FORMAL VERIFICATION OF MULTIPLIER CIRCUITS USING COMPUTER ALGEBRA

Rigorosum

Daniela Kaufmann

Institute for Formal Models and Verification
Johannes Kepler University
Linz, Austria

April 21, 2020

JOHANNES KEPLER
UNIVERSITY LINZ

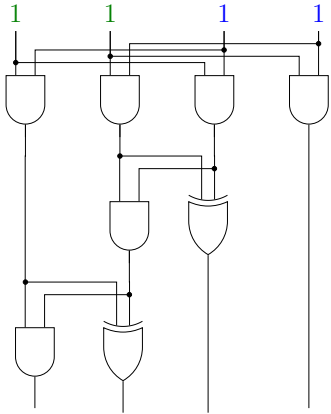
Multiplier Circuits

$$
\begin{array}{cccc}
1 & 1 & \cdot & 1 & 1 \\
\hline
 & & & 1 & 1 \\
 & & 1 & 1 & \\
 & {\scriptstyle 1} & & {\scriptstyle 1} & {\scriptstyle 0} \\
\hline
 & 1 & 0 & 0 & 1
\end{array}
$$

$3 \cdot 3 = 9$

Multiplier Circuits

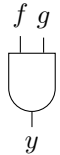
$$
\begin{array}{cccc}
1 & 1 & \cdot & 1 & 1 \\
\hline
 & & & 1 & 1 \\
 & & 1 & 1 & \\
 & 1 & 1 & 0 & \\
\hline
 & 1 & 0 & 0 & 1
\end{array}
$$



$3 \cdot 3 = 9$

Multiplier Circuits

AND-Gate



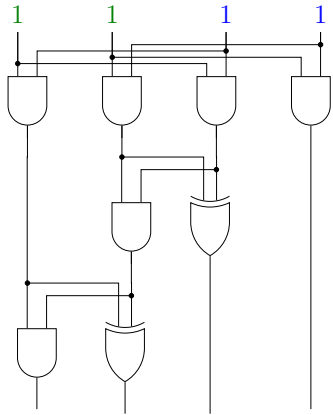
$f \wedge g = y$

f	g	y
0	0	0
0	1	0
1	0	0
1	1	1

XOR-Gate

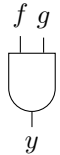
$f \oplus g = y$

f	g	y
0	0	0
0	1	1
1	0	1
1	1	0



Multiplier Circuits

AND-Gate



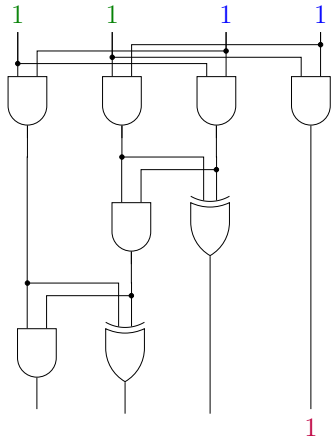
$f \wedge g = y$

f	g	y
0	0	0
0	1	0
1	0	0
1	1	1

XOR-Gate

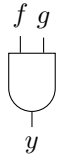
$f \oplus g = y$

f	g	y
0	0	0
0	1	1
1	0	1
1	1	0



Multiplier Circuits

AND-Gate



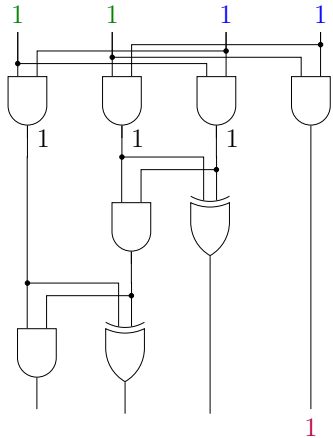
$f \wedge g = y$

f	g	y
0	0	0
0	1	0
1	0	0
1	1	1

XOR-Gate

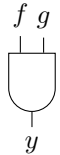
$f \oplus g = y$

f	g	y
0	0	0
0	1	1
1	0	1
1	1	0



Multiplier Circuits

AND-Gate



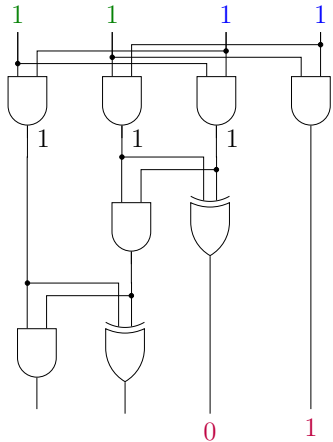
$f \wedge g = y$

f	g	y
0	0	0
0	1	0
1	0	0
1	1	1

XOR-Gate

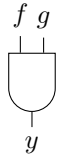
$f \oplus g = y$

f	g	y
0	0	0
0	1	1
1	0	1
1	1	0



Multiplier Circuits

AND-Gate



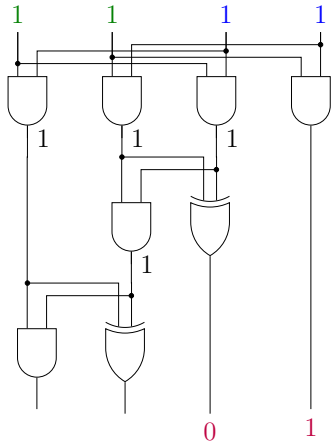
$f \wedge g = y$

f	g	y
0	0	0
0	1	0
1	0	0
1	1	1

XOR-Gate

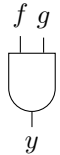
$f \oplus g = y$

f	g	y
0	0	0
0	1	1
1	0	1
1	1	0



Multiplier Circuits

AND-Gate



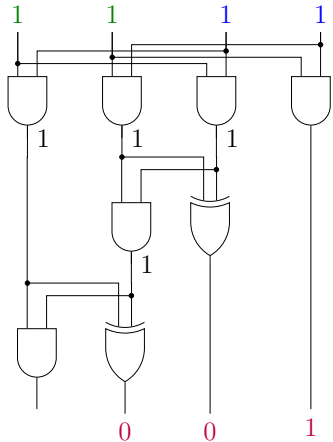
$f \wedge g = y$

f	g	y
0	0	0
0	1	0
1	0	0
1	1	1

XOR-Gate

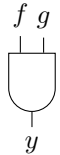
$f \oplus g = y$

f	g	y
0	0	0
0	1	1
1	0	1
1	1	0



Multiplier Circuits

AND-Gate



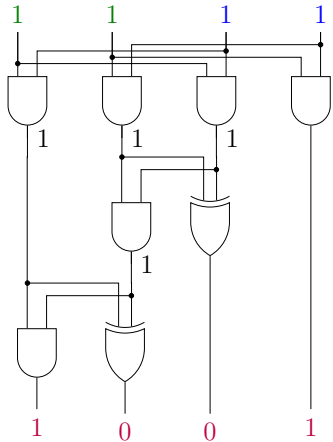
$f \wedge g = y$

f	g	y
0	0	0
0	1	0
1	0	0
1	1	1

XOR-Gate

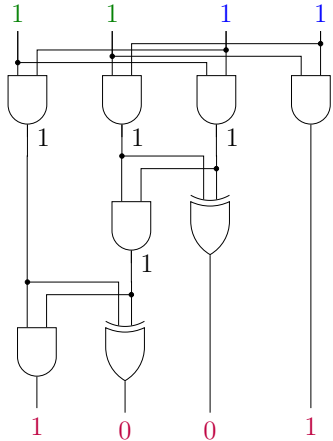
$f \oplus g = y$

f	g	y
0	0	0
0	1	1
1	0	1
1	1	0



Multiplier Circuits

$$
\begin{array}{cccc}
1 & 1 & \cdot & 1 & 1 \\
\hline
 & & & 1 & 1 \\
 & & 1 & 1 & \\
 & 1 & 1 & 0 & \\
\hline
 & 1 & 0 & 0 & 1
\end{array}
$$

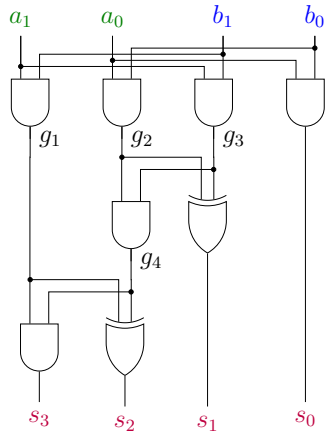


Multiplier Circuits

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

Question: For all possible $a_i, b_i \in \mathbb{B}$:

$$(2a_1 + a_0) * (2b_1 + b_0) = 8s_3 + 4s_2 + 2s_1 + s_0?$$



Intel Pentium FDIV Bug 1994

- Affected floating point unit (FPU) in early Intel processors.
- Processor might return incorrect result for division.
- Cost in 1994: 500 million dollars.

Even more than 25 years later, verification of arithmetic circuits is considered to be hard.

Formal Verification Techniques

Satisfiability Checking (SAT)

- SAT 2016 Competition
- Exponential run-time of solvers

Formal Verification Techniques

Satisfiability Checking (SAT)

- SAT 2016 Competition
- Exponential run-time of solvers

Decision Diagrams

- First technique to detect Pentium bug
- Cannot be applied fully automatically

Formal Verification Techniques

Satisfiability Checking (SAT)

- SAT 2016 Competition
- Exponential run-time of solvers

Decision Diagrams

- First technique to detect Pentium bug
- Cannot be applied fully automatically

Theorem Proving

- Used in industry
- Requires manual effort

Formal Verification Techniques

Satisfiability Checking (SAT)

- SAT 2016 Competition
- Exponential run-time of solvers

Decision Diagrams

- First technique to detect Pentium bug
- Cannot be applied fully automatically

Theorem Proving

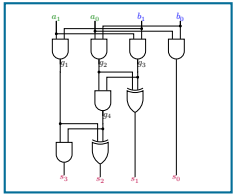
- Used in industry
- Requires manual effort

Algebraic Approach

- Seminal work: [CYBLR'15] [SGKSD'16]
- Polynomial encoding
- Works for non-trivial multiplier designs

Basic Idea of Algebraic Approach

Multiplier



Polynomials

$$
\begin{aligned}
B = \{ & \\
& x - a_0 * b_0, \\
& y - a_1 * b_1, \\
& s_0 - x * y, \\
& \dots \\
\} &
\end{aligned}
$$

Specification

$$\sum_{i=0}^{2n-1} 2^i s_i - (\sum_{i=0}^{n-1} 2^i a_i)(\sum_{i=0}^{n-1} 2^i b_i)$$

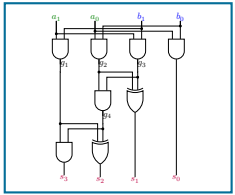
Implication

$= 0$ ✓

$\neq 0$ ✗

Basic Idea of Algebraic Approach

Multiplier



Polynomials

$$
\begin{aligned}
B = \{ & \\
& x - a_0 * b_0, \\
& y - a_1 * b_1, \\
& s_0 - x * y, \\
& \dots \\
\} &
\end{aligned}
$$

Specification

$$\sum_{i=0}^{2n-1} 2^i s_i - (\sum_{i=0}^{n-1} 2^i a_i)(\sum_{i=0}^{n-1} 2^i b_i)$$

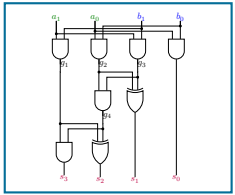
Roots

$= 0$ ✓

$\neq 0$ ✗

Basic Idea of Algebraic Approach

Multiplier



Polynomials

$$
\begin{aligned}
B = \{ & \\
& x - a_0 * b_0, \\
& y - a_1 * b_1, \\
& s_0 - x * y, \\
& \dots \\
\} &
\end{aligned}
$$

Specification

$$\sum_{i=0}^{2n-1} 2^i s_i - (\sum_{i=0}^{n-1} 2^i a_i)(\sum_{i=0}^{n-1} 2^i b_i)$$

Ideal Membership

$= 0$ ✓

$\neq 0$ ✗

Multiplier Specification

Unsigned Integers:

$$0 = \sum_{i=0}^{2n-1} 2^i s_i - \left(\sum_{i=0}^{n-1} 2^i a_i\right)\left(\sum_{i=0}^{n-1} 2^i b_i\right) \qquad \in \mathbb{Z}[X]$$

Multiplier Specification

Unsigned Integers:

$$0 = \sum_{i=0}^{2n-1} 2^i s_i - \left(\sum_{i=0}^{n-1} 2^i a_i\right)\left(\sum_{i=0}^{n-1} 2^i b_i\right) \qquad \in \mathbb{Z}[X]$$

Signed Integers:

$$0 = -2^{2n-1} s_{2n-1} + \sum_{i=0}^{2n-2} 2^i s_i - \left(-2^{n-1} a_{n-1} + \sum_{i=0}^{n-2} 2^i a_i\right)\left(-2^{n-1} b_{n-1} + \sum_{i=0}^{n-2} 2^i b_i\right) \qquad \in \mathbb{Z}[X]$$

Multiplier Specification

Unsigned Integers:

$$0 = \sum_{i=0}^{2n-1} 2^i s_i - \left(\sum_{i=0}^{n-1} 2^i a_i\right)\left(\sum_{i=0}^{n-1} 2^i b_i\right) \qquad \in \mathbb{Q}[X]$$

Signed Integers:

$$0 = -2^{2n-1} s_{2n-1} + \sum_{i=0}^{2n-2} 2^i s_i - \left(-2^{n-1} a_{n-1} + \sum_{i=0}^{n-2} 2^i a_i\right)\left(-2^{n-1} b_{n-1} + \sum_{i=0}^{n-2} 2^i b_i\right) \qquad \in \mathbb{Q}[X]$$

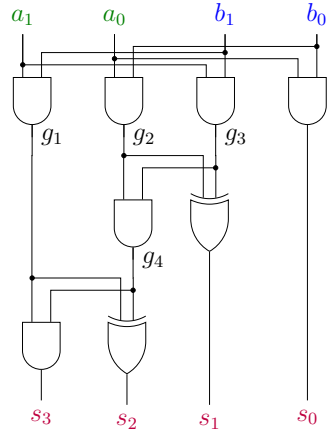
From Circuits to Polynomials

AND-Gate $f \wedge g = y$

f	g	y
0	0	0
0	1	0
1	0	0
1	1	1

XOR-Gate $f \oplus g = y$

f	g	y
0	0	0
0	1	1
1	0	1
1	1	0



From Circuits to Polynomials

AND-Gate $f \wedge g = y$

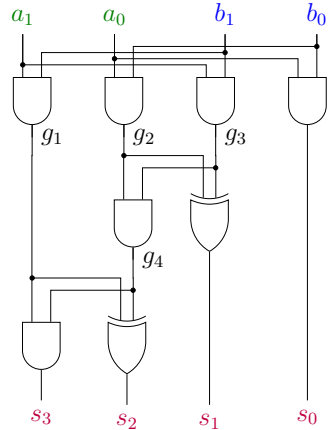
f	g	y
0	0	0
0	1	0
1	0	0
1	1	1

$$-y + fg$$

XOR-Gate $f \oplus g = y$

f	g	y
0	0	0
0	1	1
1	0	1
1	1	0

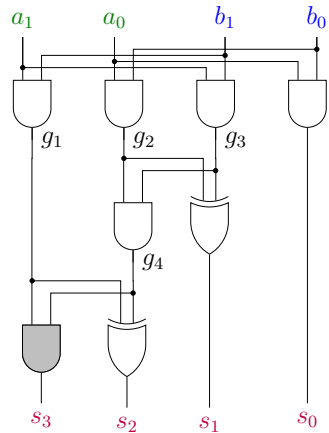
$$-y + f + g - 2fg$$



From Circuits to Polynomials

Gate polynomials $G(C)$.

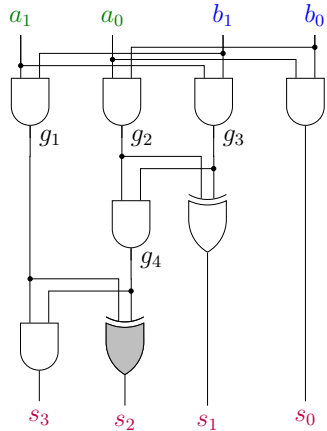
$$s_3 = g_1 \wedge g_4 \qquad -s_3 + g_4 g_1,$$



From Circuits to Polynomials

Gate polynomials $G(C)$.

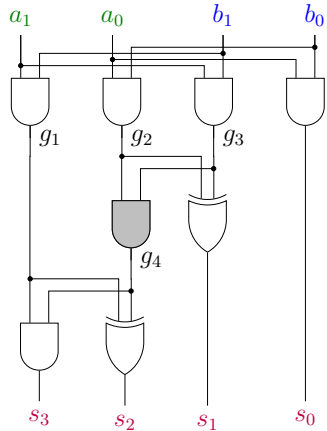
$$
\begin{aligned}
s_3 = g_1 \wedge g_4 \qquad & -s_3 + g_4 g_1, \\
s_2 = g_1 \oplus g_4 \qquad & -s_2 - 2g_4 g_1 + g_4 + g_1,
\end{aligned}
$$



From Circuits to Polynomials

Gate polynomials $G(C)$.

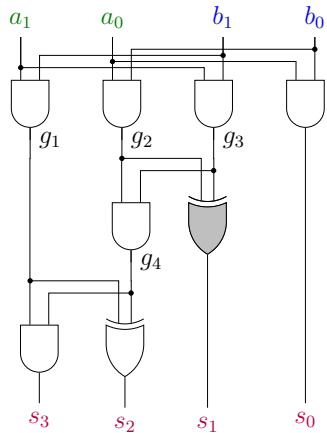
$$
\begin{aligned}
s_3 = g_1 \wedge g_4 \qquad & -s_3 + g_4 g_1, \\
s_2 = g_1 \oplus g_4 \qquad & -s_2 - 2g_4 g_1 + g_4 + g_1, \\
g_4 = g_2 \wedge g_3 \qquad & -g_4 + g_2 g_3,
\end{aligned}
$$



From Circuits to Polynomials

Gate polynomials $G(C)$.

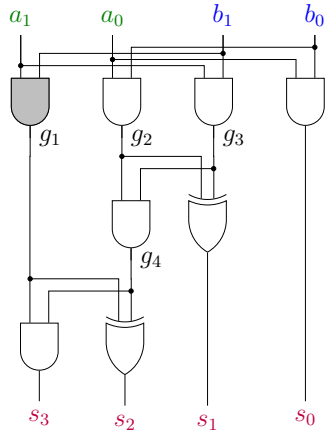
$$
\begin{aligned}
s_3 = g_1 \wedge g_4 \qquad & -s_3 + g_4 g_1, \\
s_2 = g_1 \oplus g_4 \qquad & -s_2 - 2g_4 g_1 + g_4 + g_1, \\
g_4 = g_2 \wedge g_3 \qquad & -g_4 + g_2 g_3, \\
s_1 = g_2 \oplus g_3 \qquad & -s_1 - 2g_2 g_3 + g_2 + g_3,
\end{aligned}
$$



From Circuits to Polynomials

Gate polynomials $G(C)$.

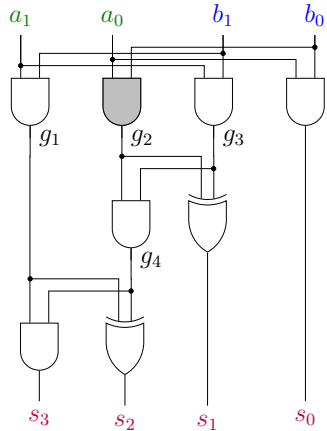
$$
\begin{aligned}
s_3 = g_1 \wedge g_4 \quad & -s_3 + g_4 g_1, \\
s_2 = g_1 \oplus g_4 \quad & -s_2 - 2g_4 g_1 + g_4 + g_1, \\
g_4 = g_2 \wedge g_3 \quad & -g_4 + g_2 g_3, \\
s_1 = g_2 \oplus g_3 \quad & -s_1 - 2g_2 g_3 + g_2 + g_3, \\
g_1 = a_1 \wedge b_1 \quad & -g_1 + a_1 b_1,
\end{aligned}
$$



From Circuits to Polynomials

Gate polynomials $G(C)$.

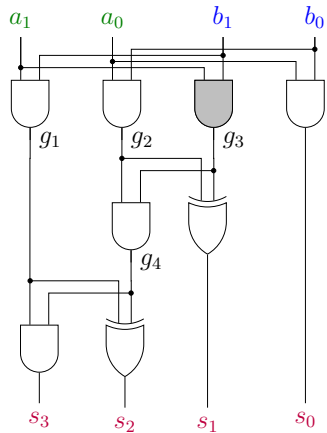
$$
\begin{aligned}
s_3 = g_1 \wedge g_4 \quad & -s_3 + g_4 g_1, \\
s_2 = g_1 \oplus g_4 \quad & -s_2 - 2g_4 g_1 + g_4 + g_1, \\
g_4 = g_2 \wedge g_3 \quad & -g_4 + g_2 g_3, \\
s_1 = g_2 \oplus g_3 \quad & -s_1 - 2g_2 g_3 + g_2 + g_3, \\
g_1 = a_1 \wedge b_1 \quad & -g_1 + a_1 b_1, \\
g_2 = a_0 \wedge b_1 \quad & -g_2 + a_0 b_1,
\end{aligned}
$$



From Circuits to Polynomials

Gate polynomials $G(C)$.

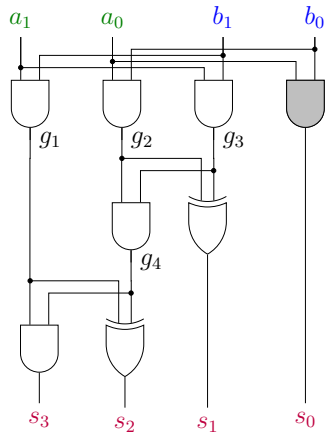
$$
\begin{aligned}
s_3 = g_1 \wedge g_4 \quad & -s_3 + g_4 g_1, \\
s_2 = g_1 \oplus g_4 \quad & -s_2 - 2g_4 g_1 + g_4 + g_1, \\
g_4 = g_2 \wedge g_3 \quad & -g_4 + g_2 g_3, \\
s_1 = g_2 \oplus g_3 \quad & -s_1 - 2g_2 g_3 + g_2 + g_3, \\
g_1 = a_1 \wedge b_1 \quad & -g_1 + a_1 b_1, \\
g_2 = a_0 \wedge b_1 \quad & -g_2 + a_0 b_1, \\
g_3 = a_1 \wedge b_0 \quad & -g_3 + a_1 b_0,
\end{aligned}
$$



From Circuits to Polynomials

Gate polynomials $G(C)$.

$$
\begin{aligned}
s_3 = g_1 \wedge g_4 \quad & -s_3 + g_4 g_1, \\
s_2 = g_1 \oplus g_4 \quad & -s_2 - 2g_4 g_1 + g_4 + g_1, \\
g_4 = g_2 \wedge g_3 \quad & -g_4 + g_2 g_3, \\
s_1 = g_2 \oplus g_3 \quad & -s_1 - 2g_2 g_3 + g_2 + g_3, \\
g_1 = a_1 \wedge b_1 \quad & -g_1 + a_1 b_1, \\
g_2 = a_0 \wedge b_1 \quad & -g_2 + a_0 b_1, \\
g_3 = a_1 \wedge b_0 \quad & -g_3 + a_1 b_0, \\
s_0 = a_0 \wedge b_0 \quad & -s_0 + a_0 b_0
\end{aligned}
$$



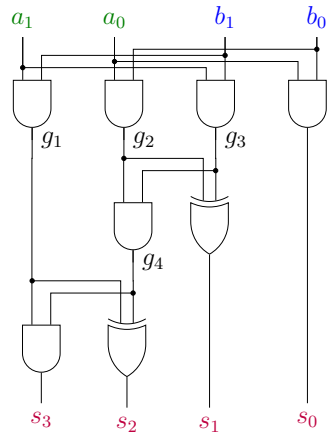
From Circuits to Polynomials

Gate polynomials $G(C)$.

$$
\begin{aligned}
s_3 = g_1 \wedge g_4 \quad & -s_3 + g_4 g_1, \\
s_2 = g_1 \oplus g_4 \quad & -s_2 - 2g_4 g_1 + g_4 + g_1, \\
g_4 = g_2 \wedge g_3 \quad & -g_4 + g_2 g_3, \\
s_1 = g_2 \oplus g_3 \quad & -s_1 - 2g_2 g_3 + g_2 + g_3, \\
g_1 = a_1 \wedge b_1 \quad & -g_1 + a_1 b_1, \\
g_2 = a_0 \wedge b_1 \quad & -g_2 + a_0 b_1, \\
g_3 = a_1 \wedge b_0 \quad & -g_3 + a_1 b_0, \\
s_0 = a_0 \wedge b_0 \quad & -s_0 + a_0 b_0
\end{aligned}
$$

Boolean value constraints $B(C)$.

$$
\begin{aligned}
a_1, a_0 \in \mathbb{B} \quad & a_1(1-a_1),\ a_0(1-a_0), \\
b_1, b_0 \in \mathbb{B} \quad & b_1(1-b_1),\ b_0(1-b_0)
\end{aligned}
$$

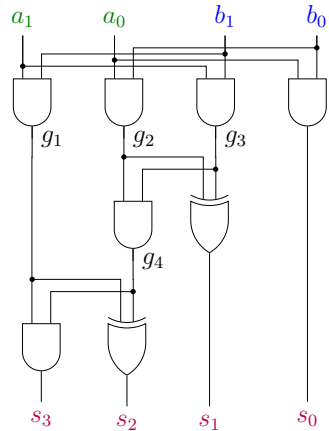


Ideals Associated to Circuits

[FMCAD'17, FMSD'19]

More circuit relations:

- $s_0 - a_0 b_0$ AND-gate
- $a_1^2 - a_1$ a_1 Boolean
- $g_2^2 - g_2$ g_2 Boolean
- $s_1 g_4$ XOR-AND constraint
- $\ldots$



Ideals Associated to Circuits

[FMCAD'17, FMSD'19]

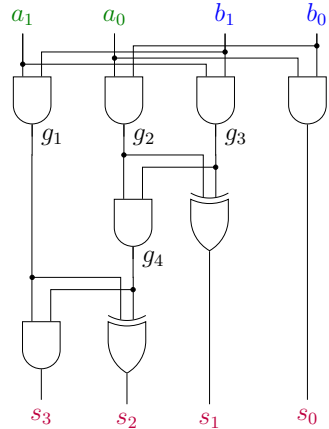
- $I(C)$ contains all relations of the circuit C.
- $I(C)$ is an ideal.

Ideal. A subset $I \subseteq R[X]$ is called an ideal if

$$\forall\, p, q \in I : p+q \in I \quad \text{and} \quad \forall\, p \in R[X]\, \forall\, q \in I : pq \in I.$$

Multiplier. A circuit C is called a multiplier if

$$\sum_{i=0}^{2n-1} 2^i s_i - \left(\sum_{i=0}^{n-1} 2^i a_i\right)\left(\sum_{i=0}^{n-1} 2^i b_i\right) \quad \in \quad I(C).$$



Ideal Membership Problem

If I is the smallest ideal containing a set G, we say G is a **basis** of I and write $I = \langle G \rangle$.

Given an arbitrary basis G of I it is not obvious how to decide ideal membership.

Ideal Membership Problem

If I is the smallest ideal containing a set G, we say G is a **basis** of I and write $I = \langle G \rangle$.

Given an arbitrary basis G of I it is not obvious how to decide ideal membership.

Solution: We need a basis with certain structural properties, called Gröbner basis.

Ideal Membership Problem

If I is the smallest ideal containing a set G, we say G is a **basis** of I and write $I = \langle G \rangle$.

Given an arbitrary basis G of I it is not obvious how to decide ideal membership.

Solution: We need a basis with certain structural properties, called Gröbner basis.

Gröbner basis.[BB'65]

- Offers decision procedure for ideal membership problem in $R[X]$, where R is a field.
- Every ideal of $R[X]$ has a finite Gröbner basis.
- Given an arbitrary basis of an ideal, we can compute a Gröbner basis.
- Buchberger's algorithm is expensive.

Ideal Membership Problem

[FMCAD'17, FMSD'19]

- We can deduce some elements of $I(C)$:
  - Gate polynomials $G(C)$
  - Boolean value constraints $B(C)$
- Let $J(C) = \langle G(C) \cup B(C) \rangle$.
- Lexicographic term order: output variable of a gate is greater than input variables.

Conjecture: [CYBLR'15, SGKSD'16]
$G(C) \cup B(C)$ is a Gröbner basis for $J(C)$.

Ideal Membership Problem

[FMCAD'17, FMSD'19]

- We can deduce some elements of $I(C)$:
  - Gate polynomials $G(C)$
  - Boolean value constraints $B(C)$
- Let $J(C) = \langle G(C) \cup B(C) \rangle$.
- Lexicographic term order: output variable of a gate is greater than input variables.

Theorem (FMCAD'17)

$G(C) \cup B(C)$ is a Gröbner basis for $J(C)$.

Proof idea: Application of Buchberger's Product criterion.

Soundness and Completeness

[FMCAD'17, FMSD'19]

Theorem

For all acyclic circuits C, we have $J(C) = I(C)$.

- $J(C) \subseteq I(C)$: soundness
- $I(C) \subseteq J(C)$: completeness

Non-Incremental Checking Algorithm

[FMCAD'17, FMSD'19]

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.

Verification

$$
\begin{aligned}
&G(C) \cup B(C) = \{ \\
&\quad -s_3 + g_1 g_4, \\
&\quad -s_2 - 2g_1 g_4 + g_4 + g_1, \\
&\quad -g_4 + g_2 g_3, \\
&\quad -s_1 - 2g_2 g_3 + g_3 + g_2, \\
&\quad -g_1 + a_1 b_1, \\
&\quad -g_2 + a_0 b_1, \\
&\quad -g_3 + a_1 b_0, \\
&\quad -s_0 + a_0 b_0, \\
&\quad -a_1^2 + a_1, \\
&\quad -a_0^2 + a_0, \\
&\quad -b_1^2 + b_1, \\
&\quad -b_0^2 + b_0\}
\end{aligned}
$$

$$
8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0
$$

Verification

$$
\begin{aligned}
&G(C) \cup B(C) = \{ \\
&-s_3 + g_1 g_4, \\
&-s_2 - 2g_1 g_4 + g_4 + g_1, \\
&-g_4 + g_2 g_3, \\
&-s_1 - 2g_2 g_3 + g_3 + g_2, \\
&-g_1 + a_1 b_1, \\
&-g_2 + a_0 b_1, \\
&-g_3 + a_1 b_0, \\
&-s_0 + a_0 b_0, \\
&-a_1^2 + a_1, \\
&-a_0^2 + a_0, \\
&-b_1^2 + b_1, \\
&-b_0^2 + b_0\}
\end{aligned}
$$

$$8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0$$

$$8g_1 g_4 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0$$

Verification

$$
\begin{aligned}
G(C) \cup B(C) = \{ & \\
& -s_3 + g_1 g_4, \\
& -s_2 - \boxed{2g_1 g_4} + g_4 + g_1, \\
& -g_4 + g_2 g_3, \\
& -s_1 - 2g_2 g_3 + g_3 + g_2, \\
& -g_1 + a_1 b_1, \\
& -g_2 + a_0 b_1, \\
& -g_3 + a_1 b_0, \\
& -s_0 + a_0 b_0, \\
& -a_1^2 + a_1, \\
& -a_0^2 + a_0, \\
& -b_1^2 + b_1, \\
& -b_0^2 + b_0 \}
\end{aligned}
$$

$$
\begin{aligned}
& 8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
& \boxed{8g_1 g_4} + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
& 4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0
\end{aligned}
$$

Verification

$$
\begin{aligned}
&G(C) \cup B(C) = \{ \\
&\quad -s_3 + g_1 g_4, \\
&\quad -s_2 - 2g_1 g_4 + g_4 + g_1, \\
&\quad -g_4 + g_2 g_3, \\
&\quad -s_1 - 2g_2 g_3 + g_3 + g_2, \\
&\quad -g_1 + a_1 b_1, \\
&\quad -g_2 + a_0 b_1, \\
&\quad -g_3 + a_1 b_0, \\
&\quad -s_0 + a_0 b_0, \\
&\quad -a_1^2 + a_1, \\
&\quad -a_0^2 + a_0, \\
&\quad -b_1^2 + b_1, \\
&\quad -b_0^2 + b_0\}
\end{aligned}
$$

$$
\begin{aligned}
8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
8g_1 g_4 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_2 g_3 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0
\end{aligned}
$$

Verification

$$
\begin{aligned}
&G(C) \cup B(C) = \{ \\
&\quad -s_3 + g_1 g_4, \\
&\quad -s_2 - 2g_1 g_4 + g_4 + g_1, \\
&\quad -g_4 + g_2 g_3, \\
&\quad -s_1 - \boxed{2g_2 g_3} + g_3 + g_2, \\
&\quad -g_1 + a_1 b_1, \\
&\quad -g_2 + a_0 b_1, \\
&\quad -g_3 + a_1 b_0, \\
&\quad -s_0 + a_0 b_0, \\
&\quad -a_1^2 + a_1, \\
&\quad -a_0^2 + a_0, \\
&\quad -b_1^2 + b_1, \\
&\quad -b_0^2 + b_0\}
\end{aligned}
$$

$$
\begin{aligned}
&8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
&8g_1 g_4 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
&4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
&\boxed{4g_2 g_3} + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
&4g_1 + 2g_3 + 2g_2 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0
\end{aligned}
$$

Verification

$$
\begin{aligned}
&G(C) \cup B(C) = \{ \\
&\quad -s_3 + g_1 g_4, \\
&\quad -s_2 - 2g_1 g_4 + g_4 + g_1, \\
&\quad -g_4 + g_2 g_3, \\
&\quad -s_1 - 2g_2 g_3 + g_3 + g_2, \\
&\quad -g_1 + \boxed{a_1 b_1}, \\
&\quad -g_2 + a_0 b_1, \\
&\quad -g_3 + a_1 b_0, \\
&\quad -s_0 + a_0 b_0, \\
&\quad -a_1^2 + a_1, \\
&\quad -a_0^2 + a_0, \\
&\quad -b_1^2 + b_1, \\
&\quad -b_0^2 + b_0 \}
\end{aligned}
$$

$$
\begin{aligned}
8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
8g_1 g_4 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_2 g_3 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_1 + 2g_3 + 2g_2 + s_0 - \boxed{4a_1 b_1} - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
2g_3 + 2g_2 + s_0 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0
\end{aligned}
$$

Verification

$$
\begin{aligned}
G(C) \cup B(C) = \{ & \\
& -s_3 + g_1 g_4, \\
& -s_2 - 2g_1 g_4 + g_4 + g_1, \\
& -g_4 + g_2 g_3, \\
& -s_1 - 2g_2 g_3 + g_3 + g_2, \\
& -g_1 + a_1 b_1, \\
& -g_2 + \boxed{a_0 b_1}, \\
& -g_3 + a_1 b_0, \\
& -s_0 + a_0 b_0, \\
& -a_1^2 + a_1, \\
& -a_0^2 + a_0, \\
& -b_1^2 + b_1, \\
& -b_0^2 + b_0 \}
\end{aligned}
$$

$$
\begin{aligned}
8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
8g_1 g_4 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_2 g_3 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_1 + 2g_3 + 2g_2 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
2g_3 + 2g_2 + s_0 - 2a_1 b_0 - \boxed{2a_0 b_1} - a_0 b_0 \\
2g_3 + s_0 - 2a_1 b_0 - a_0 b_0
\end{aligned}
$$

Verification

$$
\begin{aligned}
G(C) \cup B(C) = \{ & \\
& -s_3 + g_1 g_4, \\
& -s_2 - 2g_1 g_4 + g_4 + g_1, \\
& -g_4 + g_2 g_3, \\
& -s_1 - 2g_2 g_3 + g_3 + g_2, \\
& -g_1 + a_1 b_1, \\
& -g_2 + a_0 b_1, \\
& -g_3 + \boxed{a_1 b_0}, \\
& -s_0 + a_0 b_0, \\
& -a_1^2 + a_1, \\
& -a_0^2 + a_0, \\
& -b_1^2 + b_1, \\
& -b_0^2 + b_0 \}
\end{aligned}
$$

$$
\begin{aligned}
8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
8g_1 g_4 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_2 g_3 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_1 + 2g_3 + 2g_2 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
2g_3 + 2g_2 + s_0 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
2g_3 + s_0 - \boxed{2a_1 b_0} - a_0 b_0 \\
s_0 - a_0 b_0
\end{aligned}
$$

Verification

$$
\begin{aligned}
G(C) \cup B(C) = \{ & \\
& -s_3 + g_1 g_4, \\
& -s_2 - 2g_1 g_4 + g_4 + g_1, \\
& -g_4 + g_2 g_3, \\
& -s_1 - 2g_2 g_3 + g_3 + g_2, \\
& -g_1 + a_1 b_1, \\
& -g_2 + a_0 b_1, \\
& -g_3 + a_1 b_0, \\
& -s_0 + \boxed{a_0 b_0}, \\
& -a_1^2 + a_1, \\
& -a_0^2 + a_0, \\
& -b_1^2 + b_1, \\
& -b_0^2 + b_0 \}
\end{aligned}
$$

$$
\begin{aligned}
& 8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
& 8g_1 g_4 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
& 4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
& 4g_2 g_3 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
& 4g_1 + 2g_3 + 2g_2 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
& 2g_3 + 2g_2 + s_0 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
& 2g_3 + s_0 - 2a_1 b_0 - a_0 b_0 \\
& s_0 - \boxed{a_0 b_0} \\
& 0
\end{aligned}
$$

Non-Incremental Checking Algorithm

[FMCAD'17, FMSD'19]

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.

Non-Incremental Checking Algorithm

[FMCAD'17, FMSD'19]

Verification Algorithm

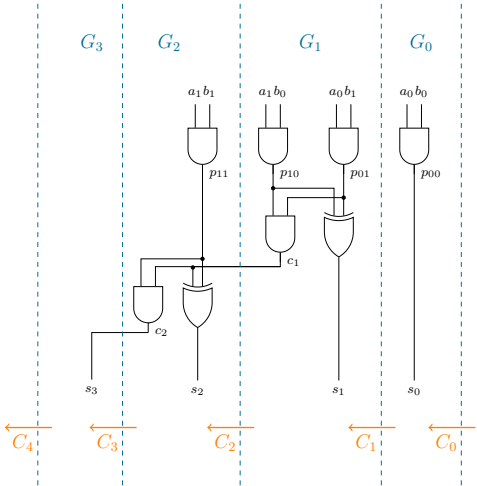
Reduce specification $\sum_{i=0}^{2n-1} 2^i s_i - (\sum_{i=0}^{n-1} 2^i a_i)(\sum_{i=0}^{n-1} 2^i b_i)$ by elements of $G(C) \cup B(C)$ until no further reduction is possible, then C is a multiplier iff remainder is zero.

Computational Problems

- The number of monomials in the intermediate results increases drastically.
- 8-bit multiplier cannot be verified within 20 minutes.

Incremental Verification

[FMCAD'17, FMSD'19]



Let $P_k = \sum_{k=i+j} a_i b_j$.

Column-Wise Checking Algorithm

Input: Circuit C with sliced Gröbner bases G_i
Output: Determine whether C is a multiplier

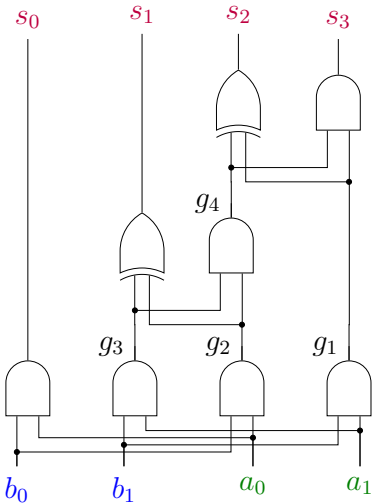
$C_{2n} \leftarrow 0$

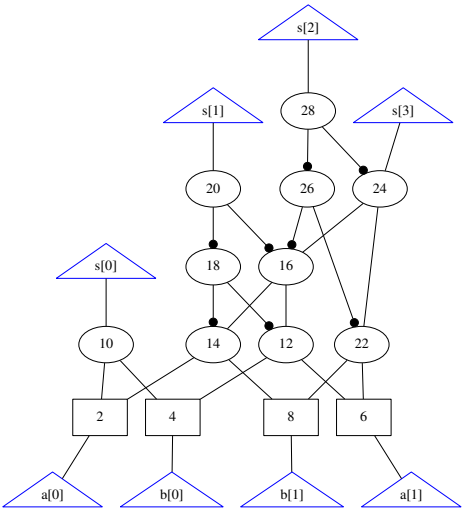
for $i \leftarrow 2n-1$ **to** 0

$C_i \leftarrow$ Remainder $(2C_{i+1} + s_i - P_i,\ \ G_i)$

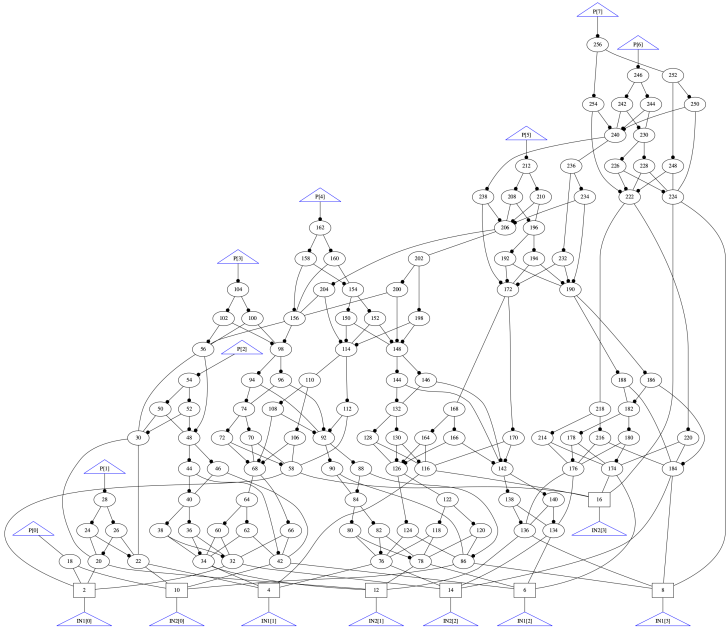
return $C_0 = 0$

AND-Inverter Graphs (AIGs)

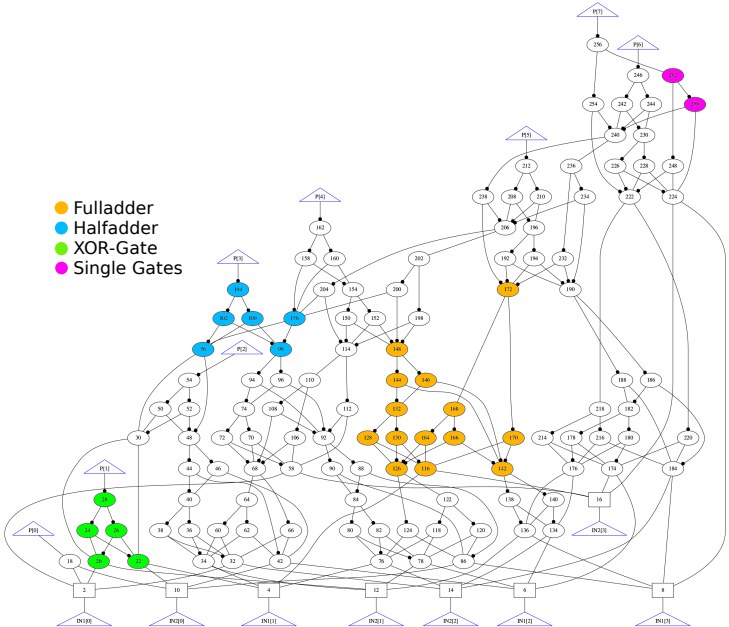




Rewriting


P[7]
256
P[6]
246
252
254
242
244
250
240
230
P[5]
212
236
228
238
248
208
210
234
222
224
250
196
P[4]
162
158
160
202
192
194
232
172
190
P[3]
164
284
154
200
102
100
156
150
152
198
56
P[2]
98
114
146
54
94
96
110
144
140
188
186
50
52
74
108
112
152
168
218
182
30
48
72
70
106
92
128
130
164
166
170
214
178
216
180
220
44
46
68
58
90
88
120
116
142
176
174
184
P[1]
28
40
64
84
122
138
140
16
P[0]
34
26
38
36
60
62
66
80
82
124
118
120
136
134
IN[2][3]
18
20
22
34
32
42
70
78
86
2
10
4
12
14
6
8
IN[1][0]
IN[2][0]
IN[1][1]
IN[2][1]
IN[2][2]
IN[1][2]
IN[1][3]

Rewriting


Fulladder
Halfadder
XOR-Gate
Single Gates
P[7]
P[6]
P[5]
P[4]
P[3]
P[2]
P[1]
P[0]
IN1[0]
IN2[0]
IN1[1]
IN2[1]
IN2[2]
IN1[2]
IN1[3]
IN2[3]

Variable Elimination

Variable elimination is based on **elimination theory of Gröbner bases**.

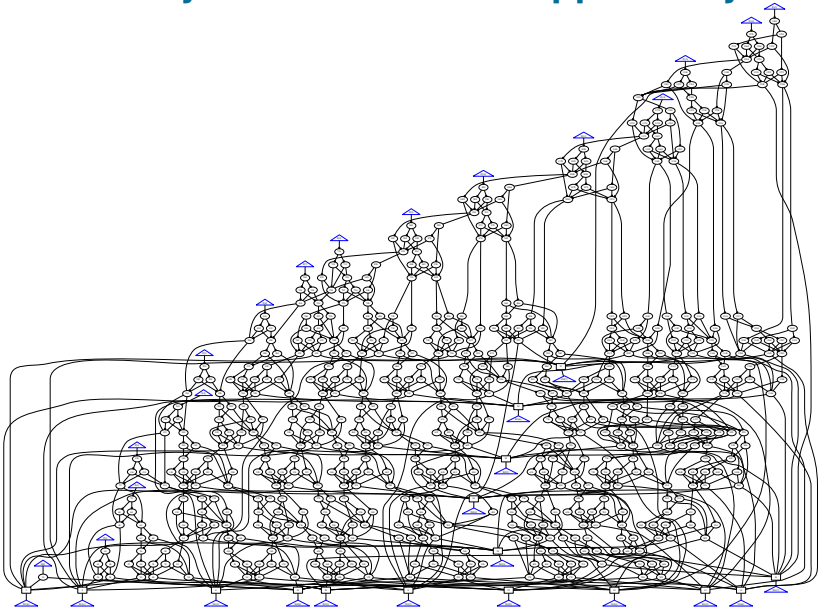
Identify sub-circuits C_S in the AIGs and eliminate internal variables [DATE'18, FMSD'19]:

- Full-adder rewriting
- Half-adder rewriting
- XOR rewriting
- Common rewriting

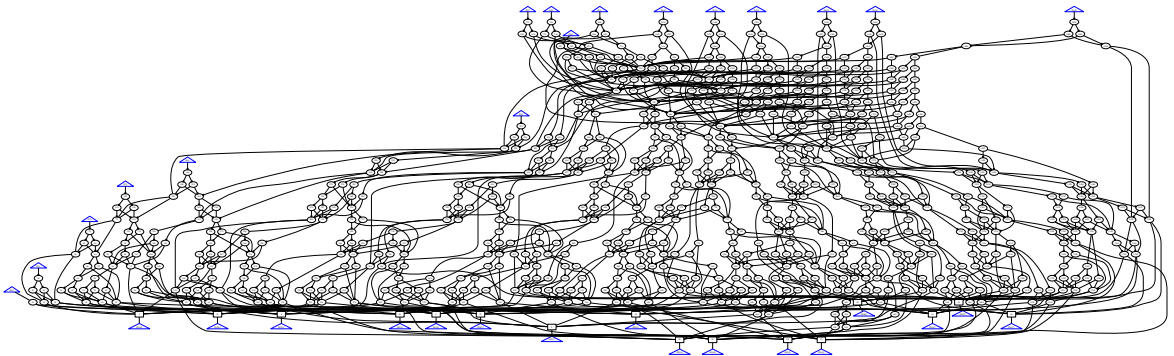
Repeated elimination of variables with single dependency [FMCAD'19]:

- Does not rely on specific patterns.

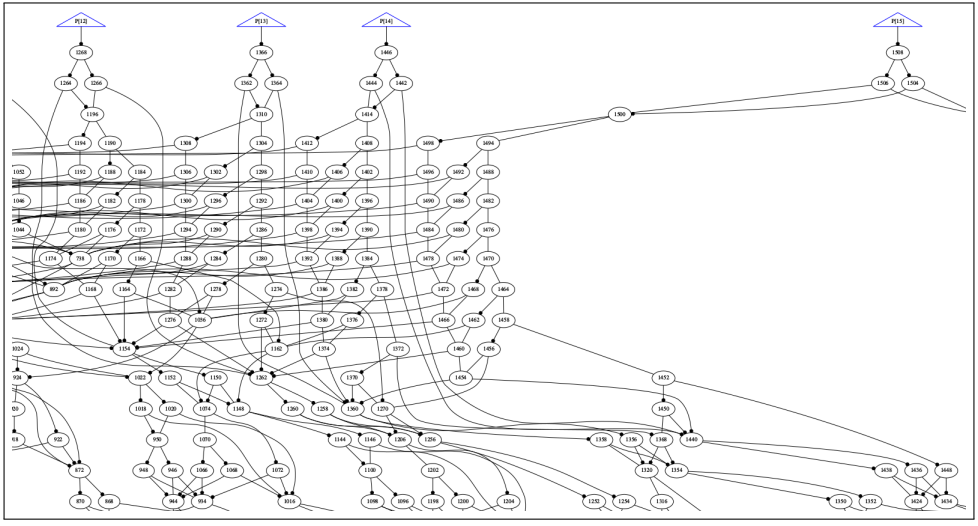
Multiplier – Array Accumulation & Ripple-Carry Adder



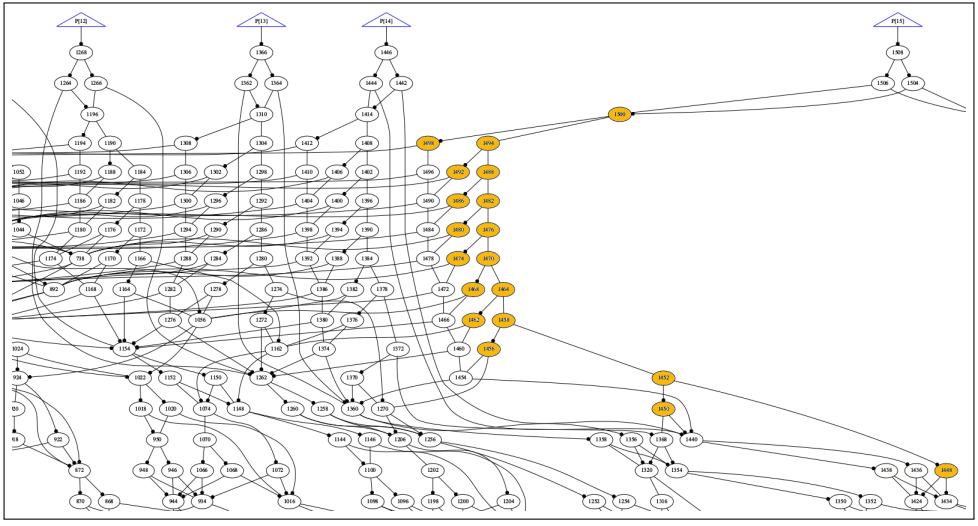
Multiplier – Wallace Tree Acc. & Carry Look-Ahead Adder



Multiplier – Wallace Tree Acc. & Carry Look-Ahead Adder

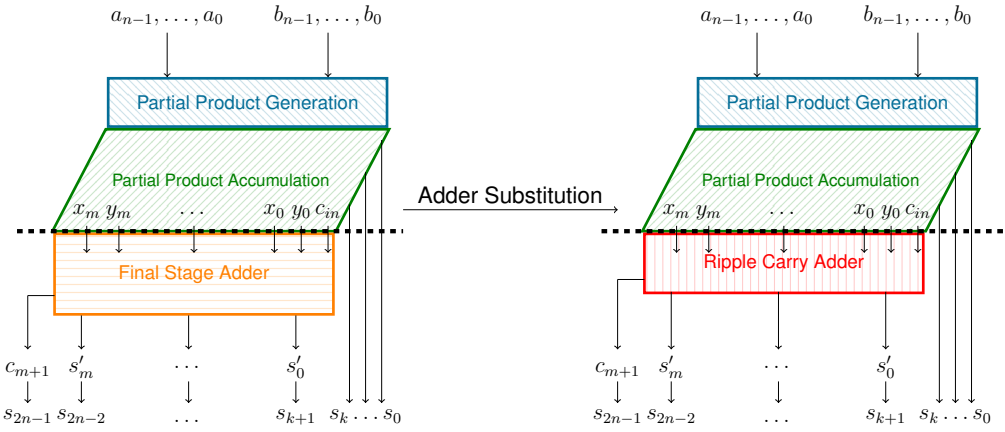


Multiplier – Wallace Tree Acc. & Carry Look-Ahead Adder



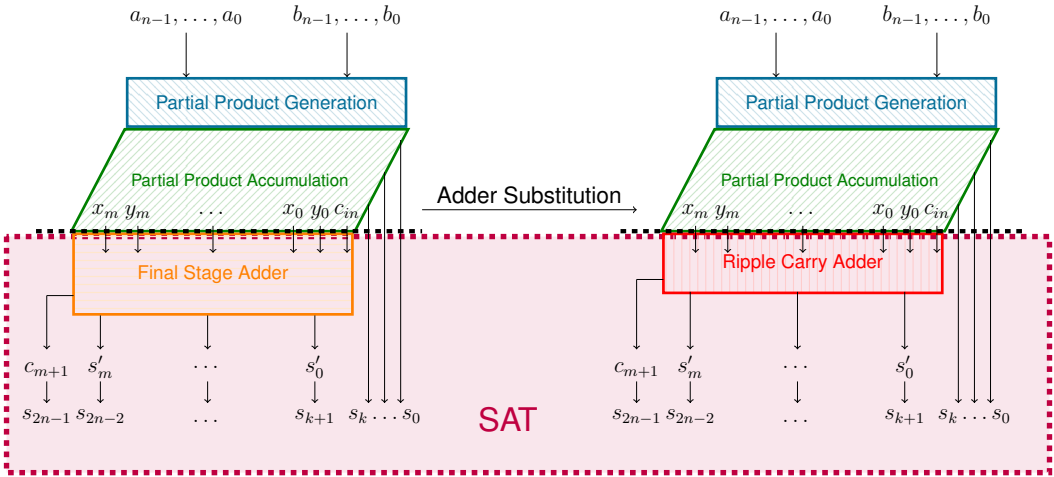
SAT & Computer Algebra

[FMCAD'19]



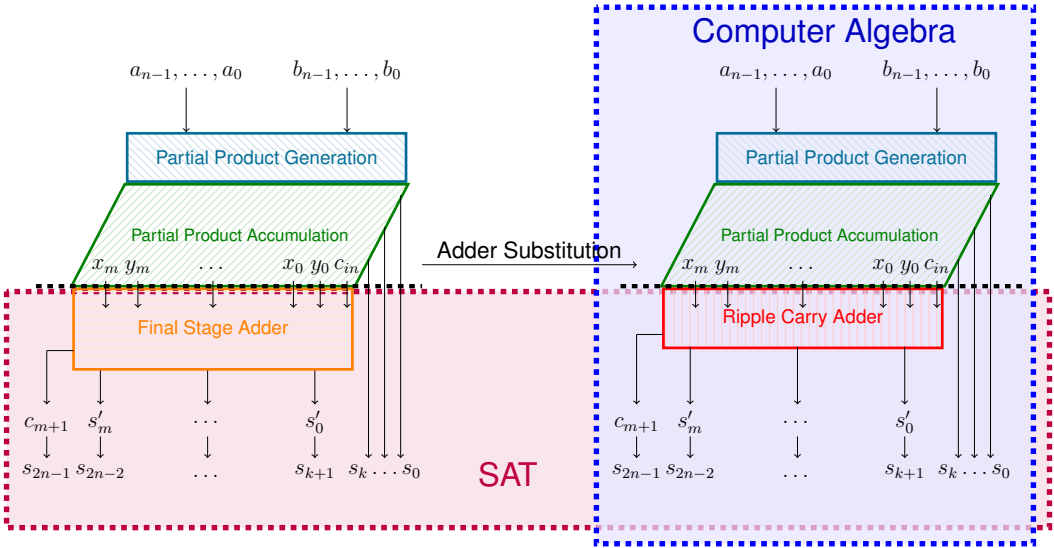
SAT & Computer Algebra

[FMCAD'19]



SAT & Computer Algebra

[FMCAD'19]



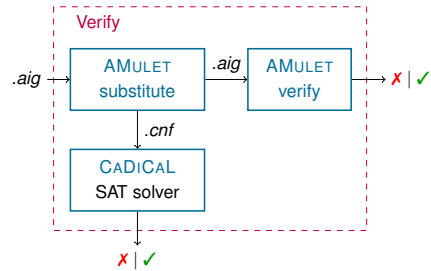
AMulet

Computer Algebra Systems [FMSD'19]

- Mathematica, Singular
- Algorithmic power
  (e.g., Mathematica has > 5000 built-in functions).
- Easy to use.
- Designed for general purposes.

AMULET [FMCAD'19, Vampire'19]

- Automatically applies adder substitution and circuit verification.
- Designed to make use of properties of polynomials.
- On-the-fly reduction of Boolean value constraints.

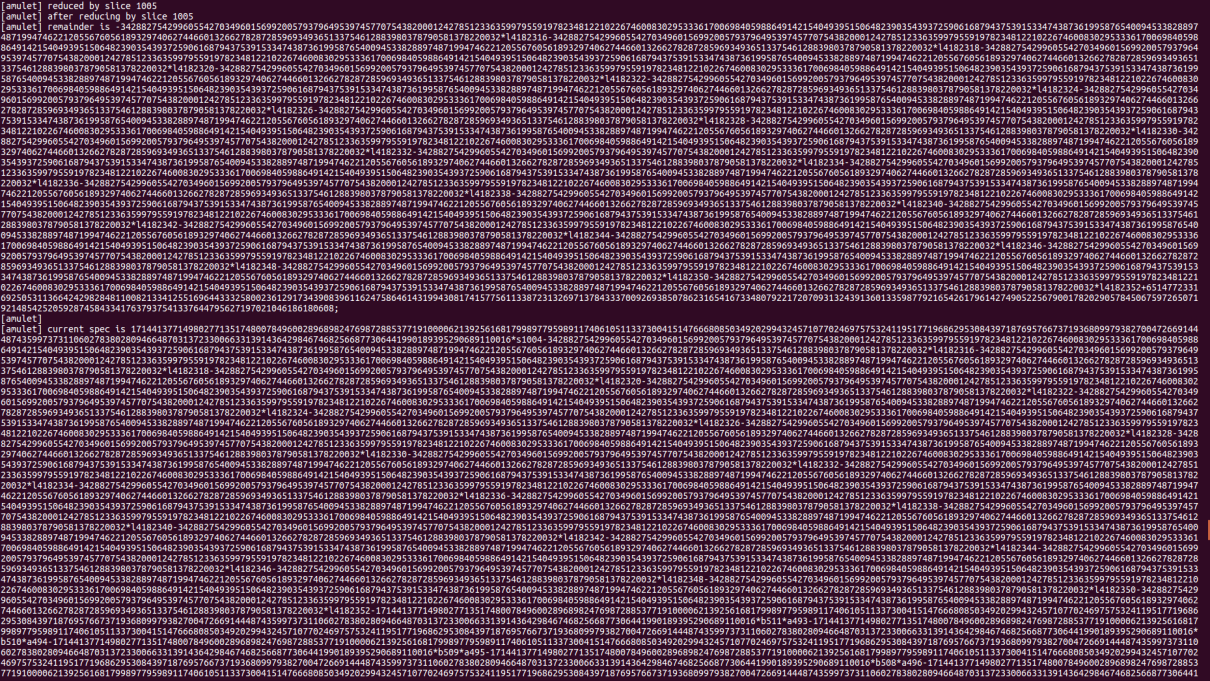


```
daniela@daninovo:~/algprop/src/amulet$ ./amulet ../benchmarks/btor/btor512.aig -verify -v1
```

```
daniela@daninovo:~/algprop/src/amulet$ ./amulet ../benchmarks/btor/btor512.aig -verify -v1
[amulet] AMulet Version 001
[amulet] Aiger multiplier examination tool
[amulet] Copyright (C) 2019, Daniela Kaufmann, Johannes Kepler University Linz
[amulet] reading '../benchmarks/btor/btor512.aig'
[amulet] found regular multiplier: N * N = 2N
[amulet] MILOA 2092544 1024 0 1024 2091520
[amulet] assuming ordering as in BTOR generated benchmarks
[amulet] (a[0], a[1], ..., a[511]) = (input[0], input[2], ..., input[1022])
[amulet] (b[0], b[1], ..., b[511]) = (input[1], input[3], ..., input[1023])
[amulet] (s[0], ..., s[1023]) = (output[0], ..., output[1023])
[amulet] maximum level of variables is 4085
[amulet] found 262144 partial products
[amulet] assuming simple pp generator
[amulet] assigning slices to variables
[amulet] totally merged 261120 variable(s)
[amulet] totally promoted 0 variable(s)
```

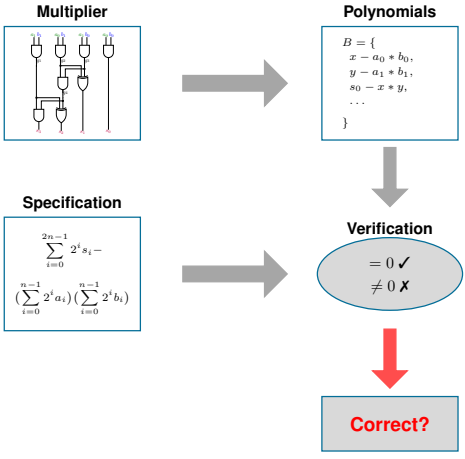
```
daniela@daninovo:~/algprop/src/amulet$ ./amulet ../benchmarks/btor/btor512.aig -verify -v1
[amulet] AMulet Version 001
[amulet] Aiger multiplier examination tool
[amulet] Copyright (C) 2019, Daniela Kaufmann, Johannes Kepler University Linz
[amulet] reading '../benchmarks/btor/btor512.aig'
[amulet] found regular multiplier: N * N = 2N
[amulet] MILOA 2092544 1024 0 1024 2091520
[amulet] assuming ordering as in BTOR generated benchmarks
[amulet] (a[0], a[1], ..., a[511]) = (input[0], input[2], ..., input[1022])
[amulet] (b[0], b[1], ..., b[511]) = (input[1], input[3], ..., input[1023])
[amulet] (s[0], ..., s[1023]) = (output[0], ..., output[1023])
[amulet] maximum level of variables is 4085
[amulet] found 262144 partial products
[amulet] assuming simple pp generator
[amulet] assigning slices to variables
[amulet] totally merged 261120 variable(s)
[amulet] totally promoted 0 variable(s)
[amulet] filled 1024 slices
[amulet] ordered 2093568 variables
[amulet] decomposed 523775 variables
[amulet] final reduction started
[amulet] current spec is 8988465674311579538646525953945123668089884894711532863671504057886633790275048156635423866120376801056005693993569667882939488440720831124642371531973706218888394671243274263815110980062304705972654147604
2502884419075341171231440736956555270413618581675255342293149119973622969239858152417678164812112068608*s1023;
[amulet] reduced by slice 1023
[amulet] after reducing by slice 1023
[amulet] remainder is -898846567431157953864652595394512366808988489471153286367150405788663379027504815663542386612037680105600569399356966788293948844072083112464237153197370621888839467124327426381511098006230470597265414760425
02884419075341171231440736956555270413618581675255342293149119973622969239858152417678164812112068608*l4185088+89884656743115795386465259539451236680898848947115328636715040578866337902750481566354238661203768010560056939935696678
829394884407208311246423715319737062188883946712432742638151109800623047059726541476042502884419075341171231440736956555270413618581675255342293149119973622969239858152417678164812112068608;
[amulet]
[amulet] current spec is 4494232837155789769323262976972561834044942447355766431835752028943316895137524078317711933060188400528002846996784833941469744220360415562321185765986853109444197335621637131907555490031152352986327073802
1251442209537670585615720368478277635206809290837627671146574559986811484619929076208839082406056034304*s1022-8988465674311579538646525953945123668089884894711532863671504057886633790275048156635423866120376801056005693993569667882939488440720831124642371531973706218888394671243274263815110980062304705972654147604250288441907534117123144073695655527041361858167525534229314911997362296923985815241767816481211206860 8*l4185088-4494232837155789769323262976972561834044942447355766431835752028943316895137524078317711933060188400528002846996784833941469744220360415562321185765986853109444197335621637131907555490031152352986327073802125144220953767058561572036847827763520680929083762767114
6574559986811484619929076208839082406056034304*b511*a511+8988465674311579538646525953945123668089884894711532863671504057886633790275048156635423866120376801056005693993569667882939488440720831124642371531973706218888394671243274263815110980062304705972654147604250288441907534117123144073695655527041361858167525534229314911997362296923985815241767816481211206860 8;
[amulet] reduced by slice 1022
[amulet] after reducing by slice 1022
[amulet] remainder is -449423283715578976932326297697256183404494244735576643183575202894331689513752407831771193306018840052800284699678483394146974422036041556232118576598685310944419733562163713190755549003115235298632707380212
5144220953767058561572036847827763520680929083762767114657455998681148461992907620883908240605603430 4*l4185070-44942328371557897693232629769725618340449424473557664318357520289433168951375240783177119330601884005280028469967848339
4146974422036041556232118576598685310944419733562163713190755549003115235298632707380212514422095376705856157203684782776352068092908376276711465745599868114846199290762088390824060560343 04*l4185072+8988465674311579538646525953945
1236680898848947115328636715040578866337902750481566354238661203768010560056939935696678829394884407208311246423715319737062188883946712432742638151109800623047059726541476042502884419075341171231440736956555270413618581675255342 2
931491199736229692398581524176781648121120686 08;
[amulet]
[amulet] current spec is 2247116418577894884661631488486280917022471223677883215917876014471658447568762039158855966530094200264001423498392416970734872110180207781160592882993426554722098667810818565953777745015576176493163536901
0625721104768835292807860184239138817603404645418813835573287279993405742309964538104419541203028017152*s1021-449423283715578976932326297697256183404494244735576643183575202894331689513752407831771193306018840052800284699678483394
1469744220360415562321185765986853109444197335621637131907555490031152352986327073802125144220953767058561572036847827763520680929083762767114657455998681148461992907620883908240605603430 4*l4185070-4494232837155789769323262976972 5
6183404494244735576643183575202894331689513752407831771193306018840052800284699678483394146974422036041556232118576598685310944419733562163713190755549003115235298632707380212514422095376705856157203684782776352068092908376276711 4
6574559986811484619929076208839082406056034304*l4185072-2247116418577894884661631488486280917022471223677883215917876014471658447568762039158855966530094200264001423498392416970734872110180207781160592882993426554722098667810818 56
5953777745015576176493163536901062572110476883529280786018423913881760340464541881383557328727999340574230996453810441954120302801715 2*b511*a510-2247116418577894884661631488486280917022471223677883215917876014471658447568762039158
8559665300942002640014234983924169707348721101802077811605928829934265547220986678108185659537777450155761764931635369010625721104768835292807860184239138817603404645418813835573287279993405742309964538104419541203028017152*b510*a
511+8988465674311579538646525953945123668089884894711532863671504057886633790275048156635423866120376801056005693993569667882939488440720831124642371531973706218888394671243274263815110980062304705972654147604250288441907534117123
1440736956555270413618581675255342293149119973622969239858152417678164812112068608;
[amulet] reduced by slice 1021
[amulet] after reducing by slice 1021
[amulet] remainder is -224711641857789488466163148848628091702247122367788321591787601447165844756876203915885596653009420026400142349839241697073487211018020778116059288299342655472209866781081856595377774501557617649316353690106
2572110476883529280786018423913881760340464541881383557328727999340574230996453810441954120302801715 2*l4185036-2247116418577894884661631488486280917022471223677883215917876014471658447568762039158855966530094200264001423498392416 9
7073487211018020778116059288299342655472209866781081856595377774501557617649316353690106257211047688352928078601842391388176034046454188138355732872799934057423099645381044195412030280171 52*l4185038-2247116418577894884661631488486
2809170224712236778832159178760144716584475687620391588559665300942002640014234983924169707348721101802077811605928829934265547220986678108185659537777450155761764931635369010625721104768835292807860184239138817603404645418813835 5
7328727999340574230996453810441954120302801715 2*l4185040+67413492557336846539848944654588427510674136710336496477536280434149753427062861174765678995902826007920042704951772509122046163305406233434817786489802796641662960034324556
97861333235046728529479490610703187716331430650587842358055271741645281021393625644150671986183998021722692989361431325862360908405145 6;
[amulet]
[amulet] current spec is 1123558209288947442330815744243140458511235611838941607958938007235829223784381019579427983265047100132000711749196208485367436055090103890580296441496713277361049333905409282976888872507788088246581768450
5312860552384417646403930092119569408801702322709406917786643639996702871154982269052209770601514008576*s1020-2247116418577894884661631488486280917022471223677883215917876014471658447568762039158855966530094200264001423498392416 97
0734872110180207781160592882993426554722098667810818565953777745015576176493163536901062572110476883529280786018423913881760340464541881383557328727999340574230996453810441954120302801715 2*l4185036-2247116418577894884661631488486 2
8091702247122367788321591787601447165844756876203915885596653009420026400142349839241697073487211018020778116059288299342655472209866781081856595377774501557617649316353690106257211047688352928078601842391388176034046454188138355 7
```

[amdlet] reduced to slice 1005
[amdlet] after reducing by slice 1005
[amdlet] after emainder 1005 1005 1005



```
[amulet]
[amulet] current spec is 512*s9+1024*l2696-1024*l2702-1024*l2704-1024*l2706-1024*l2708-1024*l2710-1024*l2712-1024*l2714-1024*l2716-512*b9*a0-512*b8*a1-512*b7*a2-512*b6*a3-512*b5*a4-512*b4*a5-512*b3*a6-512*b2*a7-512*b1*a8-512*b0*a9
+8192;
[amulet] reduced by slice 9
[amulet] after reducing by slice 9
[amulet] remainder is 512*l2560-512*l2566-512*l2568-512*l2570-512*l2572-512*l2574-512*l2576-512*l2578+3584;
[amulet]
[amulet] current spec is 256*s8+512*l2560-512*l2566-512*l2568-512*l2570-512*l2572-512*l2574-512*l2576-512*l2578-256*b8*a0-256*b7*a1-256*b6*a2-256*b5*a3-256*b4*a4-256*b3*a5-256*b2*a6-256*b1*a7-256*b0*a8+3584;
[amulet] reduced by slice 8
[amulet] after reducing by slice 8
[amulet] remainder is 256*l2440-256*l2446-256*l2448-256*l2450-256*l2452-256*l2454-256*l2456+1536;
[amulet]
[amulet] current spec is 128*s7+256*l2440-256*l2446-256*l2448-256*l2450-256*l2452-256*l2454-256*l2456-128*b7*a0-128*b6*a1-128*b5*a2-128*b4*a3-128*b3*a4-128*b2*a5-128*b1*a6-128*b0*a7+1536;
[amulet] reduced by slice 7
[amulet] after reducing by slice 7
[amulet] remainder is 128*l2336-128*l2342-128*l2344-128*l2346-128*l2348-128*l2350+640;
[amulet]
[amulet] current spec is 64*s6+128*l2336-128*l2342-128*l2344-128*l2346-128*l2348-128*l2350-64*b6*a0-64*b5*a1-64*b4*a2-64*b3*a3-64*b2*a4-64*b1*a5-64*b0*a6+640;
[amulet] reduced by slice 6
[amulet] after reducing by slice 6
[amulet] remainder is 64*l2248-64*l2254-64*l2256-64*l2258-64*l2260+256;
[amulet]
[amulet] current spec is 32*s5+64*l2248-64*l2254-64*l2256-64*l2258-64*l2260-32*b5*a0-32*b4*a1-32*b3*a2-32*b2*a3-32*b1*a4-32*b0*a5+256;
[amulet] reduced by slice 5
[amulet] after reducing by slice 5
[amulet] remainder is 32*l2176-32*l2182-32*l2184-32*l2186+96;
[amulet]
[amulet] current spec is 16*s4+32*l2176-32*l2182-32*l2184-32*l2186-16*b4*a0-16*b3*a1-16*b2*a2-16*b1*a3-16*b0*a4+96;
[amulet] reduced by slice 4
[amulet] after reducing by slice 4
[amulet] remainder is 16*l2120-16*l2126-16*l2128+32;
[amulet]
[amulet] current spec is 8*s3+16*l2120-16*l2126-16*l2128-8*b3*a0-8*b2*a1-8*b1*a2-8*b0*a3+32;
[amulet] reduced by slice 3
[amulet] after reducing by slice 3
[amulet] remainder is 8*l2080-8*l2086+8;
[amulet]
[amulet] current spec is 4*s2+8*l2080-8*l2086-4*b2*a0-4*b1*a1-4*b0*a2+8;
[amulet] reduced by slice 2
[amulet] after reducing by slice 2
[amulet] remainder is 4*l2056;
[amulet]
[amulet] current spec is 2*s1+4*l2056-2*b1*a0-2*b0*a1;
[amulet] reduced by slice 1
[amulet] after reducing by slice 1
[amulet] remainder is 0;
[amulet]
[amulet] current spec is s0-b0*a0;
[amulet] reduced by slice 0
[amulet] after reducing by slice 0
[amulet] remainder is 0;
[amulet]
[amulet]
[amulet] final remainder is 0;
[amulet]
[amulet] CORRECT MULTIPLIER
[amulet]
[amulet] maximun polynomial size is:                     1540 monomials
[amulet] maximun resident set size:                   1711.02 MB
[amulet] used time for reading/slicing/ordering:        50.39 seconds
[amulet] used time for decompositlon:                   26.19 seconds
[amulet] used time for elimination:                      0.00 seconds
[amulet] used time for reduction:                     1047.45 seconds
[amulet] used process time:                           1124.03 seconds
dantela@dantnovo:~/algprop/src/amulet$
```

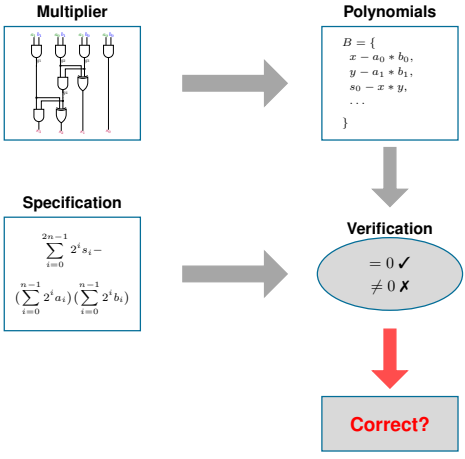
Proofs



Problem:

- Can we trust our own implementation?
- Is the verification process correct?

Proofs



Problem:

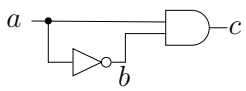
- Can we trust our own implementation?
- Is the verification process correct?

Solution:

Validate result of verification process.

- Generate machine-checkable proofs.
- Check by independent proof checkers.

Practical Algebraic Calculus



$$
\begin{aligned}
G(C) &= \{-b+1-a,\ -c+ab\} \\
B(C) &= \{-a^2+a\} \\
\text{Spec} &= c
\end{aligned}
$$

[SC2'18]

```
* : -b+1-a,        a,         -a*b+a-a^2;
* : -a^2+a,        -1,        a^2-a;
+ : -a*b+a-a^2,    a^2-a,     -a*b;
+ : -a*b,          -c+a*b,    -c;
* : -c,            -1,        c;
```

[FMCAD'19]

```
* : -b+1-a,    a,         -a*b;
+ : -a*b,      -c+a*b,    -c;
* : -c,        -1,        c;
```

[Submitted]

```
3   * 1,   a,    -a*b;
4   + 3,   2,    -c;
5   * 5,  -1,    c;
```

Proof Checker

PACTRIM [SC2'18, FMCAD'19]

- Written in C (~ 2000 lines of code).
- Supports older PAC formats, where antecedents are given explicitly.

PACHECK [Submitted]

- Written in C (~ 2200 lines of code).
- Supports new calculus.
- Backward compatible to older PAC formats.

PASTÈQUE [Submitted]

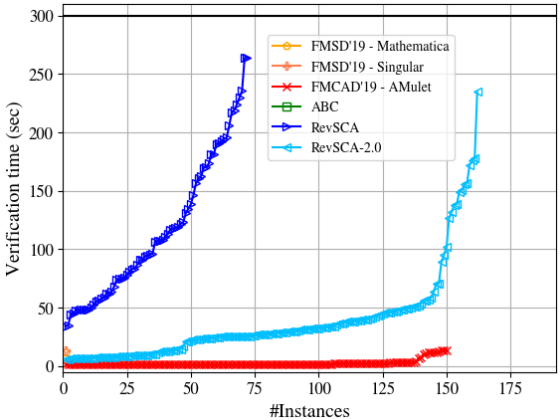
- Implemented and verified using Isabelle/HOL by M. Fleury.
- Supports new calculus.

Recent Related Work

- M. Ciesielski, C. Yu, W. Brown, D. Liu, and A. Rossi. Verification of Gate-level Arithmetic Circuits by Function Extraction. In DAC, 2015.
- C. Yu, M. Ciesielski, and A. Mishchenko. Fast Algebraic Rewriting Based on And-Inverter Graphs. IEEE TCAD, 2018.
- M. Ciesielski, T. Su, A. Yasin, and C. Yu. Understanding Algebraic Rewriting for Arithmetic Circuit Verification: a Bit-Flow Model. IEEE TCAD, 2019.

- 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'16.
- A. Mahzoon, D. Große, and R. Drechsler. PolyCleaner: Clean your Polynomials before Backward Rewriting to verify Million-gate Multipliers. In ICCAD'18.
- A. Mahzoon, D. Große, and R. Drechsler. RevSCA: Using Reverse Engineering to Bring Light into Backward Rewriting for Big and Dirty Multipliers. In DAC'19.
- A. Mahzoon, D. Große, C. Scholl, and R. Drechsler. Towards Formal Verification of Optimized and Industrial Multipliers. To appear in DATE, 2020.

Evaluation – Verification

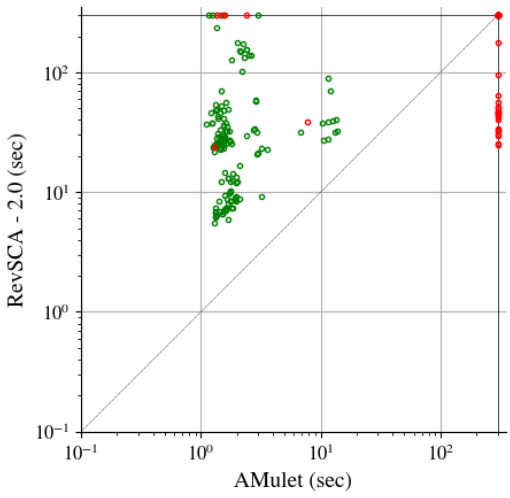


Comparing our approaches to most recent related work.

- Our work
  [FMSD'19] [FMCAD'19]
- Yu et al. [ABC]
- Mahzoon et al.
  [RevSCA] [RevSCA-2.0]

- 192 different multipliers
- Input bit-width $n = 64$
- Time-limit: 300 sec

Evaluation – Verification

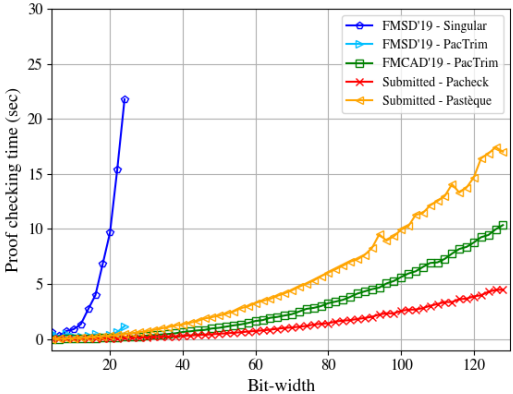


Detailed comparison of AMULET and RevSCA-2.0 on the same benchmarks.

- Multipliers using a
  - "(7,3) counter tree" or a
  - "redundant binary addition tree".
- Remaining 144 benchmarks.

		AMULET		total
		solved	TO	
RevSCA-2.0	solved	141	21	162
	TO	8	10	18
	buggy	1	11	12
total		150	42	192

Evaluation – Certification & Proof Checking



Comparison of our approaches
[FMSD'19] [FMCAD'19] [Submitted]

None of related work generates certificates.

- Array ripple-carry multiplier
- Input bit-width n in $[2, 128]$
- Time-limit: 300 sec

Evaluation – Certification & Proof Checking

Verification Tool AMulet & Proof Checker Pacheck.

multiplier	bit-width	Verify	Certify	Check	total	AIG size	proof length
btor	512	18m 44s	23m 03s	6m 34s	29m 37s	2.1 M	7.8 M
kjvnkv	512	13m 03s	15m 26s	9m 13s	24m 39s	3.1 M	12.3 M
sp-ar-rc	512	13m 00s	15m 35s	9m 46s	25m 21s	3.1 M	12.3 M
sp-dt-lf	512	26m 25s	25m 19s	11m 28s	36m 47s	3.1 M	12.3 M
sp-wt-bk	512	26m 34s	26m 09s	11m 30s	37m 39s	3.2 M	12.4 M
btor	1024	2h 58m 40s	3h 38m 31s	51m 23s	4h 29m 54s	8.4 M	31.4 M
kjvnkv	1024	1h 32m 50s	2h 52m 07s	1h 11m 56s	4h 04m 03s	12.6 M	49.2 M
btor	2048	25h 09m 50s	42h 32m 05s	7h 10m 08s	49h 42m 13s	33.5 M	125.8 M
kjvnkv	2048	19h 06m 30s	34h 37m 02s	20h 27m 33s	55h 04m 35s	50.3 M	197.0 M

Impact

Contributions:

- 10 publications (6 peer-reviewed, 2 invited papers, 1 technical report, 1 submitted)
- AMULET: Reduction engine for multipliers given as AIGs [FMCAD'19]
- PACTRIM, PACHECK: Proof checker for PAC [SC2'18, Submitted]

Awards:

- 2017: Best Paper Award [FMCAD'17]

Thesis: Formal Verification of Multiplier Circuits using Computer Algebra

- 6 publications (4 peer-reviewed, 1 invited paper, 1 submitted):
  [FMSD'19] [SC2'18] [FMCAD'19] [Vampire'19] [DATE'20] [Submitted]
- Extended with additional unpublished material.

Conclusion

Contributions covered in this talk:

1. Precise formalization of circuit verification including soundness and completeness proofs.
2. Fastest approach for multiplier verification:
   - Incremental Verification
   - Variable Elimination
   - SAT and computer algebra
3. Unique: Generation of proof certificates.

What I did NOT talk about today (but is part of my work):

- Generalization to more general polynomial rings that allow modular reasoning.
- Equivalence checking.
- Combining proof certificates from SAT and computer algebra.
- Adding extensions and deletions to PAC.

References I

[FMCAD'17] D. Ritirc, A. Biere and M. Kauers. Column-Wise Verification of Multipliers Using Computer Algebra. In Proc. of FMCAD'17, pages 23–30, IEEE, 2017.

[DATE'18] D. Ritirc, A. Biere and M. Kauers. Improving and Extending the Algebraic Approach for Verifying Gate-Level Multipliers. In Proc. of DATE'18, pages 1556–1561, IEEE, 2018.

[SC2'18] D. Ritirc, A. Biere and M. Kauers. A Practical Polynomial Calculus for Arithmetic Circuit Verification. In Proc. of SC'2, pages 61–76, CEUR-WS, 2018.

[FMSD'19] D. Kaufmann, A. Biere and M. Kauers. Incremental Column-Wise Verification of Arithmetic Circuits Using Computer Algebra. To appear in FMSD.

[FMCAD'19] D. Kaufmann, A. Biere and M. Kauers. Verifying Large Multipliers by Combining SAT and Computer Algebra. In Proc. of FMCAD'19, pages 28–36, IEEE, 2019.

References II

[Vampire'19] D. Kaufmann, A. Biere and M. Kauers. SAT, Computer Algebra, Multipliers. In Proc. of Vampire Workshop'19, EPiC series, vol. 71, pages 1–18, EasyChair, 2020.

[DATE'20] D. Kaufmann, A. Biere and M. Kauers. From DRUP to PAC and Back. To appear.

[Submitted] D. Kaufmann, M. Fleury and A. Biere. Pacheck and Pastèque Checking Practical Algebraic Calculus Proofs. Submitted.

[BB'65] B. Buchberger. Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal. PhD Thesis, University of Innsbruck, 1965.

[CYBLR'15] M. Ciesielski, C. Yu, W. Brown, D. Liu, and A. Rossi. Verification of Gate-level Arithmetic Circuits by Function Extraction. In Proc. of DAC'15, pages 52:1–52:6, ACM, 2015.

[ABC] M. J. Ciesielski, T. Su, A. Yasin and C. Yu. Understanding Algebraic Rewriting for Arithmetic Circuit Verification: a Bit-Flow Model. To appear in IEEE TCAD.

References III

[SGKSD'16] 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 Proc. of DATE'16, pages 1048–1053, IEEE, 2016.

[RevSCA] A. Mahzoon, D. Große and R. Drechsler. RevSCA: Using Reverse Engineering to Bring Light into Backwar Rewriting for Big and Dirty Multipliers. In Proc. of DAC'19, pages 185:1–185:6, ACM, 2019.

[RevSCA-2.0] A. Mahzoon, D. Große and R. Drechsler. RevSCA-2.0. `https://github.com/amahzoon/RevSCA-2.0`

FORMAL VERIFICATION OF MULTIPLIER CIRCUITS USING COMPUTER ALGEBRA

Rigorosum

Daniela Kaufmann

Institute for Formal Models and Verification
Johannes Kepler University
Linz, Austria

April 21, 2020

JOHANNES KEPLER
UNIVERSITY LINZ