Fast and Exact is Doable: Polynomial Algorithms in Test and Verification

Rolf Drechsler

Institute of Computer Science
University of Bremen/DFKI
28359 Bremen, Germany
drechsler@uni-bremen.de

***Abstract*—Ensuring the correctness of circuits and systems by test and verification is one of the central tasks in modern design flows. But most of the core algorithms have large run times, since the underlying problems can be proven to be NP- or co-NP-complete. Thus, there is little hope to find efficient algorithms that can solve all instances in polynomial time and space. But recently it has been shown in the context of *Polynomial Formal Verification* (PFV) that for a large class of practical relevant functions fast exact algorithms can be provided.**

In this talk recent developments in the field of PFV are shown and directions for future work are outlined. The similarities in this domain between test and verification problems are discussed. Experimental studies show that very large designs can be handled in PFV, while fast run time can be ensured.

***Index Terms*—circuit design, correctness, verification, test, formal methods, BDD**

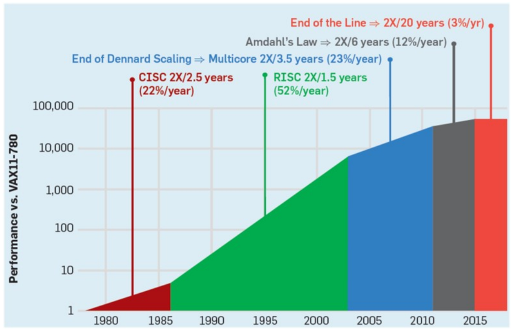

Fig. 1: Moore's law

I. Introduction

According to Moore's Law the complexity of modern designs steadily increases resulting in devices that consist of more than 20 billion transistors, and the next generation will contain up to a trillion[1]. It can be observed that the increase per year is slowing down (see Figure 1 following [1]), but in modern design flows the tools have to cope with huge instances and the underlying data structures are of utmost importance.

Along the design flow several computational hard problems have to be solved. For most of them it can be shown that they are NP- or co-NP-complete. While for logic synthesis, place and route often heuristic solutions are sufficient, in the area of test and verification exact algorithms that allow to traverse the complete search space are required.

Due to the complexity of the underlying problems exact results by algorithms with polynomial worst-case bounds cannot be expected in general. But recently the concept of *Polynomial Formal Verification* (PFV) has been introduced in [2]. Here, the core idea is to prove for a specific class of functions - adders in this case - that the complete verification process can be carried out in polynomial time and space.

Parts of this work have been supported by DFG within the Reinhart Koselleck Project *PolyVer: Polynomial Verification of Electronic Circuits* (DR 287/36-1). Furthermore, the author likes to thank Dr. Alireza Mahzoon for helpful comments and discussions.

[1]https://spectrum.ieee.org/tech-talk/semiconductors/processors/cerebras-giant-ai-chip-now-has-a-trillions-more-transistors

In this paper, the concept of PFV is reviewed. It is shown that for classes of functions, if an adequate implementation is chosen, algorithms with polynomial worst-case behavior can be designed. Next, also algorithms in the context of hard testing problems, like *Automatic Test Pattern Generation* (ATPG), are considered and similarities between test and verification are identified. Finally, directions for future work are outlined.

II. Polynomial Formal Verification

Polynomial upper and exponential lower bounds on various types of graph-based function representations are known for long (see e.g. [3]). But in this context only the representation size of the final function, e.g. for adders and multipliers, was considered. While for PFV the complete construction process has to be taken into account. It has to be ensured that the whole verification task (that might consist of symbolic simulation or backward substitution) can be carried out in polynomial time and space. It is important to notice that the efficient result not only depends on the function under consideration, but also the used architecture has a siginificant effect. For an overview on PFV and on recent results for bit- and word-level in this context see [4].

III. Polynomial Algorithms in Test

Also in the context of testing circuits, it is known for long that some of the hard problems, like ATPG, become easy, if the underlying circuit has some specific properties (see

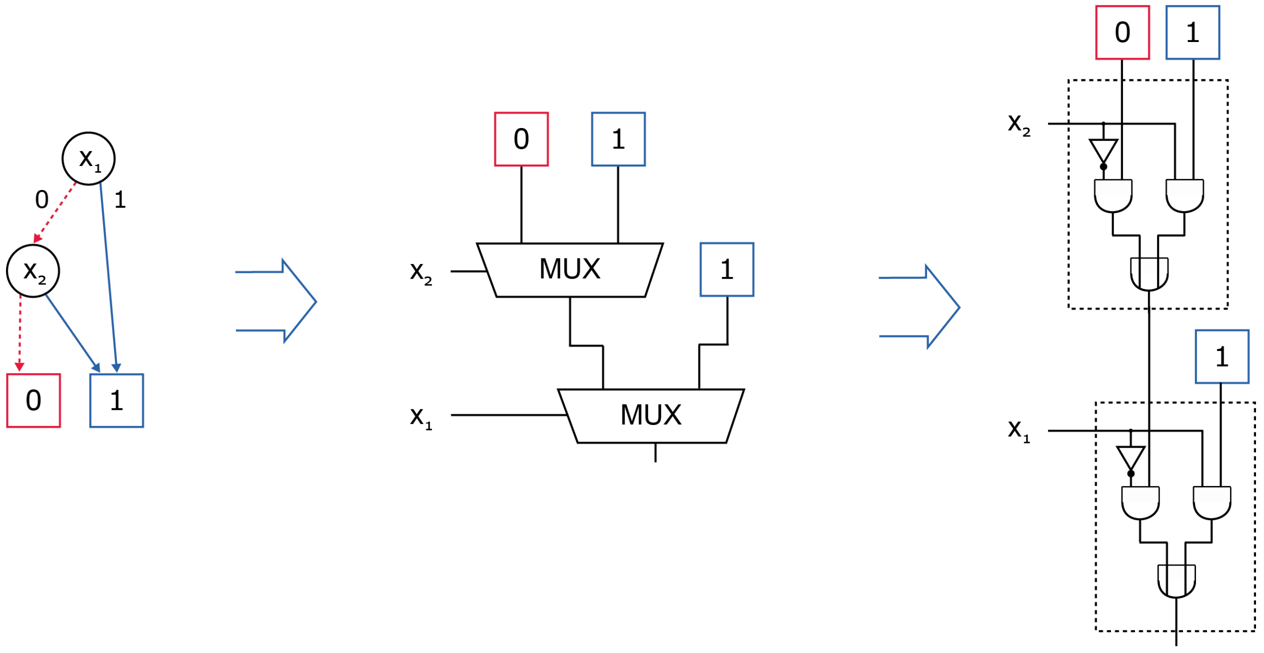

Fig. 2: BDD and resulting circuit

e.g. [5], [6]). Here, similarities between PFV and testing can be identified. For tree-like circuits and also circuits derived from a mapping of BDDs it has been proven in [7] that PFV applies. The example from this paper shows that a BDD-circuit results from a BDD by substituting each internal node by a multiplexer.

Example III.1. *For the OR-function $f_{OR} = x_1 + x_2$ the BDD is shown on the left hand side of Figure 2. The resulting circuit is shown on the right hand side, where each internal node is substituted by a multiplexer cell.*

For this type of BDD-circuits also ATPG can be done efficiently for various fault models (including the stuck-at and the path-delay fault model) [8]. A simple hardware extenstion presented as a *Design-for-Testability* (DFT) approach in [9] shows that 100% testability can be achieved. In both fields, i.e. test and verification, the problem can be simplified, if the number of reconvergent paths is limited or if the reconvegencies can be controlled, e.g. by multiplexer structures.

IV. Conclusion and Future Directions

Test and verification are central tasks in today's design flows. Since the underlying problems have high computational cost in the worst case, in general performance predictions cannot be expected. But for restricted classed of functions in combination with fitting proof engines and architecture realizations, polynomial algorithms can be designed.

While first steps in this direction are very promising, there are several directions for future work:

- There is a good understanding of BDDs (and bit-level decision diagrams in general) in the meantime. For other formal proof techniques, like SAT- and SMT-solvers, PFV is still in an early stage. This also applies to *Word-Level Decision Diagrams* (WLDDs), like BMDs or K*BMDs.
- The relation between test and verification scenarios needs a deeper understanding. Findings from one domain can then be transferred and vice versa.
- Similar to DFT also approaches for *Design for Verifiability* (DFV) would be desirable. In this context it would be good to precisely suggest architectures that the formal tools can finally handle efficiently.

Only based on a rigorous analysis of the complexity of the algorithms, next generation tools can ensure efficient run times and scalability.

References

[1] J. L. Hennessy and D. A. Patterson, "A new golden age for computer architecture," *Commun. ACM*, vol. 62, no. 2, pp. 48–60, 2019.

[2] R. Drechsler, "PolyAdd: Polynomial formal verification of adder circuits," in *International Symposium on Design and Diagnostics of Electronic Circuits and Systems*, 2021, pp. 99–104.

[3] I. Wegener, *Branching Programs and Binary Decision Diagrams - Theory and Application*. SIAM Monographs on Discrete Mathematics and Applications, 2000.

[4] R. Drechsler and A. Mahzoon, "Polynomial formal verification: Ensuring correctness under resource constraints," in *Int'l Conf. on CAD*, 2022.

[5] H. Fujiwara, "Computational complexity of controllability/observability problems for combinational circuits," *IEEE Trans. on Comp.*, vol. 39, no. 6, pp. 762–767, 1990.

[6] M. Prasad, P. Chong, and K. Keutzer, "Why is ATPG easy?" in *Design Automation Conf.*, 1999, pp. 22–28.

[7] R. Drechsler, "Polynomial circuit verification using BDDs," in *5th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT)*, 2021, pp. 49–52.

[8] B. Becker, "Synthesis for testability: Binary decision diagrams," in *Symp. on Theoretical Aspects of Comp. Science*, ser. LNCS, vol. 577. Springer Verlag, 1992, pp. 501–512.

[9] R. Drechsler, J. Shi, and G. Fey, "Synthesis of fully testable circuits from BDDs," *IEEE Trans. on CAD*, vol. 23, no. 3, pp. 440–443, 2004.