Verdi³
Automated Debug System

Overview
The Verdi™ Automated Debug System is an advanced open platform for debugging digital designs with powerful technology that helps you comprehend complex and unfamiliar design behavior, automate difficult and tedious debug processes and unify diverse and complicated design environments.

Cut Debug Time in Half
The Verdi³ system lets you focus on tasks that add more value to your designs by cutting your debug time, typically by over 50%. These time savings are made possible by unique technology that:
- Automates behavior tracing using unique behavior analysis technology
- Extracts, isolates, and displays pertinent logic in flexible and powerful design views
- Reveals the operation of and interaction between the design, assertions, and testbench

Figure 1: The unique behavior analysis technology of the Verdi³ system automates many time-consuming aspects of debug

Complete Debug System
The Verdi³ automated debug system incorporates all of the technology and capabilities you would expect in a debug system. In addition, the Verdi³ system combines advanced debugging features with support for a broad range of languages and methodologies.
Core Features
The Verdi® system provides the following fundamental debug features:
- Full-featured waveform viewer enables you to display and analyze activity over time
- Powerful waveform comparison engine allows you to isolate differences between fast signal database (FSDB) files
- Source code browser enables you to easily traverse between source code and hierarchy
- Flexible schematics and block diagrams give you the ability to display logic and connectivity using familiar symbols
- Intuitive bubble diagrams help you to reveal the operation of finite state machines

Advanced Features
The Verdi® system also includes the following advanced debug features:
- Automatic tracing of signal activity enables you to quickly trace activity across many clock cycles with powerful behavior analysis technology
- Temporal flow views provide a combined display of time and structure to help you rapidly understand cause-and-effect relationships
- Transaction-based debug with flexible transaction and message support for debugging and analyzing designs at higher levels of abstraction
- Assertion-based debug with built-in support for assertions facilitates quick traversal from assertion failure to related design activity
- SystemVerilog Testbench debug with:
  - Full source code support for SystemVerilog Testbench (SVTB) and libraries, including Universal Verification Methodology (UVM), to ensure reusability and interoperability of testbench code
  - Specialized views that help you understand testbench code, including declaration-based hierarchy browsing and navigation, class inheritance and relationship comprehension, and tracing

Languages and Methodologies
The Verdi® system supports the following languages and methodologies:
- Design components described in Verilog, VHDL, and SystemVerilog
- Automated testbench environments using SVTB, as well as UVM, OVM and VMM
- Assertions using SystemVerilog Assertions (SVA)

Optimized Open Architecture and Unified Methodology
The Verdi® automated debug system is designed so that you can take full advantage of your verification and debug methodology. The Verdi® system is built on the open Design Knowledge Architecture, which consists of compilers that extract relevant information into databases that are optimized for efficient debug. The Verdi® system also unifies your debug process by providing a single solution that operates seamlessly and consistently across multiple domains: verification tools, design/verification languages, and abstractions. This consistency reduces your learning curve and saves time as you move to new projects using different tools and languages, and allows you to further leverage your investment in the Verdi® system even as your other tools and methodology evolve.

The Design Knowledge Architecture is comprised of the following:
- Knowledge engine compilers extract design knowledge contained in HDL code, testbenches, and assertions
- Knowledge database (KDB) stores crucial design knowledge to facilitate debug and understanding of your design

Figure 2: UVM-aware debug with built-in UVM intelligent viewers
Fast signal database (FSDB) captures and stores results from simulators, emulators, and formal tools that produce time/value sequences.

Application programming interfaces (APIs) provide open access to both databases and command-and-control mechanisms, enabling you to easily integrate the Verdi\(^3\) system with other verification tools and design environments.

**Third-Generation Debug Platform**

The Verdi\(^3\) platform builds on these core values with unprecedented personalization, customization and interoperability to maximize your debug productivity. You can speed debug of your large, complex designs, tailor the debug environment for the way you work, and directly integrate custom and third-party programs into your flows.

Faster, smaller and more efficient infrastructure:
- Multi-threaded database boosts performance by more than 45%
- Compact format reduces file size by more than 30%
- Parallel simulation dumping for 30% productivity gain

Flexible graphical user interface for personalization:
- Fast access to source files, videos and user-defined documents
- Personalize window layout to fit your debug needs
- Different working modes for different debug purposes
- Centralized spotlight to search commands, preferences and documents

Seamless VC Apps integration for customized flows:
- Rich array of VC Apps programs to tailor Verdi\(^3\) deployment for user flows
- Customizable menu/toolbars enable VC Apps functions in Verdi\(^3\) user interface
- Direct launch of VC Apps-enabled third-party tools/scripts from debug environment

**Interoperability and Customization**

The open architecture and interoperability of the Verdi\(^3\) platform also enables you to leverage your investment with integration of other commercial and proprietary verification tools. Its continuously expanding ecosystem provides out-of-the-box support for a wide range of commonly used tools, including:
- Simulators
- Emulators and accelerators
- Model checkers and other formal analysis engines

You can further maximize the effectiveness of your Verdi\(^3\) deployment with custom automation programs that leverage the power of its underlying design knowledge infrastructure for data mining and manipulation applications. Seamless integration with the VC Apps platform provides everything you need to quickly create and integrate Open Source scripts/utilities with the Verdi\(^3\) software that are designed to work reliably in your flow.
Personalized Graphical User Interface
The Verdi® platform provides Qt-based GUI and flexible usage models with:

- Welcome page for access to information and debug modes including:
  - Release notes, application notes and command reference documents
  - Predefined modes for different debug tasks, such as hardware debug, testbench debug, power-aware and property debug
  - Saved sessions with screenshots and text notes to restore/continue work
  - Training videos that demonstrate how to use key features
  - Customizable user links for sharing documents and web page

- Window management system to personalize layout of debug environment:
  - All components can be docked or undocked from major window
  - Major modules like nWave, nSchema and TFV can be configured as standalone window
  - One-click access to multiple source files within single window
  - Customizable toolbar, menu and hot keys for natural fit with debug tasks

- Spotlight search for fast, efficient navigation of commands, manuals and preference settings

Power-Aware Debug Module
The Verdi® Power-aware Debug Module accelerates the comprehension of power intent and automates the process of visualizing, tracing, and analyzing the source of power-related errors. It simplifies the verification of low-power SoC designs with unique automation technologies that help you visualize power intent with UPF/CPF, comprehend the impact of power intent on HDL design, and automate debug to determine whether unexpected design behavior is caused by functional logic or power-related issues.

Debug of Power-Aware Design Intent
The Verdi® Power-aware Debug Module accelerates comprehension of power intent and automates the process of visualizing, tracing, and analyzing the source of power-related errors. It simplifies the verification of low-power SoC designs with automation technologies that help you:

- **Visualize** power intent with UPF/CPF
- **Comprehend** impact of power intent on HDL design
- **Automate** debug to determine whether unexpected design behavior is caused by functional logic or power-related errors

Low-Power Debug Challenges
Low-power design requirements add more complexity to an already burdened verification process. RTL design can represent function and timing, but not power states. Gate-level design includes all information but the verification is too tedious. IP reuse is not possible when power requirements change. The need for a consistent format for describing power intent across the entire design and verification methodology creates new challenges that:

- Introduce the new PDML language into the verification flow (UPF/CPF)
- Require power-aware design environments to understand the power specification in the HDL design
- Create gaps between HDL debugging and power-aware debug tools

Power Intent Comprehension
The Verdi® Power-aware Debug Module provides a high-level overview of power intent that can be correlated with RTL design data for a complete power-aware understanding of designs. Core features include:

- Full UPF and CPF source code support allows you to import and compile power design data into FSDB

Figure 5: Users can personalize the layout and debug modes of the Verdi® environment
Easy-to-use Power Manager browser enables you to:
- Visualize power domains, power networks and power states/modes
- Annotate power intent on all design views (nTrace, nSchema and nWave)
- Locate power-related constraints for each power domain (power switch rules and retention, isolation and level-shifter rules)
- Drag-and-drop cross-probing between PDML and all design views to identify the origin of power-related problems

Temporal Flow View of power circuits that shows retention registers with save/restore signals and enables you to automatically trace control signals

Power state monitors that automatically identify the current power status of any signal at any simulation time

Special visualization for the values caused by power intent rules, such as isolation rules and power on/off

Reporting of signals impacted by power intent and related from/to instance (driver/load) with power domain

Automatic list X of selected signals with its driver and related power status

Explore, analyze, and identify problems with clock-related logic using powerful clock tree and clock domain extraction and analysis engines

Identify and isolate potential timing, power, or hot-spot problems early in the process through analysis of the netlist, Standard Delay File (SDF) information, and simulation results

View, understand, and explore the results of third-party clock, timing, and power analysis tools integrated with the nAnalyzer solution

Debug and isolate problems identified with the built-in or third-party capabilities using the powerful tracing and exploration features provided in the Verdi³ Automated Debug System

Bridging the Power Verification Gap

The Verdi³ Power-aware Debug Module bridges the power verification gap with a universal platform for integrating power formats and innovations with HDL debug automation. By making it easier to visualize, trace and analyze complex power behaviors in both HDL design and UPF/CPF code, power issues can be resolved earlier in the process while also saving valuable verification cycles.
Clock Tree and Clock Domain Analysis

Today’s complex designs require special and often sophisticated clock schemes. Multiple clock domains introduce the need to synchronize data passed between the different domains. Low-power stipulations require special clock gating techniques. The integration of third-party IP presents unexpected clock-related challenges at the IP interface. All of these factors increase the complexity of the clock circuitry and introduce more opportunities for implementation errors and thus incorrect operation. The nAnalyzer solution provides powerful built-in analysis engines that enable comprehensive exploration and debug of common clock-related issues.

Clock Analysis Features:

- Extract clock trees, clock domains, and clock domain crossings from RTL or gate-level descriptions
- Browse analysis results in text form or display and explore them in intuitive Verdi® schematics
- Import SDC information to identify clock sources and further constrain the analysis
- Pre-qualify clock tree synthesis (CTS) scripts to check for incorrect or incomplete constraints
- Develop and export SDC and CTS scripts to drive the CTS process
- Identify common problems such as flops outside of extracted clock trees and clocks connected to non-clock pins
- Verify proper cross-domain synchronization using user-defined synchronization methods
- Import SDF information and/or static timing analysis results to extract and display timing details — level delay, skew, slack time

Timing and Power Analysis

Although dedicated tools exist that provide comprehensive analysis and verification in the timing and power analysis domains, it is often useful to interactively explore timing and power issues outside of these tools. In some cases, the data required for a comprehensive analysis may not be available. In other cases, the dedicated tool may take a long time to complete its analysis, but only a quick approximation is needed. The nAnalyzer solution provides built-in analysis capability in both the timing and power domains to meet these needs.

Timing Analysis Features

- Import reports from common static timing analysis (STA) tools
- Browse critical path details in text form or display and explore them in intuitive Verdi® schematics
- Import SDF information and annotate the delay information on schematic views
- Calculate and display longest/shortest paths based on imported delay information

Power Analysis Features

- Analyze simulation results and identify areas of heaviest switching activity
- Sort and explore switching activity by signal, module, time, or time window
- Explore related logic using Verdi®’s powerful tracing and visualization capabilities

Figure 7: Analyze and debug design implementation issues in a single, comprehensive environment
Interoperability With Third-Party Tools

Complete analysis and resolution of timing, power and clock scheme related issues often require the application of best-in-class tools dedicated to one of these domains. The nAnalyzer solution enhances your usability of these dedicated tools by importing, displaying, and enabling the exploration of the analysis results in the familiar Verdi environment.

Embedded Software Debug

The Verdi HW SW Debug works together with the Verdi Debug Automation System to provide a comprehensive multi-window hardware and software debug view of a SoC design. When operating Verdi’s unified debug environment, both hardware and software engineers can simultaneously view their design at both hardware debug views and software debug views, including both C/C++ and assembly code as well as memory, register and breakpoint windows. Please refer to Verdi HW SW Debug datasheet for more details.

For more information about Synopsys products, support services or training, visit us on the web at: www.synopsys.com, contact your local sales representative or call 650.584.5000.