COMBINING SAT AND COMPUTER ALGEBRA FOR MULTIPLIER VERIFICATION

Daniela Kaufmann
TU Wien, Vienna, Austria

Dagstuhl Seminar
SAT Encodings and Beyond
Dagstuhl, Germany

June 27, 2023

✉ daniela.kaufmann@tuwien.ac.at

Bugs in hardware are expensive!

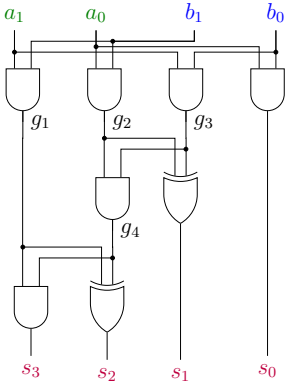
Circuit verification prevents issues like the famous Pentium FDIV bug.

Multiplier verification

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

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

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

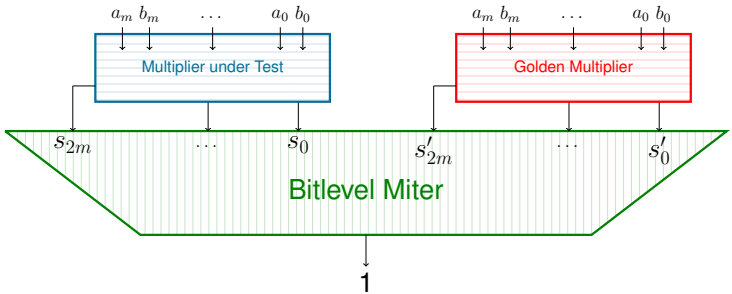


Formal Verification Techniques

Satisfiability Checking (SAT)

- SAT 2016 Competition [Biere SATComp'16]
- Exponential run-time of solvers

Equivalence checking: Miter



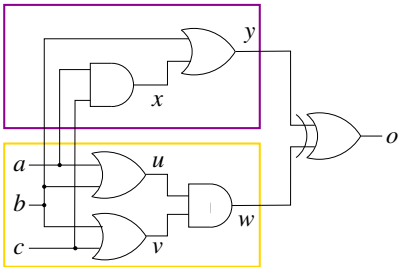
- Miter is translated to CNF.
- CNF is given to SAT solver.
- To show correctness SAT solver needs to return UNSAT.

Tseitin Transformation: Circuit to CNF

$(a \wedge c) \vee b$

=

$(a \vee b) \wedge (b \vee c)$



$$
\begin{aligned}
& o \ \wedge \\
& (x \ \leftrightarrow \ a \wedge c) \wedge \\
& (y \ \leftrightarrow \ b \vee x) \wedge \\
& (u \ \leftrightarrow \ a \vee b) \wedge \\
& (v \ \leftrightarrow \ b \vee c) \wedge \\
& (w \leftrightarrow \ u \wedge v) \wedge \\
& (o \ \leftrightarrow y \oplus w)
\end{aligned}
$$

$$o \wedge (x \rightarrow a) \wedge (x \rightarrow c) \wedge (x \leftarrow a \wedge c) \wedge \ \ldots$$

$$\boxed{o \wedge (\overline{x} \vee a) \wedge (\overline{x} \vee c) \wedge (x \vee \overline{a} \vee \overline{c}) \wedge \ \ldots}$$

Tseitin Transformation: Gate Constraints

Negation:

$$\begin{aligned} x \leftrightarrow \overline{y} &\Leftrightarrow (x \rightarrow \overline{y}) \wedge (\overline{y} \rightarrow x) \\ &\Leftrightarrow (\overline{x} \vee \overline{y}) \wedge (y \vee x) \end{aligned}$$

Disjunction:

$$\begin{aligned} x \leftrightarrow (y \vee z) &\Leftrightarrow (y \rightarrow x) \wedge (z \rightarrow x) \wedge (x \rightarrow (y \vee z)) \\ &\Leftrightarrow (\overline{y} \vee x) \wedge (\overline{z} \vee x) \wedge (\overline{x} \vee y \vee z) \end{aligned}$$

Conjunction:

$$\begin{aligned} x \leftrightarrow (y \wedge z) &\Leftrightarrow (x \rightarrow y) \wedge (x \rightarrow z) \wedge ((y \wedge z) \rightarrow x) \\ &\Leftrightarrow (\overline{x} \vee y) \wedge (\overline{x} \vee z) \wedge (\overline{(y \wedge z)} \vee x) \\ &\Leftrightarrow (\overline{x} \vee y) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z} \vee x) \end{aligned}$$

Equivalence:

$$\begin{aligned} x \leftrightarrow (y \leftrightarrow z) &\Leftrightarrow (x \rightarrow (y \leftrightarrow z)) \wedge ((y \leftrightarrow z) \rightarrow x) \\ &\Leftrightarrow (x \rightarrow ((y \rightarrow z) \wedge (z \rightarrow y)) \wedge ((y \leftrightarrow z) \rightarrow x) \\ &\Leftrightarrow (x \rightarrow (y \rightarrow z)) \wedge (x \rightarrow (z \rightarrow y)) \wedge ((y \leftrightarrow z) \rightarrow x) \\ &\Leftrightarrow (\overline{x} \vee \overline{y} \vee z) \wedge (\overline{x} \vee \overline{z} \vee y) \wedge ((y \leftrightarrow z) \rightarrow x) \\ &\Leftrightarrow (\overline{x} \vee \overline{y} \vee z) \wedge (\overline{x} \vee \overline{z} \vee y) \wedge (((y \wedge z) \vee (\overline{y} \wedge \overline{z})) \rightarrow x) \\ &\Leftrightarrow (\overline{x} \vee \overline{y} \vee z) \wedge (\overline{x} \vee \overline{z} \vee y) \wedge ((y \wedge z) \rightarrow x) \wedge ((\overline{y} \wedge \overline{z}) \rightarrow x) \\ &\Leftrightarrow (\overline{x} \vee \overline{y} \vee z) \wedge (\overline{x} \vee \overline{z} \vee y) \wedge (\overline{y} \vee \overline{z} \vee x) \wedge (y \vee z \vee x) \end{aligned}$$

Checking Commutativity of Multiplication with SAT

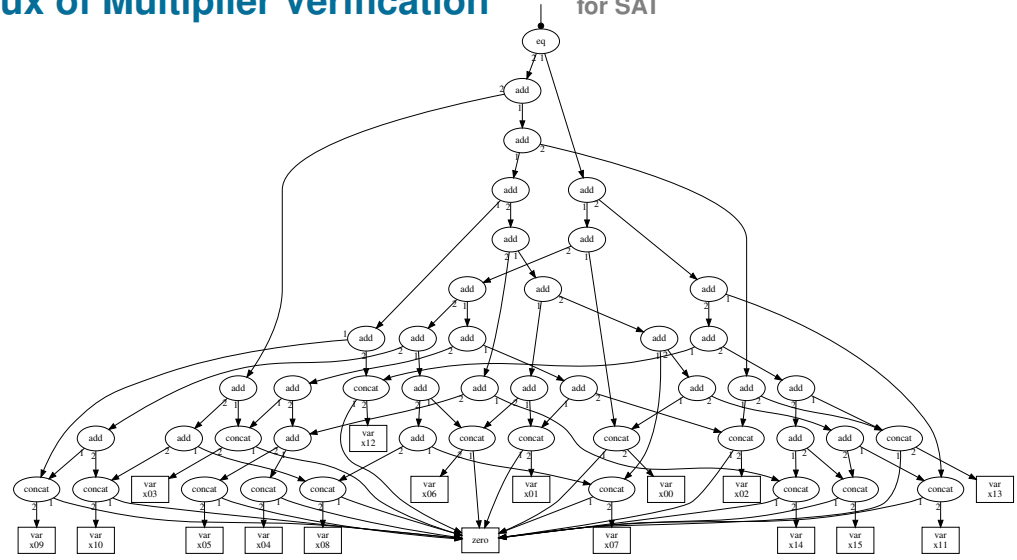
```
bits  Kissat
01      0.00
02      0.00
03      0.00
04      0.00
05      0.00
06      0.02
07      0.12
08      1.23
09      7.85
10     39.42
11    152.89
12    820.12
13    ------

limit of 900 seconds wall clock time
```

```
(set-logic QF_BV)
(declare-fun x () (_ BitVec 12))
(declare-fun y () (_ BitVec 12))
(assert (distinct (bvmul x y) (bvmul y x)))
(check-sat)
```

Crux of Multiplier Verification

for SAT



Formal Verification Techniques

Satisfiability Checking (SAT)

- SAT 2016 Competition
- Exponential run-time of solvers

Decision Diagrams

- First technique to detect Pentium bug
- Rely on manual decomposition

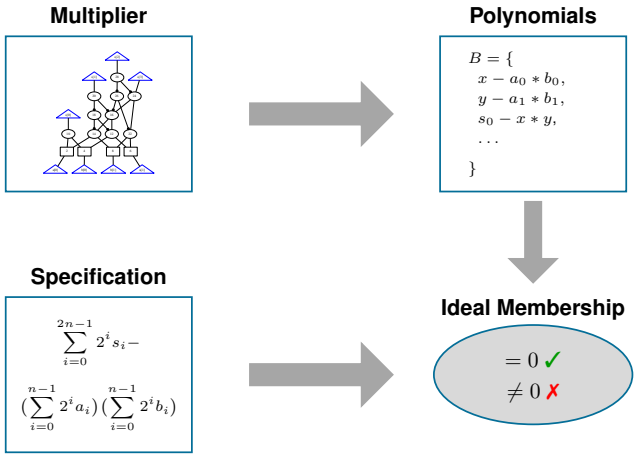
Theorem Proving

- Used in industry
- Requires manual effort
- Automated techniques rely on hierarchical information.

Algebraic Approach

- Great progress since 2015
- Polynomial encoding
- Works for non-trivial multiplier designs

Basic Idea of Algebraic Approach



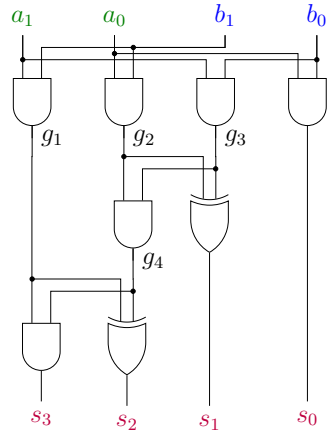
From Circuits to Polynomials

AND-Gate $f \wedge g = y$

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

XOR-Gate $f \oplus g = y$

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



From Circuits to Polynomials

AND-Gate $f \wedge g = y$

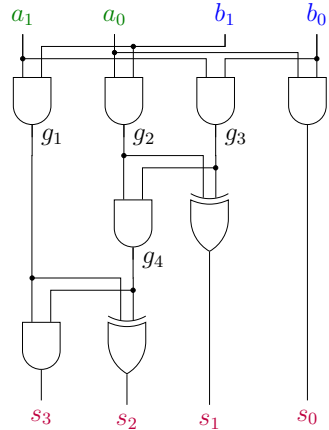
f	g	y
0	0	0
0	1	0
1	0	0
1	1	1

$$-y + fg$$

XOR-Gate $f \oplus g = y$

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

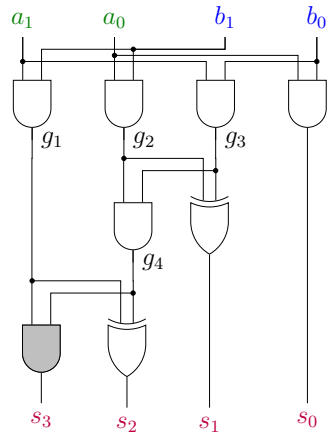
$$-y + f + g - 2fg$$



From Circuits to Polynomials

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

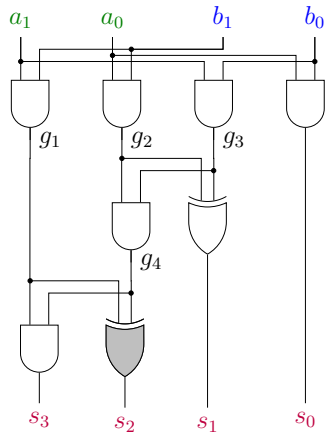
$$s_3 = g_1 \wedge g_4 \qquad -s_3 + g_4 g_1,$$



From Circuits to Polynomials

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

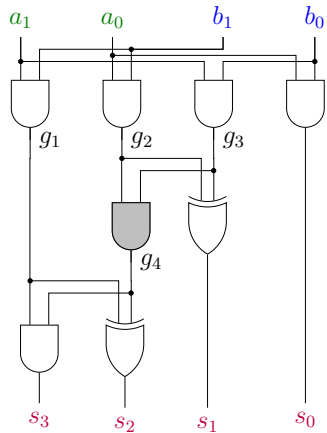
$$
\begin{aligned}
s_3 = g_1 \wedge g_4 \qquad & -s_3 + g_4 g_1, \\
s_2 = g_1 \oplus g_4 \qquad & -s_2 - 2g_4 g_1 + g_4 + g_1,
\end{aligned}
$$



From Circuits to Polynomials

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

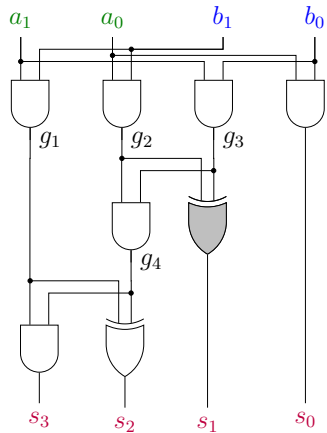
$$
\begin{aligned}
s_3 = g_1 \wedge g_4 \qquad & -s_3 + g_4 g_1, \\
s_2 = g_1 \oplus g_4 \qquad & -s_2 - 2g_4 g_1 + g_4 + g_1, \\
g_4 = g_2 \wedge g_3 \qquad & -g_4 + g_2 g_3,
\end{aligned}
$$



From Circuits to Polynomials

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

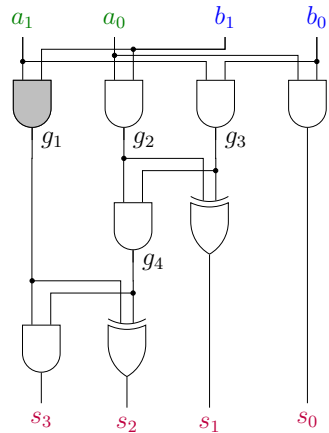
$$
\begin{aligned}
s_3 = g_1 \wedge g_4 \quad & -s_3 + g_4 g_1, \\
s_2 = g_1 \oplus g_4 \quad & -s_2 - 2g_4 g_1 + g_4 + g_1, \\
g_4 = g_2 \wedge g_3 \quad & -g_4 + g_2 g_3, \\
s_1 = g_2 \oplus g_3 \quad & -s_1 - 2g_2 g_3 + g_2 + g_3,
\end{aligned}
$$



From Circuits to Polynomials

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

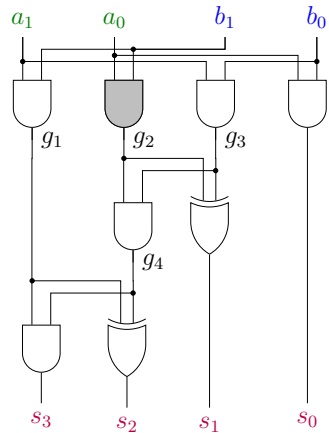
$$
\begin{aligned}
s_3 = g_1 \wedge g_4 \quad & -s_3 + g_4 g_1, \\
s_2 = g_1 \oplus g_4 \quad & -s_2 - 2g_4 g_1 + g_4 + g_1, \\
g_4 = g_2 \wedge g_3 \quad & -g_4 + g_2 g_3, \\
s_1 = g_2 \oplus g_3 \quad & -s_1 - 2g_2 g_3 + g_2 + g_3, \\
g_1 = a_1 \wedge b_1 \quad & -g_1 + a_1 b_1,
\end{aligned}
$$



From Circuits to Polynomials

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

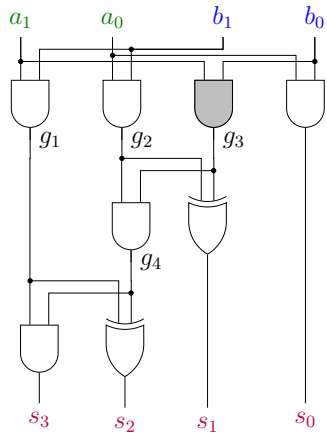
$$
\begin{aligned}
s_3 = g_1 \wedge g_4 \quad & -s_3 + g_4 g_1, \\
s_2 = g_1 \oplus g_4 \quad & -s_2 - 2g_4 g_1 + g_4 + g_1, \\
g_4 = g_2 \wedge g_3 \quad & -g_4 + g_2 g_3, \\
s_1 = g_2 \oplus g_3 \quad & -s_1 - 2g_2 g_3 + g_2 + g_3, \\
g_1 = a_1 \wedge b_1 \quad & -g_1 + a_1 b_1, \\
g_2 = a_0 \wedge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



From Circuits to Polynomials

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

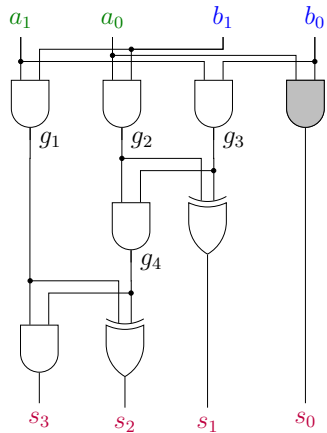
$$
\begin{aligned}
s_3 = g_1 \wedge g_4 \quad & -s_3 + g_4 g_1, \\
s_2 = g_1 \oplus g_4 \quad & -s_2 - 2g_4 g_1 + g_4 + g_1, \\
g_4 = g_2 \wedge g_3 \quad & -g_4 + g_2 g_3, \\
s_1 = g_2 \oplus g_3 \quad & -s_1 - 2g_2 g_3 + g_2 + g_3, \\
g_1 = a_1 \wedge b_1 \quad & -g_1 + a_1 b_1, \\
g_2 = a_0 \wedge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



From Circuits to Polynomials

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

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



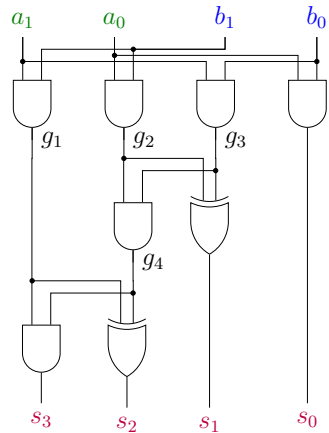
From Circuits to Polynomials

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

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

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

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



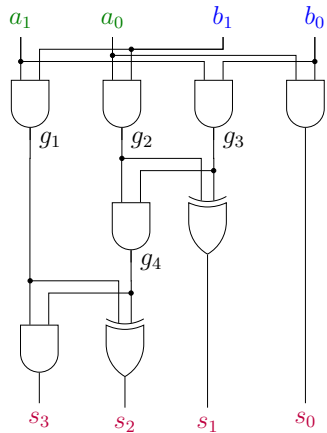
From Circuits to Polynomials

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

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

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

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



Polynomial Encoding

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

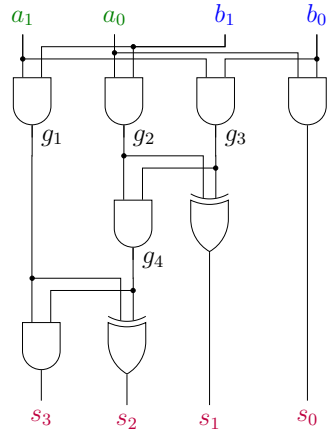
$$
\begin{array}{ll}
s_3 = g_1 \wedge g_4 & -s_3 + g_1 g_4, \\
s_2 = g_1 \oplus g_4 & -s_2 - 2g_1 g_4 + g_4 + g_1, \\
g_4 = g_2 \wedge g_3 & -g_4 + g_2 g_3, \\
s_1 = g_2 \oplus g_3 & -s_1 - 2g_2 g_3 + g_2 + g_3, \\
g_1 = a_1 \wedge b_1 & -g_1 + a_1 b_1, \\
g_2 = a_0 \wedge b_1 & -g_2 + a_0 b_1, \\
g_3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

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

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

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

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



Verification Technique

Gröbner basis

$G(C) \cup B(C)$ is a Gröbner basis w.r.t. the lexicographical ordering.

Verification Technique

Gröbner basis

$G(C) \cup B(C)$ is a Gröbner basis w.r.t. the lexicographical ordering.

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 $G(C) \cup B(C)$ based on a fixed variable order until no further reduction is possible.
Then C is a multiplier iff the final remainder is zero.

Verification Technique

Gröbner basis

$G(C) \cup B(C)$ is a Gröbner basis w.r.t. the lexicographical ordering.

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 $G(C) \cup B(C)$
based on a fixed variable order until no further reduction is possible.
Then C is a multiplier iff the final remainder is zero.

Easy: Multipliers containing a ripple-carry adder
Hard: Multipliers containing a generate-and-propagate adder, e.g., carry-lookahead adder

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

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

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

Verification

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

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

Verification

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

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

Verification

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

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

Verification

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

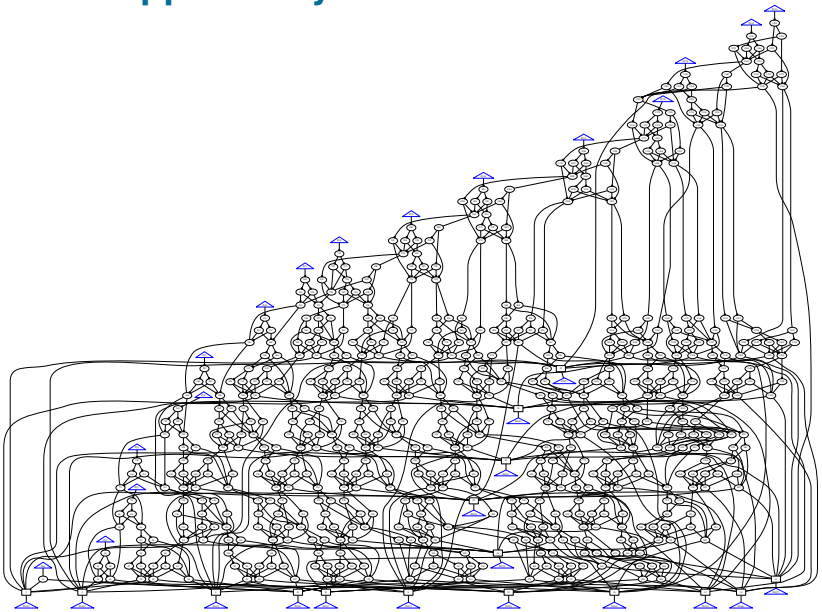
$$
\begin{aligned}
8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
8g_1 g_4 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_2 g_3 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0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

Verification

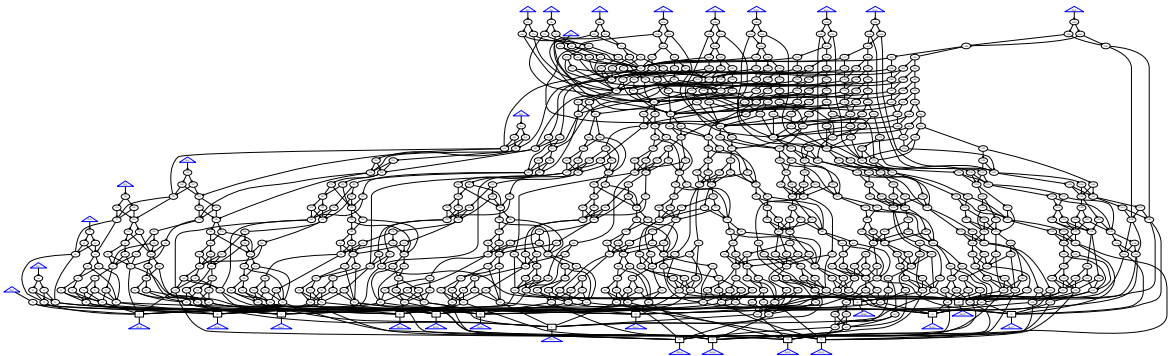
$$
\begin{aligned}
G(C) \cup B(C) = \{ & \\
& -s_3 + g_1 g_4, \\
& -s_2 - 2g_1 g_4 + g_4 + g_1, \\
& -g_4 + g_2 g_3, \\
& -s_1 - 2g_2 g_3 + g_3 + g_2, \\
& -g_1 + a_1 b_1, \\
& -g_2 + a_0 b_1, \\
& -g_3 + a_1 b_0, \\
& -s_0 + a_0 b_0, \\
& -a_1^2 + a_1, \\
& -a_0^2 + a_0, \\
& -b_1^2 + b_1, \\
& -b_0^2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

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

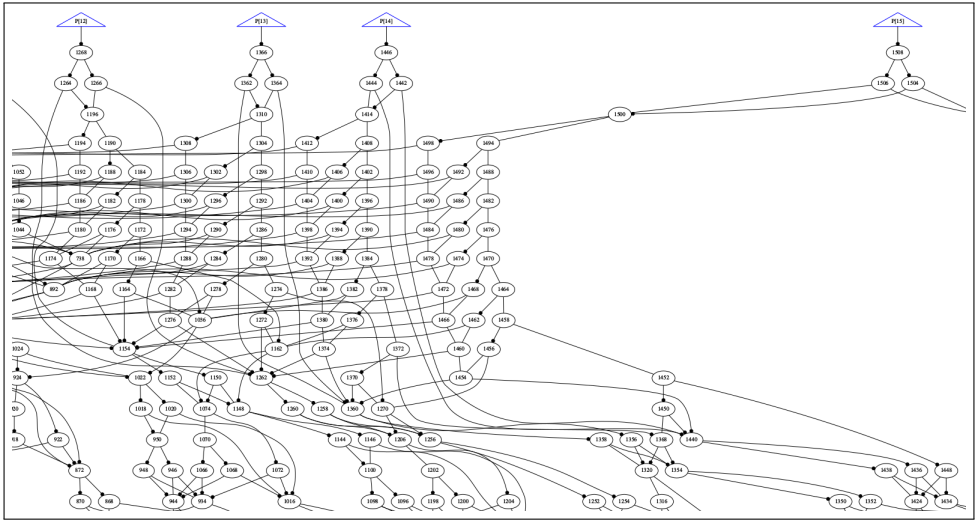
Multiplier – Ripple-Carry Adder



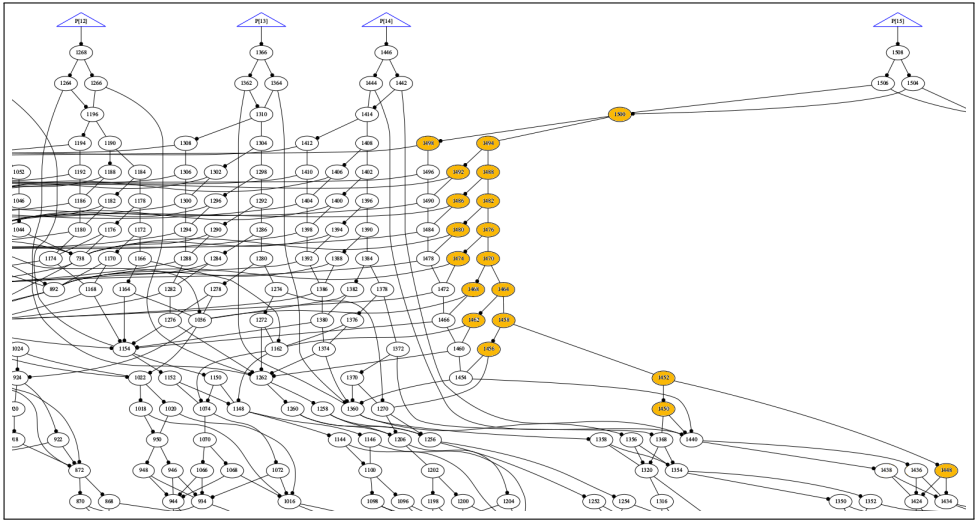
Multiplier – Carry-Lookahead Adder



Multiplier – Carry-Lookahead Adder

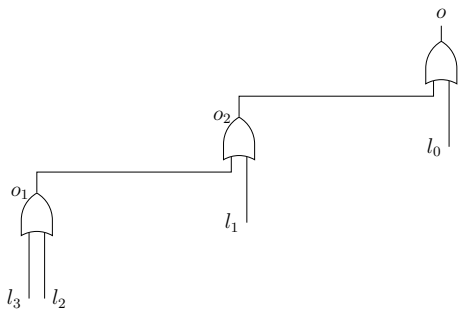


Multiplier – Carry-Lookahead Adder



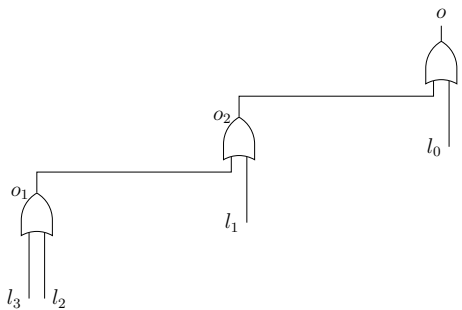
OR Gates

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



OR Gates

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

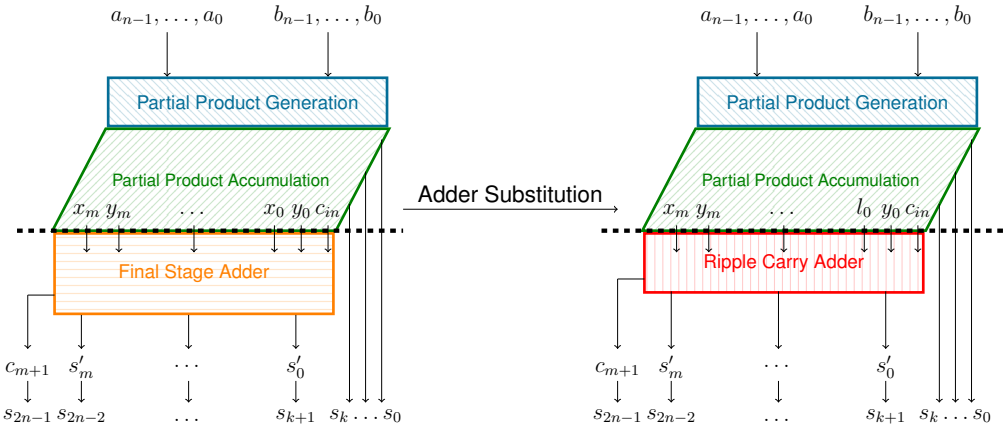


$$o = l_0 + l_1 - l_0 l_1 + l_2 - l_0 l_2 - l_1 l_2 + l_0 l_1 l_2 + l_3 - l_0 l_3 - l_1 l_3 + l_0 l_1 l_3 - l_2 l_3 + l_0 l_2 l_3 + l_1 l_2 l_3 - l_0 l_1 l_2 l_3$$

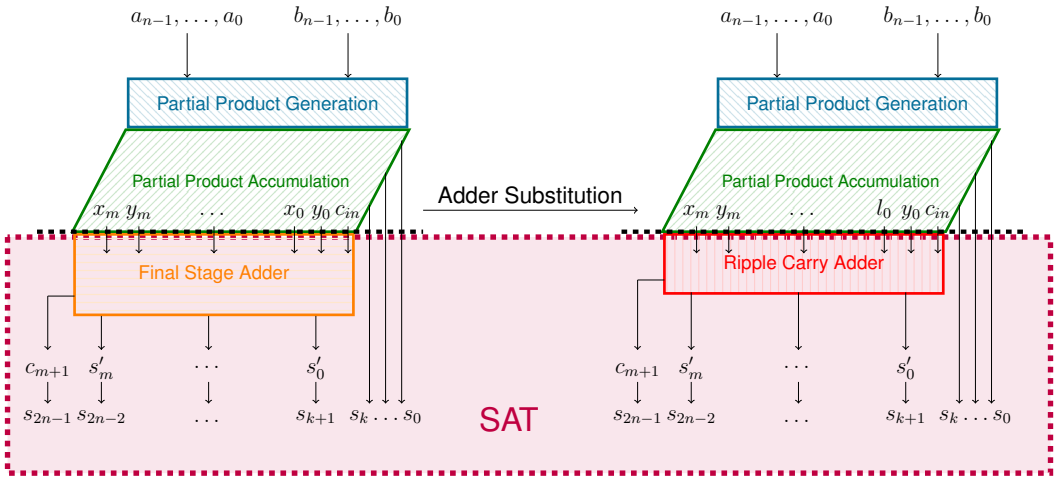
$15 = 2^4 - 1$ monomials

n OR Gates $\rightarrow 2^{n+1} - 1$ monomials

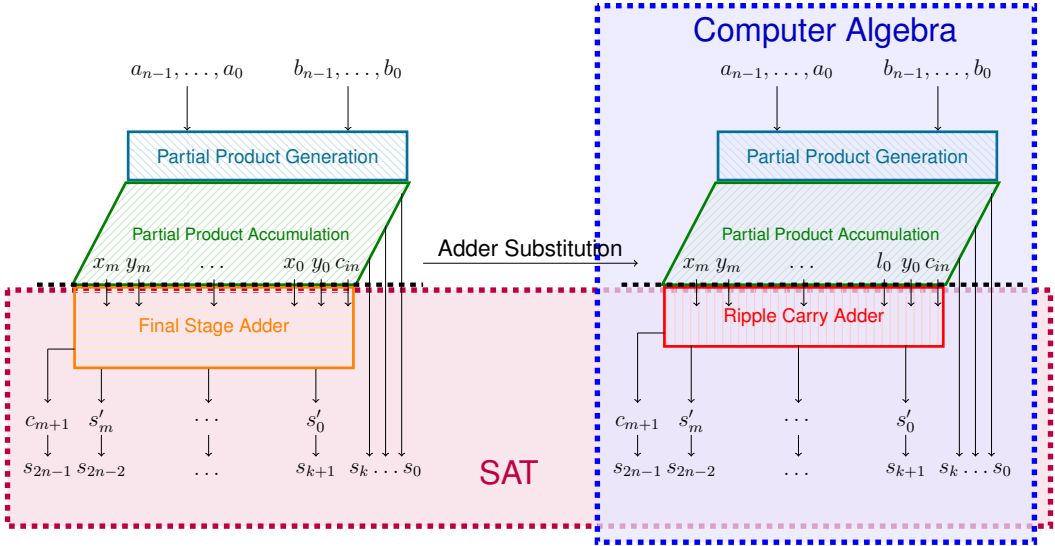
Explicit combination: SAT & Computer Algebra



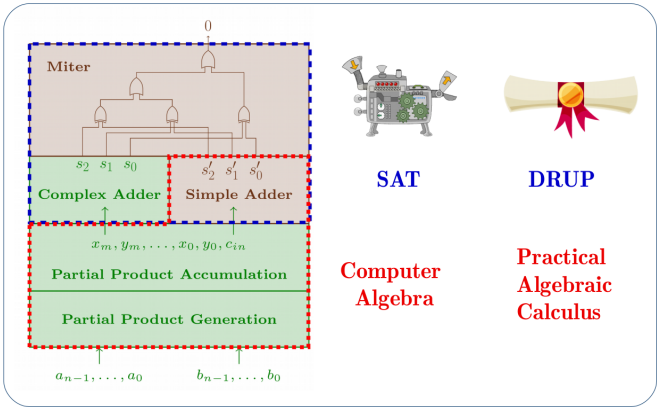
Explicit combination: SAT & Computer Algebra



Explicit combination: SAT & Computer Algebra



Problem: Proof Certificates

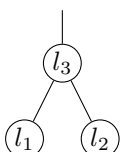


It is possible to simulate DRUP proofs in PAC, but it does not scale.

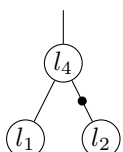
Dual Variables

[KaufmannBeameBiereNordström DATE'22]

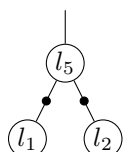
Provide a shorthand notation for inverters.



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



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



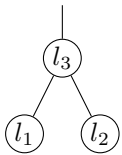
$$l_5 = \neg l_1 \wedge \neg l_2$$
$$-l_5 + l_1 l_2 - l_1 - l_2 + 1$$

Dual Variables

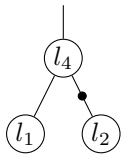
[KaufmannBeameBiereNordström DATE'22]

Provide a shorthand notation for inverters.

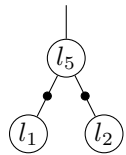
Dual variables.
Whenever two variables $l_i, f_i \in \{0, 1\}$ fulfill the relation $f_i = 1 - l_i$, we have $f_i = \mathrm{dual}(l_i)$.



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



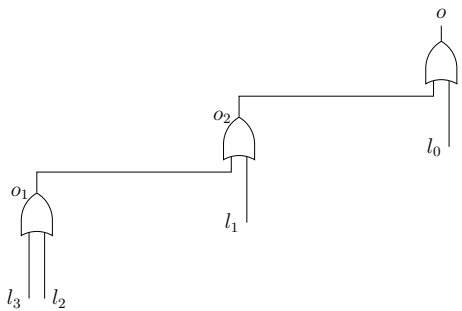
$$l_4 = l_1 \wedge \neg l_2$$
$$-l_4 - l_1 l_2 + l_1$$
$$-l_4 + l_1 f_2$$



$$l_5 = \neg l_1 \wedge \neg l_2$$
$$-l_5 + l_1 l_2 - l_1 - l_2 + 1$$
$$-l_5 + f_1 f_2$$

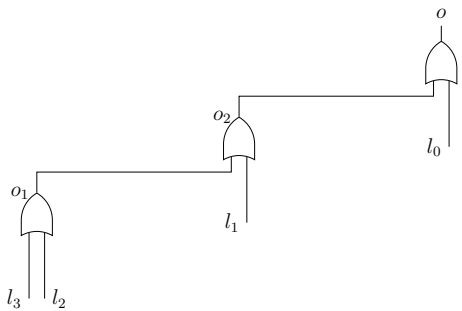
OR Gates

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



OR Gates

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



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

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

Practical Difficulty

Key Method for polynomial inference: Gröbner basis algorithm

Relies on a reduction method based on a fixed variable order that will immediately eliminate one of each pair of dual variables by re-expressing it using its partner.

Practice: During verification we always reduce the specification by the dual constraint $-f_i - l_i + 1$ of a gate variable l_i before reducing by its gate constraint. This has the effect that all occurrences of f_i in the specification will be flipped to l_i before reducing l_i.

Problem: Compact representation is unfolded.

Practical Difficulty

Key Method for polynomial inference: Gröbner basis algorithm

Relies on a reduction method based on a fixed variable order that will immediately eliminate one of each pair of dual variables by re-expressing it using its partner.

Practice: During verification we always reduce the specification by the dual constraint $-f_i - l_i + 1$ of a gate variable l_i before reducing by its gate constraint. This has the effect that all occurrences of f_i in the specification will be flipped to l_i before reducing l_i.

Problem: Compact representation is unfolded.

$\rightarrow$ We need dedicated heuristics to keep compact representation.

Calculate with Dual Variables

Proposition 1.
For all Boolean variables l_i and their dual representation $\mathrm{dual}(l_i) = f_i$ we have $l_i f_i = 0$.

"l_i and $\mathrm{dual}(l_i)$ cannot be 1 at the same time."

Proposition 2.
For all Boolean variables l_i and their dual representation $\mathrm{dual}(l_i) = f_i$ we have $l_i + f_i = 1$.

"l_i and $\mathrm{dual}(l_i)$ add up to 1."

Dual Mergeable

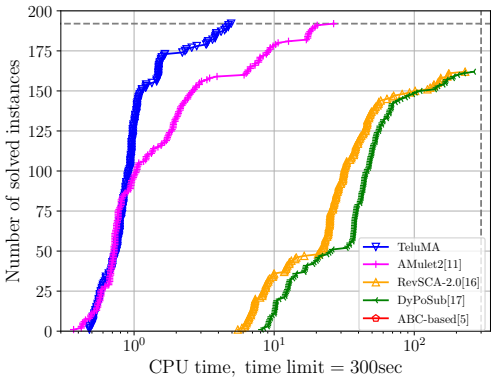
Example

Let $p = l_1 f_2 f_3 + l_1 f_2 l_3 + l_1 l_2 f_3 + f_1 f_2 + l_2 \in \mathbb{Z}[l_1, l_2, l_3, f_1, f_2, f_3]$. We write q_i to denote the polynomial q after iteration i and indicate the dual merges.

$$
\begin{aligned}
q_0 &= l_1 f_2 f_3 + l_1 f_2 l_3 + l_1 l_2 f_3 + f_1 f_2 + l_2 && r = 0 \\
q_1 &= l_1 l_2 f_3 + f_1 f_2 + \boxed{l_1 f_2} + l_2 && r = 0 \\
q_2 &= f_1 f_2 + l_1 f_2 + l_2 && r = l_1 l_2 f_3 \\
q_3 &= \boxed{f_2} + l_2 && r = l_1 l_2 f_3 \\
q_4 &= \boxed{1} && r = l_1 l_2 f_3 \\
q_5 &= 0 && r = l_1 l_2 f_3 + 1
\end{aligned}
$$

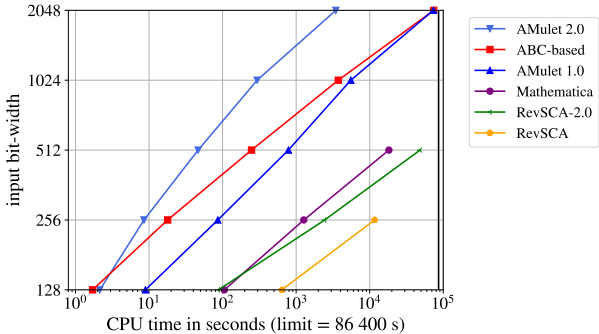
Evaluation - 64-bit Multipliers

Verification of 192 unsigned 64-bit multipliers



- **TeluMA** [KaufmannBeameBiereNordström DATE'22]
  variable elimination - incremental reduction - dual var.
- **AMulet2** [KaufmannBiere TACAS'21]
  variable elimination - incremental reduction - SAT
- **RevSCA-2.0** [MahzoonGroßeDrechsler DAC'19]
  variable elimination
- **DyPoSub** [MahzoonGroßeSchollDrechsler DATE'20]
  variable elimination - dynamic reduction

Evaluation - Large Multipliers



- Array ripple-carry multiplier
- Input bit-width n in $[128, 2048]$
- Time-limit: 86 400 sec (24h)

Conclusion

For multiplier verification using a . . .

. . . pure SAT encoding is  .

. . . pure algebraic encoding is  .

. . . combination of an algebraic and SAT encoding is  .

Conclusion

For multiplier verification using a . . .
. . . pure SAT encoding is  .
. . . pure algebraic encoding is  .
. . . combination of an algebraic and SAT encoding is  .

Future Work

- How to derive a minimal canonical encoding of polynomials using dual variables?
- How to generalize SAT concepts on polynomials?

COMBINING SAT AND COMPUTER ALGEBRA FOR MULTIPLIER VERIFICATION

Daniela Kaufmann
TU Wien, Vienna, Austria

Dagstuhl Seminar
SAT Encodings and Beyond
Dagstuhl, Germany

June 27, 2023

✉ daniela.kaufmann@tuwien.ac.at

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.

[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.

[KaufmannBiere TACAS'21] D. Kaufmann and A. Biere. AMulet 2.0 for Verifying Multiplier Circuits. Aceppted at TACAS'21, 2021.

[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.

[MahzoonGroßeScholl Drechsler 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.