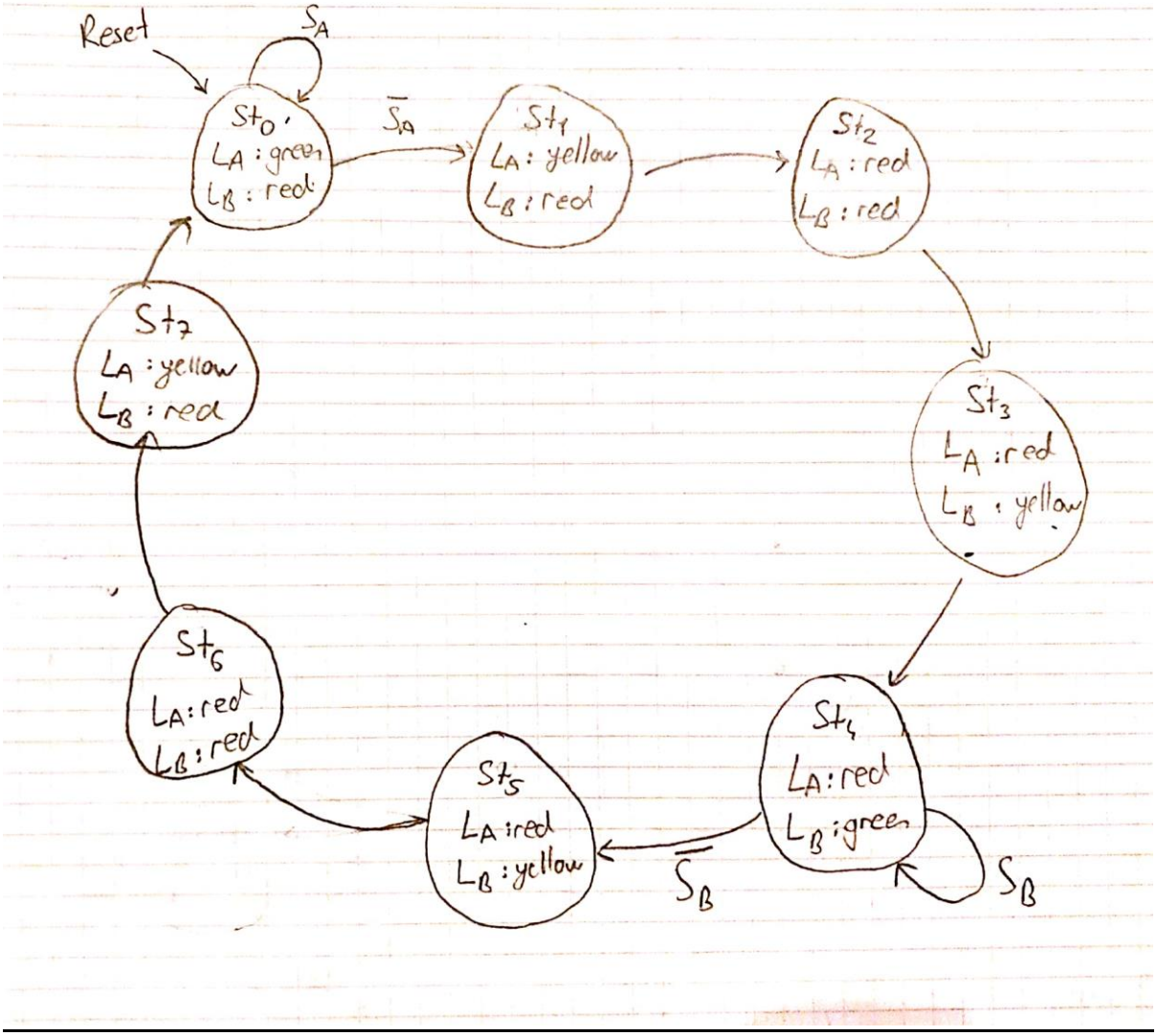
21901443

Section 02

27/11/2020

Part A:

Moore state machine transition diagram:



State encodings:

State encoding

State	Encoding $S_{2:0}$
St_0	000
St_1	001
St_2	010
St_3	011
St_4	100
St_5	101
St_6	110
St_7	111

State transition table:

State transition table

Current State S	Inputs S_A	S_B	Next state S'
St_0	0	X	St_1
St_0	1	X	St_0
St_1	X	X	St_2
St_2	X	X	St_3
St_3	X	X	St_4
St_4	X	0	St_5
St_4	X	1	St_4
St_5	X	X	St_6
St_6	X	X	St_7
St_7	X	X	St_0

Output table:

Output table

Current State			Outputs			
St_2	St_1	St_0	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

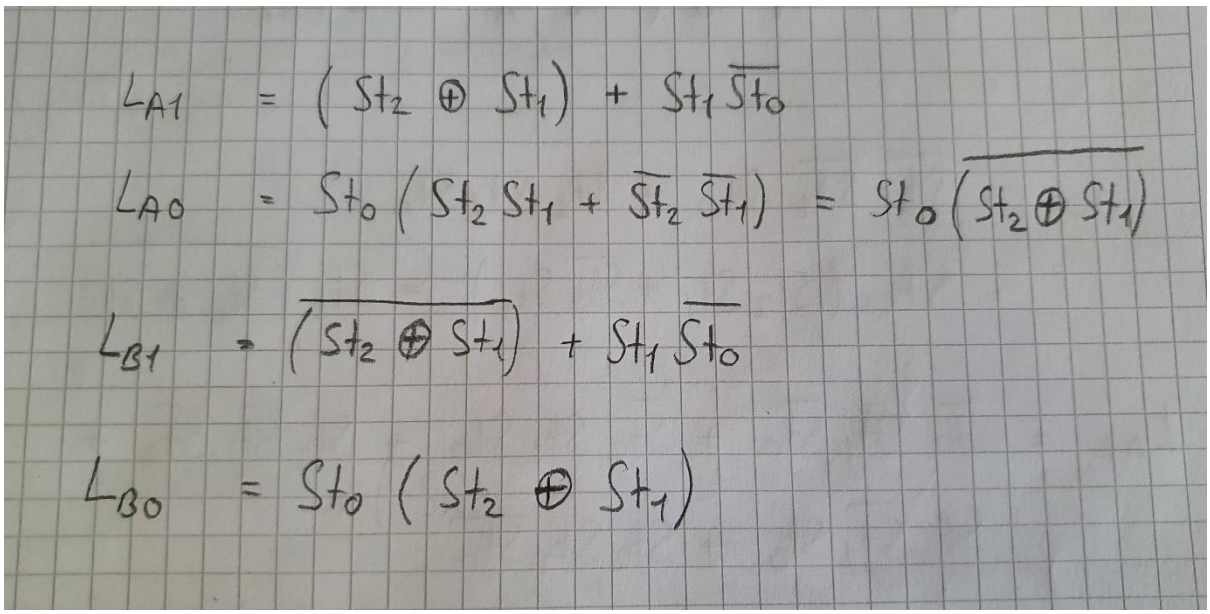
Next state equation:

$$St_2' = \overline{St_2} \oplus St_1 St_0$$

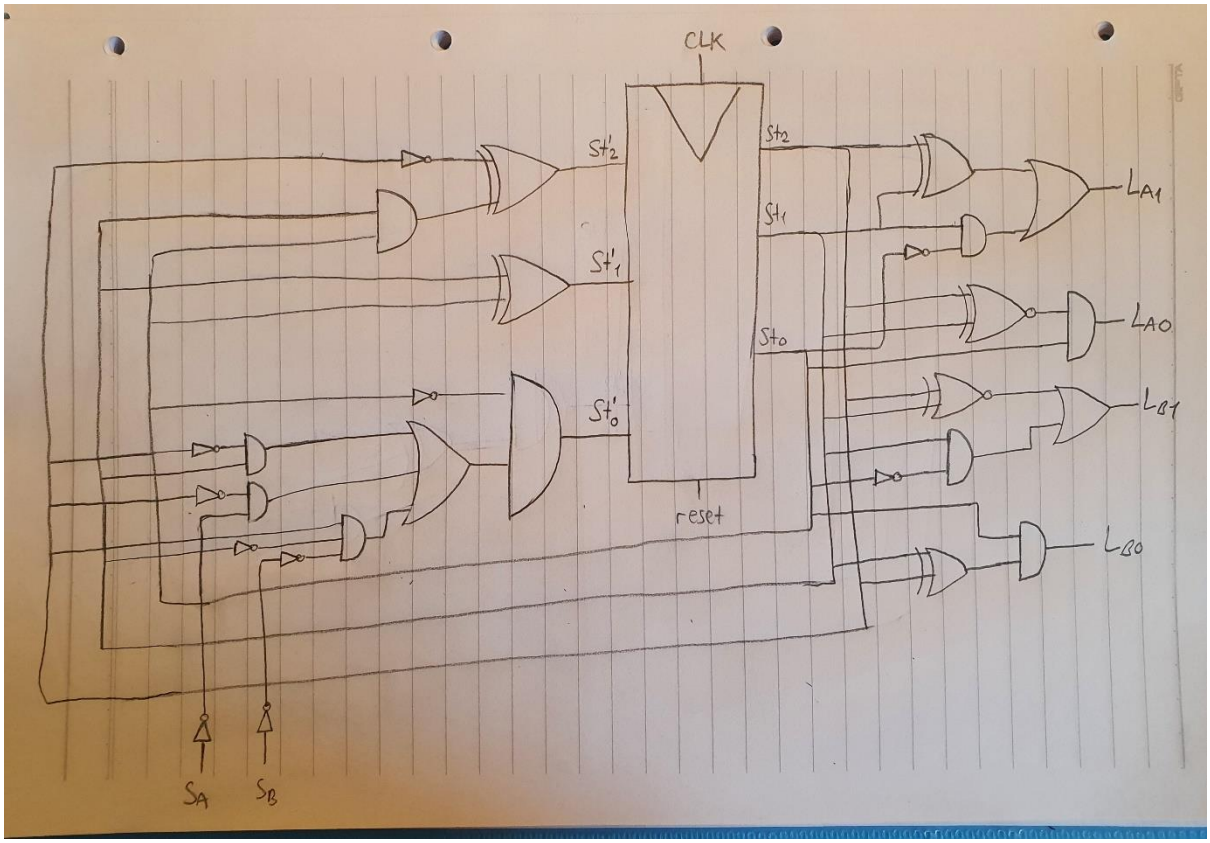
$$St_1' = St_1 \oplus St_0$$

$$St_0' = \overline{St_0}\left(\overline{St_2}\ \overline{S_A} + \overline{St_2}\ St_1 + St_2\ \overline{St_1}\ \overline{S_B}\right)$$

Output equation:



Finite State Machine schematic:



Part B:

We are going to use 3 flip-flops to implement this problem because we are using 3 bits to show all states.

Part C:

Basys3 has a frequency of 100MHz, so it equals to 100,000,000Hz. We need to think 1/3 Hz as 1Hz in order to achieve our frequency.

$$\frac{10^8 Hz}{1/3\,Hz} = 3 * 10^8 Hz$$

Hence, we need to count $3 * 10^8$ amount of rising edge of the clock.

SystemVerilog code:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Author: Arda Icoz
// Create Date: 27.11.2020 14:41:51
// Module Name: clock_cycle
// Project Name: Lab04
// Description: This is the SystemVerilog code for obtaining 3 second
delay.
// Additional Comments: This code is taken from Xilinx's official
documentation. The code was originally written in Verilog so all "reg"
variables
//      have been changed to "logic". And, the "constantNumber" variable
has been determined as 300 million to achieve the clock frequency of 1/3
Hz.
// Sources: https://reference.digilentinc.com/learn/programmable-
logic/tutorials/counter-and-clock-divider/start
//////////////////////////////////////////////////////////////////////////////////


module clock_cycle(
    input clk,
    input rst,
    output logic clk_div
    );

    localparam constantNumber = 150000000;
    logic [31:0] count;

    always @(posedge(clk), posedge(rst))
    begin
        if (rst == 1'b1)
            count <= 32'b0;
        else if (count == constantNumber - 1)
            count <= 32'b0;
```

```
        else
            count <= count + 1;
    end

    always @ (posedge(clk), posedge(rst))
    begin
        if (rst == 1'b1)
            clk_div <= 1'b0;
        else if (count == constantNumber - 1)
            clk_div <= ~clk_div;
        else
            clk_div <= clk_div;
    end
endmodule
```

Part D:

Traffic light system:

```
`timescale 1ns / 1ps

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

    typedef enum logic [2:0] {State0, State1, State2, State3, State4,
State5, State6, State7} statetype;
    statetype currentState, nextState;

    typedef enum logic[1:0] {red, yellow, green} lights;
    lights light1, light2;

    //output logic
    logic [1:0] LA;
    logic [1:0] LB;

    //synchronous logic
    always_ff @(posedge clk)
        if(reset)   currentState <= State0;
        else        currentState <= nextState;

    //combinational logic
    always_comb
        case(currentState)
            State0: begin
                light1 = green;
                light2 = red;
                if(SA)  nextState = State0;
                else    nextState = State1;
            end

            State1: begin
                light1 = yellow;
```

```
                light2 = red;
                nextState = State2;
            end

            State2: begin
                light1 = red;
                light2 = red;
                nextState = State3;
            end

            State3: begin
                light1 = red;
                light2 = yellow;
                nextState = State4;
            end

            State4: begin
                light1 = red;
                light2 = green;
                if(SB)  nextState = State4;
                else    nextState = State5;
            end

            State5: begin
                light1 = red;
                light2 = yellow;
                nextState = State6;
            end

            State6: begin
                light1 = red;
                light2 = red;
                nextState = State7;
            end

            State7: begin
                light1 = yellow;
                light2 = red;
                nextState = State0;
            end

            default: begin
                light1 = red;
                light2 = red;
                nextState = State0;
            end
        endcase

    //output assignments
    assign LA = light1;
    assign LB = light2;

    assign LA3[2] = LA[1];
    assign LA3[1] = LA[1] | LA[0];
    assign LA3[0] = 1;
    assign LB3[2] = LB[1];
    assign LB3[1] = LB[1] | LB[0];
    assign LB3[0] = 1;
endmodule
```

Traffic light system:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
///////
// Author: Arda Icoz
// Create Date: 27.11.2020 16:19:15
// Module Name: TrafficLightFSM_TB
// Project Name: Lab04
// Description: This is the testbench for TrafficLightFSM
//////////////////////////////////////////////////////////////////////////////////
///////


module TrafficLightFSM_TB();

    logic clk;
    logic reset;
    logic SAinput;
    logic SBinput;
    logic [2:0] LA3output;
    logic [2:0] LB3output;

    //device under test
    TrafficLightFSM dut(clk, reset, SAinput, SBinput, LA3output,
LB3output);
    //TrafficLightFSM_WithClock dut(clk, reset, SAinput, SBinput,
LA3output, LB3output);

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

    //testing
    initial
        begin
            reset <= 1; #100;
            reset <= 0; #100;
                SAinput <= 1; SBinput <= 0; #100;//SA is TRUE, traffic in
Road A; SB is FALSE, no traffic in Road B
                SAinput <= 0; SBinput <= 0; #100; //SA is FALSE, no traffic
in Road A; SB is FALSE, no traffic in Road B
                SAinput <= 0; SBinput <= 1; #100; //SA is FALSE, no traffic
in Road A; SB is TRUE, traffic in Road B
                SAinput <= 1; SBinput <= 1; #100; //SA is TRUE, traffic in
Road A; SB is TRUE, traffic in Road B
        end
endmodule
```

Part E:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
///////
// Author: Arda Icoz
// Create Date: 27.11.2020 22:30:12
// Module Name: TrafficLightFSM_WithClock
// Project Name: Lab04
//////////////////////////////////////////////////////////////////////////////////
///////


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

    logic clock_output;
    clock_cycle clockDivider(clk, reset, clock_output);

    TrafficLightFSM trafficLightFSM(clk, reset, SA, SB, LA3, LB3);
endmodule
```