AXI4-Lite Presilicon Verification Report

ECE 593 Verification Team

May 23, 2025

Objective

This report summarizes the verification coverage analysis of an AXI4-Lite slave design. We assess both code coverage (line, branch, condition) and functional coverage via SystemVerilog covergroups. The goal was to maximize verification completeness through a combination of constrained-random testing and directed stimulus.

1 Code Coverage Analysis

Limitations in Code Coverage Achievement

Despite a comprehensive verification strategy, certain conditions prevented 100% code coverage. The major limitations are as follows:

- **Unverifiable Slave-Driven Signals:** The signal `BVALID` is driven by the slave DUT, and hence cannot be controlled directly from the testbench. This results in uncovered conditions like:

  ```
  if (BVALID && BREADY)
  ```

  where `BVALID` being '0' could not be triggered externally, leading to incomplete branch and condition coverage.

- **Complex Compound Conditions:** For nested conditions like:

  ```
  if (AWVALID && AWREADY && WVALID && WREADY)
  ```

  none of the 4 inputs were observed to be '0' during simulation, likely due to synchronous assertion or simultaneous driving. These conditions remained 0% covered as all false-paths weren't exercised.

- **Design Restrictions:** The RTL might be structurally written to always assert valid signals together (e.g., `AWVALID` with `WVALID`), thereby limiting branching flexibility.

Code Coverage – Write Response

Coverage Details

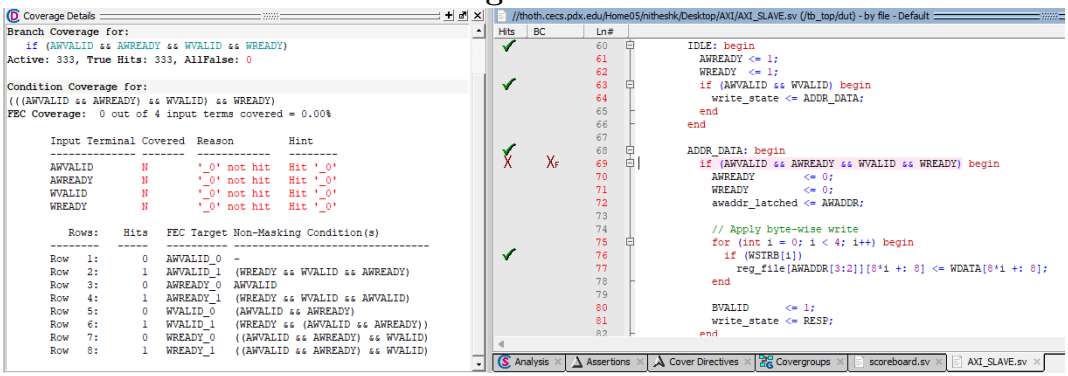
```
By file
File: AXI_SLAVE.sv
Line: 86
Branch Coverage for:
    if (BVALID && BREADY)
Active: 530, True Hits: 332, AllFalse: 198

Condition Coverage for:
(BVALID && BREADY)
FEC Coverage:  1 out of 2 input terms covered = 50.00%

        Input Terminal Covered  Reason         Hint
        -------------- -------  ------------   --------
        BVALID         N        '_0' not hit   Hit '_0'
        BREADY         Y

           Rows:    Hits   FEC Target Non-Masking Condition(s)
        --------    -----   ---------- -------------------------
        Row   1:        0   BVALID_0   -
        Row   2:        1   BVALID_1   BREADY
        Row   3:        1   BREADY_0   BVALID
        Row   4:        1   BREADY_1   BVALID
```

Code Coverage – All False Path



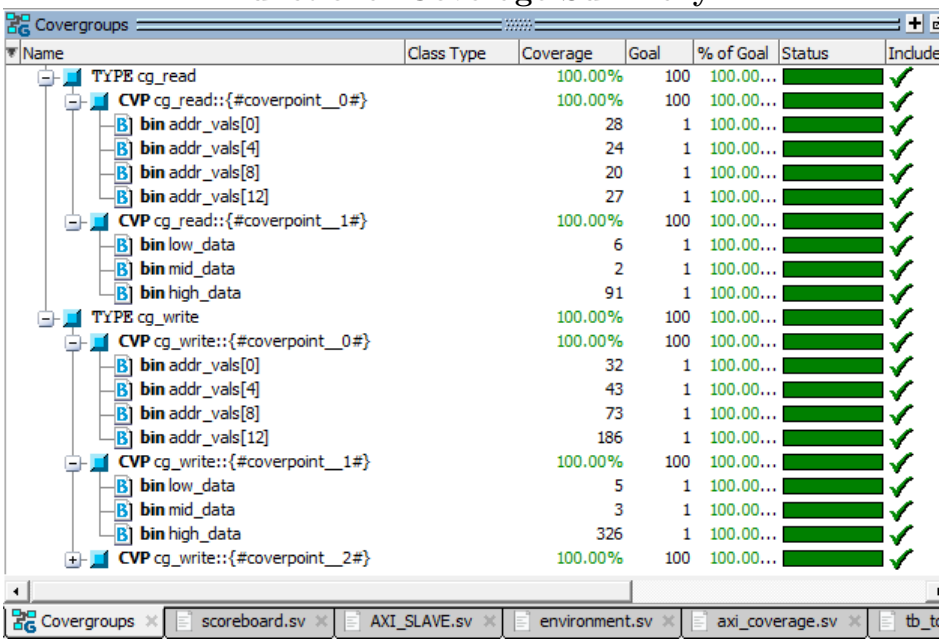
2 Functional Coverage Analysis

We utilized covergroups in the monitor to track functional scenarios such as:

- Read/Write address values (`0, 4, 8, 12`)
- Data range bins: low, mid, high
- Write strobes: 1-bit, 2-bit, full strobe

The functional coverage achieved 100% for all declared bins, confirming that the stimulus generator hit all valid scenarios for data and address combinations.

Functional Coverage Summary



3 Testing Strategies

3.1 Constrained Random Testing

Random stimulus with constraints allowed broad coverage of input space and automatic generation of address, data, and strobe combinations. This technique is excellent for:

- Hitting a wide range of legal values
- Generating different permutations of transactions
- Observing corner-case behavior

Limitation: It cannot guarantee coverage of specific branch conditions (e.g., forcing one signal low in a multi-condition branch), especially for output or feedback-controlled signals.

3.2 Directed Testing

Directed stimulus was essential to:

- Explicitly drive certain signal patterns (e.g., `AWVALID=0`, `WVALID=1`)
- Cover boundary values and unreachable states missed by randomization
- Force protocol sequences to complete, especially with master-slave handshakes

Benefit: Directed tests can close coverage holes left by constrained random tests.

4 Conclusion

Even with complete functional coverage and extensive stimulus, 100% code coverage was not attainable due to:

- Slave-controlled outputs that can't be forced in TB
- Synchronous conditions tightly tied in design logic
- Absence of inverse activation paths for some condition branches

Combining directed testing with constrained-random generation is essential for maximizing coverage in AXI protocol verification.