

---

# FormalCheck Product Notes

---

## July 2002

These Product Notes give you an overview of the features and capabilities of this release of FormalCheck.

These Product Notes discuss the following topics:

- [New Features and Enhancements](#) on page 5—Describes the features in this version of the product.
- [Hardware and Software Requirements](#) on page 9—Tells you where to locate information about the hardware and software requirements for this release.
- [Installation and Licensing](#) on page 9—Tells you how to access information and procedures required to install Cadence products.
- [Customer Support](#) on page 9—Tells you how to contact the Cadence Customer Response Center.
- [SourceLink](#) on page 9—Tells you how to obtain current information on the tools.
- [Documentation](#) on page 10—Describes the ways you can view online documentation.

## New Features and Enhancements

The following new features and enhancements have been made to FormalCheck in this release:

- Performance:
  - Launching queries
  - Design model optimized for verification (Reduction of temporaries/state variables)
- New SimVision Waves waveform viewer – consistency with the NC-Sim simulator.
  - SimVision Waves is a more consistent debugging interface than was previously provided. The interface is also intended to address any remaining colormap problems that FormalCheck users have reported in the past.

## Formal Check Product Notes July 2002

---

- ❑ The *FormalCheck User Guide* documents the new interface in SimVision Waves for going from a signal value in a failure waveform to the HDL source line responsible for a signal value.
- ❑ SimVision Waves has many new features that you can learn about by reading the *SimVision Waves User Guide*.
- Links back to HDL source improvements.
- Enhanced pragma support.
  - ❑ The FormalCheck compilation front end now uses `-lexpragma` by default. This corrects several user-reported problems regarding certain problematic combinations of ``include` statements and synthesis pragmas.
- New common SR (connectivity) generation module; refocus on FQL query interface and future standards.
  - ❑ After a study of the usage patterns and priorities of the FormalCheck customer community, Cadence made a conscious decision to phase out the relatively new “HQL” embedded query language, in order to phase in a new standardized SR generation module. This strategy will better support emerging embedded industry-standard assertion languages in future releases. The widely-used FQL FormalCheck query language remains, to support existing FormalCheck customers.
  - ❑ For help migrating any existing HQL queries to use the FQL language or other Cadence-supported assertion languages, contact `fc_support@cadence.com`.
- Quality, improved testing, and PCR backlog reduction.
- Corrections for online FormalCheck documentation, accessed via the GUI *Help* menu, and the `cdsdoc` command.
- An extended *FormalCheck Known Problems and Solutions* document, that answers most common support questions and provides you with tips using the tool.
- An examples area, located at `.../tools/fcheck/examples`. This directory includes the examples that are referenced in the documentation, install tests, a tutorial, and a quick-reference card.

## New Features and Enhancements from Previous Releases

- Simplified installation—No need to set the `LD_LIBRARY_PATH` environment variable.

## Formal Check Product Notes July 2002

---

- More robust Graphical User Interface support—Particularly for the newest functions, including:
  - The creation of properties with user-friendly templates, queries, and state variables.
  - Stronger support for the reduction manager, query runtime control, and query status monitoring in the GUI.
- Performance improvement in problematic cases for vacuity checking.
- Improved constant propagation (performance) in designs using variable indexes.
- The `fcsh` command has been replaced with the `formalcheck` command in all cases to eliminate redundancy.

This release of FormalCheck also contains the following major improvements introduced in the FormalCheck 3.0 release and repeated here since they may be new to some FormalCheck users:

- INCA™ compilers for both VHDL and Verilog RTL support
- A Tcl/Tk-based textual command-line interface

### INCA Compilers For VHDL and Verilog

FormalCheck 3.x uses the INCA compilers for both VHDL and Verilog designs, which gives you the following benefits:

- Your design uses the same parser and elaborator that is used by the Verification Cockpit, including, HDL Analysis and Lint tool (HAL), and the NC simulators.

The INCA compilers are combined in FormalCheck 3.0 to provide enhanced capabilities in the following areas:

- Simple and compound pragma statements
- Synchronous and asynchronous coding styles
- The `#include`, `'define`, `'ifdef`, and `use` statements
- The Verilog `initial` statement
- NC-simulator-style options that you use to handle library elements
- Verilog parameterization
- VHDL generics
- Concatenation and bit slices in expressions

## Formal Check Product Notes July 2002

---

- Inout ports
- Tri-state logic
- Wired logic (wired-AND/wired-OR)
- Function calls
- Arrays (one dimensional and two dimensional arrays)
- VHDL conversion functions at ports of instantiation
- Verilog concatenation operation at ports of an instantiation
- VHDL records, record of arrays, and array of records

### The FormalCheck Command-Line Interface

The FormalCheck command-line interface implements the FormalCheck 3.x programming shell environment based on Tcl/Tk technology.

### Model Checking Flows

FormalCheck 3.x supports the following model checking flows:

- A basic verification flow—In the basic verification flow, you verify queries using the default settings of the `verify` command. You should use the basic verification flow only when you are absolutely certain that your queries are correctly formulated and the complexity of the query can be handled with the computing resources available.
- Analyzing queries using *AutoCheck*—The *AutoCheck* feature ranks queries based on their degree of complexity. This improves your productivity by letting you focus on those queries that are bound to converge and by pointing out to you which queries should be deferred for batch runs.
- Verification using the clock extraction mode—The clock extraction mode gives you a simple solution to overcome state explosion problems, which can cause a query or a casequery to run out of memory.

### Want More Information on FormalCheck?

Please contact the FormalCheck Product Engineering Team by sending e-mail to:

`fc_support@cadence.com`