Polynomial Formal Verification of Arithmetic Circuits

Polynomial Formal Verification of Arithmetic Circuits

Alireza Mahzoon
University of Bremen
mahzoon@uni-bremen.de

Rolf Drechsler
University of Bremen/DFKI GmbH
drechsler@uni-bremen.de

the essence of knowledge
Boston — Delft

Foundations and Trends® in Electronic Design Automation

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is

A. Mahzoon and R. Drechsler. *Polynomial Formal Verification of Arithmetic Circuits*. Foundations and Trends® in Electronic Design Automation, vol. 14, no. 3, pp. 171–244, 2024.

ISBN: 978-1-63828-405-5

Foundations and Trends® in Electronic Design Automation
Volume 14, Issue 3, 2024

Editorial Scope

Foundations and Trends® in Electronic Design Automation publishes survey and tutorial articles in the following topics:

- System Level Design
- Behavioral Synthesis
- Logic Design
- Verification
- Test
- Physical Design
- Circuit Level Design
- Reconfigurable Systems
- Analog Design
- Embedded software and parallel programming
- Multicore, GPU, FPGA, and heterogeneous systems
- Distributed, networked embedded systems
- Real-time and cyberphysical systems

Information for Librarians

Foundations and Trends® in Electronic Design Automation, 2024, Volume 14, 4 issues. ISSN paper version 1551-3939. ISSN online version 1551-3947. Also available as a combined paper and online subscription.

Contents

Polynomial Formal Verification of Arithmetic Circuits

Alireza Mahzoon[1] and Rolf Drechsler[2]

[1] *University of Bremen, Germany; mahzoon@uni-bremen.de*
[2] *University of Bremen/DFKI GmbH, Germany; drechsler@uni-bremen.de*

ABSTRACT

In recent years, significant effort has been put into developing formal verification approaches by both academic and industrial research. In practice, these techniques often give satisfying results for some types of circuits, while they fail for others. A major challenge in this domain is that the verification techniques suffer from unpredictability in their performance. The only way to overcome this challenge is the calculation of bounds for the space and time complexities. If a verification method has polynomial space and time complexities, scalability can be guaranteed.

In this monograph, we propose *Polynomial Formal Verification* (PFV) of arithmetic circuits. We discuss the importance and advantages of PFV. Subsequently, we prove that PFV of different types of arithmetic circuits, including adders, multipliers, and *Arithmetic Logic Units* (ALUs) is possible. Furthermore, we calculate the exact upper-bound space and time complexities of verifying these circuits.

Alireza Mahzoon and Rolf Drechsler (2024), "Polynomial Formal Verification of Arithmetic Circuits", Foundations and Trends® in Electronic Design Automation: Vol. 14, No. 3, pp 171–244. DOI: 10.1561/1000000059.

1

Introduction

With the invention of the transistor in 1947, the cornerstone for the digital revolution was laid. As a fundamental building block, the transistor triggered the development of digital circuits. The mass production of digital circuits revolutionized the field of electronics, finally leading to computers, embedded systems, and the Internet. Hence, the impact of digital hardware on society, as well as the economy, was and is tremendous. Over the last decades, the enormous growth in the complexity of integrated circuits has continued as expected. Digital circuits nowadays are much more complex, sometimes even consisting of billions of transistors. Back in 2000, an Intel Pentium 4 processor had 42 million transistors, and it was working with a 1.4 GHz frequency. Thirteen years later, Intel released its Core-i Series processors. They consist of more than 5 billion transistors (i.e. 120$\times$ Pentium 4 transistors) and work with clock speeds of up to 4.4 GHz. Moreover, modern digital circuits are usually designed based on sophisticated algorithms, leading to fast and efficient but complex architectures. The complexity is even higher when it comes to the arithmetic circuits, since 1) they are usually very large, and 2) they are designed based on several optimized algorithms for each stage. As an example, a multiplier consists of three stages and each stage can be created based on different algorithms, leading to an area-efficient and fast circuit.

As modern electronic devices are getting more and more complex, the fundamental issue of functional correctness becomes more important than ever. This is evidenced by many publicly known examples of electronic failures with disastrous consequences. This includes e.g., the Intel Pentium bug in 1994 (Blum and Wasserman, 1996), the New York blackout in 2003, and a design flaw in Intel's Sandy Bridge chipset in 2011. Such costly mistakes can only be prevented by verifying the circuits before they get to production (Drechsler, 2004; Drechsler, 2017). Exhaustive simulation (i.e., checking the outputs for each provided test-vector) is not a feasible approach to ensure correctness since it is impossible to cover the whole input space in the case of large digital circuits. As a result, significant effort has been put into developing formal verification techniques by both academic and industrial research. Essentially, formal verification aims to formally prove that an implementation is correct with respect to its specification. Formal verification methods take advantage of rigorous mathematical reasoning to ensure that a design meets its specification. Nowadays, formal verification is an essential task in industry since it is the only way to ensure the 100% correctness of an implementation. They are extensively used to prove the correctness of arithmetic circuits such as adders, multipliers, and *Arithmetic Logic Units* (ALUs).

Several bit-level and word-level formal verification algorithms have been proposed in recent years to prove the correctness of digital circuits (see, for example, Russinoff *et al.*, 2022; Kaivola and O'Leary, 2022). In practice, they might give satisfying results for some types of circuits, but they might also fail due to non-efficient run-time and memory usage if the size of the circuits increases. As a result, these verification algorithms suffer from **unpredictability in their performance**. The time and space complexities of many formal methods are unknown when it comes to verifying various types of designs. It cannot be predicted before actually invoking the verification tool whether (a) it will successfully terminate or (b) run for an indefinite amount of time. It is a serious challenge in the verification phase and can dramatically affect the time schedule for the implementation and fabrication of a digital circuit. This obstacle can only be overcome by calculating the verification complexity of different types of circuits. We are particularly interested in verification techniques whose space and time complexities are polynomially bounded.

Polynomial Formal Verification (PFV) was first introduced in Drechsler (2021) for adders. Shortly, researchers put a lot of effort into proving the polynomial bounds for the existing methods and proposed new PFV approaches (Drechsler and Mahzoon, 2022). In general, calculating the space and time complexities and proving the polynomial bounds provide us with three main advantages:

- We can predict before running a verification engine whether it returns the results in a limited period. As a result, we can avoid the verification methods with exponential space and time complexities.
- We can ensure the scalability of a verification method when it comes to proving the correctness of a specific type of circuit. Thus, the verification run-time and memory usage increase polynomially with respect to the size of the circuit. It is particularly important when there is a resource constraint for the verification process.
- We can compare the upper-bound space and time complexities of two verification methods when they are applied to a specific type of circuit. Consequently, we can realize which method performs better in terms of run-time and memory usage.

In this monograph, we prove that PFV of arithmetic circuits including adders, multipliers, and ALUs is possible. Furthermore, we calculate the exact upper-bound complexity of verifying different types of adders and multipliers, as well as an ALU.

We first provide an overview of formal verification techniques and clarify the importance of PFV in Section 2. Then, we calculate the upper-bound complexity of verifying various adder architectures using *Binary Decision Diagrams* (BDDs) in Section 3. Subsequently, we prove that PFV of complex multipliers is possible using *Symbolic Computer Algebra* (SCA) and BBDs in Section 4. Section 5 provides the proof for the PFV of ALUs. Finally, Section 6 concludes the work.

References

Blum, M. and H. Wasserman. (1996). “Reflections on the Pentium division bug”. *IEEE Transactions on Computers*. 45(4): 385–393.

Booth, A. D. (1951). “A signed binary multiplication technique”. *The Quarterly Journal of Mechanics and Applied Mathematics*. 4(2): 236–240.

Brace, K. S., R. L. Rudell, and R. E. Bryant. (1990). “Efficient Implementation of a BDD Package”. In: *Design Automation Conference*. 40–45.

Bryant, R. E. (1986). “Graph-Based Algorithms for Boolean Function Manipulation”. *IEEE Transactions on Computers*. 35(8): 677–691.

Bryant, R. E. (1991). “On the Complexity of VLSI Implementations and Graph Representations of Boolean Functions with Application to Integer Multiplication”. *IEEE Transactions on Computers*. 40(2): 205–213.

Bryant, R. E. and Y. A. Chen. (1995). “Verification of Arithmetic Circuits with Binary Moment Diagrams”. In: *Design Automation Conference*. 535–541.

Ciesielski, M., T. Su, A. Yasin, and C. Yu. (2020). “Understanding Algebraic Rewriting for Arithmetic Circuit Verification: A Bit-Flow Model”. *IEEE Transactions on Computer Aided Design of Circuits and Systems*. 39(6): 1346–1357.

Disch, S. and C. Scholl. (2007). "Combinational Equivalence Checking Using Incremental SAT Solving, Output Ordering, and Resets". In: *Asia and South Pacific Design Automation Conference*. 938–943.

Drechsler, R. (2004). *Advanced Formal Verification*. Kluwer Academic Publishers.

Drechsler, R. (2017). *Formal System Verification: State-of the-Art and Future Trends*. Springer.

Drechsler, R. (2021). "PolyAdd: Polynomial Formal Verification of Adder Circuits". In: *IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems*. 99–104.

Drechsler, R., B. Becker, and S. Ruppertz. (1997). "The K*BMD: A Verification Data Structure". *IEEE Design & Test of Computers*. 14(2): 51–59.

Drechsler, R. and S. Höreth. (1998). "Manipulation of *BMDs". In: *Asia and South Pacific Design Automation Conference*. 433–438.

Drechsler, R. and A. Mahzoon. (2022). "Polynomial Formal Verification: Ensuring Correctness under Resource Constraints : (Invited Paper)". In: *International Conference on Computer-Aided Design*. 1–9.

Farahmandi, F. and B. Alizadeh. (2015). "Gröbner basis based formal verification of large arithmetic circuits using Gaussian elimination and cone-based polynomial extraction". USenglish. *Microprocessors and Microsystems*. 39(2): 83–96.

Farahmandi, F. and P. Mishra. (2016). "Automated Test Generation for Debugging Arithmetic Circuits". In: *Design, Automation and Test in Europe*. 1351–1356.

Ghandali, S., C. Yu, D. Liu, W. Brown, and M. Ciesielski. (2015). "Logic Debugging of Arithmetic Circuits". In: *IEEE Annual Symposium on VLSI*. 113–118.

Hamaguchi, K., A. Morita, and S. Yajima. (1995). "Efficient construction of binary moment diagrams for verifying arithmetic circuits". In: *International Conference on Computer-Aided Design*. 78–82.

Kaivola, R. and J. O'Leary. (2022). "Verification of Arithmetic and Datapath Circuits with Symbolic Simulation". In: *Handbook of Computer Architecture*. Ed. by A. Chattopadhyay. Singapore: Springer Nature Singapore. 1–52.

Kaufmann, D., A. Biere, and M. Kauers. (2020). "Incremental column-wise verification of arithmetic circuits using computer algebra". *Formal Methods in System Design: An International Journal*. 56(1): 22–54.

Kogge, P. M. and H. S. Stone. (1973). "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations". *IEEE Transactions on Computers*. C-22(8): 786–793.

Konrad, A., C. Scholl, A. Mahzoon, D. Große, and R. Drechsler. (2022). "Divider Verification Using Symbolic Computer Algebra and Delayed Don't Care Optimization". In: *Formal Methods in Computer-Aided Design*. 1–10.

Koren, I. (2001). *Computer Arithmetic Algorithms*. 2nd. A. K. Peters, Ltd.

Kuehlmann, A. and F. Krohm. (1997). "Equivalence Checking Using Cuts And Heaps". In: *Design Automation Conference*. 263–268.

Kuehlmann, A., V. Paruthi, F. Krohm, and M. K. Ganai. (2002). "Robust Boolean reasoning for equivalence checking and functional property verification". *IEEE Transactions on Computer Aided Design of Circuits and Systems*. 21(12): 1377–1394.

Ladner, R. E. and M. J. Fischer. (1980). "Parallel Prefix Computation". *Journal of the ACM*. 27(4): 831–838.

Mahzoon, A., D. Große, and R. Drechsler. (2018a). "Combining Symbolic Computer Algebra and Boolean Satisfiability for Automatic Debugging and Fixing of Complex Multipliers". In: *IEEE Annual Symposium on VLSI*. 351–356.

Mahzoon, A., D. Große, and R. Drechsler. (2018b). "PolyCleaner: Clean your Polynomials before Backward Rewriting to Verify Million-gate Multipliers". In: *International Conference on Computer-Aided Design*. 129:1–129:8.

Mahzoon, A., D. Große, and R. Drechsler. (2019). "RevSCA: Using Reverse Engineering to Bring Light into Backward Rewriting for Big and Dirty Multipliers". In: *Design Automation Conference*. 185:1–185:6.

Mahzoon, A., D. Große, and R. Drechsler. (2021). "GenMul: Generating Architecturally Complex Multipliers to Challenge Formal Verification Tools". In: *Recent Findings in Boolean Techniques*. Ed. by R. Drechsler and D. Große. Springer. 177–191.

Mahzoon, A., D. Große, and R. Drechsler. (2022a). "RevSCA-2.0: SCA-Based Formal Verification of Nontrivial Multipliers Using Reverse Engineering and Local Vanishing Removal". *IEEE Transactions on Computer Aided Design of Circuits and Systems*. 41(5): 1573–1586.

Mahzoon, A., D. Große, C. Scholl, and R. Drechsler. (2020). "Towards Formal Verification of Optimized and Industrial Multipliers". In: *Design, Automation and Test in Europe*. 544–549.

Mahzoon, A., D. Große, C. Scholl, A. Konrad, and R. Drechsler. (2022b). "Formal Verification of Modular Multipliers using Symbolic Computer Algebra and Boolean Satisfiability". In: *Design Automation Conference*. 1183–1188.

Mishchenko, A., S. Chatterjee, and R. K. Brayton. (2006). "DAG-aware AIG rewriting a fresh look at combinational logic synthesis". In: *Design Automation Conference*. 532–535.

Mishchenko, A., S. Cho, S. Chatterjee, and R. K. Brayton. (2007). "Combinational and sequential mapping with priority cuts". In: *International Conference on Computer-Aided Design*. 354–361.

Navabi, Z. (2005). *Verilog Digital System Design: Register Transfer Level Synthesis, Testbench, and Verification*. McGraw-Hill Professional.

Pan, P. and C.-C. Lin. (1998). "A New Retiming-based Technology Mapping Algorithm for LUT-based FPGAs". In: *FPGAs for Custom Computing Machines*. 35–42.

Parhami, B. (2000). *Computer arithmetic - algorithms and hardware designs*. Oxford University Press.

Pratt, V. (1995). "Anatomy of the Pentium bug". In: *TAPSOFT '95: Theory and Practice of Software Development*. Berlin, Heidelberg: Springer Berlin Heidelberg. 97–107.

Ritirc, D., A. Biere, and M. Kauers. (2017). "Column-Wise Verification of Multipliers Using Computer Algebra". In: *Formal Methods in Computer-Aided Design*. 23–30.

Russinoff, D., J. Bruguera, C. Chau, M. Manjrekar, N. Pfister, and H. Valsaraju. (2022). "Formal Verification of a Chained Multiply-Add Design: Combining Theorem Proving and Equivalence Checking". In: *IEEE Symposium on Computer Arithmetic*. 120–126.

Sabbagh, N. A. and B. Alizadeh. (2021). "Arithmetic Circuit Correction by Adding Optimized Correctors Based on Groebner Basis Computation". In: *European Test Symposium*. 1–6.

Sayed-Ahmed, A., D. Große, M. Soeken, and R. Drechsler. (2016). "Equivalence Checking Using Gröbner Bases". In: *Formal Methods in Computer-Aided Design*. 169–176.

Scholl, C. and A. Konrad. (2020). "Symbolic Computer Algebra and SAT Based Information Forwarding for Fully Automatic Divider Verification". In: *Design Automation Conference*. 1–6.

Scholl, C., A. Konrad, A. Mahzoon, D. Große, and R. Drechsler. (2021). "Verifying Dividers Using Symbolic Computer Algebra and Don't Care Optimization". In: *Design, Automation and Test in Europe*. 1110–1115.

Somenzi, F. (2018). "CUDD: CU Decision Diagram Package Release 2.7.0". URL: https://github.com/ivmai/cudd.

Temel, M., A. Slobodová, and W. A. Hunt. (2020). "Automated and Scalable Verification of Integer Multipliers". In: *Computer Aided Verification*. 485–507.

Testa, E., M. Soeken, L. Amarù, and G. D. Micheli. (2019). "Reducing the Multiplicative Complexity in Logic Networks for Cryptography and Security Applications". In: *Design Automation Conference*. 1–6.

Tseitin, G. S. (1983). "On the Complexity of Derivation in Propositional Calculus". In: *Automation of Reasoning: 2: Classical Papers on Computational Logic 1967–1970*. Berlin, Heidelberg: Springer Berlin Heidelberg. 466–483.

Wallace, C. S. (1964). "A Suggestion for a Fast Multiplier". *IEEE Transactions on Electronic Computers*. EC-13(1): 14–17.

Wegener, I. (2000). *Branching Programs and Binary Decision Diagrams*. SIAM.

Wolf, C. (2022). "Yosys open synthesis suit". URL: https://github.com/YosysHQ/yosys.

Yang, W., L. Wang, and A. Mishchenko. (2012). “Lazy man’s logic synthesis”. In: *International Conference on Computer-Aided Design*. 597–604.

Yasin, A., T. Su, S. Pillement, and M. J. Ciesielski. (2019). “Functional Verification of Hardware Dividers using Algebraic Model”. In: *VLSI of System-on-Chip*. 257–262.

Yu, C., W. Brown, D. Liu, A. Rossi, and M. Ciesielski. (2016). “Formal verification of arithmetic circuits by function extraction”. USenglish. *IEEE Transactions on Computer Aided Design of Circuits and Systems*. 35(12): 2131–2142.

Yu, C., M. Ciesielski, and A. Mishchenko. (2017). “Fast Algebraic Rewriting Based on And-Inverter Graphs”. USenglish. *IEEE Transactions on Computer Aided Design of Circuits and Systems*. 37(9): 1907–1911.

Zimmermann, R. (1997). “Binary Adder Architectures for Cell-Based VLSI and their Synthesis”. *PhD thesis*. Swiss Federal Institute of Technology.

Zuras, D. and W. H. McAllister. (1986). “Balanced delay trees and combinatorial division in VLSI”. *IEEE Journal of Solid-State Circuits*. 21(5): 814–819.