AMULET 2.0 FOR VERIFYING MULTIPLIER CIRCUITS

Daniela Kaufmann and Armin Biere

✉️ daniela.kaufmann@jku.at
Johannes Kepler University
Linz, Austria

TACAS 2021
online
March 31, 2021
Bugs in hardware are expensive!

Circuit verification prevents issues like the famous Pentium FDIV bug.

**Multiplier verification**

**Given:** Gate-level integer multiplier for fixed bit-width.

**Input format:** AND-Inverter Graph

**Question:** For all possible \( a_i, b_i \in \mathbb{B} \):

\[
(2a_1 + a_0) \times (2b_1 + b_0) = 8s_3 + 4s_2 + 2s_1 + s_0?
\]
Formal Verification Techniques

Satisfiability Checking (SAT)
- SAT 2016 Competition [Bie16]
- Exponential run-time of solvers

Theorem Proving
- Used in industry
- Past: Requires manual effort
- 2020: Progress in ACL2 [TSH20]

Decision Diagrams
- First technique to detect Pentium bug [CB95]
- Cannot be applied fully automatically

Algebraic Approach
- Seminal work: [CYB+15] [SGK+16]
- Polynomial encoding
- Works for non-trivial multiplier designs
Basic Idea of Algebraic Approach

Multiplier

Polynomials

Specification

Ideal Membership

For more details on circuit verification using computer algebra see e.g. [Kau20].
From Circuits to Polynomials

Gate polynomials $G(C) \subseteq \mathbb{Z}[X]$.

$$-s_3 + l_{24}$$
$$-s_2 + l_{28}$$
$$-s_1 + l_{20}$$
$$-s_0 + l_{10}$$
$$-l_{28} + l_{26}l_{24} - l_{26} - l_{24} + 1$$
$$-l_{26} + l_{22}l_{16} - l_{22} - l_{16} + 1$$
$$-l_{24} + l_{22}l_{16}$$

$$-l_{22} + a_1 b_1$$
$$-l_{20} + l_{18}l_{16} - l_{18} - l_{16} + 1$$
$$-l_{18} + l_{14}l_{12} - l_{14} - l_{12} + 1$$
$$-l_{16} + l_{14}l_{12}$$

Boolean value constraints $B(C) \subseteq \mathbb{Z}[X]$.

$$a_1, a_0 \in \mathbb{B}$$
$$b_1, b_0 \in \mathbb{B}$$

$$-a_1^2 + a_1, -a_0^2 + a_0,$$

$$-b_1^2 + b_1, -b_0^2 + b_0$$

Specification $S_n \in \mathbb{Z}[X]$.

$$8s_3 + 4s_2 + 2s_1 + s_0 - (2a_1 + a_0)(2b_1 + b_0)$$
Verification Technique

Verification Algorithm

Reduce specification $\sum_{i=0}^{2n-1} 2^i s_i - (\sum_{i=0}^{n-1} 2^i a_i) (\sum_{i=0}^{n-1} 2^i b_i)$ by elements of $G(C) \cup B(C)$ until no further reduction is possible, then $C$ is a multiplier iff remainder is zero.

Optimizations

- Preprocessing based on variable elimination
- Incremental column-wise verification algorithm
Partial Product Generation

Partial Product Accumulation

Final Stage Adder

Adder Substitution

Partial Product Generation

Partial Product Accumulation

Ripple Carry Adder

\[ a_{n-1}, \ldots, a_0 \]

\[ b_{n-1}, \ldots, b_0 \]

\[ x_m y_m \]

\[ \ldots \]

\[ x_0 y_0 c_in \]

\[ s_0 \]

\[ s_{k+1} \]

\[ s_k \]

\[ s_{2n-2} \]

\[ s_{2n-1} \]

\[ c_{m+1} \]

\[ s'_m \]

\[ s'_0 \]
SAT & Computer Algebra [KBK19]

\[ a_{n-1}, \ldots, a_0 \quad \text{and} \quad b_{n-1}, \ldots, b_0 \]

Partial Product Generation

Partial Product Accumulation

Final Stage Adder

Adder Substitution

Ripple Carry Adder

SAT

\[ x_m y_m \quad \ldots \quad x_0 y_0 c_{in} \]
AMulet

AMULET 1.0 [KBK19]

- C implementation (single file)
- Automatically applies adder substitution and circuit verification.
- Designed to make use of properties of polynomials.
- Provides proof certificates in practical algebraic calculus and Nullstellensatz.

AMULET 2.0

- Modular C++ re-implementation
- Enhanced heuristics for adder substitution.
- Refined techniques for deciding on reduction order.
- Improved memory management for representation of polynomials.
AMulet 2.0

- substitute
  .cnf Miter
  .aig Rewritten AIG

- certify
  Multiplier correct?
  .pol Gate constraints
  .prf Proof
  .spc Specification

- verify
  Multiplier correct?

Substitution Engine
Polynomial Solver
Gate Library
AIG Parser
Substitution
Elimination
Slicing
Proofs
Polynomial Library
Monomial
Term
Variable
Hashing

Mode
Input AIG
Rewritten AIG
Multiplier correct?
Gate constraints
Proof
Specification

AMulet2.0
AMulet 2.0

Source
Open source under MIT license

- Webpage (incl. source code): http://fmv.jku.at/amulet2
- Artifact: http://fmv.jku.at/amulet2_artifact
- Maintained version: https://github.com/d-kfmnn/amulet2

Requirements & Compilation

- AIGER library (provided with source code) [BHW11]
- GMP library (needs to be preinstalled) [Gt16]
- To compile execute ./configure.sh && make
Evaluation

64-bit multipliers (384 instances)

Array multiplier

Input bit-width

CPU time, time limit = 300 sec

CPU time, time limit = 86 400 sec (24 h)
AMULET 2.0 FOR VERIFYING MULTIPLIER CIRCUITS

Daniela Kaufmann and Armin Biere
✉ daniela.kaufmann@jku.at
Johannes Kepler University
Linz, Austria

TACAS 2021
online
March 31, 2021
References I

[BHW11] Armin Biere, Keijo Heljanko, and Siert Wieringa.
AIGER 1.9 And Beyond.
Technical report, FMV Reports Series, JKU Linz, Austria, 2011.

[Bie16] Armin Biere.
Collection of Combinational Arithmetic Miters Submitted to the SAT Competition 2016.
In SAT Competition 2016, volume B-2016-1 of Dep. of Computer Science Report Series B, pages 65–66. University of Helsinki, 2016.

[CB95] Yirng-An Chen and Randal E. Bryant.
Verification of Arithmetic Circuits with Binary Moment Diagrams.
In Design Automation Conference, DAC 1995, pages 535–541. ACM, 1995.

[CSYY20] Maciej J. Ciesielski, Tiankai Su, Atif Yasin, and Cunxi Yu.
Understanding Algebraic Rewriting for Arithmetic Circuit Verification: a Bit-Flow Model.
IEEE TCAD, 39(6):1346–1357, 2020.
References II

[CYB+15] Maciej J. Ciesielski, Cunxi Yu, Walter Brown, Duo Liu, and André Rossi.
Verification of Gate-level Arithmetic Circuits by Function Extraction.
In Design Automation Conference, DAC 2015, pages 52:1–52:6. ACM, 2015.

[Gt16] Torbjörn Granlund and the GMP development team.
GNU MP: The GNU Multiple Precision Arithmetic Library, 2016.
Version 6.1.2.

[Kau20] Daniela Kaufmann.
*Formal Verification of Multiplier Circuits using Computer Algebra.*
PhD thesis, Informatik, Johannes Kepler University Linz, 2020.

[KBK19] Daniela Kaufmann, Armin Biere, and Manuel Kauers.
Verifying Large Multipliers by Combining SAT and Computer Algebra.
In FMCAD 2019, pages 28–36. IEEE, 2019.
References III

[MGD19] Alireza Mahzoon, Daniel Große, and Rolf Drechsler.
RevSCA: Using Reverse Engineering to Bring Light into Backward Rewriting for Big and Dirty Multipliers.
In DAC 2019, pages 185:1–185:6. ACM, 2019.

[SGK+16] Amr Sayed-Ahmed, Daniel Große, Ulrich Kühne, Mathias Soeken, and Rolf Drechsler.
Formal verification of integer multipliers by combining gröbner basis with logic reduction.
In Design, Automation and Test in Europe Conference and Exposition, DATE 2016, pages 1048–1053. IEEE, 2016.

[TSH20] Mertcan Temel, Anna Slobodová, and Warren A. Hunt.
Automated and scalable verification of integer multipliers.
In CAV (1), volume 12224 of Lecture Notes in Computer Science, pages 485–507. Springer, 2020.