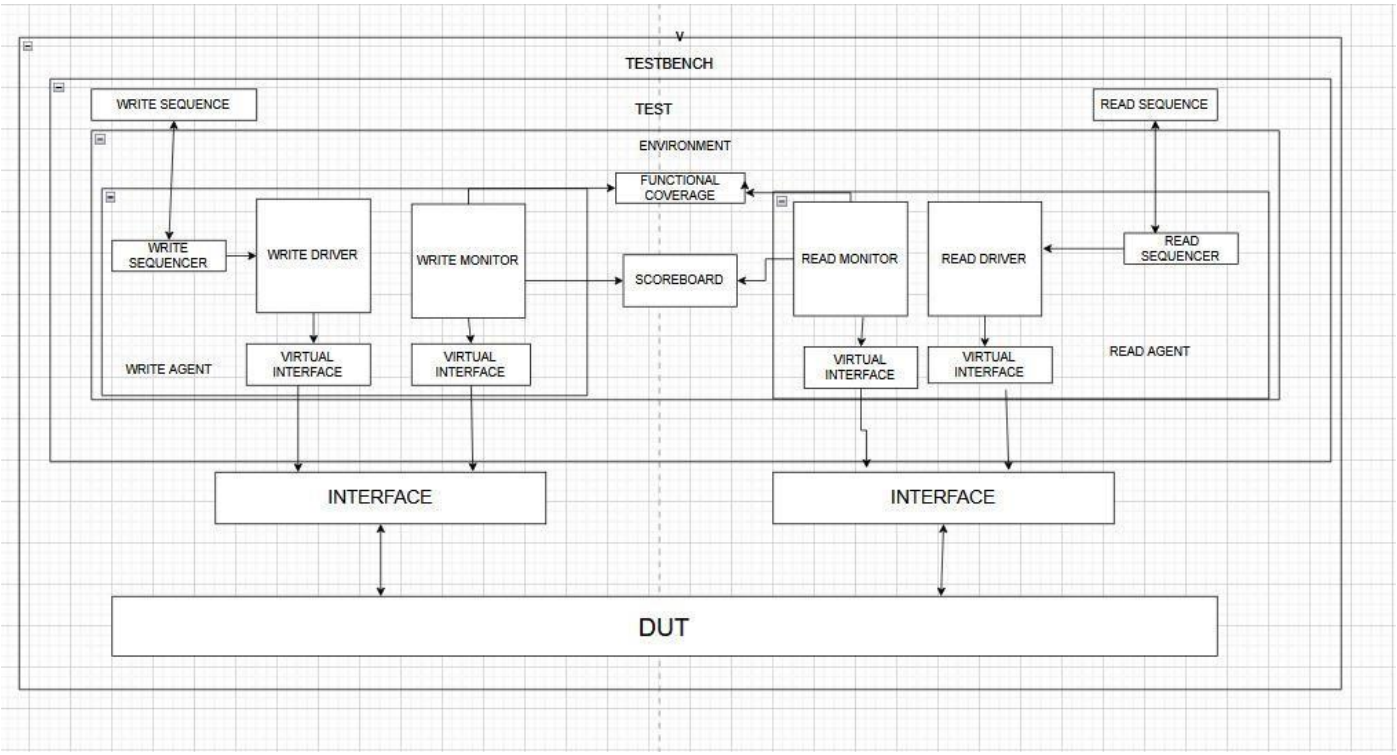
UVM Testbench Verification Plan

Below is the implementation of the UVM testbench hierarchy.



In this UVM Hierarchy, we have a testbench module in which we instantiate the test class.

In test class, we create an instance of environment and we also have test libraries declared here.

Test Cases :

1) Fifo_base_test

2) fifo_full_and_Empty test

3) Fifo_random_test

4) Concurrent_wr_rd_test

In Environment we had 2 agents instantiated, known as Read Agent and Write Agent. Agent is a key component in the UVM verification environment, primarily used for testing and verifying. It encapsulates the mechanisms for generating stimulus, collecting responses, and monitoring the behaviour of the design under test (DUT). Typically, a UVM agent includes a sequencer for driving test scenarios, a driver for applying these scenarios to the DUT, a monitor for observing and collecting output from the DUT, and an analysis component to evaluate the DUT's performance against expected results.

Write Agent:

Write Sequence: Generates the transactions.

Write Sequencer: It is responsible for generating and orchestrating the flow of the transactions to the driver.

Write Driver: Drives the transaction level details to signal level details.

Read Agent:

Read Sequence: Enables the read pointer.

Read Sequencer: It is responsible for generating and orchestrating the flow of the transactions to the driver.

Read Driver: Drives the transaction level details to signal level details.

Write & Read Monitor: It observes and collect data from the interface of the design under test (DUT) without influencing its behavior and broadcasts this data to scoreboard and coverage components.

Functional Coverage: This class is mainly focussed to verify the functionality of the DUT. Thus, it contains multiple covergroups and coverpoints.

Scoreboard: It is the golden DUT used to compare data with the original DUT.

Github Link : https://github.com/swaroop-chandrashekar/team_9_Aysnc_FIFO.git