

The Project Report  
Entitled  
**FPGA-BASED TRAFFIC LIGHT  
CONTROLLER & SMART PARKING  
SYSTEM**

Submitted in  
Partial Fulfilment for the award of the Degree of  
**Bachelor of Technology - (Electrical)**

by

**Mr. Vibhanshu Botke - (U21EE071)**

**Ms. Aditi Tapariya - (U21EE081)**

**Mr. Rishav Raj - (U21EE090)**

**Mr. Akshay Kankaria - (U21EE101)**

**Mr. Gourav Gupta - (U21EE113)**  
**(B. TECH. IV(DoEE), 7<sup>th</sup> Semester)**

Guided by

**Dr. Anandita Chowdhury**  
**Professor, DoEE**



DEPARTMENT OF ELECTRICAL ENGINEERING  
SARDAR VALLABHBHAI NATIONAL INSTITUTE OF TECHNOLOGY  
DECEMBER-2024



# Sardar Vallabhbhai National Institute Of Technology

Surat - 395 007, Gujarat, India

## DEPARTMENT OF ELECTRICAL ENGINEERING



## CERTIFICATE

This is to certify that the FINAL YEAR PROJECT REPORT entitled “**FPGA-BASED TRAFFIC LIGHT CONTROLLER & SMART PARKING SYSTEM**” is presented & submitted by Candidates **Mr. Vibhanshu Botke (U21EE071)**, **Ms. Aditi Tapariya (U21EE081)**, **Mr. Rishav Raj (U21EE090)**, **Mr. Akshay Kankaria (U21EE101)**, **Mr. Gourav Gupta (U21EE113)** of B.Tech. IV, 7th Semester in the partial fulfillment of the requirement for the award of B.Tech. Degree in Electrical Engineering for academic year 2024 - 25.

We have successfully and satisfactorily completed our Final Year Project in all respects. We certify that the work is comprehensive, complete and fit for evaluation.

**Dr. Anandita Chowdhury**  
Professor & Seminar Guide

Name of Examiners	Signature with Date
-------------------	---------------------

1. Dr. H.G Patel	_____
------------------	-------

2. Dr. Suresh Lakhimsetty	_____
---------------------------	-------

**Dr. P.B Darji**  
Head & Professor  
DoEE, SVNIT

Seal of The Department  
(December 2024)



# Acknowledgement

We would like to express our sincere gratitude and deep regards to director of SVNIT Dr. Anupam Shukla, our guide Dr. Anandita Chawdhury, Professor, Electrical Engineering Department, SVNIT, for giving us opportunity to gain and enhance our knowledge in this domain and present a final year project report, and for constantly guiding and directing us throughout my work.

We also want to express our gratitude to our department's head, Dr.P.B Darji, for his continuous support to us and for providing us the facilities at department for our project work.

We extend our heartfelt thanks to Dr. Suresh Lakhimsetty Sir, Dr. Pinalkumar J. Engineer Sir and Ms. Zahida Ma'am for their invaluable guidance and support during our project.

At last but not the least, We would like to thank our parents and our friends for their always being supportive, motivating, and morally inspiring.



# **Abstract**

This report proposes the design and implementation of two innovative FPGA-based systems: a real-time traffic light control algorithm and a smart parking solution. The traffic light controller, implemented on the Artix-7 FPGA, leverages parallel processing to dynamically optimize signal timings at a three-way intersection with two lanes per direction, based on real-time traffic and pedestrian data. This results in reduced congestion, minimized wait times, and enhanced traffic efficiency. The smart parking system utilizes sensor data for real-time tracking of available parking spots, alongside password-based authentication for secure access control, ensuring efficient space allocation and improved user experience. Both systems, simulated on Vivado, demonstrate the scalability, flexibility, and efficiency of FPGA technology in addressing modern urban challenges, such as traffic management and smart city infrastructure, offering promising solutions for future urban development.





# Table of Contents

	Page
<b>Acknowledgements</b> . . . . .	v
<b>Abstract</b> . . . . .	vii
<b>Table of Contents</b> . . . . .	ix
<b>List of Figures</b> . . . . .	xi
<b>Chapters</b>	
1 Introduction . . . . .	1
1.1 Overview . . . . .	1
1.2 Motivation . . . . .	2
1.3 Objective . . . . .	3
1.4 Organisation . . . . .	3
2 Literature Review . . . . .	5
3 System Architecture . . . . .	7
3.1 Traffic Light Controller Design . . . . .	7
3.1.1 Problem Analysis and Road Layout . . . . .	7
3.1.2 Traffic Signal Indications . . . . .	7
3.1.3 Time Delays and State Transitions . . . . .	8
3.1.4 State Diagram . . . . .	9
3.1.5 State Table . . . . .	9
3.1.6 Timing Control and State Logic . . . . .	10
3.2 Smart Parking System Design . . . . .	11
3.2.1 Basic Functionality . . . . .	11
3.2.2 Safety Procedures and Time Synchronization . . . . .	12
3.3 Components Used . . . . .	13
3.3.1 Hardware . . . . .	13
3.3.2 Software . . . . .	16
4 Project Workflow in Vivado Design Suite 2024.1 . . . . .	21
4.0.1 Key Notes on Constraints . . . . .	27
5 Observations and Results . . . . .	29
5.1 Traffic Light Controller . . . . .	29
5.2 Smart Parking System . . . . .	31
6 Conclusion . . . . .	33
6.1 Future Scope . . . . .	34
<b>Bibliography</b> . . . . .	35



# List of Figures

3.1	Road Directions . . . . .	7
3.2	Different States . . . . .	8
3.3	State Diagram . . . . .	9
3.4	State Table . . . . .	10
3.5	Setup of Parking Area . . . . .	11
3.6	Algorithm for Smart Parking . . . . .	13
3.7	Artix-7 FPGA . . . . .	14
3.8	Digital Signal Oscilloscope . . . . .	15
3.9	Other Components . . . . .	16
3.10	Vivado Design Suite 2024.1 . . . . .	17
3.11	Icarus Verilog . . . . .	18
3.12	GTKWave . . . . .	19
4.1	Vivado Start-Up Window . . . . .	21
4.2	Create Project Dialog . . . . .	22
4.3	Enter Project Name . . . . .	23
4.4	Select Project Type . . . . .	23
4.5	Add Sources . . . . .	24
4.6	Add Constraint Files . . . . .	24
4.7	Select FPGA Board . . . . .	25
4.8	Create Project Summary . . . . .	26
4.9	Vivado Project Window . . . . .	26
4.10	XDC File for Traffic Light Controller . . . . .	27
4.11	Schematic for Traffic Light Controller . . . . .	28
5.1	Hardware implemented Traffic Light Controller with different states . . . . .	30
5.2	GTKWave Waveforms for Traffic Light Controller . . . . .	30
5.3	GTKWave Waveforms for Smart Parking . . . . .	31



# Chapter 1

## Introduction

Traffic light controllers play a crucial role in managing traffic flow and ensuring intersection safety. Traditional systems that rely on microcontrollers often struggle with complex real-time and parallel operations. To address these limitations, Field Programmable Gate Arrays (FPGAs) have emerged as a more effective solution. FPGAs provide benefits such as hardware-level parallelism, adaptability, and fast processing speeds, making them well-suited for dynamic and adaptive traffic management. By transitioning to FPGA-based designs, traffic control systems achieve improved efficiency, scalability, and reliability, effectively handling a wide range of traffic conditions.

### 1.1 Overview

Traffic signal controllers play a vital role in optimizing traffic flow and ensuring safety at intersections, serving as key components of modern traffic management systems. While traditional systems often rely on microcontrollers, these are well-suited for basic, predefined tasks but face challenges with complex real-time operations, scalability, and parallel processing. As traffic density and dynamic requirements grow, the limitations of sequential processing in microcontroller-based systems become increasingly apparent.

Field Programmable Gate Arrays (FPGAs) offer a powerful alternative, providing hardware-level parallelism, high-speed processing, and adaptability. Unlike microcontrollers, FPGAs excel at handling multiple tasks simultaneously, enabling real-time responses to dynamic factors such as traffic density and sensor inputs. This capability makes FPGAs particularly effective in addressing the evolving demands of contemporary traffic systems. For example, FPGA-based systems have been shown to deliver up to ten times faster processing speeds in real-time applications compared to microcontrollers, with significantly reduced latency. Furthermore, their reconfigurability allows for system updates without hardware modifications, reducing maintenance costs and enhancing adaptability.

In traffic management, FPGAs have demonstrated exceptional performance, achieving up to 95% efficiency in managing varying traffic conditions compared to traditional microcontroller-based systems. Their capacity to process large datasets and execute complex algorithms—such as adaptive signal timing driven by machine learning or

heuristic methods—makes them particularly suited to the demands of modern urban traffic systems.

By harnessing these advantages, FPGA-based traffic light controllers represent a transformative approach to design, delivering greater reliability, scalability, and efficiency, especially in scenarios that require robust real-time decision-making and high-performance system operations.

## **1.2 Motivation**

The growing complexity of urban traffic systems and the shortcomings of conventional microcontroller-based solutions in meeting real-time and adaptive requirements are the driving forces behind the switch to FPGA-based traffic light controllers. Traffic systems must handle dynamic and unpredictable situations, such as shifting traffic volumes, emergency vehicle prioritizing, and peak-hour congestion, as cities grow and the number of vehicles increases. Because of their sequential processing and limited computational power, microcontroller-based systems are naturally limited in their ability to effectively handle these difficulties, which frequently leads to extended delays and decreased system responsiveness.

With their unmatched hardware parallelism that allows for the simultaneous execution of several operations, FPGAs offer a revolutionary option. In real-time traffic management, when operations including processing sensor data, dynamically modifying signal timings, and managing pedestrian crossings must all take place simultaneously, this capacity is crucial. Additionally, because FPGAs can be reconfigured, systems can incorporate sophisticated algorithms or adjust to changing traffic patterns without needing to reinvent the hardware. FPGAs are an affordable and future-ready solution for contemporary traffic systems because of their versatility, which guarantees long-term scalability.

The ability of FPGAs to facilitate sophisticated traffic control methods, such machine learning-based adaptive signal optimization, is another important incentive. FPGA-based controllers may proactively modify signal timings and predict congestion patterns by utilizing predictive algorithms. This improves overall traffic flow and drastically cuts down on waiting times. Additionally, FPGA-driven systems contribute to sustainability goals by minimizing fuel consumption and vehicular emissions through more efficient traffic management.

The shift towards FPGAs is also motivated by their robustness and reliability in high-performance applications. Unlike software-dependent microcontrollers, FPGA-based

systems are less prone to software bottlenecks and can achieve higher fault tolerance, ensuring consistent operation even in high-traffic environments. These advantages, combined with the growing need for intelligent and sustainable traffic solutions, drive the adoption of FPGAs in traffic light controller design, offering a pathway to more efficient, adaptable, and eco-friendly urban transportation systems.

## **1.3 Objective**

The objective of this project is to use the FPGA Artix-7 platform to design and implement two integrated systems: a smart parking system and a real-time traffic light control algorithm.

By optimizing traffic flow according to current conditions, the traffic light controller was created to effectively operate a three-way intersection with two lanes in each direction. The system simultaneously interprets sensor data, modifies signal timings, and prioritizes pedestrian traffic by utilizing the FPGA's parallel processing capabilities. This reduces traffic, shortens wait times, and improves traffic efficiency. This system's capacity to adjust to changing traffic levels and enhance intersection performance was demonstrated during successful testing and implementation on the Artix-7 FPGA board.

Additionally, a smart parking system was developed, incorporating sensor-based tracking of parking space occupancy along with password-based authentication for security. This system ensures secure access to the parking facility, monitors real-time availability of parking spots, and allocates spaces efficiently based on user preferences and time of arrival. The smart parking algorithm was designed to optimize the parking process, reduce the time spent searching for available spots, and enhance overall user experience. The system was also simulated on Vivado to ensure accurate operation and seamless integration with the FPGA platform.

These systems collectively highlight the advantages of FPGA-based solutions in urban management applications, providing scalable, efficient, and adaptable infrastructure for modern smart cities.

## **1.4 Organisation**

Chapter 1 - This chapter includes a brief introduction to the report highlighting the basic concepts used in it.

Chapter 2 - This chapter talks about the literature survey about the various papers that were useful during the course of research for this report.

Chapter 3 - This chapter focuses on the hardware utilized, software employed, methodology adopted, and the implementation of the smart parking system.

Chapter 4 - This chapter focuses on implementing the project on Vivado Design Suite 2024.1 software.

Chapter 5 - This chapter includes the results obtained and the observations derived from our model.

Chapter 6 - This part frames the Conclusion for the report and the further possible works.



## Chapter 2

# Literature Review

The following section provides an overview of some of the relevant and significant research that has been done on the implementation of an advanced traffic light system using FPGA hardware.

The research carried out by Poorna Shree.T et. al. [1] introduces an FPGA-based Smart Traffic Light Control System (STLCS) to tackle the persistent problem of traffic congestion in urban areas. Their work focuses on using sensors to detect the presence and volume of vehicles, which then informs a timing algorithm to dynamically adjust the signal durations. This system prioritizes smooth traffic flow and error minimization by incorporating fixed maximum and minimum green light durations into the algorithm. The system is implemented using the Spartan-3E FPGA trainer kit and simulated with ISim software, and the design uses Verilog HDL for functionality. A state-based control mechanism ensures systematic traffic regulation across four directions, and FPGA is used for its speed, performance, and flexibility in handling such tasks. In comparison, our research extends the scope of FPGA-based systems by not only addressing traffic light management but also integrating a smart parking solution. Our system, implemented on the Artix-7 FPGA, employs real-time data from vehicles and pedestrians to optimize signal timings dynamically, reducing congestion at a three-way intersection with two lanes per direction. Unlike the other study, which uses linear or electromagnetic sensors, our approach integrates parallel processing for enhanced efficiency.

The research done by D. Bhavana et. al. [2] focuses on designing a traffic light controller using FPGA technology to enhance traffic flow management in urban settings. Their project introduces a sequential machine modeled as a finite state machine to manage traffic signal switching. The system uses Verilog HDL for design and implementation on Nexys-2 FPGA hardware using Xilinx software. The functionality of the system was demonstrated through simulation waveforms and hardware outputs represented by LEDs and a 7-segment display. The key features of this research include managing traffic based on time intervals and implementing both day and night modes to adjust signal timings dynamically depending on traffic density. A significant aspect of their system is its flexibility to accommodate changes in traffic flow by manually adjusting states, ensuring compatibility with various road structures.

In this research published by Apoorva Banerjee [3], the research focuses on designing an Intelligent Traffic Light Controller (I-TLC) system for a four-way intersection using Verilog and implementing it on a Xilinx Spartan-3E FPGA. The study integrates sen-

sors to detect vehicle and pedestrian activity, enabling dynamic signal adjustments to minimize waiting times and improve traffic flow. The implementation utilizes a Moore finite state machine (FSM) for adaptive timing control, with states configured to prioritize main road traffic while accommodating side-road demands based on real-time sensor input. Simulations and hardware testing demonstrate the system's efficiency, with minimal hardware utilization and delay, highlighting the advantages of FPGAs in speed, reconfigurability, and parallel processing capabilities for such applications.

Kishore et. al. [4] conducted research and explored the development of a traffic light controller using FPGA technology to address traffic congestion at intersections. Their work focuses on a four-junction road system utilizing Verilog HDL for design and implementation. The system employs a finite state machine (FSM) to manage the sequence of red and green signals, alongside timers that display countdown durations for waiting vehicles. The proposed controller was implemented on the Artix-7 Nexys4 FPGA developmental kit, with a maximum operating frequency of 290 MHz, showcasing the benefits of FPGA in terms of speed, and resource optimization. Additionally, the paper suggests potential enhancements, such as integrating traffic density sensors and pedestrian crossing lights, emphasizing the adaptability of their design.

The research published by T. Bala Obula Reddy et. al. [5] presented an FPGA-based traffic light control system designed to address the challenges of urban traffic management. Their system employed infrared (IR) sensors to dynamically allocate signal timings based on traffic density, prioritizing emergency vehicles using sound sensors and capturing images of traffic violators with a camera module. The system was implemented using Verilog HDL and tested on Xilinx software, the authors highlighted the flexibility and efficiency of FPGA over conventional microcontrollers or microprocessors in managing traffic flow at four- and six-road intersections. Their design aimed to minimize waiting times, improve traffic throughput, and enhance safety through real-time responsiveness to changing traffic conditions.

# Chapter 3

## System Architecture

### 3.1 Traffic Light Controller Design

#### 3.1.1 Problem Analysis and Road Layout

The traffic light controller system is developed to manage a T-shaped intersection that experiences heavy vehicular traffic. The road layout consists of three primary directions (M1, MT, M2) and one secondary side route (S). The control logic is derived using state diagrams and state tables, ensuring a systematic representation of the traffic flow and the transitions between signal states. The intersection requires efficient traffic management due to its complexity, and six unique states (S1, S2, S3, S4, S5, S6) are defined to control traffic signals for the entire system.

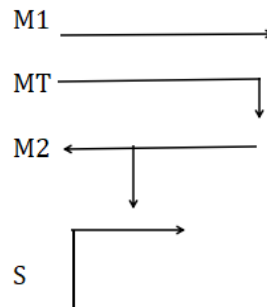


Figure 3.1: Road Directions

Each state corresponds to a specific configuration of traffic signals, balancing the flow of vehicles while minimizing congestion. This structure allows us to design the signal timing and transitions that address the dynamics of heavy traffic at the intersection.

#### 3.1.2 Traffic Signal Indications

To regulate traffic effectively, three types of signal lights are used, each conveying a specific message to vehicles at the intersection:

- **Green Light:** Indicates unrestricted movement for vehicles in the corresponding direction, ensuring a smooth flow of traffic.
- **Red Light:** Restricts vehicle movement, signaling that the route is blocked due to traffic congestion or prioritized flow in other directions.

- **Yellow Light:** Acts as a transitional phase, warning vehicles of an upcoming change in the signal from Green to Red or vice versa.

This combination of signal lights ensures clarity and allows traffic flow to be coordinated in a logical sequence, preventing confusion among drivers.

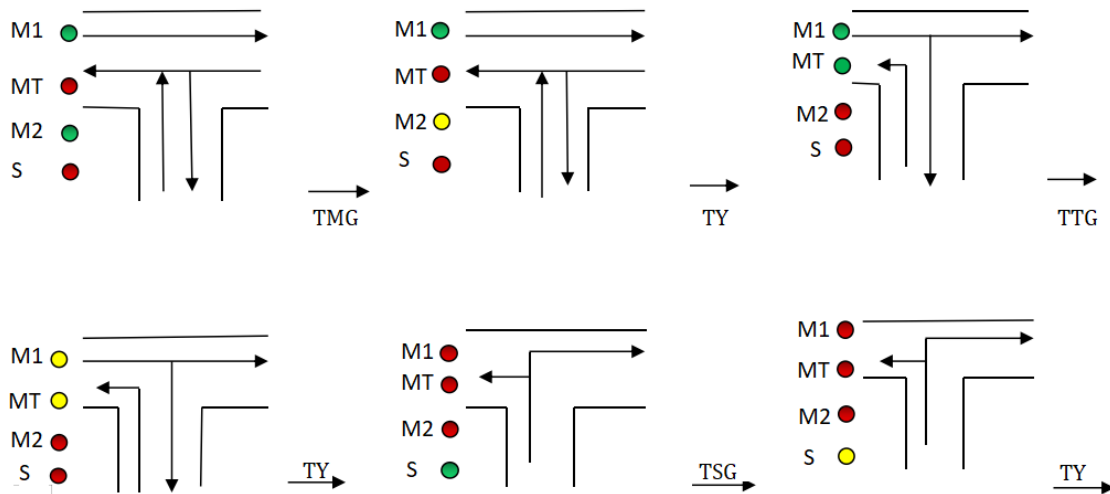


Figure 3.2: Different States

### 3.1.3 Time Delays and State Transitions

Each traffic light state is assigned a predefined time delay, which determines the duration for which the signal remains active before transitioning to the next state. The delays are carefully configured to optimize traffic flow, reduce waiting times, and manage congestion efficiently. The specific time delays for the state transitions are:

- **TMG (7 seconds):** Represents the time duration for the Green light in direction M1.
- **TY (2 seconds):** Accounts for the Yellow light's transitional phase between states.
- **TTG (5 seconds):** Indicates the Green light duration for the MT direction.
- **TSG (3 seconds):** Denotes the Green light duration for the S direction.

The transitions follow a cyclic sequence. For example:

- **From S1 to S2:** After TMG (7 seconds), the system transitions to the next state.
- **From S2 to S3:** After TY (2 seconds), the cycle moves forward, and so on.

This cycle ensures equitable traffic distribution across all directions while accounting for real-world traffic dynamics.

### 3.1.4 State Diagram

The state diagram (Figure 3.3) graphically represents the transitions between the six states of the traffic signal system. Each state corresponds to a unique configuration of signals across the four directions (M1, MT, M2, and S). For example:

- In S1, M1 and M2 are Green, while MT and S are Red.
- In S3, M1, MT, and M2 transition to specific configurations, as defined in the state table.

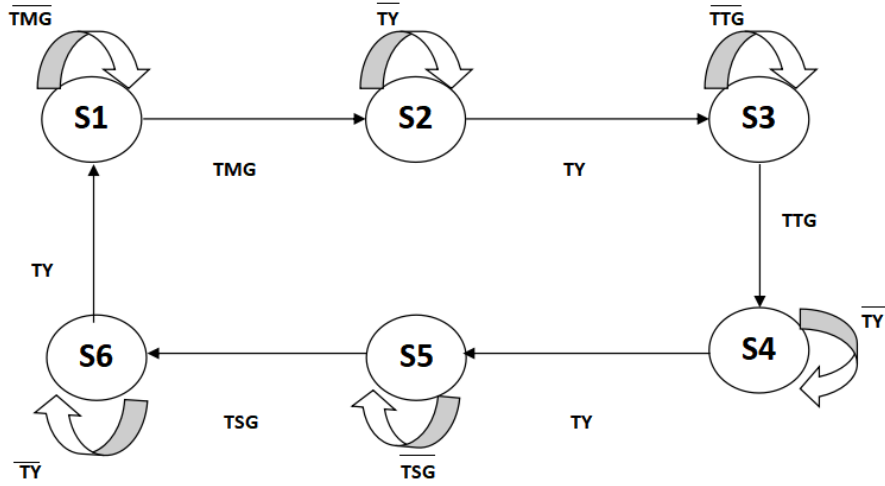


Figure 3.3: State Diagram

This diagram provides a concise view of how the system evolves over time, ensuring that all states are accounted for in the cycle.

### 3.1.5 State Table

A state table (Figure 3.4) complements the state diagram by providing detailed information about the signal configurations at each state. It uses the RYG (Red, Yellow, Green) coding scheme to represent the state of traffic signals in each direction:

$$S1(001) = \begin{cases} M1, & \text{Green}(RYG = 001) \\ MT, & \text{Red}(RYG = 100) \\ M2, & \text{Green}(RYG = 001) \\ S, & \text{Red}(RYG = 100) \end{cases} \quad (3.1)$$

$$S2(010) = \begin{cases} M1, & \text{Green}(RYG = 001) \\ MT, & \text{Red}(RYG = 100) \\ M2, & \text{Yellow}(RYG = 010) \\ S, & \text{Red}(RYG = 100) \end{cases} \quad (3.2)$$

This systematic table ensures a clear understanding of how traffic signal transitions are orchestrated across the six states.

PRESENT STATE	INPUT	NEXT STATE	ST	M1	M2	MT	S
ABC		A*B*C*		RYG	RYG	RYG	RYG
000	-	001	1	000	000	000	000
001	$\overline{\text{TMG}}$	001	0	001	001	100	100
	TMG	010	1				
010	$\overline{\text{TY}}$	010	0	001	010	100	100
	TY	011	1				
011	$\overline{\text{TTG}}$	011	0	001	100	001	100
	TTG	100	1				
100	$\overline{\text{TY}}$	100	0	010	100	010	100
	TY	101	1				
101	$\overline{\text{TSG}}$	101	0	100	100	100	001
	TSG	110	1				
110	$\overline{\text{TY}}$	110	0	100	100	100	010
	TY	001	1				
111	-	000	0	000	000	000	000

Figure 3.4: State Table

### 3.1.6 Timing Control and State Logic

The timing control is built into the state transitions using time delays and logical gates implemented through Verilog code. The controller ensures that the system operates in the following sequence:

1. S1 (M1 and M2 Green)  $\rightarrow$  S2 (M2 transitions to Yellow)
2. S2  $\rightarrow$  S3 (MT becomes Green)

3.  $S3 \rightarrow S4$  (MT transitions to Yellow)
4.  $S4 \rightarrow S5$  (S becomes Green)
5.  $S5 \rightarrow S6$  (S transitions to Yellow)
6.  $S6 \rightarrow S1$  (Cycle restarts)

This cyclic pattern is repeated indefinitely, ensuring seamless traffic management at the intersection.

## 3.2 Smart Parking System Design

By automating parking management, the smart parking system maximizes parking space usage, improves security, and ensures effective control of vehicle entry and exit. To control vehicle movement at the parking entrance and exit gates, the system uses sensor-based detection and password authentication techniques.

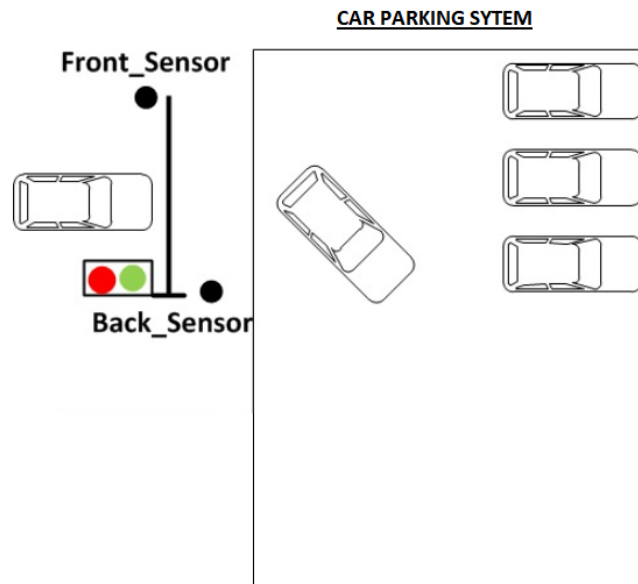


Figure 3.5: Setup of Parking Area

### 3.2.1 Basic Functionality

At the core of the smart parking system is the use of sensors and logic controllers to detect vehicles and manage gate operations. The following workflow outlines the basic operational mechanism:

- 1. Vehicle Detection at the Entrance :** A sensor installed at the parking lot entrance detects the presence of an incoming vehicle. Once the sensor is triggered, the sys-

tem initiates the process to authenticate the vehicle's entry. The detection mechanism ensures that only vehicles intending to enter the parking lot are processed, minimizing unnecessary operations.

**2. Entry Authentication by Password :** The system uses an input interface to ask the driver to enter a password after identifying a vehicle. By serving as an authentication method, this password makes sure that only cars with permission can enter the parking lot. Logic circuits created in Verilog HDL are used to implement the password validation procedure, which compares the input to a password that has been stored in the system's memory beforehand.

- The entrance gate opens automatically to let the car access the parking lot if the password is correct.
- The mechanism keeps the gate locked and asks the motorist to enter the proper password again if the password is incorrect. This guarantees the parking facility's security and stops unwanted access.

**3. Observation of Parking Spaces :** The technology uses sensors placed throughout the parking lot to track the vehicle's whereabouts as it arrives. Drivers can more effectively find available parking spaces by using this data for real-time parking spot monitoring.

**4. Synchronization and Exit Detection :** A comparable sensor-based detection system at the exit determines when a car is prepared to leave the parking lot. To prevent many cars from entering or exiting at the same time, which could lead to operational issues, the system synchronizes the entrance and leave procedures. For example: The system will momentarily shut the entrance gate to keep another car from entering if it detects a car leaving the parking lot. This prevents crowding at the entry or exit and guarantees efficient traffic flow.

**5. Password Needed to Leave :** In some setups, the system can also ask for a password at the exit gate to confirm that the car leaving the parking lot is the same one that was permitted to enter. Unauthorized or unintentional usage of the parking facility is prevented by this extra security measure.

### 3.2.2 Safety Procedures and Time Synchronization

The system incorporates time synchronization mechanisms for gate operation to further improve functionality. In order to account for vehicle movement and guarantee that the gates stay open or locked for an adequate amount of time, time delays are implemented. Safety measures, such sensors that track gate closure, are put in place to stop unintentional collisions or damage.





contemporary digital and embedded systems. This FPGA, which is based on a 28nm process, is designed to have a high logic density and great performance at low power consumption. With its high-speed serial interface and support for sophisticated digital designs, Artix-7's 101,440 logic cells allow for smooth integration with external systems and peripherals. Because of its adaptability, it is perfect for uses including communication systems, industrial automation, and real-time data processing.

Artix-7's memory and clocking capabilities are its most notable features. It supports SPI Flash for setup and storage, high-speed DDR3 SDRAM for data-intensive activities, and a variety of built-in clock inputs, including fixed sources operating at 50 MHz and 100 MHz. Applications like embedded vision, audio processing, and real-time control systems can benefit from its built-in DSP slices, which can offer outstanding performance for computationally taxing tasks like signal processing.

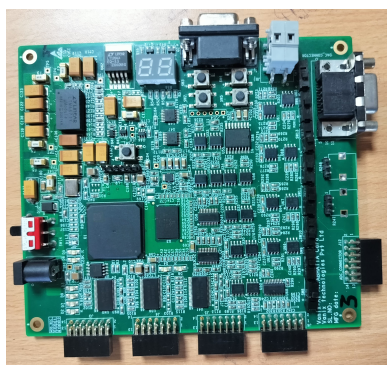


Figure 3.7: Artix-7 FPGA

The Aryabhata Card is a unique hardware platform that gives engineers and researchers access to a feature-rich development environment by utilizing the Artix-7 FPGA's capabilities. For networking applications, it combines the Artix-7 XC7A100T or XC7A35T FPGA with peripherals including 1GB DDR3 memory, 32Mb SPI Flash, and 10/100 Mbps Ethernet. The card also has a Bluetooth interface for wireless connectivity and supports UART for serial communication. Real-time monitoring and control in industrial and research applications are made possible by the 8-channel ADC and DAC modules, which convert analog signals to digital and vice versa.

Additionally, the Aryabhata Card has user-friendly interfaces that offer versatility for debugging and prototyping, such as a seven-segment display, DIP switches, status LEDs, and 48 GPIO pins that may be configured. Because of this, the platform is ideal for a variety of tasks, including control applications, signal processing systems, and Internet of Things devices. It enables developers to efficiently design end-to-end sys-

tems with integrated support for both analog and digital signal processing. Additionally, the FPGA's clock and reset control guarantee accurate and dependable performance in crucial activities.

#### Digital Signal Oscilloscope

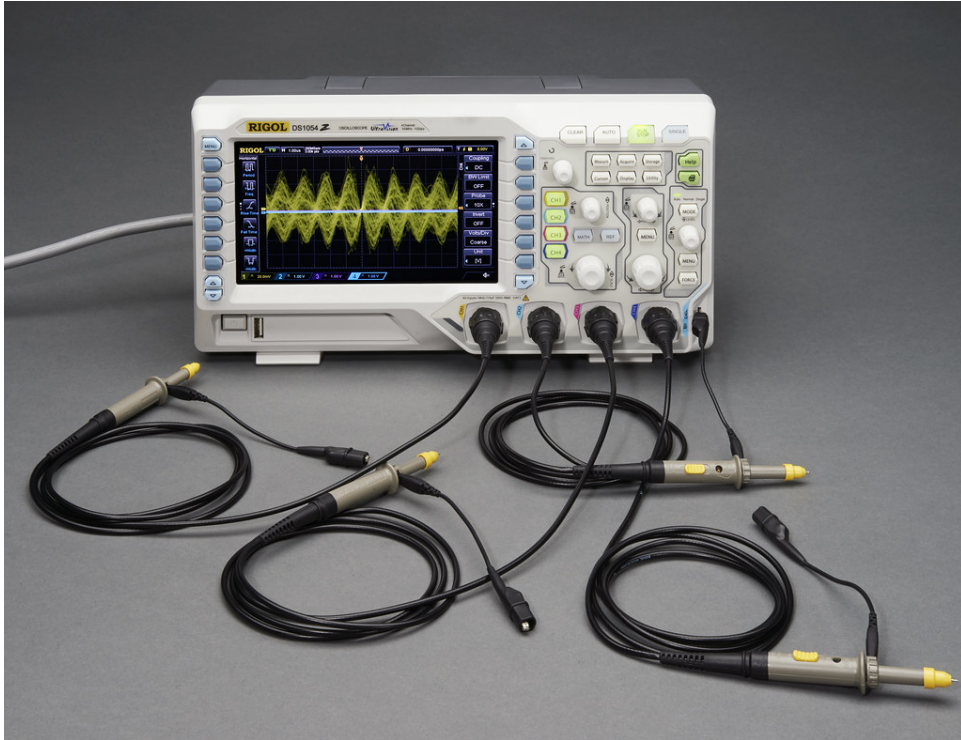


Figure 3.8: Digital Signal Oscilloscope

A Digital Storage Oscilloscope (DSO) is an essential tool for analyzing and visualizing electrical signals in digital and analog systems. Unlike traditional oscilloscopes, a DSO digitizes input signals, enabling precise waveform capture, storage, and analysis. With its ability to measure voltage, frequency, and timing characteristics of signals, the DSO is widely used in circuit debugging, signal testing, and system design. Its versatility allows users to observe real-time signal variations, making it indispensable for validating and troubleshooting complex electronics.

In the context of the traffic light controller and smart parking system project, the DSO was crucial in several phases of the implementation. First, it was used to modify the clock frequency, where the FPGA's 50 MHz clock was scaled down to generate a stable 50 Hz signal. This frequency modulation was essential for timing applications like controlling LED transitions or generating pulses for other circuitry. Additionally, the DSO was employed to test each GPIO pin for precise PWM wave generation, ensuring proper

functionality and accurate duty cycles required for smooth operation. Finally, the DSO was used to validate the implemented circuitry, such as LED drivers and current-limiting resistors, confirming signal integrity and performance. The ability to visually inspect and analyze waveforms provided by the DSO ensured the reliability and correctness of the design, making it an indispensable tool in the development process.

### Other Components

The implementation of a traffic light controller or smart parking system using the Artix-7 FPGA on the Aryabhata Card is enhanced by integrating essential components such as resistors, LEDs, a breadboard, and a digital multimeter (DMM). Resistors play a critical role in controlling current flow, protecting LEDs from excessive current. Green, red, and yellow LEDs can be used to mimic traffic light signals: green for "go," red for "stop," and yellow for "caution," making them ideal for traffic management systems.

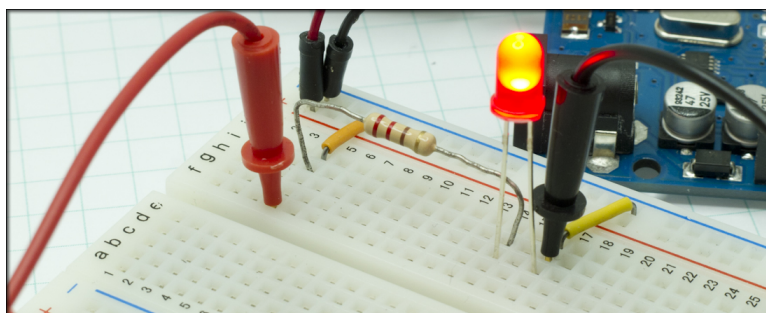


Figure 3.9: Other Components

In a smart parking system, these LEDs can also indicate parking space status—green for available, red for occupied, and yellow for reserved or transitioning. A breadboard facilitates the quick assembly and testing of these circuits without permanent connections, allowing easy modifications. A digital multimeter (DMM) is invaluable for measuring voltage, current, and resistance, ensuring the circuit operates correctly. These basic components, combined with the FPGA's programmable logic and high-speed processing, create a robust and efficient solution for real-time urban infrastructure systems.

## 3.3.2 Software

### Vivado Design Suite 2024.1

The Vivado Design Suite 2024.1, developed by Xilinx, serves as a comprehensive platform for designing, simulating, and implementing digital logic systems on FPGAs. It was a cornerstone for the implementation of the traffic light controller and smart parking system on the Artix-7 FPGA. Vivado's simulation environment was used extensively during the early stages to validate the RTL (Register Transfer Level) design.

This simulation ensured that the system behaved as intended under various input conditions. The suite also provided extensive debugging tools, allowing designers to trace issues in the design logic and optimize performance even before moving to hardware. The synthesis phase in Vivado converted the Verilog HDL code into a gate-level netlist optimized for the Artix-7 FPGA. This ensured efficient utilization of the FPGA's resources while meeting critical timing and area constraints. Vivado's synthesis reports, including resource utilization and timing estimation, provided insights into the feasibility of the design before proceeding to the implementation stage.



Figure 3.10: Vivado Design Suite 2024.1

The implementation tools in Vivado further refined the system by performing placement and routing of the synthesized design onto the FPGA's physical resources. The tool's static timing analysis ensured the design met timing requirements, which is particularly crucial for real-time systems like traffic light controllers and smart parking systems. Layout analysis provided a visual representation of the placement and routing process, allowing for further optimization to minimize delays and improve efficiency. Vivado's bitstream generation module produced a configuration file that was subsequently used to program the Artix-7 FPGA, enabling the hardware to function according to the verified design.

#### **Icarus Verilog**

Icarus Verilog is an open-source Verilog simulator that was employed during the initial stages of the project to simulate and validate the designs for the traffic light controller and smart parking system. This tool was critical for functional verification, allowing the behavior of the Verilog-based HDL designs to be tested under various input conditions and scenarios. By running simulations with Icarus Verilog, the correctness of the logic and the system's overall behavior could be thoroughly examined. Icarus Verilog excels in its ability to handle complex behavioral models and ensure compliance with the Verilog standard, making it an ideal choice for early-stage verification. It also allowed iterative refinement of the designs, as errors in the logic were identified and corrected through multiple simulation cycles. This saved significant time and effort

by addressing potential issues before proceeding to synthesis and hardware implementation.



Figure 3.11: Icarus Verilog

Icarus Verilog's versatility is derived from its support for a large number of simulation capabilities, such as test benches and hierarchical designs. In order to verify the operation of certain modules, including the parking sensors and the traffic light logic, under various circumstances, test benches were heavily utilized in this project. Icarus Verilog's simulation logs included comprehensive information on time, component interactions, and signal transitions. These logs were essential for comprehending the system's dynamic behavior and making sure it followed the design guidelines. A smooth verification process was made possible by the tool's extensive simulation capabilities and ease of use, which served as the cornerstone for its successful FPGA implementation.

### **GTKWave**

GTKWave, a powerful open-source waveform viewer, was employed in conjunction with Icarus Verilog to analyze and debug the simulation results. After simulating the designs in Icarus Verilog, GTKWave was used to visualize the output waveforms, offering an intuitive and detailed representation of the system's behavior. By displaying signal transitions, delays, and relationships between different signals, GTKWave allowed designers to assess the temporal behavior of the traffic light controller and smart parking systems with precision. This level of analysis was critical for debugging and verifying that the outputs met the expected functionality under various input scenarios. Signal mismatches or incorrect timings identified in GTKWave were addressed through iterative modifications to the design, ensuring a robust system before moving to the synthesis stage.

Another key strength of GTKWave lies in its ability to handle large-scale designs with complex signal interactions, making it suitable for analyzing multi-module systems like those in this project. The viewer supported hierarchical browsing, allowing detailed exploration of specific signals or modules, which was particularly useful for debugging intricate logic errors. For instance, the interactions between parking sensors, password

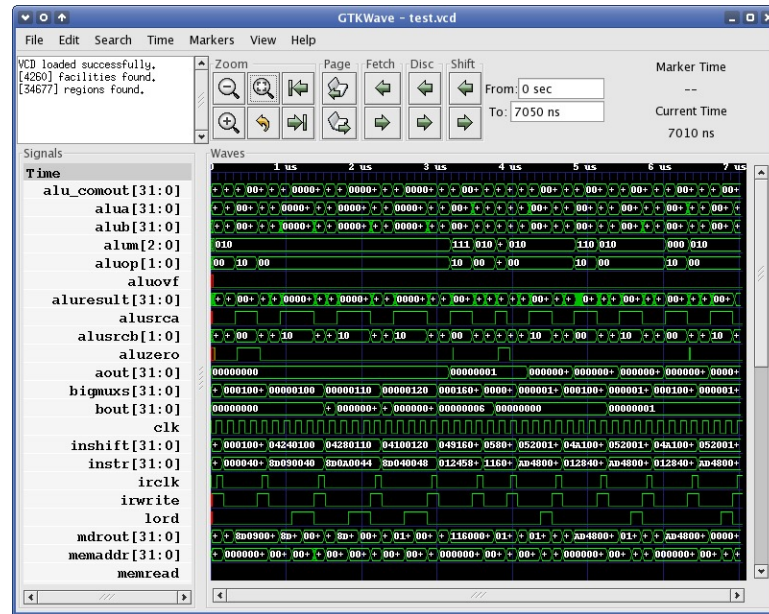


Figure 3.12: GTKWave

verification modules, and traffic light transitions could be visualized and debugged efficiently. By using GTKWave, potential issues were resolved early in the development process, leading to a smoother implementation on the FPGA. Together with Icarus Verilog, GTKWave formed a robust simulation and debugging framework that ensured the reliability and correctness of the system before hardware deployment.





# Chapter 4

## Project Workflow in Vivado Design Suite

### 2024.1

This chapter outlines the process of creating and managing the Vivado project for FPGA designs. Vivado projects consist of directory structures containing both user-created source files and system-generated files. These files support the design, simulation, and implementation of FPGA projects. Below are detailed steps to create a Vivado project, along with explanations for each stage.

#### 1. Starting Vivado Design Suite 2024.1

The method to open the software using the two below-mentioned operating systems is:

- Windows: Click Vivado's desktop shortcut to launch it.
- Linux: Use the relevant command to launch Vivado from a terminal.

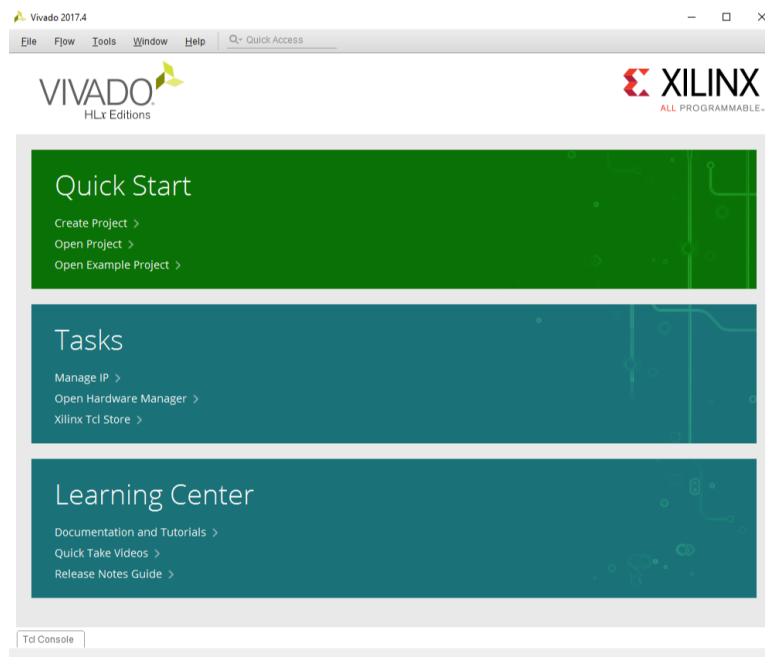


Figure 4.1: Vivado Start-Up Window

The Vivado start-up window displays after it has been launched (see Figure 4.1).

#### 2. Establishing a Project In the Quick Start panel, select Create Project.

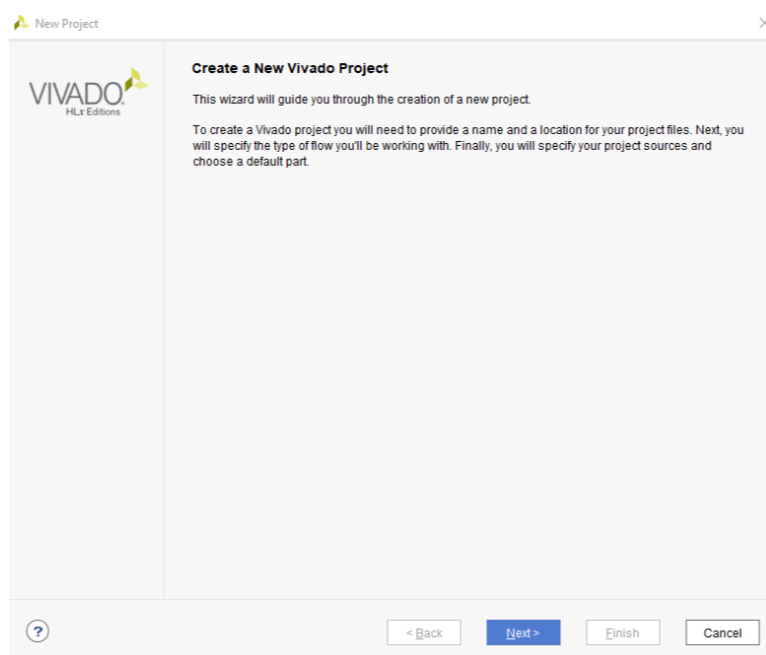


Figure 4.2: Create Project Dialog

The dialog box for the New Project Wizard appears (Figure 4.2). To continue, click Next.

### 3. Determining the Project's Name and Address

Name of the Project: Give the project a meaningful name.

Project Location: Select the project files' directory. To make management easier, group related projects together. To avoid compatibility problems, keep paths and names free of spaces.

### 4. Choosing the Type of Project

Select the type of project:

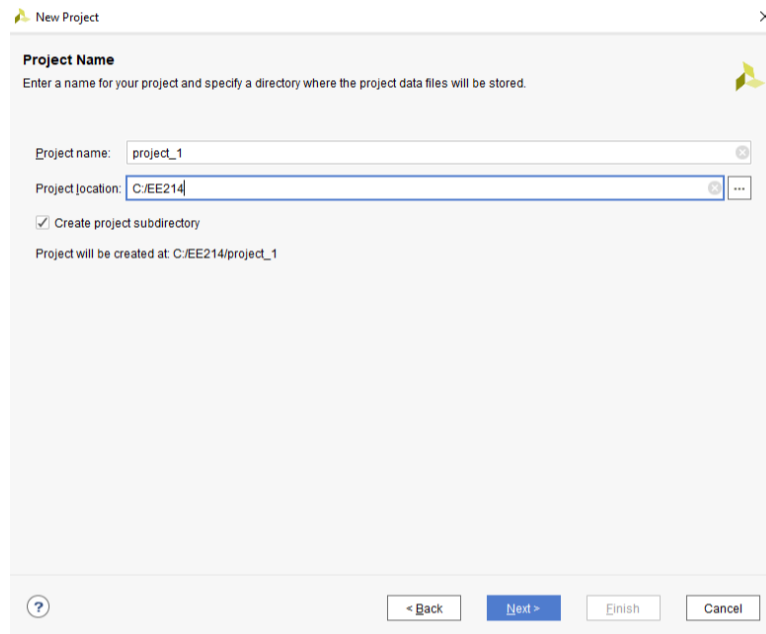
RTL Project: Verilog or VHDL are the hardware description languages used for the majority of FPGA designs.

Post-synthesis or simulation-only projects are examples of other kinds.

Click Next after choosing RTL Project (see Figure 4.4) for new designs.

### 5. Including Sources for Design



You can upload any existing HDL source files you may have (such as Verilog or VHDL) here:



**New Project**

**Project Name**  
Enter a name for your project and specify a directory where the project data files will be stored.

Project name:

Project location:   

☒ Create project subdirectory

Project will be created at: C:/EE214/project\_1






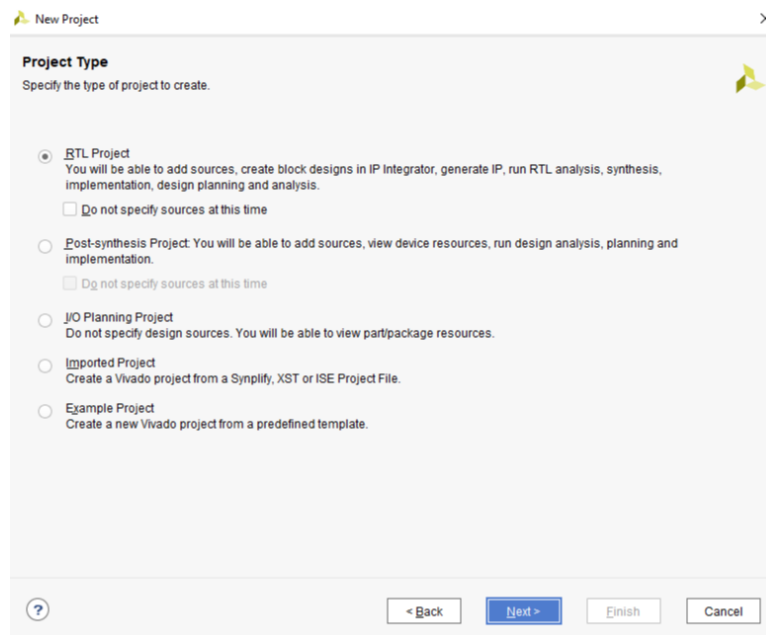
    

Figure 4.3: Enter Project Name



**New Project**

**Project Type**  
Specify the type of project to create.

☒ **RTL Project**  
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.  
☐ Do not specify sources at this time

☐ **Post-synthesis Project** You will be able to add sources, view device resources, run design analysis, planning and implementation.  
☐ Do not specify sources at this time

☐ **I/O Planning Project**  
Do not specify design sources. You will be able to view part/package resources.

☐ **Imported Project**  
Create a Vivado project from a Synplify, XST or ISE Project File.

☐ **Example Project**  
Create a new Vivado project from a predefined template.






    

Figure 4.4: Select Project Type

To access the file location, click Add Files.  
Click Next to bypass this step for new designs.

## 6. Including Restrictions

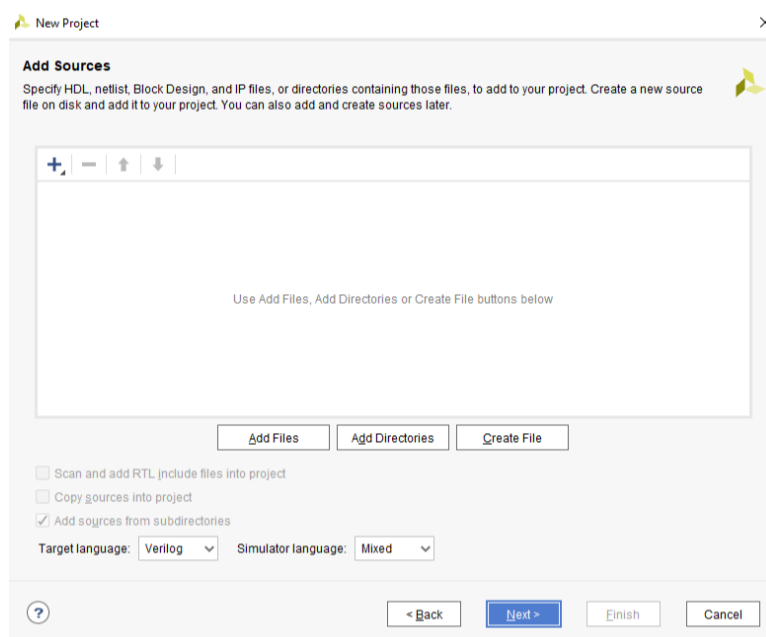


Figure 4.5: Add Sources

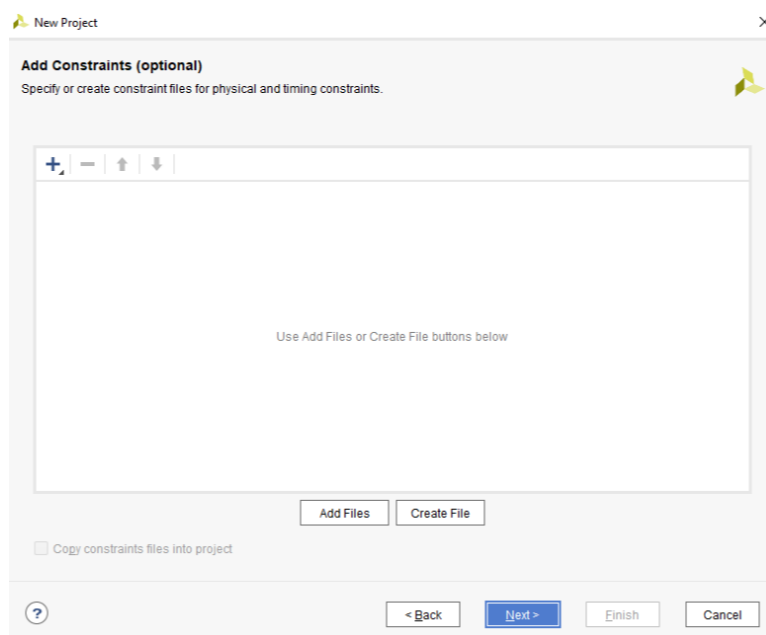


Figure 4.6: Add Constraint Files

Physical implementation specifics, such as clock settings or pin assignments, are specified by constraints:

Files called Xilinx Design Constraints (XDC) are frequently utilized.

If there are already constraint files available, click Add Files to add them.

---

Click Next if there aren't any constraints available yet (see Figure 4.6).

## 7. Selecting the Target Device

Specify the FPGA part or evaluation board used for the design:

Use details like device family, package, and part number.

To find this information, refer to the markings on the IC, the board's schematic, or online documentation. After selecting, click Next.

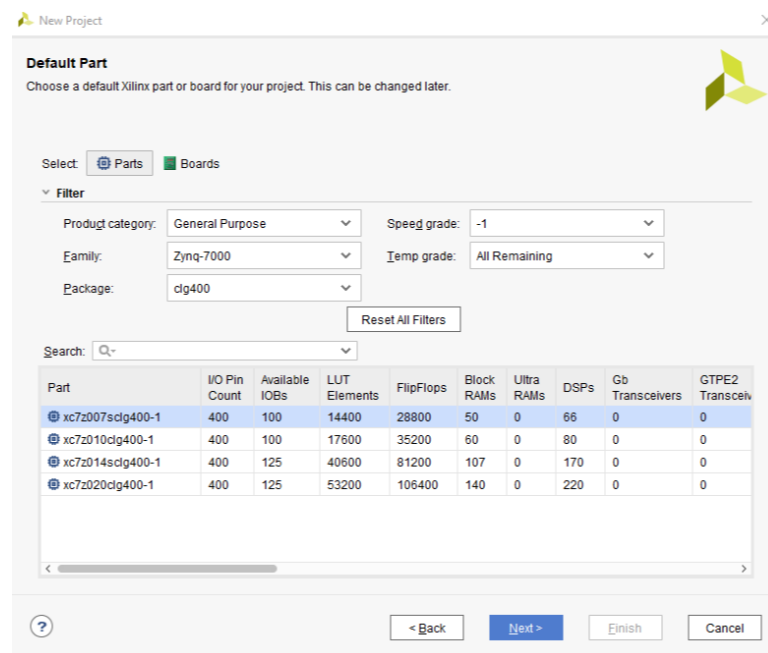


Figure 4.7: Select FPGA Board

## 8. Reviewing the Project Configuration

The Project Summary page displays a configuration overview:

Verify the project name, type, and selected part.

If corrections are needed, click Back to make changes.

Click Finish to create the project.

## 9. Navigating the Vivado Project Window

The Vivado project window opens after project creation:

Flow Navigator: Launches tasks like synthesis and implementation.

Project Manager: Manages files and settings.

Console Pane: Displays messages, logs, and TCL commands.



---

## 4.0.1 Key Notes on Constraints

XDC files are vital for synthesizing designs on hardware.

These files configure:

I/O Constraints: Define physical pin mappings.

Clock Constraints: Set up timing parameters.

Creating constraints will be detailed in a later section.

By following these steps, you can set up and manage Vivado projects effectively, ensuring a streamlined FPGA design process.

### Constraints for Traffic Light Controller

The Xilinx Design Constraints (XDC) file is critical for mapping the design to the FPGA. This file specifies the physical connections between the FPGA pins and external components, such as LEDs and switches, used in the traffic light controller.

Below is the XDC file and the schematic diagram of the project.

```
set_property IOSTANDARD LVCMOS33 [get_ports clk]
create_clock -period 20.000 [get_ports clk]
set_property PACKAGE_PIN W19 [get_ports clk]

set_property IOSTANDARD LVCMOS33 [get_ports {light_M1[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {light_M1[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {light_M1[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {light_M2[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {light_M2[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {light_M2[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {light_MT[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {light_MT[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {light_MT[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {light_S[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {light_S[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {light_S[0]}]
set_property PACKAGE_PIN V17 [get_ports {light_M1[2]}]
set_property PACKAGE_PIN W20 [get_ports {light_M1[1]}]
set_property PACKAGE_PIN V19 [get_ports {light_M1[0]}]
set_property PACKAGE_PIN V18 [get_ports {light_M2[2]}]
set_property PACKAGE_PIN U20 [get_ports {light_M2[1]}]
set_property PACKAGE_PIN T18 [get_ports {light_M2[0]}]
set_property PACKAGE_PIN V20 [get_ports {light_MT[2]}]
set_property PACKAGE_PIN E18 [get_ports {light_MT[1]}]
set_property PACKAGE_PIN C17 [get_ports {light_MT[0]}]
set_property PACKAGE_PIN D17 [get_ports {light_S[2]}]
set_property PACKAGE_PIN A16 [get_ports {light_S[1]}]
set_property PACKAGE_PIN E17 [get_ports {light_S[0]}]
```

Figure 4.10: XDC File for Traffic Light Controller

**1. XDC File** The following XDC file defines the pin assignments for the Traffic Light

Controller design. It maps logical design signals to specific FPGA pins and configures timing constraints.

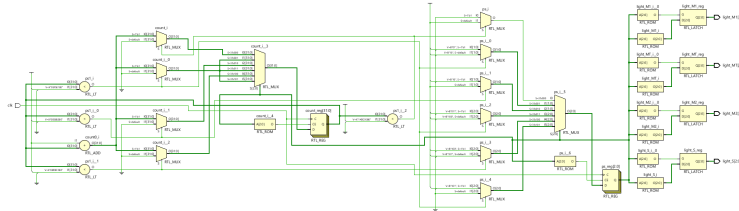


Figure 4.11: Schematic for Traffic Light Controller

**2. Schematic** The schematic diagram provides a visual representation of the connections between the FPGA and external components in the Traffic Light Controller system. This includes:

LEDs for red, yellow, and green lights.

External connections for the clock input.

GitHub repository at the following link:

FPGA-Based Traffic Light Controller and Smart Parking System -  
<https://github.com/adititapariya/TLVS-TrafficLight-Verilog-System-with-Smart-Parking>



# Chapter 5

## Observations and Results

Compared to conventional microcontroller-based systems, FPGA-based traffic light controllers provide a number of clear advantages. Their capacity to effectively manage traffic flow by dynamically modifying signal timings in response to current traffic circumstances is one of their main advantages. In addition to ensuring seamless transitions between the many signal phases—such as Green, Yellow, and Red—this dynamic control lessens congestion. Furthermore, many traffic lights may be managed simultaneously without delays thanks to FPGA’s real-time processing capabilities, which makes the system incredibly quick and efficient.

Energy efficiency is another noteworthy benefit. Generally speaking, FPGA designs use less power than their microcontroller equivalents, which makes these systems more affordable over time. Furthermore, their intrinsic adaptability makes it simple to adjust to a variety of intersection layouts, ranging from straightforward crossroads to intricate multi-lane networks. This system’s scalability guarantees that it can meet future demands, providing a reliable and long-lasting traffic control solution.

The improved security and operating efficiency offered by the FPGA-based smart parking system are exceptional. Password authentication provides an extra degree of security by guaranteeing that only vehicles with permission can enter. Additionally, integrated sensors are essential for directing cars straight to open places, cutting down on time spent looking for spots and increasing overall parking usage.

Real-time performance is guaranteed by the system’s high-speed processing capability. For example, the system instantly updates gate access and space availability when a car enters or quits the parking area. Because of its cost-effectiveness and reactivity, FPGA is a great option for smart parking systems. It can also manage real-time operations. It is also appropriate for a range of applications due to its scalability.

### 5.1 Traffic Light Controller

The proposed system was tested using 12 LEDs to demonstrate how the system would work in a 3-way junction in the real world with 2 lanes each. The figure 4.1 shows all the 6 states mentioned in the figure 3.10 using 4 sets of 3 LEDs depicting state of the traffic light installed on the path M1, M2, MT, and S respectively. Starting from the left the 3 LEDs of each path are in the order red (stop), yellow (slow down/stop),

and green (Go).

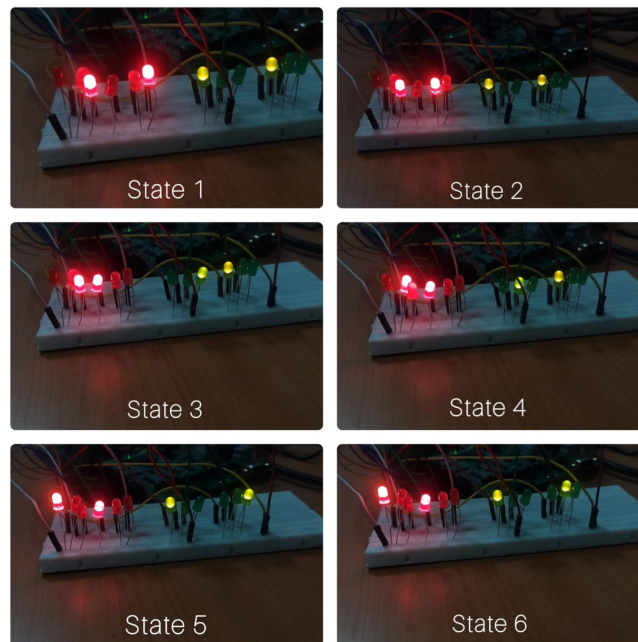


Figure 5.1: Hardware implemented Traffic Light Controller with different states

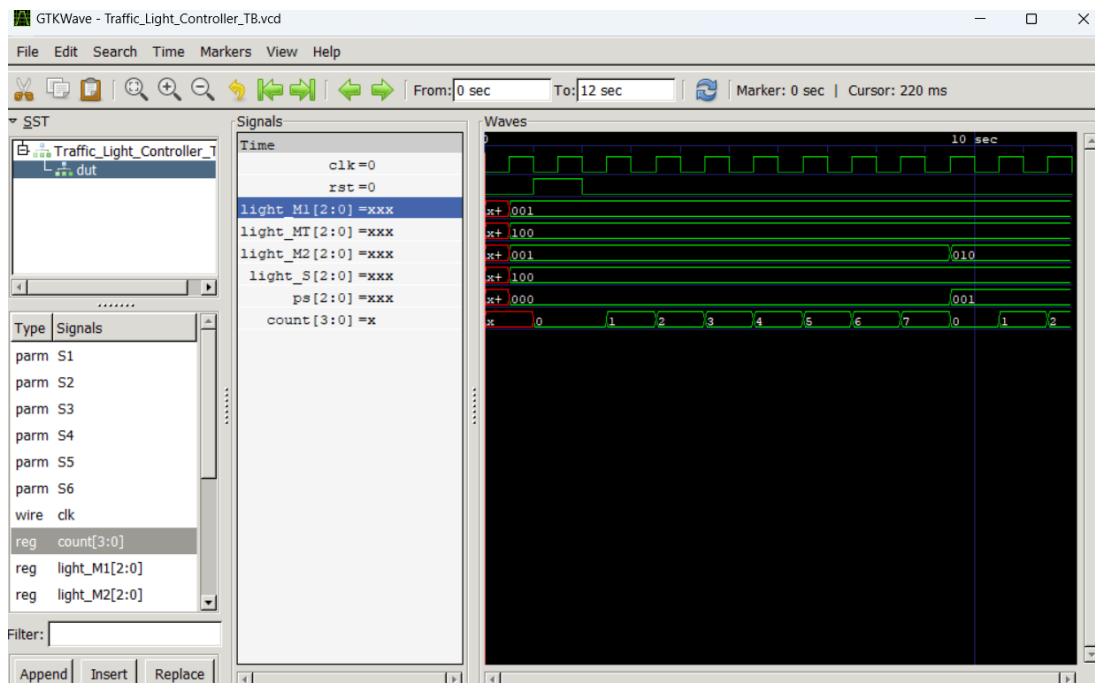


Figure 5.2: GTKWave Waveforms for Traffic Light Controller

Additionally, the figure 4.2 shows the GTKWave panel which shows the waveform of

each path (M1, M2, MT, S) as per the timing sequence. The waveform of each path shows the exact state of that particular path. For example, in the figure 4.2, the path M1 waveform is currently in state 001, indicating that the state of LED for the path M1 is currently red. Similarly, the state of LEDs of path MT, M2, and S are green, red, and green respectively.

## 5.2 Smart Parking System

The GTKWave waveforms in the figure 4.3, demonstrate the logic state of front and rear sensors of a parking area. As per the logic state of the front sensor it will follow the procedure mentioned in the figure 3.12 and conclude if the vehicle can be parked in that particular parking area.

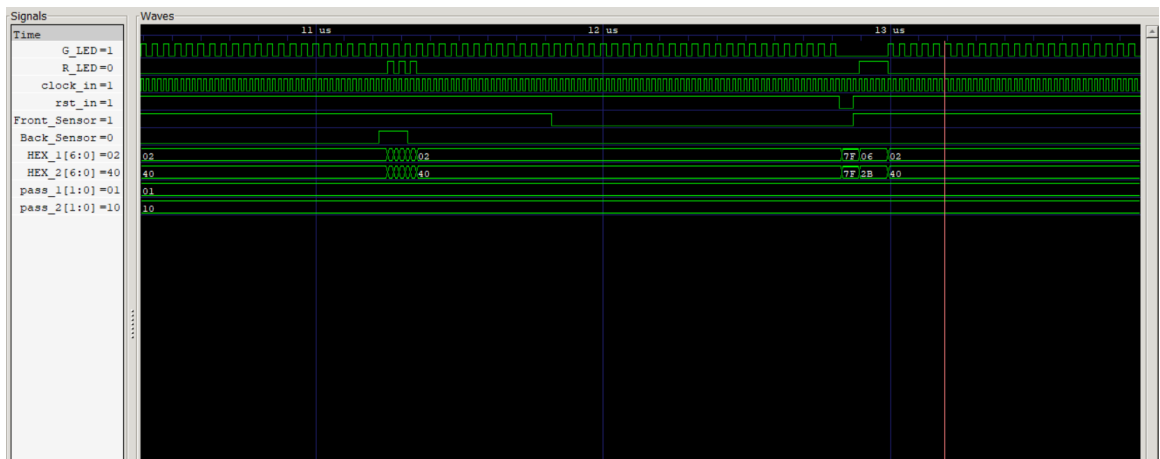


Figure 5.3: GTKWave Waveforms for Smart Parking



# Chapter 6

## Conclusion

This report has explored the design and implementation of two innovative systems: a traffic light controller and a smart parking system, both utilizing FPGA technology for optimized performance. The traffic light controller leverages FPGA's parallel processing capabilities to efficiently manage traffic flow at a T-shaped intersection with varying levels of congestion. By incorporating multiple states and predefined time delays, the system effectively minimizes traffic delays and improves road safety. The use of FPGA in this context ensures real-time processing, greater flexibility, and lower power consumption compared to traditional microcontroller-based systems.

The smart parking system, on the other hand, utilizes sensors and password-based authentication to enhance both security and convenience. The system is designed to automatically detect vehicles, authenticate their entry, and guide them to available parking spots. By using FPGA for real-time processing, the system is able to respond instantaneously to changes, ensuring seamless operation and reducing the time spent in the parking lot. Furthermore, the system's scalability and energy efficiency make it a cost-effective solution for modern parking management.

Both applications demonstrate the significant potential of FPGA in developing efficient, responsive, and scalable solutions for real-world challenges. The high-speed processing, parallelism, and flexibility inherent in FPGA technology make it an ideal choice for applications that demand real-time performance and adaptability. The successful implementation of these systems highlights the advantages of FPGA over traditional hardware in terms of performance, power efficiency, and system scalability.

As urbanization continues to accelerate, the need for smarter infrastructure becomes increasingly vital. Moving towards smart cities, these technologies, such as intelligent traffic management and automated parking systems, are essential components of modern urban planning. By incorporating such systems into everyday city infrastructure, cities can significantly improve traffic management, reduce congestion, enhance road safety, and optimize the use of resources. These projects underscore the growing role of FPGA technology in enabling smart city developments, where efficiency, sustainability, and real-time data processing are critical for improving quality of life.

In conclusion, the traffic light controller and smart parking system not only exemplify the power of FPGA in solving complex real-time problems but also pave the way for future advancements in intelligent transportation systems and automated parking management. These projects contribute to the realization of smart cities, enhancing operational efficiency, sustainability, and resource optimization, which are integral to addressing the challenges of rapidly growing urban environments.

## **6.1 Future Scope**

Future expansions of the traffic light controller could include integration with a wider range of sensors, such as vehicle detection and environmental sensors, to further optimize signal timings based on real-time traffic patterns, weather conditions, and emergency vehicle prioritization. Additionally, the system could be connected to a central cloud infrastructure for cross-junction coordination, enabling dynamic traffic flow adjustments across an entire city, contributing to improved overall traffic management and reduced congestion.

The smart parking system can be expanded to accommodate a larger scale of smart city applications by integrating it with mobile apps for real-time parking availability updates and reservation systems. Furthermore, using machine learning algorithms, the system could predict parking space availability based on historical usage data, optimizing parking management.

# Bibliography

- [1] P. Shree.T, D. C. S. Manikandababu, and D. M. Jagadeeswari, “Advanced traffic light controller system using fpga implementation,” *International Journal for Research in Applied Science Engineering Technology (IJRASET)*, 2022.
- [2] D. Bhavana, D. R. Tej, P. Jain, G. Mounika, R. Mohini, and Bhavana, “Traffic light controller using fpga,” *International Journal of Engineering Research and Applications*, 2015.
- [3] A. Banerjee, “Fpga implementation of an intelligent traffic light controller (i-tlc) in verilog,” *arXiv:2401.13345*, 2024.
- [4] Kishore, S. Venkata, Sreeja, Vasavi, Gupta, Vibhuti, Videesha, V., Raju, I. B. K., Rao, and K. Madhava, “Fpga-based traffic light controller,” pp. 469–475, 2017.
- [5] T. B. O. Reddy and V. Sowmya, “An advanced traffic light controller using verilog hdl,” *International Journal of VLSI System Design and Communication Systems*, 2017.
- [6] S. Shankaran and L. Rajendran, “Real-time adaptive traffic control system for smart cities,” *2021 International Conference on Computer Communication and Informatics (ICCCI)*, 2021.
- [7] Amaresh, K. S. Bhat, A. G, Bhagyashree, and Aishwarya, “Density-based smart traffic control system for congregating traffic information,” *International Conference on Intelligent Computing and Control Systems (ICICCS 2019)*, 2019.
- [8] S. Kaur, Varshitha, Siddharth, Tejus, and P. Kumar, “Adaptive traffic light controller using fpga,” *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT-2018)*, 2018.
- [9] N. Rida, M. Ouadoud, A. Hasbi, and S. Chebli, “Adaptive traffic light control system using wireless sensors networks,” *ICEMI’09 9th International Conference*, 2018.
- [10] X. Wen-Juan and L. Jian-Feng, “Application of vision sensing technology in urban intelligent traffic control system,” *4th International Conference on Computer and Technology Applications*, 2018.
- [11] S. V. Kishore, V. Sreeja, V. Gupta, V. Videesha, I. B. K. Raju, and K. M. Rao, “Fpga based traffic light controller,” *International Conference on Trends in Electronics and Informatics (ICEI 2017)*, 2017.

- [12] S. M. Shinde, "Adaptive traffic light control system," *IEEE Transactions on Intelligent Transportation Systems*, 2017.
- [13] B. Ghazal, K. E. Khatib, K. Chahine, and M. Kherfan, "Smart traffic light control system," *International Journal of Scientific and Research Publications*, 2016.
- [14] S. M. Deshmukh and B. N. Savant, "Traffic congestion alerting system," *International Journal of Software and Web Sciences (IJSWS)*, 2016.
- [15] M. S. V. Lahade and M. S. R. Hirekhan, "Intelligent and adaptive traffic light controller (ia-tlc) using fpga," *International Conference on Industrial Instrumentation and Control (ICIC)*, 2015.
- [16] B. chan Jeon, W. ki Park, H. young Lee, Y.-H. Lee, and S.-C. Lee, "Design implementation of a solar powered led road traffic sign control system," *ISOCC 2015*, 2015.
- [17] L. Bhaskar, A. Sahai, D. Sinha, G. Varshney, and T. Jain, "Intelligent traffic light controller using inductive loops for vehicle detection," *1st International Conference on Next Generation Computing Technologies (NGCT-2015)*, 2015.
- [18] J. Shridhar, Ruchin, and P. Whig, "Design and simulation of power efficient traffic light controller (ptlc)," *International Conference on Computing for Sustainable Global Development (INDIACom)*, 2014.
- [19] P. K. Singh and P. Daniel, "Advanced real traffic light controller system design using cortex-m0 ip on fpga," *IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*, 2014.
- [20] M. Ramzanzad and H. R. Kanan, "A new method for design and implementation of intelligent traffic control system based on fuzzy logic using fpga," *13th Iranian Conference on Fuzzy Systems (IFSC)*, 2013.
- [21] M. F. M. Sabri, M. H. Husin, W. A. W. Z. Abidin, K. M. Tay, and H. M. Basri, "Design of fpga-based traffic light controller system," *2nd International Conference on Emerging Trends in Engineering and Technology*, 2011.
- [22] S. Singh and S. C. Badwaik, "Design and implementation of fpga-based adaptive dynamic traffic light controller," *5th International Conference on Intelligent Transportation Systems*, 2011.
- [23] J. E. Ortiz and R. H. Klenke, "Simple traffic light controller: A digital systems design project," *IEEE International Conference*, 2010.



- [24] W. Wen, "A dynamic and automatic traffic light control expert system for solving the road congestion problem," *Expert Systems with Applications*, vol. 34, no. 4, p. 2370–2381, 2008.
- [25] S. B. Kale and G. P. Dhok, "Design of intelligent ambulance and traffic control," *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 2, no. 5, p. 2278–3075, 2013.
- [26] P. K. Singh and P. Daniel, "Advanced real traffic light controller system design using cortex-m0 ip on fpga," p. 1023–1026, 2014.
- [27] M. Sabri, M. Husin, W. Abidin, K. Tay, and H. Basri, "Design of fpga-based traffic light controller system," p. 114–118, 2011.
- [28] M. A. Qureshi, A. Aziz, and S. H. Raza, "A verilog model of adaptable traffic control system using mealy state machines," *International Journal of Computer and Electrical Engineering*, vol. 4, no. 3, p. 400, 2012.
- [29] S. Singh and S. C. Badwaik, "Design and implementation of fpga-based adaptive dynamic traffic light controller," p. 324–330, 2011.
- [30] B. Dilip, Y. Alekhya, and P. D. Bharathi, "Fpga implementation of an advanced traffic light controller using verilog hdl," *International Journal of Advanced Research in Computer Engineering Technology (IJARCET)*, vol. 1, no. 7, p. 2278–1323, 2012.
- [31] W. El-Medany and M. Hussain, "Fpga-based advanced real traffic light controller system design," p. 100–105, 2007.
- [32] Y.-S. Huang, "Design of traffic light control systems using statecharts," *The Computer Journal*, vol. 49, no. 6, p. 634–649, 2006.
- [33] V. N. Ivanov, "Using a picoblaze processor to traffic light control," *Cybernetics and Information Technologies*, vol. 15, no. 5, p. 131–139, 2015.
- [34] S. Iokhande, P. S. morade, and P. M. joshi, "Smart car parking system using fpga and e-application," *International Research Journal of Engineering and Technology (IRJET)*, vol. 03, 2016.