SPI UVM TESTBENCH ARCHITECTURE

INTRODUCTION:

UVM test bench components are constructed by extending UVM classes.

SPI has master slave configuration, can have either single slave or multiple salve as shown below and short description of the signals.

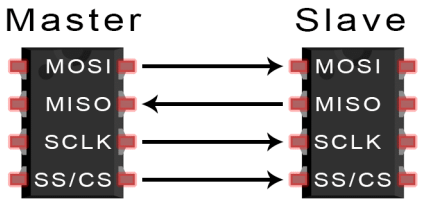

Fig 1: SPI configuration with a Master and a Slave

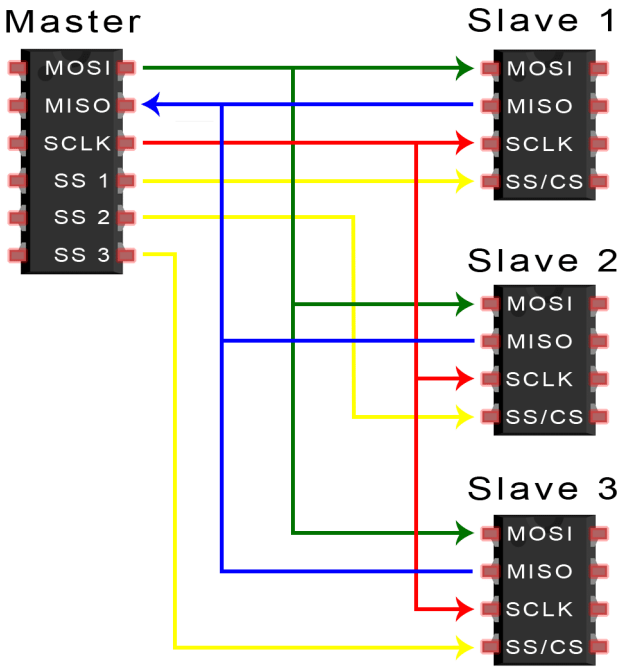

Fig 2: Multislave SPI Configuration

4 WIRE SPI DEVICES HAS FOUR SIGNALS AS BELOW:

1.Clock(SCLK),master generates the clock.

2.Chip select(SS/CS),From master used to select slave.

3.Master Out Slave In(MOSI),Transmits data from master to the slave.

4.Master In Slave Out(MISO),Transmits data from slave to the master.

ARCHITECTURE:

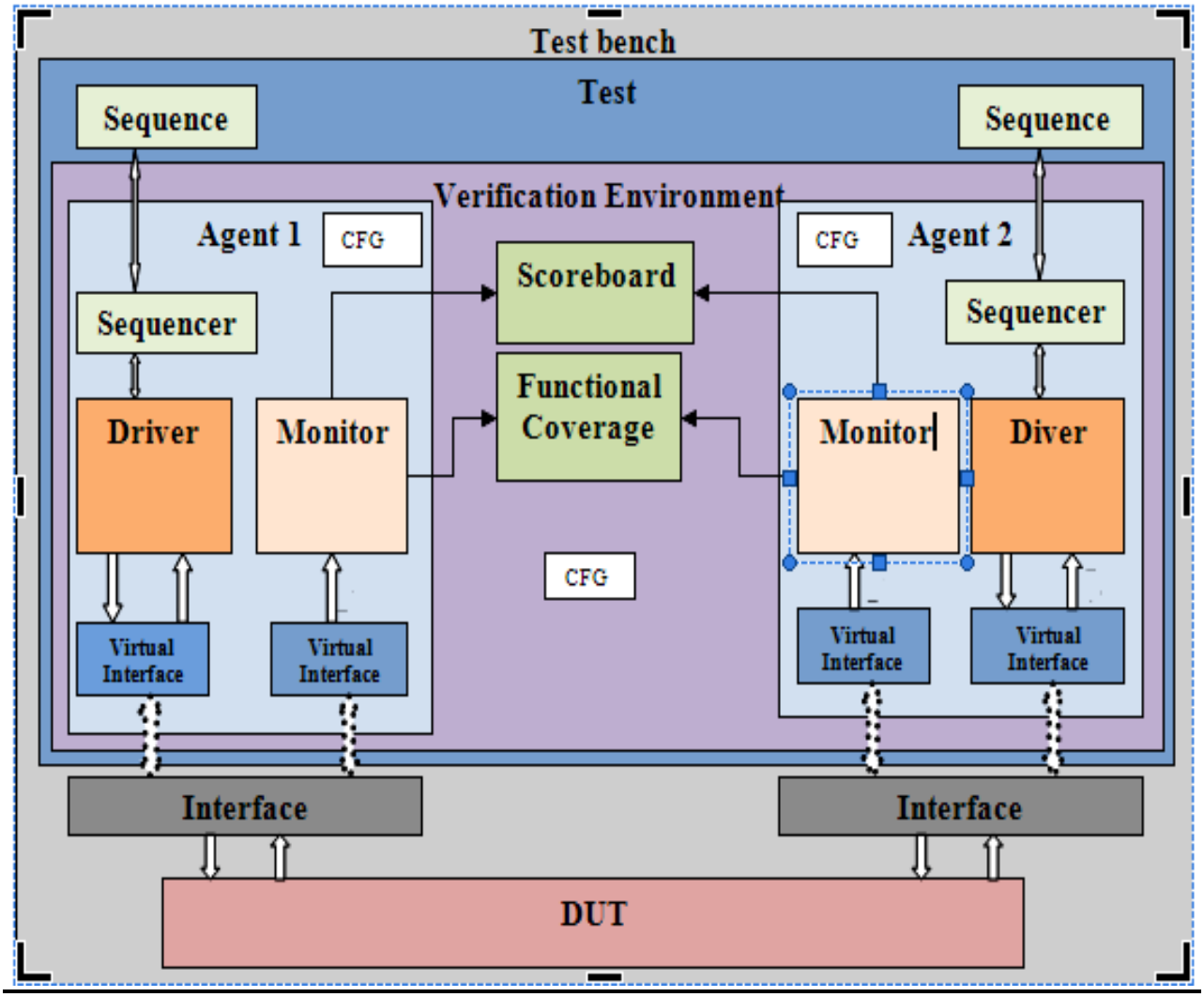

Fig 3:uvm testbench architecture

COMPONENTS:

1.UVM TB TOP:

TestBench top is the module, it connects the DUT and Verification environment components.

2.TEST:

Test is the topmost class and is responsible for

- Configuring the testbench,
- Constructing the lower classes like environment
- Initiating the stimulus

3.ENVIRONMENT:

User-defined environment is derived from uvm_env, uvm_env is inherited from uvm_component. Environment is the container class, It contains one or more agents, as well as other components such as scoreboard, top level monitor, and checker.

4.UVM AGENT(UNIVERSAL VERIFICATION COMPONENT):

User-defined agent is extended from uvm_agent, it is inherited by uvm_component.An agent typically contains: a driver, sequencer, and monitor. Agents can be configured either active or passive

5.UVM SEQUENCE ITEM:

Sequence item consist of data fields required for generating the stimulus.In order to generate the stimulus, the sequence items are randomized in sequences. Therefore data properties in sequence item should generally be declared as rand and can have constraints defined.

6.UVM SEQUENCE:

Sequence generates series of sequence_item's and send to the driver via sequencer, Sequence is written by extending the uvm_sequence.

7.UVM DRIVER:

Responsible for driving the stimulus to DUT according to the protocol using virtual interface fetching the stimulus from sequencer.

8.UVM SEQUENCER:

The sequencer control the flow of request and response sequence items between sequences and the driver,Responsible for routing the data sequence_item generated in sequence to the driver or vice verse.

9.UVM MONITOR:

Extracts signal information from the dut via virtual interface and translates the pin level to transaction level.

10.UVM SCOREBOARD:

Receives data from monitors and compares the expected and the actual values.

11. UVM CONFIG DB:

The configuration database provide access to a centralized database, where type specific information can be stored and received. Config_db can contain scalar objects, class handles, queues, lists, or even virtual interfaces.

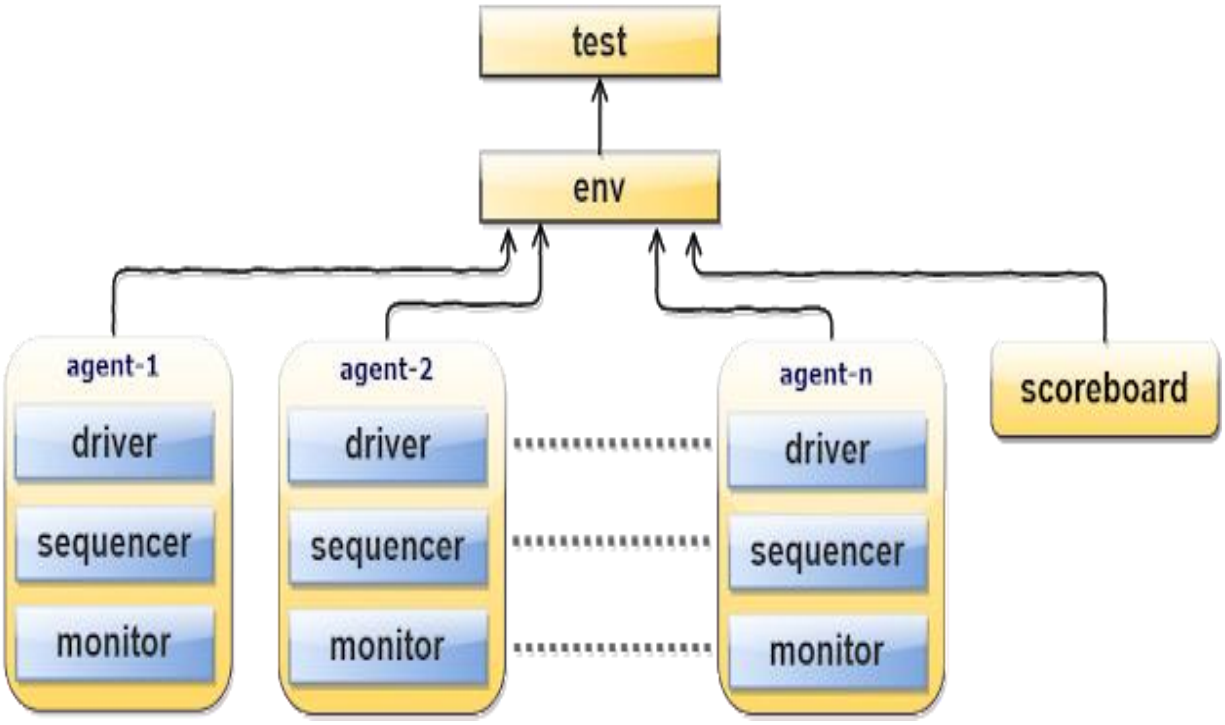

Fig3:Multislave configuration(multiple agents for slave)