A blue and black logo

Description automatically generated

***Faculty of Engineering Ain Shams University***

***ECE413s - ASIC Design and Automation***

**Fall 2024**

**Traffic Light Controller**

|  |  |  |
| --- | --- | --- |
| **Name** | **ID** | **Task** |
| Mustafa Ahmed Hashem | 2100467 | Controller |
| Kareem Ehab Ibrahim | 2100913 | Test bench |
| Michael George Naem | 2100709 | FSM |
| Aya Faried Hussain | 2001967 | Light algorithm |

# Traffic Light Controller System Report

## System Design

This Traffic Light Controller system is designed to manage traffic flow at a four-way intersection. The design focuses on efficient traffic regulation by dynamically adjusting light durations based on real-time sensor inputs. The system ensures fairness and reduces congestion by prioritizing active lanes while maintaining safe transitions between signals. The key design considerations include:  
- Scalability: The system can adapt to different configurations and input parameters.  
- Responsiveness: Real-time sensor inputs enable dynamic adjustments to traffic light durations.  
- Safety: A robust state machine ensures safe transitions and prevents conflicts.

## Light Algorithm

The traffic light algorithm determines the behavior of each light at the intersection. The system operates in four main phases corresponding to the four directions (North, South, East, West). Each phase includes green, yellow, and red states, with durations influenced by sensor inputs. The algorithm prioritizes lanes with active traffic (indicated by sensor\_1th signals) and extends green durations if extended traffic is detected (sensor\_5th signals). The sequence ensures all directions are served equitably.

## Controller Specifications

The controller operates on a finite state machine (FSM) with the following parameters:  
- DEFAULT\_GREEN\_TIME: The base green light duration is set to 9 time units.  
- EXTENDED\_GREEN\_TIME: If extended traffic is detected, the green light duration is increased to 18 time units.  
- YELLOW\_TIME: A fixed duration of 2 time units for the yellow light ensures safe transitions.  
- RED\_TIME: A fixed duration of 2 time units ensures all lights are red before transitioning to the next green phase.  
Inputs:  
- clk: Clock signal to synchronize operations. (recommend clock 5 second)  
- rst: Reset signal to initialize the system to a safe state.  
- sensor\_1th: 4-bit input indicating the presence of traffic in each direction.  
- sensor\_5th: 4-bit input indicating extended traffic in each direction.  
Outputs:  
- Light\_north, Light\_south, Light\_east, Light\_west: 2-bit signals representing the state (red, yellow, green) of lights in each direction.

## FSM Design (Hybrid state machine -> Moore for Light & Mealy for timer)

The Finite State Machine (FSM) for the traffic light controller has the following states:  
- ALL\_RED: All lights are red; this is the default and safe state during reset or transitions.  
- TLNG, TLEG, TLSG, TLWG: Green light states for North, South, East, and West directions.  
- TLNY, TLEY, TLSY, TLWY: Yellow light states for respective directions.  
State transitions are determined by:  
- Sensor inputs indicating active traffic.  
- Timer values to ensure adequate duration for each phase.  
The FSM ensures that no two conflicting directions are green simultaneously and maintains fair rotation among directions.

![A white background with black and white clouds

Description automatically generated]()A diagram of a flowchart

Description automatically generatedA diagram of a computer program

Description automatically generated with medium confidence

A diagram of a system

Description automatically generated

They both describe the system accurately but the handwritten is more detailed

**Notes:**

-Green time depends on the Sensor 1st & 5th

-Any outputs aren’t mentioned are by default 0

Red = 0, Green = 2, Yellow =1

## Testbench

The testbench simulates the Traffic Light Controller to validate its functionality under different scenarios. The testbench includes:  
- Clock Generation: A periodic clock signal drives the FSM.  
- Reset Conditions: The system is initialized to the ALL\_RED state for safety.  
- Sensor Inputs: Stimuli simulate the presence and intensity of traffic in each direction.  
Test cases include:  
1. No sensors active: Verifies the default behavior.  
2. Single direction active: Validates green light activation for active lanes.  
3. Extended green duration: Confirms longer green light periods when extended traffic is detected.  
4. Multiple directions active: Ensures proper prioritization and rotation among active lanes.  
5. Reset: Tests the FSM's ability to return to a safe state after reset.

## Simulation Results

Simulation results demonstrate the correct operation of the Traffic Light Controller. Key observations include:  
- Proper transitions between green, yellow, and red phases.  
- Dynamic adjustment of green light durations based on sensor inputs.  
- Safe operation under reset and conflict scenarios.  
The system successfully meets the design objectives, ensuring efficient and safe traffic management.

## Verilog Code

The following is the Verilog code for the Traffic Light Controller and its testbench:

module trafficLightController(

    input clk,                  // Clock signal

    input rst,                  // Reset signal

    input [3:0] sensor\_1th,     // Sensors to select active traffic light

    input [3:0] sensor\_5th,     // Sensors to extend green duration

    output reg [1:0] Light\_north ,Light\_south,Light\_east,Light\_west

);

    // Timing Parameters

    parameter DEFAULT\_GREEN\_TIME = 4'd9;

    parameter EXTENDED\_GREEN\_TIME = 5'd18;

    parameter YELLOW\_TIME = 4'd2;

    parameter RED\_TIME = 4'd2;

    // State Parameters

    parameter ALL\_RED = 4'b0000,

              TLNG = 4'b0001, TLEG = 4'b0010, TLSG = 4'b0100, TLWG = 4'b1000,

              TLNY = 4'b1001, TLEY = 4'b1010, TLSY = 4'b1011, TLWY = 4'b1100;

    reg [3:0] current\_state, next\_state; // Current and next state

    reg [4:0] timer; // Timer for light durations

    // State transition logic

    always @(posedge clk or posedge rst) begin

        if (rst) begin

            current\_state <= ALL\_RED;

            timer <= 4'd0;

             //{Light\_north,Light\_east,Light\_south,Light\_west} = 8'b00000000; // All lights off

        end else if (timer == 0) begin

            current\_state <= next\_state;

            case (next\_state)

                TLNG: timer <= (sensor\_1th[0] & sensor\_5th[0]) ? EXTENDED\_GREEN\_TIME : DEFAULT\_GREEN\_TIME;

                TLEG: timer <= (sensor\_1th[1] & sensor\_5th[1]) ? EXTENDED\_GREEN\_TIME : DEFAULT\_GREEN\_TIME;

                TLSG: timer <= (sensor\_1th[2] & sensor\_5th[2]) ? EXTENDED\_GREEN\_TIME : DEFAULT\_GREEN\_TIME;

                TLWG: timer <= (sensor\_1th[3] & sensor\_5th[3]) ? EXTENDED\_GREEN\_TIME : DEFAULT\_GREEN\_TIME;

                TLNY, TLEY, TLSY, TLWY: timer <= YELLOW\_TIME;

                ALL\_RED: timer <= RED\_TIME;

            endcase

        end else begin

            timer <= timer - 1; // Decrement timer

        end

    end

    // Next state logic for fair rotation

    always @(\*) begin

        case (current\_state)

            ALL\_RED: begin

                if (sensor\_1th[0]) next\_state = TLNG;

                else if (sensor\_1th[1]) next\_state = TLEG;

                else if (sensor\_1th[2]) next\_state = TLSG;

                else if (sensor\_1th[3]) next\_state = TLWG;

                else next\_state = TLNG; // Default start

            end

            // Green to Yellow transitions

            TLNG: next\_state = TLNY;

            TLEG: next\_state = TLEY;

            TLSG: next\_state = TLSY;

            TLWG: next\_state = TLWY;

            // Yellow to next Green transitions

            TLNY: next\_state = TLEG;

            TLEY: next\_state = TLSG;

            TLSY: next\_state = TLWG;

            TLWY: next\_state = TLNG;

            default: next\_state = ALL\_RED; // Safety fallback

        endcase

    end

    // Traffic light outputs

    always @(\*) begin

        case (current\_state)

            TLNG: {Light\_north,Light\_east,Light\_south,Light\_west} = 8'b10000000; // North green

            TLEG: {Light\_north,Light\_east,Light\_south,Light\_west} = 8'b00100000; // South green

            TLSG: {Light\_north,Light\_east,Light\_south,Light\_west} = 8'b00001000; // East green

            TLWG: {Light\_north,Light\_east,Light\_south,Light\_west} = 8'b00000010; // West green

            TLNY: {Light\_north,Light\_east,Light\_south,Light\_west} = 8'b01000000; // North yellow

            TLEY: {Light\_north,Light\_east,Light\_south,Light\_west} = 8'b00010000; // South yellow

            TLSY: {Light\_north,Light\_east,Light\_south,Light\_west} = 8'b00000100; // East yellow

            TLWY: {Light\_north,Light\_east,Light\_south,Light\_west} = 8'b00000001; // West yellow

            default: {Light\_north,Light\_east,Light\_south,Light\_west} = 8'b00000000; // All red

        endcase

    end

endmodule

// Test Bench

`timescale 1s/1ms

module trafficLightController\_tb;

    // Inputs

    reg clk;

    reg rst;

    reg [3:0] sensor\_1th;

    reg [3:0] sensor\_5th;

    // Outputs

    wire [1:0] Light\_north ,Light\_south,Light\_east,Light\_west;

    // Instantiate the trafficLightController module

    trafficLightController uut (

        .clk(clk),

        .rst(rst),

        .sensor\_1th(sensor\_1th),

        .sensor\_5th(sensor\_5th),

        .Light\_north(Light\_north),

        .Light\_east(Light\_east),

        .Light\_west(Light\_west),

        .Light\_south(Light\_south)

    );

    // Clock generation

    initial begin

        clk = 0;

        forever #2.5 clk = ~clk; // Clock with 10-time unit period

    end

    // Test sequence

    initial begin

        // Monitor changes

        $monitor("Time=%0d | rst=%b | sensor\_1th=%b | sensor\_5th=%b | Light\_north=%b | Light\_south=%b | Light\_east=%b | Light\_west=%b |", $time, rst, sensor\_1th, sensor\_5th, Light\_north ,Light\_south,Light\_east,Light\_west);

        // Test case 1: No sensors active

        rst = 1; sensor\_1th = 4'b0000; sensor\_5th = 4'b0000;

        #100 rst = 0;

        // Test case 2: North light active

        #200 sensor\_1th = 4'b0001; sensor\_5th = 4'b0000;

        // Test case 3: North light extended green

        #200 sensor\_1th = 4'b0001; sensor\_5th = 4'b0001;

        // Test case 4: South light active

        #400 sensor\_1th = 4'b0011; sensor\_5th = 4'b0000;

        // Test case 5: South light extended green

        #400 sensor\_1th = 4'b1100; sensor\_5th = 4'b1010;

        // Test case 6: Multiple sensors active (priority North)

        #400 sensor\_1th = 4'b1011; sensor\_5th = 4'b1011;

        // Test case 7: Reset the module

        #700 rst = 1; #10 rst = 0;

        #100 $finish; // End simulation

    end

endmodule

## Wave form

The following is the waveform snapshots for the Traffic Light Controller:

Testcase 1:

A screen shot of a computer

Description automatically generated

Testcase 2:

A screen shot of a computer

Description automatically generated

Testcase 3:

A screen shot of a computer

Description automatically generated

Testcase 4:

A screenshot of a video game

Description automatically generated

Testcase 5:

A screenshot of a computer

Description automatically generated

Testcase 6:

A screen shot of a computer

Description automatically generated

Testcase 7:

A screen shot of a computer

Description automatically generated