cadence®

C-to-Silicon Compiler High-Level Synthesis

Automated high-level synthesis for design and verification

Cadence® C-to-Silicon Compiler high-level synthesis enables hardware design and verification to move up to the transaction level of abstraction, delivering quantum leaps in productivity and project turnaround time. Its analysis and optimization works in conjunction with embedded production logic synthesis, delivering performance, power, and area (PPA) results that meet or beat those of handwritten register-transfer logic (RTL) for any mixture of datapath and control logic design. By automating the path from transaction-level SystemC to RTL, C-to-Silicon Compiler enables the main focus of functional verification to shift to a higher level of abstraction, greatly speeding the overall verification cycle.

System-Level Design and Verification Challenge

Today's system-on-chip (SoC) market is extremely competitive. In order for chip design projects to succeed, designers must create and verify differentiated hardware more quickly than their competitors. Unfortunately, today's design flows begin by manually writing an RTL description that functions the same as the C/C++ specification model written by the architects. The RTL description also codifies the micro-architecture that will be implemented, largely determining the final PPA with minimal ability to explore more optimal alternatives. This inflexibility has negatively affected the risk/reward tradeoff of developing new intellectual property (IP), instead pushing companies toward outsourcing or re-using more and more of their SoCs in order to meet schedule demands, reducing their ability to differentiate.

The history of chip design has seen a progression of leaps in design and verification abstraction that deliver a corresponding leap in productivity. Starting with the transistor level, moving up to gate level, and then to RTL, hardware design productivity has kept pace with the advances in silicon capacity (see Figure 1). But productivity has lagged recently because the shift from RTL has taken longer than expected due to the lack of production-worthy high-level synthesis.

The Solution: High-Level Synthesis

In order to make the next productivity leap, high-level synthesis must deliver an automated path for the entire design from transaction-level models (TLMs) to RTL while delivering PPA that is at least as good as what is achieved with handwritten RTL. C-to-Silicon Compiler is the first high-level synthesis product to synthesize datapath and control logic together, described in an industry-standard language, to generate PPA that meets or beats that of handwritten RTL.

C-to-Silicon Compiler reads in a SystemC description of the hardware architecture and generates a Verilog RTL micro-architecture utilizing the

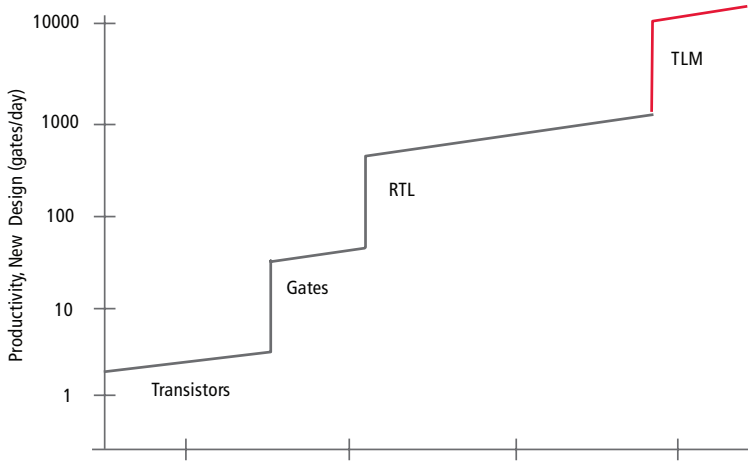

Figure 1: Raising design abstraction levels increases productivity

high-level constraints that are unique to the target product requirements and process library. These constraints are fine-tuned based on visual feedback from a rich graphical design environment and an incremental database. Because the implementation constraints are kept separate from the design's functionality, the same verified SystemC model is easily re-targeted for different end products with different requirements and process libraries (see Figure 2).

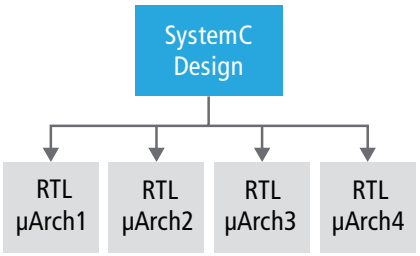

Figure 2: Generate different micro-architectures for different end-product needs from the same verified SystemC design.

Cadence has developed a full methodology to extend design and verification to SystemC TLM. By starting the metric-driven verification methodology with TLM, most bugs are eliminated before RTL is even created, greatly speeding the overall turnaround time of what is typically the critical path of a design project. Because the production synthesis guides generation of the RTL and synthesis constraints, the path into existing implementation flows is smooth. And if bugs need to be fixed or features added later during implementation, the automated engineering change order (ECO) capability applies a small patch that is optimized and formally verified so the project can stay on schedule.

C-to-Silicon Compiler Features

Micro-architecture exploration

- SystemC describes only the functionality, while C-to-Silicon implements the micro-architecture, enabling full exploration of the PPA solution space before committing to an RTL micro-architecture
- Graphical environment delivers source-connected visual analysis of such items as the control and data flow, critical timing path, area utilization, and power consumption
- Identify micro-architectures that have a greater power reduction than is possible in RTL synthesis, while meeting performance goals

Production quality of results for ASIC or FPGA

- Industry-leading scheduling and optimization, including resource sharing, various pipelining approaches, speed grade control, and carry-save adder (CSA) optimization.
- Cadence RTL Compiler synthesis is embedded under-the-hood for ASIC flows to accurately characterize resources for analysis and optimization
- Utilizes Liberty libraries for ASIC flows, along with wireload models or RTL Compiler physical layout estimation (PLE)
- Supports Xilinx and Altera FPGA devices and synthesis flows

ECO from TLM to GDSII

- Incremental synthesis minimizes changes to RTL while meeting constraints when applying an ECO patch
- Tight links to Cadence Encounter® Conformal ECO applies patches downstream to anywhere in the implementation flow, from netlist to placement with routing to metal-only ECO

Successful on all types of hardware

- Synthesizes datapath and control logic together, supporting all type of designs in the mixed datapath-control spectrum
- Has been used in production on projects spanning the datapath-control spectrum. Examples include H.265 video codec, high-speed image processor, 3G equalizer, 100Gbps optical networking, AES encryption, L2 cache control, and low-density parity checker

Built to enhance existing methodologies

- Supports IEEE SystemC and OSCI TLM with a single-source model used as a virtual prototype in simulation and then taken through high-level synthesis
- Generates RTL and constraints compatible with Cadence and third-party RTL synthesis and logic equivalence checking
- Methodology available to extend today's RTL-driven metric-driven verification environments to TLM-driven

Better PPA for ASIC or FPGA

High-level synthesis takes in untimed algorithms and implements micro-architectures and pipelines with the freedom to distribute and share logic across registers in order to satisfy constraints. Thus it is able to choose from a much broader and more powerful array of implementation options in order to deliver better PPA than is delivered by manually constructing designs with RTL.

C-to-Silicon Compiler embeds production logic synthesis technology in order to provide accurate analysis and optimization guidance. It characterizes the timing, power, and area of logic components in the context in which they are used in the design in order to more accurately model wire timing effects.

For ASIC flows, C-to-Silicon Compiler utilizes RTL Compiler synthesis along with the production Liberty library (see Figure 3). For FPGA flows, it utilizes either

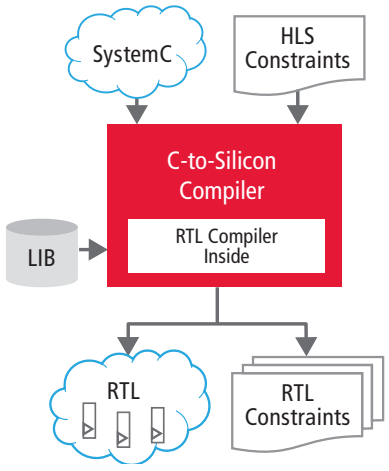

Figure 3: C-to-Silicon ASIC flow

Xilinx or Altera synthesis and device information. Utilizing production RTL synthesis delivers more accurate characterization, allowing for more aggressive optimization while generating RTL that will predictably deliver the desired quality of results after synthesis.

Graphical Exploration and Control

Successfully transforming an algorithm into production-quality hardware requires designers to make micro-architecture decisions in order to achieve the desired PPA balance. C-to-Silicon Compiler's graphical environment is a full cockpit for running synthesis, analyzing results, and making necessary adjustments.

After the SystemC design is compiled, the structure and flow of the design is displayed in the control-dataflow graph. This high-level visualization of the loops, branches, operations, and timing elements is cross-linked to the SystemC source to help the designer better understand the flow and interdependencies of the design. And it is updated after each step of the synthesis process (see Figure 4).

Once the design has been scheduled and resources have been allocated, all aspects of PPA are examined. The critical path viewer displays the timing-critical paths annotated with timing results from RTL synthesis. The area and power tree map viewers quickly highlight the blocks that consume the largest area or power as measured in RTL synthesis. Adjustments are made directly in the GUI and applied to the next run. Finally, the generated RTL structure can also be viewed in schematic form to visualize the resulting micro-architecture.

Apply ECO Anywhere from TLM-GDSII

Changes to the design can become necessary at any point during the design flow. Rather than re-running high-level synthesis followed by completely re-doing the entire RTL-GDSII implementation for the design, an ECO patch can minimize the impact of the change on the design and thus the schedule.

Most high-level synthesis tools are far removed from the RTL-GDSII implementation flow. However, because C-to-Silicon Compiler embeds production synthesis, it can combine this capability with its incremental database to generate a patched RTL design that will meet timing constraints downstream.

This capability, working in conjunction with Cadence Conformal ECO Designer, enables the patch to then be applied downstream on the netlist, the placed design, the routed design, or the post-mask netlist if a metal-only ECO is possible. Conformal ECO combines Cadence Conformal Equivalence Checker with RTL Compiler's synthesis optimization to ensure the patch is correct and that the design still meets its quality of results goals.

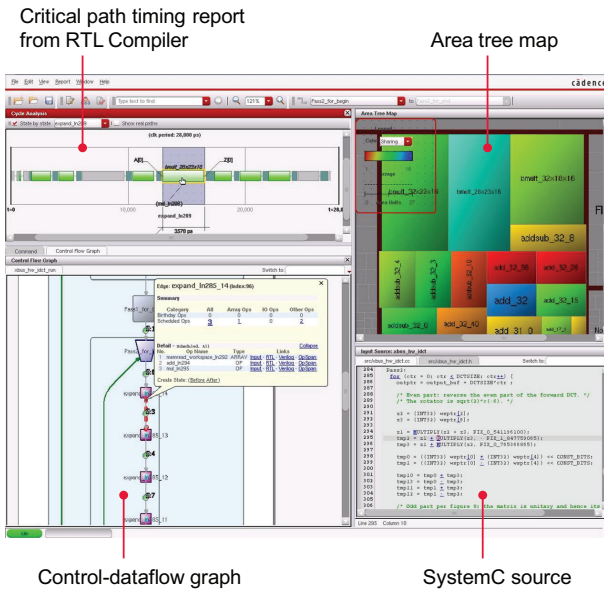

Figure 4: Complete graphical analysis, fully cross-linked.

Faster Time to Verified RTL

Functional verification is the schedule bottleneck in most design projects. Designing at a higher level of abstraction requires fewer details, which means fewer opportunities for designers to introduce bugs, faster simulation runtimes, and speedier debug of errors. However, if TLM verification is simply added onto an existing verification methodology, it becomes an extra step that does not help.

Cadence has extended its metric-driven verification methodology to take advantage of starting verification at a higher level of abstraction. Most features—specifically the core functionality and the protocol timing—are verified as SystemC TLM. Because C-to-Silicon Compiler offers an automated path to RTL, these features can just be regressed once in RTL, where verification can then focus only on newly added features introduced with signal-level interfaces and register timing.

The result is an overall reduction in verification turnaround time that has been proven to reduce the verification cycle by 35-50% (see Figure 5).

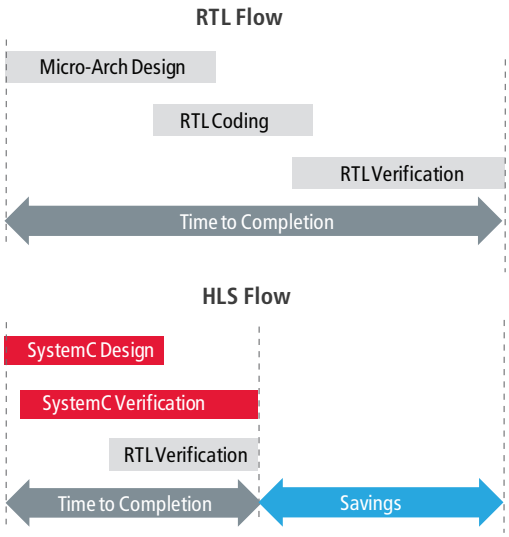

Figure 5: Verification savings from SystemC design and verification

Specifications

C-to-Silicon Compiler L	
Design format and language support	• Input design language: IEEE SystemC and OSCI TLM 1.0 • High-level synthesis constraints and scripting: Tcl • Output design language: IEEE 1364 Verilog • Output constraints: Synopsys Design Constraints
Graphical user interface	Full cross-linked graphical design environment with new design wizard, control-dataflow graph, critical path viewer, area and power tree maps, resource viewer, pipeline viewer, source code viewer, RTL schematic viewer
Embedded logic synthesis	• ASIC: Cadence RTL Compiler • FPGA: Xilinx or Altera
Analysis	Built-in timing, power, and area analysis utilizing context-aware characterization from embedded logic synthesis
Database	Fully incremental design database that stores behavior, structure, and timing information
ECO synthesis	Incremental synthesis that uses similarity as the primary cost function
Clock gating	Fine-grained and coarse-grained clock gating
Memory support	Flatten array, built-in RAM, prototype memory, vendor RAM, external memory
Ramp-up support	• Self-paced tutorials • Extensive design and script examples • Methodology services available
Design IP	• Math functions • Fixed-point data types • Floating-point data types • Flex channel block-to-block communication • AXI3 • AXI4-Lite
Testbench generation	Automatically generates SystemC wrappers to enable RTL verification with SystemC testbenches
Simulation model generation	Automatically generates I/O cycle-accurate simulation models, assertions, and scripts for simulation

Supported Workstations and Operating Systems*

- X86 instruction-set architecture workstations
- OS type: Linux
  - RHEL 5 (32-bit and 64-bit), RHEL 6 (32-bit and 64-bit)
  - SuSE 10 (32-bit and 64-bit), SuSE 11 (32-bit and 64-bit)

** Please check with your Cadence representative for the latest information and additional details as they are subject to change without notice.*

Cadence Services and Support

- Cadence application engineers can answer your technical questions by telephone, email, or Internet—they can also provide technical assistance and custom training
- Cadence certified instructors teach more than 70 courses and bring their real-world experience into the classroom
- More than 25 Internet Learning Series (iLS) online courses allow you the flexibility of training at your own computer via the Internet
- Cadence Online Support gives you 24x7 online access to a knowledgebase of the latest solutions, technical documentation, software downloads, and more

cadence®

Cadence is transforming the global electronics industry through a vision called EDA360. With an application-driven approach to design, our software, hardware, IP, and services help customers realize silicon, SoCs, and complete systems efficiently and profitably. **www.cadence.com**