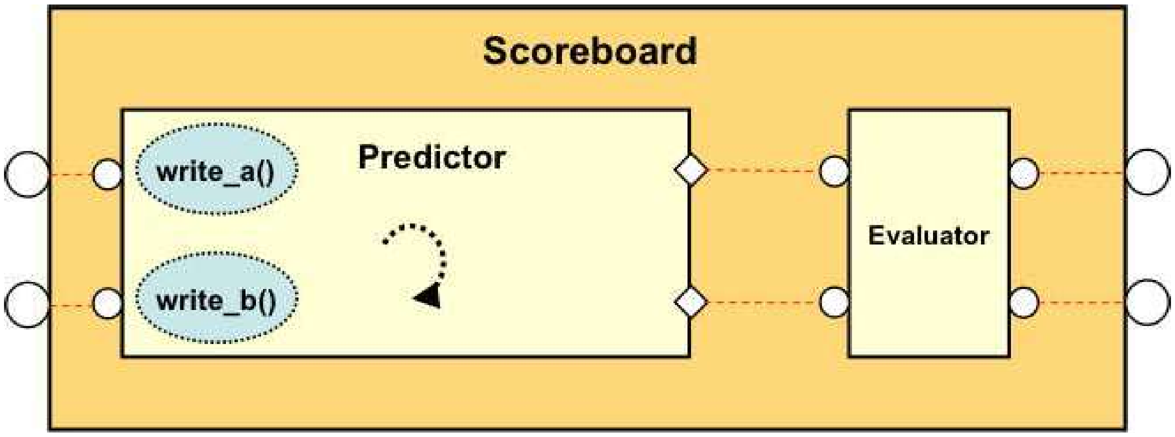
Predictors

Overview

A Predictor is a verification component that represents a "golden" reference model of all or part of the DUT functionality. It takes the same input stimulus that is sent to the DUT and produces expected response data that is by definition correct. When you send random stimulus into your DUT, you need an automatic way to check the result as you can no longer check the output by hand. A Predictor generates expected output that is compared to the actual DUT output to give a pass / fail.

Predictors in the Testbench Environment



Predictors are the part of the Scoreboard component that generates expected transactions. They should be separate from the part of the Scoreboard that performs the evaluation. A Predictor can have one or more input streams, which are the same input streams that are applied to the DUT.

Construction

Predictors are typical analysis components that are subscribers to transaction streams. The inputs to a Predictor are transactions generated from monitors observing the input interfaces of the DUT. The Predictors take the input transaction(s) and process them to produce expected output transactions. Those output transactions are broadcast through analysis ports to the evaluator part of the scoreboard, and to any other analysis component that needs to observe predicted transactions. Internally, Predictors can be written in C, C++, SV or SystemC, and are written at an abstract level of modeling. Since Predictors are written at the transaction level, they can be readily chained if needed.

Example

```
class alu_tlm extends uvm_subscriber #(alu_txn);
 `uvm_component_utils(alu_tlm)

 uvm_analysis_port #(alu_txn) results_ap;

 function new(string name, uvm_component parent );
  super.new( name , parent );
 endfunction

 function void build_phase( uvm_phase phase );
```

```
  results_ap = new("results_ap", this);
 endfunction

 function void write( alu_txn t);
  alu_txn out_txn;
  $cast(out_txn,t.clone());
  case(t.mode)
   ADD: out_txn.result = t.val1 + t.val2;
   SUB: out_txn.result = t.val1 - t.val2;
   MUL: out_txn.result = t.val1 * t.val2;
   DIV: out_txn.result = t.val1 / t.val2;
  endcase
  results_ap.write(out_txn);
 endfunction

endclass
```

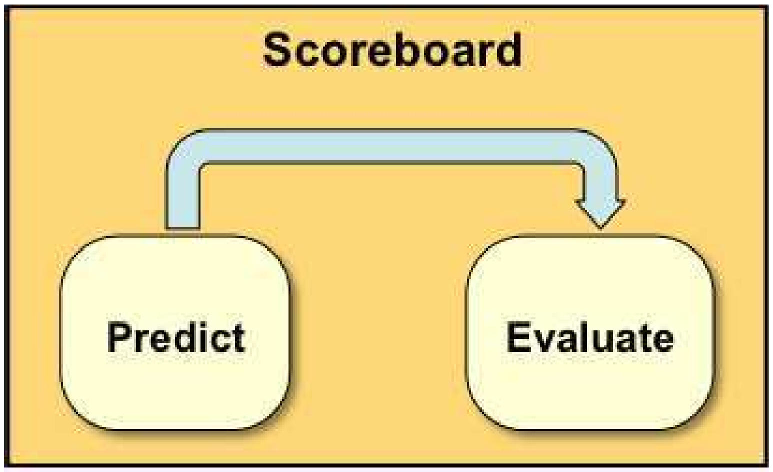
Predictor as a Proxy for the DUT

Another use of a Predictor is to act as a proxy DUT while the DUT is being written. Typically, since the Predictor is written at a higher level of abstraction, it takes less time to write and is available earlier than the DUT. As a proxy for the DUT, it allows testbench development and debugging to proceed in parallel with DUT development.

Scoreboards

Overview

The Scoreboard's job is to determine whether or not the DUT is functioning properly. The scoreboard is usually the most difficult part of the testbench to write, but it can be generalized into two parts: The first step is determining what exactly is the correct functionality. Once the correct functionality is predicted, the scoreboard can then evaluate whether or not the actual results observed on the DUT match the predicted results. The best scoreboard architecture is to separate the prediction task from the evaluation task. This gives you the most flexibility for reuse by allowing for substitution of predictor and evaluation models, and follows the best practice of separation of concerns.



In cases where there is a single stream of predicted transactions and a single stream of actual transactions, the scoreboard can perform the evaluation with a simple comparator. The most common comparators are an in-order and out-of-order comparator.