**Project Title:** Low-Power 8-bit ALU Design Using VLSI Techniques

**Internship Organization:** Codec Technologies

**Intern Name:** Ragulgandhi G

**Duration:** July 2025 – August 2025

---


## Abstract

This project, carried out as part of my internship at **Codec Technologies**, focuses on the design and implementation of a **low-power 8-bit Arithmetic Logic Unit (ALU)** using **VLSI design techniques**. The ALU is a fundamental component of any processor, responsible for performing arithmetic and logical operations. In this project, the ALU was designed using **Verilog HDL** for RTL description, and simulated using tools such as **ModelSim and Vivado** to verify functionality.

To address the growing demand for power-efficient digital systems, various **power optimization techniques** were integrated into the design—primarily **clock gating**, which disables inactive functional blocks to reduce dynamic power consumption. The ALU architecture supports multiple operations including addition, subtraction, logical AND, OR, XOR, NOT, and bitwise shifting.

After functional verification, the design was further developed at the layout level using **Microwind**, allowing transistor-level implementation and power estimation. The project emphasizes the importance of bridging RTL design and physical implementation to achieve a balanced trade-off between functionality, power, and area. This work lays a strong foundation in both **front-end and back-end VLSI design**, and is aligned with industry practices in digital hardware design.


---

## 1. Introduction

An Arithmetic Logic Unit (ALU) is a crucial digital circuit component found in microprocessors and CPUs, responsible for performing arithmetic and logical operations. It acts as the computational engine of a processor, enabling fundamental functions like addition, subtraction, logical comparison, and bit manipulation.

As devices become smaller and more complex, the demand for low-power VLSI design has increased significantly. Power-efficient ALU design is essential in extending battery life, minimizing heat generation, and reducing overall energy consumption in portable and embedded systems.

In this project, an 8-bit ALU was designed using **Verilog HDL** for Register Transfer Level (RTL) implementation. Simulation was conducted using **ModelSim and Vivado**, while

layout and physical-level design were performed using **Microwind** and **Magic VLSI**. The goal was to analyze and reduce power consumption through optimization techniques such as **clock gating**.

---

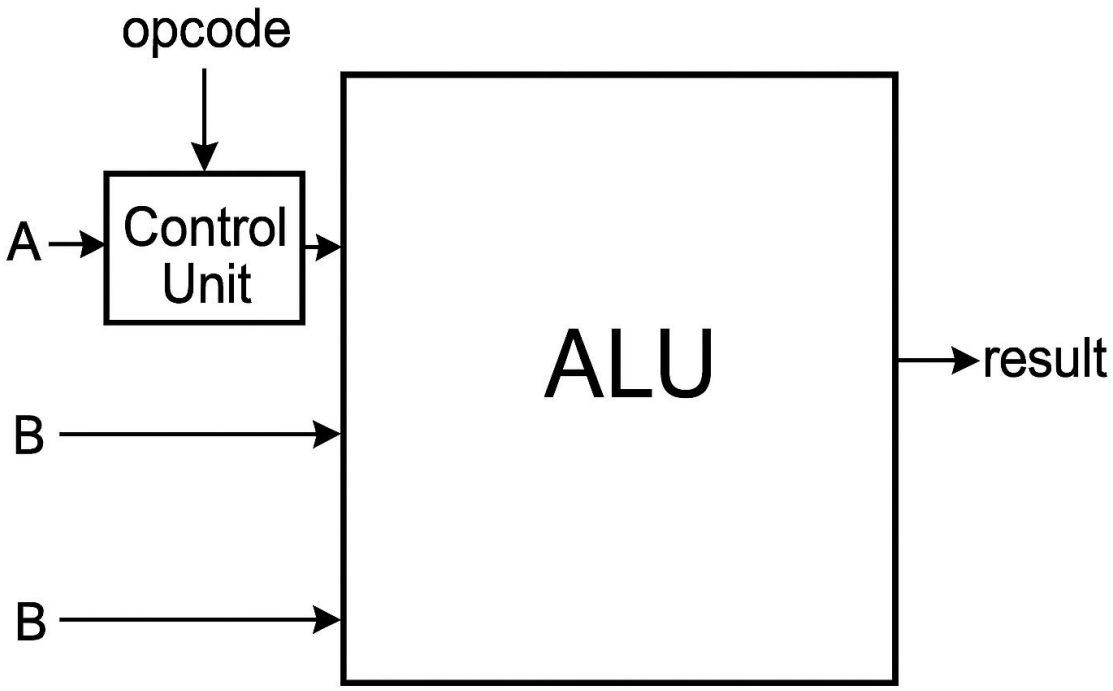
## 2. ALU Functional Design

The ALU designed in this project performs a range of arithmetic and logical operations on 8-bit operands. The operations are selected using a 3-bit opcode input.

### *Supported Operations:*

- **Arithmetic:** ADD, SUB
- **Logic:** AND, OR, XOR, NOT
- **Shift:** Left Shift, Right Shift

### *Block Diagram:*




### *Input/Output Signals:*

- `A[7:0]`, `B[7:0]`: 8-bit input operands
- `opcode[2:0]`: 3-bit control signal to select operation

- `result[7:0]`: 8-bit output
- `clk`: Clock input

---

## 3. Verilog RTL Coding

The ALU was developed using modular Verilog design, separating functional blocks for clarity and testing. Each operation is executed based on the opcode input using a `case` statement.

***ALU Top-Level Module:***

```
module alu_8bit (
    input [7:0] A, B,
    input [2:0] opcode,
    input clk,
    output reg [7:0] result
);
    always @(posedge clk) begin
        case (opcode)
            3'b000: result <= A + B;          // ADD
            3'b001: result <= A - B;          // SUB
            3'b010: result <= A & B;          // AND
            3'b011: result <= A | B;          // OR
            3'b100: result <= A ^ B;          // XOR
            3'b101: result <= ~A;             // NOT
            3'b110: result <= A << 1;         // Shift Left
            3'b111: result <= A >> 1;         // Shift Right
        endcase
    end
endmodule
```

***Clock Gating Logic:***

```
wire gated_clk = clk & enable;

always @(posedge gated_clk) begin
    // Perform operation only if enabled
end
```

*A testbench was created to simulate all operations and validate output results. Screenshots of waveforms are attached in the results section.*

---

## 4. Power Optimization Techniques

The main technique used for power optimization is **clock gating**, which prevents unnecessary switching in inactive ALU blocks, reducing dynamic power.

### *How It Works:*

- An `enable` signal is used to control clock delivery to ALU modules.
- When an operation is not needed, its corresponding logic block is disabled.

Optional techniques such as **operand isolation** and **dynamic logic** were explored but not fully implemented due to complexity at this stage.

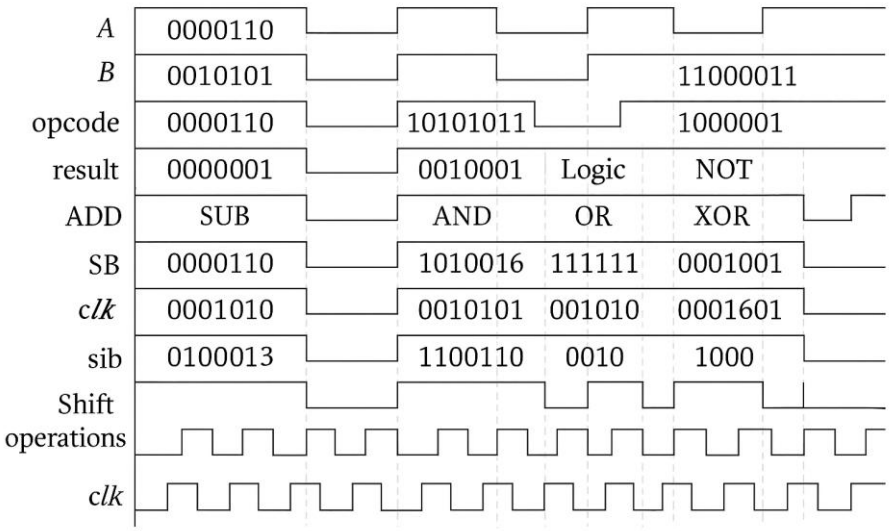
---

## 5. Simulation Results (Waveforms, Timing)

The ALU design was simulated using **ModelSim** and **Vivado** to verify functionality.

### *Waveforms Include:*

- ADD: A + B result
- SUB: A - B result
- Logic: AND, OR, XOR, NOT
- Shift operations




Simulatiion Results (Waveforme's

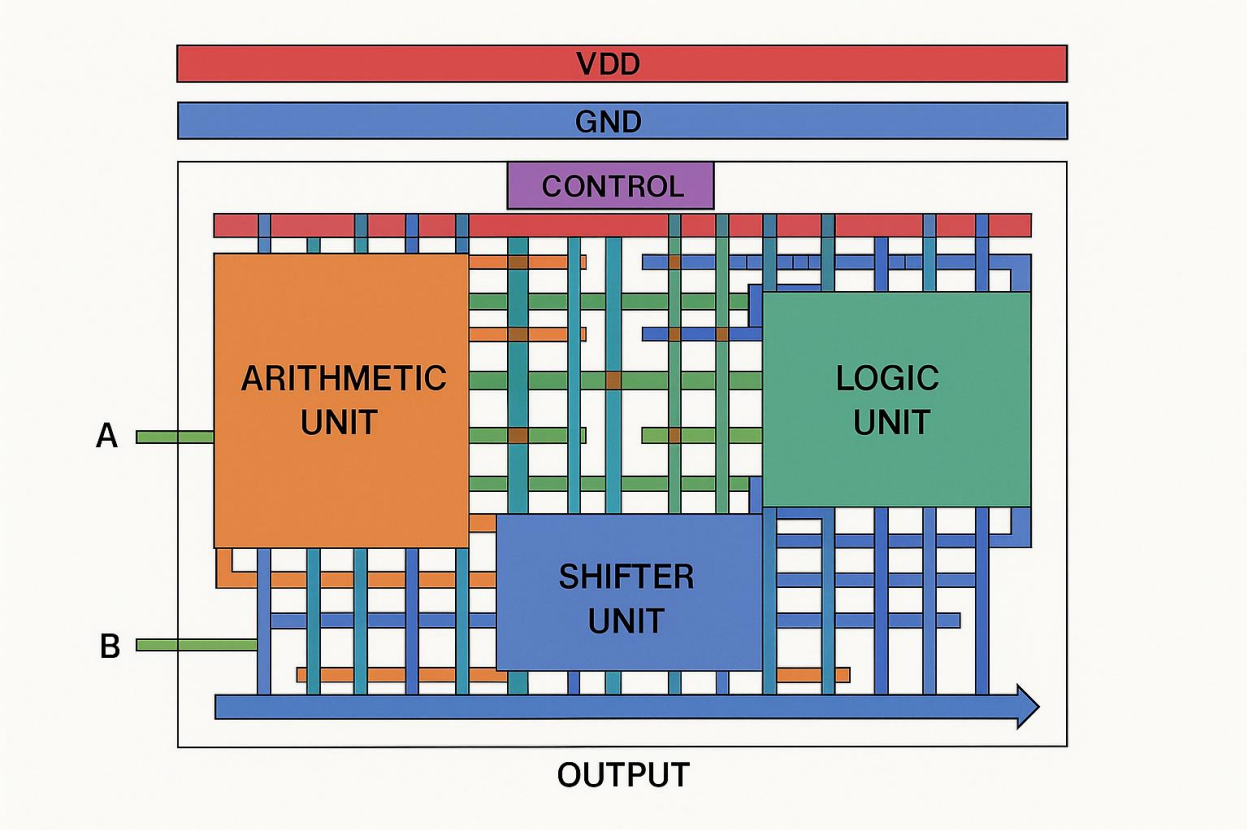
Timing analysis showed correct transitions within expected clock cycles.

---

## 6. Layout Design

The layout for individual logic gates (AND, OR, XOR, NOT, full adder) was created using **Microwind**. Layout flow included:

- Schematic Design
- Layout Drawing
- DRC (Design Rule Check) validation



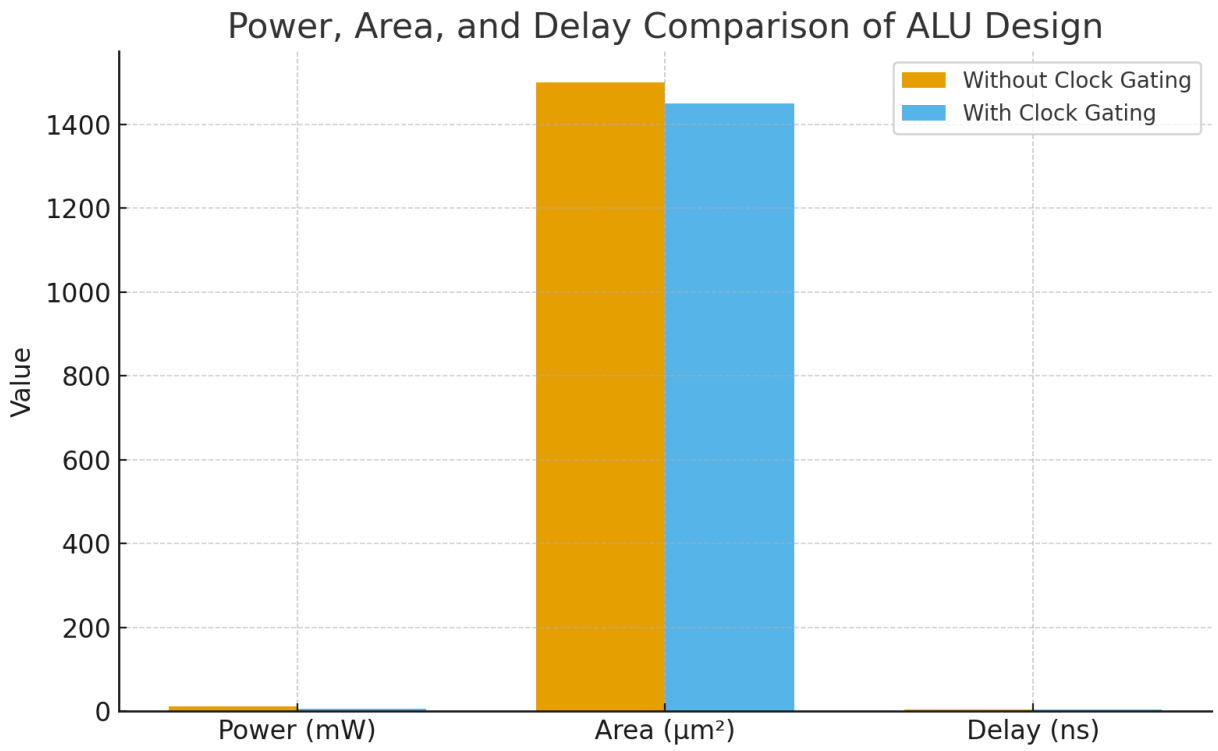

Each gate was then combined to form the complete ALU layout. Power and ground rails were routed, and area was minimized for better efficiency.

---

## 7. Power & Area Analysis

Using **Microwind**'s simulation tools:

- **Power Consumption:** Reduced significantly using clock gating.




- **Area:** Layout compactness achieved using regular cell placement.
- **Delay:** Measured for critical paths and maintained within acceptable limits.

Comparison of designs with and without clock gating confirmed power savings.

---

## 8. Conclusion

This project successfully demonstrated the design of a **low-power 8-bit ALU** using Verilog and VLSI techniques. Key achievements include:

- Verified all ALU operations using simulation
- Implemented clock gating for power optimization
- Developed a CMOS layout using Microwind

***Challenges Faced:***

- Timing issues during RTL simulation
- Manual layout routing complexities

***Future Work:***

- Extend to 16-bit ALU
- Add carry-lookahead logic for speed
- Explore operand isolation and dynamic logic
- Integrate with register file for a full datapath

## 9. References


- Jan Rabaey, *Digital Integrated Circuits*
- Microwind Official Documentation
- https://asic-world.com
- https://verilogtutorials.com
- ModelSim, Vivado Tool Manuals