SYNOPSYS®

DATASHEET

VC Formal

Next-Generation Formal Verification

Delivering the highest performance and capacity with more design bugs found, more proofs, and faster coverage closure

Overview

SoC design complexity demands fast and comprehensive verification methods to accelerate verification and debug, as well as shorten overall schedule and improve predictability. The VC Formal™ next-generation formal verification solution has the capacity, speed and flexibility to verify some of the most difficult SoC design challenges, and includes comprehensive analysis and debug techniques to quickly identify root causes in the Verdi® debug platform. The VC Formal solution consistently delivers higher performance and capacity, with more bugs found, more proofs on larger designs and achieves faster coverage closure through the native integration with VCS® functional verification solution.

Verification Challenges and Modern Formal Verification

The VC Formal solution includes a comprehensive set of formal applications (Apps), including Property Verification (FPV), Automatic Extracted Properties (AEP), Coverage Analyzer (FCA), Connectivity Checking (CC), Sequential Equivalence Checking (SEQ), Register Verification (FRV), X-Propagation Verification (FXP), Testbench Analyzer (FTA), Regression Mode Accelerator (RMA), Datapath Validation (DPV), Functional Safety (FuSa), and a portfolio of Assertion IPs (AIP) for verification of standard bus protocols.

Formal methods are techniques that can perform analysis on the design independent of, or in conjunction with simulation, and have the power to identify design problems that can otherwise be missed until very late in the project schedule, or even in the manufactured silicon, when changes are expensive and debug is highly challenging and time consuming. When applied early in the design cycle, these methods can identify RTL issues such as functional correctness and completeness well before the simulation test environment is up and running.

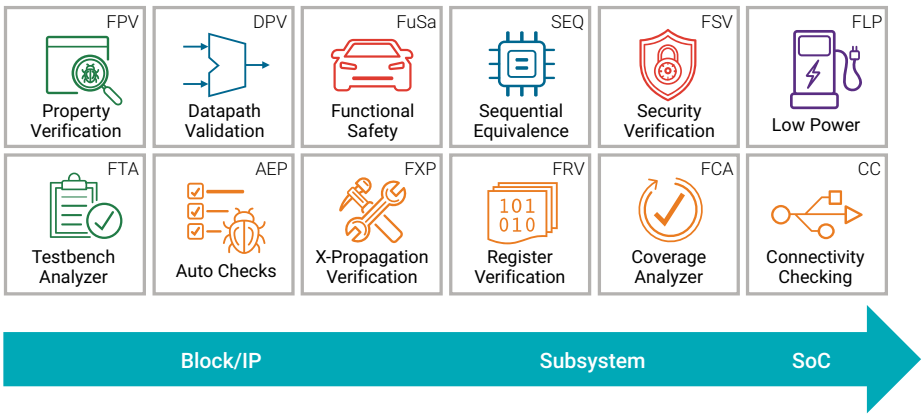


Figure 1: VC Formal Apps

Additional applications of formal technology can verify SoC connectivity correctness and completeness, and help isolate differences between two disparate versions of the design RTL.

Once the simulation environment is available, formal methods can complement simulation to add additional analysis for even better results, for example, for unreachable coverage goals.

By employing formal techniques at the appropriate time in the design and verification process, bugs can be caught significantly earlier in the project, including hard-to-find-bugs that typically elude verification until late in the project. The result is a higher quality design and overall schedule improvements as well as better predictability.

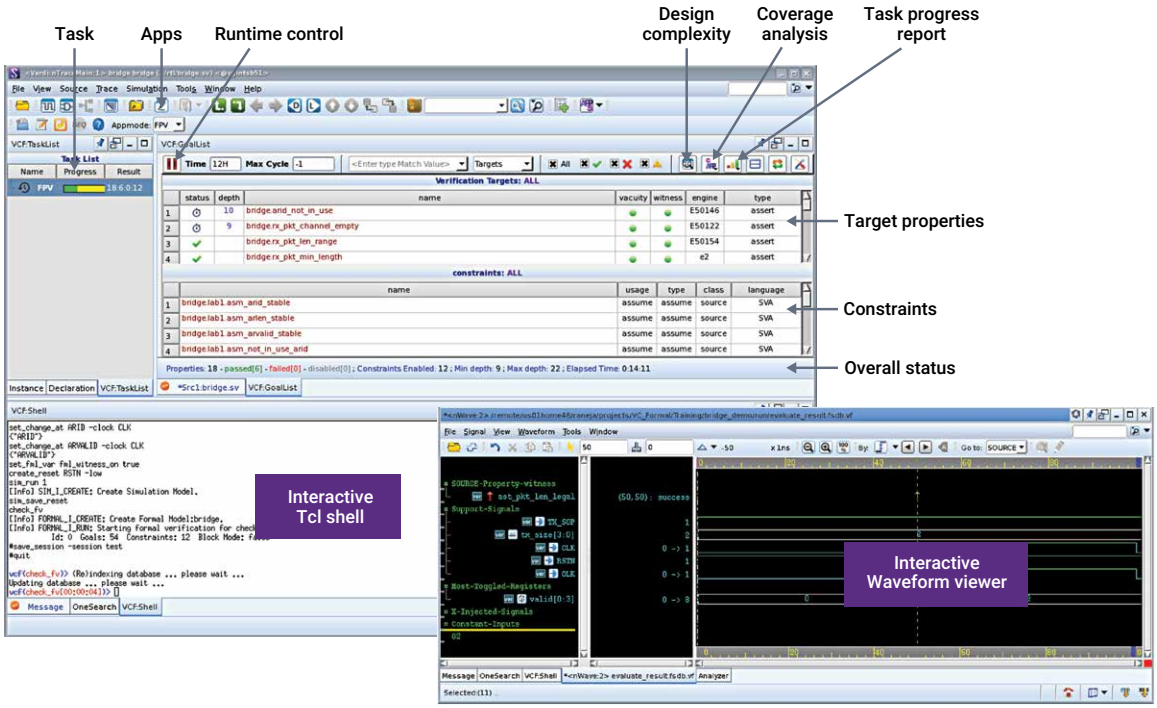


Figure 2: Unified Verdi Debug for Formal and Simulation

VC Formal

VC Formal is a high capacity, high performance formal verification solution that includes best-in-class algorithms, methodologies, databases and user interfaces. Built from the ground up, this solution was architected to address today's most challenging verification tasks, and provides the very latest and best formal verification engines available.

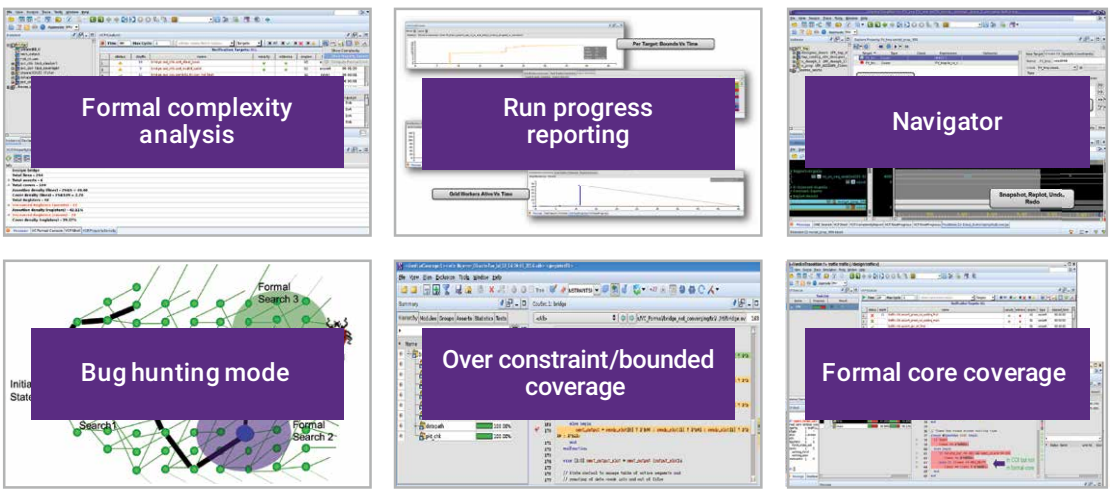


Figure 3: VC Formal Property Verification (FPV)

Key Features and Benefits

- **Assertion-Based Property Verification (FPV):** Formal proof-based techniques to verify SystemVerilog Assertion (SVA) properties to ensure correct operation across all possible design activity even before the simulation environment is available. Advanced assertion visualization, property browsing, grouping and filtering allows simple concise access to results.
- **Datapath Validation (DPV):** Integrated HECTOR™ technology within VC Formal and contains custom optimizations and engines for datapath verification (ALU, FPU, DSP etc.) using transaction level equivalence. This app leverages the Verdi graphical user interface for debug.
- **Automatic Extracted Property Checks (AEP):** Automatic functional analysis for out of bound arrays, arithmetic overflow, X-assignments, simultaneous set/reset, full case, parallel case, multi driver/conflicting bus and floating bus checks without the need for dedicated tests
- **Formal Coverage Analysis (FCA):** Complementing simulation flows, VC formal provides proof that uncovered points in coverage goals are indeed unreachable, allowing them to be removed from further analysis—saving significant manual effort
- **Connectivity Checking (CC):** Verification of connectivity at the SoC level. Flexible input format ensures ease of integration. Powerful debugging, including value annotation, schematic viewing, source code browsing and analysis reporting speeds analysis. Automatic root-cause analysis of unconnected connectivity checks saves significant debug time
- **Sequential Equivalency Checking (SEQ):** Verify modifications to the designs that do not affect the output functionality. For example, changes after register retiming, insertion of clock gating for power optimization or microarchitecture changes can be exhaustively verified without running any simulations.
- **Register Verification (FRV):** Formally verify that the attributes and behaviors of configuration and status register addresses and fields are correctly implemented and accessed. For example, attributes such as "read only", "read/write" or "reset value" can be defined in IP-XACT and formally verified, eliminating the need for directed simulation tests.
- **X-Propagation Verification (FXP):** Checks for unkown signal value (X) propagation through the design and allows tracing of the failed property to source X in the Verdi schematic and waveform.
- **Formal Testbench Analyzer with Certitude Integration (FTA):** Certitude™ provides the unique capability to assess the quality of formal environment. The native integration of Certitude with VC Formal provides meaningful property coverage measurements as part of formal signoff, and identifies any weaknesses such as missing or incorrect properties or constraints. The native integration delivers 5-10X faster performance compared to stand-alone fault injection methods
- **Security Verification (FSV):** Helps formally verify that secure data should not reach non-secure destinations and ensures data integrity, where non-secure data should not over-write (or reach) secure destinations
- **Regression Mode Accelerator (RMA):** Provides significant performance improvement to formal property verification using Machine Learning technology. Use of this app accelerates formal propety verification to achieve better convergence of formal proofs for subsequent runs and enables significant saving of compute resources in nightly formal regressions.
- **Assertion IP (AIP):** A portfolio of high performance and optimized Assertion IPs for standard bus protocols are available for all VC Formal Apps as well as VCS and ZeBu®. Some of the most popular titles available are Arm® AMBA® APB, AHB, AXI3, AXI4, ACE-lite, ACE, and CHI protocols.
- **Functional Safety (FuSa):** Functional safety verification is an essential requirement for automotive SoC and IP designs. This App formally identifies and classifies faults based on observability or detectability criteria. Complementing fault simulation with Z01X™, the combined solution reduces the effort and time required to achieve the test coverage closure.
- **Advanced Debug and Interactivity:** Advanced debugging interface built on Unified Verdi GUI based RTL and waveform visualization solutions, including schematic value annotation, on the fly assertion and constraint editing, proof progress feedback, and cone of influence analysis allows users greater visibility and control
- **Formal Scoreboard:** Exhaustively verifies the data integrity of data path designs. Ensures that data is transported through the design without being lost, re-ordered, corrupted or duplicated
- **Formal Coverage:** Advanced technologies enable formal metrics and methodology to achieve signoff for property verification

Unique Values

- **Industry-leading performance and capacity:** Ability to efficiently run on large designs. At least 5X performance and capacity gains
- **Excellent ease of adoption and ease-of-use:** Use model and commands tightly aligned with Synopsys implementation tools. VC Formal scripts are very similar to Design Compiler® TCL script, and share common commands and syntax
- **Run control features:** Enabling grid, pause/resume and save/restore
- **Excellent connectivity checking features, including schematic value annotation and root-cause analysis:** Significantly improving state-of-the-art debug, including debug of disconnected nets
- **Engine analysis and control:** Ability to examine and control engine activity across the run, and on the fly to better ensure closure on even the most challenging formal problems

Conclusion

Adoption of advanced formal verification techniques is growing rapidly to improve design verification. The automation of certain verification tasks, such as SoC connectivity verification, formal coverage analysis, and sequential equivalence checking significantly accelerates the ease of technology adoption. In addition, the common script environment and common setup makes adding launching new formal apps as simple as typing in a few new commands into a previously created script, or a simple click on the GUI. With the integration of industry-standard VCS® simulation and Verdi® debug solutions, the true power of formal verification can be realized.

By employing next-generation formal verification at opportune points in the design verification process, challenging bugs can be caught significantly earlier in the verification schedule, resulting in a higher quality design, overall schedule improvements as well as better predictability.

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

SYNOPSYS®
Silicon to Software™

©2022 Synopsys, Inc. All rights reserved. Synopsys is a trademark of Synopsys, Inc. in the United States and other countries. A list of Synopsys trademarks is available at synopsys.com/copyright.html. All other names mentioned herein are trademarks or registered trademarks of their respective owners.
08/10/22.CS11919_VCFormal_DS-0610019.