# 2-Bit ALU Design and Implementation

This project provides a comprehensive implementation of a 2-bit Arithmetic Logic Unit (ALU) using discrete 7400 series logic gates. It includes detailed hardware implementation guides, software simulation, and testing procedures.

## Project Overview

An Arithmetic Logic Unit (ALU) is a fundamental component of any CPU, responsible for performing various arithmetic and logical operations. This project implements a simplified 2-bit ALU that can perform the following operations:

- Bitwise AND
- Bitwise OR
- Bitwise XOR
- Addition with carry
- Subtraction using 2's complement
- NOT (1's complement)

Mermaid diagram removed for PDF compatibility

## Project Contents

### Documentation

- `docs/project_report.md`: Detailed project report with theoretical background and analysis
- `hardware_design/circuit_implementation.md`: Step-by-step hardware implementation guide

### Software Simulation

- `src/alu_simulator.py`: Command-line simulation of the 2-bit ALU operations
- `src/alu_visualizer.py`: GUI-based visualization showing ALU operations and circuit diagrams

## Getting Started

### Software Requirements

- Python 3.6 or higher
- Tkinter (for GUI visualization)
- NumPy (for numerical operations)

### Hardware Requirements

- Breadboard
- Logic Gate ICs:
  - 1× 7408 (Datasheet): Quad 2-input AND gates
  - 1× 7432 (Datasheet): Quad 2-input OR gates
  - 1× 7486 (Datasheet): Quad 2-input XOR gates
  - 1× 7404 (Datasheet): Hex inverter (NOT gates)
- 6× SPDT switches for inputs
- 3× LEDs with appropriate resistors (330Ohm)
- 6× 10kOhm pull-up resistors
- 4× 0.1uF decoupling capacitors
- Jumper wires
- 5V power supply

### Installation

1. Clone this repository:
2. Install required dependencies:

### Running the Simulator

1. Run the command-line simulator:
2. Run the GUI visualizer:

### Building the Hardware

See the detailed instructions in `hardware_design/circuit_implementation.md`.

The basic steps are: 1. Set up the power supply for all ICs 2. Connect input switches with pull-up resistors 3. Wire the logic gates according to the operation implementation diagrams 4. Connect the output LEDs through current-limiting resistors

Mermaid diagram removed for PDF compatibility

## Operation Codes

The ALU uses a 2-bit operation selector:

| Operation | Op Code (S1:S0) | Description |
|---|---|---|
| AND | 00 | Bitwise AND |
| OR | 01 | Bitwise OR |
| XOR/ADD | 10 | Addition with carry |
| NOT/SUB | 11 | Subtraction with borrow |

## Example Operations

### AND Operation Example

### Addition Example

### Subtraction Example

## Circuit Diagram

Mermaid diagram removed for PDF compatibility

For detailed circuit diagrams, see `hardware_design/circuit_implementation.md`.

## Troubleshooting

### Common Issues and Solutions

1. **No Power to ICs**
   - Check +5V at pin 14 and GND at pin 7 of all ICs
   - Verify power supply is providing 5V
2. **Inconsistent Results**
   - Add 0.1uF decoupling capacitors near each IC
   - Check for floating inputs
   - Verify pull-up resistors are properly connected
3. **Switch Bounce Issues**
   - Add debounce circuits (RC filters) to input switches

## Video Demonstrations

- ALU Hardware Demo (placeholder)
- Software Simulator Walkthrough (placeholder)

## Additional Resources

- Digital Logic Design Tutorial
- Ben Eater's 8-bit Computer Series
- Online Logic Circuit Simulator
- TTL Logic Databook
- Interactive Simulator

## Project Extensions

Want to extend this project? Here are some ideas:

1. Expand to 4-bit or 8-bit ALU
2. Add multiplication and division operations
3. Create a PCB design instead of breadboard
4. Connect to a simple control unit to create a basic CPU
5. Implement using a modern FPGA instead of discrete logic

## License



## Acknowledgments

- Course instructor and teaching assistants
- Texas Instruments for 7400 series datasheets
- Ben Eater's educational videos on digital logic