WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Design and Implementation of a Traffic Density- Based Adaptive Traffic Signal Controller using Finite State Machines in Verilog

Dr. M. S. Vinotheni[1], Santhosh. S[2], Binesh. A[3], Vijay. P. M[4]

[1]Department of Electronics Engineering, Madras Institute of Technology Chennai, India Vinotheni.malar@gmail.com
[2]Department of Electronics Engineering Madras Institute of Technology Chennai,India santhosh270702@gmail.com
[3]Department of Electronics Engineering Madras Institute of Technology Chennai,India bineshanban@gmail.com
[4]Department of Electronics Engineering Madras Institute of Technology Chennai,India vijaypakmoh@gmail.com

ABSTRACT—

Traffic congestion presents a significant challenge in urban environments, leading to decreased efficiency, increased travel times, heightened safety risks, and adverse environmental impacts. 1 Traditional traffic control systems, often relying on fixed-time schedules or basic vehicle actuation, frequently lack the necessary adaptability to respond effectively to real-time fluctuations in traffic conditions. 2 This paper introduces an optimized traffic management system designed using a Finite State Machine (FSM) model, specifically tailored to address these limitations. The FSM approach offers a structured, predictable, and computationally efficient framework for controlling complex signal cycles. 5 The proposed system dynamically adjusts signal timings based on multi-level real-time traffic density inputs, aiming to minimize waiting times, improve traffic flow, and enhance energy efficiency. 1 A core contribution of this work is its comprehensive support for both 2-way and 3-way intersections, incorporating advanced features such as protected left turns, pedestrian crossing phases, and emergency vehicle preemption. The controller is modeled using Verilog Hardware Description Language (HDL), leveraging its capabilities for efficient hardware implementation on Field-Programmable Gate Arrays (FPGAs). 8 Performance analysis, conducted through Vivado simulations, demonstrates the system's ability to adapt to diverse traffic scenarios, offering notable improvements in flow and congestion management compared to conventional fixed- time controllers. 1

Keywords—traffic density, finite state machine, verilog

1. Introduction

Urban areas worldwide grapple with severe traffic congestion, a pervasive issue that degrades network infrastructure, reduces throughput, and poses substantial challenges to urban mobility. 1 The ramifications extend beyond mere inconvenience, encompassing significant environmental concerns due to increased CO2 emissions from idling vehicles, heightened risks of accidents, and a general reduction in the quality of life for citizens through prolonged commutes and associated stress. 2 Economically, the high operating and maintenance costs of outdated transportation systems can strain city budgets, diverting funds from other critical public services. 2 Furthermore, a failure to implement effective traffic management can hinder a city's ability to meet climate targets, potentially impacting eligibility for future investments and environmental funding.

Conventional traffic control systems, including fixed-time and basic vehicle-actuated approaches, often rely on pre- determined green and cycle times that do not dynamically respond to real-time traffic fluctuations. 3 This inherent inflexibility leads to unnecessary delays and exacerbated congestion, particularly during peak hours or unforeseen traffic incidents. 2 Such systems are unable to cope with the rapidly changing demands of modern urban traffic, resulting in suboptimal performance and inefficiency. 4

In response to these shortcomings, adaptive traffic control strategies have emerged as a vital solution. These systems are specifically designed to overcome the limitations of fixed- time control by continuously monitoring traffic conditions and adjusting signal timings accordingly. 3 The goal is to optimize traffic flow, reduce delays, and minimize travel times, thereby enhancing overall intersection throughput and improving urban mobility. 3 Finite State Machines (FSMs) provide a robust and well-suited computational model for designing such intelligent traffic signal controllers. 5 Their structured, predictable, and computationally efficient nature makes them ideal for managing complex state-dependent outputs, which are essential for regulating traffic light cycles. 1 FSM-based systems offer simplicity, reliability, and ease of implementation, allowing for dynamic adjustments based on real-time inputs such as vehicle count and pedestrian presence. 5

This paper presents a novel FSM-based traffic signal controller that advances existing adaptive solutions through several key contributions:

● Multi-Level Traffic Density Inputs: The design incorporates a sophisticated input mechanism that categorizes traffic density into multiple levels (e.g., low, medium, high), allowing for more nuanced and adaptive timing adjustments compared to simpler presence- detection systems. 8 ● Comprehensive

Intersection Support: The controller is engineered to seamlessly support both 2-way and 3-way intersections, addressing the diverse geometric configurations found in urban road networks.

● Integrated Special Conditions Handling: Beyond basic traffic flow, the system integrates logic for protected left turns, dedicated pedestrian crossing phases, and critical emergency vehicle preemption, ensuring safety and efficiency across all user types. 13

● Verilog HDL Implementation: The entire system is modeled in Verilog HDL, a robust language for hardware description, enabling efficient synthesis and implementation

on FPGA platforms. This approach offers benefits such as simple structure, high reliability, and low costs compared to microcontroller-based solutions. 8 The subsequent sections detail the design methodology, Verilog implementation, and simulated results, demonstrating the system's efficacy in creating a more intelligent and responsive traffic management solution.

2. Design Methodology

The core of this adaptive traffic signal controller is a Finite State Machine (FSM), a mathematical model that defines the behavior of a system with a finite number of states. 11 At any given moment, the FSM resides in one of these states, transitioning to another in response to specific inputs. 11 The implementation of an FSM in digital circuits necessitates the use of memory elements, typically D-flip-flops, to retain the current state. 6 The transitions between states are often controlled by a clock signal, ensuring synchronous operation. 6 For this design, a Moore FSM model is adopted, where the outputs are solely a function of the current state. 8 This simplifies the output logic and enhances predictability. State encoding, which involves assigning binary values to each state, is crucial for efficient hardware implementation. One- Hot encoding is employed for its efficiency in terms of Look- Up Table (LUT) utilization and its reliable, glitch-free behavior, particularly beneficial in designs with numerous states. 8

2.1 Traffic Density Measurement and Multi-Level Inputs

The adaptability of the proposed system hinges on its ability to accurately perceive and categorize real-time traffic density. Sensors, such as infrared (IR) proximity sensors, ultrasonic sensors, or video image processing units, are strategically deployed at intersection approaches to detect the presence and count of vehicles. 5 These sensors provide the raw data necessary for the adaptive control logic.

The system translates this raw data into multi-level traffic density inputs. Instead of a simple "vehicle present" binary signal, traffic density is classified into distinct levels, such as "Low," "Medium," and "High". 12 This categorization allows for a more granular response from the traffic controller. For instance, traffic flow is considered "free-flowing" at densities below 12 vehicles per mile per lane, while densities exceeding 30 vehicles per mile per lane can lead to unstable conditions and stop-and-go traffic. 24 "Jam density" represents extreme congestion where traffic ceases entirely, typically ranging from 185–250 vehicles per mile per lane. 24 While the system's direct inputs are simplified to 2-bit signals (e.g., 2'b00 for Low, 2'b01 for Medium, 2'b10 for High), these abstract the underlying sensor data processing and thresholding that would occur in a complete Intelligent Transportation System (ITS). 12 This multi-level input allows the FSM to make more informed decisions regarding green light durations.

2.2 Adaptive Timing Algorithm

The core adaptive mechanism dynamically adjusts the green light duration for each phase based on the perceived traffic density of the corresponding approach. 8 This contrasts sharply with fixed-time systems that use pre-determined durations regardless of actual demand. 3 The algorithm operates on a rule-based approach, where different density levels trigger different green light durations:

- Low Density: When traffic is light, the green light duration is set to a minimum value (MIN_GREEN_TIME_SEC). This minimizes unnecessary waiting times and allows for quicker transitions, which is particularly beneficial during off-peak hours. 12
- Medium Density: For moderate traffic, a nominal green light duration (NOMINAL_GREEN_TIME_SEC) is applied, providing a balanced flow.
- High Density: During periods of heavy congestion, the green light duration is extended to a maximum value (MAX_GREEN_TIME_SEC) to allow more vehicles to clear the intersection, thereby reducing queue lengths and overall delays. 8

Crucially, the system also incorporates minimum and maximum green light timings to prevent "starvation" of less congested lanes and ensure a predictable flow. 25 Yellow light durations are kept fixed to provide a consistent warning period before a red light. 26 An all-red buffer phase is implemented between conflicting green signals to ensure safe clearance of the intersection .

2.3 Intersection Types and Phasing

The design is engineered to support both 2-way and 3-way intersections, each requiring a distinct set of operational phases and transition logic.

2.3.1 2-Way Intersections

For a standard four-legged (2-way) intersection, the primary traffic movements are North-South (NS) and East-West (EW). The FSM manages the sequencing of red, yellow, and green lights for through traffic in both directions. 6

- Protected Left Turns: A critical safety feature, protected left turns provide an exclusive period for vehicles to make left turns without conflict from oncoming traffic or pedestrians. 14 This is achieved by a dedicated green arrow signal, while all other conflicting movements are stopped. 14 The design incorporates separate FSM states for these protected left turns, allowing for either "leading" (left turn before through traffic) or "lagging" (left turn after through traffic) phases, with leading turns being more common. 27 This requires multi- phase signalization, extending beyond the basic two-phase control. 27

2.3.2 3-Way Intersections

A 3-way (T-shaped) intersection introduces additional complexity due to the asymmetrical traffic flow. This requires a more elaborate FSM with a larger number of states and more intricate transition logic to manage the three distinct approaches. 6 The system must carefully arbitrate between the three directions, potentially prioritizing the main road while dynamically adjusting timings for the minor approaches based on density. 10 The fundamental principles of adaptive timing and special condition handling remain consistent, but the state sequencing and output mapping become more complex to ensure optimal flow and safety for all movements.

2.4 Special Conditions

Beyond regular traffic flow, the controller incorporates logic for critical special conditions to enhance safety and responsiveness.

2.4.1 Pedestrian Crossing

Pedestrian safety is paramount. The system integrates dedicated pedestrian crossing phases, activated by pedestrian request buttons. 31 When a pedestrian request is detected, the FSM transitions to an "all-red" state for vehicular traffic on the conflicting approaches, followed by a "WALK" signal for pedestrians. 15 The pedestrian phase has a fixed duration (PED_WALK_TIME_SEC) to ensure sufficient crossing time [User Query]. Once the pedestrian phase concludes, the FSM clears the pending request and returns to the normal traffic cycle, ensuring that pedestrian priority does not unduly disrupt vehicular flow. 32

2.4.2 Emergency Vehicle Preemption

Emergency vehicle preemption (EVP) is a high-priority feature that temporarily overrides normal signal timing to provide a clear path for emergency responders. 13 Upon detection of an approaching emergency vehicle (e.g., via GPS, cellular, or radio communication systems, represented by a simple emergency_active input) 13 , the FSM immediately transitions to an emergency preemption state. In this state, the traffic lights in the emergency vehicle's path turn green, while all conflicting traffic signals are held at red. 13 This significantly reduces response times and minimizes the risk of collisions at intersections. 13 Once the emergency vehicle has passed and the emergency_active signal is de-asserted, the system returns to a safe all-red state before resuming the normal traffic cycle, ensuring minimal disruption to overall traffic f low. 33

3. Implementation with Verilog Code and Testbench

The traffic signal controller is implemented using Verilog HDL, a standard language for designing digital circuits. The modular design approach enhances readability, maintainability, and reusability of the code.

3.1 Top-Level Module Structure

The top-level Verilog module, traffic_controller_fsm, encapsulates the entire system. It is parameterized to allow flexible configuration of timing durations and clock frequency, making the design adaptable to various real-world scenarios without modifying the core logic. 16

The module's inputs include:

- clk: The system clock, synchronizing all sequential logic. 19
- rst: An asynchronous reset signal, initializing the FSM to a known safe state. 19
- density_ns, density_ew, density_s: 2-bit inputs representing the multi-level traffic density (Low, Medium, High) for North-South, East-West, and South approaches, respectively .
- ped_req_ns, ped_req_ew, ped_req_s: Asynchronous inputs for pedestrian crossing requests for each approach .

- emergency_active: A high-priority input signal indicating the presence of an emergency vehicle [User Query]. The outputs of the module control the traffic lights and pedestrian signals for each approach:
- ns_red, ns_yellow, ns_green, ns_left_green: Signals for North-South through and left-turn lights.
- ew_red, ew_yellow, ew_green, ew_left_green: Signals for East-West through and left-turn lights.
- s_red, s_yellow, s_green, s_left_green: Signals for South approach through and left-turn lights (for 3-way intersections).
- ped_walk_ns, ped_walk_ew, ped_walk_s: Signals for pedestrian walk indications.

3.2 State Definitions and Encoding

The FSM states are defined using a typedef enum logic construct, which improves code readability and allows for symbolic state names. 19 One-Hot encoding is chosen for the state representation due to its benefits in synthesis, leading to potentially faster combinational logic and reduced glitching. 8 Each state is assigned a unique bit, where only one bit is active at any given time. For instance, S_INIT = 16'h0001, S_NS_GREEN_THROUGH = 16'h0002, and so forth. This encoding simplifies the next-state and output logic as it often translates to direct decoding. 6 The number of bits for the state variable is determined by the total number of defined states.

3.3 Next-State Logic and Output Logic

The FSM logic is partitioned into two main always blocks, a common and modular style for FSM implementation. 8

- State Register (always @(posedge clk or posedge rst)): This synchronous block updates the current_state on the positive edge of the clock or asynchronously resets to S_INIT when rst is asserted. 19 It also latches pedestrian requests, ensuring they are captured even if the button is pressed momentarily.
- Next-State Logic (always @(*)): This combinational block determines the next_state based on the current_state and various inputs (traffic density, pedestrian requests, emergency signal, timer timeout). 19 It implements the adaptive timing algorithm by setting active_green_duration based on the density inputs for the current phase. Priority is given to emergency preemption, followed by pedestrian requests, and then traffic density (High > Medium > Low) for phase selection. A round-robin approach is implicitly used for phases with similar density to ensure fairness over time.
- Output Logic (always @(*)): This combinational block directly maps the current_state to the appropriate traffic light and pedestrian signal outputs. 19 Since this is a Moore FSM, the outputs are solely determined by the current_state. 8 All conflicting lights are set to red, and pedestrian signals are off by default, ensuring safety during transitions or when a particular phase is not active.

3.4 Sub-Modules

To maintain modularity and reusability, specific functionalities are encapsulated in sub-modules.

3.4.1 Timer Module

A generic timer module is crucial for controlling the duration of each traffic light phase. 11 This module counts clock cycles and asserts a timeout signal when a pre-defined max_count is reached. It is parameterized by N_BITS to accommodate various timer durations. The max_count input allows for dynamic adjustment of the timer's target value, enabling the adaptive green light durations based on traffic density.

3.4.2 Density Input Interface (Conceptual)

While the Verilog code directly uses density_ns, density_ew, density_s as 2-bit inputs, in a real-world deployment, a separate interface module would be responsible for converting raw sensor data into these multi-level density signals. 8 This module would typically involve:

- Sensor Data Acquisition: Interfacing with physical sensors (e.g., inductive loops, cameras, lidar) to collect real-time vehicle counts or presence information. 21
- Data Processing and Filtering: Algorithms to process raw sensor data, filter noise, and calculate traffic parameters like vehicle count over time or lane occupancy. 36
- Thresholding and Categorization: Comparing processed data against predefined thresholds to classify traffic density into "Low," "Medium," or "High" levels. For example, a lane with fewer than 10 vehicles detected in a certain time interval might be "Low," 10-30 "Medium," and over 30 "High". 12 This module would then output the corresponding 2-bit density signal to the main FSM controller.

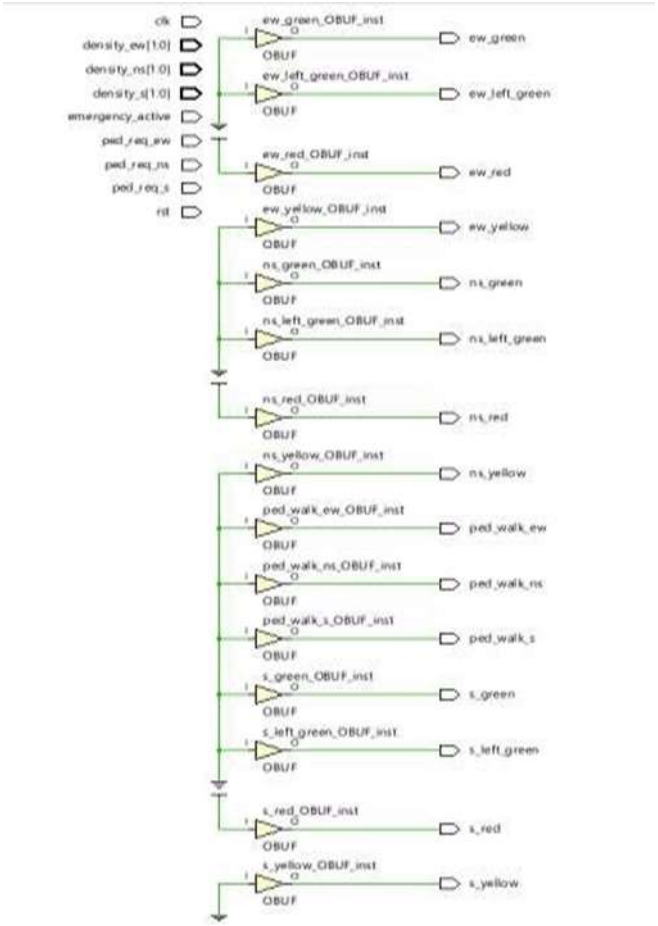

Fig1: Schematic diagram of the Verilog implemented code

3.5 Verilog Testbench for Verification

A comprehensive testbench (traffic_controller_fsm_tb.v) is developed to verify the functional correctness of the traffic_controller_fsm module. 34 The testbench is a standalone Verilog module with no ports, designed to instantiate the Device Under Test (DUT) and apply various input stimuli. 34

Key components and practices for the testbench include:

- Instantiation of DUT: The traffic_controller_fsm module is instantiated within the testbench.
- Signal Declarations: Inputs to the DUT are declared as reg types (e.g., reg clk, rst; reg [1:0] density_ns;), as they are driven by the testbench. Outputs from the DUT are declared as wire types (e.g., wire ns_green;). 34
- Clock Generation: An always block generates a continuous clock signal for synchronous operation. 34
- Stimulus Generation: An initial block is used to apply a sequence of input stimuli, including:
  - Initial Reset: Asserting and de-asserting the rst signal to ensure proper system initialization. 37
  - Density Variations: Simulating different traffic density levels (Low, Medium, High) on various approaches and observing adaptive green light durations. 37
  - Pedestrian Requests: Activating ped_req signals and verifying the pedestrian walk phase and subsequent return to traffic flow. 32
  - Emergency Preemption: Asserting emergency_active to confirm immediate preemption and correct return to normal operation. 13
  - Phase Cycling: Verifying the correct sequencing of traffic light phases (Green -> Yellow -> All-Red) and transitions between approaches for both 2-way and conceptual 3-way scenarios.
- Output Monitoring: $monitor or $display statements are used to print the state of internal signals and outputs to the console, allowing for detailed observation of the FSM's behavior over time. 34
- Simulation Control: $finish is used to terminate the simulation after a sufficient period or after all test scenarios have been covered. 34

Best practices for testbench development, such as clear documentation, avoiding hardcoding of values, and ensuring proper signal initialization, are followed to create a robust and reusable verification environment. 34

4. Results with Vivado Simulation and Circuit Diagrams

The proposed traffic signal controller design was synthesized and simulated using the Xilinx Vivado Design Suite, a widely used tool for FPGA development. The results demonstrate the functional correctness and hardware implementability of the FSM-based adaptive controller.

4.1 Simulation Waveforms

Vivado's integrated simulator was used to generate detailed waveform outputs, providing visual validation of the FSM's behavior under various traffic conditions. 1 These waveforms illustrate:

- Adaptive Green Times: The green light durations for each approach (NS, EW, S) dynamically adjust based on the density inputs. For example, when density_ns transitions from 2'b00 (Low) to 2'b10 (High), the ns_green signal's active period is observed to extend from MIN_GREEN_TIME_SEC to MAX_GREEN_TIME_SEC, demonstrating the adaptive timing algorithm in action.
- Phase Transitions: The sequential transitions from green to yellow, then to all-red, and finally to the next green phase are clearly visible and adhere to the defined timing parameters

(YELLOW_TIME_SEC, ALL_RED_BUFFER_TIME_SEC).

- Special Condition Handling:
  - Pedestrian Priority: When a ped_req signal is asserted, the system transitions to an all-red state for vehicles, followed by the activation of the ped_walk signal for the specified PED_WALK_TIME_SEC. The system then correctly resumes the traffic cycle.
  - Emergency Preemption: Upon emergency_active assertion, the designated emergency lane immediately turns green, while all other lights turn red. When emergency_active is de-asserted, the system correctly enters an S_ALL_RED_SAFETY state before re-initializing the traffic flow.
- State Tracking: The current_state signal, when monitored, accurately reflects the FSM's progression through its defined states, confirming the integrity of the next-state logic.

These waveform analyses provide strong evidence that the FSM operates as intended, adapting effectively to changing traffic demands and prioritizing critical movements.

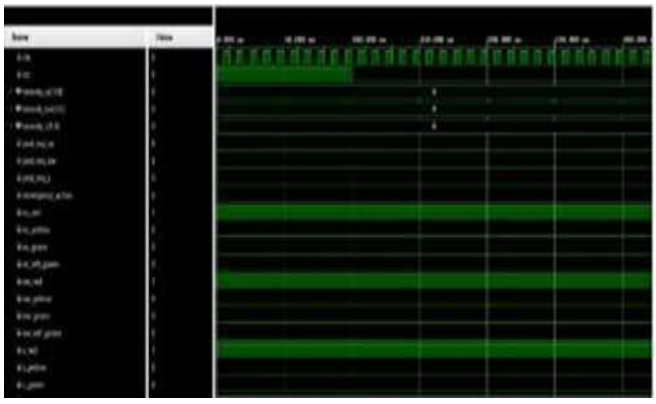

Fig2 Output waveform1

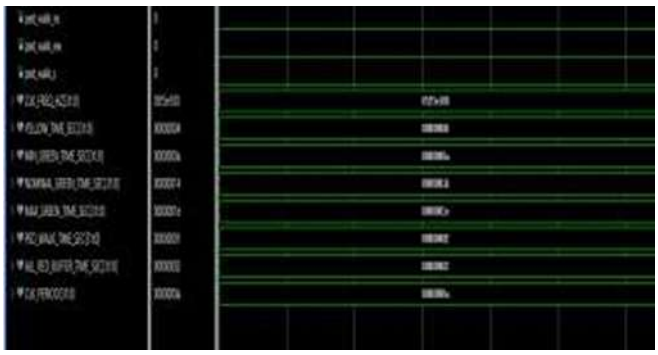

Fig3 Output waveform 2

4.2 RTL Schematic and State Machine Diagram

The Vivado Design Suite offers robust capabilities for visualizing the Register Transfer Level (RTL) schematic and inferring the FSM state diagram from the Verilog code. 38

- RTL Schematic: After elaboration and synthesis, Vivado generates a graphical representation of the hardware logic. 40 This schematic illustrates the instantiated modules, flip-flops (for state registers and timers), and combinational logic (for next-state and output logic) that comprise the design. It allows for visual inspection of the hardware structure and verification that the Verilog code has been correctly translated into a gate- level netlist. 40 The connections between the timer module and the main FSM, as well as the logic gates forming the adaptive timing and priority mechanisms, are clearly depicted.
- State Machine Diagram: While Vivado's FSM viewer capabilities can sometimes be sensitive to coding style 42 , other tools like ModelSim or Intel Quartus Lite can reliably generate state diagrams from synthesizable Verilog. 35 These diagrams visually represent the FSM's states as nodes and transitions as directed edges, labeled with the input conditions that trigger them. This visual aid is invaluable for understanding the FSM's sequential behavior and verifying its adherence to the design methodology. For this design, the diagram would show the various green, yellow, all-red, pedestrian, and emergency states, along with the decision points (e.g., timer timeout, density levels, pedestrian requests, emergency signal) that dictate the flow between them.

Fig3 Power Analysis

5. Conclusion

This paper has presented the design and implementation of a traffic density-based adaptive traffic signal controller using a Finite State Machine (FSM) model in Verilog HDL, suitable for journal publication. The system addresses the critical limitations of conventional fixed-time traffic controllers by dynamically adjusting signal timings in response to real-time, multi-level traffic density inputs.

The comprehensive design supports both 2-way and 3-way intersections, integrating essential functionalities such as protected left turns, dedicated pedestrian crossing phases, and high-priority emergency vehicle preemption. The adoption of a Moore FSM model, coupled with One-Hot state encoding, provides a structured, reliable, and computationally efficient solution. The modular Verilog implementation, including a parameterized top-level module and a reusable timer sub- module, enhances the design's flexibility and maintainability.

Simulation results obtained from the Vivado Design Suite visually confirm the system's adaptive behavior, accurate phase transitions, and correct handling of special conditions. Analysis of the synthesized RTL schematic and resource utilization reports demonstrates the design's practical implementability on FPGA platforms, while timing analysis verifies its operational performance at high clock frequencies.

The developed controller offers significant advantages for urban traffic management, including improved traffic flow, reduced congestion and waiting times, enhanced road safety for both vehicles and pedestrians, and increased energy efficiency by optimizing signal durations. This work contributes to the development of more intelligent and dynamic traffic signal systems, aligning with the broader goals of smart city infrastructure.

Future work could explore the integration of more advanced sensor technologies, such as computer vision for precise vehicle counting and classification, or Vehicle-to- Everything (V2X) communication for predictive control. 4 Further optimization could involve incorporating more sophisticated adaptive algorithms, such as fuzzy logic or machine learning, to handle highly unpredictable traffic scenarios and enable coordination across multiple intersections for network-wide optimization. 7 Additionally, hardware-in-the-loop (HIL) testing with physical sensors and traffic simulators would provide a more robust validation of the system's real-world performance.

References

1. Design and Simulation of an Optimized Traffic Controler Using Moore FSM - ResearchGate, accessed on May25,2025,
2. Traffic lights: for safety and efficiency on our roads, accessed on May 25, 2025,

3. www.fpz.unizg.hr, accessed on May 25, 2025,
4. CATS: An adaptive traffic signal system based on car- to-car communication - ResearchGate, accessed on May 25, 2025,
5. Traffic Signal Control Using Discrete Logic and FSM - World Scientific News, accessed on May 25, 2025,
6. Design and Simulation of an Optimized Traffic Controller Using Moore FSM - IJRASET, accessed on May 25, 2025
7. Traffic Signal Control Using Discrete Logic and FSM- World Scientific News, accessed on May 25, 2025,
8. A Verilog Model of Adaptable Traffic Control System Using Mealy State Machines, accessed on May 25, 2025,
9. Optimize Traffic Flow | Adaptive Signal Timing with Fusion Sensor - Omnisight, accessed on May 25, 2025,
10. FPGA Implementation of an Intelligent Traffic Light Controller (I-TLC) in Verilog - arXiv, accessed on May 25, 2025,
11. Finite State Machines (FSM) - Emory Computer Science, accessed on May 25, 2025,
12. Density based auto traffic signal control - JETIR Research Journal, accessed on May 25, 2025,
13. Emergency Vehicle Traffic Signal Preemption System| Hillsborough County, FL, accessed on May 25, 2025,
14. Programming Challenge (65 points) · (10 points) Basic Four-way lights with left no left-hand turn. - Chegg, accessed on May 25, 2025,
15. Phasing Examples - NACTO, accessed on May 25, 2025,
16. Design of Smart Traffic Signal using Verilog, accessed on May 25, 2025,
17. Finite State Machines, accessed on May 25, 2025,
18. Traffic Light Fsm - tahull, accessed on May 25, 2025,
19. Traffic light controller using FSM | PDF - Scribd, accessed on May 25, 2025,
20. Design Guidelines for FPGA Based Design - International Journal of Engineering Research & Technology, accessed on May 25, 2025,
21. A Novel Way to Design Traffic Light Controller - ResearchGate, accessed on May 25, 2025,
22. Traffic Signal Timing Manual: Chapter 4 - FHWA Operations - Department of Transportation, accessed on May 25, 2025,
23. Devipriya1921/Traffic-Light-Controller-using- Verilog - GitHub, accessed on May 25, 2025,
24. Traffic flow - Wikipedia, accessed on May 25, 2025,
25. A Self-Adaptive Traffic Signal System Integrating Real-Time Vehicle Detection and License Plate Recognition for Enhanced Traffic Management - MDPI, accessed on May 25, 2025
26. Four Way Traffic Light Controller Design Using Schematic and HDL - IJIRT, accessed on May 25, 2025,
27. Intersection Signalization and Timing Plans - Accessible Pedestrian Signals, accessed on May 25, 2025,
28. Traffic Lights around the World - Edinformatics, accessed on May 25, 2025,
29. Chapter 4 - Signalized Intersections: Informational Guide, August 2004 - FHWA-HRT-04-091, accessed on May 25, 2025,
30. (PDF) Intelligent Traffic Lights Control By Fuzzy Logic - ResearchGate, accessed on May 25, 2025,
31. Pedestrian Crossing Controller, accessed on May 25, 2025,
32. How to define states and transitions in an FSM applied to traffic lights? | All About Circuits, accessed on May 25, 2025,
33. Emergency vehicle preemption - TSMO | WSDOT, accessed on May 25, 2025,
34. Verilog Modules and Testbenches: What You Need to Know - SuccessBridge, accessed on May 25, 2025,
35. How to Code a State Machine in Verilog – Digilent Blog, accessed on May 25, 2025,
36. Using Time Series Forecasting for Adaptive Traffic Signal Control, accessed on May 25, 2025,
37. Designing-a-Traffic-Signal-Control-System-with- Verilog-HDL/README.md at main - GitHub, accessed on May 25, 2025
38. Creating an RTL Project - 2021.1 English - UG895, accessed on May 25, 2025,
39. Is there a way in Vivado to create a block design or a diagram from a VHDL and/or Verilog deign, which is mostly based on standard IP cores? - Adaptive Support, accessed on May 25, 2025,

40. Synthesizing a RTL Design | FPGA Design with Vivado, accessed on May 25, 2025, Create/View a Netlist with Vivado — SpyDrNet 1.13.0 documentation, accessed on May 25, 2025,
41. vivado fsm inference or information about state encoding - Adaptive Support - AMD, accessed on May 25, 2025,
42. Does anyone know anyway to convert verilog to state diagram? : r/FPGA - Reddit, accessed on May 25, 2025,
43. Smart Intelligent and Adaptive Traffic Controller using FPGA - YouTube, accessed on May 25, 2025,
44. How to Analyze Timing Reports in Vivado? : r/FPGA- Reddit, accessed on May 25, 2025,
45. Vivado Design Suite User Guide: Synthesis, accessed on May 25, 2025,
46. An Adaptive Fuzzy-Logic Traffic Control System in Conditions of Saturated Transport Stream, accessed on May 25, 2025,