FORMAL VERIFICATION OF INTEGER MULTIPLIER CIRCUITS USING SAT AND COMPUTER ALGEBRA

Keynote

Daniela Kaufmann

Johannes Kepler University

Linz, Austria

14th International Workshop on Boolean Problems

Bremen, Germany (online)

September 24, 2020

JOHANNES KEPLER UNIVERSITY LINZ

Multiplication

$$3 \cdot 3 = ?$$

Multiplication

$$3 \cdot 3 = 9$$

Multiplication

$$11 \cdot 11 = ?$$

Multiplication

$$\begin{array}{r} 1\ 1 \\ \cdot\ 1\ 1 \\ \hline \end{array}$$

$3 \cdot 3 = 9$

Multiplication

$$\begin{array}{rrcrr} 1 & 1 & \cdot & 1 & 1 \\ \hline & & & 1 & 1 \end{array}$$

$3 \cdot 3 = 9$

Multiplication

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

$3 \cdot 3 = 9$

Multiplication

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

$3 \cdot 3 = 9$

Multiplication

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

$3 \cdot 3 = 9$

Multiplication

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

$3 \cdot 3 = 9$

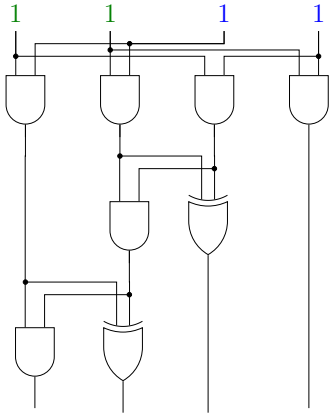
Multiplication

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

$3 \cdot 3 = 9$

Multiplication

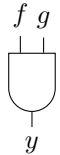
$$
\begin{array}{cccc}
1 & 1 & \cdot & 1 & 1 \\
\hline
 & & & 1 & 1 \\
 & & 1 & 1 & \\
 & 1 & 1 & 0 & \\
\hline
 & 1 & 0 & 0 & 1
\end{array}
$$



$3 \cdot 3 = 9$

Multiplication

AND-Gate



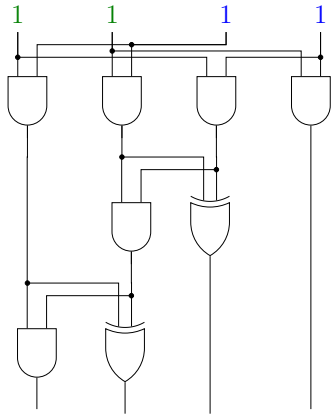
$f \wedge g = y$

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

XOR-Gate

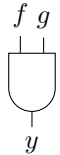
$f \oplus g = y$

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



Multiplication

AND-Gate



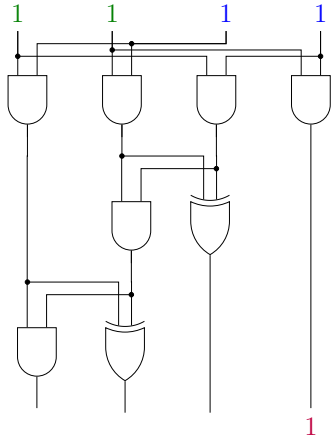
$f \wedge g = y$

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

XOR-Gate

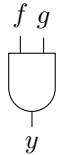
$f \oplus g = y$

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



Multiplication

AND-Gate



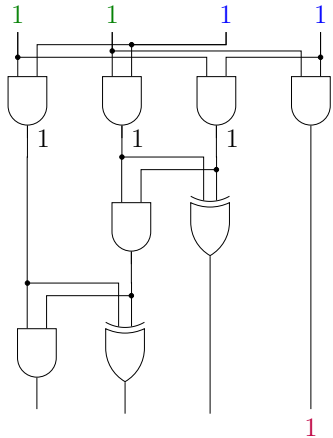
$f \wedge g = y$

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

XOR-Gate

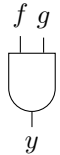
$f \oplus g = y$

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



Multiplication

AND-Gate



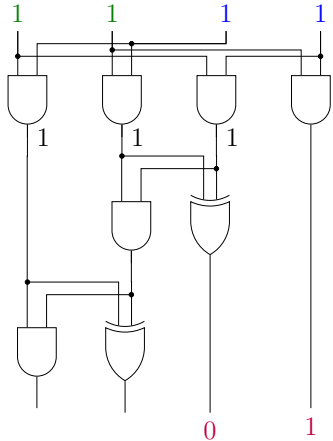
$f \wedge g = y$

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

XOR-Gate

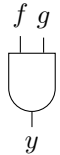
$f \oplus g = y$

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



Multiplication

AND-Gate



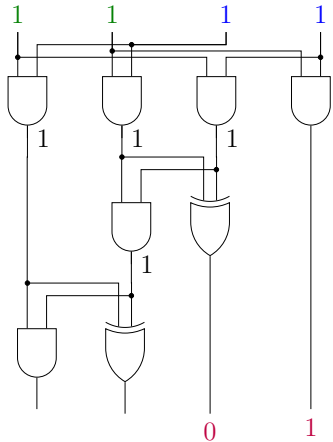
$f \wedge g = y$

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

XOR-Gate

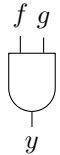
$f \oplus g = y$

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



Multiplication

AND-Gate



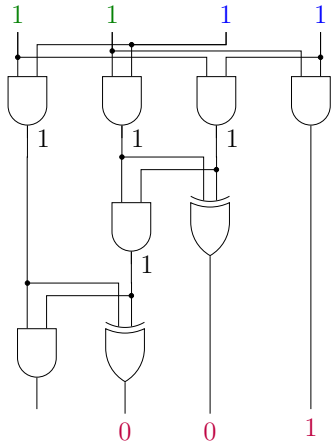
$f \wedge g = y$

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

XOR-Gate

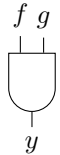
$f \oplus g = y$

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



Multiplication

AND-Gate



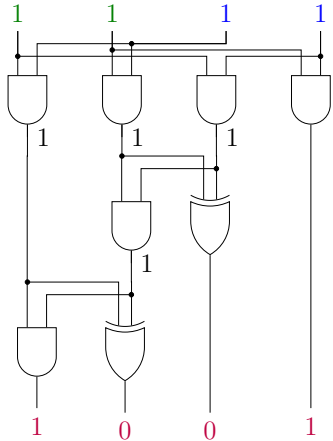
$f \wedge g = y$

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

XOR-Gate

$f \oplus g = y$

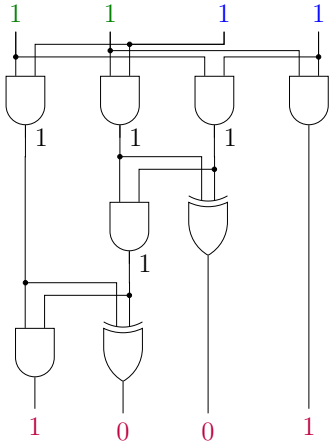
f	g	y
0	0	0
0	1	1
1	0	1
1	1	0



Is the circuit correct?

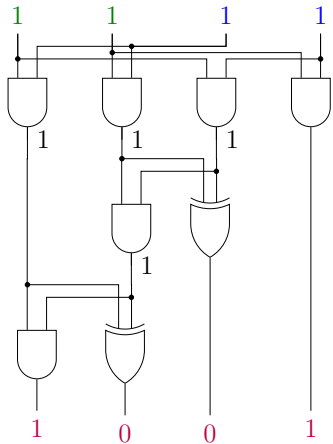
Multiplier circuit

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



Multiplier circuit

Circuit seems correct!



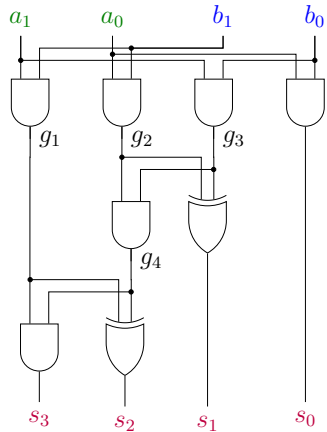
Is the circuit really correct?

Multiplier Circuits

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

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

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



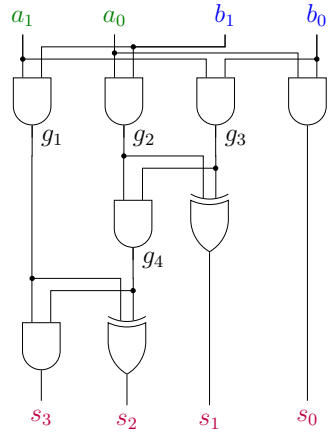
Intel Pentium FDIV Bug 1994

- 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.

Testing

a_1a_0	·	b_1b_0
00	·	00
00	·	01
00	·	10
00	·	11
01	·	00
01	·	01
01	·	10
01	·	11
10	·	00
10	·	01
10	·	10
10	·	11
11	·	00
11	·	01
11	·	10
11	·	11



Testing

Bit-width	Cases
2	16
3	64
4	256
5	1 024
6	4 096
7	16 384
8	65 536

Testing

Bit-width	Cases	
2	16	
3	64	
4	256	
5	1 024	
6	4 096	
7	16 384	
8	65 536	
⋮	⋮	
32	18 446 744 073 709 551 616	quintillion

Testing

Bit-width	Cases	
2	16	
3	64	
4	256	
5	1 024	
6	4 096	
7	16 384	
8	65 536	
⋮	⋮	
32	18 446 744 073 709 551 616	quintillion
⋮	⋮	
64	340 282 366 920 938 463 463 374 607 431 768 211 456	undecillion

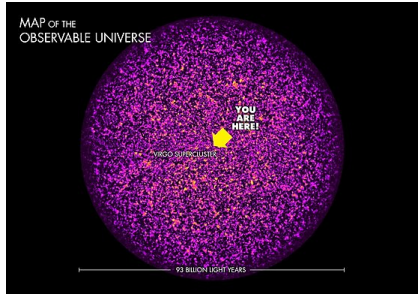
Testing

Bit-width	Cases	
2	16	
3	64	
4	256	
5	1 024	
6	4 096	
7	16 384	
8	65 536	
⋮	⋮	
32	18 446 744 073 709 551 616	quintillion
⋮	⋮	
64	340 282 366 920 938 463 463 374 607 431 768 211 456	undecillion
⋮	⋮	
150	$\sim 10^{90}$	

Testing

Bit-width	Cases
2	16
3	64
4	256
5	1 024
6	4 096
7	16 384
8	65 536
⋮	⋮
32	18 446 744 073 709 551 616
⋮	⋮
64	340 282 366 920 938 463 463 374 607 431 768 211 456
⋮	⋮
150	$\sim 10^{90}$

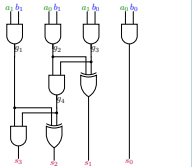
It is estimated that there are between 10^{78} to 10^{82} atoms in the known, observable universe.



Robert Wille: How the World is Run by 0 and 1, TEDxLinz Talk, 2019.

Formal Verification

System



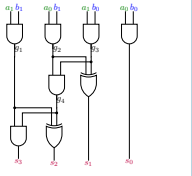
Specification

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

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

Formal Verification

System



Specification

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

$(2a_1 + a_0) * (2b_1 + b_0) =$

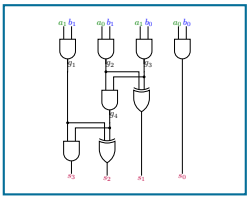
$8s_3 + 4s_2 + 2s_1 + s_0$?

Mathematical Model

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

Formal Verification

System



Specification

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

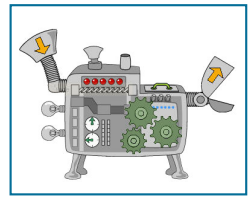
$(2a_1 + a_0) * (2b_1 + b_0) =$

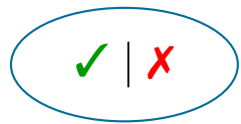
$8s_3 + 4s_2 + 2s_1 + s_0$?

Mathematical Model

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

Automated Decision Process





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
- Past: Requires manual effort
- 2020: Progress in ACL2

  [TemelSlobodovaHunt CAV'20]

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
- Past: Requires manual effort
- 2020: Progress in ACL2

  [TemelSlobodovaHunt CAV'20]

Algebraic Approach

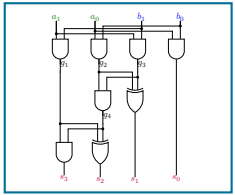
- Seminal work:

  [CiesielskiYuBrownLiuRossi DAC'15]

  [SayedGroßeKühneSoekenDrechsler DATE'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

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

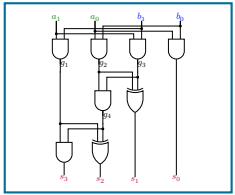
Implication

$= 0$ ✓

$\neq 0$ ✗

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

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

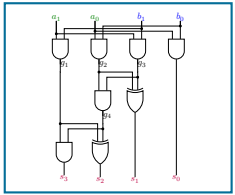
Roots

$= 0$ ✓

$\neq 0$ ✗

Basic Idea of Algebraic Approach

Multiplier



Polynomials

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

Specification

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

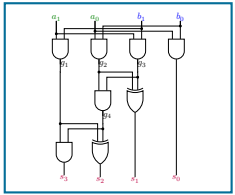
Ideal Membership

$= 0$ ✓

$\neq 0$ ✗

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:

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

Multiplier Specification

Unsigned Integers:

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

Signed Integers:

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

Multiplier Specification

Unsigned Integers:

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

Signed Integers:

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

Multiplier Specification

Unsigned Integers:

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

Signed Integers:

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

Multiplier Specification

Unsigned Integers:

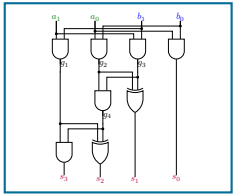
$$\sum_{i=0}^{2n-1} 2^i s_i - \left(\sum_{i=0}^{n-1} 2^i a_i\right)\left(\sum_{i=0}^{n-1} 2^i b_i\right) \quad \in \mathbb{Q}[X]$$

Signed Integers:

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

Basic Idea of Algebraic Approach

Multiplier



Polynomials

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

Specification

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

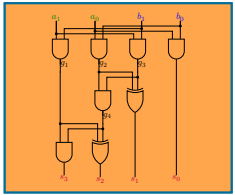
Ideal Membership

$= 0$ ✓

$\neq 0$ ✗

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

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

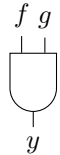
Ideal Membership

$= 0$ ✓

$\neq 0$ ✗

From Circuits to Polynomials

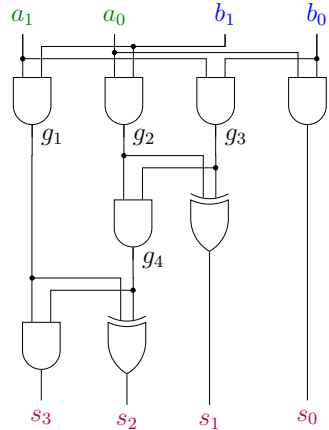
AND-Gate $f \wedge g = y$



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 $f \wedge g = y$

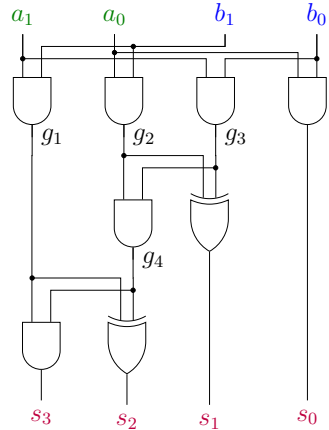
f	g	y
0	0	0
0	1	0
1	0	0
1	1	1

$$-y + fg$$

XOR-Gate $f \oplus g = y$

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

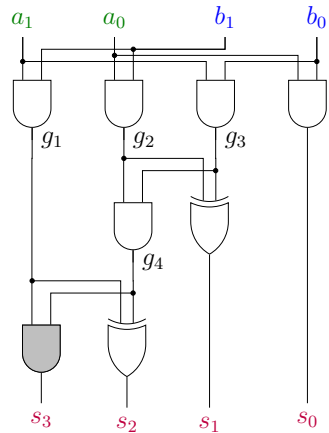
$$-y + f + g - 2fg$$



From Circuits to Polynomials

Gate polynomials $G(C)$.

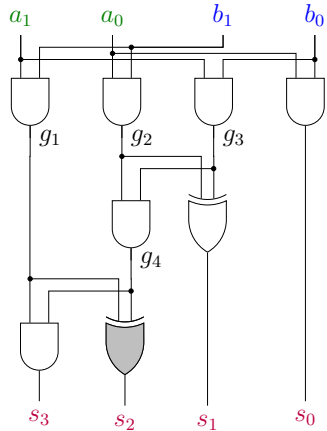
$$s_3 = g_1 \wedge g_4 \qquad -s_3 + g_4 g_1,$$



From Circuits to Polynomials

Gate polynomials $G(C)$.

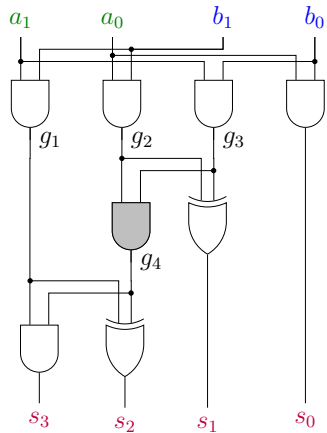
$$
\begin{aligned}
s_3 = g_1 \wedge g_4 \qquad & -s_3 + g_4 g_1, \\
s_2 = g_1 \oplus g_4 \qquad & -s_2 - 2g_4 g_1 + g_4 + g_1,
\end{aligned}
$$



From Circuits to Polynomials

Gate polynomials $G(C)$.

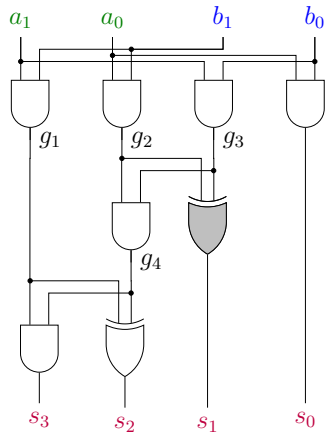
$$
\begin{aligned}
s_3 = g_1 \wedge g_4 \qquad & -s_3 + g_4 g_1, \\
s_2 = g_1 \oplus g_4 \qquad & -s_2 - 2g_4 g_1 + g_4 + g_1, \\
g_4 = g_2 \wedge g_3 \qquad & -g_4 + g_2 g_3,
\end{aligned}
$$



From Circuits to Polynomials

Gate polynomials $G(C)$.

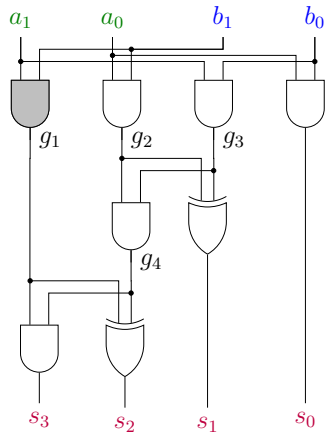
$$
\begin{aligned}
s_3 = g_1 \wedge g_4 \quad & -s_3 + g_4 g_1, \\
s_2 = g_1 \oplus g_4 \quad & -s_2 - 2g_4 g_1 + g_4 + g_1, \\
g_4 = g_2 \wedge g_3 \quad & -g_4 + g_2 g_3, \\
s_1 = g_2 \oplus g_3 \quad & -s_1 - 2g_2 g_3 + g_2 + g_3,
\end{aligned}
$$



From Circuits to Polynomials

Gate polynomials $G(C)$.

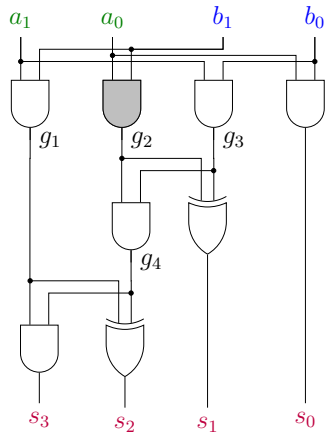
$$
\begin{aligned}
s_3 = g_1 \wedge g_4 \quad & -s_3 + g_4 g_1, \\
s_2 = g_1 \oplus g_4 \quad & -s_2 - 2g_4 g_1 + g_4 + g_1, \\
g_4 = g_2 \wedge g_3 \quad & -g_4 + g_2 g_3, \\
s_1 = g_2 \oplus g_3 \quad & -s_1 - 2g_2 g_3 + g_2 + g_3, \\
g_1 = a_1 \wedge b_1 \quad & -g_1 + a_1 b_1,
\end{aligned}
$$



From Circuits to Polynomials

Gate polynomials $G(C)$.

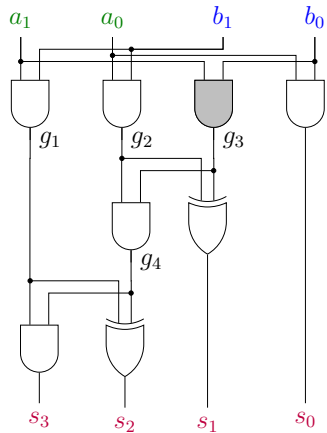
$$
\begin{aligned}
s_3 = g_1 \wedge g_4 \quad & -s_3 + g_4 g_1, \\
s_2 = g_1 \oplus g_4 \quad & -s_2 - 2g_4 g_1 + g_4 + g_1, \\
g_4 = g_2 \wedge g_3 \quad & -g_4 + g_2 g_3, \\
s_1 = g_2 \oplus g_3 \quad & -s_1 - 2g_2 g_3 + g_2 + g_3, \\
g_1 = a_1 \wedge b_1 \quad & -g_1 + a_1 b_1, \\
g_2 = a_0 \wedge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



From Circuits to Polynomials

Gate polynomials $G(C)$.

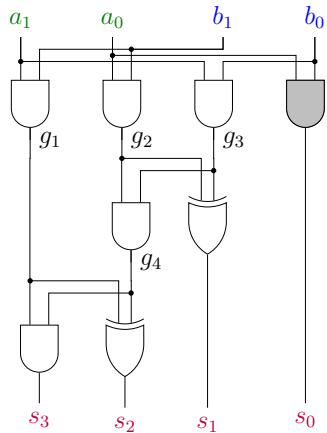
$$
\begin{aligned}
s_3 = g_1 \wedge g_4 \quad & -s_3 + g_4 g_1, \\
s_2 = g_1 \oplus g_4 \quad & -s_2 - 2g_4 g_1 + g_4 + g_1, \\
g_4 = g_2 \wedge g_3 \quad & -g_4 + g_2 g_3, \\
s_1 = g_2 \oplus g_3 \quad & -s_1 - 2g_2 g_3 + g_2 + g_3, \\
g_1 = a_1 \wedge b_1 \quad & -g_1 + a_1 b_1, \\
g_2 = a_0 \wedge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



From Circuits to Polynomials

Gate polynomials $G(C)$.

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



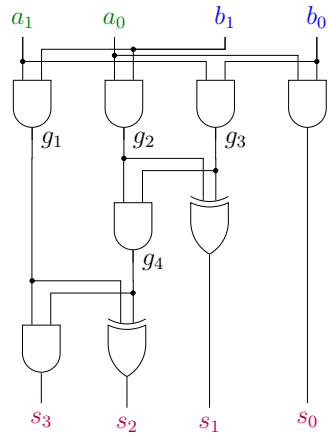
From Circuits to Polynomials

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 $B(C)$.

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

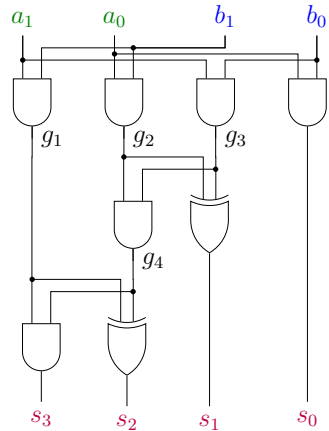


Ideals Associated to Circuits

[RitircBiereKauers FMCAD'17, KaufmannBiereKauers FMSD'19]

More circuit relations:

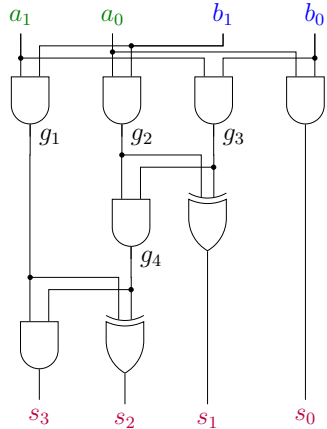
- $s_0 - a_0 b_0$ AND-gate
- $a_1^2 - a_1$ a_1 Boolean
- $g_2^2 - g_2$ g_2 Boolean
- $s_1 g_4$ XOR-AND constraint
- $\ldots$



Ideals Associated to Circuits

[RitircBiereKauers FMCAD'17, KaufmannBiereKauers 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

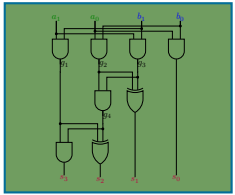
$$\forall\, p, q \in I : p+q \in I \quad \text{and} \quad \forall\, p \in R[X]\, \forall\, q \in I : pq \in I.$$

Multiplier. A circuit C is called a multiplier if

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

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

$= 0$ ✓

$\neq 0$ ✗

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. [Buchberger'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

[RitircBiereKauers FMCAD'17, KaufmannBiereKauers 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: Reverse topological
  Output variable of a gate is greater than input variables [LvKallaEnescu TCAD'13].

Theorem

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

Proof idea: Application of Buchberger's Product criterion.

Soundness and Completeness

[RitircBiereKauers FMCAD'17, KaufmannBiereKauers 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

Verification Algorithm

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

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

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

Verification

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

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

Verification

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

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

Verification

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

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

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

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

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

Verification

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

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

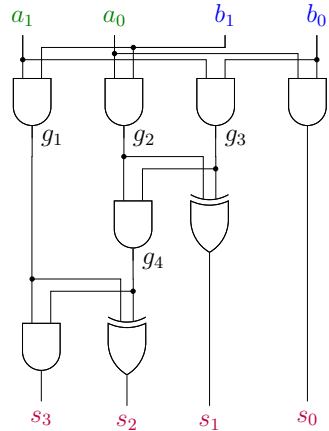
Faulty multiplier

Gate polynomials $G(C)$.

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

Boolean value constraints $B(C)$.

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



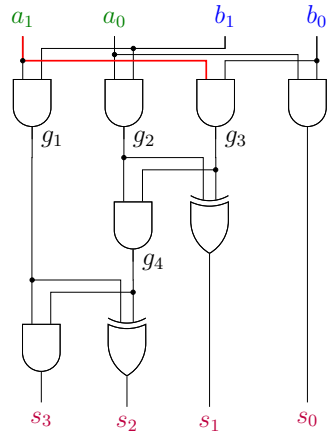
Faulty multiplier

Gate polynomials $G(C)$.

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

Boolean value constraints $B(C)$.

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



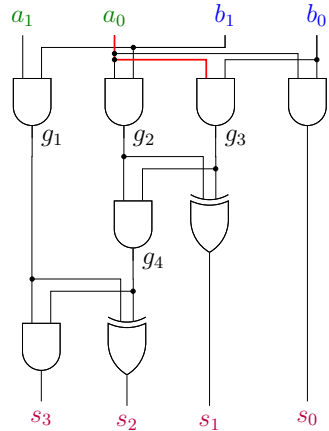
Faulty multiplier

Gate polynomials $G(C)$.

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

Boolean value constraints $B(C)$.

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



Verification of a faulty multiplier

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

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

Verification of a faulty multiplier

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

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

Verification of a faulty multiplier

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

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

Verification of a faulty multiplier

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

$$8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0$$

$$8g_1g_4 + 4s_2 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0$$

$$4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0$$

$$4g_2g_3 + 4g_1 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0$$

$$4g_1 + 2g_3 + 2g_2 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0$$

$$2g_3 + 2g_2 + s_0 - 2a_1b_0 - 2a_0b_1 - a_0b_0$$

$$2g_3 + s_0 - 2a_1b_0 - a_0b_0$$

$$s_0 + a_0b_0$$

$$2a_0b_0$$

Counter Example

Remainder: $2a_0b_0 \neq 0$

Counter Example

Remainder: $2a_0b_0 \neq 0$

$a_1 = 0, \quad a_0 = 1$
$b_1 = 0, \quad b_0 = 1$

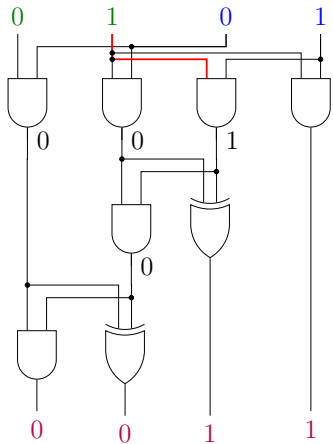
$01 * 01 = 0001$

Counter Example

Remainder: $2a_0b_0 \neq 0$

$a_1 = 0, \quad a_0 = 1$
$b_1 = 0, \quad b_0 = 1$

$01 * 01 = 0001$



Non-Incremental Checking Algorithm

Verification Algorithm

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

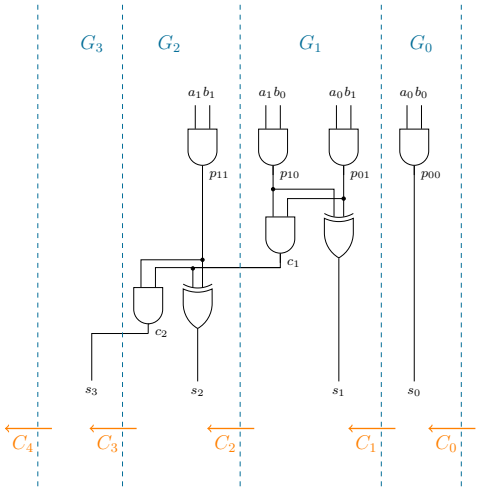
Verification Algorithm

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



[RitircBiereKauers FMCAD'17, KaufmannBiereKauers 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

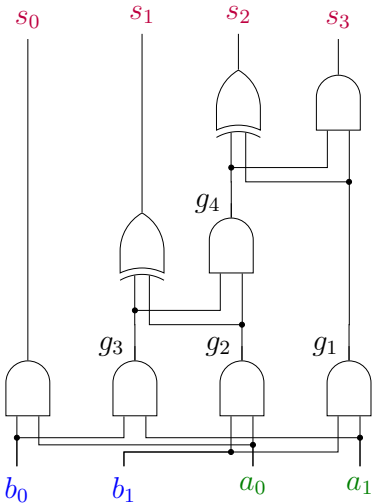
$C_{2n} \leftarrow 0$

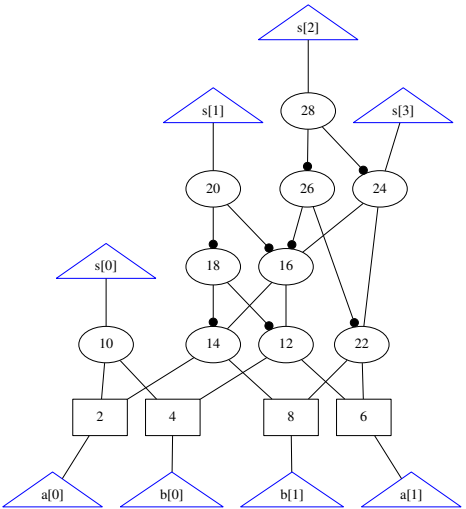
for $i \leftarrow 2n-1$ **to** 0

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

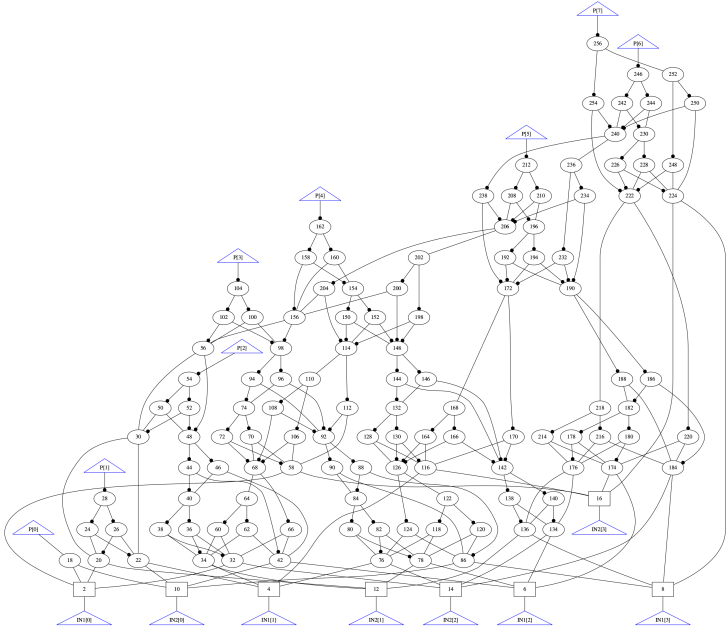
return $C_0 = 0$

AND-Inverter Graphs (AIGs)

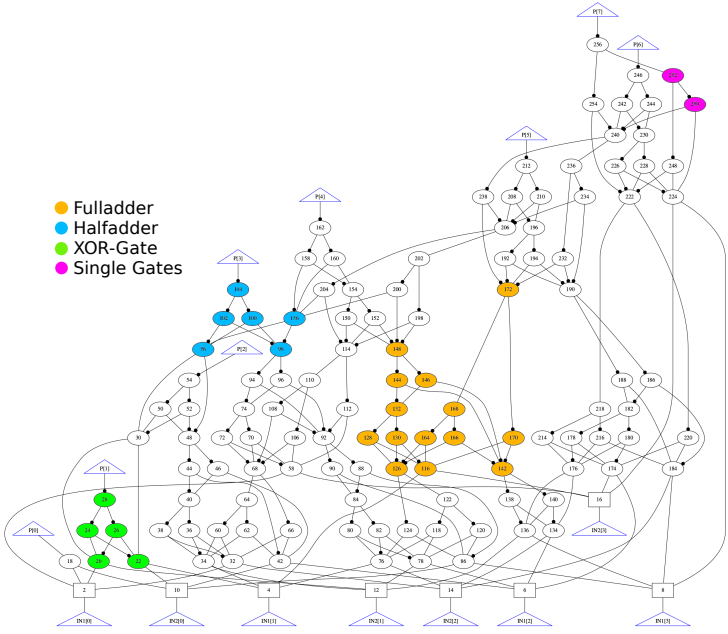




Rewriting


P[7]
256
P[6]
246
252
254
242
244
250
240
230
P[5]
212
236
228
228
248
238
208
210
234
222
224
250
196
P[4]
162
158
160
202
192
194
232
P[3]
164
204
154
200
172
190
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
126
130
164
166
170
214
178
216
180
220
P[1]
44
46
68
58
90
88
120
116
142
176
174
182
28
40
64
84
122
138
140
16
P[0]
34
26
38
36
60
62
66
80
82
124
118
120
136
134
IN2[3]
18
20
22
34
32
42
70
78
86
2
10
4
12
14
6
8
IN1[0]
IN2[0]
IN1[1]
IN2[1]
IN2[2]
IN1[2]
IN1[3]

Rewriting



Variable Elimination

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

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

- Full-adder rewriting
- Half-adder rewriting
- XOR rewriting
- Eliminating nodes with single dependency

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

- Does not rely on specific patterns.

Elimination theory of Gröbner bases

Elimination order. Let $X = Y \dot{\cup} Z$ and we want to eliminate Z. Order the terms such that for all terms σ, τ where a variable from Z is contained in σ but not in τ, we obtain $\tau < \sigma$.

Elimination ideal. The elimination ideal J where the Z-variables are eliminated of $I \subseteq \mathbb{Q}[X] = \mathbb{Q}[Y, Z]$ is defined by

$$J = I \cap \mathbb{Q}[Y].$$

Elimination theorem. Given an ideal $I \subseteq \mathbb{Q}[X] = \mathbb{Q}[Y, Z]$. Further let G be a Gröbner basis of I with respect to an elimination order $Y < Z$. Then the set

$$H = G \cap \mathbb{Q}[Y]$$

is a Gröbner basis of the elimination ideal $J = I \cap \mathbb{Q}[Y]$, in particular $\langle H \rangle = J$.

Variable Elimination

Example: $I \subseteq \mathbb{Q}[o, x, y, a, b]$. Eliminate y.

Ideal

$$I = \langle$$
$$-o + xy - x - y + 1,$$
$$-x + ab - a - b + 1,$$
$$-y + ab \rangle$$

Extended Ideal

$$I = \langle$$
$$-o + xy - x - y + 1,$$
$$-o + xab - x - ab + 1,$$
$$-x + ab - a - b + 1,$$
$$-y + ab \rangle$$

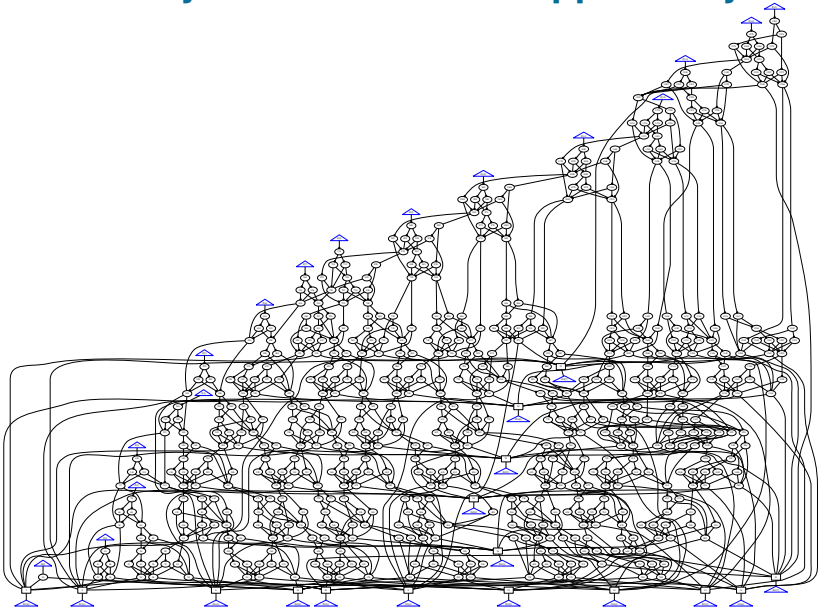
Rewritten ideal

$$I = \langle$$
$$-o + xab - x - ab + 1,$$
$$-x + ab - a - b + 1,$$
$$-y + ab \rangle$$

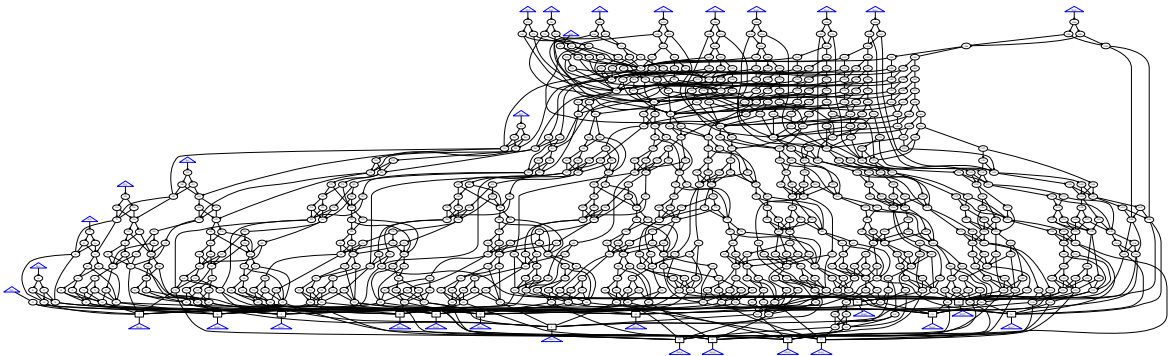
Elimination ideal

$$I \cap \mathbb{Q}[o, x, a, b] = \langle$$
$$-o + xab - x - ab + 1,$$
$$-x + ab - a - b + 1 \rangle$$

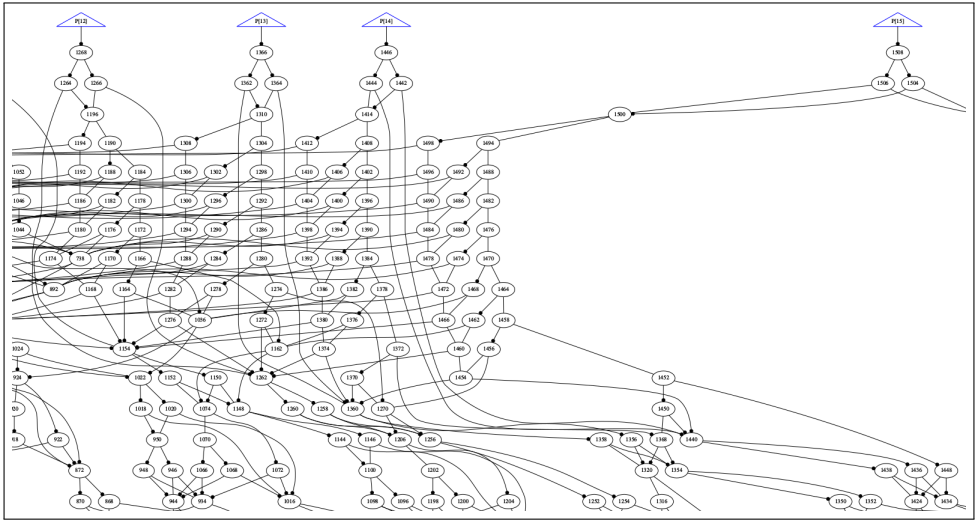
Multiplier – Array Accumulation & Ripple-Carry Adder



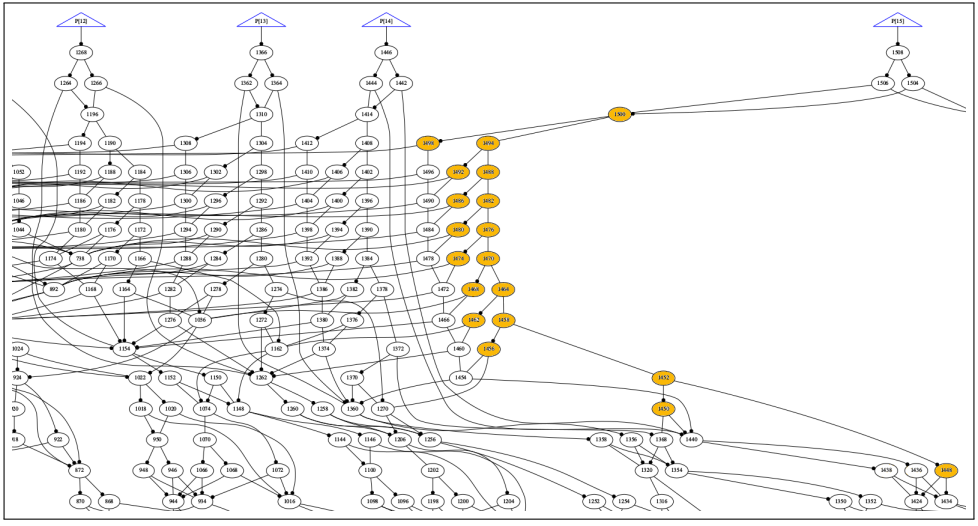
Multiplier – Wallace Tree Acc. & Carry Look-Ahead Adder



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

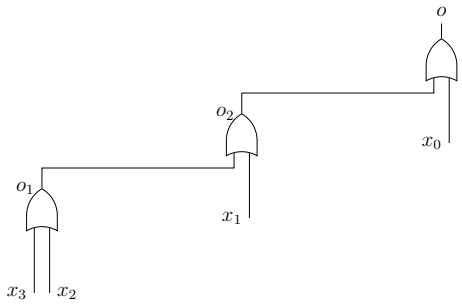


Multiplier – Wallace Tree Acc. & Carry Look-Ahead Adder



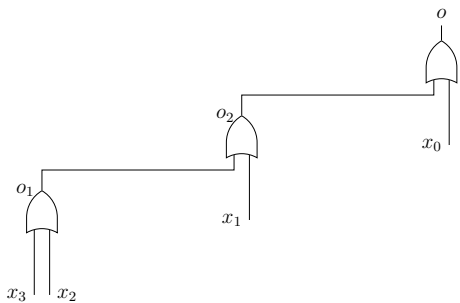
OR Gates

$$
\begin{aligned}
o = o_2 \vee x_0 \quad & -o + o_2 + x_0 - o_2 x_0, \\
o_2 = o_1 \vee x_1 \quad & -o_2 + o_1 + x_1 - o_1 x_1, \\
o_1 = x_3 \vee x_2 \quad & -o_1 + x_3 + x_2 - x_3 x_2
\end{aligned}
$$



OR Gates

$$
\begin{aligned}
o = o_2 \vee x_0 \quad & -o + o_2 + x_0 - o_2 x_0, \\
o_2 = o_1 \vee x_1 \quad & -o_2 + o_1 + x_1 - o_1 x_1, \\
o_1 = x_3 \vee x_2 \quad & -o_1 + x_3 + x_2 - x_3 x_2
\end{aligned}
$$

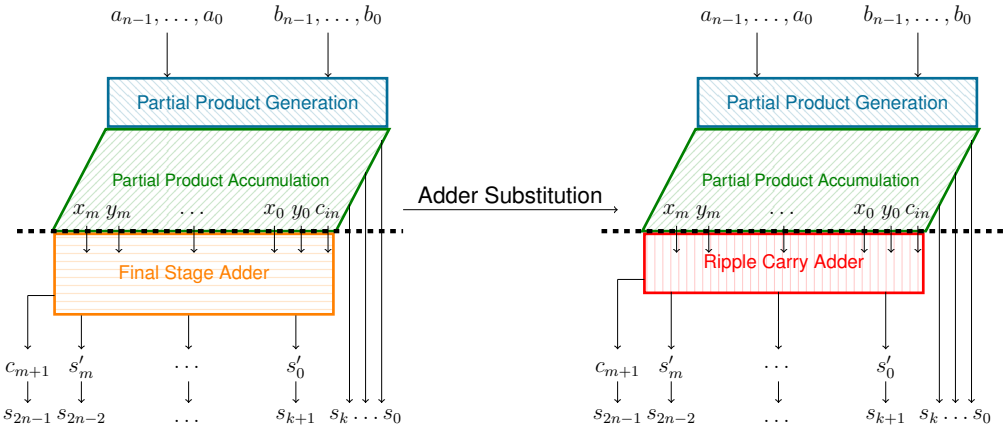


$$o = x_0 + x_1 - x_0x_1 + x_2 - x_0x_2 - x_1x_2 + x_0x_1x_2 + x_3 - x_0x_3 - x_1x_3 + x_0x_1x_3 - x_2x_3 + x_0x_2x_3 + x_1x_2x_3 - x_0x_1x_2x_3$$

$15 = 2^4 - 1$ monomials

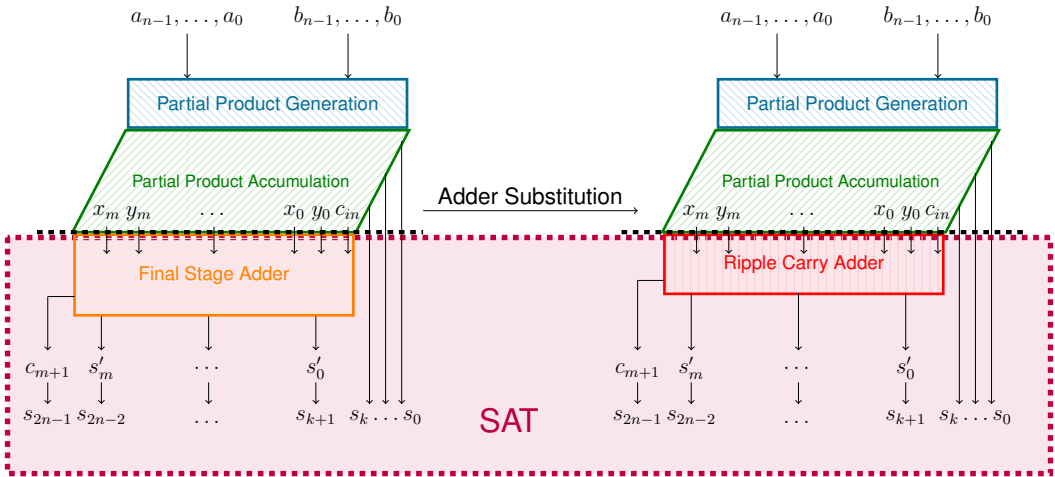
SAT & Computer Algebra

[KaufmannBiereKauers FMCAD'19]



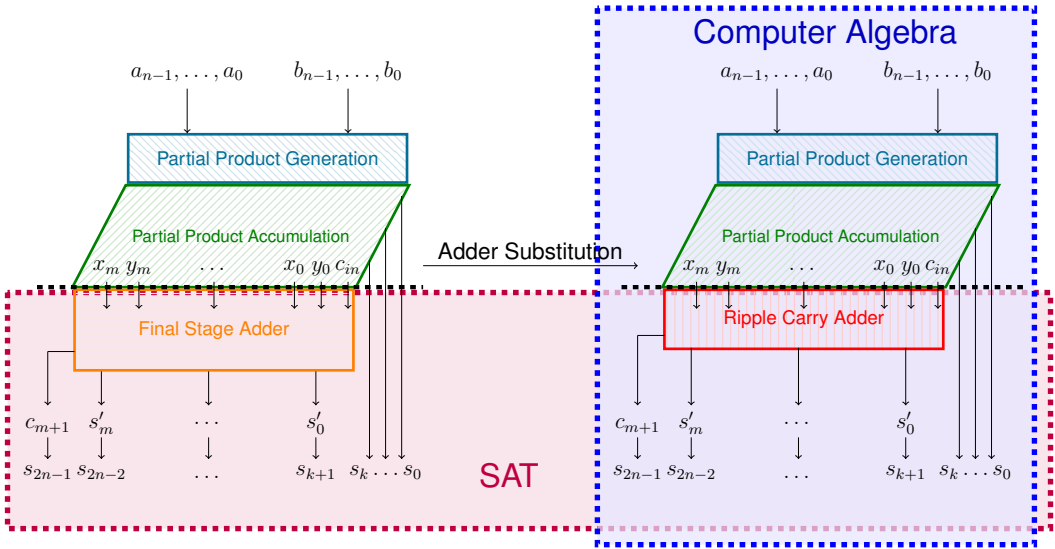
SAT & Computer Algebra

[KaufmannBiereKauers FMCAD'19]

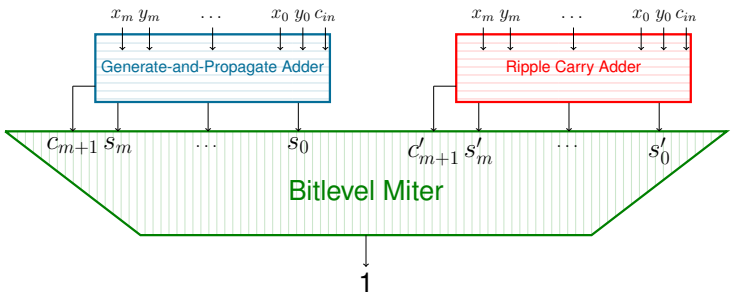


SAT & Computer Algebra

[KaufmannBiereKauers FMCAD'19]



SAT



- AIG is translated to CNF.
- CNF is given to SAT solver.
- To show correctness SAT solver needs to return UNSAT.

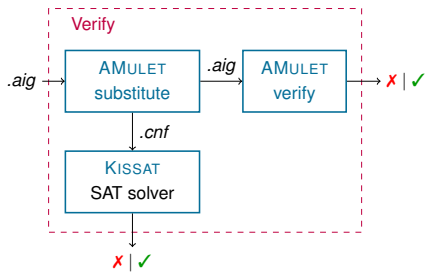
AMulet

Computer Algebra Systems [KaufmannBiereKauers FMSD'19]

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

AMULET [KaufmannBiereKauers 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)
```

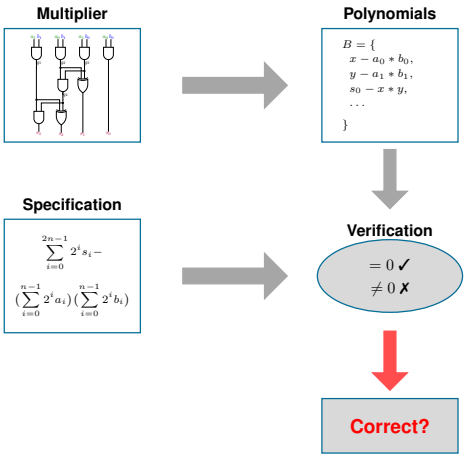
```
daniela@daninovo:~/algprop/src/amulet$ ./amulet ../benchmarks/btor/btor512.aig -verify -v1
[amulet] AMulet Version 001
[amulet] Aiger multiplier examination tool
[amulet] Copyright (C) 2019, Daniela Kaufmann, Johannes Kepler University Linz
[amulet] reading '../benchmarks/btor/btor512.aig'
[amulet] 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+8988465674311579538646525953945123668089884894711532863671504057886633790275048156635423866120376801056005693993569667
8293948844072083112464237153197370621888839467124327426381511098006230470597265414760425028844190753411712314407369565552704136185816752553422931491199736229692398581524176781648121120686 08;
[amulet]
[amulet] current spec is 4494232837155789769323262976972561834044942447355766431835752028943316895137524078317711933060188400528002846996784833941469744220360415562321185765986853109444197335621637131907555490031152352986327073802
12514422095376705856157203684782776352068092908376276711465745599868114846199290762088390824060560343 04*s1022-8988465674311579538646525953945123668089884894711532863671504057886633790275048156635423866120376801056005693993569667 88
29394884407208311246423715319737062188883946712432742638151109800623047059726541476042502884419075341171231440736956555270413618581675255342293149119973622969239858152417678164812112068608*l4185088-44942328371557897693232629769725
6183404494244735576643183575202894331689513752407831771193306018840052800284699678483394146974422036041556232118576598685310944419733562163713190755549003115235298632707380212514422095376705856157203684782776352068092908376276711 4
6574559986811484619929076208839082406056034304*b511*a511+898846567431157953864652595394512366808988489471153286367150405788663379027504815663542386612037680105600569399356966788293948844072083112464237153197370621888839467124327 42
638151109800623047059726541476042502884419075341171231440736956555270413618581675255342293149119973622969239858152417678164812112068608;
[amulet] reduced by slice 1022
[amulet] after reducing by slice 1022
[amulet] remainder is -449423283715578976932326297697256183404494244735576643183575202894331689513752407831771193306018840052800284699678483394146974422036041556232118576598685310944419733562163713190755549003115235298632707380212
51442209537670585615720368478277635206809290837627671146574559986811484619929076208839082406056034304*l4185070-4494232837155789769323262976972561834044942447355766431835752028943316895137524078317711933060188400528002846996784833 9
414697442203604155623211857659868531094441973356216371319075554900311523529863270738021251442209537670585615720368478277635206809290837627671146574559986811484619929076208839082406056034304*l4185072+8988465674311579538646525953945
123668089884894711532863671504057886633790275048156635423866120376801056005693993569667882939488440720831124642371531973706218888394671243274263815110980062304705972654147604250288441907534117123144073695655527041361858167525534 22
9314911997362296923985815241767816481211206 8608;
[amulet]
[amulet] current spec is 2247116418577894884661631488486280917022471223677883215917876014471658447568762039158855966530094200264001423498392416970734872110180207781160592882993426554722098667810818565953777745015576176493163536901
062572110476883529280786018423913881760340464541881383557328727999340574230996453810441954120302801 7152*s1021-4494232837155789769323262976972561834044942447355766431835752028943316895137524078317711933060188400528002846996784833 94
146974422036041556232118576598685310944419733562163713190755549003115235298632707380212514422095376705856157203684782776352068092908376276711465745599868114846199290762088390824060560343 04*l4185070-44942328371557897693232629769725
6183404494244735576643183575202894331689513752407831771193306018840052800284699678483394146974422036041556232118576598685310944419733562163713190755549003115235298632707380212514422095376705856157203684782776352068092908376276711 4
6574559986811484619929076208839082406056034304*l4185072-22471164185778948846616314884862809170224712236778832159178760144716584475687620391588559665300942002640014234983924169707348721101802077811605928829934265547220986678108185 6
595377774501557617649316353690106257211047688352928078601842391388176034046454188138355732872799934057423099645381044195412030280171 52*b511*a510-224711641857789488466163148848628091702247122367788321591787601447165844756876203915 8
8559665300942002640014234983924169707348721101802077811605928829934265547220986678108185659537777450155761764931635369010625721104768835292807860184239138817603404645418813835573287279993405742309964538104419541203028017152*b510*a
511+8988465674311579538646525953945123668089884894711532863671504057886633790275048156635423866120376801056005693993569667882939488440720831124642371531973706218888394671243274263815110980062304705972654147604250288441907534117123
144073695655527041361858167525534229314911997362296923985815241767816481211206 8608;
[amulet] reduced by slice 1021
[amulet] after reducing by slice 1021
[amulet] remainder is -224711641857789488466163148848628091702247122367788321591787601447165844756876203915885596653009420026400142349839241697073487211018020778116059288299342655472209866781081856595377774501557617649316353690106
25721104768835292807860184239138817603404645418813835573287279993405742309964538104419541203028017152*l4185036-22471164185778948846616314884862809170224712236778832159178760144716584475687620391588559665300942002640014234983924169
7073487211018020778116059288299342655472209866781081856595377774501557617649316353690106257211047688352928078601842391388176034046454188138355732872799934057423099645381044195412030280171 52*l4185038-224711641857789488466163148848 6
2809170224712236778832159178760144716584475687620391588559665300942002640014234983924169707348721101802077811605928829934265547220986678108185659537777450155761764931635369010625721104768835292807860184239138817603404645418813835 5
73287279993405742309964538104419541203028017152*l4185040+67413492557336846539848944654588427510674136710336496477536280434149753427062861174765678995902826007920042704951772509122046163305406233434817786489802796641662960034324556
978613332350467285294794906107031877163314306505878423580552717416452810213936256441506719861839980217226929893614313258623609084051456;
[amulet]
[amulet] current spec is 1123558209288947442330815744243140458511235611838941607958938007235829223784381019579427983265047100132000711749196208485367436055090103890580296441496713277361049333905409282976888872507788088246581768450
5312860552384417646403930092119569408801702322709406917786643639996702871154982269052209770601514008576*s1020-2247116418577894884661631488486280917022471223677883215917876014471658447568762039158855966530094200264001423498392416 97
07348721101802077811605928829934265547220986678108185659537777450155761764931635369010625721104768835292807860184239138817603404645418813835573287279993405742309964538104419541203028017152*l4185036-2247116418577894884661631488486 2
8091702247122367788321591787601447165844756876203915885596653009420026400142349839241697073487211018020778116059288299342655472209866781081856595377774501557617649316353690106257211047688352928078601842391388176034046454188138355 57
```

[amdlet] reduced by slice 1005
[amdlet] after reducing by slice 1005
[amdlet] remainder: 0x0000000000000000

```
[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] maximum polynomial size is:                     1540 monomials
[amulet] maximum resident set size:                   1711.02 MB
[amulet] used time for reading/slicing/ordering:        50.39 seconds
[amulet] used time for decomposition:                   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$
```

Is the circuit really really correct?

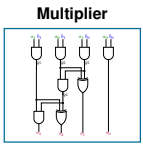
Proofs

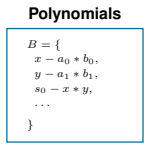


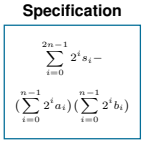
Problem:

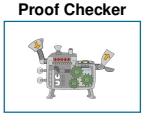
- Can we trust our own implementation?
- Is the verification process correct?

Proofs









Problem:

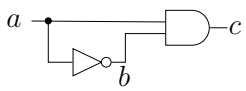
- Can we trust our own implementation?
- Is the verification process correct?

Solution:

Validate result of verification process.

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

Practical Algebraic Calculus



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

[RitircBiereKauers 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;
```

[KaufmannBiereKauers FMCAD'19]

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

[KaufmannFleuryBiere FMCAD'20]

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

Proof Checker

PACTRIM [RitircBiereKauers SC2'18]

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

PACHECK [KaufmannFleuryBiere FMCAD'20]

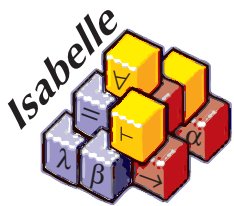
- Written in C (~ 1700 lines of code).
- Supports new calculus.
- Backward compatible to older PAC formats.

Is the circuit really really really correct?

PASTÈQUE

[KaufmannFleuryBiere FMCAD'20]

Theorem Prover Isabelle/HOL



Refinement Approach, relying on Isabelle's Refinement Framework

- abstract specification on ideals: specification in ideal
- final step: executable checker

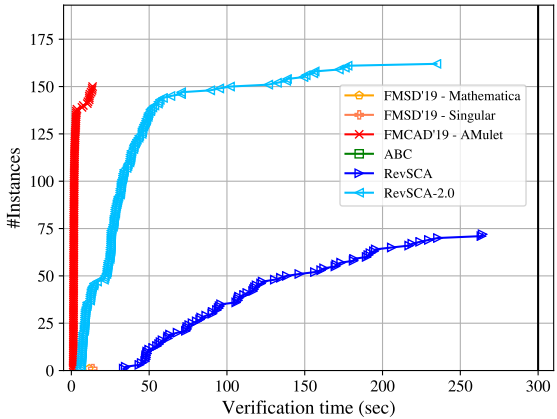
Isabelle's Archive of Formal Proofs 8000 lines of code

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. In DATE, 2020.

Evaluation – Verification

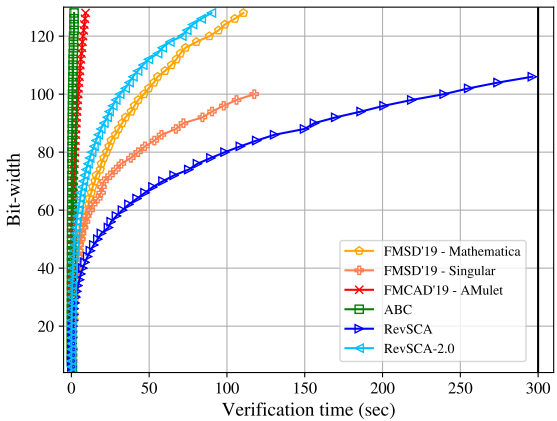


Comparing our approaches to most recent related work.

- Our work
  [KaufmannBiereKauers FMSD'19]
  [KaufmannBiereKauers 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

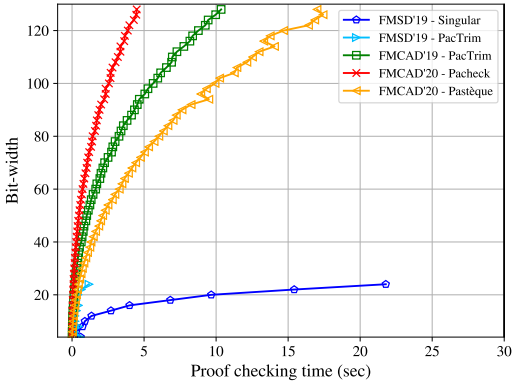


Comparing our approaches to most recent related work.

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

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

Evaluation – Certification & Proof Checking



Comparison of our approaches

[KaufmannBiereKauers FMSD'19]

[KaufmannBiereKauers FMCAD'19]

[KaufmannFleuryBiere FMCAD'20]

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 size
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

Conclusion

- **Is the circuit correct?**

  Yes, because we have tested some cases.

- **Is the circuit really correct?**

  Yes, because we applied formal verification.

- **Is the circuit really really correct?**

  Yes, because we generated and checked a proof certificate.

- **Is the circuit really really really correct?**

  Yes, because we used a verified proof checker.

FORMAL VERIFICATION OF INTEGER MULTIPLIER CIRCUITS USING SAT AND COMPUTER ALGEBRA

Keynote

Daniela Kaufmann

Johannes Kepler University

Linz, Austria

14th International Workshop on Boolean Problems

Bremen, Germany (online)

September 24, 2020

JOHANNES KEPLER UNIVERSITY LINZ

References I

[RitircBiereKauers 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.

[RitircBiereKauers 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.

[RitircBiereKauers 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.

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

[KaufmannBiereKauers 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.

[KaufmannBiereKauers 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.

References II

[KaufmannFleuryBiere FMCAD'20] D. Kaufmann, M. Fleury and A. Biere. Pacheck and Pastèque Checking Practical Algebraic Calculus Proofs. To appear at FMCAD 2020.

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

[TemelSlobodovaHunt CAV'20] M. Temel, A. Slobodova and W. Hunt. Automated and Scalable Verification of Integer Multipliers. In Proc. of CAV'20, pages 485–507, Springer, 2020.

[LvKallaEnescu TCAD'13] J. Lv, P. Kalla, F. Enescu. Efficient Gröbner Basis Reductions for Formal Verification of Galois Field Arithmetic Circuits. In IEEE TCAD, vol. 32, pages 1409–1420, 2013.

[CiesielskiYuBrownLiuRossi DAC'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

[SayedGroßeKühneSoekenDrechsler DATE'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`