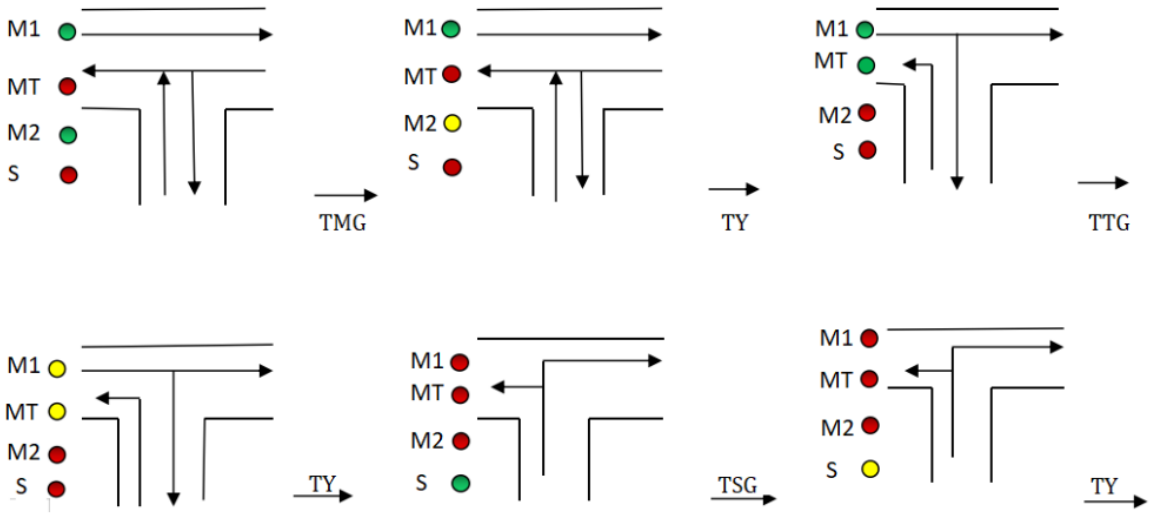
Traffic light Controller using Verilog

Introduction

Traffic control is a major challenge in urban areas due to high vehicle density and dynamic traffic patterns, leading to accidents and time losses. This project proposes a Verilog HDL-based design to optimize waiting times at signals. Verilog simplifies hardware design by reducing errors associated with manual circuit assembly, enhancing efficiency in simulation and debugging. A T-shaped road traffic light system was designed and tested using, employing FPGA for its speed, scalability, and reconfigurability. The system addresses traffic jams and inefficiencies by dynamically adjusting signal delays based on real-time traffic levels.

Problem statement:

- Green light indicates that there is no traffic and there is easy flow of vehicles in that route/direction.
- Red light indicates that there is a traffic jam and that route is blocked for the vehicles to move and,
- Yellow light indicates that the route has medium flow of vehicles.



Verilog code:

```
`timescale 1ns / 1ps


module Traffic_Light_Controller(
    input clk,
```

```
    input rst,
    output reg [2:0] light_M1,
    output reg [2:0] light_S,
    output reg [2:0] light_MT,
    output reg [2:0] light_M2
);

    // State parameters
    parameter S1 = 0, S2 = 1, S3 = 2, S4 = 3, S5 = 4, S6 = 5;
    reg [2:0] ps; // Present state
    reg [3:0] count; // Counter for timing
    parameter sec7 = 7, sec5 = 5, sec2 = 2, sec3 = 3; // Timing constants

    // State transition block
    always @(posedge clk or posedge rst) begin
        if (rst == 1) begin
            ps <= S1;
            count <= 0;
        end else begin
            case (ps)
                S1: if (count < sec7) begin
                        ps <= S1;
                        count <= count + 1;
                    end else begin
                        ps <= S2;
                        count <= 0;
                    end
```

```
S2: if (count < sec2) begin
        ps <= S2;
        count <= count + 1;
    end else begin
        ps <= S3;
        count <= 0;
    end
S3: if (count < sec5) begin
        ps <= S3;
        count <= count + 1;
    end else begin
        ps <= S4;
        count <= 0;
    end
S4: if (count < sec2) begin
        ps <= S4;
        count <= count + 1;
    end else begin
        ps <= S5;
        count <= 0;
    end
S5: if (count < sec3) begin
        ps <= S5;
        count <= count + 1;
    end else begin
        ps <= S6;
        count <= 0;
```

```
            end
        S6: if (count < sec2) begin
                ps <= S6;
                count <= count + 1;
            end else begin
                ps <= S1;
                count <= 0;
            end
        default: ps <= S1;
    endcase
  end
end

// Output control block
always @(ps) begin
  case (ps)
    S1: begin
      light_M1 <= 3'b001; // Green
      light_M2 <= 3'b001; // Green
      light_MT <= 3'b100; // Red
      light_S  <= 3'b100; // Red
    end
    S2: begin
      light_M1 <= 3'b001; // Green
      light_M2 <= 3'b010; // Yellow
      light_MT <= 3'b100; // Red
      light_S  <= 3'b100; // Red
```

```
end
S3: begin
   light_M1 <= 3'b001; // Green
   light_M2 <= 3'b100; // Red
   light_MT <= 3'b001; // Green
   light_S  <= 3'b100; // Red
end
S4: begin
   light_M1 <= 3'b010; // Yellow
   light_M2 <= 3'b100; // Red
   light_MT <= 3'b010; // Yellow
   light_S  <= 3'b100; // Red
end
S5: begin
   light_M1 <= 3'b100; // Red
   light_M2 <= 3'b100; // Red
   light_MT <= 3'b100; // Red
   light_S  <= 3'b001; // Green
end
S6: begin
   light_M1 <= 3'b100; // Red
   light_M2 <= 3'b100; // Red
   light_MT <= 3'b100; // Red
   light_S  <= 3'b010; // Yellow
end
default: begin
   light_M1 <= 3'b000; // Off
```

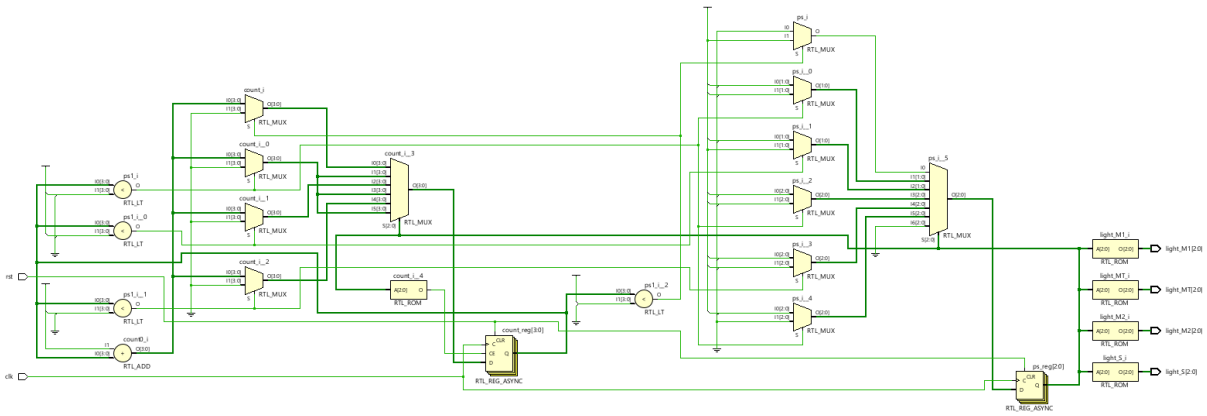
```
            light_M2 <= 3'b000; // Off
            light_MT <= 3'b000; // Off
            light_S  <= 3'b000; // Off
        end
    endcase
end

endmodule
```

Test Bench:

```
`timescale 1ns / 1ps

module Traffic_Light_Controller_TB;

    // Input signals
    reg clk, rst;

    // Output signals
    wire [2:0] light_M1;
    wire [2:0] light_S;
    wire [2:0] light_MT;
    wire [2:0] light_M2;

    // Device Under Test (DUT)
    Traffic_Light_Controller dut (
        .clk(clk),
        .rst(rst),
        .light_M1(light_M1),
```

```
        .light_S(light_S),
        .light_M2(light_M2),
        .light_MT(light_MT)
    );
     // Clock generation
    initial begin
        clk = 1'b0;
        forever #(1000000000 / 2) clk = ~clk; // Toggle clock every half period
    end
    // Reset sequence
    initial begin
        rst = 0;            // Initial reset
        #1000000000;        // Wait 1 second
        rst = 1;            // Activate reset
        #1000000000;        // Hold reset for 1 second
        rst = 0;            // Deactivate reset
        #(1000000000 * 200); // Simulate for additional 200 seconds
        $finish;            // End simulation
    end

endmodule
```

RTL Design:



Result:

The traffic light control system operates in fixed stages, alternating green, yellow, and red signals to manage traffic flow at intersections. Verilog HDL is used to design and simulate the system, ensuring orderly vehicle movement. While the system uses fixed time allocation for signals, it cannot adjust dynamically to varying traffic densities, which may cause inefficiencies. Traffic signal controllers reduce accidents and improve safety, but many areas still lack these systems, leading to frequent collisions. Implementing such systems in underserved areas is essential for better traffic management and safety.