cadence conformal lec user guide

Cadence Conformal LEC User Guide: Mastering Logical Equivalence Checking

cadence conformal lec user guide is an essential resource for engineers and designers working with digital verification and formal equivalence checking. Whether you are a newcomer to the field or an experienced professional aiming to streamline your design verification process, understanding how to effectively use Cadence Conformal LEC (Logical Equivalence Checking) can significantly enhance your workflow. This article unpacks the essentials of Cadence Conformal LEC, offering a clear path to mastering its features, commands, and best practices while naturally weaving in related concepts such as RTL verification, netlist comparison, and design signoff.

What is Cadence Conformal LEC?

Before diving into the user guide specifics, it’s important to understand what Cadence Conformal LEC is and why it’s crucial for modern chip design. Cadence Conformal LEC is a tool designed to perform logical equivalence checking between two versions of a digital design. It compares RTL (Register Transfer Level) code with synthesized netlists, or two netlists, to ensure they are functionally equivalent. This verification step is key to catching discrepancies introduced during synthesis, optimization, or design changes, thereby preventing costly errors late in the design cycle.

Why Use Logical Equivalence Checking?

Logical equivalence checking is a cornerstone of design verification because it provides:

- Assurance that design transformations preserve intended functionality
- Early detection of bugs that simulation might miss
- Automated comparison between RTL and gate-level netlists
- Faster design closure and improved confidence in signoff quality

Cadence Conformal LEC specializes in handling large, complex designs efficiently, making it a preferred choice in the semiconductor industry.

Getting Started with Cadence Conformal LEC

If you’re new to Cadence Conformal LEC, starting with the right setup and understanding the basic workflow can save you hours of frustration.

Installation and Environment Setup

First, ensure you have the correct license and installation package from Cadence. The tool typically runs on Linux environments, so having a supported OS like CentOS or RHEL is recommended. After installation, set up environment variables like `CONFORMAL_HOME` and update your `PATH` so you can invoke

Conformal commands from the shell easily.

Basic Workflow Overview

The general flow of a Conformal LEC project involves:

1. **Input Preparation**: Gather your RTL files and synthesized netlists. These can be in Verilog, VHDL, or EDIF formats.
2. **Run Setup Scripts**: Create a setup file (usually a `.tcl` script) where you specify design files, top modules, and any configuration options.
3. **Launch the Tool**: Execute the conformal command to start the equivalence checking process.
4. **Review Reports**: Analyze the output reports for equivalence results, any mismatches, or warnings.
5. **Debug Discrepancies**: Use Conformal's debug capabilities to pinpoint and resolve equivalence failures.

Key Features of Cadence Conformal LEC Explored

Understanding the core features will help you leverage the tool to its full potential.

Netlist and RTL Comparison

Conformal LEC excels at comparing RTL against synthesized gate-level netlists. It supports various design languages and can handle complex constructs like clock gating and resets. This feature helps verify that synthesis tools have not altered the intended design functionality.

Incremental Equivalence Checking

One of the most powerful aspects of Conformal LEC is its ability to handle incremental verification. When only parts of the design change, you don't need to re-run equivalence checks on the entire design. Instead, incremental LEC focuses on affected modules, saving considerable time during regression runs.

Handling Complex Design Constructs

Modern SoC designs include multiple clock domains, asynchronous resets, and power management features. Conformal LEC provides advanced options to properly interpret these constructs, ensuring accurate equivalence checking without false positives.

How to Write a Cadence Conformal LEC Setup Script

The setup script is the heart of your equivalence checking session. It tells the tool which files to consider, which top modules to verify, and how to handle specific design features.

Basic Setup Script Structure

A typical Conformal LEC setup script includes commands such as:

```tcl
# Load design files
add_design_file -rtl rtl_design.v
add_design_file -gate synthesized_netlist.v

# Define top modules
set_top rtl_top_module
set_top gate_top_module

# Run equivalence check
run_equivalence
```

This minimal script instructs Conformal to load the RTL and gate-level designs, sets the top modules for each, and performs the equivalence check.

Advanced Options in Setup Scripts

You can customize the verification by adding commands like:

- `set_clock` to define clock signals explicitly
- `add_ignore` to exclude certain signals or modules from comparison
- `set_reset` for asynchronous or active-low resets
- `set_power` to account for power domains or retention cells

These commands help tailor the tool’s behavior to your specific design needs and reduce false mismatch reports.

Tips for Effective Use of Cadence Conformal LEC

Maximizing the benefits of Conformal LEC requires some practical know-how. Here are tips from experienced users:

Organize Your Design Files Clearly

Keep your RTL and gate-level netlists organized in separate directories with consistent naming conventions. This helps prevent errors in file referencing within setup scripts.

Use Incremental Checks Whenever Possible

Running full equivalence checks on large designs can be time-consuming. Make use of incremental checking to focus only on changed portions, speeding up regression cycles.

Leverage Debug Features

When mismatches occur, use generated reports and waveforms to drill down to the root cause. Conformal provides options to dump failing signals, trace logic cones, or even generate debug scripts.

Match Clocks and Resets Precisely

Equivalence failures often stem from clock domain mismatches or reset handling differences. Define these signals explicitly in your setup to avoid false positives.

Common Challenges and How to Overcome Them

Even with a powerful tool like Cadence Conformal LEC, some challenges are common. Understanding and addressing them improves your verification experience.

False Mismatches Due to Synthesis Optimizations

Synthesis tools sometimes optimize away logic or rename signals, causing apparent mismatches. To handle this:

- Use the `add_equiv` command to specify equivalent signals manually.
- Exclude non-functional signals from comparison.
- Utilize ‘smart’ matching features in Conformal that recognize common synthesis transformations.

Multi-Clock Domain Issues

Designs with multiple asynchronous clocks can confuse equivalence checking. Ensure clocks are properly defined, and use clock domain crossing constraints to help the tool understand timing relationships.

Handling Black Boxes and IP Blocks

Third-party IP or black-box modules may not have RTL sources available for comparison. Isolate these blocks in the setup script or use abstract models to avoid mismatches.

Integrating Cadence Conformal LEC into Your Verification Flow

For modern design teams, Conformal LEC fits into a broader verification ecosystem, often integrated with other Cadence tools like JasperGold or Xcelium.

Automating Equivalence Checking

By scripting Conformal runs and integrating them into build systems (e.g., Jenkins), you can automate regression equivalence checks. This continuous verification approach catches issues early in the design cycle.

Combining Formal and Simulation

While simulation tests design behavior under various scenarios, formal equivalence checking with Conformal LEC ensures that the implementation matches the intended RTL logic thoroughly, complementing simulation coverage.

Conclusion: Becoming Proficient with Cadence Conformal LEC

Mastering Cadence Conformal LEC is a valuable skill for anyone involved in digital design verification. This user guide overview has highlighted the essentials—from understanding the tool's purpose, setup, and features to practical tips for effective use and troubleshooting. By integrating Conformal LEC into your verification flow and leveraging its advanced capabilities, you can improve design quality, reduce verification time, and gain confidence in your chip's correctness. Keep exploring the official documentation and experimenting with different options to unlock the full potential of this powerful equivalence checking tool.

Frequently Asked Questions

What is the Cadence Conformal LEC User Guide?

The Cadence Conformal LEC User Guide is an official documentation that provides detailed instructions on how to use the Logical Equivalence Checking (LEC) tool within the Cadence Conformal suite. It covers installation, setup, command references, and best practices.

Where can I find the Cadence Conformal LEC User Guide?

The Cadence Conformal LEC User Guide is available on the Cadence support website or through the Cadence Help portal, accessible to licensed users. It can also be accessed via the Conformal tool’s help menu.

What are the main features explained in the Cadence Conformal LEC User Guide?

The guide explains features such as design setup, equivalence checking flows, constraint handling, reporting, debugging mismatches, and integration with other Cadence tools.

How does the Cadence Conformal LEC User Guide help with debugging mismatches?

The guide provides step-by-step instructions on interpreting mismatch reports, isolating differences, and applying debug techniques to resolve logical equivalence issues between designs.

Does the Cadence Conformal LEC User Guide cover scripting and automation?

Yes, the User Guide includes sections on using Tcl scripts and command-line options to automate equivalence checking processes and customize tool behavior.

Can the Cadence Conformal LEC User Guide help new users get started with LEC?

Absolutely. It contains introductory chapters that guide new users through basic concepts, initial setup, and running their first equivalence check.

What types of designs are supported according to the Cadence Conformal LEC User Guide?

The guide specifies support for RTL, gate-level netlists, synthesized designs, and various design formats commonly used in ASIC and FPGA verification.

How often is the Cadence Conformal LEC User Guide updated?

The User Guide is typically updated with each major release of the Conformal tool, reflecting new features, enhancements, and changes in workflows.

Are there troubleshooting tips available in the Cadence Conformal LEC User Guide?

Yes, the guide includes a troubleshooting section that addresses common issues, error messages, and recommended solutions to help users resolve problems efficiently.

Additional Resources

Cadence Conformal LEC User Guide: Unlocking Advanced Formal Verification

cadence conformal lec user guide serves as an essential resource for engineers and verification professionals aiming to harness the full potential of Cadence's Conformal Logic Equivalence Checking (LEC) tool. As semiconductor designs grow increasingly complex, ensuring that different representations of a design remain functionally equivalent is critical. The Cadence Conformal LEC user guide not only offers detailed instructions for effective tool usage but also provides insights into best practices and troubleshooting strategies, making it indispensable for both novice and experienced users.

Understanding Cadence Conformal LEC

Cadence Conformal LEC is a formal verification tool designed to perform logic equivalence checking between two versions of a digital design—commonly the RTL and gate-level netlist. It helps verify that optimizations, synthesis transformations, or changes during design stages do not alter the intended functionality. Unlike simulation-based verification, which tests specific input scenarios, Conformal LEC uses mathematical algorithms to prove equivalence exhaustively.

The user guide delves into the fundamental architecture of the tool, describing how it leverages advanced techniques like formal analysis, SAT solving, and structural analysis to expedite verification. The guide also explains how Conformal LEC fits into the broader Cadence verification ecosystem, often working alongside tools like JasperGold for enhanced coverage.

Key Features Highlighted in the User Guide

The Cadence Conformal LEC user guide outlines several features that distinguish the tool in the competitive landscape of logic equivalence checking:

- **Scalability:** Ability to handle designs ranging from small IP blocks to large SoCs with billions of gates.
- **Incremental Checking:** Supports iterative design flows by efficiently verifying only the changed portions of a design.
- **Debugging Support:** Provides detailed reports and graphical representations of mismatches to facilitate swift root cause analysis.
- **Automatic Constraint Handling:** Integrates design constraints seamlessly, reducing manual effort and potential errors.
- **Integration with Verification Flows:** Compatible with scripting interfaces and continuous integration pipelines.

These capabilities are not only described in procedural terms but are also accompanied by practical examples, helping users appreciate their application in real-world scenarios.

Getting Started: Installation and Setup

The Cadence Conformal LEC user guide begins with a thorough walkthrough of installation prerequisites, environment configuration, and initial setup. Users are guided through:

1. System requirements, including supported operating systems and hardware recommendations.
2. Licensing setup, which is crucial for enabling tool features and ensuring compliance.
3. Configuration of environment variables and paths to integrate with existing Cadence tools.
4. Verification of installation through sample projects, enabling users to confirm operational readiness.

The guide emphasizes the importance of aligning the tool version with the rest of the software stack, as mismatches can lead to compatibility issues.

Command-line Interface and GUI Usage

One of the strengths of Conformal LEC is its versatile user interface options. The user guide details how to operate the tool via:

- **Command-line Interface (CLI)**: Suitable for batch processing and automation, the CLI section outlines key commands, syntax, and scripting tips to streamline workflows.
- **Graphical User Interface (GUI)**: Ideal for interactive analysis, the GUI chapter explains navigation, report visualization, and graphical debugging tools.

By covering both interfaces, the guide caters to diverse user preferences and use cases, ensuring accessibility for teams with varying expertise levels.

Workflow and Best Practices

Central to the Cadence Conformal LEC user guide is the detailed exposition of typical equivalence checking workflows. From preparing input files to interpreting results, the guide encourages a methodical approach:

Preparation of Input Files

The guide stresses the importance of consistent and clean input design representations. It advises on:

- Ensuring RTL and netlist files are properly preprocessed and free of synthesis artifacts that could confound equivalence checking.
- Managing constraints files to represent don't-care conditions and timing assumptions accurately.
- Using blackboxing judiciously to exclude non-essential IP blocks while maintaining verification integrity.

Running the Equivalence Check

Users are walked through command sequences and parameter tuning strategies to optimize runtime and coverage. The guide explains approaches to handle common challenges such as:

- Dealing with intentional design differences like pipeline balancing or retiming.
- Employing heuristics and abstraction techniques to reduce complexity.
- Utilizing incremental checking to focus on changes and accelerate verification cycles.

Analyzing and Debugging Results

When equivalence fails, the guide equips users with diagnostic tools including:

- Mismatch reports that pinpoint the exact location and nature of discrepancies.
- Simulation waveforms generated from counterexamples to aid in understanding failures.
- Interactive debugging sessions through the GUI to traverse design hierarchies and signals.

Such capabilities are crucial in shortening the feedback loop during design iterations.

Comparative Insights and Industry Applications

In the broader context of logic equivalence checking tools, the Cadence Conformal LEC user guide provides comparative insights that help users evaluate its fit relative to alternatives like Synopsys Formality or Mentor Questa LEC. It highlights Conformal's advantages in scalability, user interface flexibility, and seamless integration within Cadence's toolchain.

Industries such as semiconductor IP development, automotive electronics, and data center chip design benefit significantly from Conformal LEC's robust verification capabilities. The user guide includes case studies illustrating how the tool has enabled faster signoff cycles and reduced verification costs in complex projects.

Pros and Cons Addressed in the Guide

While the guide primarily focuses on maximizing tool benefits, it also candidly discusses limitations:

- **Pros:** High accuracy, extensive debugging features, and strong automation support.
- **Cons:** Steep learning curve for beginners and resource-intensive processing on very large designs.

This balanced view equips users with realistic expectations and encourages proactive planning when adopting the tool.

Advanced Topics Covered in the User Guide

For power users, the guide explores advanced functionalities such as:

- **Custom Scripting:** Leveraging TCL scripts to customize workflows and automate repetitive tasks.
- **Integration with Continuous Integration/Continuous Deployment (CI/CD):** Embedding equivalence checking into automated build pipelines.
- **Formal Property Verification:** Combining equivalence checking with property checking to enhance design correctness assurance.

These sections demonstrate how the Cadence Conformal LEC user guide is not merely an instruction manual but a strategic document for elevating verification methodologies.

Exploring the Cadence Conformal LEC user guide reveals it as a comprehensive companion for mastering logic equivalence checking. Its depth and clarity empower engineers to confidently apply formal verification techniques, ultimately supporting the creation of reliable and high-quality semiconductor products.

Cadence Conformal Lec User Guide

cadence conformal lec user guide: *The Functional Verification of Electronic Systems* Brian Bailey, 2005-01-30 Addressing the need for full and accurate functional information during the design process, this guide offers a comprehensive overview of functional verification from the points of view of leading experts at work in the electronic-design industry.

cadence conformal lec user guide: A Practical Approach to VLSI System on Chip (SoC) Design Veena S. Chakravarthi, 2019-09-25 This book provides a comprehensive overview of the VLSI design process. It covers end-to-end system on chip (SoC) design, including design methodology, the design environment, tools, choice of design components, handoff procedures, and design infrastructure needs. The book also offers critical guidance on the latest UPF-based low power design flow issues for deep submicron SOC designs, which will prepare readers for the challenges of working at the nanotechnology scale. This practical guide will provide engineers who aspire to be VLSI designers with the techniques and tools of the trade, and will also be a valuable professional reference for those already working in VLSI design and verification with a focus on complex SoC designs. A comprehensive practical guide for VLSI designers; Covers end-to-end VLSI SoC design flow; Includes source code, case studies, and application examples.

cadence conformal lec user guide: *CPU Design* Chandra Thimmannagari, 2005-12-02 I am honored to write the foreword for Chandra Thimmannagari’s book on CPU design. Chandra’s book provides a practical overview of Microprocessor and high end ASIC design as practiced today. It is a valuable addition to the literature on CPU design, and is made possible by Chandra’s unique combination of extensive hands-on CPU design experience at companies such as AMD and Sun Microsystems and a passion for writing. Technical books related to CPU design are almost always written by researchers in academia or industry and tend to pick one area, CPU architecture/Bus architecture/ CMOS design that is the area of expertise of the author, and present that in great detail. Suchbooks are of great value to students and practitioners in that area. However, engineers working on CPU design need to develop an understanding of areas outside their own to be effective. CPU design is a multi dimensional problem and one dimensional optimization is often counterproductive.

cadence conformal lec user guide: User's Manual for Program CONFORM (CONFORMal Contact Stress Problems). Burton Paul, 1978

Related to cadence conformal lec user guide

Cadence□□□Altium□□□□□□□□AD□□ - □□ 5.□□AD□Cadence□□□□□□□□□□□ □□AD□□□□□□□Cadence□□□□ □□□
60□□□CADENCE-□□□□□□□□□□□□□□□□□□□□□ □□□□□□20□□□□□□□□□□□□□□□□□□□□□□□□□□3□□□□□□□□□□□ □Cadence□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□
□□□□Cadence□ - □□ □□□Cadence□□□□□□□□Cadence□□□ □□□□□□□□□□□□□□□□□□□□□□□

□□□**Cadence IC**□□□□**PDF**□□□□□□□ **-** □□ Cadence SKILL□Cadence□□□□□□Cadence EDA□□□□□□□□□□□□□□□□□Cadence EDA□□□□□□□□□□□□□□□ □□□□□SKILL□□□□□□□□□□□□□□□□□□□

□**Cadence**□□□□□□□□□□□□□□□□□**?** **-** □□ □□□□□□□□□□□□□□□□□□□ Cadence □□ □□□□□□□□□□□□□□□□□ 1. □□□□□□□□□□□□□□□□SAD ADC□□□□□□□□□□□□□□□□□ □□□□□ □□□□□□□□□□

Cadence□□□□□□□□□□□□□□□□□□□□□□□□□□□□ **-** □□ Cadence□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□cadence□□ □□□□□

cadence virtuoso □□□□□□□□□□□□□□□□□□□□□□□□□ hiSetFont□□□hiSetFont□□export CDS_2DFORM_FONT_SCALING=1□□□□□□□□□□ □

Cadence17.4DRC□□□□□□□□**DRC**□□□□□□□□□□ **-** □□ Cadence Allegro 17.4□□□DRC□□□□□□□□□□□□□□□□□□□□bug□□□□□□□□□□□□□□DRC□□□□□□□□□□□□□□□□

Cadence□**Altium Designer**□□□□□□□ **-** □□ cadence □□□□□□□□□□□ 3D □□ ad □□□□□ □□□□□□□□□□□□□□□□□□□□ AD□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□cadence □□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□ **-** □□ □□cadence□□□□□ 2□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□

Cadence□□□**Altium**□□□□□□□□□□**AD**□□ **-** □□ 5.□□AD□Cadence□□□□□□□□□□□□□ □□AD□□□□□□□□□Cadence□□□

60□□□**CADENCE-**□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□20□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□3□□□□□□□□□□□□□Cadence□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□

□□□□**Cadence**□ **-** □□ □□□Cadence□□□□□□□□□Cadence□□

□□□**Cadence IC**□□□□**PDF**□□□□□□□ **-** □□ Cadence SKILL□Cadence□□□□□□Cadence EDA□□□□□□□□□□□□□□□□□Cadence EDA□□□□□□□□□□□□□□□ □□□□□SKILL□□□□□□□□□□□□□□□□□□□

□**Cadence**□□□□□□□□□□□□□□□□□**?** **-** □□ □□□□□□□□□□□□□□□□□□□ Cadence □□ □□□□□□□□□□□□□□□□□ 1. □□□□□□□□□□□□□□□□SAD ADC□□□□□□□□□□□□□□□□ □□□□□ □□□□□□□□□

Cadence□□□□□□□□□□□□□□□□□□□□□□□□□□□□ **-** □□ Cadence□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□cadence□□ □□□□

cadence virtuoso □□□□□□□□□□□□□□□□□□□□□□□□□ hiSetFont□□□hiSetFont□□export CDS_2DFORM_FONT_SCALING=1□□□□□□□□□□ □

Cadence17.4DRC□□□□□□□□**DRC**□□□□□□□□□□ **-** □□ Cadence Allegro 17.4□□□DRC□□□□□□□□□□□□□□□□□□□□bug□□□□□□□□□□□□□□DRC□□□□□□□□□□□□□□□□

Cadence□**Altium Designer**□□□□□□□ **-** □□ cadence □□□□□□□□□□□ 3D □□ ad □□□□□ □□□□□□□□□□□□□□□□□□□□ AD□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□cadence □□

□□□□□□□□□□□□□□□□□□□□□□□□□□□ **-** □□ □□cadence□□□□□ 2□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□

Cadence□□□**Altium**□□□□□□□□□□**AD**□□ **-** □□ 5.□□AD□Cadence□□□□□□□□□□□□□ □□AD□□□□□□□□□Cadence□□□

60□□□**CADENCE-**□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□20□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□3□□□□□□□□□□□□□Cadence□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□

□□□□**Cadence**□ **-** □□ □□□Cadence□□□□□□□□□Cadence□□

□□□**Cadence IC**□□□□**PDF**□□□□□□□ **-** □□ Cadence SKILL□Cadence□□□□□□Cadence EDA□□□□□□□□□□□□□□□□□Cadence EDA□□□□□□□□□□□□□□□ □□□□□SKILL□□□□□□□□□□□□□□□□□□□

□**Cadence**□□□□□□□□□□□□□□□□□**?** **-** □□ □□□□□□□□□□□□□□□□□□□ Cadence □□ □□□□□□□□□□□□□□□□□ 1. □□□□□□□□□□□□□□□□SAD ADC□□□□□□□□□□□□□□□□ □□□□□ □□□□□□□□□

Cadence□□□□□□□□□□□□□□□□□□□□□□□□□□□□ **-** □□ Cadence□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□cadence□□ □□□□

cadence virtuoso □□□□□□□□□□□□□□□□□□□□□□□□□ hiSetFont□□□hiSetFont□□export CDS_2DFORM_FONT_SCALING=1□□□□□□□□□□ □

Cadence17.4DRC□□□□□□□□**DRC**□□□□□□□□□□ **-** □□ Cadence Allegro 17.4□□□DRC□□□□□□□□□□□□□□□□□□□□bug□□□□□□□□□□□□□□DRC□□□□□□□□□□□□□□□□

Cadence□**Altium Designer**□□□□□□□ **-** □□ cadence □□□□□□□□□□□ 3D □□ ad □□□□□ □□□□□□□□□□□□□□□□□□□□ AD□

□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□cadence □□

□□□□□□□□□□□□□□□□□□□□□□□□ - □□ □□□cadence□□□□□ 2□□□□□□□□□□□□□□□□□□□□□□□□□□ □□

Cadence□□□Altium□□□□□□□□□□AD□□ - □□ 5.□□AD□Cadence□□□□□□□□□□□□ □□AD□□□□□□□□□Cadence□□□

60□□□CADENCE-□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□20□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□3□□□□□□□□□□□□□Cadence□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□

□□□□Cadence□ - □□ □□□Cadence□□□□□□□□Cadence□□□

□□□Cadence IC□□□□PDF□□□□□□□ - □□ Cadence SKILL□Cadence□□□□□□Cadence EDA□□□□□□□□□□□□□□□□□Cadence EDA□□□□□□□□□□□□□□□ □□□□□SKILL□□□□□□□□□□□□□□□□□□□□

□Cadence□□□□□□□□□□□□□□□□□? - □□ □□□□□□□□□□□□□□□□□□□□ Cadence □□ □□□□□□□□□□□□□□□□□□ 1. □□□□□□□□□□□□□□□□□SAD ADC□□□□□□□□□□□□□□□□□ □□□□ □□□□□□□□□□□

Cadence□□□□□□□□□□□□□□□□□□□□□□□□□□□ - □□ Cadence□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□cadence□□□ □□□□□

cadence virtuoso □□□□□□□□□□□□□□□□□□□□□□□□□ hiSetFont□□hiSetFont□□export CDS_2DFORM_FONT_SCALING=1□□□□□□□□□□ □

Cadence17.4DRC□□□□□□□□DRC□□□□□□□□□ - □□ Cadence Allegro 17.4□□□DRC□□□□□□□□□□□□□□□□□□□□bug□□□□□□□□□□□□□□DRC□□□□□□□□□□□□□□□□□

Cadence□Altium Designer□□□□□□□□ - □□ cadence □□□□□□□□□□□ 3D □□ ad □□□□ □□□□□□□□□□□□□□□□□□□□ AD□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□cadence □□□

□□□□□□□□□□□□□□□□□□□□□□□□ - □□ □□□cadence□□□□□ 2□□□□□□□□□□□□□□□□□□□□□□□□□□ □□

Cadence□□□Altium□□□□□□□□□□AD□□ - □□ 5.□□AD□Cadence□□□□□□□□□□□□ □□AD□□□□□□□□□Cadence□□

60□□□CADENCE-□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□20□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□3□□□□□□□□□□□□□Cadence□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□

□□□□Cadence□ - □□ □□□Cadence□□□□□□□□Cadence□□□

□□□Cadence IC□□□□PDF□□□□□□□ - □□ Cadence SKILL□Cadence□□□□□□Cadence EDA□□□□□□□□□□□□□□□□□Cadence EDA□□□□□□□□□□□□□□□ □□□□□SKILL□□□□□□□□□□□□□□□□□□□□

□Cadence□□□□□□□□□□□□□□□□□? - □□ □□□□□□□□□□□□□□□□□□□□ Cadence □□ □□□□□□□□□□□□□□□□□□ 1. □□□□□□□□□□□□□□□□□SAD ADC□□□□□□□□□□□□□□□□□ □□□□ □□□□□□□□□□

Cadence□□□□□□□□□□□□□□□□□□□□□□□□□□□ - □□ Cadence□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□cadence□□ □□□□

cadence virtuoso □□□□□□□□□□□□□□□□□□□□□□□□□ hiSetFont□□hiSetFont□□export CDS_2DFORM_FONT_SCALING=1□□□□□□□□□□ □

Cadence17.4DRC□□□□□□□□DRC□□□□□□□□□ - □□ Cadence Allegro 17.4□□□DRC□□□□□□□□□□□□□□□□□□□□bug□□□□□□□□□□□□□□DRC□□□□□□□□□□□□□□□□□

Cadence□Altium Designer□□□□□□□□ - □□ cadence □□□□□□□□□□□ 3D □□ ad □□□□ □□□□□□□□□□□□□□□□□□□□ AD□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□cadence □□

□□□□□□□□□□□□□□□□□□□□□□□□ - □□ □□□cadence□□□□□ 2□□□□□□□□□□□□□□□□□□□□□□□□□□ □□

Related to cadence conformal lec user guide

Cadence Introduces the Conformal Smart Logic Equivalence Checker (Design-Reuse8y) SAN JOSE, Calif., Sept. 13, 2017 – Cadence Design Systems, Inc. (NASDAQ: CDNS) today announced the Cadence® Conformal® Smart Logic Equivalence Checker (LEC), the next-generation equivalence checking

Cadence Introduces the Conformal Smart Logic Equivalence Checker (Design-Reuse8y) SAN

JOSE, Calif., Sept. 13, 2017 – Cadence Design Systems, Inc. (NASDAQ: CDNS) today announced the Cadence® Conformal® Smart Logic Equivalence Checker (LEC), the next-generation equivalence checking