



**Tracking the memory market?**

## CommsDesign

### Inside a hybrid verification model

Nilesh Ranpura

Oct 04, 2004 (9:12 AM)

URL: <http://www.commsdesign.com/showArticle.jhtml?articleID=48800521>

Over the past decade, the semiconductor industry has seen the evolution of myriad high-level languages for both design and verification. This proliferation has been compounded by the advent of a variety of point tools for each step of the ASIC design cycle. Last but not least, shrinking process geometries have enabled multimillion-gate systems-on-chip, adding another dimension of complexity to the development flow.

This combination of languages, tools, intellectual property and methodologies has morphed the traditional ASIC design cycle into a "hybrid model" process. With no readily available cookbook for selecting the best option for each stage of the cycle, many semiconductor companies have adopted the hybrid methodology to reduce risk and optimize time-to-market.

Verification consumes 60 to 70 percent of the total resources in a typical chip design cycle. The ASIC design flow requires verification at each level of abstraction, from architecture to silicon prototype. The challenge for engineers is verifying spec adherence for multiple abstractions. By its very nature, the verification process depends on language, methodology and SoC complexity in size, functionality and application.

Today, as more embedded software is being added to systems, software developers do not have access to the target hardware until there is a physical prototype. This leads to a need for common guidelines for verification methodologies, such as reusable methodologies, directed or coverage-driven approaches and co-verification.

Hardware verification has become more challenging. A 2002 survey by Collett International Research found that only 40 percent of designs were bug-free at silicon; 60 percent contained logic or functional flaws. More than 20 percent required one or more silicon re-spins. Interestingly enough, 50 percent of the total time was spent on verification.

Because there is no panacea for all verification problems, most verification processes for today's SoC designs follow the hybrid verification model. We also use this model at eInfochips.

In general, there are four major abstractions in the overall verification process: architectural validation; functional verification of each interface, using verification IP and assertions; formal verification; and virtual prototyping.

We chose SystemC for architectural validation of DMA and encapsulation blocks. In SystemC, hardware constructs were present within the context of C++ as a class library, and cycle-accurate verification modeling up to 10k cycles/second was possible. Co-simulation and event/message-level abstraction were also possible.

The basic goal of architectural validation was to validate customized DMA and proprietary encapsulation models. The overall performance and functional correctness were checked against the specifications of the SoC. SystemC's hardware constructs facilitated such modeling-performance-driven aspects as latency and throughput.

Since primary design blocks were available in-house, functional verification was carried out, so as to use readymade verification IP: eVC (e Verification Component). PCI-X and Fibre Channel eVCs were integrated with the overall testbench, along with the memory block.

The key benefit of eVCs is a coverage-driven, reusable methodology. Using a coverage model of the testbench, as opposed to writing thousands of test cases, helped identify missing scenarios in the test list. Built-in random generation of SpecmanElite allowed the generation of all possible ranges of scenarios. Just a few test cases were enough to manage and modify the overall testbench.

The testbench comprised scoreboarding features that compared data at every transaction and ensured data checking. Assertions were used extensively to make customized proprietary blocks such as DMA and encapsulation bug-free, since there was no verification IP available. RTL for both blocks had built-in assertions for protocol validity and signal checking at the timing level.

Formal verification was performed to ensure gate-level validation. Functional properties between the RTL and gate level were checked.

In virtual prototype modeling, the entire SoC RTL was transferred to multiple FPGAs by means of a specially built card. The card plugged in to a hardware platform, which had an PCI-X bus and Fibre Channel link device to generate and execute real-time traffic. Several key application tests were used to validate real-time functionality on first silicon, which was an FPGA prototype.

### **The downside**

Along with its benefits, the hybrid verification model can involve certain risks, related to:

- ✗ The integration of multiple methodologies;
- ✗ The impact of multiple languages on simulation of the environment/verification process because of multiple calls to the operating system;
- ✗ Separate skill set requirements for each language or tool;
- ✗ The need to select the right tools with due compatibility; and
- ✗ Cost vs. effectiveness on bug discovery, as well as re-spins due to functional bugs.

Since time-to-verification has become a major concern, the hybrid verification model is often the only viable alternative, since it addresses all the parameters of the verification process. The next step would be conversion of the hybrid model to a uniform model.

Nilesh Ranpura ([nilesh.ranpura@einfochips.com](mailto:nilesh.ranpura@einfochips.com)) is director of the ASIC Verification Group at eInfochips (Santa Clara, Calif.).



[See related chart](#)



**Wondering what's new  
in the memory market?**

[Copyright © 2003 CMP Media, LLC](#) | [Privacy Statement](#)