PANEL: Formal Verification Methods: Getting around the Brick Wall

Chair: David Dill, Stanford University

Organizers: Nate James, Prover Technology/KVO Public Relations and Shishpal Rawat, Intel Corporation

Abstract

Do formal verification tools and methodologies require a drastic overhaul to move beyond equivalence checking? Equivalence checking catches errors in synthesis and local hand-modifications to designs. However, powerful formal verification technologies are emerging to combat "behavioral" errors, which represent today's biggest verification problems. Nonetheless, formal verification experts are split on how formal tools should adapt to this challenge. Some of our panelists feel that designers can sufficiently benefit from new formal verification technologies by making incremental changes to current methodologies. Others, however, argue that major changes are required to reap meaningful benefits from these new technologies. Just how much change is enough, what is the capacity of our current tools and what is limiting the full deployment of FV technology?

Our panel of experts, consisting of users, tool providers, and core engine builders, will answer these challenging questions. The panel will debate these issues while discussing real life examples from the user base. They will provide a perspective of how the progression of technology will bring the real promise of formal verification to the user base.

Position Statements

Gérard Berry
Esterel Technologies

Thanks to continuous progress in Boolean and arithmetic computation engines, designers can now use formal verification techniques to validate their designs. These techniques systematically explore design behavior and especially corner cases in a way inaccessible to human reasoning. They are applicable to module or module interconnection verification at various abstraction levels. They can automatically check dynamic properties of designs. They can also analyze the true behavior coverage of test suites, fully dealing with complex concurrent component interaction coverage in a way that traditional code coverage cannot reveal.

The current design flow mostly operates at RTL-level and can be called Design for Synthesis. Since many existing verification tools rely on internal RTL descriptions, they are immediately usable within this flow. However, we think that matching the correctness constraints for larger future designs will require one more step up, Design for Verification, where automatic techniques will play a major role. The current HDLs may rapidly run out of steam since they work at too low an abstraction level to fully optimize the use of verification engines. New or enhanced HDLs will be needed to help the designer by providing higher behavioral description levels, simpler paths to synthesis, and built-in support for verification.

Limor Fix
Intel Corporation

Formal Property Verification, targeted at either catching "behavioral" errors or for certifying the correct behavior of a design, has been used in our industry, and in particular at Intel, for almost ten years. Our findings indicate that formal property verification is a validation vehicle that brings value not achievable in simulation. Complex control and data-intensive circuits that cannot be exhaustively checked by simulation can achieve high level of coverage using formal verification methods. However, the deployment of formal property verification is still limited due to (1) lack of specification and (2) limited capacity of FV technology.

Both roadblocks can be overcome by changing the design methodology. Unfortunately, small (large) changes will yield small (large) benefits. For example**,** if designers will specify embedded assertions in the RTL and designs employ parameterized RTL blocks and arrays, then these incremental changes only partially solve the lack of specification and the limited capacity of the tools that design teams continuously face.

While we fully support these changes, we hope to utilize the full power of the FV technology by allowing more drastic changes to the front-end design methodology. For example, by unifying the efforts of developing Cluster-Test-Environments and Formal Specification, or by developing high level models which are written in a formal language which can serve as formal specification for the RTL.

Harry Foster
Verplex Systems, Inc.

Proving functional correctness using formal techniques (e.g., model-checking) has established its worth with leading-edge companies. Yet, transitioning the technology into mainstream design environments (beyond the bounds of research and early adopters) has been limited by the following factors: (a) lack of a universally accepted formal property language, (b) lack of commercial tools—until recently, (c) lack of methodology guidelines for effective use (e.g., addressing scalability), and (d) unpersuasive ROI—*what's the gain for the pain?* The lack of a formal property language has recently been addressed through the efforts of the Accellera standardization organization. Likewise, commercial tool offerings are proliferating as the technology matures. Yet, to clearly demonstrate the value of formal or semi-formal techniques requires a change in design methodology. That is, a formal interface specification is required for any individual block we wish to verify. Many argue that in real designs, the block-level interfaces are changing too rapidly to maintain up-to-date interface specifications. However, a counter-argument is that complexity of block-level interfaces could be reduced if serious

effort were placed on developing and maintaining interface specifications. The additional *gain for the pain* is an improved simulation process (e.g., the block-level interface specifications can serve as monitors during system integration).

Rajeev Ranjan
Real Intent, Inc.

The increasing high premium of functional correctness in designs with growing complexity demands that verification needs to be concurrent with the design process. Formal verification can facilitate this objective while making only incremental changes to the mainstream HDL based design methodology. In this incremental methodology, automatically identified assertions as well as user specified design assertions within RTL are exhaustively analyzed before testbenches are developed. The formal analysis of these assertions can either confirm desired behavior or eliminate undesired behavior as RTL is being developed.

While the capacity of formal analysis is substantially lower than completely constrained simulation, use of innovative methodologies for adequate and easy constraint specification and for seamless assume-guarantee reasoning enables meaningful formal analysis to obtain 100% confidence for assertions verified.

The biggest barrier to adoption is the absence of clear metrics for performance and quality of the design process. It ranges from cobbling up RTL and throw it over the wall for the verification engineers to developing proper specification and developing RTL from it. A clean guideline on enforcing a quality sign-off criteria for every designer would proliferate usage of formal techniques. The tool vendors can help by reducing the designer's effort needed to get tangible benefits from formal techniques by providing easy specification and debugging methods.

Gunnar Stålmarck
Prover Technology

Although formal verification has made progress in recent years, several factors continue to limit its use in mainstream design flows. Proof of concept remains a problem for the industry, as does a lack of standardized FV languages. Also, one cannot overlook insufficient proof power when considering the reasons why formal verification has not progressed beyond standard applications such as equivalence checking and model checking.

If proof power were greater today, say 100 times greater, formal verification would already be a critical part of every design flow. Instead, designers have been forced to bridge the gap between proof power available and proof power needed by adapting design flows to apply formal verification only where absolutely critical. Increasing the proof power of formal verification tools is the key to unlocking formal verification's potential as a powerful high-level tool for designers. Once the issue of proof power is addressed, revolutionary new standards and methodologies will emerge and be adopted for use on a large scale. Although both incremental and revolutionary paths will be explored, the benefits of revolutionary new methodologies with increased proof power will prevail, enabling adventurous new designs, faster time to market, and complete verification coverage.

Curt Widdoes
0-In Design Automation

Formal verification technologies today are vastly more thorough than simulation in finding corner-case problems in RTL designs. However, hardware design teams have historically been reluctant to abandon established methodologies until they have no choice. In the case of verification, mainstream design teams will continue to rely on simulation and will adopt formal technology incrementally only where it can reduce verification time and effort within the existing simulation flow.

Assertion-based verification (ABV) is an effective way to bring formal verification into the traditional simulation flow. In ABV, the design team instruments the RTL design with statements of design intent (assertions), and simulates with the assertions in place. Then, they use formal technology to directly target the assertions. Constraints, which are necessary for formal verification, are simply assertions that check for illegal interface behavior; in ABV, constraints are simulated along with all the other assertions.

Assertion-based verification fits into the simulation flow with minimal disruption, and the resulting assertions improve the effectiveness of simulation. By enabling formal technology to be used in the simulation flow, ABV increases the thoroughness of verification and decreases the verification time and effort. As a result, ABV is being used today by many companies to verify large designs with complex interfaces. Gradually, as users achieve success with incremental use of formal verification, they will adopt new languages and flows that are designed to maximize its benefits. Eventually, formal verification will play as large a role in mainstream verification as simulation does today.