ALU Verification Plan

Table of Contents

CHAPTER 1 - PROJECT OVERVIEW AND OBJECTIVES

1.1 PROJECT OVERVIEW

1.2 VERFICATION OBJECTIVES

1.3 DUT INTERFACE

CHAPTER 2 - TESTBENCH ARCHITECTURE AND METHODOLOGY

2.1 TESTBENCH ARCHITECTURE

2.2 COMPONENT DETAILS AND FLOWCHART

CHAPTER 1 - VERIFICATION RESULTS AND ANALYSIS

3.1 ERROR IN DUT

3.2 COVERAGE REPORT

CHAPTER 1

PROJECT OVERVIEW

The goal of this project is to verify ALU that can perform various arithmetic and logic functions. It includes custom commands, shift/rotate functionality, and intelligent error detection.

We started by understanding the functional requirements such as the need for ADD, SUB, CMP, INC, DEC, AND, OR, XOR, SHL, SHR, ROL, and ROR. These were mapped to the CMD signal with proper encoding for both arithmetic and logical modes.

Inputs like operand A and B, clock, reset, input validity, and carry-in are driven into the ALU. Based on the command selected and the operation mode, the ALU generates results along with flags like overflow, equality, carry, and error.

If the operand A is valid and if we get operand B with in or at 16th clock cycle then the particular operation is valid otherwise the error flag will set high.

In this project, our main focus is on verifying it using a System Verilog based testbench. This ensures our ALU behaves correctly for all supported operations and under all possible input conditions.

VERIFICATION OBJECTIVES

The primary objective of the verification process is to confirm the correctness, stability, and completeness of the ALU under a wide range of scenarios. This includes validating the output for all defined operations, ensuring correct propagation of flags such as carry-out, overflow, and comparison results, and verifying robustness against invalid inputs and timing violations. 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.

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:

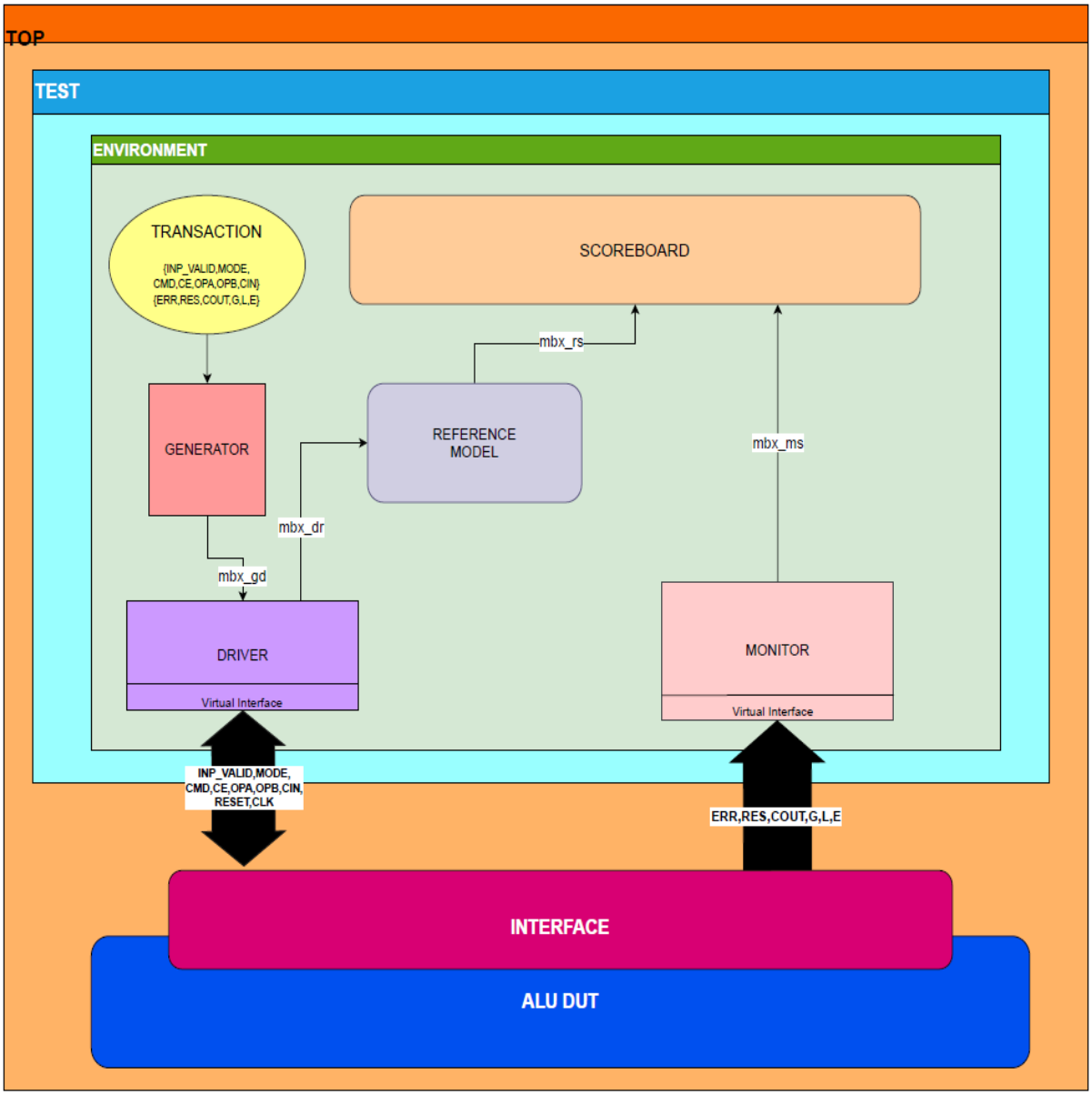
The modport groups and specifies the direction to the signals specified within the interface.

CLOCKING BLOCK:

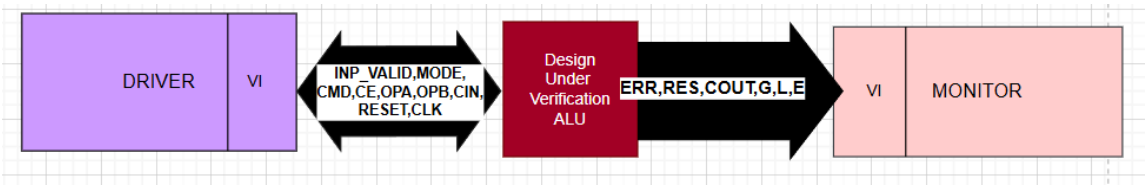
A clocking block in SystemVerilog defines the timing and synchronization for signal interaction between the testbench and DUT. It specifies a clock event as a reference, the signals to be sampled or driven, and the timing of these actions relative to the clock. This helps ensure stable and predictable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

CHAPTER 2

TESTBENCH ARCHITECTURE



1.INTERFACE



SystemVerilog interface with clocking blocks and assertions.

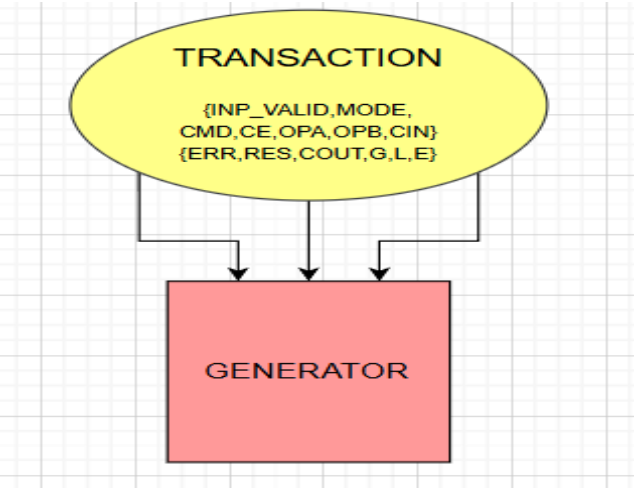
Key Features:

- **Clocking Blocks:** drv_cb, mon_cb, ref_cb for synchronization
- **Modports:** DRV, MON, REF_SB for component-specific access
- **Built-in Assertions:** Reset, timeout, error condition checking
- **Signal Definitions:** All ALU input/output signals with proper widths

2.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.

3.TRANSACTION



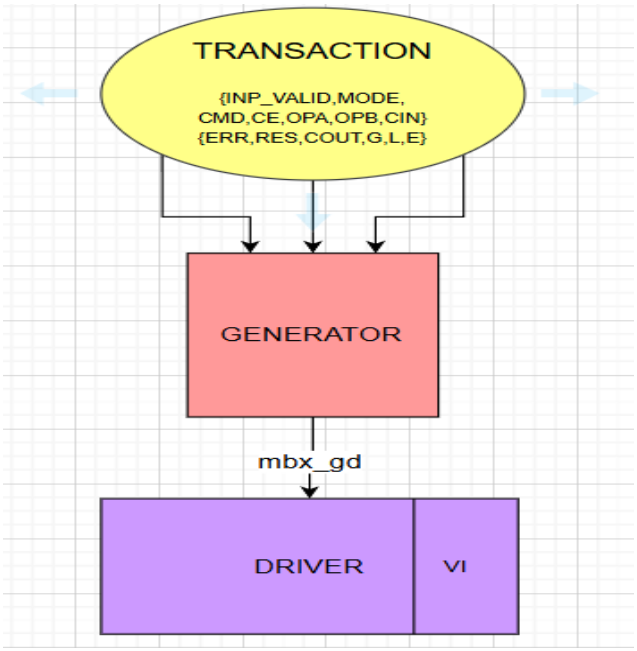
The alu_transaction class encapsulates all ALU input/output data for verification.

Components:

- **Randomized Input Stimuli:** rand keyword signals for randomization
    - INP_VALID, MODE, CMD, CIN, OPA, OPB

- **Output Signals:** Non-randomized response signals
    - ERR, RES, OFLOW, COUT, G, L, E
- **Constraints:** signals constraint forces CE == 1
- **Deep Copy Method:** copy() function creates independent transaction copies
- **Extended Classes:** Specialized constraint classes for directed testing

4.GENERATOR

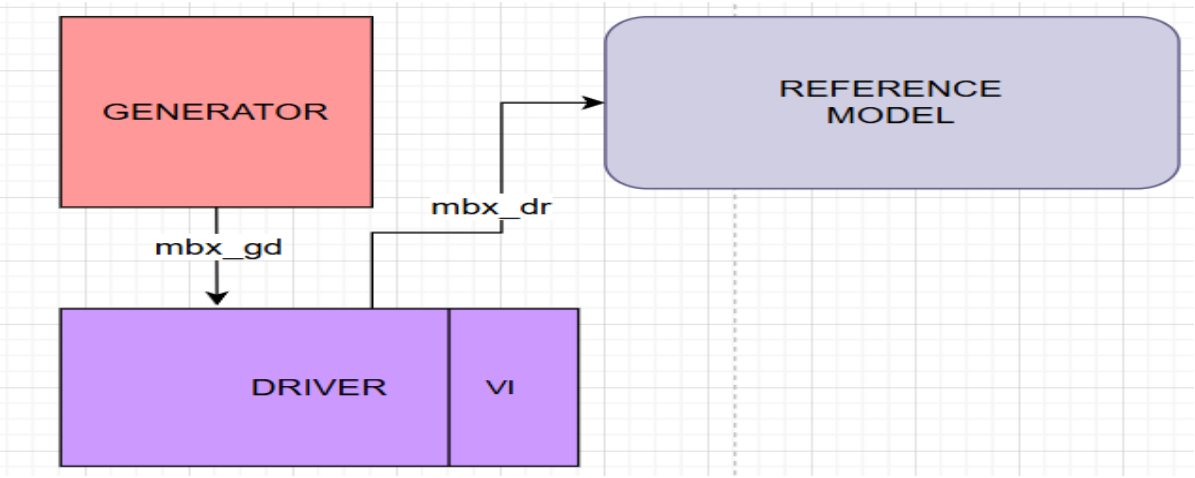


Generates randomized transactions and sends them to driver.

Key Features:

- blueprint transaction handle for ALU operations
- mbx_gd mailbox connecting to driver
- start() task randomizes blueprint and sends copies through mailbox
- Displays transaction details for debug

5.DRIVER

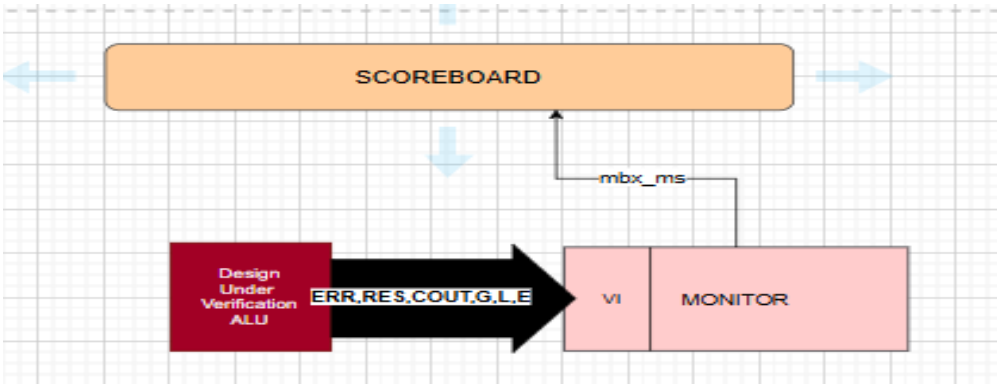


Drives transactions from generator to DUV and reference model.

Key Features:

1. **Two Mailboxes:**
    - mbx_gd: Receives transactions from generator
    - mbx_dr: Sends transactions to reference model
2. **Virtual Interface:**
    - vif: Virtual interface with clocking blocks for DUV communication
    - drive_VI(): Applies stimuli to DUV through interface
3. **Functional Coverage Group (DRV_cg):**
    - Covers INPUT_VALID, COMMAND, MODE, CLOCK_ENABLE combinations
    - Cross coverage for MODE_X_CMD, OPA_X_OPB interactions
4. **Special Handling:**
    - 16-cycle timeout logic for partial operand scenarios
    - Multi-cycle timing for multiplication operations (CMD 9,10)

6.MONITOR

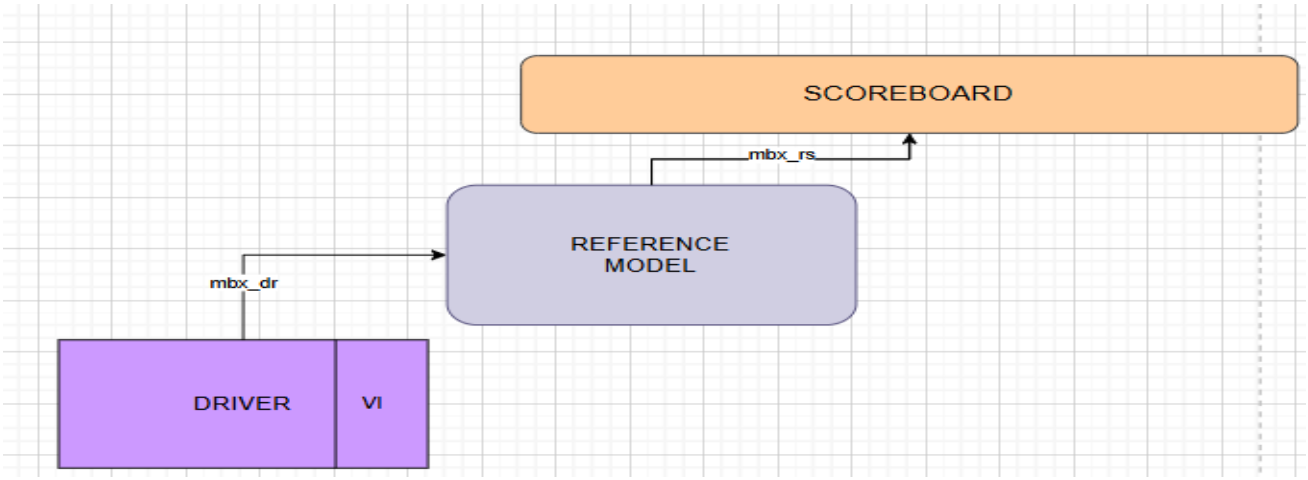


Captures DUV output transactions and forwards to scoreboard.

Key Features:

- **Mailbox Communication:** mbx_ms sends captured transactions
- **Output Coverage Group (MON_output_cg):** Covers RES, COUT, OFLOW, ERR, G, L, E
- **Functionality:**
    - start(): Continuously monitors DUV interface
    - Multi-cycle sampling for multiplication operations
    - Samples outputs and forwards to scoreboard

7.REFERENCE MODEL

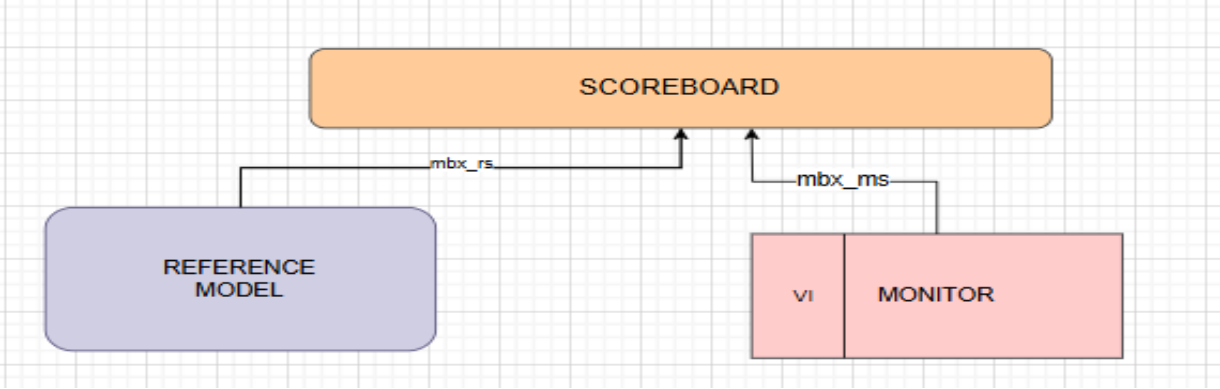


Golden model producing expected outputs for comparison.

Key Features:

1. **Two Mailboxes:**
    - mbx_dr: Receives input stimuli from driver
    - mbx_rs: Sends expected results to scoreboard
2. **Functionality:**
    - start(): Continuously processes driver transactions
    - Implements golden reference for arithmetic operations (ADD, SUB, MUL, etc.)
    - Implements logical operations (AND, OR, XOR, shifts, rotates)
    - Handles timeout conditions and error detection

8.SCOREBOARD



Compares actual DUV results against expected results.

Key Features:

- **Mailbox Communication:**
    - mbx_rs: Receives expected results from reference model

- mbx_ms: Receives actual results from monitor

- **Field-wise Comparison:** Individual counters for each output field
- **Functionality:**
  - compare_report(): Compares expected vs actual for all fields
  - print_summary(): Provides detailed statistics and success rates

9.MAILBOXES

Mailboxes are used for communication between testbench components. Each component pair (e.g., generator to driver, monitor to scoreboard) communicates through its dedicated mailbox. This ensures safe, synchronized, and transaction-level data transfer without race conditions.

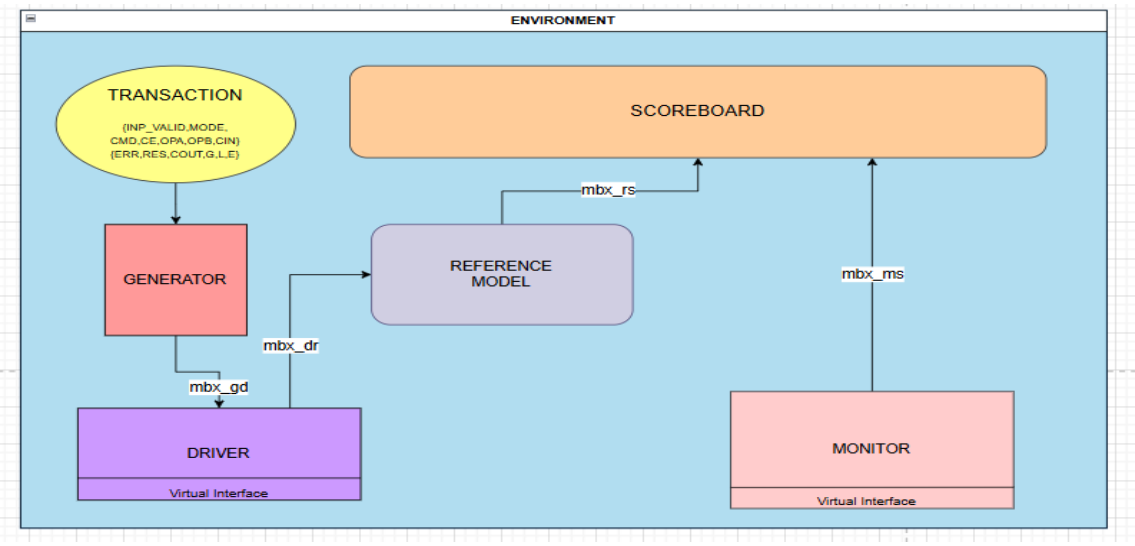
• mbx_gd: Generator to Driver

• mbx_dr: Driver to Reference Model

• mbx_rs: Reference Model to Scoreboard

• mbx_ms: Monitor to Scoreboard

10.ENVIRONMENT



Encapsulates all testbench components and coordinates verification flow.

Key Features:

- Instantiates generator, driver, monitor, reference model, scoreboard
- Creates four mailbox connections (mbx_gd, mbx_dr, mbx_rs, mbx_ms)
- build() task constructs all components
- start() task launches all processes using fork-join
- Manages overall verification coordination

11.TEST

Top-level tests configuring and running verification environment.

Key Features:

- **Base Test:** alu_test with basic run functionality
- **Specialized Tests:**
    - test1/test2: MODE_CHECK1/0 for arithmetic/logical modes
    - test3: CMD_MUL_ONLY_CHECK for multiplication testing
    - test4: CE_LOW for clock enable testing
    - test5/6/7: INP_VALID_0/1/2 for input validation testing
- **Regression Test:** test_regression runs all arithmetic/logical command combinations

12.TOP

This is the top most module of the SV testbench architecture where control signals like clock and reset are generated. Its code is written inside a module and the interface, the DUV and the test are instantiated in it

CHAPTER 3

ERROR IN DUT

S.No	Bug Category	Issue Description
1	Timeout Logic	ERR flag not set after 16 clock cycles timeout condition
2	Single-Operand Operations	INC_A/DEC_A/INC_B/DEC_B only work with INP_VALID=11, not 01/10
3	Carry-Out Logic	COUT incorrect/missing in increment operations
4	Increment A Operation	INC_A holds same value as OPA instead of incrementing
5	Increment B Operation	INC_B decrements value instead of incrementing
6	Shift Operations	Right shift OPA/OPB fail, Left shift OPB fails
7	Decrement B Operation	DEC_B increments value instead of decrementing
8	Overflow Logic	OFLOW missing for decrement operations (0→-1)
9	Carry-In Timing	CIN appears one clock cycle late in ADD_CIN/SUB_CIN
10	Rotate Error Detection	ROR_A_B error condition always remains zero
11	Multiplication Operation	Issues in MULT_INC and MULT_SHIFT functionality
12	Logical OR Operation	OR operation fails with valid inputs and timing

COVERAGE REPORT

Overall Coverage:

top
alu_pkg

Number of tests run:	1
Passed:	0
Warning:	0
Error:	1
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 %
top	86.93%	66.14%
intrf	74.77%	60.00%
DUV	89.13%	82.96%
alu_pkg	23.72%	94.69%
alu_transaction/copy	100.00%	100.00%
MODE_CHECK1/copy	100.00%	100.00%
MODE_CHECK0/copy	100.00%	100.00%
CMD_MUL_ONLY_CHECK/copy	100.00%	100.00%
CE_LOW/copy	100.00%	100.00%
INP_VALID_0/copy	100.00%	100.00%
INP_VALID_1/copy	100.00%	100.00%

Coverage Summary by Type:

Total Coverage:					24.24%	74.39%
Coverage Type	**Bins**	**Hits**	**Misses**	**Weight**	**% Hit**	**Coverage**
Covergroups	66558	15184	51374	1	22.81%	83.45%
Statements	864	837	27	1	96.87%	96.87%
Branches	146	138	8	1	94.52%	94.52%
FEC Conditions	49	41	8	1	83.67%	83.67%
Toggles	304	267	37	1	87.82%	87.82%
Assertions	8	0	8	1	0.00%	0.00%

Scope: /alu_pkg/alu_monitor

Covergroup type:

MON_output_cg

Summary	Total Bins	Hits	Hit %
Coverpoints	268	265	98.88%
Crosses	0	0	0.00%

Search:

CoverPoints	Total Bins	Hits	Misses	Hit %	Goal %	Coverage %
#coverpoint_0#	256	256	0	100.00%	100.00%	100.00%
#coverpoint_1#	2	2	0	100.00%	100.00%	100.00%
#coverpoint_2#	2	2	0	100.00%	100.00%	100.00%
#coverpoint_3#	2	2	0	100.00%	100.00%	100.00%
#coverpoint_4#	2	1	1	50.00%	50.00%	50.00%
#coverpoint_5#	2	1	1	50.00%	50.00%	50.00%
#coverpoint_6#	2	1	1	50.00%	50.00%	50.00%

Scope: /alu_pkg/alu_driver

Covergroup type:

DRV_cg

Summary	Total Bins	Hits	Hit %
Coverpoints	554	553	99.81%
Crosses	65736	14366	21.85%

Search:

CoverPoints	Total Bins	Hits	Misses	Hit %	Goal %	Coverage %
CARRY_IN	2	2	0	100.00%	100.00%	100.00%
CLOCK_ENABLE	2	1	1	50.00%	50.00%	50.00%
COMMAND	32	32	0	100.00%	100.00%	100.00%
INPUT_VALID	4	4	0	100.00%	100.00%	100.00%
MODE	2	2	0	100.00%	100.00%	100.00%
OPERAND_A	256	256	0	100.00%	100.00%	100.00%
OPERAND_B	256	256	0	100.00%	100.00%	100.00%

Search:

Crosses	Total Bins	Hits	Misses	Hit %	Goal %	Coverage %
INP_VALID_X_CMD	128	128	0	100.00%	100.00%	100.00%
MODE_X_CMD	64	64	0	100.00%	100.00%	100.00%
MODE_X_INP_VALID	8	8	0	100.00%	100.00%	100.00%
OPA_X_OPB	65536	14166	51370	21.61%	21.61%	21.61%

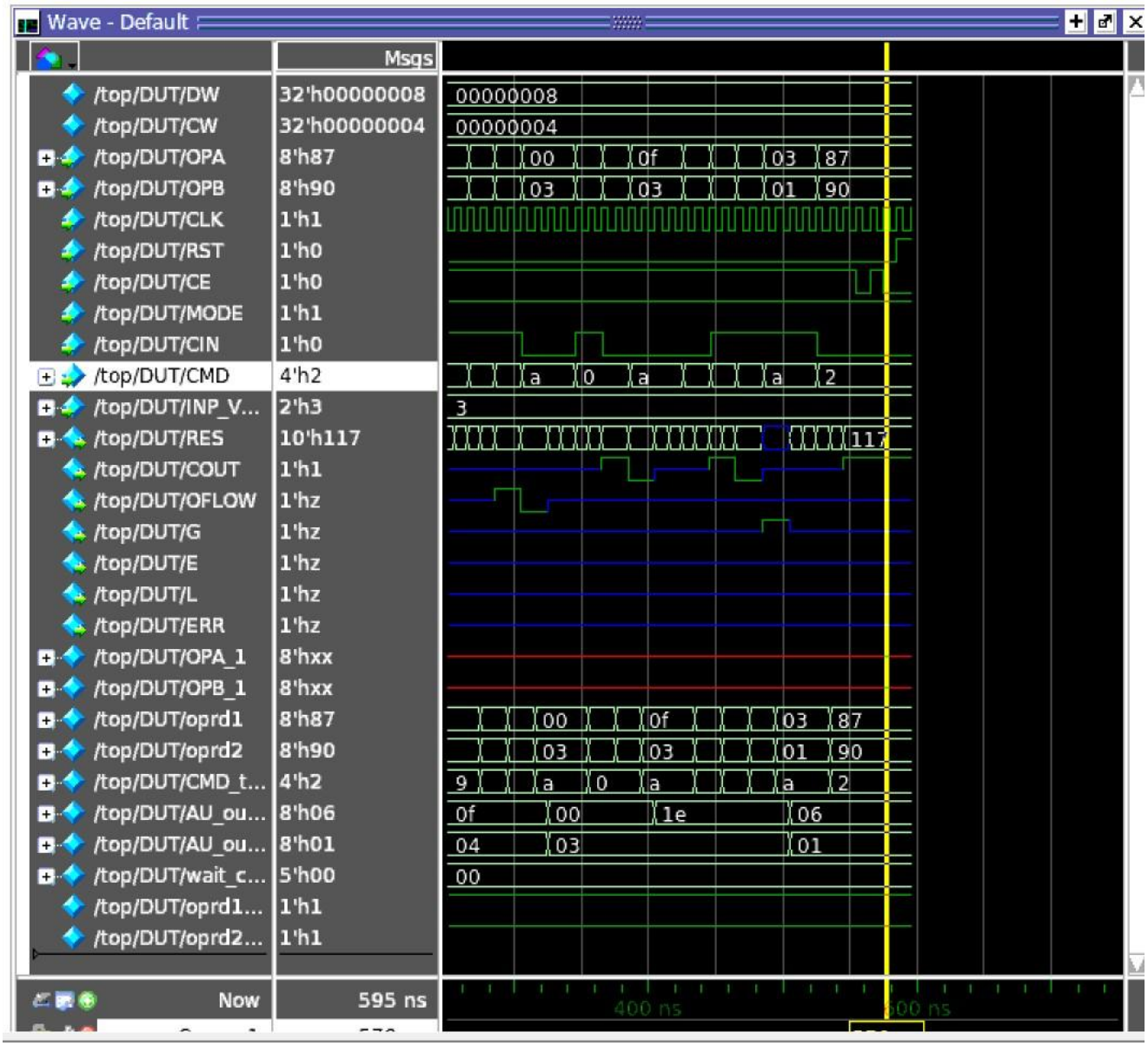
Assertions Coverage:

Assertions Coverage Summary:

Search:

Assertions	Failure Count	Pass Count	Attempt Count	Vacuous Count	Disable Count	Active Count	Peak Active Count	Status
/top/intrf/assert__0	238	30	30759	30490	1	0	5	Failed
/top/intrf/assert__1	1627	0	30759	29132	0	0	2	Failed
/top/intrf/assert__2	853	4	30759	29902	0	0	2	Failed
/top/intrf/assert__3	3655	81	30759	27023	0	0	2	Failed
/top/intrf/assert__ppt_clock_enable	0	0	30759	30758	1	0	0	ZERO
/top/intrf/assert__ppt_reset	2	0	30759	30757	0	0	2	Failed
/top/intrf/assert__ppt_timeout_arithmetic	3027	83	30759	27648	1	0	17	Failed
/top/intrf/assert__ppt_timeout_logical	5702	199	30759	24857	1	0	17	Failed
/work.alu_if/assert__0	238	30	30759	30490	1	0	5	Failed
/work.alu_if/assert__1	1627	0	30759	29132	0	0	2	Failed
/work.alu_if/assert__2	853	4	30759	29902	0	0	2	Failed
/work.alu_if/assert__3	3655	81	30759	27023	0	0	2	Failed
/work.alu_if/assert__ppt_clock_enable	0	0	30759	30758	1	0	0	ZERO
/work.alu_if/assert__ppt_reset	2	0	30759	30757	0	0	2	Failed
/work.alu_if/assert__ppt_timeout_arithmetic	3027	83	30759	27648	1	0	17	Failed
/work.alu_if/assert__ppt_timeout_logical	5702	199	30759	24857	1	0	17	Failed

Waveform:


Wave - Default
Msgs
/top/DUT/DW 32'h00000008
/top/DUT/CW 32'h00000004
/top/DUT/OPA 8'h87
/top/DUT/OPB 8'h90
/top/DUT/CLK 1'h1
/top/DUT/RST 1'h0
/top/DUT/CE 1'h0
/top/DUT/MODE 1'h1
/top/DUT/CIN 1'h0
/top/DUT/CMD 4'h2
/top/DUT/INP_V... 2'h3
/top/DUT/RES 10'h117
/top/DUT/COUT 1'h1
/top/DUT/OFLOW 1'hz
/top/DUT/G 1'hz
/top/DUT/E 1'hz
/top/DUT/L 1'hz
/top/DUT/ERR 1'hz
/top/DUT/OPA_1 8'hxx
/top/DUT/OPB_1 8'hxx
/top/DUT/oprd1 8'h87
/top/DUT/oprd2 8'h90
/top/DUT/CMD_t... 4'h2
/top/DUT/AU_ou... 8'h06
/top/DUT/AU_ou... 8'h01
/top/DUT/wait_c... 5'h00
/top/DUT/oprd1... 1'h1
/top/DUT/oprd2... 1'h1
Now
595 ns
400 ns