Verifying Multipliers using Computer Algebra

Daniela Kaufmann   Supervisor: Prof. Armin Biere
Johannes Kepler University Linz, Altenbergerstr. 69, 4040 Linz, Austria
daniela.kaufmann@jku.at

Abstract—**Formal verification is used to guarantee the correctness of a given circuit with respect to a certain specification. However arithmetic circuits, and most prominently gate-level multipliers, impose a challenge for existing verification techniques and in practice still require substantial manual effort. The focus of this thesis is to improve automated formal verification of multiplier circuits using computer algebra to make it practically applicable for non-trivial and optimized multiplier designs. In the thesis a rigorous formalization of the problem is given and several optimization techniques are developed which make automated verification an order of magnitude faster than related approaches. As a further contribution a dedicated reduction engine is developed which in addition allows to generate proof certificates, validating the result of the verification approach.**

I. Introduction

Digital circuits are extensively used in computers and digital systems and it is crucial to guarantee that these circuits are correct. A subclass of digital circuits are combinational logic circuits where the output is a function of the present input only. Combinational logic is used in computer circuits to perform Boolean algebra. For example, the part of an arithmetic logic unit (ALU) in a CPU, which is responsible for mathematical calculations, is constructed using combinational logic. If a circuit implements an arithmetic operation it is called arithmetic circuit.

Formal verification is used to prove or disprove the correctness of a given software or hardware system with respect to a certain specification. To this end the system is translated into a mathematical model and automated decision processes are applied to derive the desired correctness properties. The different formal verification approaches are distinguished by the mathematical formalism used in the verification process.

Formal verification of arithmetic circuits is extremely important to help to prevent issues like the famous Pentium FDIV bug, which was detected in 1994 and is reported to have cost Intel almost half a billion dollar. This bug affected the floating point unit of the early Intel Pentium processors in such a way that the processor might return incorrect binary floating point results when dividing a number. Even more than 25 years after detecting this bug the problem of formally verifying arithmetic circuits, and especially multiplier circuits, is still considered to be hard. There have been many attempts since then to verify such circuits, but even today verifying designs with arithmetic parts cannot be applied fully automated. For instance, a common approach in the industry is to black-box multipliers and verify them separately, which often requires additional knowledge about the multiplier design.

Supported by Austrian Science Fund (FWF), NFN S11408-N23 (RiSE).

Up to now several solving techniques have been developed for multiplier verification. A common approach models the problem as a satisfiability (SAT) problem. A large set of such encodings was submitted to the SAT 2016 competition [1]. However the results indicated that SAT solvers need exponential run-time when verifying simple multipliers.

The first technique which was shown to detect the Pentium bug is based on binary decision diagrams, precisely on binary moment diagrams [3]. Nevertheless, this method requires structural knowledge of the multipliers. Contrarily theorem provers in combination with SAT are able to certify industrial multipliers [7], however this approach is not fully automated.

The currently most effective technique for fully automated verification of multipliers uses computer algebra [5], [14], [18]. In this method each logic gate as well as the specification of the multiplier is represented by a polynomial. If the gate polynomials are ordered according to their reverse topological appearance in the circuit, they form a Gröbner basis. As a consequence, the question whether a circuit implements a correct multiplier can be answered by reducing the specification polynomial by the Gröbner basis. The multiplier is correct if and only if the reduction returns zero. The main issue of the algebraic approach is that without preprocessing the size of intermediate reduction results increases drastically.

Most recent algebraic techniques [4] are able to verify large simple multipliers but fail for optimized multiplier designs. In related work [12] various optimizations are presented, which make the algebraic technique scale to large non-trivial multiplier designs, but it remains unclear whether their technique is able to verify optimized and synthesized multipliers.

The aim of this thesis is to investigate and improve formal verification of multiplier circuits using computer algebra to make it practically applicable for non-trivial and optimized multiplier designs. The first theoretical step is to show the correctness of the algebraic approach for circuit verification, which has not been done so far. In [15][1] we propose a simple and precise mathematical formalization of the algebraic approach for arithmetic circuit verification, including rigorous proofs of soundness and completeness. After this crucial goal has been achieved we focus on practical challenges of the problem. We investigate possible reasons of the monomial blow-up in the intermediate reduction results. Based on these results we develop reasoning techniques which overcome this issue, involving theoretical proofs, which guarantee that soundness and completeness is maintained.

[1]Some of my work is published under my maiden name "Ritirc".

One of our technical contributions is a new incremental column-based verification approach for multipliers [15], [8]. In this approach the multiplier circuit is divided into several slices and the correctness of the circuit is shown by incrementally verifying the correctness of each slice. The main advantage of this approach is that only one small part of the global specification is used for reduction, which reduces the size of the intermediate results. Experiments in [15] show that this approach is an order of magnitude faster than previous non-incremental methods.

Additionally we develop optimization techniques based on variable elimination which rewrite and simplify the Gröbner basis in advance [17], [2]. In our rewriting techniques we extract subcircuits, more precisely full- and half-adder structures, from the circuit and eliminate the internal variables of these subcircuits from the Gröbner basis. For this optimization we introduce the necessary theory and present a technical theorem which allows to rewrite only local parts of the Gröbner basis in such a way that the result is again a Gröbner basis.

In our most recent works [9], [10] we generalize the approach of circuit verification to be applicable in more general polynomial rings which allow modular reasoning. Modular reasoning allows us to verify not only signed and unsigned integer multipliers, but also truncated multipliers. Furthermore performance is improved substantially. Additionally we formalize a more general version of variable elimination, which does not require to identify syntactic patterns in the circuits. As a further novelty we invoke SAT in the algebraic approach. In our technique complex final stage adders are detected and replaced by simple adders. These simplified multipliers are verified by computer algebra techniques and correctness of the replacement step by SAT solvers.

In our initial work [15], [17] we used existing computer algebra systems for applying the reduction algorithm. However these systems showed to be too general. We implemented a new dedicated reduction engine AMULET [9], which is tailored to the specific structure of the polynomials. Implementing our own tool gave a speed-up of three orders of magnitude compared to computer algebra systems.

A common approach to increase confidence in the verification results consists of generating proofs which are checked by independent proof checkers. For algebraic reasoning the main focus has been on proof complexity [13], [11], however, generating proofs for practical applications is not well studied. In order to validate verification techniques based on computer algebra we show in [16] how the abstract polynomial calculus [6] can be instantiated to yield a practical algebraic calculus (PAC). Proofs in this format can be obtained as by-product of verifying multipliers with AMULET and can be checked with our proof checker tool PACTRIM [16]. Our experiments produce quadratic polynomial proofs which certify the correctness of certain multipliers. None of the related work [4], [12] produces certificates in their verification approaches.

Our experiments in [9] show that we are able to verify and certify various simple and complex multiplier architectures of bitwidth 64 within seconds and we are an order of magnitude faster than most recent related work [4], [12]. A short excerpt of these experiments is depicted in Table I. It can be seen that our approach scales up to verification and certification of multipliers of input bitwidth 2048.

TABLE I
VERIFYING VARIOUS MULTIPLIERS OF DIFFERENT INPUT SIZE

Architecture	Bitwidth	Verification	[12]	[4]	Certification	Check
sp-ar-rc	64	1 s	NA_1	0 s	2 s	3 s
sp-dt-lf	64	2 s	31 s	NA_2	3 s	3 s
sp-wt-cl	64	11 s	96 s	NA_2	12 s	10 s
sp-bd-ks	64	3 s	162 s	NA_2	4 s	4 s
sp-ar-ck	64	1 s	143 s	NA_2	2 s	3 s
bp-ar-rc	64	1 s	53 s	NA_2	2 s	3 s
bp-ct-bk	64	2 s	119 s	NA_2	2 s	3 s
bp-os-cu	64	2 s	95 s	NA_2	3 s	4 s
bp-wt-cs	64	1 s	75 s	NA_2	2 s	3 s
btor	2048	25.2 h	NA_1	41.9 h	42.5 h	7.2 h
kojvnkv	2048	19.4 h	NA_1	20.7 h	34.6 h	20.5 h

NA_1: tool not yet available NA_2: incompleteness

REFERENCES

[1] A. Biere. Collection of Combinational Arithmetic Miters. In *SAT Competition 2016*, volume B-2016-1, pages 65–66. Univ. Helsinki, 2016.

[2] A. Biere, M. Kauers, and D. Ritirc. Challenges in verifying arithmetic circuits using computer algebra. In *SYNASC*, pages 9–15. IEEE, 2017.

[3] Y. Chen and R. E. Bryant. Verification of arithmetic circuits with binary moment diagrams. In *DAC*, pages 535–541. ACM Press, 1995.

[4] M. Ciesielski, T. Su, A. Yasin, and C. Yu. Understanding Algebraic Rewriting for Arithmetic Circuit Verification: a Bit-Flow Model. *IEEE TCAD*, pages 1–1, 2019.

[5] M. J. Ciesielski, C. Yu, W. Brown, D. Liu, and A. Rossi. Verification of gate-level arithmetic circuits by function extraction. In *DAC*, pages 52:1–52:6. ACM, 2015.

[6] M. Clegg, J. Edmonds, and R. Impagliazzo. Using the groebner basis algorithm to find proofs of unsatisfiability. In *STOC*, pages 174–183. ACM, 1996.

[7] W. A. Hunt, M. Kaufmann, J. Strother Moore, and A. Slobodova. Industrial hardware and software verification with ACL2. *Philos. Trans. Royal Soc. A*, 375:20150399, 10 2017.

[8] D. Kaufmann, A. Biere, and M. Kauers. Incremental Column-wise verification of arithmetic circuits using computer algebra. *FMSD*, 2019.

[9] D. Kaufmann, A. Biere, and M. Kauers. Verifying Large Multipliers by Combining SAT and Computer Algebra. In *FMCAD 2019*, pages 28–36, 2019.

[10] D. Kaufmann, A. Biere, and M. Kauers. From DRUP to PAC and Back. In *DATE 2020*, To appear.

[11] M. Lauria and J. Nordström. Graph Colouring is Hard for Algorithms Based on Hilbert's Nullstellensatz and Gröbner Bases. In *CCC*, volume 79 of *LIPIcs*, pages 2:1–2:20. Schloss Dagstuhl, 2017.

[12] A. Mahzoon, D. Große, and R. Drechsler. RevSCA: Using Reverse Engineering to Bring Light into Backward Rewriting for Big and Dirty Multipliers. In *DAC*, page 185. ACM, 2019.

[13] M. Miksa and J. Nordström. A generalized method for proving polynomial calculus degree lower bounds. In *CCC*, volume 33 of *LIPIcs*, pages 467–487. Schloss Dagstuhl, 2015.

[14] T. Pruss, P. Kalla, and F. Enescu. Equivalence Verification of Large Galois Field Arithmetic Circuits using Word-Level Abstraction via Gröbner Bases. In *DAC*, pages 152:1–152:6. ACM, 2014.

[15] D. Ritirc, A. Biere, and M. Kauers. Column-wise verification of multipliers using computer algebra. In *FMCAD*, pages 23–30. IEEE, 2017.

[16] D. Ritirc, A. Biere, and M. Kauers. A Practical Polynomial Calculus for Arithmetic Circuit Verification. In *SC-Square Workshop 2018*, pages 61–76. CEUR-WS, 2018.

[17] D. Ritirc, A. Biere, and M. Kauers. Improving and extending the algebraic approach for verifying gate-level multipliers. In *DATE*, pages 1556–1561. IEEE, 2018.

[18] A. Sayed-Ahmed, D. Große, U. Kühne, M. Soeken, and R. Drechsler. Formal verification of integer multipliers by combining Gröbner basis with logic reduction. In *DATE*, pages 1048–1053. IEEE, 2016.