TRAFFIC LIGHT CONTROLLER PROJECT

---

1. Introduction

Traffic signals are an essential part of road management systems. This project focuses on the **Verilog implementation** of a **Traffic Light Controller** that operates at a **junction** using a **finite state machine (FSM)** approach. The design ensures that only one direction receives a green signal at a time, while others remain red, closely mirroring real-world traffic systems.

2. Problem Statement

The goal of this project is to design a **realistic traffic light controller** that follows the correct sequence of Red, Yellow, and Green signals for each direction, ensuring smooth traffic flow and safety. The system follows a well-defined **FSM-based operation** to control the lights and manage road intersections effectively.

---

4. Verilog Code Implementation

4.1 Traffic Light Controller (Design Code)

```
module Traffic_Light_Controller(
   input clk, reset,
   output reg [1:0] ns_light,  // North-South Traffic Light (00 = Red, 01 = Yellow, 10 = Green)
   output reg [1:0] ew_light,  // East-West Traffic Light (00 = Red, 01 = Yellow, 10 = Green)
   output reg ns_left, ew_left // Left Turn Allowed (1 = Yes, 0 = No)
);
   // Define State Encoding (Verilog-2005 compatible)
   parameter S0 = 3'b000,  // North-South Green
             S1 = 3'b001,  // North-South Yellow
             S2 = 3'b010,  // All Stop
             S3 = 3'b011,  // East-West Green
             S4 = 3'b100,  // East-West Yellow
             S5 = 3'b101;  // All Stop
```

```
reg [2:0] current_state, next_state;
reg [3:0] counter; // Timing Counter
// State Transition Logic
always @(posedge clk or posedge reset) begin
   if (reset)
      current_state <= S0;
   else if (counter == 10)  // Change state every 10 clock cycles
      current_state <= next_state;
end
// Next State Logic
always @(*) begin
   case (current_state)
      S0: next_state = S1;  // NS Green → Yellow
      S1: next_state = S2;  // NS Yellow → All Stop
      S2: next_state = S3;  // All Stop → EW Green
      S3: next_state = S4;  // EW Green → Yellow
      S4: next_state = S5;  // EW Yellow → All Stop
      S5: next_state = S0;  // All Stop → NS Green
      default: next_state = S0;
   endcase
end
// Output Logic for Traffic Lights
always @(current_state) begin
   case (current_state)
      S0: begin  // North-South Green, East-West Red
```

```
    ns_light = 2'b10; ew_light = 2'b00;
    ns_left = 1; ew_left = 0;
end
S1: begin  // North-South Yellow, East-West Red
    ns_light = 2'b01; ew_light = 2'b00;
    ns_left = 1; ew_left = 0;
end
S2: begin  // All Stop (Red)
    ns_light = 2'b00; ew_light = 2'b00;
    ns_left = 0; ew_left = 0;
end
S3: begin  // North-South Red, East-West Green
    ns_light = 2'b00; ew_light = 2'b10;
    ns_left = 0; ew_left = 1;
end
S4: begin  // North-South Red, East-West Yellow
    ns_light = 2'b00; ew_light = 2'b01;
    ns_left = 0; ew_left = 1;
end
S5: begin  // All Stop (Red)
    ns_light = 2'b00; ew_light = 2'b00;
    ns_left = 0; ew_left = 0;
end
default: begin
    ns_light = 2'b00; ew_light = 2'b00;
```

```
            ns_left = 0; ew_left = 0;
        end
    endcase
  end
  // Counter for Timing
  always @(posedge clk) begin
    if (counter == 10)
        counter <= 0;
    else
        counter <= counter + 1;
  end
endmodule
```

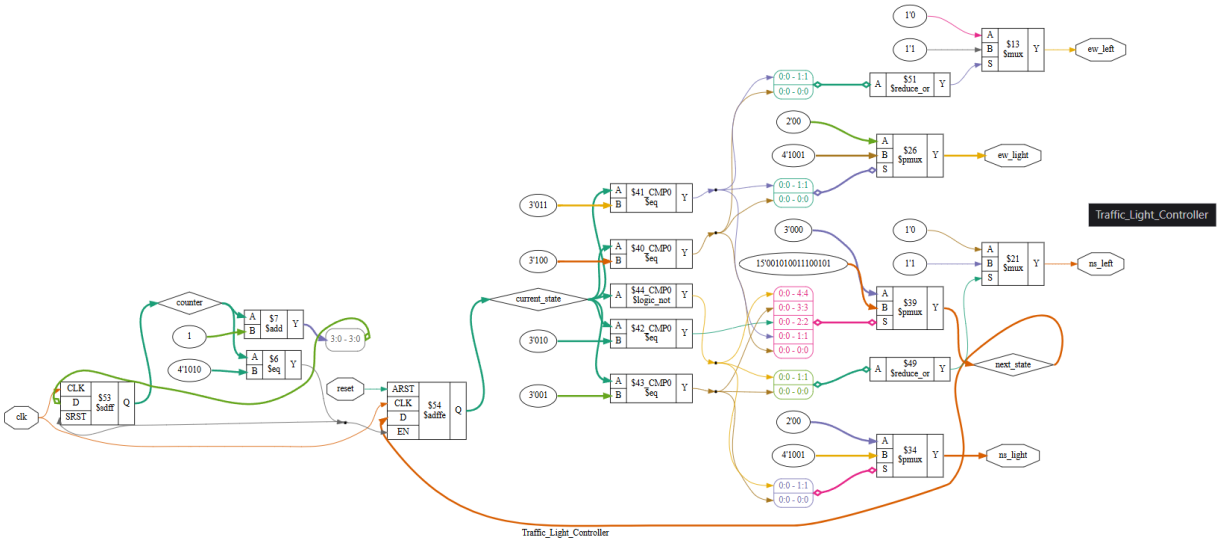
4.2 Testbench Code

```
module testbench;
  reg clk, reset;
  wire [1:0] ns_light, ew_light;
  wire ns_left, ew_left;
  // Instantiate the Traffic Light Controller
  Traffic_Light_Controller uut (
    .clk(clk),
    .reset(reset),
    .ns_light(ns_light),
    .ew_light(ew_light),
    .ns_left(ns_left),
```

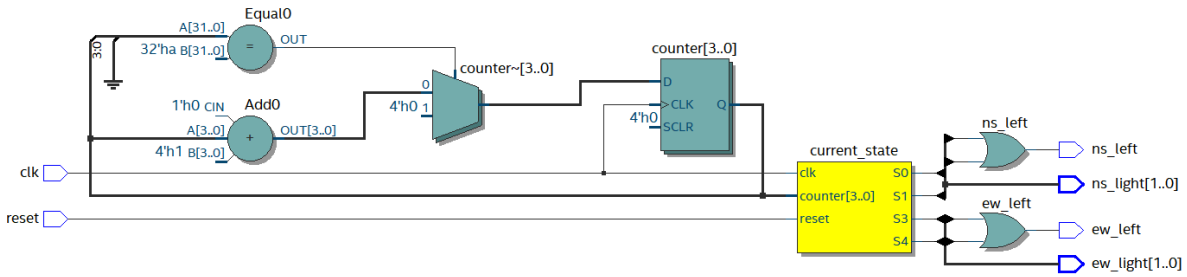
```
        .ew_left(ew_left)
    );
    always #5 clk = ~clk;  // Toggle every 5 time units
    initial begin
        // Initialize Signals
        clk = 0;
        reset = 1;
        #10 reset = 0;  // Release Reset

        // Run Simulation for 100 time units
        #100 $stop;
    end
endmodule
```

5. Simulation Results & RTL Schematic

5.1 Finite State Machine (FSM) Diagram:

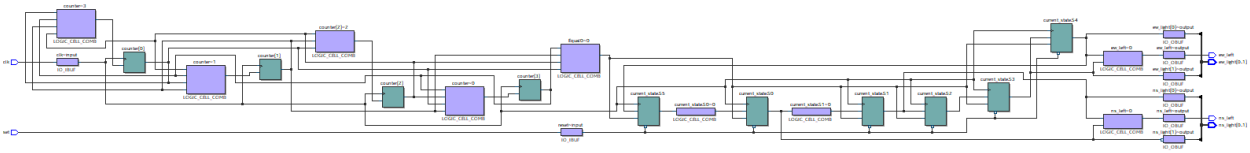


5.2 RTL Schematic

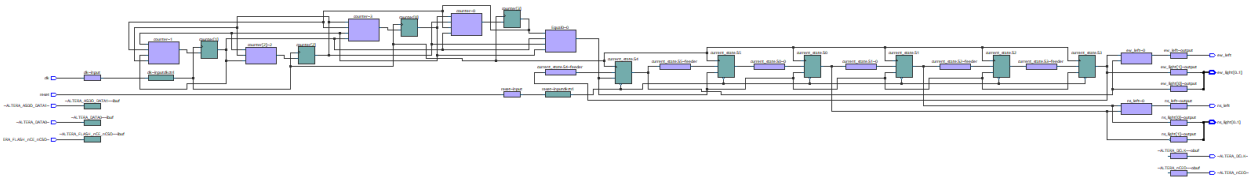


5.3 FPGA Gate Level Implementation

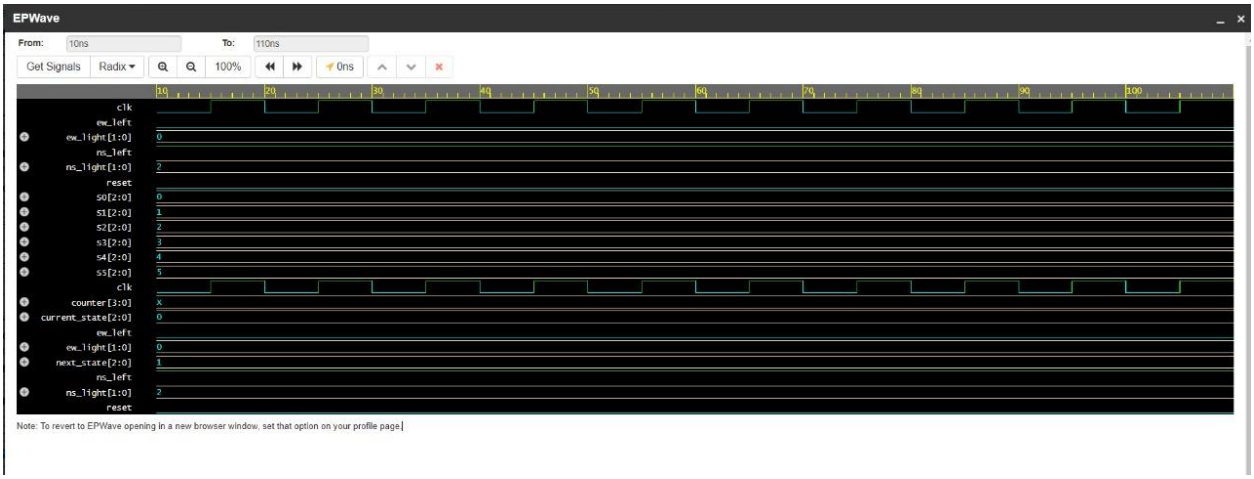
Post-Mapping:



Post-Fitting:

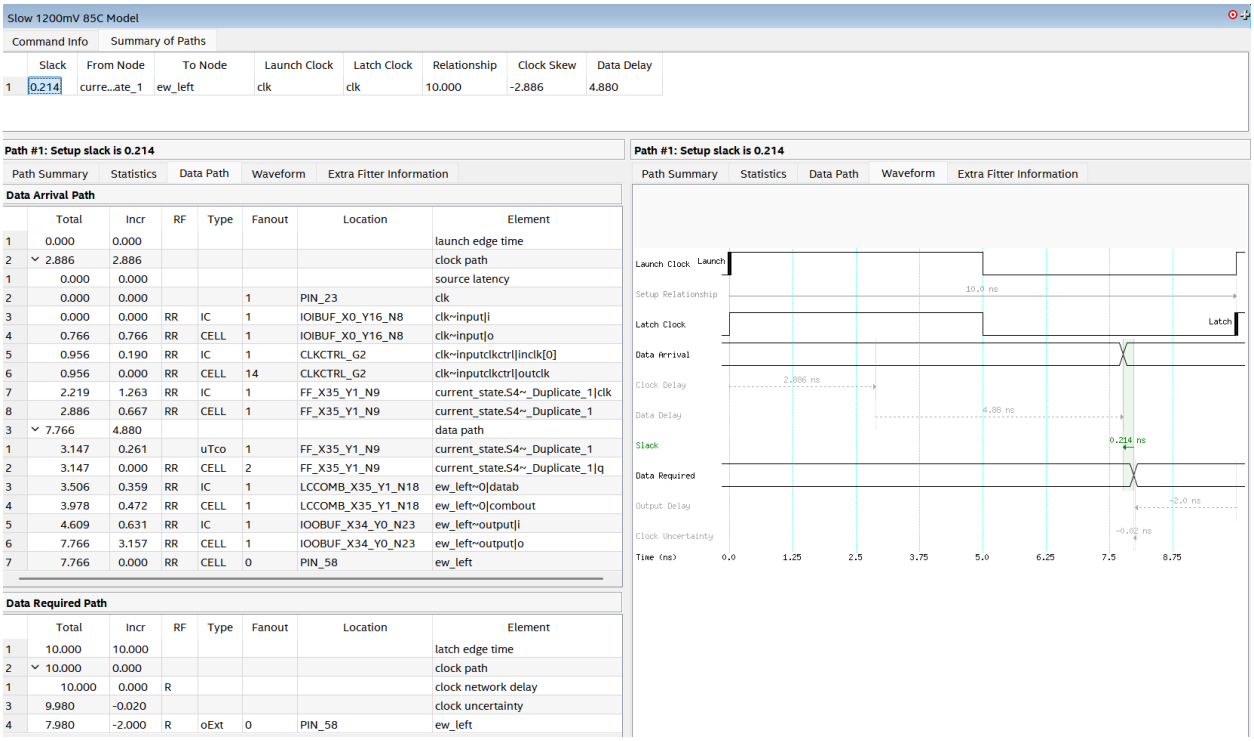


5.4: Output Waveform:



6. Reports After Synthesis

- **Timing Report**



---

7. Conclusion

The designed Traffic Light Controller successfully simulates the FSM-based traffic system. The simulation results confirm that the system transitions correctly between states, ensuring smooth and safe traffic management.

---

8. References

1. FPGA-based Digital Design References
2. NPTEL Lectures on Digital Design
3. ASIC-World Verilog FSM Guide
4. Cyclone IV E EP4CE22E22C8 Pinouts