ALU PROJECT

SUBMITTED BY:
SWATHI RATHOD
EMP ID-6116

INTRODUCTION

An Arithmetic Logic Unit (ALU) is a fundamental component of digital systems, responsible for executing arithmetic and logic operations on binary data. This project focuses on the design and implementation of a parameterized ALU capable of performing a wide range of arithmetic, logical, shift, and rotate operations. The ALU is designed with flexibility in mind, supporting both signed and unsigned operations based on configurable modes and commands.

The ALU accepts two primary input operands OPA and OPB, along with control signals such as CLK, RST, CE, INP_VALID, MODE, and CMD, which tells the type and validity of the operations to be performed. The design incorporates functionality for unsigned addition, subtraction, logical operations (AND, OR, XOR, NOT, etc.), shift operations, and advanced rotate instructions. It also includes comparator outputs (Greater than, Less than, Equal), overflow detection, and error flagging based on operand constraints

OBJECTIVES

- To develop a parameterized ALU design
- Support essential arithmetic operations (addition, subtraction, increment, decrement, etc.) and logical operations (AND, OR, XOR, NOT, etc.).
- Include advanced functions such as shift, rotate, and comparison operations.
- Utilize a MODE signal to switch between arithmetic and logical operation modes effectively.
- Design the ALU to accept external control signals (CLK, RST, CE, INP_VALID, and CMD) and generate output flags (COUT, OFLOW, G, L, E, ERR) to indicate operation status.
- To generate essential output results and status flags

The ALU is designed to interpret control signals to dynamically switch between operation modes and execute commands based on a 4-bit input. It also incorporates operand validation, comparator logic, and the generation of status flags such as carry-out, overflow, and error detection.

ARCHITECTURE

The ALU (Arithmetic Logic Unit) architecture is designed to efficiently handle a wide range of arithmetic and logical operations using a modular and configurable structure. The system consists of three main blocks: the Input Unit, Processing Unit, and Output Unit, coordinated by control signals and a multiplexer for operation selection.

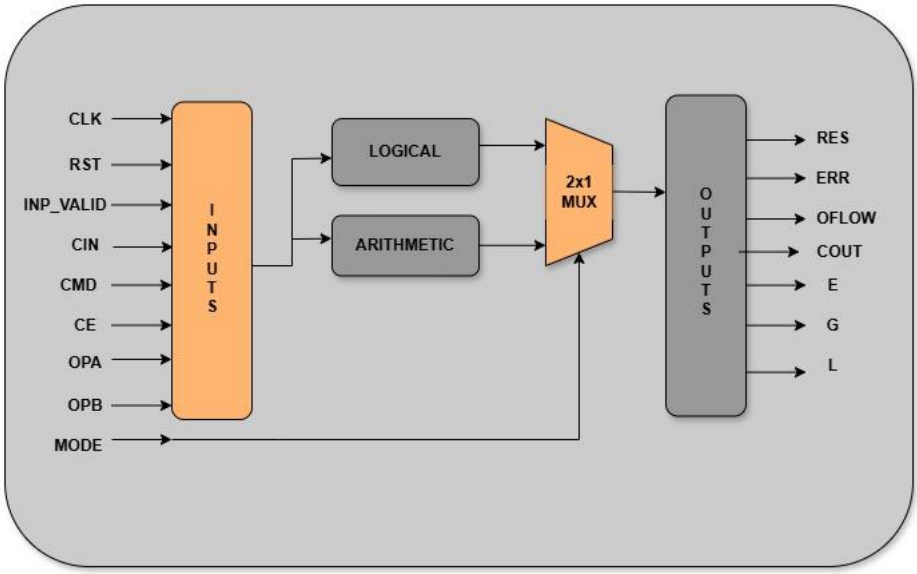


Figure 1:Design Architecture.

Input Unit

This unit receives all necessary control and data signals including,Clock (CLK),Reset (RST),Operand Validity (INP_VALID),Carry-in (CIN),Command (CMD),Clock Enable (CE),Operands (OPA, OPB),Mode Selector (MODE)

2. **Processing Unit**

This section contains two core functional sub-blocks:

Arithmetic Block: Executes operations such as addition, subtraction, increment, decrement, carry-based operations, and other complex arithmetic functions.

Logical Block: Performs bitwise operations like AND, OR, XOR, NOT, NAND, NOR, etc., as well as shifting and rotating data.

The MODE signal is used to select between the arithmetic and logical blocks using a 2×1 multiplexer (MUX). If MODE = 1, the ALU operates in arithmetic mode; if MODE = 0, it switches to logical mode.

Output Unit

The selected result from the MUX is sent to the output block, which delivers:

- RES: Final result of the operation
- ERR: Error signal for invalid command or input conditions
- OFLOW: Overflow flag for arithmetic operations
- COUT: Carry-out signal from arithmetic operations
- G, L, E: Comparator outputs indicating if Operand A is greater than, less than, or equal to Operand B

WORKING

The ALU operates based on the combination of input control signals and operand values, executing either arithmetic or logical operations depending on the mode selection. The core functionality is driven by a clocked process, and the result is generated after interpreting the command and processing the inputs accordingly.

Upon receiving a valid clock (CLK) and with the clock enable (CE) signal asserted, the ALU begins its operation. The RST signal, when active, resets all internal states and output signals to their default values. Operand validity is checked using the INP_VALID signal to ensure proper data is available for computation.

The MODE signal determines the type of operation:

When MODE = 1, the ALU performs arithmetic operations such as addition, subtraction, increment, decrement, and other extended functions using CMD.

When MODE = 0, it executes logical operations such as AND, OR, XOR, NOT, as well as shift and rotate operations.

A 4-bit CMD input selects the specific operation to perform. For arithmetic operations, the operands OPA and OPB are used along with optional CIN to compute the result, with status flags like carry-out (COUT), overflow (OFLOW), and comparator outputs (G, L, E) being updated accordingly. For logical operations, the result is directly obtained from bitwise manipulations or shifts, and in case of invalid operand configurations (e.g., incorrect rotate settings), the ERR flag is raised. A 2x1 multiplexer internally selects the output from either the arithmetic or logical block based on the MODE signal

RESULTS

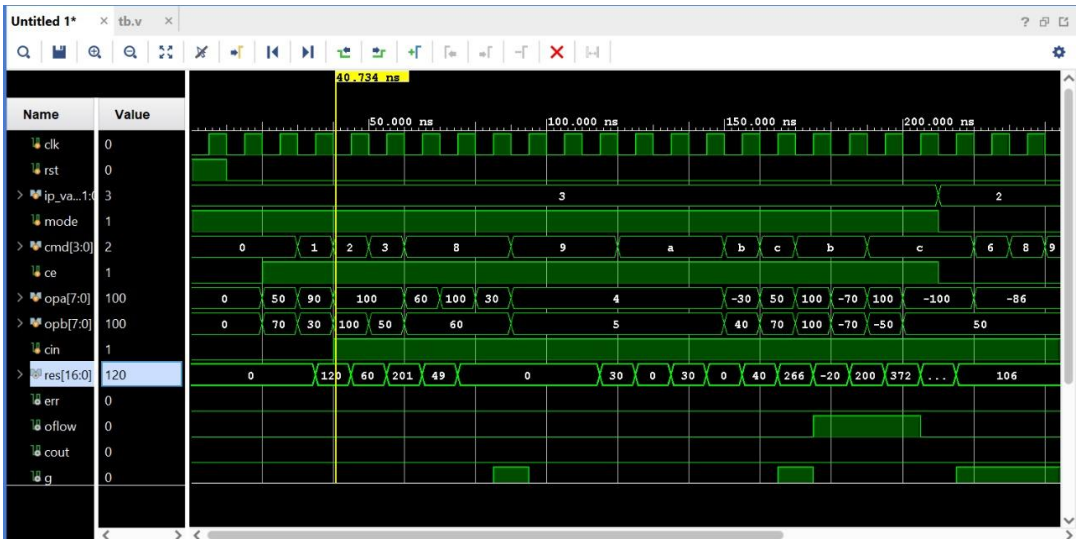
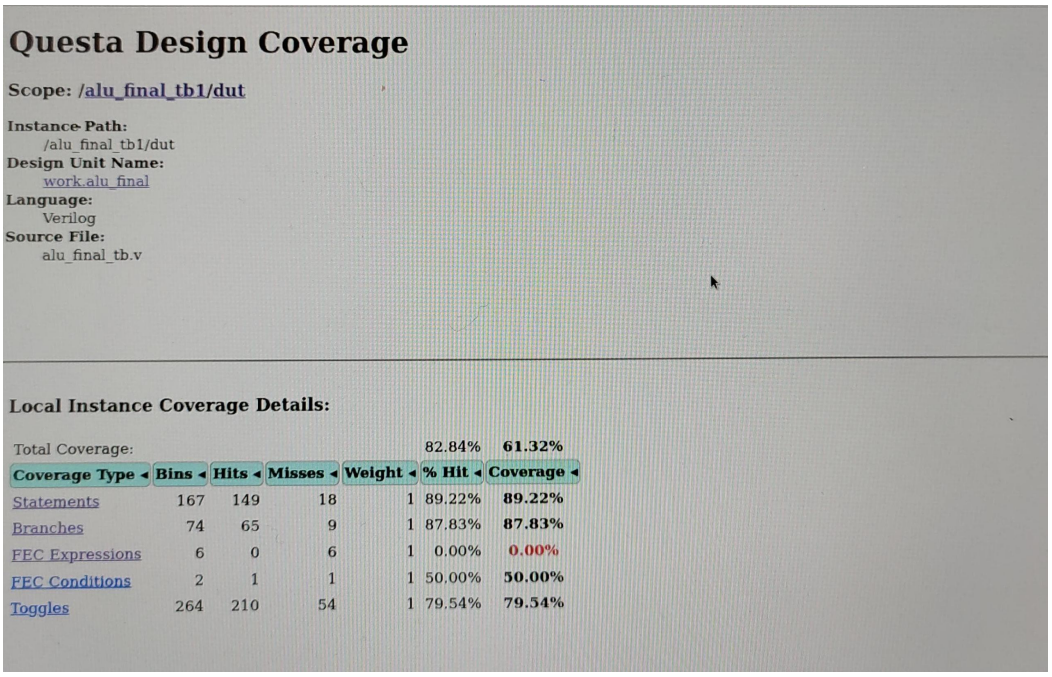


Figure 2:Waveform of different operations.



Questa Design Coverage

Scope: /alu_final_tb1/dut

Instance Path:
/alu_final_tb1/dut
Design Unit Name:
work.alu_final
Language:
Verilog
Source File:
alu_final_tb.v

Local Instance Coverage Details:

Total Coverage:					82.84%	**61.32%**
Coverage Type	**Bins**	**Hits**	**Misses**	**Weight**	**% Hit**	**Coverage**
Statements	167	149	18	1	89.22%	**89.22%**
Branches	74	65	9	1	87.83%	**87.83%**
FEC Expressions	6	0	6	1	0.00%	**0.00%**
FEC Conditions	2	1	1	1	50.00%	**50.00%**
Toggles	264	210	54	1	79.54%	**79.54%**

Figure 3:Code coverage.

CONCLUSION

The design and implementation of the Arithmetic Logic Unit (ALU) were successfully completed, meeting all functional and architectural objectives. The ALU supports a wide range of arithmetic and logical operations, including addition, subtraction, shifting, rotation, and bitwise functions, with efficient handling of control signals and operand validation. It also includes robust error detection and status flag generation such as carry-out, overflow, and comparator outputs.

The parameterized design offers flexibility in operand width, making it adaptable for various applications in digital systems and embedded platforms. Through simulation and waveform analysis, the ALU's behavior was verified for correctness across different input scenarios. Overall, this project demonstrates the successful integration of digital design principles, modular hardware architecture, and effective use of SystemVerilog to build a reliable and scalable ALU module.