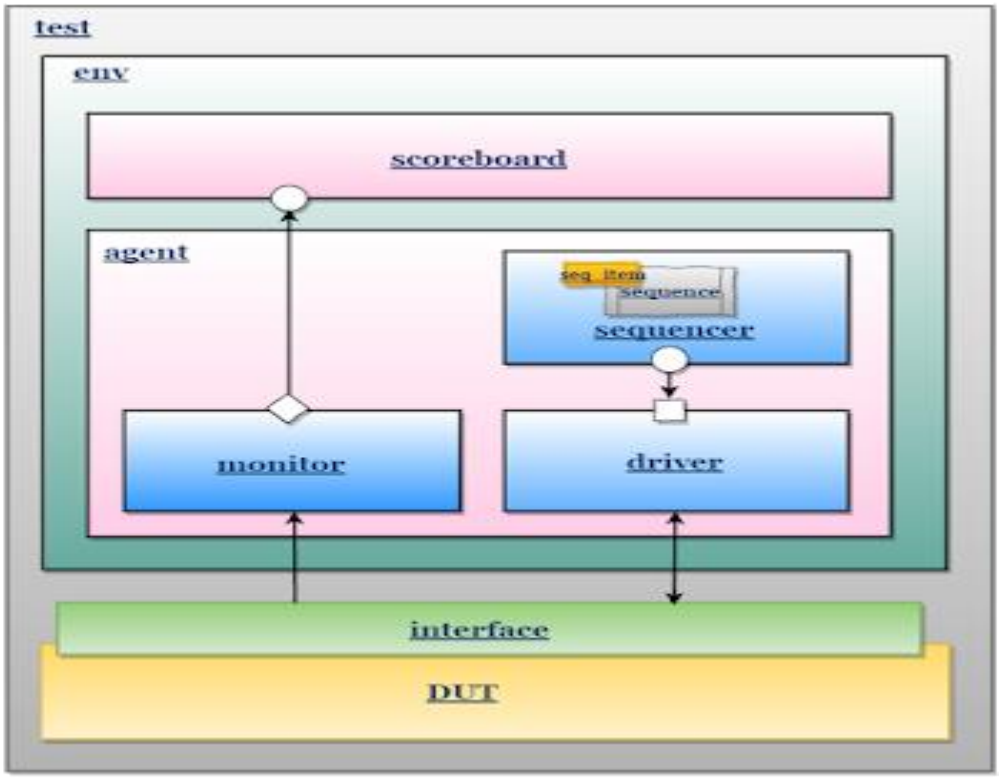
Class – Based TestBench Implementation



Testbench Structure

The testbench follows a hierarchical structure where:

- The Testbench module serves as the top module.
- It creates an instance of the Test class.
- The Test class instantiates the Environment class.

The Environment class is responsible for creating essential test components, including:

1. Scoreboard
2. Monitor
3. Driver
4. Generator
5. Coverage

1. Scoreboard

- Ensures the correctness of the DUT by comparing expected and actual outputs.
- Receives DUT output data from the Monitor for validation.

- Checks if the DUT behaves as per the test expectations and scenarios.
- Generates pass/fail results based on comparisons.
- Tracks the progress and outcomes of test cases.
- Highlights discrepancies between expected and observed DUT behavior for debugging.

2. Monitor

- Observes and captures input and output signals of the DUT during simulation.
- Logs data transactions for debugging and further analysis.
- Tracks specific DUT interfaces and signals of interest.
- Acts as a data source for the scoreboard to verify DUT behavior.
- Provides real-time logging and monitoring for deeper visibility into DUT responses.

3. Driver

- Converts high-level test scenarios into low-level signal transactions.
- Interfaces between the test environment and the DUT.
- Translates test inputs into appropriate signal-level interactions.
- Sends stimulus to the DUT while ensuring compliance with protocol and timing requirements.
- Facilitates seamless communication between the testbench and DUT.
- Helps execute test cases efficiently by ensuring proper signal generation.

4. Generator

- Generates test scenarios and sequences based on specifications.
- Produces stimuli to evaluate various DUT functionalities.
- Covers different operational modes and edge cases.
- Enables customization of test sequences to stress-test specific features.
- Provides randomized or directed input variations for enhanced test coverage.
- Automates scenario creation, making testing more efficient and comprehensive.

5. Coverage

- Tracks and assesses signal transitions and state changes during simulation.
- Measures how effectively the test suite verifies the DUT.
- Observes and logs signal activity across DUT interfaces.
- Records coverage metrics to determine tested vs. untested functionalities.

- Generates coverage reports showing the percentage of exercised features.
- Identifies gaps in testing and provides insights for test enhancement.
- Guides refinements in test scenarios to improve overall coverage.