

Formality

Equivalence checking for DC Ultra

Overview

Formality[®] is an equivalence-checking (EC) solution that uses formal, static techniques to determine if two versions of a design are functionally equivalent.

The size and complexity of today's designs, coupled with the challenges of meeting timing, area, power and schedule, requires that the newest, most advanced synthesis optimizations be fully verifiable.

Formality supports all of the out-of-the-box DC Ultra optimizations and so provides the highest quality of results that are fully verifiable.

Formality supports verification of power-up and power-down states, multi-voltage, multi-supply and clock gated designs.

Formality's easy-to-use, flow-based graphical user interface and auto-setup mode helps even new users successfully complete verification in the shortest possible time.

Key benefits

- ▶ Perfect companion to DC Ultra – supports all DC Ultra default optimizations
- ▶ Intuitive flow-based graphical user interface
- ▶ Verifies low-power designs including power-up and power-down states
- ▶ Auto setup mode reduces “false failures” caused by incorrect or missing setup information
- ▶ Built-in distributed verification boosts performance
- ▶ Automated guidance boosts completion with DC Ultra
- ▶ Verifies full-custom and memory designs when including ESP technology

The most comprehensive equivalence checking solution

Formality delivers superior completion on designs compiled with DC Ultra, which uses Topographical Technology to achieve accurate correlation with post-layout timing, area and power, and provides advanced optimizations such as retiming, phase inversion and ungrouping. Formality is also fully compatible with DC Graphical used to predict and alleviate routing congestion. Designers no longer need to disable DC Ultra's powerful optimizations to get equivalence checking to pass. DC Ultra combined with Formality delivers maximum Quality of Results (QoR) that is fully verifiable.

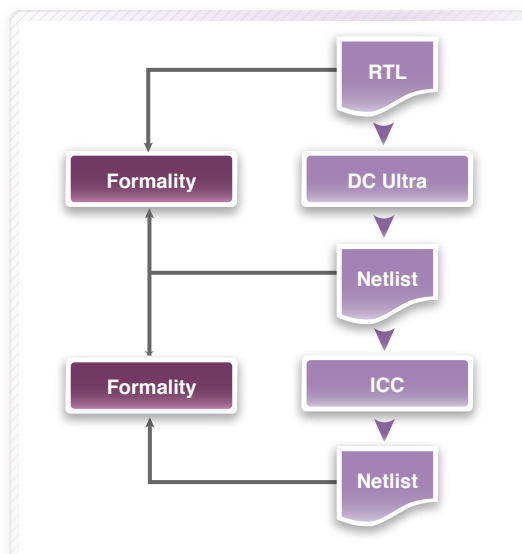


Figure 1: Formality equivalence checking solution.

Formality is easy to use

Auto-setup mode

Formality's auto-setup mode simplifies verification by reducing false failures caused by incorrect or missing setup information. Auto-setup applies setup information in Formality to match the assumptions made by DC Ultra, including naming styles, unused pins, test inputs and clock gating.

Critical files such as RTL, netlists and libraries are automatically located. All auto-setup information is listed in a summary report.

Guided setup

Formality can account for synthesis optimizations through the use of a guided setup file automatically generated by DC Ultra. Guided setup includes information about name changes, register optimizations, multiplier architectures and many other transformations that may occur during synthesis. This correct-by-construction information improves performance and first-pass completion by utilizing the most efficient algorithms during matching and verification.

Formality-guided setup is a standard, documented format that removes unpredictability found in tools relying on log file parsing.

Independent verification

Every aspect of a guided setup flow is either implicitly or explicitly verified, and all content is available for inspection in an ASCII file.

Graphical user interface

Formality provides a flow-based graphical user interface designed to promote out-of-the-box usability.

Reference and implementation designs can be shown simultaneously. Automatic cone pruning reduces complexity by showing only the differences between the reference and implementation designs.

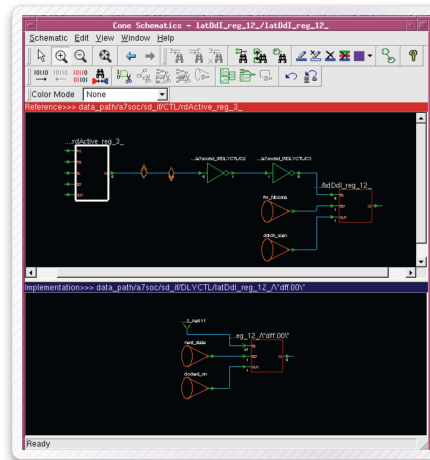


Figure 2: Automatic cone pruning improves schematic readability when debugging.

Hier-IQ™ technology

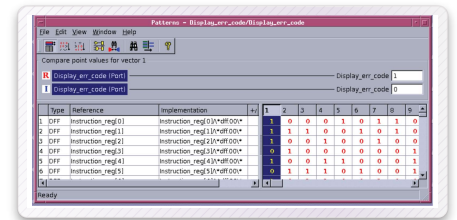
Patented Hier-IQ technology provides the performance benefits of hierarchical verification with flat verification's out-of-the-box usability.

Error-ID technology

Error-ID identifies the exact logic causing real functional differences between two design representations. Error-ID can isolate and report several logic differences when multiple discrepancies exist. Error-ID will also present alternative logic that can be changed to correct a given functional difference; this flexibility allows the designer to select the change that is easiest to implement.

Failing pattern display window

All failing input patterns can be viewed in a familiar spreadsheet-like format. The failing pattern window is an ideal way to quickly identify trends indicating the cause of a failing verification or improper setup.



Type	Reference	Implementation	1	2	3	4	5	6	7	8	9
1. DFF	Instruction_reg_01	Instruction_reg_01*nrff00*	0	0	0	0	1	0	1	1	0
2. DFF	Instruction_reg_01	Instruction_reg_01*nrff00*	1	1	0	0	1	0	1	0	0
3. DFF	Instruction_reg_02	Instruction_reg_02*nrff00*	1	0	0	1	0	0	1	0	0
4. DFF	Instruction_reg_03	Instruction_reg_03*nrff00*	0	1	0	0	0	0	0	0	1
5. DFF	Instruction_reg_04	Instruction_reg_04*nrff00*	1	0	1	1	0	0	0	0	1
6. DFF	Instruction_reg_05	Instruction_reg_05*nrff00*	0	1	1	0	1	0	0	0	1

Figure 3: Problem areas can be easily identified by visual inspection of the Failing Pattern Window.

Power-aware verification

Formality is fully compatible with Power Compiler and verifies power-up and power-down states, multi-voltage, multi-supply and clock gated designs.

When a reference design block is powered up, Formality verifies functionality. If the implementation design powers up differently, failing points will occur.

Formality functionally verifies that the implementation powers down when the reference powers down and will detect functional states where the implementation does not power down as expected. The valid power states are defined in the power state table (PST).

Power intent is supplied to Formality through IEEE 1801 Unified Power Format (UPF).

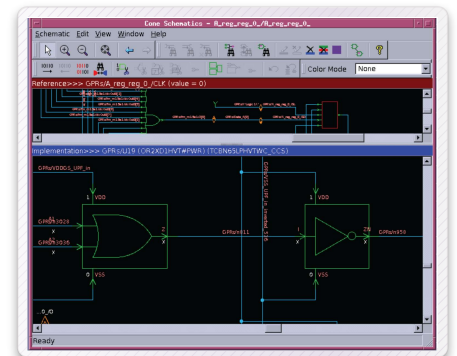


Figure 4: Power connectivity is easy to see and debug from the schematic view.

Accelerated time to results

Distributed verification

Formality's performance is enhanced with distributed verification. This inherent Formality feature verifies the design using up to four processors simultaneously to reduce verification time.

Other time-saving features

Formality's Hierarchical Scripting provides a method to investigate sub-blocks without additional setup and is ideal for isolating problems and verifying fixes.

The Source Browser opens RTL and netlist source files to highlight occurrences of a selected instance. This can help users correlate between the RTL and gate-level design versions.

Error Region Correlation provides a quick, visual identification of the logic from one design that correspond to the errors isolated by Error-ID within the other.

Command Line Editing allows you to take advantage of history and common text editor commands when working from Formality's command line.

ECO verification and implementation

Formality can help debug and implement functional ECOs by identifying areas of logic that need to be changed to bring two design representations into functional correlation. Additionally, Formality can be used to prove that ECOs have not had any unintended functional impact.

Transistor verification

ESP combines with Formality to offer fast verification of custom circuits, embedded memories and complex I/Os. ESP technology directly reads existing SPICE and behavioral RTL models and does not require restrictive mapping or translation.

Input formats

- ▶ Synopsys DC, DDC, Milkyway
- ▶ IEEE 1800 SystemVerilog
- ▶ Verilog-95, Verilog-2001
- ▶ VHDL-87, VHDL-93
- ▶ IEEE 1801 Unified Power Format (UPF)
- ▶ Spice (Formality-ESP)

Guided setup formats

- ▶ Synopsys V-SDC
- ▶ Formality Guide Files (SVF)

Platform support

- ▶ SPARC Solaris
- ▶ IBM AIX
- ▶ Linux Suse, Red Hat and Enterprise

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.