Enhanced Traffic Light Controller

VERILOG CODE->

- **Traffic_light_controller.v**

```
module traffic_light_controller(
    input clk,               // Clock signal
    input reset,             // Reset signal
    input pedestrian_button,  // Pedestrian button signal
    input traffic_sensor,     // Traffic sensor signal
    output reg [2:0] main_light,   // Traffic light for main road (Red, Yellow, Green)
    output reg [2:0] side_light,   // Traffic light for side road (Red, Yellow, Green)
    output reg pedestrian_signal   // Pedestrian crossing signal
);

// State encoding
parameter RED    = 3'b100;
parameter YELLOW = 3'b010;
parameter GREEN  = 3'b001;

// State definitions
parameter S_MAIN_GREEN = 3'b000;
parameter S_MAIN_YELLOW = 3'b001;
parameter S_SIDE_GREEN = 3'b010;
parameter S_SIDE_YELLOW = 3'b011;
parameter S_PED_CROSS = 3'b100;

reg [2:0] state, next_state;
reg [15:0] timer;  // Timer for delay

// State transition
always @(posedge clk or posedge reset) begin
    if (reset) begin
        state <= S_MAIN_GREEN;
        timer <= 16'd0;
    end else begin
        state <= next_state;
        if (timer > 0)
            timer <= timer - 1;
    end
end

// Next state logic and output control
always @(*) begin
    case (state)
        S_MAIN_GREEN: begin
            main_light = GREEN;
            side_light = RED;
            pedestrian_signal = 0;
            if (traffic_sensor || timer == 16'd5000) // Change based on timer or sensor
                next_state = S_MAIN_YELLOW;
            else
                next_state = S_MAIN_GREEN;
        end
        S_MAIN_YELLOW: begin
            main_light = YELLOW;
            side_light = RED;
            pedestrian_signal = 0;
            if (timer == 16'd1000)
                next_state = S_SIDE_GREEN;
            else
                next_state = S_MAIN_YELLOW;
        end
        S_SIDE_GREEN: begin
            main_light = RED;
            side_light = GREEN;
            pedestrian_signal = 0;
```

```
      if (pedestrian_button || timer == 16'd5000) // Change based on timer or pedestrian button
        next_state = S_SIDE_YELLOW;
      else
        next_state = S_SIDE_GREEN;
    end
    S_SIDE_YELLOW: begin
      main_light = RED;
      side_light = YELLOW;
      pedestrian_signal = 0;
      if (timer == 16'd1000)
        next_state = S_PED_CROSS;
      else
        next_state = S_SIDE_YELLOW;
    end
    S_PED_CROSS: begin
      main_light = RED;
      side_light = RED;
      pedestrian_signal = 1;
      if (timer == 16'd3000)
        next_state = S_MAIN_GREEN;
      else
        next_state = S_PED_CROSS;
    end
    default: begin
      next_state = S_MAIN_GREEN;
    end
  endcase
end
endmodule
```

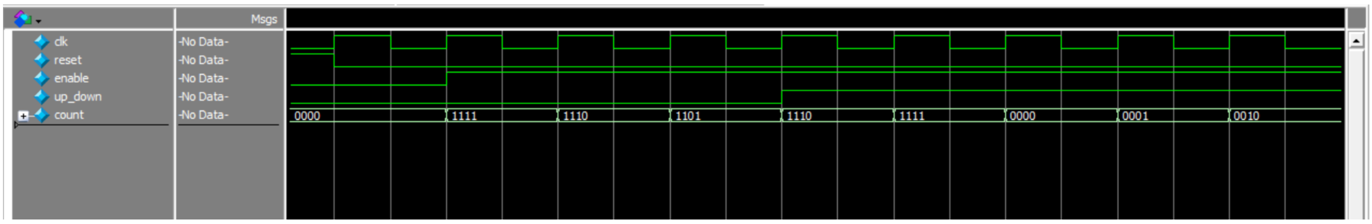
Traffic_light_controller_tb.v->

```
`timescale 1ns/1ps
module traffic_light_controller_tb;

  // Testbench signals
  reg clk;
  reg reset;
  reg pedestrian_button;
  reg traffic_sensor;
  wire [2:0] main_light;
  wire [2:0] side_light;
  wire pedestrian_signal;

  // Instantiate the Traffic Light Controller module
  traffic_light_controller uut (
    .clk(clk),
    .reset(reset),
    .pedestrian_button(pedestrian_button),
    .traffic_sensor(traffic_sensor),
    .main_light(main_light),
    .side_light(side_light),
    .pedestrian_signal(pedestrian_signal)
  );

  // Clock generation (100 MHz)
  always #5 clk = ~clk;  // 10ns period

  initial begin
    // Initialize signals
    clk = 0;
    reset = 1;          // Assert reset
    pedestrian_button = 0;
```

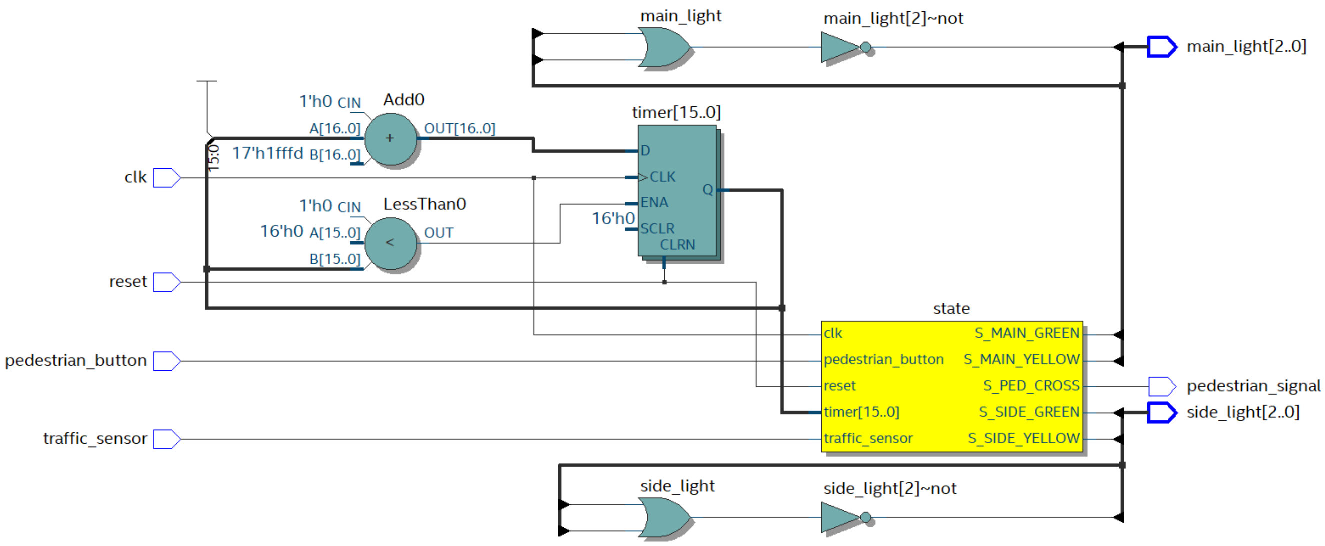
```
        traffic_sensor = 0;

        // Hold reset for a while to ensure proper initialization
        #20 reset = 0;        // Deassert reset after 20ns

        // Test traffic sensor activation after reset
        #50 traffic_sensor = 1; // Traffic sensor active
        #100 traffic_sensor = 0; // Traffic sensor inactive

        // Test pedestrian button press after some time
        #100 pedestrian_button = 1;
        #50 pedestrian_button = 0; // Button released after 50ns

        // Further simulation to see system behavior over time
        #300 traffic_sensor = 1;   // Traffic sensor active again
        #50 pedestrian_button = 1; // Pedestrian button pressed
        #20 pedestrian_button = 0; // Button released
        #100 traffic_sensor = 0;   // Traffic sensor inactive

        // Allow simulation to run for a while
        #1000;

        // End simulation
        $finish;
    end
    // Monitor the inputs and outputs
    initial begin
        $monitor("Time=%0t | clk=%b | reset=%b | pedestrian_button=%b | traffic_sensor=%b | main_light=%b | side_light=%b | pedestrian_signal=%b",
                 $time, clk, reset, pedestrian_button, traffic_sensor, main_light, side_light, pedestrian_signal);
    end
endmodule
```

RESULTS

MODELSIM SIMULATION OUTPUT->



QUARTUS PRIME SYNTHESIS OUTPUT->


main_light
main_light[2]~not
main_light[2..0]
1'h0 CIN
Add0
A[16..0]
OUT[16..0]
17'h1fffd B[16..0]
+
15:0
timer[15..0]
D
CLK
ENA
Q
16'h0
SCLR
CLRN
clk
1'h0 CIN
LessThan0
16'h0 A[15..0]
<
OUT
B[15..0]
reset
state
clk
pedestrian_button
reset
timer[15..0]
traffic_sensor
S_MAIN_GREEN
S_MAIN_YELLOW
S_PED_CROSS
S_SIDE_GREEN
S_SIDE_YELLOW
pedestrian_button
traffic_sensor
pedestrian_signal
side_light[2..0]
side_light
side_light[2]~not