Design Document: Parameterized ALU in Verilog

1. Introduction

The Arithmetic Logic Unit (ALU) is a key component of digital systems, responsible for executing arithmetic and logic operations. This project implements a parameterized ALU in Verilog, designed to support a wide range of operations with flexible data width and opcode size. It also includes delayed response for operations.

---

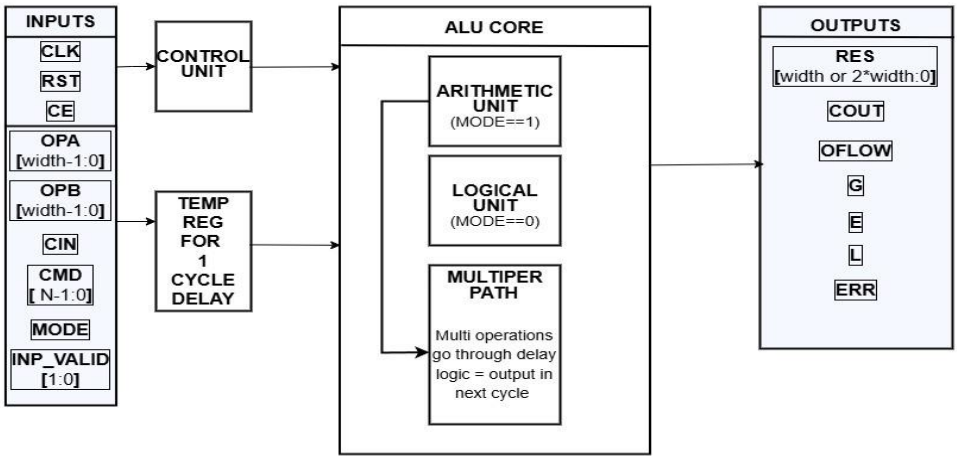
2. Objectives

- To design a fully parameterized ALU in Verilog.
- To implement arithmetic (ADD, SUB, etc.), logical (AND, OR, etc.), shift/rotate operations, and special commands (INC_MUL, SHIFT_MUL).
- To introduce 2 cycle delay for multiplication-based operations.
- To introduce 1 cycle delay for rest of the operations.
- To develop a robust testbench that validates all operations using a stimulus file.

---

3. Architecture Overview

The ALU is designed with the following modular blocks:



3.1 Input Unit

- **OPA, OPB:** Primary input operands.
- **INP_VALID:** Indicates which inputs are valid (e.g., A only, B only, both).
- **CMD:** 4-bit opcode to select the operation.
- **MODE:** Selects between arithmetic (MODE=1) and logical (MODE=0) operation.
- **CIN:** Carry input for some arithmetic ops.
- **CE, CLK, RST:** Control signals.

3.2 Control Unit

- Decodes the opcode and input validity.
- Directs the data to the arithmetic, logic, or shift sub-blocks.

3.3 Computation Unit

- **Arithmetic Operations:** ADD, SUB, CMP, SIGN_ADD, SIGN_SUB, etc.
- **Logical Operations:** AND, OR, XOR, NOT, etc.
- **Shift/Rotate:** SHL, SHR, ROL, ROR.
- **Multiplication Logic:** Supports INC_MUL and SHIFT_MUL with 2-cycle delayed output using a pipeline register (`res_delay`).

3.4 Output Unit

- **RES:** Final result output, with width depending on `MUL` mode.
- **Flags:** COUT, OFLOW, G, E, L, ERR.

---

4. Working Principle

- The **input stage** latches data on the clock's rising edge when `CE` is high.
- Based on `MODE` and `INP_VALID`, the **control logic** enables the corresponding block (arithmetic or logic).
- **Arithmetic operations** are computed combinationally with support for overflow and carry-out.
- **Logic operations** are applied using bitwise operators or shift/rotate instructions.
- **Status flags** (`E`, `G`, `L`) are updated based on comparison operations.
- Multiplication operations use a pipeline to introduce 2-cycle delay (`res_delay`).
- Results are **synchronously latched** to the output port (`RES`) along with corresponding flags.
- If `CE=0`, outputs are cleared.

---

5. Delay Handling

- **1-cycle delay** for all regular operations.
- **2-cycle delay** for `INC_MUL` and `SHIFT_MUL`:
  - Result is computed in combinational block and held in `res_delay`.
  - On the next clock, `res_delay` is assigned to `RES`.

---

6. Testbench Structure

- Reads binary test vectors from `stimulus.txt`.
- Each vector includes:
  - Inputs (OPA, OPB, CMD, etc.)
  - Expected output (RES, FLAGS)
- Compares simulation output with expected result using a **scoreboard**.
- Reports **PASS/FAIL** in `results.txt`.

---

7. Simulation Result

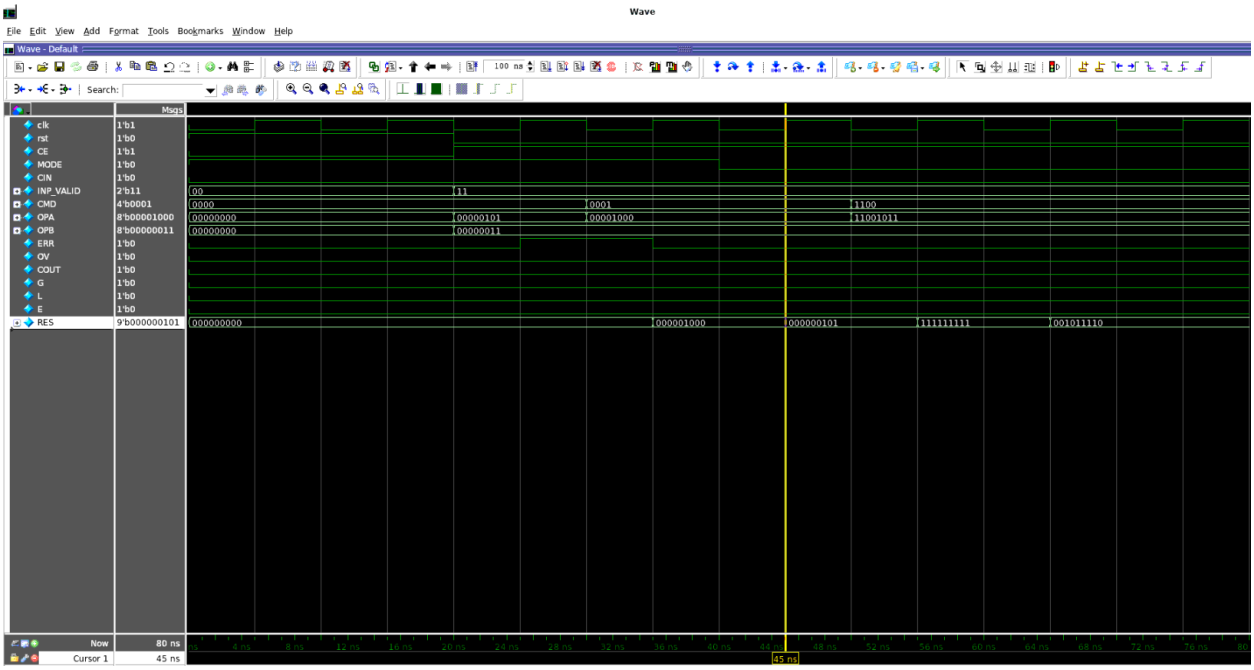


Figure 1: Output waveform for normal arithmetic operation

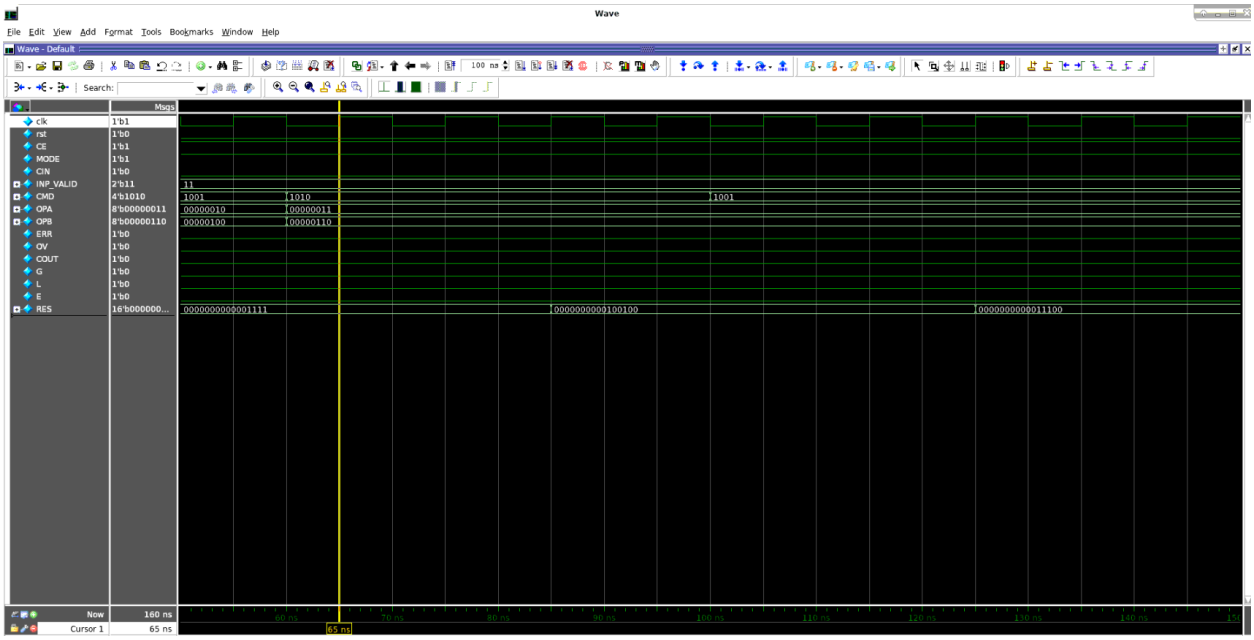


Figure 2: Output waveform for multiplication operation

- Waveform confirms:
  - Correct timing behavior (delays)
  - Accurate RES output and flag updates
  - Proper error signaling for invalid commands

---

8. Conclusion

The Verilog ALU design:

- Successfully supports arithmetic and logic operations.
- Properly handles delay for multiply operations.
- Performs accurately under extensive simulation-based testing.
- Is modular and synthesizable.

---

9. Future Improvements

- Add pipelined execution for throughput.
- Extend opcode width to include more functions.
- Implement signed/unsigned division.
- Integrate formal verification and coverage analysis.
