ALU Verification Plan

N.Gowthaman

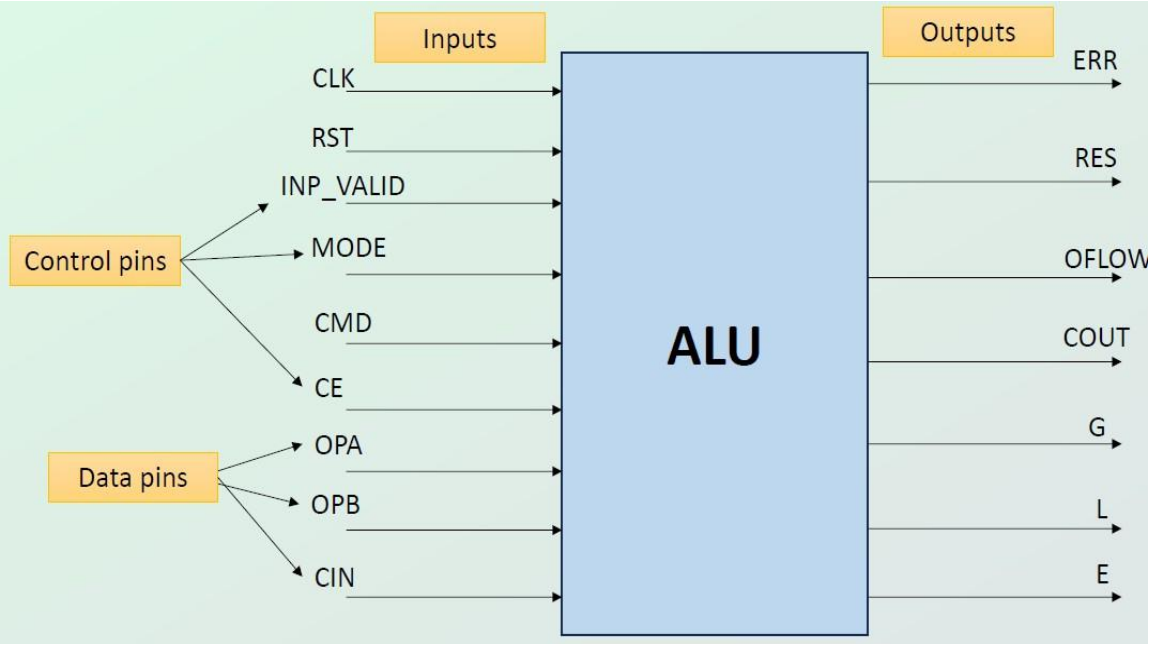
EMP ID:6086

VERIFICATION DOCUMENT- ALU

CHAPTER 1 – DESIGN OVERVIEW

1. ALU:

The Arithmetic Logic Unit (ALU) is a fundamental combinational circuit that performs essential arithmetic and logical operations within a digital system. In this design, the ALU is implemented using Verilog, a hardware description language widely used for modelling digital systems.



This pin-out diagram provides a structural overview of how the ALU interfaces with external components. The design features clearly categorized inputs and outputs to enable smooth data flow and operation control. Control pins manage the operation mode and synchronization, while data pins provide the operands required for computation. The outputs reflect the results of the operations along with relevant status indicators such as comparison and error flags.
By structuring the ALU in this way, the design ensures modularity, reusability, and ease of integration into larger digital systems such as CPUs and signal processors.

1.2 Advantages of ALU:

- Does math problems like add, subtract, multiply, divide.
- Used to make logical decisions like AND, OR, NOT.
- Used to compares numbers which is bigger, smaller, or equal.
- It is fast accurate and occupies less space inside processor.

1.3 Disadvantages of ALU:

- It has no memory and hence cannot remember previous calculations.
- Cannot work faster than the processor allows.
- Supports only specific bit widths such as 32-bit and 64-bit; larger data sizes are truncated during processing..
- Power consumption increases rapidly during high-complex calculations

1.4 Use cases of ALU:

The ALU is a core component in digital systems and processors. Key use cases include:

- **Microprocessors and Microcontrollers** – Executes arithmetic and logic instructions (e.g., add, subtract, AND, OR).
- **DSPs (Digital Signal Processors)** – Used for filtering, FFT, and signal transformations.
- **Embedded Systems** – Performs calculations for automation and control logic.
- **Cryptographic Units** – Handles bitwise and modular arithmetic for encryption algorithms.
- **AI Accelerators / SIMD Units** – ALU arrays process matrix/vector operations efficiently.

1.5 Project Overview of ALU:

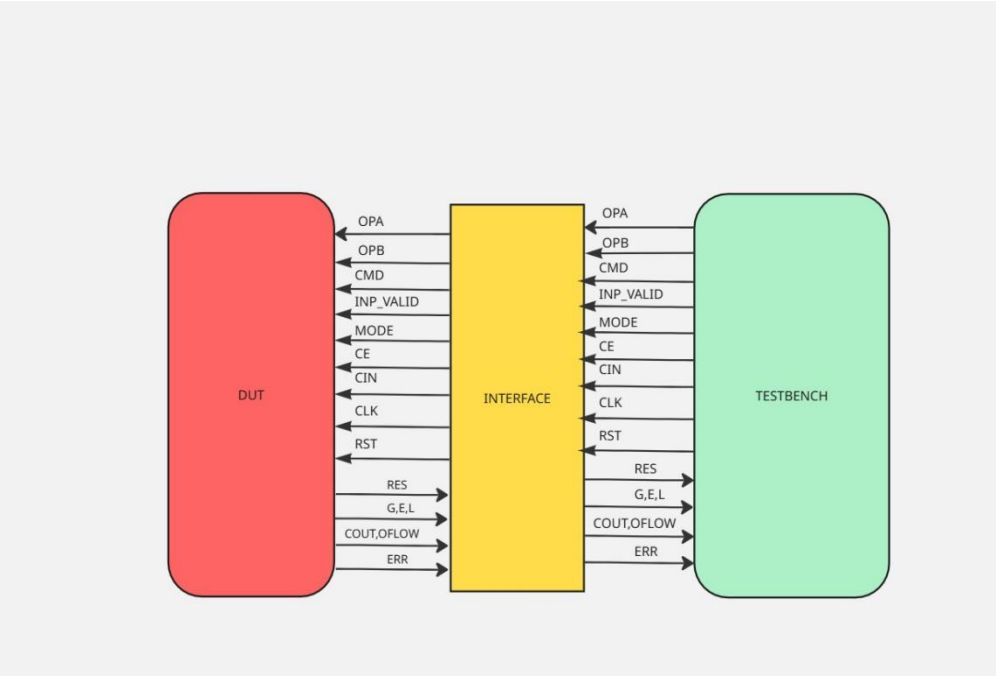
1.This project implements a parameterized ALU in Verilog that supports a variety of operations including arithmetic, logical, comparison, and bitwise shift/rotation. The ALU design is tested using a SystemVerilog testbench with functional coverage and assertions.

2. the design ensures modularity, reusability, and ease of integration into larger digital systems such as CPUs and signal processors.

1.6 Design Features:

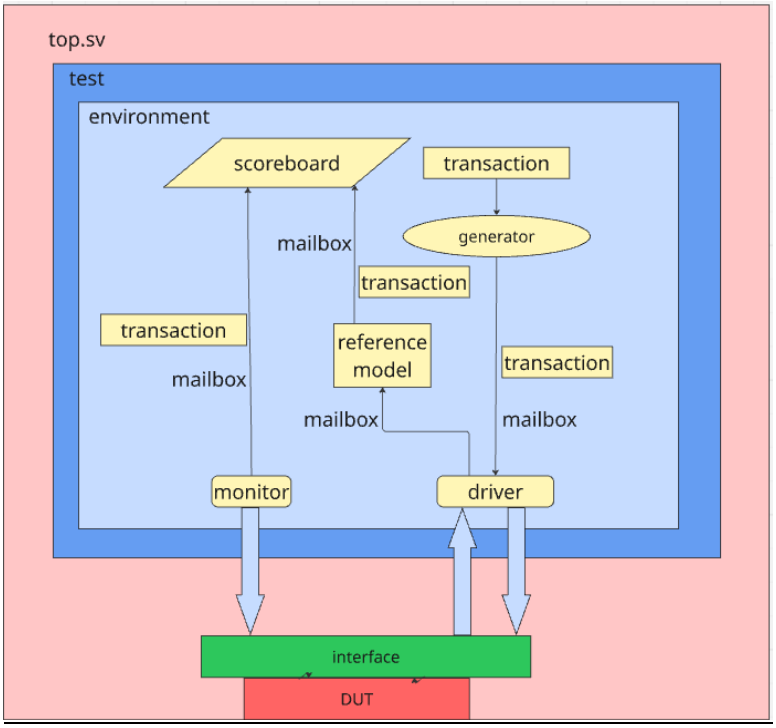
- Arithmetic
- Addition ,subtraction ,Comparision,Increment,Decrement,Multiplication
- Logic
- AND,NAND,OR,NOR, XOR,XNOR, NOT
- Others
- Rotate right/left , Shift right/left

1.7 Design diagram with interface signals:



CHAPTER 2 - Verification Architecture

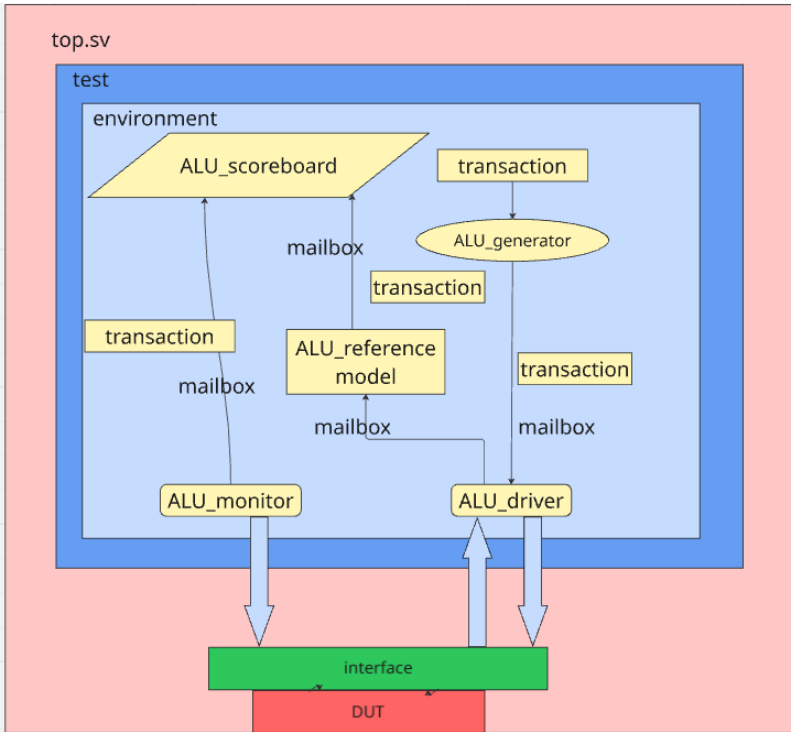
2.1 Verification Architecture :



This diagram shows a typical testbench setup used to verify digital designs.

- The testbench environment includes several parts:
- A generator creates input test cases (transactions), often using random values.
- These inputs go to the driver, which converts them into signals and sends them to the DUV through a virtual interface.
- The DUV's output is captured by a monitor, which passes the data to the scoreboard.
- The scoreboard checks if the output is correct by comparing it to the expected results from a reference model.
- All these components are organized within the environment block, which manages their interaction.
- This setup helps verify that the DUV works correctly by comparing actual and expected behavior.

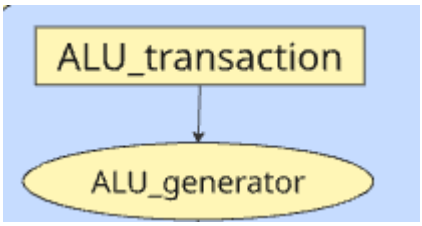
2.2 Verification Architecture for ALU:



2.3 FLOW CHART OF SV COMPONENTS :

1.Transaction class

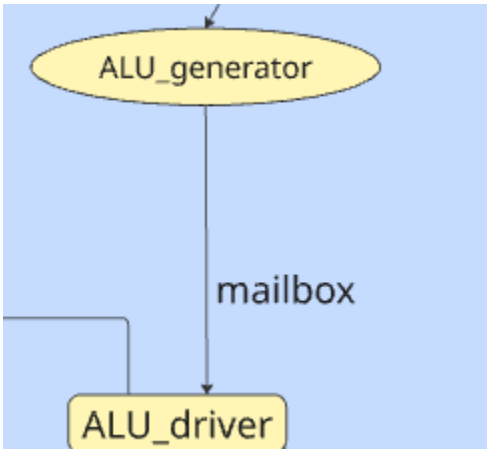
The transaction class encapsulates the input stimuli and output responses.



Deep Copy Method

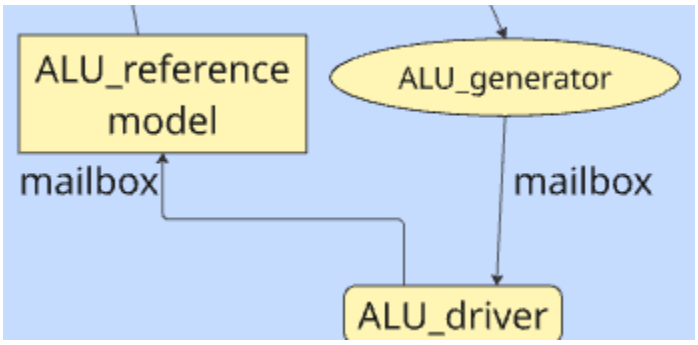
Implements a copy() function following the blueprint pattern for creating independent transaction copies used across testbench components.

2.Generator Class: -



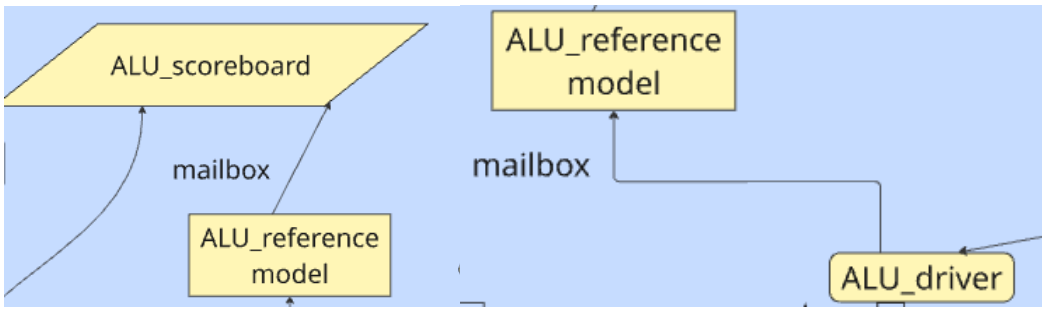
The generator consists of the ALU transaction class handle, the mailbox handle which connects to the driver, and the start() task which randomizes transactions and sends the randomized transactions to the driver through the mailbox.

3.Driver Class: -



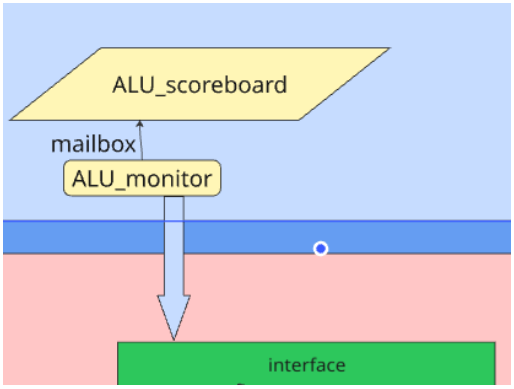
The driver class drives the input stimuli to the reference model.It receives the input stimuli generated by generator.It Communicates between testbench and DUV with proper synchronization.

4.Reference model



Replicates the DUT without delays. Continuously gets transactions from driver, computes expected results, and forwards to scoreboard.Implements golden reference for all ALU operations

5.Monitor:-



The ALU Monitor captures transactions from the DUV interface and forwards them to the scoreboard for result comparison.

6.Scoreboard:-

The ALU Scoreboard compares actual DUV results against expected results from the reference model to determine test pass/fail status.Compares expected vs actual results and reports mismatches