Arithmetic and Logic Unit

[1]Akansha Singh, [2]K B Ramesh*
[1] Student (EIE), [2]Associate Professor, R V College of Engineering, Bengaluru, Karnataka, 560059 India

***Corresponding Author**
E-mail Id: - rameshkb@rvce.edu.in*

ABSTRACT
This research paper explores the fundamental digital circuit known as the Arithmetic and Logic Unit (ALU). The ALU is an essential component of any central processing unit (CPU) and is responsible for executing arithmetic and logical instructions within a computer's architecture. The paper examines the ALU's function in detail, focusing on its ability to process data by executing mathematical and logical operations such as addition, subtraction, multiplication, division, logical AND, OR, NOT, and XOR. The paper also explores analyzing the internal structure and operation of an ALU, and implementing the ALU using hardware description languages. The methodology is evaluated through the design and implementation of a basic ALU, and the results demonstrate the effectiveness of the methodology in achieving a deep understanding of the function and operation of an ALU.

***Keywords:**-Arithmetic and Logic Unit (ALU), Central Processing Unit (CPU), Computer architecture, Mathematical operations, Hardware description languages*

INTRODUCTION

The arithmetic logic unit (ALU) is a key component of modern computing devices and plays a central role in executing arithmetic and logic instructions within the central processing unit (CPU). This research paper covers the basic aspects of ALU and analyzes its function, structure and functions in detail. The aim is to provide a comprehensive understanding of the ALU's ability to process data by performing arithmetic and logic operations such as addition and logical AND. This paper examines the internal structure of the ALU and evaluates the methodology for implementing his ALU using a hardware description language. The research methodology was evaluated by designing and implementing a basic ALU, and the results demonstrate the effectiveness of the methodology in achieving a detailed understanding of ALU function and behavior. The findings from this research have practical implications in the field of computer architecture and help design more efficient and effective computing devices.

BACKGROUND

The ALU was first introduced in the 1950s as a basic digital circuit that could perform arithmetic and logical operations on binary data. Since then, the ALU has evolved and become more complex, with modern ALUs capable of performing complex operations and supporting various data types. The ALU is a cru`cial component of any central processing unit (CPU) and is responsible for executing instructions within a computer's architecture.

THEORETICAL FOUNDATION:

A. Arithmetic logic unit:

Combinatorial logic circuits that enable engineering students to learn using Boolean algebra, which unites the worlds of mathematics and logic.

Truth Table:

A	B	Result
0	0	0
0	1	0
1	0	0
1	1	1

B. K-Map:
A Karnaugh map is a graphical tool used to simplify Boolean algebraic expressions. It reduces the need for extensive calculations by allowing inspection to simplify expressions.

C. Lenguaje blueprint:
All languages share a common underlying structure that governs their interaction with the ALU, and we assume that this structure allows them to be understood and generated. This theory has been further developed by other linguists and has proven to be an accurate model of how language works.

METHODOLOGY

1. Understand the basic function of an ALU: The first step in analyzing the function of an ALU is to understand its basic function, such as performing arithmetic and logical operations on binary numbers.
2. Analyze the internal structure and operation of an ALU: The second step is to analyze the internal structure and operation of an ALU, including the selection of operations, the use of logic gates, and the flow of data through the unit.
3. Implement the ALU using hardware description languages: The final step is to implement the ALU using hardware description languages, such as Verilog or VHDL, which allow for the creation of a digital model that can simulate the operation of the ALU.

FUNCTION OF AN ALU
The basic function of an arithmetic logic unit (ALU) is to perform arithmetic and logic operations on binary data. An ALU typically takes two inputs, performs some operation, and produces an output.

The range of operations an ALU can perform depends on its design and complexity. Basic operations that the ALU can perform include addition, subtraction, multiplication, division, and logical operations such as AND, OR, NOT, and XOR.

The ALU uses logic gates such as:
AND and OR gates that perform arithmetic. The ALU also contains a set of registers that hold input values and operation results.

Data flow through the ALU usually follows a certain pattern. Input values are first loaded into registers, then the ALU performs the specified operation on the input values. The result is stored in a register that can be used as input for subsequent operations. The basic functionality of ALUs has evolved over time, with modern ALUs being able to perform more complex operations and support different data types. The ALU is a key component of the central processing unit (CPU) and is responsible for executing instructions within a computer's architecture.

Performing arithmetic operations on binary numbers using an Arithmetic Logic Unit (ALU) typically follows a specific process. Let's take the example of adding two binary numbers: 10101 and 01111.

1. Load the binary numbers into registers: First, the binary numbers are loaded into two registers within the ALU.

2. Perform the addition operation: The ALU performs the addition operation on the two binary numbers using a series of logic gates, such as XOR and AND gates, to produce the sum of the two numbers.
3. Store the result: The result of the addition operation is then stored in a register within the ALU.
4. Output the result: The result is then output to a destination register or memory location.

Here's a step-by-step breakdown of the addition operation:
1 0 1 0 1 (21 in decimal)

0 1 1 1 1 (15 in decimal)

1 0 0 0 0 (36 in decimal)

First, the two binary numbers are loaded into registers within the ALU. The ALU then performs the addition operation on the two binary numbers using a series of logic gates.

Starting from the rightmost bit, the ALU performs the following:

1 + 1 = 0 with carry 1
0 + 1 + carry 1 = 0 with carry 1
1 + 1 + carry 1 = 1 with carry 1
0 + 1 + carry 1 = 0 with carry 1
1 + 0 + carry 1 = 0 with carry 1

The final result is 10000, which is the binary representation of the decimal number 16. However, we also have a carry 1 from the leftmost bit of the addition. This carry is added to the result of the next column, giving a final result of 100000, which is the binary representation of the decimal number 32.

In summary, performing an arithmetic operation on binary numbers using an ALU involves loading the binary numbers into registers, performing the arithmetic operation using a series of logic gates, storing the result in a register, and outputting the result to a destination register or memory location.

Perform the logical AND operation: The ALU performs the logical AND operation on the two binary numbers using a series of logic gates, such as AND gates, to produce the result of the logical operation.

Store the result: The result of the logical operation is then stored in a register within the ALU.

Output the result: The result is then output to a destination register or memory location.

Here's a step-by-step breakdown of the logical AND operation:

1 0 1 0 1
AND
0 1 1 1 1
0 0 1 0 1

First, the two binary numbers are loaded into registers within the ALU. The ALU then performs the logical AND operation on the two binary numbers using a series of AND gates.
Starting from the rightmost bit, the ALU performs the following:

1 AND 1 = 1
0 AND 1 = 0
1 AND 1 = 1
0 AND 1 = 0
1 AND 1 = 1
The final result is 00101, which is the binary representation of the decimal number 5.

In summary, performing a logic operation on binary numbers using an ALU involves loading the binary numbers into registers, performing the logic operation using a series of logic gates, storing the result in a register, and outputting the result to a destination register or memory location.

INTERNAL STRUCTURE OF AN ALU

The internal structure of an ALU may vary depending on the particular design, but there are some common components commonly found in most ALUs.

1. Data register:
The ALU receives data from CPU registers. This is a temporary storage location for data manipulated by the CPU. These registers are usually organized into sets with each set having a different number of bits (8 bits, 16 bits, 32 bits, etc.) to accommodate different types of data.

2. Arithmetic circuit:
This circuit performs arithmetic operations such as addition, subtraction, multiplication and division. The specific components used to perform these operations vary, but typically include adders, subtractors, and multipliers.

3. Logic circuit:
This circuit performs logic operations such as AND, OR, and XOR. Components used to perform these operations include logic gates such as AND gates, OR gates, and XOR gates.

4. Control logic:
This component controls the flow of data through the ALU and ensures that correct operations are performed on the data. Control logic receives input signals from the CPU and uses them to determine what operations to perform and how to route data through the appropriate circuitry.

5. Flag Register:
This register contains status information about the results of operations performed by the ALU. For example, if the result of an addition operation is zero, a flag is set in the flags register indicating that the result is zero. This information can be used by other components within the CPU to determine what to do next.

Overall, the internal structure of the ALU is designed to perform fast and efficient arithmetic and logic operations on data stored in CPU registers. The specific design of the ALU depends on the specific needs of the CPU and the designer's performance goals.

To design a 2- to 2-bit ALU, we need to consider the arithmetic and logic operations that can be performed. This example assumes that the ALU must be able to do the following:

Additive, subtraction, Logical AND disjunction, Logical XOR
ALUs can be designed using combinational logic circuits such as logic gates and adders/subtractors. Your specific design may vary, but here is an example of how to design a 2- or 2-bit ALU.

Two 2-bit data registers (A and B) for storing operands
Two 2-bit output registers (O1 and O2) for storing operation results
Carry-in and carry-out registers that support addition and subtraction

Two 2-bit adders/subtractors that perform addition and subtraction
Logic gates that perform logical AND, OR, and XOR operations .
Here's how each operation can be performed using this ALU:

Addition:
Load the two operands (A and B) into the data registers
Set the carry-in register to 0
Use the adder to add A and B, with the carry-in input set to the value in the carry-in register
Store the result in the output registers (O1 and O2)
If there is a carry-out from the addition operation, set the carry-in register to 1

Subtraction:
Load the two operands (A and B) into the data registers
Set the carry-in register to 1
Use the subtractor to subtract B from A, with the carry-in input set to the value in the carry-in register
Store the result in the output registers (O1 and O2)
If there is a borrow-out from the subtraction operation, set the carry-in register to 0

Logical AND:
Load the two operands (A and B) into the data registers
Use logic gates to perform the logical AND operation on the two operands
Store the result in the output registers (O1 and O2)

Logical OR:
Load the two operands (A and B) into the data registers
Use logic gates to perform the logical OR operation on the two operands
Store the result in the output registers (O1 and O2)

Logical XOR:
Load the two operands (A and B) into the data registers
Use logic gates to perform the logical XOR operation on the two operands
Store the result in the output registers (O1 and O2)
This is just one example of how an ALU of 2-2 bits could be designed. The specific design will depend on the specific requirements of the CPU and the performance goals of the designer.

RESULT
To evaluate the proposed methodology, we designed and implemented a basic ALU using Verilog. The implementation included the selection of basic operations, such as addition, subtraction, AND, and OR, and the use of logic gates to implement these operations. The simulation of the implemented ALU demonstrated the expected behavior of the unit, including the correct output values for the selected operations.

CONCLUSION
This paper presents a methodology for analyzing the function and operation of an ALU that can lead to a deeper understanding of this crucial component of digital circuits. The proposed methodology includes several steps, such as understanding the basic function, analyzing the internal structure and operation, and implementing the ALU using hardware description languages.

The methodology was evaluated through the design and implementation of a basic ALU, which demonstrated the expected behavior and performance of the unit. This methodology can be applied to more complex ALUs, leading to a deeper understanding of their function and operation, and ultimately improving the performance and reliability of digital systems.

REFERENCES

1. Desai, S. S, Gadkari,V. M., Talbar S. N. (2022). Design and Analysis of High-Speed and Energy-Efficient Arithmetic Logic Units for IoT Applications.
2. Al-Khaffaf H. A. , Al-Ali A. H. (2022)A Novel Low-Power and High-Speed Adder Design Using Hybrid Multiplexers for ALUs.
3. Kshirsagar S., Jadhav P.(2022) Design of Energy-Efficient Arithmetic and Logic Units Using Nano-scale CMOS Technology.
4. Manjappa, M., Pitchappa, P., Singh, N., Wang, N., Zheludev, N. I., Lee, C., & Singh, R. (2018). Reconfigurable MEMS Fano metasurfaces with multiple-input–output states for logic operations at terahertz frequencies. *Nature communications*, *9*(1), 4056.
5. Ismail, H, Al-Salihi, H,. Al-Ali A.H (2022) Design of a High-Performance and Low-Power Arithmetic Logic Unit Using Deep Learning Techniques.