Innovation University

Faculty of Computers and Information Technology (FCIT)

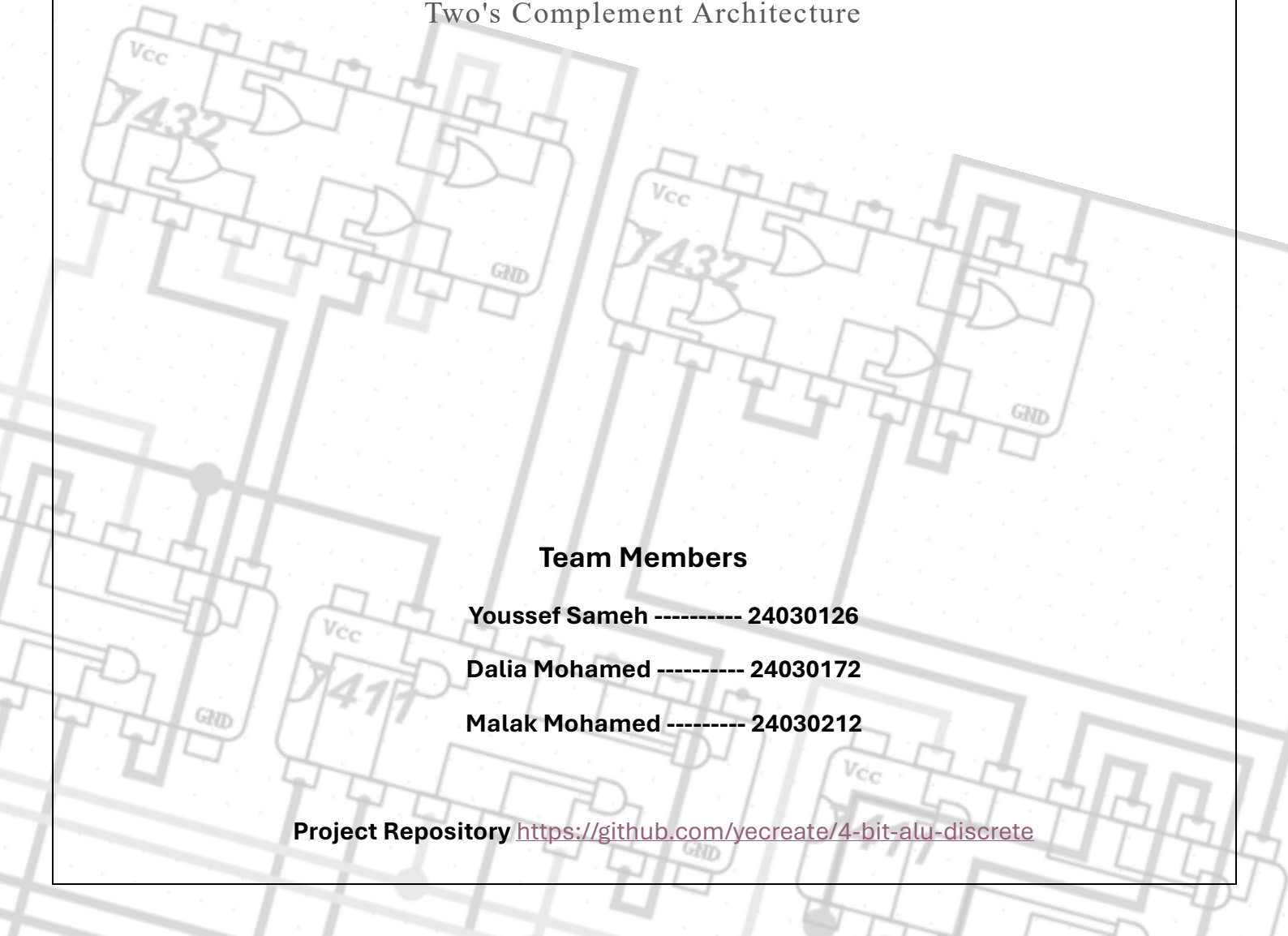
CF225: Computer Architecture

Level 2 - Fall Semester

December 2025

DESIGN & HARDWARE IMPLEMENTATION OF A 4-BIT ARITHMETIC LOGIC UNIT

A Discrete Logic Approach using 74HC Series CMOS Gates & Two's Complement Architecture



Team Members

Youssef Sameh ---------- 24030126

Dalia Mohamed ---------- 24030172

Malak Mohamed --------- 24030212

Project Repository https://github.com/yecreate/4-bit-alu-discrete

Contents

1. Introduction 3
2. Project Specifications 3
2.1 Functional Requirements 3
2.2 Hardware Architecture 3
3. Hardware Design & Implementation 4
3.1 Power Supply and Protection 4
3.2 The Arithmetic Unit (ADD/SUB) 4
3.3 The Logic Unit (AND/OR) 4
3.4 Multiplexing (Routing) 5
3.4.1 Hybrid Implementation Strategy 5
3.4.2 Discrete Logic Mux Design 5
4. Simulation and Verification 6
5. Testing and Results 6
5.1 Test Cases (Truth Table Verification) 6
5.2 The "CD4511 Limitation" (Technical Note) 8
5.3 Proposed Solutions for Full Hexadecimal Display 9
5.4 Scalability: Expanding to 8-Bit Architecture 9
6. Implementation Challenges 10
7. Future Proposals 10
8. Conclusion 10
9. Bill of Materials (BOM) 11
10. References 12
A. Component Datasheets (Technical Specifications) 12
B. Theoretical Concepts & Educational Resources 12
C. Software Tools 12
D. Digital Appendices & Source Files 12

1. Introduction

The **Arithmetic Logic Unit (ALU)** is the fundamental building block of the **Central Processing Unit (CPU)**, responsible for performing all **integer arithmetic and bitwise logical operations**. This project aims to design, simulate, and physically construct a functional **4-bit ALU using standard discrete logic gates (74HC series).**

The designed system accepts **two 4-bit binary inputs** (Operand **A** and Operand **B**) and performs one of four selectable operations: **AND, OR, ADD,** or **SUB**. The result is displayed on a **7-segment display**, utilizing a strong power management system to ensure reliability in any environment.

---

2. Project Specifications

2.1 Functional Requirements

The system is controlled by a 2-bit Opcode (Selection Lines **S1** and **S0**) which determines the operation path as explained in the following table:

S1	S0	Operation	Description	Active Component
0	**0**	AND	Bitwise Logical AND	74HC08
0	**1**	OR	Bitwise Logical OR	74HC32
1	**0**	ADD	Arithmetic Addition (A + B)	74HC283
1	**1**	SUB	Subtraction (A - B)	74HC283 + 74HC86

2.2 Hardware Architecture

The system is divided into four main subsystems:

1. **Power Supply Unit:** Regulated 5V DC.
2. **Input/Control Unit:** DIP switches for Data and Opcode.
3. **Processing Unit:** Logic gates and Arithmetic Adder.
4. **Output Unit:** Multiplexers and Display Drivers.

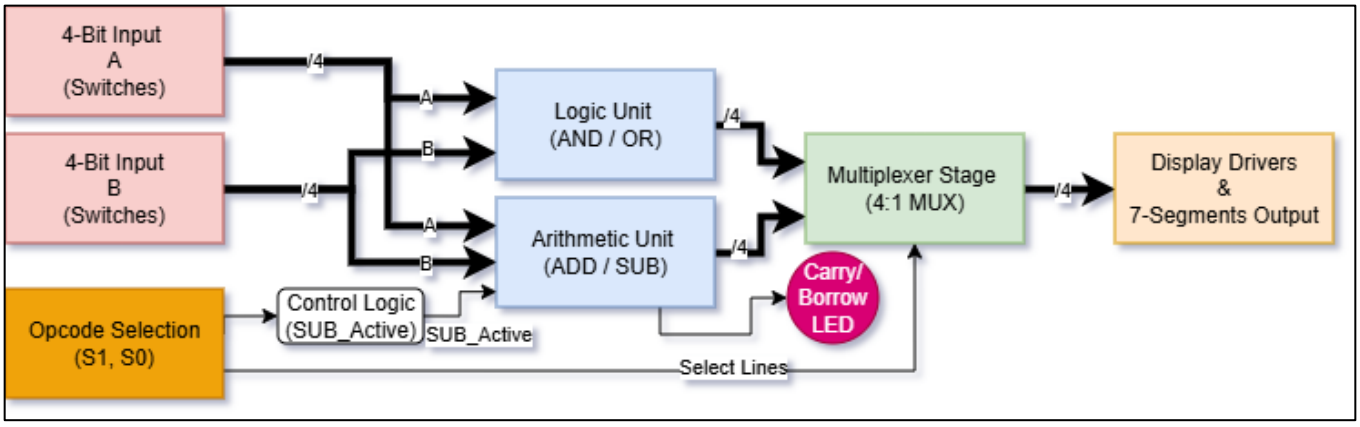


Figure 1 High-Level System Block Diagram showing Inputs, Control Logic, Processing Units, and Output.

3. Hardware Design & Implementation

3.1 Power Supply and Protection

To ensure stability and protect sensitive CMOS logic gates from voltage changeability, we moved away from direct battery connections.

- **Regulation:** A **LM2596 Buck Converter** steps down a 9V Battery to a precise 5.0V.
- **Protection:** A **5.1V Zener Diode** acts as an overvoltage clamp.
- **Noise Filtering: 100nF ceramic capacitors** are placed near logic ICs to filter high-frequency noise, and **10µF electrolytic capacitors** smooth the power rails.

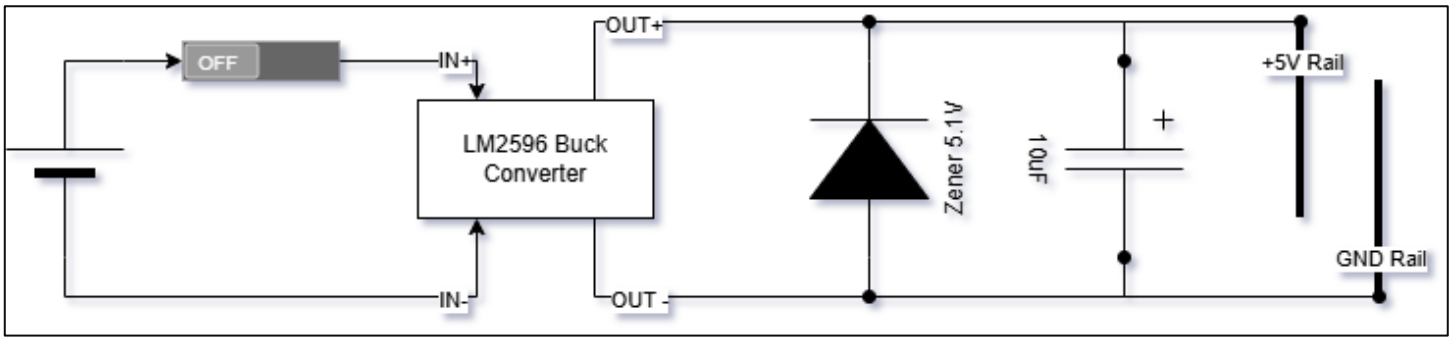


Figure 2 Power Regulation Unit. The 9V source is stepped down to 5V. A Zener Diode 5.1V is placed in parallel voltage spikes, protecting the 74HC logic gates.

3.2 The Arithmetic Unit (ADD/SUB)

The arithmetic core uses a **74HC283 4-bit Binary Full Adder**.

- **Addition:** Performed directly (A + B)
- **Subtraction:** Implemented using **Two's Complement Logic** (A + NOT(B) + 1)
  - We generate a **SUB_Active** control signal (**Active HIGH** when **S1=1**, **S0=1**).
  - This signal feeds into **74HC86 (XOR)** gates to invert Input B
  - The signal also connects to the **Carry-In (C_0)** of the Adder to add the necessary **+1**

3.3 The Logic Unit (AND/OR)

- **AND Operation**
  - Implemented using **74HC08** Quad 2-Input AND Gate IC
- **OR Operation**
  - Implemented using **74HC32** Quad 2-Input OR Gate IC

3.4 Multiplexing (Routing)

To select the correct result based on the Opcode, our design requires two 4:1 Multiplexing units (one for the lower 2 bits, one for the upper 2 bits).

3.4.1 Hybrid Implementation Strategy

Due to an unexpected component shortage of the 74HC153 IC during the final build phase and strict project deadlines, we adopted a hybrid engineering approach. We demonstrated adaptability by implementing the multiplexing logic in two different ways:

- **Lower Nibble (Bits 0 & 1):** Handled by a standard 1x 74HC153 (Dual 4:1 Multiplexer) IC.
- **Upper Nibble (Bits 2 & 3):** Handled by a **Discrete Logic Replacement Circuit**.

3.4.2 Discrete Logic Mux Design

We successfully replicated the functionality of the second 74HC153 using basic logic gates.

- **Logic:** We utilized 74HC11 (3-Input AND) gates for input selection, 74HC04 (NOT) for opcode inversion, and 74HC32 (OR) to sum the outputs.
- **Outcome:** While this increased wiring complexity, it successfully routed the correct signals for Bits 2 and 3, proving that logical function can be preserved even when specific integrated circuits are unavailable.

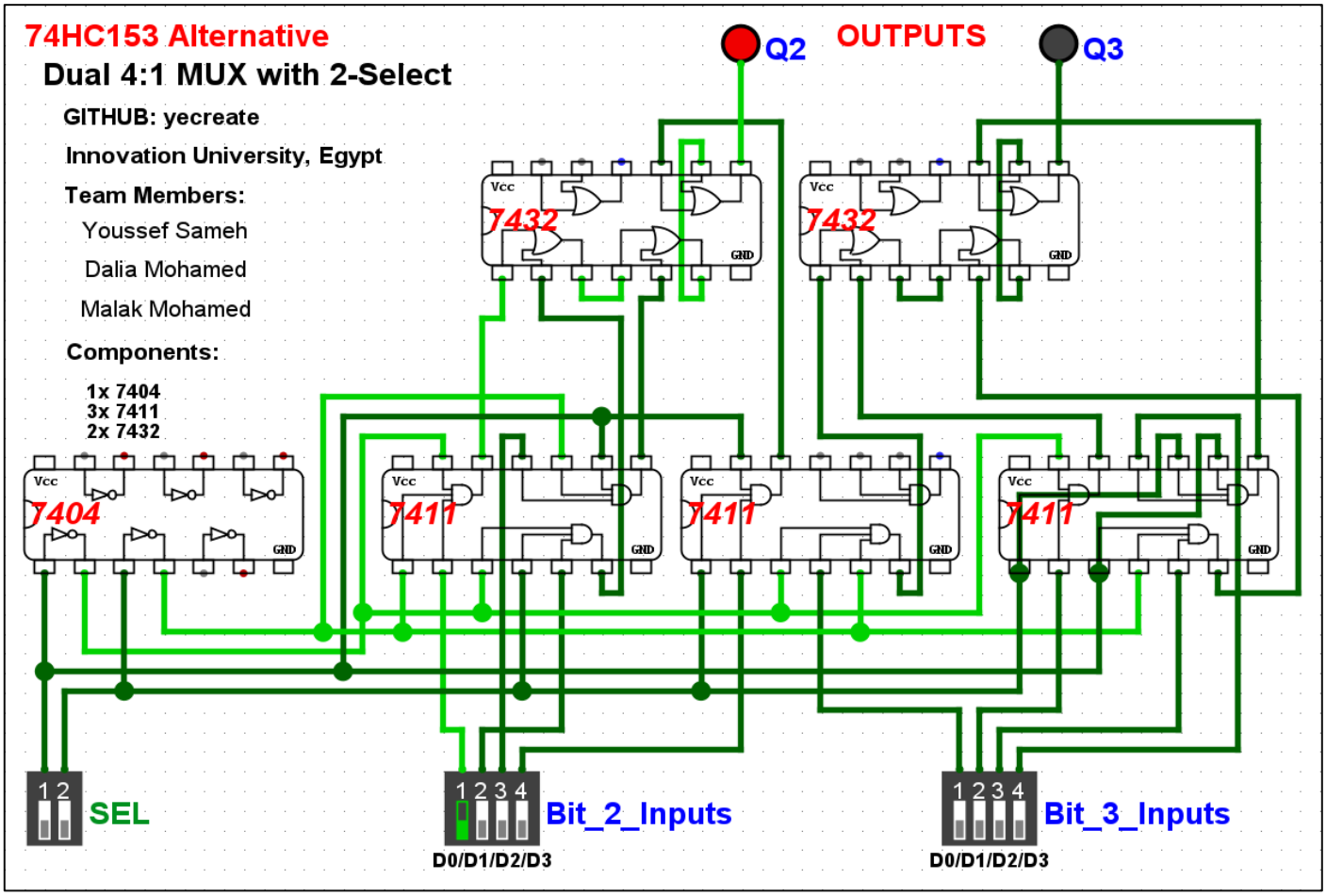


Figure 3 Discrete Logic Replacement Circuit Design for the second Multiplexer unit in Logisim Evolution, designed using 74HC11, 74HC32, and 74HC04

4. Simulation and Verification

Before hardware assembly, the logic was verified using **Logisim Evolution**. The simulation confirmed the **Two's Complement** logic and the **multiplexer** routing.

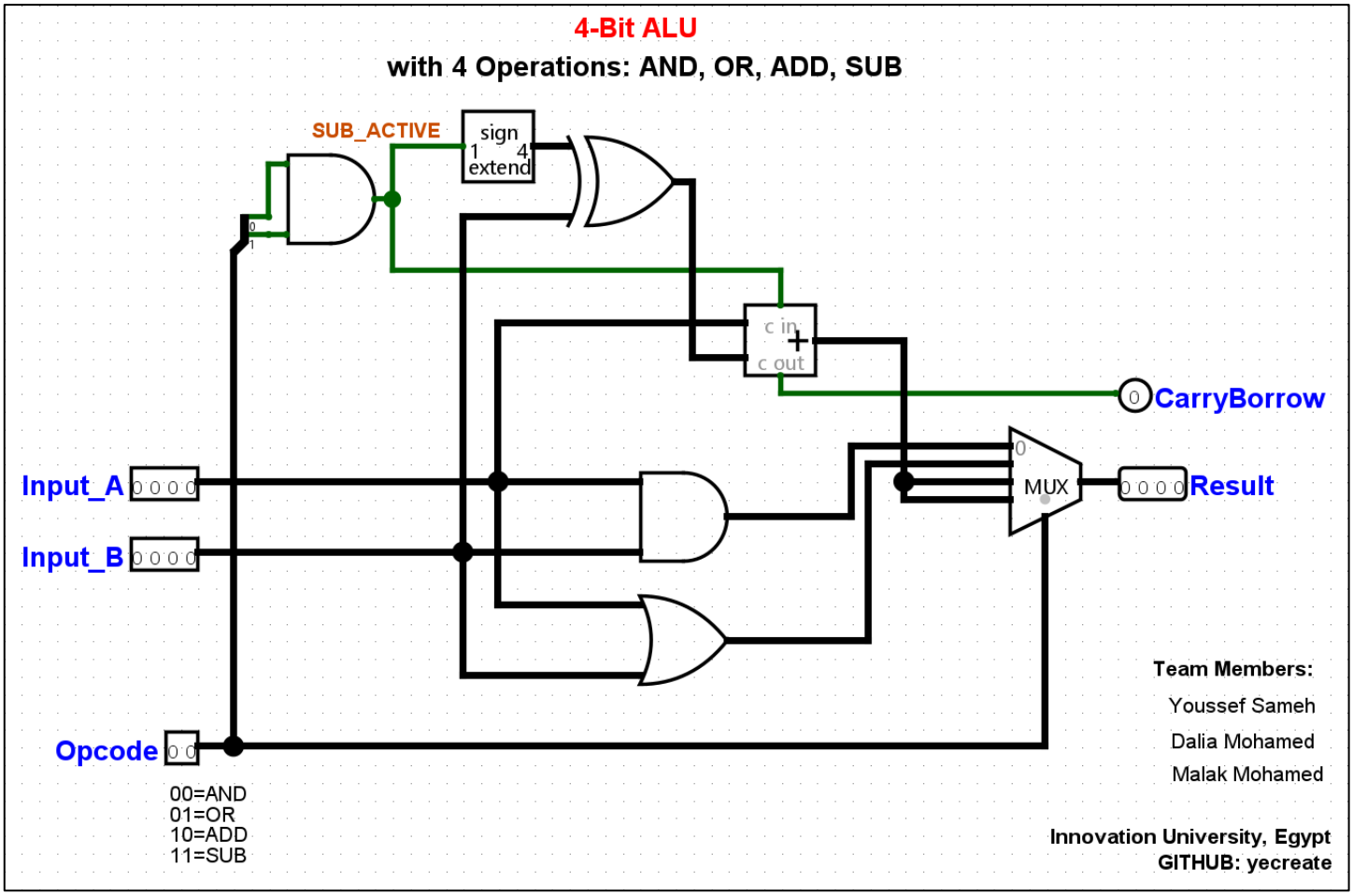


Figure 4 Full ALU Circuit Design in Logisim Evolution

---

5. Testing and Results

5.1 Test Cases (Truth Table Verification)

The following table documents the validation of the ALU across all **four operations**. Special attention is given to the **Red LED (Carry/Out)**, which **changes meaning** depending on whether the system is **adding** or **subtracting**.

- **Key for Red LED Status:**
  - **OFF (0)** Logic **LOW** (0V)
  - **ON (1)** Logic **HIGH** (5V)

Comprehensive ALU Hardware Verification and Truth Table Analysis

Test Case	*Opcode*	*Operation*	*Input A*	*Input B*	*Expected Result (Binary)*	*Display Output*	*Red LED (Cout)*	*Explanation*
1	**00**	**AND**	**1100** (12)	**1010** (10)	**1000**	**8**	Ignored*	Bitwise AND: Only overlapping 1s remain (1 . 1 = 1)
2			**0101** (5)	**0011** (3)	**0001**	**1**	Ignored*	Bitwise AND: 0101 . 0011 = 0001
3	**01**	**OR**	**0101** (5)	**0011** (3)	**0111**	**7**	Ignored*	Bitwise OR: Any 1 results in 1 (1+0=1)
4			**1000** (8)	**0001** (1)	**1001**	**9**	Ignored*	Bitwise OR: Combines bits to form 9
5	**10**	**ADD**	**0011** (3)	**0010** (2)	**0101**	**5**	**OFF**	Standard Addition. 3+2=5 Result fits in 4 bits; No Overflow.
6			**1001** (9)	**0001** (1)	**1010**	**(Blank)**	**OFF**	Result is 10 (1010_2). **CD4511 Blanking** occurs (See Section 5.2). No Overflow
7			**1111** (15)	**0001** (1)	**0000**	**0**	**ON**	**Overflow Case.** 15+1=16. 4-bit limit is 15, so result wraps to 0. LED indicates Overflow (16)
8			**1000** (8)	**1000** (8)	**0000**	**0**	**ON**	**Overflow Case.** 8+8=16. Result wraps to 0. LED indicates the 5th bit (Overflow)
9	**11**	**SUB**	**0111** (7)	**0010** (2)	**0101**	**5**	**ON**	**Positive Result (A >= B).** In 2's Comp, Carry=1 means **No Borrow**. Result is positive
10			**1111** (15)	**1111** (15)	**0000**	**0**	**ON**	**Zero Result.** 15-15=0. Carry=1 (No Borrow)
11			**0100** (4)	**0111** (7)	**1101**	**(Blank)**	**OFF**	**Negative Result (A < B).** 4-7 = -3 In 4-bit 2's comp, -3 is 1101 (13) LED OFF means **Borrow Active** (Negative)

Technical Note on the "Red LED" (Cout)

To ensure accurate interpretation of the results, the behavior of the **Red LED** connected to the **Carry Out (C_4)** pin of the **74HC283 Adder** must be understood in context:

1. **During ADD Operations (A + B):**
    - **LED ON:** Indicates **Arithmetic Overflow**. The sum exceeded **15** (1111_2), and the result shown on the display is the shortened 4-bit value.
    - **LED OFF:** The sum fits within the standard 4-bit range (0-15).
2. **During SUB Operations (A - B):**

    The subtraction is performed using **Two's Complement Logic** (A + NOT(B) + 1). In this mode, the Carry bit functions as an **Active-Low Borrow** indicator:

    - **LED ON (Carry = 1)**
        - Indicates **No Borrow**. The result is **Positive or Zero** (A >= B)
    - **LED OFF (Carry = 0)**
        - Indicates **Borrow Active**. The result is **Negative** (A < B)
    - *Note: Since the 7-segment display is unsigned, a negative result (like -3) will appear as its two's complement binary equivalent (13, or 1101), causing the CD4511 to blank.*
3. **During LOGIC Operations (AND / OR):**
    - **Status: Ignored**
    - *Explanation:* The 74HC283 Adder is permanently powered and receiving inputs A and B even when the Opcode selects AND/OR. Therefore, the Red LED may light up if A+B > 15, but this signal is electrically irrelevant to the Logic Unit's output.

5.2 The "CD4511 Limitation" (Technical Note)

We utilized the **CD4511 BCD-to-7-Segment Decoder** for result visualization. It is crucial to note the following hardware characteristics:

- **Constraint:** The CD4511 is a **BCD** (Binary Coded Decimal) decoder, not a full Hexadecimal decoder.
- **Observation:** It correctly displays results from **0 to 9**. However, if the ALU result is **10, 11, 12, 13, 14, or 15** (**1010_2** to **1111_2**), the display enters a "Blanking State" (turns off).
- **Conclusion:** This is **expected hardware behavior** documented in the CD4511 datasheet, not a circuit wiring error.

5.3 Proposed Solutions for Full Hexadecimal Display

To address the blanking issue and display Hexadecimal characters **(A, b, C, d, E, F)** for results **10-15**, future iterations of this project could implement one of the following engineering solutions:

A. Specialized Hex Decoder ICs Replacing the **CD4511** with a **MC14495** or **DM9368**. These chips are pin-compatible logic decoders specifically designed to map binary inputs 1010 through 1111 to the 7-segment patterns for **'A'** through **'F'**. These components are currently **obsolete** and difficult to source, which forced our use of **CD4511**

B. ROM-Based Decoding Using a small **EEPROM** (Electrically Erasable Programmable Read-Only Memory) chip. The 4-bit ALU result would address the memory, and the data output would be the exact 7-bit segment pattern required. This allows for custom character shapes but adds **significant complexity**.

C. Microcontroller Driver Interfacing the 4-bit result to a microcontroller (e.g. ATmega328P). The software would read the nibble and drive the display segments directly. **While effective**, this **deviates** from the "**Pure Discrete Logic**" requirement of this course project.

5.4 Scalability: Expanding to 8-Bit Architecture

The modular design of this 4-bit ALU allows for direct scalability to 8-bit, 16-bit, or higher-order architectures through a method known as **Cascading**

- **Ripple Carry Configuration**: To create an 8-bit ALU, **two identical 4-bit ALU units** can be connected in series.
    - **Low-Order Nibble:** The first ALU processes **bits 0-3**. Its Carry Input ($\mathbf{C_0}$) is connected to the Opcode selection (for the initial **+1** in subtraction).
    - **High-Order Nibble:** The second ALU processes bits **4–7**. Crucially, the Carry Out ($\mathbf{C_4}$) of the first ALU is connected directly to the Carry In ($\mathbf{C_0}$) of the second ALU.

- **Unified Control:** The **Opcode** selection lines (S0, S1) would be shared in **parallel** across both units, ensuring both ALUs perform the **same operation** (e.g., ADD or AND) at the same time.

- **Trade-off:** While cascading increases bit-width capacity, it introduces **Propagation Delay**. The high-order ALU must wait for the carry signal to "ripple" through the low-order adder logic *before* its final result is valid. In modern processors, this is softened using **Look-Ahead Carry Logic**, but for this discrete logic implementation, the **Ripple Carry** method remains the most **hardware-efficient scaling solution**.

6. Implementation Challenges

Moving from Logisim verification to physical hardware presented distinct engineering obstacles:

- **Hybrid Multiplexing:** The unavailability of a second **74HC153** chip required immediate adaptation. We engineered a discrete logic replacement using **74HC11**, **74HC32**, and **74HC04** gates. This "hybrid" approach validated our reach of internal component logic, effectively replacing the missing integrated circuit chip with a functional discrete circuit.
- **Signal Stability:** Early tests showed erratic counter behavior due to breadboard parasitic capacitance. We resolved this by placing **100nF ceramic decoupling capacitors** near each IC and **10µF electrolytic capacitors** on power rails, ensuring stable operation for the high-speed 74HC logic.
- **Wire Management:** To mitigate "spaghetti wiring" risks, we implemented utilized flush-mount wiring, significantly easing debugging and improving mechanical reliability.

---

7. Future Proposals

While the prototype meets all functional requirements, future iterations could offer enhanced performance:

- **PCB Fabrication:** Transitioning from breadboards to a custom two-layer PCB would minimize noise, reduce the physical footprint, and eliminate connection fragility.
- **Hexadecimal Decoding:** As detailed in *Section 5.2*, the CD4511 blanks on results 10–15. A future upgrade should incorporate a ROM look-up table or a microcontroller driver to display full Hexadecimal characters (A–F) for values greater than 9.
- **Input Latching:** Replacing DIP switches with momentary buttons and D-Flip-Flops would improve user interaction, mimicking commercial calculator input methods.

---

8. Conclusion

This project successfully demonstrated the design and construction of a functional 4-bit ALU. By integrating the **74HC283 Arithmetic Unit** with discrete logic gates, the system effectively performs **AND, OR, ADD,** and **SUB** operations.

A key achievement was the hardware implementation of **Two's Complement Subtraction** using XOR gates and Carry-In manipulation, proving the efficiency of binary arithmetic.

Furthermore, the team's ability to **troubleshoot real-time constraints**. specifically engineering a ***discrete multiplexer solution***, demonstrated a practical mastery of digital logic beyond theoretical simulation. The final prototype stands as a robust, verified educational platform that fully satisfies the **CF225 Computer Architecture** course requirements.

9. Bill of Materials (BOM)

Category	Component	Qty	Description / Function
Integrated Circuits (Logic)	**74HC283**	1	4-bit Binary Full Adder (Arithmetic Unit)
	74HC153	1	Dual 4-Input Multiplexer (Output Selection)
	74HC08	1	Quad 2-Input AND Gate (Logic Unit)
	74HC11	3	Triple 3-Input AND Gate (Discrete Mux Logic)
	74HC32	3	Quad 2-Input OR Gate (Logic Unit/Discrete MUX Logic)
	74HC86	1	Quad 2-Input XOR Gate (SUB/Inversion)
	74HC00	1	Quad 2-Input NAND Gate (Control Logic)
	74HC04	2	Hex Inverter / NOT Gate (Control Logic/MUX)
	CD4511	2	BCD to 7-Segment Latch/Decoder
Power Management	**LM2596**	1	DC-DC Buck Converter (9V to 5V Regulation)
	9V Battery	1	Primary Power Source
	Battery Holder	1	9V Case with Integrated ON/OFF Switch
	Zener Diode	1	5.1V (5V1) Overvoltage Protection
Input / Output	**7-Segment Display**	2	Common Cathode (0.56")
	DIP Switch (4-Way)	2	Input Operands A and B
	DIP Switch (2-Way)	1	Operation Select (S1, S0)
	LED (Green, 3mm)	4	Logic High Indicators (Operand A)
	LED (Yellow, 3mm)	4	Logic High Indicators (Operand B)
	LED (Red)	1	Carry / Borrow Indicator
	LED Spacers	9	5mm Height
Passive Components	**Resistor Network**	3	10kΩ, 5-pin (4 resistors) - Input Pull-downs
	Resistor Network	1	220Ω, 9-pin (8 resistors) - Display Current Limiting
	Resistor Network	1	220Ω, 5-pin (4 resistors) - LED Current Limiting
	Resistor (Discrete)	15	330Ω – 7-segment and Carry LED Current Limiting
	Ceramic Capacitor	8	100nF (104) - IC Decoupling/Noise Filtering
	Electrolytic Cap	4	10µF 25V - Power Rail Smoothing
Prototyping	**Breadboard (830)**	2	Large Tie-point (Main Build Area)
	Breadboard (400)	5	Medium Tie-point (Sub-circuits)
	Breadboard (170)	1	Mini Tie-point (Power Module)
	Flat Jumpers	3 Pks	Flush-mount wires for neat routing

10. References

A. Component Datasheets (Technical Specifications)

- **74HC283 (4-Bit Binary Full Adder):**
  - Nexperia. (2021). *74HC283; 74HCT283 4-bit Binary Full Adder with Fast Carry*.
- **74HC153 (Dual 4-Input Multiplexer):**
  - Texas Instruments. (1998). *SN74HC153 Dual 4-Line to 1-Line Data Selectors/Multiplexers*.
- **CD4511 (BCD to 7-Segment Latch/Decoder):**
  - Texas Instruments. (2004). *CD4511B CMOS BCD-to-7-Segment Latch Decoder Drivers*.
- **LM2596 (Buck Converter):**
  - Texas Instruments. (2016). *LM2596 SIMPLE SWITCHER® Power Converter 150-kHz 3-A Step-Down Voltage Regulator*.
- **74HC08 (Quad 2-Input AND):**
  - Nexperia. *74HC08; 74HCT08 Quad 2-input AND gate*
- **74HC32 (Quad 2-Input OR):**
  - Nexperia. *74HC32; 74HCT32 Quad 2-input OR gate*
- **74HC86 (Quad 2-Input XOR):**
  - Nexperia. *74HC86; 74HCT86 Quad 2-input XOR gate*.

B. Theoretical Concepts & Educational Resources

- **Two's Complement Arithmetic:**
  - Cornell University. (2014). *ECE 4760: Binary Arithmetic and Two's Complement*.
- **Digital Logic & Multiplexing:**
  - Electronics Tutorials. (2023). *The Multiplexer (MUX) and Multiplexing*.
- **Power Supply Design (Decoupling Capacitors):**
  - Analog Devices. (2009). *MT-101: Decoupling Techniques*.
- **CD4511 Hexadecimal Limitation:**
  - Create Arduino. (2020). *Understanding the CD4511 BCD to 7 Segment Decoder*.

C. Software Tools

- **Logisim Evolution:**
  - *Open-source educational tool for designing and simulating digital logic circuits*.
- **Draw.io (Diagrams.net):**
  - *Browser-based diagramming software used for system architecture design*

D. Digital Appendices & Source Files

The complete Logisim Evolution simulation files (.circ), high-resolution schematics, and digital documentation for this project are open-source and available at:

- **GitHub Repository:** https://github.com/yecreate/4-bit-alu-discrete