



Put "phrases in quotes"

[News & Analysis](#) | [SolutionSource™](#) | [Academic Insights](#) | [Expert's Corner](#) | [Industry Access](#) | [Classifieds](#) | [About Us](#)

You are here: [Home Page](#) / [News & Analysis](#) / [Formal – rocket science or mainstream technology? A deeper look](#)

Thursday, May, 6th 2010

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

In My Opinion

Formal – rocket science or mainstream technology? A deeper look

By Michael Siegel

07/15/09

Is formal verification only for experts? This is probably one of the most debated questions in functional RTL verification in recent years. The answer to this question is critical to anyone considering adopting formal verification because it determines the necessary investment in skills and technology – and the return on investment (ROI) that companies can expect.

A recent SCDsource [survey of 16 formal user companies](#) – spanning both experienced users and recent adopters – gives the answer: “It depends on what formal is used for.” The survey data shows that formal use has diversified considerably since the introduction of standardized assertion languages, which in turn has changed the adoption pattern of formal, turning it into a mainstream technology.

Historical use of formal

Leading companies such as HP, IBM and Intel used – and still use – formal methods to crack verification’s real hard nuts. These are the parts of designs where simulation alone does not deliver sufficiently high confidence results. By definition, these parts have high logical complexity and intricate functionality and thus are hard to verify – irrespective of the technology used.

Exhaustive formal verification of such hard nuts is itself hard. The user has to struggle with the capacity and performance limitations of the formal tool, necessitating the use of sophisticated techniques such as abstractions, expert tool switches, decomposition, assume/guarantee reasoning, and so on – concepts that puzzle or even scare many simulation users. These techniques require specialized skills and a lot of experience.

Effort and skills requirements increase by orders of magnitude when users try to push the limits of formal – this is and will remain the domain of experts. This history has given formal the reputation of being powerful but hard to learn and hard to use – an expert-only technology. In turn, this has led to a perception in many companies that the learning effort and risk involved in adopting formal outweigh its benefits.

What has changed?

The survey gives us a look at what has changed in the employment of formal. To the best of my knowledge, it is the first published detailed analysis of the actual employment of formal methods based on data from *user* companies. The analysis is based on 17 use cases that leverage formal technology, of which exhaustive verification – the historically predominant use case – is but one (see figure 1).

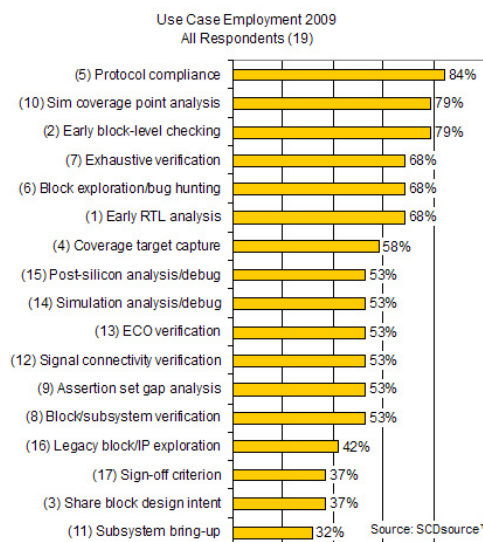


Figure 1: The broad diversity of formal use today (Source: SCDsource)

The bar chart shows that formal technology is used today for a broad and diverse range of RTL analysis and verification tasks. Most interestingly, the majority of the formal use cases are *not* about pushing the limits of formal technology – they focus on faster results for standard, mainstream verification tasks such as automatic RTL code analysis, early block-level checking, and analysis of coverage gaps in simulation.

Our users report that these use cases require little learning and no specialized formal skills – in fact, they tell us that these use cases are often easier to learn than writing good testbenches, require less effort, and deliver earlier better results than simulation.

The diversification of formal applications has changed the typical user profile. Of the top six formal verification use cases, three are clearly used by designers. Designers employ formal to increase baseline quality while the RTL is still under construction or immediately after coding, before any testbench is available. This eliminates the effort to develop throw-away block-level testbenches, while reducing costly late iterations and bug fixes. It also provides block integration conditions that are reused later to speed integration verification and system simulation bring-up. What everybody knows in principle – fixing bugs earlier is orders of magnitude cheaper than fixing them later – is now both possible and best practice through the use of formal technology. And these are not formal's rocket science applications.

Clearly, formal has evolved from an expert-only technology into mainstream use to improve a multitude of analysis and verification tasks. And, as the survey shows, the employment of all 17 use cases will significantly increase – seeing adoption and deployment across design, verification and integration teams, many of which do not have a historically strong formal verification background – let alone the oft-cited “PhD in formal.”

In fact, the adoption pattern for formal today is similar to that of testbench automation technology in the last decade. Those new to testbench automation did not start employing it by developing sophisticated, aspect-oriented, constrained random testbenches to verify the most intricate designs in an effort to reach 100 percent functional coverage. Rather, they employed testbench automation capabilities in a gradual fashion, learning more advanced concepts and applications along the way. The same is true of formal now. The survey shows that those new to formal start with the low-hanging fruit – simple use cases – and adopt more advanced applications along the way. Their focus is not on learning the rocket science applications, but on fast results that benefit the overall verification and minimize initial learning and adoption effort.

The pivotal role of standard languages

The standardization and broad deployment of SystemVerilog assertions (SVA) and Property Specification Language (PSL) have catalyzed the emergence of new formal use cases through the linking of formal and dynamic verification, and the reuse of assertions across verification technologies. The adoption of assertions in dynamic verification in combination with simple formal use cases has considerably eased the learning curve for becoming productive using formal assertion-based verification. In fact, users tell us they became productive in days.

Furthermore, these standard languages have dramatically reduced the cost of switching formal tools. Users are no longer locked into one point tool with legacy assertions written in a proprietary language – removing another adoption hurdle and risk.

Take-Aways

Formal technology is no longer only an on-top-of-simulation task to improve verification quality. Its key strengths are the numerous applications in which it delivers better results with less effort compared to simulation. Formal and dynamic methods have coexisted for a long time but, driven by formal's diversification and the change in who uses it, we now see the combined use of formal and simulation that exploits their complementary strengths to improve overall verification productivity and quality.

In the article “Formal Property Checking – what the users say” [Viresh Paruthi](#), technical lead for formal verification at IBM Systems Group said: "In my mind, the debate over the value proposition of formal is over. It's clear from everybody's experience that this is a technology that should be leveraged to improve design and verification productivity."

The recent SCDsource user survey shows that this view has become a general consensus. And the learning path for new adopters is clear: start with the simple use cases and expand the application domain while you go.

“Rocket science or mainstream technology?” You decide.

Dr. Michael Siegel is director of product marketing at [OneSpin Solutions](#). He has a Ph.D. degree in computer science from Kiel University, Germany.

Further Reading

- Mixing Formal and Dynamic, [Part I + Part 2](#). SCDsource. B. Murray, Special Technology Report, May 2009
- [Formal property checking -- what the users say](#). SCDsource. R. Goering, February 2008
- [OneSpin advances formal assertion/RTL debug automation](#). SCDsource. B. Murray. July 2009.

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

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

[Back to Home Page](#)

All materials on this site Copyright © 2007-2009 Tech Source Media, Inc. All Rights Reserved | [Privacy Statement](#)