cadence®

Perspec System Verifier

Use-case-driven SoC verification

Cadence® Perspec™ System Verifier is a software-driven system-on-chip (SoC) verification solution. The Perspec System Verifier improves SoC quality and saves time by reducing development effort for complex SoC-level use cases, creating coverage-driven automation of system use-case generation, 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 allow not only for vertical (IP to SoC) reuse and horizontal (cross-platform) reuse, but most importantly it must provide a way to capture and share use cases and deliver a top-down verification approach. As shown in Figure 1, the Perspec System Verifier addresses these three key requirements for SoC verification and creation of portable stimulus. By offering an abstract model-based approach, the Perspec System Verifier not only enables capture of use cases, but through abstraction makes reuse and sharing 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 platforms depicted in the figure.

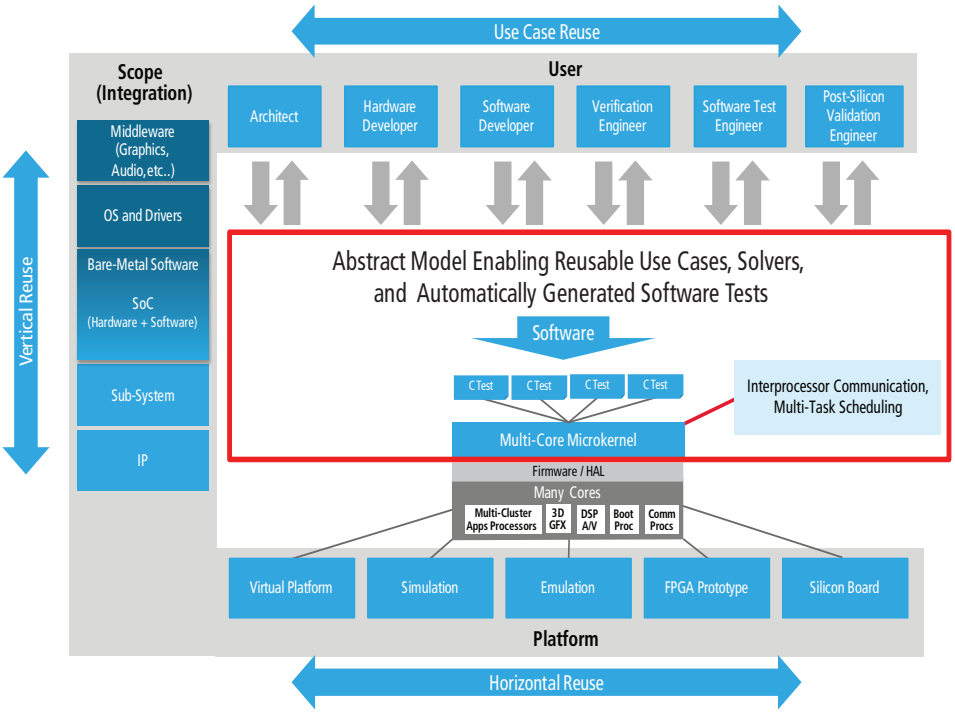

Figure 1: Perspec System Verifier Technology

Product Overview

Today, the verification of electronic systems (SoCs) and subsystems is generally achieved using C tests. There are several reasons for this, including:

- A need to exercise system use cases from a programmer's view
- Portability across platforms (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 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 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.

The Perspec System Verifier is a model-based, goal-directed SoC verification product developed to meet these challenges. Developed in conjunction with key customers, the Perspec System Verifier was released to production in 2013, and since has been used in production projects with several customers. 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
3a. Use the Perspec System Verifier to solve the abstract use case to create concrete use case or scenario
3b. The Perspec System Verifier generates C tests for the concrete scenario mapped to specific execution platform
4. Run the tests on the targeted platform
5. Debug the test and review coverage results

The Perspec System Verifier supports the capture of abstract models of system actions and resources using the Perspec System Verifier's 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 System Verifier 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

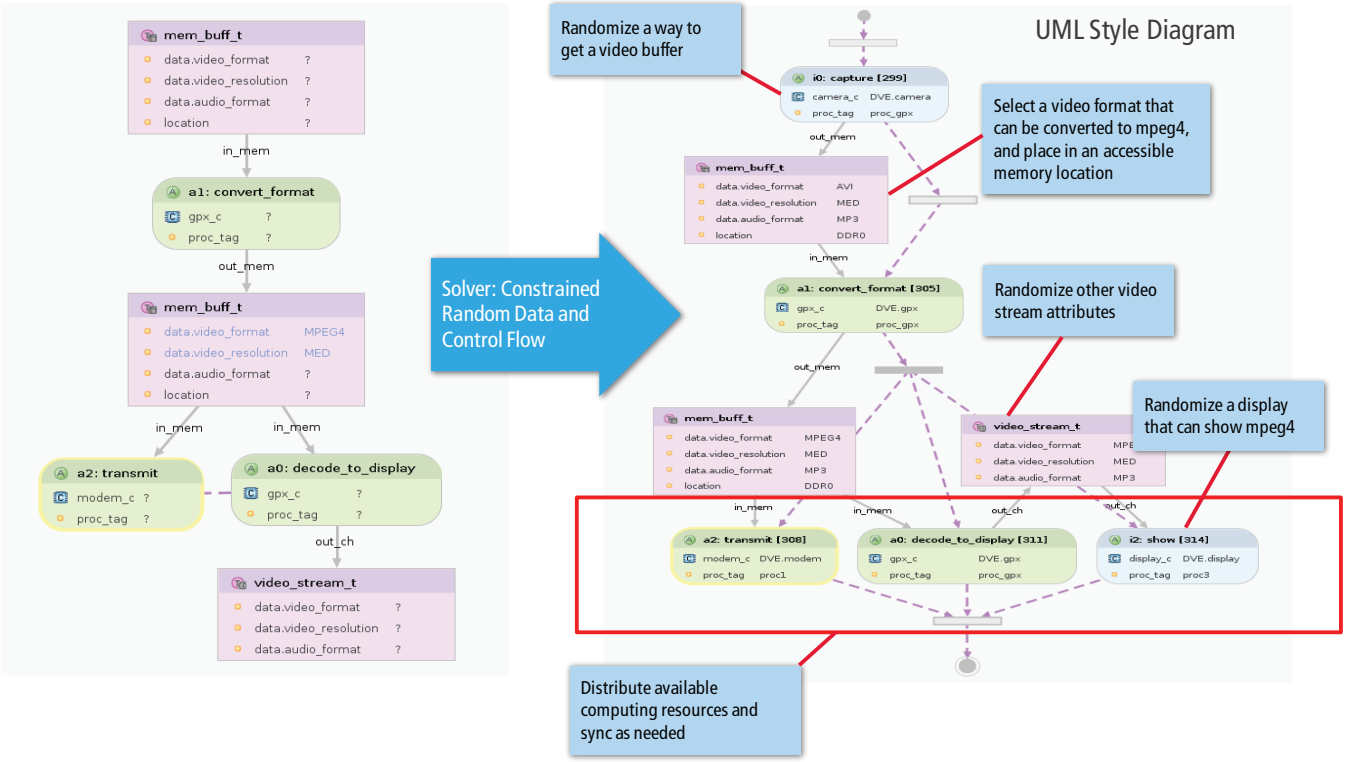

Figure 2: Abstract Use Case >> Solver >> Concrete Solution

rules defined in the models (Step 3a). In addition, the coverage for the scenario is calculated at generation (gen-time coverage) and can be compared with actual execution results once tests are run (runtime coverage). In Figure 2, the abstract scenario is shown on the left and a concrete solution, 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 System Verifier automates the generation of a C test that fulfills the scenario, including the interprocessor communication and multi-task scheduling required (Step 3b). The C tests can run natively at speed on simulation, emulation, FPGA prototype, and even post-silicon boards and can also be expanded to take full advantage of faster platforms (Step 4). When the test runs, it generates a log that can be used to debug the test with Cadence's Incisive® Debug Analyzer and coverage results can be analyzed in the context of the verification plan using the Incisive vManager™ solution (Step 5).

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 System Verifier, 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 3 highlights how the Perspec System Verifier 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 System Verifier automates the generation of the complex C tests for any target platform.

Features

Capture 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:

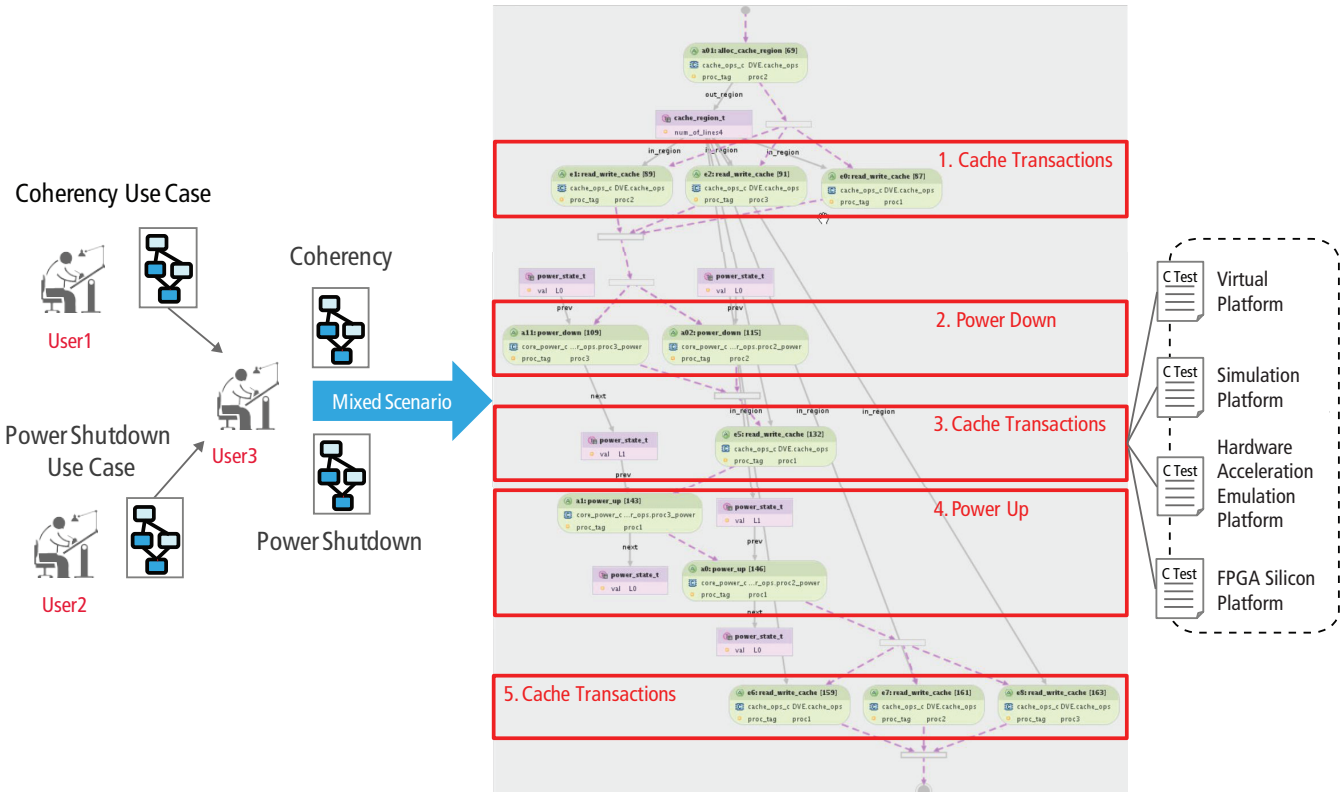

Figure 3: Creating SoC-Level Mixed Scenarios

  - 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).

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
- Maximizes technology for accelerated coverage closure

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 on all platforms 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

Abstract debug capabilities

- 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

Related Products

Incisive Debug Analyzer

The Perspec System Verifier's automated C tests are generated to include built-in logging information. This information can be viewed using the Cadence Incisive Debug Analyzer, 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.

Incisive vManager Solution

Cadence's Incisive vManager solution 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 Incisive vManager solution 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®

Cadence Design Systems enables global electronic design innovation and plays an essential role in the creation of today's electronics. Customers use Cadence software, hardware, IP, and expertise to design and verify today's mobile, cloud, and connectivity applications. **www.cadence.com**