Team2: Hierarchy of class based testbench.

The hierarchy of class based testbench architecture for verifying an asynchronous FIFO design.

1. Top-Level Testbench (`asynchFIFO_tb.sv`):

- Serves as the entry point for simulation and testbench control.
- Adapts the simulation flow by invoking compilation, elaboration, and execution of test sequences.

2. Testbench Modules:

a. Definitions (`definitions.sv`):

- The "definitions" package provides parameter definitions commonly used across the asynchronous FIFO design and testbench. It includes parameters, which specify the size and behavior of the FIFO. This package ensures consistency and ease of parameter management throughout the design and testbench components.

b. Transaction (`transaction.sv`):

- The "transaction" class represents a single transaction in the asynchronous FIFO testbench. It encapsulates the write data (wdata) as a random variable and provides a display method to print the transaction details. This class serves as a container for transaction data and facilitates modular transaction generation and manipulation within the testbench environment.

c. Environment (`environment.sv`):

- The "environment" class encapsulates the test environment for an asynchronous FIFO design in SystemVerilog. It coordinates the generation and driving of transactions between the testbench and the DUT through the "generator" and "driver" instances, respectively. A mailbox acts communication between the generator and driver. The class defines tasks for pre-test initialization, test execution, and post-test verification. During the test, the environment adapts the generation and driving of transactions, ensuring synchronization and completion criteria are met before finishing the simulation. Overall, the environment class provides a structured framework for verifying the functionality and performance of the asynchronous FIFO design.

d. FIFO Interface (`fifoInterface.sv`):

- Declares the interface signals required for communication between the testbench and the DUT.This interface facilitates seamless communication and interaction between the testbench and the DUT during verification.

e. Driver (`driver.sv`):

- The "driver" class serves as the component responsible for driving stimuli to the asynchronous FIFO DUT (Design Under Test). It manages the generation of write and read transactions, ensuring proper synchronization with the FIFO's clock domains. The class maintains counters for write and read cycles, ensuring that write operations occur at the specified WRITE_PERIOD interval and read operations occur at the specified READ_PERIOD interval. Additionally, it handles reset operations and tracks the number of transactions and burst counts. The class communicates with the testbench interface through a mailbox mechanism, enabling seamless interaction with the test environment.

f. Test (`test.sv`):

- Defines test scenarios and sequences to be executed by the testbench.
- The "test" program serves as the top-level entity in the class-based testbench for the asynchronous FIFO. It instantiates an environment object and sets up the test environment by configuring the generator's repeat count. Then, it initiates the test by calling the environment's "run" task, which starts the execution of the generator and driver main tasks.

g. Monitor (`monitor.sv`):

- The "monitor" class serves as a passive observer in the class-based testbench for the asynchronous FIFO. It samples the interface signals of the FIFO design under test and constructs transaction objects representing the observed behavior. These transaction objects are then sent to the scoreboard for analysis via a mailbox mechanism. The monitor operates continuously, sampling both the write and read interface signals, ensuring comprehensive monitoring of the FIFO's behavior during simulation.

h. Scoreboard(`scoreboard.sv`):

- The "scoreboard" class functions as the verification component in the class-based testbench for the asynchronous FIFO. It receives transaction objects representing observed behavior from the monitor via a mailbox mechanism. The scoreboard then compares the received data with the expected data stored in memory. If there is a mismatch, it reports an error, indicating the inconsistency between the expected and actual results. Additionally, it keeps track of the number of transactions processed for monitoring purposes.

3. Communication Flow:

- The testbench modules interact through well-defined interfaces and transactions.
- The environment instantiates the DUT and connects it to the testbench components via interface signals.
- The test module adapts the test sequence, driving transactions through the driver, and monitors DUT outputs for correctness.
- Coverage and results analysis are also included to ensure comprehensive verification which includes Code coverage and Functional Coverage.

Overall, this hierarchical structure facilitates modular design, reusability, and scalability of the testbench, enabling efficient verification of the asynchronous FIFO design. Each component plays a crucial role in the testing process, contributing to thorough verification and validation of the DUT.