CS 223 Digital Design

Section 5

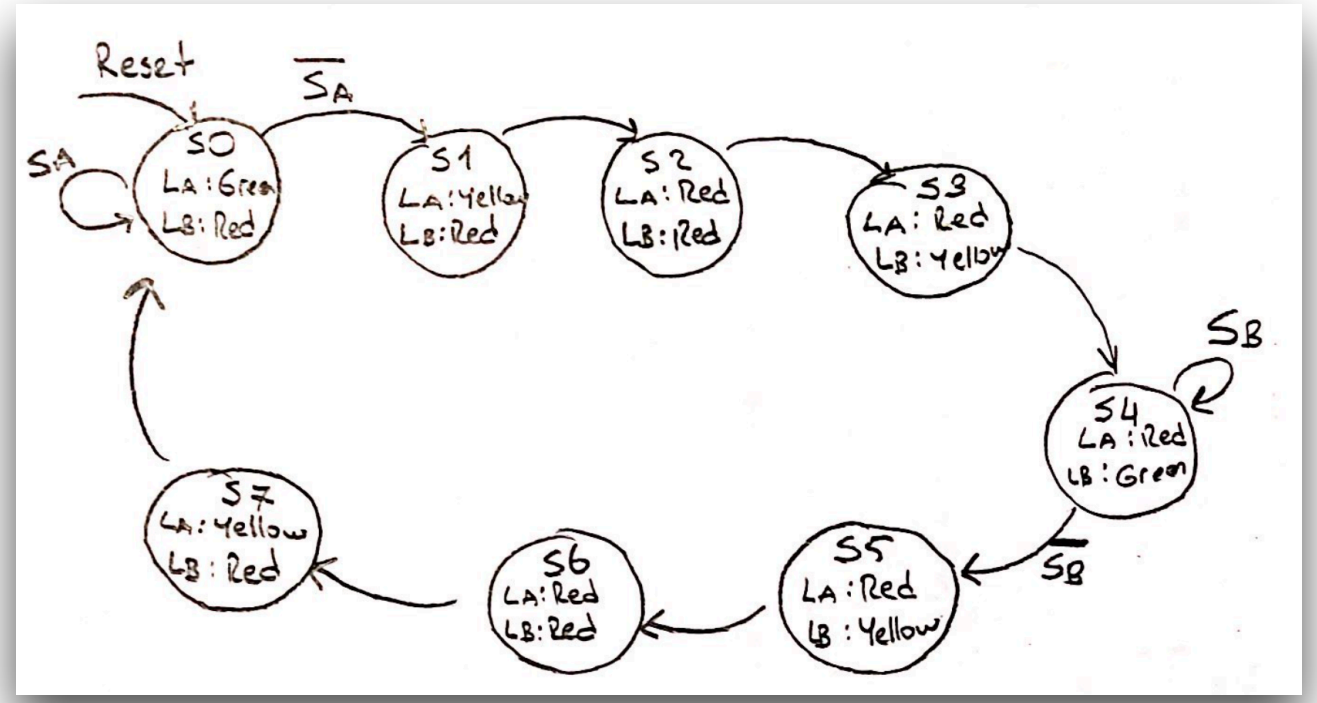
Lab 4

Name: Esad Ismail Tok

ID: 21801679

Date: 22.11.2020

State Transition diagram for the Traffic Light Controller



State and Output Encodings

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

Output	Encoding
Green	011
Yellow	001
Red	111

State Transition Table

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

Output Table

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

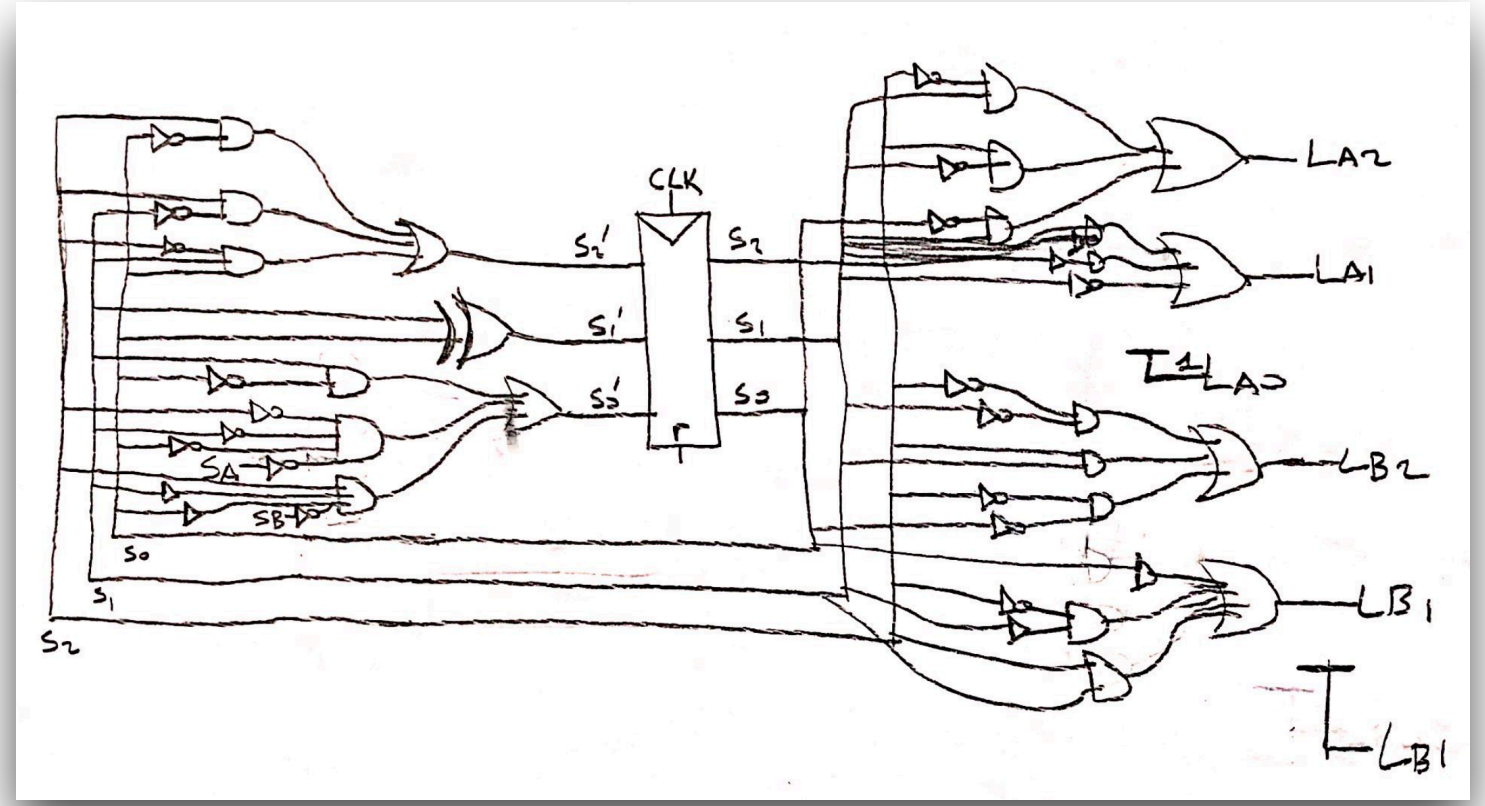
Next State Boolean Equations

$$S_2' = S_2\overline{S_0} + S_2\overline{S_1} + \overline{S_2}S_1S_0$$
$$S_1' = S_1 \oplus S_0$$
$$S_0' = S_1\overline{S_0} + \overline{S_2}\,\overline{S_1}\,\overline{S_0}\,\overline{S_A} + S_2\overline{S_1}\,\overline{S_0}\,\overline{S_B}$$

Output Boolean Equations

$$L_{A2} = \overline{S_2}S_1 + S_2\overline{S_1} + S_1\overline{S_0}$$
$$L_{A1} = \overline{S_0} + \overline{S_2}S_1 + S_2\overline{S_1}$$
$$L_{A0} = 1$$
$$L_{B2} = \overline{S_2}\,\overline{S_1} + S_2S_1 + \overline{S_2}\,\overline{S_0}$$
$$L_{B1} = \overline{S_0} + \overline{S_2}\,\overline{S_1} + S_2S_1$$
$$L_{B0} = 1$$

Traffic Light System Circuit Schematic



How Many Flip-Flops are Needed for This Problem?

- The number of flip-flops needed to implement this FSM is 3.
- Because there are 8 states and those 8 states can be represented by using 3 bits of state and we need 3 flip-flops for 3 bits of state.

Clock Generator with Three Seconds Period

- The clock frequency of the Basys3 board is 100 *MHz.* It is needed a clock signal with the period of three seconds which means that it is needed a clock with a frequency of 1/3 *Hz.*
- The needed count number to get a desired frequency can be calculated by the formula below, in case the counter starts at 1.

Count Number = 100 *MHz* / 2 * Desired Frequency (*Hz*)

- By using the formula above it can be understood that the clock generator should count up to 15 * 10^7 and then toggle the newly generated clock signal and then counter returns to the beginning again.
- Therefore it is needed to count 15 * 10^7 clock edges to toggle the signal, from 0 -> 1 or 1 -> 0, of the generated clock.

SystemVerilog Code for the Clock Generator

```
`timescale 1ns / 1ps


// Low level module to generate a clock with 3 seconds period
module clockCounter(input logic clk, output logic clk_out = 0);

    // Count number = 100MHz / 2 x desired frequency (Hz)
    // We need to count 15 x 10^7 times the clock edge, by starting the counter from 1, to have a 1/3 Hz frequency (3 seconds period)
    localparam division_value = 150000000; // 1/3 Hz

    logic [31:0] counter = 1; // 32 bit counter starting from 1 to go maximum of 15 x 10^7

    // Loop to generate our 3 secnds period clock
    always_ff@ (posedge clk) begin
        // If we reach the division value we toggle the output clock and reset the counter to start again
        if (counter == division_value) begin
            counter <= 1;
            clk_out <= ~clk_out;
        end

        // Increment the clock until reach the division value
        else
            counter <= counter + 1;
    end
endmodule
```

SystemVerilog Code for Traffic Light System and It's TestBench

```
`timescale 1ns / 1ps


// Low level FSM that simulates a traffic light controller
module trafficFSM(input logic clk, input logic reset, input logic sa, input logic sb, output
logic [2:0] la, output logic [2:0] lb);
    typedef enum logic [2:0] {S0, S1, S2, S3, S4, S5, S6, S7} statetype;
    statetype [2:0] currentState, nextState;

    // Parameters for the outputs
    parameter red = 3'b111;
    parameter green = 3'b011;
    parameter yellow = 3'b001;

    // State Register
    always_ff@ (posedge clk, posedge reset) begin
        if (reset)
            currentState <= S0;
        else
            currentState <= nextState;
    end

    // Next State Logic
    always_comb
        case(currentState)
            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
    always_comb
        case(currentState)
```

```
            S0: {la, lb} = {green, red};
            S1: {la, lb} = {yellow, red};
            S2: {la, lb} = {red, red};
            S3: {la, lb} = {red, yellow};
            S4: {la, lb} = {red, green};
            S5: {la, lb} = {red, yellow};
            S6: {la, lb} = {red, red};
            S7: {la, lb} = {yellow, red};
        endcase
endmodule


// TestBench for the Traffic Light Controller
module fsmTB();

    logic clock, reset;
    logic sa, sb;
    logic [2:0] la, lb;

    // Instantiating the device under test
    trafficFSM dut(clock, reset, sa, sb, la, lb);

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

    // Testing the FSM for the all possible input combinations
    initial begin
        // Testings
        reset = 1; sa = 0; sb = 0; #10;
        reset = 0; sa = 1; sb = 1; #10;
        sa = 1; sb = 0; #10;
        sa = 0; sb = 0; #10;
        sa = 0; sb = 1; #10;
        sa = 1; sb = 0; #10;
        sa = 1; sb = 1; #10;
        sa = 0; sb = 1; #10;
        sa = 0; sb = 1; #10;
        sa = 0; sb = 0; #10;
        sa = 1; sb = 1; #10;
        sa = 0; sb = 0; #10;
        sa = 1; sb = 0; #10;
    end
endmodule
```

SystemVerilog Code for the Combined Clocked Traffic Light Controller FSM

```
`timescale 1ns / 1ps


// Top level FSM that simulates the traffic light controller with the 3 seconds clock period
module clockedTrafficFSM(input logic clk, input logic reset, input logic sa, input logic sb,
output logic [2:0] la, output logic [2:0] lb);

    logic clk3sec; // Local signl that keeps the clock with 3 seconds period

    // Wrapper for the Clock Counter
    clockCounter wrapper(clk, clk3sec);

    // Generating the clocked FSM
    trafficFSM fsm(clk3sec, reset, sa, sb, la, lb);

endmodule
```