



## Open Verification Methodology

The **Open Verification Methodology (OVM)** provides the first open, interoperable SystemVerilog verification methodology in the industry. The OVM provides a library of base classes that allow users to create modular, reusable verification environments in which components talk to each other via standard transaction-level modeling (TLM) interfaces. It also enables intra- and inter-company reuse through a common methodology and classes for virtual sequences and block-to-system reuse. As a joint development initiative between Mentor Graphics® and Cadence® Design Systems, the OVM is supported on multiple verification platforms making it the de facto standard methodology that is ideally suited to both novice and expert verification engineers.

### Open Verification Methodology (OVM)

Built on the success of the Advanced Verification Methodology (AVM) from Mentor Graphics and the Universal Reuse Methodology (URM) from Cadence, the OVM brings the combined power of these two leading companies together to deliver on the promise of SystemVerilog. The OVM offers established interoperability mechanisms for verification IP (VIP), transaction-level and RTL models, and full integration with other languages commonly used in production flows.

### Benefits

#### Interoperability

- Jointly developed and tested on both the Mentor Graphics Questa® and Cadence Incisive® verification platforms
- Utilizes standard transaction-level modeling (TLM) interfaces to improve modularity and reuse
- Provides architecture, utilities and infrastructure, including specialized base classes, to create higher-level verification components for block-to-system and project-to-project reuse
- Can be adopted incrementally to enhance existing module-based testbench methodologies
- Backward-compatible with AVM 3.0 and URM 6.2

#### Advanced Capabilities

- Unsurpassed flexibility and configurability allows test case customization without rewriting the underlying code
- Standard test phases for coordinating activities of multiple components, which are customizable to suit any organization's needs
- Unified customizable message formatting and reporting.
- Powerful and flexible sequence specification for test scenarios, from simple stimulus to complex multi-layer protocols

#### Open Source

Distributed under the standard open-source Apache™ 2.0 license, the OVM base classes and examples may be downloaded and distributed freely. Access to the source code simplifies debug, enhances code quality, facilitates collaboration, and ensures adherence to the IEEE-1800 SystemVerilog standard.

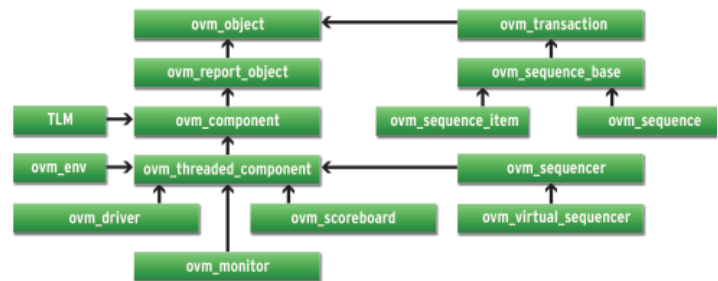


Figure 1. The OVM object hierarchy

The OVM code runs on any SystemVerilog-compliant simulator, and the open-source distribution greatly accelerates the adoption of SystemVerilog throughout the verification ecosystem.

### OVM Base Class Library

#### Verification Infrastructure

The OVM provides all the building blocks needed to construct a suitable test environment (Figure 1). Components may be encapsulated and instantiated hierarchically and are controlled through an extendable set of phases to initialize, run, and complete each test. These phases are defined in the base class library but can be extended to meet specific project needs.

#### Transaction-Level Modeling

The OVM components communicate via standard TLM interfaces, which improves reuse. The TLM Standard, originally developed by OSCI, defines a standard set of interface methods to define communication semantics but separates the interfaces from their implementations. Using a SystemVerilog implementation of TLM in the OVM, a component may communicate via its interface to any other component that implements that interface. Thus, one component may be connected at the transaction level to others implemented at multiple levels of abstraction. The common semantics of TLM communication permit components to be swapped in and out without affecting the rest of the environment.

### OVM Messaging

To facilitate debug, results checking, and overall consistency, the OVM includes a standard set of methods for reporting messages to *stdout* and/or to one or more text files. Messages may be controlled using verbosity or other customizable filters on an individual or hierarchical basis. Standard severity levels (INFO, WARNING, ERROR, FATAL) are supported.

### OVM Automation

OVM users also have access to advanced automation to simplify environment creation. Macros dramatically reduce coding and improve readability. Alternatively, users can access the same capabilities through function calls.

### Flexible Configuration

The OVM uniquely provides three forms of configuration to decouple the “test” from the “testbench.” The testbench is the complete topology for the specific components needed to perform verification, while the test becomes a straightforward configuration of that topology. These three forms of configuration are all available at run time as follows:

- The test may choose from a set of precompiled environments
- The chosen environment and the components therein may be configured to modify their behavior and/or layout
- Stimulus sequences may be added or modified on-the-fly to exercise different scenarios

The key benefit of this unique configuration capability is that the test writer is able to modify the behavior of the underlying environment without having to modify or extend the code.

## Sequential Stimulus

The OVM provides class support for building hierarchical sequences, making it easy to specify complex layered stimulus scenarios. By defining sequences as objects, the OVM separates the test behavior from the testbench structure, allowing greater modularity and reuse of stimulus scenarios. At any level of abstraction, sequences enable a simple test writer interface, including built-in semantics for controlling the randomization of transactions.

### Simple Test Writer Interface

The OVM sequences are written at the transaction level, independent of the underlying testbench infrastructure. The test writer simply specifies procedurally the type of transaction to be generated, along with an optional set of constraints and the transaction is automatically randomized and sent to the driver. In addition, sequences may call subsequences or may spawn parallel sequences (Figure 2). New tests are created simply by specifying new sequences, which can be executed via simple configuration calls from the test without modifying the underlying environment.

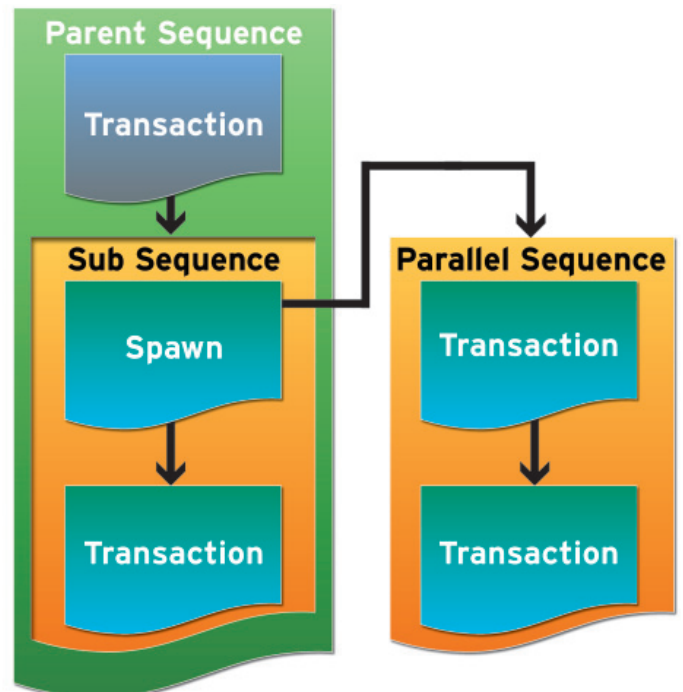


Figure 2: Hierarchical sequences

### Sophisticated Virtual Sequences

The OVM sequences can also be layered hierarchically to build and maintain sophisticated protocols as well as a library of tests. Virtual sequences may be used to coordinate the execution of multiple sequences on different interfaces, allowing more sophisticated tests to exercise more complex corner cases of the design under test (DUT).

While sequences provide the stimulus, scoreboards monitor the results. The OVM implements scoreboards using configurable classes, making them easy to build and maintain.

## System Level Verification

The use of TLM interfaces in the OVM naturally leads to the development of verification environments at the transaction level. This facilitates the early development of testbenches against SystemC™ models prior to RTL coding. The modularity and reuse provided in the OVM allows these system-level models to be refined down to RTL while still exposing the same TLM interfaces to the test environment. The original SystemC model can also be incorporated back into the environment as a golden model to help verify that the RTL behaves correctly.

At a higher level, the OVM architecture facilitates the development of protocol-level golden VIP for plug-and-play reuse. These OVM components encapsulate all of the functionality needed to verify a particular protocol or interface, including generating protocol-level stimulus, signal-level behavior and

checking, and functional coverage. The built-in architectural management capabilities then enable the VIP built for block-level verification to be encapsulated and reconfigured for system-level verification without modifying the code in the VIP (Figure 3).

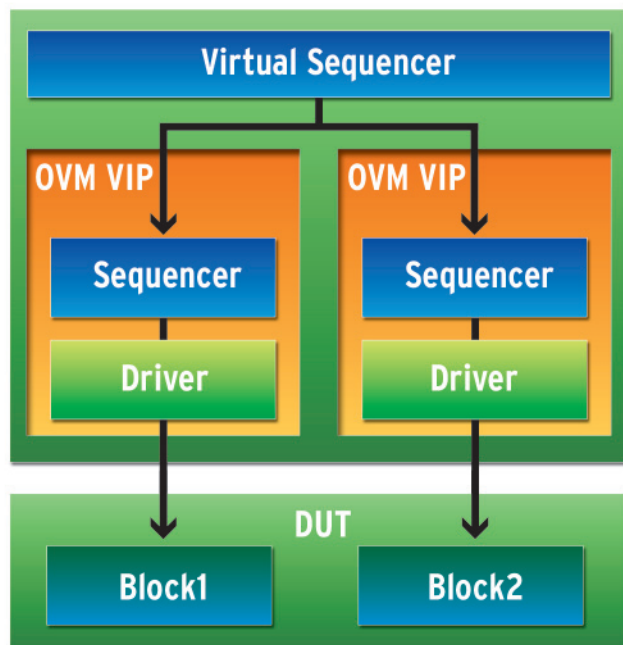


Figure 3: Verification IP

## The OVM World

### Ecosystem

The OVM inherits the well-established Cadence and Mentor verification ecosystems. This ensures that OVM users have multiple sources for training, VIP, and verification services.

### Support

The OVM is jointly supported by both Mentor Graphics and Cadence through normal support channels. Each company offers complete support for the entire OVM offering, including documentation and examples. The open-source nature of the OVM fosters contributions and feedback from the user community, which will be evaluated by both companies and potentially included in future releases.

### Learning More

Cadence and Mentor Graphics have jointly created the OVM World <http://www.ovmworld.org>. At this site you can find the OVM download, including source code, examples, and documentation. You can also find additional information such as a white paper, events calendar, and the OVM partner links.

Copyright © 2007 Mentor Graphics Corporation and Cadence Design Systems, Inc.  
All company or product names are the registered trademarks or trademarks of their respective owners.

**cadence**<sup>™</sup>

2655 Seely Avenue  
San Jose, CA 95134  
Phone: 408.943.1234  
Fax: 408.428.5001  
[www.cadence.com](http://www.cadence.com)

**Mentor  
Graphics**<sup>®</sup>

8005 SW Boeckman Road  
Wilsonville, OR 97070-7777  
Phone: 503.685.7000  
Fax: 503.685.1204  
Sales and Product Information  
Phone: 800.547.3000  
[www.mentor.com](http://www.mentor.com)