Formal Verification – Is It Real Enough?

Yaron Wolfsthal
IBM Haifa Research Laboratory
ISRAEL
wolfstal@il.ibm.com

Rebecca M. Gott
IBM Systems and Technology Group
Poughkeepsie, NY, USA
gott@us.ibm.com

Categories and Subject Descriptors
B.6.3 **[Logic Design]:** Design Aids – verification.

General Terms
Design, Measurement, Verification.

Keywords
Formal Verification, Functional Verification.

1. INTRODUCTION

While Formal Verification (FV) of logic designs has been described in an industrial context for over a decade, it has not yet become a mainstream methodology. Our purpose in this report is to summarize a body of experience in the application of industrial-scale FV. We aim to present our insights and recommendations to practicing engineers and managers, who wish to evaluate the inclusion of FV as a part of their design methodology. In doing so, we hope to contribute to the understanding of the full potential of FV, based on our positive experience with this paradigm (cf. [1-2]). We focus on providing practical information about the process of FV as possible within the limited scope of this text[1]. Our analysis is based on observations and data collected from the application of over 100 projects across IBM and customers[2].

2. SCOPE

For the foreseeable future, verification will continue to rely on simulation. FV, in turn, can complement simulation, and in certain cases can take a central role – e.g. in control-dominated logic with limited datapath function. A recent IBM report described a methodology where up to 40% of the logic in a typical design project can be subjected to formal analysis with careful planning [3]. This investment is justified by the fact that the logic bugs detected by FV are complementary to those found by simulation, hence providing a worthy return on the investment.

[1] We limit our discussion to FV techniques based on Model Checking.

[2] The FV tool used in the referenced projects is RuleBase PE [2] with PSL.

Copyright is held by the author/owner(s).
DAC 2005, June 13–17, 2005, Anaheim, California, USA.
ACM 1-59593-058-2/05/0006.

3. BENEFITS EXPLAINED

Two highly acclaimed benefits of FV are that it increases quality and shortens time-to-market. In closer analysis, the underlying drivers of these high-level attributes are early availability, higher coverage, and the enablement of integration:

1 Early Availability – FV can typically start earlier than simulation, as soon as the logic is compiled, since it requires a very limited setup cost relative to simulation testbenches.

2 High Coverage – inherently, FV can systematically cover large state spaces, or significant parts thereof, thereby allowing the detection of bugs which would have been difficult to manually target with simulation.

3 Enabling Effective Integration – building on the above points, when FV is used at the unit level, system simulation time is significantly reduced, and stabilization is expedited.

4. HINDERING FACTORS

The main technical limitation of FV is well known – an exponential state space explosion prevents the application of FV as a comprehensive system-level verification solution. Indeed, FV remains a unit level verification tool, capable of addressing logic models of limited size. The actual limits depend on the design at hand, the technique used, and the verification goals (proof of design properties vs. their falsification). As a yardstick, FV is used in our projects for proving properties of design modules reaching up to few thousands of state-holding elements, and semi-formal techniques – focused at falsification –applied to modules whose size is about an order of magnitude higher.

A second hindering factor in the application of FV is that engineers and managers often exhibit reluctance to use them. This, in our experience, is the combination of three issues. First, formal methods have a (false) reputation of being difficult to use. Secondly, the demonstration of the value proposition requires an upfront investment. Lastly, many design houses are comfortable with the "good enough" nature of simulation, and readily forego the potential benefits of FV, in favor of the familiar simulation-only design methodology. The ongoing flow of reports on the successful application of FV in the field appears to increasingly, steadily, influence design teams to seriously consider the deployment of FV. In the accompanying presentation, we aim to contribute to this cause by quantitatively demonstrating successful application of FV in our projects.

5. BUG CLASSIFICATION

The nature of bugs found by FV is exemplified below, following a study of an IBM team who reviewed all bugs found by FV during 12 months (several 100's in all). This study has yielded the following bug classification, which sheds some light on the impact FV has on a design project:

Bugs Found Due to Schedule Advantage: as FV can typically start earlier than unit simulation, some bugs discovered by FV early in the verification cycle are simply due to the fact the FV was the first to evaluate the logic. The resolution of this class of bugs helps provide a more stable model for the start of simulation.

Hole in Simulation Coverage: Once simulation has started, bugs discovered by FV may expose holes in simulation coverage. This class of bugs is important as it drives appropriate improvements to the simulation environment so that it will provide better coverage in the larger system simulation context.

True Corner Case Bugs: As mentioned earlier, FV can expose corner case bugs that are extremely hard to hit in simulation, even with carefully planned simulation environments developed with advanced tools. With that, FV complements simulation.

Performance Bugs: These are cases where computation results are intact, while the performance fails to meet requirements. This is an important class of bugs since, especially in the area of high-performance logic, performance IS function and performance related problems have been known to be a reason for silicon respins.

Impossible Bugs: Some bugs found by FV may be deemed impossible (i.e. they arise from a sequence of input signals considered illegal for the neighboring block). The importance of those bugs is in that they encourage the designer to consider and provide a fix. This provides a protection against cases where behavior on the interface may change at some future point in time, thus allowing the currently impossible behavior and exposing the bug.

6. RECOMMENDATIONS FOR DEPLOYMENT

Some recommendations on how to integrate FV as a stable part of the overall verification process are listed below.

- Size the FV work at the start of the project, considering simulation coverage schedules, logic availability, and logic complexity. Staff the FV team accordingly.
- Commitment needs to remain throughout the project cycle, including
  - regressing FV as the design is modified, and
  - continued communications with the design team to ensure FV assumptions are current with the design
- To better gauge FV's impact, track the type of bugs found as discussed in Section 5.
- The FV staff across projects should follow a common form of documenting their work, and hold regular reviews with the design and simulation teams. The FV documentation should be incorporated into the larger verification plan for the project.
- Design teams need to commit to providing adequate design documentation for the FV staff to work from. It greatly slows the verification process when the FV engineer is forced to independently discern the function of the logic.
- A formal FV tapeout criterion should be defined for projects with FV requirements.

3. SUMMARY

Formal Verification, in our experience, is a realistic means to successfully address the growing complexities of contemporary design. However, as the preceding discussion suggests, it is no silver bullet. Introducing FV into the design flow is a strategic decision that requires investment in engineering resources (training and methodology adjustment) as well as support and commitment from management. When appropriately applied, FV is a powerful verification vehicle which contributes to increasing design quality and shortening time to market, with a notable return on the investment in engineering resources.

REFERENCES

[1] Ben-David, S., Eisner, C., Geist, D., and Wolfsthal, Y. Model Checking in IBM, Formal Methods in System Design 22, 2 (2003), 101-108.

[2] See pertinent experience reported at the RuleBase homepage http://www.haifa.il.ibm.com/projects/verification/RB_Homepage

[3] Ginzburg, Y., Using Sugar at the IBM Haifa Design Labs, PSL/Sugar Consortium Meeting at DAC'03, http://www.pslsugar.org/papers/ABV-in-IBM-Haifa.pdf