Microprocessor Verification Based on Datapath Abstraction and Refinement

Zaher S. Andraus, Mark H. Liffiton, and Karem A. Sakallah

Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI 48109-2122

{zandrawi,liffiton,karem}@eecs.umich.edu

Counterexample Guided Abstraction Refinement (CEGAR for short) has been shown to be an effective paradigm in a variety of hardware and software verification scenarios. Originally pioneered by Kurshan [7], it has since been adopted by several researchers as a powerful means for coping with verification complexity. The wide-spread use of such a paradigm hinges, however, on the automation of its abstraction *and* refinement phases. Without automation, CEGAR requires laborious user intervention to choose the right abstractions and refinements based on a detailed understanding of the intricate interactions among the components of the design being verified. Clarke et al. [3], Jain et al. [5], and Dill et al. [2] have successfully demonstrated the automation of abstraction and refinement in the context of model checking for safety properties of hardware and software systems. In particular, these approaches create a smaller abstract transition system from the underlying concrete transition system and iteratively refine it with the spurious counterexamples produced by the model checker. The approaches in [3] and [5] are additionally based on the extraction of unsatisfiability explanations derived from the infeasible counterexamples to provide stronger refinement of the abstract model and to significantly reduce the number of refinement iterations.

All of these approaches are examples of *predicate abstraction* which essentially projects the concrete model onto a given set of relevant predicates to produce an abstraction suitable for model checking a given property. In contrast, we describe in [1] a methodology for *datapath abstraction* that is particularly suited for equivalence checking. In this approach, datapath components in behavioral Verilog models are automatically abstracted to uninterpreted functions and predicates while refinement is performed manually using the ACL2 theorem prover [6].

The use of (near)-minimal explanations of unsatisfiability forms the basis of another class of abstraction methods. These include the work of Gupta et al. [4] and McMillan et al. [8] who employ "proof analysis" techniques to create an abstraction from an unsatisfiable concrete bounded model checking (BMC) instance of a given depth.

In this talk we explore the application of CEGAR in the context of microprocessor correspondence checking. The approach is based on automatic datapath abstraction as in [1] augmented with automatic refinement using minimal unsatisfiable subset (MUS) extraction. One of our main conclusions is the necessity of basing refinement on the extraction of MUSes from both the abstract and concrete models. Additionally, refinement tends to converge faster when multiple MUSes are extracted in each iteration. Finally, localization and generalization of spurious counterexamples are also shown to be crucial for refinement to converge quickly. We will describe our implementation of these ideas in the Reveal system and discuss the effectiveness of the various refinement options in the verification of a few benchmarks.

REFERENCES

[1] Z. S. Andraus and K. A. Sakallah, "Automatic Abstraction of Verilog Models", In Proceedings of 41st Design Automation Conference 2004, pp. 218-223.

[2] S. Das and D. Dill, "Successive Approximation of Abstract Transition Relations" in 16th Annual IEEE Symposium on Logic in Computer Science (LICS) 2001.

[3] E. Clarke, O. Grumberg. S. Jha, Y. Lu and H. Veith, "Counterexample-Guided Abstraction Refinement," In CAV 2000, pp. 154-169.

[4] A. Gupta, M. Ganai, Z. Yang, and P. Ashar, "Iterative Abstraction Using SAT-based BMC with Proof Analysis." In Proc. of the International Conference on CAD, pp. 416-423, Nov. 2003.

[5] H. Jain, D. Kroening and E. Clarke, "Predicate Abstraction and Verification of Verilog," Technical Report CMU-CS-04-139.

[6] M. Kaufmann and J. Moore, "An Industrial Strength Theorem Prover for a Logic Based on Common Lisp." IEEE Transactions on Software Engineering 23(4), April 1997, pp. 203-213.

[7] R. Kurshan, "Computer-Aided Verification of Coordinating Processes: The Automata-Theoritic Approach," Princeton University Press, 1999.

[8] K. L. McMillan and N. Amla, "Automatic Abstraction without Counterexamples." In International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS'03), pp. 2-17, Warsaw, Poland, April, 2003, LNCS 2619.