FROM CIRCUITS TO IDEALS: GRÖBNER BASES FOR FORMAL VERIFICATION

Daniela Kaufmann
TU Wien, Austria

Plenary Talk
ALge**B**r**A**ic me**T**hods fo**R** p**O**lynomial **S**ystem **S**olving Workshop/Summer School

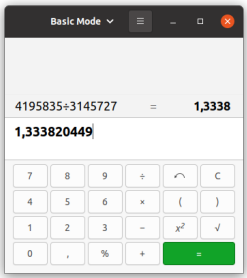
September 4, 2026

TU WIEN Informatics

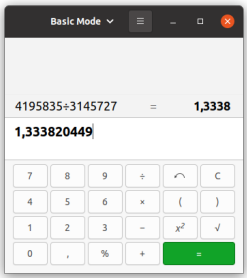
FWF Austrian Science Fund

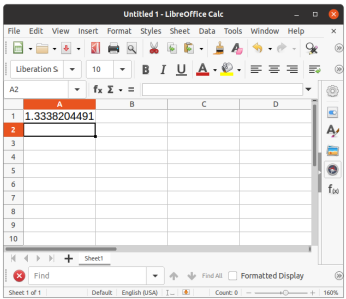
$$4\,195\,835 \div 3\,145\,727 = ?$$

$$4\,195\,835 \div 3\,145\,727 = ?$$

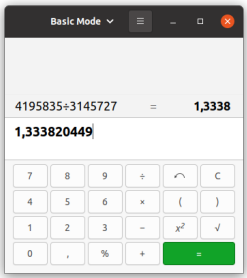


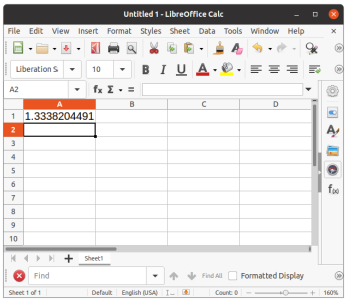
$$4\,195\,835 \div 3\,145\,727 = ?$$

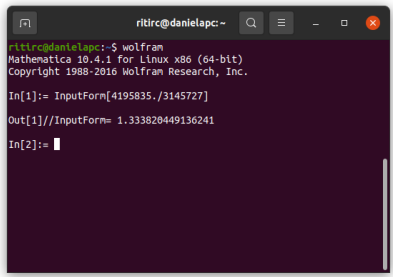


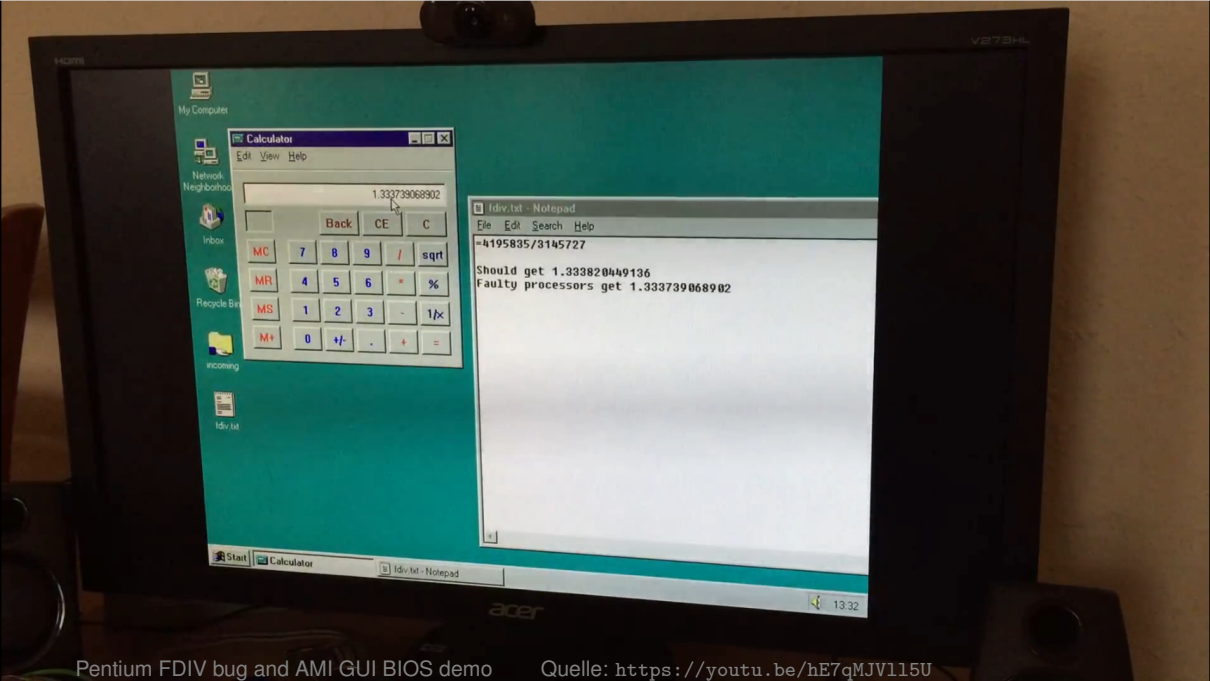


$$4\,195\,835 \div 3\,145\,727 = ?$$









Pentium FDIV bug and AMI GUI BIOS demo Quelle: https://youtu.be/hE7qMJV115U

Intel Pentium FDIV-Bug 1994

Quelle: http://neology.com.au/portfolios/a80502-90-sx923/

- 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 30 years later verification of arithmetic circuits is considered to be hard. Correctness proofs are not fully automated yet.

Challenge: Integer multipliers

Integer Multiplication

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

$3 \cdot 3 = 9$

Integer Multiplication

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

$3 \cdot 3 = 9$

Integer Multiplication

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

$3 \cdot 3 = 9$

Integer Multiplication

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

$3 \cdot 3 = 9$

Integer Multiplication

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

$3 \cdot 3 = 9$

Integer Multiplication

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

$3 \cdot 3 = 9$

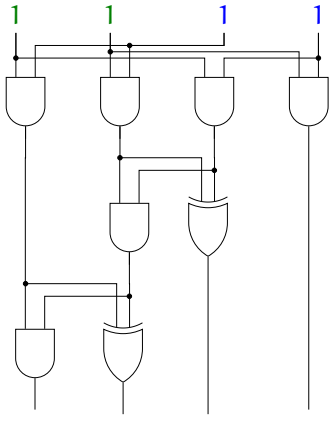
Integer Multiplication

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

$3 \cdot 3 = 9$

Integer Multiplication

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



$3 \cdot 3 = 9$

Integer Multiplication

AND-Gate

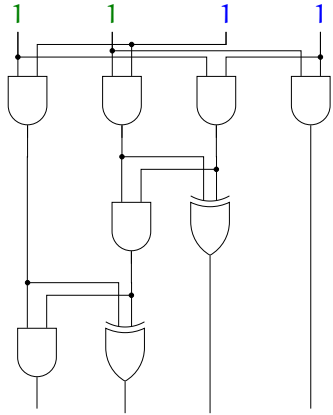
$f \wedge g = y$

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



Integer Multiplication

AND-Gate

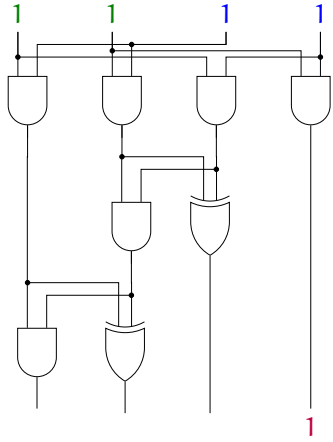
$f \wedge g = y$

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



Integer Multiplication

AND-Gate

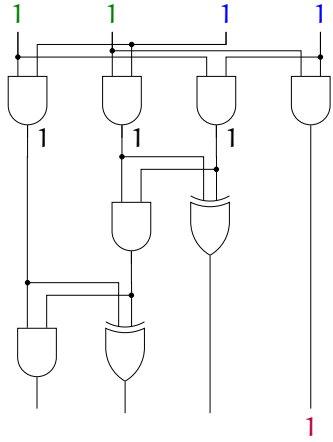
$f \wedge g = y$

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



Integer Multiplication

AND-Gate

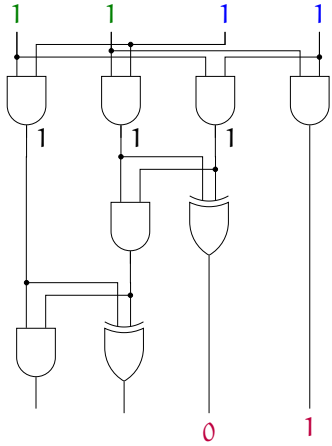
$f \wedge g = y$

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



Integer Multiplication

AND-Gate

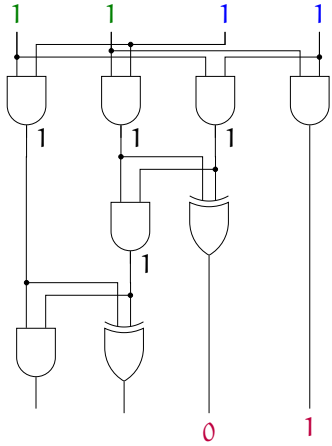
$f \wedge g = y$

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



Integer Multiplication

AND-Gate

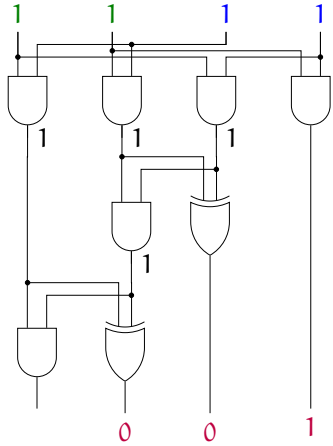
$f \wedge g = y$

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



Integer Multiplication

AND-Gate

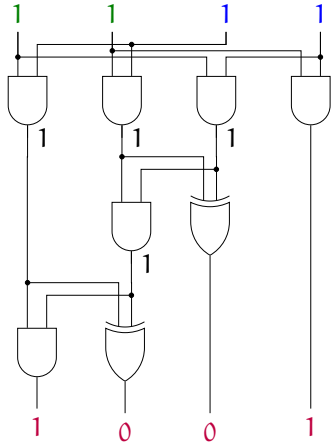
$f \wedge g = y$

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

XOR-Gate

$f \oplus g = y$

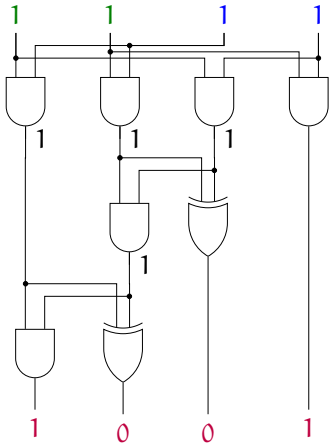
f	g	y
0	0	0
0	1	1
1	0	1
1	1	0



Is the circuit correct?

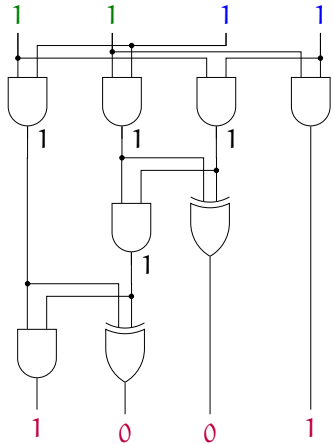
Multiplier circuit

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



Multiplier circuit

Circuit seems correct!



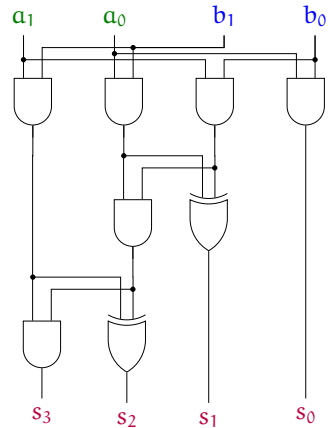
Is the circuit really correct?

Multiplier Circuits

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

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

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

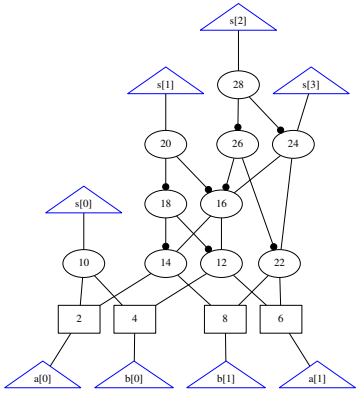


Multiplier Circuits

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

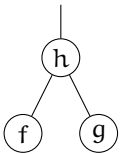
Question: For all possible $a_i, b_i \in \mathbb{B}$:

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

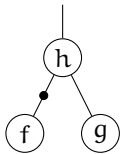


And-Inverter Graph

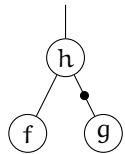
And-Inverter Graph



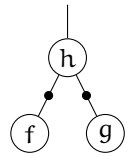
$h = f \wedge g$



$h = \neg f \wedge g$

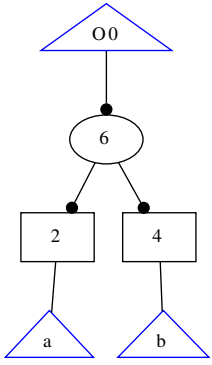


$h = f \wedge \neg g$

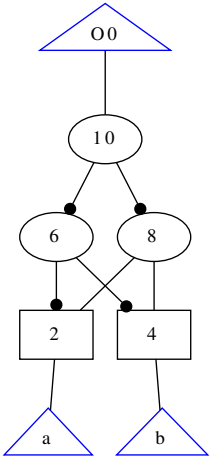


$h = \neg f \wedge \neg g$

And-Inverter Graph



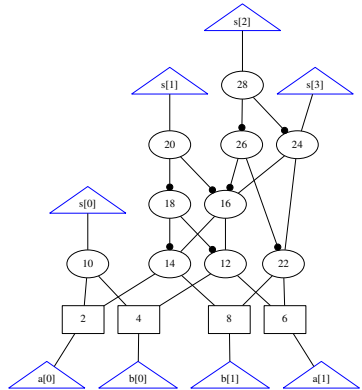
$a \vee b \equiv \neg(\neg a \wedge \neg b)$



$a \otimes b \equiv \neg(\neg a \wedge \neg b) \wedge \neg(a \wedge b)$

And-Inverter Graph

```
aig 14 4 0 4 10
2
4
6
8
10
20
28
24
10 6 2
12 6 4
14 8 2
16 14 12
18 15 13
20 19 17
22 8 4
24 22 16
26 23 17
28 27 25
```

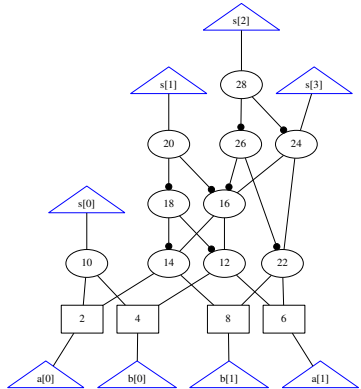


And-Inverter Graph

Circuit Verification

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

$3 \cdot 3 = 9$

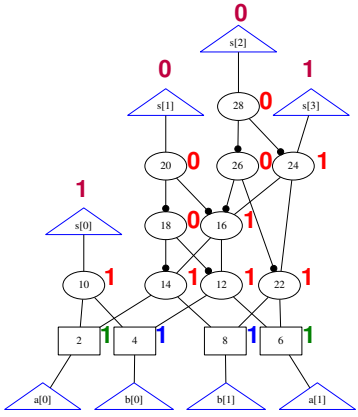


And-Inverter Graph

Circuit Verification

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

$3 \cdot 3 = 9$



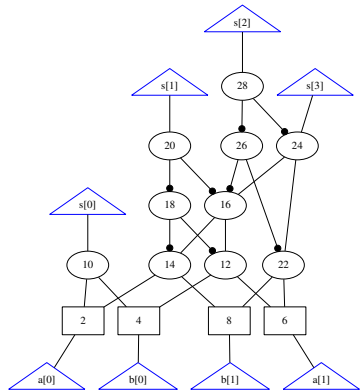
And-Inverter Graph

Circuit Verification

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

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

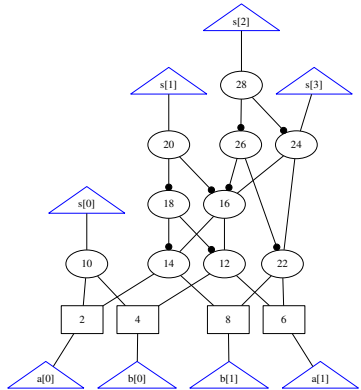
$$(2a_1 + a_0) * (2b_1 + b_0) = 8s_3 + 4s_2 + 2s_1 + s_0?$$



And-Inverter Graph

Simulation

$a_1 a_0$	·	$b_1 b_0$
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

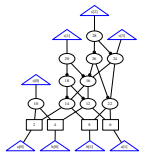


Simulation

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

Formal Verification

System



Specification

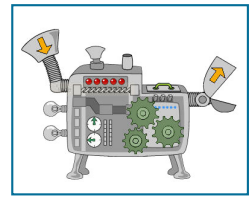
For all $a_i, b_i \in \mathbb{B}$:

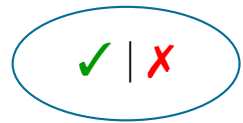
$$(2a_1 + a_0) * (2b_1 + b_0) = 8s_3 + 4s_2 + 2s_1 + s_0?$$

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

Decision Diagrams

- First technique to detect Pentium bug

  [ChenBryant DAC'95]

- Requires knowledge of the layout

Formal Verification Techniques

Decision Diagrams

- First technique to detect Pentium bug
  [ChenBryant DAC'95]
- Requires knowledge of the layout

Theorem Proving

- Used in industry, e.g., ACL2
  [TemelSlobodovaHunt CAV'20]
- Automated on the RTL level
  [Temel TACAS'24]

Formal Verification Techniques

Decision Diagrams

- First technique to detect Pentium bug [ChenBryant DAC'95]
- Requires knowledge of the layout

Theorem Proving

- Used in industry, e.g., ACL2 [TemelSlobodovaHunt CAV'20]
- Automated on the RTL level [Temel TACAS'24]

Satisfiability Checking (SAT)

- SAT 2016: Exponential run-time of solvers [Biere SATComp'16]
- SAT 2024: Equivalence checking of structural similar circuits [BiereFallerFazekasFleuryFroleyksPollitt SATComp'24]

Formal Verification Techniques

Decision Diagrams

- First technique to detect Pentium bug [ChenBryant DAC'95]
- Requires knowledge of the layout

Satisfiability Checking (SAT)

- SAT 2016: Exponential run-time of solvers [Biere SATComp'16]
- SAT 2024: Equivalence checking of structural similar circuits [BiereFallerFazekasFleuryFroleyksPollitt SATComp'24]

Theorem Proving

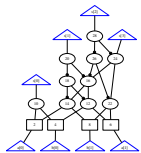
- Used in industry, e.g., ACL2 [TemelSlobodovaHunt CAV'20]
- Automated on the RTL level [Temel TACAS'24]

X^1 Algebraic Approach

- Seminal work: [LvKallaEnescu TCAD'13, CiesielskiYuBrownLiuRossi DAC'15] [SayedGroßeKühneSoekenDrechsler DATE'16]
- Polynomial encoding
- Works for non-trivial multiplier designs

Basic Idea of Algebraic Approach

Multiplier



Polynomials

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

Specification

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

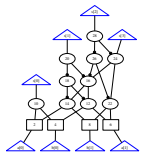
Implication

$= 0$ ✓

$\neq 0$ ✗

Basic Idea of Algebraic Approach

Multiplier



Polynomials

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

Specification

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

Ideal Membership

$= 0$ ✓

$\neq 0$ ✗

Multiplier Specification

Unsigned Integers:

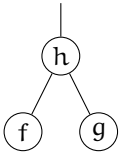
$$\sum_{i=0}^{2n-1} 2^i s_i - \left(\sum_{i=0}^{n-1} 2^i a_i\right)\left(\sum_{i=0}^{n-1} 2^i b_i\right) \quad \in \mathbb{Q}[X]$$

Multiplier Specification

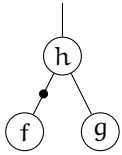
Unsigned Integers:

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

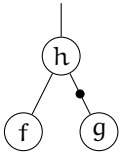
From AIGs to Polynomials



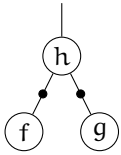
$h = f \wedge g$



$h = \neg f \wedge g$

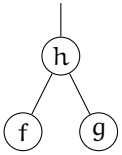


$h = f \wedge \neg g$



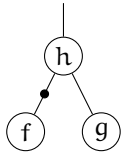
$h = \neg f \wedge \neg g$

From AIGs to Polynomials



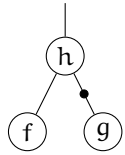
$h = f \wedge g$

f	g	h
0	0	0
0	1	0
1	0	0
1	1	1



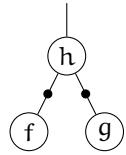
$h = \neg f \wedge g$

f	g	h
0	0	0
0	1	1
1	0	0
1	1	0



$h = f \wedge \neg g$

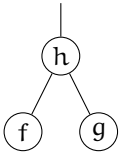
f	g	h
0	0	0
0	1	0
1	0	1
1	1	0



$h = \neg f \wedge \neg g$

f	g	h
0	0	1
0	1	0
1	0	0
1	1	0

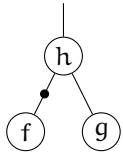
From AIGs to Polynomials



$h = f \wedge g$

f	g	h
0	0	0
0	1	0
1	0	0
1	1	1

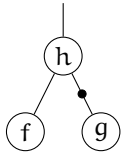
$-h + fg$



$h = \neg f \wedge g$

f	g	h
0	0	0
0	1	1
1	0	0
1	1	0

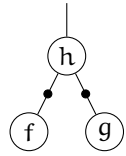
$-h - fg + g$



$h = f \wedge \neg g$

f	g	h
0	0	0
0	1	0
1	0	1
1	1	0

$-h - fg + f$



$h = \neg f \wedge \neg g$

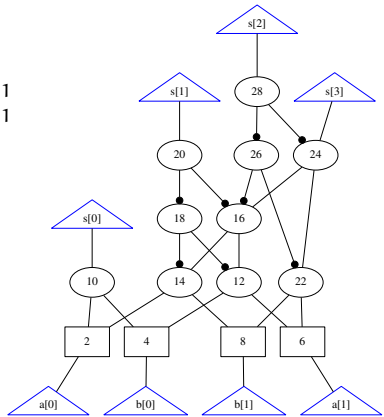
f	g	h
0	0	1
0	1	0
1	0	0
1	1	0

$-h + fg - f - g + 1$

From AIGs to Polynomials

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

$$
\begin{aligned}
&-s_3 + l_{24} \\
&-s_2 + l_{28} \\
&-s_1 + l_{20} \\
&-s_0 + l_{10} \\
&-l_{28} + l_{26}l_{24} - l_{26} - l_{24} + 1 \\
&-l_{26} + l_{22}l_{16} - l_{22} - l_{16} + 1 \\
&-l_{24} + l_{22}l_{16}
\end{aligned}
\qquad
\begin{aligned}
&-l_{22} + a_1 b_1 \\
&-l_{20} + l_{18}l_{16} - l_{18} - l_{16} + 1 \\
&-l_{18} + l_{14}l_{12} - l_{14} - l_{12} + 1 \\
&-l_{16} + l_{14}l_{12} \\
&-l_{14} + a_0 b_1 \\
&-l_{12} + a_1 b_0 \\
&-l_{10} + a_0 b_0
\end{aligned}
$$



From AIGs to Polynomials

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

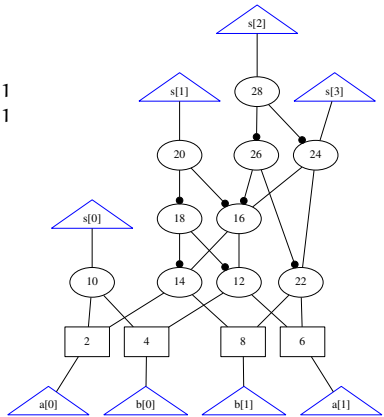
$$
\begin{aligned}
&-s_3 + l_{24} && -l_{22} + a_1 b_1\\
&-s_2 + l_{28} && -l_{20} + l_{18} l_{16} - l_{18} - l_{16} + 1\\
&-s_1 + l_{20} && -l_{18} + l_{14} l_{12} - l_{14} - l_{12} + 1\\
&-s_0 + l_{10} && -l_{16} + l_{14} l_{12}\\
&-l_{28} + l_{26} l_{24} - l_{26} - l_{24} + 1 && -l_{14} + a_0 b_1\\
&-l_{26} + l_{22} l_{16} - l_{22} - l_{16} + 1 && -l_{12} + a_1 b_0\\
&-l_{24} + l_{22} l_{16} && -l_{10} + a_0 b_0
\end{aligned}
$$

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

$$
\begin{aligned}
&a_1, a_0 \in \mathbb{B} && -a_1^2 + a_1,\ -a_0^2 + a_0,\\
&b_1, b_0 \in \mathbb{B} && -b_1^2 + b_1,\ -b_0^2 + b_0
\end{aligned}
$$

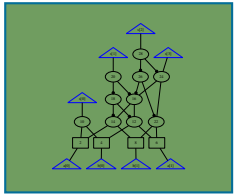
Specification $\mathcal{S}_n \in \mathbb{Z}[X]$.

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



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

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

Ideal Membership

$= 0$ ✓

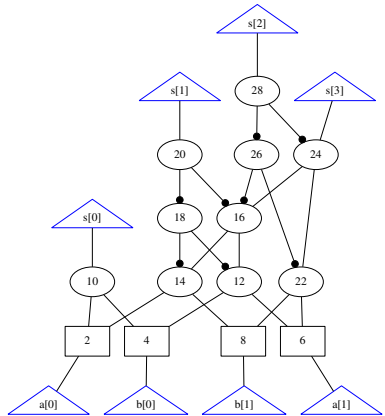
$\neq 0$ ✗

Ideals Associated to Circuits

[RitircBiereKauers FMCAD'17, KaufmannBiereKauers FMSD'20]

More circuit relations:

- $s_0 - a_0 b_0$ AND-gate
- $a_1^2 - a_1$ a_1 Boolean
- $l_{22}^2 - l_{22}$ l_{22} Boolean
- $l_{20} l_{16}$ XOR-AND constraint
- $\ldots$



Polynomial Circuit Constraints.

A polynomial $p \in \mathbb{Q}[X]$ is a polynomial circuit constraint (PCC) if for all

$$(a_0, \ldots, a_{n-1}, b_0, \ldots, b_{n-1}) \in \{0,1\}^{2n}$$

and resulting values $l_1, \ldots, l_k, s_0, \ldots, s_{2n-1}$ implied by the gates of the circuit C the substitution of these values into p gives zero.

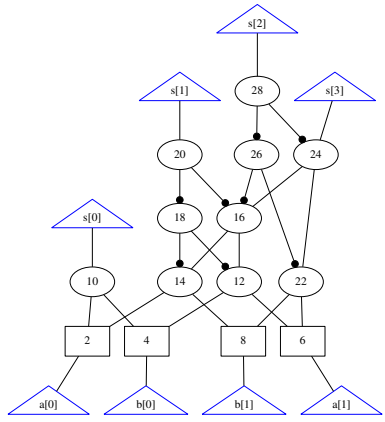
Ideals Associated to Circuits

[RitircBiereKauers FMCAD'17, KaufmannBiereKauers FMSD'20]

- The set of all PCCs for C is denoted by $I(C)$.
- $I(C)$ contains all relations of the circuit C.
- $I(C)$ is an ideal.

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

[RitircBiereKauers FMCAD'17]

- We can deduce some elements of $\mathrm{I}(\mathrm{C})$:
  - Gate polynomials $\mathrm{G}(\mathrm{C})$
  - Boolean input constraints $\mathrm{B}(\mathrm{C})$

Ideal Membership Problem

[RitircBiereKauers FMCAD'17]

- We can deduce some elements of $\mathrm{I}(\mathrm{C})$:
  - Gate polynomials $\mathrm{G}(\mathrm{C})$
  - Boolean input constraints $\mathrm{B}(\mathrm{C})$
- Let $\mathrm{J}(\mathrm{C}) = \langle \mathrm{G}(\mathrm{C}) \cup \mathrm{B}(\mathrm{C}) \rangle$.
- Lexicographic term order: Reverse topological
  Output variable of a gate is greater than input variables [LvKallaEnescu TCAD'13].

Ideal Membership Problem

[RitircBiereKauers FMCAD'17]

- We can deduce some elements of $\mathrm{I}(\mathrm{C})$:
  - Gate polynomials $\mathrm{G}(\mathrm{C})$
  - Boolean input constraints $\mathrm{B}(\mathrm{C})$
- Let $\mathrm{J}(\mathrm{C}) = \langle \mathrm{G}(\mathrm{C}) \cup \mathrm{B}(\mathrm{C}) \rangle$.
- Lexicographic term order: Reverse topological
  Output variable of a gate is greater than input variables [LvKallaEnescu TCAD'13].

Theorem

$\mathrm{G}(\mathrm{C}) \cup \mathrm{B}(\mathrm{C})$ *is a Gröbner basis for* $\mathrm{J}(\mathrm{C})$.

Proof idea: Application of Buchberger's Product criterion.

Soundness and Completeness

[RitircBiereKauers FMCAD'17]

Theorem

For all acyclic circuits C*, we have* $\mathrm{J}(\mathrm{C}) = \mathrm{I}(\mathrm{C})$*.*

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

Verification Algorithm

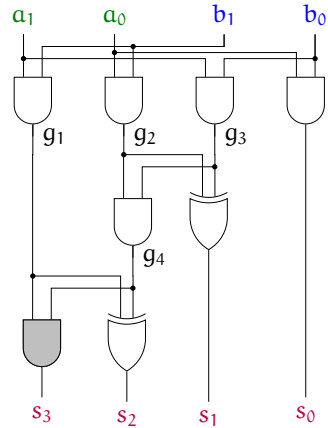
Verification Algorithm

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

Gate polynomials $G(C)$.

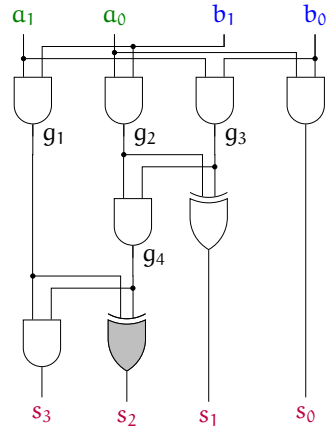
$$s_3 = g_1 \wedge g_4 \qquad -s_3 + g_4 g_1,$$



Verification

Gate polynomials $G(C)$.

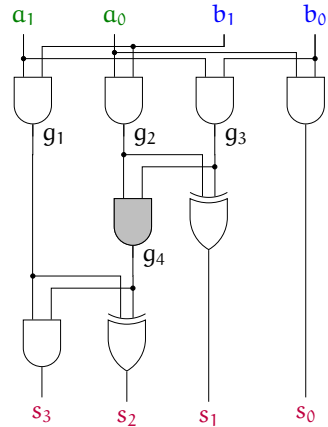
$$s_3 = g_1 \wedge g_4 \qquad -s_3 + g_4 g_1,$$
$$s_2 = g_1 \oplus g_4 \qquad -s_2 - 2g_4 g_1 + g_4 + g_1,$$



Verification

Gate polynomials $G(C)$.

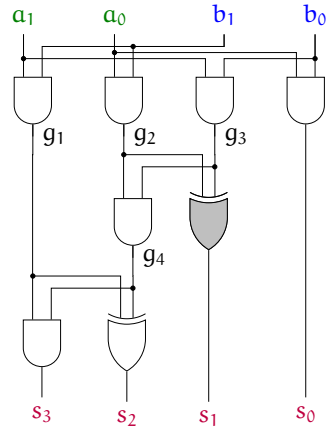
$$
\begin{aligned}
s_3 = g_1 \wedge g_4 \qquad & -s_3 + g_4 g_1, \\
s_2 = g_1 \oplus g_4 \qquad & -s_2 - 2g_4 g_1 + g_4 + g_1, \\
g_4 = g_2 \wedge g_3 \qquad & -g_4 + g_2 g_3,
\end{aligned}
$$



Verification

Gate polynomials $G(C)$.

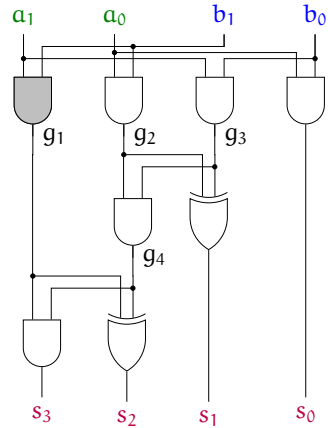
$$
\begin{aligned}
s_3 = g_1 \wedge g_4 \qquad & -s_3 + g_4 g_1, \\
s_2 = g_1 \oplus g_4 \qquad & -s_2 - 2g_4 g_1 + g_4 + g_1, \\
g_4 = g_2 \wedge g_3 \qquad & -g_4 + g_2 g_3, \\
s_1 = g_2 \oplus g_3 \qquad & -s_1 - 2g_2 g_3 + g_2 + g_3,
\end{aligned}
$$



Verification

Gate polynomials $G(C)$.

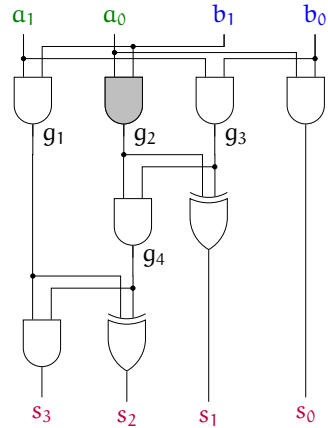
$$
\begin{aligned}
s_3 = g_1 \wedge g_4 \quad & -s_3 + g_4 g_1, \\
s_2 = g_1 \oplus g_4 \quad & -s_2 - 2g_4 g_1 + g_4 + g_1, \\
g_4 = g_2 \wedge g_3 \quad & -g_4 + g_2 g_3, \\
s_1 = g_2 \oplus g_3 \quad & -s_1 - 2g_2 g_3 + g_2 + g_3, \\
g_1 = a_1 \wedge b_1 \quad & -g_1 + a_1 b_1,
\end{aligned}
$$



Verification

Gate polynomials $G(C)$.

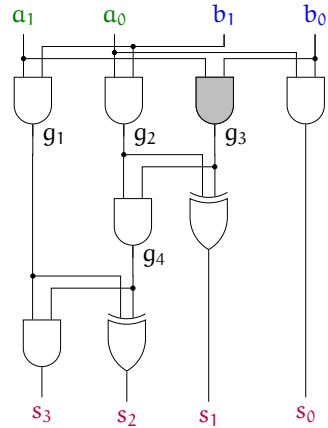
$$
\begin{aligned}
s_3 = g_1 \wedge g_4 \quad & -s_3 + g_4 g_1, \\
s_2 = g_1 \oplus g_4 \quad & -s_2 - 2g_4 g_1 + g_4 + g_1, \\
g_4 = g_2 \wedge g_3 \quad & -g_4 + g_2 g_3, \\
s_1 = g_2 \oplus g_3 \quad & -s_1 - 2g_2 g_3 + g_2 + g_3, \\
g_1 = a_1 \wedge b_1 \quad & -g_1 + a_1 b_1, \\
g_2 = a_0 \wedge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



Verification

Gate polynomials $G(C)$.

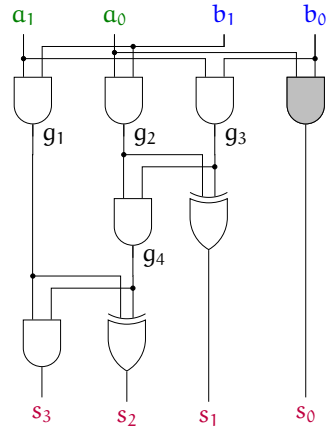
$$
\begin{aligned}
s_3 = g_1 \wedge g_4 \quad & -s_3 + g_4 g_1, \\
s_2 = g_1 \oplus g_4 \quad & -s_2 - 2g_4 g_1 + g_4 + g_1, \\
g_4 = g_2 \wedge g_3 \quad & -g_4 + g_2 g_3, \\
s_1 = g_2 \oplus g_3 \quad & -s_1 - 2g_2 g_3 + g_2 + g_3, \\
g_1 = a_1 \wedge b_1 \quad & -g_1 + a_1 b_1, \\
g_2 = a_0 \wedge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



Verification

Gate polynomials $G(C)$.

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



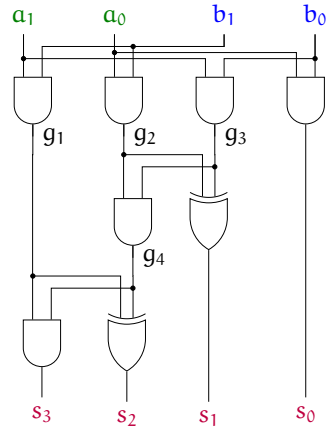
Verification

Gate polynomials $\mathrm{G}(C)$.

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

Boolean input constraints $\mathrm{B}(C)$.

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



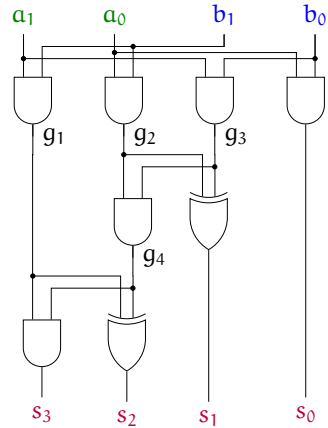
Verification

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 input constraints $B(C)$.

$$
\begin{aligned}
a_1, a_0 \in \mathbb{B} \quad & -a_1^2 + a_1,\ -a_0^2 + a_0, \\
b_1, b_0 \in \mathbb{B} \quad & -b_1^2 + b_1,\ -b_0^2 + b_0
\end{aligned}
$$



Verification

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

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

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

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

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

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

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

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

Verification

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

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

Verification

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

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

Verification

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

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

Verification

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

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

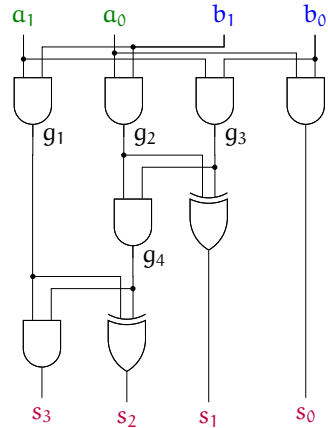
Faulty multiplier

Gate polynomials $\mathrm{G}(C)$.

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

Boolean input constraints $\mathrm{B}(C)$.

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



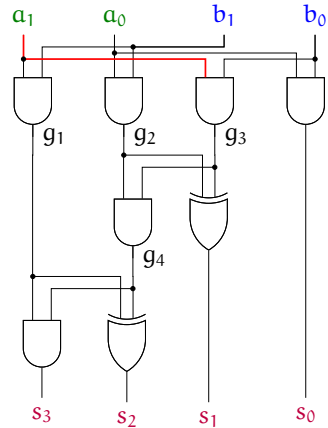
Faulty multiplier

Gate polynomials $\mathrm{G}(C)$.

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

Boolean input constraints $\mathrm{B}(C)$.

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



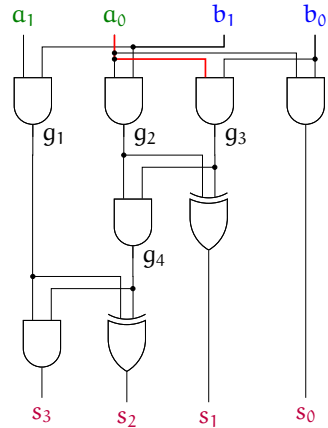
Faulty multiplier

Gate polynomials $\mathrm{G}(C)$.

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

Boolean input constraints $\mathrm{B}(C)$.

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



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

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

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

Verification of a faulty multiplier

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

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

Verification of a faulty multiplier

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

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

Counter Example

Remainder: $-2a_1b_0 + 2a_0b_0 \neq 0$

Counter Example

Remainder: $-2a_1b_0 + 2a_0b_0 \neq 0$

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

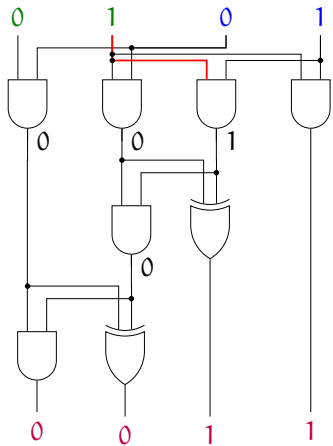
$$01 \cdot 01 = 0001$$

Counter Example

Remainder: $-2a_1b_0 + 2a_0b_0 \neq 0$

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

$01 \cdot 01 = 0001$



Verification Algorithm

Theorem (Kaufmann, Biere, Kauers FMCAD'17) For any circuit C, the set $G(C) \cup B(C)$ forms a Gröbner basis for $I(C)$ w.r.t. any lexicographic monomial order where "gate inputs $<$ gate output".

Verification Algorithm

Theorem (Kaufmann, Biere, Kauers FMCAD'17) For any circuit C, the set $G(C) \cup B(C)$ forms a Gröbner basis for $I(C)$ w.r.t. any lexicographic monomial order where "gate inputs $<$ gate output".

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 Algorithm

Theorem (Kaufmann, Biere, Kauers FMCAD'17) For any circuit C, the set $G(C) \cup B(C)$ forms a Gröbner basis for $I(C)$ w.r.t. any lexicographic monomial order where "gate inputs $<$ gate output".

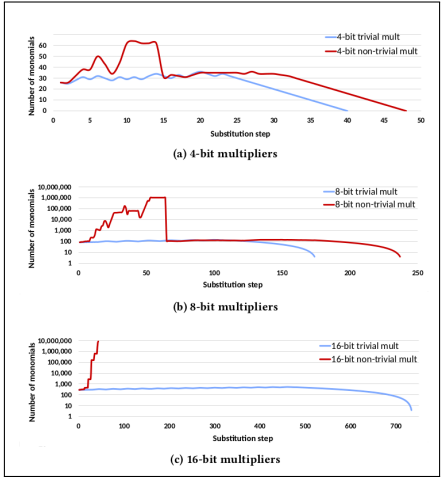
Reduce specification $\sum_{i=0}^{2n-1} 2^i s_i - \left(\sum_{i=0}^{n-1} 2^i a_i\right)\left(\sum_{i=0}^{n-1} 2^i b_i\right)$ by elements of $G(C) \cup B(C)$ until no further reduction is possible, then C is a multiplier iff remainder is zero.

Computational Problems

- 👎 The number of monomials in the intermediate results blows-up.
- 👎 8-bit multiplier cannot be verified within 20 minutes.

Verification Algorithm

[MahzoonGroßeDrechsler ICCAD'18]



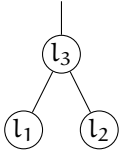
Strategies

- Encoding
  - Embedding negations as literals [KaufmannBeameBiereNordström DATE'22, KonradScholl FMCAD'24]

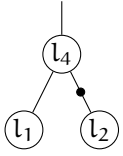
Encoding

[KaufmannBeameBiereNordström DATE'22]

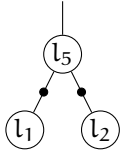
Provide a shorthand notation for inverters.



$$l_3 = l_1 \wedge l_2$$
$$-l_3 + l_1 l_2$$



$$l_4 = l_1 \wedge \neg l_2$$
$$-l_4 - l_1 l_2 + l_1$$



$$l_5 = \neg l_1 \wedge \neg l_2$$
$$-l_5 + l_1 l_2 - l_1 - l_2 + 1$$

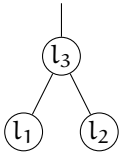
Encoding

[KaufmannBeameBiereNordström DATE'22]

Provide a shorthand notation for inverters.

Dual variables.

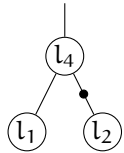
Whenever two variables $l_i, f_i \in \{0, 1\}$ fulfill the relation $f_i = 1 - l_i$, we have $f_i = \mathrm{dual}(l_i)$.



$$l_3 = l_1 \wedge l_2$$

$$-l_3 + l_1 l_2$$

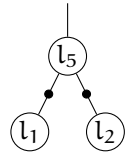
$$-l_3 + l_1 l_2$$



$$l_4 = l_1 \wedge \neg l_2$$

$$-l_4 - l_1 l_2 + l_1$$

$$-l_4 + l_1 f_2$$



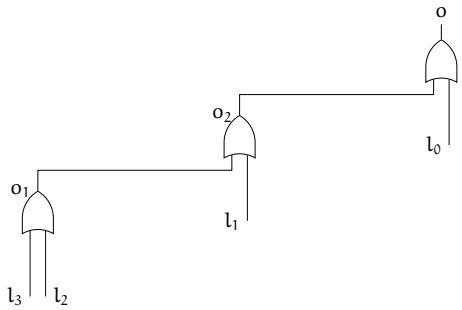
$$l_5 = \neg l_1 \wedge \neg l_2$$

$$-l_5 + l_1 l_2 - l_1 - l_2 + 1$$

$$-l_5 + f_1 f_2$$

OR Gates

$$
\begin{aligned}
o = o_2 \vee x_0 \quad & -o + o_2 + l_0 - o_2 l_0, \\
o_2 = o_1 \vee l_1 \quad & -o_2 + o_1 + l_1 - o_1 l_1, \\
o_1 = l_3 \vee l_2 \quad & -o_1 + l_3 + l_2 - l_3 l_2
\end{aligned}
$$



$$o = l_0 + l_1 - l_0 l_1 + l_2 - l_0 l_2 - l_1 l_2 + l_0 l_1 l_2 + l_3 - l_0 l_3 - l_1 l_3 + l_0 l_1 l_3 - l_2 l_3 + l_0 l_2 l_3 + l_1 l_2 l_3 - l_0 l_1 l_2 l_3$$

$$o = 1 - f_0 f_1 f_2 f_3$$

Strategies

- Encoding
  - Embedding negations as literals [KaufmannBeameBiereNordström DATE'22, KonradScholl FMCAD'24]
- Preprocessing
  - Variable Elimination [MahzoonGroßeDrechsler DAC'19, RitircBiereKauers DATE'18]
- Reduction
  - Incremental Algorithm [RitircBiereKauers FMCAD'17]
  - Dynamic Reduction Order [MahzoonGroßeSchollDrechsler DATE'20, KonradScholl FMCAD'24]

Strategies

- Encoding
  - Embedding negations as literals [KaufmannBeameBiereNordström DATE'22, KonradScholl FMCAD'24]
- Preprocessing
  - Variable Elimination [MahzoonGroßeDrechsler DAC'19, RitircBiereKauers DATE'18]
- Reduction
  - Incremental Algorithm [RitircBiereKauers FMCAD'17]
  - Dynamic Reduction Order [MahzoonGroßeSchollDrechsler DATE'20, KonradScholl FMCAD'24]

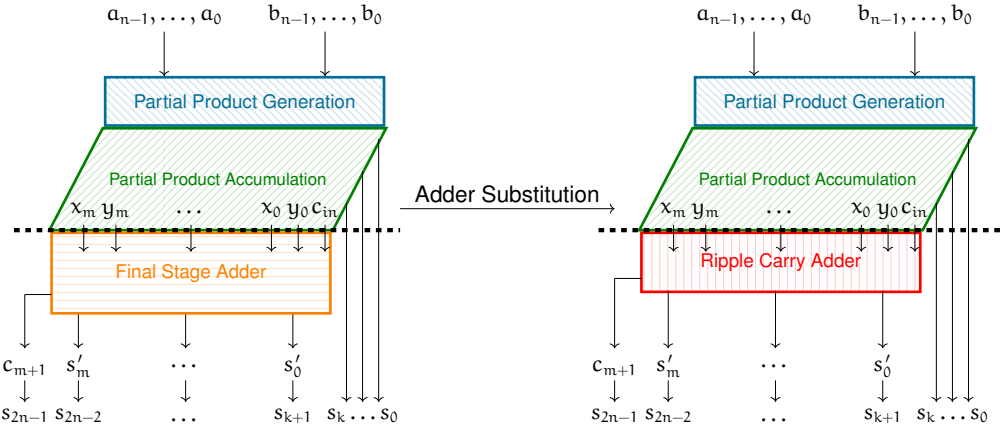
Demo: `d-kfmnn.github.io/circa`

Strategies

- Encoding
  - Embedding negations as literals [KaufmannBeameBiereNordström DATE'22, KonradScholl FMCAD'24]
- Preprocessing
  - Variable Elimination [MahzoonGroßeDrechsler DAC'19, RitircBiereKauers DATE'18]
- Reduction
  - Incremental Algorithm [RitircBiereKauers FMCAD'17]
  - Dynamic Reduction Order [MahzoonGroßeSchollDrechsler DATE'20, KonradScholl FMCAD'24]
- Tricky: Final Stage Adder
  - Include SAT or BDDs [KaufmannBiereKauers FMCAD'19, DrechslerMahzoon ISEEIE'22]

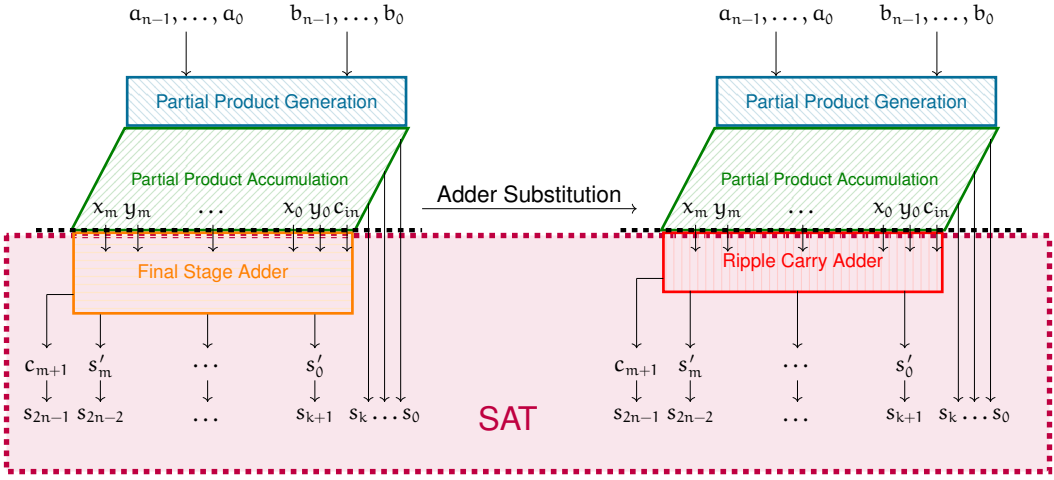
SAT & Computer Algebra

[KaufmannBiereKauers FMCAD'19]



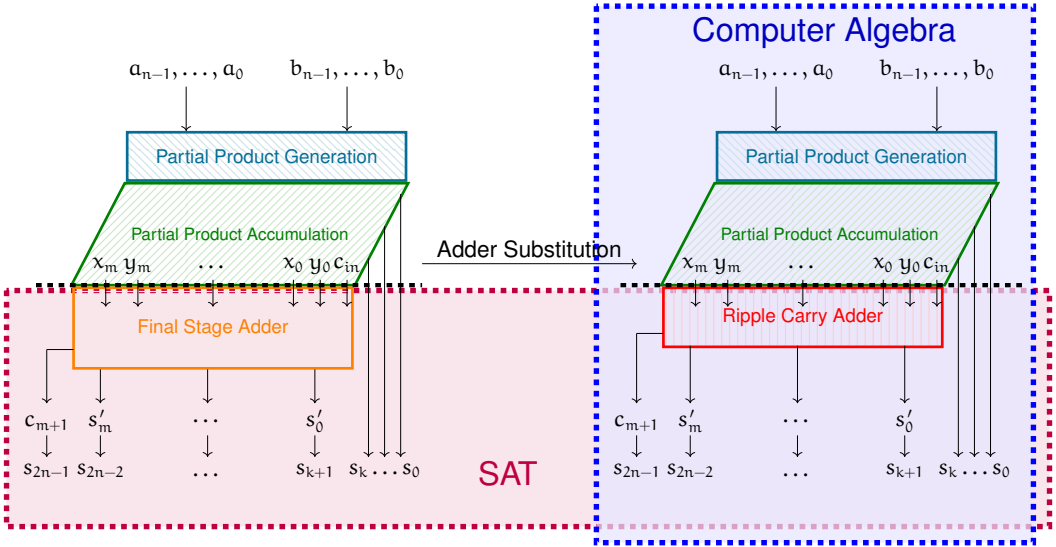
SAT & Computer Algebra

[KaufmannBiereKauers FMCAD'19]



SAT & Computer Algebra

[KaufmannBiereKauers FMCAD'19]



```
daniela@daninovo:~/algprop/src/amulet2$ ./amulet ../benchmarks/btor/btor512.aig -verify -v2
```

```
daniela@dantnovo:~/algprop/src/amulet2$ ./amulet ../benchmarks/btor/btor512.aig -verify -v2
[amulet2] AMulet Version 2.0
[amulet2] Aiger multiplier examination tool
[amulet2] Copyright(C) 2020, Daniela Kaufmann, Johannes Kepler University Linz
[amulet2] selected mode: verification
[amulet2] reading '../benchmarks/btor/btor512.aig'
[amulet2] MILOA 2092544 1024 0 1024 2091520
[amulet2] assuming ordering as in BTOR generated benchmarks
[amulet2] (a[0], a[1], ..., a[511]) =(input[0], input[2], ..., input[1022])
[amulet2] (b[0], b[1], ..., b[511]) =(input[1], input[3], ..., input[1023])
[amulet2] (s[0], ..., s[1023]) =(output[0], ..., output[1023])
[amulet2] allocating 2093568 gates
[amulet2] found 262144 partial products
[amulet2] assuming simple pp generator
[amulet2] found 522752 xor-gates
[amulet2] marking xor chain gates
[amulet2] marked 0 xor gates in last slice
[amulet2] remove internal xor gates
```

```
daniela@daninovo:~/algprop/src/amulet2$ ./amulet ../benchmarks/btor/btor512.aig -verify -v2
[amulet2] AMulet Version 2.0
[amulet2] Aiger multiplier examination tool
[amulet2] Copyright(C) 2020, Daniela Kaufmann, Johannes Kepler University Linz
[amulet2] selected mode: verification
[amulet2] reading '../benchmarks/btor/btor512.aig'
[amulet2] MILOA 2092544 1024 0 1024 2091520
[amulet2] assuming ordering as in BTOR generated benchmarks
[amulet2] (a[0], a[1], ..., a[511]) =(input[0], input[2], ..., input[1022])
[amulet2] (b[0], b[1], ..., b[511]) =(input[1], input[3], ..., input[1023])
[amulet2] (s[0], ..., s[1023]) =(output[0], ..., output[1023])
[amulet2] allocating 2093568 gates
[amulet2] found 262144 partial products
[amulet2] assuming simple pp generator
[amulet2] found 522752 xor-gates
[amulet2] marking xor chain gates
[amulet2] marked 0 xor gates in last slice
[amulet2] remove internal xor gates
[amulet2] removed 522752 internal xor gates
[amulet2] slicing based on xor
[amulet2] remove single occurence gates
[amulet2] removed 0 single occurence gates
[amulet2] moved 0 gates to smaller slices
[amulet2] remove gates that are not assigned to slices
[amulet2] removed 0 gates that are not assigned to slices
[amulet2]
[amulet2]
[amulet2] started reducing
[amulet2] reducing by slice 1023
[amulet2] after reducing by slice 1023
[amulet2] remainder is -8988465674311579538646525953945123668089884894711532863671504057886633790275048156635423866120376801056005693993569667882939488440720831124642371531973706218888394671243274263815110980062304705972654147604250
2884419075341171231440736956555270413618581675255342293149119973622969239858152417678164812112068608*l4185088+89884656743115795386465259539451236680898848947115328636715040578866337902750481566354238661203768010560056939935696678829
394884407208311246423715319737062188883946712432742638151109800623047059726541476042502884419075341171231440736956555270413618581675255342293149119973622969239858152417678164812112068608;
[amulet2]
[amulet2] reducing by slice 1022
[amulet2] after reducing by slice 1022
[amulet2] remainder is -4494232837155789769323262976972561834044942447355766431835752028943316895137524078317711933060188400528002846996784833941469744220360415562321185765986853109444197335621637131907555490031152352986327073802125
1442209537670585615720368478277635206809290837627671146574559986811484619929076208839082406056034304*l4185072-4494232837155789769323262976972561834044942447355766431835752028943316895137524078317711933060188400528002846996784833941 4
6974422036041556232118576598685310944419733562163713190755549003115235298632707380212514422095376705856157203684782776352068092908376276711465745599868114846199290762088390824060560343 04*l4185070+898846567431157953864652595394512366
8089884894711532863671504057886633790275048156635423866120376801056005693993569667882939488440720831124642371531973706218888394671243274263815110980062304705972654147604250288441907534117123144073695655527041361858167525534229314911
9973622969239858152417678164812112068608;
[amulet2]
[amulet2] reducing by slice 1021
[amulet2] after reducing by slice 1021
[amulet2] remainder is -2247116418577894884661631488486280917022471223677883215917876014471658447568762039158855966530094200264001423498392416970734872110180207781160592882993426554722098667810818565953777745015576176493163536901062
5721104768835292807860184239138817603404645418813835573287279993405742309964538104419541203028017152*l4185040-22471164185778948846616314884862809170224712236778832159178760144716584475687620391588559665300942002640014234983924169707
34872110180207781160592882993426554722098667810818565953777745015576176493163536901062572110476883529280786018423913881760340464541881383557328727999340574230996453810441954120302801 7152*l4185038-224711641857789488466163148848628091
70224712236778832159178760144716584475687620391588559665300942002640014234983924169707348721101802077811605928829934265547220986678108185659537777450155761764931635369010625721104768835292807860184239138817603404645418813835573287 27
9993405742309964538104419541203028017152*l4185036+67413492557336846539848944654588427510674136710336496477536280434149753427062861174765678995902826007920042704951772509122046163305406233434817786489802796641662960034324556978613332
3504672852947949061070318771633143065058784235805527174164528102139362564415067198618399802172269298936143132586236090840514 56;
[amulet2]
[amulet2] reducing by slice 1020
[amulet2] after reducing by slice 1020
[amulet2] remainder is -1123558209288947442330815744243140458511235611838941607958938007235829223784381019579427983265047100132000711749196208485367436055090103890580296441496713277361049333905409282976888872507788088246581768450531
2860552384417646403930092119569408801702322709406917786643639996702871154982269052209770601514008576*l4184992-11235582092889474423308157442431404585112356118389416079589380072358292237843810195794279832650471001320007117491962084853
67436055090103890580296441496713277361049333905409282976888872507788088246581768450531286055238441764640393009211956940880170232270940691778664363999670287115498226905220977060151400857 6*l4184990-112355820928894744233081574424314045
8511235611838941607958938007235829223784381019579427983265047100132000711749196208485367436055090103890580296441496713277361049333905409282976888872507788088246581768450531286055238441764640393009211956940880170232270940691778664363
9996702871154982269052209770601514008576*l4184988-11235582092889474423308157442431404585112356118389416079589380072358292237843810195794279832650471001320007117491962084853674360550901038905802964414967132773610493339054092829768888
7250778808824658176845053128605523844176464039300921195694088017023227094069177866436399967028711549822690522097706015140085 76*l4184986+449423283715578976932326297697256183404494244735576643183575202894331689513752407831771193306018
84005280028469967848339414697442203604155623211857659868531094441973356216371319075554900311523529863270738021251442209537670585615720368478277635206809290837627671146574559986811484619929076208839082406056034304;
```

5162325512880128*l4182880+23316027292373176903837729067545639397816168702712411297976008450938838872478610025451996723029541872856460082685956474916072281416664753588702440802544098773361619478017526624276258340615719604726428300365029129564274304197859718208744223626663688902106923535445915756829095313676110665757759533718962176;
[amulet2]
[amulet2] reducing by slice 1006
[amulet2] after reducing by slice 1006
[amulet2] remainder is -685765508599211085406992031398401158759299079491541508764000248557024672719959118395646962442045349201660590667234013968119772982843080987903012964780708787451812337588750783066948774723991753080189067657794974398949244241113521123786594812548932026532556574571938698730267509225767960757581162756440064*l4182640-685765508599211085406992031398401158759299079491541508764000248557024672719959118395646962442045349201660590667234013968119772982843080987903012964780708787451812337588750783066948774723991753080189067657794974398949244241113521123786594812548932026532556574571938698730267509225767960757581162756440064*l4182638-685765508599211085406992031398401158759299079491541508764000248557024672719959118395646962442045349201660590667234013968119772982843080987903012964780708787451812337588750783066948774723991753080189067657794974398949244241113521123786594812548932026532556574571938698730267509225767960757581162756440064*l4182636-685765508599211085406992031398401158759299079491541508764000248557024672719959118395646962442045349201660590667234013968119772982843080987903012964780708787451812337588750783066948774723991753080189067657794974398949244241113521123786594812548932026532556574571938698730267509225767960757581162756440064*l4182634-685765508599211085406992031398401158759299079491541508764000248557024672719959118395646962442045349201660590667234013968119772982843080987903012964780708787451812337588750783066948774723991753080189067657794974398949244241113521123786594812548932026532556574571938698730267509225767960757581162756440064*l4182632-685765508599211085406992031398401158759299079491541508764000248557024672719959118395646962442045349201660590667234013968119772982843080987903012964780708787451812337588750783066948774723991753080189067657794974398949244241113521123786594812548932026532556574571938698730267509225767960757581162756440064*l4182630-685765508599211085406992031398401158759299079491541508764000248557024672719959118395646962442045349201660590667234013968119772982843080987903012964780708787451812337588750783066948774723991753080189067657794974398949244241113521123786594812548932026532556574571938698730267509225767960757581162756440064*l4182628-685765508599211085406992031398401158759299079491541508764000248557024672719959118395646962442045349201660590667234013968119772982843080987903012964780708787451812337588750783066948774723991753080189067657794974398949244241113521123786594812548932026532556574571938698730267509225767960757581162756440064*l4182626-685765508599211085406992031398401158759299079491541508764000248557024672719959118395646962442045349201660590667234013968119772982843080987903012964780708787451812337588750783066948774723991753080189067657794974398949244241113521123786594812548932026532556574571938698730267509225767960757581162756440064*l4182624-685765508599211085406992031398401158759299079491541508764000248557024672719959118395646962442045349201660590667234013968119772982843080987903012964780708787451812337588750783066948774723991753080189067657794974398949244241113521123786594812548932026532556574571938698730267509225767960757581162756440064*l4182622-685765508599211085406992031398401158759299079491541508764000248557024672719959118395646962442045349201660590667234013968119772982843080987903012964780708787451812337588750783066948774723991753080189067657794974398949244241113521123786594812548932026532556574571938698730267509225767960757581162756440064*l4182620-685765508599211085406992031398401158759299079491541508764000248557024672719959118395646962442045349201660590667234013968119772982843080987903012964780708787451812337588750783066948774723991753080189067657794974398949244241113521123786594812548932026532556574571938698730267509225767960757581162756440064*l4182618-685765508599211085406992031398401158759299079491541508764000248557024672719959118395646962442045349201660590667234013968119772982843080987903012964780708787451812337588750783066948774723991753080189067657794974398949244241113521123786594812548932026532556574571938698730267509225767960757581162756440064*l4182616-685765508599211085406992031398401158759299079491541508764000248557024672719959118395646962442045349201660590667234013968119772982843080987903012964780708787451812337588750783066948774723991753080189067657794974398949244241113521123786594812548932026532556574571938698730267509225767960757581162756440064*l4182614-685765508599211085406992031398401158759299079491541508764000248557024672719959118395646962442045349201660590667234013968119772982843080987903012964780708787451812337588750783066948774723991753080189067657794974398949244241113521123786594812548932026532556574571938698730267509225767960757581162756440064*l4182612-685765508599211085406992031398401158759299079491541508764000248557024672719959118395646962442045349201660590667234013968119772982843080987903012964780708787451812337588750783066948774723991753080189067657794974398949244241113521123786594812548932026532556574571938698730267509225767960757581162756440064*l4182610-685765508599211085406992031398401158759299079491541508764000248557024672719959118395646962442045349201660590667234013968119772982843080987903012964780708787451812337588750783066948774723991753080189067657794974398949244241113521123786594812548932026532556574571938698730267509225767960757581162756440064*l4182608-685765508599211085406992031398401158759299079491541508764000248557024672719959118395646962442045349201660590667234013968119772982843080987903012964780708787451812337588750783066948774723991753080189067657794974398949244241113521123786594812548932026532556574571938698730267509225767960757581162756440064*l4182606+12343779154785799537325856565171220857667383430847747157752004474026444108959264131121645323956816285629890632010212251426155913691175457782254233366052758174132622076597514095205077945031851555443403217840309539181086396340043380228158706625880776477586018342294896577144815166063823293636460929615921152;
[amulet2]
[amulet2] reducing by slice 1005
[amulet2] after reducing by slice 1005
[amulet2] remainder is -342882754299605542703496015699200579379649539745770754382000124278512336359979559197823481221022674600830295333617006984059886491421540493951506482390354393725906168794375391533474387361995876540094533828897487199474622120556760561893297406274466013266278287285969349365133754612883980378790581378220032*l4182352-342882754299605542703496015699200579379649539745770754382000124278512336359979559197823481221022674600830295333617006984059886491421540493951506482390354393725906168794375391533474387361995876540094533828897487199474622120556760561893297406274466013266278287285969349365133754612883980378790581378220032*l4182350-342882754299605542703496015699200579379649539745770754382000124278512336359979559197823481221022674600830295333617006984059886491421540493951506482390354393725906168794375391533474387361995876540094533828897487199474622120556760561893297406274466013266278287285969349365133754612883980378790581378220032*l4182348-342882754299605542703496015699200579379649539745770754382000124278512336359979559197823481221022674600830295333617006984059886491421540493951506482390354393725906168794375391533474387361995876540094533828897487199474622120556760561893297406274466013266278287285969349365133754612883980378790581378220032*l4182346-342882754299605542703496015699200579379649539745770754382000124278512336359979559197823481221022674600830295333617006984059886491421540493951506482390354393725906168794375391533474387361995876540094533828897487199474622120556760561893297406274466013266278287285969349365133754612883980378790581378220032*l4182344-342882754299605542703496015699200579379649539745770754382000124278512336359979559197823481221022674600830295333617006984059886491421540493951506482390354393725906168794375391533474387361995876540094533828897487199474622120556760561893297406274466013266278287285969349365133754612883980378790581378220032*l4182342-342882754299605542703496015699200579379649539745770754382000124278512336359979559197823481221022674600830295333617006984059886491421540493951506482390354393725906168794375391533474387361995876540094533828897487199474622120556760561893297406274466013266278287285969349365133754612883980378790581378220032*l4182340-342882754299605542703496015699200579379649539745770754382000124278512336359979559197823481221022674600830295333617006984059886491421540493951506482390354393725906168794375391533474387361995876540094533828897487199474622120556760561893297406274466013266278287285969349365133754612883980378790581378220032*l4182338-342882754299605542703496015699200579379649539745770754382000124278512336359979559197823481221022674600830295333617006984059886491421540493951506482390354393725906168794375391533474387361995876540094533828897487199474622120556760561893297406274466013266278287285969349365133754612883980378790581378220032*l4182336-342882754299605542703496015699200579379649539745770754382000124278512336359979559197823481221022674600830295333617006984059886491421540493951506482390354393725906168794375391533474387361995876540094533828897487199474622120556760561893297406274466013266278287285969349365133754612883980378790581378220032*l4182334-342882754299605542703496015699200579379649539745770754382000124278512336359979559197823481221022674600830295333617006984059886491421540493951506482390354393725906168794375391533474387361995876540094533828897487199474622120556760561893297406274466013266278287285969349365133754612883980378790581378220032*l4182332-342882754299605542703496015699200579379649539745770754382000124278512336359979559197823481221022674600830295333617006984059886491421540493951506482390354393725906168794375391533474387361995876540094533828897487199474622120556760561893297406274466013266278287285969349365133754612883980378790581378220032*l4182330-342882754299605542703496015699200579379649539745770754382000124278512336359979559197823481221022674600830295333617006984059886491421540493951506482390354393725906168794375391533474387361995876540094533828897487199474622120556760561893297406274466013266278287285969349365133754612883980378790581378220032*l4182328-342882754299605542703496015699200579379649539745770754382000124278512336359979559197823481221022674600830295333617006984059886491421540493951506482390354393725906168794375391533474387361995876540094533828897487199474622120556760561893297406274466013266278287285969349365133754612883980378790581378220032*l4182326-342882754299605542703496015699200579379649539745770754382000124278512336359979559197823481221022674600830295333617006984059886491421540493951506482390354393725906168794375391533474387361995876540094533828897487199474622120556760561893297406274466013266278287285969349365133754612883980378790581378220032*l4182324-342882754299605542703496015699200579379649539745770754382000124278512336359979559197823481221022674600830295333617006984059886491421540493951506482390354393725906168794375391533474387361995876540094533828897487199474622120556760561893297406274466013266278287285969349365133754612883980378790581378220032*l4182322-342882754299605542703496015699200579379649539745770754382000124278512336359979559197823481221022674600830295333617006984059886491421540493951506482390354393725906168794375391533474387361995876540094533828897487199474622120556760561893297406274466013266278287285969349365133754612883980378790581378220032*l4182320-342882754299605542703496015699200579379649539745770754382000124278512336359979559197823481221022674600830295333617006984059886491421540493951506482390354393725906168794375391533474387361995876540094533828897487199474622120556760561893297406274466013266278287285969349365133754612883980378790581378220032*l4182318-342882754299605542703496015699200579379649539745770754382000124278512336359979559197823481221022674600830295333617006984059886491421540493951506482390354393725906168794375391533474387361995876540094533828897487199474622120556760561893297406274466013266278287285969349365133754612883980378790581378220032*l4182316+651477233169250531136642429828481100821334125516964433258002361291734390839611624758646143199430817415775611338723132697137843337009269385078623165416733480792217207093132439136013359877921654261796142749052256790017820290578450675972650719214854252059287458433417637937541337644795271970210

```
[amulet2] remainder is -4096*l3040-4096*l3038-4096*l3036-4096*l3034-4096*l3032-4096*l3030-4096*l3028-4096*l3026-4096*l3024-4096*l3022+4096*l3016+40960;
[amulet2]
[amulet2] reducing by slice 11
[amulet2] after reducing by slice 11
[amulet2] remainder is -2048*l2870-2048*l2868-2048*l2866-2048*l2864-2048*l2862-2048*l2860-2048*l2858-2048*l2856-2048*l2854+2048*l2848+18432;
[amulet2]
[amulet2] reducing by slice 10
[amulet2] after reducing by slice 10
[amulet2] remainder is -1024*l2716-1024*l2714-1024*l2712-1024*l2710-1024*l2708-1024*l2706-1024*l2704-1024*l2702+1024*l2696+8192;
[amulet2]
[amulet2] reducing by slice 9
[amulet2] after reducing by slice 9
[amulet2] remainder is -512*l2578-512*l2576-512*l2574-512*l2572-512*l2570-512*l2568-512*l2566+512*l2560+3584;
[amulet2]
[amulet2] reducing by slice 8
[amulet2] after reducing by slice 8
[amulet2] remainder is -256*l2456-256*l2454-256*l2452-256*l2450-256*l2448-256*l2446+256*l2440+1536;
[amulet2]
[amulet2] reducing by slice 7
[amulet2] after reducing by slice 7
[amulet2] remainder is -128*l2350-128*l2348-128*l2346-128*l2344-128*l2342+128*l2336+640;
[amulet2]
[amulet2] reducing by slice 6
[amulet2] after reducing by slice 6
[amulet2] remainder is -64*l2260-64*l2258-64*l2256-64*l2254+64*l2248+256;
[amulet2]
[amulet2] reducing by slice 5
[amulet2] after reducing by slice 5
[amulet2] remainder is -32*l2186-32*l2184-32*l2182+32*l2176+96;
[amulet2]
[amulet2] reducing by slice 4
[amulet2] after reducing by slice 4
[amulet2] remainder is -16*l2128-16*l2126+16*l2120+32;
[amulet2]
[amulet2] reducing by slice 3
[amulet2] after reducing by slice 3
[amulet2] remainder is -8*l2086+8*l2080+8;
[amulet2]
[amulet2] reducing by slice 2
[amulet2] after reducing by slice 2
[amulet2] remainder is 4*l2056;
[amulet2]
[amulet2] reducing by slice 1
[amulet2] after reducing by slice 1
[amulet2] remainder is 0;
[amulet2]
[amulet2] reducing by slice 0
[amulet2] after reducing by slice 0
[amulet2] remainder is 0;
[amulet2]
[amulet2]
[amulet2] CORRECT MULTIPLIER
[amulet2]
[amulet2] maximun resident set size:                1966.98 MB
[amulet2] used time for initializing:                 1.33 seconds
[amulet2] used time for slicing/elimination:          7.62 seconds
[amulet2] used time for reduction:                   30.72 seconds
[amulet2] used time for freeing memory:               3.40 seconds
[amulet2] total process time:                        43.07 seconds
daniela@danilnovo:~/algprop/src/amulet2$
```

Strategies

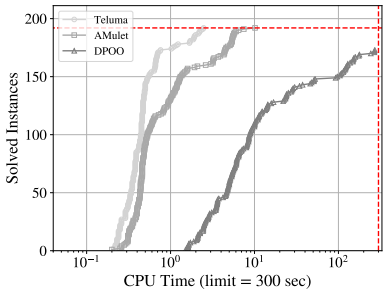
- Encoding
  - Embedding negations as literals [KaufmannBeameBiereNordström DATE'22, KonradScholl FMCAD'24]
- Preprocessing
  - Variable Elimination [MahzoonGroßeDrechsler DAC'19, RitircBiereKauers DATE'18]
- Reduction
  - Incremental Algorithm [RitircBiereKauers FMCAD'17]
  - Dynamic Reduction Order [MahzoonGroßeSchollDrechsler DATE'20, KonradScholl FMCAD'24]
- Tricky: OR Gates in final stage adder
  - Include SAT or BDDs [KaufmannBiereKauers FMCAD'19, DrechslerMahzoon ISEEIE'22]

All of these strategies rely on a **lexicographic term ordering** and aim to find an optimal reduction order.

Strategies

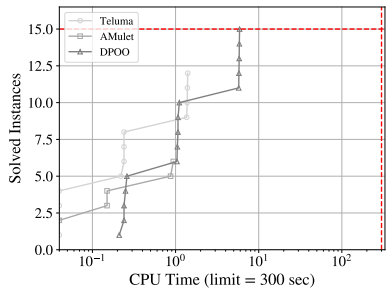
Unoptimized multipliers

- 192 architectures
- 64 bit inputs



Optimized multipliers

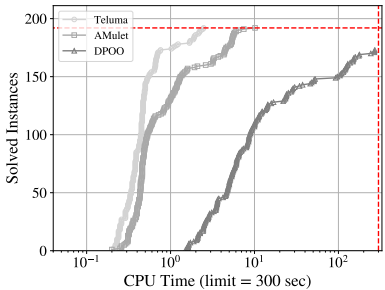
- 5 optimization scripts
- for 32, 64, 128 bit inputs



Strategies

Unoptimized multipliers

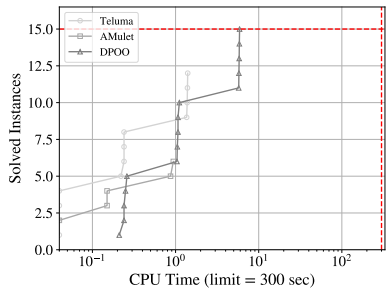
- 192 architectures
- 64 bit inputs



👎 Fast approaches are overfitted

Optimized multipliers

- 5 optimization scripts
- for 32, 64, 128 bit inputs



A New Idea

- Change monomial order to degree-lexicographic order (Kaufmann, Berthomieu TACAS'25)

	$\prec_{\text{lex}}$	$\prec_{\text{dlex}}$
GB Computation	✅ Easy	⚠️ Hard
Spec Reduction	⚠️ Hard	✅ Easy

A New Idea

Change monomial order to degree-lexicographic order (Kaufmann, Berthomieu TACAS'25)

	$\prec_{\text{lex}}$	$\prec_{\text{dlex}}$
GB Computation	Easy	Hard
Spec Reduction	Hard	Easy

Theorem (Kaufmann, Berthomieu TACAS'25) If the specification polynomial is linear, then the linear polynomials in a $\prec_{\text{dlex}}$-GB suffice to verify the circuit.

Verification via Linearization

Input: Circuit C, specification polynomial S
Output: Determine whether C fulfils S

1 $G \leftarrow G(C) \cup B(C)$
2 $S_{\text{lin}}, G \leftarrow$ Linearize-Spec(S, G)
3 $L \leftarrow$ LINEAR-POLIES-OF-$\prec_{\text{dlex}}$-GB(G)
4 $S_{\text{lin}} \leftarrow$ Linear-Reduce(S_{lin}, L)
5 **return** $S_{\text{lin}} \stackrel{?}{=} 0$

Verification via Linearization

Input: Circuit C, specification polynomial S
Output: Determine whether C fulfils S

1 $G \leftarrow G(C) \cup B(C)$
2 $S_{\text{lin}}, G \leftarrow$ Linearize-Spec(S, G)
3 $L \leftarrow$ LINEAR-POLIES-OF-$\prec_{\text{dlex}}$-GB(G)
4 $S_{\text{lin}} \leftarrow$ Linear-Reduce(S_{lin}, L)
5 **return** $S_{\text{lin}} \stackrel{?}{=} 0$

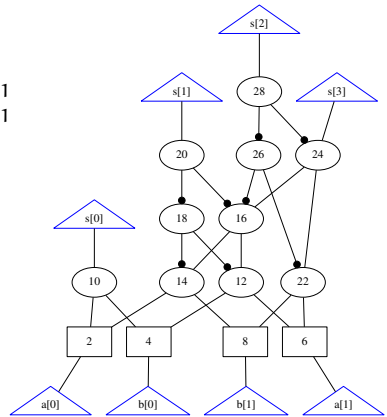
$S_{\text{lin}}, G \leftarrow$ **Linearize-Spec**(S, G)

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

$$-s_3 + l_{24}$$
$$-s_2 + l_{28}$$
$$-s_1 + l_{20}$$
$$-s_0 + l_{10}$$
$$-l_{28} + l_{26}l_{24} - l_{26} - l_{24} + 1$$
$$-l_{26} + l_{22}l_{16} - l_{22} - l_{16} + 1$$
$$-l_{24} + l_{22}l_{16}$$
$$-l_{22} + a_1 b_1$$
$$-l_{20} + l_{18}l_{16} - l_{18} - l_{16} + 1$$
$$-l_{18} + l_{14}l_{12} - l_{14} - l_{12} + 1$$
$$-l_{16} + l_{14}l_{12}$$
$$-l_{14} + a_0 b_1$$
$$-l_{12} + a_1 b_0$$
$$-l_{10} + a_0 b_0$$

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

$$8s_3 + 4s_2 + 2s_1 + s_0 - 4b_1a_1 - 2b_1a_0 - 2b_0a_1 - b_0a_0$$



$S_{\text{lin}}, G \leftarrow$ **Linearize-Spec**(S, G)

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

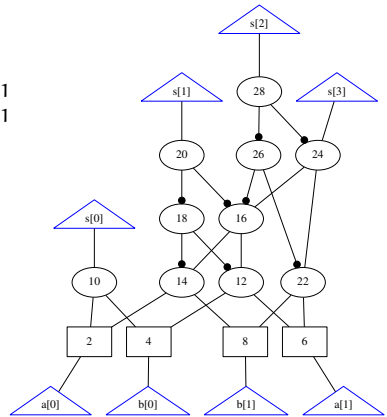
$$-s_3 + l_{24}$$
$$-s_2 + l_{28}$$
$$-s_1 + l_{20}$$
$$-s_0 + l_{10}$$
$$-l_{28} + l_{26}l_{24} - l_{26} - l_{24} + 1$$
$$-l_{26} + l_{22}l_{16} - l_{22} - l_{16} + 1$$
$$-l_{24} + l_{22}l_{16}$$
$$-l_{22} + a_1 b_1$$
$$-l_{20} + l_{18}l_{16} - l_{18} - l_{16} + 1$$
$$-l_{18} + l_{14}l_{12} - l_{14} - l_{12} + 1$$
$$-l_{16} + l_{14}l_{12}$$
$$-l_{14} + a_0 b_1$$
$$-l_{12} + a_1 b_0$$
$$-l_{10} + a_0 b_0$$

Extension polynomials $G \subseteq \mathbb{Z}[X]$.

$$-t_{11} + a_1 b_1 \qquad -t_{01} + a_0 b_1$$
$$-t_{10} + a_1 b_0 \qquad -t_{00} + a_0 b_0$$

Linear Specification $S_{\text{lin}} \in \mathbb{Z}[X]$.

$$8s_3 + 4s_2 + 2s_1 + s_0 - 4t_{11} - 2t_{10} - 2t_{01} - t_{00}$$



$S_{lin}, G \leftarrow$ **Linearize-Spec**(S, G)

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

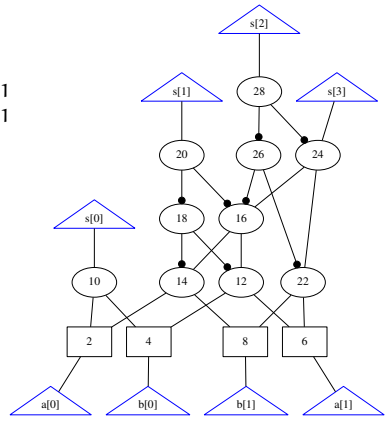
$$-s_3 + l_{24}$$
$$-s_2 + l_{28}$$
$$-s_1 + l_{20}$$
$$-s_0 + l_{10}$$
$$-l_{28} + l_{26}l_{24} - l_{26} - l_{24} + 1$$
$$-l_{26} + l_{22}l_{16} - l_{22} - l_{16} + 1$$
$$-l_{24} + l_{22}l_{16}$$
$$-l_{22} + a_1 b_1$$
$$-l_{20} + l_{18}l_{16} - l_{18} - l_{16} + 1$$
$$-l_{18} + l_{14}l_{12} - l_{14} - l_{12} + 1$$
$$-l_{16} + l_{14}l_{12}$$
$$-l_{14} + a_0 b_1$$
$$-l_{12} + a_1 b_0$$
$$-l_{10} + a_0 b_0$$

Extension polynomials $G \subseteq \mathbb{Z}[X]$.

$$-t_{11} + a_1 b_1 \qquad -t_{01} + a_0 b_1$$
$$-t_{10} + a_1 b_0 \qquad -t_{00} + a_0 b_0$$

Linear Specification $S_{lin} \in \mathbb{Z}[X]$.

$$8s_3 + 4s_2 + 2s_1 + s_0 - 4t_{11} - 2t_{10} - 2t_{01} - t_{00}$$



$S_{\text{lin}}, G \leftarrow$ **Linearize-Spec**(S, G)

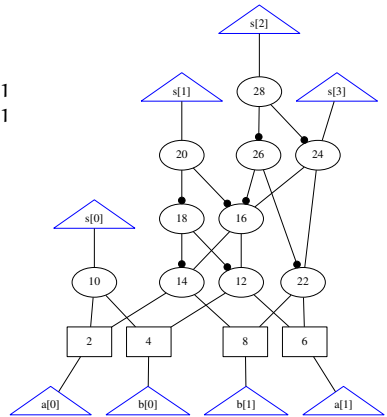
Gate polynomials $G(C) \subseteq \mathbb{Z}[X]$.

$$-s_3 + l_{24}$$
$$-s_2 + l_{28}$$
$$-s_1 + l_{20}$$
$$-s_0 + l_{10}$$
$$-l_{28} + l_{26}l_{24} - l_{26} - l_{24} + 1$$
$$-l_{26} + l_{22}l_{16} - l_{22} - l_{16} + 1$$
$$-l_{24} + l_{22}l_{16}$$
$$-l_{22} + a_1 b_1$$
$$-l_{20} + l_{18}l_{16} - l_{18} - l_{16} + 1$$
$$-l_{18} + l_{14}l_{12} - l_{14} - l_{12} + 1$$
$$-l_{16} + l_{14}l_{12}$$
$$-l_{14} + a_0 b_1$$
$$-l_{12} + a_1 b_0$$
$$-l_{10} + a_0 b_0$$

Extension polynomials $G \subseteq \mathbb{Z}[X]$.

Linear Specification $S_{\text{lin}} \in \mathbb{Z}[X]$.

$$8s_3 + 4s_2 + 2s_1 + s_0 - 4l_{22} - 2l_{14} - 2l_{12} - l_{10}$$



Verification via Linearization

Input: Circuit C, specification polynomial S
Output: Determine whether C fulfils S

1 $G \leftarrow G(C) \cup B(C)$
2 $S_{\text{lin}}, G \leftarrow$ Linearize-Spec(S, G)
3 $L \leftarrow$ LINEAR-POLIES-OF-$\prec_{\text{dlex}}$-GB(G)
4 $S_{\text{lin}} \leftarrow$ Linear-Reduce(S_{lin}, L)
5 **return** $S_{\text{lin}} \stackrel{?}{=} 0$

Verification via Linearization

Input: Circuit C, specification polynomial S
Output: Determine whether C fulfils S

1 $G \leftarrow G(C) \cup B(C)$
2 $S_{\text{lin}}, G \leftarrow$ Linear-Spec(S, G)
3 $L \leftarrow$ LINEAR-POLIES-OF-$\prec_{\text{dlex}}$-GB(G)
4 $S_{\text{lin}} \leftarrow$ Linear-Reduce(S_{lin}, L)
5 **return** $S_{\text{lin}} \stackrel{?}{=} 0$

compute $\prec_{\text{dlex}}$-GB of G

Verification via Linearization

Input: Circuit C, specification polynomial S
Output: Determine whether C fulfils S

1 $G \leftarrow G(C) \cup B(C)$
2 $S_{\text{lin}}, G \leftarrow$ Linearize-Spec(S, G)
3 $L \leftarrow$ LINEAR-POLIES-OF-$\prec_{\text{dlex}}$-GB(G)
4 $S_{\text{lin}} \leftarrow$ Linear-Reduce(S_{lin}, L)
5 **return** $S_{\text{lin}} \stackrel{?}{=} 0$

compute $\prec_{\text{dlex}}$-GB of G

Computing a $\prec_{\text{dlex}}$-GB for the whole circuit is infeasible.

Verification via Linearization

Input: Circuit C, specification polynomial S
Output: Determine whether C fulfils S

1 $G \leftarrow G(C) \cup B(C)$
2 $S_{\text{lin}}, G \leftarrow$ Linearize-Spec(S, G)
3 $L \leftarrow$ LINEAR-POLIES-OF-$\prec_{\text{dlex}}$-GB(G)
4 $S_{\text{lin}} \leftarrow$ Linear-Reduce(S_{lin}, L)
5 **return** $S_{\text{lin}} \stackrel{?}{=} 0$

compute $\prec_{\text{dlex}}$-GB of $G' \subseteq G$

Computing a $\prec_{\text{dlex}}$-GB for the whole circuit is infeasible.

$\rightsquigarrow$ work locally: extract sub-circuits and only compute $\prec_{\text{dlex}}$-GBs for them

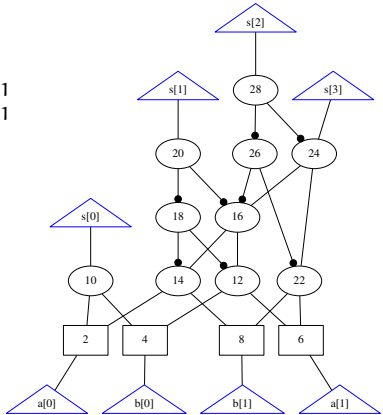
Local Linearization

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

$$
\begin{aligned}
&-s_3 + l_{24} && -l_{22} + a_1 b_1 \\
&-s_2 + l_{28} && -l_{20} + l_{18} l_{16} - l_{18} - l_{16} + 1 \\
&-s_1 + l_{20} && -l_{18} + l_{14} l_{12} - l_{14} - l_{12} + 1 \\
&-s_0 + l_{10} && -l_{16} + l_{14} l_{12} \\
&-l_{28} + l_{26} l_{24} - l_{26} - l_{24} + 1 && -l_{14} + a_0 b_1 \\
&-l_{26} + l_{22} l_{16} - l_{22} - l_{16} + 1 && -l_{12} + a_1 b_0 \\
&-l_{24} + l_{22} l_{16} && -l_{10} + a_0 b_0
\end{aligned}
$$

Specification $\mathcal{S}_n \in \mathbb{Z}[X]$.

$$8s_3 + 4s_2 + 2s_1 + s_0 - 4l_{22} - 2l_{14} - 2l_{12} - l_{10}$$



Local Linearization

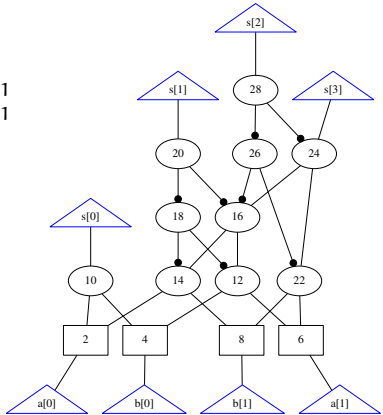
Gate polynomials $G(C) \subseteq \mathbb{Z}[X]$.

$$
\begin{aligned}
&-s_3 + l_{24} && -l_{22} + a_1 b_1 \\
&-s_2 + l_{28} && -l_{20} + l_{18} l_{16} - l_{18} - l_{16} + 1 \\
&-s_1 + l_{20} && -l_{18} + l_{14} l_{12} - l_{14} - l_{12} + 1 \\
&-s_0 + l_{10} && -l_{16} + l_{14} l_{12} \\
&-l_{28} + l_{26} l_{24} - l_{26} - l_{24} + 1 && -l_{14} + a_0 b_1 \\
&-l_{26} + l_{22} l_{16} - l_{22} - l_{16} + 1 && -l_{12} + a_1 b_0 \\
&-l_{24} + l_{22} l_{16} && -l_{10} + a_0 b_0
\end{aligned}
$$

Specification $\mathcal{S}_n \in \mathbb{Z}[X]$.

$$8s_3 + 4s_2 + 2s_1 + s_0 - 4l_{22} - 2l_{14} - 2l_{12} - l_{10}$$

$$4s_2 + 2s_1 + s_0 + 8l_{24} - 4l_{22} - 2l_{14} - 2l_{12} - l_{10}$$



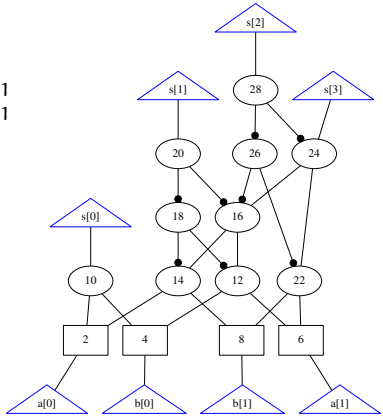
Local Linearization

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

$$
\begin{aligned}
&-s_3 + l_{24} && -l_{22} + a_1 b_1 \\
&-s_2 + l_{28} && -l_{20} + l_{18} l_{16} - l_{18} - l_{16} + 1 \\
&-s_1 + l_{20} && -l_{18} + l_{14} l_{12} - l_{14} - l_{12} + 1 \\
&-s_0 + l_{10} && -l_{16} + l_{14} l_{12} \\
&-l_{28} + l_{26} l_{24} - l_{26} - l_{24} + 1 && -l_{14} + a_0 b_1 \\
&-l_{26} + l_{22} l_{16} - l_{22} - l_{16} + 1 && -l_{12} + a_1 b_0 \\
&-l_{24} + l_{22} l_{16} && -l_{10} + a_0 b_0
\end{aligned}
$$

Specification $\mathcal{S}_n \in \mathbb{Z}[X]$.

$$
\begin{aligned}
8s_3 + 4s_2 + 2s_1 + s_0 - 4l_{22} - 2l_{14} - 2l_{12} - l_{10} \\
4s_2 + 2s_1 + s_0 + 8l_{24} - 4l_{22} - 2l_{14} - 2l_{12} - l_{10} \\
4l_{28} + 8l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12}
\end{aligned}
$$



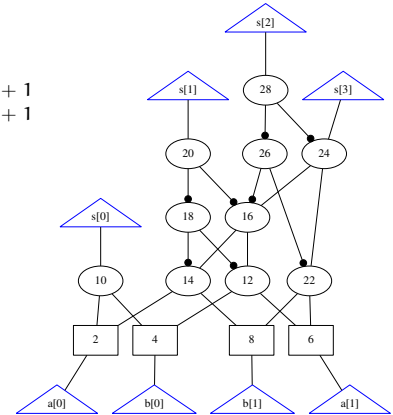
Local Linearization

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

$$-s_3 + l_{24}$$
$$-s_2 + l_{28}$$
$$-s_1 + l_{20}$$
$$-s_0 + l_{10}$$
$$\boxed{-l_{28} + l_{26}l_{24} - l_{26} - l_{24} + 1}$$
$$-l_{26} + l_{22}l_{16} - l_{22} - l_{16} + 1$$
$$-l_{24} + l_{22}l_{16}$$
$$-l_{22} + a_1 b_1$$
$$-l_{20} + l_{18}l_{16} - l_{18} - l_{16} + 1$$
$$-l_{18} + l_{14}l_{12} - l_{14} - l_{12} + 1$$
$$-l_{16} + l_{14}l_{12}$$
$$-l_{14} + a_0 b_1$$
$$-l_{12} + a_1 b_0$$
$$-l_{10} + a_0 b_0$$

Specification $\mathcal{S}_n \in \mathbb{Z}[X]$.

$$8s_3 + 4s_2 + 2s_1 + s_0 - 4l_{22} - 2l_{14} - 2l_{12} - l_{10}$$
$$4s_2 + 2s_1 + s_0 + 8l_{24} - 4l_{22} - 2l_{14} - 2l_{12} - l_{10}$$
$$4l_{28} + 8l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12}$$



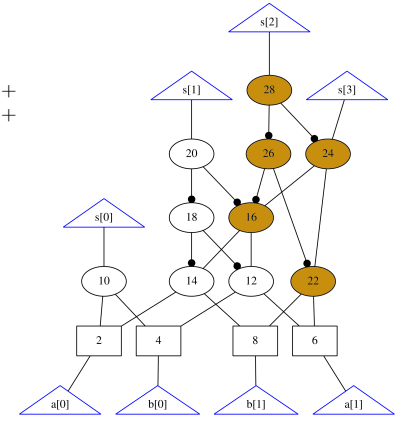
Local Linearization

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

$$-s_3 + l_{24}$$
$$-s_2 + l_{28}$$
$$-s_1 + l_{20}$$
$$-s_0 + l_{10}$$
$$\boxed{-l_{28} + l_{26}l_{24} - l_{26} - l_{24} + 1}$$
$$-l_{26} + l_{22}l_{16} - l_{22} - l_{16} + 1$$
$$-l_{24} + l_{22}l_{16}$$
$$-l_{22} + a_1 b_1$$
$$-l_{20} + l_{18}l_{16} - l_{18} - l_{16} +$$
$$-l_{18} + l_{14}l_{12} - l_{14} - l_{12} +$$
$$-l_{16} + l_{14}l_{12}$$
$$-l_{14} + a_0 b_1$$
$$-l_{12} + a_1 b_0$$
$$-l_{10} + a_0 b_0$$

Specification $\mathcal{S}_n \in \mathbb{Z}[X]$.

$$8s_3 + 4s_2 + 2s_1 + s_0 - 4l_{22} - 2l_{14} - 2l_{12} - l_{10}$$
$$4s_2 + 2s_1 + s_0 + 8l_{24} - 4l_{22} - 2l_{14} - 2l_{12} - l_{10}$$
$$4l_{28} + 8l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12}$$



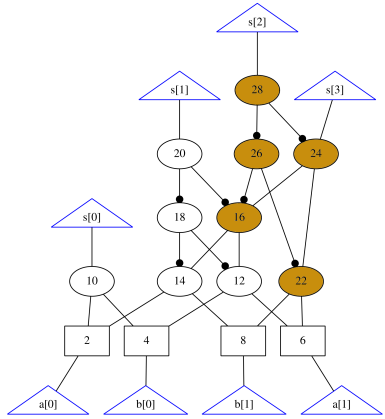
Local Linearization

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

$$
\begin{aligned}
&-s_3 + l_{24} && -l_{22} + a_1 b_1 \\
&-s_2 + l_{28} && -l_{20} + l_{18} l_{16} - l_{18} - l_{16} + 1 \\
&-s_1 + l_{20} && -l_{18} + l_{14} l_{12} - l_{14} - l_{12} + 1 \\
&-s_0 + l_{10} && -l_{16} + l_{14} l_{12} \\
&-l_{28} - l_{26} - l_{24} + 1 && -l_{14} + a_0 b_1 \\
&-l_{26} + l_{24} - l_{22} - l_{16} + 1 && -l_{12} + a_1 b_0 \\
&-l_{24} + l_{22} l_{16} && -l_{10} + a_0 b_0
\end{aligned}
$$

Specification $\mathcal{S}_n \in \mathbb{Z}[X]$.

$$
\begin{aligned}
8s_3 + 4s_2 + 2s_1 + s_0 - 4l_{22} - 2l_{14} - 2l_{12} - l_{10} \\
4s_2 + 2s_1 + s_0 + 8l_{24} - 4l_{22} - 2l_{14} - 2l_{12} - l_{10} \\
4l_{28} + 8l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12}
\end{aligned}
$$



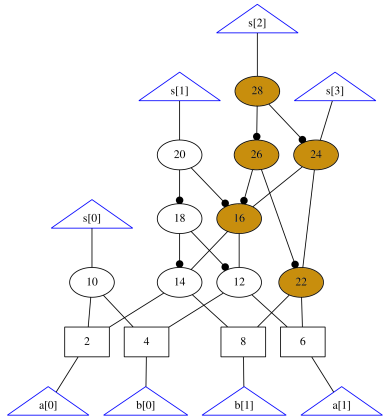
Local Linearization

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

$$
\begin{aligned}
&-s_3 + l_{24} && -l_{22} + a_1 b_1 \\
&-s_2 + l_{28} && -l_{20} + l_{18} l_{16} - l_{18} - l_{16} + 1 \\
&-s_1 + l_{20} && -l_{18} + l_{14} l_{12} - l_{14} - l_{12} + 1 \\
&-s_0 + l_{10} && -l_{16} + l_{14} l_{12} \\
&-l_{28} - l_{26} - l_{24} + 1 && -l_{14} + a_0 b_1 \\
&-l_{26} + l_{24} - l_{22} - l_{16} + 1 && -l_{12} + a_1 b_0 \\
&-l_{24} + l_{22} l_{16} && -l_{10} + a_0 b_0
\end{aligned}
$$

Specification $\mathcal{S}_n \in \mathbb{Z}[X]$.

$$
\begin{aligned}
8s_3 + 4s_2 + 2s_1 + s_0 - 4l_{22} - 2l_{14} - 2l_{12} - l_{10} \\
4s_2 + 2s_1 + s_0 + 8l_{24} - 4l_{22} - 2l_{14} - 2l_{12} - l_{10} \\
4l_{28} + 8l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12} \\
- 4l_{26} + 4l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12} + 4
\end{aligned}
$$



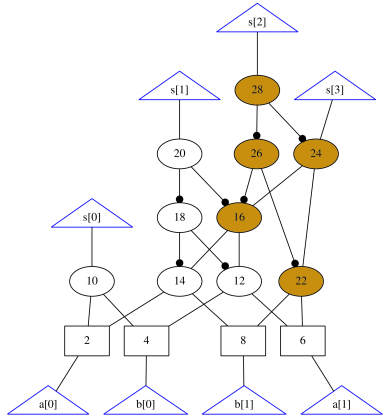
Local Linearization

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

$$
\begin{aligned}
&-s_3 + l_{24} && -l_{22} + a_1 b_1 \\
&-s_2 + l_{28} && -l_{20} + l_{18} l_{16} - l_{18} - l_{16} + 1 \\
&-s_1 + l_{20} && -l_{18} + l_{14} l_{12} - l_{14} - l_{12} + 1 \\
&-s_0 + l_{10} && -l_{16} + l_{14} l_{12} \\
&-l_{28} - l_{26} - l_{24} + 1 && -l_{14} + a_0 b_1 \\
&-l_{26} + l_{24} - l_{22} - l_{16} + 1 && -l_{12} + a_1 b_0 \\
&-l_{24} + l_{22} l_{16} && -l_{10} + a_0 b_0
\end{aligned}
$$

Specification $\mathcal{S}_n \in \mathbb{Z}[X]$.

$$
\begin{aligned}
8s_3 + 4s_2 + 2s_1 + s_0 - 4l_{22} - 2l_{14} - 2l_{12} - l_{10} \\
4s_2 + 2s_1 + s_0 + 8l_{24} - 4l_{22} - 2l_{14} - 2l_{12} - l_{10} \\
4l_{28} + 8l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12} \\
-4l_{26} + 4l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12} + 4 \\
2l_{20} + 4l_{16} - 2l_{14} - 2l_{12}
\end{aligned}
$$



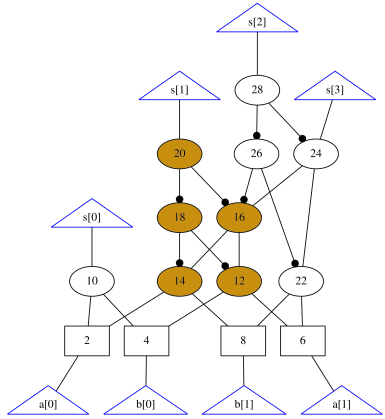
Local Linearization

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

$$
\begin{aligned}
&-s_3 + l_{24} && -l_{22} + a_1 b_1 \\
&-s_2 + l_{28} && -l_{20} + l_{18} l_{16} - l_{18} - l_{16} + 1 \\
&-s_1 + l_{20} && -l_{18} + l_{14} l_{12} - l_{14} - l_{12} + 1 \\
&-s_0 + l_{10} && -l_{16} + l_{14} l_{12} \\
&-l_{28} - l_{26} - l_{24} + 1 && -l_{14} + a_0 b_1 \\
&-l_{26} + l_{24} - l_{22} - l_{16} + 1 && -l_{12} + a_1 b_0 \\
&-l_{24} + l_{22} l_{16} && -l_{10} + a_0 b_0
\end{aligned}
$$

Specification $\mathcal{S}_n \in \mathbb{Z}[X]$.

$$
\begin{aligned}
8s_3 + 4s_2 + 2s_1 + s_0 - 4l_{22} - 2l_{14} - 2l_{12} - l_{10} \\
4s_2 + 2s_1 + s_0 + 8l_{24} - 4l_{22} - 2l_{14} - 2l_{12} - l_{10} \\
4l_{28} + 8l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12} \\
-4l_{26} + 4l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12} + 4 \\
2l_{20} + 4l_{16} - 2l_{14} - 2l_{12}
\end{aligned}
$$



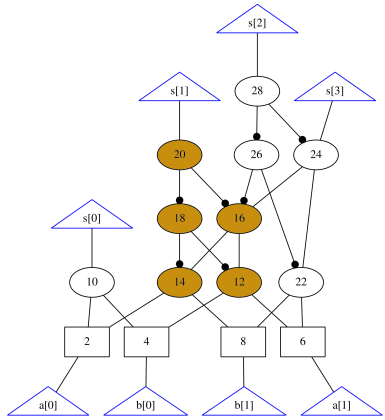
Local Linearization

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

$$
\begin{aligned}
&-s_3 + l_{24} && -l_{22} + a_1 b_1 \\
&-s_2 + l_{28} && -l_{20} - l_{18} - l_{16} + 1 \\
&-s_1 + l_{20} && -l_{18} + l_{16} - l_{14} - l_{12} + 1 \\
&-s_0 + l_{10} && -l_{16} + l_{14} l_{12} \\
&-l_{28} - l_{26} - l_{24} + 1 && -l_{14} + a_0 b_1 \\
&-l_{26} + l_{24} - l_{22} - l_{16} + 1 && -l_{12} + a_1 b_0 \\
&-l_{24} + l_{22} l_{16} && -l_{10} + a_0 b_0
\end{aligned}
$$

Specification $\mathcal{S}_n \in \mathbb{Z}[X]$.

$$
\begin{aligned}
8s_3 + 4s_2 + 2s_1 + s_0 - 4l_{22} - 2l_{14} - 2l_{12} - l_{10} \\
4s_2 + 2s_1 + s_0 + 8l_{24} - 4l_{22} - 2l_{14} - 2l_{12} - l_{10} \\
4l_{28} + 8l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12} \\
- 4l_{26} + 4l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12} + 4 \\
2l_{20} + 4l_{16} - 2l_{14} - 2l_{12} \\
- 2l_{18} + 2l_{16} - 2l_{14} - 2l_{12} + 2
\end{aligned}
$$



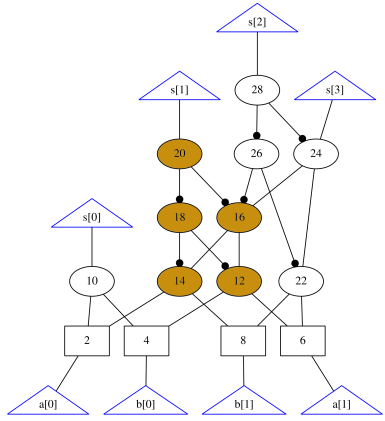
Local Linearization

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

$$
\begin{aligned}
&-s_3 + l_{24} && -l_{22} + a_1 b_1\\
&-s_2 + l_{28} && -l_{20} - l_{18} - l_{16} + 1\\
&-s_1 + l_{20} && -l_{18} + l_{16} - l_{14} - l_{12} + 1\\
&-s_0 + l_{10} && -l_{16} + l_{14} l_{12}\\
&-l_{28} - l_{26} - l_{24} + 1 && -l_{14} + a_0 b_1\\
&-l_{26} + l_{24} - l_{22} - l_{16} + 1 && -l_{12} + a_1 b_0\\
&-l_{24} + l_{22} l_{16} && -l_{10} + a_0 b_0
\end{aligned}
$$

Specification $\mathcal{S}_n \in \mathbb{Z}[X]$.

$$
\begin{aligned}
8s_3 + 4s_2 + 2s_1 + s_0 - 4l_{22} - 2l_{14} - 2l_{12} - l_{10}\\
4s_2 + 2s_1 + s_0 + 8l_{24} - 4l_{22} - 2l_{14} - 2l_{12} - l_{10}\\
4l_{28} + 8l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12}\\
- 4l_{26} + 4l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12} + 4\\
2l_{20} + 4l_{16} - 2l_{14} - 2l_{12}\\
- 2l_{18} + 2l_{16} - 2l_{14} - 2l_{12} + 2\\
0
\end{aligned}
$$

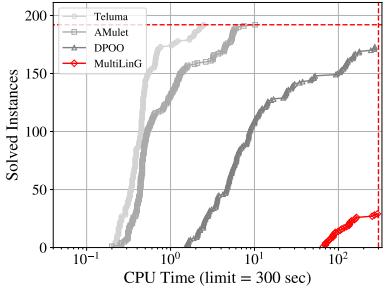


First Approach

(Kaufmann, Berthomieu TACAS'25)

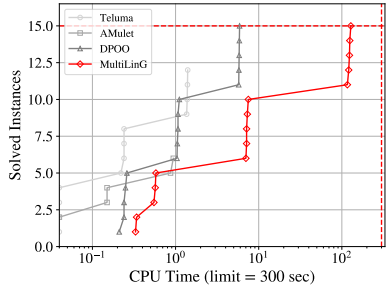
Unoptimized multipliers

- 192 architectures
- 64 bit inputs



Optimized multipliers

- 5 optimization scripts
- for 32, 64, 128 bit inputs

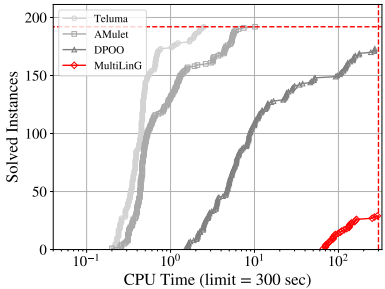


First Approach

(Kaufmann, Berthomieu TACAS'25)

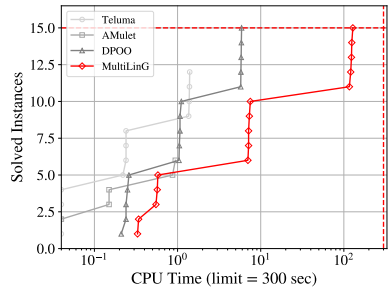
Unoptimized multipliers

- 192 architectures
- 64 bit inputs



Optimized multipliers

- 5 optimization scripts
- for 32, 64, 128 bit inputs



- 👍 Approach seems to be more robust against optimization
- 👎 Black-box GB computation is too costly (can handle ≤ 8 vars)
- 👎 Completely ignores existing structure

Two Targeted Approaches

(Hofstadler, Kaufmann, CP'25)

FGLM-style Linearization

Guess & Prove Linearization

Two Targeted Approaches

(Hofstadler, Kaufmann, CP'25)

FGLM-style Linearization

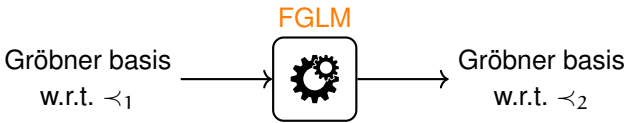
Guess & Prove Linearization

FGLM-style Linearization

- Exploit that we already have ($\prec_{\text{lex}}$-)Gröbner basis.

FGLM-style Linearization

Exploit that we already have ($\prec_{\text{lex}}$-)Gröbner basis.

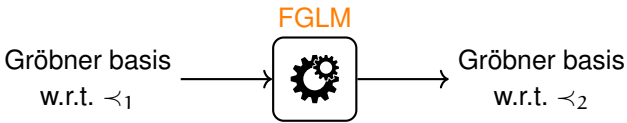


One can adapt FGLM to compute all linear polynomials and then stop.

Exploit existing structure $\rightsquigarrow$ a lot faster than GB computation

FGLM-style Linearization

- Exploit that we already have ($\prec_{\text{lex}}$-)Gröbner basis.



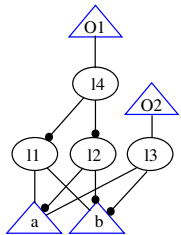
One can adapt FGLM to compute all linear polynomials and then stop.

- Exploit existing structure $\rightsquigarrow$ a lot faster than GB computation
- Runtime exponential in number of variables (can handle ≤ 15 vars)

FGLM-style Linearization

Ideal $I \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3, l_4]$ **generated by:**

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$



FGLM-style Linearization

Ideal $I \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3, l_4]$ generated by:

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

1. Compute Normalforms:

$$l_1 = ab, \quad l_2 = ab - a - b + 1, \quad l_3 = -ab + a, \quad l_4 = -2ab + a + b$$

FGLM-style Linearization

Ideal $I \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3, l_4]$ generated by:

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

1. Compute Normalforms:

$$l_1 = ab, \quad l_2 = ab - a - b + 1, \quad l_3 = -ab + a, \quad l_4 = -2ab + a + b$$

2. Build Coefficient Matrix A:

$$\begin{array}{ccccccc} \mathrm{NF}_G(1) & \mathrm{NF}_G(a) & \mathrm{NF}_G(b) & \mathrm{NF}_G(l_1) & \mathrm{NF}_G(l_2) & \mathrm{NF}_G(l_3) & \mathrm{NF}_G(l_4) \\ \left(\begin{array}{ccccccc} 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & -1 & -2 \end{array}\right) & \begin{array}{c} 1 \\ a \\ b \\ ab \end{array} \end{array}$$

FGLM-style Linearization

Ideal $I \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3, l_4]$ generated by:

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

3. Compute Kernel of A:

$$\begin{matrix} 1 & a & b & l_1 & l_2 & l_3 & l_4 \end{matrix}$$
$$\begin{pmatrix} 0 & -1 & -1 & 2 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 & 1 & 0 \\ -1 & 1 & 1 & -1 & 1 & 0 & 0 \end{pmatrix}$$

FGLM-style Linearization

Ideal $I \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3, l_4]$ generated by:

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

3. Compute Kernel of A:

$$\begin{array}{c} \begin{array}{ccccccc} 1 & a & b & l_1 & l_2 & l_3 & l_4 \end{array} \\ \begin{pmatrix} 0 & -1 & -1 & 2 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 & 1 & 0 \\ -1 & 1 & 1 & -1 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

4. Read off Linear Polynomials:

$$l_4 + 2l_1 - a - b, \qquad l_3 + l_1 - a, \qquad l_2 - l_1 + a + b - 1$$

Two Targeted Approaches

(Hofstadler, Kaufmann, CP'25)

FGLM-style Linearization

Guess & Prove Linearization

Two Targeted Approaches

(Hofstadler, Kaufmann, CP'25)

FGLM-style Linearization

Guess & Prove Linearization

Guess & Prove Linearization

- Exploit that we can compute (a lot of) points in the variety $\mathrm{V}(\mathrm{G})$

Guess & Prove Linearization

- Exploit that we can compute (a lot of) points in the variety $V(G)$

  - ✓ our ideals are radical
  - ✓ our ideals are zero-dimensional
  - ✓ our ideals don't have additional roots in algebraic closure
  - ✓ we can efficiently test ideal membership of linear polynomials

Guess & Prove Linearization

Given: $G \subseteq \mathbb{K}[X]$ satisfying the assumptions from before

Compute: $\mathbb{K}$-VS basis of $\{ f \in \langle G \rangle \mid \deg(f) \leq 1 \}$

Guess & Prove Linearization

Given: $G \subseteq \mathbb{K}[X]$ satisfying the assumptions from before

Compute: $\mathbb{K}$-VS basis of $\{ f \in \langle G \rangle \mid \deg(f) \leq 1 \}$

Algorithm

1. Sample points $P \subseteq V(G)$
2. Compute all linear polynomials F that vanish on P
3. **if** $F \subseteq \langle G \rangle$
4. **return** F
5. **else** add witness points to P and repeat

Guess & Prove Linearization

Given: $G \subseteq \mathbb{K}[X]$ satisfying the assumptions from before

Compute: $\mathbb{K}$-VS basis of $\{ f \in \langle G \rangle \mid \deg(f) \leq 1 \}$

Algorithm

1. Sample points $P \subseteq V(G)$ "guess"
2. Compute all linear polynomials F that vanish on P
3. **if** $F \subseteq \langle G \rangle$
4. **return** F
5. **else** add witness points to P and repeat

Guess & Prove Linearization

Given: $G \subseteq \mathbb{K}[X]$ satisfying the assumptions from before

Compute: $\mathbb{K}$-VS basis of $\{ f \in \langle G \rangle \mid \deg(f) \leq 1 \}$

Algorithm

1. Sample points $P \subseteq V(G)$ "guess"
2. Compute all linear polynomials F that vanish on P
3. **if** $F \subseteq \langle G \rangle$ "prove"
4. **return** F
5. **else** add witness points to P and repeat

Guess & Prove Linearization

Given: $G \subseteq \mathbb{K}[X]$ satisfying the assumptions from before

Compute: $\mathbb{K}$-VS basis of $\{ f \in \langle G \rangle \mid \deg(f) \leq 1 \}$

Algorithm

1. Sample points $P \subseteq V(G)$ "guess"
2. Compute all linear polynomials F that vanish on P
3. **if** $F \subseteq \langle G \rangle$ "prove"
4. **return** F "repair"
5. **else** add witness points to P and repeat

Guess & Prove Linearization

Let $G \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3, l_4]$ consist of:

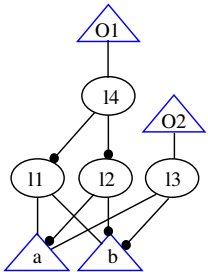
$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

Guess & Prove Linearization

Let $G \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3, l_4]$ **consist of:**

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

Sample: $(0, 0, 0, 1, 0, 0)$, $(1, 0, 0, 0, 1, 1)$, $(0, 1, 0, 0, 0, 1) \in V(G)$



Guess & Prove Linearization

Let $G \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3, l_4]$ **consist of:**

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

Sample: $(0,0,0,1,0,0),\ (1,0,0,0,1,1),\ (0,1,0,0,0,1) \in V(G)$

Matrix A**:**

$$\begin{array}{c} \begin{matrix} 1 & a & b & l_1 & l_2 & l_3 & l_4 \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Guess & Prove Linearization

Let $G \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3, l_4]$ **consist of:**

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

Kernel A**:**

$$\begin{array}{c} \begin{array}{ccccccc} 1 & a & b & l_1 & l_2 & l_3 & l_4 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & -1 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Guess & Prove Linearization

Let $G \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3, l_4]$ consist of:

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

Kernel A:

$$\begin{array}{c} \begin{array}{ccccccc} 1 & a & b & l_1 & l_2 & l_3 & l_4 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & -1 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Kernel yields Guesses:

$$l_4 - b - a \qquad l_3 - a \qquad l_2 + a + b - 1 \qquad l_1$$

Guess & Prove Linearization

Let $G \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3]$ consist of

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

Guess: $l_4 - b - a$ $l_3 - a$ $l_2 + a + b - 1$ l_1

Guess & Prove Linearization

Let $G \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3]$ consist of

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

Guess: $l_4 - b - a$ $l_3 - a$ $l_2 + a + b - 1$ l_1

Prove:

👎 Algebraic proof (reduction by G) does not scale!

Guess & Prove Linearization

Let $G \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3]$ consist of

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

Guess: $l_4 - b - a$ $l_3 - a$ $l_2 + a + b - 1$ l_1

Prove:

- 👎 Algebraic proof (reduction by G) does not scale!
- 💡 Use a SAT solver!

Guess & Prove Linearization

Let $G \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3]$ consist of

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

Guess: $l_4 - b - a$ $l_3 - a$ $l_2 + a + b - 1$ l_1

Prove:

- Algebraic proof (reduction by G) does not scale!
- Use a SAT solver!

Encode $\bigwedge_{g \in G} (g = 0) \wedge (\text{guess} \neq 0)$ as propositional formula and give to a SAT solver.

Guess & Prove Linearization

Let $G \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3]$ consist of

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

Guess: $l_4 - b - a$ $l_3 - a$ $l_2 + a + b - 1$ l_1

Prove:

- Algebraic proof (reduction by G) does not scale!
- Use a SAT solver!

Encode $\bigwedge_{g \in G} (g = 0) \wedge (\text{guess} \neq 0)$ as propositional formula and give to a SAT solver.

If the result is

UNSAT, then guess $\in \langle G \rangle$

SAT, then we get $p \in V(G) : \text{guess}(p) \neq 0$

Guess & Prove Linearization

Let $G \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3]$ consist of

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

Guess: $l_4 - b - a$ $l_3 - a$ $l_2 + a + b - 1$ l_1

Prove:

- Algebraic proof (reduction by G) does not scale!
- Use a SAT solver!

Encode $\bigwedge_{g \in G} (g = 0) \wedge (\text{guess} \neq 0)$ as propositional formula and give to a SAT solver.

If the result is

UNSAT, then guess $\in \langle G \rangle$

SAT, then we get $p \in V(G) : \text{guess}(p) \neq 0$

In this case, SAT solver returns SAT and $p = (1, 1, 1, 0, 0, 0)$.

Guess & Prove Linearization

Let $G \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3]$ consist of

$$l_1 - ab \qquad l_2 - (1-a)(1-b) \qquad l_3 - a(1-b) \qquad a^2 - a \qquad b^2 - b$$

Repair Samples $(0,0,0,1,0,0),\ (1,0,0,0,1,1),\ (0,1,0,0,0,1), (1,1,1,0,0,0) \in V(G)$

Matrix A:

$$\begin{array}{c} \begin{matrix} 1 & a & b & l_1 & l_2 & l_3 & l_4 \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Guess & Prove Linearization

Let $G \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3]$ consist of

$$l_1 - ab \qquad l_2 - (1-a)(1-b) \qquad l_3 - a(1-b) \qquad a^2 - a \qquad b^2 - b$$

Repair Samples $(0,0,0,1,0,0),\ (1,0,0,0,1,1),\ (0,1,0,0,0,1), (1,1,1,0,0,0) \in V(G)$

Kernel A:

$$\begin{array}{c} \begin{matrix} 1 & a & b & l_1 & l_2 & l_3 & l_4 \end{matrix} \\ \begin{pmatrix} 0 & -1 & -1 & 2 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 & 1 & 0 \\ -1 & 1 & 1 & -1 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Kernel yields:

$$l_4 + 2l_1 - a - b, \qquad l_3 + l_1 - a, \qquad l_2 - l_1 + a + b - 1$$

Guess & Prove Linearization

Works really well also for larger circuits (can handle ≥ 200 variables)!

Guess & Prove Linearization

Works really well also for larger circuits (can handle ≥ 200 variables)!

One might wonder...

...why not interpolate on all of $V(G)$?

Guess & Prove Linearization

Works really well also for larger circuits (can handle ≥ 200 variables)!

One might wonder...

...why not interpolate on all of $V(G)$? $|V(G)| \approx 2^{80} \approx$ stars in universe

Guess & Prove Linearization

Works really well also for larger circuits (can handle ≥ 200 variables)!

One might wonder...

...why not interpolate on all of $V(G)$? $|V(G)| \approx 2^{80} \approx$ stars in universe

...how many samples do we need?

Guess & Prove Linearization

Works really well also for larger circuits (can handle ≥ 200 variables)!

One might wonder...

...why not interpolate on all of $V(G)$? $|V(G)| \approx 2^{80} \approx$ stars in universe

...how many samples do we need? $3\times$ subcircuit size

Guess & Prove Linearization

Works really well also for larger circuits (can handle ≥ 200 variables)!

One might wonder...

...why not interpolate on all of $V(G)$? $|V(G)| \approx 2^{80} \approx$ stars in universe

...how many samples do we need? $3\times$ subcircuit size

...how do we find good samples?

Guess & Prove Linearization

Works really well also for larger circuits (can handle ≥ 200 variables)!

One might wonder...

...why not interpolate on all of $V(G)$? $|V(G)| \approx 2^{80} \approx$ stars in universe

...how many samples do we need? $3\times$ subcircuit size

...how do we find good samples? Use weighted uniform-like sampling (`CMSGen`)

Guess & Prove Linearization

Works really well also for larger circuits (can handle ≥ 200 variables)!

One might wonder...

...why not interpolate on all of $V(G)$? $|V(G)| \approx 2^{80} \approx$ stars in universe

...how many samples do we need? $3\times$ subcircuit size

...how do we find good samples? Use weighted uniform-like sampling (`CMSGen`)

...how many iterations of the repair loop do we need?

Guess & Prove Linearization

Works really well also for larger circuits (can handle ≥ 200 variables)!

One might wonder...

...why not interpolate on all of $V(G)$? $|V(G)| \approx 2^{80} \approx$ stars in universe

...how many samples do we need? $3\times$ subcircuit size

...how do we find good samples? Use weighted uniform-like sampling (`CMSGen`)

...how many iterations of the repair loop do we need? ≤ 3

Verification via Linearization

Input: Circuit C, specification polynomial S
Output: Determine whether C fulfils S

1 $G \leftarrow G(C) \cup B(C)$
2 $S_{\text{lin}}, G \leftarrow$ Linearize-Spec(S, G)
3 $L \leftarrow$ LINEAR-POLIES-OF-$\prec_{\text{dlex}}$-GB(G)
4 $S_{\text{lin}} \leftarrow$ Linear-Reduce(S_{lin}, L)
5 **return** $S_{\text{lin}} \stackrel{?}{=} 0$

compute $\prec_{\text{dlex}}$-GB of $G' \subseteq G$

Verification via Linearization

Input: Circuit C, specification polynomial S
Output: Determine whether C fulfils S

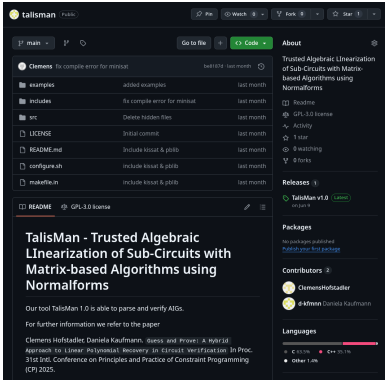
1 $G \leftarrow G(C) \cup B(C)$
2 $S_{\text{lin}}, G \leftarrow$ Linearize-Spec(S, G)
3 $L \leftarrow$ LINEAR-POLIES-OF-$\prec_{\text{dlex}}$-GB(G)
4 $S_{\text{lin}} \leftarrow$ Linear-Reduce(S_{lin}, L)
5 **return** $S_{\text{lin}} \stackrel{?}{=} 0$

pick $G' \subseteq G$
if $|G'|$ is small use FGLM
else use Guess & Prove

TALISMAN

https://github.com/d-kfmnn/talisman/

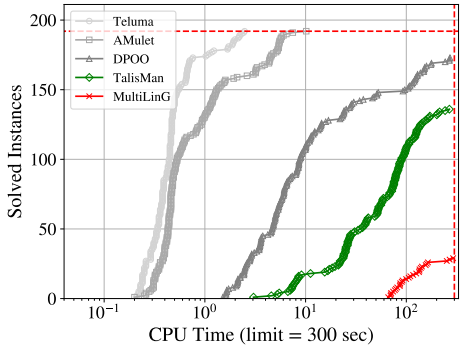
- Uses FLINT for matrix computations
- Guess and Prove Algorithm:
  - Uses PBLIB to translate guessed polynomial to CNF
  - Uses KISSAT as a SAT solver
  - Uses caching of isomorphic subcircuits



Evaluation – Multiplier Circuits

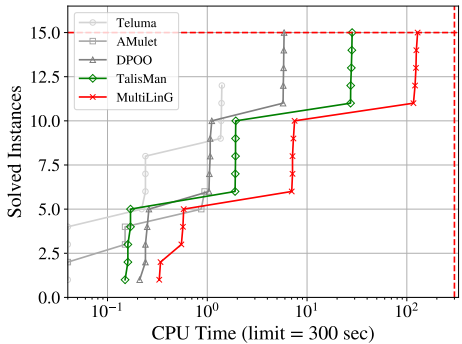
Unoptimized multipliers

- 192 architectures
- 64 bit inputs



Optimized multipliers

- 5 optimization scripts
- for 32, 64, 128 bit inputs



Big Coefficients

$$S \rightsquigarrow \cdots \rightsquigarrow \sum_{i=1}^{D} c_i \cdot m_i \rightsquigarrow \cdots \rightsquigarrow 0$$

Big Coefficients

$$S \rightsquigarrow \cdots \rightsquigarrow \sum_{i=1}^{D} c_i \cdot m_i \rightsquigarrow \cdots \rightsquigarrow 0$$

Number of terms

D

Can grow exponentially

Extensive prior work:

Encodings, Preprocessing, Dynamic division order, SAT/BDD-reasoning

Big Coefficients

$$S \rightsquigarrow \cdots \rightsquigarrow \sum_{i=1}^{D} c_i \cdot m_i \rightsquigarrow \cdots \rightsquigarrow 0$$

Number of terms

D

Can grow exponentially

Extensive prior work:

Encodings, Preprocessing, Dynamic division order, SAT/BDD-reasoning

Coefficients

c_i

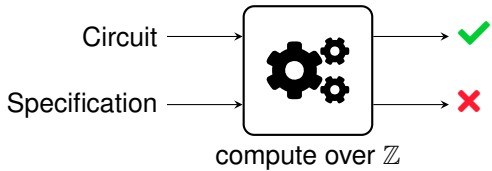
Values up to $\geq 2^{127}$

👎 Arbitrary precision arithmetic
👎 CNF Translation explodes

Largely ignored so far!

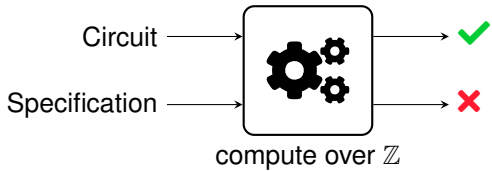
Big Coefficients

Classical Verification



Big Coefficients

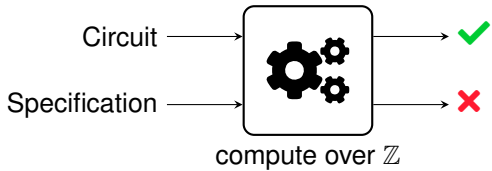
Classical Verification



Specification for a 64-bit multiplier: $\left(\sum_{i=0}^{63} 2^i a_i\right) \cdot \left(\sum_{i=0}^{63} 2^i b_i\right) - \sum_{i=0}^{127} 2^i s_i$

Big Coefficients

Classical Verification

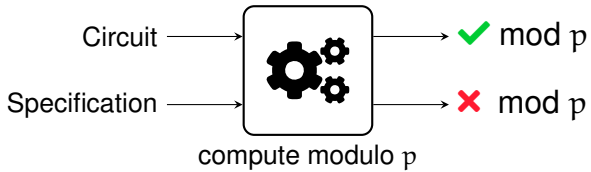


Specification for a 64-bit multiplier: $\left(\sum_{i=0}^{63} 2^i a_i\right) \cdot \left(\sum_{i=0}^{63} 2^i b_i\right) - \sum_{i=0}^{127} 2^i s_i$

Observation: Spec can only take values between -2^{128} and 2^{128}.

Big Coefficients

Modular Verification (Kaufmann, Biere, Kauers FMCAD'19)

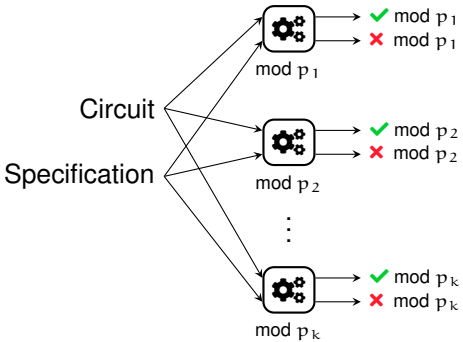


Specification for a 64-bit multiplier: $\left(\sum_{i=0}^{63} 2^i a_i\right) \cdot \left(\sum_{i=0}^{63} 2^i b_i\right) - \sum_{i=0}^{127} 2^i s_i$

Observation: Spec can only take values between -2^{128} and 2^{128}.

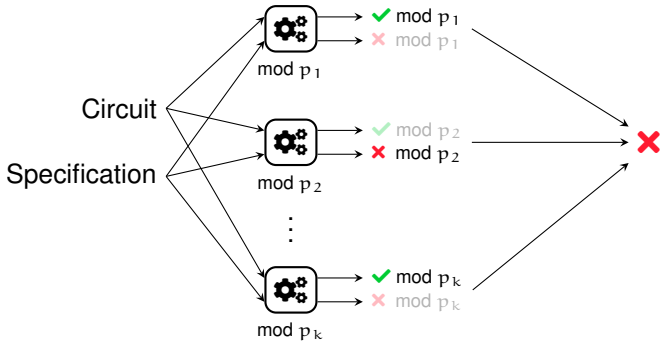
Multimodular Verification

[HofstadlerKaufmannChen IJCAR'26]



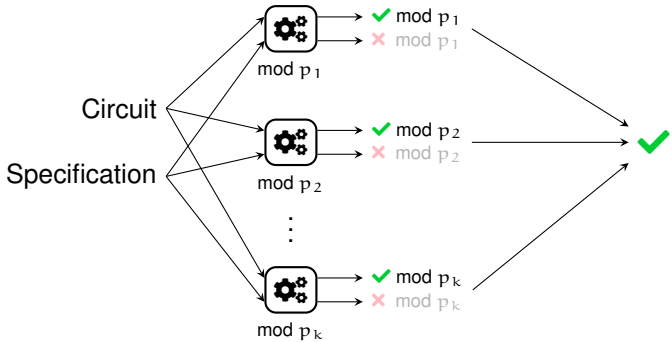
Multimodular Verification

[HofstadlerKaufmannChen IJCAR'26]



Multimodular Verification

[HofstadlerKaufmannChen IJCAR'26]

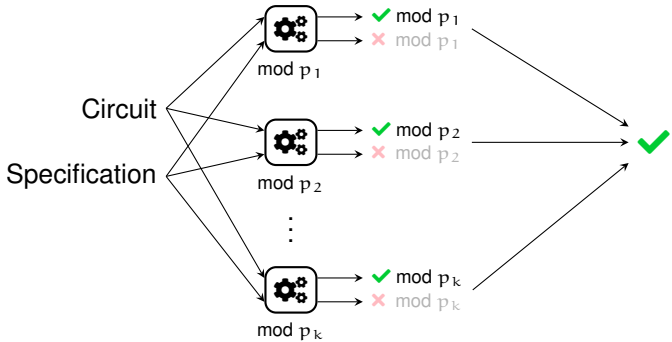


Theorem (Hofstadler, Kaufmann, Chen '26)

If $p_1 p_2 \cdots p_k \geq 2^{128}$, this is sound and complete!

Multimodular Verification

[HofstadlerKaufmannChen IJCAR'26]

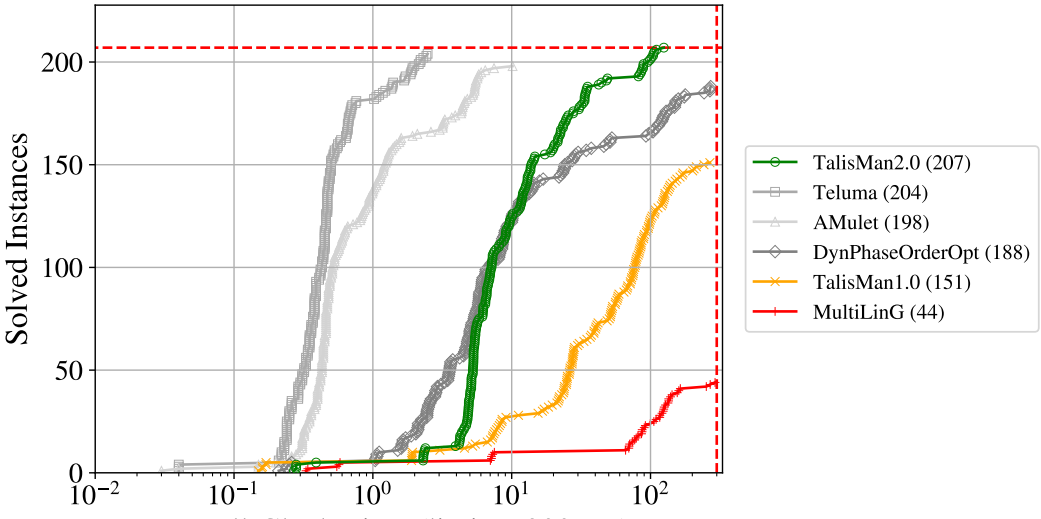


Theorem (Hofstadler, Kaufmann, Chen '26)

If $p_1 p_2 \cdots p_k \geq B$, this is sound and complete!

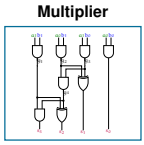
Multimodular verification implemented in C++ tool TalisMan2.0.

207 optimized and unoptimized multipliers for 32/64/128 bit inputs.



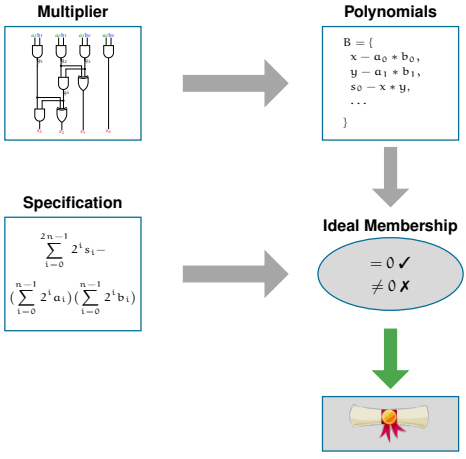
Is the circuit really really correct?

Proof Logging



Problem: Verification might not be error free

Proof Logging

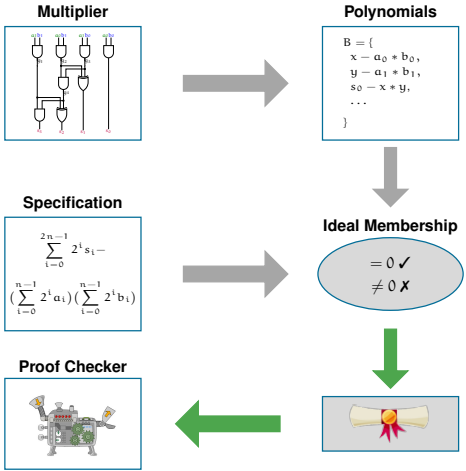


Problem: Verification might not be error free

Goal: Validate result of verification process

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

Proof Logging



Problem: Verification might not be error free

Goal: Validate result of verification process

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

How to derive a certificate for $f \xrightarrow{\{f_1,\ldots,f_s\}} r$?

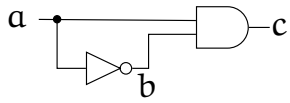
Nullstellensatz Proofs

Input: $f_1, \ldots, f_s, f$
Output: $a_1, \ldots, a_s, r$
$a_1 := 0; \ldots; a_s := 0; r := 0$
$p := f$
WHILE $p \neq 0$ DO
    $i := 1$
    divisionoccurred := false
    WHILE $i \leq s$ AND divisionoccurred = false DO
        IF LT(f_i) divides (p) THEN
            $a_i := a_i + \mathrm{LT}(p)/\mathrm{LT}(f_i)$
            $p := p - (\mathrm{LT}(p)/\mathrm{LT}(f_i))f_i$
            divisionoccurred:= true
        ELSE
            $i := i + 1$
    IF divisionoccurred = false THEN
        $r := r + \mathrm{LT}(p)$
        $p := p - \mathrm{LT}(p)$

- Provides list of co-factors $a_1, \ldots, a_s$.
- Correctness is checked by expanding linear combination $f = \sum a_i f_i$.
- Condensed proof format.
- Not ideal for debugging.

Beame, P., Impagliazzo, R., Krajicek, J., Pitassi, T., Pudlák, P.: Lower Bounds on Hilbert's Nullstellensatz and Propositional Proofs. In: Proc. London Math. Society. vol. s3-73, pp. 1-26 (1996)

Example



$$
\begin{aligned}
G &= \{ \quad -b + 1 - a, && b = \neg a \\
&\qquad -c + ab, && c = a \wedge b = a \wedge \neg a \\
&\qquad a^2 - a, \} && a \in \mathbb{B} \\
f &= c
\end{aligned}
$$

$$\mathrm{red}_G(f) = 0$$

$$f = c = a(-b + 1 - a) - 1(-c + ab) + 1(a^2 - a)$$

$$P = (a, -1, 1)$$

Polynomial Calculus*

Let $G \subseteq R[X]$ and $f \in R[X]$.

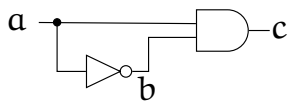
Proof: Sequence $P = (p_1, \ldots p_n)$, where each p_k is obtained by one of the two rules:

Addition	$\dfrac{p_i \qquad p_j}{p_i + p_j}$	p_i, p_j appearing earlier in the proof or are contained in G
Multiplication	$\dfrac{p_i}{q p_i}$	p_i appearing earlier in the proof or is contained in G and $q \in R[X]$ being arbitrary

If $p_n = f$ we have $f \in \langle G \rangle$.

Clegg, M., Edmonds, J., Impagliazzo, R.: Using the Groebner basis algorithm to find proofs of unsatisfiability. In: STOC. pp. 174–183. ACM (1996)

Example



$$
\begin{aligned}
G &= \{ && -b + 1 - a, && b = \neg a \\
&&& -c + ab, && c = a \wedge b = a \wedge \neg a \\
&&& a^2 - a, \} && a \in \mathbb{B} \\
f &= && c
\end{aligned}
$$

$$\mathrm{red}_G(f) = 0$$

$$
\begin{array}{lc}
* & \dfrac{-b + 1 - a}{} \\
+ & \dfrac{-ab + a - a^2 \qquad a^2 - a}{} \\
& + \quad \dfrac{-ab \qquad\qquad\qquad -c + ab}{} \\
& \qquad\qquad * \quad \dfrac{-c}{c}
\end{array}
$$

$$P = (-ab + a - a^2,\ -ab,\ -c,\ c)$$

Practical Algebraic Calculus

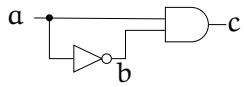
We translate the polynomial calculus into a more concrete proof format:

- For correctness it is important to know how the polynomials in the proof where derived
- Usually known $\rightarrow$ store this information

Practical Algebraic Calculus (PAC)

allows automated proof checking

Practical Algebraic Calculus




```
P =   -b+1-a,
      -c+a*b,
      -a^2+a
Target =  c
```

```
* : -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;
```

D. Ritirc, A. Biere, M. Kauers, A Practical Polynomial Calculus for Arithmetic Circuit Verification, SC2-Workshop, 2018.

Proof Checking

A proof **rule** contains four components:

$$\mathrm{o} : \nu, w, \ \mathrm{p};$$

Proof checking:

- Connection property: ν, w are given polynomials or conclusions $\mathrm{p_i}$ of previous rules
- Inference property: verify correctness of each rule, e.g. $\mathrm{p} = \nu + w$ for $\mathrm{o} = "+"$
- Target check: at least one $\mathrm{p_i}$ is equal to f.

Proof Checking Algorithm

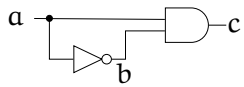
input G sequence of given polynomials
$r_1 \cdots r_k$ sequence of PAC proof rules

output "incorrect", "correct-proof", or "correct-refutation"

$P_0 \leftarrow G$
for $i \leftarrow 1 \ldots k$
  let $r_i = (o_i, v_i, w_i, p_i)$
  case $o_i = +$
    if $v_i \in P_{i-1} \ \wedge \ w_i \in P_{i-1} \ \wedge \ p_i = v_i + w_i$ **then** $P_i \leftarrow \text{append}(P_{i-1}, p_i)$
    else return "incorrect"
  case $o_i = *$
    if $v_i \in P_{i-1} \ \wedge \ p_i = v_i * w_i$ **then** $P_i \leftarrow \text{append}(P_{i-1}, p_i)$
    else return "incorrect"
for $i \leftarrow 1 \ldots k$
  if $p_i = f$ **then return** "target-checked"
return "correct-proof"

Boolean Variables

Handle Boolean-value constraints implicitly to reduce number of proof steps.



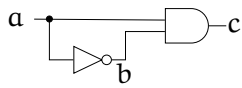

```
P =   -b+1-a,
      -c+a*b,
      -a^2+a
Spec =  c
```

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

D. Kaufmann, M. Fleury, A. Biere, The Proof Checkers Pacheck and Pastèque for the Practical Algebraic Calculus, FMCAD, 2020.

Indices

Introduce indices to reduce proof size.



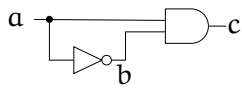

```
P =   1 -b+1-a;
      2 -c+a*b;
Spec = c
```

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

D. Kaufmann, M. Fleury, A. Biere, The Proof Checkers Pacheck and Pastèque for the Practical Algebraic Calculus, FMCAD, 2020.

Deletion Rule

Introduce a deletion rule to reduce the memory usage of the proof checker.



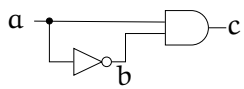

```
P =   1 -b+1-a;
      2 -c+a*b;
Spec = c
```

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

D. Kaufmann, M. Fleury, A. Biere, The Proof Checkers Pacheck and Pastèque for the Practical Algebraic Calculus, FMCAD, 2020.

LPAC

Switch from explicit addition and multiplication rules to linear combinations.



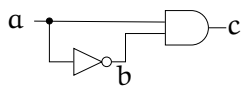

```
P =   1 -b+1-a;
      2 -c+a*b;
Spec =  c
```

LPAC simulates PAC

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

LPAC

Switch from explicit addition and multiplication rules to linear combinations.




```
P =   1 -b+1-a;
      2 -c+a*b;
Spec =  c
```

LPAC simulates PAC

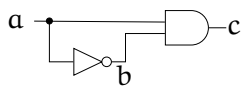
```
3   % 1*(a), -a*b;
1   d;
4   % 3*(1)+2*(1), -c;
2   d;
3   d;
5   % 4*(-1), c;
```

LPAC

```
3   % 1*(a)+2*(1), -c;
1   d;
2   d;
4   % 4*(-1), c;
```

LPAC

Switch from explicit addition and multiplication rules to linear combinations.




```
P =   1 -b+1-a;
      2 -c+a*b;
Spec =  c
```

LPAC simulates PAC

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

LPAC

```
3   % 1*(a)+2*(1), -c;
1   d;
2   d;
4   % 4*(-1), c;
```

LPAC simulates NSS

```
3   % 1*(-a)+2*(-1), -c;
```

D. Kaufmann, M. Fleury, A. Biere, M. Kauers, Practical Algebraic Calculus and Nullstellensatz with the Checkers Pacheck and Pastèque and Nuss-Checker FMSD, 2022.

Proof Checker

PACHECK

- Written in C++
- Verifies that the polynomial in `<target>` is contained in the ideal generated by the polynomials in `<input>` using the rules provided in `<proof>`.

Proof Checker

PACHECK

- Written in C++
- Verifies that the polynomial in `<target>` is contained in the ideal generated by the polynomials in `<input>` using the rules provided in `<proof>`.

PASTÈQUE

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

Evaluation: Circuit Verification

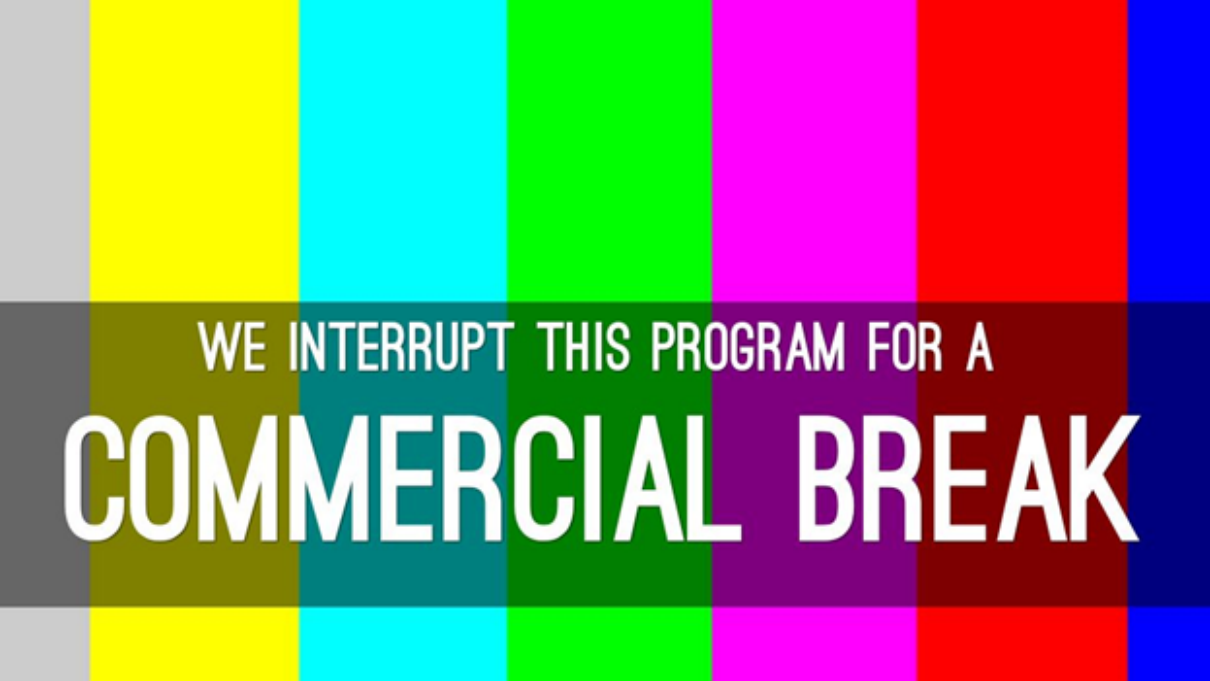
PACHECK

		LPAC simulates PAC			LPAC			LPAC simulates NSS		
multiplier	n	steps	sec	MB	steps	sec	MB	steps	sec	MB
btor	128	0.3	5	94	0.1	2	94	1	2	98
btor	256	1.3	26	367	0.3	8	367	1	7	385
btor	512	5.2	149	1468	1.0	37	1496	1	35	1555

PASTÈQUE

		LPAC simulates PAC			LPAC			LPAC simulates NSS		
multiplier	n	steps	sec	MB	steps	sec	MB	steps	sec	MB
btor	128	0.3	14	1305	0.1	7	1305	1	53	2044
btor	256	1.3	67	3467	0.3	37	3816	1	762	8819
btor	512	5.2	351	14651	1.0	238	16173	1	14347	41712

Conclusion


WE INTERRUPT THIS PROGRAM FOR A
COMMERCIAL BREAK

Funded by the European Union

European Research Council
Established by the European Commission

POLARIS

Advancing Polynomial and Logical Approaches for Trusted Automated Reasoning over Integrated Systems

SMT SOLVING
Satisfiability Modulo Theories at the core

COMPUTER ALGEBRA
Polynomial reasoning at word level

PROOF LOGGING
Machine-checkable evidence for every result

AUTOMATED REASONING
Powerful reasoning for complex systems

PSEUDO-BOOLEAN SOLVING
Efficient reasoning for arithmetic constraints

BIT-VECTOR SOLVING
Bit-precise reasoning for memory handling

FINITE FIELD REASONING
Polynomials over finite fields for cryptography

I am hiring PhD students and PostDocs

Join the **POLARIS** team and help us reach for the stars.

daniela.kaufmann@tuwien.ac.at

Conclusion

Algebraic circuit verification (GB ✅, Reduction ⚠️)

Conclusion

Algebraic circuit verification (GB ✅, Reduction ⚠️)

💡 Flip things around: $\prec_{\text{lex}} \rightsquigarrow \prec_{\text{dlex}}$ (GB ⚠️, Reduction ✅)

Conclusion

Algebraic circuit verification (GB ✅, Reduction ⚠️)

💡 Flip things around: $\prec_{\text{lex}} \rightsquigarrow \prec_{\text{dlex}}$ (GB ⚠️, Reduction ✅)

Three approaches

Compute GB

- no requirements
- full GB computation
- double exponential runtime
- small circuits (≤ 10 variables)

FGLM

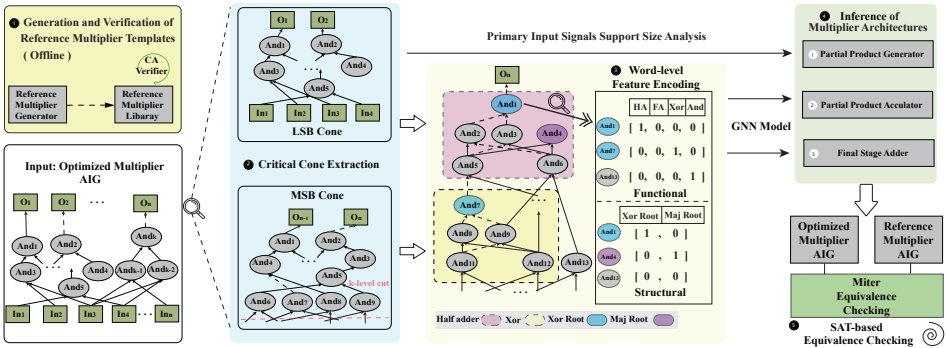
- requires a Gröbner basis
- first steps of FGLM
- exponential runtime
- moderate circuits (≤ 15 variables)

Guess & Prove

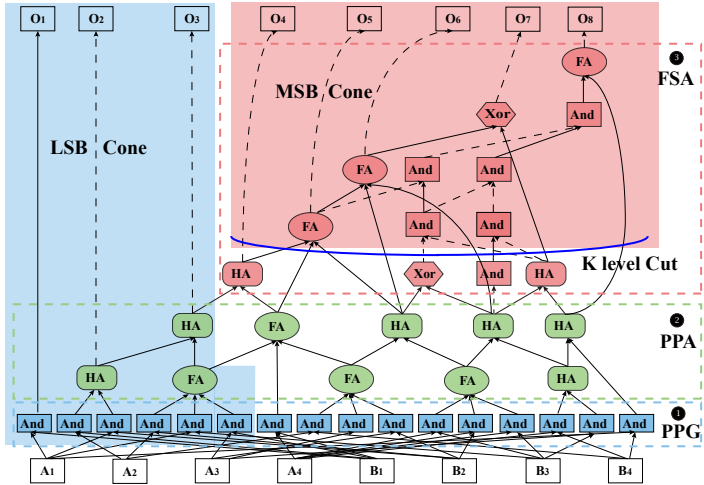
- requires variety
- interpolation (guess)
- uses SAT solver (prove)
- larger circuits (≥ 200 variables)

What about using GNNs?

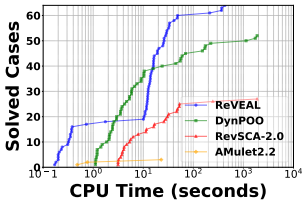
ReVEAL - GNN-Guided Reverse Engineering for Formal Verification of Optimized Multipliers [ChenKaufmannDengSongZhangYu TACAS'26]



What about using GNNs?



What about using GNNs?



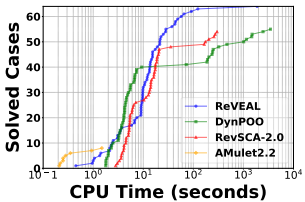


Table 3: Inference Accuracy for Each Stage

Stage	TOP 1*	TOP 2*	TOP 3*
Stage PPG	100%	–	–
Stage PPA	100%	–	–
Stage FSA	85%	97%	98%

* TOP 1, TOP 2, and TOP 3 represent the prediction accuracy when the ground truth is among the model's top 1, top 2, and top 3 ranked predictions, respectively.

Table 4: Solver Performance for Miter Solving

Opt	dc2			resyn3		
	Solved	Avg Time (s)	#. fastest	Solved	Avg Time (s)	#. fastest
cec	60/64	692.57	0	60/64	1007.20	1
&cec	64/64	8.50	46	64/64	6.83	50
&fraig -y	36/64	1848.53	12	39/64	1877.33	8
&fraig -x	37/64	1765.14	0	39/64	1726.71	5
kissat	64/64	96.79	6	64/64	186.55	0

References I

[Biere SATComp'16] A. Biere. Collection of Combinational Arithmetic Miters Submitted to the SAT Competition 2016. In SAT Competition 2016, pages 65–66, Dep. of Computer Science Report Series B, University of Helsinki, 2016.

[BiereFallerFazekasFleuryFroleyksPollitt SATComp'24] A. Biere, T. Faller, K. Fazekas, M. Fleury, N. Froleyks and F. Pollitt CaDiCaL, Gimsatul, IsaSAT and Kissat Entering the SAT Competition 2024. In SAT Competition 2024, pages 8–10, Dep. of Computer Science Report Series B, University of Helsinki, 2024.

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

[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.

[ChenBryant DAC'95] Y. Chen and R. Bryant. Verification of Arithmetic Circuits with Binary Moment Diagrams. In Proc. of DAC'95, pages 535–541, ACM, 1995.

References II

[ChenKaufmannDengSongZhangYu TACAS'26] C. Chen, D. Kaufmann, C. Deng, Z. Song, H. Zhang, and C. Yu. ReVEAL: GNN-Guided Reverse Engineering for Formal Verification of Optimized Multipliers. In Proc. of TACAS'26, Springer, 2026.

[DrechslerMahzoon ISEEIE'22] R. Drechsler and A. Mahzoon. Design Modification for Polynomial Formal Verification. In Proc. of ISEEIE'22, pages 187–194, IEEE, 2022.

[KaufmannBeameBiereNordström DATE'22] D. Kaufmann, P.Beame, A. Biere and J. Nordström. Adding Dual Variables to Algebraic Reasoning for Gate-Level Multiplier Verification. In Proc. of DATE'22, pages 1431–1436, IEEE, 2022.

[HofstadlerKaufmannChen IJCAR'26] C. Hofstadler, D. Kaufmann, and C. Chen. Avoiding Big Integers: Parallel Multimodular Algebraic Verification of Arithmetic Circuits. In Proc. of IJCAR'26.

[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 FMSD'20] D. Kaufmann, A. Biere and M. Kauers. Incremental Column-Wise Verification of Arithmetic Circuits Using Computer Algebra. In FMSD, vol 56, pages 22–54, 2020.

References III

[KaufmannFleuryBiere FMCAD'20] D. Kaufmann, M. Fleury and A. Biere. Pacheck and Pastèque Checking Practical Algebraic Calculus Proofs. In Proc. of FMCAD'20, pages 264–269, TU Vienna Academic Press, 2020.

[KaufmannFleuryBiereKauers FMSD'21] D. Kaufmann, M. Fleury, A. Biere and M. Kauers. Practical Algebraic Calculus and Nullstellensatz with the Checkers Pacheck and Pastèque and Nuss-Checker. In FMSD, online first, 2021.

[KonradScholl FMCAD'24] A. Konrad and C. Scholl. Practical Algebraic Calculus and Nullstellensatz with the Checkers Pacheck and Pastèque and Nuss-Checker. In FMSD, online first, 2021.

[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.

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

References IV

[MahzoonGroßeDrechsler ICCAD'18] A. Mahzoon, D. Große and R. Drechsler. PolyCleaner: Clean your Polynomials before Backward Rewriting to verify Million-gate Multipliers. In Proc. of ICCAD'18, pages 129:1–129:8, ACM, 2018.

[MahzoonGroßeSchollDrechsler DATE'20] A. Mahzoon, D. Große, Christoph Scholl and R. Drechsler. Towards Formal Verification of Optimized and Industrial Multipliers. In Proc. of DATE'20, pages 544–549, DATE, 2020.

[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 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.

[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.

References V

[SchollKonradMahzoonGroßeDrechsler DATE'21] C. Scholl, A. Konrad, A. Mahzoon, D. Große and R. Drechsler. Verifying Dividers Using Symbolic Computer Algebra and Don't Care Optimization. In Proc. of DATE'21, pages 1110–1115, IEEE, 2021.

[Temel TACAS'24] M. Temel eSCMul: Verified Implementation of S-C-Rewriting for Multiplier Verification. In Proc. of TACAS'24, pages 485–507, Springer, 2024.

[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.