Traffic Light Controller using FSM in Verilog

Project Report

Submitted by: Kalai Kumar M

Tools: Verilog, GTKWave, Icarus Verilog, VS Code

1. Introduction

This project implements a simple Traffic Light Controller using a Finite State Machine (FSM) in Verilog HDL.

The system cycles through Red, Green, and Yellow lights based on clock cycles. The FSM logic is written in Verilog,

and the simulation is performed using Icarus Verilog and visualized in GTKWave.

2. FSM Design Explanation

The FSM has four states: RED, GREEN, YELLOW, and RESET. The state transitions occur based on clock signals.

Each state corresponds to a specific output value on the `lights[2:0]` bus:

- 3'b100 -> RED

- 3'b010 -> YELLOW

- 3'b001 -> GREEN

Transitions: RED -> GREEN -> YELLOW -> RED

3. Verilog Code

FSM Module

```
module traffic_light(input clk, input reset, output reg [2:0] lights);
    reg [1:0] state;
    parameter RED=2'b00, GREEN=2'b01, YELLOW=2'b10;

    always @(posedge clk or posedge reset) begin
        if (reset)
            state <= RED;
        else begin
            case (state)
                RED: state <= GREEN;
                GREEN: state <= YELLOW;
                YELLOW: state <= RED;
                default: state <= RED;
            endcase
        end
    end

    always @(*) begin
        case (state)
            RED: lights = 3'b100;
            GREEN: lights = 3'b001;
            YELLOW: lights = 3'b010;
            default: lights = 3'b000;
        endcase
    end
endmodule
```

Testbench Module

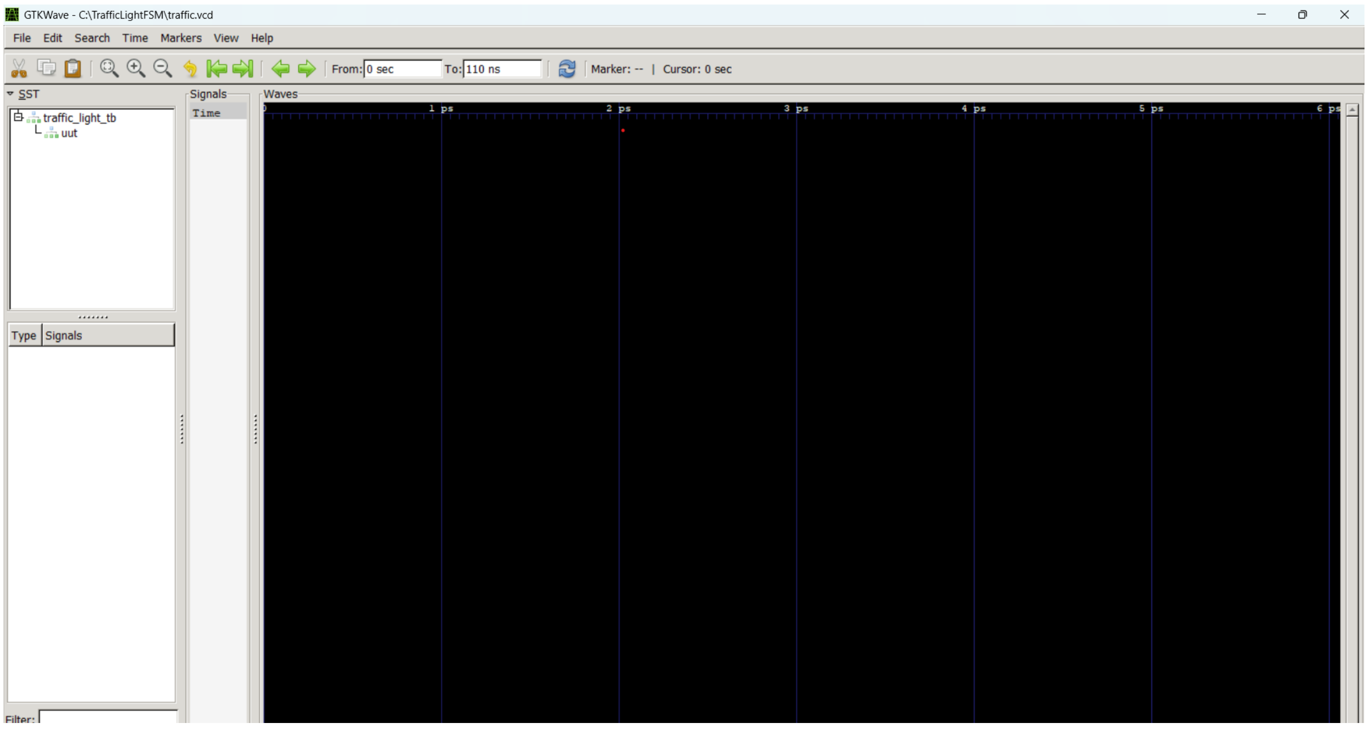
```
module traffic_light_tb;
    reg clk, reset;
    wire [2:0] lights;

    traffic_light uut (.clk(clk), .reset(reset), .lights(lights));

    initial begin
        $dumpfile("traffic.vcd");
        $dumpvars(0, traffic_light_tb);
        clk = 0; reset = 1;
        #5 reset = 0;
        #100 $finish;
    end

    always #5 clk = ~clk;
endmodule
```

4. Simulation Result (GTKWave)


GTKWave - C:\TrafficLightFSM\traffic.vcd

From: 0 sec To: 110 ns Marker: -- | Cursor: 0 sec

The waveform shows the clk, reset, state, and lights[2:0] signals. The output lights transitions through

001 (Green), 010 (Yellow), 100 (Red) repeatedly, matching the expected FSM behavior.

5. Conclusion

This project demonstrated the design and simulation of a traffic light controller using FSM in Verilog. The simulation results confirmed the correct behavior of the FSM and the output signal transitions. This practical exercise helped understand RTL design, state machines, and waveform analysis using GTKWave.