

OCP Verification Component

The Verilab OCP *uVC* is a *mixed language* verification component that supports a major subset of *Open Core Protocol Specification 2.2 (OCP)*.

The OCP *uVC* is implemented using *SystemVerilog* and *e* verification languages and is compliant with both the *Open Verification Methodology (OVM)* and *e Reuse Methodology (eRM)*.

Applications

The OCP *uVC* enables validation of OCP interface behaviour using constrained-random and coverage-driven verification techniques, including sequence-based stimulus generation, assertion-based protocol checking and functional coverage.

The verification component can be used as an *OVM Verification Component (OVC)* in SystemVerilog-only applications without the Specman layer (or license), or it can be used in Specman-based verification environments as a regular *e Verification Component (eVC)*.

Benefits

The OCP *uVC* can be used as a more *cost-effective* replacement for existing OCP *eVCs*, or as a *superior* solution in new verification environments.

This state-of-the-art implementation of a fully functional SystemVerilog *OVC* provides a direct, or migration, path to SystemVerilog-based environments.

The combination of profile-based configuration, transaction level encapsulation, SystemVerilog signal interface and SVA property checks ensure reliable operation and ease of integration.

Contact

Please contact ocp_info@verilab.com for additional information and license costs.

Key Features

- ★ OCP 2.2 verification component
- ★ OVM and eRM compliant
- ★ Multiple use-models
 - SystemVerilog *OVC*
 - Specman *eVC*
- ★ Multiple tool support
 - simulators supporting OVM
- ★ Cost-effective, easy to use and integrate
- ★ Constrained-random sequences
- ★ Protocol checking
- ★ Functional coverage

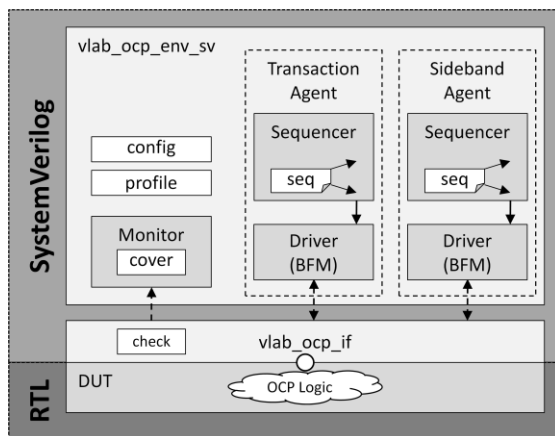
User Features

- ★ Proactive Master sequence stimulus
- ★ Reactive Slave response generation
- ★ Passive Monitor coverage and hooks
- ★ SVA protocol checks
- ★ SystemVerilog signal interface
- ★ Profile-based configuration
- ★ Separate transaction and sideband sequence drivers
- ★ *uVC* model control
 - control pipeline depth
 - force response order
- ★ Logical transaction encapsulation
 - single transaction bursts
 - constrain all aspects together
 - monitor homogenous transaction
- ★ Major subset of OCP behaviour
 - support for all features planned
 - most implemented and released

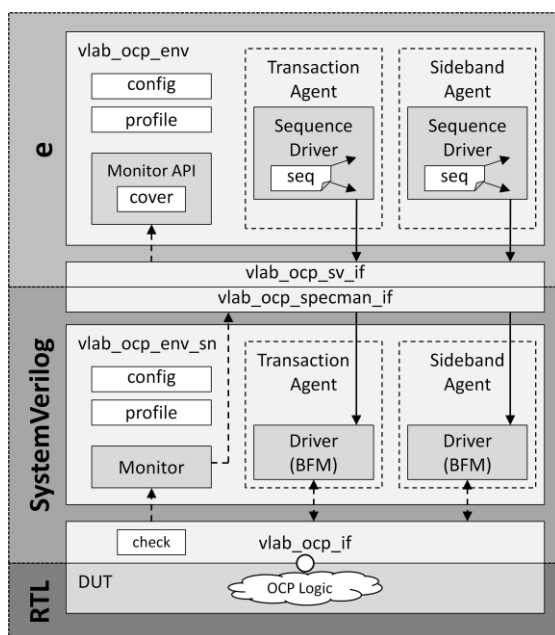
Architecture

All of the complex OCP components (such as drivers, bus functional models (BFMs), monitors and protocol checks) for the OCP uVC are implemented in SystemVerilog and connected to the device-under-test (DUT) using a SystemVerilog interface.

The higher-level components (such as config, profile, coverage, constraints, sequences and monitor API) are active either in *e* or SystemVerilog depending on the use-model, as shown in the diagrams:



SystemVerilog OVC Architecture



Specman eVC Architecture

Operating Modes

Each instance of the OCP uVC can be individually constrained to behave as a *pro-active Master* (initiating transactions), a *re-active Slave* (responding to transactions) or a *passive Monitor* (monitoring traffic, collecting coverage and checking protocol).

Configuration

The OCP parameter configuration is encapsulated in *profile* objects which are validated at run-time using configuration compliance checks from OCP 2.2. All aspects of OCP uVC behaviour are *profile-aware*, including checks, monitoring and connectivity. Additional configuration fields are available to control uVC model behaviour (e.g. force response order, or limit pipeline depth).

Stimulus Constraints

The transaction and sideband stimulus are independently controlled by constraining sequence and item generation. The user has full control over all aspects of transactions including field values and corresponding delays. OCP transactions are encapsulated as a single homogeneous object (i.e. all beats of a burst are part of the same transaction) enabling efficient constraining and monitoring, even when the OCP bus reorders the responses.

Integration & Coverage

The Monitor API provides hooks for integrating to external verification environment components (such as scoreboards) and efficient per-instance functional coverage for all operating modes.

Protocol Checks

All of the required Protocol Compliance Checks identified in OCP 2.2 are implemented using SVA property assertions.