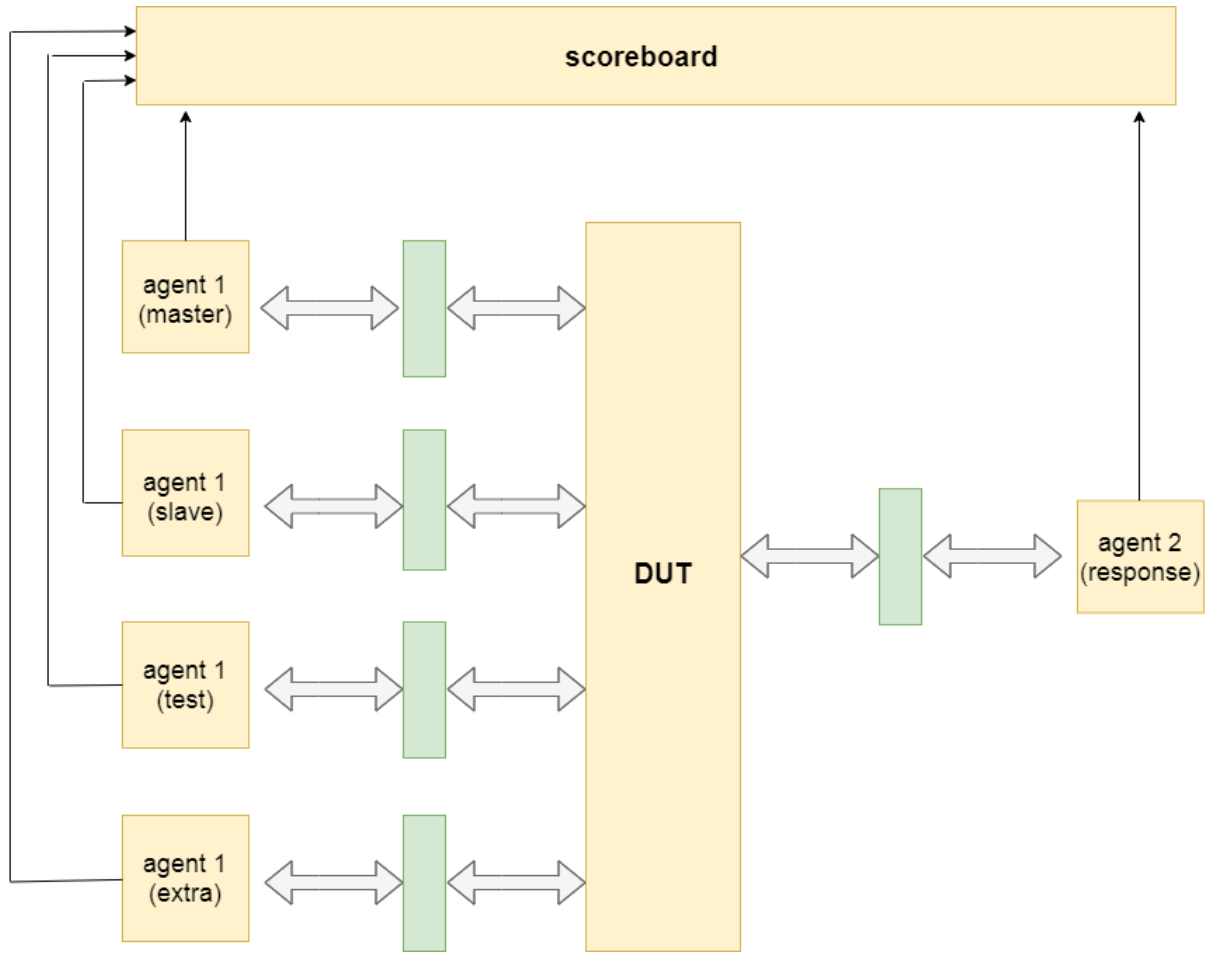
**Name: Guy Arad, Date: 02.09.2021, Course: SQLabs Verification**

# Arbiter Verification Program

## Block Diagram:




## General:

The verification program consist of 1 agent and 1 interface for the hosts (each receives different instances of them), and also 1 agent and 1 interface for the slave response. It also contains the arbiter DUT and a scoreboard.

The hosts' agents push random transactions to the DUT, and the slave response returns an answer to the corresponding host. The corresponding host then pushes a new transaction. The process repeats itself until all the transactions have completed.

The scoreboard compares the host transactions to the corresponding response transactions. If an ack was received, it will also compare the ack and the data read. It uses the round-robin algorithm to keep track of the hosts queue.

Inside the verification program there are also assert tests, and coverage sampling.

Agents:

2 types of agents - active agent for the hosts, and active agent for the slave response.
Each host receives a different instance of the hosts' agent.

The host's agent has a sequencer, a driver and a monitor.
The slave response's agent has a driver and a monitor.

The host's sequencer generates disable/read/write transactions and sends them to the driver, which pushes them to the DUT.

The host's monitor checks the interface twice for each transaction (it checks the inputs at the beginning, and the outputs after an ack was received), and it sends them as two transactions to the scoreboard.

The slave response's monitor does a similar thing. The difference is, if a timeout has occurred then the second transaction won't be sent. Also, if the arbiter has entered idle state, then it sends an empty transaction to the scoreboard, to inform it.

The slave response's driver generates data read and ack and pushes them to the DUT.

Interfaces:

2 types of interfaces - 1 for the hosts, and 1 for the slave response.
Each host receives a different instance of the hosts' interface.

Each interface contains the 6 inputs (cpu,addr,rd,wr,be,dwr) and 2 outputs (drd,ack), in addition to the clock & reset signals.

There's only 1 difference - the hosts' ack is 2 bit, and the slave response's ack is 1 bit.

Scoreboard:

The scoreboard contains 5 fifos (one for each agent).

Each fifo receives transactions from his corresponding agent.

The scoreboard compares the current transaction from the slave response's fifo to the corresponding transaction from a host fifo.

Each loop cycle, it compares the initial transaction (with only inputs) from the slave response to the corresponding host transaction. It then waits for the complete transaction (with the ack and data read) to arrive to the slave response's fifo and the host's fifo, and it does another comparison. If a timeout has occurred, then it skips the second comparison.

The scoreboard maintains a round-robin algorithm, to keep track of the hosts queue. If an empty transaction (no read and no write) was received, then we move to "idle state".

Type of sequences:

1. Sequence of disable transactions.
2. Sequence of write transactions.
3. Sequence of read transactions.
4. Sequence of random (disable/write/read) transactions.
5. Sequence of sequences.

Main Test:

The main test sends a big number of random transactions for each host. It generates a random delay for each transaction, and occasionally creates edge cases like timeouts and idle state. Also, there are assert tests, and coverage sampling.

Asserts:

1. Checking that the slave response's inputs reset after an ack.
2. Checking that exactly 1 host has cpu='1', at all times.
3. Checking that the slave response doesn't create ack for empty transactions.
4. Checking that the slave response doesn't get rd='1' and wr='1' at the same time.
5. Checking that only one host receives the ack.
6. Checking that the ack resets after exactly one clock cycle.
7. Checking that the inputs don't change in the slave response's interface, while waiting for an ack.

Coverage:

1. Code coverage.
2. Functional coverage for all the hosts signals.
3. Functional coverage for interesting values (round-robin state & amount of hosts currently active).