A Blueprint for SoC Verification Success with SystemVerilog

The development schedule for today’s advanced systems-on-chips (SoCs) is largely dominated by functional verification. With verification effort increasing exponentially with design complexity, many chip developers are reconsidering traditional approaches to verification in a search for significantly higher productivity and quality. The good news is that over the past few years, advanced verification methodologies such as coverage-driven, constrained-random test generation combined with verification reuse have been proven to achieve impressive productivity gains. In addition, the IEEE 1800 SystemVerilog hardware design and verification language has emerged with broad industry support to provide an efficient, industry-standard foundation on which to build reusable and interoperable verification environments using these techniques.

Constrained-random, coverage-driven methodologies enabled by SystemVerilog provide higher productivity and quality verification for large, complex designs compared to the traditional directed-test methodology. In the directed-test approach, engineers write individual tests to verify specific design features. This process can be very time consuming, especially for complex devices with large numbers of interacting features so time consuming, in fact, that often designs must be taped out prior to reaching the verification quality target, leading to an increased chance for an expensive silicon re-spin.

With a constrained-random, coverage driven approach, engineers write high-level constraints that describe realistic input scenarios for the design. An advanced verification tool will use these constraints to automatically generate many thousands of tests – even tests that the engineer didn’t think of – to verify the design. Functional and code coverage are used to guide verification and measure progress. The result is that verification quality targets can more easily be met prior to tape-out, even for the most complex designs.

ARM and Synopsys Collaborate on Verification Methodology

Recognizing the importance of advanced verification methodologies, and the opportunity presented by creation and standardization of SystemVerilog, ARM and Synopsys have collaborated to create an open reference verification methodology to help improve SoC verification productivity and to foster the growth of an ecosystem of interoperable verification components and tools. The methodology is documented in a jointly-authored book titled Verification Methodology Manual for SystemVerilog.
The Verification Methodology Manual (VMM) for SystemVerilog provides engineers with architecture guidelines and industry best practices that enable more effective and faster functional verification of complex SoCs. It also provides verification IP developers with standard verification architectures and libraries to foster the development of interoperable verification IP. This approach is applicable to multiple levels of design abstraction, verifying transaction-level models written in SystemC™ as well as the RTL implementation of the design. The techniques address block-level, subsystem-level, and full-chip verification, encompassing both formal analysis and simulation-based methods. The VMM for SystemVerilog was peer-reviewed by over 30 industry experts and provides a blueprint for SoC verification success with SystemVerilog.

This article provides a brief overview of a few of the key elements of the methodology described in the VMM for SystemVerilog.

**VMM Layered Testbench Architecture Enables Reuse**

The VMM for SystemVerilog defines a layered testbench architecture that places various elements of the verification environment into independent levels of hierarchy, communicating over well-defined, high-level channel interfaces. The layered approach allows different parts of the environment to be developed independently, without requiring a detailed knowledge of the surrounding components. This layered approach is especially valuable to geographically-dispersed SoC development teams, easing the development of interoperable and reusable verification components.

A VMM testbench typically consists of five layers:

- **The lowest layer** is the signal layer that connects the testbench to the RTL design and is implementation-specific.

- **The command layer** contains lower-level driver and monitor components, as well as the assertions (properties) that check design intent. This layer has a procedural or object interface to the layer above and drives the physical pins via the signal layer.

- **The functional layer** contains higher-level driver and monitor components, as well as the self-check that determines whether tests pass or fail. Additional checking, for example protocol checkers, can span the command and functional layers.

- **The scenario layer** uses generators to produce streams or sequences of transactions that are applied to the functional layer. The generators have a set of weights, constraints or scenarios specified by the test layer. The randomness of constrained-random testing is introduced within this layer.

- **Finally, the test layer** is where the tests are located. The tests can define new sequences of transactions using the scenario layer, synchronize multiple transaction streams, generate sequences by interacting directly with the functional or command layers, or supply directed stimulus directly to the command layer.

The VMM layered architecture allows the environment to easily drive various models of the design, from high-level reference models written in SystemC, to register-transfer level (RTL) models, to synthesized designs running in a hardware-assisted verification environment. This concept of a “golden testbench” helps ensure design consistency as the design moves through different levels of abstraction, and enables a significant portion of the verification environment to be developed and validated prior to having RTL available.

**VMM Standard Library Speeds Development, Enables Interoperability**

To help implement the methodology, the VMM for SystemVerilog includes the complete specification for a standard library of commonly used verification functions, such as stimulus generation, transactors, channels, simulation control, messaging and coverage analysis. These testbench building blocks can be used to quickly implement an advanced testbench that easily integrates with other internally-developed VMM-compliant components.

Synopsys provides a pre-compiled version of this VMM Standard Library to its customers, and offers a no-cost source-code license for its implementation of the library to other EDA vendors to aid in the wide-scale adoption of these techniques and promote the emergence of an ecosystem of interoperable VMM-compliant tools, services, and IP.

**Extensible Verification Components for Efficient System Level Verification**

With the ever-increasing design complexity of system-level designs, support for coordination of multiple verification components is essential to provide a working platform for the construction of complex verification scenarios. The VMM for SystemVerilog describes how individual transactors can be structured into system-level transactors, referred to in the book as extensible verification components or XVCs. XVCs provide a foundation for modular, scalable and reusable system-level verification environments, with the aim of minimizing test setup overhead.

An XVC is a container for verification IP divided into two elements. The top layer is a user-extensible library of test actions with a defined action interface for integration into a verification environment. The bottom layer integrates individual transactors for implementing the actions on a physical- or transaction-level interface. These layers align with the VMM layered testbench architecture.

The VMM for SystemVerilog also defines the XVC manager for high level synchronization of XVCs. The synchronization and XVC control mechanisms can be user-defined according to the needs of the system using test scenario descriptions written as external configuration files in a plain text format. These files can be reused or modified to new requirements with relative
The ability to direct the test via simple text input files enables the user to quickly achieve effective test results without a detailed understanding of the internals of the verification environment.

**Summary**
The increasing verification challenge for complex SoCs is driving wider adoption of advanced coverage-driven, constrained random verification methodologies. This trend is further amplified by the emergence, standardization, and widespread vendor support of IEEE 1800 SystemVerilog, which enables interoperable and reusable environments to be built using these techniques.

ARM and Synopsys have collaborated to author the Verification Methodology Manual for SystemVerilog, the industry's first open reference methodology for advanced functional verification. The VMM for SystemVerilog helps development teams take maximum advantage of SystemVerilog's features while implementing a verification environment that is both flexible and reusable. The combination of a widely-supported language and a standard, comprehensive methodology will help improve verification productivity and quality, foster the emergence of interoperable verification components and tools, and give project managers the confidence they need to successfully tape out their next complex SoC. For more information on the book can be found at www.vmm-sv.org.

---

When you need proven, robust TCP/IP software for your ARM processor-based connected devices design, you can be sure that InterNiche Technologies will meet the challenge. For more than 15 years, InterNiche has been a specialist provider of embedded Internet protocol stacks and applications that offer an unrivalled combination of compact memory footprint, smooth system integration and royalty-free licensing. From the core NicheStack™ IPv4 offering through a range of IPv6, SNMP, HTTP server and security protocols, NicheStack products form a modular and flexible family of technologies that fits both your requirements and your resource constraints.

When networking devices, be sure to make this connection first: [www.interniche.com](http://www.interniche.com).

**The Specialist Provider of Software and Expertise for Device Networking**