TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

A PROJECT REPORT ON

DIGITAL LOGIC SUITE

GUI BASED TOOL

SUBMITTED BY:

NABARAJ BHANDARI (081BCT041)

NIKUNJ BHUSAL (081BCT043)

NIRDESH JOSHI (081BCT044)

SUPERVISED BY:

SENIOR FACULTY MEMBER, DAYA SAGAR BARAL

SUBMITTED TO:

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

INSTITUTE OF ENGINEERING, PULCHOWK CAMPUS

SUBMISSION DATE:

2082 Bhadra 02, Monday

Acknowledgment

We would like to express our sincere gratitude to our supervisor, **Mr. Daya Sagar Baral**, Senior Faculty Member, Department of Electronics and Computer Engineering, IOE Pulchowk Campus, for his amazing guidance, helpful feedback, and constant support. His knowledge of programming and digital logic helped us understand and build this project better.

This project, **Digital Logic Suite**, has been carried out as part of the subject **Object-Oriented Programming with C++**, and it would not have been possible without the support provided by our supervisor. His expertise and insights have greatly contributed to the successful design and implementation of this project.

We extend our sincere thanks to the faculty members of the Department of Electronics and Computer Engineering at Pulchowk Campus for providing us with the necessary resources and an excellent learning environment. Their lectures on C++ programming were really helpful for this project. Our heartfelt appreciation goes to our classmates and friends who offered constructive suggestions during its development phases.

Finally, we thank Tribhuvan University for including projects in our studies, letting us use what we learned in a real project. This work has greatly improved our skills in C++ programming, software design, and working as a team.

Abstract

The Digital Logic Suite is a user-friendly app built with C++ and SFML (Simple Fast Multimedia Library) to provide features like designing, testing, and studying digital logic circuits. It uses object-oriented programming ideas to make a helpful tool for students.

Users can place logic gates like AND, OR, NOT, INPUT, and OUTPUT, then connect them with wires to build circuits. This application can be used to see how circuits work in real time and makes truth tables for the designs. Users can also enter Boolean expressions and see their simplified form on the screen.

This tool is a lighter option compared to expensive software, works well on different systems, and includes key features for learning digital logic. The project shows how to use C++ programming effectively and sets the stage for future improvements in circuit simulation.

Table of Contents

Acknowledgement.tex i

Table of Contents iii

1 Objectives 1

2 Introduction 2

3 Application 3

4 Literature Survey 4

5 Existing Systems 5

6 Methodology 7

7 Implementation 8

8 Project Scope 9

9 Project Schedule 10

10 Libraries and Tools 10

1 Objectives

The primary objective of the **Digital Logic Suite** project is to develop a practical GUI-based tool that helps students in analyzing digital logic circuits, simplifying Boolean expressions and understanding digital logic concepts. The specific objectives are:

- To apply object-oriented programming concepts such as encapsulation and abstraction to enhance code maintainability and scalability.
- To use C++ for implementing the core functionalities of the tool to understand its standard libraries.
- To develop and organize custom header files which can help to improve code reusability and modularity throughout the project.
- To learn the basics of graphical user interfaces using SFML which can enable clear and interactive visualization of digital circuits.
- To design an intuitive and accessible interface that allows users to easily construct, modify, and analyze digital logic circuits.
- To make the program run faster and work smoothly in older hardware.
- To gain skills to handle larger, more complex projects in the future.
- To work together as a team to build and complete the project successfully.

2 Introduction

Digital logic design is the foundation of modern electronics and computer engineering. It helps to create systems from basic calculators to complex devices. Tools like Logisim and Digital Works let us design and test digital circuits, but they can be hard to use with command lines, and heavy to run.

The Digital Logic Suite fixes these problems with a user-friendly tool made with SFML (Simple and Fast Multimedia Library) in C++. Instead of old command-line tools, this one has a clear graphical interface. We can see logic gates, circuits, and truth tables easily, and it uses object-oriented programming.

The Digital Logic Suite has an interactive canvas where a user can place logic gates like AND, OR, NOT along with INPUT, and OUTPUT components. It offers real-time simulation, so the user can see how the circuit works based on inputs that we choose. The tool automatically creates truth tables for the circuits designed by a user and it can also simplify the boolean expressions that the user enters

The Digital Logic Suite is designed with modularity, using core object-oriented programming ideas like encapsulation, inheritance, and polymorphism in classes such as Gate, Wire, and Simulator. The SFML graphical interface works on different platforms (tested in Windows and Linux) and renders efficiently.

3 Application

The Digital Logic Suite serves as a versatile tool with applications that are listed below:

- Students can learn digital logic by building and testing circuits visually. It can make understanding gates and circuits easier.
- Users can create and simulate basic electronic systems quickly.
- The tool generates truth tables for circuits automatically. It saves time for students who are studying logic behavior.
- Users can input boolean expressions and get simpler versions. This can help to optimize circuit designs.
- The tool works on multiple platforms, making it easy to use in classrooms or personal projects.
- Users can model and test logic algorithms (e.g., adders, multiplexers) for correctness and understanding.
- Unlike resource-heavy commercial tools, this project's lightweight SFML-based GUI runs smoothly on older hardware too.

4 Literature Survey

Digital logic simulators are widely used simulation models that are used in digital circuit designs to model and test logic circuits before physical implementation. Early simulators, such as Logisim (Berg, 2008), provided a graphical interface for designing combinational and sequential circuits, allowing students to interactively test logic gates and simple circuits. However, it cannot parse Boolean expressions directly, and lacks features for automating analysis or generating truth tables and Karnaugh maps through scripts.

Another digital logic simulator, Digital Work, developed by Mecanique Ltd. enables users to construct and analyze digital logic circuits through real-time simulation. It can perform simple simulations of circuits and show how signals change over time. The software allows users to design circuits using basic gates (AND, OR, NOT, etc.), flip-flops (D, JK, RS). However, it does not support direct Boolean expression evaluation or advanced minimization features.

With the rise of modern graphics libraries, several implementations have explored using frameworks like Qt, SDL, or SFML for improved user interfaces and real-time simulation. Wired Panda is an open-source digital logic simulator developed in C++ using the Qt framework. It provides a real-time, interactive environment for designing and simulating combinational and sequential digital circuits. It supports a variety of logic gates as well as flip flops.

Some websites provide online calculators to evaluate Boolean expressions or generate truth tables. While they can be convenient for quick tasks, they do not support advanced features like K-map minimization, simulation, or visualizing circuits. They also require internet access, which is not always available or reliable in every environment.

Despite the usefulness of these tools, there is a clear gap in the availability of a lightweight, cross-platform suite that combines Boolean expression parsing, truth table generation, K-map minimization, circuit simulation. Our Digital Logic Suite aims to fill this gap by providing a single, modular tool that allows users to perform all these tasks in one place, with the possibility of extending the suite in the future.

5 Existing Systems

Several tools for digital logic design are available, but they have limitations that our Digital Logic Suite aims to address. Some of those systems are listed along with their limitations below:

- Logisim: Logisim is a popular tool for learning how to design and simulate digital circuits, widely used by students and educators, but it doesn't work with modern C++. It is built in Java, which makes it slow on older or less powerful computers.
- Digital Works: This is a paid software that allows users to simulate basic logic gates and simple circuits. However, it lacks features like generating truth tables or simplifying Boolean expressions, which makes it less helpful for students.
- Logic Friday: Logic Friday is a free tool focused on simplifying Boolean expressions. It only works on Windows (does not have a cross-platform support), has no circuit simulation, and uses an old, hard-to-use interface..
- Online Boolean Calculators: Tools like CircuitVerse are web-based and allow users to design circuits and perform basic logic analysis online. However, they require a constant internet connection, can be slower than local applications, and may not offer the same performance as a C++-based tool like ours.

Key Improvements in Our System:

- Our Digital Logic Suite uses SFML to create a clear and responsive graphical interface, making it easy for users to place logic gates and build circuits. Our application runs smoothly even on less powerful computers.
- Our tool supports the full process of designing circuits, simulating them, generating truth tables, and simplifying Boolean expressions. This all-in-one approach makes it more useful for students.
- The Digital Logic Suite works on multiple operating systems, including Windows, Linux, and macOS, ensuring that users can access it regardless of their device.
- Unlike web-based tools like CircuitVerse that depend on an internet connection, our app runs entirely offline which can make it more reliable and faster for users in areas with poor internet.

6 Methodology

This project, “Digital Logic Suite,” is a C++ application that uses the SFML graphics library and follows object-oriented programming (OOP) paradigm. We made different classes with private and public members to keep the code organized and safe. We used code reusability and data abstraction to make the project better. Each object has its own functions to handle events, and these work together to create the interactive simulation.

The Digital Logic Suite solves problems found in old digital logic simulators by giving users an easy-to-use tool made in C++ with SFML (Simple and Fast Multimedia Library). Unlike older tools that use only the command line, this simulator has a clear graphical interface. Users can see logic gates, circuit connections, and truth tables easily. By using OOP, we organized the code into classes, making it easier to reuse and maintain. Users can interact with the simulation, place components, and see how circuits work in real time. This makes it useful for learning and designing digital logic circuits.

Basic C++ graphics were not enough for our needs. To make a better and more interactive interface, we used SFML version 3.0.0. SFML gave us good graphics, worked on different systems, and it was easy to use for drawing, handling events, and getting input. We used VS Code as our IDE and the GCC compiler, which helped us to code, compile, and debug on different platforms.

Before starting, we learned SFML by reading tutorials, official documents, and online guides. We also reviewed OOP concepts to design a system that is easy to manage. SFML is used for features like making the grid layout, displying connections with wires, simulating logic gates, showing the component palette, drawing the window, creating buttons and the visual layout of the program.

7 Implementation

8 Project Scope

The Digital Logic Suite is mainly created to help students, teachers, and anyone interested in learning about digital logic circuits. It is made for academic and educational purposes, so it focuses on providing clear and useful tools to understand how digital logic works.

This project provides a command-line interface (CLI) tool, which means users can type commands in a terminal or console to use the program. This makes the tool lightweight and easy to run on many different computers without needing a complex graphical interface.

The suite is designed to be modular. This means it is built in separate parts or modules that can work independently but also fit together smoothly. Because of this modular design, it will be easier to add new features or improve existing ones in the future without needing to rewrite the entire system.

The main focus of the Digital Logic Suite is on combinational logic, which is a type of digital logic where the output depends only on the current inputs. The tool will provide features to analyze these logic circuits, simplify or minimize the logic expressions to make them easier to understand and use, and simulate how the logic behaves with different inputs.

Overall, this project aims to be a simple, effective, and flexible tool that supports learning and experimentation in digital logic.

9 Project Schedule

Week	Planned Activities
Week 1	Requirements gathering, and system design.
Week 2	Implementation of CLI and Boolean expression parser.
Week 3	Truth table generation and Karnaugh map minimization modules.
Week 4	Circuit simulation and Graphviz visualization integration.
Week 5	Testing, documentation, and final report preparation.

Table 1: Project Schedule for Digital Logic Suite

10 Libraries and Tools

- **C++ Standard Template Library (STL):** Utilized for object-oriented programming concepts and basic data structures.
- **Graphviz:** Visualization of logic circuits and truth tables.
- **Visual Studio:** Development environment.
- **Git:** Version control and collaboration.