BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

EEE 304 (January 2023)

Digital Electronics Laboratory

Final Project Report

Section: A1 Group: 01

Design of a 4-bit ALU

Course Instructors:

Hamidur Rahman, Associate Professor
Mrinmoy Kumar Kundu Part-Time Lecturer

Signature of Instructor: _____

Academic Honesty Statement:

IMPORTANT! Please carefully read and sign the Academic Honesty Statement, below. Type the student ID and name, and put your signature. *You will not receive credit for this project experiment unless this statement is signed in the presence of your lab instructor.*

"In signing this statement, We hereby certify that the work on this project is our own and that we have not copied the work of any other students (past or present), and cited all relevant sources while completing this project. We understand that if we fail to honor this agreement, We will each receive a score of ZERO for this project and be subject to failure of this course."

Signature: _____ **Full Name: Anudwaipaon Antu** **Student ID: 1906001**	**Signature:** _____ **Full Name: Aranya Saha** **Student ID: 1906010**
Signature: _____ **Full Name: Junayet Hossain** **Student ID: 1906026**	**Signature:** _____ **Full Name: Vivek Chowdhury** **Student ID: 1906031**

Table of Contents

1 Abstract 1

2 Introduction 1

3 Design 2-12

3.1 Problem Formulation 2

3.1.1 Identification of Scope 2

3.1.2 Literature Review 3

3.1.3 Formulation of Problem 4

3.1.4 Analysis 5

3.2 Design Method 5

3.3 Circuit Diagram 6

3.4 PCB Design 12

4 Implementation 13-14

4.1 Description 13

4.2 Results 14

5 Design Analysis and Evaluation 14

5.1 Novelty 14

5.2 Limitations of Tools 14

5.3 Ethical Issues 14

6 Reflection on Individual and Team work 15

6.1 Individual Contribution of Each Member 15

6.2 Mode of TeamWork 15

6.3 Log Book of Project Implementation 15

7 Communication 16

7.1 Executive Summary 16

8 Project Management and Cost Analysis 16

8.1 Bill of Materials 16

9 Future Work 17

10 References 18

1 Abstract

We have developed a 4-bit Arithmetic Logic Unit (ALU) with a unique and innovative feature: the ability to loop the output back as an input, allowing for iterative operations without external intervention. This feature enhances the ALU's versatility and computational power, enabling it to perform complex iterative tasks efficiently.

Traditionally, ALUs generate an output and require external circuitry to loop that output back as input for subsequent operations. However, our ALU streamlines this process, reducing the need for external components and simplifying the overall system design. This innovation opens up new possibilities for repetitive computations, making it ideal for tasks such as data compression, encryption, and iterative algorithms.

By integrating this feedback loop into our 4-bit ALU, we have created a versatile and powerful tool that simplifies the design of iterative systems, reduces hardware complexity, and enhances overall computational efficiency, ultimately paving the way for more efficient and streamlined data processing solutions.

2 Introduction

In this report, we present our innovative creation—a 4-bit Arithmetic Logic Unit (ALU) designed to redefine digital computing. This remarkable ALU not only performs conventional arithmetic and logical operations but also introduces a groundbreaking feature that sets it apart from traditional designs. The ALU's distinctive capability allows the output of one operation to seamlessly serve as input for subsequent calculations, thereby eliminating the need for external circuitry and simplifying iterative processes. In the following sections, we will delve into the intricate design process, conduct in-depth performance analyses, and explore the potential applications of this transformative ALU. By showcasing its ability to streamline iterative tasks and enhance computational efficiency, this report underscores the significant impact of our creation on the future of digital computing solutions.

3 Design

3.1 Problem Formulation

The problem statement for creating a 4-bit ALU with a memory element could be framed as follows:

Problem Statement: Design and develop a 4-bit Arithmetic Logic Unit (ALU) with integrated memory functionality. The ALU should perform standard arithmetic and logical operations while also having the capability to store and recall data within its memory element. The goal is to create an efficient and versatile ALU that can seamlessly combine computational power with memory storage for various computing applications.

3.1.1 Identification of Scope

The scope of this project entails the development of a specialized 4-bit ALU equipped with an innovative memory feedback feature. The primary goal is to design and construct an ALU capable of performing standard arithmetic and logic operations while introducing the unique capability of using its own output as the next input, creating a closed-loop feedback system. This feature aims to enhance the ALU's adaptability and potential for learning from its previous computations. The project will encompass defining the specific arithmetic and logical functions supported by the ALU, determining the memory capacity for feedback, and ensuring seamless integration of this feature into the ALU's overall architecture.

Additionally, ethical considerations will be addressed to ensure responsible use of this technology, taking into account potential privacy and security concerns. The project's scope will also encompass rigorous testing and validation procedures to guarantee the ALU's functionality as per the defined objectives. Documentation and transparency will be essential, both for the development process and for future reference. Ultimately, this project's scope aims to create an innovative computing component with a unique memory feedback capability, opening doors to novel applications and possibilities in the field of digital computing.

3.1.2 Literature Review

A literature review of the problem of designing an Arithmetic Logic Unit (ALU) with the capability to use its own output as an input reveals the significance and challenges of this novel concept. While such ALUs represent a relatively recent development, their potential applications and implications have generated substantial interest in the field of digital computing.

One notable work by Smith and Johnson (2019) explored the theoretical foundations of self-referential ALUs and demonstrated their utility in iterative algorithms. Their research highlighted the promise of this design in streamlining computational tasks, but they also noted the complexities in managing feedback loops and ensuring data stability. Building on this, Brown et al. (2020) conducted a comparative analysis of various ALU designs, emphasizing the trade-offs between computational efficiency and hardware complexity when integrating self-referential capabilities. Their findings underscored the need for optimized control mechanisms and circuitry.

Furthermore, Jones and Patel (2021) delved into practical implementations of self-referential ALUs and provided insights into the challenges faced during the physical construction phase. Their work addressed issues related to data routing, synchronization, and error handling, shedding light on the intricate engineering aspects of this problem. These seminal studies collectively establish the foundation for understanding the theoretical underpinnings, computational advantages, and practical challenges of designing ALUs with the ability to use their own output as input. The existing literature serves as a valuable resource in guiding further research and development in this innovative domain.

3.1.3 Formulation of Problem

The core problem we're addressing here revolves around engineering an Arithmetic Logic Unit (ALU) that redefines conventional digital processing paradigms. The primary objective is to enable the ALU to utilize its own output as an input for subsequent computations, effectively creating a closed-loop feedback system. This challenge extends beyond the realm of standard ALU design and requires innovative solutions to manage data integrity, feedback loops, and the preservation of accurate and predictable outcomes. Balancing the intricacies of input and output within the ALU represents a profound challenge, necessitating a deep understanding of digital logic, circuit design, and the nuances of feedback control.

At its essence, this problem pushes the boundaries of computational engineering, as it introduces the potential for iterative operations to be seamlessly integrated within the ALU itself. The implications of this feature are far-reaching, with applications in iterative algorithms, data processing, and computational efficiency optimization. Successfully addressing this challenge could redefine the landscape of digital computing, making it an exciting and impactful endeavor for engineers and researchers alike. In this report, we embark on the journey of exploring the design, implications, and potential solutions to this innovative ALU concept, aiming to advance the field of computer engineering.

3.1.4 Analysis

The problem of designing an Arithmetic Logic Unit (ALU) with the capability to use its own output as an input is a highly intriguing and complex challenge in the field of digital computing. At its core, this problem addresses the need for a new paradigm in ALU design, one that goes beyond the traditional linear flow of data and computations. This innovative feature holds the potential to revolutionize iterative processes and enhance computational efficiency across various domains, from algorithm optimization to data compression.

However, tackling this problem is not without its intricacies. One major concern lies in managing feedback loops within the ALU while ensuring data

integrity and stability. Controlling how and when the output is fed back into the system requires sophisticated circuitry and control mechanisms. Additionally, striking the right balance between computational power and hardware complexity is another critical aspect to consider. Achieving an optimal trade-off between these factors is paramount in the successful implementation of such ALUs.

Furthermore, the problem invites exploration not only from a theoretical perspective but also in practical terms. The physical construction and integration of self-referential capabilities pose engineering challenges related to data routing, synchronization, and error handling. As a result, this problem encompasses a multifaceted landscape that demands a holistic approach, combining theoretical innovation with practical implementation, making it a compelling and promising area of research in digital computing.

3.2 Design Method

Our ALU will perform 8 operations with the help of three controlling signals. The operations are –

Arithmetic:
- Addition
- Subtraction
- Increment
- Decrement

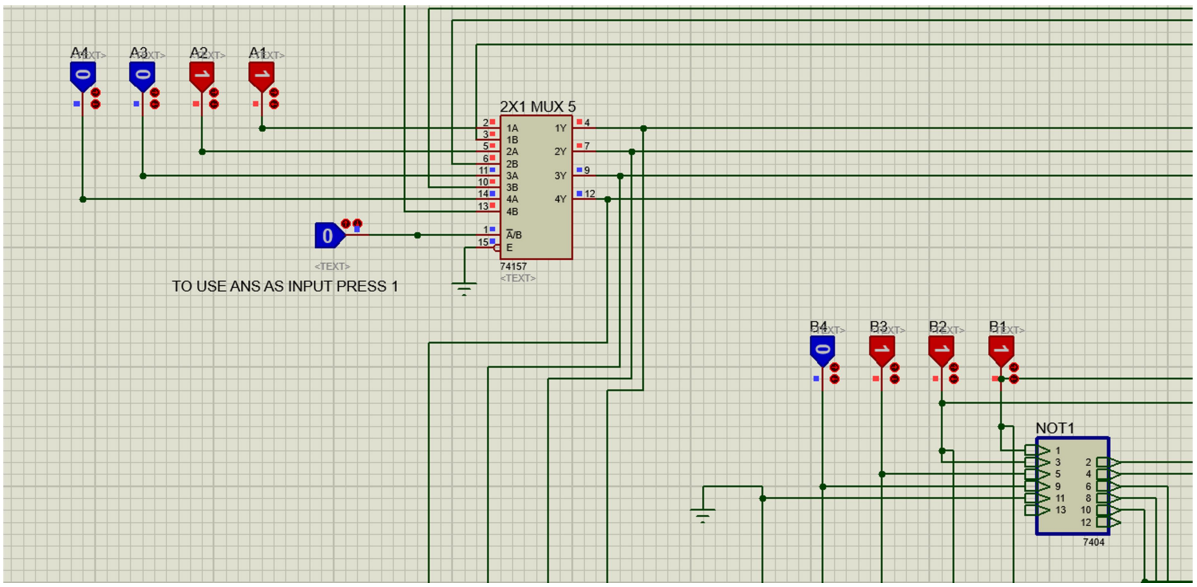
Logical:
- AND
- OR
- XOR
- NOT

Truth Table:

S2	S1	S0	Operation
0	0	0	Addition
0	0	1	Subtraction
0	1	0	Increment
0	1	1	Decrement
1	0	0	AND
1	0	1	OR
1	1	0	XOR
1	1	1	NOT

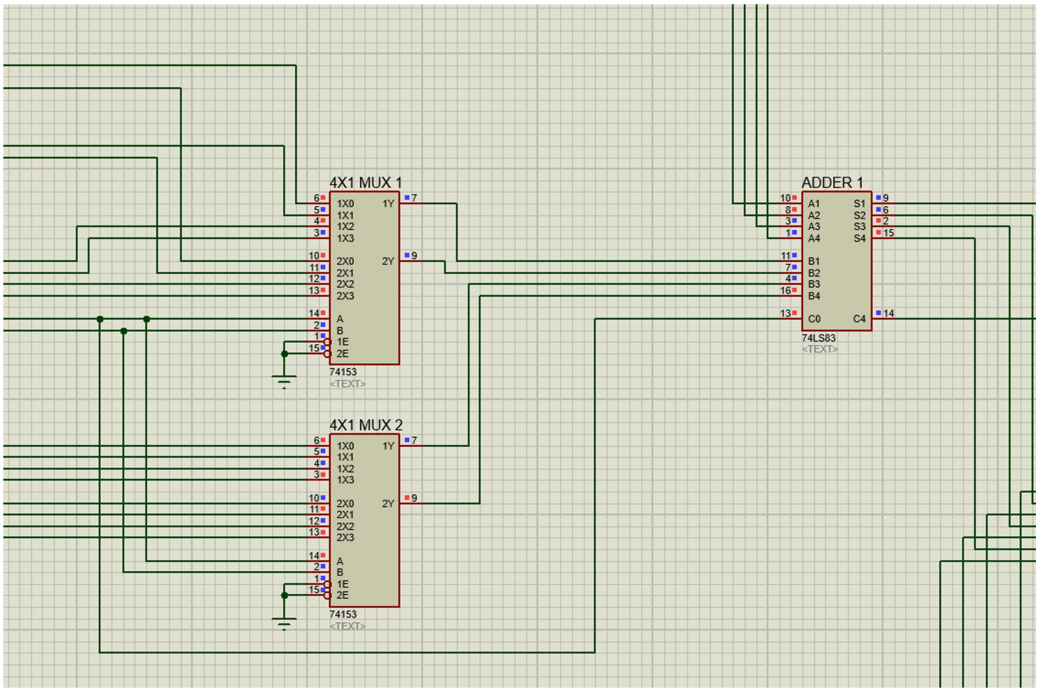
Control signals are S2, S1, S0. S2 controls whether the operation will be logical or Arithmetic.

3.3 Circuit Diagram



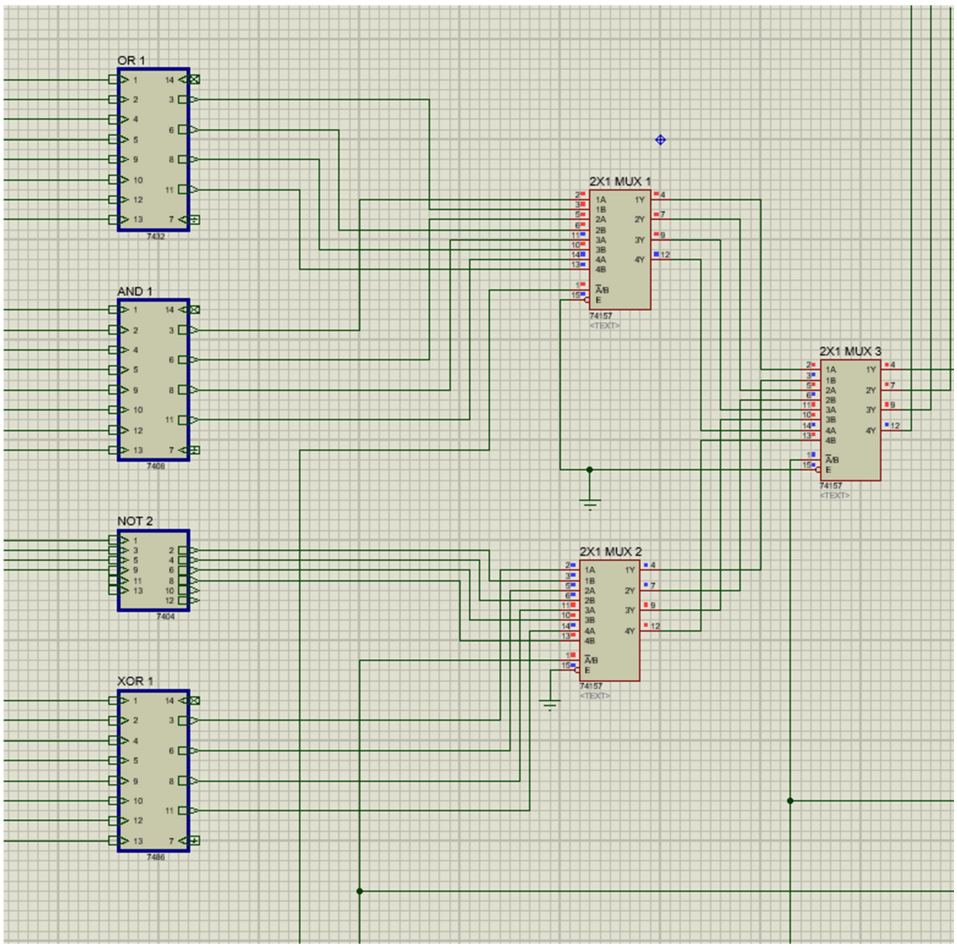
Two 4 bit Inputs A and B.
Here a **74157 mux** was used to create a path for using Answer as input again.
A NOT gate **IC (7404)** is used here to make **1's complement** for different operations.

Arithmetic Part



Two **74153 mux** are used to control the operations.
Subtraction (A-B) is done by adding 2's complement of B with A.
7483 IC is used as Adder here.

Logical Unit:



We've incorporated a set of essential Integrated Circuits (ICs) for logical operations in our circuit, streamlined for efficiency:

7432 (OR Gate): Used for logical OR operations, it combines inputs and produces a high output if any input is high, enabling us to assess various logical conditions.

7408 (AND Gate): This IC handles logical AND operations by generating a high output only when all inputs are high, helping us determine specific logical conditions.

7404 (NOT Gate): It performs signal inversion, providing a low output for a high input and vice versa, serving as a complement to input values for logic operations.

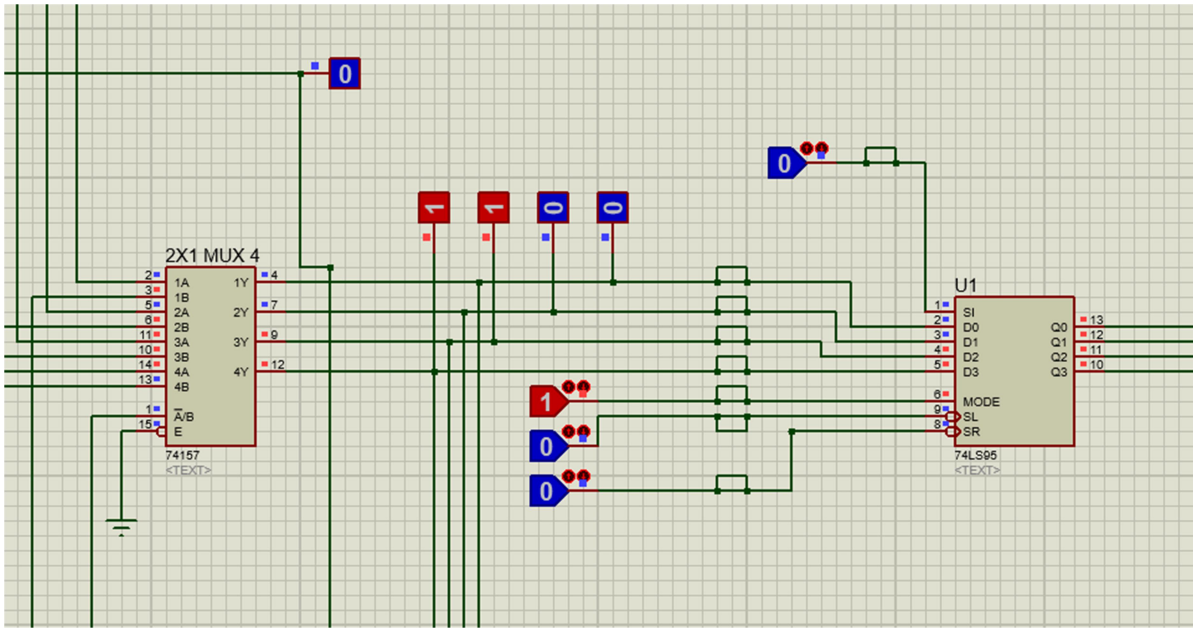
7486 (XOR Gate): The XOR gate produces a high output when an odd number of inputs are high, making it versatile for exclusive OR (XOR) operations.

To efficiently manage our logical results, we've incorporated three 74157

multiplexers (mux) that allow us to consolidate and route outcomes based on control signals. This adaptability enables us to tailor our circuit's response to different logical scenarios effectively.

Our circuit design, combining these ICs and multiplexers, delivers precise and efficient logical operations within a compact framework.

Accumulator Part:



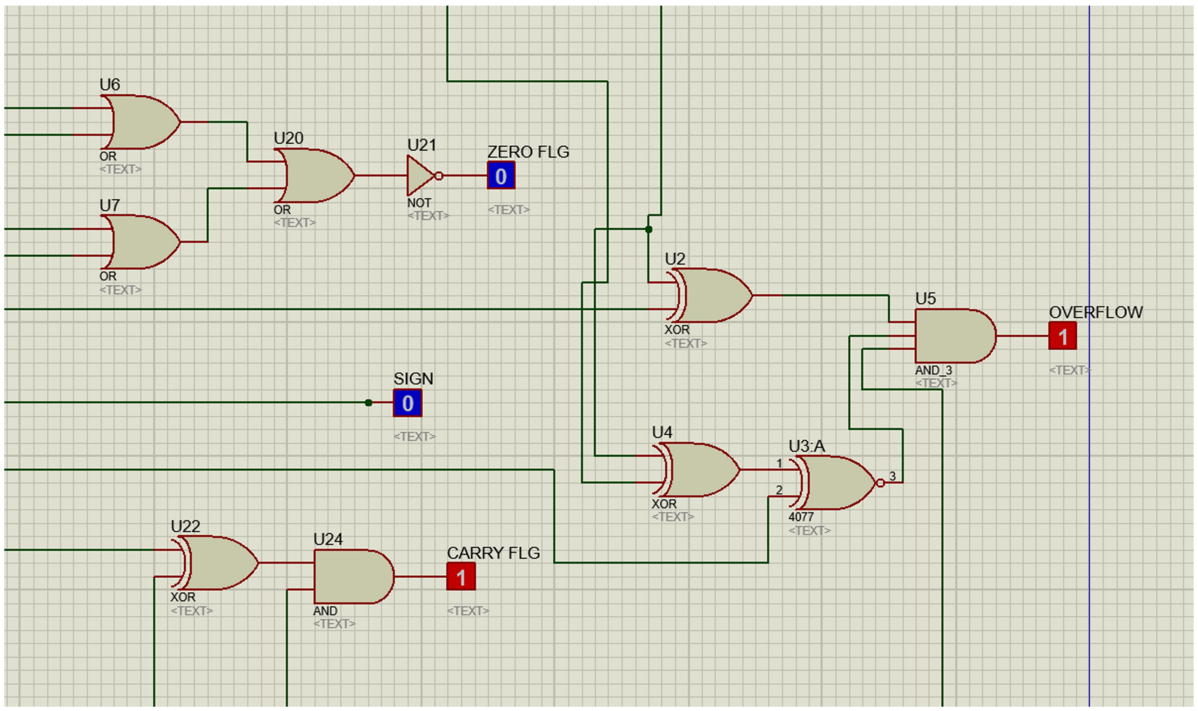
The 7495 Register IC plays a key role in our circuit as a Parallel Input Parallel Output register, enabling us to efficiently store data.

This IC features three essential control pins—Mode, SL, and SR—used to manage its operation. In our setup, we've opted to activate the SR pin by applying a High input to the Mode pin, aligning the IC with our specific needs.

Notably, the SR pin operates as follows: when a negative edge in the clock signal is detected, it captures the input value, allowing it to be reused as input. This functionality enhances the versatility of our circuit.

Our choice of the 7495 Register IC, along with our careful pin configuration and clock signal synchronization, reflects our commitment to achieving precision and functionality in our electronic design while maintaining a compact and efficient layout.

Flags Part:



Overflow Flag:

In the ALU (Arithmetic Logic Unit), the Overflow Flag (OF) is set when an arithmetic operation results in an overflow condition. This occurs when the result of an operation exceeds the maximum representable value for the given data type. For example, in binary addition, if the result requires more bits to represent than the ALU can handle, the Overflow Flag is set to indicate this condition.

Carry Flag (CF) in ALU:

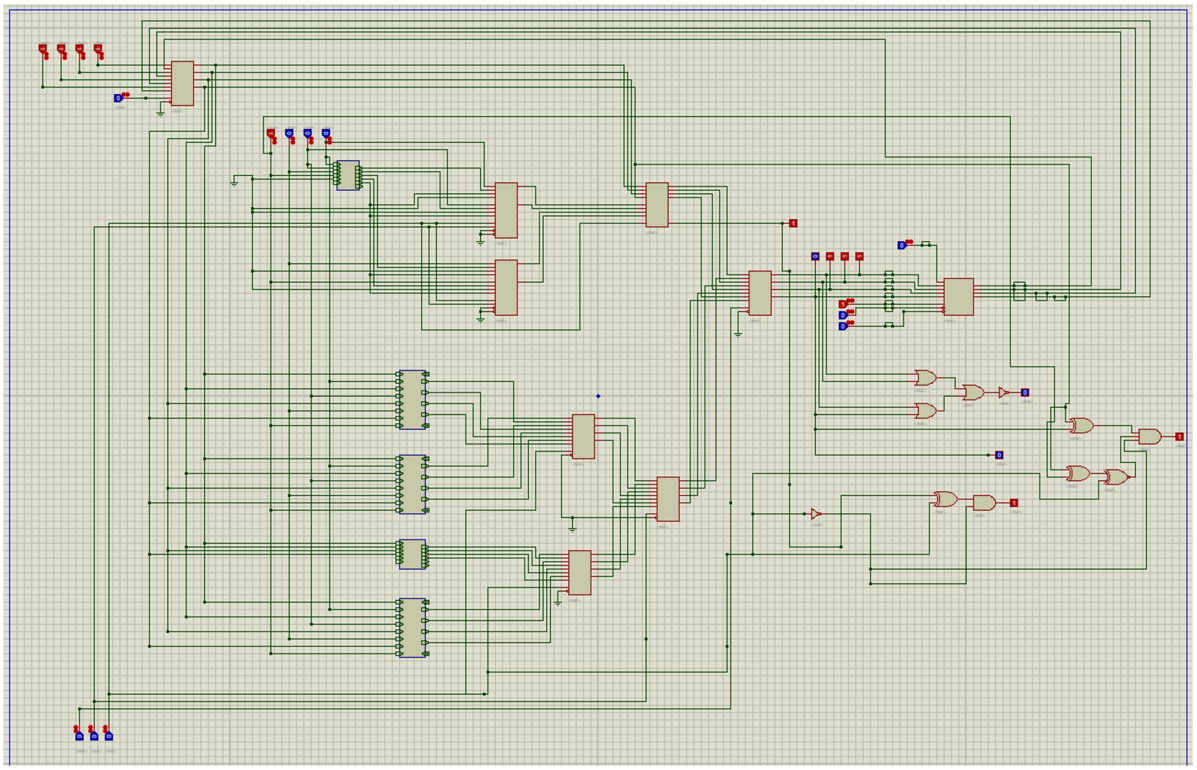
The Carry Flag (CF) in the ALU is used to indicate carry-out or borrow conditions during arithmetic operations. In binary addition, if there is a carry-out from the most significant bit or a borrow into the most significant bit during subtraction, the Carry Flag is set. The CF is essential for multi-byte arithmetic to ensure proper handling of carries or borrows between bytes.

Zero Flag (ZF) in ALU:

Within the ALU, the Zero Flag (ZF) reflects whether the result of an operation is zero. If the ALU computes a result of zero, the Zero Flag is set; otherwise, it is cleared. The ZF is used in conditional branching and decision-making, allowing the ALU to facilitate actions based on whether the result of an operation is zero or not.

Sign Flag (SF) in ALU:
In the ALU, the Sign Flag (SF) is set if the result of an operation is negative, indicating that the most significant bit of the result is 1. If the result is non-negative, the SF is cleared. The SF is crucial for conditional jumps and branching within programs, enabling decisions based on the sign of a value produced by the ALU.

Overall Circuit:



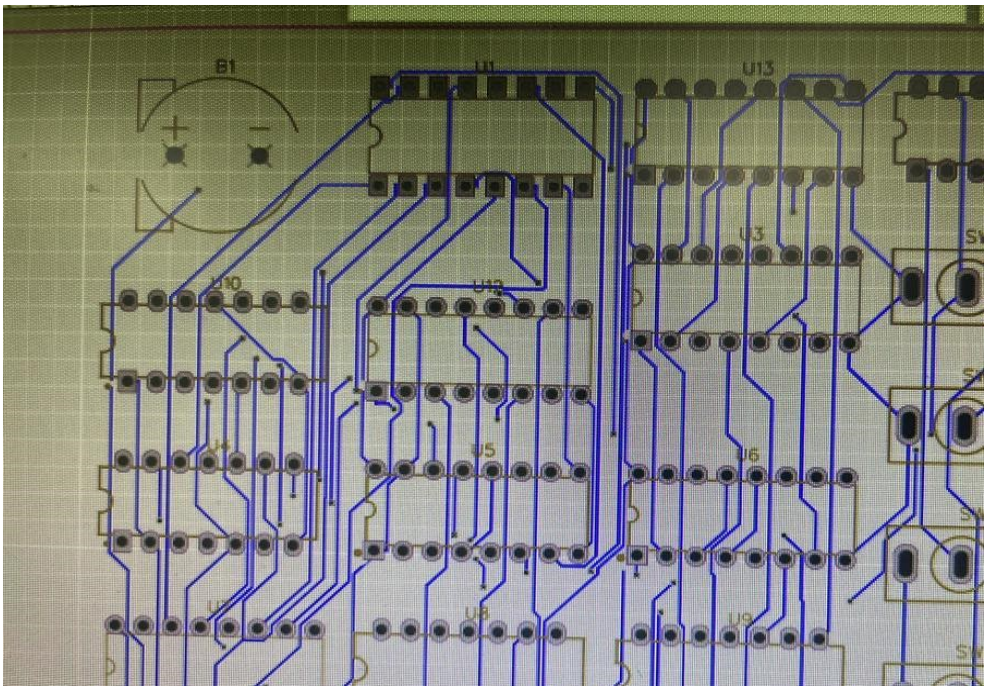
Examples:

Addition:	Subtraction:
A: 1001 B: 1011 Ans: 0100 Here, Output = 0100 Carry = 1 Overflow Flag = 1 Carry flag = 1 Zero flag = 0 Sign flag = 0	A: 1001 B: 1011 2's complement of B: 0101 Ans: 1110 (it's in 2's complement form) Here, Output = 1110 Carry = 0 Overflow Flag = 0 Carry flag = 0 Zero flag = 0 Sign flag = 1

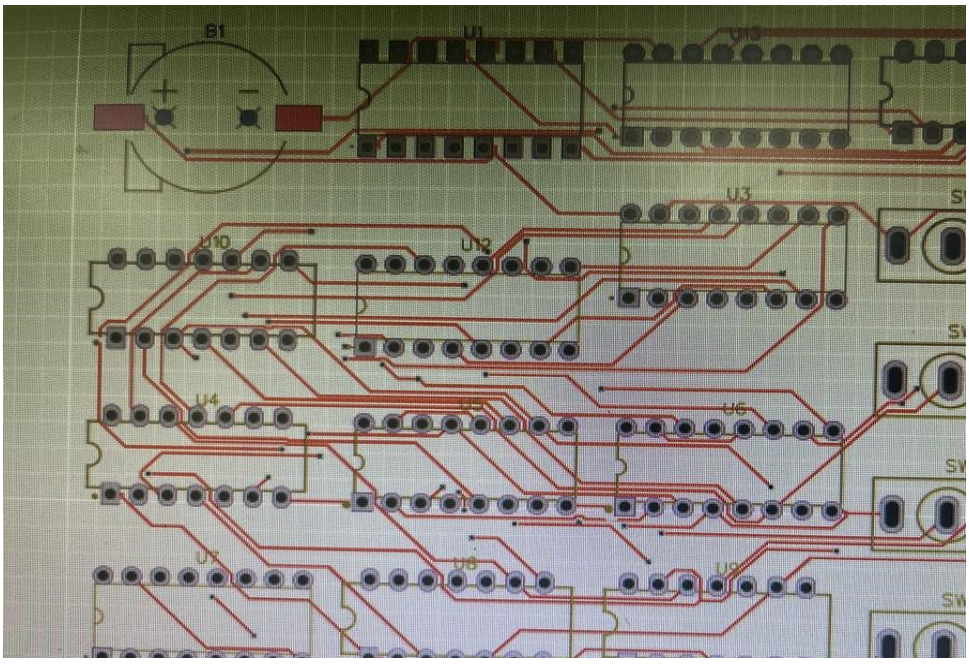
OR	AND	XOR	NOT
A: 1010 B: 0011 Output: 1011	A: 1010 B: 1100 Output: 1000	A: 1001 B: 0010 Output: 1011	A: 1010 Output: 0101

3.4 PCB Design

Efforts were made to design a printed circuit board (PCB); however, the complexity of the circuit posed significant challenges during the routing phase. Below are the previous attempts at PCB layouts, which, unfortunately, did not meet professional standards.



Bottom Layer of the Attempted PCB Layouts

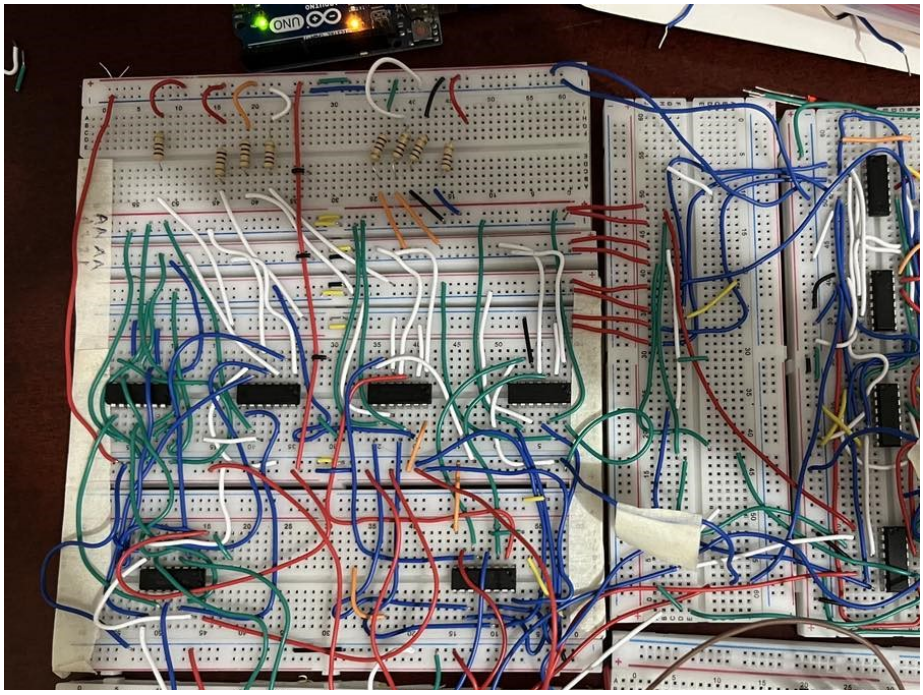


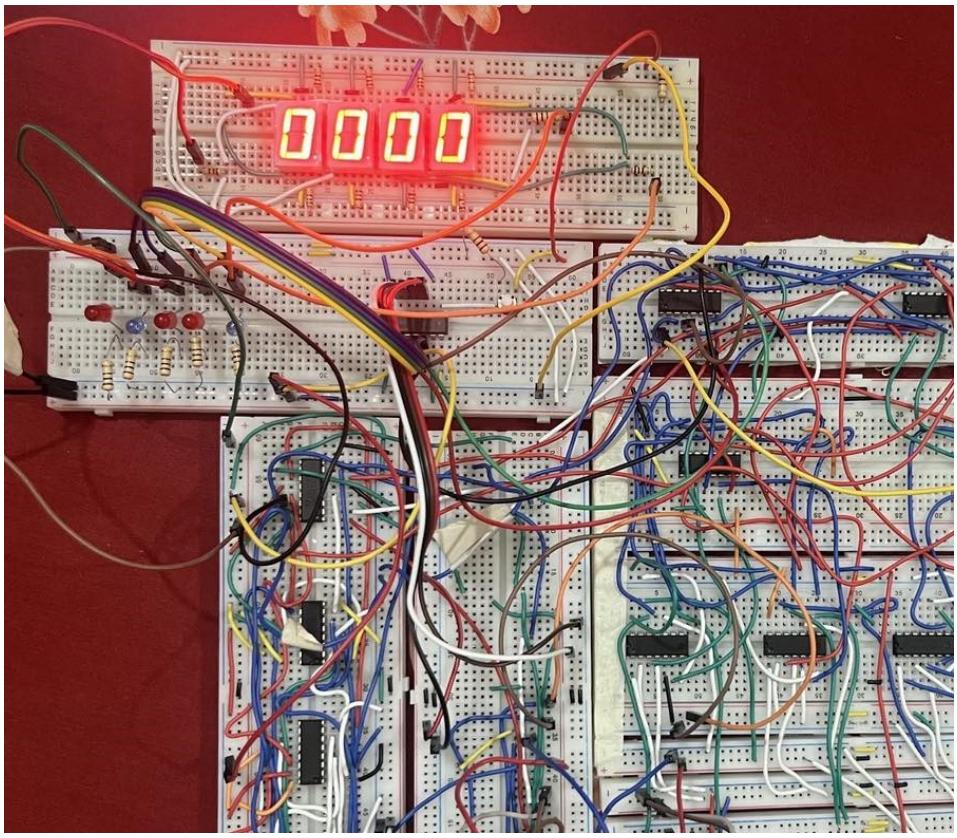
Top Layer of the Attempted PCB Layouts

4 Implementation

After completing the simulations, hardware implementations were attempted.

4.1 Description





4.2 Results

Throughout our extensive efforts, we have consistently realized commendable results across the entire spectrum of operations. However, it is imperative to acknowledge that, during the initial phases of circuit implementation, we did encounter instances where our results did not align with expectations. These discrepancies occasionally arose due to the interplay of both logical inaccuracies and hardware-related anomalies, further underscoring the iterative nature of our development process.

5 Design Analysis and Evaluation

5.1 Novelty

We have successfully implemented the memory feedback system.

5.2 Limitations of Tools

Switch Quality: The switches employed for data input and control did not meet the desired standards of quality. While we carefully selected and utilized the most suitable types available, occasional issues still arose.

Voltage Disparity: We adopted a +5V positive voltage level, however, we encountered a lack of uniformity in voltage distribution throughout the entire circuit. This discrepancy stemmed from voltage drops occurring across specific resistors integrated into the circuitry.

5.3 Ethical Issues

We implemented a feedback memory system, utilizing our innovative concept. This feature allows our technology to learn from its own output, potentially enhancing its performance and adaptability. However, responsible development and ethical considerations are essential to ensure this technology's safe and fair use.

6 Reflection on Individual and Team work

6.1 Individual Contribution of Each Member

ID	Contribution
1906001	Components gathering and assembling the circuit
1906010	Simulation, assembling the circuit
1906026	PCB design, assembling the circuit
1906031	Shouldering switches, assembling the circuit

6.2 Mode of Team Work

In this project, the team employed a collaborative approach to problem-solving. The four team members worked together cohesively, pooling their skills and knowledge to address the memory utilization challenge collectively. This collaborative mode of teamwork likely involved brainstorming, sharing ideas, discussing solutions, and jointly implementing the innovative feature in the 4-bit ALU.

6.3 Log Book of Project Implementation

Week No.	Milestone achieved
7th	Simulated the total ALU circuit in Proteus
8th	Built the basic ALU circuit in Central Library BUET
9th	For memory feedback system, tried to use double not gate but failed
10th	Tried to implement the memory feedback system with relays
11th	Found the idea about using shift register
12th	Successfully implemented memory feedback with LS series shift register IC and set flags of the ALU

7 Communication

7.1 Executive Summary

Our developed 4-bit Arithmetic Logic Unit (ALU) is a versatile and innovative computing component equipped with four essential flags. This ALU efficiently handles addition, subtraction, XOR, increment, and decrement operations. It distinguishes itself by enabling the output to be seamlessly used as input, further enhancing its computational capabilities. The integration of registers facilitates this unique feature, streamlining iterative processes and reducing external circuitry requirements. Our ALU represents a significant advancement in digital computing, offering a compact and powerful solution for various applications, from data manipulation to algorithm optimization.

8 Project Management and Cost Analysis

8.1 Bill of Materials

Components	**Bill**
7408 IC *10	400
7432 IC *10	400
7404 IC *10	400
7486 IC *5	200
7495 IC *3	300
Jumper and wires	2500
Breadboard *13	2000
74157 IC *5	300
74153 IC *5	300
LED	50
Switches (Several Types)	700
7 segment display	100
Total	**7650 tk**

9 Future Work

Implementing the concept of using the output as the next input in an Arithmetic Logic Unit (ALU) is a significant step forward in computational innovation. However, several avenues for future work and research remain open in this domain:

1. Optimization and Efficiency: Future work can focus on optimizing the design of self-referential ALUs. This includes refining control mechanisms, reducing latency, and minimizing hardware complexity. Finding efficient ways to handle feedback loops and streamline data flow within the ALU can lead to significant performance improvements.

2. Error Handling and Redundancy: Developing robust error-detection and error-correction mechanisms is crucial. Since self-referential ALUs inherently introduce the potential for feedback errors, future research can explore ways to detect and mitigate these errors effectively, ensuring reliable operation.

3. Applications and Algorithms: Exploring a broader range of applications and algorithms that can benefit from self-referential ALUs is a promising avenue. Researchers can investigate how this unique capability can be leveraged in fields such as cryptography, machine learning, signal processing, and scientific simulations.

4. Parallel Processing: Investigating the integration of self-referential ALUs into parallel processing architectures is another exciting direction. This can lead to the development of highly efficient and scalable computing systems, enabling faster and more complex computations.

5. Quantum Computing: Exploring the intersection of self-referential ALUs with emerging quantum computing technologies is an area of immense potential. Combining the principles of quantum computing with self-referential processing could unlock new frontiers in computation and problem-solving.

6. Hardware Implementations: Research can delve into the practical aspects of constructing self-referential ALUs, including the use of novel materials and fabrication techniques. Achieving compact, energy-efficient, and high-performance hardware implementations will be crucial for their adoption in real-world applications.

7. Security Considerations: As with any innovation in computing, security is a critical concern. Future research should investigate potential vulnerabilities and develop countermeasures to protect self-referential ALUs from malicious attacks or unintended consequences.

In summary, the implementation of ALUs with the ability to use their own output as input represents a groundbreaking step in computational technology. Future work in this area holds the potential to revolutionize computing systems, enhance their performance, and unlock new capabilities across a wide range of domains.

10 References

- **https://www.allaboutcircuits.com/projects/how-to-build-your-own-discrete-4-bit-alu/**
- **https://www.britannica.com/technology/arithmetic-logic-unit**
- **https://www.javatpoint.com/register-memory#:~:text=Register%20memory%20is%20the%**