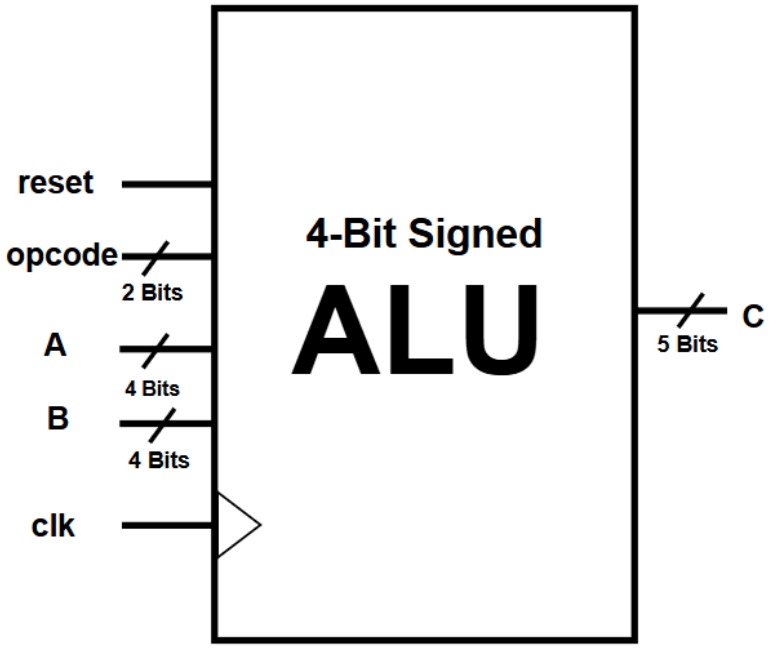
ANALOG DEVICES

Summer 2025’ Digital Communication and Digital Design Internship

ALU Signed 4-Bit

Verification Plan



By: Khaled Ahmed Hamed

Project on Github: [Github Repo]

Contents

Introduction

This document outlines the verification test plan for a 4-Bit Arithmetic Logic Unit (ALU) with asynchronous reset. The goal is to ensure the ALU behaves as expected under all operational conditions, including addition, subtraction, bitwise inverting, bitwise oring and reset functionalities. The verification includes directed test cases to validate functionality, corner cases, and robustness of the design.

IP Design Details

Parameter	Description
clk	Input clock signal
reset	Input reset asynchronous signal
opcode	2-Bit input opcode
A	4-Bit Input data A in 2's complement
B	4-Bit Input data B in 2's complement
C	5-Bit ALU output in 2's complement

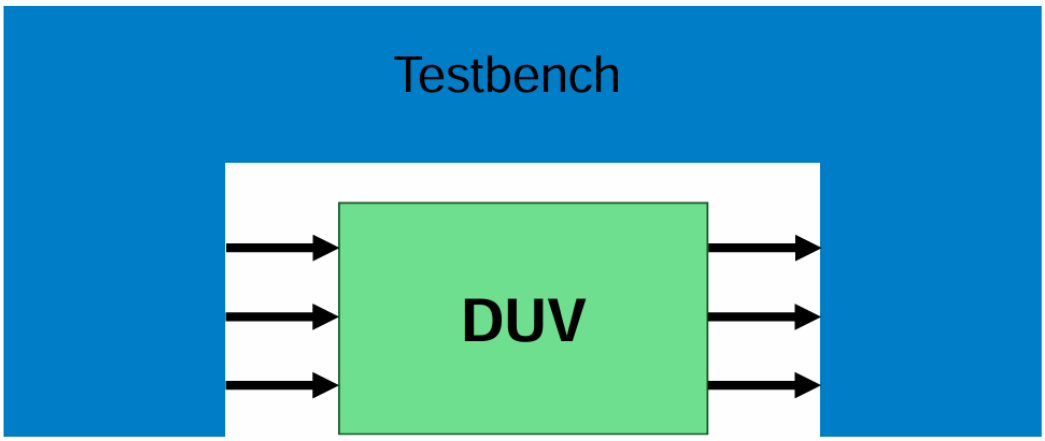
Verification Strategy

Type:

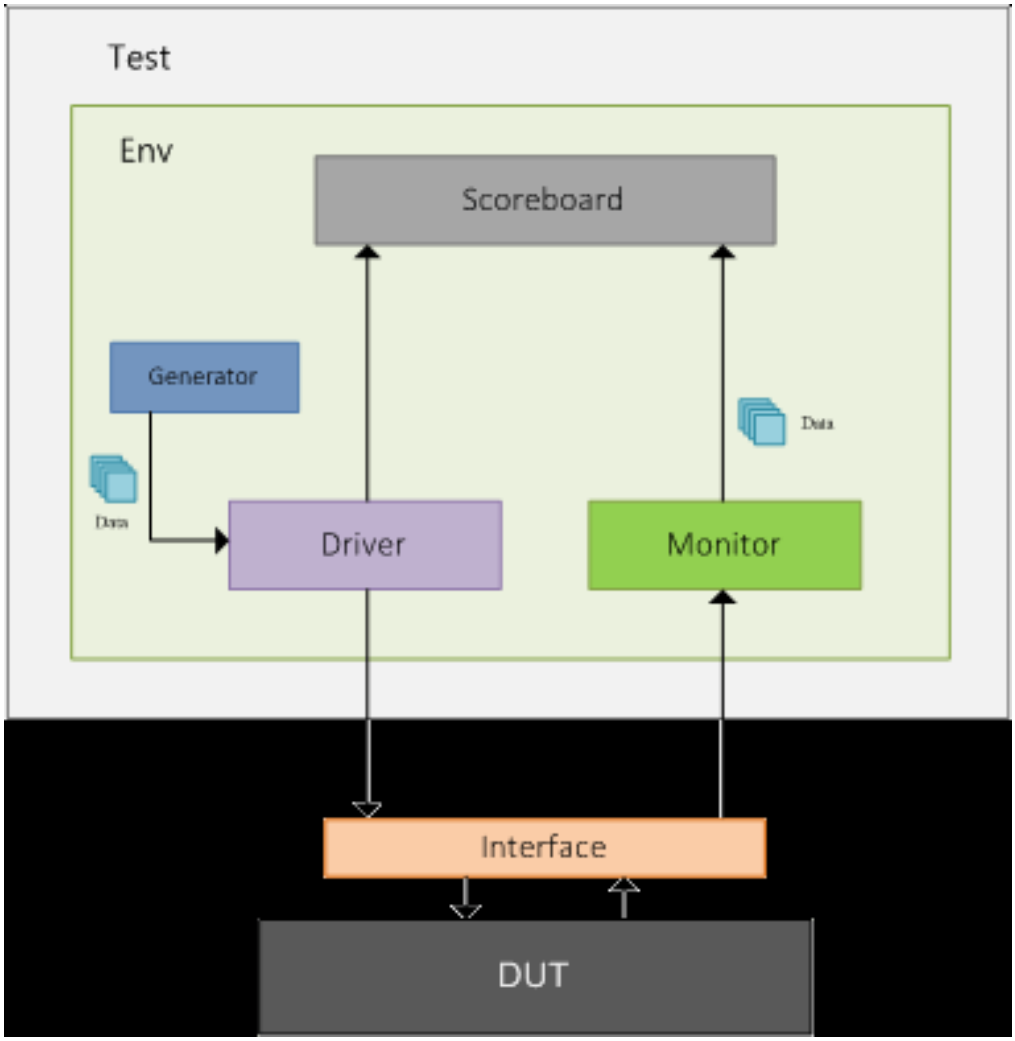
1. Class-Based SV TB
2. Module-Based SV TB using tasks
3. CoCotb

Environment Diagram

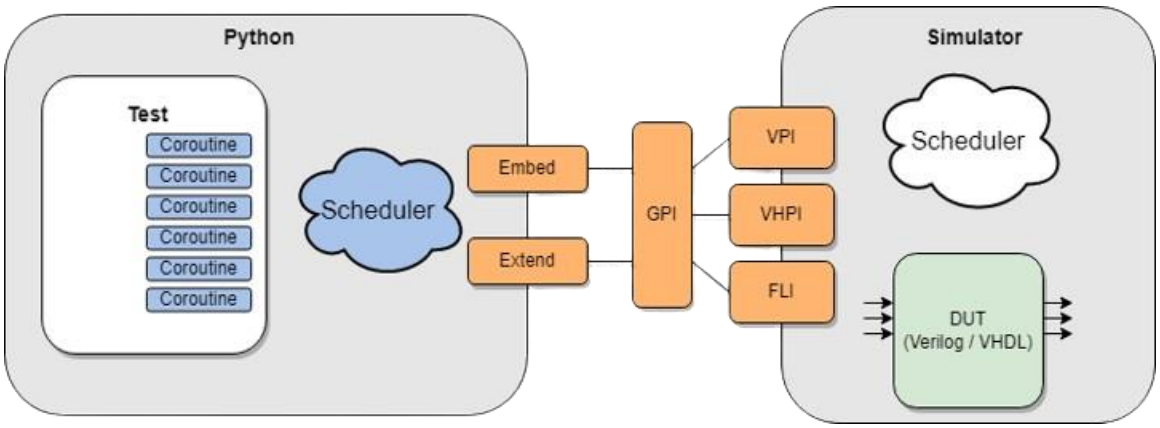
Module-Based:



Class-Based:



CoCotb:

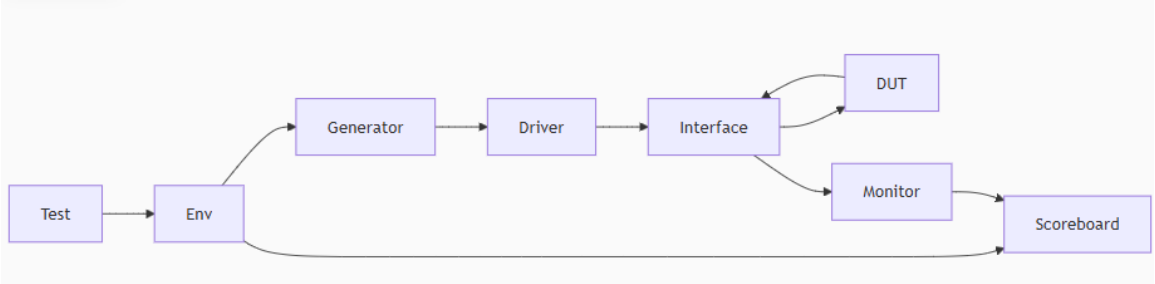


Tools: Questa-Sim, Xcelium, VS Code & EDA Playground.

Testbench Structure:

Class-Based Structure:

Data Flow:



- Test → Env: Configures the environment and starts the test.
- Generator → Driver → Interface → DUT**:** Stimulus path.
- DUT → Interface → Monitor → Scoreboard**:** Response checking path.
- Scoreboard**:** Central point for verification (pass/fail decisions).

Module-Based Structure:

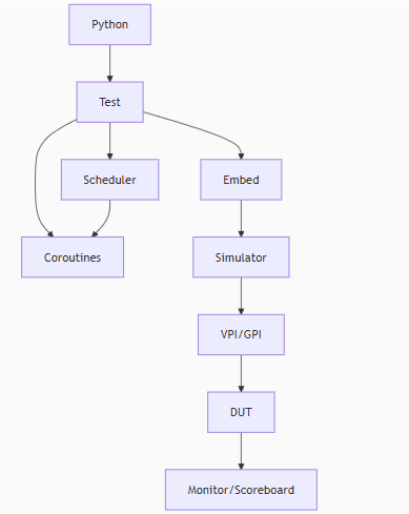
- Signals Declaration
- DUV Instantiation
- Stimulus Generator
- Task to check ALU result
- Task to check reset functionality
- Test Monitor & Results

CoCotb Structure:

Data Flow:

- **Python → Test**: Orchestrates test execution and coroutines.
- **Test → Coroutines**: Concurrent tasks (e.g., stimulus, checks) run under the test.
- **Coroutines → Scheduler**: Handles timing/events (e.g., `await RisingEdge(clk)`).
- **Scheduler ↔ GPI/VPI**: Syncs Python with the simulator's event loop.

- **GPI/VPI → Simulator → DUT**: Drives signals into the DUT (stimulus path).
- **DUT → Simulator → GPI/VPI → Monitor**: Captures DUT outputs (response path).
- **Monitor → Scoreboard**: Validates results (pass/fail decisions).
- **Extend/FU**: Custom utilities (e.g., protocol drivers) plug into the flow.



Test Items (Applied on Class-Based)

Reset Test

1. **Reset Functionality**
   - Assert reset and verify output `C_dut` becomes 0
   - De-assert reset and ensure normal operation resumes

Not_A Operation Tests

2. **Not_A with MAXNEG input**
   - `A = -8 (MAXNEG)`, verify `C_dut = ~A`
3. **Not_A with MAXPOS input**
   - `A = +7 (MAXPOS)`, verify `C_dut = ~A`
4. **Not_A with Zero input**
   - `A = 0`, verify `C_dut = ~A`

Add & Subtract Boundary Tests

5. **MAXNEG + MAXNEG**
   - `A = -8`, `B = -8`, verify `C_dut = A + B` and `A - B`
6. **MAXNEG + 0**

- A = -8, B = 0, verify C_dut = A + B and A - B

7. **MAXNEG + MAXPOS**
   - A = -8, B = +7, verify C_dut = A + B and A - B
8. **0 + MAXNEG**
   - A = 0, B = -8, verify C_dut = A + B and A - B
9. **0 + MAXPOS**
   - A = 0, B = +7, verify C_dut = A + B and A - B
10. **MAXPOS + MAXNEG**
    - A = +7, B = -8, verify C_dut = A + B and A - B
11. **MAXPOS + 0**
    - A = +7, B = 0, verify C_dut = A + B and A - B
12. **0 + 0**
    - A = 0, B = 0, verify C_dut = A + B and A - B
13. **MAXPOS + MAXPOS**
    - A = +7, B = +7, verify C_dut = A + B and A - B

ReductionOR_B Tests

14. **ReductionOR_B with MAXNEG input**
    - B = -8, verify C_dut = |B (should be 1, since any non-zero input gives 1)
15. **ReductionOR_B with MAXPOS input**
    - B = +7, verify C_dut = |B (should be 1)
16. **ReductionOR_B with Zero input**
    - B = 0, verify C_dut = |B (should be 0)

Counter Checks

17. **Error & Correct Counters**

At the end, display:

- Total errors (error_counter) & Total correct results (correct_counter)

Test Case Table

Label	Description	Stimulus Generation	Functionality Check
LABEL1	When the reset is asserted, the output value should be low	Directed at the start of the simulation	A checker in the testbench to make sure the output is correct
LABEL2	Verifing maximum negative value on A with bitwise invert opcode	Directed during the simulation	A checker in the testbench to make sure the output is correct
LABEL3	Verifing maximum positive value on A with bitwise invert opcode	Directed during the simulation	A checker in the testbench to make sure the output is correct
LABEL4	Verifing zero value on A with bitwise invert opcode	Directed during the simulation	A checker in the testbench to make sure the output is correct
LABEL5	Verifing maximum negative value on A and maximum negative value on B	Directed during the simulation	A checker in the testbench to make sure the output is correct
LABEL6	Verifing maximum negative value on A and zero value on B	Directed during the simulation	A checker in the testbench to make sure the output is correct
LABEL7	Verifing maximum negative value on A and maximum positive value on B	Directed during the simulation	A checker in the testbench to make sure the output is correct
LABEL8	Verifing maximum zero value on A and maximum negative value on B	Directed during the simulation	A checker in the testbench to make sure the output is correct
LABEL9	Verifing maximum zero value on A and maximum positive value on B	Directed during the simulation	A checker in the testbench to make sure the output is correct
LABEL10	Verifing maximum positive value on A and maximum negative value on B	Directed during the simulation	A checker in the testbench to make sure the output is correct
LABEL11	Verifing maximum positive value on A and zero value on B	Directed during the simulation	A checker in the testbench to make sure the output is correct
LABEL12	Verifing zero value on A and zero value on B	Directed during the simulation	A checker in the testbench to make sure the output is correct
LABEL13	Verifing maximum positive value on A and maximum positive value on B	Directed during the simulation	A checker in the testbench to make sure the output is correct
LABEL14	Verifing maximum negative value on A with reduction OR opcode	Directed during the simulation	A checker in the testbench to make sure the output is correct
LABEL15	Verifing maximum positive value on A with reduction OR opcode	Directed during the simulation	A checker in the testbench to make sure the output is correct
LABEL16	Verifing zero value on A with reduction OR opcode	Directed during the simulation	A checker in the testbench to make sure the output is correct
LABEL17	Verifing error counter and correct counter	Calculated during and at the end of the simulation	A checker in the testbench to make sure the number of error counts is zero and number of correct counts is correct