print | close

10 Best Verification Practices for Hardware Emulation

Electronic Design

Lauro Rizzatti

Wed, 2016-06-29 09:24

I recently attended an invited talk by a senior manager of a design group within a large networking company. He described the group's verification flow and it quickly became obvious that hardware emulation sits at the center of the flow. In one example he gave to the standing-room only crowd, hardware emulation was used in testbench acceleration mode to improve productivity and reduce design risk.

On top of the list was the speed of execution the group gets from emulation, allowing them to perform a level of design verification, including embedded software validation, not possible with software simulation.

What captured my attention was the list of issues that emulation cannot address or should not be used for, including some hints on how to best deploy it. It was well thought out and any engineering or verification group developing a verification strategy would benefit from using it. I expanded the list a bit with some of my own observations, based on my experience and what I hear from experts and users.

Related

11 Myths About Hardware Emulation

Implementing Functional Coverage with Hardware Emulation

Hardware Emulation: A Weapon of Mass Verification

1. Verification productivity is job #1.

With the time necessary for the verification effort, purported to be close to 70% of the entire project cycle, productivity and thoroughness must be considerations. Designs must not fail at first silicon. A re-spin at 28 nm of a large design exceeding 100 million gates would curtail a design budget by more than $10 million. Not to mention that missing a schedule by one month may have a disastrous effect on revenue.

Hardware emulation can process millions of lines of code and run billions of execution cycles in minutes or a few hours that are necessary for thorough embedded design verification/validation, or for executing processing-intensive operations such as decoding high-definition video.

The bottom line is that it is mandatory to have a well-tested verification strategy and proven methodology. It means having the right tools—everything from simulation, simulation acceleration, emulation, and virtual prototyping to silicon validation—that should be interoperable, in addition to a well-trained group of verification engineers.

To better ensure productivity, the verification phase should start as early as possible, while development time

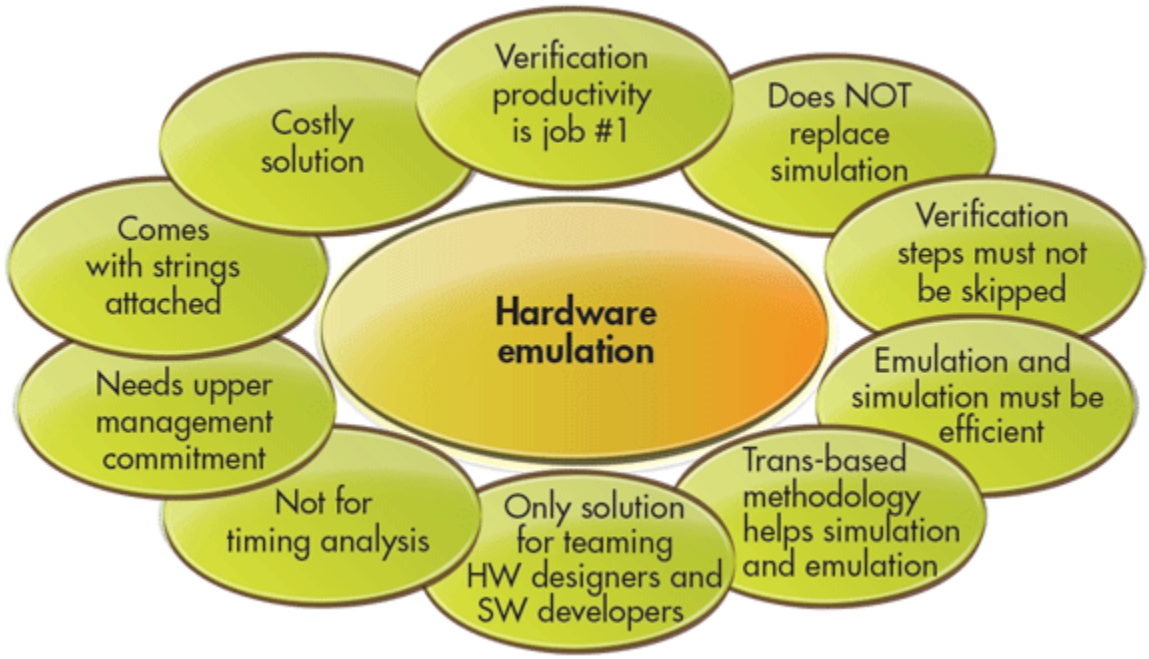
l effort should be kept to a minimum.

2. Hardware emulation is not a replacement for simulation.

Industry watchers have predicted the demise of hardware-description-language (HDL) simulators for years, but they continue to be part of the verification mix and that won't change any time soon. Knowing when to use them and to what purpose is the key. Simulators are most effective for hardware debug in the early stages of the design cycle or at the block level of design, tracking coverage and verification flow. They work best with smaller designs, under 20 million gates or thereabout.

Their interactive hardware debug and fast compilation support several design iterations per day. They don't work for embedded software validation because they run way too slowly, in the kilohertz range at best. As design sizes increase, their speed of execution drops to a few tens of hertz, or less.

That's where hardware emulation takes over, because it operates at one or several megahertz. They are marginally defeated by the increase in design sizes, from one to a few billion gates.



3. Added performance is not a justification for not verifying functionality at lower levels—verification steps must not be skipped.

The rule of thumb is that the verification engineer should find most functional bugs as soon as possible. When applied to a hierarchical design, the rule calls for finding most bugs at the block level. By the time a design is verified at the chip level, he or she should uncover only bugs related to block connectivity and integration.

Finding bugs at the chip level are by far more expensive—10 times more costly—than at the block level. Furthermore, hardware emulation is an expensive resource that should be used only for what simulation cannot handle.

4. Simulation and emulation tests need to be efficient.

Another thing to keep in mind is tests. Often, engineers create and run many tests instead of merging them into smaller numbers. To do so would require additional thinking, but the payoff could be significant. This should happen both in simulation as well as in emulation to maximize productivity.

Emulation cannot handle timing

Emulation performs cycle-accurate functional verification. It cannot handle accurate timing. The propagation delays within a design under test (DUT) mapped onto the emulator are orders of magnitude larger than in the real silicon, leading to inaccurate timing. However, the cycle-accurate aspect of the analysis can be used for DUT performance characterization.

6. Emulation brings together the pre-silicon verification team with the software-development team.

Another significant benefit of emulation is that it can bring together the software side and the pre-silicon side of the design team. Traditionally, the software-development team started embedded software validation when the first engineering samples came back from the foundry.

Today, the software-development team works together with the pre-silicon team to verify on the same the chip design way ahead of silicon availability. By the time the chip comes back from the foundry, all of the software and validation tests have been validated together with the hardware through verification.

Basically, within a week or two from first silicon, the chip can be in the customer's hands.

7. Transaction-based methodology helps simulation and emulation.

A transaction-based verification methodology can be extended from simulation to emulation. The presenter reported that with a byproduct of making the above transition, the team saw improvement in simulation performance. Specifically, when the memory was changed to support emulation, it doubled simulation speed.

They also separated the HDL timing-driven analysis from the object-oriented analysis, which further improved simulation performance.

8. Hardware emulation comes with strings attached.

Yes, this is true. Hardware emulators can accommodate any design size, but they require a long setup time and are relatively slow to compile, compared to simulation.

While emulators can process billions of cycles in a relatively short time, on smaller designs, the limitations may hinder the benefits of the fast speed. A simulation session of one hour may lead to higher productivity than an emulation session on the same design running in 10 seconds. In an eight-hour day, a verification engineer can run more design iterations, including compilation-execution-debug, than with emulation.

The best use of hardware emulation is for system-level debug and integration, especially with embedded software. Emulation can find almost all design bugs uncovered in the hardware via simulation, as well as in the system-on-chip's (SoC) embedded software.

Emulation allows for verifying the robustness of a design and helps optimize the design for improved performance.

9. Emulation needs long-term upper-management commitment to build an infrastructure.

While easier to operate than earlier generations, hardware emulation is capital equipment that requires training in order to use it. That takes management-level commitment. Having a senior-level champion for hardware emulation is a must. While it has a number of benefits to satisfy upper management's long-term objectives, such as its ability to be used throughout the SoC development cycle, they need to be communicated by someone who understands those objectives.

Emulators are costly.

Pricing has been a knock against hardware emulation, even though it is the best verification tool to thoroughly test the functionality of a SoC design. Over time, the cost of ownership has been reduced from more than one dollar per gate to less than a penny per gate. Part of the ROI analysis must be its improved system reliability and the usage model, multi-user and remote-access capabilities, and its ability to reduce the number of re-spins. Because emulation is considered a high-value resource able to alleviate unnecessary risk, it is considered cost-effective with a justifiable ROI.

Looking for parts? Go to SourceESB.

Source URL: http://electronicdesign.com/test-measurement/10-best-verification-practices-hardware-emulation