
بسم الله الرحمن الرحيم

**Project Title:** Transistor-Level 4-BIT ALU DESIGN

**Module:** 4-BIT ALU UNIT
Design


- Author: Ahmed Assem Mohamed


May 18, 2025

21 ذو القعدة، 1446

## 1. Abstraction

This report presents the design and implementation of a **4- Bit Arithmetic Logic Unit (ALU)** using 130nm **CMOS** technology**.** The ALU can perform both arithmetic and logical operations. Control logic allows the selection between these two classes of operations via the most significant bit (MSB) of a 4-bit selection line. The circuit was meticulously developed and simulated using Cadence Virtuoso.

## 2. Logic Function and Truth Table

| Sel[3:0] | Mode | Operation | Output = f(A, B) | Notes |
|---|---|---|---|---|
| **0000** | Arithmetic | Increment A | A + 1 | Wraps around on overflow |
| **0001** | Arithmetic | Increment B | B + 1 | Wraps around on overflow |
| **0010** | Arithmetic | Transfer A | A | Pass-through |
| **0011** | Arithmetic | Transfer B | B | Pass-through |
| **0100** | Arithmetic | Decrement A | A - 1 | Wraps around on underflow |
| **0101** | Arithmetic | Multiply A and B | A × B | 8-bit result shown |
| **0110** | Arithmetic | Add A and B | A + B | 4-bit result |
| **0111** | Arithmetic | Subtract A - B | A - B | A assumed ≥ B |
| **1000** | Logic | 1’s Complement A | ~A | Bitwise NOT |

| | | | | |
|---|---|---|---|---|
| **1001** | Logic | 1’s Complement B | ~B | Bitwise NOT |
| **1010** | Logic | AND | A & B | Bitwise AND |
| **1011** | Logic | OR | (A \| B) | Bitwise OR |
| **1100** | Logic | XOR | A ^ B | Bitwise XOR |
| **1101** | Logic | XNOR | ~(A ^ B) | Bitwise XNOR |
| **1110** | Logic | NAND | ~(A & B) | Bitwise NAND |
| **1111** | Logic | NOR | ~(A \| B) | Bitwise NOR |

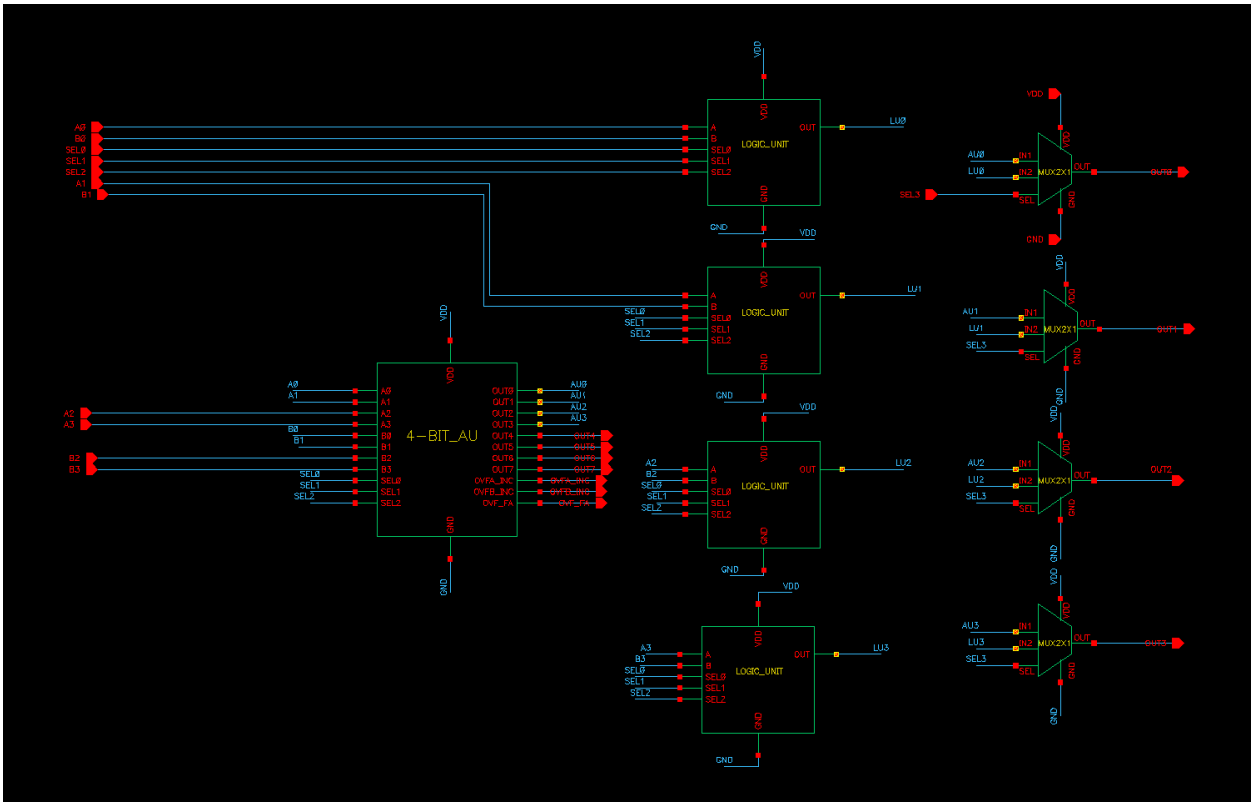
Some sample test cases. Example Input/Output Combinations (for A = 0101 (5), B = 0011 (3))

| Sel[3:0] | Operation | Output (Binary) | Output (Decimal) |
|---|---|---|---|
| **0000** | A + 1 | 0110 | 6 |
| **0001** | B + 1 | 0100 | 4 |
| **0010** | Transfer A | 0101 | 5 |
| **0011** | Transfer B | 0011 | 3 |
| **0100** | A - 1 | 0100 | 4 |
| **0101** | A × B | 00001111 (15) | 15 |
| **0110** | A + B | 1000 | 8 |
| **0111** | A - B | 0010 | 2 |
| **1000** | ~A | 1010 | 10 |
| **1001** | ~B | 1100 | 12 |
| **1010** | A & B | 0001 | 1 |
| **1011** | A \| B | 0111 | 7 |
| **1100** | A ^ B | 0110 | 6 |
| **1101** | ~(A ^ B) | 1001 | 9 |

| | | | |
|---|---|---|---|
| **1110** | ~(A & B) | 1110 | 14 |
| **1111** | ~(A \| B) | 1000 | 8 |

# 3. Circuit Design

## 3.1. Schematic



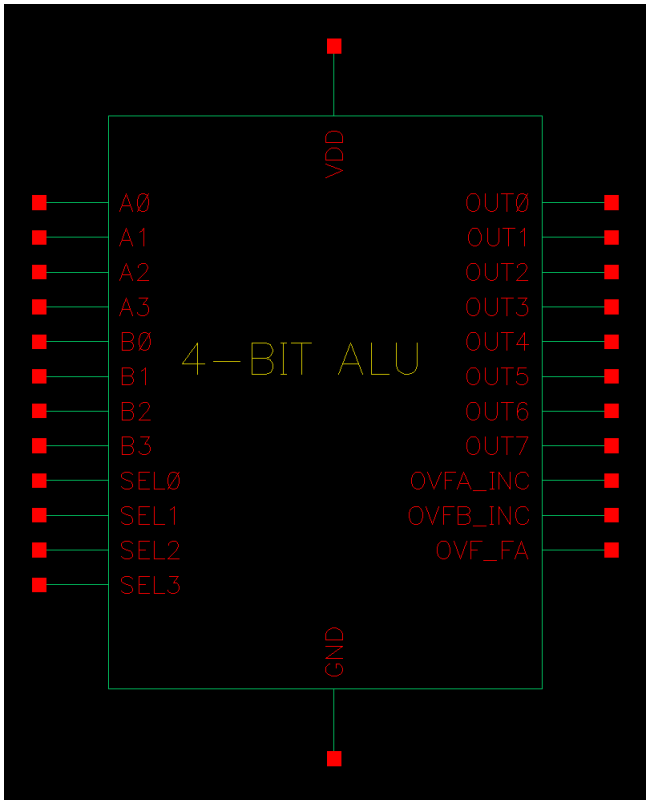

## 3.2. Design Approach

- 130nm CMOS process
- 1.2V VDD
- INPUTS:  4-BIT IN0, 4-BIT IN1, 4 SEL
- OUTPUTS: 8-BIT OUT, 3-BIT OVF
- Input parameter: WN_ALU
- Sizes: All gates have this size(even if it is the parameter for MUX-8X1) why? as already each gate is designed alone,

which is suitable size, this parameter is only to reduce number of variables

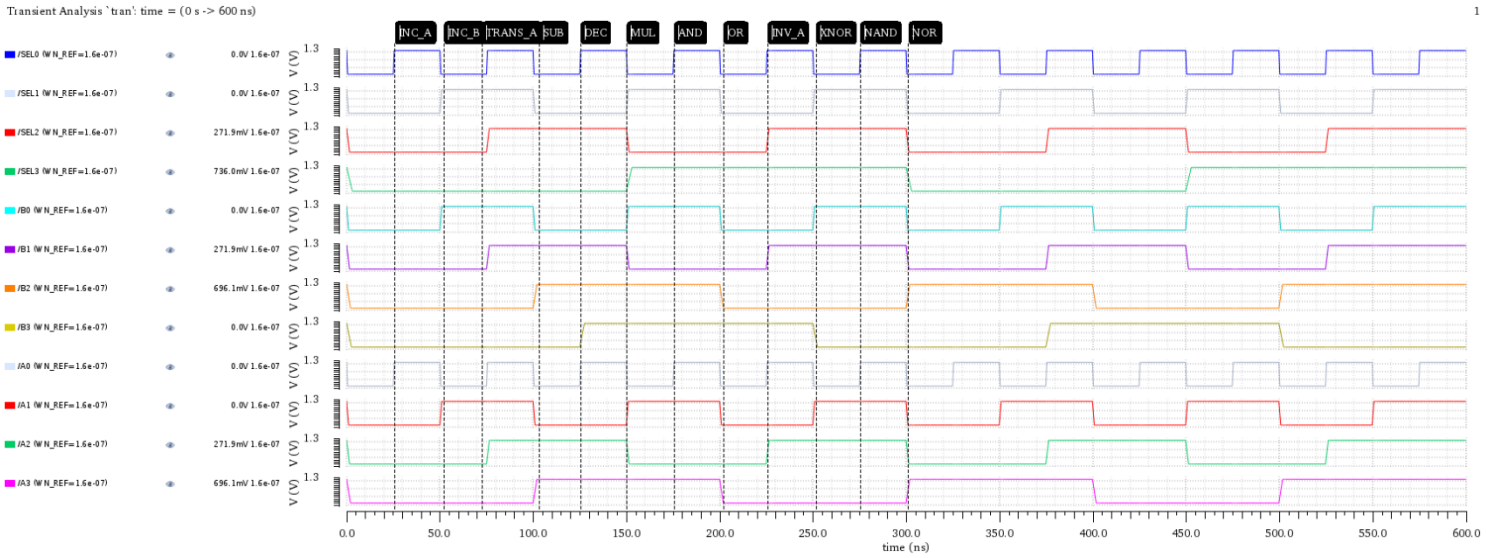
- Design Approach: 4-BIT ALU is the integration between 4*(1-BIT LU), 1*(4-BIT AU), and 4*(MUX_2X1)
- Although there is 8-bit output, all these bits are important only with MUL-circuit, for the rest of operations only the first 4 is important.
- OVF Flags: OVF_FA, OVFA_INC, OVFB_INC
  The first one is used only with FA-circuit, The second one is used only with INC/DEC-circuit for A input, The third one is used only with INC-circuit for B input.
- NOTE: OVF flags are designed on the assumption that the inputs are signed numbers.

## 3.3. Symbol

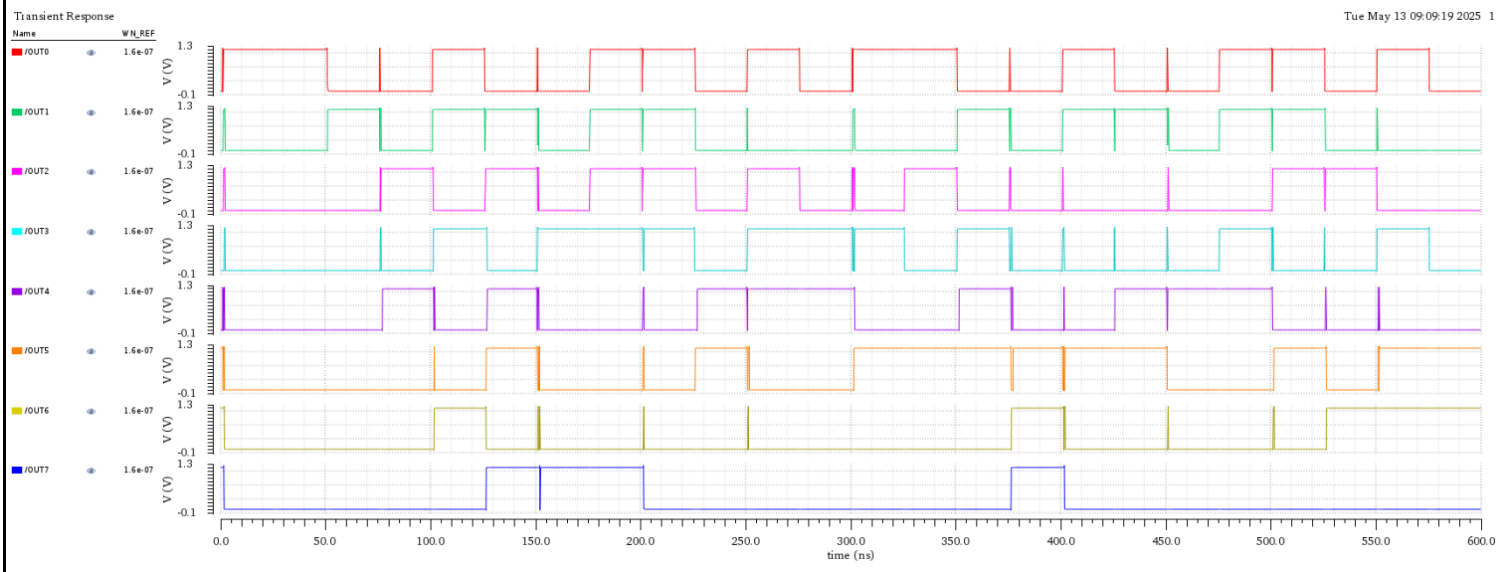


# 4. Simulation and Results

## Inputs




## Outputs




Note: the simulation result is performed with a max input f = 20.0MHZ