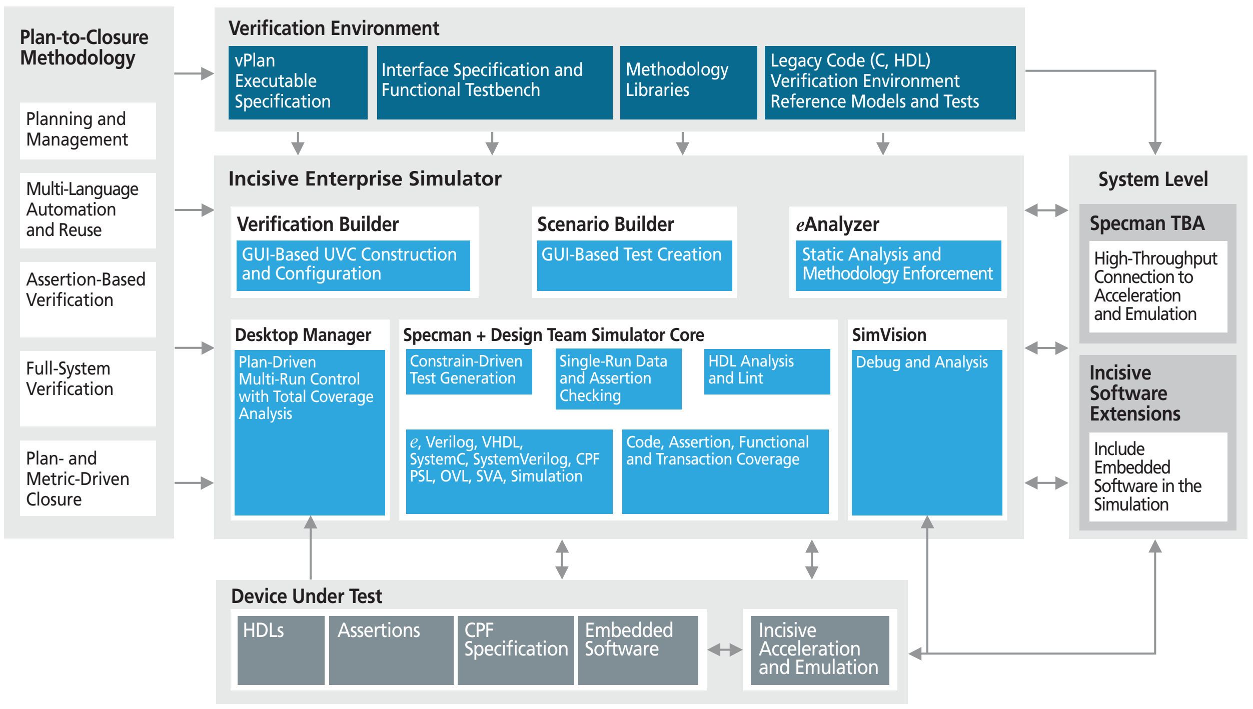
Incisive Enterprise Simulator

Multi-language simulation for low-power, metric-driven, mixed-signal verification

Created for verification teams developing complex system-level environments, Cadence® Incisive® Enterprise Simulator simplifies and accelerates your workflow. Its blend of leading-edge process automation technology, high-performance engines, power analysis, and advanced debug capabilities—integrated with the Incisive platform—helps you verify the most complex chips and systems. With support for all IEEE-standard languages, Si2's Common Power Format, and the comprehensive Plan-to-Closure Methodology, Incisive Enterprise Simulator improves productivity, project predictability, and product quality, helping you take the risk out of verification.



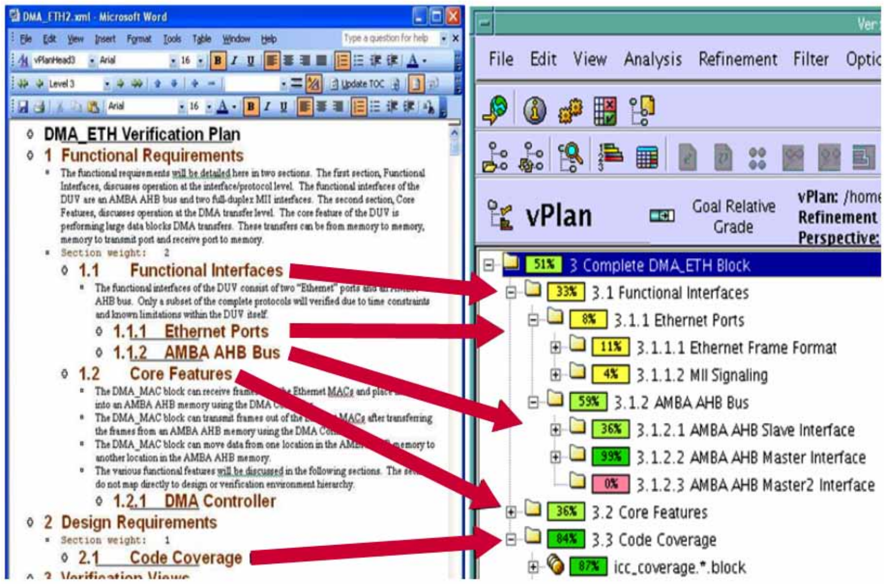
Plan-to-Closure Methodology
Planning and Management
Multi-Language Automation and Reuse
Assertion-Based Verification
Full-System Verification
Plan- and Metric-Driven Closure
Verification Environment
vPlan Executable Specification
Interface Specification and Functional Testbench
Methodology Libraries
Legacy Code (C, HDL) Verification Environment Reference Models and Tests
Incisive Enterprise Simulator
Verification Builder
GUI-Based UVC Construction and Configuration
Scenario Builder
GUI-Based Test Creation
eAnalyzer
Static Analysis and Methodology Enforcement
Desktop Manager
Plan-Driven Multi-Run Control with Total Coverage Analysis
Specman + Design Team Simulator Core
Constrain-Driven Test Generation
Single-Run Data and Assertion Checking
HDL Analysis and Lint
e, Verilog, VHDL, SystemC, SystemVerilog, CPF PSL, OVL, SVA, Simulation
Code, Assertion, Functional and Transaction Coverage
SimVision
Debug and Analysis
System Level
Specman TBA
High-Throughput Connection to Acceleration and Emulation
Incisive Software Extensions
Include Embedded Software in the Simulation
Device Under Test
HDLs
Assertions
CPF Specification
Embedded Software
Incisive Acceleration and Emulation

Figure 1: Incisive Enterprise Simulator combines Incisive Enterprise Specman® Simulator, SimVision, Desktop Manager, Verification Builder, Scenario Builder, eAnalyzer, and the Plan-to-Closure Methodology in a single optimized package. Engineers can "hot-swap" the state of the software-based simulation in and out of the Incisive Xtreme® series of accelerators and emulators for additional performance. System-level capabilities include a connection to Incisive Software Extensions and support for testbench acceleration with Palladium® XP.

Incisive Enterprise Simulator

Incisive Enterprise Simulator is the only product on the market that supports all IEEE-standard languages and design abstractions, from the gate level all the way up to system modeling and verification. Additionally it supports the verification plan (vPlan) executable specification, Si2's Common Power Format (CPF) specification, and all Plan-to-Closure Methodology flows.

Verification engineers can extend the functionality of Enterprise Simulator with Incisive Software Extensions that provide a high-throughput channel between the testbench and the device under test (DUT), and enables automated Plan-to-Closure verification of embedded software exactly as if it were another part of the DUT. With other elements from the Incisive platform, including verification IP, hardware acceleration and emulation, analog/mixed-signal/RF verification, and formal assertion verification, Enterprise Simulator supports any testbench, HDL, CPF file, software, and assertion IP created with the Incisive platform's other simulators: the HDL Simulator and the Design Team Simulator; or vPlans created by Enterprise Manager.


DMA_ETH2.xml - Microsoft Word
File Edit View Insert Format Tools Table Window Help
Type a question for help
vPlanHead3
Arial
16
Level 3
Update TOC
File Edit View Analysis Refinement Filter Optic
vPlan
Goal Relative Grade
vPlan: /home
Refinement
Perspective:

DMA_ETH Verification Plan

1 Functional Requirements

- The functional requirements will be detailed here in two sections. The first section, Functional Interfaces, discusses operation at the interface/protocol level. The functional interfaces of the DUV are an AMBA AHB bus and two full-duplex MII interfaces. The second section, Core Features, discusses operation at the DMA transfer level. The core feature of the DUV is performing large data blocks DMA transfers. These transfers can be from memory to memory, memory to transmit port and receive port to memory.
- Section weight: 2

1.1 Functional Interfaces

- The functional interfaces of the DUV consist of two "Ethernet" ports and an AMBA AHB bus. Only a subset of the complete protocols will verified due to time constraints and known limitations within the DUV itself.

1.1.1 Ethernet Ports

1.1.2 AMBA AHB Bus

1.2 Core Features

- The DMA_MAC block can receive frame[illegible] Ethernet MACs and place [illegible] into an AMBA AHB memory using the DMA C[illegible]
- The DMA_MAC block can transmit frames out of the [illegible] MACs after transferring the frames from an AMBA AHB memory using the DMA Co[illegible]
- The DMA_MAC block can move data from one location in the AM[illegible] memory to another location in the AMBA AHB memory.
- The various functional features will be discussed in the following sections. The se[illegible] do not map directly to design or verification environment hierarchy.

1.2.1 DMA Controller

2 Design Requirements

- Section weight: 1

2.1 Code Coverage

3 Verification Views

51% 3 Complete DMA_ETH Block
33% 3.1 Functional Interfaces
8% 3.1.1 Ethernet Ports
11% 3.1.1.1 Ethernet Frame Format
4% 3.1.1.2 MII Signaling
59% 3.1.2 AMBA AHB Bus
36% 3.1.2.1 AMBA AHB Slave Interface
99% 3.1.2.2 AMBA AHB Master Interface
0% 3.1.2.3 AMBA AHB Master2 Interface
36% 3.2 Core Features
84% 3.3 Code Coverage
87% icc_coverage.*.block

Figure 2: The Desktop Manager component of Incisive Enterprise Simulator can create an executable vPlan from a written specification. This plan is then used to drive all simulations, with the coverage results from multiple simulaton runs automatically backannotated into the vPlan for a "total coverage" view.

Benefits

Predictability

- Drives the complete verification process from plan to closure with the vPlan executable specification
- Ensures higher quality system verification by applying coverage-driven verification and debugging of complex HW/SW interactions and embedded software

High performance

- Delivers the highest possible performance for mixed-language designs, across multiple levels of abstraction
- Enables "hot-swap" of the software-based simulation in and out of the Xtreme accelerators and emulators for additional performance

Productivity

- Reduces "time to first test" for verification novices and experts alike
- Speeds the construction, configuration, understanding, and usage of verification IP using Incisive Verification Builder and Incisive Scenario Builder
- Accelerates the set up, debug, and integration of verification environment components using *e*Analyzer's static analysis and methodology compliance checking
- Automates generation, assertion checking, and functional and code coverage collection to provide a "total coverage" view
- Reduces the time to discover and correct logic and transaction modeling errors via an integrated graphical debug environment

Comprehensive standards support

- Supports all IEEE-standard languages and interfaces, enabling full legacy and third-party IP usage
- Supports the Si2 CPF, an open specification language that captures all power-specific design, constraint, and functionality requirements in a single file

Features

Executable verification plan

Enterprise Simulator incorporates Desktop Manager, a subset of Incisive Enterprise Manager, to drive the verification process right from the planning stage. When combined with the guidelines in the planning and management component of the Plan-to-Closure Methodology, verification teams can automatically capture the verification objectives from the written verification plan, create a vPlan executable specification, and automatically compare progress against the vPlan.

Constraint-driven stimulus generation

Using either the IEEE 1647 *e* language or IEEE 1800 SystemVerilog (or both together), Enterprise Simulator provides constraint-driven stimulus generation, which automates the process of generating functional verification tests. By specifying constraints, verification engineers can target the generator quickly and easily to create any test in the functional test plan. They can even generate tests on-the-fly based on the current design state, making it possible to detect hard-to-reach corner cases.

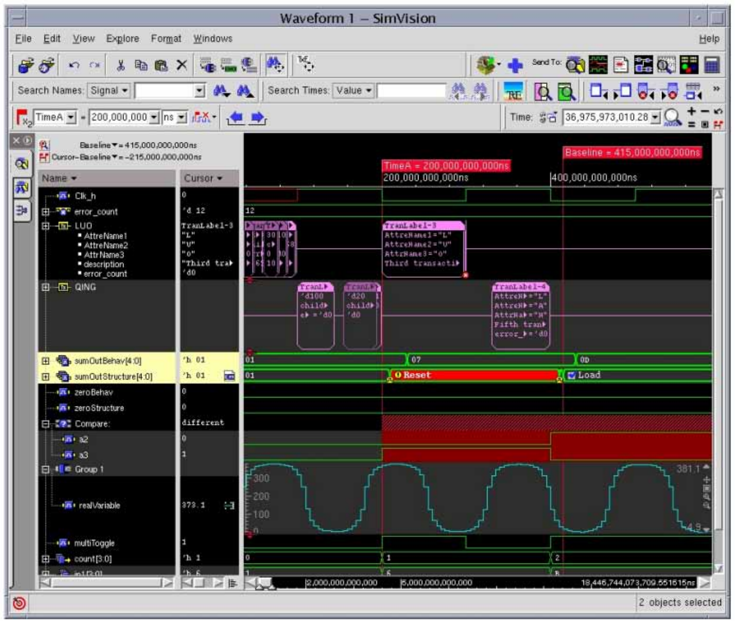


Figure 3: The SimVision integrated debug environment supports signal-level and transaction-based flows across all IEEE-standard design, testbench, and assertion languages, in addition to concurrent visualization of hardware, software, and analog domains.

Rapid creation of libraries of reusable tests

Enterprise Simulator fully supports the testbench reuse component of the Plan-to-Closure Methodology, which describes how to create reusable verification components in any IEEE-standard language, and provides guidelines for setting up multi-language interfaces to existing IP for maximum operational flexibility.

The process is based on the time-tested *e* Reuse Methodology (*e*RM) and System Verification Methodology (SVM); and with the class-based language support inside Enterprise Simulator, it extends to encompass class-based reuse features found in the IEEE 1800 SystemVerilog-compliant, open-source Open Verification Methodology (OVM) library. Examples include recommendations for structuring classes, virtual interfaces, and packages for unrestricted portability.

With the Verification Builder component, users can quickly create and configure reusable verification components in either *e* or SystemVerilog, as per Plan-to-Closure Methodology specifications. Verification Builder also supports plug-ins that help users automatically connect the DUT to the testbench. There is a similar plug-in for Incisive Software Extensions that supports the rapid binding of the testbench to the processor model(s).

Additionally, the Scenario Builder component helps engineers create test cases in a fraction of the time it would take to write them in a hardware verification language, eliminating the need to learn a verification language and object-oriented programming. Users simply drag-and-drop randomized verification elements and adjust their constraints to build a scenario. Scenario Builder makes it just as easy to build complex system-level test cases: users define stimuli for each interface that can be synchronized on existing or user-defined events.

When combined with the “sequence” feature of the Plan-to-Closure Methodology, stimuli that exercise corner-case scenarios at the block level can be reused at the system level to verify how the entire chip behaves in that corner case. Furthermore, users can graphically create these sequences as easily as test cases, which they can use in other tests hierarchically or in other sequences by dragging and dropping in a new scenario. All scenarios are written out into a reusable format, supporting the creation of portable libraries of protocol checkers and other structured behaviors for reuse on future projects, or system-level verification of the same project.

Data and assertion checking

Powerful temporal constructs allow engineers to capture complex protocols for assertion checking. On-the-fly data checking and generation provides context-specific expected values. Enterprise Simulator also supports any combination of gray-, black-, or white-box checking.

Total coverage analysis

Enterprise Simulator supports “total coverage analysis” including all coverage metrics commercially available—functional, HDL code, and assertion coverage—which are viewed together in one GUI. This ensures that all functionality is fully tested, allowing the team to achieve first-pass silicon success.

Leveraging functional coverage in particular, an executable functional test plan (the vPlan) measures the progress of verification, and functional analysis automatically identifies holes in the test coverage. Since functional coverage is a meaningful and direct measure of the completeness of verification, this analysis increases the predictability of the verification schedule.

Failure analysis

Enterprise Simulator further simplifies the overall debugging effort and shortens simulation-failure debug time by separating design failures from simulation failures, sorting and grouping these failures for easy selection and action. It then identifies the least costly simulation to exhibit the failure and the optimal case for repeated debugging.

Heterogeneous single-kernel architecture

A heterogeneous single-kernel architecture enables unified simulation through behavioral, transaction, register-transfer, and gate levels of abstraction. It uses a unique interleaved native-compiled architecture that supports all open and IEEE-standard languages including Verilog®, SystemVerilog, VHDL, *e*, SystemC®, the SystemC Verification (SCV) Library, CPF, PSL, SystemVerilog Assertions (SVA), OVL, and the OVM class library. Design and testbench models can be interleaved in any language and any level of abstraction without the performance and integration overhead caused by co-simulation.

Enterprise Simulator also produces efficient machine code for high-speed execution. Linked-list scheduling of the resulting data structures pre-processes signal actions while maximizing the effectiveness of modern caching algorithms available in leading compute platforms.

"Hot-swap" with Incisive Xtreme series

To significantly boost verification throughput, the simulation state in Enterprise Simulator can be "hot-swapped" in and out of the Incisive Xtreme series of accelerators and emulators in real time. Specifically, during simulation, engineers can swap the state of the Xtreme engine into Enterprise Simulator to interactively debug the design and continue with software simulation. When the circuit is fully debugged and the problem isolated, engineers can swap the simulation state value back into the Xtreme engine for maximum performance.

Unified SimVision debug environment

SimVision provides a unified simulation and debug environment that allows Enterprise Simulator to manage multiple simulation runs easily and to analyze both design and testbench behavior at any point in the verification process—regardless of the composition.

Throughout the design and verification flow, SimVision provides hardware analysis checks, source browsing, transaction and waveform viewing, and complete code/transaction/assertion coverage analysis. Application programming interfaces based on industry standards are available at all levels to enable user-defined checks and analysis. Bottom line: engineers only have to learn one debug environment.

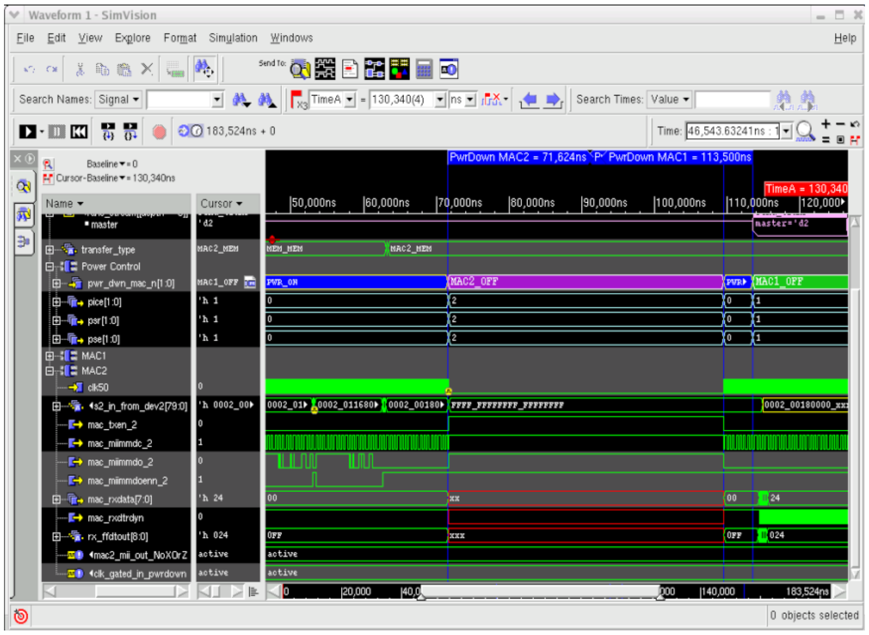


Figure 4: Engineers can determine the DUT's power characteristics very early in the project life cycle. In this example, the user is verifying a full cycle of power shut off (note the Xs in the center) and correct state restoration.

Native low-power analysis

As the industry pushes toward smaller process geometries, the automation of low-power design and verification techniques is critical. Enterprise Simulator address this need with native support for the CPF, an open specification language that captures all power-specific design, constraint, and functionality requirements, such as multi-supply voltage and power shut-off, in a single file. This enables automation of power-reduction techniques inside the Enterprise Simulator core, integrated display and debug of power behaviors in SimVision, and seamless connections to downstream implementation flows.

The result: design engineers can determine the DUT's power characteristics significantly earlier in the project life cycle, yielding greater power savings than from exclusive reliance on implementation methodologies.

Native PSL, SVA, and OVL support

The Incisive single-kernel architecture provides native PSL, SVA, and OVL support. Also included is the Incisive Assertion Library, which allows users to insert common assertions into their designs quickly. Assertions enable design teams to instrument their interfaces and logic with monitors that automatically detect specified error conditions or critical coverage situations.

The resulting increase in observability means verification engineers detect more bugs, detect bugs closer to the source, guarantee that all interface activity is legal, and identify coverage holes. Native-kernel support also means that assertions have minimal impact on runtime and capacity. This ensures the appropriate assertions are always active, so that no bugs go undetected.

HDL and testbench analysis

HDL analysis catches design bugs and coding surprises early in the design cycle. It performs more than 500 checks to flag syntactic, semantic, and functional errors. A flow that includes HDL analysis before simulation will check the code for race conditions, problems with clock domain synchronization, semantic ambiguity issues, and synthesizability traps.

Similarly, the *e*Analyzer component of Enterprise Simulator focuses on static analysis of an *e* language testbench, allowing engineers to identify reuse and porting issues, and generally accelerating the set up, debug, and integration of the testbench environment.

HDL and testbench analysis also come with rules that check compliance with the Plan-to-Closure Methodology reuse guidelines, and can be expanded to include corporate style guidelines. These powerful rule-definition GUIs and graphical analysis tools help engineers write working code correctly the first time.

Plan-to-Closure Methodology

The Plan-to-Closure Methodology provides a system of best-known principles, guidelines, and procedures that increase project productivity and predictability, and ensure overall system-level quality. It includes documented best practices, golden examples that serve as templates to help designers learn and apply the methodology, and libraries (code building blocks and utilities) that automate the process and eliminate many redundant verification-coding tasks.

In addition, unlike any other commercially available methodology, the Plan-to-Closure "Knowledge System" (a web-based portal technology) allows you to easily research topics of interest, and then customize the methodology to your specific needs. The Plan-to-Closure Methodology spans verification planning and management, assertion-based verification, testbench automation and reuse, and system-level verification.

System-Level Verification

Traditionally, the "ESL" acronym has been used to describe the need for scalability and increased abstraction to accommodate growing HDL designs. Testbench acceleration and Incisive Software Extensions build on this traditional flow to enable systems engineers, logic designers, software engineers, verification engineers, and system validation teams to do "in-system" verification at the block, chip, and full system levels. This is all made possible by using the system verification capabilities of Incisive Enterprise Specman and Incisive Software Extensions to leverage the metric-driven Plan-to-Closure Methodology to span design and verification—from an initial system specification and system verification plan to full system-level integration and closure. The result: rapid discovery of deeply buried, cross-domain bugs that are impossible to find otherwise.

Incisive Software Extensions

Incisive Software Extensions give your testbench access to embedded software exactly as if it were another part of the HDL DUT. Using extensions to the familiar Incisive SimVision debug tool and the vPlan, engineers can simultaneously control and verify software methods, procedures, variables, registers, and other elements with a traditional hardware-centric DUT using the same time-tested coverage driven verification process described in the Plan-to-Closure Methodology. Embedded software tracing provides non intrusive probing, break point setting, and source viewing of embedded software with the hardware debug environment. Furthermore, Incisive Software Extensions rise above hardware/software co-verification limitations by supporting processor models in any form: workstation-based host-code execution, Instruction Set Simulators (ISS), full RTL CPU models, hardware acceleration and emulation, and even prototype silicon.

Transaction-based acceleration (TBA)

TBA utilizes message-level communication between the Enterprise Simulator testbench and the acceleration/emulation hardware. By communicating at the message level instead of the signal level, TBA significantly reduces the amount of testbench-DUT communication, greatly increasing overall throughput over simulation alone (typically from 100x to 1,000x when used in combination with the Incisive Palladium or Xtreme series of hardware accelerators and emulators). This level of performance enables full system-level validation that includes embedded and application-layer software.

When TBA and Incisive Software Extensions are combined with the vPlan-driven Plan-to-Closure flow, the result is a predictable, metric-driven process for both hardware and software development based on an executable plan that enforces adherence to the system specification. In short, this real-time, automatic annotation of metrics against the specification gives engineers the information they need to confidently argue for additional time to explore more high-risk corner cases, or to ask for a new pen to autograph the signoff form.

Specifications

Simulation

- Single-kernel simulation engine
  - Verilog (IEEE 1364-1995 and IEEE 1364-2001 extensions)
  - SystemVerilog (IEEE 1800)
  - *e* (IEEE 1647)
  - VHDL (IEEE 1076-1987, IEEE 1076-1993, IEEE 1076.4-2000 (VITAL 2000))
  - SystemC (IEEE 1666, OSCI® SystemC v2.2)
  - PSL (IEEE 1850)
  - SVA (IEEE 1800)
  - Si2 Common Power Format (CPF)
- Compile
  - Native compilation technology goes directly to host processor machine code, which maximizes performance
  - Intelligent incremental compile reduces compile times
- Capacity
  - Typical 10M gate equivalents in 32-bit OS (4GB addressable)
  - Typical 100M gate equivalents in 64-bit OS
- Server farm
  - Platform computing LSF
  - Sun Microsystems gridware

Assertion support

- Supports SVA and PSL, the standard assertion languages from IEEE
  - Handles Verilog, VHDL, and SystemC versions of PSL
- Supports the Open Verification Library (OVL) standard
- Supports *e* testbench assertions
- Includes the Incisive Assertion Library
- Common compile and elaboration mechanism with the Incisive platform
- Common user interface with the Incisive platform
- Dynamic assertion evaluation
  - Natively compiled with HDL for highest performance

- Assertions can be embedded in the HDL or in separate file(s)
- Recorded as transactions for direct display in waveform window
- PSL and SVA assertions treated as first-class simulation objects for easy debugging

SimVision results analysis

- Debug and GUI
  - Waveform window
  - Register window
  - Unified transaction/signal viewing
  - Schematic tracer
  - Expression calculator
  - Signal flow browser
  - Source viewer
  - Error browser
  - Tcl/Tk scripting for customizable displays
  - Log signal and transaction data to SST database
- Performance analysis software outlines areas of code where most simulation time is spent
- Code coverage
  - Supports Verilog, SystemVerilog, VHDL, and mixed-language designs
  - Automatic finite state machine extraction
  - Coverage attributes supported include blocks, paths, expressions, variables, gates, FSM (states, sequences), and toggle
  - Coverage reuse
  - Rank order coverage contributions
  - Bit-wise expression scoring
- Functional coverage analysis
  - Supports Verilog, SystemVerilog, VHDL, *e*, SystemC, SCV, PSL, SVA, and OVL
  - Logs data to SST2 database
  - Tcl/Tk scripting for custom analysis

HDL analysis

- 500+ checks to lint and analyze code for:
  - Synthesizability
  - Race conditions
  - Code reusability
  - Clock domain synchronization
  - FSM coding
  - Acceleration policy checks
- Gate-level netlist analysis for any DFT errors introduced during synthesis
- Verilog, SystemVerilog, VHDL, and mixed-language support
- Powerful customization capability using VPI/VHPI
- Graphical interface to sort, filter, and analyze messages with source code

e testsbench analysis

- 200+ checks to lint and analyze code for:
  - Code reusability per *e* Reuse Methodology (*e*RM) compliance rules vPerformance analysis vRace conditions
  - Pre-defined coding style rules
  - Generation constraints
- Graphical interface to sort, filter, and analyze messages with source code

Verification builder

- GUI support for configuration of Plan-to-Closure Universal Verification Components (UVCs)
- Outputs *e* (IEEE 1647) or (IEEE 1800) Plan-to-Closure UVCs

Cadence IP support

- Design IP
  - Cadence SoC Functional Verification Kit
- Verification IP
  - Supports all simulation-based Universal Verification Components (UVCs), transaction-based VIP, assertion-based VIP, and SpeedBridge® rate adapters used in emulation
  - Supports the full portfolio of Cadence UVCs: PCI Express, AHB, AXI, AMBA™, USB, and Ethernet, PCI, SATA, and OCP
  - UVCs are designed to use all elements of Enterprise Simulator, featuring a comprehensive Compliance Management System that leverages vPlans to exhaustively verify protocol compliance

Third-party support

- ASIC libraries
  - More than 30 ASIC vendors have certified their libraries for the Incisive platform
  - More than 150 unique libraries
- Models
  - Third-party model support through the Cadence Verification IP Partner program
- Software
  - Third-party software support through the Connections® program with more than 30 verification company partners

Interfaces

- PLI (IEEE 1364)
- DPI (IEEE 1800)
- VPI (PLI 2.0, IEEE 1364)
- OMI (IEEE 1499)
- VHPI
- Compiled SDF

Platforms

- Sun Solaris
- HP-UX
- Linux

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

cādence®

Cadence is transforming the global electronics industry through a vision called EDA360. With an application-driven approach to design, our software, hardware, IP, and services help customers realize silicon, SoCs, and complete systems efficiently and profitably. **www.cadence.com**

21508 11/11 MK/DM/PDF