ALU Verification Plan

CHAPTER 1 - PROJECT OVERVIEW AND OBJECTIVES

1.1 PROJECT OVERVIEW

Arithmetic logic unit (ALU) is a combinational digital circuit that performs arithmetic and bitwise operations on integer binary numbers. It is responsible for carrying out arithmetic operations such as addition and subtraction, as well as logical operations like AND, OR, and NOT. This project presents the implementation of a parameterized ALU designed using syst em verilog. The ALU supports a wide range of operations including arithmetic, logical, shift, and rotate instructions. The goal is to create a modular, synthesizable, and simulation-verified ALU that can serve as a reliable component in larger digital design projects.

1.2 VERIFICATION OBJECTIVES

The main objective of verifying the ALU is to check that it performs all its supported operations correctly. This includes making sure that arithmetic operations like addition, subtraction, and increment, as well as logical operations like AND, OR, and NOT, produce the correct output for all types of inputs. It is also important to check that the output flags—such as overflow, carry, and comparison flags (greater, less, equal)—are set properly depending on the operation. Additionally, the goal is to test the ALU under different conditions, such as changing inputs quickly or giving the same input in different ways, to ensure the design is stable and works in all possible situations. Apart from functional checks, the verification also includes making sure that all types of operations are tested (functional coverage) and that written checks (assertions) are triggered correctly when something goes wrong.

1.3 DUT INTERFACES

The alu interface defines a SystemVerilog interface that encapsulates all input and output signals between the testbench and the DUT (Design Under Test). This interface ensures clean, modular, and synchronized communication using clocking blocks and modports.

The ALU interface include:

INPUTS:

• OPA, OPB: Operand inputs (parameterized width N)
• CMD: Operation command (4 bits to select the desired ALU operation)
• INP_VALID: Indicates which operands are valid (00, 01, 10, 11)
• MODE: 1 for arithmetic, 0 for logical operations
• CIN: Carry input used in arithmetic operations
• CLK: Clock signal, positive edge triggered
• RST: Asynchronous reset
• CE: Clock enable

OUTPUTS:

• RES: Result of the ALU operation
• COUT: Carry-out from addition/subtraction
• OFLOW: Indicates overflow
• ERR: Indicates invalid conditions (e.g. wrong rotation inputs)
• G, L, E: Comparison flags (Greater, Less, Equal)

MODPORTS:

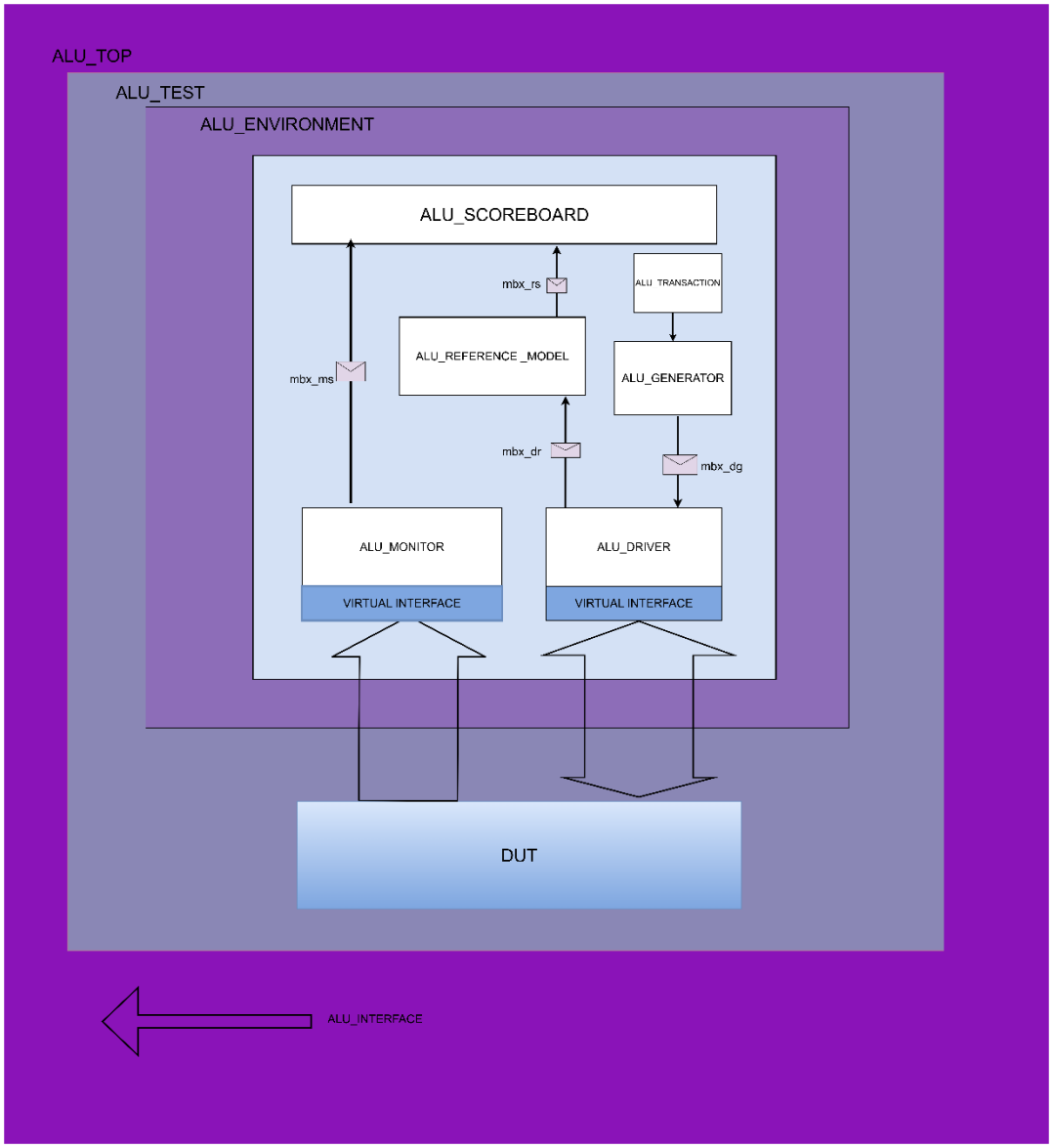
The modport groups and specifies the direction to the signals specified within the interface.

CLOCKING BLOCK:

A clocking block in SystemVerilog defines the timing and synchronization for signal interaction between the testbench and DUT. It specifies a clock event as a reference, the signals to be sampled or driven, and the timing of these actions relative to the clock. This helps ensure stable and predictable communication between the testbench and DUT.

CHAPTER 2 TESTBENCH ARCHITECTURE AND METHODOLOGY

2.1 TESTBENCH ARCHITECTURE



1. Top Module – TEST

The TEST module acts as the top-level wrapper that instantiates the test environment, connects it to the DUT, and manages simulation control.

2. Environment

The environment contains all the essential components such as the driver, monitor, generator, scoreboard, reference model, and communication mailboxes. These elements work together to provide stimulus, capture outputs, generate expected results, and perform verification comparisons.

3. Generator

The Generator creates transactions including fields like opa, opb, mode, cmd, inp_valid, ce, cin, etc. It sends the transactions to the Driver via the mailbox mbx_gd.

4. Driver

The Driver receives the randomized values from the generator and drives it to the DUT through the interface. It also sends the values to the reference model through mailbox.

5. Monitor

The Monitor observes and captures the DUT outputs such as res, cout, oflow, err, g, l, and e. It sends this data to the Scoreboard (via mbx_ms) for comparison.

6. Reference Model

It receives transactions and performs the expected operation, and sends the results to the Scoreboard.

7. Scoreboard

The Scoreboard performs comparison between actual DUT outputs and the expected outputs from the Reference Model. It logs pass/fail information for each transaction and reports mismatches.

8. Mailboxes

Mailboxes provide a transaction-level communication channel between various testbench components. They include:

• mbx_gd: Generator to Driver

• mbx_dr: Driver to Reference Model

• mbx_rs: Reference Model to Scoreboard

• mbx_ms: Monitor to Scoreboard

9. Design Under Test (DUT)

The DUT receives driven inputs from the Driver and sends responses back to the Monitor. It is treated as a black-box connected through virtual interfaces.

10.INTERFACE

An interface is a bundle of signals or nets through which a testbench communicates with a design. The interface construct is used to connect the design and testbench.

11.VIRTUAL INTERFACE

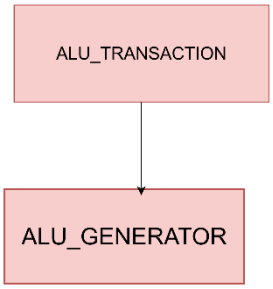
A virtual interface is a variable of an interface type that is used in classes to provide access to the interface signals. A virtual interface is a variable that represents an interface instance.

12.TRANSACTION

Transaction consists of ALU inputs that need to be randomized and outputs that are not randomized.

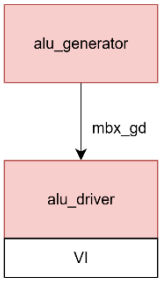
2.2 FLOWCHART OF SV COMPONENTS

1.Transaction class



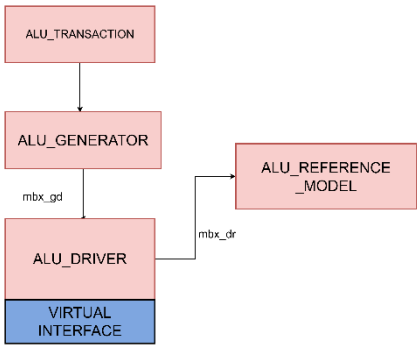
The transaction class consists of randomized inputs and non-randomized outputs.
It consists of constraints and a deep copy function for creating transaction copies.

2.Generator class



This component generates randomized input values and is sent to the driver through a mailbox(mbx_gd).

3.Driver class

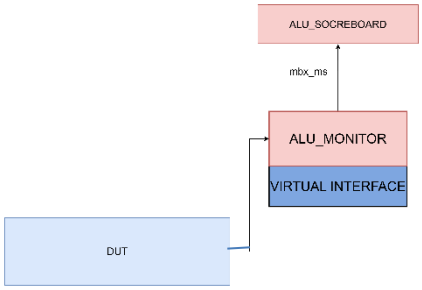


It consists of two mailbox:

Generator to driver mailbox(mbx_gd): It transfers randomized transactions from the generator to the driver
Driver to reference model mailbox(mbx_dr): It sends the same transactions to the reference model

The task applies the stimulus to the DUV based on the received transaction

4.Monitor class



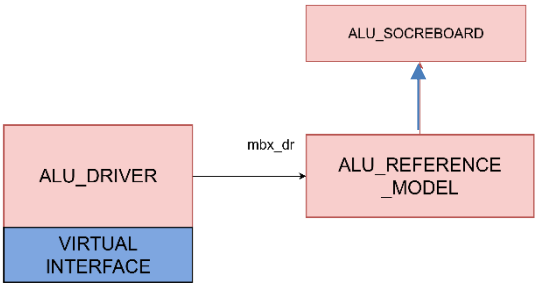
The ALU Monitor observes the DUT's outputs and converts them into transaction objects for validation.

It consists of one mailbox:

Monitor to Scoreboard Mialbox: It transfers captured output transactions to the scoreboard

Detects valid output transactions, samples the result, and sends it as a transaction object to the scoreboard

5.Reference model



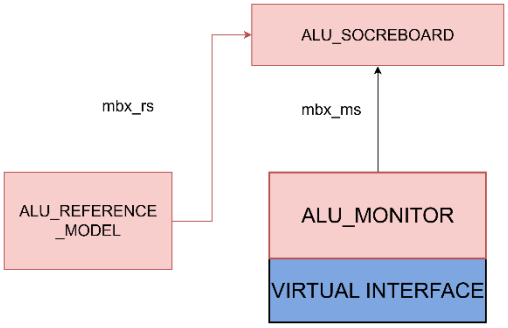
Acts as the reference for result validation. It processes the same inputs as the DUV and produces the expected outputs.

It consists of two Mailboxes:

Driver to reference model mailbox: It receives transactions from the driver

Reference to scoreboard mailbox: It sends computed expected results to the scoreboard

6.Scoreboard



The scoreboard verifies the correctness of the DUV by comparing its outputs with the reference model's expected results.

It consists of two mailbox:

Reference model to scoreboard mailbox : It receives the expected results from the reference model

Monitor to Scoreboard mailbox: It receives actual results from the monitor

Compares the actual and expected results, and flags mismatches and generates final statistics, including pass/fail counts and functional coverage details.

CHAPTER 3 VERIFICATION RESULTS AND ANALYSIS

3.1 DESIGN BUGS

1. 16-Clock Cycle Timeout Logic

Error flag is not set even after waiting for 16 clock cycles when the expected timeout condition occurs.

2. Single-Operand Operations

DUT does not perform increment/decrement operations when only the corresponding input valid signal is high. It only works when INP_VALID == 2'b11, which violates the design specification.

3. Carry-Out Logic for Increment Operations

COUT logic is incorrect or missing.

It remains at the default value.

COUT flag should assert when operand reaches the maximum value (e.g., 255 incrementing to 256).

4. Increment A Operation

INC_A operation holds the same value as OPA, instead of incrementing it.

This results in no change to the operand value.

5. Increment B and Decrement B Operations

INC_B operation incorrectly decrements the value instead of incrementing.

6. Shift Operation

Right shift operations on both OPA and OPB fail.

Left shift OPB operation also fails to execute correctly.

7. Decrement B Operations

DEC_B operation incorrectly increments the value instead of decrementing.

8. Overflow Logic for Decrement Operations

OVER_FLOW logic is incorrect or missing.It remains at the default value.

OVER_FLOW flag should assert when operand at minimum value (e.g., 0) is decremented to -1.

9. Carry-In Signal Timing

CIN signal appears with a one-clock-cycle delay in addition to CIN and subtraction with CIN operations.

This causes timing misalignment.

10. Rotate Right A by B Error Condition

ROR_A_B error condition detection is incorrect and always remains zero.

It fails to set appropriate error flags even when an error condition occurs.

11. Multiplication Operation

Issues exist in the multiplication functionality that affect correct arithmetic results.

12. Logical OR

Logical OR operation fails to execute correctly even with valid inputs and proper timing.

3.2 COVERAGE REPORT:

Questa Coverage Report

Number of tests run:	1
Passed:	1
Warning:	0
Error:	0
Fatal:	0

List of tests included in report...

List of global attributes included in report...

List of Design Units included in report...

Coverage Summary by Structure:

Design Scope	Hits %	Coverage %
test_bench_alu	86.47%	95.06%
read_stimulus	100.00%	100.00%
driver	100.00%	100.00%
global_init	100.00%	100.00%
monitor	100.00%	100.00%
score_board	100.00%	100.00%
dut_reset	100.00%	100.00%
gen_report	100.00%	100.00%
dut	95.35%	97.14%

Coverage Summary by Type:

Total Coverage:					86.47%	95.06%
Coverage Type	Bins	Hits	Misses	Weight	% Hit	Coverage
Statements	264	264	0	1	100.00%	100.00%
Branches	80	79	1	1	98.75%	98.75%
FEC Expressions	4	4	0	1	100.00%	100.00%
Toggles	924	753	171	1	81.49%	81.49%

Report generated by Questa (ver. 10.6c) on Sun 08 Jun 2025 10:49:10 PM IST with command line:
vcover report -html test_bench_alu.ucdb -htmldir covReport

FUNCTIONAL COVERAGE:

OUTPUT COVERAGE-

Covergroup type:

c2

Summary	Total Bins	Hits	Hit %
Coverpoints	524	473	90.26%
Crosses	0	0	0.00%

Search:

CoverPoints	Total Bins	Hits	Misses	Hit %	Goal %	Coverage %
COUT	2	2	0	100.00%	100.00%	100.00%
E	2	2	0	100.00%	100.00%	100.00%
ERR	2	1	1	50.00%	50.00%	50.00%
G	2	2	0	100.00%	100.00%	100.00%
L	2	2	0	100.00%	100.00%	100.00%
OFLOW	2	2	0	100.00%	100.00%	100.00%
RES	512	462	50	90.23%	90.23%	90.23%

Scope: /alu_pkg/alu_driver

Covergroup type:

c

Summary	Total Bins	Hits	Hit %
Coverpoints	28	22	78.57%
Crosses	101	40	39.60%

Search:

CoverPoints	Total Bins	Hits	Misses	Hit %	Goal %	Coverage %
CIN_CP	2	2	0	100.00%	100.00%	100.00%
CMD_CP	14	11	3	78.57%	78.57%	78.57%
INP_VALID_CP	4	1	3	25.00%	25.00%	25.00%
MODE_CP	2	2	0	100.00%	100.00%	100.00%
OPA_CP	3	3	0	100.00%	100.00%	100.00%
OPB_CP	3	3	0	100.00%	100.00%	100.00%

3.3 OUTPUT WAVEFORM