

In My Opinion

 [Print](#)  [Email](#)  [Bookmark](#)

Time to reconcile the design/verification divorce

By Kathryn Kranen, Jasper Design Automation

05/20/08

In the 1980s, designers verified their own or their peers' designs by composing specific directed tests (stimuli) to exercise and check all design behaviors as they were developed. As design complexity grew, aided by synthesis and faster simulation, the engineering effort required to develop directed tests became an overall schedule bottleneck. The EDA industry responded with constrained-random simulation, which harnessed compute power to automatically generate abundant verification stimulus.

As part of the team at Verisity that brought constrained-random simulation, or "testbench automation," to market during the late 1990s, I recall the key adoption challenges we faced. Constrained-random testbench development required object-oriented software programming skills that most RTL designers lacked. Testbench development took weeks or months. Users were required to learn a new and proprietary verification language.

The answer to overcoming these adoption challenges was to persuade system-on-chip (SoC) design teams to establish separate verification teams of engineers with object-oriented software backgrounds -- in effect, to parallelize design and verification. Unfortunately, constrained-random simulation, for all its benefits, brought about the "divorce" of design and verification.

In the early 2000s, RTL designers and verification engineers became more and more specialized. RTL designers shifted their focus to IP integration and deep submicron design challenges, and sophisticated power management techniques. Meanwhile, verification engineers grappled with soaring complexity. Sophisticated testbenches contained more lines of code than the designs under verification, and all of those lines of code had to be debugged.

Since stimulus was generated randomly, coverage methodologies and reporting tools were required to track verification progress. New EDA tools designed to help manage the massive data associated with constrained-random simulation emerged. These tools required learning and resource investment.

Over the years, verification teams grew and grew, often outnumbering design teams by 2 to 1, or even more. Today, verification is said to represent 70% of overall chip development costs.

Too much detachment

While verification specialists bring indisputable value to projects, the pendulum has swung too far towards detachment of design and verification. Many designers now bear too little responsibility for assuring the quality of their RTL. They must stitch together thousands of lines of RTL code and rush it into the hands of the verification team after performing only the most rudimentary directed tests. Early schedule milestones for delivering rough "place-holder" RTL to physical design teams create pressure on designers to code *all* of the RTL before verifying *any* of it.

The two main consequences of the design/verification divorce are a lack of cross-functional awareness and incentives, and increased bug lifespans -- that is, the number of days from when a bug is introduced into the RTL until it is corrected. The result: increased costs and schedules.

The ideal design process would employ verification-aware design, as well as design-aware verification. Engineers who have performed only either design or verification throughout their careers are missing half of the big picture. An "over the wall" syndrome exists. Each team optimizes its own tools and processes and throws problems over the wall to the other team.

Pushing bug-finding responsibility to the verification team dictates that most bugs aren't found until a critical mass of RTL (usually from multiple designers) and testbench code (developed by the verification engineer) has been integrated and debugged. Unfortunately, the longer the lifespan of a bug, the more costly are its ripple effects.

When a bug has a lifespan of weeks or months, new code will be developed on top of the buggy code, creating layered dependencies that are difficult to repair. The later a bug is found, the longer it will take to debug its root cause, as the code will no longer be fresh in the RTL designer's mind. A single bug might manifest itself in dozens, or even hundreds of simulation failures, each requiring investigation and closure.

Bugs not detected until late in the physical design cycle may cause massive rework. The most costly bugs escape to silicon, may not be discovered until another design team reuses the legacy RTL. Sophisticated teams track the lifespan of every bug and strive to reduce the average. Clearly, designers are in the best position to eliminate bugs in their infancy – and to reduce the chances of creating bugs in the first place.

Improving RTL

The answer to taming the verification beast and reducing overall schedules is to raise the baseline quality of RTL before it leaves the hands of the original designer.

Unfortunately, the drive to get designers to raise the baseline quality of their code must be balanced against two basic realities – designers are the keystone to chip creation, and initial design creation gates practically every other task. Therefore, any approach to improving the initial quality of the RTL must be:

- Low overhead to learn and implement
- Robust and intuitive to use
- Fast and easy to drive to completion
- Worth the effort, over and above the benefit to the RTL quality alone

New EDA solutions are needed to help designers to raise the baseline quality of their RTL. What would be the criteria for the ideal solution?

To begin with, learning and setup overhead must be kept low. The new solution should not require the user to learn a new language or to spend weeks setting up a verification environment. And it should not require complete input constraints, as some current solutions now do. Incidentally, these are the two primary reasons why the assertion-based formal verification (ABV) flow has seen slow traction in the world of designers.

To be applied very early in the RTL design phase, the ideal solution should be able to operate on partial or incomplete RTL, unlike simulation. The solution should overcome controllability and observability challenges that are prevalent with simulation. And, for this solution to offer the highest “return on effort” for busy designers, it should not focus on exhaustive completeness (as is the focus of the standard assertion-based formal verification methodology), but rather on interactive and immediate feedback.

To run quickly and drive to completion, the ideal solution would need the ability to quickly verify fundamental design decisions – far more than protocol compliance or structural design rules. Lastly, the ideal solution would need to be able to leverage the work performed by the designer throughout the design cycle, from architecture to first silicon, and to aid in knowledge communication throughout the design team.

Formal verification is a technology that is uniquely capable of delivering early RTL verification, even before RTL is complete, of providing an “executable specification” that unifies RTL design and verification, and of spanning multiple phases of the design cycle.

Expect more from formal verification!

Kathryn Kranen is president and CEO of [Jasper Design Automation](#). She previously served as president and CEO of Verisity Design, Inc., and was vice president of North American sales at Quickturn Systems. She started her career as a design engineer at Rockwell International, and later joined Daisy Systems, an early EDA company.

Related articles

[Constrained random test struggles to live up to challenges](#)

[Formal property checking – what the users say](#)

[Formal property checking – what the vendors say](#)



Add Comment - please [log-in](#) to comment

SCDsource newsletter subscribers may post a comment - [Register for free!](#)

[Back to Home Page](#)