Continued Relevance of Bit-Level Verification Research

R. Brayton, N. Een, A. Mishchenko
Berkeley Verification and Synthesis Research Center
EECS Dept., University of California, Berkeley

Introduction

Formal verification methods are making great strides. Motivated by the improvements in the core technologies and successful industrial applications, a number of new areas have emerged. These are exemplified by the list of topics discussed at FMCAD 2010: SMT, verification of RTL designs, software, systems on the chip, protocols, concurrent systems, etc.

With the shifting of attention to these new areas and applications, there seems to be a tendency to believe that bit-level verification is either

a) a solved problem; or
b) a hopelessly difficult one

and therefore not worth the effort of continued research. This goes hand-in-hand with the reduction of governmental and institutional funding for these core areas in formal verification.

We believe that bit-level verification is not a solved problem, nor it is a hopelessly difficult one in practical situations. The peculiarity of this area is that solutions to the challenges are often found not in a new theoretical discovery, but in a better implementation of old algorithms.

A well-known example is the 2001 paper by the group of Sharad Malik at Princeton, appropriately titled "Chaff: Engineering an Efficient SAT Solver" [7]. This has become one of the most cited papers on formal verification and represents a major break-through, even though the contribution of this paper is not a new theoretical formulation. The contribution was in the exploitation of the synergy between algorithms and data-structures, most of which were known before. Yet this was a break-through in SAT solver technology, and was responsible for stimulating research that led to the majority of the recent successes of formal verification.

After almost 10 years of significant progress, we believe it is not the time to give up on improving bit-level verification for the following reasons:

1. Emergence of new verification engines

At the Hardware Model Checking Competition (HWMCC) affiliated with CAV 2010, a new complete proof-method [1] was proposed by Aaron Bradley from University of Colorado, Boulder. His one-engine solver (IC3) based on this new method came in third in the competition and lost, by a small margin, only to mature multi-engine solvers based on a variety of techniques, including sequential synthesis, induction, interpolation, BDD-based reachability, etc. The same method, unmodified, often detects deep bug-traces, and often may be faster than SAT-based bounded model checking that starts from the initial state. It is probably fair to say that IC3 is better than any single engine used within the multi-engine solvers that beat it and almost as good as the combined multi-engine methods.

Beyond this, we see Bradley's method, which we call *property directed reachability* (PDR), as a paradigm for discovering new proof methods or improving the old ones. Indeed, PDR has elements of induction, interpolants, and over-approximate reachability, which exposes a number of similarities that can be explored. It gives encouragement that there is much room for significant innovation in the core proof-technologies for bit-level verification.

2. The need for specialized SAT solvers

In the last few years, the SAT competitions have seen minor changes in the SAT solver technology, largely limited to better tuning on difficult benchmarks and new CNF preprocessing schemes. Because of the lack of major advances, the SAT research community has largely shifted its attention to SMT, quantified Boolean formulae, parallel SAT, SAT on FPGAs, etc.

However, we believe that core SAT solver technology is not saturated, as evidenced by our experience with several experimental implementations developed recently, which motivate the need for developing *application-specific* SAT solvers.

One example was detecting sequential equivalences in large hardware designs as implemented in ABC [2]. In this problem, it was necessary to solve a large number of closely-related, relatively-simple, incremental SAT problems. A specialized light-weight circuit-based SAT solver was developed that worked directly on the AIG structure. By replacing a fine-tuned implementation of MiniSAT [3] with this new solver, a substantial speedup (over 5x on large designs) was observed. A similar experience is in our implementation of PDR, which carefully orchestrates its various steps using a single SAT instance, repeatedly called through the incremental interface.

Another occasion for an application-specific SAT solver is the problem of don't-care-based resubstitution [6]. We have an implementation which works quite well on smaller problems but is too slow for large designs, say, those of over a half million logic gates. On the other hand, we have seen that there is a lot left on the table in terms of optimization potential. We expect that developing a specialized SAT solver, possibly similar to an ATPG engine, could lead to a speedup in this important application used in both synthesis and verification.

In all of these examples, a common source of improvement is that MiniSAT requires converting the circuit into CNF and setting up complex data-structures. This is not needed if a sequence of easy, incremental problems can be solved directly on the circuit. Additionally, a solution can lie in the ability to re-think the problem formulation and customize little-known features of the SAT solver, such as an incremental interface and proof logging..

We believe there are many opportunities where such an approach can be useful. The development of application-specific SAT solvers and application-specific use of decision engines at the bit-level is an interesting research direction, which may lead to speeding up core verification engines, making them scale to larger problems, and enhancing their performance on specific types of verification problems.

3. Exploiting problem structure

Most the problems encountered in formal verification are either circuit-based or can be viewed and formulated as such. A circuit can be a source of important information about the problem, but this is rarely exploited in present-day bit-level verification engines. For example, the majority of verification engines, such as BMC, interpolation, reachability analysis, convert the circuit into a CNF or BDD form. After this, structural information is lost and most methods make no attempt to exploit the circuit structure.

Meanwhile, there are several possible ways to make use of the circuit structure: (a) guide decision heuristics in circuit-based SAT, (b) derive circuit-based invariants to facilitate proofs. In addition, there is a lot of structural information lost that comes from the higher levels of problem specification, such as word or bit groupings, existence of arithmetic and other special operators. In some applications, structural information has been lost due to legacy code or hardware. An interesting problem is how to reverse-engineer the circuit structure to re-discover hierarchical or word-level information.

Some types of verification problems are easier in practice because they contain similar sub-structures. An example is equivalence checking, where two versions of the same design are compared. These problems can be either combinational (CEC) or sequential (SEC). Although SEC is PSPACE-complete [4], in practice almost 80-90% of the SEC problems resulting from logic synthesis can be solved by general purpose model checkers. This is due to synthesis leaving many structures unchanged and hence many common sub-structures. For the remaining hard problems, it is possible that re-discovering lost structures may be the key to their solvability.

We know that if all lost substructures are found, then SEC becomes NP-complete. This is because they can be used to form a set of equivalences, which is an inductive invariant [5]. Similarly, the new PDR method is based on discovering a set of cubes which is an inductive invariant. Other forms of inductive invariants can possibly be explored.

Another example occurred in a problem of proving sequential equivalence between two circuits where one was changed only slightly to include circuitry to do sequential clock gating for saving power. Although the sequential structure of the circuit was modified, knowledge about the origin of the problem and a subsequent theorem [8] allowed the SEC problem to be reduced to CEC where the circuit structure was unrolled interleaving the original and synthesized circuits. This led to very deep CEC problems, which were solved by modifying a CEC engine to take advantage of the special structures of these problems. The resulting CEC engine, which was motivated by this particular use, was later used in a CEC problem derived from proving the equivalence of a prototype encryption algorithm implemented on FPGAs to a high level description.

Conclusion

Research in bit-level verification has not saturated. We believe that with renewed interest and dedicated effort, a substantial progress can be made in the following directions: (1) discovering new engines, (2) developing application-specific SAT solvers, (3) exploiting the specific structure of verification problems. Future effort in revitalizing bit-level methods should pay very close attention to the implementation details and make the most of the synergies between algorithms and data-structures.

References

[1] A. Bradley, “k-step relative inductive generalization”, 2010, arXiv:1003.3649. http://arxiv.org/abs/1003.3649

[2] R. Brayton and A. Mishchenko, "ABC: An academic industrial-strength verification tool", Proc. CAV'10, Springer, LNCS 6174, pp. 24-40. http://www.eecs.berkeley.edu/~alanmi/publications/2010/cav10_abc.pdf

[3] N. Een and N. Sörensson, “An extensible SAT-solver”. Proc. SAT ‘03 , pp. 502-508. http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat/

[4] J.-H. Jiang and R. Brayton. “Retiming and resynthesis: A complexity perspective”. TCAD’06, Vol. 25(12), pp. 2674-2686. http://www.eecs.berkeley.edu/~brayton/publications/2006/tcad06_r&r.pdf

[5] A. Mishchenko and R. Brayton, "Recording synthesis history for sequential verification", Proc. FMCAD'08, pp. 27-34. http://www.eecs.berkeley.edu/~alanmi/publications/2008/fmcad08_case.pdf

[6] A. Mishchenko, R. Brayton, J.-H. R. Jiang, and S. Jang, "Scalable don't care based logic optimization and resynthesis", Proc. FPGA'09, pp. 151-160. http://www.eecs.berkeley.edu/~alanmi/publications/2009/fpga09_mfs.pdf

[7] M. Moskewicz, C. Madigan, Y. Zhao, L.Zhang, S. Malik. “Chaff: engineering an efficient SAT solver”, Proc. DAC ’01, pp. 530-535. http://citeseer.ist.psu.edu/moskewicz01chaff.html

[8] H. Savoj, D. Berthelot, A. Mishchenko and R. Brayton, “Combinational Techniques for Sequential Equivalence Checking”, FMCAD Oct., 2010