**Etc..Transaction Class: -**

* It’ s contains data members that represent the various signals or values that are part of the transaction.

**Generator: -**

* Responsible for stimulus/traffic (transaction items) generation and keep the same in mailbox which is further processed by driver.

i.e., stimulus generation could be through randomization (preferred), through a file,

hardcoded values, DPI etc.

**Driver: -**

* Who's responsible to takes transaction or sequence level activity(stimulus) coming from generator and convert it into the pin or system level activity.
* Driver drives the pin or system level activity via interface (bus) as per the protocol.
* It basically drives input data to design adhering to the protocol.

**Monitor: -**

* Who's responsible to take pin or system level activity coming from bus (interface) and convert it into the transaction or sequence level activity.
* Monitor collect transaction or sequence level activity from interface (bus) as per the protocol.
* It basically sample/monitor interface data adhering to the protocol and send it to other component (i.e. scoreboard, reference model/predictor, coverage collector etc.)

**Reference Model: -**

* Reference model is a verification component where you write a logic to generate expected output (checker logics). (Predicting the output). Also known as predictor.

**Scoreboard: -**

* Scoreboard is responsible to check whether your design output is correct or not.
* It's collects expected value from reference model, actual value from monitor and compare those values and log the status.
* Functional Coverage can be part of Scoreboard

**Why virtual interface? :-**

* **Interface is a static standalone entity whereas our verification components are class, and because of this the interface cannot be instantiated inside this dynamic components (i.e. driver, monitor, etc.), hence we need virtual interface.**

**What virtual interface? :-**

* **Virtual interface is just a pointer, who points to actual interface.**

**Virtual\_interface = actual\_interface**

**h1 = h2**

**virtual <intraface\_name> <instance\_ name>;**

**Testcase Steps (ref sanity)**

1. Apply reset

2. Do write operation with address 5 and data 10

3. wait for one cycle

4. Do read operation on address 5.

5. verify the read data is 10.