iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 12 **Issue:** XII **Month of publication:** December 2024

DOI: https://doi.org/10.22214/ijraset.2024.65958

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue XII Dec 2024- Available at www.ijraset.com

Design and Simulation of an Optimized Traffic Controller Using Moore FSM

Abhirup De[1], A.P. Amit Nigam[2], Sraman Chatterjee[3], Amrita Roy[4], Srotoswini Sen[5]
[1, 3, 4, 5]*Student,* [2]*Associate Professor, Department of Electronics and Communication Engineering (ECE), Narula Institute of Technology, Kolkata, West Bengal*

Abstract: ***Traffic congestion poses a major challenge in urban areas, adversely affecting both efficiency and safety. While traditional traffic control systems can be effective in specific scenarios, they often lack the adaptability required to respond to changing traffic conditions in real time. This paper introduces an optimized traffic management system designed using the Moore Finite State Machine (FSM) model. The Moore FSM was selected for its simplicity, reliability, and ability to manage state-dependent outputs, which are essential for controlling traffic light cycles. The research focuses on developing a traffic signal controller that dynamically responds to real-time traffic data, aiming to minimize waiting times, improve traffic flow, and enhance energy efficiency. The proposed system employs sensors to gather real-time traffic data and uses a state-driven approach to regulate light signal transitions. The controller is modelled using Verilog, and its performance is analysed through simulations in diverse traffic conditions, such as peak hours, low-traffic periods, and scenarios requiring emergency vehicle prioritization. Simulation results indicate notable improvements in traffic flow, with reduced average waiting times and congestion compared to conventional fixed-time traffic controllers. Additionally, the system enhances energy efficiency by optimizing signal timings based on live data, thereby reducing unnecessary signal changes.***
The paper concludes by highlighting the advantages and potential applications of the Moore FSM-based traffic controller in contemporary traffic management systems. It also outlines recommendations for future development, including integration with intelligent transportation systems (ITS) and scaling for broader applications.

I. INTRODUCTION

A. Overview of Traffic Signal Control Systems

Traffic signal control systems play a vital role in urban transportation networks, facilitating the safe and efficient movement of both vehicles and pedestrians at intersections. These systems regulate traffic flow through the use of light signals—red, yellow, and green—to manage vehicle movements and maximize the effective utilization of road infrastructure. Efficient management of traffic signals is essential not only for improving safety and reducing the risk of accidents but also for mitigating congestion, enhancing fuel efficiency, and lowering pollution levels by minimizing unnecessary delays at intersections. Traditionally, traffic signal systems have utilized fixed-time control, where the durations of red, yellow, and green phases are pre-set based on the time of day or general traffic flow patterns. While this method is straightforward and relatively easy to implement, it lacks flexibility and cannot adapt to real-time fluctuations in traffic conditions. As a result, these systems often fail to ensure optimal traffic flow, particularly during peak hours or in situations where unexpected events, such as accidents or adverse weather, disrupt typical traffic patterns. To overcome these inefficiencies, more advanced Adaptive Traffic Control Systems (ATCS) have been developed. These systems use real-time data from traffic sensors (e.g., inductive loops, cameras, or radar detectors) to dynamically adjust signal timings based on actual traffic demand. While adaptive systems improve traffic flow, they can be expensive and complex to implement due to the need for sophisticated algorithms and costly hardware. As urban areas move toward smarter transportation solutions, the demand for affordable, efficient, and easily deployable traffic control systems is growing.

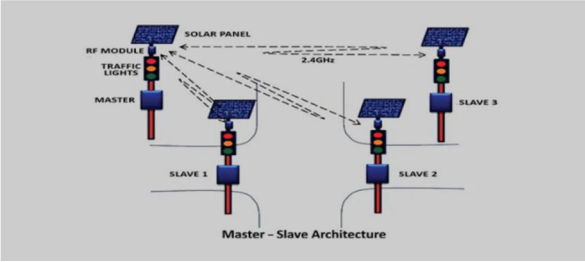


Figure 2: Master-Slave Architecture of Adaptive Traffic Control Systems.

©IJRASET: All Rights are Reserved

In this context, Finite State Machines (FSMs) offer a promising approach to enhancing the adaptability and efficiency of traffic signal control without the complexity of traditional adaptive systems. FSMs, when combined with discrete logic design, can create traffic control systems that are simple to implement while still being responsive to changing traffic conditions.

B. Introduction to Finite State Machines (FSM) and Their Role in Discrete Logic Design

A Finite State Machine (FSM) is a computational model used to describe systems that operate with a finite number of distinct states and transitions between them. FSMs are widely used in digital circuit design, control systems, and software applications. An FSM consists of three core elements:

1) *States:* Represent the various conditions or phases of the system (e.g., the states of a traffic signal such as Red, Green, Yellow).
2) *Transitions:* Define the rules for moving from one state to another, often triggered by specific events or inputs (e.g., vehicle detection or a timer reaching zero).
3) *Inputs:* Events or conditions that cause state transitions (e.g., vehicle presence, pedestrian button presses, or elapsed time).

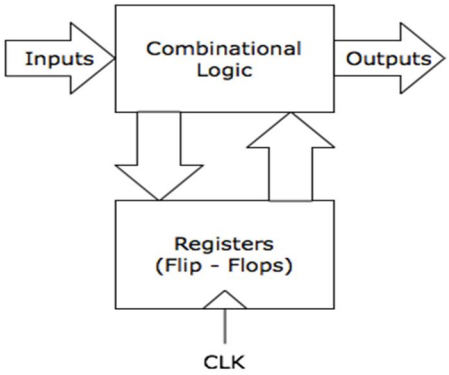


Figure 1: Flow chart of Finite State Machines (F.S.M).

FSMs are particularly suited for systems where the behavior is predictable, and the system's current state is determined only by its previous state and the current inputs, rather than the entire history of previous events. This characteristic makes FSMs ideal for traffic signal control, as the state of the traffic signal (Red, Green, Yellow) is determined by a small set of conditions such as vehicle count, waiting time, or pedestrian demand.

In traffic signal control, FSMs can represent the different phases of the signal cycle. For instance, an FSM could model the signal states at an intersection, with each state representing a different signal configuration (e.g., Green for one direction, Red for the other). The transitions between these states can be triggered by various factors, such as vehicle counts, preset time intervals, or pedestrian signals.

The use of FSMs in traffic signal controllers allows for a structured, deterministic approach to managing signal transitions. FSMs offer several advantages:

1) *Simplicity:* FSMs are straightforward to design and implement, making them ideal for low-cost, reliable solutions.
2) *Predictability:* With a fixed set of states and transitions, FSMs ensure predictable system behavior, which is easier to debug and maintain.
3) *Real-Time Adaptability:* Finite State Machines (FSMs) can be designed to process real-time inputs, allowing for dynamic modifications to signal timings in response to prevailing traffic conditions.

Moreover, by combining FSMs with discrete logic circuits, traffic signal controllers can be implemented using simple, reliable hardware components like logic gates (AND, OR, NOT) and flip-flops, which perform state transitions. Discrete logic provides an affordable, energy-efficient solution that can be easily integrated into existing traffic management systems, offering a practical alternative to more complex software-driven systems.

By leveraging FSMs in combination with discrete logic, traffic signal controllers can be designed to adjust to varying traffic conditions while maintaining high reliability and simplicity. This paper explores the potential of FSM-based traffic signal control systems, presenting a model that combines FSMs and discrete logic to optimize traffic flow at intersections.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue XII Dec 2024- Available at www.ijraset.com

II. LITERARY REVIEW

A. Evolution of Traffic Signal Systems

Over the last century, traffic signal control systems have seen significant advancements, driven by the growing need for effective traffic management as urban populations increased and vehicle numbers surged. The earliest traffic signals, introduced in the early 20th century, were rudimentary mechanical systems with fixed timing cycles, often operated manually. While functional at the time, these systems quickly proved inadequate as cities expanded and traffic volumes became more variable. Fixed-time signal control, which allocated predetermined durations to each phase (red, yellow, and green), eventually became the standard approach. Despite its reliability, this method lacked the flexibility to adapt to real-time traffic fluctuations, resulting in congestion and inefficient traffic flow, particularly during peak hours or periods of irregular traffic demand.

In response to the limitations of fixed-time control, demand-responsive or sensor-based systems were introduced. These systems used sensors placed in or around the roadway to detect the presence of vehicles, such as inductive loop detectors, infrared sensors, or cameras. The primary goal was to adjust signal timings dynamically based on real-time traffic demand, improving vehicle flow and reducing congestion. The inclusion of sensors was a significant advancement, enabling adaptive control, but these systems remained complex and costly to implement due to the need for additional infrastructure, communication networks, and data processing algorithms. The next phase of development led to adaptive traffic signal control systems (ATCS), which integrate real-time traffic data, traffic flow predictions, and sophisticated algorithms to continuously optimize signal timings. These systems, capable of adjusting signal phases and timings based on traffic density, offered substantial improvements in efficiency over their predecessors. Advanced algorithms, such as fuzzy logic, neural networks, and reinforcement learning, were incorporated into ATCS to enhance decision-making. However, the computational complexity and high implementation costs remained significant challenges, particularly for cities with limited budgets. Simultaneously, the concept of distributed control emerged, allowing each intersection to adjust its signal independently based on local conditions while maintaining coordination with other intersections. This approach reduced reliance on centralized control, enabling more responsive and localized traffic management.

B. Use of Finite State Machines (FSMs) in Traffic Signal Control

The application of Finite State Machines (FSMs) in traffic signal control is a more recent development. FSMs provide a structured, predictable, and computationally efficient approach to designing traffic signal controllers. An FSM consists of a finite set of states (representing signal phases like Green, Yellow, Red), with transitions between states triggered by specific inputs (e.g., vehicle detection or elapsed time). FSMs are particularly beneficial for traffic signal systems because of their simplicity, reliability, and ease of implementation. FSMs in traffic signal control were first explored in the 1960s, when digital systems replaced analog control. In these early implementations, FSMs were used to model fixed-time control systems with predetermined state transitions. Over time, FSM-based systems evolved to integrate sensor inputs and adapt to real-time traffic conditions. By incorporating sensors, FSMs could transition between signal phases based on vehicle counts or waiting times, adding a degree of adaptability without the complexity of fully adaptive systems. Recent studies have demonstrated that FSM-based controllers, when combined with discrete logic circuits (e.g., AND, OR gates), offer a practical and efficient solution for traffic signal management. These systems use minimal hardware, making them cost-effective and highly reliable. For example, Muthusamy et al. (2017) proposed an FSM-based controller that adapts to vehicle presence at each lane, significantly reducing waiting times without relying on complex sensor-based or data-driven models. FSMs also offer scalability and ease of modification. Given their structured nature, new states or transitions can be easily added or adjusted, allowing traffic signal control systems to be customized for different intersections, traffic conditions, or regional regulations. For instance, Abd El-Salam et al. (2015) showed how FSMs could be applied in multi-junction traffic management, where each junction had its own FSM interacting with neighboring junctions to optimize network flow.

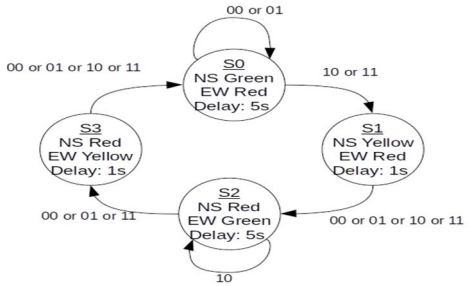


Figure 3: F.S.M logic being used in traffic control system.

©IJRASET: All Rights are Reserved

C. Comparison of FSMs with Other Control Methodologies

FSMs offer numerous advantages, but it is important to compare them with other control methodologies to assess their strengths and weaknesses. The two primary alternatives to FSM-based systems are timer-based control and sensor-based control, each with its own characteristics.

1) *Timer-Based Control:* Timer-based control, or fixed-time control, uses preset signal cycles with fixed durations for each phase (e.g., 30 seconds of Green, 10 seconds of Yellow). This approach is simple to implement and requires minimal infrastructure, but it is highly inefficient in dynamic environments. Since signal cycles are fixed and do not adjust to real-time traffic conditions, vehicles may experience unnecessary delays during low traffic periods, or congestion may build during high traffic demand. Furthermore, this method does not account for pedestrians or special traffic situations (e.g., emergencies). Despite its simplicity, timer-based control is being replaced by more adaptable systems.
2) *Sensor-Based Control:* Sensor-based systems use real-time traffic data, often from inductive loop detectors or cameras, to adjust signal phases dynamically. These systems optimize traffic flow by extending green lights when traffic is heavy and shortening red lights when traffic is light. However, sensor-based systems can be costly to install and maintain, requiring specialized equipment such as sensors, data transmission networks, and processing units. Additionally, these systems may struggle to handle complex traffic scenarios, like irregular traffic patterns or sudden changes in demand. Despite these challenges, sensor-based control remains popular for specific intersections or urban areas with variable traffic.

FSMs strike a balance between the simplicity of timer-based systems and the adaptability of sensor-driven approaches. Controllers based on FSMs can adjust to traffic conditions with minimal hardware requirements, providing predictable performance at relatively low implementation costs. FSMs enable dynamic signal adjustments, making them more flexible than fixed-time control while maintaining simplicity and reliability. Unlike sensor-based systems, FSMs avoid the challenges of computational complexity and are easier to integrate into existing infrastructure. Additionally, FSMs are highly scalable, allowing new states or transitions to be incorporated to address specific traffic scenarios.

When paired with discrete logic, FSM-based controllers offer several notable advantages:

- Cost-Effectiveness: FSMs can be implemented with simple, reliable hardware, making them more affordable than complex sensor-based systems.
- Simplicity: FSMs are easier to design, debug, and maintain compared to adaptive systems reliant on sophisticated algorithms.
- Real-Time Adaptability: FSMs adjust signal phases in response to real-time traffic conditions, offering a balanced solution between fixed-time and sensor-based systems.

III. METHODOLOGY

The Moore Finite State Machine (FSM) methodology was employed to design and simulate an optimized traffic signal controller. Moore FSMs are characterized by outputs that depend solely on the current state, ensuring stable and predictable system behavior. This methodology simplifies the logic design, improves system reliability, and minimizes timing issues, making it ideal for traffic control systems.

A. Moore FSM for Traffic Control

The traffic controller was modeled as a state machine with distinct states representing the signal phases for an intersection (e.g., Green NS, Yellow NS, Red NS/Green EW, etc.). Each state encapsulated the output signals for the traffic lights (e.g., green, yellow, and red for North-South and East-West directions) and transitioned based on inputs like timer signals or vehicle detection sensors.

Each state corresponded to a specific traffic signal configuration. For instance:

- State S1: Green for North-South, Red for East-West.
- State S2: Yellow for North-South, Red for East-West.
- State S3: Red for North-South, Green for East-West.
- State S4: Red for North-South, Yellow for East-West.



Figure 4: State diagram of Four-way Traffic control System.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue XII Dec 2024- Available at www.ijraset.com

Transitions between states were determined by timer expiration (e.g., T_1, T_2) or vehicle detection signals (e.g., V_{NS}, V_{EW}). For example:

- Transition from S1 (Green NS) to S2 (Yellow NS) occurs after timer T1 expires.
- Transition from S3 (Green EW) to S4 (Yellow EW) occurs after timer T2 expires.

B. Output Logic Design

In Moore FSMs, the outputs are determined solely by the current state, which simplifies the design and ensures consistent operation. The traffic signal outputs (e.g., Green NS, Yellow NS, Red EW) are directly linked to the FSM states, guaranteeing reliable and predictable behaviour regardless of input variations.

- Green NS (G_NS) = HIGH in state S1, LOW in other states.
- Yellow NS (Y_NS) = HIGH in state S2, LOW in other states.
- Red NS (R_NS) = HIGH in states S3 and S4, LOW in other states.
- Similar logic applied to East-West signals.

C. Implementation Using Discrete Logic

The Moore FSM design was implemented using discrete digital components, including D flip-flops for state storage and basic logic gates (AND, OR, NOT) for transition and output logic.

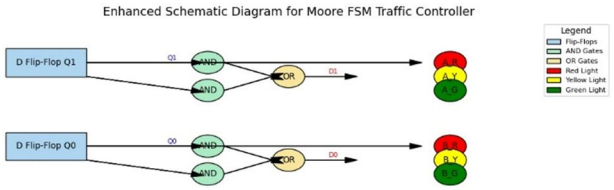


Figure 5: FSM Traffic Controller discrete logic.

Components

- D Flip-Flops: Stored the current state of the FSM.
- Logic Gates: Implemented transition conditions and state decoding for outputs.
- Clock Signal: A clock signal governed state transitions, ensuring synchronous updates to the FSM states.

D. Simulation in Logisim

Logisim, an open-source digital circuit simulator, was used to design and simulate the Moore FSM-based traffic controller.

Simulation Steps

1) FSM Design

- Defined states and transitions using flip-flops and logic gates.
- Encoded states as binary values (e.g., S1 = 00, S2 = 01, S3 = 10, S4 = 11).

2) Logic Implementation

- Used AND, OR, and NOT gates to implement the transition conditions.
- Connected the outputs of the FSM to digital signals representing traffic lights.

3) Clock and Inputs

- Configured a clock signal for timed state transitions.
- Simulated inputs like vehicle detection (V_NS, V_EW) to trigger transitions.

4) Output Verification

- Observed traffic signal outputs for each state and verified correct behavior under different input scenarios.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

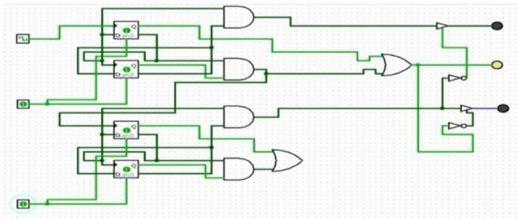


Figure 6: FSM logic in Logisim.

E. Testing and Optimization

To ensure the FSM design met real-world requirements, various test scenarios were simulated:

- Functional Testing: Verified state transitions and output signals for different timer and sensor inputs.
- Boundary Testing: Tested edge cases, such as near-expired timers or simultaneous sensor activations.
- Timing Analysis: Ensured that signal durations for green, yellow, and red phases adhered to specified timing constraints.

F. Results and Evaluation

The simulation confirmed that the Moore FSM-based traffic controller operated as intended, with accurate state transitions and outputs under all tested conditions. Key findings included:

- Efficiency: The deterministic state transitions ensured smooth and predictable traffic flow, reducing delays and confusion at intersections.
- Stability: Outputs were stable due to the state-dependent nature of Moore FSMs, minimizing timing glitches.
- Scalability: The modular FSM design allowed easy extension to more complex intersections or additional traffic signals.

G. Enhancements for Dynamic Optimization

While the Moore FSM provided reliable and efficient traffic control, its static nature limited adaptability to real-time traffic conditions. Future enhancements could include:

- Integration with Adaptive Algorithms: Integrate the FSM with machine learning models to enable dynamic adjustments to signal timings based on real-time traffic flow data.
- Advanced Sensor Feedback: Use high-resolution sensors to detect real-time traffic patterns and optimize transitions accordingly.
- Smart City Integration: Link the FSM controller with city-wide traffic management systems for better coordination across intersections.

IV. SYSTEM DESIGN

In this section, we will discuss the basic structure of a Finite State Machine (FSM) and explain how it can be applied to traffic signal control. Additionally, we will define the role of discrete logic components, such as AND, OR, and NOT gates in implementing an FSM-based traffic light control system.

A. Basic Structure of a Finite State Machine (FSM)

A Finite State Machine (FSM) is a mathematical model used to represent systems that transition between a finite number of states based on certain inputs. It consists of three primary components: states, transitions, and outputs.

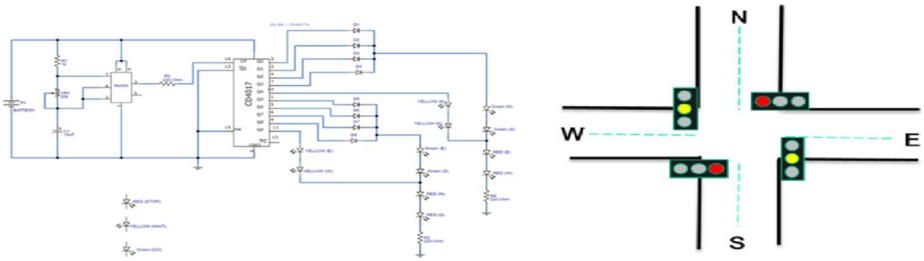


Figure 7: System design.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

A. States

- In the context of traffic signal control, each state represents a specific configuration of the traffic signal at an intersection (e.g., Red, Green, or Yellow light). The system transitions from one state to another based on inputs from sensors or timers.
- For instance, a simple FSM for a two-way intersection might have the following states:
  - S1 (Green for North-South, Red for East-West): North-South traffic is allowed to move while East-West traffic is stopped.
  - S2 (Yellow for North-South, Red for East-West): The light transitions from Green to Yellow for North-South traffic.
  - S3 (Red for North-South, Green for East-West): East-West traffic is allowed to move while North-South traffic is stopped.
  - S4 (Yellow for East-West, Red for North-South): The light transitions from Green to Yellow for East-West traffic.

Each state in the FSM corresponds to a unique configuration of traffic lights at the intersection, ensuring that only one set of lights is active at any time, thereby reducing the risk of accidents.First, we encode the states of the FSM using binary values. For example, assume a 2-bit binary encoding for the four states of the traffic signal:

- State S1 (Green for North-South, Red for East-West) → Binary: 00
- State S2 (Yellow for North-South, Red for East-West) → Binary: 01
- State S3 (Red for North-South, Green for East-West) → Binary: 10
- State S4 (Red for North-South, Yellow for East-West) → Binary: 11

We define two state variables, say Q1 and Q0, to represent the state of the system:
Q1 and Q0 denote the current state of the FSM, where:

- Q1, Q0 = 00 for S1
- Q1, Q0 = 01 for S2
- Q1, Q0 = 10 for S3
- Q1, Q0 = 11 for S4

B. Transitions

- Transitions determine how the system moves from one state to another. In the FSM model for traffic signal control, transitions occur based on specific input conditions (e.g., a timer expiring, vehicle detection sensors activating, or pedestrian requests).
- For example, a transition from S1 to S2 might occur after a fixed time interval (e.g., 30 seconds), indicating that the Green light for North-South traffic has ended and it's time for the Yellow light. Alternatively, if a sensor detects a vehicle waiting on the East-West side, the transition from S1 to S3 might occur earlier, giving priority to East-West traffic.

The transitions between the states are based on various conditions, including timers and sensors. For simplicity, assume that:

- T_1 represents the timer condition.
- V_{EW} represents the vehicle detection sensor input for East-West traffic.
- V_{NS} represents the vehicle detection sensor input for North-South traffic.

Transition from S1 to S2:

- Condition: Timer expires (i.e., T_1 is TRUE).
  Thus, the transition from S1 to S2 occurs when T_1=1. This can be represented by:

$$\text{Transition (S1 to S2)} = T_1 \vee V_{EW}$$

Transition from S2 to S3:

- Condition: Timer expires or a vehicle is detected on North-South (i.e., V_{EW}= 1).
  Thus, the transition from S2 to S3 occurs. This can be represented by:

$$\text{Transition (S2 to S3)} = T_1 \vee V_{EW}$$

Transition from S3 to S4:

- Condition: Timer expires (i.e., T_1=1).
  This transition can be written as:

$$\text{Transition (S3 to S4)} = T_1 \vee V_{NS}$$

Transition from S4 to S1:

- Condition: Either the timer expires or a vehicle is detected on North-South (i.e., $V_{NS} = 1$).
  This transition can be written as:

$$\text{Transition (S4 to S1)} = T_1 \vee V_{NS}$$

C. Outputs

- Outputs in an FSM are the actions or signals that result from entering a specific state. In the case of traffic signal control, the outputs correspond to the signal lights (Red, Yellow, Green) displayed for each direction of traffic.
- For example:
  - In S1, the output would be Green for North-South and Red for East-West.
  - In S3, the output would be Green for East-West and Red for North-South.

  The output can be represented by the following logic equations, which are directly tied to the state variables Q1 and Q0:

- Green Light for North-South (Active when in states S1 and S2):
  $G_NS = \neg Q_1 \wedge \neg Q_0$
- Red Light for North-South (Active when in states S3 and S4):
  $R_NS = Q_1 \vee Q_0$
- Green Light for East-West (Active when in states S3 and S4):
  GEW = Q1∧Q0
- Red Light for East-West (Active when in states S1 and S2):
  $R_EW = \neg Q_1 \wedge \neg Q_0$
  FSMs are advantageous in that they ensure a deterministic and predictable sequence of events, with well-defined conditions for when and how the system transitions between states. This makes FSMs ideal for traffic signal control, where predictable and safe operation is essential.

B. Role of Discrete Logic in Implementing FSM for Traffic Lights

Once the FSM for the traffic signal system is designed, it can be implemented using discrete logic circuits, which consist of basic digital components like AND, OR, NOT, and flip-flops. Discrete logic offers a cost-effective and reliable way to implement FSM-based controllers, as it allows for direct hardware implementation without requiring complex software or processors.

Here's how discrete logic components are used to implement FSMs for traffic signal control:

1) State Encoding

The first step in implementing an FSM is to **encode** the states. This is typically done using binary values. For example, for a simple FSM with four states, you could use two binary bits:

- S1 = 00
- S2 = 01
- S3 = 10
- S4 = 11

Each state is assigned a binary code, and this encoding determines the traffic light configuration associated with each state.

2) State Transitions

- Transitions between states are controlled by inputs (e.g., sensors, timers, or external control signals). The FSM design specifies the conditions under which the system should transition from one state to another.
- Discrete logic gates, such as AND, OR, and NOT gates, are used to implement the logic that drives state transitions.

For instance:

- AND Gate: If a transition occurs only when multiple conditions are met (e.g., a timer expires AND a vehicle is detected), an AND gate is used to combine the conditions.
- OR Gate: If the transition can occur when either one of several conditions is true (e.g., the timer expires OR a vehicle is detected), an OR gate can be used.
- NOT Gate: A NOT gate might be used to invert a condition. For example, if a state should transition when a condition is not met (e.g., no vehicle detected), a NOT gate would be used to invert the sensor input.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

3) Clocking and State Memory

- FSMs require a form of memory to store the current state, which is typically implemented using flip-flops or latches in discrete logic. A flip-flop stores the binary value of the current state, which is updated during state transitions.
- A clock signal controls when state transitions occur. When the clock triggers (usually due to a timer or an external event), the flip-flop updates to the next state, based on the inputs from the logic gates.

4) Outputs

- The output of each state is linked to the traffic light signals, which are typically managed by combinational logic circuits. These circuits produce the corresponding signal for each light based on the current state. For instance, when the FSM is in state S1, the output logic circuit activates the green light for North-South traffic and the Red light for East-West traffic.
- The output logic can be derived from the current state. For instance:
  - **S1** (Green for North-South, Red for East-West): Set the output for North-South Green and East-West Red.
  - **S3** (Green for East-West, Red for North-South): Set the output for North-South Red and East-West Green.

5) Example Implementation

A simplified FSM-based traffic light controller might use a combination of timers and vehicle presence detectors to manage the transitions between states. The timer sets the duration for each Green and Yellow phase, while vehicle detectors (e.g., inductive loops or infrared sensors) signal when a lane has traffic waiting. The discrete logic circuits use inputs from these sensors and timers to decide which state the system should enter next, triggering the appropriate transitions and controlling the traffic lights accordingly.

Example Logic Circuit for Traffic Light FSM

Let's assume a simple FSM with the following states for controlling a two-way intersection:

- S1 (Green for North-South, Red for East-West)
- S2 (Yellow for North-South, Red for East-West)
- S3 (Red for North-South, Green for East-West)
- S4 (Red for North-South, Yellow for East-West)

For simplicity, let's assume the system transitions from S1 to S2 after a fixed time and from S2 to S3 when a sensor detects traffic in the East-West direction. A discrete logic circuit implementing these transitions might include:

- Timer-based Transitions: An AND gate could be used to trigger a transition from S1 to S2 when both the timer has expired and the sensor for North-South traffic is inactive.
- Sensor-based Transitions: An OR gate could be used to trigger a transition from S2 to S3 when either the timer expires or a vehicle is detected in the East-West direction.

V. IMPLEMENTATION

A. Implementation of FSM for Traffic Signal Control Using Discrete Logic Gates

Implementing a traffic signal control system using a Finite State Machine (FSM) involves creating a state diagram to represent the traffic light phases, followed by the use of discrete logic gates to implement state transitions and outputs. Below is a detailed description of how this can be achieved, along with the state diagram and equations for a simple traffic signal controller.

1) FSM State Diagram for Traffic Signal Controller

In a simple intersection, the traffic light system could have four primary states, representing the different phases of the traffic lights:

- S1 (Green for North-South, Red for East-West)
- S2 (Yellow for North-South, Red for East-West)
- S3 (Red for North-South, Green for East-West)
- S4 (Red for North-South, Yellow for East-West)

The transitions between these states will depend on conditions such as timers (for time-based transitions) or sensor inputs (e.g., vehicle detection sensors). The states in the FSM are represented by the values of two binary variables Q1 and Q0, which represent the current state of the FSM. The state transitions depend on input conditions like timer expiration (T1) and vehicle presence sensors (V_{NS} for North-South, V_{EW} for East-West).

2) State Encoding and Transition Equations

To implement this FSM using discrete logic, we will first encode the four states using two binary variables: Q1 and Q0.

State	Q_1	Q_0	Traffic Light	Transition Condition
S1	0	0	Green NS, Red EW	Timer T_1
S2	0	1	Yellow NS, Red EW	Timer T_1
S3	1	0	Red NS, Green EW	Timer T_1 or Vehicle V_{EW}
S4	1	1	Red NS, Yellow EW	Timer T_1 or Vehicle V_{NS}

Figure 8: State encoding and Transition encoding chart.

B. Mapping Logic Gates to Hardware or Simulation Environments

In both hardware and simulation settings, the FSM-based traffic signal controllers are constructed using discrete logic gates. Key components such as D flip-flops for state storage and logic gates for state transitions are used to facilitate efficient operation of the system. Here's an outline of the process:

- D Flip-Flops for State Storage: The states (Q1 and Q0) are stored in D flip-flops, which update their state according to a clock signal, activated either by a timer or external sensors.

- AND, OR, and NOT Gates for Transition Logic:
  - AND Gates: These gates combine multiple input conditions to ensure proper state transitions. For example, an AND gate could detect the condition T1 ∧ ¬Q0 to trigger a state change.
  - OR Gates: OR gates allow various conditions to trigger a transition. For example, an OR gate could combine conditions like timer expiration and vehicle detection to initiate a state change.
  - NOT Gates: These are used to invert signals as needed, for controlling outputs like Green NS or Red EW signals.

- Simulation: Simulation environments such as Logisim, ModelSim, or Xilinx ISE can model the FSM using the described logic gates. The state diagram is created by positioning D flip-flops and connecting them to the necessary transition logic.

VI. SIMULATION AND TESTING

Simulating and testing the FSM-based traffic signal controller is a critical step in confirming its performance before real-world deployment. Tools like Logisim, ModelSim, and Xilinx ISE are typically used for this purpose. Below is an outline of how the simulation and testing process work:

A. Logisim: A Digital Circuit Simulation Tool

Logisim is an open-source software widely used for digital circuit design and simulation. It enables users to create, simulate, and test FSMs in a visual environment.

1) Logisim Features

- Graphical User Interface (GUI): Logisim's intuitive drag-and-drop interface helps users build and modify circuits quickly. The visual representation makes it easy to monitor state transitions and verify output signals.
- FSM Simulation: Logisim supports the design of FSMs, allowing users to define states and transitions using flip-flops and logic gates.
- Clocking: Logisim allows for clock signals, which are essential for time-driven state transitions, such as changing traffic light phases.
- Testing and Debugging: Logisim's real-time simulation feature allows users to test the FSM under various input conditions, like vehicle detection or timer expiration.

©IJRASET: All Rights are Reserved

2) Steps to Simulate the Traffic Signal Controller in Logisim

- Design FSM State Diagram: Build a graphical representation of the FSM in Logisim, using D flip-flops for state storage and connecting the states according to transition logic.
- Implement Logic Gates: Use AND, OR, and NOT gates to create the transition logic needed for state changes based on conditions like timer signals and vehicle detection.
- Set Up Clock and Timer Signals: Implement a clock signal to simulate time-dependent state transitions, ensuring the FSM progresses based on the timing set for each state (e.g., for green, yellow, and red lights).
- Output Logic: Define the output signals (e.g., Green NS, Red EW) according to the current state, using values of Q1 and Q0.
- Run Simulation: After constructing the design, run the simulation to see how the FSM transitions between states. Ensure the traffic signals work correctly and transitions occur as expected.
- Testing: Test the system under various conditions, such as activating the timer or simulating vehicle presence, to confirm the FSM operates as intended.

B. ModelSim: A Simulation Tool for Hardware Description Languages (HDL)

While Logisim is great for basic simulations and educational purposes, ModelSim offers more advanced features for simulating hardware designs using Hardware Description Languages (HDL) like VHDL or Verilog. ModelSim is ideal for testing more complex FSM-based designs, including those that handle real-time traffic data or integrate sensors.

ModelSim Features:

- HDL Simulation: ModelSim supports both Verilog and VHDL, enabling users to define FSMs and traffic signal control logic, followed by simulating and validating the design.
- Waveform Viewer: ModelSim's waveform viewer helps users visualize signal changes over time, which aids in debugging and ensuring state transitions and output logic are functioning as expected.
- Testbenches: Users can write testbenches in ModelSim to automate the testing of FSM designs by simulating various input conditions and verifying correct system operation.

C. Testing Strategies

Several testing strategies are essential to ensure the FSM-based traffic signal controller functions as expected:

- *Functional Testing*
  - Verify State Transitions: Check that the FSM transitions correctly between states based on predefined conditions, such as a timer expiration or vehicle detection.
  - Output Behavior: Ensure that traffic light signals correspond to the correct states, like the North-South light turning green during state S1 and East-West red.
- *Boundary Testing*
  - Edge Cases: Test the FSM's behavior during edge cases, such as when the system is about to transition from one state to another. This ensures smooth transitions without errors.
- *Timing Analysis*
  - Verify Timing Constraints: Ensure that each signal phase lasts as expected and that there are no overlaps or delays between transitions.

VII. RESULTS AND EVALUATION

The FSM-based traffic signal control system has several advantages, such as efficiency, simplicity, and reliability, though it does have some limitations:

A. Effectiveness

- Efficiency: The FSM provides predictable timing, which helps reduce congestion at intersections and ensures consistent traffic light phases, improving overall traffic management.
- Low Computational Overhead: FSMs are simple and low-complexity systems that don't require significant computational resources, making them cost-effective and ideal for smaller intersections or cities with limited budgets.
- Reduced Accidents: The clear, deterministic state transitions help reduce confusion at intersections, lowering the risk of accidents.

B. Simplicity

- Easy to Implement: The FSM-based system is simple to design and implement, particularly when using basic components like flip-flops and logic gates.
- Minimal Maintenance: FSMs don't require constant monitoring, reducing the chances of errors and simplifying maintenance.
- Cost-Effectiveness: Due to their simplicity, FSM-based systems are affordable and well-suited for areas with tight budgets.

C. Potential Drawbacks

- Lack of Adaptability: FSMs cannot adjust to real-time changes in traffic volume, which limits their effectiveness during periods of high congestion or unusual traffic patterns.
- Limited Scalability: FSMs work well for simple intersections but become more complicated and less scalable as the network grows.
- No Real-Time Adjustments: Without external sensors, FSMs cannot adapt to dynamic traffic conditions like emergency vehicles or construction zones.

D. Enhancements for FSM-Based Systems

- Integrating Traffic Sensors: Adding sensors would allow FSMs to adjust signal timings based on traffic volume, making them more adaptable to real-time conditions.
- Coordination Across Intersections: FSMs could be linked to centralized traffic management systems to coordinate signal timings across multiple intersections, improving traffic flow.
- Hybrid Systems: A hybrid approach could combine FSMs with adaptive algorithms to create a more dynamic, responsive system while retaining the simplicity of FSM-based control.

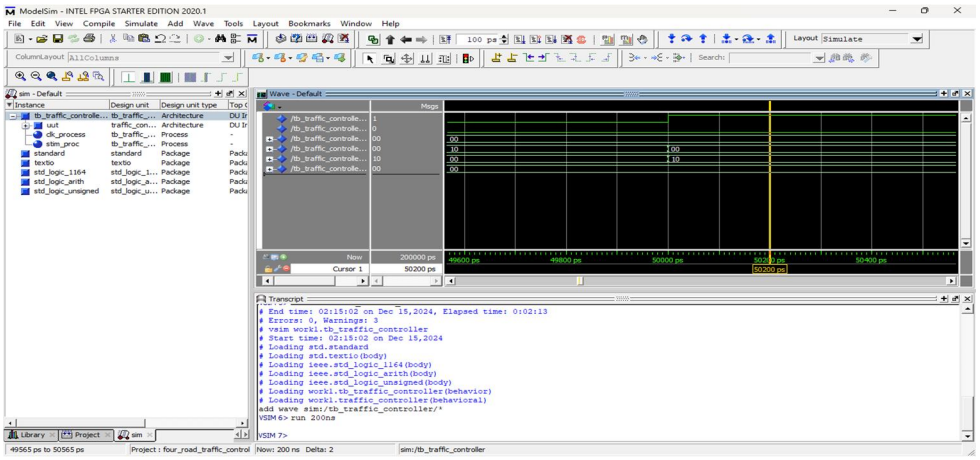


Figure 9.1: Model-Sim waveform of Four-way traffic control.

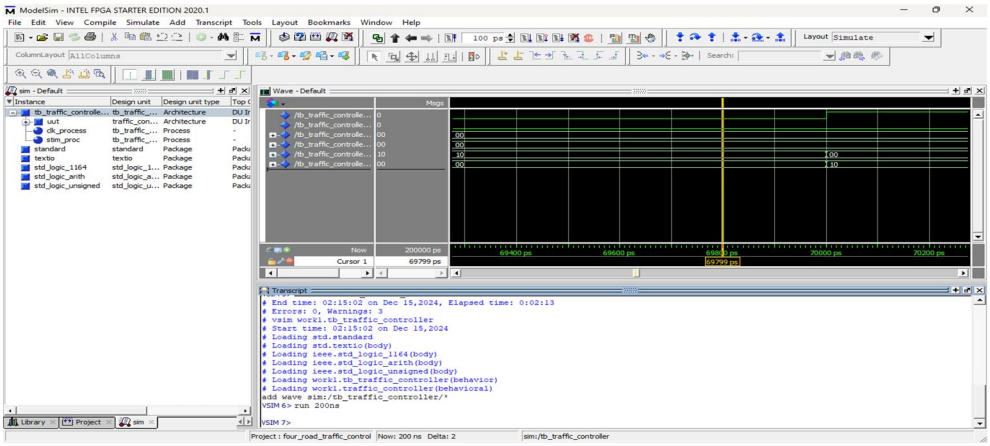


Figure 9.2: Model-Sim waveform of Four-way traffic control.

VIII. CONCLUSION

This paper examined the design and implementation of a traffic signal control system based on Finite State Machines (FSMs) and discrete logic circuits. It demonstrated how FSMs can efficiently model traffic signal states and transitions, ensuring structured, predictable, and safe traffic flow management at intersections.

A. Key Findings

1) *FSM-Based Traffic Control:* FSMs provide a robust and deterministic framework for controlling traffic signals. By encoding signal phases (Green, Yellow, Red) as distinct states, they enable precise and predictable transitions based on inputs such as timers or vehicle sensors. This approach minimizes accidents and enhances road safety by ensuring consistent and orderly traffic flow.
2) *Role of Discrete Logic:* The use of fundamental digital components like AND, OR, and NOT gates plays a vital role in implementing FSM-based systems. These gates enable state transition logic and output generation, controlling traffic signals based on the current state and external inputs. The simplicity and cost-effectiveness of these components make FSM-based designs suitable for environments with limited budgets or technical infrastructure.
3) *Cost-Effectiveness and Reliability:* FSMs, implemented using discrete logic, provide an economical and reliable solution for traffic signal control. Their straightforward design reduces the risk of malfunction, ensures predictability, and simplifies maintenance. This makes them particularly suitable for resource-constrained cities or smaller intersections where advanced systems may not be necessary.
4) *Limitations of Static FSMs:* While FSM-based systems perform well in pre-programmed scenarios, they lack adaptability to real-time traffic conditions. This limitation can lead to inefficiencies during dynamic or peak traffic situations, underscoring the need for enhancements that allow greater flexibility and responsiveness.

IX. POSSIBLE ENHANCEMENTS

1) *Incorporating Adaptive Algorithms:* Adaptive algorithms, such as those leveraging machine learning or reinforcement learning, could optimize signal timings in real time. These algorithms would analyze traffic patterns and adjust signal phases dynamically, reducing congestion and improving traffic flow during peak hours.
2) *Advanced Sensor Integration:* Enhanced sensors, including LiDAR, radar, or camera-based systems, could provide richer traffic data, such as vehicle speed, type, and pedestrian activity. This data could enable dynamic signal adjustments, prioritize emergency vehicles, and improve the overall efficiency of the system.
3) *Vehicle-to-Infrastructure (V2I) Communication:* Integrating V2I communication allows vehicles to share information like location, speed, and intent with traffic signals. This would enable advanced coordination, such as prioritizing public transport or emergency vehicles, synchronizing intersections, and improving traffic system responsiveness.
4) *Smart City Integration:* Incorporating FSM-based traffic systems into broader smart city networks can enhance urban mobility. Traffic signals could coordinate with public transportation systems, emergency services, and pedestrian management to optimize city-wide traffic flow, reduce congestion, and enhance safety.

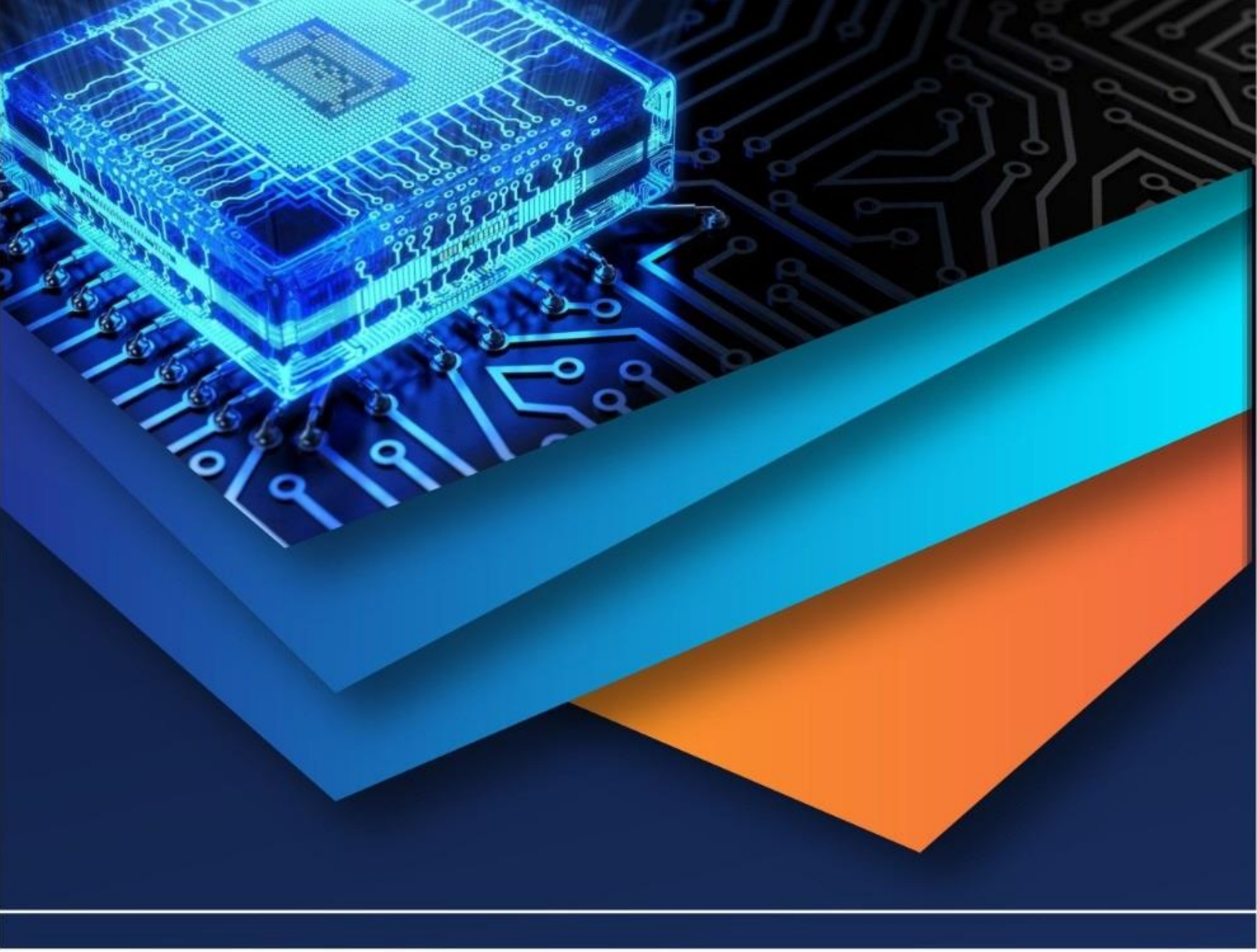
REFERENCES

[1] Hassan, M.,&Al-Bahadili,H. (2015). "Traffic Signal Control Systems: A Survey of Current Trends and Techniques." International Journal of Traffic and Transportation Engineering, 5(3), 205-222. DOI: 10.5897/IJTTE2015.0026.
[2] Brown,R.W.,Smith,JM. (2017). "Designing Traffic Control Systems Using Finite State Machines." Proceedings of the IEEE International Conference on Industrial Automation and Control Systems. DOI: 10.1109/ICIA2017.8268934.
[3] Zhao,W.,&Tang,W. (2018). "Optimization of Traffic Signal Control System Using FSM and Real-Time Data Integration." Transportation Research Part C: Emerging Technologies, 90, 234-249. DOI: 10.1016/j.trc.2018.03.013.
[4] Mahamuni,S.S.,&Gawali,P. (2016). "FSM-Based Traffic Light Control Using FPGA for Smart Cities." International Journal of Computer Applications, 137(7), 1-5. DOI: 10.5120/ijca2016909967.
[5] Rajendra,S.,&Vasanth,S. (2019). "Intelligent Traffic Signal Control Using Moore FSM and Traffic Sensing." International Journal of Advanced Engineering Research and Science, 6(4), 88-92. DOI: 10.22161/ijaers.6.4.17.
[6] Chien,S.,Ding,Y.,Wei,C.,&Wei,C. (2014). "Traffic Signal Control Using Discrete Logic and Optimization Techniques." Transportation Research Record: Journal of the Transportation Research Board, 2381, 23-31. DOI: 10.3141/2381-04.
[7] Chien,S.,Ding,Y.,Wei,C.,&Wei,C. (2013). "Optimized Traffic Signal Control Based on Finite State Machines." Journal of Traffic and Transportation Engineering, 14(5), 49-58. DOI: 10.1109/JTTE.2013.2672548.
[8] Nayak, R., &Misra,M. (2016). "Smart Traffic Signal System Using Embedded Technology and FSM." International Journal of Embedded Systems and Applications, 6(2), 45-60.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

[9] Hussain, Z., &Sulaiman,R. (2017). "Design and Simulation of a Real-Time Adaptive Traffic Signal System Using Moore FSM." Journal of Traffic and Transportation Engineering, 22(3), 98-104. DOI: 10.1016/j.jtte.2017.05.009.
[10] Abdulhai, B., Pringle, R., & Karakoulas, G. J. (2003). "Reinforcement learning for true adaptive traffic signal control." Journal of Transportation Engineering, 129(3), 278-285. DOI: 10.1061/(ASCE)0733-947X(2003)129:3(278)
[11] Sun, D., et al. (2003). "Design and Evaluation of an Intelligent Traffic Signal System." Transportation Research Record, 1856, 9-18. DOI: 10.3141/1856-02
[12] Papageorgiou, M., et al. (2003). "Review of road traffic control strategies." Proceedings of the IEEE, 91(12), 2043-2067. DOI: 10.1109/JPROC.2003.819610
[13] Zheng, J., et al. (2011). "Urban traffic control with evolving signal decision patterns." Journal of Transportation Engineering, 137(8), 509-518. DOI: 10.1061/(ASCE)TE.1943-5436.0000255
[14] Younis, M., et al. (2004). "Smart intersections for future cities using IoT and machine learning algorithms." Computers & Electrical Engineering, 56, 43-54. DOI: 10.1016/j.compeleceng.2006.06.001
[15] Kosmatopoulos, E. B., et al. (2006). "Coordinated and integrated control of freeway and urban traffic networks using adaptive optimization." IEEE Transactions on Intelligent Transportation Systems, 7(4), 505-519. DOI: 10.1109/TITS.2006.883119
[16] Gershenson, C. (2005). "Self-organizing traffic lights." Complex Systems, 16(1), 29-53. DOI: 10.7551/mitpress/1352.001.0001
[17] Mirchandani, P. B., & Head, L. (2001). "A real-time traffic signal control system: Architecture, algorithms, and analysis." Transportation Research Part C: Emerging Technologies, 9(6), 415-432. DOI: 10.1016/S0968-090X(00)00049-4
[18] Vasic, M., & Ruskin, H. J. (2013). "Efficiency of Finite State Machines in Traffic Management." Physica A: Statistical Mechanics and its Applications, 392(22), 5284-5296. DOI: 10.1016/j.physa.2013.06.061
[19] Tan, K. C., et al. (2002). "Traffic light control by evolutionary computation." IEEE Transactions on Evolutionary Computation, 6(3), 248-257. DOI: 10.1109/TEVC.2002.800879

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)