The Cadence® Perspec™ System Verifier is a portable stimulus, system-on-chip (SoC) verification solution. The Perspec System Verifier improves SoC quality and saves time by reducing development effort for defining complex SoC-level use cases, amplifying use-case exploration of state space and timing on fast platforms, creating coverage-driven automation of system use-case generation, bridging UVM and SoC verification methodology, and shrinking the time required to reproduce, debug, and fix complex SoC-level bugs.

**Requirements for SoC Verification**

While the bottom-up approach offered by UVM-constrained random and coverage-driven verification revolutionized IP and unit-level testing, it doesn’t meet the requirements for SoC-level verification. To fully address SoC-level verification, a solution must extend from UVM and allow for vertical (IP to SoC) reuse and horizontal (verification engine portability) reuse. And most importantly it must provide a way to capture, share, and automatically amplify use cases to speed test-case creation and leverage fast verification engines.

The Perspec System Verifier addresses these three key requirements highlighted in Figure 1. By offering an abstract model-based approach, the

---

**Figure 1: SoC Verification with Portable Stimulus**
Perspec solution not only enables capture of use cases, but through abstraction makes reuse, sharing, and amplification of the use cases easy. In order to deliver real tests, a solver is required to automate the creation of concrete use cases either through randomization or if requested through coverage filling. These concrete use cases can be used to automatically generate C tests that can be run natively and at speed on any of the verification engines depicted in the figure.

Today, the verification of electronic systems (SoCs) and subsystems is generally achieved using C tests. These tests compliment the IP-level verification performed with UVM. There are several reasons for the use of C tests, including:

- A need to exercise system use cases from a software programmer’s view
- Enable portability across the Cadence Xcelium™ Parallel Simulator, Cadence Palladium® Z1 Platform, and Cadence Protium® S1 FPGA-Based Prototyping Platform (including post-silicon execution)
- Widespread knowledge of the C language and the availability of C compilers
- Ever-growing software layers that need to be verified with the hardware components (both test software and production-level software are likely to be used as part of the verification process)
- The challenge to emulate real-life scenarios in synthetic testbenches

C tests are typically created manually or by basic code generators, and lag far behind the UVM automation that has become mainstream for hardware functional verification. The effort of test creation and maintenance, test reuse that spans subsystems and systems, and leveraging these tests for future system derivatives, are not addressed properly by manually creating C tests.

Furthermore, the overall flow—defining goals, automating stimuli creation, launching tests to meet the goals, and collecting the results into a concise and intuitive dashboard—are challenges for productive system validation.

**Product Overview**

The Perspec System Verifier is a model-based, goal-directed SoC verification product developed to meet these challenges. The Perspec usage flow includes the following steps:

1. Capture the SoC actions needed to create a desired use case, if not already captured
2. Compose the desired use case
3. Use the Perspec System Verifier to solve the abstract use case to create concrete use cases or scenarios
4. The Perspec solution generates C tests for the concrete scenario mapped to specific execution platform
5. Run the tests on the targeted platform
6. Debug the test and review coverage results

**Figure 2: Abstract Use Case >> Solver >> Concrete Solution**
The Perspec solution supports the capture of abstract models of system actions and resources by using System-Level Notation (SLN) (Step 1). The abstract models are visualized using Unified Modeling Language (UML)-based graphical notation to simplify creation, modification, and use of complex use cases or scenarios (Step 2).

The Perspec solution includes a constraint solver that generates concrete scenarios from user-directed, random-selection, or coverage-driven fills of both data and control flow from the abstract scenario and can show both legal and illegal concrete scenarios based upon the rules defined in the models (Step 3).

In addition, the coverage for the scenario is calculated when generated (gen-time coverage) and can be compared with actual execution results once tests are run (runtime coverage). In Figure 3, the abstract scenario is shown on the left and a UML activity diagram, is shown on the right. The solver fills in everything needed to create an executable scenario. The callouts identify some of the choices made by the solver.

From a concrete scenario, the Perspec solution automates the generation of a C test that fulfills the scenario, including the inter-processor communication and multi-task scheduling required (Step 4). The C tests can run natively at speed on the Xcelium, Palladium Z1, Protium S1, and even post-silicon boards, and can also be automatically amplified to take full advantage of the faster platforms (Step 5). The C tests can also be generated to connect to an existing UVM environment for constrained random verification of IP and interface scenarios using the Xcelium simulator.

When the test runs, it generates a log that can be used to debug the test with Cadence’s Indago™ Debug Analyzer and coverage results can be analyzed in the context of the verification plan using the vManager™ Metric-Driven Signoff Platform (Step 6).

Connecting It Together

As shown above, the combination of abstract system actions with a constrained random solver that can randomize both control and data offers significant productivity improvement over manual creation of tests, but the real value is the ability to capture complex SoC-level use cases that would otherwise go unverified and to find bugs in the implementation that would go undetected until the problem occurs in actual use.

To illustrate, let’s consider how to verify a use case where multiple cache coherent processors are operating on different tasks and where you need to verify that the cache coherency is maintained even when powering some of the processors on and off. Most customers today avoid tackling the development of a directed test for this type of complex use case because of the level of expertise required in both coherency and power management and the complexity of the software required to create this use case. Instead, they rely on their production software to validate these types of complex use cases as best they can. With the Perspec solution, it is easy to take advantage of use cases developed by domain experts and to create new, more complex SoC use cases by mixing use cases.

Figure 4 highlights how the Perspec solution not only makes it possible to create the complex use case of mixing power shutdown and coherency, but as depicted allows other users to take advantage of use cases created by domain experts without needing to become a domain expert. In addition, the Perspec solution automates the generation of the complex C tests for any target platform.
Features

Use-Case Composer GUI
- Defines use cases in goal-oriented terms using a UML-like GUI editor
- Allows reviewing and sharing use cases and system flows between teams
- For advanced use cases:
  - Enables composition of sub-scenarios to create advanced use cases and flows
  - Scalable solution supports thousands of actions
  - Allows definition of reusable flows for test construction
  - Supports operators for random scenario selection, generation-time repetition/filling and run-time repetition (for long tests).

Libraries
- System Modeling Library with building blocks for modeling common system components
- SoC Library for modeling and creating scenarios to verify sophisticated multi-core CPU subsystems including cache coherency, DVM, and low-power modes

Advanced constraint solver
- Randomizes both system control flow and data
- Automatic memory management and planning of legal resource distribution
- Randomizes hard-to-achieve scenarios and also spans multiple dimensions around them with fill capability
- Automatic amplification of use-case state space and timing exploration for faster verification engines

Multi-core microkernel
- Allows runtime synchronization of multiple heterogeneous cores and parallel testbench activities
- Emulates multi-threading on single thread cores
- Dynamic runtime management of resources to enable efficient and concise tests
- Exports messages from embedded cores

Coverage metric goals to ensure completeness
- Supports both generation-time for regression planning and runtime coverage
- Collects functional coverage from all verification engines including simulation, acceleration, emulation, FPGAs, and post-silicon
- Includes explicit user-defined coverage goals and implicit exhaustive coverage definitions
- Delivers interval utility to enable coverage of hardware and software events, latency, and event shmooing
- Plan-driven approach enables the user to select the desired verification plan goal and the tool generates an optimized set of tests to fill it in

Checking
- Enables both runtime and post simulation checking
- Allows creation of assertions involving both hardware and software events around latency, expected value comparison, and end-to-end tests
Model debug capabilities

- Interactive GUI debugging of model contradictions, UML graphs, solver operation, and C test generation
- Supports automatic synchronization of C test message log, UML-based graph nodes, and waveform transactions
- Supports adding messages, abstract transactions, and waveforms based on events and intervals

Verification Engines

Out-of-the-box test generation and execution support for using Cadence verification engines:
- Xcelium Parallel Logic Simulator
- Palladium Z1 Enterprise Emulation Platform
- Protium S1 FPGA-Based Prototyping Platform

VIP

The Cadence Verification IP (VIP) Catalog can be used for UVM simulation of IP interface protocols, as well as accelerated VIP for SoC verification and system emulation. Many VIP for UVM are available with C cores for fast execution and portability, and come with the TripleCheck™ tool, which includes an extensive library of test sequences, a coverage model for tracking and measuring verification progress, and a verification plan that mirrors the protocol specification.

The Perspec solution can generate tests that drive UVM simulation tests in the VIP for coverage driven verification. The Perspec solution can also use the same portable stimulus to generate tests for acceleration with accelerated Verification IP.

Related Products

Indago Debug Platform

The Perspec System Verifier’s automated C tests are generated to include built-in logging information. This information can be viewed using the Cadence Indago Debug Platform, which provides post-process debug from any platform used to execute the tests. In addition, the post-process debug uses the same UML activity diagram to provide the same use-case-focused debug perspective that the Perspec System Verifier uses to speed creation of use cases.

Perspec supports the ability to include embedded software in the design under test, and generate tests that drive the API of that layer of software. The Indago Embedded Software Debugger provides additional, complementary debug support such as software variable and register tracing.

vManager Metric-Driven Signoff Platform

Cadence’s vManager Metric-Driven Signoff Platform enables a plan-driven approach for SoC verification. The user can select a section in the hierarchical verification plan and call the Perspec System Verifier to create an optimized subset of tests that will address the sub-section goals. The Perspec System Verifier is also integrated to provide uniform management and analysis of the coverage resulting from the C tests executed. The vManager Metric-Driven Signoff Platform can display both the calculated coverage created at the time of test generation and the end coverage based upon results from the test execution. The coverage can also be merged with HDL, HVL, and assertion-based coverage. As with debug, the logging of the coverage points in the Perspec System Verifier is embedded in the C tests and can be post-processed to show the results from the test.

Cadence Services and Support

- Cadence application engineers can answer your technical questions by telephone, email, or Internet—they can also provide technical assistance and custom training.
- Cadence-certified instructors teach more than 70 courses and bring their real-world experience into the classroom.
- More than 25 Internet Learning Series (iLS) online courses allow you the flexibility of training at your own computer via the Internet.
- Cadence Online Support gives you 24x7 online access to a knowledgebase of the latest solutions, technical documentation, software downloads, and more.
- For more information, please visit www.cadence.com/support for support and www.cadence.com/training for training.