Smart Traffic System using Verilog and Python Integration

[1]V.Kiran Lal [2]C. Harikrishna [3]P.Poojitha [4]M.Pavani [5]E.Nandini [6]V.Preethi

Departmnt of ECE

Narayana Engineering College,524004,Andhra Pradesh ,India.

Abstract: *The main objective of this project is to design an intelligent traffic control system. Traffic congestion is a critical challenge faced by urban areas worldwide, leading to significant delays and inefficiencies in transportation. This paper presents the design of a Real-Time Traffic Control System using Verilog. It aims to create an intelligent traffic management system that integrates Finite State Machine (FSM) principles with Pythonbased GUI customization for real-time adaptability. The system is implemented using Verilog for the core traffic signal logic and Python for interfacing and visualization. The Python GUI provides user-friendly control with features such as: - Inputting traffic conditions (e.g., vehicle density in East and West directions). - Indicating ambulance arrival for prioritization. - Managing pedestrian crossings. By employing a finite state machine (FSM) approach, the traffic light controller manages the sequence of red, yellow, and green lights, ensuring smooth transitions. The system also integrates emergency vehicle detection, enabling immediate signal changes to facilitate the passage of ambulances. The implementation will be verified using the Xilinx ISE tool, and the simulation results demonstrate the effectiveness of the design in reducing traffic delays and enhancing road safety. This innovative approach bridges hardware simulation with interactive software visualization, making it a robust solution for modern smart traffic systems.*

Keywords*: Smart Traffic System, Verilog, Tkinter, Traffic Management, GUI*

I.INTRODUCTION

Traffic congestion is a growing concern in urban areas, leading to long travel times, excessive fuel consumption, and increased pollution levels. Traditional traffic management systems operate on fixed-time signals, which do not adapt to real-time traffic conditions, often causing unnecessary delays or inefficient traffic flow.An advanced approach to traffic control can optimize signal timings based on logical conditions, ensuring smoother vehicle movement and reducing wait times at intersections. By utilizing a hardware description language for traffic signal control and integrating it with an interactive graphical interface, traffic management becomes more efficient and user-friendly. This approach not only enhances traffic flow but also improves overall road safety and reduces environmental impact.

II.FUNCTIONAL OVERVIEW

The system is designed to efficiently manage traffic flow by integrating hardware-based traffic control logic with a graphical user interface for real-time monitoring. The key functionalities include:

1. **Traffic Signal Control Logic (Verilog-Based):**
   - Implements a predefined logic to control traffic lights based on time cycles.
   - Simulates real-world traffic conditions by switching between Red, Yellow, and Green signals.

- Ensures smooth traffic flow and minimizes congestion at intersections.

2. **Graphical User Interface (Tkinter-Based):**
   - Provides a visual representation of traffic signals.
   - Allows users to monitor and interact with the system in real time.
   - Displays the status of each traffic signal and updates dynamically based on logic execution.

3. **Integration of Verilog with Tkinter:**
   - Verilog processes the signal control logic, while Tkinter displays the results interactively.
   - Enables real-time simulation of traffic management, offering insights into system behavior.

4. **Efficiency and Optimization:**
   - Reduces waiting time at signals by efficiently switching based on logical conditions.
   - Can be further enhanced with real-time sensor data and adaptive control mechanisms

III. DESIGN

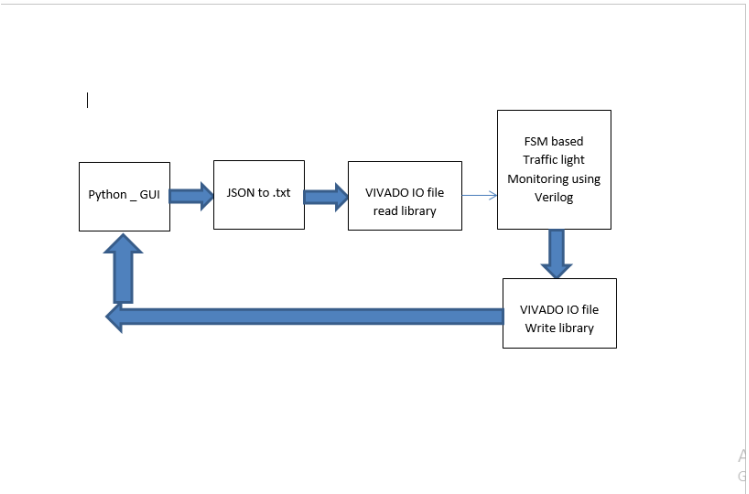


Fig.1 Block diagram

The system is structured into two main components:

1. Traffic Signal Control Logic (Verilog-Based)

- The core logic of the traffic control system is implemented using Verilog, a hardware description language.
- Traffic lights follow a predefined sequence of Red, Yellow, and Green signals based on time cycles.
- The logic ensures efficient switching between signals to minimize congestion at intersections.
- Simulations are performed to verify the correctness of the traffic control algorithm before integration.

2. Graphical User Interface (Tkinter-Based)

- A Python Tkinter GUI provides **a** real-time visual representation of traffic signals.
- The interface displays signal status dynamically, updating according to the Verilog logic execution.
- Users can monitor and interact with the traffic system through the GUI.
- The GUI consists of signal indicators, timers, and control buttons for better visualization.

3. Integration of Verilog with Tkinter

- The Verilog-based traffic logic is connected to the Tkinter GUI, allowing real-time simulation.
- Outputs from the Verilog simulation are fed into the Tkinter interface, ensuring synchronization between the control logic and visual display.
- This integration helps analyze traffic behavior and improve signal efficiency.

4. System Workflow

- Verilog initializes the traffic control logic and sets the default signal timings.
- Traffic lights change based on predefined conditions, switching between Red, Yellow, and Green.
- Verilog outputs are transferred to the Tkinter GUI, updating the traffic signals on the display.
- Users can monitor the signal changes and test different traffic scenarios using the GUI

IV.WORKING

The simulation was executed in **two phases**:

1. **Verilog Processing:**

- The Verilog module read input values from `1.txt`.
- Based on the **Finite State Machine (FSM)** logic, it processed signal control.
- The calculated green, yellow, and red light durations were stored in `op.txt`.
- outputs were verified through **Verilog console logs.**

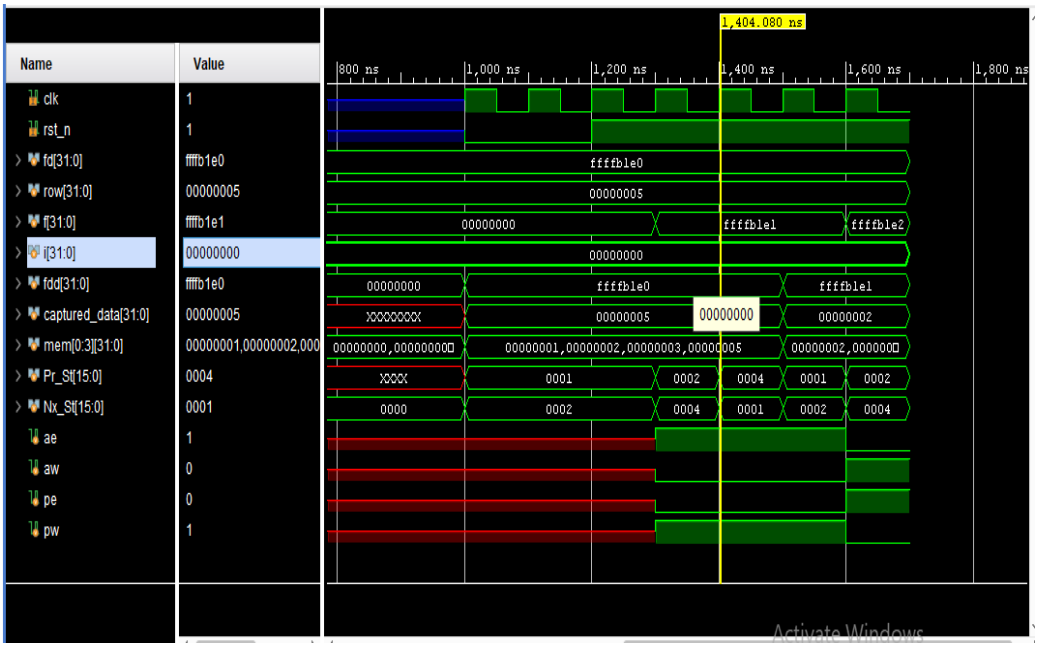


Fig .1 Simulation in Xilinx

2.GUI – Tkinter Visualization:

- The `op.txt` output from Verilog was used to simulate traffic lights in Tkinter-Turtle.
- The signals changed dynamically based on real-time traffic conditions.
- Observations confirmed that the **signal transitions were smooth and accurate**.

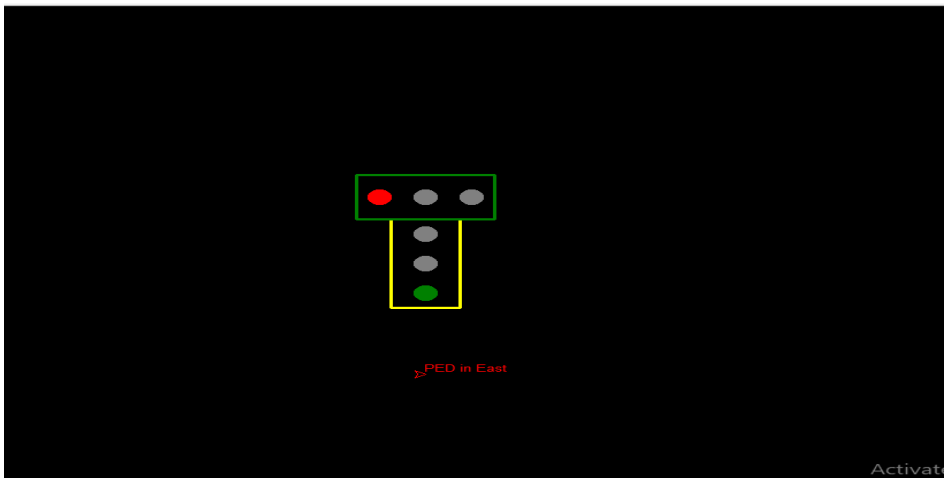


Fig.3 Transistion of Lights using GUI

Step 1: **Reading Input from GUI**

- The user inputs traffic data using the python GUI
- The GUI collects parameters like vehicle count ,emergency vehicle status and road conditions

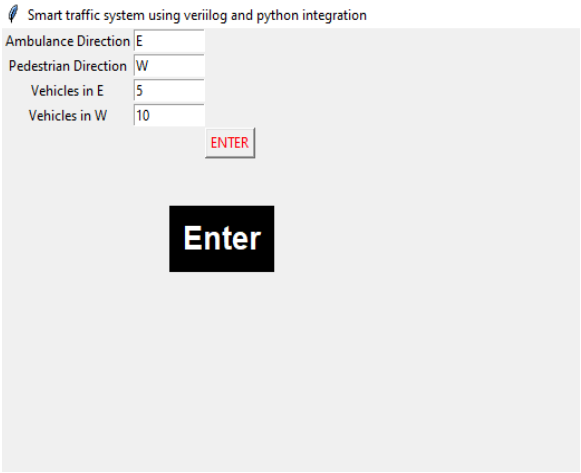


Fig .4 Input Python GUI

Step 2: **Storing Data in a Text File**

- The input from the GUI is written into a text file for further processing
- This file acts as an intermediate data storage for the Verilog module

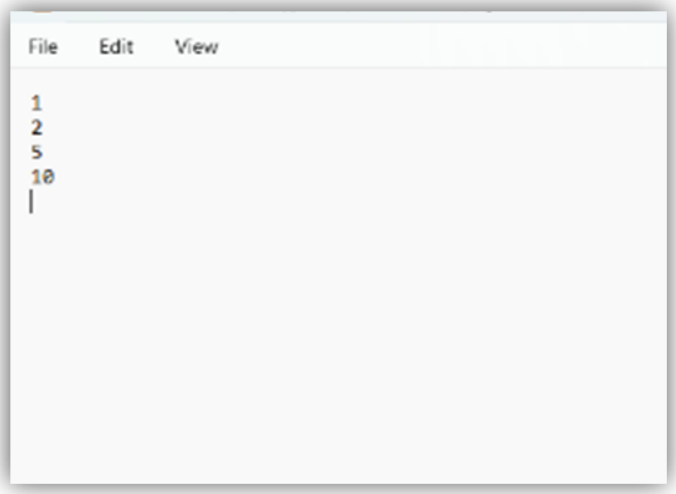


Fig.5 Input Text File

Step 3: **Processing Input in Verilog**

The Verilog code reads data from the text file and execute logic to determine optimal signal timing

It uses condition like vehicle density, pedestrian crossing and emergency vehicle priority

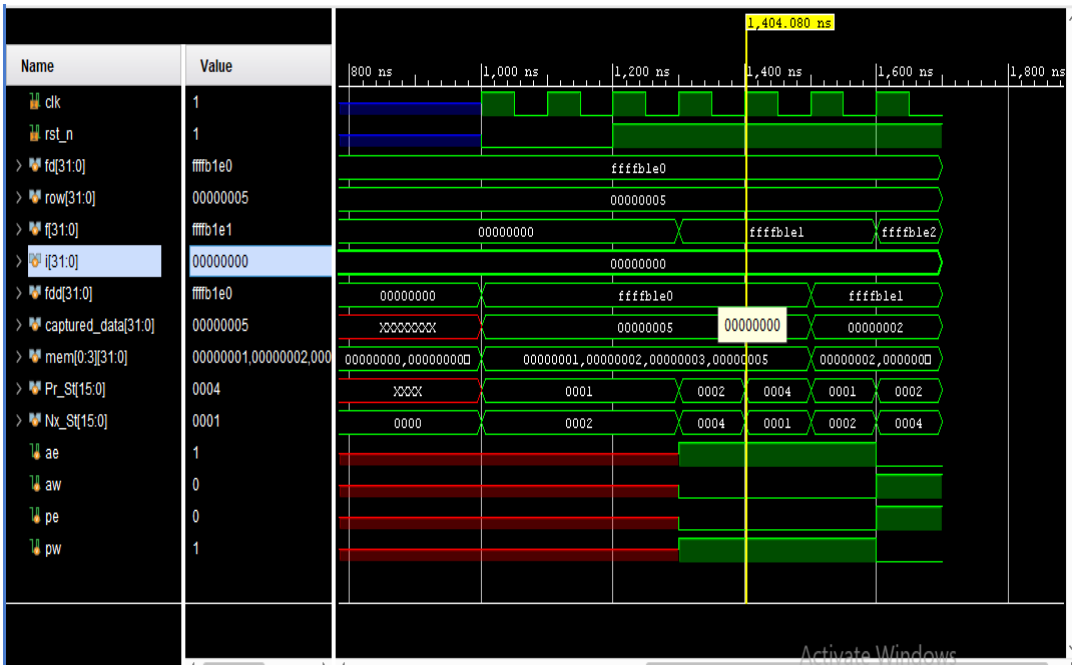


Fig.6 Processing Input in Verilog

Step 4: **Writing Output to a File**

The Verilog output (optimized signal timings) is written to another text file.

This output file is then used by the GUI to update traffic signals data

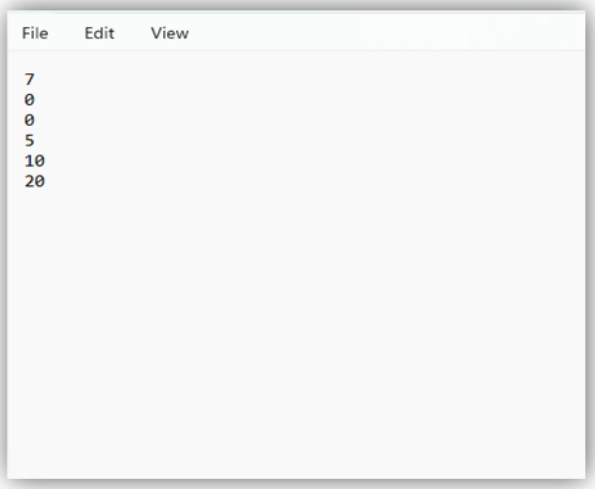


Fig.7 Writing Output to a File

Step 5: **Displaying Traffic signals in GUI**

The Tkinter GUI reads the output file and updates the traffic light display
The signals dynamically change based on Verilog calculations

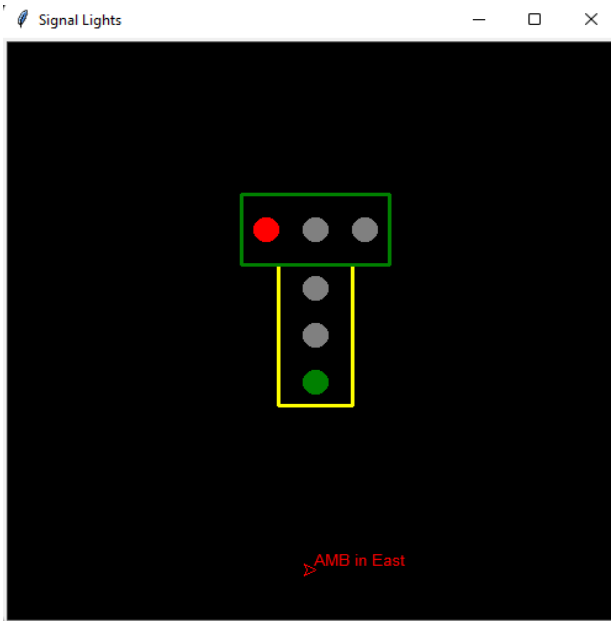


Fig .8 Displaying Traffic signals in GUI - 1

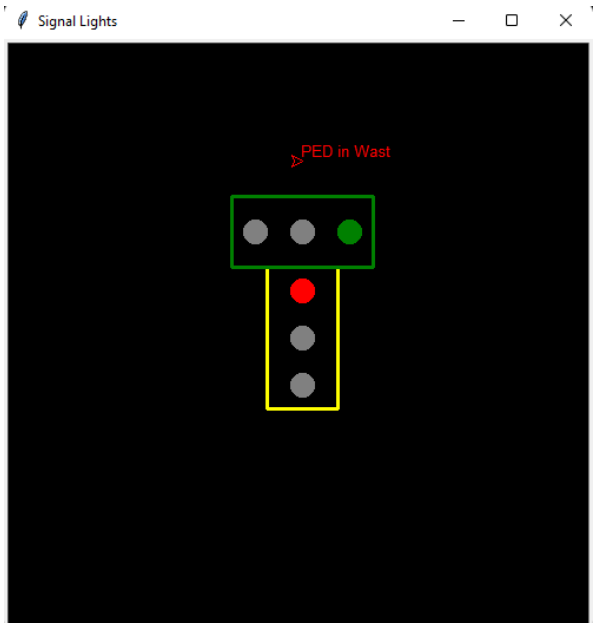


Fig.9 Displaying Traffic signals in GUI - 2

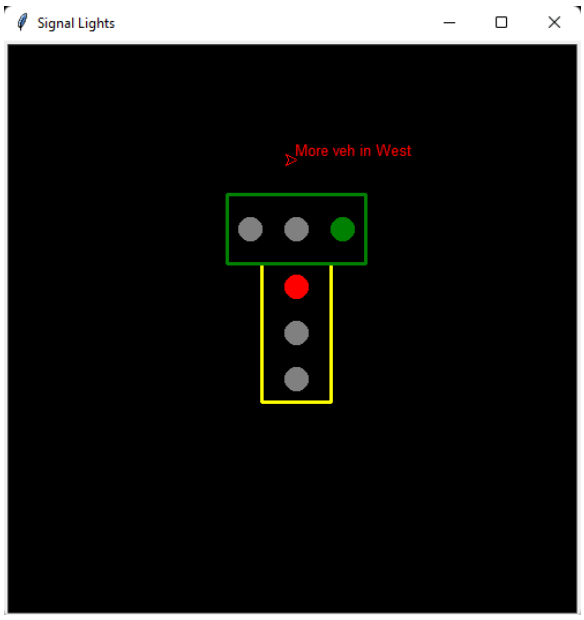
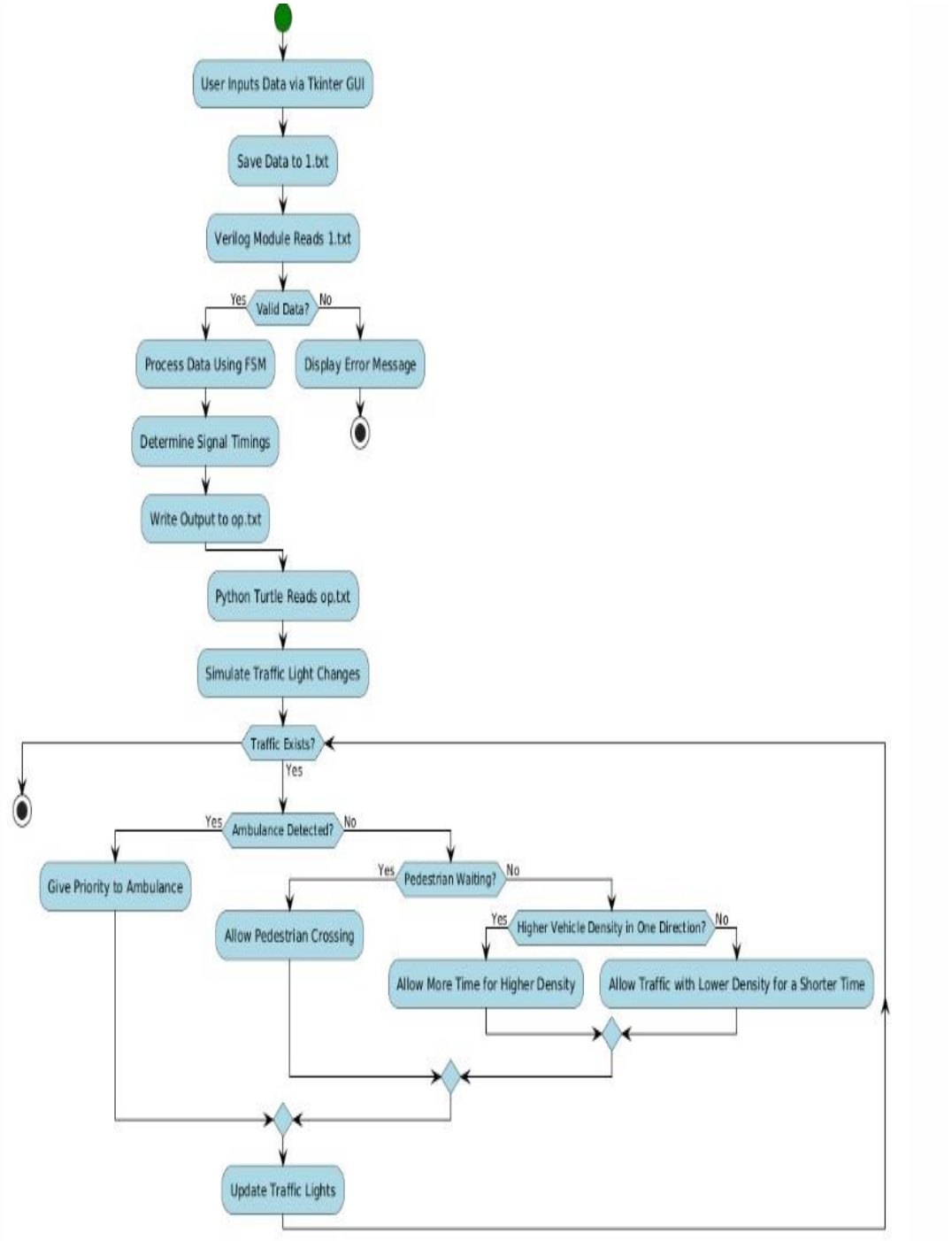


Fig .10 Displaying Traffic signals in GUI -3

Flow of Execution:



V. INSTALLATION

Step 1: Install Xilinx Vivado

1. Download and Install Vivado

1. Go to the **Xilinx Vivado official website**: https://www.xilinx.com/
2. Click on **Download Vivado** and select the latest version.
3. Choose the **Vivado HLx Edition (WebPACK or Full Version)**.
4. Create or log in to your Xilinx account.
5. Download the **Vivado installer** based on your OS (Windows/Linux).
6. Run the installer and follow the on-screen instructions:
    - **Select Installation Type** → Choose **Vivado HLx with Verilog support**.
    - **Set Installation Path** → Choose the desired directory.
    - **Accept License Agreement** and proceed.
7. Wait for the installation to complete.
8. Restart your system if required.

2. Verify Vivado Installation

- Open **Vivado** from the Start Menu or Terminal.
- Check if the tool launches correctly.

Step 2: Install Python and IDLE

1. Download and Install Python

1. Go to the **official Python website**: https://www.python.org/downloads/
2. Click **Download Python 3.x** (latest version).
3. Run the installer and check the box **"Add Python to PATH"**.
4. Click **Install Now** and wait for the installation to complete.

2. Verify Python Installation

- Open **Command Prompt (Windows) / Terminal (Linux/macOS)** and type:

```
bash
CopyEdit
python --version
```

- If installed correctly, it will display the Python version.

3. Install Required Libraries

- Open **Command Prompt/Terminal** and install Tkinter:

```
bash
CopyEdit
pip install tkinter numpy customtkinter
```

Step 3: Connect Xilinx Vivado with Python IDLE

1. **Generate Traffic Signal Logic in Verilog using Vivado**
   - Open **Vivado** and create a **new project**.
   - Add your **Verilog files** that define traffic signal logic.
   - Run **simulation** and verify the results.
2. **Export Verilog Simulation Output**
   - Use `write_verilog` command in Vivado to export design files.
   - Save the output in a `.vcd` or `.txt` file format.
3. **Read Verilog Output in Python**
   - Write a Python script to **read the Verilog output file** and process signal data.
   - Example Python script to read a `.txt` output file from Verilog:

```
python
CopyEdit
with open("verilog_output.txt", "r") as file:
    data = file.readlines()
    print("Traffic Signal Output:", data)
```

   - Use Tkinter to **display real-time traffic signal updates** based on Verilog output.

Final Step: Run the Smart Traffic System

- **Step 1:** Start the Python script to open the **Tkinter GUI**.

```
bash
CopyEdit
python traffic_gui.py
```

- **Step 2:** Enter the **traffic conditions** in the GUI.
- **Step 3:** The Python script will **send inputs to the Verilog module**.
- **Step 4:** Vivado will process the signals and generate an output.
- **Step 5:** Python will **read the Verilog output** and update the GUI dynamic

VI . CONCLUSION

The Smart Traffic Control System successfully integrates a Verilog-based traffic signal controller with a Tkinter-based graphical user interface (GUI) to optimize traffic management. By dynamically adjusting signal timings based on real-time inputs such as vehicle count, ambulance presence, and pedestrian crossings, the system enhances traffic flow efficiency and emergency response time.

The hardware-independent design ensures compatibility with various FPGA platforms and traffic monitoring systems, making it adaptable for real-world implementation. The use of Python for GUI

interaction and Verilog for signal processing enables a seamless connection between software and hardware, allowing for accurate, real-time traffic control.

This system can be further enhanced by incorporating machine learning algorithms for predictive traffic analysis, IoT sensors for real-time data collection, and cloud integration for remote monitoring. With its scalability and automation capabilities, this approach can significantly improve urban traffic management and road safety.

VII. REFERENCES

[1] https://www.python.org/doc/

[2] https://docs.python.org/3/library/tkinter.html

[3] https://www.xilinx.com/