DEPARTMENT OF ELECTRICAL ENGINEERING

COLLEGE OF E&ME, NUST, RAWALPINDI

Design & Implementation of 4-bit (ALU)

EE – 221 Digital Logic Design

Semester Project Report

DE – 46 – EE – A

Course Instructor: Dr. Shahzad Amin Sheikh
Lab Engineer: Ali Waris

NAME	CMS ID
HAFSA AHMED	506854
MUNEEBA AROOJ	504277
DIYYAT ZAHRA	503852

Submission Deadline: 09-Dec-2025

Project Elements

Operation	Output / Description
Addition	Output C = A+B Status1 = Carry_out Status2 = Overflow
Subtraction	Output C = A-B Status1 = Carry_out Status2 = Overflow
2's Complement of A or B	Output C = 2's Complement of A or B Status = Don't Care
1's Complement of A or B	Output C = 1's Complement of A or B Status = Don't Care
Increment A or B	Output C = A + 1 or B + 1 Status1 = Carry_out Status2 = Overflow
Decrement A or B	Output C = A - 1 or B - 1 Status1 = Carry_out Status2 = Overflow
Bitwise OR	Output C = Bitwise OR of A, B Status = Don't Care
Bitwise AND	Output C = Bitwise AND of A, B Status = Don't Care
Bitwise XOR	Output C = Bitwise XOR of A, B Status = Don't Care
Load A	**Load A into a 4 Bit Counter** Output C = A Status = Don't Care
Count Up	**Count Up from initial value of A** Output C = Count Value Status = Don't Care
Comparison	**Compare A and B** Status1 = 1 if A and B are equal, else zero Status2 = 1 if A > B, else zero Output C = Don't Care

Arithmetic Logic Unit:

Arithmetic Logic Unit (ALU) is the main component of the central processing unit (CPU). It has the ability to perform all processes related to arithmetic and logic operations. Also, binary numbers can accomplish mathematical and bitwise operations. The operands used by the ALU instruct the unit which operations have to be performed according to the input data. When the ALU completes the processing of input, the information is sent to the computer's memory.

Abstract:

This project implements a 4-bit ALU capable of addition, subtraction, 1's and 2's complement, increment, decrement, bitwise AND/OR/XOR, load, count, and comparison using standard TTL ICs (7483, 7485, 74163, 7404, 74157, etc.). A single select line or small control bus chooses the operation; status outputs indicate carry and overflow where relevant.

Aim:

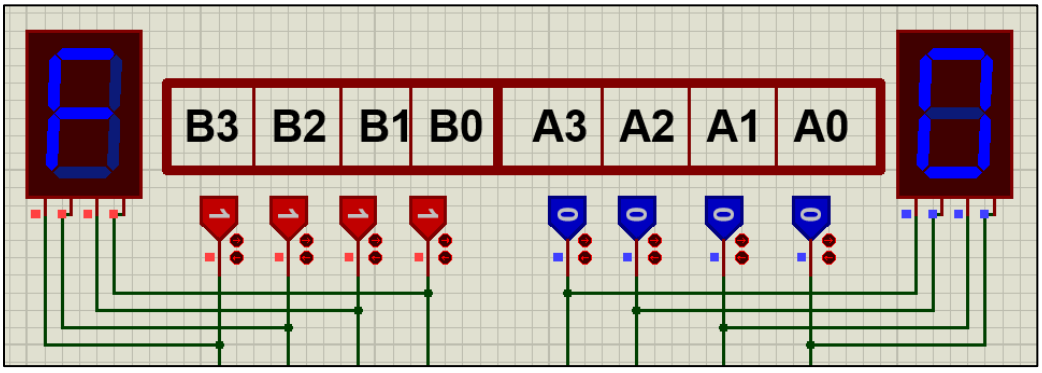
To design and implement a 4-bit ALU using TTL ICs that performs arithmetic and logic operations on two 4-bit inputs A and B, and to demonstrate the correct outputs.

Components:

- 4-bit binary adder IC-**7483**
- 4-bit comparator IC-**7485**
- 4-bit synchronous counter / loadable IC-**74163**
- Multiplexer IC-**74157** (to select A or B)
- Inverter IC-**7404** (for bitwise inversion / 1's complement)
- Bitwise logic gates / ICs for AND/OR/XOR are 7408, 7432, 7486 respectively
- Also XOR gates for overflow detection (7486).
- LEDs for outputs as well as for carry, overflow, equality, greater….
- Breadboard, wires, power supply (+5V) & resistors (optional)

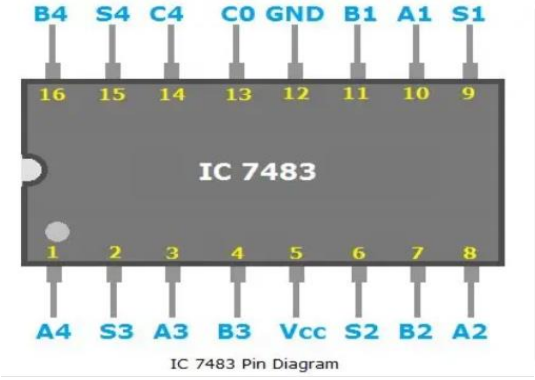
Project Elements Implementation & Explanation

First we set the two 4-bit inputs A and B for this whole project.

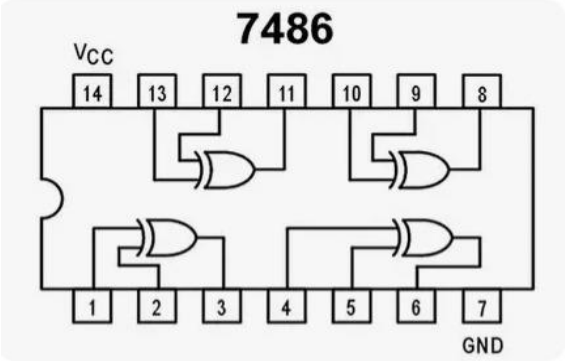


1) Addition and Subtraction

First we used a 7483 adder then used XOR (7486) on B and a select line to switch between addition and subtraction.
→ Select line 0 is for **Addition**
→ Select line 1 is for **Subtraction**



IC 7483 Pin Diagram



Explanation:

This 7483 chip will add two 4-bit numbers **A** and **B** and gives sum and a carry_out. To subtract (A − B), we set the 4-bit **inverted B** into the adder and set the adder's carry-in = 1 which basically depends on the select line, that produces A + (¬B) + 1 = A − B (2's complement subtraction).
We used XOR gates on each bit of B with the operation select line S_0. When **S_0=0**, XOR output = B (so adder adds). When **S_0=1**, XOR output = ¬B (so adder subtracts after carry-in=1). Also we use one select line to set carry-in = SUB (0 for add, 1 for subtract).

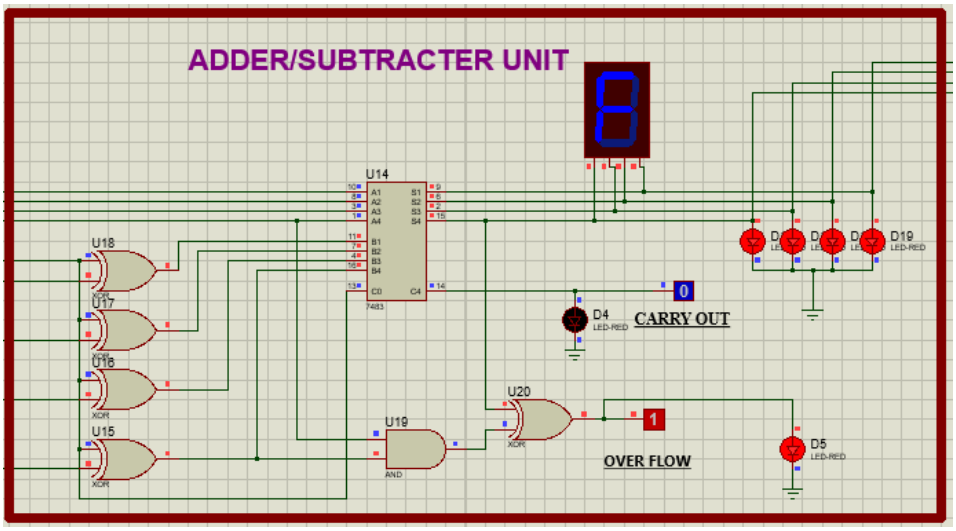
Now it will display the adder result on LEDs, also we've another LED for adder's carry_out.

Overflow:

- Overflow occurs when carry into MSB ≠ carry out of MSB. You implemented this by XORing the carry into MSB and the carry out of MSB; the XOR result = overflow.

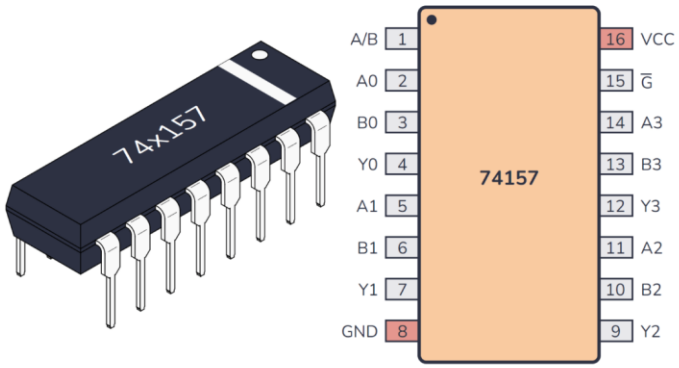
Testing:

- For addition try A=3 (0011), B=5 (0101) will gives result 8 (1000), carry 0.
- For subtraction try A=5, B=3 will gives result 2. Try A=2, B=5 this will produce two's complement negative result.



2) 1's & 2's Complement of A or B

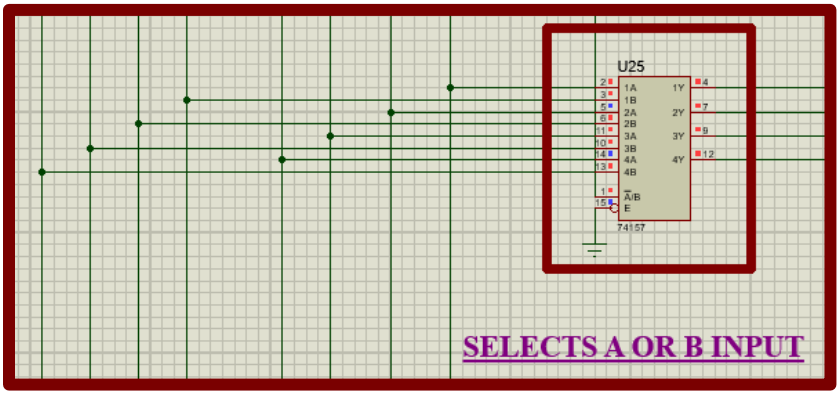
We used 74157 to select A or B. It takes two 4-bit words as inputs and lets you choose which you want to send to the output using select line. Then again7404 to invert bits for 1's complement and use 7483 for 2's complement.



74157 Quad 2-1 Line Data Multiplexers

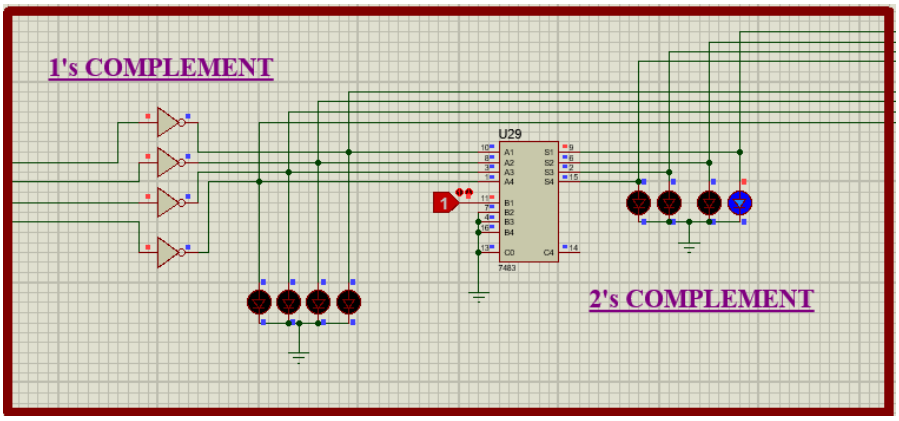
Explanation:

In this operation we use a multiplexer (74157) to choose which operand to operate on. The input (A bar/B) at Pin 1 will decide which input is connected to the output. If select line (S_1) = 0 use A and if select line (S_1) = 1 use B.



For **1's complement** we pass the selected 4 bits through the inverter (7404). That will flips/invert each bit. This is the 1's complement.
For **2's complement** we simply take the output of 1's complement (means the inverted output of a 4 bit number) and then add 1 **(0001)** by using 7483 adder chip. This is the 2's complement.

Finally we showed the results on LEDs.

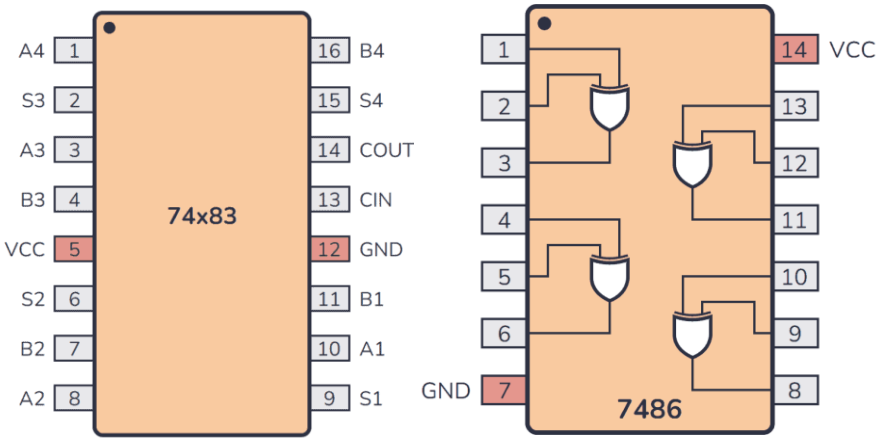


Testing:

- Pick a number A= (0101) it's 1's complement would be (1010) & 2's complement will be 1011 (which is −5 in 2's complement).

3) Increment and Decrement

We used 7483 and display overflow using XOR gate which is 7486.



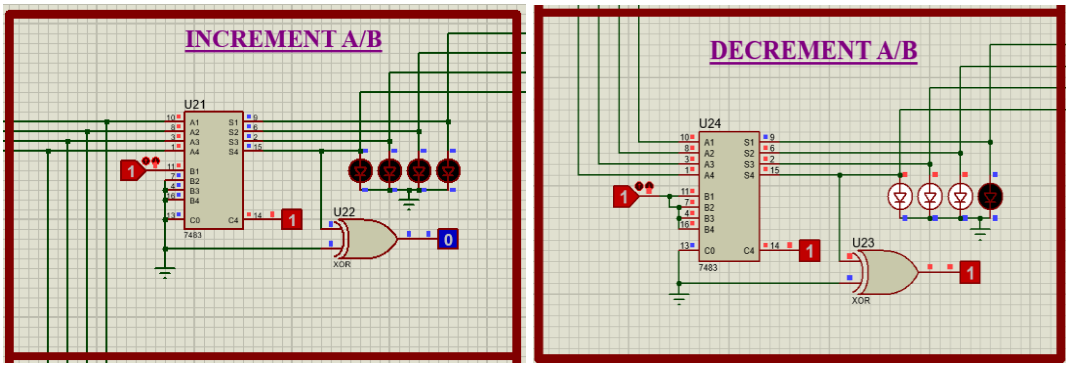
Explanation:

To increment, set A and add with 0001 using the 7483. Once again, the multiplexer (74157) is used to decide between inputs A and B. (The input (A bar/B) at Pin 1 will decide which input is connected to the output. If select line = 0 it will A and if select line = 1 it will B.)
To decrement, we add 1's complement of 0001 and add it with the input from the 74157 multiplexer. Also we use XOR to show overflow if increment causes overflow past 4 bits.

In the end it will display the incremented/decremented output using LEDs.

Testing

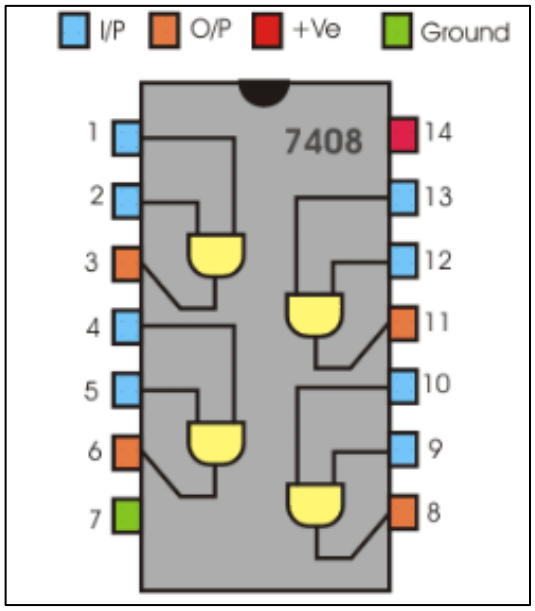
- A=0111, increment will be 1000. If A=1111, increment will be 0000 with carry (overflow), indicate overflow LED.



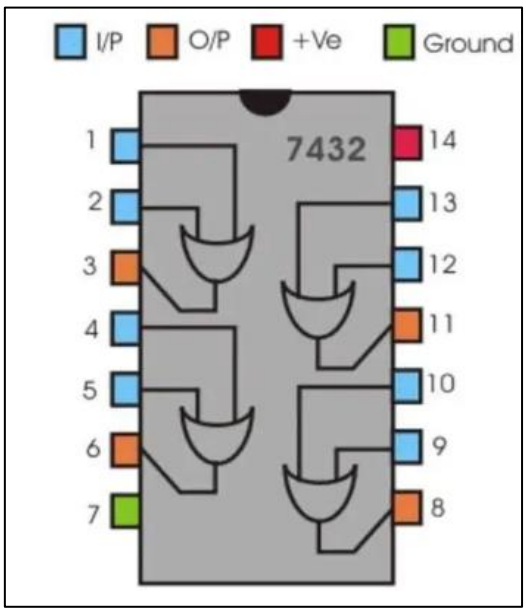
4) Bitwise AND, OR, XOR

Standard logic gate chips (**7408, 7432, and 7486**) for these operations.

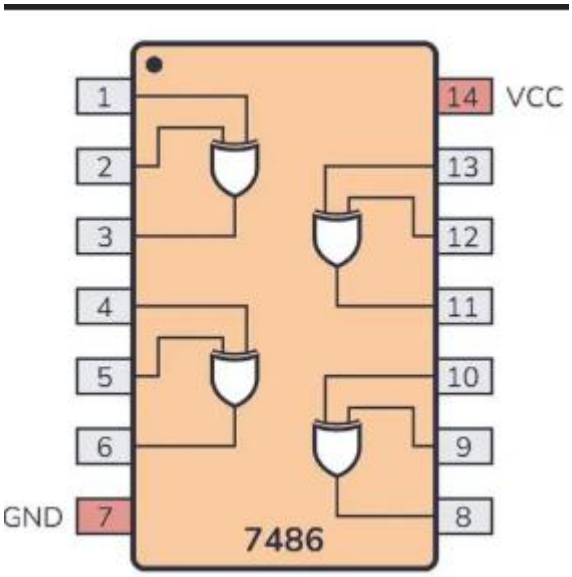
AND: **IC-7408**



OR: **IC-7432**



XOR: **IC-7486**

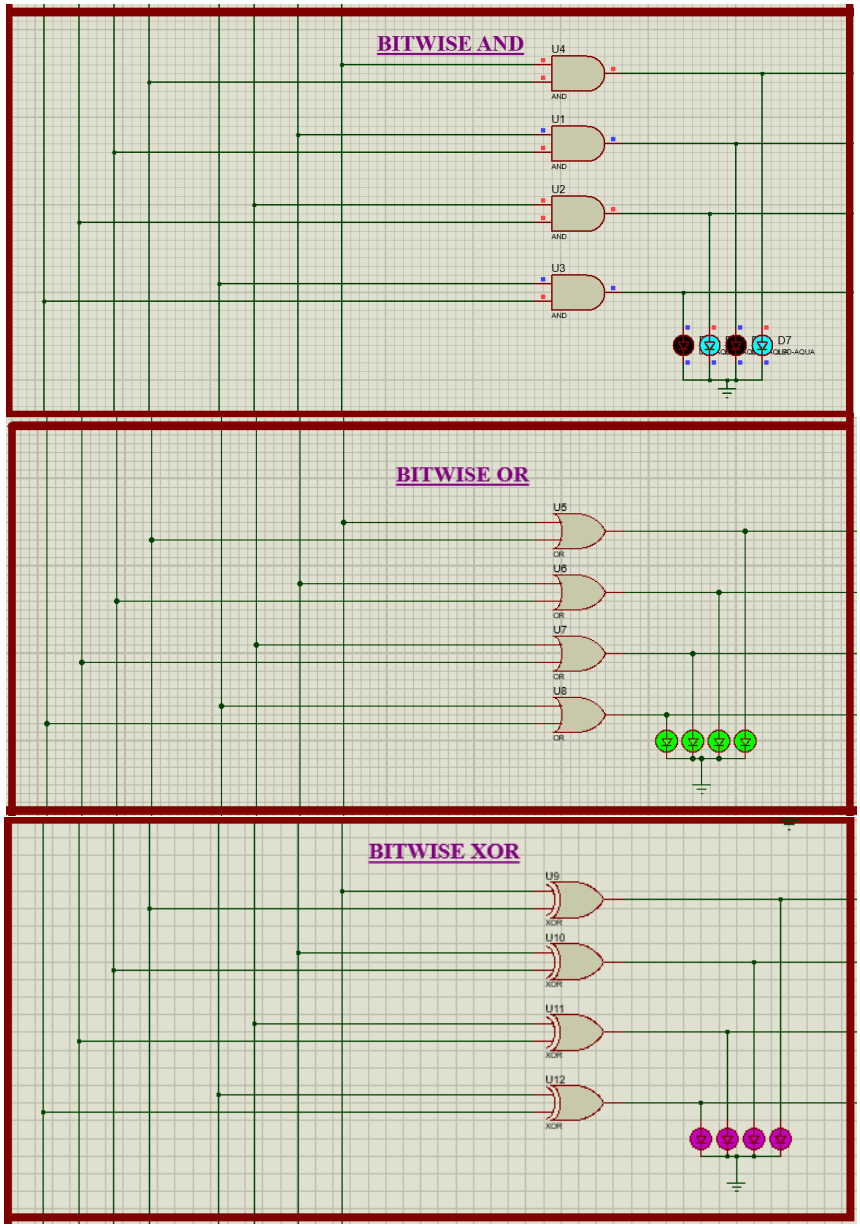


Explanation:

Each AND, XOR, and OR gate will receive both input A and B's respective bit. One gate receiving A0 and B0. 2nd gate receiving A1 and B1 and so on respectively till all bits are fed into a gate. The outputs will show on LEDs. Each output bit is fed into 1 of 4 (16 x 1) multiplexers. The 16 x 1 multiplexer shows the output depending on the select lines combination selected.

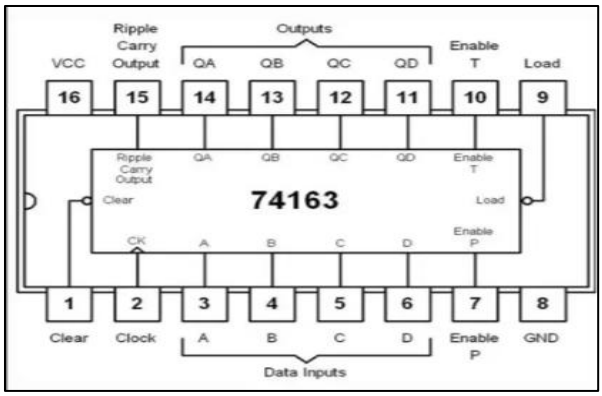
Testing:

- A=1010, B=1100, the AND = 1000, OR = 1110, XOR = 0110.



5) Load and Counter (74163)

A **74163** IC counter chip is used to load A, and to count from **0 -15 (0000-1111)**.



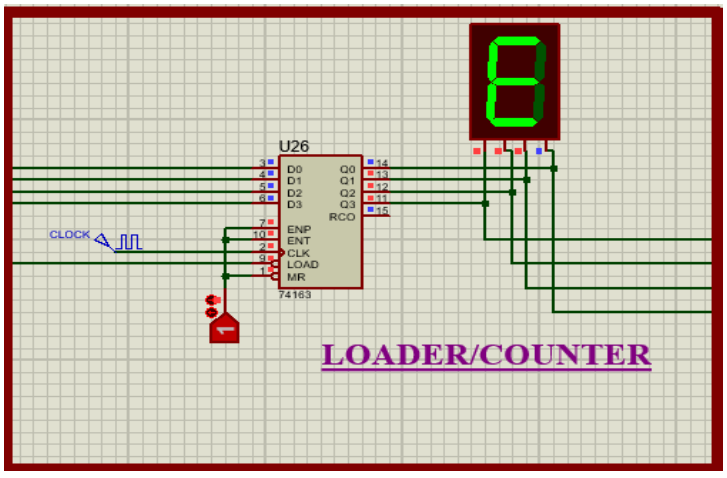
Explanation:

1. **Load:**

   Connect A [3:0] to the parallel load inputs of 74163. Assert the load control to put A into the counter output instantly. This lets you **"load A into the counter"**

2. **Count up:**

   Enable the count input and provide clock pulses; the 74163 increments on each clock. The counter output shows the current count value. You can display the counter bits on LEDs or send them to other parts of the ALU as needed.

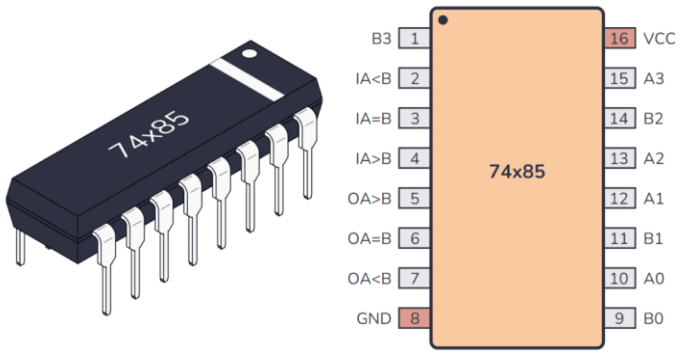


Testing:

- Load A = 0101, then apply clock using oscilloscope and you'll see 0110, 0111, etc.

6) Comparison (7485)

We use the 7485 comparator chip.

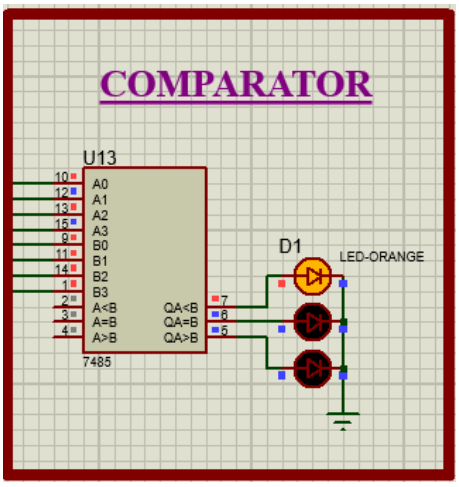


Explanation:

1. 7485 compares two 4-bit numbers and provides outputs which shows:

    - **A = B**
    - **A > B**
    - **A < B**

2. Use those outputs directly as status indicators or LEDs in your circuit.
3. If required, connect cascading inputs for wider comparisons, but for a single 4-bit design the 7485 outputs suffice.



Testing:

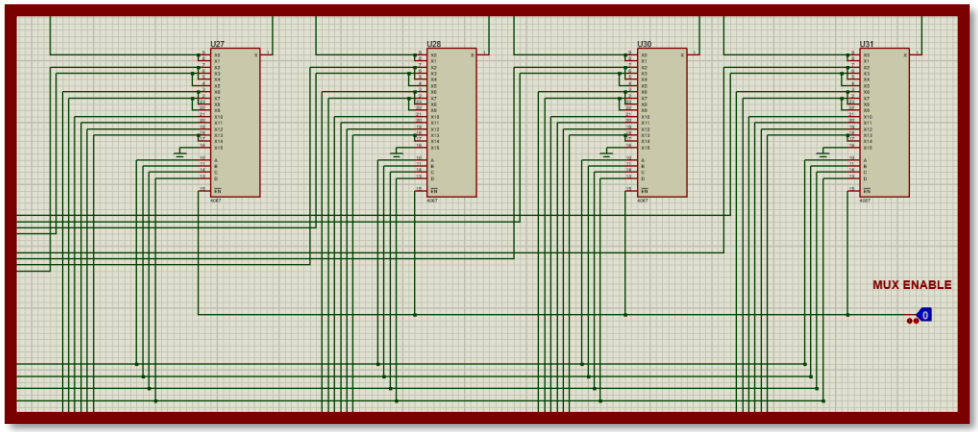
- A=0100, B=0100 equality LED on. A=1000, B=0111 means A>B so this LED will on.

OP SEL:

Four **16-to-1 Multiplexers** were used as Operation Selectors. All outputs from these twelve operations were input into these four MUXs. And the four select lines of each MUX were selected as the OP SEL. Also, the output of each MUX was drawn as the final output of the circuit. The operation table for OpSel is as under:

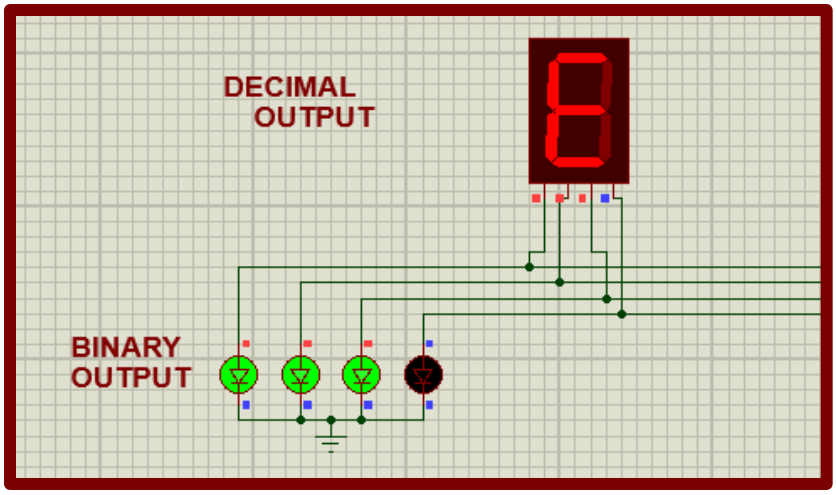
S.no	OP SEL	OPERATIONS
0	0000	Addition
1	0001	Subtraction
2	0010	1's Complement of B
3	0011	2's Complement of B
4	0100	1's Complement of A
5	0101	2's Complement of A
6	0110	B Increment
7	0111	B Decrement
8	1000	A Increment
9	1001	A Decrement
10	1010	Bitwise OR
11	1011	Bitwise AND
12	1100	Bitwise XOR
13	1101	Load A
14	1110	Counter
15	1111	**DON'T CARE**

4 (16 x 1 MUXs) to show 4-bit Output:



1. Use select line inputs (**high and low inputs**) to pick the operation to perform.
2. These inputs are fed to the **16 x 1 multiplexer** and pick and chose which output to display.
3. Ensure proper power **(+5V and GND)**. Provide each IC with its grounds at its specific pin and provide **Vcc** to power the chip.
4. Being mindful of all the **correct bits entering the correct multiplexer**, means the multiplexer with the MSB from the adder/ subtracter must also receive the MSB of the bitwise operators, the increment/ decrement and the 1's and 2's compliment.
5. Inputs should all be **controlled from 1 place**. Those inputs should then be supplied via jumper wires to other circuits in the ALU.

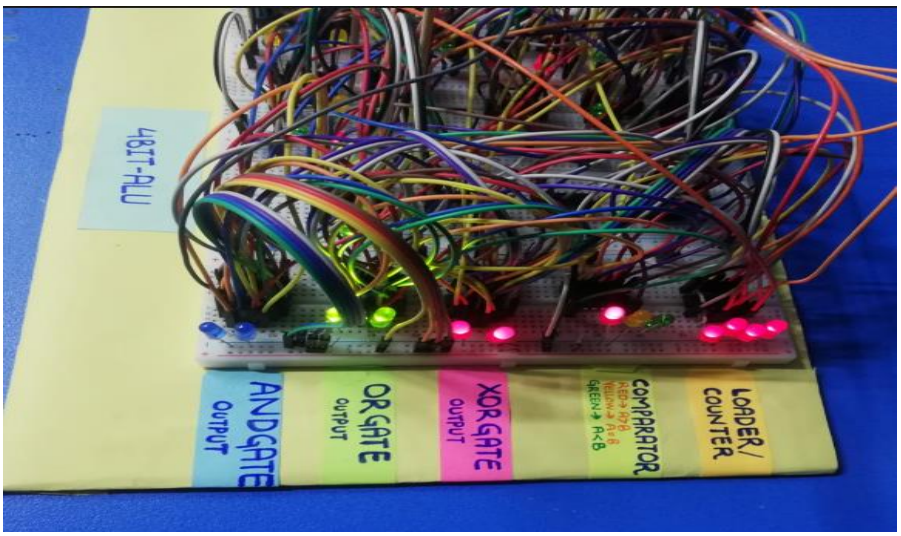
FINAL OUTPUT:



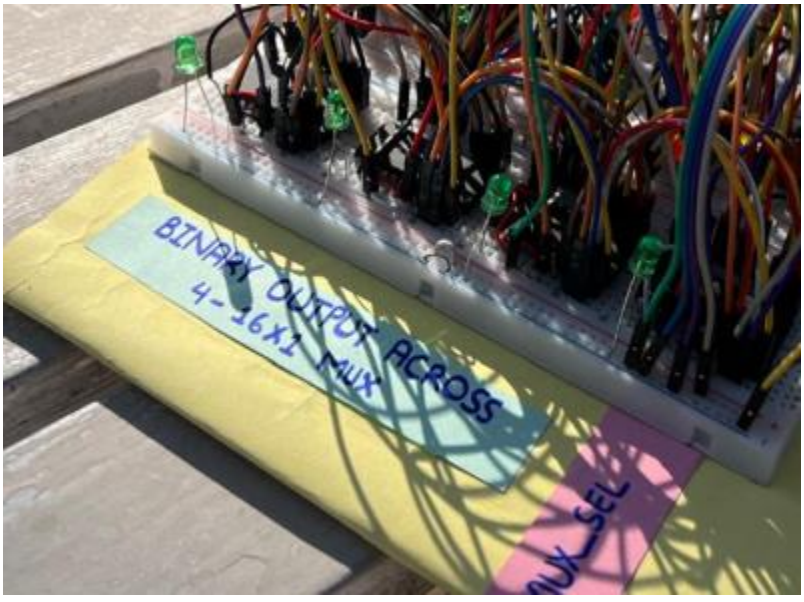
Testing Procedure:

1. Power up the circuit and check supply rails with multimeter.
2. **Test each IC separately** for example feed test patterns to 7483 and verify sum.
3. **Test adder/subtracter mode** verify add when **S_0=0** and subtract when **S_0=1**.
4. **Test complements of A & B,** set A and B using select line and verify 1's and 2's complement outputs.
5. **Test increment/decrement**: with edge cases (like 1111).
6. **Test bitwise gates** with varied A/B.
7. **Test load and count** first load A, then pulse clock.
8. **Test comparator:** with equal, greater and less cases of two 4 bit numbers.
9. Record results in table form for your report.

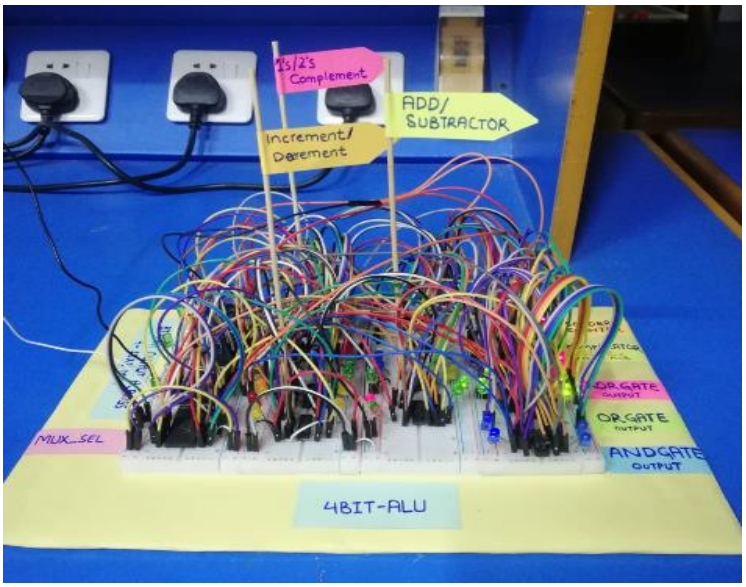
Hardware Implementation:



1. Blue LEDs will show the AND Gate Output
2. Green LEDs will show the OR Gate Output
3. Red LEDs will show the XOR Gate Outputs
4. Now these three different color LEDs are for Comparator Output
    a. Red LED → **A > B**
    b. Yellow LED → **A = B**
    c. Green LED → **A < B**
5. Now these four Red LEDs will show the counter or also load the A input
6. These 4 green LEDs showing the final 4-bit output across the 4 (16x1) MUXs



Final Circuit:



Challenges & How We Solved Them

Challenge 1: Managing multiple operations with limited hardware
With many arithmetic and logic functions, routing outputs without conflicts was complex.
Solution: Designed a centralized OP-SEL control bus and used 4× (16:1) multiplexers to cleanly select the final 4-bit output, similar to a real ALU datapath.

Challenge 2: Implementing subtraction without a separate subtractor
Adding extra hardware would increase complexity and wiring.
Solution: Used XOR gates on B and controlled the carry-in of the 7483 adder to perform 2's complement subtraction using a single adder.

Challenge 3: Correct overflow detection in arithmetic operations
Overflow behavior was confusing during early testing.
Solution: Detected overflow by XORing the carry into and carry out of the MSB, ensuring accurate signed arithmetic indication.

Challenge 4: Signal integrity and wiring complexity on breadboard
Dense wiring made debugging difficult and increased error chances.

Solution: Modularized the design (adder block, logic block, counter, comparator) and tested each IC independently before full integration**.**

Conclusion:

- The 4-bit ALU successfully performs arithmetic and logic operations using **TTL chips**.
- Using XOR on B and carry-in switching allowed a single adder chip to both add and subtract.
- 1's and 2's complement operations work using **7404 (NOT) + 7483(ADDER)**.
- The comparator and counter chips provided reliable comparison and loading/counting features.
- Overflow and carry status correctly indicate arithmetic limits.