ALU Verification Document

Introduction

This document outlines the verification process for an Arithmetic Logic Unit (ALU) design, ensuring that the implemented functionality adheres to the specified requirements. The ALU is a critical component in digital systems, responsible for performing arithmetic and logical operations, and its correct operation is essential for overall system reliability.

The verification process involves a thorough review of the design specification, development of a comprehensive testbench, and execution of test cases to validate the ALU's functionality under various scenarios. The testbench will include randomized tests to cover all conditions.

Key Objectives

Key objectives of the project are :-

- Understanding the ALU design specification, including supported operations (e.g., addition, subtraction, AND, OR, XOR, shifts) and control signals.
- Developing a structured testbench using a hardware verification language (e.g., SystemVerilog) with assertions and coverage metrics.
- Executing functional verification to confirm that all operations produce expected results.
- Analyzing coverage (code, functional, and assertion coverage) to ensure all design aspects are thoroughly tested.

DUT INTERFACES

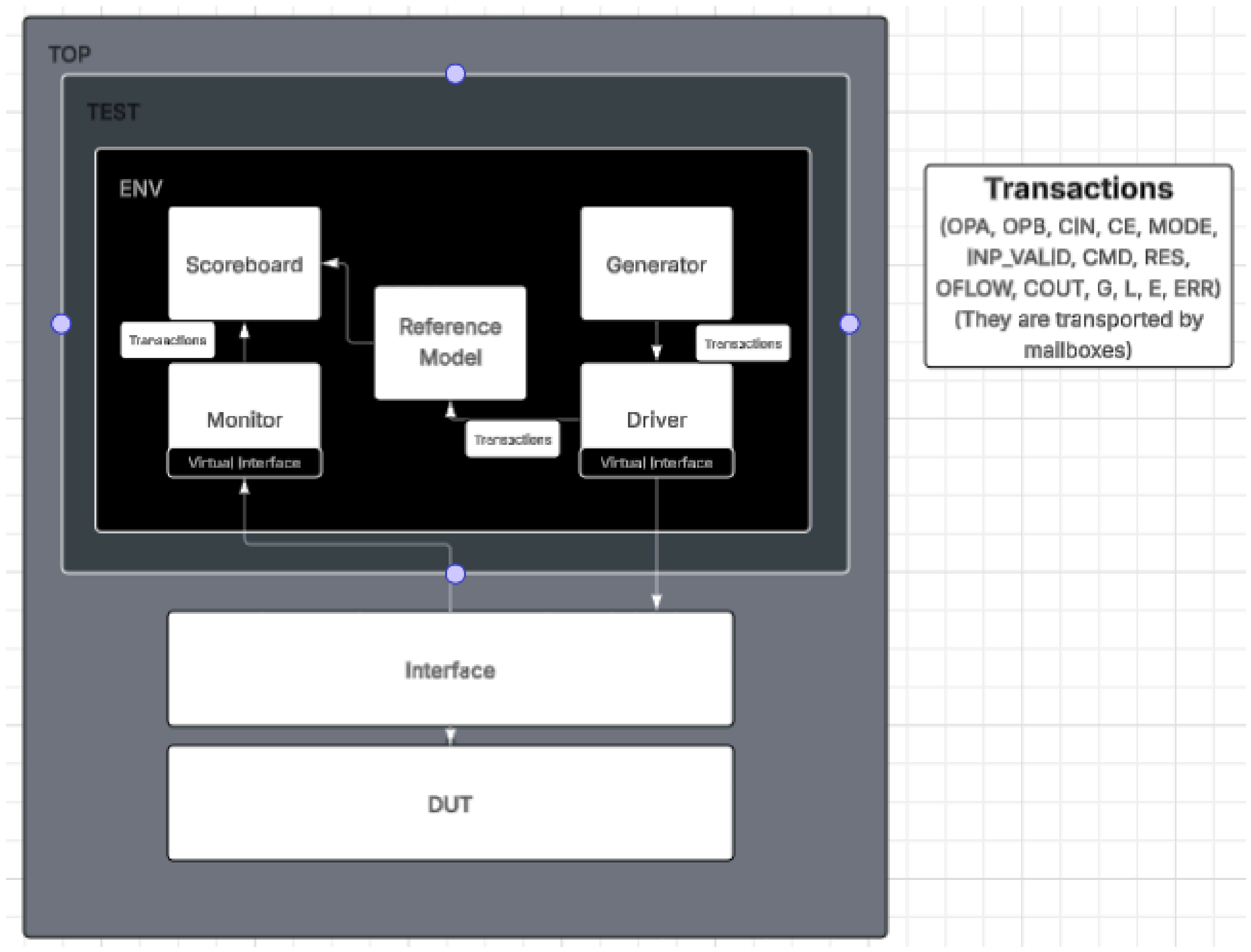
The DUT consists of the following input and output pins.

PIN name	Direction	Width	Description
OPA	INPUT	Parameterized	Parameterized operand 1
OPB	INPUT	Parameterized	Parameterized operand 2

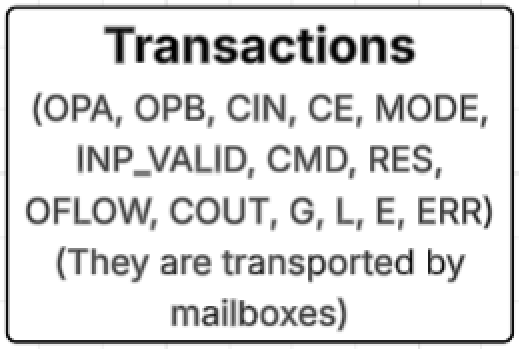
CIN	INPUT	1	This is the active high carry in input signal of 1-bit.
CLK	INPUT	1	This is the clock signal to the design and it is edge sensitive.
RST	INPUT	1	This is the active high asynchronous reset to the design.
CE	INPUT	1	This is the active high clock enable signal 1 bit.
MODE	INPUT	1	MODE signal 1 bit is high, then this is an Arithmetic Operation otherwise it is logical operation
INP_VALID INPUT	INPUT	2	Operands are valid as per below table :- 00 : No operand is valid. 01: Operand A is valid. 10: Operand B is valid. 11: Both Operands are valid.
CMD	INPUT	4	Selects the command to be executed.
RES	OUTPUT	Parameterized + 1	This is the total parameterized plus 1 bits result of the instruction performed by the ALU.
OFLOW	OUTPUT	1	This 1-bit signal indicates an output overflow, during Addition/Subtraction
COUT	OUTPUT	1	This is the carry out signal of 1-bit, during Addition/Subtraction

G	OUTPUT	1	This is the comparator output of 1-bit,which indicates that the value of OPA is greater than the value of OPB.
L	OUTPUT	1	This is the comparator output of 1-bit,which indicates that the value of OPA is lesser than the value of OPB/
E	OUTPUT	1	This is the comparator output of 1-bit,which indicates that the value of OPA is equal to the value of OPB.
ERR	OUTPUT	1	When Cmd is selected as 12 or 13 and mode is logical operation , if 4th ,5th ,6th and 7th bit of OPB are 1, then ERR bit will be 1 else it is high impedance .

Testbench Architecture



Transactions

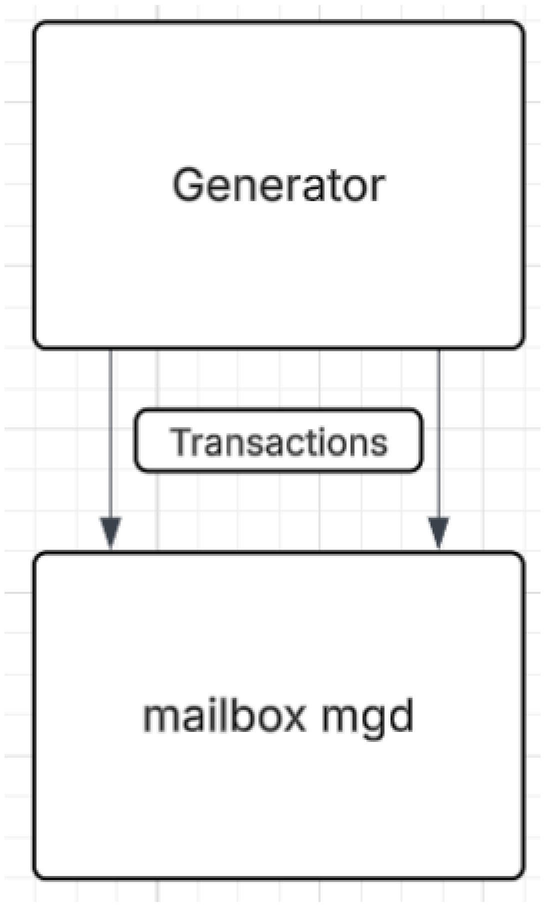

Transactions
(OPA, OPB, CIN, CE, MODE, INP_VALID, CMD, RES, OFLOW, COUT, G, L, E, ERR) (They are transported by mailboxes)

Transactions are class objects that encapsulate all input and output signals of the ALU. They serve as the primary data structure passed between testbench components (via mailboxes). They consist of the following signals.

1. Inputs
    - ➢ OPA, OPB (Operands)
    - ➢ CIN (Carry-in)
    - ➢ CE (Clock Enable)
    - ➢ MODE (Operation Mode)
    - ➢ CMD (Command/Opcode)

2. Outputs
    - ➢ RES (Result)
    - ➢ OFLOW (Overflow)
    - ➢ COUT (Carry-out)
    - ➢ G, L, E (Greater, Less, Equal flags)
    - ➢ ERR (Error flag

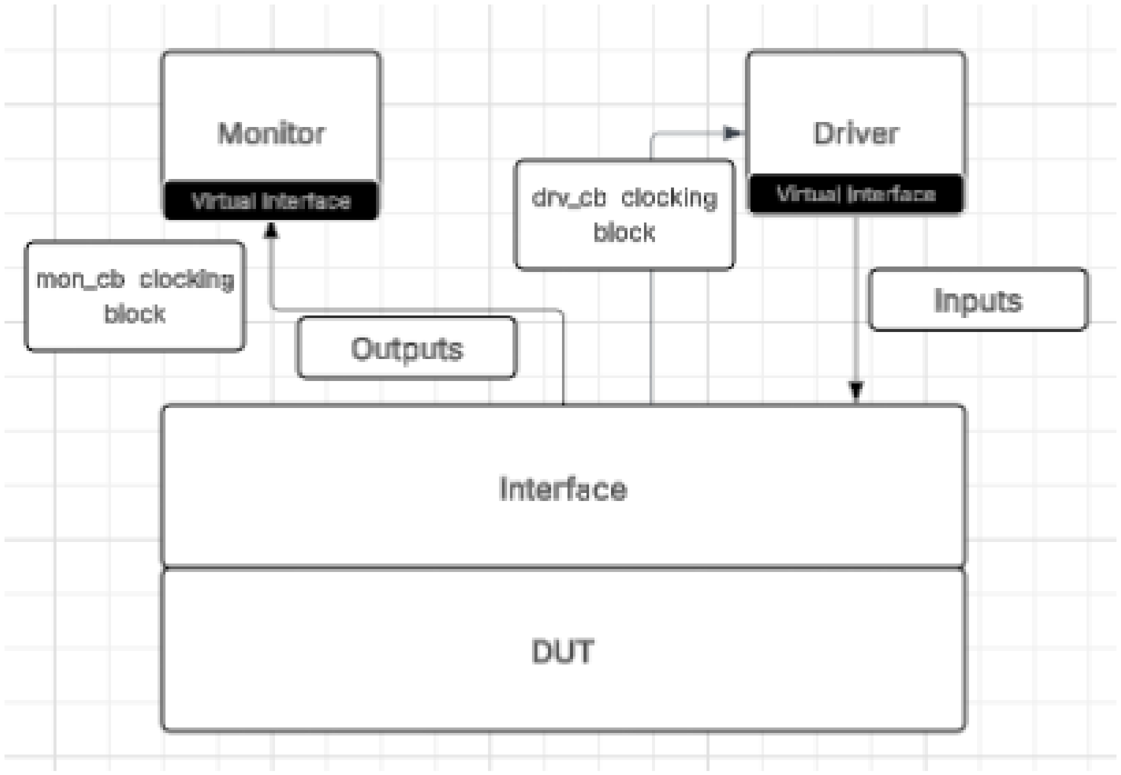
Generator



- Generates transactions containing randomized ALU inputs (OPA, OPB, CMD, CIN, MODE, CMD).

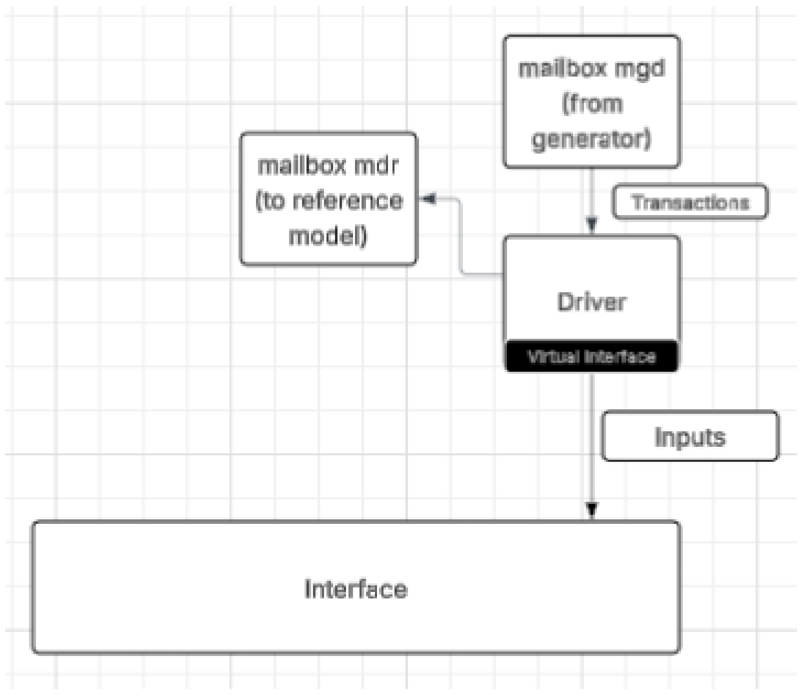
- Controls test variability by applying constraints to randomization (e.g., valid opcodes, corner-case operands).
- Sends transactions to the Driver via a mailbox for execution on the DUT.

Interface



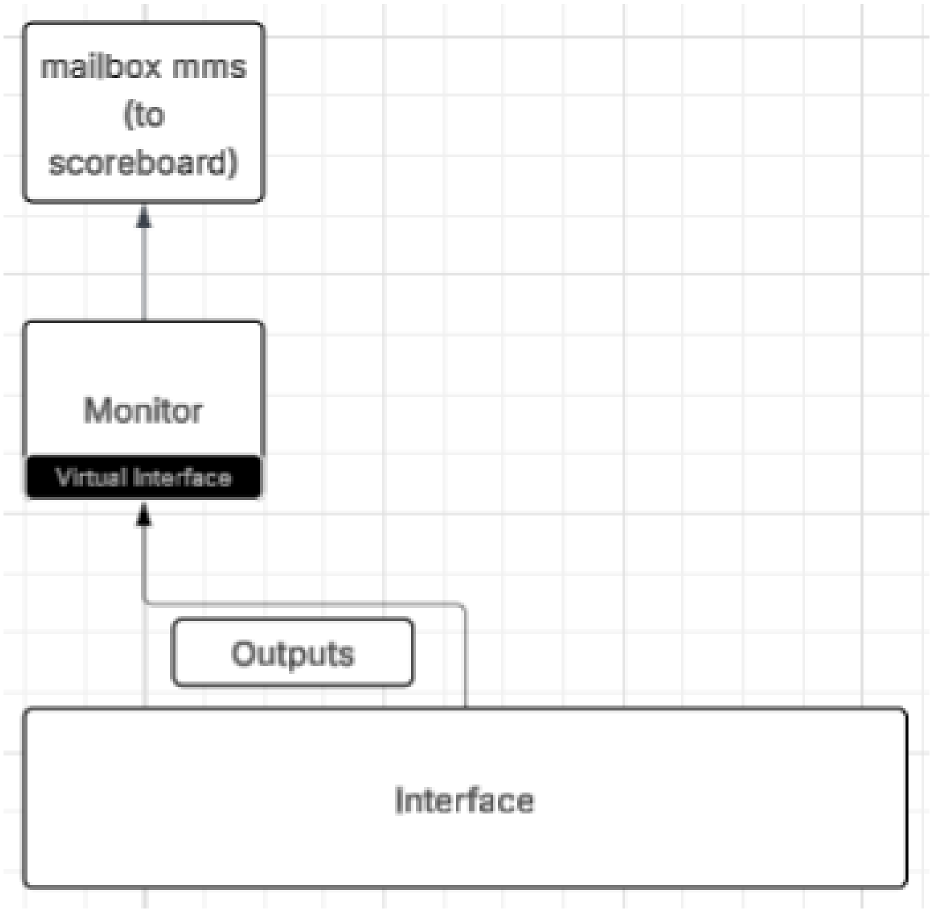
- It is used to bundle all inputs and outputs of DUT, so we don't have to instantiate DUT every time and can directly use an interface.
- It is used to synchronize different components of the testbench to work together via clocking blocks namely drv_cb (for driver), mon_cb(for monitor), ref_cb(for reference model.
- It is used to drive the inputs from the driver to the DUT, and used to access output values from DUT to the monitor.

Driver


mailbox mgd (from generator)
Transactions
mailbox mdr (to reference model)
Driver
Virtual Interface
Inputs
Interface

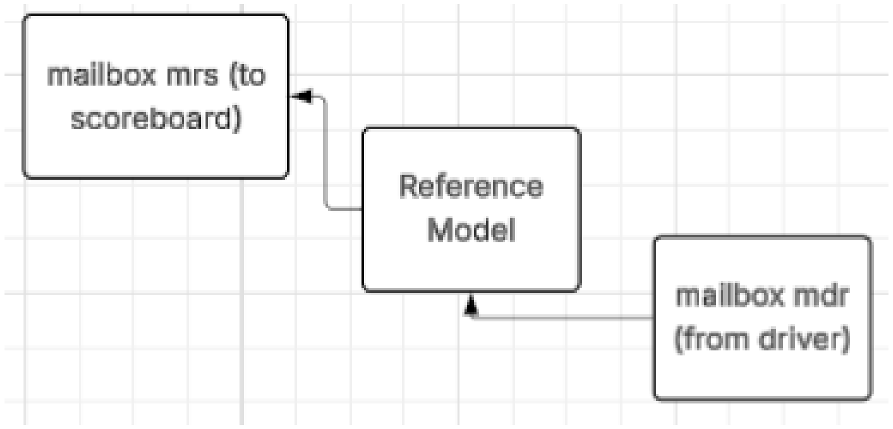
- Driver receives transactions from Generator via mailbox mgd.
- Drives inputs to the DUT through a virtual interface.
- Handles special cases (single operand commands, multi operand commands with wrong inp_valid, multiplication operations).
- Send transactions to the reference model via mailbox mdr.
- Tracks functional coverage of inputs being driven to DUT.

Monitor



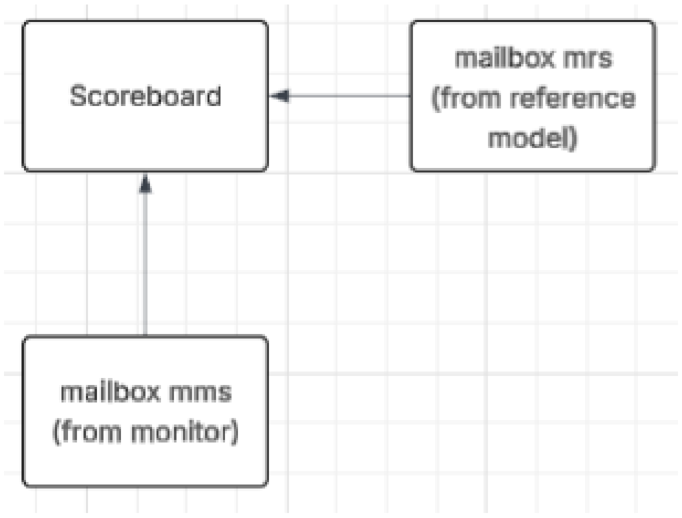
- Monitor as the name states is used to monitor the outputs from the DUT via the interface.
- The Monitor then takes these outputs and puts them in a mailbox which is sent to the scoreboard.

Reference Model



- It allows us to replicate the ALU operations.
- We take inputs from transaction objects received from the driver via a mailbox.
- We take these inputs, feed it to our ALU, perform the operations and generate the outputs.
- These outputs are sent to the scoreboard via a mailbox, where they get compared with the outputs from the DUT.

Scoreboard


Scoreboard
mailbox mrs (from reference model)
mailbox mms (from monitor)

- It compares the DUT outputs with the outputs generated by our reference model.
- It calculates how many test cases have passed and how many have failed.
- It lets us know whether the DUT works as intended.