External Project Report on Digital Logic Design (EET1211)

[ALU 1-bit]

<u>**Submitted by**</u>

Name 01 : SK SHAKEEL AKHTAR	Reg. No.: 2341001063
Name 02 : ANUSKA BASANTIA	Reg. No.: 2341002058
Name 03 : SAHIB KUMAR BAL	Reg. No.: 2341002006

B. Tech. CSE(AIML) 3rd Semester (Section – 23412C3)

INSTITUTE OF TECHNICAL EDUCATION AND RESEARCH
(FACULTY OF ENGINEERING)
SIKSHA 'O' ANUSANDHAN (DEEMED TO BE UNIVERSITY), BHUBANESWAR, ODISHA

Declaration

We, the undersigned students of B. Tech. of **(AIML)** Department hereby declare that we own the full responsibility for the information, results etc. provided in this PROJECT titled “**(**ALU 1-bit**)**” submitted to **Siksha ‘O’ Anusandhan Deemed to be University, Bhubaneswar** for the partial fulfillment of the subject **Digital Logic Design (EET 1211)**. We have taken care in all respect to honor the intellectual property right and have acknowledged the contribution of others for using them in academic purpose and further declare that in case of any violation of intellectual property right or copyright we, as the candidate(s), will be fully responsible for the same.

SK SHAKEEL AKHTAR
Registration No.: 2341001063

ANUSKA BASANTIA
Registration No.: 2341002058

SAHIB KUMAR BAL
Registration No.: 2341002006

DATE: 20/12/2024

PLACE: ITER, Jagamara Nagar, 751030

Abstract

Design and Implementation of a 1-Bit Arithmetic Logic Unit (ALU)

This project presents the design and implementation of a 1-bit Arithmetic Logic Unit (ALU) capable of performing fundamental arithmetic and logical operations. The ALU is designed to process two 1-bit inputs (A and B) and generate a 2-bit output (O1, O0) based on control signals (C1, C0). The control signals dictate the operation, which can be one of the following:

1. **Addition**
2. **Subtraction**
3. **Logical AND**
4. **Logical OR**

Operation and Output

- The **outputs (O1, O0)** represent:
  - **O1**: Carry (for addition), Borrow (for subtraction), or 0 (for logical operations).
  - **O0**: The result of the operation.

For arithmetic operations, the design incorporates carry and borrow logic to handle bit-level addition and subtraction. Logical operations directly compute the result using basic logic gates.

Design Approach

The implementation utilizes five 2-to-4 line decoders for operation control and output generation:

1. A **control decoder** selects the active operation based on the control signals.
2. Four additional decoders manage specific operations (addition, subtraction, AND, OR) and produce the respective outputs.

This modular design ensures clarity and systematic control signal processing. By using decoders, the architecture simplifies the logic, making it efficient and scalable.

Applications and Educational Significance

This 1-bit ALU serves as a foundational building block for multi-bit ALU designs, commonly used in digital processors and embedded systems. Beyond its practical applications, the project is an excellent educational tool, enabling students and enthusiasts to understand the integration of logical and arithmetic circuits. The design bridges theoretical concepts such as Boolean algebra and combinational logic with practical circuit implementation.

In summary, this project demonstrates the core principles of digital logic design, emphasizing simplicity, modularity, and scalability in constructing arithmetic and logical units for diverse applications.

Contents

1. Introduction

1-Bit Arithmetic Logic Unit (ALU): A Modular Design

An Arithmetic Logic Unit (ALU) is a fundamental component of any digital processor, responsible for executing arithmetic and logical operations that form the core of computational tasks. This project centers on designing a simple yet functional 1-bit ALU capable of performing essential operations, including addition, subtraction, logical AND, and logical OR, on two single-bit inputs (A and B).

The type of operation is determined by two control inputs (C1, C0), allowing the ALU to adapt dynamically to different computational requirements. This versatility makes it an excellent foundational building block for more advanced digital systems.

Output Description

The 1-bit ALU generates a 2-bit output:

- **O1**: Represents the carry (for addition), borrow (for subtraction), or a default value of 0 (for logical operations).
- **O0**: Represents the result of the operation performed.

Design Features

To achieve modularity and clarity, the design employs **five 2-to-4 line decoders**:

1. **Control Decoder**: Interprets the control inputs (C1, C0) to select the desired operation.
2. **Operation Decoders**: Four additional decoders compute outputs for each specific operation—addition, subtraction, logical AND, and logical OR.

This approach not only simplifies the circuit's logical structure but also enhances its scalability for more complex designs.

Educational and Practical Value

This project demonstrates the integration of **combinational logic** and **modular circuit design principles**, providing a practical tool for understanding ALU functionality. The simplicity of the 1-bit ALU makes it an ideal prototype for exploring digital system design.

Furthermore, it serves as an educational platform to learn fundamental concepts like:

- Arithmetic and logical operations in hardware.
- Decoder-based circuit implementation.
- The basics of digital processor design.

Future Applications

The 1-bit ALU's modular nature makes it a scalable building block for constructing multi-bit ALUs used in advanced digital systems, such as:

- Central Processing Units (CPUs).
- Embedded systems.
- Signal processing devices.

This project is a hands-on demonstration of how simple digital circuits form the foundation of modern computing.

2. Problem Statement

I. Explanation of problem and identification of input and output variables.

Explanation of Problem and Identification of Input and Output Variables
The goal of this project is to design a 1-bit Arithmetic Logic Unit (ALU) capable of performing fundamental operations—addition, subtraction, logical AND, and logical OR—on two single-bit inputs (A and B). The selection of the operation is determined by a 2-bit control input (C1, C0). The ALU should produce two output bits (O1 and O0), where:

O1: Represents the carry for addition, borrow for subtraction, or 0 for logical operations.
O0: Represents the result of the selected operation, such as sum, difference, logical AND, or logical OR.
The circuit design should utilize five 2-to-4 line decoders. The first decoder selects the operation based on the control inputs, while the other four decoders produce the corresponding outputs for each operation.

II. Highlighting the constraints.

Control Constraints:

The control input (C1, C0) must uniquely identify one of the four operations:
00 for addition
01 for subtraction
10 for logical AND
11 for logical OR
Logic Implementation Constraints:

The design must use exactly five 2-to-4 line decoders: one for operation selection and four for operation-specific output generation.
All logical and arithmetic operations must be implemented using these decoders, avoiding additional complex components like multiplexers or adders.
Output Constraints:

Outputs must strictly adhere to the operation definitions:
For addition, O1 is the carry and O0 is the sum.
For subtraction, O1 is the borrow and O0 is the difference.
For logical AND/OR, O1 must always be 0, and O0 must represent the logical result.
Input Constraints:

Inputs (A and B) and control signals (C1, C0) are single-bit values.
The circuit must function correctly for all possible combinations of inputs and control signals.
This design provides a modular and scalable solution that serves as a foundation for understanding basic ALU functionality and combinational circuit design.

3. Methodology

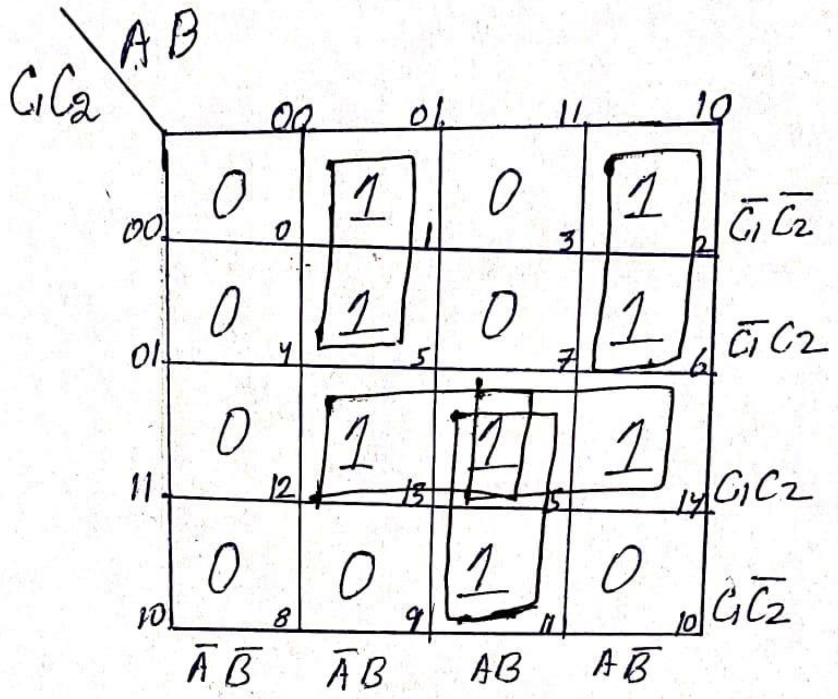
I. Generating the solution to the problem by the use of Truth table/excitation table, K- map and (or) Boolean algebra.

II. Finding out the different digital ICs to be used in the optimized design.

Truth Table

C1	C2	A	B	Operation	O1(Carry/Borrow)	O0(Result)
0	0	0	0	Addition	0	0
0	0	0	1	Addition	0	1
0	0	1	0	Addition	0	1
0	0	1	1	Addition	1	0
0	1	0	0	Subtraction	0	0
0	1	0	1	Subtraction	1	1
0	1	1	0	Subtraction	0	1
0	1	1	1	Subtraction	0	0
1	0	0	0	Logical AND	0	0
1	0	0	1	Logical AND	0	0
1	0	1	0	Logical AND	0	0
1	0	1	1	Logical AND	0	1
1	1	0	0	Logical OR	0	0
1	1	0	1	Logical OR	0	1
1	1	1	0	Logical OR	0	1
1	1	1	1	Logical OR	0	1

K-MAP



$$\boxed{\text{Result, } O_0 = \overline{C_1}\,\overline{A}B + \overline{C_1}A\overline{B} + C_1C_2B + C_1C_2A + C_1AB}$$

ICs USED:

Arithmetic Operations:

1. **Adder/Subtractor:**
   - **74LS83: A 4-bit binary adder with a fast carry feature, ideal for multi-bit arithmetic operations.**
   - **For 1-bit operations: This IC can be adapted for simplicity or extended to handle larger bits by cascading.**
2. **1-bit Addition with Carry:**
   - **74LS86 (XOR Gate): Useful for building half or full adders. Combine it with other gates for complete arithmetic operations like adding a carry input.**

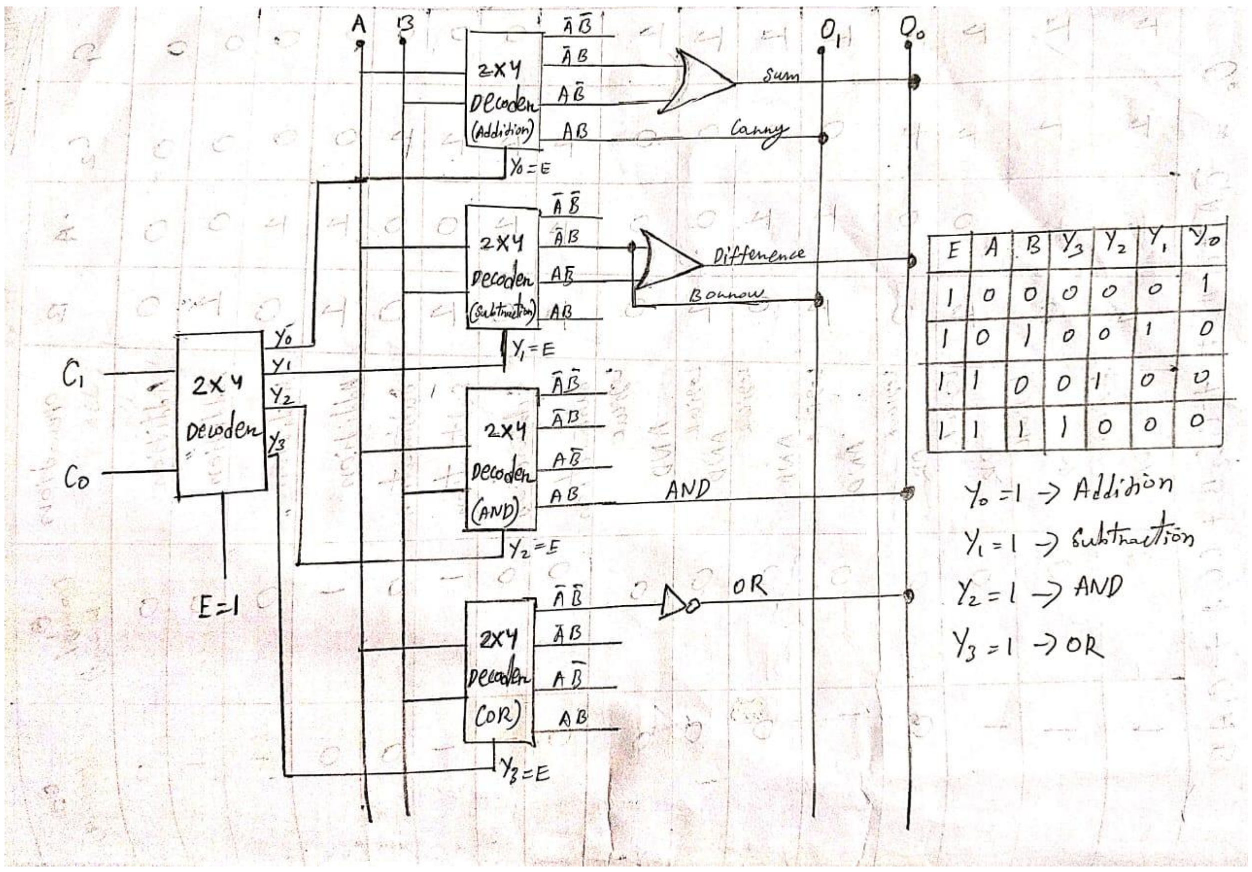
Logical Operations:

1. **AND/OR Gates:**
   - **74LS08: Quad 2-input AND gates for logical conjunctions.**
   - **74LS32: Quad 2-input OR gates for logical disjunctions.**
2. **NOT Gate:**
   - **74LS04: Hex inverter for simple logic inversion operations.**

4. Implementation

I. Drawing the logic diagram using different logic gates.

II. Program



E	A	B	Y_3	Y_2	Y_1	Y_0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

$Y_0 = 1 \rightarrow$ Addition

$Y_1 = 1 \rightarrow$ Subtraction

$Y_2 = 1 \rightarrow$ AND

$Y_3 = 1 \rightarrow$ OR

VHDL CODE

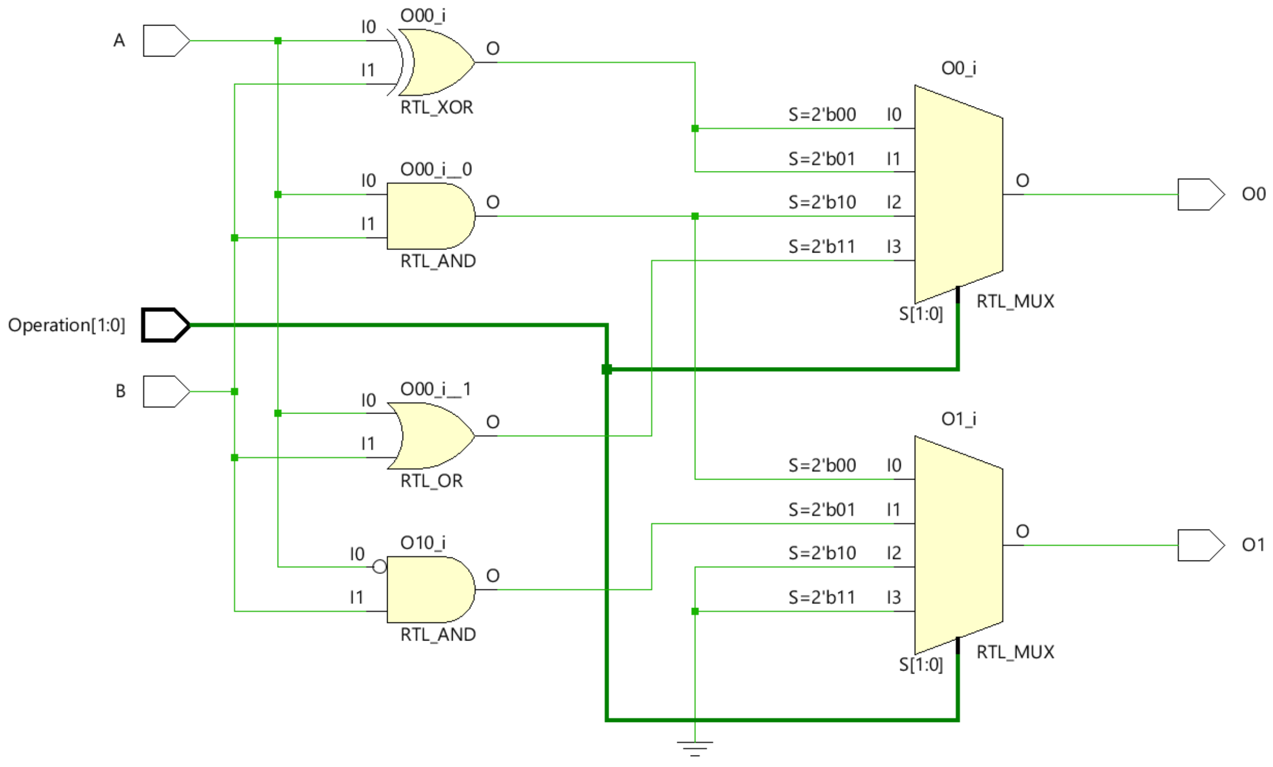
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity ALU_1Bit is
  Port (
     A        : in  STD_LOGIC;
     B        : in  STD_LOGIC;
     Operation : in  STD_LOGIC_VECTOR(1 downto 0);
     O0       : out STD_LOGIC;
     O1       : out STD_LOGIC
  );
end ALU_1Bit;

architecture Behavioral of ALU_1Bit is
begin
  process(A, B, Operation)
  begin
     case Operation is
        -- Addition
        when "00" =>
           O0 <= A XOR B;
           O1 <= A AND B;

        -- Subtraction (A - B)
        when "01" =>
           O0 <= A XOR B;
           O1 <= NOT A AND B;  -- Borrow condition

        -- Logical AND
        when "10" =>
           O0 <= A AND B;
           O1 <= '0';         -- No carry/borrow for AND

        -- Logical OR
        when "11" =>
           O0 <= A OR B;
        O1 <= '0';        -- No carry/borrow for OR

      -- Default case (should not occur)
      when others =>
        O0 <= '0';
        O1 <= '0';
    end case;
  end process;
end Behavioral;
```

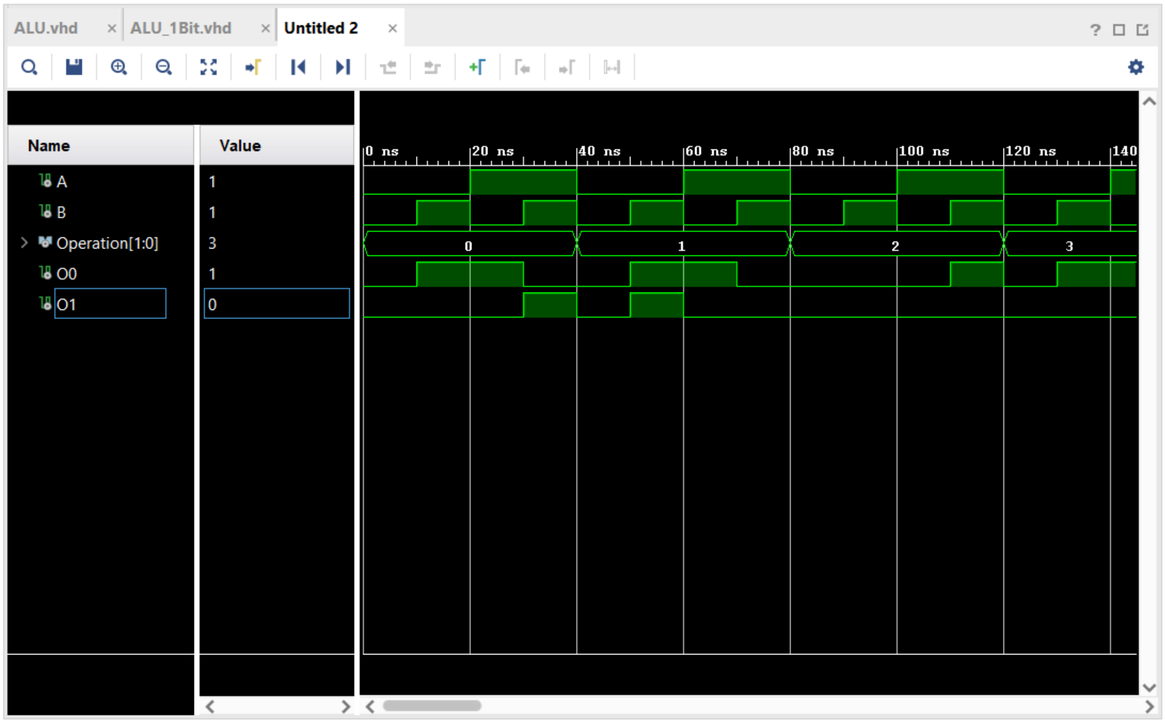
TESTBENCH CODE

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ALU_1Bit_TB is
end ALU_1Bit_TB;

architecture TestBench of ALU_1Bit_TB is
  -- Component declaration
  component ALU_1Bit
    Port(
      A : in  STD_LOGIC;
      B : in  STD_LOGIC;
      Operation : in  STD_LOGIC_VECTOR(1 downto 0);
      O0 : out STD_LOGIC;
      O1 : out STD_LOGIC
    );
  end component;

  -- Signal declarations
  signal A, B : STD_LOGIC := '0';
  signal Operation : STD_LOGIC_VECTOR(1 downto 0) := "00";
  signal O0, O1 : STD_LOGIC;

begin
  -- Instantiate the ALU component
  UUT: ALU_1Bit
    port map(
      A => A,
            B => B,
            Operation => Operation,
            O0 => O0,
            O1 => O1
        );

    -- Stimulus process
    process
    begin
        -- Test Case 1: Addition (Operation = "00")
        A <= '0'; B <= '0'; Operation <= "00"; wait for 10 ns;  -- Expected: O1 = 0, O0 = 0
        A <= '0'; B <= '1'; Operation <= "00"; wait for 10 ns;  -- Expected: O1 = 0, O0 = 1
        A <= '1'; B <= '0'; Operation <= "00"; wait for 10 ns;  -- Expected: O1 = 0, O0 = 1
        A <= '1'; B <= '1'; Operation <= "00"; wait for 10 ns;  -- Expected: O1 = 1, O0 = 0

        -- Test Case 2: Subtraction (Operation = "01")
        A <= '0'; B <= '0'; Operation <= "01"; wait for 10 ns;  -- Expected: O1 = 0, O0 = 0
        A <= '0'; B <= '1'; Operation <= "01"; wait for 10 ns;  -- Expected: O1 = 1, O0 = 1
        A <= '1'; B <= '0'; Operation <= "01"; wait for 10 ns;  -- Expected: O1 = 0, O0 = 1
        A <= '1'; B <= '1'; Operation <= "01"; wait for 10 ns;  -- Expected: O1 = 0, O0 = 0

        -- Test Case 3: Logical AND (Operation = "10")
        A <= '0'; B <= '0'; Operation <= "10"; wait for 10 ns;  -- Expected: O1 = 0, O0 = 0
        A <= '0'; B <= '1'; Operation <= "10"; wait for 10 ns;  -- Expected: O1 = 0, O0 = 0
        A <= '1'; B <= '0'; Operation <= "10"; wait for 10 ns;  -- Expected: O1 = 0, O0 = 0
        A <= '1'; B <= '1'; Operation <= "10"; wait for 10 ns;  -- Expected: O1 = 0, O0 = 1

        -- Test Case 4: Logical OR (Operation = "11")
        A <= '0'; B <= '0'; Operation <= "11"; wait for 10 ns;  -- Expected: O1 = 0, O0 = 0
        A <= '0'; B <= '1'; Operation <= "11"; wait for 10 ns;  -- Expected: O1 = 0, O0 = 1
        A <= '1'; B <= '0'; Operation <= "11"; wait for 10 ns;  -- Expected: O1 = 0, O0 = 1
        A <= '1'; B <= '1'; Operation <= "11"; wait for 10 ns;  -- Expected: O1 = 0, O0 = 1

        -- End Simulation
        wait;
    end process;

end TestBench;
```

5. Results & Interpretation

SCHMATIC DIAGRAM & WAVEFRONT:





Name	Value
A	1
B	1
Operation[1:0]	3
O0	1
O1	0

6. Conclusion

The design and implementation of a 1-bit Arithmetic Logic Unit (ALU) effectively demonstrate the principles of modular circuit design and the use of combinational logic components. By utilizing five 2-to-4 line decoders, the ALU performs four fundamental operations—addition, subtraction, logical AND, and logical OR—on two 1-bit inputs (A and B). The control inputs (C1, C0) allow for seamless operation selection, ensuring flexibility and functionality.

The output of the ALU is clearly defined: O1 represents carry or borrow for arithmetic operations or remains 0 for logical operations, while O0 provides the result of the selected operation. The constraints and modularity of the design make it both an educational tool and a foundational component for larger, more complex ALUs in digital processors.

This project not only highlights the practical applications of decoders in digital logic but also showcases the simplicity and elegance of a well-structured combinational circuit. The design is scalable, making it suitable for further expansion into multi-bit ALUs or integration into larger computational systems. It serves as a stepping stone for exploring more advanced topics in digital system design.

References

M. Morris Mano and M. D. Ciletti, Digital Design: With an Introduction to the Verilog HDL, VHDL, and SystemVerilog, 6th ed., Pearson, 2018.

J. F. Wakerly, Digital Design: Principles and Practices, 4th ed., Pearson, 2005.

A. S. Sedra and K. C. Smith, Microelectronic Circuits, 7th ed., Oxford University Press, 2015.

IEEE Standard for Binary Floating-Point Arithmetic, IEEE Standard 754-2008, IEEE Computer Society, 2008.

R. S. Gaonkar, Fundamentals of Microcontrollers and Applications in Embedded Systems (with the PIC18 Microcontroller Family), 2nd ed., Penram International Publishing, 2013.

H. Taub and D. Schilling, Digital Integrated Electronics: Analysis and Design, McGraw-Hill, 1977.

IEEE Std 91-1984, IEEE Standard Graphic Symbols for Logic Functions, IEEE, 1984.

T. L. Floyd, Digital Fundamentals, 11th ed., Pearson, 2015.

Ensure proper citation of references in the report, adhering to IEEE citation formats.

Appendices

Justification of the Architecture and Digital ICs Used

The architecture of the 1-bit ALU was carefully chosen to ensure modularity, simplicity, and scalability. By utilizing five 2-to-4 line decoders, the design achieves clear separation of control and operation logic, minimizing complexity while maintaining efficiency. Below is the justification for the selected components:

1. 2-to-4 Line Decoders (IC 74LS139)

Functionality:

The 2-to-4 line decoder takes 2 input lines and activates one of its 4 output lines based on the input combination.

One decoder handles the operation selection (control logic), and four additional decoders perform output-specific logic for addition, subtraction, AND, and OR operations.

Why IC 74LS139?

Compact dual 2-to-4 line decoder in a single package.

Active-low outputs make it suitable for integrating with other logic gates.

Fast propagation delay, ensuring efficient signal processing.

2. Basic Logic Gates

AND Gates (IC 7408):

Functionality: Performs logical AND and generates the carry for addition operations.

Why IC 7408?

Quad 2-input AND gates in a single package.

High reliability and wide availability.

OR Gates (IC 7432):

Functionality: Performs logical OR operations.

Why IC 7432?

Quad 2-input OR gates in a single package.

Simplifies logical OR implementation.

XOR Gates (IC 7486):

Functionality: Performs addition and subtraction logic (Sum/Difference).

Why IC 7486?

Quad 2-input XOR gates in a single package.

Essential for arithmetic logic operations.

NOT Gates (IC 7404):

Functionality: Inverts inputs for subtraction (borrow calculation).

Why IC 7404?

Hex inverter in a single package.

Simple and effective for complementing inputs.

3. Multiplexers (Optional)

Functionality: Multiplexers can further optimize the design by selectively passing outputs from the decoders to the final output stage.

Why Multiplexers?

Reduces the need for additional wiring and logic gates.

Enhances modularity and scalability.

Datasheets of ICs Used

IC 74LS139 - Dual 2-to-4 Line Decoder

Datasheet Link: IC 74LS139 Datasheet (Texas Instruments)

IC 7408 - Quad 2-input AND Gates

Datasheet Link: IC 7408 Datasheet (Texas Instruments)

IC 7432 - Quad 2-input OR Gates

Datasheet Link: IC 7432 Datasheet (Texas Instruments)

IC 7486 - Quad 2-input XOR Gates

Datasheet Link: IC 7486 Datasheet (Texas Instruments)

IC 7404 - Hex Inverter (NOT Gates)

Datasheet Link: IC 7404 Datasheet (Texas Instruments)

The selected architecture and components are justified for their efficiency, modularity, and ease of implementation. These digital ICs are widely available and provide the necessary functionality for constructing the 1-bit ALU as per the design requirements.