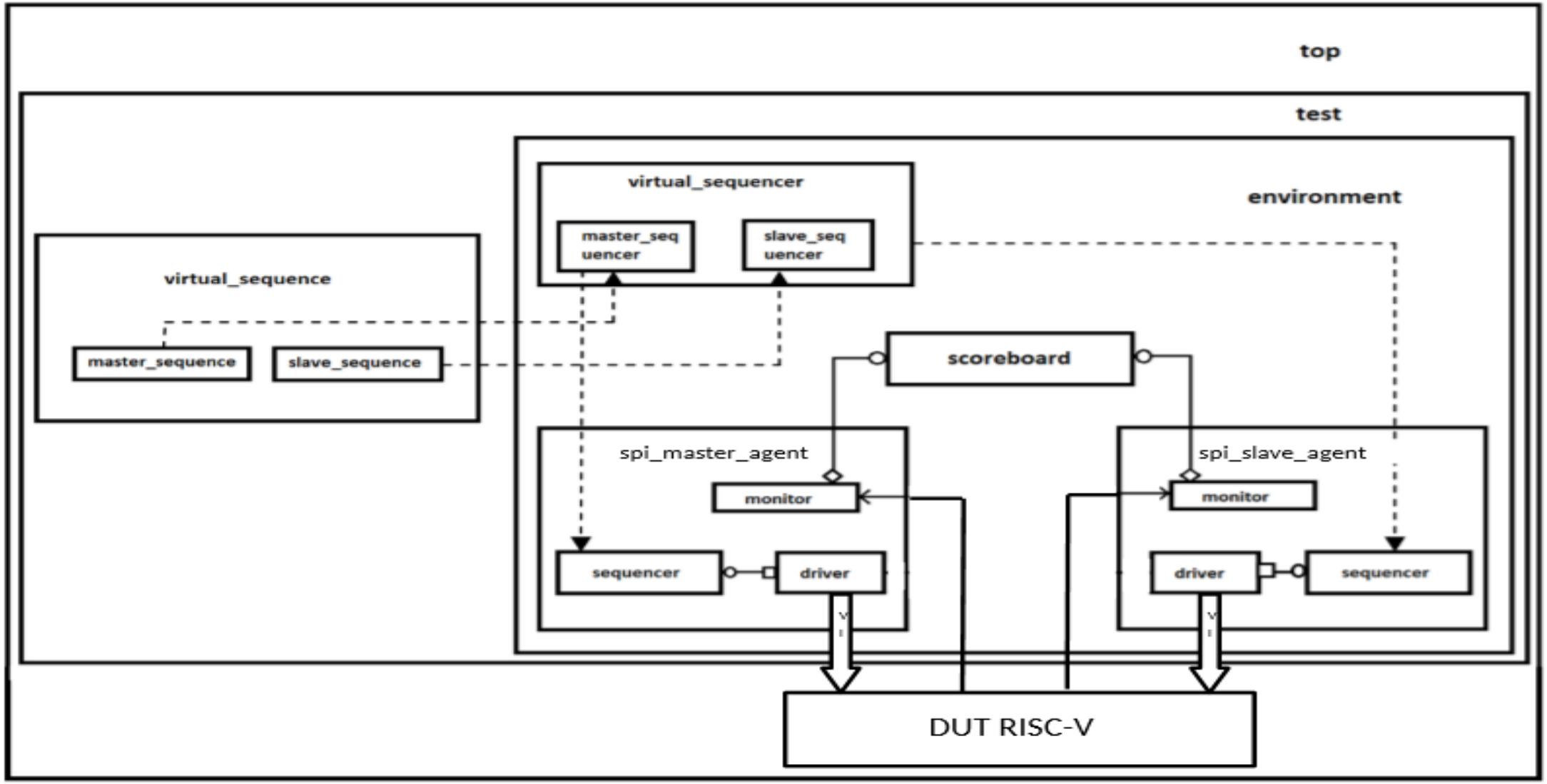
SERIAL PERIPHERAL INTERFACE VIP

SPI Testbench Architecture



SPI VIP UVM Testbench Components

Interface:
The SPI interface spi_if contains signals information. The timings of the respective signals are defined by the clocking blocks.
The master_cb defines the timings from the bus-master point of view, while the slave_cb defines the timings from the bus-slave point of view. Also has the modport lists, which define the directions of the signals within the interface

Transactions:
The SPI master_transaction & slave_transaction class defines the spi based on data length, shift direction and other characteristics

Sequence:
The SPI master_sequence & slave_sequence creates the transaction of the SPI being generated. Sequences are made up of several data items which can be put together in different ways to create scenarios

Sequencer:
The SPI master_sequencer & slave_sequencer controls the flow of request and response sequence items between sequences and the driver

Driver:
The SPI master_driver & slave_driver receives the master_transaction & slave_transaction through the seq_item_port. It defines how signal should be timed so that target protocol becomes valid. Transaction level objects are obtained from sequencer & driver drives them to design via interface handle

Monitor:
The SPI master_monitor & slave_monitor collects bus or signal interface through virtual interface. Collected data used for protocol checking & coverage. Collected data is exported via analysis port.

Agent:
The SPI master_agent & slave_agent encapsulates respective sequencer, driver & monitor into single entity & connecting together via TLM interfaces.

Agent_Configuration:
Here, master_agent_configuration & slave_agent_configuration configures agent as Active or Passive

Scoreboard:
Compares the data from master_monitor & slave_monitor. Receive data via analysis port

Environment:
To provide a conclusion, env class contains all the verification components

Test:
Verification plan lists all the features & other functional items that needs to be verified & test needs to cover each of them

Top:
The top module, which instantiates the DUT module, registers the spi_if in the database, and runs the test. The top module is responsible for clock generation as well.

Virtual Sequence:
Tests that require coordinated generation of stimulus using multiple driving agents need to use virtual sequences. A virtual sequence can run multiple transaction types on multiple real sequencers

Virtual Sequencer:
If we have multiple driving agents and stimulus coordination is required, we need a virtual sequencer