DATASHEET

Formality

Equivalence Checking and Interactive ECO

Independent formal verification of Design Compiler and Fusion Compiler synthesis results, with built-in intelligence delivering the highest verifiable QoR

Overview

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

Formality delivers capabilities for ECO assistance and advanced debugging to help guide the user in implementing and verifying ECOs. These capabilities significantly shorten the ECO implementation cycle.

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 Design Compiler® and Fusion Compiler™ 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.

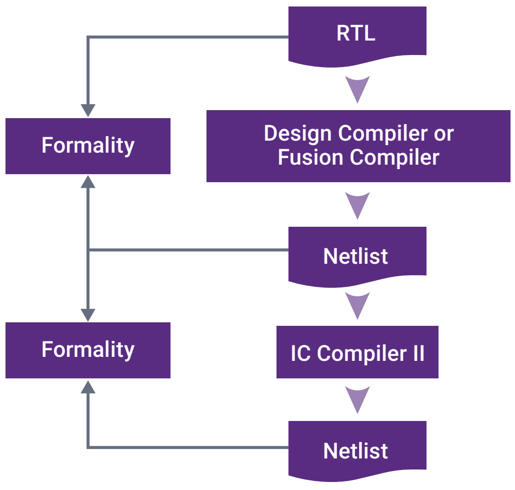


Figure 1: Formality equivalence checking solution

Key Benefits

- Perfect companion to Design Compiler and Fusion Compiler—supports all default optimizations
- Intuitive flow-based graphical user interface
- Verifies low-power designs including power-up and power-down states
- ECO implementation assistance, fast verification of the ECO, and advanced debugging
- Auto setup mode reduces “false failures” caused by incorrect or missing setup information
- Multicore verification boosts performance
- Automated guidance boosts completion with Design Compiler and Fusion Compiler
- Verifies full-custom and memory designs when including ESP technology

Formality

The Most Comprehensive Equivalence Checking Solution

Formality delivers superior completion on designs compiled with Design Compiler or Fusion Compiler. Design Compiler is the industry leading family of RTL Synthesis solutions. Fusion Compiler is the next generation RTL-to-GDSII implementation system architected to address the complexities of advanced process node design. Designers no longer need to disable the powerful optimizations available with Design Compiler or Fusion Compiler to get equivalence checking to pass. Design Compiler/Fusion Compiler combined with Formality delivers maximum quality of results (QoR) that are fully verifiable.

Easy to Use with 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 Design Compiler or Fusion Compiler, 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 using a guided setup file automatically generated by Design Compiler or Fusion Compiler. 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.

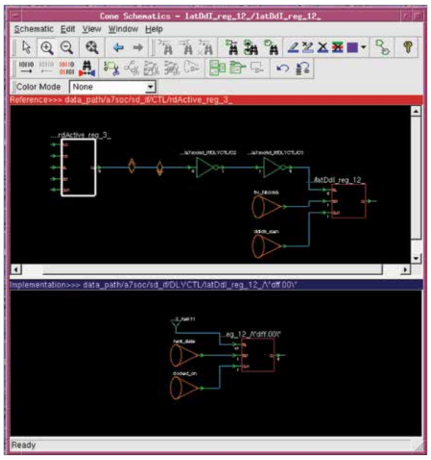

Cone Schematics - latDdI_reg_12_/latDdI_reg_12_
Schematic Edit View Window Help
Color Mode None
Reference>>> data_path/a7soc/sd_if/CTL/rdActive_reg_3_
...rdActive_reg_3_
/latDdI_reg_12_
Implementation>>> data_path/a7soc/sd_if/DLYCTL/latDdI_reg_12_/*dff.00*
eg_12_/*dff.00*
Ready

Figure 2: Automatic cone pruning improves schematic readability when debugging

Hier-IQ Technology

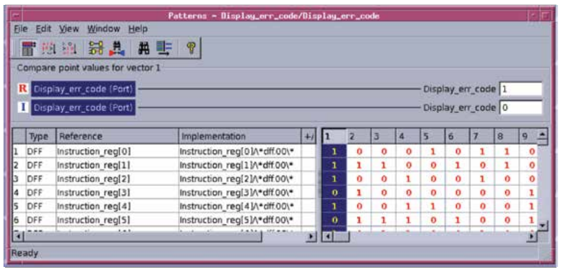
Patented Hier-IQ technology provides the performance benefits of hierarchical verification with flat verification's out-of- the-box usability.

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.


Patterns - Display_err_code/Display_err_code
File Edit View Window Help

Compare point values for vector 1

R Display_err_code (Port) — Display_err_code 1

I Display_err_code (Port) — Display_err_code 0

	Type	Reference	Implementation	+/	1	2	3	4	5	6	7	8	9
1	DFF	Instruction_reg[0]	Instruction_reg[0]/*dff.00*		1	0	0	0	1	0	1	1	0
2	DFF	Instruction_reg[1]	Instruction_reg[1]/*dff.00*		1	1	1	0	0	1	0	1	0
3	DFF	Instruction_reg[2]	Instruction_reg[2]/*dff.00*		1	0	0	1	0	0	1	0	0
4	DFF	Instruction_reg[3]	Instruction_reg[3]/*dff.00*		0	1	0	0	0	0	0	0	1
5	DFF	Instruction_reg[4]	Instruction_reg[4]/*dff.00*		1	0	0	1	1	0	0	0	1
6	DFF	Instruction_reg[5]	Instruction_reg[5]/*dff.00*		0	1	1	1	0	1	0	0	1

Ready

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

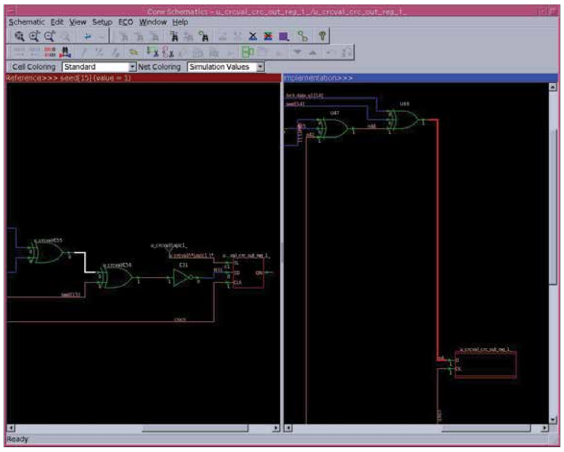
Power-aware Verification

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).



Cell Coloring Standard Net Coloring Simulation Values

Ready

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

Accelerated Time to Results

Formality's performance is enhanced with multicore verification. This Formality capability allows verification of the design using up to four cores 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.

Interactive ECO

Key Benefits

Provides GUI-driven ECO implementation assistance, fast ECO verification, and advanced debugging. Formality guides the user through the implementation of ECOs, and then quickly verifies only the changed logic.

Formality Interactive ECO Flow

Formality uses the ECO RTL and an unmodified netlist. Guided GUI driven changes are made to the netlist. Once the ECO has been implemented, a quick verification is run on only the affected logic cones, eliminating the need for a full verification run on the design to verify that the ECO was implemented correctly.

Once all ECO's are implemented and fully verified, a list of IC Compiler™ commands is generated to assist in implementing the physical changes to the design.

ECO Guidance

Formality highlights equivalent nets between the reference and implementation designs, and nets that have lost their equivalence due to the ECO changes in the reference. This helps the designer quickly identify where the change should be made in the implementation.

Implementing the ECO

Editing commands in Formality are used to modify the netlist in-place using the GUI.

Rapid ECO Verification

Formality can identify and verify just the portion of the design affected by the ECO. This ensures that the ECO was implemented correctly. If the ECO verification fails, the ECO can be interactively "undone" and new edits can be made again. Once the partial verification passes, the changes are committed. This partial verification eliminates having to verify the entire design to assure that the ECO was implemented correctly, dramatically reducing the total time required to implement and verify the ECO.

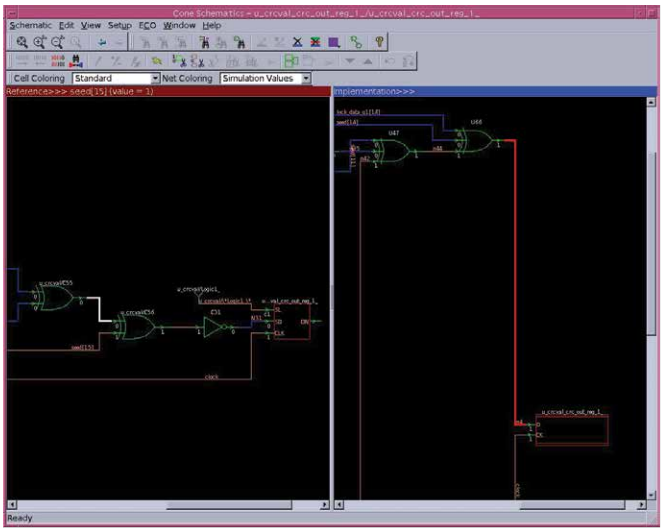


Figure 5: Equivalent net is highlighted between Reference design (left) and Implementation design (right)

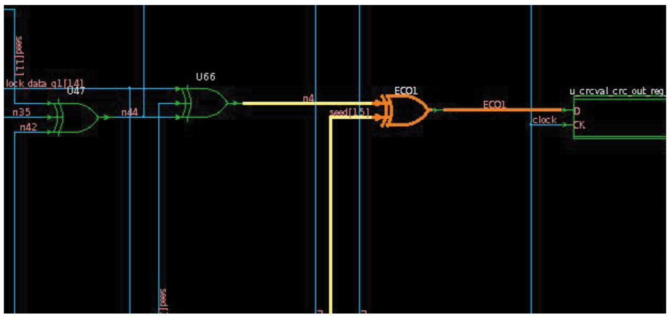


Figure 6: On a completed ECO, the schematic shows the nets affected by ECO in yellow, and the new component and net in orange

```
Verification SUCCEEDED
ATTENTION: Only a subset of the compare points was verified.
           Use remove_verify_points to do a full verification.
ATTENTION: synopsys_auto_setup mode was enabled.
           See Synopsys Auto Setup Summary for details.
-------------------------------------------------------------
```

Figure 7: Formality transcript shows a successful partial verification of the portion of the design that was affected by the ECO

Interface with IC Compiler II

Once the ECO's are implemented and verified, a final complete verification run is performed to assure that the ECO RTL and the ECO netlist are functionally equivalent.

Formality produces IC Compiler II compatible ECO command file, easing the implementation in the physical design.

Advanced Debugging

Formality incorporates advanced debugging capabilities that help the designer identify and debug verifications that do not pass. The designer can find compare points, equivalences (and inverted-equivalences) between reference and implementation designs, perform "what if" analysis by interactively modifying the designs, and verify equivalence between two (or multiple) points.

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)

Guided Setup Formats

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

Platform Support

- Linux Suse, Red Hat and Enterprise
- SPARC Solaris

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.