Class Based TB Implementation Report

Overview

A class-based verification environment was implemented for the MIPS Lite CPU to enable scalable, reusable, and coverage-driven verification. This environment replaces basic directed testing with a modular architecture that separates stimulus generation, monitoring, checking, and coverage.

Verification Architecture

The testbench consists of the following components:

Generator

- Produces instruction transactions.
- Predicts expected ALU outputs based on instruction decoding.
- Sends transactions to driver and scoreboard via mailboxes.

Driver

- Provides timing synchronization.
- Acts as execution pacing element rather than direct signal driver.
- Helps emulate realistic CPU execution flow.

Input Monitor

- Observes instruction fetch interface.
- Displays PC progression and instruction execution.

Output Monitor

- Captures DUT outputs:
  - ALU result
  - Instruction execution state
- Sends observed data to scoreboard.

Scoreboard

- Compares expected ALU results with observed results.
- Maintains pass/fail counters.
- Generates final verification summary.

Coverage Collector

- Functional coverage implemented using SystemVerilog covergroups.
- Tracks opcode usage, register utilization, and ALU result distributions.

Verification Achievements

- Successfully executed directed instruction sequences.
- Functional coverage collected for major datapath elements.
- Modular verification environment established for future expansion.

Observed Issue (Scoreboard Failures)

The scoreboard currently reports mismatches due to:

- Pipeline latency misalignment between expected and observed outputs.
- Generator assumes zero-latency execution.
- Output monitor captures results after pipeline delay.

This causes all transactions to appear as failures even though waveform inspection confirms correct functionality.

Planned Fix

Future improvements include:

- Pipeline stage latency modeling in scoreboard.
- Transaction tagging with cycle alignment.
- Enhanced synchronization between generator predictions and DUT outputs.

Conclusion

The class-based verification environment is functionally operational, provides meaningful coverage data, and establishes a scalable verification infrastructure. Remaining issues are limited to scoreboard synchronization rather than functional CPU errors.