Traffic Light Controller

Prepared by

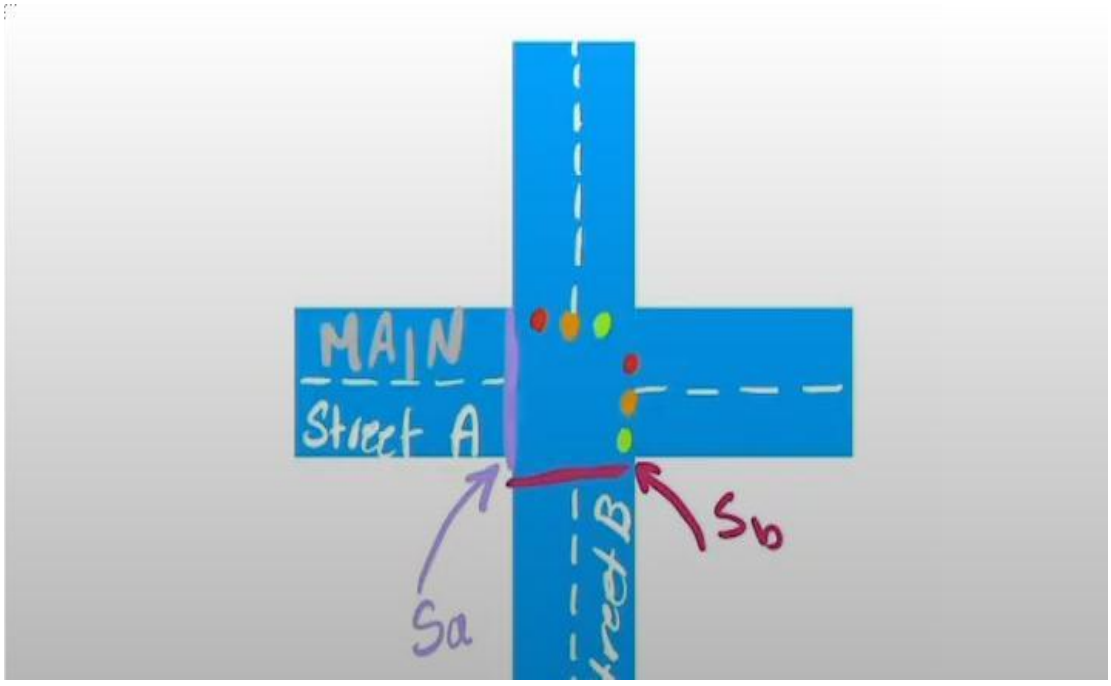
Ali Abdel Aleem Ali

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.

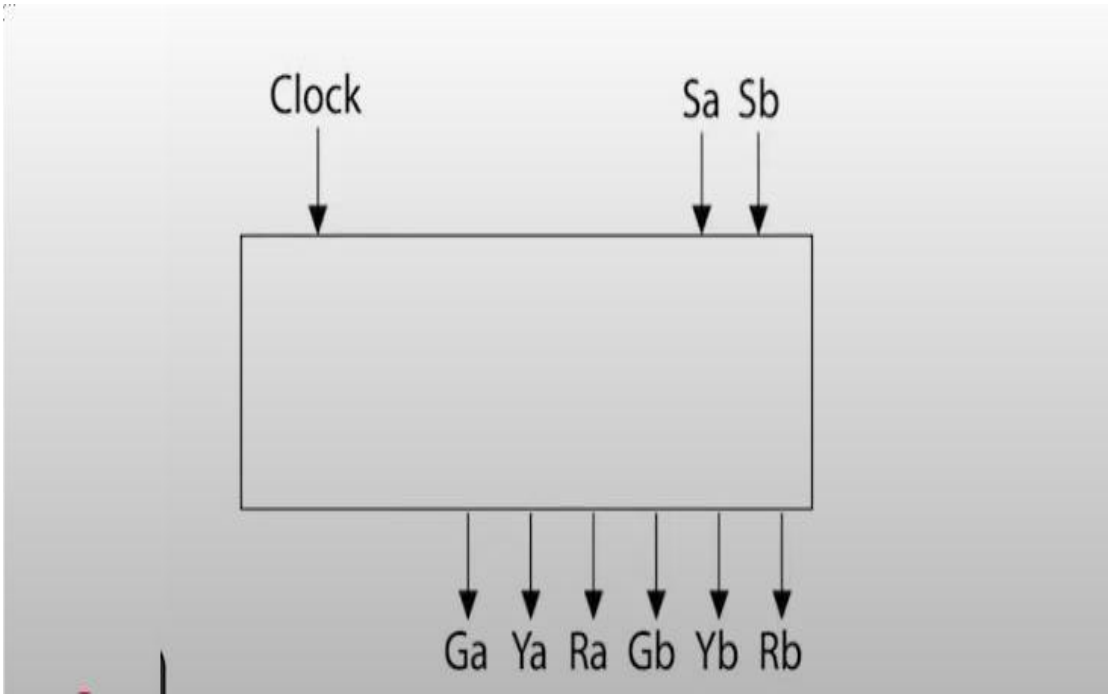
Design Procedure:

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

.

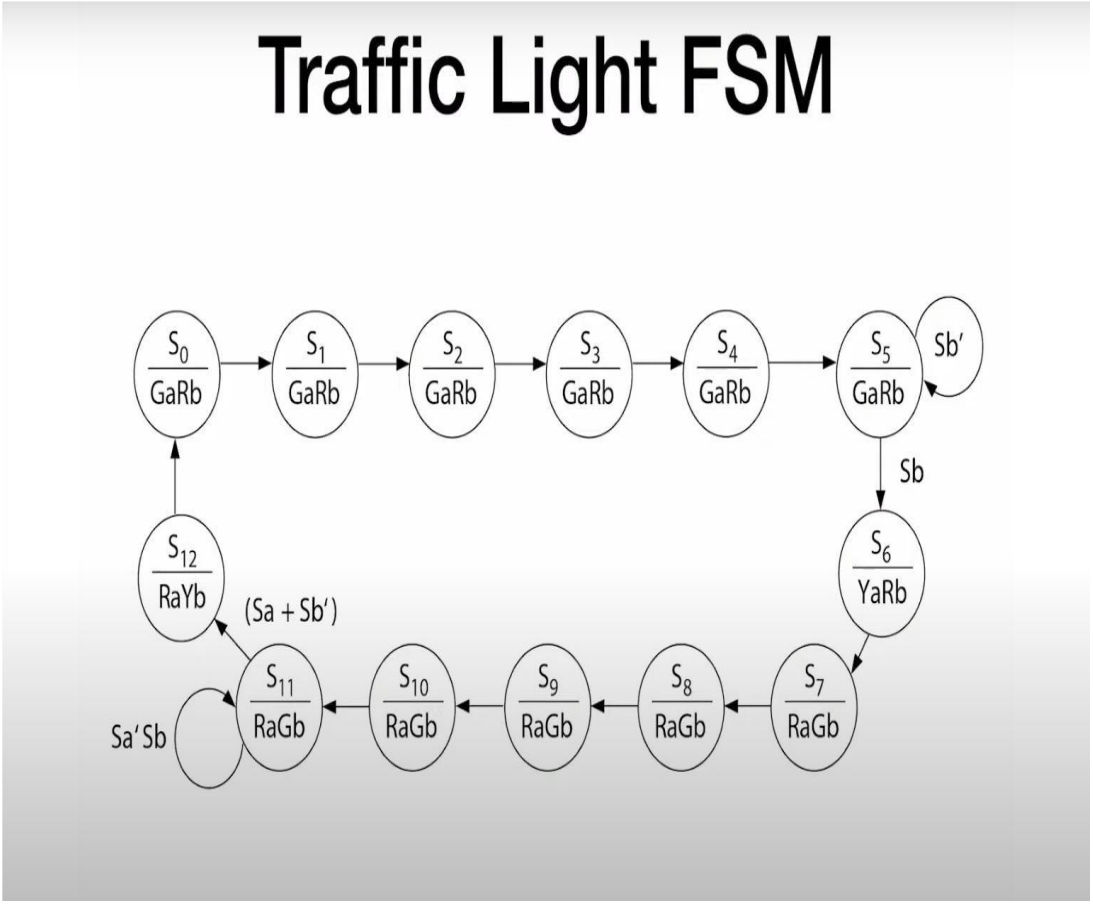
- Street A is a main street and has a green light until a car approaches approaches on B, then the lights change, and B has a green light .

- 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:

```
21
22
23  module traffic_light_controller(
24      input clk, reset_n,
25      input Sa, Sb,
26      output reg Ra, Ya, Ga,
27      output reg Rb, Yb, Gb
28      );
29
30
31      reg [3:0] state_reg, state_next;
32      localparam  s0 = 0, s1 = 1, s2 = 2, s3 = 3,
33                  s4 = 4, s5 = 5, s6 = 6, s7 = 7,
34                  s8 = 8, s9 = 9, s10 = 10, s11 = 11,
35                  s12 = 12;
36
```

Sequential State:

```
36
37      // Sequential state register
38      always @(posedge clk, negedge reset_n)
39      begin
40          if (~reset_n)
41              state_reg <= s0;
42          else
43              state_reg <= state_next;
44      end
```

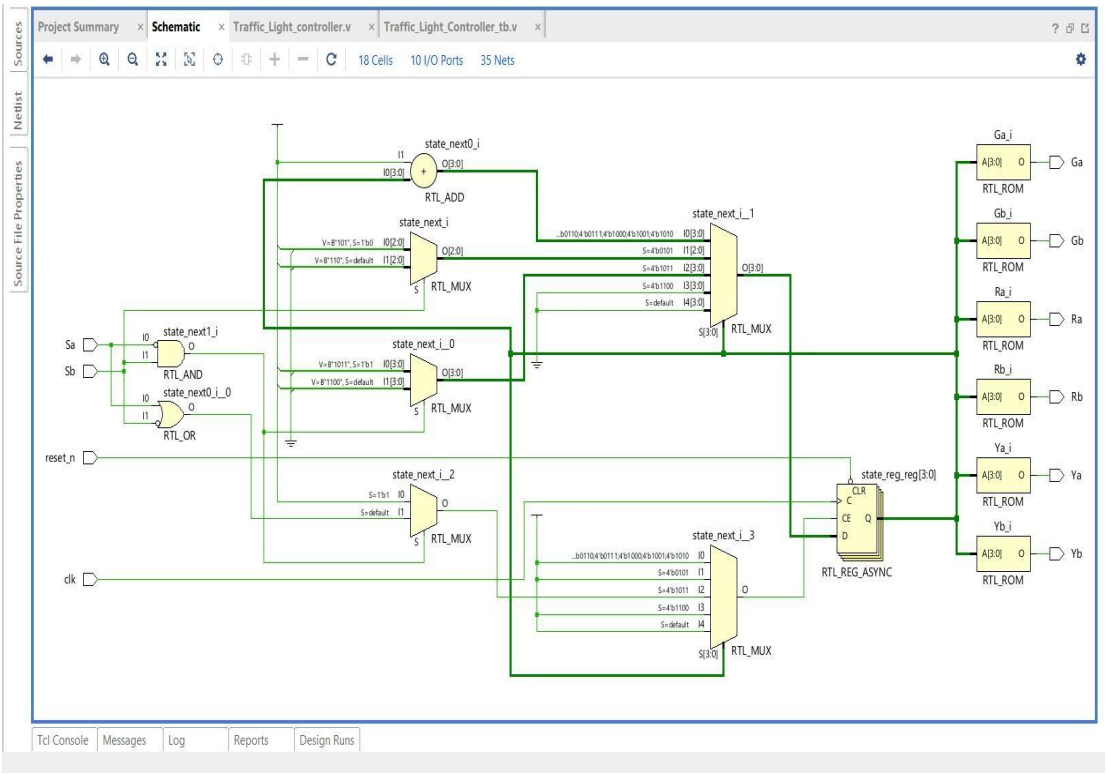
Next State:

```
    // 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:

```
    // 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:

```
23  module traffic_light_controller_tb(
24
25      );
26
27      reg clk, reset_n, Sa, Sb;
28      wire Ra, Ya, Ga;
29      wire Rb, Yb, Gb;
30
31
32      // Instantiate unit under test
33      traffic_light_controller CTRL0(
34          .clk(clk),
35          .reset_n(reset_n),
36          .Sa(Sa),
37          .Sb(Sb),
38          .Ra(Ra),
39          .Ya(Ya),
40          .Ga(Ga),
41          .Rb(Rb),
42          .Yb(Yb),
43          .Gb(Gb)
44      );
45
```

Assign new values to simpler the outputs:

```
46      // To make the output of the simulator simpler to read
47      // reduce the 6 different lights into light_A and light_B
48      // with values R, Y, G
49      wire [1:0] light_A, light_B;
50
51      localparam  R = 0;
52      localparam  Y = 1;
53      localparam  G = 2;
54
55      assign light_A = Ra? R: Ya ? Y: Ga? G: light_A;
56      assign light_B = Rb? R: Yb ? Y: Gb? G: light_B;
57
58
```

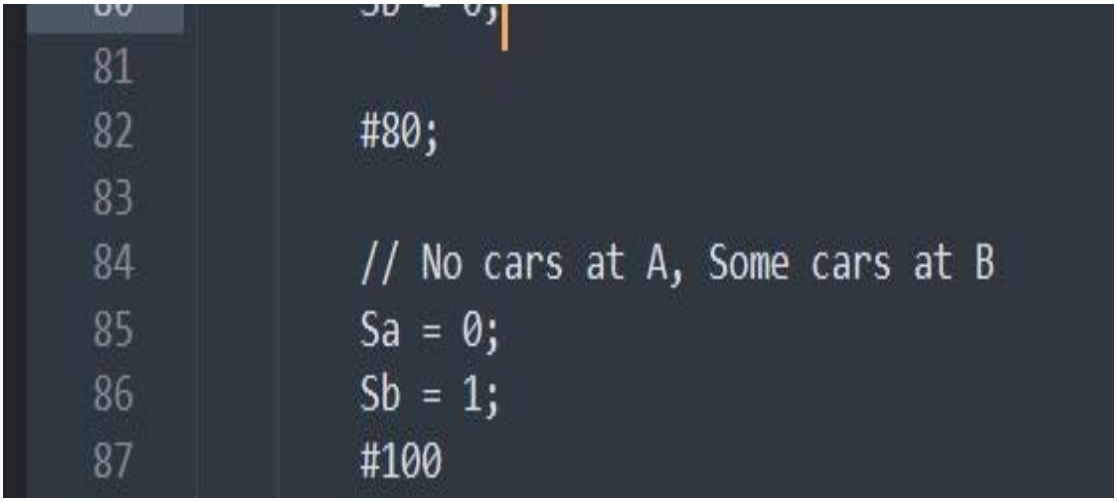
Generate Clock

```
58
59      // Generate stimuli
60
61      // Generating a clk signal
62      localparam T = 10;
63      S
64      begin
65          clk = 1'b0;
66          #(T / 2);
67          clk = 1'b1;
68          #(T / 2);
69      end
```

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

```
69      end
70
71      initial
72      begin
73          // issue a quick reset for 2 ns
74          reset_n = 1'b0;
75          #2
76          reset_n = 1'b1;
77
78          // No cars at either streets
79          Sa = 0;
80          Sb = 0;
```

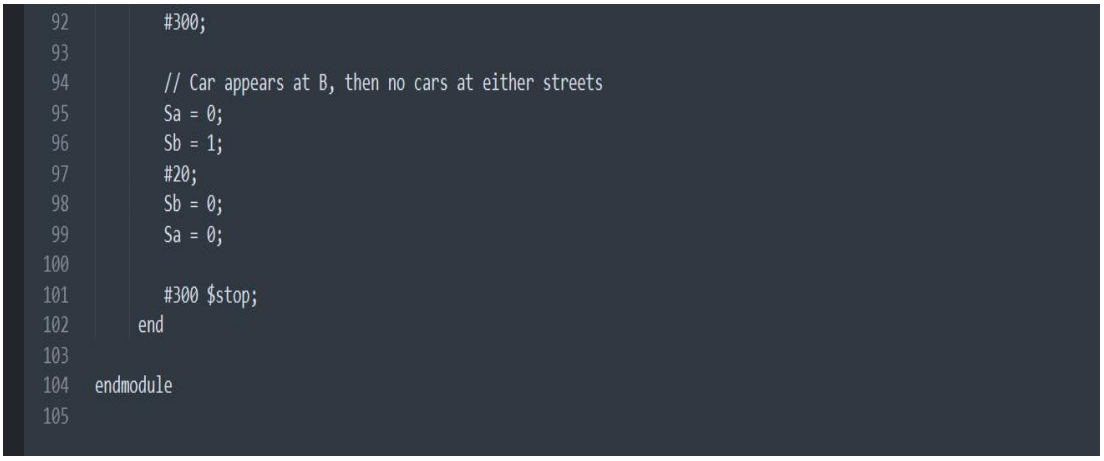
No cars at A, there is a car in B:



Cars at both streets:

```
88
89          // Cars at both streets
90          Sa = 1;
91          Sb = 1;
92          #300;
```

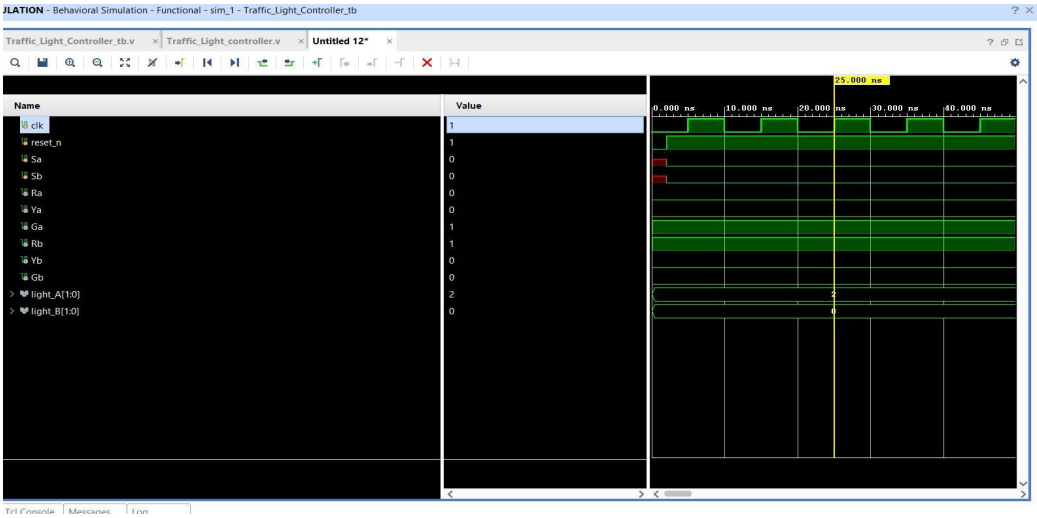
Car appears at B, then no cars at either streets:



```
92          #300;
93
94          // Car appears at B, then no cars at either streets
95          Sa = 0;
96          Sb = 1;
97          #20;
98          Sb = 0;
99          Sa = 0;
100
101         #300 $stop;
102       end
103
104   endmodule
105
```

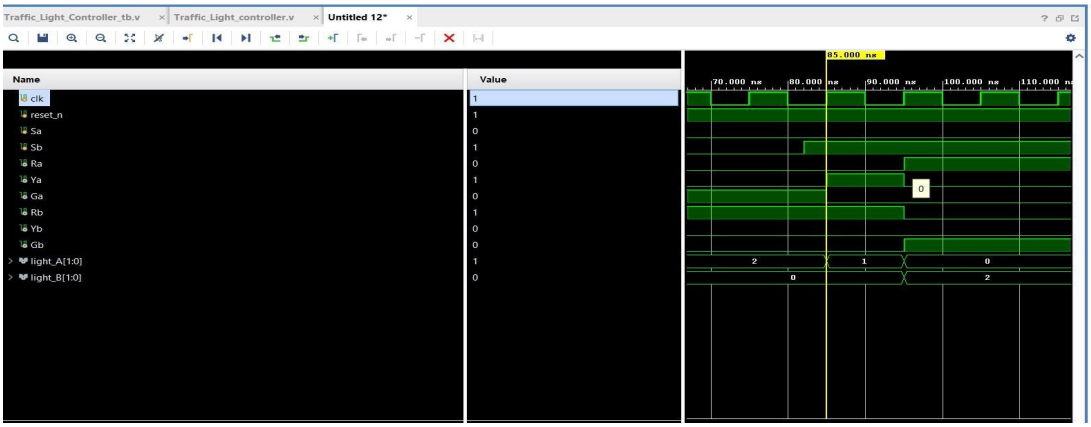
Simulation:

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



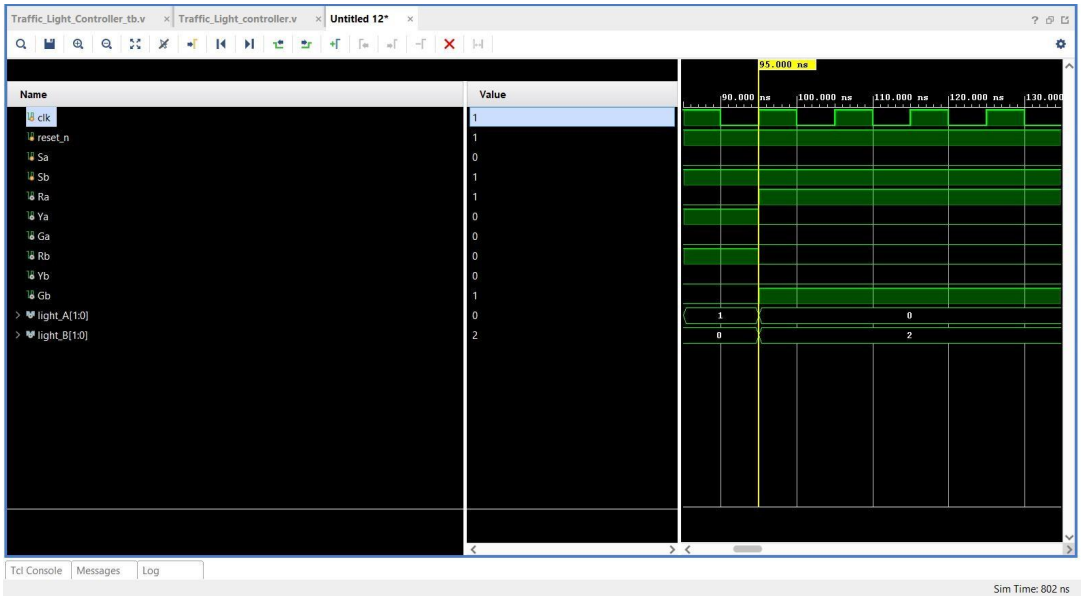
So: GA√ and RB√

- After, cars approach B:



So, YA√ and RB√

- No cars at A, car approach B:



So, RA√ and GB√