

# Integrating Virtual Prototypes with IC Verification

By Steve Brown, Cadence Design Systems and Michel Genard, Wind River

Virtualized systems development (VSD) enables faster, more effective system-on-chip (SoC) development. Using VSD, design teams can draw upon libraries of fast processor and peripheral models to quickly build virtual prototypes for early software development. A Wind River and Cadence collaboration offers a unified VSD flow for hardware and software, which starts at a high level of abstraction and extends to implementation.

## Contents

Introduction.....	1
Collaboration Addresses Challenges .....	2
A Look at Simics .....	3
Verification for Hardware Design .....	4
A Look at Incisive Software Extensions .....	5
Integrating Simics and Incisive Software Extensions.....	6
A Look at the Palladium XP Verification Computing Platform .....	7
Connecting Simics to RTL Design and Verification.....	8
Co-Simulation with Simics and Palladium XP.....	8
Conclusion.....	9
References.....	9

## Introduction

Increasingly complex systems-on-chip (SoCs) are enabling sophisticated applications in mobile communications, consumer electronics, medical devices, networking, automotive, and many other markets. But SoC development costs are rapidly rising with complexity and capacity, and are expected to reach \$100 million by the 32nm process node. Studies show that much, if not most, of that cost will be embedded software development. Meanwhile, semiconductor companies are increasingly expected to provide some or all of the embedded software stack with their silicon.

The need for faster and better SoC and software development has led to a methodology called virtualized systems development (VSD). This approach lets design teams quickly build “virtual platforms” or “virtual prototypes” comprising software models of system hardware, drawing upon libraries of fast processor and peripheral models. By running simulations using these models, systems architects and software developers can then perform hardware/software partitioning, develop and debug hardware-dependent code, and start applications software development before any system hardware is actually built.

Virtual prototypes are an important part of the System Realization process described in the recent EDA360 vision paper<sup>1</sup>. System Realization represents the integration of complete hardware/software platforms ready for applications development and deployment. EDA360 proposes an application-driven methodology in which the end applications are envisioned first, and customized hardware and embedded software are then developed to support the applications.

While virtual prototypes are a critical enabler of application-driven development, they are expensive to build, and this expense has limited their deployment. One reason virtual prototypes are costly is that they have remained largely disconnected from the downstream design and verification

flow. High-level models created for virtual prototypes are typically not used for downstream implementation, forcing hardware developers to manually rewrite models at the register-transfer level (RTL) for functional verification and synthesis.

Verification often doesn't begin until the register-transfer level, and is disconnected from the virtual prototype environment. Inconsistencies between virtual prototyping and RTL verification result in extra work, more bugs, and numerous iterations between hardware and software design.

The virtual prototyping environment is not sufficient for all phases of software development and debugging. Virtual prototypes use highly abstract models with no notion of timing, and are not accurate enough to find all bugs in hardware/software interfaces. Nor are they accurate enough to run a detailed power analysis. RTL simulation provides much more accuracy, but is extremely slow. Many design teams are finding that emulation and acceleration, which run orders of magnitude faster than software-based simulation while maintaining the same accuracy, are essential tools for hardware/software integration and debugging.

Meanwhile, software verification has traditionally been an ad-hoc process using debuggers and "printf" statements. Software developers have not had the advantages that are available today in hardware verification environments, including executable verification plans, coverage metrics, constrained-random stimulus generation, and formal property checking.

### Collaboration Addresses Challenges

Collaboration between Cadence and Wind River addresses these problems by linking virtual prototypes to the RTL verification environment. The collaboration began in 2009 with Cadence and Virtutech, which was then an independent virtual prototype provider. That collaboration integrated Cadence® Incisive® Software Extensions with the Virtutech Simics virtual prototype environment. Incisive Software Extensions brings techniques such as verification planning, constrained-random stimulus, and coverage metrics to hardware/software co-verification.

The collaboration was extended in April 2010 after Intel purchased both Wind River and Virtutech, and Wind River integrated Simics into its product line. Cadence, meanwhile, introduced Palladium® XP, a verification computing platform that unites simulation, acceleration, and emulation. The new partnership between Cadence and Wind River provides direct links between the Simics environment and Palladium XP.

This collaboration allows the synthesis of Simics models into RTL code using Cadence C-to-Silicon Compiler, providing a high productivity, direct path to hardware implementation and verification. The RTL models can then be converted to gate-level representations using RTL synthesis. As such, the same models used for virtual prototypes can become the starting point for IC implementation.

The RTL models can also be brought into the Palladium XP environment and then co-verified with Simics models in a transaction-based acceleration environment (Figure 1). In this combined environment, some of the design components will run in the Simics environment while cycle-accurate, RTL hardware models reside in Palladium XP. This allows the combination of a functional virtual prototype and an accurate hardware representation at speeds that are orders of magnitude faster than software-based simulations.

With this new collaboration, software applications can be developed first and drive the downstream embedded software and hardware development processes. The result is a significant step toward application-driven System Realization as described in the EDA360 vision paper. With the capabilities provided by the Wind River/Cadence partnership, virtual prototypes no longer need to be isolated from the rest of the design flow.

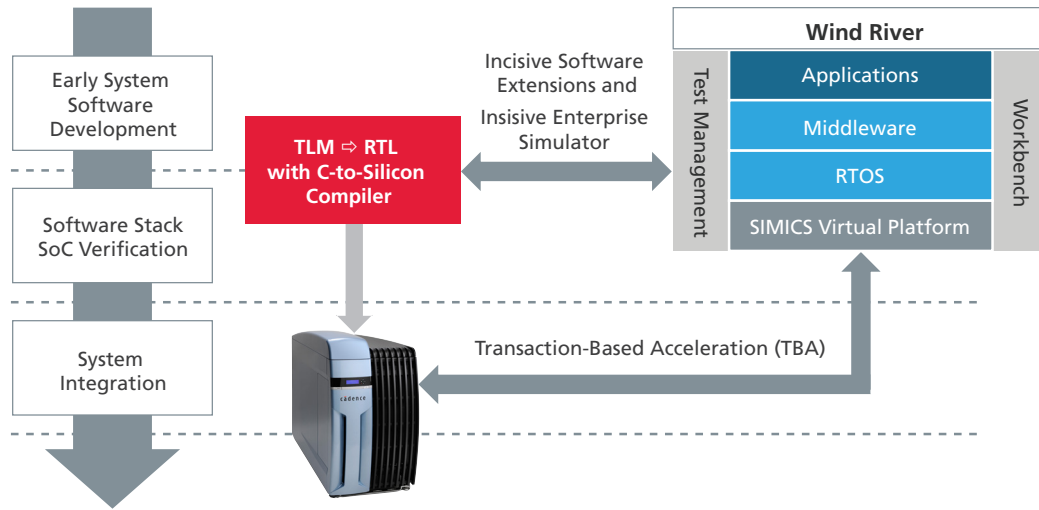


Figure 1: The partnership between Wind River and Cadence links the Simics virtual prototyping environment with the RTL environment and the Palladium XP verification computing platform

This whitepaper includes brief overviews of Simics, Incisive Software Extensions, and Palladium XP, and shows how these environments can be linked together to facilitate hardware/software integration.

## A Look at Simics

Wind River Simics is a flexible and scalable software solution that models electronic systems with high performance and fidelity. By allowing early software development and debugging, it speeds time to market. Simics also enables higher product quality and lowers project risks.

With Simics, design teams can create a complete transaction-level model (TLM) virtual prototype by leveraging an extensive library of target devices, operating systems, CPU architectures, and SoC families. Once the prototype is created, developers can run the same unmodified binaries that would run on the actual hardware, including full applications software, in a simulation environment that runs at hundreds of MIPS. Simics provides deterministic execution and offers features such as non-intrusive profiling and fault injection.

The Simics family consists of the following components:

- **Simics Model Builder** – provides the development platform used to create new device models
- **Simics Virtual Platform** – provides a model of the physical target hardware that is being simulated
- **Simics Accelerator** – allows acceleration on multi-processor, multi-core hosts
- **Simics Ethernet Networking** – provides connectivity between virtual systems inside a simulation
- **Simics Hindsight** – lets developers run and debug code, provides visibility into registers, and supports reverse execution that runs code backward to find the source of a defect
- **Simics Analyzer** – provides the ability to do analysis on the complete system including statement code coverage and execution analysis
- **Simics Extension Builder** – allows developers to build customized extensions on Simics to integrate it with their particular workflow

It's important to note that Simics technology models complete systems, not just SoCs or boards. Simics can also model multi-core and multi-processor systems, and greatly ease programming and debugging for parallel hardware. Although multi-core systems are inherently non-deterministic, Simics can re-introduce control and repeatability in the debugging process. This makes debugging a multi-processor as easy as debugging a single program on a single processor.

A virtual prototyping environment such as Simics enables many capabilities that would be impossible in actual hardware, including control over time, inspection of all hardware and software states in the system, the ability to span multiple software domains, and visibility into hardware/software boundaries.

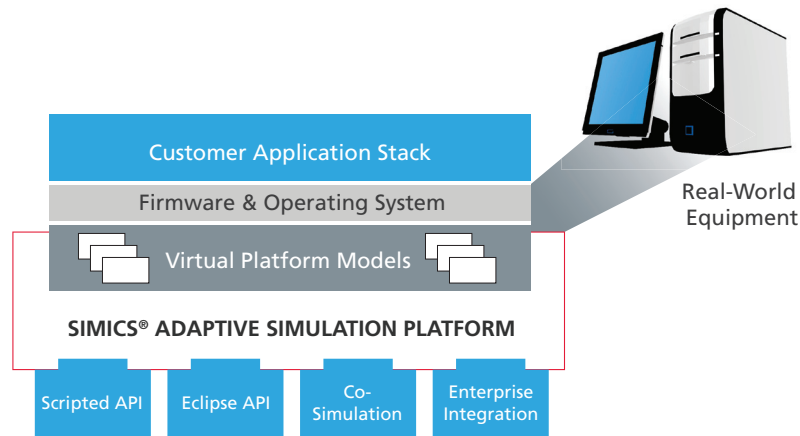


Figure 2: Simics provides a simulation environment based on virtual prototypes

## Verification for Hardware Design

While software verification is largely an ad-hoc process, great strides have been taken toward building a rigorous verification methodology for hardware design. Functional verification, which begins as soon as a hardware specification is produced, employs a variety of powerful tools and techniques. For example, IC functional verification tools include simulation, hardware emulation and acceleration, assertion-based verification, equivalence checking, and formal (static) property checking.

In recent years, functional verification has moved from a reliance on directed tests to constrained-random stimulus, which can uncover unexpected bugs that directed tests are unlikely to find. Furthermore, engineers increasingly rely on coverage metrics to determine when the verification task is completed. These metrics include code coverage, which measures whether lines of code have been executed, as well as functional coverage, which defines scenarios that should be tested.

Recently, a methodology called metric-driven verification (MDV) has been developed to make the IC verification process more effective and predictable. MDV starts with a verification plan that has measurable goals based on coverage metrics. As verification proceeds, metrics are captured and used to guide the verification process.

A recent article<sup>2</sup> identified five steps of MDV:

- Analyze the device specifications
- Scope the verification objectives
- Identify the feature set of the design
- Design detailed coverage metrics
- Select aggregate metrics to track progress

These steps can be applied to hardware/software co-verification and debugging as well.

## A Look at Incisive Software Extensions

The Cadence Incisive Software Extensions product brings the constrained-random stimulus generation and MDV capabilities of the Incisive product family to the embedded software portions of a design. It runs with Incisive Enterprise Simulator, which is a multi-language, multi-level functional verification solution, or with the Palladium XP Verification Computing Platform. It can also be used with Incisive Enterprise Manager to provide verification planning and management.

Initially aimed at hardware/software co-verification, Incisive Software Extensions gives the verification testbench access to software executing on processor models. Using the Incisive GUI, the verification team can apply MDV to system-level behavior including software processes, function calls, and variables, along with processor and register behavior and interaction with hardware events such as interrupts.

Incisive Software Extensions works with processor models in any form. It can run with RTL models, TLM models, Palladium XP, host code execution, or silicon. It allows fast processor models to co-simulate with fast TLM 2.0 models (Figure 3). Incisive Software Extensions provides post-process software debug with no need to re-run software, and lets users trace backwards and forwards. If a system-level verification run fails, Incisive Software Extensions can show what was happening in the software when things went wrong.

When Incisive Software Extensions is used within a complete MDV solution, Incisive Enterprise Planner creates a verification plan containing the goals for functional verification closure. The plan can include software functions, input parameters, variables, and data structures. This plan is used to measure progress and guide verification teams in making the most effective decisions to reach closure with optimal resource utilization. Metrics are collected and used to guide the verification effort.

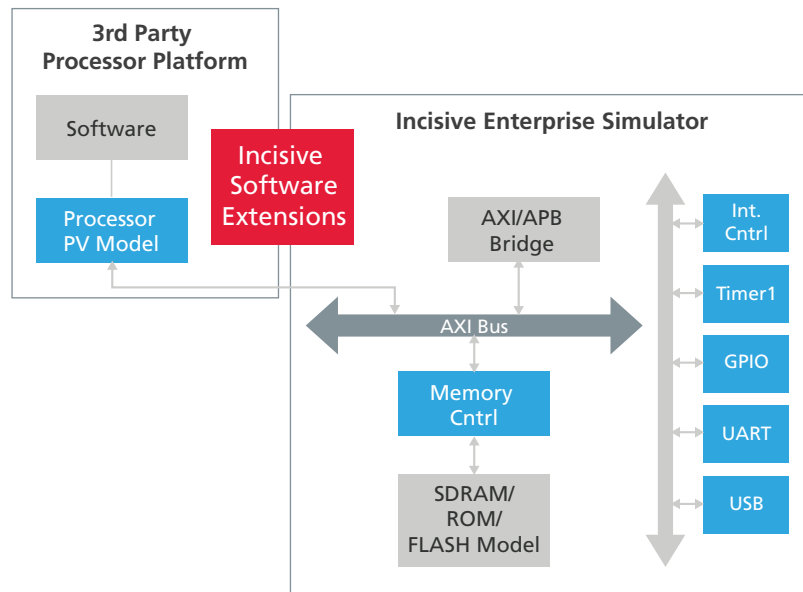


Figure 3: Incisive Software Extensions links the Incisive Enterprise Simulator with software running on processor models.

With Incisive Software Extensions, users can build software tests that automate stimulus generation. Incisive Software Extensions permits constrained-random variability in parameter values. Software tests can randomly vary sequence and timing within desired constraints, and can react to hardware states. Incisive Software Extensions can monitor hardware and software in tandem to check across domains and provide combined cross-functional coverage.

Incisive Software Extensions has an embedded software configuration that gives SimVision (the Incisive GUI) software debug features such as single-stepping, breakpoints, stepping forward and backward, signal tracing, and probing.

## Integrating Simics and Incisive Software Extensions

On a technical level, the integration of Simics and Incisive Software Extensions is straightforward. Incisive Software Extensions has been extended to read Simics function calls and to understand the specific needs and uses of virtual prototype environments. As a result of the integration, Incisive Software Extensions can access the virtual prototype non-intrusively. Prior to the integration, it would have been necessary to create an Incisive Software Extensions agent that would have consumed memory and network bandwidth.

Simics has not been modified in any way, nor is additional software required to take advantage of the integration. All that's needed is Simics and Incisive Software Extensions running with a simulation engine such as Palladium XP, Incisive Enterprise Simulator, Host code execution, or the silicon itself.

The integration brings verification planning and management, embedded software tracing, and MDV to Simics users (Figure 4). It provides stimulus generation and coverage measurement for software development and virtual platform verification.

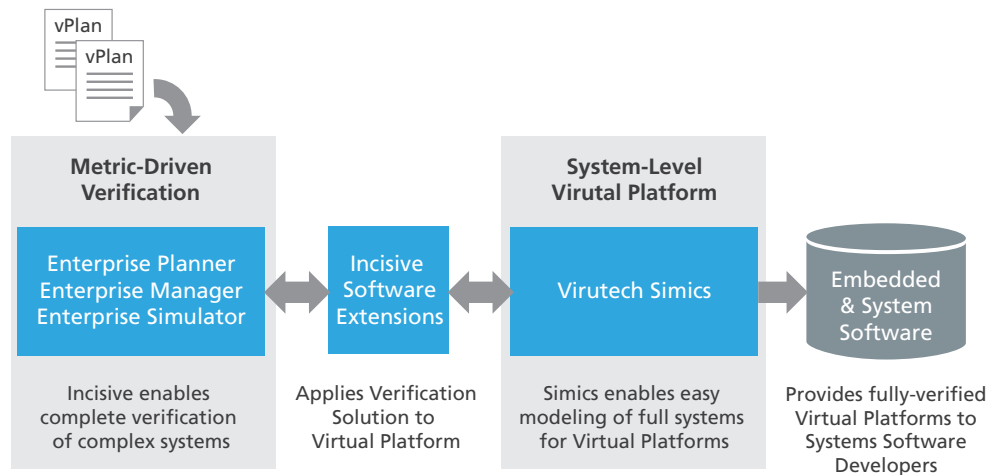


Figure 4: The integration of Incisive Software Extensions with Simics brings verification planning and metric-driven verification into the virtual platform environment

Simics and Incisive Software Extensions are highly complementary. While Simics provides strong features for non-intrusively isolating and identifying bugs, Incisive Software Extensions enable constrained-random testing, making it easy to provoke bugs. Users can work within the Incisive GUI to generate stimulus, track coverage, set breakpoints, and step through code. Meanwhile, Simics lets users snapshot, or checkpoint, any state of the virtual platform for storage and for later replay.

Who benefits from this integration? First of all, software developers using Simics can move beyond directed tests and employ constrained-random testing to find unexpected corner cases. They can develop a verification plan that determines the items that need to be exercised and monitored. And they can set measurable goals and track those goals through coverage metrics.

Secondly, hardware developers and systems architects can test the hardware/software interface, and can quickly identify whether errors are occurring in hardware or software. This boosts confidence that the system they have architected is functioning in accordance with system specifications.

Third, any verification plan or testbench developed with Incisive Software Extensions can provide an excellent starting point for RTL verification engineers. The verification testbench developed through Incisive Software Extensions can be used with the first TLM design with only minor enhancements; with further refinements, it can be used in the RTL design. In this sense, the virtual prototype environment flows easily into the chip design flow, beginning at the TLM level and extending down to RTL.

## A Look at the Palladium XP Verification Computing Platform

Cadence Palladium XP is a verification computing platform that unites simulation, emulation, and acceleration into a single environment (Figure 5). It unifies capabilities from Cadence Incisive Xtreme® accelerators and Incisive Palladium emulators, combining the best capabilities from each platform. A “hot-swap” technology lets users switch between simulation and acceleration in real time without re-compilation.

Palladium XP is integrated with Incisive Enterprise Manager and thus supports MDV flows. It also works with the Incisive Palladium Dynamic Power Analysis option. Palladium XP provides performance up to 4MHz, has capacity for up to 2 billion gates, and can accommodate 512 concurrent users.



Figure 5: Palladium XP combines simulation, emulation, and acceleration in a single verification computing platform

The difference between acceleration and emulation is subtle, but important. In brief, these technologies can be described as follows:

- In acceleration, the design is typically running on the hardware while a simulation testbench runs on the workstation. If the testbench is transaction-based, the result is transaction-based acceleration, or as some say, co-emulation
- In emulation, the entire design and verification environment is generally running on the hardware. In addition to the hardware in the emulation box, portions of the design or the testbench may also be running on external target hardware through in-circuit emulation

An accelerator speeds up a simulator and may provide most or all of the benefits of the simulator. Palladium XP, for example, can support various simulators, but is optimized to accelerate the Incisive Enterprise Simulator. As such, it supports such features as executable verification plans (vPlans), metric-driven verification (MDV), pseudo-random test generation, and coverage metrics in acceleration mode. Metrics can be collected and analyzed by Incisive Enterprise Manager.

Emulation provides a self-contained verification environment. It provides extremely fast, testbench-independent verification speeds—up to 4MHz with Palladium XP—and offers fast bring-up times. Emulation makes it possible to directly run software applications on target hardware connected via a JTAG port or SpeedBridge adapters. Emulation does not, however, provide the stimulus randomization capability of simulation and acceleration, and coverage metrics are limited to the design since the testbench is ported into the hardware.

During the hardware/software integration process, therefore, designers might start with a software virtual prototype in Simics and then find that they need more accuracy to debug hardware/software interactions or run a power analysis. They will first turn to acceleration mode in Palladium XP by moving some portions of the design into hardware while running the verification environment on a workstation. Then they will turn to emulation mode

by moving everything (including the testbench) into hardware, and perhaps run software applications on target hardware. Palladium XP also supports a hybrid (acceleration plus emulation) mode in which a design or testbench runs on the workstation while the emulator is connected to a target.

## Connecting Simics to RTL Design and Verification

Cadence C-to-Silicon Compiler automatically generates synthesizable RTL from untimed C/C++ or SystemC® transaction-level models. Users provide timing, area, and/or power constraints, and they can perform what-if analyses and generate different micro-architectures by changing these parameters. C-to-Silicon Compiler embeds Cadence Encounter® RTL Compiler under the hood and can synthesize both dataflow and control logic.

As part of the 2010 partnership, Cadence and Wind River are collaborating to convert the non-processor Simics models to synthesizable RTL using C-to-Silicon Compiler. Simics models are written in the Device Modeling Language (DML), which compiles to untimed C/C++. If care is taken to follow synthesizable coding guidelines, C-to-Silicon Compiler can read these input models. Once synthesized, the resulting RTL models can be used for hardware implementation through RTL synthesis, and can also be brought into Palladium XP for acceleration and emulation.

High-level synthesis will work well for Simics algorithmic models, but results are less predictable for Simics device and processor models. That's because these models are typically not created for implementation, and as such, may not be sufficiently optimized in synthesis for silicon implementation. One solution is to develop models that are more optimized for implementation; another is to reuse the legacy RTL that is often available for devices and processors.

Once C/C++ models are compiled, users can utilize the powerful C-to-Silicon Compiler GUI to manipulate loops, function calls, arrays, and latency constraints to explore possible micro-architectures. For maximum performance in acceleration, for instance, a user might tell C-to-Silicon Compiler to in-line all the function calls, add a state transition to the main loop, and flatten the arrays. Then the design is scheduled using the provided constraints, and a standard synthesizable RTL-Verilog model is generated. The RTL model can be connected to a testbench and compiled into Palladium XP.

## Co-Simulation with Simics and Palladium XP

As noted, transaction-based acceleration takes place when a testbench or some of the behavioral design components are running on a workstation and some or all of the hardware design is in the accelerator. Palladium XP uses the Accellera Standard Co-Emulation Modeling Interface3 (SCE-MI 2.0) to provide a transactional interface to Simics, using a C-based API. In this way, cycle-accurate RTL models residing on the Palladium XP platform can be brought into a transaction-based acceleration environment with Simics models, allowing hardware/software co-verification in the virtual prototyping environment.

The RTL interface function is brought into the Simics model device driver, and when the Simics model runs, Palladium XP intercepts any device driver function calls. It stimulates the RTL model and returns signal or transaction values to the Simics model. As a result, engineers can debug both the virtual prototype environment and the RTL model concurrently, using the C breakpoint capability in Simics and the RTL state breakpoint in Palladium XP. Hardware designers can then use a conventional waveform environment to complete their debugging.

There are various ways to use the co-simulation capability. Some components of the design could be placed in Palladium XP, while others remain in Simics, and the stimulus could come from embedded software directly connected to Simics. Stimulus could also come through a JTAG port connected to target hardware. Or, it could come from a verification testbench on the workstation in SystemVerilog or e languages. Regardless of the approach used, leaving more of the design in Simics will result in faster execution speeds, while porting more of the design to Palladium XP will allow much higher accuracy.

## Conclusion

The virtual prototypes used for early software development no longer need to be isolated from the rest of the design flow. Collaboration between Wind River and Cadence links the Simics virtual prototype environment with the downstream IC design and verification flow, facilitating hardware/software integration and enabling System Realization as described in the EDA360 vision paper.

In the first phase of this collaboration, announced in 2009, Cadence Incisive Software Extensions was integrated into the Simics environment. This brings hardware verification capabilities (such as random stimulus generation and metric-driven verification) to engineers performing hardware/software co-verification.

In the second phase, announced in 2010, Cadence and Wind River built a bridge from Simics to RTL with Cadence C-to-Silicon Compiler. Design teams can compile Simics models into synthesizable RTL, which can be used for implementation with RTL synthesis or verification with the Cadence Palladium XP verification computing platform. Moreover, transaction-based acceleration between Simics and Palladium XP brings those RTL models back into a single simulation environment.

A unified design and verification flow for hardware and software, starting at a very high level of abstraction and extending down to implementation, has thus become possible.

## References

1. "EDA360: The Way Forward for Electronic Design." Cadence Design Systems, <http://www.cadence.com/eda360>
2. "Project Management is All About Planning and Execution." Chip Design Magazine, <http://www.chipdesignmag.com/display.php?articleId=264>
3. Accellera interface technical committee home page, <http://www.accellera.org/activities/itc>



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](http://www.cadence.com)