Name : Rhudhresh

Reg. No: 212223050039

Aim:

To design and simulate a traffic light controller for an intersection of three main roads, where each road has equal priority. The controller should regulate the traffic flow efficiently, ensuring safety and smooth movement of vehicles while diverting the traffic to path 1 direction and disabling control in other directions.

Apparatus Required:

1. Hardware Description Language (HDL) simulation environment such as Verilog or VHDL.
2. Simulation software like ModelSim for testing and verification.
3. FPGA development board (optional) for hardware implementation.

Procedure:

1. Define the State Machine
2. Implement the State Machine in HDL
3. Simulate the Design
4. Test the Design
5. Optimize and Refine

State Table :

- To define the states, inputs, outputs, and state transitions.
- Let's denote the three main roads as MR1, MR2, and MR3.
- Each road can have three possible states for its traffic light: Red, Yellow, and Green. Here's the state table:

Current State	Next State	MR1 (Red/Yellow/Green)	MR2 (Red/Yellow/Green)	MR3 (Red/Yellow/Green)	Counter
S_RED	S_GREEN_MR1	Red	Red	Green	0 to 9
S_GREEN_MR1	S_GREEN_MR2	Green	Yellow	Red	0 to 9
S_GREEN_MR2	S_GREEN_MR3	Yellow	Green	Red	0 to 9
S_GREEN_MR3	S_RED	Red	Yellow	Green	0 to 9

- Each row represents a state transition along with the corresponding outputs for each main road.
- The "Counter" column indicates the counts for each state before transitioning to the next state.
- The "Current State" column represents the current state of the state machine.
- The "Next State" column indicates the state to transition to after completing the counts.

- The "MR1", "MR2", and "MR3" columns specify the traffic light states for each main road in the current state.
- "Red", "Yellow", and "Green" denote the states of the traffic lights.
- The counter counts from 0 to 9, where each count corresponds to one clock cycle.

Code:

```
module traffic_light_controller (
    input clk,          // Clock input
    input reset,        // Reset input
    output reg red_MR1, // Red signal for main road 1
    output reg yellow_MR1, // Yellow signal for main road 1
    output reg green_MR1,  // Green signal for main road 1
    output reg red_MR2, // Red signal for main road 2
    output reg yellow_MR2, // Yellow signal for main road 2
    output reg green_MR2,  // Green signal for main road 2
    output reg red_MR3, // Red signal for main road 3
    output reg yellow_MR3, // Yellow signal for main road 3
    output reg green_MR3  // Green signal for main road 3
);

// Internal signals
reg [1:0] state;    // State variable

// State definitions
parameter S_RED = 2'b00;
parameter S_GREEN_MR1 = 2'b01;
parameter S_GREEN_MR2 = 2'b10;
parameter S_GREEN_MR3 = 2'b11;

// Counter for state transitions
reg [3:0] counter;

always @(posedge clk or posedge reset) begin
```

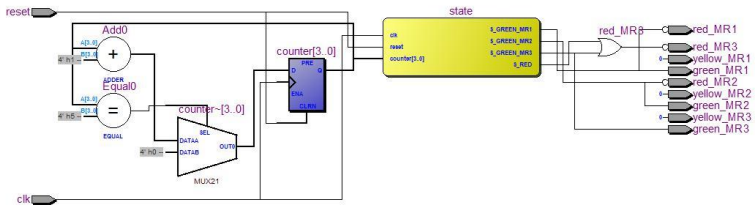
```
if (reset) begin
    // Initialize state and counter
    state <= S_RED;
    counter <= 4'b0000;
end else begin
    // State transition
    case (state)
        S_RED: begin
            if (counter == 4'b0101) begin
                state <= S_GREEN_MR1;
                counter <= 4'b0000;
            end else begin
                counter <= counter + 1;
            end
        end
        S_GREEN_MR1: begin
            if (counter == 4'b0101) begin
                state <= S_GREEN_MR2;
                counter <= 4'b0000;
            end else begin
                counter <= counter + 1;
            end
        end
        S_GREEN_MR2: begin
            if (counter == 4'b0101) begin
                state <= S_GREEN_MR3;
                counter <= 4'b0000;
            end else begin
                counter <= counter + 1;
            end
        end
```

```
            S_GREEN_MR3: begin
                if (counter == 4'b0101) begin
                    state <= S_RED;
                    counter <= 4'b0000;
                end else begin
                    counter <= counter + 1;
                end
            end
        endcase
    end
end

// Output logic based on state
always @(state) begin
    case (state)
        S_RED: begin
            // Disable all signals except MR3
            red_MR1 <= 1'b1;
            yellow_MR1 <= 1'b0;
            green_MR1 <= 1'b0;
            red_MR2 <= 1'b1;
            yellow_MR2 <= 1'b0;
            green_MR2 <= 1'b0;
            red_MR3 <= 1'b0;
            yellow_MR3 <= 1'b0;
            green_MR3 <= 1'b0;
        end
        S_GREEN_MR1: begin
            // Enable MR1 green signal
            red_MR1 <= 1'b0;
            yellow_MR1 <= 1'b0;
```

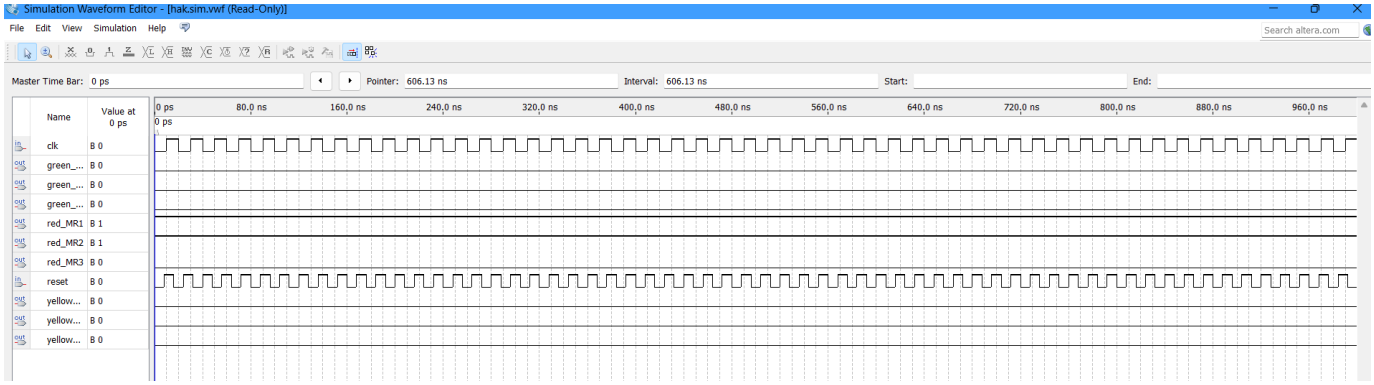
```
        green_MR1 <= 1'b1;
        // Disable MR2 and MR3 signals
        red_MR2 <= 1'b1;
        yellow_MR2 <= 1'b0;
        green_MR2 <= 1'b0;
        red_MR3 <= 1'b1;
        yellow_MR3 <= 1'b0;
        green_MR3 <= 1'b0;
    end
    S_GREEN_MR2: begin
        // Enable MR2 green signal
        red_MR1 <= 1'b1;
        yellow_MR1 <= 1'b0;
        green_MR1 <= 1'b0;
        red_MR2 <= 1'b0;
        yellow_MR2 <= 1'b0;
        green_MR2 <= 1'b1;
        // Disable MR1 and MR3 signals
        red_MR3 <= 1'b1;
        yellow_MR3 <= 1'b0;
        green_MR3 <= 1'b0;
    end
    S_GREEN_MR3: begin
        // Enable MR3 green signal
        red_MR1 <= 1'b1;
        yellow_MR1 <= 1'b0;
        green_MR1 <= 1'b0;
        red_MR2 <= 1'b1;
        yellow_MR2 <= 1'b0;
        green_MR2 <= 1'b0;
        red_MR3 <= 1'b0;
```

```
            yellow_MR3 <= 1'b0;
            green_MR3 <= 1'b1;
        end
    endcase
end

endmodule
```

RTL Schematic View



Output Waveforms


Simulation Waveform Editor - [hak.sim.vwf (Read-Only)]
File Edit View Simulation Help
Search altera.com
Master Time Bar: 0 ps
Pointer: 606.13 ns
Interval: 606.13 ns
Start:
End:
Name
Value at 0 ps
clk B 0
green_... B 0
green_... B 0
green_... B 0
red_MR1 B 1
red_MR2 B 1
red_MR3 B 0
reset B 0
yellow... B 0
yellow... B 0
yellow... B 0
0 ps
80,0 ns
160,0 ns
240,0 ns
320,0 ns
400,0 ns
480,0 ns
560,0 ns
640,0 ns
720,0 ns
800,0 ns
880,0 ns
960,0 ns