Report on

Traffic Light ControlSystem Using FSM

Mini-project

Contents

Introduction:

Traffic congestion is a common problem in busy cities, leading to delays and pollution. To address this, an efficient traffic control system is needed to manage traffic at intersections. The **3-Way Traffic Control System** project aims to design a traffic light controller for a three-way intersection using a **Finite State Machine (FSM)**.

In this system, traffic lights for three roads are controlled by a clock-driven FSM, which moves between five different states. Each state controls the lights (green, yellow, or red) for the roads in a specific order. The output (the traffic light signals) depends only on the current state, making the system simple and predictable.

This system is implemented using **Verilog** and can be tested on hardware platforms like **FPGA**. The goal is to provide a reliable and efficient solution for managing traffic flow at intersections and reducing congestion.

Background:

Traffic control systems are used to manage the flow of traffic at intersections to ensure safety and efficiency. In a 3-way intersection, traffic lights are used to control the flow of vehicles from three different roads. The traffic lights need to switch between red, yellow, and green to allow vehicles to move while preventing accidents and congestion. The system you've designed uses a finite state machine (FSM) to control these traffic lights.

Objective:

The primary objective of this project is to design and implement a reliable and efficient traffic light controller system using Verilog HDL. The specific objectives include.

Traffic Regulation: Develop a system that efficiently controls traffic flow at a three-way intersection, ensuring safe and smooth movement of vehicles.

Finite State Machine Implementation: Utilize finite state machines (FSMs) to manage the sequential operation of traffic lights for different directions.

Customizable Timing: Provide flexibility in adjusting the timing for red, yellow, and green signals to accommodate various traffic conditions.

Simulation and Verification: Simulate and test the Verilog code using standard tools to validate functionality, timing accuracy, and reliability.

Methodology:

The design of the Traffic light control system follows a structured methodology:

The development of the traffic light controller project is structured into a series of systematic steps, ensuring an efficient design process, accurate functionality, and scalability. The methodology includes the following phases.

1. Requirements Analysis: Identify the functional requirements of the traffic light system, including the sequence of traffic signals, timing durations, and the number of roads/intersections.
2. System Design: Finite State Machine (FSM) Design: Define states for the traffic light (e.g., Green, Yellow, Red) and transitions between them.
3. Verilog Code Implementation:Develop modular Verilog code to represent. FSM Module: Manages the sequence and transitions of traffic light states. Timer Module: Implements

time delays for each state. Control Logic Module: Ensures proper operation based on inputs like clock and reset.

4. Simulation and Testing: Simulate the Verilog design using tools like ModelSim,Vivado.

Simulation:

Before synthesizing the Verilog code on the FPGA, **simulation** is a crucial step to verify the correctness of the design. Using simulation tools such as **Vivado**, we can simulate the behavior of the FSM under different input scenarios. This allows the designer to ensure that the FSM correctly handles:

- **Normal operations:** Changing traffic lights to green, yellow, and red at appropriate times based on traffic flow and preset schedules.
- **Edge cases:** Handling simultaneous vehicle requests from multiple directions, managing pedestrian crosswalk requests, traffic flow disruptions (e.g., accidents), or abnormal traffic conditions.
- **Error states:** Managing invalid inputs (e.g., faulty sensors), system errors (e.g., malfunctioning signal controllers), or out-of-range requests (e.g., improper signal durations).

Testing:

After simulating the FSM, the system will be tested in a real-world scenario on an FPGA board. The FPGA will receive actual inputs, such as vehicle presence sensors, pedestrian crosswalk requests, and traffic signal timing feedback, and will generate the appropriate control signals for the traffic light changes. The testing phase will include a variety of scenarios, such as:

- Normal user operation with sequential traffic light changes.
- Multiple vehicle detections with varying traffic volumes and priorities.
- Simultaneous vehicle requests from different directions or lanes.
- System error scenarios, such as failure to switch lights or unexpected behavior in traffic light cycles.

Verilog Code:

```
module traffic_light_controller (
   input clk,
   input rst,
   input x,
   output reg [2:0] light1,
   output reg [2:0] light2,
   output reg [2:0] light3
);
   // Define states
   parameter State1 = 3'b001,
             State2 = 3'b010,
             State3 = 3'b011,
```

```
            State4 = 3'b100,
            State5 = 3'b101;

reg [2:0] State, next_state;

    // State transition logic
    always @(posedgeclk or posedgerst) begin
        if (rst)
            State <= State1;
        else
            State <= next_state;
    end

    // Next state logic
    always @(*) begin
        case (State)
            State1: next_state = x ? State2 : State1;
            State2: next_state = x ? State3 : State2;
            State3: next_state = x ? State4 : State3;
            State4: next_state = x ? State5 : State4;
            State5: next_state = x ? State1 : State5;
            default: next_state = State1;
endcase
    end

    // Output logic
    always @(*) begin
        case (State)
            State1: begin
                light1 = 3'b001; light2 = 3'b100; light3 = 3'b100;
            end
            State2: begin
```

```
            light1 = 3'b100; light2 = 3'b010; light3 = 3'b100;
        end
        State3: begin
            light1 = 3'b100; light2 = 3'b001; light3 = 3'b100;
        end
        State4: begin
            light1 = 3'b100; light2 = 3'b100; light3 = 3'b010;
        end
        State5: begin
            light1 = 3'b100; light2 = 3'b100; light3 = 3'b001;
        end
        default: begin
            light1 = 3'b100; light2 = 3'b100; light3 = 3'b100;
        end
endcase
   end
endmodule
```

TESTBENCH:

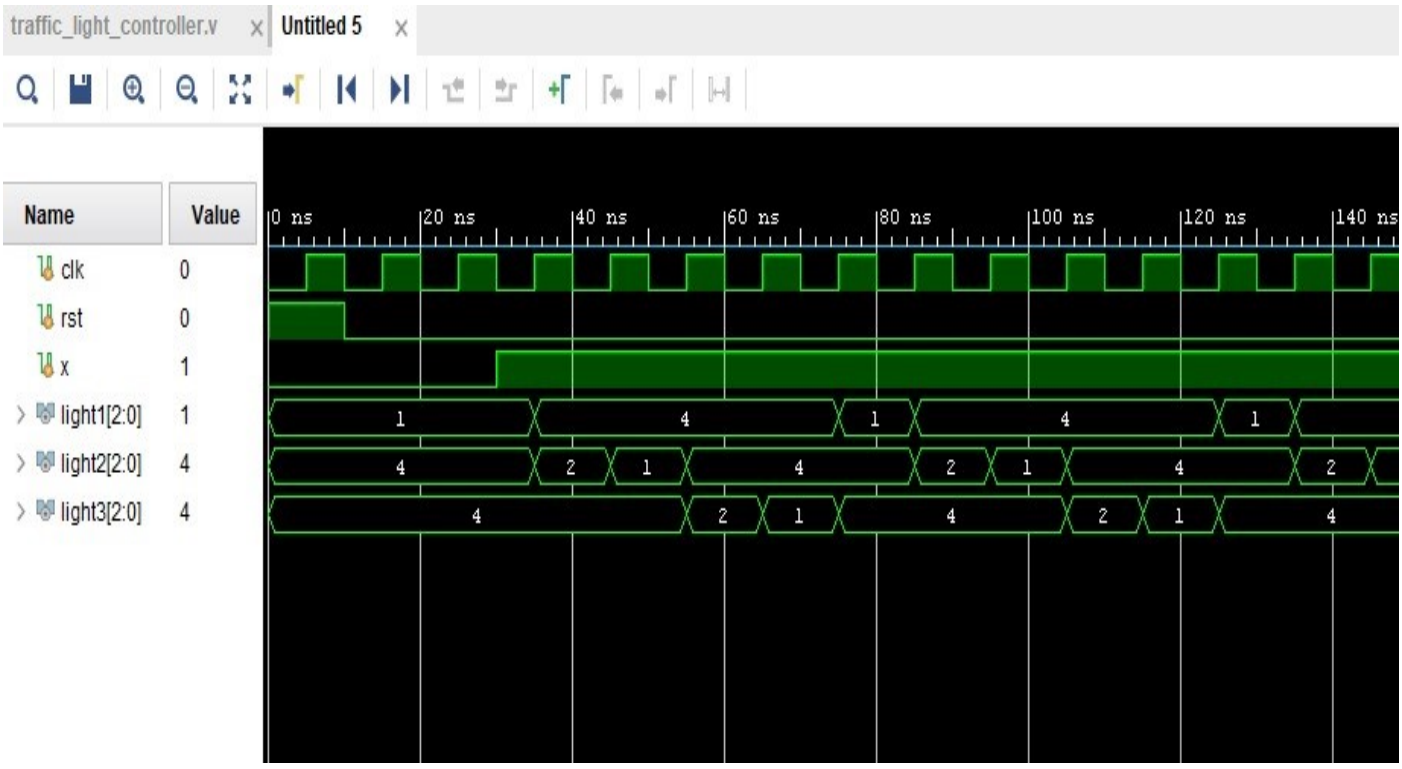
```
module tb_traffic_light_controller;
regclk;
regrst;
   reg x;
   wire [2:0] light1, light2, light3;

   // Instantiate the traffic_light_controller module
traffic_light_controlleruut (
.clk(clk),
.rst(rst),
       .x(x),
.light1(light1),
.light2(light2),
.light3(light3)
```

```
    );

    // Clock generation
    initial begin
clk = 0;
        forever #5 clk = ~clk;  // 10ns clock period
    end

    // Testbench stimulus
    initial begin
        // Initialize inputs
rst = 1;
        x = 0;

        // Apply reset
        #10 rst = 0;

        // Test case 1: Hold x = 0 to remain in the same state
        #20 x = 1;

        // Test case 2: Set x = 1 to test state transitions
        #20 x = 1;  // Transition to State2
        #20 x = 1;  // Transition to State3
        #20 x = 1;  // Transition to State4
        #20 x = 1;  // Transition to State5
        #20 x = 1;  // Transition to State6
        #20 x = 1;  // Transition back to State1
// Test case 3: Toggle x randomly
        #20 x = 1;
        #20 x = 1;
        #20 x = 0;
        #20 x = 1;
```

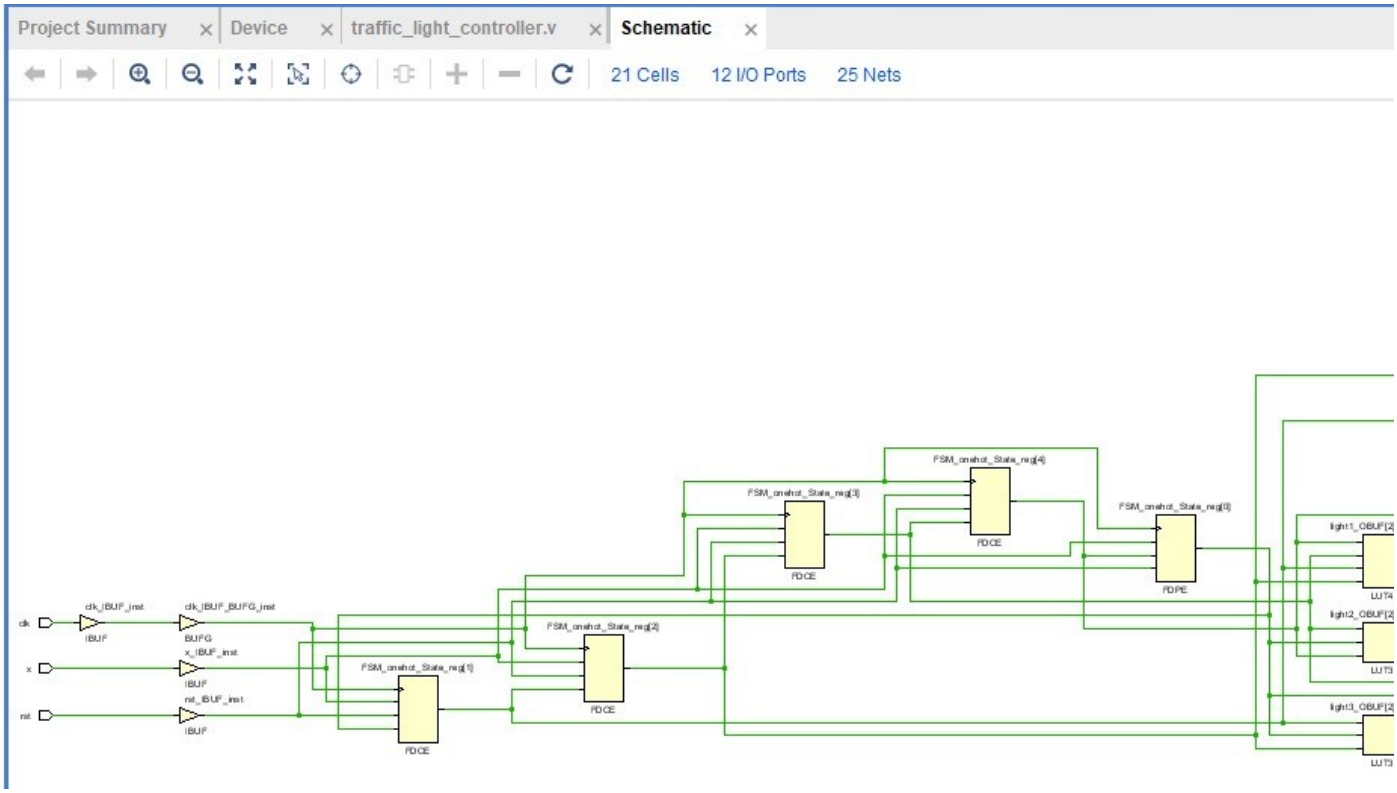
```
        // End simulation

    end

    // Monitor outputs
    initial begin
        $monitor("Time=%0t | rst=%b | x=%b | State=%b | light1=%b | light2=%b | light3=%b",
                 $time, rst, x, uut.State, light1, light2, light3);
    end
endmodule
```

Results:

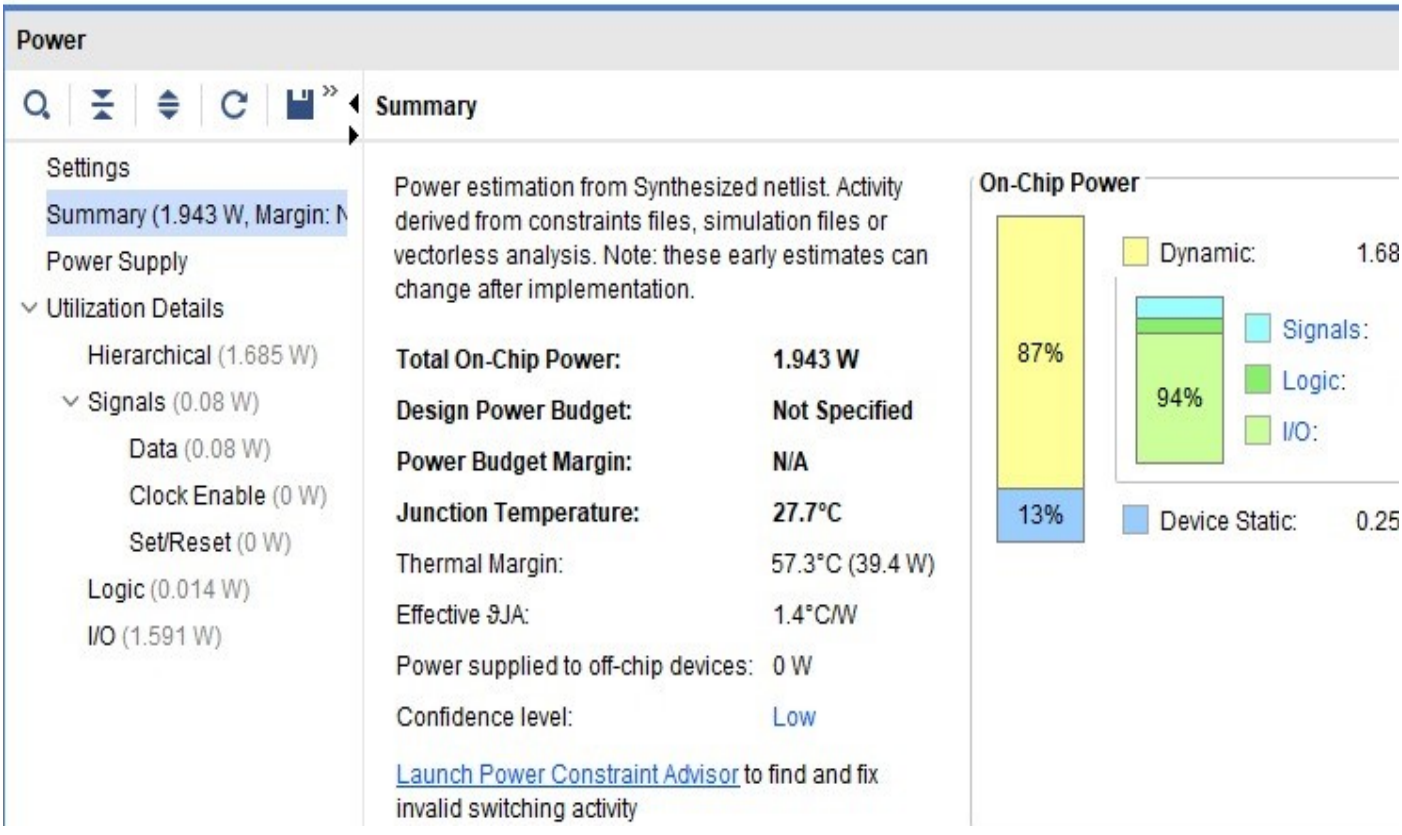
Waveforms



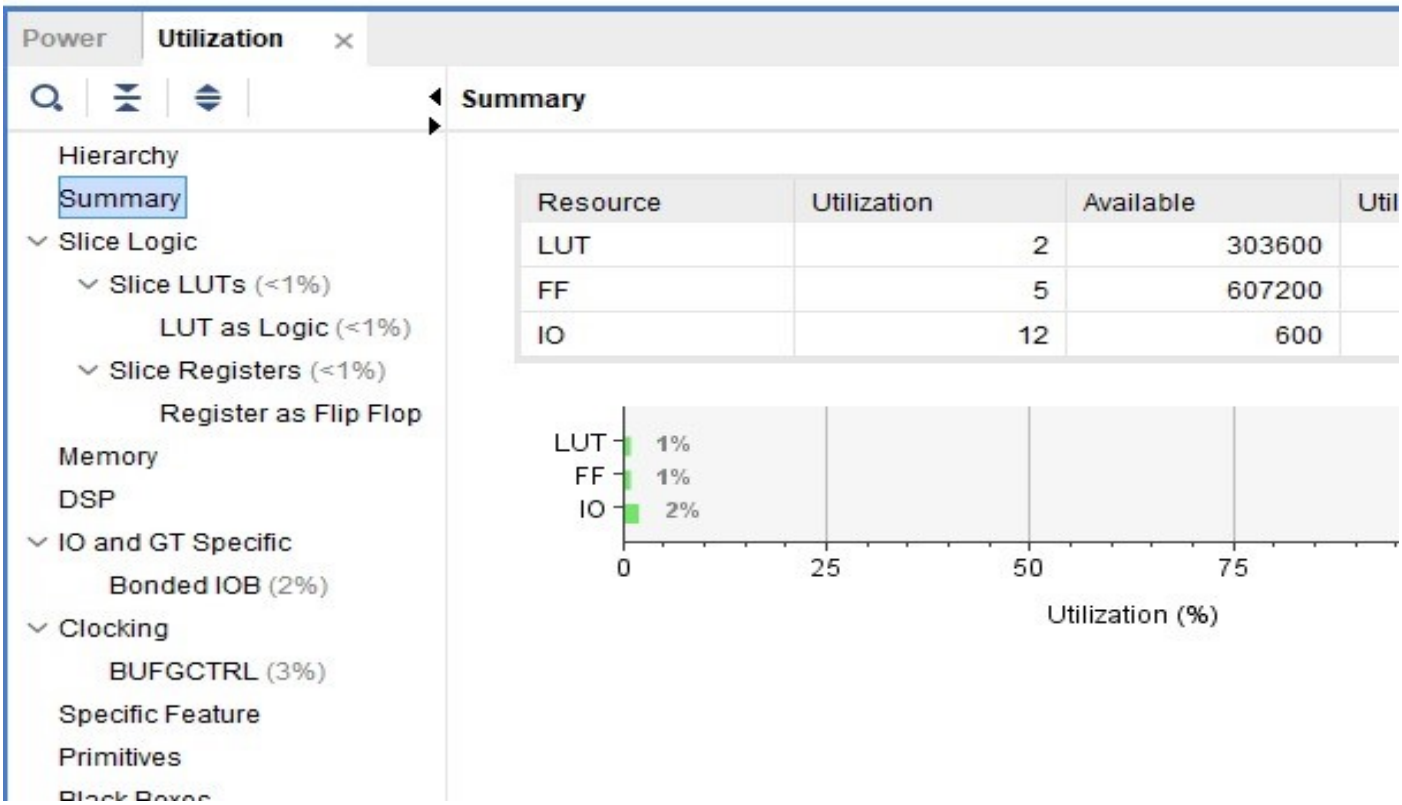
Synthesis



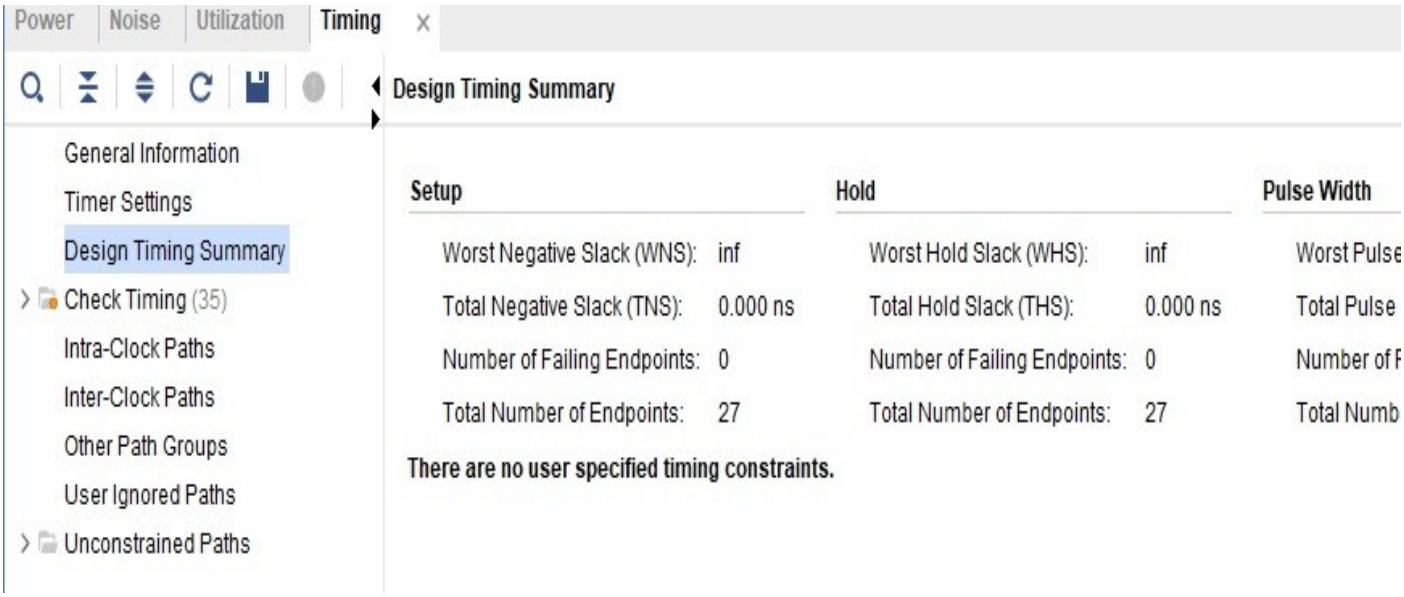
Power Report



Area Report



Timing Report



Conclusion:

The traffic light control system for a 3-way intersection, implemented using a Moore Finite State Machine (FSM), is designed to efficiently manage the traffic lights in a cyclic pattern. The system ensures that each road gets a green light in turn, and it handles transitions from green to yellow to red lights smoothly, promoting safe and efficient traffic flow.

The testbench simulation verifies that the traffic light system works correctly by checking the state transitions and ensuring the lights are controlled as expected. With the FSM approach, the system operates predictably, reducing the chance of errors and making it easy to implement for real-world traffic control applications.

In summary, the design is functional, reliable, and suitable for use in controlling traffic lights at a 3-way intersection.

Future Work:

- Expand to support multi-intersection traffic control systems with centralized request management.
- Implement advanced features like:
    - Vehicle count sensing for traffic flow optimization.
    - Pedestrian detection and priority management for safer crossings.

References:

1.http://www.asic-world.com/tidbits/verilog_fsm.html