



Find and fix problems early in the design cycle

THE NEED FOR EXPENSIVE VERIFICATION CYCLES PROVES THAT ELECTRONICS DESIGNERS AND EDA VENDORS ARE HUMAN. BETTER TO EMPLOY TOOLS AND METHODS THAT AVOID MOST MISTAKES.

At a glance.....62
Read more about it62
Breaking down the silent-productivity barrier: What's next for debugging?.....64
For more information66

DESIGN VERIFICATION REQUIRES significant time and resources from every design team. Its cost is directly proportional to the size of the design and the diversity of disciplines that the design's development involves. Designs that require hardware and software co-design or use both digital and analog techniques are also the most complex to verify. Advances in process technology enable engineers

to use more gates with each successive decrease in feature size. Today, an average design uses millions of gates; many designs use tens of millions of gates; and state-of-the-art processes will soon allow 100 million gates on a single IC. Advances in verification technologies are lagging behind semiconductor-fabrication capabilities. But this difficulty, a significant contributor to the design gap, is not new. The problems of verifying the correctness of a design before releasing the product to manufacturing and providing sufficient and efficient manufacturing test suites have existed for at least 35 years. The

gap between manufacturing capabilities and verification has grown steadily, and, consequently, verification costs have escalated in lock step with design complexity. The purpose of design verification is to ensure correctness. But this purpose assumes different meanings, depending on the development phase your design has achieved. Correctness means compliance with specifications, design rules, industry standards, and corporate design practices. It also means equivalency of two implementations at different levels of abstractions, such as RTL (register-transfer level) and gate level.

The design team must also ensure the correct design and implementation of protocols for hardware and software interfacing. As imposing as the list seems, it only partially describes the possible design and implementation problems that you must test and verify. During the keynote speech at the Synopsys Users Group meeting on Sept 8, 2003, Aart de Geus, chairman and chief executive officer of Synopsys, stated that 61% of all new ICs and ASICs require at least one respin. Of these devices, 43% exhibit errors due to functional logic errors; current verification techniques would fail to identify the causes of only 3% of the malfunctions. Yet, engineers lack the time and resources to identify and correct all design problems. The impact of design verification on the cost, length, and difficulty of product development is significant.

A FRACTURED LANDSCAPE

Obviously, avoiding problems is more efficient than fixing them, yet little of the design community favors this approach. For example, VHDL is more robust and well-defined than Verilog. Designers using VHDL can automatically avoid some types of errors that users of Verilog must find through verification, yet most engineers have opted to use Verilog, because it is easier to learn and simulates faster than VHDL. Still, simpler languages do not support the more rigorous methodology designers require when developing complex designs. This limitation is the principal reason behind the development of SystemVerilog and Verilog 2005.

Verification engineers cannot count on a well-established, industry-standard, verification methodology. Verification tools run the gamut from software-development languages, such as C, to hardware-description languages, such as VHDL, to special-purpose test languages, such as OpenVera from Synopsys and "e" from Verisity. The field is in flux, and the instability has a direct impact on the development of reusable modules and practices. The problem is most serious at the architectural level of abstraction, at which avoiding or at least finding and eliminating errors is most beneficial to profitability.

Kazu Yamada, general manager of the technology-foundation-development division at NEC Electronics believes verification comprises three primary stages:

AT A GLANCE

- ▶ The gap between manufacturing capabilities and engineering productivity continues to grow.
- ▶ No standard verification methodology exists.
- ▶ Formal proofs reduce designers' dependence from functional verification.
- ▶ Circuit verification is gaining in importance due to the many electrophysical issues in deep-submicron fabrication.

system verification, logic verification, and timing verification. He affirms that the most critical area in system-on-chip design verification is system verification. NEC, failing to find a satisfactory commercial answer to this problem, developed its own high-level-design environment to improve system verification for its designers and customers. Although approaches such as NEC's can shorten design times and lower development costs, they are still far from optimal. The problem rests in the tools they use. For many years, EDA vendors' marketing departments have pointed out the savings a customer would obtain by finding and fixing problems earlier in the development cycle. Unfortunately, these presentations seldom depict the benefits of avoiding mistakes. One way to implement such a strategy is by verifying the design at the highest possible level of abstraction. This approach works, because designs introduce more and more details

READ MORE ABOUT IT

Over the last three years, *EDN* has covered many aspects of design verification. You can review these articles by visiting www.edn.com and clicking on the "archives" button. Once there, search for "design verification" or other related subjects. The Web site also offers a list of EDA companies organized by market segment. Choose "EDA Tools" under "Technical Resources" and then look for the EDA-vendor matrix. The spreadsheet provides you with contact information for most EDA companies, including those that offer solutions to a range of verification problems.

as they progress, making the verification problem larger and thus more difficult to find and fix.

In the last couple of years, verification-tool vendors have turned their attention to designers and marketing professionals working at the specification level because the initial phase of product design and development is an untapped market and because the technological and financial barriers to entry are lower than what you need to develop and market a back-end tool, such as an IC-layout product. To lower the adoption barrier, many EDA vendors are using the software-programming language C or one of its derivatives, such as C++ or SystemC, as the description language for the specification and architectural design of electronic products. These vendors base their decisions on the fact that, because C and the Unix operating system related to it are inexpensive to obtain and maintain, most universities use it to teach programming. They are counting on the fact that most electronics engineers have some familiarity with C.

Unfortunately, C has neither the constructs nor the robustness to precisely describe an electronic product. Its derivative, C++, is better for software development but difficult to master and use. Unwise use of its powerful constructs can lead developers to generate errors that are difficult to find and costly to fix. SystemC attempts to introduce into C++ some hardware constructs, such as clock cycles and concurrency, but still fails to capture the essence of hardware design. Thus, it falls short in helping architects to clearly decide, for example, between hardware and software implementations of portions of a design. Engineering teams to date successfully using SystemC for design verification comprise senior and experienced engineers. How the methodology would work for the average engineer is still unknown. The experiments so far show only that no substitute exists for experience, regardless of the tools you use.

More appropriate languages, such as Esterel, which products from Esterel Technologies use, are available. EDActive Computing is developing tools based on the Rosetta language, and KeesDA is developing tools based on the B method. Esterel, B, and Rosetta provide robust methods of defining and specifying design requirements and constraints. Engi-

neers using them find that designs contain fewer errors and that it is easier to obtain formal proofs of design correctness. According to Prakash Narain, president and chief executive officer of Real Intent Inc, "The paradigms of abstraction and divide and conquer have not been developed and deployed for functional verification." He notes that "assertions and property-based specification are the most realistic paradigms to implement divide and conquer," but abstract design representations are much further off. Narain believes chip design is a precise art and that it is unclear what abstract and precise methodology will emerge. The industry, therefore, is still trying to find—instead of avoid—errors.

FORMAL PROOFS

Design teams spend most of a project's computer time running logic simulations to functionally verify their designs. Logic simulation continues to be the most ef-

fective tool for identifying design errors and verifying that the subsequent corrective measures have fixed the problem. Two factors negatively impact the efficiency of logic simulation. Its execution is based on mimicking the functions of the real circuit by modeling the propagation of logic signals through the circuit. As the size of the circuit increases, so does the time required for the simulation and the size of the results database. Engineers must monitor more logic functions, and the amount of test vectors they employ also grows significantly.

Developing test vectors is not a deterministic technique. The effectiveness of a set of test vectors depends mostly on the intuition and creativity of verification engineers. Therefore, you can never definitely conclude that you have tested all possible sources of errors when dealing with even moderately complex circuits. To address this challenge, verification methodology has evolved from manual-

ly created tests to constraints-driven random stimulus-generation techniques. Constrained random verification exposes conditions that are impossible to comprehend manually, but the random nature of its methodology makes it difficult to determine whether a design has been adequately verified.

Formal verification attempts to address the drawbacks of functional verification. It does not require test vectors, although a commonly used technique, semiformal verification, relies on logic simulation to establish the starting state for formal proof. Formal verification uses mathematical analysis to determine the state of the circuit and the relationships between consecutive states. This technique executes faster than computing the logic states of all the nodes in the circuit. Pure formal verification is difficult to use because it requires engineers to express designs in a form that suits mathematical proofs. Although some of the afore-

BREAKING DOWN THE SILENT-PRODUCTIVITY BARRIER: WHAT'S NEXT FOR DEBUGGING?

By Scott Sandler, *Novas Software Inc*

The complex silicon technologies, design flows, and organizations that the development of today's ICs require lead to less familiarity with design behavior and its causes. This situation creates a productivity crisis in engineering-intensive operations that sheer EDA tool horsepower, such as faster simulation, cannot solve.

Most design teams report that they spend more than 50% of their verification cycles debugging—trying to understand how designs are supposed to work or why they don't function as intended. The economic impact is significant in terms of direct costs but even more significant in the business ramifications of missed product schedules. For some companies, excess time spent on debugging could mean the difference between success and survival.

Designers are facing a discontinuity in debugging, particularly as engineering resources

become scarce and advanced system-on-chip methodologies become a requirement. The key to breaking through the resulting productivity bottleneck combines more efficient ways to accelerate design comprehension, automation of much of the interactive effort that engineers currently expend in debugging, and simplification of complex tool environments through the application of a "universal debugging window."

The cornerstone of the next-generation debugging environment will be the shift from today's fragmented viewing mechanisms with each tool having its own version of a graphical user interface to a more universal approach that employs a common user interface across the entire design-and-verification process. Such a scenario will ensure consistency in viewing results from multiple tools, allow cross-fertilization of data throughout the chip-develop-

ment process, and significantly reduce learning curves.

Debugging systems must fully embrace advanced technologies, such as formal methods and synthesis, to automate the more mundane parts of tracing and to eliminate bugs. These new techniques enable the debugging paradigm to shift from tracking events and design structure section by section to an automated approach that presents designers with a complete picture of device behavior. The prudent use of synthesis and formal-verification technologies during debugging can immediately expose the cause-and-effect relationships behind identified design problems. As these new techniques come into play, next-generation debugging tools must help automate the debugging of not only designs, but also transactions, testbenches, and assertions.

Integration of bus-protocol monitors and techniques for

capturing, analyzing, and visualizing behavioral operation will aid verification of both standard and proprietary interfaces at the system level. These kinds of capabilities raise the abstraction of debugging knowledge from basic structural elements and signal transfer to protocol and stream data transfer both within design blocks and across module interfaces. They will allow the debugging function to cover a broader range of design-and-verification scenarios, essential to finding bugs as early as possible in the design process.

What's next for debugging lies beyond waveforms, and continued investments in debug automation and methodologies will reduce the barrier to productivity for this critical aspect of chip verification.

AUTHOR'S BIOGRAPHY
Scott Sandler is president and chief executive officer of Novas Software Inc.

mentioned emerging languages can support the required formality, the cost of training and retooling large numbers of engineers is slowing acceptance. Therefore, formal-verification methods tend to require engineers to divide circuits into smaller modules and use assertions or define design properties to guide the proof engine.

Many formal-verification vendors support assertion-based verification. It improves implementation coverage by providing assertions and coverage points within the design, which, in turn, improve observability, provide coverage feedback for both simulation and formal verification and enable the use of automated formal-analysis tools to improve verification coverage. Richard Ho, chief architect at 0-In Design Automation points to arbitration logic as an example of a circuit susceptible to assertion-based verification. He notes that you can target this “verification hot spot” with assertions or verification IP (intellectual property), such as 0-In’s CheckerWare and then completely verify it with exhaustive formal tools. According to Ho, this approach provides complete coverage of the arbitration logic without millions of cycles of tedious and repetitious simulation.

Both Cadence and Synopsys offer verification systems that use a mixture of functional and formal techniques to verify a design. Hybrid RTL formal verification uses formal techniques to drive the built-in simulation engine, and the proof engine continuously performs formal analysis to verify whether the design satisfies a given property. It provides deterministic results that are free of false-neg-

ative errors; conditions that are never going to occur while the chip is in operation cause these errors.

The inherent limitation of formal proof engines has narrowed the scope of most formal tools to localized, implementation-specific assertions, often embedded in the RTL code. Jasper Design Automation offers a block-level pure formal-verification product that operates at the highest level of assertion-based verification—the specification level—to provide end-to-end properties that verify a block’s intended “black-box” behavior.

FINDING THE ERRORS

Brian Bailey, chief technologist in the design-verification and test division of Mentor Graphics, points out that engineers are still fundamentally missing the closure mechanism for verification. They need a way to decide when a verification task is complete, whether they have adequately covered the important aspects of the design, whether they have determined if required function is missing, and whether the tests cover all aspects of the design. To answer these questions, engineers must be able to understand functional coverage and its relationship to the design process. Bailey states that when designers achieve this understanding, they will be able to think about complete, optimal verification. Until then, they must just do more simulation or take their chances on a tape-out and then debug the actual silicon implementation.

It is not uncommon today for engineers to describe a single design with more than one HDL, especially if the design uses third-party cores. Often, the design team will also use a testbench lan-

guage, such as “e” or OpenVera; an assertion language for formal verification; and some form of C. The typical tool flow for an IC or ASIC design can include as many as 30 tools from a number of vendors. Many of these tools have their own debugging environments. The net result is that any engineer on a team will be unfamiliar with some of the user interfaces, complicating communications among team members. (See **sidebar** “Breaking down the silent-productivity barrier: What is next for debugging?” for a new approach that unifies the user interface and handles both the hierarchy of the design and the different forms of abstraction used throughout the development.)

Design and verification engineers encounter the same type of fragmentation when dealing with logic simulation and test development. Frequently, tests are written using a language different from that used in hardware design. Many development teams are decentralized and scattered around the globe. Teams use different languages for development, and designs often integrate third-party models that are available in only one language. The VSI Alliance is beginning to address the problem of third-party-core verification; one of its working groups has just finished work on a Quality Metric User Guide for virtual components.

Both Mentor Graphics and Synopsys now offer logic-simulation environments that support multiple modeling languages. The companies offer simulation platforms that support mixed-language and -signal simulation, testbenches, and assertions. Cadence’s Incisive platform provides the same functions and offers closely coupled integration with hardware

FOR MORE INFORMATION...

For more information on products such as those discussed in this article, contact any of the following manufacturers or organizations directly, and please let them know you read about their products in *EDN*.

Aptix
www.aptx.com

CoWare
www.coware.com

Jasper Design Automation
www.jasper-da.com

NEC Electronics
www.necel.com

Synopsys
www.synopsys.com

Verity
www.verity.com

ARC International
www.arc.com

EDaptive Computing
www.edaptive.com

KeesDA
www.keesda.com

Novas
www.novas.com

Synplicity
www.synplicity.com

VSI Alliance
www.vsia.org

Axis
www.axiscorp.com

Esterel Technologies
www.esterel-technologies.com

Mentor Graphics
www.mentor.com

Pittsburgh Simulation
www.pitsim.com

Tensilica
www.tensilica.com

0-In Design Automation
www.0-in.com

Cadence Design Systems
www.cadence.com

EVE
www.eve-team.com

Nassda
www.nassda.com

Quickturn
www.quickturn.com

Texas Instruments
www.ti.com

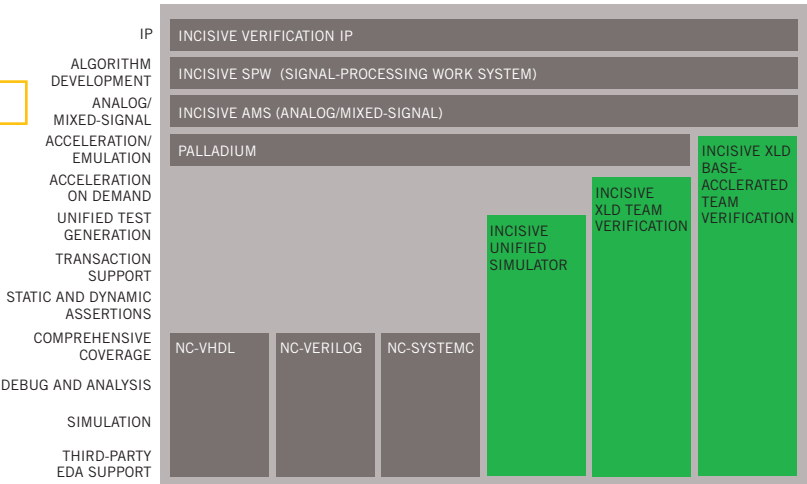
Real Intent Inc
www.realintent.com

Tharas Systems
www.tharas.com

emulation and acceleration (Figure 1).

Engineers developing pc boards using one or more ASICs often need to prototype the de-

Figure 1



Cadence's verification platform supports multilanguage and both remote and local hardware acceleration and emulation.

sign. Synplicity markets Certify, which allows designers to prototype the ASIC using FPGA devices and seamlessly connect the emulation board to the pc board under development. Synopsys has introduced DFV (Design for Verification), a method that combines simulation and formal analysis. It leverages a designer's intent and knowledge to ensure the correctness of the modules that make up the circuit and their integration. DFV allows designers to use design assertions as a focal point of specification to capture design assumptions, properties, and interface constraints. Engineers employing DFV also use multilevel interface design to ensure consistency between transaction-level specifications and register-level implementations. The fundamental idea behind the method is to enable designers to "say what they mean," instead of forcing them to paraphrase the design intent through the limitations of existing EDA tools.

VERIFYING THE SOFTWARE

The increase in the number of available gates on the die has made it possible to include a significant amount of firmware on the chip. The ability to include firmware on an IC provides marketing with the flexibility to generate a product family that may require little, if any, new logic design for new members of the family. It also diminishes the amount of electronic design required and, in theory, should decrease development time. But this strategy has one major drawback that you can characterize as a "chicken-or-the-egg" situation. When designing an ASIC device, you must make a trade-off between which functions to implement in hardware and which to leave to firmware developers. Once you make the decision, programmers must have the hardware available to debug their code. Yet, if the team develops the hardware and the firmware sequentially, the development time is too lengthy to ensure a profitable market entry. Sequential development also means that the hardware architecture is frozen, turning a hardware/software trade-off into an expensive redesign effort.

EDA vendors have developed various

ways to allow developers to proceed in parallel with hardware and firmware development. It is important to realize that hardware/software co-development requires a disciplined process, because both teams must proceed in a way that provides the other team with required pieces of the implementation at the correct time. Team managers must pay close attention to the development schedule and be flexible to rearrange priorities and assignments if unforeseen requirements occur. Tools for hardware/software co-development include ISSs (instruction-set simulators), hardware emulators and accelerators, bus-functional models, and hardware prototypes.

Tensilica supplies configurable and extensible microprocessor cores. It provides its customers with tools, including the Xtensa Xplorer, an ISS that integrates software-development, processor-optimization, and multiple-processor system-on-chip architecture tools in one common design environment. Xplorer serves as a cockpit for basic design management, invocation of Tensilica processor-configuration tools, and software-development tools. It is particularly useful for the development of TIE (Tensilica Instruction Extension) instructions—designer-defined instruction extensions to the Xtensa processor that maximize performance for certain applications. You can save processors and TIE configurations, profile them against C or C++ software, and compare them.

ARC International provides the MetaWare tool chain, which allows engineers

to replace the processor RTL model with an ISS. ARC has designed the tool to perform hardware and software checking at the system level when packets of data are injected into the system, instruction sequences are executed, and the system interactions are checked.

Aptix, Axis, Mentor Graphics, and Eve market emulators allow designers to implement a functional image of the hardware they are developing by mapping the equivalent functions onto FPGA devices. In this manner, software engineers have a copy of the developing hardware that they can use for firmware development. Mentor Graphics, Pittsburgh Simulation, Quickturn (a Cadence company), and Tharas Systems offer hardware accelerators that substantially improve the throughput of logic simulators by compiling nonbehavioral portions of the design into hardware primitives within the accelerator. Accelerators are expensive capital-equipment items, so few design teams can justify their cost. Mentor and Cadence, for example, have set up pay-for-use licenses that allow customers to rent time on accelerators, using the Internet to download the design in a secure manner.

Bus-functional models offer another way to lower simulation costs and increase throughput. Developers must be willing to trade off cycle accuracy and abstraction level with shorter simulation time. This method is particularly useful when you are verifying a bus protocol or evaluating whether to implement an algorithm in hardware or firmware. Most companies that support SystemC provide



this capability. They include Cadence, CoWare, Mentor, and Synopsys.

CIRCUIT VERIFICATION

IC design became more complex once process geometries became smaller than the wavelength of visible light. The industry reached a watershed when process nodes hit the 250-nm mark. Until then, logic designers could verify a design's functions and hand a GDSII tape to a foundry, confident that the finished chip would have the same functional characteristics as the model they verified. Although problems began to manifest themselves at 180 nm, 130 and 90 nm present serious challenges to designers. In addition to reticle corrections to compensate for diffusion inaccuracies, the size of the gates and interconnections are sources of numerous physical effects that impact the circuit's ability to function. Clive Bittlestone, a fellow at Texas Instruments, says that, once a customer delivers the GDSII tape, it takes several days to verify one ASIC design. The manufacturing team must perform many types of verification: layout versus schematic to ensure that designers used the proper geometries; IR drop and electromigration for power analysis; and gate oxide and channel hot carrier to ensure that designers have observed process rules.

Processes at 130 nm and smaller require complex design rules. Foundries and EDA vendors work together for as long as 18 months to ensure that the EDA tools enforce the correct set of rules and produce no incorrect side effects. The required collaboration is a major obstacle to market entry by new EDA companies targeting both the synthesis and layout markets. Start-ups must convince foundries of their credibility and viability—from both a technical and a financial standpoint—to secure the partnership agreements required to enter the market. Deep-subwavelength rules are getting too complex, and the industry will soon need a new method of verification, based on process models rather than process rules. It will also be important to find a way to score the severity of errors designers find in mask and to ensure that designers fix only those discrepancies in the active portion of mask.

For digital circuits running at greater than 700 MHz, designers need a hybrid timing analysis, such as Hanex from Nassda, to combine the exhaustive coverage of traditional static-timing analysis with dynamic analysis of nanometer signal-integrity effects. At such small dimensions, noise effects due to circuit operation are as important as, if not more important than, problems that bad placement cause.

The tremendous growth of the wired- and wireless-communication markets has required the development of a large number of mixed-signal ICs. In both cases, engineers must use circuit-verification techniques to ensure that their designs function properly. Analog simulators have slower throughput than digital simulators, because they employ solvers that must deal with many more variables and use more complex equations. It is impractical to separate the digital and the analog portion of the design throughout the development process. Although two designers work on the two parts of the system, they must eventually integrate their work and validate the performance of the entire system.

Ju-Hsien Chern, vice president and general manager of the deep-submicron division of Mentor Graphics, contends that "the viable alternative is a multilevel approach to mixed-signal design." Certainly, the ability to simulate systems at various levels of abstraction allows companies to make proper trade-offs between design integrity and time to market. Companies that make the transition to using a single interface face the possibility of combining all types of simulation engines—from large-signal model simulation, to digital simulation, to fast-Spice and RF simulation—into one environment. Tools in this category use a single netlist hierarchy and allow designers to combine VHDL, Verilog, VHDL-AMS, Verilog-A, Spice, and C anywhere in the design. Design teams can analyze both the digital and the analog portions of the design using the appropriate methods in a fully integrated environment. As complexity at both the front and the back ends of the design process continues to grow, verification will continue to be the subject of much development by EDA vendors. □

You can reach Technical Editor Gabe Moretti at 1-941-497-9880, fax 1-941-497-9887, e-mail gmoretti@edn.com.

