ECEN 248 - Lab Report

Lab Number:	4
Lab Title:	Simple Arithmetic Logic Unit
Section Number:	511
Student's Name:	Anahita Kaur
Student's UIN:	235004385
Date:	October 8th, 2025
TA:	Grant Mayfield

Objectives:

This lab assignment focuses on designing, implementing, and testing a simple 4-bit Arithmetic Logic Unit (ALU) capable of performing elementary computations such as addition, subtraction, and bitwise AND operations. The main goals are to understand Two's Complement arithmetic, learn about multiplexers, and integrate multiple functional units into a cohesive ALU design. This lab demonstrates how complex digital systems are built by combining simpler building blocks.

Design:

The ALU design integrates a 4-bit bitwise AND unit, an addition/subtraction unit based on a 4-bit ripple carry adder (SN74HC283E) with XOR gates for two's complement handling, and a 4-bit 2:1 multiplexer (SN74CT257N) for operation selection. Control signals c_0 (selects AND vs. arithmetic) and c_1 (selects add vs. subtract) route the appropriate output. Overflow detection is implemented using sign bits from inputs and result.

Table 1: ALU Control Signals and Operations

$\mathbf{c_0}$	$\mathbf{c_1}$	Operation	Circuit Action
0	X	AND	MUX selects AND unit output.
1	0	Addition ($A + B$)	MUX selects ± unit output; $c_1 = 0$ (CI=0) performs addition.
1	1	Subtraction ($A - B$)	MUX selects ± unit output; $c_1 = 1$ (CI=1) performs subtraction.

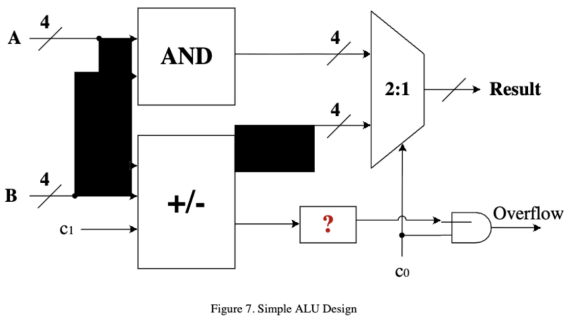

Figure 1: Complete ALU Gate-Level Schematic

Implementation:

Experiment Part 1: Addition/Subtraction Unit

This was built using the SN74HC283E 4-bit adder and a 74LS86 XOR gate chip. The control signal c_1 was wired simultaneously to the four XOR inputs (to invert B for 1's complement) and to the adder's Cin (to do the 2's complement). This allowed the signal c_1 to

control the addition ($c_1 = 0$) and subtraction ($c_1 = 1$). Inputs A and B were done with wires varying between power and ground, visually displaying the output on the LED's.

Experiment Part 2: 4-bit 2:1 MUX Testing

The SN74CT257N multiplexer, which is the ALU's selector switch, was also tested. The output enable pin ($\overline{\text{OE}}$) was permanently grounded, keeping the outputs active. By checking the input patterns with the multimeter on the MUX's A and B banks, we checked that the MUX was working properly.

Experiment Part 3: Complete ALU Integration

The final ALU was completed by connecting the outputs of the AND unit (74LS08) to the MUX's A inputs and the outputs of the addition/subtraction unit (Part 1) to the MUX's B inputs. The control signal $\mathbf{c_0}$ was connected to the MUX select line, routing either the logic (AND) or arithmetic ($\pm$) to the LEDs. The ALU was tested by going through all combinations of c_0 and c_1 across different inputs and demonstrated to the TA.

Results:

Experiment Part 1: Addition/Subtraction Unit Results

The addition/subtraction unit operated correctly for all test cases. When the Sub control signal was low (0), the circuit performed addition. When Sub was high (1), the XOR gates inverted the B input and the carry-in of 1 completed the two's complement, resulting in proper subtraction.

Experiment Part 2: MUX Testing Results

The SN74CT257N multiplexer correctly selected between the A and B input banks based on the select signal. All four bits switched simultaneously and correctly when toggling the select line.

Experiment Part 3: Complete ALU Results

The integrated ALU successfully performed all three operations based on the control signals c_0 and c_1. The operation table created in the pre-lab was verified through testing.

The demonstration to the TA confirmed proper operation of all three ALU functions.

Post-lab Deliverables:

Deliverable 1: Completed Operation Table

Table 2: ALU Operations with Two's Complement Results

c_0	c_1	Operation	A	B	Result	Overflow
0	0	AND	0100 (4)	0110 (6)	0100	N
0	1	AND	0110 (6)	1101 (-3)	0100	N
1	0	Addition	0100 (4)	0110 (6)	1010	Y
1	0	Addition	0100 (4)	1101 (-3)	0001	N
1	0	Addition	1101 (-3)	1001 (-7)	0110	Y
1	1	Subtraction	0100 (4)	0111 (7)	1101	N
1	1	Subtraction	0110 (6)	1001 (-7)	1101	Y

Deliverable 2: Maximum Gate Delay and Critical Path

Assuming each 2-input gate has a delay of 1 unit, the maximum delay is along the path:

Path 1: Through AND Operation

- Input → AND gate (1) → MUX (1) → Output.
- Total delay: 2 gate delays.

Path 2: Through Addition/Subtraction Operation

- Input B → XOR gate (1) → 4-bit ripple carry adder.
- 4-bit Ripple Carry Adder: Worst-case carry propagation through 4 stages (2 delays per stage for carry chain) = 8 delays.
- Adder output → MUX (1) → Output.
- Total delay: 1 (XOR) + 8 (Ripple Carry Adder) + 1 (MUX) = 10 gate delays.

Maximum gate delay = 10 gate delays.

The critical path is: Input B → XOR → Ripple Carry Adder (full propagation) → MUX → Output. This path is highlighted in red on the gate-level schematic in Figure 2.

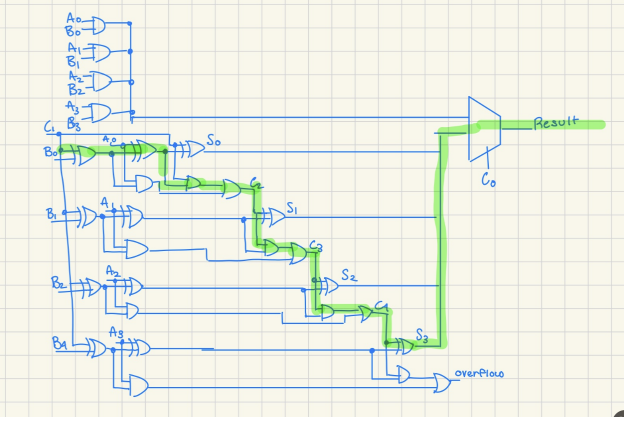

Figure 2: Maximum Gate Delay

Deliverable 3: Overflow Detection Unit Design

For two's complement arithmetic, overflow occurs when adding two numbers of the same sign produces a result with a different sign.

Signals:

- A_3: Sign bit of input A
- B_3: Sign bit of input B
- $Result_3$: Sign bit of result
- c_1: Operation select (we only check overflow for add/subtract)

$$Overflow = c_1 \cdot (A_3 \cdot B_3 \cdot \overline{Result_3} + \overline{A_3} \cdot \overline{B_3} \cdot Result_3) \tag{1}$$

$$Overflow = c_1 \cdot (\overline{A_3 \oplus B_3}) \cdot (A_3 \oplus Result_3) \tag{2}$$

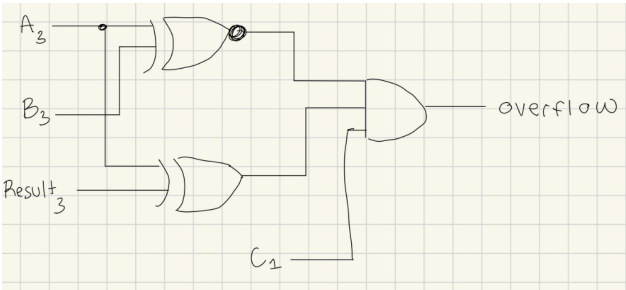

Figure 3: Overflow Detection Unit Schematic

Checking: Test the overflow circuit with the examples from Table 1 where overflow occurs.

Conclusion

This lab successfully demonstrated the design and implementation of a 4-bit Arithmetic Logic Unit (ALU) capable of performing AND, addition, and subtraction operations using two's complement arithmetic, a multiplexer for selection, and overflow detection. Key learnings included modular integration of components like the ripple carry adder and XOR gates for efficient add/subtract functionality, the role of multiplexers in resource sharing, and critical path analysis revealing a 10-gate delay bottleneck. These principles are foundational to computer architecture, providing hands-on experience in building essential computational blocks.