# CS 223 - Digital Design

Section 4

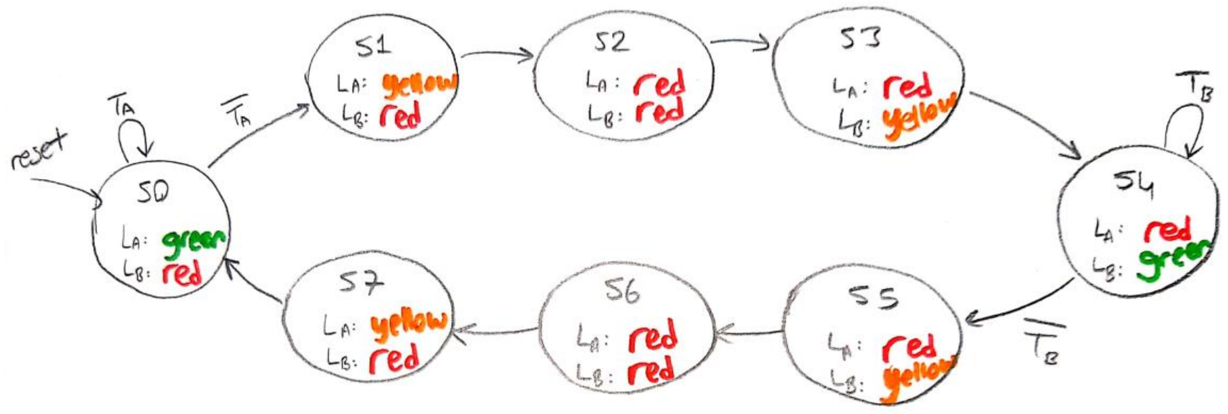
Melike Demirci 21702346

Lab05

26/04/2020

b) Number of flip-flops needed = 3 because there are 8 states ($2^3 = 8$)

c) *State Transition Diagram*



*State Transition Table*

| **Current State S** | **Inputs** | | **Next State S'** |
|---|---|---|---|
| | Ta | Tb | |
| S0 | 1 | X | S0 |
| S0 | 0 | X | S1 |
| S1 | X | X | S2 |
| S2 | X | X | S3 |
| S3 | X | X | S4 |
| S4 | X | 1 | S4 |
| S4 | X | 0 | S5 |
| S5 | X | X | S6 |
| S6 | X | X | S7 |
| S7 | X | X | S0 |

| **Current State S** | | | **Inputs** | | **Next State S'** | | | **Output** | |
|---|---|---|---|---|---|---|---|---|---|
| **$S_2$** | **$S_1$** | **$S_0$** | **Ta** | **Tb** | **$S_2'$** | **$S_1'$** | **$S_0'$** | **La** | **Lb** |
| 0 | 0 | 0 | 1 | X | 0 | 0 | 0 | 111 | 100 |
| 0 | 0 | 0 | 0 | X | 0 | 0 | 1 | 111 | 100 |
| 0 | 0 | 1 | X | X | 0 | 1 | 0 | 110 | 100 |
| 0 | 1 | 0 | X | X | 0 | 1 | 1 | 100 | 100 |
| 0 | 1 | 1 | X | X | 1 | 0 | 0 | 100 | 110 |
| 1 | 0 | 0 | X | 1 | 1 | 0 | 0 | 100 | 111 |
| 1 | 0 | 0 | X | 0 | 1 | 0 | 1 | 100 | 111 |
| 1 | 0 | 1 | X | X | 1 | 1 | 0 | 100 | 110 |
| 1 | 1 | 0 | X | X | 1 | 1 | 1 | 100 | 100 |
| 1 | 1 | 1 | X | X | 0 | 0 | 0 | 110 | 100 |

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

| Output | Encoding |
|---|---|
| Green | 111 |
| Yellow | 110 |
| Red | 100 |

**Next State:**

$S_2' = S_2 S_0' + S_2 S_1' + S_2' S_1 S_0$

$S_1' = S_1 \oplus S_0$

$S_3' = S_1 S_0' + S_2' S_1' S_0' T_a' + S_2 S_1' S_0' T_b'$

**Output:**

| | |
|---|---|
| $L_{a0} = S_2' S_1' S_0'$ | $L_{b0} = S_2 S_1' S_0'$ |
| $L_{a1} = S_2' S_1' + S_2 S_1 S_0$ | $L_{b1} = S_2 S_1' + S_2' S_1 S_0$ |
| $L_{a2} = 1$ | $L_{b2} = 1$ |

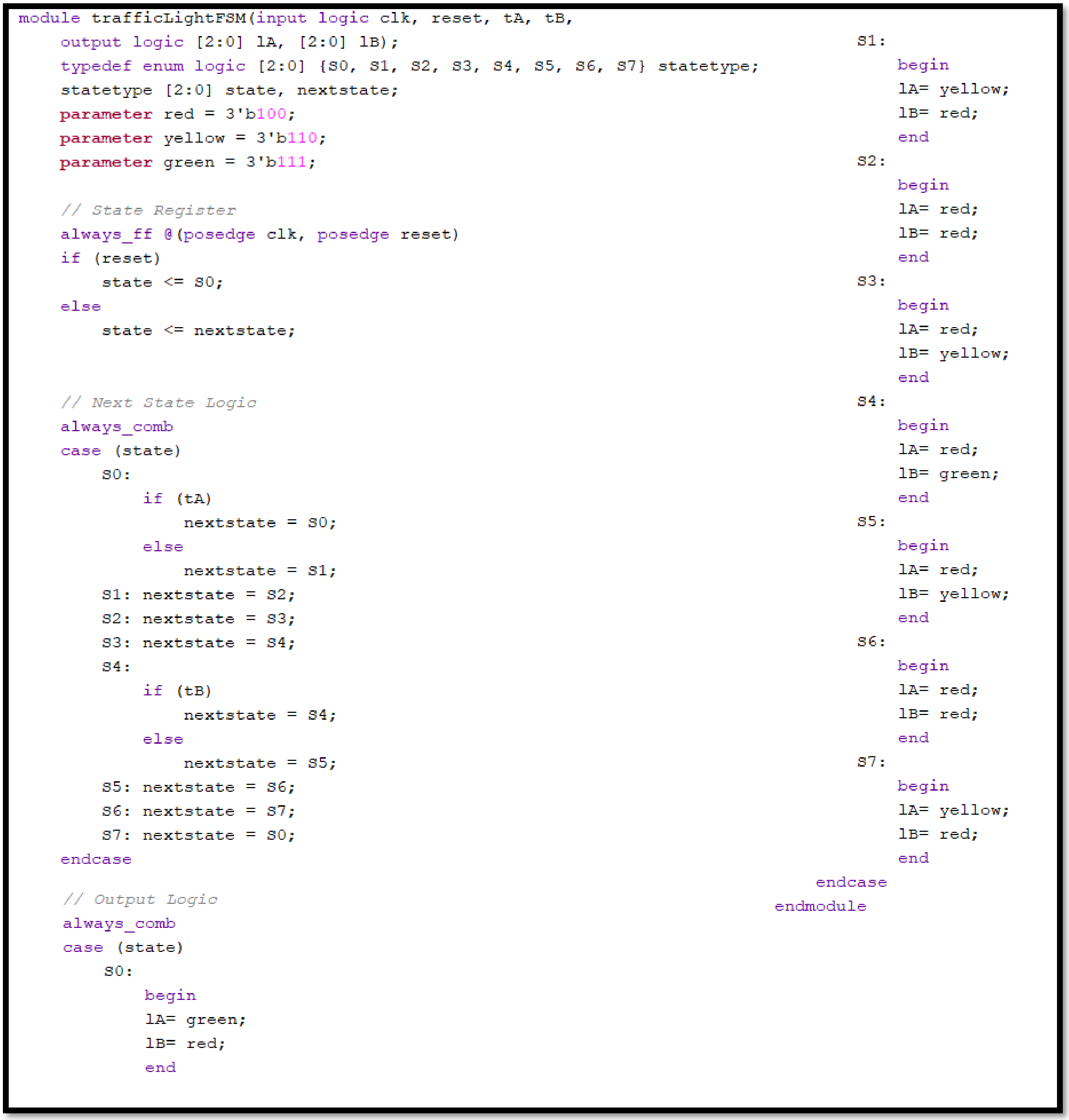
d)

```
module trafficLight_forFSM(input logic clk, reset, tA, tB,
    output logic [2:0] lA, [2:0] lB);

logic clk2;
clockDivider cd(clk, clk2);
trafficLightFSM fsm(clk2, reset, tA, tB, lA, lB);

endmodule
```

```
module clockDivider(input logic clk,
output logic slowerClk);
logic [27:0] clk_counter;
always @(posedge clk)
    begin
        if(clk_counter >= 199999999)
        clk_counter <= 0;
    else
        clk_counter <= clk_counter + 1;
    end
assign slowerClk =(clk_counter == 199999999)? 1:0;
endmodule
```

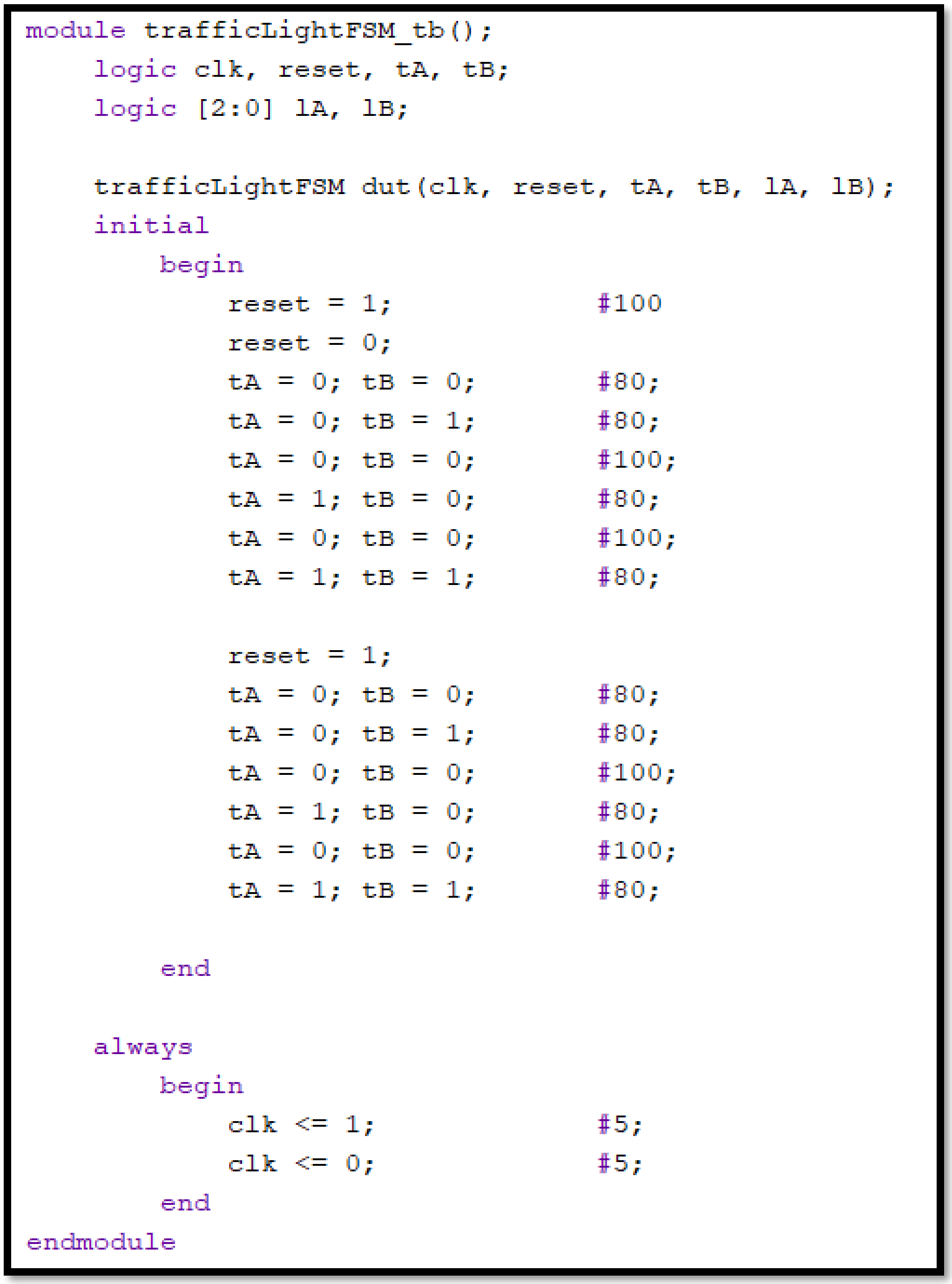


```
module trafficLightFSM(input logic clk, reset, tA, tB,
    output logic [2:0] lA, [2:0] lB);
    typedef enum logic [2:0] {S0, S1, S2, S3, S4, S5, S6, S7} statetype;
    statetype [2:0] state, nextstate;
    parameter red = 3'b100;
    parameter yellow = 3'b110;
    parameter green = 3'b111;

    // State Register
    always_ff @(posedge clk, posedge reset)
    if (reset)
        state <= S0;
    else
        state <= nextstate;


    // Next State Logic
    always_comb
    case (state)
        S0:
            if (tA)
                nextstate = S0;
            else
                nextstate = S1;
        S1: nextstate = S2;
        S2: nextstate = S3;
        S3: nextstate = S4;
        S4:
            if (tB)
                nextstate = S4;
            else
                nextstate = S5;
        S5: nextstate = S6;
        S6: nextstate = S7;
        S7: nextstate = S0;
    endcase

    // Output Logic
    always_comb
    case (state)
        S0:
            begin
            lA= green;
            lB= red;
            end
        S1:
            begin
            lA= yellow;
            lB= red;
            end
        S2:
            begin
            lA= red;
            lB= red;
            end
        S3:
            begin
            lA= red;
            lB= yellow;
            end
        S4:
            begin
            lA= red;
            lB= green;
            end
        S5:
            begin
            lA= red;
            lB= yellow;
            end
        S6:
            begin
            lA= red;
            lB= red;
            end
        S7:
            begin
            lA= yellow;
            lB= red;
            end
    endcase
endmodule
```



```
module trafficLightFSM_tb();
    logic clk, reset, tA, tB;
    logic [2:0] lA, lB;

    trafficLightFSM dut(clk, reset, tA, tB, lA, lB);
    initial
        begin
            reset = 1;              #100
            reset = 0;
            tA = 0; tB = 0;         #80;
            tA = 0; tB = 1;         #80;
            tA = 0; tB = 0;         #100;
            tA = 1; tB = 0;         #80;
            tA = 0; tB = 0;         #100;
            tA = 1; tB = 1;         #80;

            reset = 1;
            tA = 0; tB = 0;         #80;
            tA = 0; tB = 1;         #80;
            tA = 0; tB = 0;         #100;
            tA = 1; tB = 0;         #80;
            tA = 0; tB = 0;         #100;
            tA = 1; tB = 1;         #80;

        end

    always
        begin
            clk <= 1;               #5;
            clk <= 0;               #5;
        end
endmodule
```