TECHNICAL PAPER

cadence

INCISIVE PLAN-TO-CLOSURE METHODOLOGY: DESIGN TEAM VERIFICATION

TABLE OF CONTENTS

1. SCOPE

This technical paper provides a brief overview of the Cadence® Incisive® Plan-to-Closure methodology as it applies to design teams that seek to improve the effectiveness of their functional verification process. The methodology described in this paper was built around the Incisive Design Team family of products, including Design Team Manager, Design Team Simulator, Formal Verifier, and the Xtreme® Server accelerator/emulator. The methodology is provided to Incisive customers in the form of a series of documented best practices, "golden" executable examples, libraries, and training materials. The design team verification methodology is also a subset of a broader Incisive Enterprise Plan-to-Closure process, which includes additional steps for top-down planning, system-level verification, enterprise-wide VIP reuse standards, hardware/software co-verification, and other processes that are described separately in a technical paper entitled, "Incisive Enterprise Plan-to-Closure Methodology."

2. INTRODUCTION

Most electronics companies have come to realize that early, thorough functional verification is essential to reducing the risk of costly silicon re-spins and product introduction delays. Consequently, design teams are now often tasked with both design and verification of the RTL code they create. Unlike specialized verification teams that can focus 100% of their time on the verification process, design engineers have unique challenges: they must be proficient in the use of simulation, synthesis, and timing analysis tools, fluent in both RTL and testbench description languages, and well-versed in both speed/area tradeoffs and advanced verification techniques—a full plate to say the least.

SystemVerilog holds the promise of making it easier for designers to perform more effective verification. However, the language by itself is not enough. Equally important are a set of SystemVerilog-based tools and methodologies that enable design engineers to achieve high coverage and high quality on their designs without devoting months of effort, without needing to learn complex object-oriented programming techniques, and without assuming the risk of disruptive changes to their verification process.

The Incisive Plan-to-Closure methodology provides just that. It defines a series of incremental steps that each design team can take to improve its verification process automation (VPA) in three key areas:

- Making the verification process predictable—that is, knowing where you are, when you're done, and hitting schedules
- Finding most design flaws early, at the block level, where they're easiest to detect and fix
- Dealing with integration-level complexity, when the blocks are assembled and need to be verified together

Most design teams face one or more of the above challenges, and hence the Plan-to-Closure methodology for design teams is organized into three major sections that offer incremental solutions to these problems:

- How to use planning, coverage, and automated management to improve the predictability of your design process
- How to thoroughly verify blocks using assertion-based verfication (ABV), formal analysis, and randomized test generation
- How to thoroughly verify clusters and full-chip functionality using simulation farms and acceleration

The Incisive Plan-to-Closure methodology accommodates design teams that are responsible for the entire verification process as well as those that operate as part of a broader enterprise process, including multiple specialists (system-level verification, hardware/software co-verification, emulation).

3. IMPROVING PREDICTABILITY WITH PLANNING AND AUTOMATED MANAGEMENT

The best way to improve the predictability of a project is to begin with the end in mind. That is, start with a comprehensive plan that defines "verification closure" and then continuously measure progress against that plan throughout the project. The verification plan acts as a catalyst to align the entire design team to the goals and status of the verification effort. In addition, with an accurate view of project status, managers can then optimize the allocation of people, machines, and tools to avoid schedule slips.

A good verification plan starts with a focus on the design: what features/behaviors need to be verified, under what conditions, and with what constraints. It also indicates how those features will be verified: with what technologies (simulation, formal analysis, acceleration) and with what verification infrastructure (testbenches, transaction sequence generators, response checkers). Such a plan also identifies how coverage will be assessed (functional coverage metrics including FSM coverage, assertion-based coverage, control (condition/sequence) coverage, data coverage), so that we can measure progress toward complete verification of the design. The most comprehensive gauge of progress is referred to as "total coverage," which is the collection of coverage information from all sources, including code coverage, FSM coverage, assertion-language-based coverage, and SystemVerilog covergroups. Together these provide a complete and coherent picture of verification status at any given time.

While the verification plan can exist in many forms (white board, Word document, or cafeteria napkin), capturing the verification plan in an executable form has a number of advantages. An executable plan, such as the one used by Incisive Design Team Manager, can be used to drive regression test runs, combine multiple sources of coverage, and automatically track progress toward closure. Since the executable plan is an active component of the process and not just a reference document, it is also more likely to be kept up to date.

Although formalized top-down planning has all the above advantages, the up-front effort involved in the planning process may not be acceptable to some design teams, especially those working on smaller designs or on blocks within a design. To retain some of the benefits of an executable plan (without having to create one manually), Incisive Design Team Manager provides automatic derivation of a verification plan from coverage points and assertions that have been added to the design. These may include SystemVerilog assertions and covergroups, IEEE 1850 PSL assertions, and assertion/coverage checkers from a library. The block-level plan can be leveraged later on as part of a more formalized verification plan for the cluster or chip level.

4. FINDING BUGS EARLY WITH THOROUGH BLOCK-LEVEL VERIFICATION

While most designers perform some level of functional verification of their blocks prior to synthesis and integration, the effectiveness of the verification performed varies significantly from team to team and person to person. Directed tests are often the mainstay of block-level verification but are usually insufficient to achieve adequate coverage prior to integration. Hence, many design teams would like to extend the coverage of their block-level testing and are looking to the use of assertions, formal analysis, and randomized test generation as the means to that end.

4.1 USING ASSERTION-BASED VERIFICATION (ABV)

Adding assertions to RTL blocks is a highly leveraged activity, since the same assertions can dramatically impact coverage measurement and error localization, while also enabling the use of formal analysis. In addition, the same assertions used for block-level verification can be used later for cluster/full-chip simulation and acceleration.

The Incisive Design Team family supports all the popular assertion languages, including SVA (SystemVerilog Assertions), PSL, and OVL. In keeping with the theme of incremental, low-risk changes to the verification process, the Incisive Plan-to-Closure methodology enables a gradual approach to the use of assertions. Designers can get started by utilizing the Incisive Assertion Library (IAL) provided with the Incisive Design Team family. The IAL is a superset of the OVL standard and contains a set of assertions for verifying common structures such as multi-port FIFOs and CRC generators. Once designers have become familiar with the use of the IAL, they will likely want to adopt SVA and/or PSL to specify assertions for special cases that are beyond the scope of the IAL.

Assertions can also be used to validate the inputs of each block, checking for “legal” stimulus at all times. This helps detect bugs in the testbench and it is extremely valuable for problem isolation at the cluster or chip-level integration phases. Such input-checking assertions act as a kind of “perimeter defense” that prevents neighboring blocks from passing illegal values into your block, and can thus help you avoid hours of needless debugging time chasing someone else's problem.

Assertions are useful not only in automatically detecting bugs, but also in providing valuable coverage data. Combining implicit coverage information from assertions and explicit functional coverage constructs (cover points and covergroups), the information ensures that all the important functionality is tested at the block level. In addition, the assertions and coverage points can be leveraged at the cluster and chip level with no additional effort, and they contribute to the total coverage metric tracked by Incisive Design Team Manager.

4.2 USING FORMAL ANALYSIS

Formal analysis tools use mathematical techniques to exhaustively verify a design's functionality according to the assertions you have defined. Traditionally, formal analysis required a lot of specialized knowledge and expertise, and therefore was only usable by a few verification specialists. That is not the case with Incisive Formal Verifier, which was designed explicitly with design engineers in mind. By leveraging the same user interface as Incisive Simulator and the same assertions that are relevant for dynamic verification, Incisive Formal Verifier is a key ingredient of a comprehensive ABV methodology optimized for design teams.

The primary advantage of formal analysis is that it enables verification to start months earlier than traditional simulation since there is no need for a testbench. In addition, formal analysis improves quality since it often exposes corner-case bugs that are difficult or even impossible to find using simulation, acceleration, or even emulation. Formal analysis is especially effective for verifying control functions including state machines, arbitration logic, pipeline control, speculative operations, and exception handling. Any additional coverage obtained using formal analysis will be added to the total coverage calculation performed by Incisive Design Team Manager. The formal technology can also be used to identify paths that are impossible to reach so they can be excluded from coverage analysis.

4.3 USING SYSTEMVERILOG FOR DIRECTED AND RANDOMIZED TEST GENERATION

SystemVerilog provides special language constructs such as data type extensions, interface, and package, which make it easier for design teams to build and maintain their testbenches. The language also enables testbench randomization using the *randomize* construct and the associated *with* clause to specify constraints. These and other more advanced SystemVerilog features of Incisive Design Team Simulator dramatically improve verification productivity.

Where possible, a good first step with SystemVerilog is to build bus-functional models (BFMs) for the major interfaces of the block under test. These BFMs can then be used to apply both directed and randomized tests. In both cases it is important to enable coverage data collection to measure testbench efficiency and progress toward closure. As mentioned previously, SVA/PSL assertions and SystemVerilog *covergroups* provide a means for control and data coverage analysis, respectively. The *covergroup* construct also enables cross-coverage analysis. As with each step in the verification process, Incisive Design Team Manager incorporates this coverage data into the total coverage calculation.

5 OVERCOMING THE COMPLEXITY OF CLUSTER/CHIP-LEVEL VERIFICATION

Thorough block-level verification is important because it is much easier and much less costly to find and fix bugs at the block level than at higher levels of integration. As blocks are combined into clusters and eventually into the full chip, the number of combinations of possible interactions between blocks rises exponentially. At the same time, simulation performance degrades with increasing model complexity. These two effects compound one another to make cluster and chip-level verification a major challenge, and not necessarily a simple extension of the verification strategy used at the block level.

5.1 USING ASSERTIONS TO IMPROVE DEBUG EFFICIENCY

As with block-level verification, assertions can also be used at the cluster/full-chip level during simulation and acceleration to improve observability and help isolate the root cause of failures. This is especially valuable at the cluster/chip levels where runtimes can be very long, and assertions eliminate the extra iterations that are often required for debugging. Assertions also catch errors that might otherwise take a long time to propagate to the outputs.

5.2 USING SIMULATION FARMS EFFECTIVELY

Verification at the cluster/chip level relies heavily on assertions to flag issues at the block boundaries, and on more complex constrained randomization of the testbench to check for corner-case interactions between blocks. With the increasing simulation load, a simulation server "farm" consisting of at least several servers becomes a requirement. These servers can be used concurrently to simulate the same model with different sets of stimuli, or in some cases, to simulate overlapping clusters as a way to achieve integration-level coverage without simulating the whole chip at once. In other cases, full-chip simulation is mandatory and the chip-level model is simulated on many computers in parallel, but with each server applying a different set of constrained random tests based on different seeds. The number of servers required will depend on the complexity of the model and the magnitude of the verification regression suite.

5.3 MANAGING THE MOUNTAIN OF DATA

Once in the realm of simulation farms and constrained random test generation, simulation data management becomes a major issue. Parallel simulations create a mountain of data that must be analyzed and managed, including simulator and testbench log files, assertion messages, code coverage files, functional coverage files, and so on. Automating the management of both the compute resources and the resulting data is essential to increasing project predictability while achieving the highest resource utilization.

Automated management technology like Incisive Design Team Manager is essential for this task, and can help automate the execution, analysis, and debugging processes. The executable verification plan is used to initiate the launch of the simulation sessions, leveraging dispatch technology such as LSF to optimize the use of workstation and license resources. Incisive Design Team Manager tracks session progress, and any failing simulations can be automatically rerun overnight with debug flags turned on so that users can immediately focus on debugging when they arrive in the morning.

Incisive Design Team Manager also provides rapid identification of the most important simulation failures resulting from one or more sessions. For example, failures can be correlated between simulation runs to determine if there is a particular bug that has a broader impact beyond the local context in which the error message was first flagged.

To achieve rapid verification closure, engineers must be able to create unique simulations with as little redundancy as possible. Coverage-hole analysis helps identify the largest, easily targeted, uncovered areas. Ranking coverage contribution identifies those simulations that contribute the most to the overall coverage goals. By ranking the simulations based on their relative contribution, the design team can elect to run the smallest number of high-impact simulations that yield the greatest increases in coverage. This is especially useful during the final stages of the project when the regression tests need to complete as quickly as possible to avoid schedule slips.

5.4 USING ACCELERATION EFFECTIVELY

When working with large ASICs or SoCs, full chip-level simulation speeds are often measured in only tens of cycles/second. Even with the use of simulation farms, these rates are simply too slow for devices such as network processors, graphics chips, and embedded processors, which require long contiguous streams of stimuli based on protocol data packets, video frames, or software execution. For devices like these, hardware-assisted verification is a must. The challenge is how to find an accelerator/emulator that fits well into the design flow and doesn't require a lot of specialized expertise. Incisive Xtreme Server offers such a solution and is optimized specifically for use by design teams.

Two key requirements for the effective integration of acceleration into the design flow are standards-based assertion support (known as assertion-based acceleration, or ABA) and transaction-based acceleration (TBA). As with simulation, the use of assertions is key to detecting problems close to their source. This is especially important given the size of the model and the length of the simulation sequences at the chip level. Transaction-based acceleration is essential for performance because it enables efficient, transaction-level communication between the simulator and the accelerator. This methodology also facilitates debugging at the transaction level and complete reuse of the simulation-based tests in acceleration mode without sacrificing performance.

Most design teams begin to use acceleration at the cluster/full-chip level once the bug-discovery rate using simulation has fallen to a fairly low level. The combination of transaction-oriented testbenches and assertions is a great strategy for balancing the tradeoff among speed, debugging, and testbench reuse. As with simulation and formal analysis, coverage data from the acceleration process is merged into the total coverage view maintained by Incisive Design Team Manager.

6 VERIFICATION CLOSURE

With the Incisive Plan-to-Closure methodology, closure analysis takes place throughout the verification process, as the test results at each step are compared against the verification plan and the progress toward closure is measured. Thus, the coverage gained at the block/cluster/chip levels (using a mix of simulation, formal analysis, and acceleration) all contribute to the overall coverage score. This score then provides an objective measure that all the functions and use-cases defined in the verification plan have been verified. While the ultimate decision to tape out still relies on engineering judgment, the planning, coverage, and automated management provided by Incisive Design Team Manager makes the decision much more data-driven and objective.

7 CONCLUSION

Design teams are shouldering more and more of the verification burden and need to adopt new methodologies to keep pace with the complexity of their designs, while minimizing the risks associated with process change.

The Incisive Design Team family provides the technologies and methodologies required to find and fix problems early in the design cycle, overcome the performance and complexity challenges of full-chip verification, and manage the entire verification process from initial plan to final closure.

The Incisive Design Team Plan-to-Closure methodology described in this paper is provided to Incisive customers as a set of documented best practices, “golden” executable examples, libraries, and training materials. The primary Cadence products that implement this methodology include:

- Incisive Design Team Manager
- Incisive Design Team Simulator
- Incisive Formal Verifier
- Incisive Xtreme Server

In addition to the products and methodologies discussed in this paper, the Incisive Design Team solution is a subset of a broader Incisive Enterprise family of products and the corresponding Enterprise Plan-to-Closure methodology. As such, design teams can adopt pieces of the Enterprise methodology to further automate their verification process and/or contribute their verification efforts to specialized verification teams that use the Enterprise methodology. Cadence also has more than 200 verification experts in the field who will work with your design teams to assess their needs and help them quickly adopt the appropriate products and methodologies. With the Cadence Incisive Design Team solution and expert support, you can be assured that you will dramatically increase the overall productivity, predictability, and quality of your verification process.

For more information on the Incisive Design Team family please visit http://www.cadence.com/products/functional_ver/incisive_design_team/.

Cadence Design Systems, Inc.

Corporate Headquarters
2655 Seely Avenue
San Jose, CA 95134
800.746.6223
408.943.1234
www.cadence.com

6604 11/05