Incisive Simulation Acceleration Deployment

April 2003
# TABLE OF CONTENTS

- **Cadence Incisive Verification Platform**
- **Application Note Overview**

1. Introduction to Simulation Acceleration (lock-step mode) ....................................................... 1
2. Incisive Acceleration .................................................................................................................. 2
3. Differences Between Incisive Simulation and Incisive Acceleration ........................................... 3
4. Methodology Flow ..................................................................................................................... 4
   4.1 Prepare for Simulation Acceleration ..................................................................................... 4
   4.2 Optimize Design and Testbench .......................................................................................... 5
   4.3 Compile Design for Acceleration ......................................................................................... 6
   4.4 Create Design Shell ............................................................................................................. 7
   4.5 Compile/Elaborate Testbench and Design Shell ................................................................. 7
   4.6 Run Acceleration and Debug ............................................................................................. 7
5. See Also .................................................................................................................................... 8
Verifying today’s complex ICs requires the speed and efficiency that can only be provided in a unified verification methodology. The Cadence Incisive verification platform enables the development of a unified methodology from system design to system design-in for all design domains. A unified verification methodology consists of many different tools, technologies and processes all working together in a common environment. The Incisive verification platform provides the tools, technologies, a common user environment and the support needed to develop a unified methodology. This application note details specific topics for using the tools and technologies in the Incisive platform to help create a unified methodology to verify your design.

APPLICATION NOTE OVERVIEW

PURPOSE
This application note describes the benefits of migrating from an NC-Sim or an Incisive software–only simulation environment to an environment that uses Cadence’s Incisive platform with hardware-based acceleration-on-demand to perform simulation acceleration. Incisive customers using lock-step acceleration typically achieve simulation speeds 5 times to 50 times faster than with software simulation alone. This document also describes the migration process.

AUDIENCE
This document is for traditional users of workstation-based HDL simulators who wish to greatly improve simulation performance through hardware acceleration with minimal changes to their design and testbench environment. In addition, users planning future environments for maximum verification performance will find this document useful. This document describes Cadence LDV 4.1 and above and Palladium 1.1.1 and above.

1 INTRODUCTION TO SIMULATION ACCELERATION (LOCK-STEP MODE)
In simulation acceleration (lock-step mode), a simulation engine runs a behavioral testbench (TB) on a workstation while a hardware accelerator runs a synthesizable design (DUV). The simulator and accelerator exchange information every clock phase, enabling acceleration of the DUV while still using existing HDL testbenches. In addition, this model allows the user to continue to use the same simulation interface for control and debug. So the user quickly gains higher simulation performance with minimal changes to the verification environment.

![Diagram of Incisive Simulation Acceleration Environment](image1)

*Figure 1. Incisive Simulation Acceleration Environment*
Lock-step acceleration provides benefits for customers running long tests with high DUV activity. The real benefit occurs when accelerating tests that take hours, days or even weeks in software simulation. In many cases, acceleration enables tests that would be impractical in software simulation because of time constraints.

2 INCISIVE ACCELERATION

A high performance acceleration environment must include a tightly coupled, high performance accelerator. As the acceleration engine for the Incisive platform, Cadence’s Palladium technology provides many key features critical to providing performance as well as reducing cost.

Incisive customers using lock-step acceleration typically achieve simulation speeds 5 times to 50 times faster than with software simulation alone. One customer observed that a set of regression tests taking over 3 days in a software simulator completed on the Incisive platform with lock step acceleration in less than 1.5 hours! Some customers have seen up to 200 times improvement when migrating from a third party simulator to the Incisive platform with acceleration.

The user can estimate the increase in performance from the percentage of simulation time spent on executing the testbench. For example, if a simulation spends 10% of the time executing the testbench, the user can expect an acceleration of up to 10 times when using lock-step acceleration. The actual speed increase depends on design and implementation considerations.

Here are the features that make the Incisive platform a high performance and low cost solution.

- Multi-user capability allows the verification team to simultaneously share the accelerator during every step of the verification process. The accelerator can support from 1 to 16 simultaneous users depending on the size of the system. With this feature, the verification team can verify different blocks at the same time or multiple copies of the same block to improve throughput. Once the top level is ready, the verification team can verify one or more copies of the full design as well.

- Multiple use models allow the verification team to get value at every stage. This document focuses on lock-step acceleration, but the same system can be used to verify with transaction-based testbenches, synthesizable testbenches, and in-circuit emulation.

- The acceleration-on-demand capability allows users to run simulation or acceleration and to switch dynamically between them. For example, a verification team can run simulation during the day or when many small blocks must be verified. With the same set of licenses, the verification team can run long regression tests overnight or begin the full verification of the entire system plus software when available.

- Support for mixed-level TBs & DUVs means that the user can partition unsupported behavioral constructs from the DUV to execute on the workstation. In addition, synthesizable constructs in the testbench can be moved into the accelerator for improved performance.

- The Incisive platform supports soft IP, synthesizable, behavioral HDL or C/C++ as well as TLM/SystemC models. The platform supports PLI-compliant C/C++ models.

- Support for hard IP either inside Cadence’s IP Chassis or on a custom target board allows the verification team to test its design against a real implementation, which might be an ARM processor, a TI DSP, or a custom design. This testing gives higher confidence in the correctness of the design, higher performance, and possibly lower cost over soft models.
• The Incisive accelerator can accommodate up to 64 Gbytes of design memory, allowing the user to simulate large on-chip or system memory in real hardware memory which provides higher performance than simulating memory on the software simulator. In addition, customers can purchase, from Cadence, synthesizable memory models for many of the common memory types.

• Fast compile time allows the user to build a new acceleration model shortly after making an RT-level change to a design. The Incisive accelerator compiles 4 million gates per hour from the RTL.

• The Full Vision feature allows the user to display any signal at runtime. Once the user turns on Full Vision, the value of every design signal becomes available for display.

• Acceleration policy checks help the user identify issues that might degrade performance or limit functionality in an acceleration environment.

3 DIFFERENCES BETWEEN INCISIVE SIMULATION AND INCISIVE ACCELERATION

In some cases, logic implemented in real hardware behaves differently than the same logic implemented in software. In these cases, the accelerator characteristics are similar to those of the real silicon, allowing the user to identify some of these differences before tape out. If not understood, these differences can consume time when migrating to acceleration. This section discusses these differences and describes how the Incisive platform helps minimize the time in dealing with the differences between hardware and software.

PARALLEL EXECUTION

Logic implemented in real hardware executes in parallel. On the other hand, software simulators generally execute design code sequentially, which can hide certain bugs. Just as in real silicon, a processor-based system such as the Incisive accelerator executes design code in parallel, which can expose bugs created by race conditions in the design. Exposing these race conditions helps the customer improve the design, but also might consume time during the migration to acceleration.

The Incisive platform contains acceleration policy checks to help the user identify this type of race condition.

TWO-STATE SIMULATION ENGINE

Real hardware does not have states such as ‘X’ or ‘Z’. In general, all values resolve to ‘0’ or ‘1’, which is also true in the Incisive accelerator. However, unlike real silicon the Incisive accelerator resolves ‘Z’ states on tri-state nodes to known, deterministic values. Also, the Incisive accelerator automatically initializes storage elements to a known, deterministic state. Therefore, the Incisive accelerator reproduces the same initial conditions after every download, which can aid significantly in debugging and bringing up a design.

SYNTHESIZABLE RTL

Again as with real silicon, the Incisive accelerator requires synthesizable HDL constructs for implementing the DUV. However, Incisive acceleration does support some non-synthesizable constructs such as system tasks (for example, $display) inside the DUV. Also, Cadence has solutions for implementing common design structures such as memories inside the accelerator (for example, SDRAMs). The balance of non-synthesizable constructs can be supported in the Incisive unified simulator and/or be made synthesizable.
4 METHODOLOGY FLOW

This section guides a new user through the steps to migrate to an acceleration environment.

Prepare for simulation acceleration

Optimize design and testbench

Compile design for accelerator

Create design shell

Compile/elaborate testbench and design shell

Run acceleration and debug

Figure 2. Migration Flowchart for Incisive Lock-step Acceleration

4.1 PREPARE FOR SIMULATION ACCELERATION

An initial analysis of the design and testbench helps identify the steps required to migrate from a software simulation environment to a simulation acceleration environment.

RUN ACCELERATION POLICY CHECKS

The Incisive acceleration policy checks provide the user with information for use later in the flow.

- **Identification of all the testbench clocks and their time periods.** This determines the frequency of synchronization necessary between the accelerator and Incisive engines.

- **Identification of Out-of-Module References (OOMRs) between the testbench and the design.** The user must identify OOMRs at design compile time to allow access to them during simulation.

- **Identification of behavioral modules and memories in the design.** The user can then partition them to run on the Incisive engine or remodel them as synthesizable to boost simulation performance.

Test Incisive acceleration policy checks with this command:

```
UNIX> hal -palladium <snapshot>
```

In addition to the checks listed above, acceleration policy checking alerts users to various issues in the design coding style that might affect functionality or performance. For example, the race conditions described in the “Parallel Execution” section above.
IDENTIFY OTHER USER REQUIREMENTS

- Identify physical components to attach to the Incisive accelerator.
- Identify synthesizable memories in the design that need to be loaded from the testbench. These memories will be loaded from the simulator command line using Incisive accelerator commands.

RUN GOLDEN INCISIVE SIMULATION WITH PROFILING

Run a golden simulation with profiling turned on to determine the potential speed improvement and to identify potential speed optimizations. Invoke the Incisive simulator as shown below to profile the simulation.

UNIX> ncsim -profile <snapshot>

The profiling report is dumped to a file called ncprof.out.

The “Maximizing Simulation Performance” section in the NC-Verilog Simulator Help describes the profiling report in detail. The “Most Active Modules” section of the profiling report is of particular interest because it shows the portion of time spent running the design and testbench. With this information, the user can estimate the best case speed improvement when accelerating the DUV.

The profiling report also gives users information to help in optimizing the performance in acceleration.

4.2 OPTIMIZE DESIGN AND TESTBENCH

The user can modify the design and testbench to improve performance or ensure correct functionality based on information obtained from the acceleration policy checks and the profiling report.

Here are a few examples of typical changes users make.

- Converting memory models from behavioral to synthesizable, allowing them to run on the Incisive accelerator and improving the lock-step acceleration performance. The Incisive acceleration compiler performs this conversion automatically for simple memory models. Cadence can provide a synthesizable version of more complex memories such as SDRAMs.

- Remodeling inefficient testbench blocks flagged in the profiling report to reduce the testbench overhead, further increasing the lock-step acceleration speed.

- Rewriting code with race conditions and poor modeling style to improve both performance and behavior in the accelerator.
The next three sections describe the main SW flow through the Incisive platform.

### 4.3 Compile Design for Acceleration

At this step, the user compiles the design for the Incisive accelerator using the accelerator GUI or the accelerator’s Tcl/Tk-based scripting language called QEL. At the end of this phase, the user has a database ready to load into the hardware configuration specified. Because the compiler might provide additional warnings and error messages, run this step early in the design cycle to avoid issues later when starting lock-step acceleration. In addition, this step allows the user to verify that the configuration provides sufficient logic and memory capacity for the design.

Here is a sample QEL compile script (in this case called compile.qel).

```
saImport {-eng palladium –cosim –cosimperiod 1ns –d directives.dct designtop.v -v myip.pv} design designtop QTSACELL emulatorConfiguration –add {host pd_host}{boards 0.1+0.2} compilerOption –add {mode ice} saPrepare top compile
```

This compile script is run as

```
UNIX> questQel compile.qel
```

For more details on all the compiler switches and other accelerator information, refer to the Incisive simulation acceleration documentation. For more details on QEL commands, refer to the QEL documentation for the Incisive accelerator. The steps to run in the Incisive acceleration environment are discussed in the rest of this document.

#### Out-of-Module References (OOMRs)

An OOMR is a hierarchical reference to a net somewhere outside the current module. The Incisive platform supports OOMRs, meaning that the testbench can refer to any signal in the DUV.

#### Behavioral Model Extraction (BME)

Behavioral Model Extraction (BME) allows the software to extract behavioral models from the DUV and run them on the software simulator. The user can specify the behavioral modules in the design with a compiler directive, allowing the user to start acceleration even when some modules only have a behavioral representation. Note that the user does not need to modify the design files to move the behavioral components into the simulator.
Soft IP
The user can include soft IP in two ways.

- If a synthesizable version of the IP exists, the user can map it directly into the accelerator. If the IP comes from a third party vendor, Cadence provides a utility to protect the source code. Using this utility, the vendor can distribute protected HDL as well as protected memory contents.

- The user can include C/C++ models in the acceleration. The user can link in any PLI-compliant C/C++ programs.

Physical IP
The Incisive accelerator supports physical IP in acceleration. Physical IP can be third party IP such as an ARM processor or a customer-designed target board. The physical IP can either reside in an accelerator IP Chassis or on a physical PCB external to the accelerator. The only constraint is that any physical components must allow for their clocks, which will be controlled by the simulation, to be stopped without adversely affecting operation.

4.4 CREATE DESIGN SHELL
The Incisive unified simulator requires a module reference for the DUV. The cosimprep utility creates a “shell file” representing the design on the accelerator from the Incisive unified simulator perspective. The user will include this file in the Incisive software compile.

4.5 COMPILe/ELABORATE TESTBENCH AND DESIGN SHELL
The user compiles and elaborates using the standard Incisive unified simulator flow. The user only compiles the testbench and HDL shell file for the Incisive unified simulator. The HDL shell provides the simulation with the module interface for the DUV residing in the Incisive accelerator.

At this point, the user can statically or dynamically link custom C code as well as other tools or C-based soft IP.

Refer to the Incisive unified simulator documentation for more details on compilation and elaboration.

4.6 RUN ACCELERATION AND DEBUG
At this step, the verification team can begin simulation of the testbench and DUV by invoking the Incisive simulation environment.

As the figure below demonstrates, the verification engineer uses the same features as if the entire simulation were running in the software simulator alone. Verification engineers familiar with the Incisive simulation environment can quickly become productive with the simulation acceleration environment.

Through the Incisive verification environment, the user has access to all the Incisive simulation commands and the Incisive acceleration commands. The user controls both the software simulator and accelerator and can display the results in the same environment.
Refer to the Incisive unified simulator documentation and the Incisive simulation acceleration documentation for more details on the debug environment.

5 SEE ALSO
The following documents might be useful for additional reading.
1. Incisive simulation acceleration documentation
2. Incisive unified simulator documentation
3. Incisive hardware analysis tool documentation
4. QEL documentation for the Incisive accelerator