GUESS AND PROVE: A HYBRID APPROACH TO LINEAR POLYNOMIAL RECOVERY IN CIRCUIT VERIFICATION

Clemens Hofstadler
Johannes Kepler University Linz, Austria

Daniela Kaufmann
TU Wien, Austria

31st Intl. Conf. on Principles and Practice of Constraint Programming
Glasgow, Scotland

August 12, 2025

TU WIEN Informatics

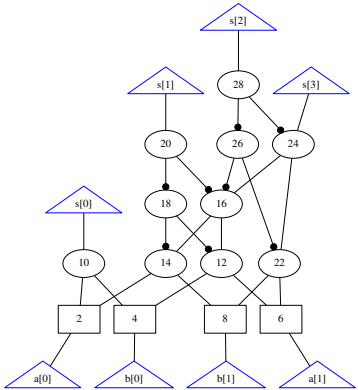
FWF Austrian Science Fund

And-Inverter Graphs & Multiplier Circuits

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

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

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

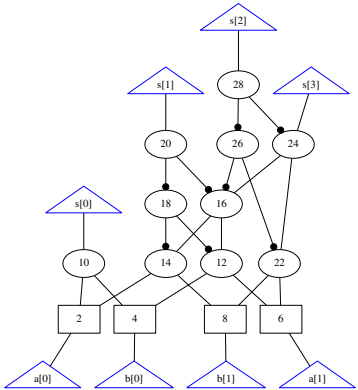


And-Inverter Graphs & Multiplier Circuits

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

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

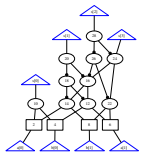
$$(2a_1 + a_0) * (2b_1 + b_0) = 8s_3 + 4s_2 + 2s_1 + s_0?$$



IMPORTANT: Our approach is not limited to multipliers!

Basic Idea of Algebraic Approach

Graph



Polynomials

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

Specification

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

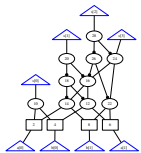
Implication

$= 0$ ✓

$\neq 0$ ✗

Basic Idea of Algebraic Approach

Graph



Polynomials

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

Specification

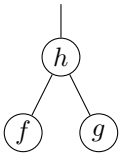
$$\sum_{i=0}^{2n-1} 2^i s_i - (\sum_{i=0}^{n-1} 2^i a_i)(\sum_{i=0}^{n-1} 2^i b_i)$$

Ideal Membership

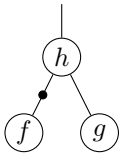
$= 0$ ✓

$\neq 0$ ✗

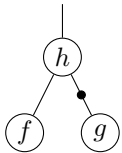
From AIGs to Polynomials



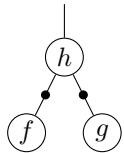
$h = f \wedge g$



$h = \neg f \wedge g$

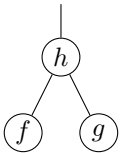


$h = f \wedge \neg g$



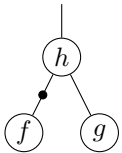
$h = \neg f \wedge \neg g$

From AIGs to Polynomials



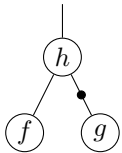
$h = f \wedge g$

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



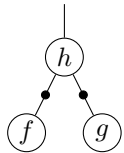
$h = \neg f \wedge g$

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



$h = f \wedge \neg g$

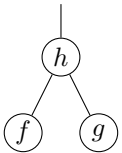
f	g	h
0	0	0
0	1	0
1	0	1
1	1	0



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

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

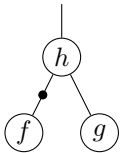
From AIGs to Polynomials



$h = f \wedge g$

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

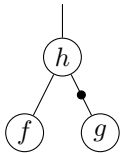
$-h + fg$



$h = \neg f \wedge g$

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

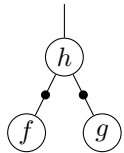
$-h + (1 - f)g$



$h = f \wedge \neg g$

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

$-h + f(1 - g)$



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

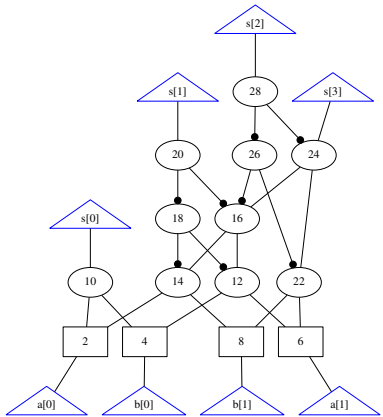
f	g	h
0	0	1
0	1	0
1	0	0
1	1	0

$-h + (1 - f)(1 - g)$

From AIGs to Polynomials

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

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



From AIGs to Polynomials

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

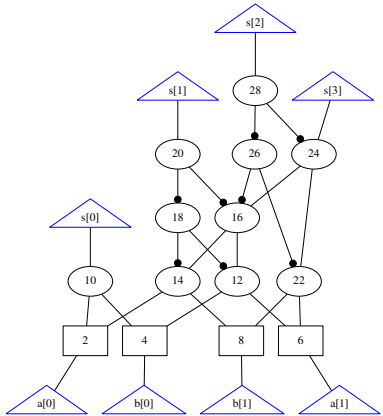
$$
\begin{array}{ll}
-s_3 + l_{24} & -l_{22} + a_1 b_1 \\
-s_2 + l_{28} & -l_{20} + l_{18} l_{16} - l_{18} - l_{16} + 1 \\
-s_1 + l_{20} & -l_{18} + l_{14} l_{12} - l_{14} - l_{12} + 1 \\
-s_0 + l_{10} & -l_{16} + l_{14} l_{12} \\
-l_{28} + l_{26} l_{24} - l_{26} - l_{24} + 1 & -l_{14} + a_0 b_1 \\
-l_{26}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

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

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

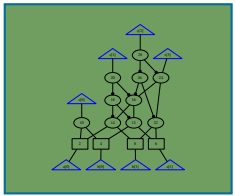
Specification $\mathcal{S} \in \mathbb{Q}[X]$.

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



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

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

Ideal Membership

$= 0$ ✓

$\neq 0$ ✗

Ideal

Ideal. A subset $I \subset \mathbb{K}[X]$ is an ideal if it satisfies:

- $0 \in I$
- If $f, g \in I$, then $f + g \in I$.
- If $f \in I$ and $h \in \mathbb{K}[X]$ then $hf \in I$.

Ideal

Ideal. A subset $I \subset \mathbb{K}[X]$ is an ideal if it satisfies:

- $0 \in I$
- If $f, g \in I$, then $f + g \in I$.
- If $f \in I$ and $h \in \mathbb{K}[X]$ then $hf \in I$.

Ideal generated by a finite number of polynomials.
Let $f_1, \ldots, f_s \in \mathbb{K}[X]$. Then we set

$$\langle f_1, \ldots, f_s \rangle = \{h_1 f_1 + \cdots + h_s f_s \mid h_1, \ldots, h_s \in \mathbb{K}[X]\}.$$

$\langle f_1, \ldots, f_s \rangle$ is an ideal and is called the ideal generated by $f_1, \ldots, f_s$.

We can think of $\langle f_1, \ldots, f_s \rangle$ as consisting of all "polynomial consequences" of the equations $f_1 = f_2 = \ldots = f_s = 0$.

Gröbner Bases

A Gröbner basis is a special kind of generating set for an ideal with the following properties:

1. Every $f \in \mathbb{K}[X]$ has a unique computable remainder $r \in \mathbb{K}[X]$ with respect to G.

Gröbner Bases

A Gröbner basis is a special kind of generating set for an ideal with the following properties:

1. Every $f \in \mathbb{K}[X]$ has a unique computable remainder $r \in \mathbb{K}[X]$ with respect to G.

2. $f - r \in \langle G \rangle$. Hence, $f \in \langle G \rangle$ iff the remainder of f with respect to G is zero.

Gröbner Bases

A Gröbner basis is a special kind of generating set for an ideal with the following properties:

1. Every $f \in \mathbb{K}[X]$ has a unique computable remainder $r \in \mathbb{K}[X]$ with respect to G.

2. $f - r \in \langle G \rangle$. Hence, $f \in \langle G \rangle$ iff the remainder of f with respect to G is zero.

3. Every ideal $I \subseteq \mathbb{K}[X]$ has a Gröbner basis G w.r.t. a fixed monomial order.

Gröbner Bases

A Gröbner basis is a special kind of generating set for an ideal with the following properties:

1. Every $f \in \mathbb{K}[X]$ has a unique computable remainder $r \in \mathbb{K}[X]$ with respect to G.

2. $f - r \in \langle G \rangle$. Hence, $f \in \langle G \rangle$ iff the remainder of f with respect to G is zero.

3. Every ideal $I \subseteq \mathbb{K}[X]$ has a Gröbner basis G w.r.t. a fixed monomial order.

4. Given an arbitrary basis, we can compute G (double-exponential).

Ideal Membership Problem for Circuits

[RitircBiereKauers FMCAD'17]

- Polynomial Encoding:
  - Gate polynomials $G(C)$
  - Boolean input constraints $B(C)$
- Let $J(C) = \langle G(C) \cup B(C) \rangle$.

Ideal Membership Problem for Circuits

[RitircBiereKauers FMCAD'17]

- Polynomial Encoding:
  - Gate polynomials $G(C)$
  - Boolean input constraints $B(C)$
- Let $J(C) = \langle G(C) \cup B(C) \rangle$.
- Lexicographic term order: Output variable of a gate is greater than input variables.

Ideal Membership Problem for Circuits

[RitircBiereKauers FMCAD'17]

- Polynomial Encoding:
  - Gate polynomials $G(C)$
  - Boolean input constraints $B(C)$
- Let $J(C) = \langle G(C) \cup B(C) \rangle$.
- Lexicographic term order: Output variable of a gate is greater than input variables.
- $G(C) \cup B(C)$ is a Gröbner basis for $J(C)$.

Ideal Membership Problem for Circuits

[RitircBiereKauers FMCAD'17]

- Polynomial Encoding:
  - Gate polynomials $G(C)$
  - Boolean input constraints $B(C)$
- Let $J(C) = \langle G(C) \cup B(C) \rangle$.
- Lexicographic term order: Output variable of a gate is greater than input variables.
- $G(C) \cup B(C)$ is a Gröbner basis for $J(C)$.

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.

Strategies

1. Encoding
   - Embedding different phases
     [KaufmannBeameBiereNordström DATE'22, KonradScholl FMCAD'24]
2. Preprocessing
   - Variable Elimination [MahzoonGroßeDrechsler DAC'19, RitircBiereKauers DATE'18]
3. Reduction
   - Incremental Algorithm [RitircBiereKauers FMCAD'17]
   - Dynamic Reduction Order
     [MahzoonGroßeScholIDrechsler DATE'20, KonradScholl FMCAD'24]
4. Final stage adder
   - Include SAT or BDDs [KaufmannBiereKauers FMCAD'19, DrechslerMahzoon ISEEIE'22]

Strategies

1. Encoding
   - Embedding different phases
     [KaufmannBeameBiereNordström DATE'22, KonradScholl FMCAD'24]
2. Preprocessing
   - Variable Elimination [MahzoonGroßeDrechsler DAC'19, RitircBiereKauers DATE'18]
3. Reduction
   - Incremental Algorithm [RitircBiereKauers FMCAD'17]
   - Dynamic Reduction Order
     [MahzoonGroßeScholIDrechsler DATE'20, KonradScholl FMCAD'24]
4. Final stage adder
   - Include SAT or BDDs [KaufmannBiereKauers FMCAD'19, DrechslerMahzoon ISEEIE'22]

All of these strategies rely on a **lexicographic term ordering**.

TACAS 2025: Change of Order

[KaufmannBerthomieu TACAS'25]

	$\prec_{\mathrm{lex}}$	$\prec_{\mathrm{drl}}$
GB Computation	✅ Easy	⚠️ Hard
Spec Reduction	⚠️ Hard	✅ Easy

TACAS 2025: Change of Order

[KaufmannBerthomieu TACAS'25]

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

If the specification polynomial is linear,

a Gröbner basis with respect to a degree-compatible term ordering

contains linear polynomials that suffice to derive correctness of the circuit.

Recovering Linear Polynomials

Algorithm 1: Verification using linear extractions

Input : Circuit C in AIG format, Specification polynomial $\mathcal{S}$

Output: Determine whether C fulfills the specification

$G_{\text{init}} \leftarrow$ Polynomial-Encoding(C);

$\mathcal{S}_{\text{lin}}, G_{\text{ext}} \leftarrow$ Linearize-Spec($\mathcal{S}, G_{\text{init}}$);

$\mathbb{L}(G_{\text{ext}}) \leftarrow$ EXTRACT-LINEAR-POLY(G_{ext});

$\mathcal{S}_{\text{lin}} \leftarrow$ Linear-Reduce($\mathcal{S}_{\text{lin}}, \mathbb{L}(G_{\text{ext}})$);

return $\mathcal{S}_{\text{lin}} = 0$;

Recovering Linear Polynomials

Algorithm 2: Verification using linear extractions

Input : Circuit C in AIG format, Specification polynomial $\mathcal{S}$

Output: Determine whether C fulfills the specification

$G_{\text{init}} \leftarrow$ Polynomial-Encoding(C);

$\mathcal{S}_{\text{lin}}, G_{\text{ext}} \leftarrow$ Linearize-Spec($\mathcal{S}, G_{\text{init}}$);

$\mathbb{L}(G_{\text{ext}}) \leftarrow$ EXTRACT-LINEAR-POLY(G_{ext});

$\mathcal{S}_{\text{lin}} \leftarrow$ Linear-Reduce($\mathcal{S}_{\text{lin}}, \mathbb{L}(G_{\text{ext}})$);

return $\mathcal{S}_{\text{lin}} = 0$;

EXTRACT-LINEAR-POLY = **Compute a** $\prec_{\text{drl}}$ **Gröbner basis** for the circuit.

Recovering Linear Polynomials

Algorithm 3: Verification using linear extractions

Input : Circuit C in AIG format, Specification polynomial $\mathcal{S}$

Output: Determine whether C fulfills the specification

$G_{\text{init}} \leftarrow$ Polynomial-Encoding(C);

$\mathcal{S}_{\text{lin}}, G_{\text{ext}} \leftarrow$ Linearize-Spec($\mathcal{S}, G_{\text{init}}$);

$\mathbb{L}(G_{\text{ext}}) \leftarrow$ EXTRACT-LINEAR-POLY(G_{ext});

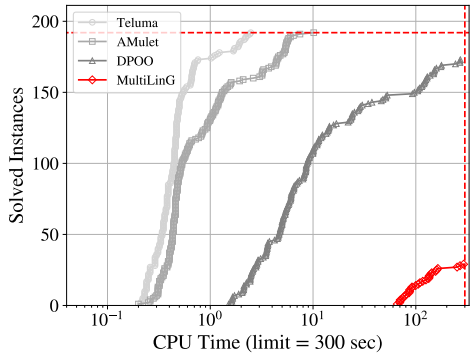
$\mathcal{S}_{\text{lin}} \leftarrow$ Linear-Reduce($\mathcal{S}_{\text{lin}}, \mathbb{L}(G_{\text{ext}})$);

return $\mathcal{S}_{\text{lin}} = 0$;

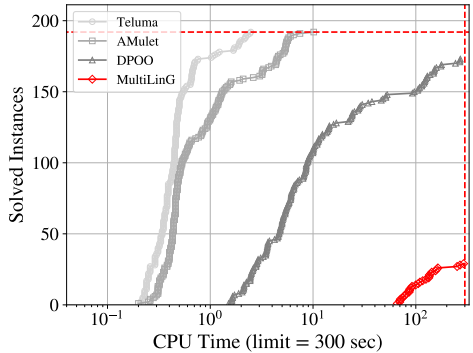
EXTRACT-LINEAR-POLY = **Compute a** $\prec_{\text{drl}}$ **Gröbner basis** for the circuit.

Computing a Gröbner basis for the whole circuit is too expensive $\rightarrow$ Linearize Sub-Circuits

TACAS 2025: Results

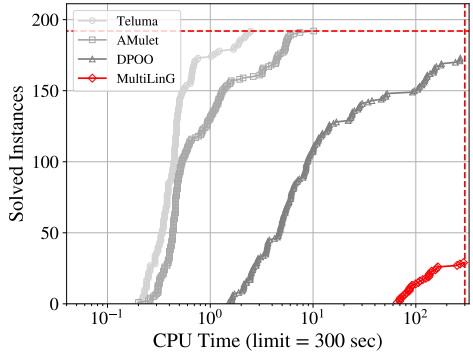


TACAS 2025: Results



- Computing a Gröbner basis is costly.
- Ignores existing structure of encoding.

TACAS 2025: Results



We propose two alternative methods:

1. FGLM-style Linearization
2. Guess-and-Prove Linearization

FGLM Algorithm

- Developed by Faugère, Gianni, Lazard, and Mora in 1993.
- Converts Gröbner bases from one term ordering to another.
- Computes multiplication matrices for the quotient ring and uses linear algebra to determine the new basis elements.
- Polynomial complexity in number of roots.

FGLM-style Linearization

Ideal $I \subseteq \mathbb{Q}[a, b, g_1, g_2, g_3, g_4]$ generated by:

$$g_1 - ab, \quad g_2 - (1-a)(1-b), \quad g_3 - a(1-b), \quad g_4 - (1-g_1)(1-g_2), \quad a^2 - a, \quad b^2 - b$$

FGLM-style Linearization

Ideal $I \subseteq \mathbb{Q}[a, b, g_1, g_2, g_3, g_4]$ generated by:

$$g_1 - ab, \quad g_2 - (1-a)(1-b), \quad g_3 - a(1-b), \quad g_4 - (1-g_1)(1-g_2), \quad a^2 - a, \quad b^2 - b$$

Compute Normalforms:

$$g_1 = ab, \quad g_2 = ab - a - b + 1, \quad g_3 = -ab + a, \quad g_4 = -2ab + a + b$$

FGLM-style Linearization

Ideal $I \subseteq \mathbb{Q}[a, b, g_1, g_2, g_3, g_4]$ generated by:

$$g_1 - ab, \quad g_2 - (1-a)(1-b), \quad g_3 - a(1-b), \quad g_4 - (1-g_1)(1-g_2), \quad a^2 - a, \quad b^2 - b$$

Compute Normalforms:

$$g_1 = ab, \quad g_2 = ab - a - b + 1, \quad g_3 = -ab + a, \quad g_4 = -2ab + a + b$$

Matrix A:

$$\begin{array}{ccccccc} \mathrm{NF}_G(1) & \mathrm{NF}_G(a) & \mathrm{NF}_G(b) & \mathrm{NF}_G(g_1) & \mathrm{NF}_G(g_2) & \mathrm{NF}_G(g_3) & \mathrm{NF}_G(g_4) \end{array}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & -1 & -2 \end{pmatrix} \begin{matrix} 1 \\ a \\ b \\ ab \end{matrix}$$

FGLM-style Linearization

Ideal $I \subseteq \mathbb{Q}[a, b, g_1, g_2, g_3, g_4]$ generated by:

$$g_1 - ab, \quad g_2 - (1-a)(1-b), \quad g_3 - a(1-b), \quad g_4 - (1-g_1)(1-g_2), \quad a^2 - a, \quad b^2 - b$$

Kernel A:

$$\begin{matrix} 1 & a & b & g_1 & g_2 & g_3 & g_4 \end{matrix}$$
$$\begin{pmatrix} 0 & -1 & -1 & 2 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 & 1 & 0 \\ -1 & 1 & 1 & -1 & 1 & 0 & 0 \end{pmatrix}$$

FGLM-style Linearization

Ideal $I \subseteq \mathbb{Q}[a, b, g_1, g_2, g_3, g_4]$ generated by:

$$g_1 - ab, \quad g_2 - (1-a)(1-b), \quad g_3 - a(1-b), \quad g_4 - (1-g_1)(1-g_2), \quad a^2 - a, \quad b^2 - b$$

Kernel A:

$$\begin{array}{ccccccc} 1 & a & b & g_1 & g_2 & g_3 & g_4 \end{array}$$
$$\begin{pmatrix} 0 & -1 & -1 & 2 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 & 1 & 0 \\ -1 & 1 & 1 & -1 & 1 & 0 & 0 \end{pmatrix}$$

Kernel yields:

$$g_4 + 2g_1 - a - b, \qquad g_3 + g_1 - a, \qquad g_2 - g_1 + a + b - 1$$

FGLM-style Linearization

Algorithm 4: Verification using linear extractions

Input: Gröbner basis G of $I \subseteq K[x_1, \ldots, x_n]$
Output: Vector space basis L of $L(I)$
Compute $\mathsf{NF}_G(1), \mathsf{NF}_G(x_1), \ldots, \mathsf{NF}_G(x_n)$;
Construct coefficient matrix A;
Compute $\ker(A)$;
Return basis vectors as polynomials: $c_0 + c_1 x_1 + \cdots + c_n x_n$;

- Normalform computation can be exponential in the worst case.
- Works only for “small” sub-circuits.

Algorithm 2: Guess-and-Prove Linearization

When FGLM is too expensive: Use interpolation.

Algorithm 2: Guess-and-Prove Linearization

When FGLM is too expensive: Use interpolation.

1. Sample N input points using sampling function
2. Build matrix A from these points
3. Solve $\ker(A)$ to get candidate polynomials
4. Verify each candidate for correctness
5. If incorrect: add witness point as sample and repeat

Guess I

Ideal $I \subseteq \mathbb{Q}[a, b, g_1, g_2, g_3, g_4]$ generated by:

$$g_1 - ab, \quad g_2 - (1-a)(1-b), \quad g_3 - a(1-b), \quad g_4 - (1-g_1)(1-g_2), \quad a^2 - a, \quad b^2 - b$$

Guess I

Ideal $I \subseteq \mathbb{Q}[a, b, g_1, g_2, g_3, g_4]$ generated by:

$$g_1 - ab, \quad g_2 - (1-a)(1-b), \quad g_3 - a(1-b), \quad g_4 - (1-g_1)(1-g_2), \quad a^2 - a, \quad b^2 - b$$

Sample (a, b): $(0, 0), (1, 0), (0, 1)$

Guess I

Ideal $I \subseteq \mathbb{Q}[a, b, g_1, g_2, g_3, g_4]$ generated by:

$$g_1 - ab, \quad g_2 - (1-a)(1-b), \quad g_3 - a(1-b), \quad g_4 - (1-g_1)(1-g_2), \quad a^2 - a, \quad b^2 - b$$

Sample (a, b): $(0, 0), (1, 0), (0, 1)$

Matrix A:

$$\begin{matrix} 1 & a & b & g_1 & g_2 & g_3 & g_4 \end{matrix}$$
$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Guess I

Ideal $I \subseteq \mathbb{Q}[a, b, g_1, g_2, g_3, g_4]$ generated by:

$$g_1 - ab, \quad g_2 - (1-a)(1-b), \quad g_3 - a(1-b), \quad g_4 - (1-g_1)(1-g_2), \quad a^2 - a, \quad b^2 - b$$

Kernel A:

$$\begin{array}{c} \begin{array}{ccccccc} 1 & a & b & g_1 & g_2 & g_3 & g_4 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & -1 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Guess I

Ideal $I \subseteq \mathbb{Q}[a, b, g_1, g_2, g_3, g_4]$ generated by:

$$g_1 - ab, \quad g_2 - (1-a)(1-b), \quad g_3 - a(1-b), \quad g_4 - (1-g_1)(1-g_2), \quad a^2 - a, \quad b^2 - b$$

Kernel A:

$$\begin{array}{c} \begin{array}{ccccccc} 1 & a & b & g_1 & g_2 & g_3 & g_4 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & -1 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Kernel yields:

$$g_1, \qquad g_2 + a + b - 1, \qquad g_3 - a, \qquad g_4 - b - a$$

Prove

Prove

Algebraic Proof: Reduction of g_1 by $G(C) \cup B(C) \rightarrow$ does not scale!

Prove

Use SAT: Encoding of Circuit:

$$(\neg a \vee \neg b \vee g_1) \wedge (a \vee \neg g_1) \wedge (b \vee \neg g_1)$$
$$(a \vee b \vee g_2) \wedge (\neg g_2 \vee \neg a) \wedge (\neg g_2 \vee \neg b)$$
$$(\neg a \vee b \vee g_3) \wedge (a \vee \neg g_3) \wedge (\neg b \vee \neg g_3)$$
$$(g_1 \vee g_2 \vee g_4) \wedge (\neg g_4 \vee \neg g_1) \wedge (\neg g_4 \vee \neg g_2)$$

Encoding of guessed polynomial as $(g_1 > 0) \vee (g_1 < 0)$ using PBlib.

Prove

Use SAT: Encoding of Circuit:

$$(\neg a \vee \neg b \vee g_1) \wedge (a \vee \neg g_1) \wedge (b \vee \neg g_1)$$
$$(a \vee b \vee g_2) \wedge (\neg g_2 \vee \neg a) \wedge (\neg g_2 \vee \neg b)$$
$$(\neg a \vee b \vee g_3) \wedge (a \vee \neg g_3) \wedge (\neg b \vee \neg g_3)$$
$$(g_1 \vee g_2 \vee g_4) \wedge (\neg g_4 \vee \neg g_1) \wedge (\neg g_4 \vee \neg g_2)$$

Encoding of guessed polynomial as $(g_1 > 0) \vee (g_1 < 0)$ using PBlib.

If polynomial is correct the SAT solver returns UNSAT, otherwise we get a sample that violates the polynomial.

Prove

Use SAT: Encoding of Circuit:

$$(\neg a \vee \neg b \vee g_1) \wedge (a \vee \neg g_1) \wedge (b \vee \neg g_1)$$
$$(a \vee b \vee g_2) \wedge (\neg g_2 \vee \neg a) \wedge (\neg g_2 \vee \neg b)$$
$$(\neg a \vee b \vee g_3) \wedge (a \vee \neg g_3) \wedge (\neg b \vee \neg g_3)$$
$$(g_1 \vee g_2 \vee g_4) \wedge (\neg g_4 \vee \neg g_1) \wedge (\neg g_4 \vee \neg g_2)$$

Encoding of guessed polynomial as $(g_1 > 0) \vee (g_1 < 0)$ using PBlib.

SAT Solver returns SAT and gives us $a = 1$, $b = 1$ as violating sample.

Guess II

Ideal $I \subseteq \mathbb{Q}[a, b, g_1, g_2, g_3, g_4]$ generated by:

$$g_1 - ab, \quad g_2 - (1-a)(1-b), \quad g_3 - a(1-b), \quad g_4 - (1-g_1)(1-g_2), \quad a^2 - a, \quad b^2 - b$$

Sample (a, b): $(0,0), (1,0), (0,1), (1,1)$

Matrix A:

$$\begin{array}{c} 1 \quad a \quad b \quad g_1 \quad g_2 \quad g_3 \quad g_4 \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Guess II

Ideal $I \subseteq \mathbb{Q}[a, b, g_1, g_2, g_3, g_4]$ generated by:

$$g_1 - ab, \quad g_2 - (1-a)(1-b), \quad g_3 - a(1-b), \quad g_4 - (1-g_1)(1-g_2), \quad a^2 - a, \quad b^2 - b$$

Sample (a, b): $(0,0), (1,0), (0,1), (1,1)$

Kernel A:

$$\begin{array}{c} \begin{array}{ccccccc} 1 & a & b & g_1 & g_2 & g_3 & g_4 \end{array} \\ \begin{pmatrix} 0 & -1 & -1 & 2 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 & 1 & 0 \\ -1 & 1 & 1 & -1 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

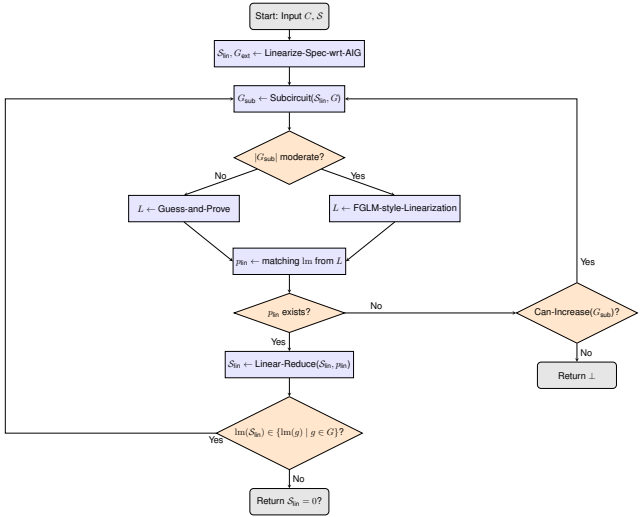
Kernel yields:

$$g_4 + 2g_1 - a - b, \qquad g_3 + g_1 - a, \qquad g_2 - g_1 + a + b - 1$$

Guess-and-Prove Linearization

- Works for large circuits.
- In the worst case we have to enumerate all input points.
- In our example we only require 10k samples out of $\sim 2^{30}$ possibilities

Recovering Linear Polynomials



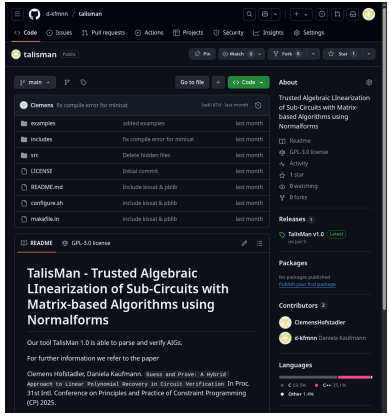
Optimizations

- **Caching:** Reuse linear basis L for isomorphic subcircuits via standardized variable mappings. Yields $> 99\%$ reuse.
- **Linear Extraction:** Extract all linear polynomials from L, not just p_{lin} as they may be needed for later reductions.

TALISMAN

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

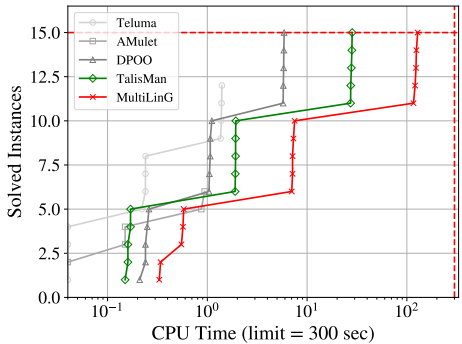
- Builds on MULTILING, written in C++
- Uses FLINT for matrix computations
- Guess and Prove Algorithm:
  - Uses PBLIB to translate guessed polynomial to CNF
  - Uses KISSAT as a SAT solver



Evaluation - Multiplier Circuits

Optimized multipliers

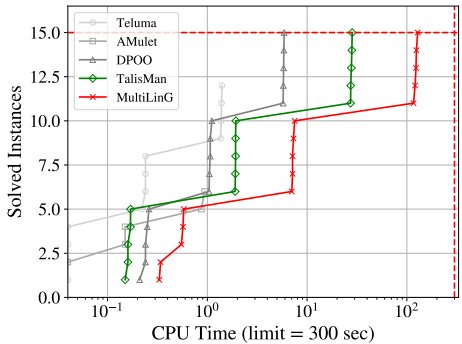
- 5 architectures
- bit-widths 32, 64 & 128



Evaluation - Multiplier Circuits

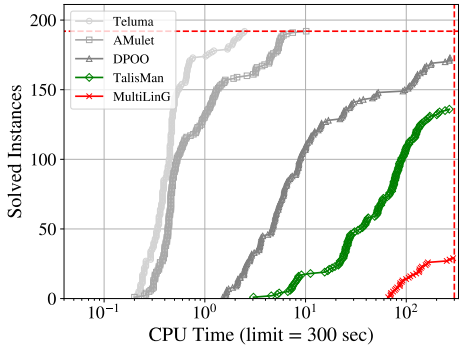
Optimized multipliers

- 5 architectures
- bit-widths 32, 64 & 128



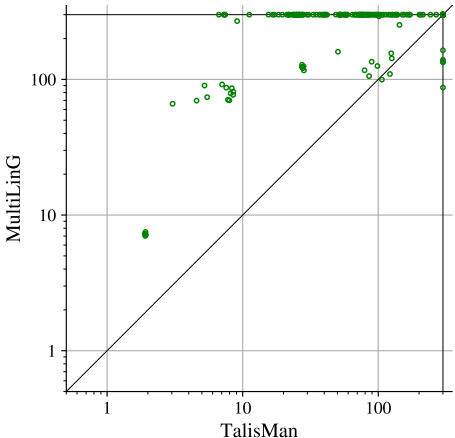
Unsigned 64-bit multipliers

- 192 architectures
- bit-width 64

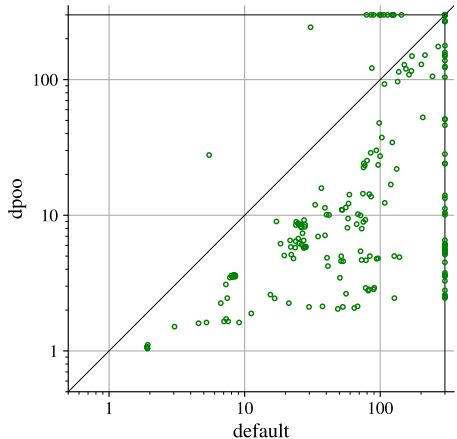


Evaluation - Multiplier Circuits

5 benchmarks not solved by TALISMAN



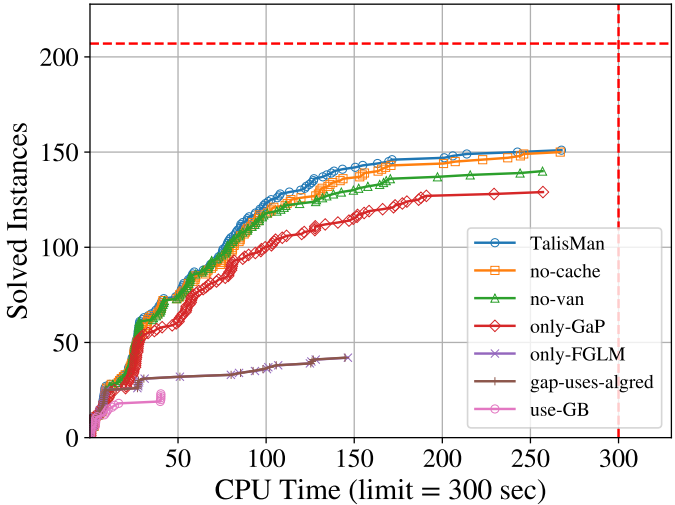
11 benchmarks not solved by DPOO



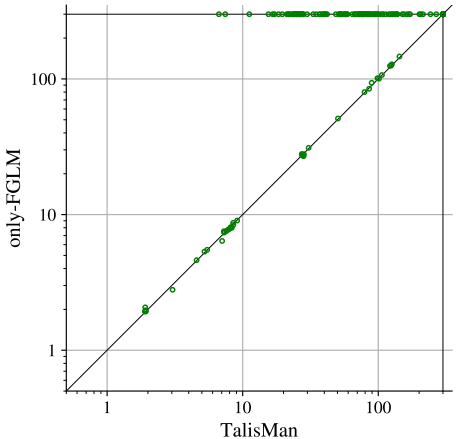
Evaluation - Optimized Multipliers

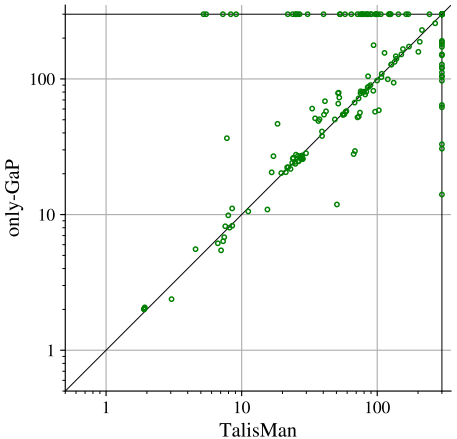
Name	Total	Subcircuits		FGLM		Guess and Prove					
	Time	#	%Ch	#	Time	#	Time	Guess	Eval	%Co	m.It.
abc64-cmp	1.9	4033	99.7	12	0.0	0	0	0	0	0	0
abc64-rsn2	1.9	4033	99.7	12	0.0	0	0	0	0	0	0
bp-ar-ks	205.9	6040	98.7	74	0.6	3	197.6	15087	3855	28.5	5
bp-ba-hc	113.2	6147	97.6	144	0.6	1	104.2	4432	2754	48.0	3
bp-bd-rb	75.6	5561	98.2	100	0.6	2	67.9	4452	1364	80.3	3
bp-ct-csv	68.3	5204	88.2	614	40.4	1	17.3	864	864	100	1
sp-ar-rc	3.0	6509	99.7	18	0.0	0	0	0	0	0	0
sp-ba-csf	26.3	8012	98.9	87	0.0	1	18.0	883	883	100	1
sp-bd-lf	81.5	6403	99.4	40	0.0	1	76.3	5733	2077	52.4	5
sp-ct-hc	98.2	7756	97.7	174	0.1	1	89.0	3675	2154	52.5	3

Evaluation - Ablation Study



Evaluation - Ablation Study





Conclusion

Two alternative algorithms to $\prec_{\mathrm{drl}}$-Gröbner bases for extracting linear relations from ideals.

FGLM-Approach

- requires a given Gröbner basis
- matrix based of normal forms
- computes exact solutions
- works for small circuits

Guess-and-Prove

- requires zero-dimension of ideal
- matrix based on samples
- guesses solutions
- works for bigger circuits

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

[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.

[KaufmannBerthomieu TACAS'25] D. Kaufmann and J. Berthomieu Extracting Linear Relations from Gröbner Bases for Formal Verification of And-Inverter Graphs In Proc. of TACAS'25, LNCS, 2025.

[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.

[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.

References III

[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.

[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.

References IV

[MahzoonGroßeScholIDrechsler 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.

[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.

References V

[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.