Traffic Light Controller

Prepared by

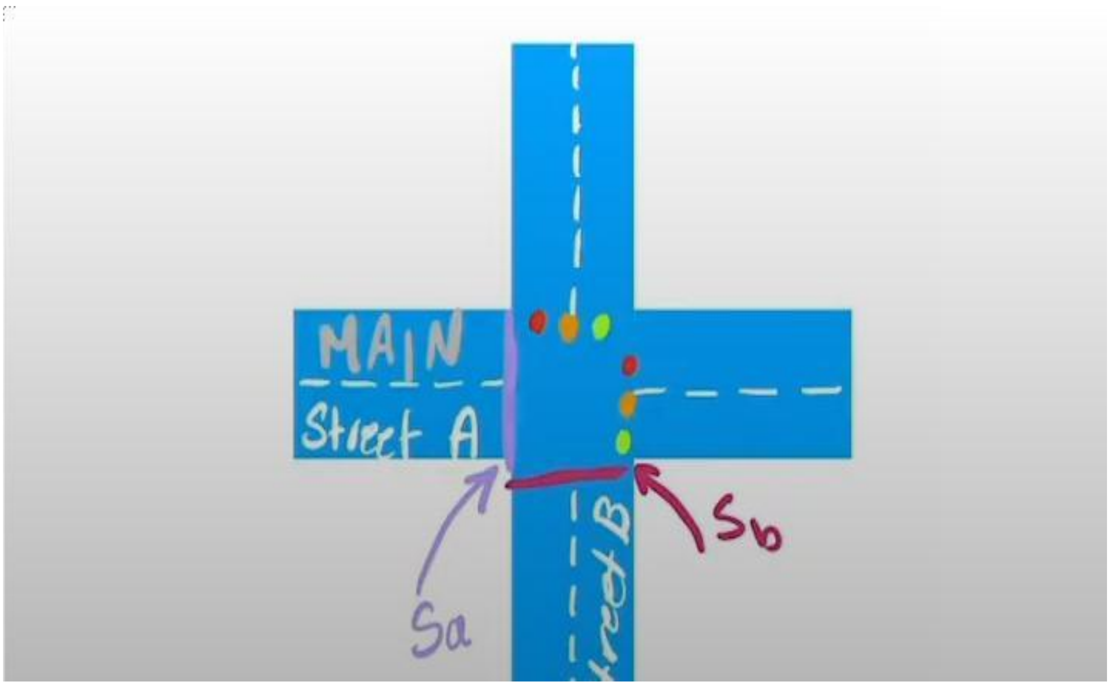
Ali Abdel Aleem Ali

Overview:

This project presents the design and implementation of a Traffic Light Controller based on a Finite State Machine (FSM) model. The controller is developed to manage traffic flow at a typical intersection by providing systematic switching between red, yellow, and green lights. Using FSM ensures a structured representation of states, transitions, and timing sequences, enabling efficient and predictable operation. The design is modeled in Verilog HDL and thoroughly verified through a testbench, which simulates different traffic scenarios to validate timing accuracy, correctness of state transitions, and overall system reliability. This FSM-based approach demonstrates how digital design techniques can be effectively applied to real-world traffic management systems, ensuring safety, robustness, and scalability for more complex intersections.

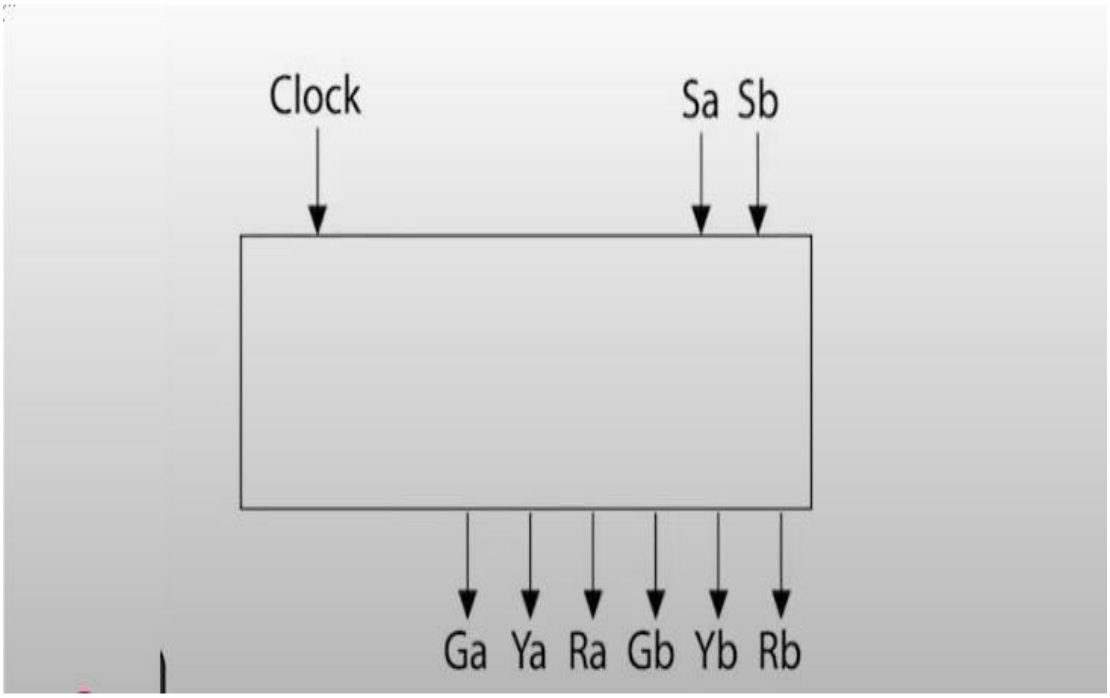
Design Procedure:

Design a traffic light controller for the intersection of Street A and Street B.
The controller uses a single synchronous clock domain. A clock divider produces a low-frequency tick, that enables the FSM to advance only once per tick, so green/yellow/red duration's map directly to real seconds. A synchronous, active-high reset initializes the system, and any asynchronous inputs are passed through a two-flip-flop synchronizer. This scheme improves timing predictability, simplifies simulation/testbench checks, and avoids metastability issues.



Each street has traffic sensors, which detect the presence of vehicles approaching or stopped at the intersection.

The controller has 6 outputs (Ga, Ya, and Ra) drive the green, yellow and red lights on street A, and (Gb, Yb, and Rb) drive the corresponding lights on street B.



specification:

- If Sa = 1 means a vehicle approaching on street A.

- Sb = 1 means a vehicle is approaching on street B

.

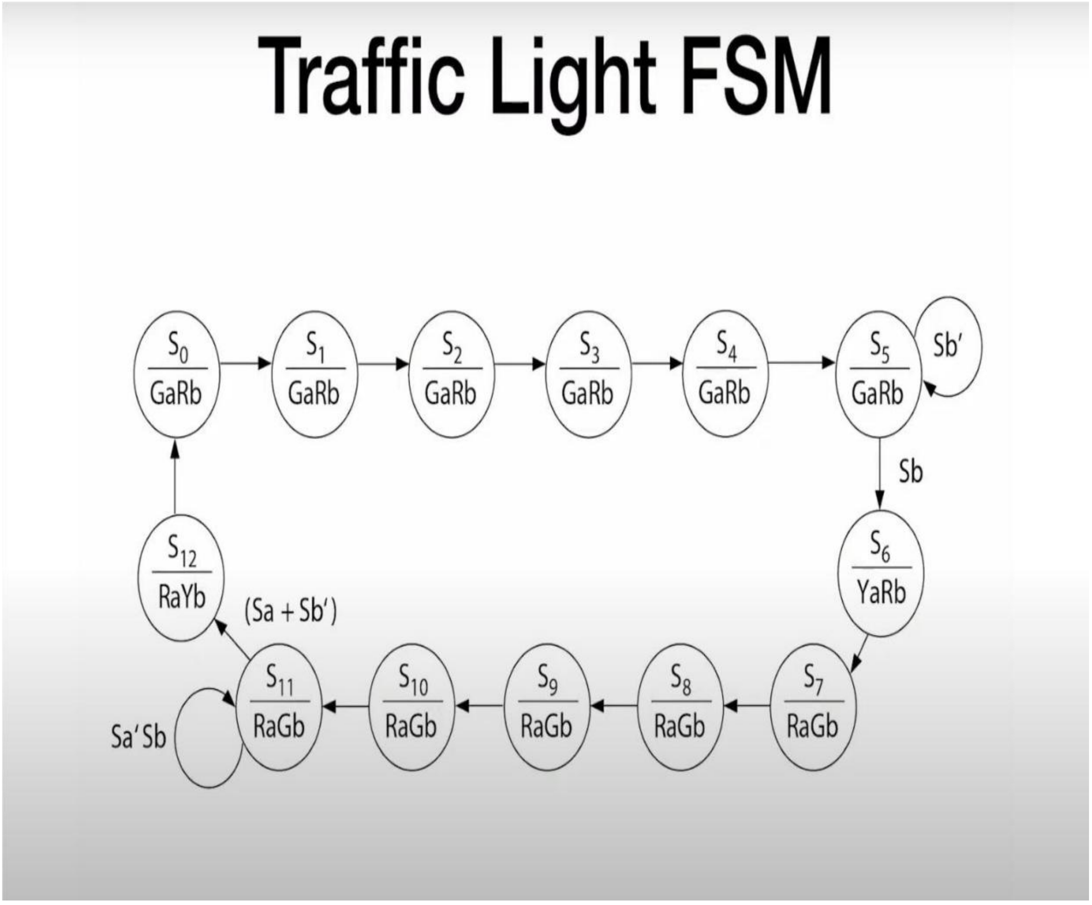
- Street A is a main street and has a green light until a car approaches approaches on B, then the lights change, and B has a green light .

- At the end of 50 seconds, the lights change back unless there is a car on street B and none on A, in which case the B cycle is extended for 10 additional seconds.

- If cars continue to arrive on street B and no car appears on street A, B continues to have a green light.

- When A is green, it remains green at least 60 seconds, and then the lights change only when a car approaches on B.

FSM State Diagram:



Inputs and Outputs ports:

```
module traffic_light_controller(
    input clk, reset_n,
    input Sa, Sb,
    output reg Ra, Ya, Ga,
    output reg Rb, Yb, Gb
    );


    reg [3:0] state_reg, state_next;
    localparam  s0 = 0, s1 = 1, s2 = 2, s3 = 3,
                s4 = 4, s5 = 5, s6 = 6, s7 = 7,
                s8 = 8, s9 = 9, s10 = 10, s11 = 11,
                s12 = 12;
```

Sequential State:

```
    // Sequential state register
    always @(posedge clk, negedge reset_n)
    begin
        if (~reset_n)
            state_reg <= s0;
        else
            state_reg <= state_next;
    end
```

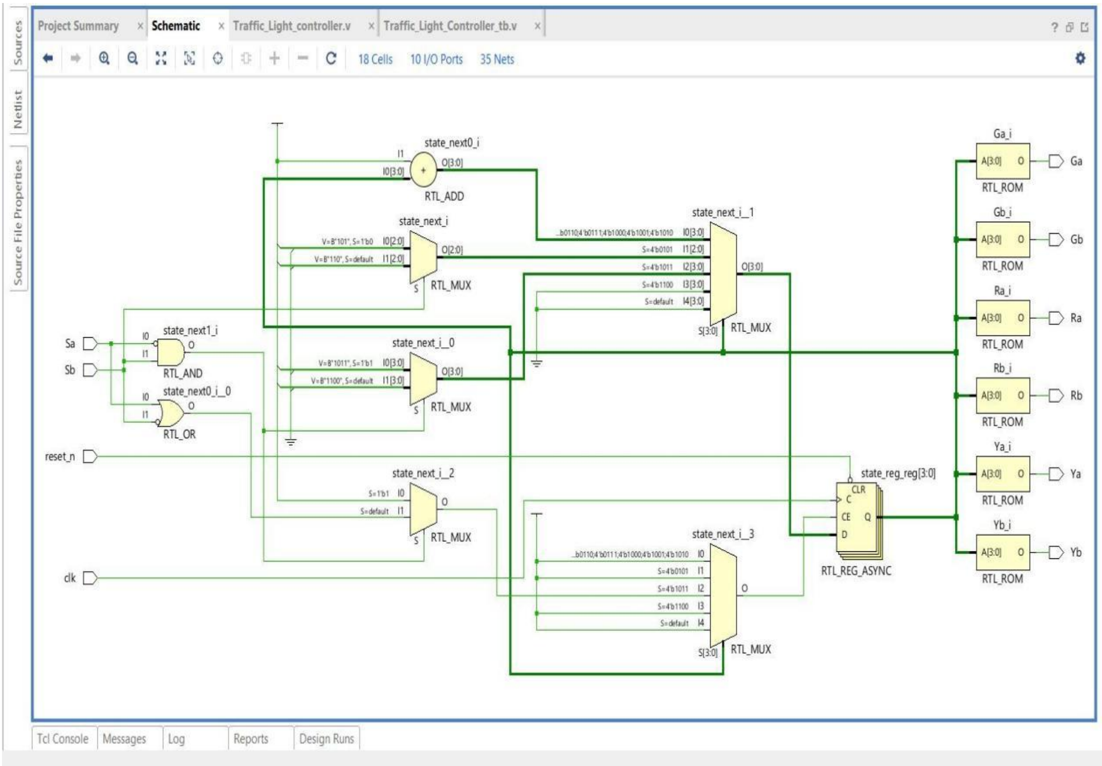
Next State:

```
    // Next state logic
    always @(*)
    begin
        state_next = state_reg;
        case(state_reg)
            s0, s1, s2, s3, s4, s6, s7, s8, s9, s10:
                state_next = state_reg + 1;
            s5: if (~Sb)
                    state_next = s5;
                else
                    state_next = s6;
            s11: if (~Sa & Sb)
                    state_next = s11;
                 else if(Sa | ~Sb)
                    state_next = s12;
            s12: state_next = s0;
            default: state_next = s0;
        endcase
    end
```

Output Logic:

```verilog
    // Output logic
    always @(*)
    begin
        Ra = 1'b0;
        Ya = 1'b0;
        Ga = 1'b0;
        Rb = 1'b0;
        Yb = 1'b0;
        Gb = 1'b0;
        case(state_reg)
            s0, s1, s2, s3, s4, s5:
                begin
                    Ga = 1'b1;
                    Rb = 1'b1;
                end
            s6:
                begin
                    Ya = 1'b1;
                    Rb = 1'b1;
                end
            s7, s8, s9, s10, s11:
                begin
                    Ra = 1'b1;
                    Gb = 1'b1;
                end
            s12:
                begin
                    Ra = 1'b1;
                    Yb = 1'b1;
                end
        endcase
    end
endmodule
```

RTL analysis:



Testbench parameter:

```
module traffic_light_controller_tb(

    );

    reg clk, reset_n, Sa, Sb;
    wire Ra, Ya, Ga;
    wire Rb, Yb, Gb;


    // Instantiate unit under test
    traffic_light_controller CTRL0(
        .clk(clk),
        .reset_n(reset_n),
        .Sa(Sa),
        .Sb(Sb),
        .Ra(Ra),
        .Ya(Ya),
        .Ga(Ga),
        .Rb(Rb),
        .Yb(Yb),
        .Gb(Gb)
    );

```

Assign new values to simpler the outputs:

```
    // To make the output of the simulator simpler to read
    // reduce the 6 different lights into light_A and light_B
    // with values R, Y, G
    wire [1:0] light_A, light_B;

    localparam  R = 0;
    localparam  Y = 1;
    localparam  G = 2;

    assign light_A = Ra? R: Ya ? Y: Ga? G: light_A;
    assign light_B = Rb? R: Yb ? Y: Gb? G: light_B;


```

Generate Clock:

```
// Generate stimuli

// Generating a clk signal
localparam T = 10;
S
begin
    clk = 1'b0;
    #(T / 2);
    clk = 1'b1;
    #(T / 2);
end
```

Reset condition and no cars in both street A, B:

```
end

initial
begin
    // issue a quick reset for 2 ns
    reset_n = 1'b0;
    #2
    reset_n = 1'b1;

    // No cars at either streets
    Sa = 0;
    Sb = 0;
```

No cars at A, there is a car in B:

```
        Sb = 0;

        #80;

        // No cars at A, Some cars at B
        Sa = 0;
        Sb = 1;
        #100
```

Cars at both streets:

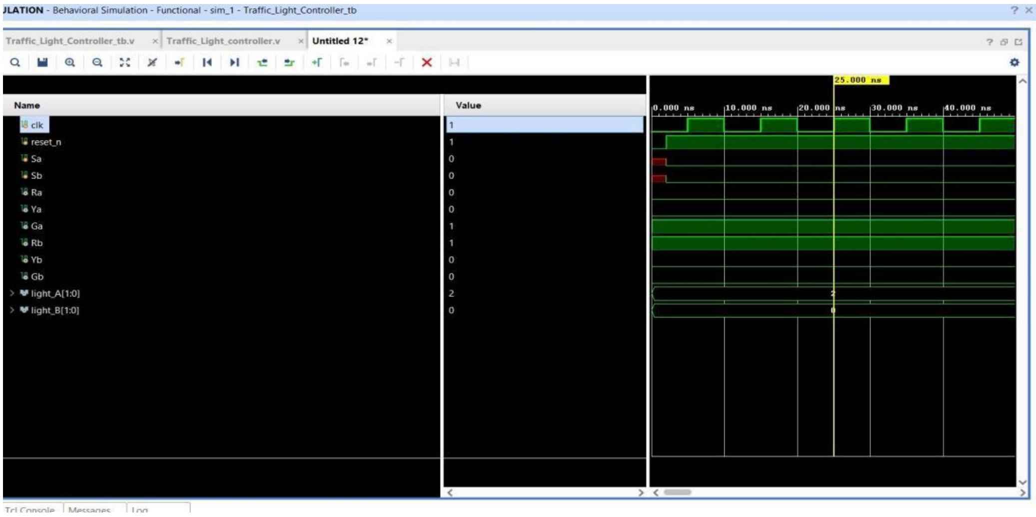
```

        // Cars at both streets
        Sa = 1;
        Sb = 1;
        #300;
```

Car appears at B, then no cars at either streets:

```
        #300;

        // Car appears at B, then no cars at either streets
        Sa = 0;
        Sb = 1;
        #20;
        Sb = 0;
        Sa = 0;

        #300 $stop;
    end

endmodule

```

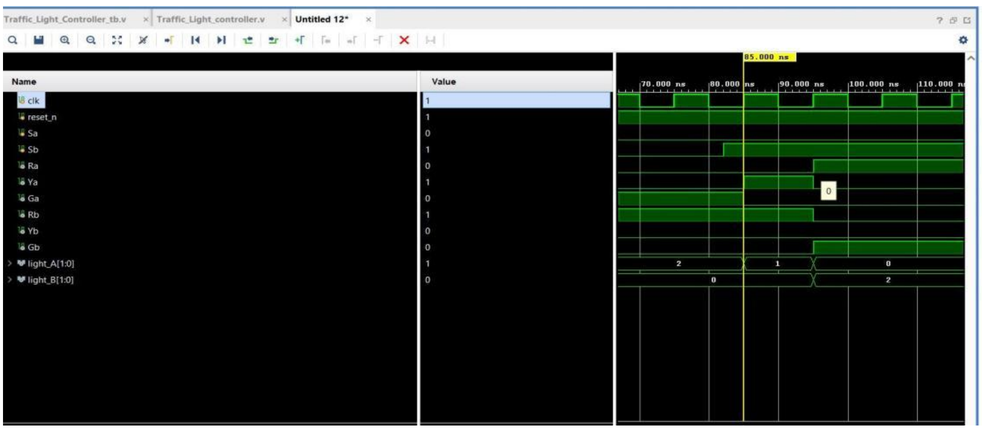
Simulation:

- At the beginning no cars on both streets but A is main Street.



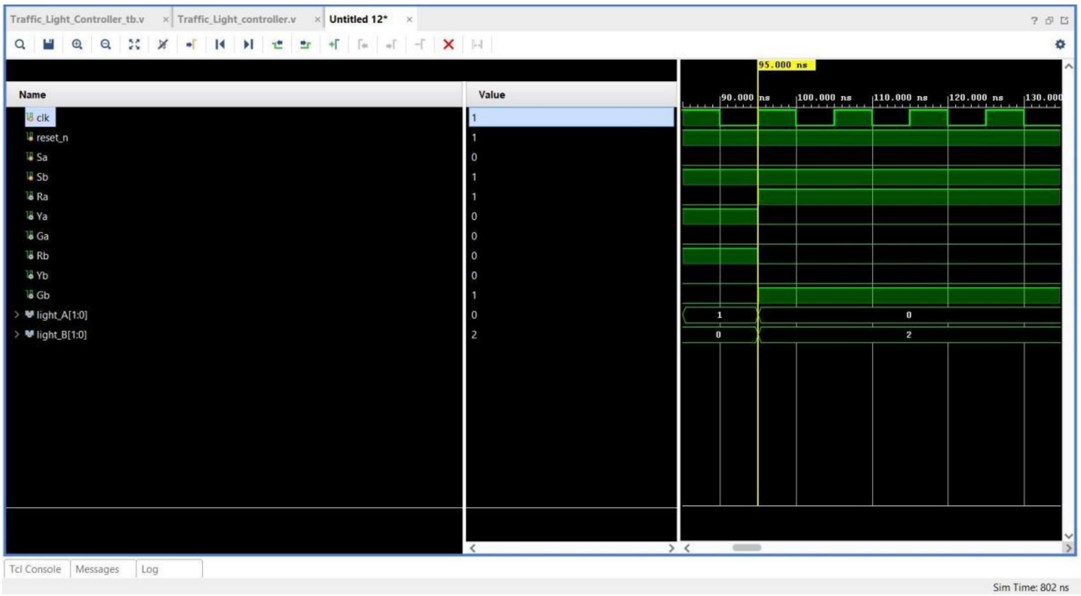
So: GA√ and RB√

- After, cars approach B:



So, YA√ and RB√

- No cars at A, car approach B:



So, RA√ and GB√