Project Report

For the Course

EE-221 Digital Logic Design

For

B.E. Electrical Engineering

Sr.	Name	CMS ID
1	Ayeza Faisal	517498
2	M. Shahryar Ameer	520657
3	Seemal Rizwan	505272

Degree- Syndicate: 46-A

Submission Date: 9-12-2025

Instructor: Dr. Shahzad Amin Sheikh
Lab Engineer: Sir Ali Waris
College of Electrical & Mechanical Engineering (CEME),
NUST, Pakistan

CMOS-Based 4-Bit Arithmetic Logic Unit (ALU)

1. Introduction:

An Arithmetic Logic Unit (ALU) is the fundamental computational block of a digital system and is responsible for performing all arithmetic and logical operations within a CPU. In this project, a 4-bit ALU was designed and simulated using Proteus software, utilizing CMOS logic ICs. The ALU supports sixteen different arithmetic and logical operations on two 4-bit binary inputs, A and B, including addition, subtraction, increment, decrement, bitwise operations, complements, loading, and counting.

The design integrates combinational logic, multi-input multiplexing, and dedicated IC-based arithmetic blocks to construct a stable, efficient, and flexible ALU architecture.

2. Objectives:

1. To design and simulate a 4-bit ALU using CMOS logic ICs.
2. To implement arithmetic and logical operations using standard ICs such as XOR, AND, OR, Adders, Counters, and Multiplexers.
3. To understand the internal structure and functionality of various CMOS ICs used in digital systems.
4. To combine multiple sub-circuits into a single integrated ALU design.
5. To verify outputs of all 16 operations using Proteus simulation.

3. List of Components Used:

ICs

- **4030** – Quad 2-Input XOR Gate
- **4081** – Quad 2-Input AND Gate
- **4071** – Quad 2-Input OR Gate
- **7404** – Hex Inverter (NOT Gate)
- **CD4030** – CMOS XOR Gate (additional XOR functionality)
- **LS74283** – 4-Bit Binary Full Adder
- **74157** – Quad 2-to-1 Multiplexer (for OpSel output control)
- **74193** – 4-Bit Synchronous Up/Down Counter
- **7485** – 4-Bit Magnitude Comparator

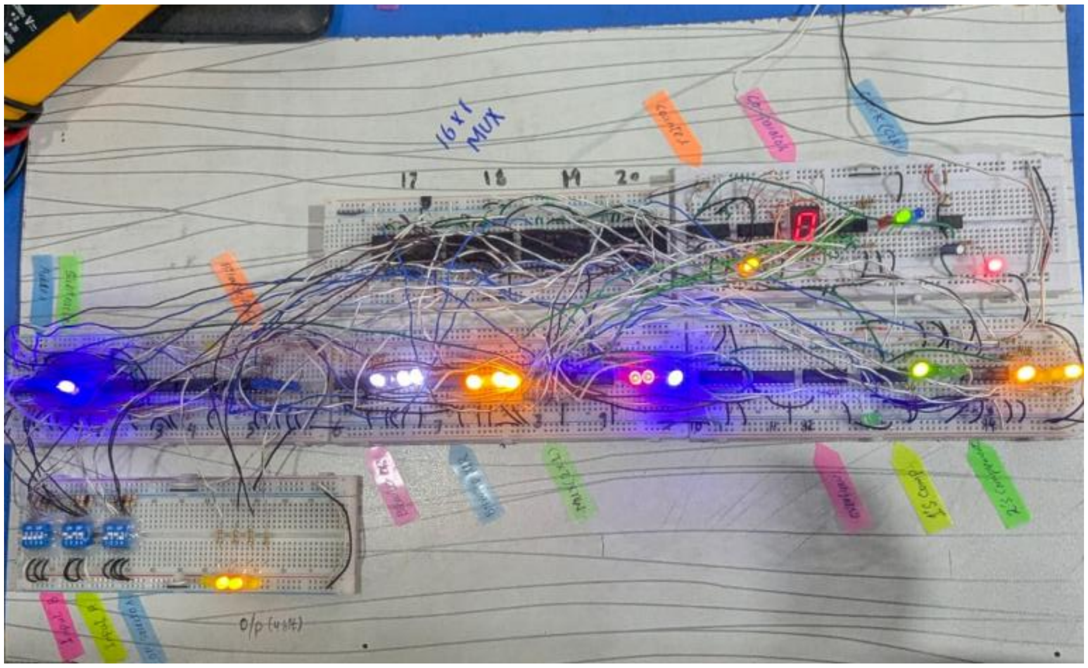
Discrete Components

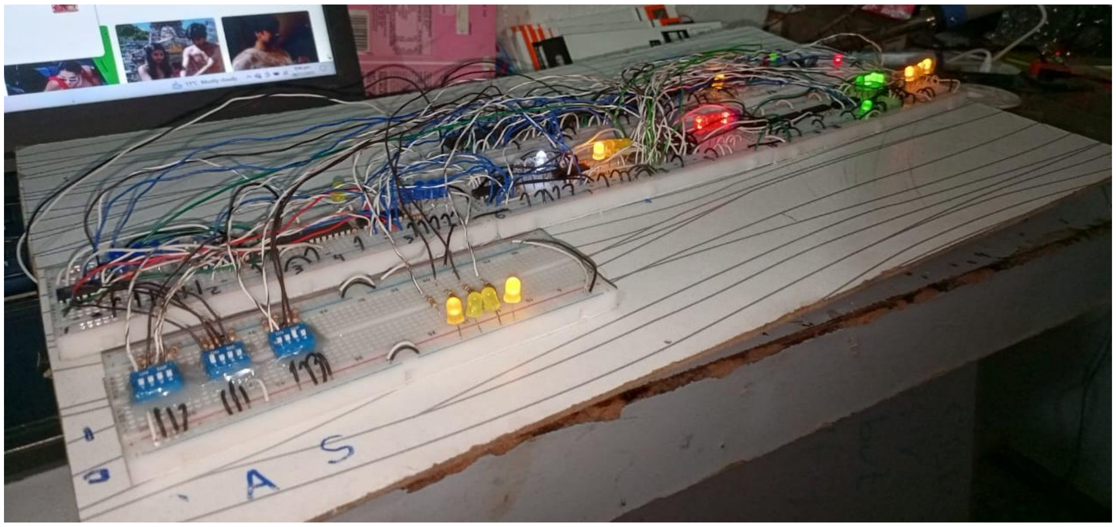
- LEDs for output display
- Resistors (330–470Ω) for LED current limiting
- Jumper wires
- 5V regulated power supply

Software

- **Proteus 8 Professional** – Circuit simulation

Hardware Circuit:



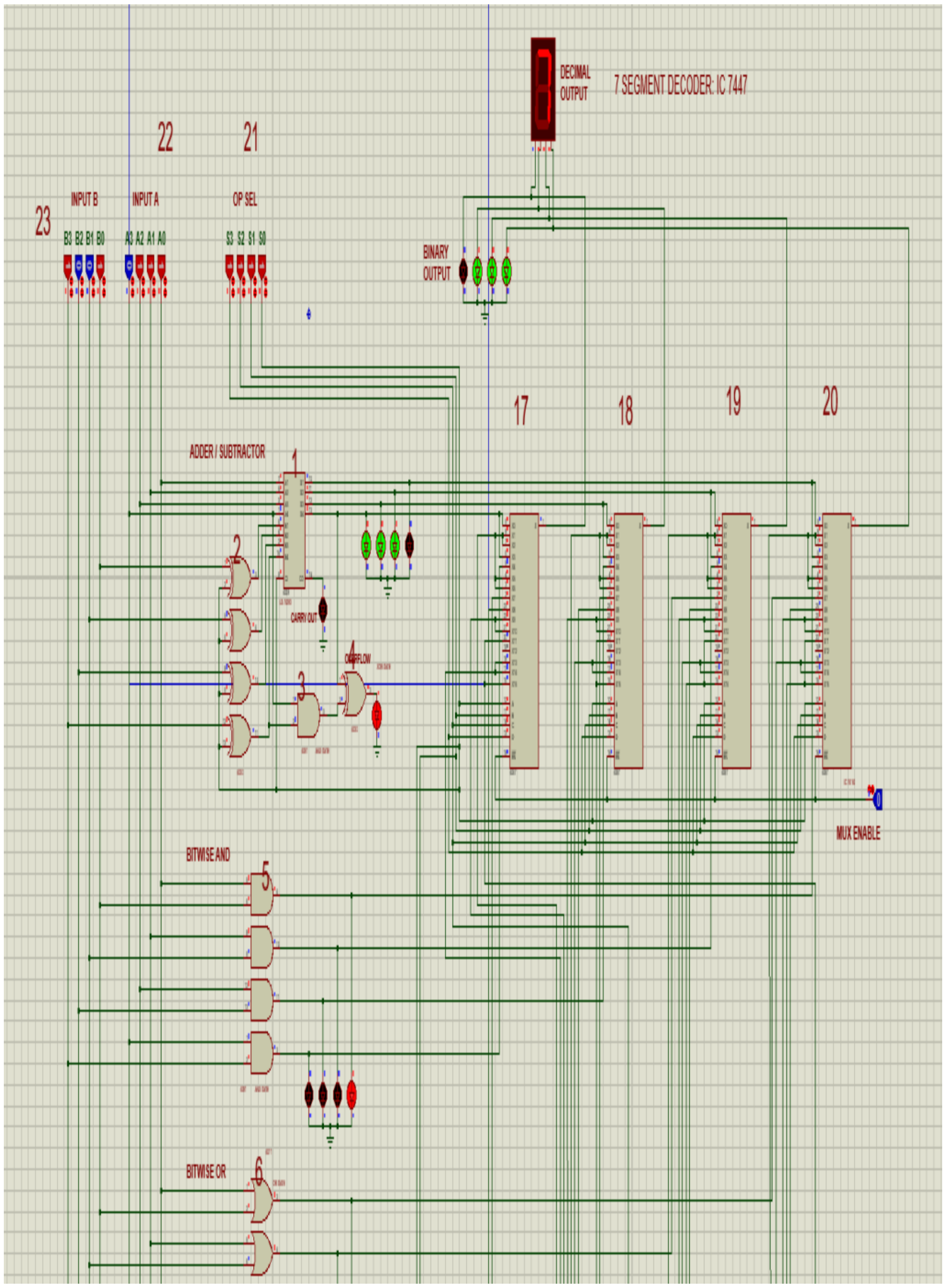


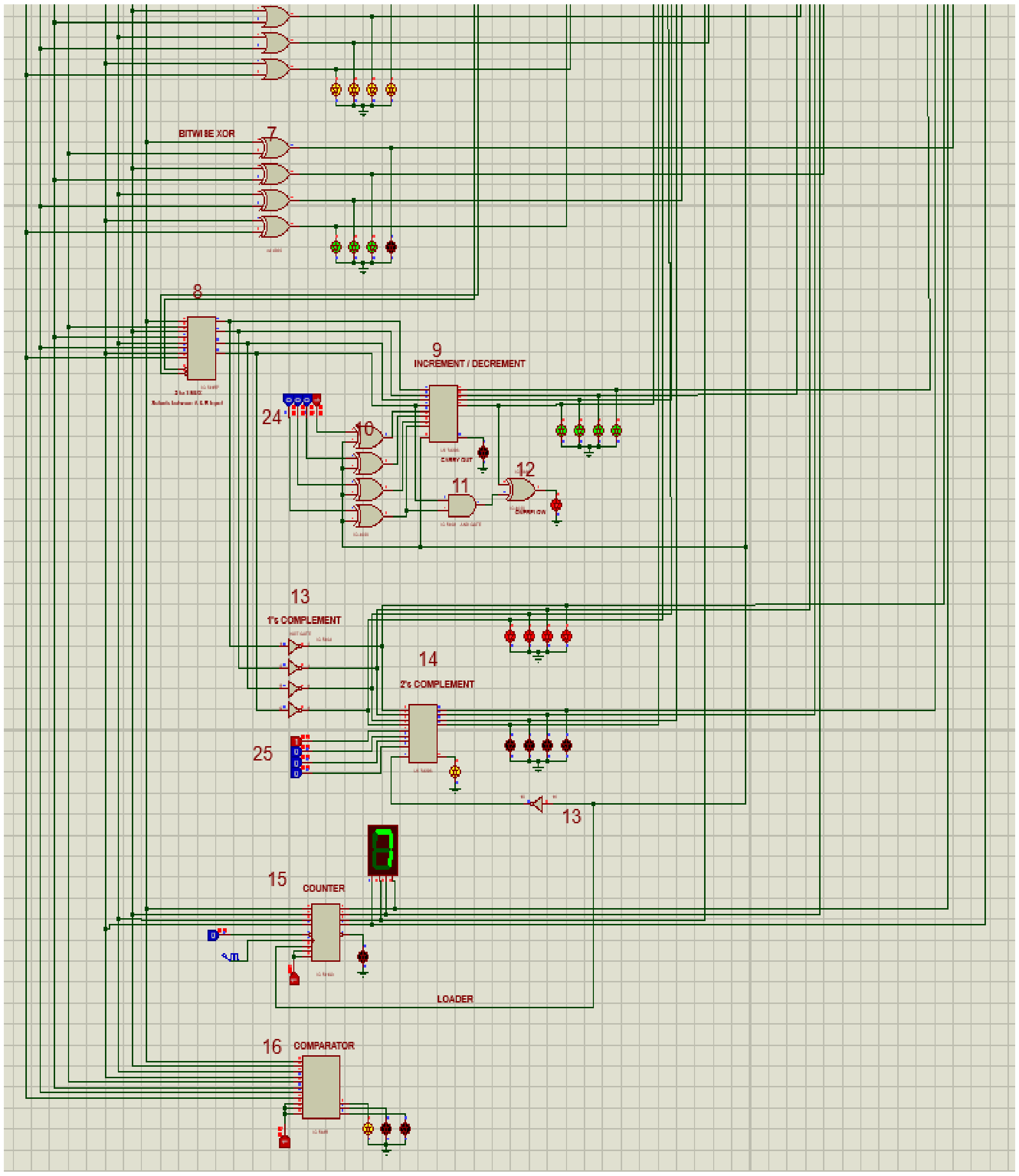
4. Block Diagram Overview:

The ALU consists of the following major functional blocks:

1. **Arithmetic Unit**
   - Addition, subtraction, increment, decrement, 2's complement
   - Implemented using adders, AND gates, OR gates, XOR gates, inverters
2. **Logic Unit**
   - AND, OR, XOR, NOT
   - Implemented using 4081, 4071, 4030, and 7404 ICs
3. **Increment/ Decrement Unit**
   - MUX to select between A & B.
   - XOR gates + adders
4. **1's complement / 2's compelement**:
   - MUX to select between A & B
   - Not gates for 1's complement
   - Adder for 2's complement
5. **Counter/Load Unit**
   - Using 4029 counter IC
   - Supports load and auto-increment counting
6. **Comparator Unit**
   - Using 4585 IC to compare A and B
7. **Operation Selector (OpSel)**
   - A set of 74157 multiplexers selects one out of the 16 outputs
   - Outputs the selected result to the final output lines

6. Proteus Simulation:



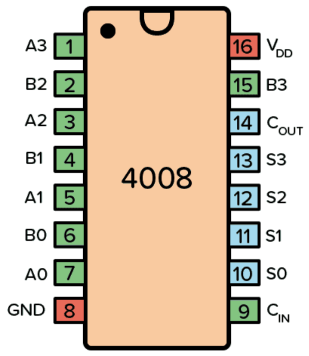

BITWISE XOR
7
8
9
INCREMENT / DECREMENT
24
10
CARRY OUT
11
12
OVERFLOW
13
1's COMPLEMENT
14
2's COMPLEMENT
25
13
15
COUNTER
LOADER
16
COMPARATOR

7. Operation Table (OpSel Codes):

OpSel	Operation
0000	Addition (A + B)
0001	Increment B
0010	Increment A
0011	Subtraction (A – B)
0100	Bitwise AND
0101	Decrement B
0110	Decrement A
0111	Bitwise OR
1000	Bitwise XOR
1001	B’s 1’s Complement
1010	A’s 1’s Complement
1011	Counter
1100	Do Not Care (Reserved)
1101	B’s 2’s Complement
1110	A’s 2’s Complement
1111	Load A

8. Detailed Description of ALU Operations

<u>Operation 0 – Code 0000: Adder (A + B)</u>



Purpose: To add two 4-bit binary numbers, A and B.
How It Works:

- Inputs A and B are applied to the CD4008 4-bit adder.
- Carry-in (Cin) is grounded (0) to perform simple addition.
- The IC generates a 4-bit sum (S0–S3) and an optional carry-out.

Result: The binary sum of A and B is displayed on LEDs.

<u>Operation 1 – Code 0001: B Increment (B + 1)</u>

Purpose: To increase the value of B by one.
How It Works:

- B is connected to the LS74283 adder.
- The second input is fixed at 0001.
- Cin = 0 for normal addition

Result: The output shows B + 1 on the LEDs.

<u>Operation 2 – Code 0010: A Increment (A + 1)</u>

Purpose: To increase A by one.
How It Works:

- A is connected to LS74283.
- The other input is fixed at 0001.
- Cin = 0.

Result: Displays A + 1.

<u>Operation 3 – Code 0011: Subtractor (A – B)</u>

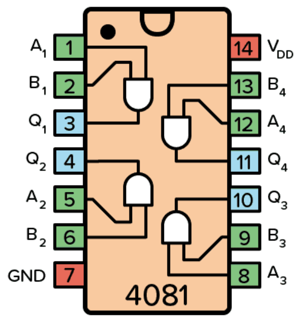
Purpose: To subtract B from A.
How It Works:

- B is inverted using XOR gates (4030) by applying logic 1 to the control input.

- Cin = 1, forming the 2's complement of B (−B).
- LS74283 performs A + (2's complement of B).

Result: Output = A − B.

Operation 4 – Code 0100: Bitwise AND (A AND B)



Purpose: Perform AND operation on each bit of A and B.
How It Works:

- All 4 AND gates of IC 4081 are used.
- Each bit pair (A0,B0)…(A3,B3) is ANDed.

Result: Produces a 4-bit AND output.

Operation 5 – Code 0101: B Decrement (B − 1)

Purpose: Reduce B by 1.
How It Works:

- LS74283 adds B + (1111), since 1111 represents −1 in 2's complement.
- Cin = 0.

Result: Displays B − 1.

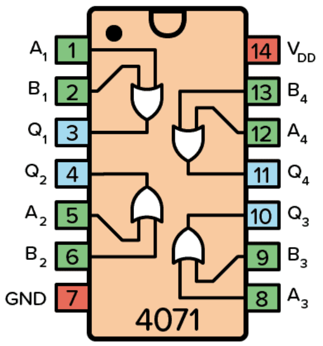
Operation 6 – Code 0110: A Decrement (A − 1)

Purpose: Reduce A by 1.
How It Works:

- LS74283 adds A + 1111 (−1).
- Cin = 0.

Result: Displays A − 1.

Operation 7 – Code 0111: Bitwise OR (A OR B)



Purpose: Perform OR operation on each bit.

How It Works:

- IC 4071 OR gates handle all 4 bit pairs.
- High output appears if either input bit is high.

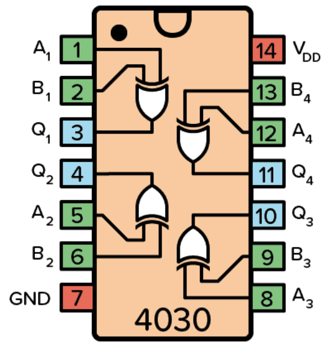
Result: Outputs A OR B.

Operation 8 – Code 1000: Bitwise XOR (A XOR B)

Purpose: XOR each bit of A and B.
How It Works:

- IC 4030 provides XOR operations for all 4 bits.
- Output is high only when input bits differ.

Result: Outputs A XOR B.

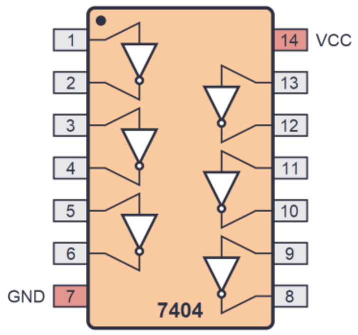


Operation 9 – Code 1001: 1's Complement of B

Purpose: Invert all bits of B.
How It Works:

- Each bit of B passes through XOR gates (4030) with logic '1' applied to invert the bits.

Result: Produces ~B.



Operation 10 – Code 1010: 1's Complement of A

Purpose: Invert A completely.
How It Works:

- A is sent through XOR gates with logic '1'.

Result: Produces ~A.

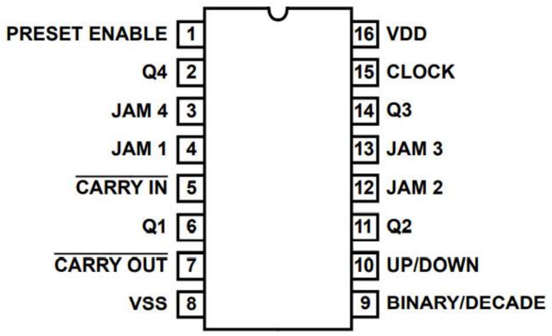
Operation 11 – Code 1011: Counter

Purpose: Count upward starting from a loaded value.

How It Works:

- IC 4029 synchronous counter is used.
- Load value is applied to the parallel inputs.
- Every clock pulse increments the count.

Result: Displays the next count value.



<u>Operation 12 – Code 1100: DC (Don't Care / Reserved)</u>

Purpose: Reserved operation (no function).
How It Works:

- No specific signal is routed.

Result: Output remains undefined or logic low depending on design.

<u>Operation 13 – Code 1101: 2's Complement of B</u>

Purpose: Produce the negative binary form of B.
How It Works:

- B is inverted using XOR gates.
- LS74283 adds 0001 to produce ~B + 1.

Result: 2's complement of B.

<u>Operation 14 – Code 1110: 2's Complement of A</u>

Purpose: Produce the negative binary form of A.
How It Works:

- A is inverted.
- LS74283 adds +1 to produce ~A + 1.

Result: 2's complement of A.

<u>Operation 15 – Code 1111: Loader (B)</u>

Purpose: Load B directly to output.
How It Works:

- Direct B lines are selected via multiplexer input.

Result: Output = B.

<u>Comparator Section</u>

Although not included in the 16 OpSel codes, a **7485 magnitude comparator** is used for:

- A > B
- A = B
- A < B
  Each comparison result is displayed using LEDs.

<u>Final Integration:</u> Connection to 4067 Multiplexer

All sixteen operations generate their own **4-bit output buses**.
These sixteen 4-bit results are then routed into a **4067 multiplexer (16-to-1 MUX)** as follows:

- Each operation output is connected to one data input of the 4967.
- The 4-bit OpSel code selects **exactly one** of the 16 operation results.
- The selected result is forwarded to the **final 4-bit ALU output bus**.

***Purpose of MUX 4067*:**

- It acts as the **Operation Selector**.
- Ensures only one operation's output reaches the final output at any time.
- Provides clean selection, organization, and reduces wiring complexity.

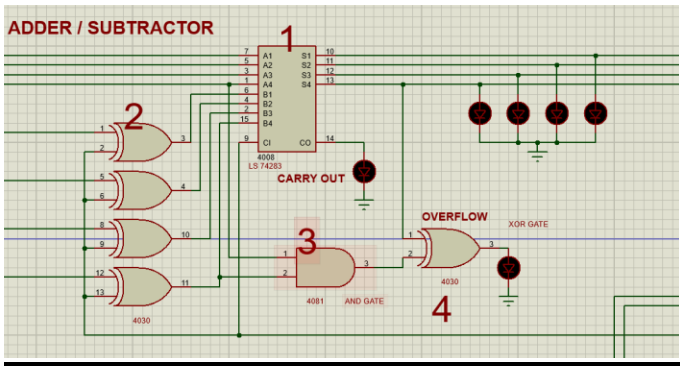
Final Output Flow:
Operation Outputs → MUX 4967 → Final 4-bit ALU Output

9. Bill of Material:

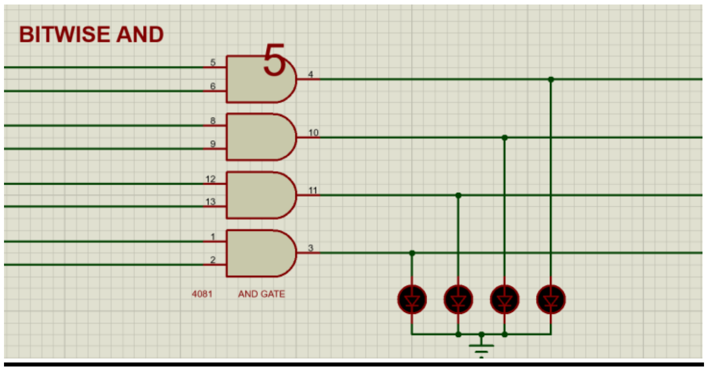
COMPONENTS	QUANTITY	PRICE (PER UNIT)	PRICE
ADDER (4008)	5	170	850
XOR (4030)	6	180	1080
AND (4081)	4	100	400
OR (4071)	4	100	400
MUX (2X1)-4019	2	300	600
NOT (7404)	5	60	300
COUNTER (4029)	3	200	600
COMPARATOR (4585)	3	300	900
MUX (16X1)-4067	7	525	3675
7 -SEGMENT IC (7447)	2	300	600
RESISTORS	50	10	150
LEDS	30	5	150
TOTAL			**9705**

10. Circuit Diagrams:

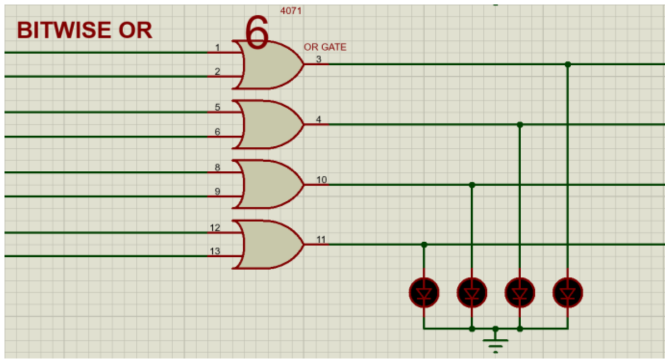
Adder/Subtractor :



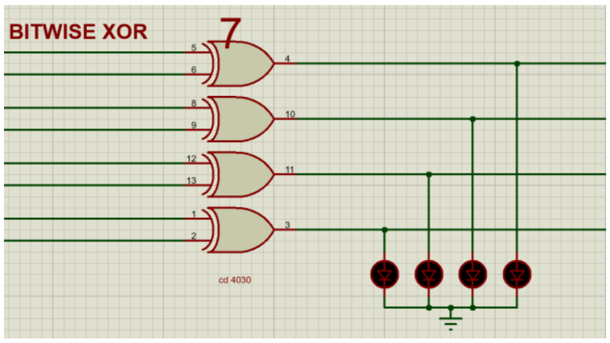
Bitwise AND:



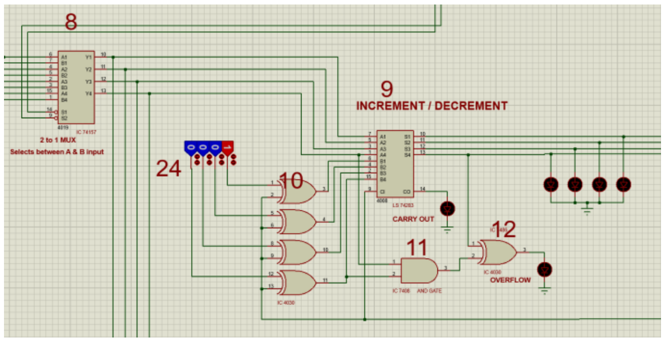
Bitwise OR:



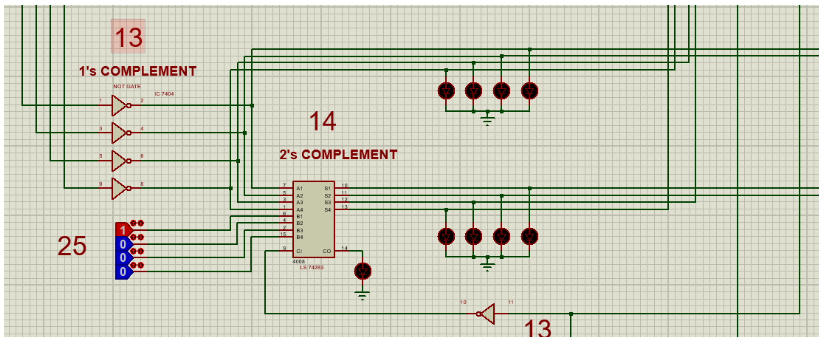
Bitwise XOR:



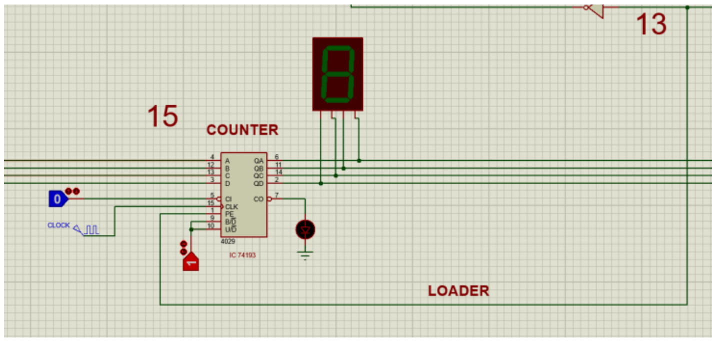
Increment / Decrement:



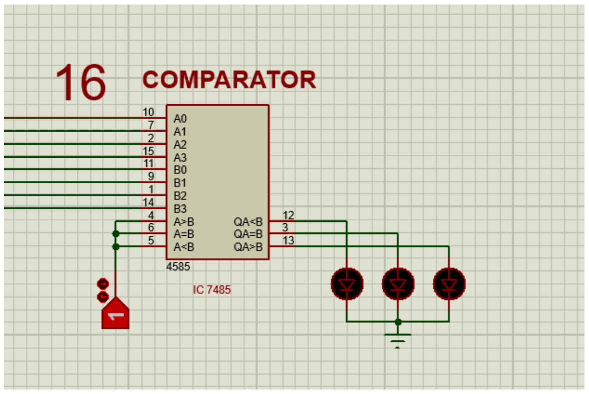
1's / 2's Complement:



Counter:



Comparator:



11. Troubleshooting Guide:

Problem: Output LEDs not turning ON

Causes:

- Incorrect LED polarity
- Missing current-limiting resistors
- No 5V supply

Fix:

- Reverse LED direction
- Add 330–470Ω resistors
- Check Vcc and GND connections

Problem: Incorrect arithmetic results

Causes:

- Wrong wiring on LS74283
- Carry-in pin incorrectly connected
- Bit-order mismatch

Fix:

- Recheck adder pin configuration
- Verify CIN is correctly set (0 for add, 1 for subtract)
- Ensure A0 corresponds to B0, etc.

Problem: Logic operations giving false results

Causes:

- Floating inputs
- Incorrect gate input pairing

Fix:

- Tie unused inputs to GND
- Double-check XOR/AND/OR routing

Problem: Multiplexer not selecting correct output

Causes:

- Incorrect OpSel wiring
- Missing enable signal

Fix:

- Verify select lines S0–S3
- Ensure enable pin of 74157 is low (active)

Problem: Counter not functioning

Causes:

- Clock not connected
- Clear/Load pins incorrect

Fix:

- Provide stable clock
- Tie CLR to high to disable reset

12.Conclusion:

The CMOS-based 4-bit ALU was successfully designed, implemented, and simulated using Proteus. The final system integrates arithmetic, logical, complement, and counter functionalities into a unified digital circuit using ICs such as 4030, 4081, 4071, LS74283, 74193, 7485, and 74157. All sixteen operations performed accurately under test conditions, confirming the correctness of the design.

This project strengthened understanding of CMOS digital logic, hierarchical design, multi-bit data processing, and the functioning of dedicated integrated circuits. The successful implementation demonstrates how an ALU serves as the core computational block in digital processors, capable of executing complex operations through combinational and sequential logic integration. The simulation results validate that the designed ALU functions with reliability, minimal errors, and stable logic behavior.