COMBINING SAT AND COMPUTER ALGEBRA TO SUCCESSFULLY VERIFY LARGE MULTIPLIER CIRCUITS

Daniela Kaufmann, Armin Biere and Manuel Kauers
Johannes Kepler University
Linz, Austria

P Lunch Talk
December 6, 2019
Pittsburgh, PA, USA
Bugs in circuits are expensive!

⇒ Use formal verification to guarantee that circuits are correct.

**Challenge:** Gate-level multiplier circuits.
Multiplier circuit

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

\[3 \cdot 3 = 9\]
Multiplier circuit

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

\[3 \cdot 3 = 9\]
Multiplier circuit

\[
\begin{array}{ccc}
1 & 1 & \cdot \\
\hline
& 1 & 1 \\
1 & 1 & 1 \\
& 1 & 1 \\
\hline
1 & 0 & 0 \\
\end{array}
\]

\[3 \cdot 3 = 9\]
Multiplier circuit

**AND-Gate**

\[
f \land g = y
\]

<table>
<thead>
<tr>
<th>( f )</th>
<th>( g )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**XOR-Gate**

\[
f \oplus g = y
\]

<table>
<thead>
<tr>
<th>( f )</th>
<th>( g )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Multiplier circuit

AND-Gate

\[ f \land g = y \]

<table>
<thead>
<tr>
<th>f</th>
<th>g</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

XOR-Gate

\[ f \oplus g = y \]

<table>
<thead>
<tr>
<th>f</th>
<th>g</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Multiplier circuit

**AND-Gate**

\[ f \land g = y \]

<table>
<thead>
<tr>
<th>( f )</th>
<th>( g )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**XOR-Gate**

\[ f \oplus g = y \]

<table>
<thead>
<tr>
<th>( f )</th>
<th>( g )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

![AND-Gate Circuit](image1)

![XOR-Gate Circuit](image2)
Multiplier circuit

AND-Gate

\[ f \land g = y \]

<table>
<thead>
<tr>
<th>( f )</th>
<th>( g )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

XOR-Gate

\[ f \oplus g = y \]

<table>
<thead>
<tr>
<th>( f )</th>
<th>( g )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Multiplier circuit

**AND-Gate**

\[ f \land g = y \]

<table>
<thead>
<tr>
<th>( f )</th>
<th>( g )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**XOR-Gate**

\[ f \oplus g = y \]

<table>
<thead>
<tr>
<th>( f )</th>
<th>( g )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Multiplier circuit

**AND-Gate**

\[ f \land g = y \]

<table>
<thead>
<tr>
<th>f</th>
<th>g</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**XOR-Gate**

\[ f \oplus g = y \]

<table>
<thead>
<tr>
<th>f</th>
<th>g</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Multiplier circuit

**AND-Gate**

\[ f \land g = y \]

<table>
<thead>
<tr>
<th>( f )</th>
<th>( g )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**XOR-Gate**

\[ f \oplus g = y \]

<table>
<thead>
<tr>
<th>( f )</th>
<th>( g )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Is the circuit correct?
Multiplier circuit

\[
\begin{array}{cccc}
1 & 1 & \cdot & 1 & 1 \\
\hline
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 \\
\hline
1 & 0 & 0 & 1 \\
\end{array}
\]
Multiplier circuit

Circuit seems correct!
Is the circuit really correct?
Circuits

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

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

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

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

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

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

**Solving Techniques**

- SAT using CNF encoding
- Binary Moment Diagrams (BMD)
- Algebraic reasoning
Basic Idea of Algebraic Approach

Multiplier

Polynomials

\[ B = \{ 
\begin{align*}
  x - a_0 \cdot b_0, \\
  y - a_1 \cdot b_1, \\
  s_0 - x \cdot y, \\
  \ldots
\end{align*}
\} \]

Specification

Ideal Membership Problem

\[ \sum_{i=0}^{2n-1} \sum_{i=0}^{n-1} 2^i \cdot s_i - (\sum_{i=0}^{n-1} 2^i \cdot a_i) \cdot (\sum_{i=0}^{n-1} 2^i \cdot b_i) = 0 \checkmark \]

\[ \neq 0 \times \]
Recent Work on Algebraic Circuit Verification

- D. Kaufmann, A. Biere, and M. Kauers. Verifying Large Multipliers by Combining SAT and Computer Algebra. FMCAD, Oct 2019
- D. Kaufmann, A. Biere, and M. Kauers. Incremental Column-wise verification of arithmetic circuits using computer algebra. FMSD, Feb 2019

Content

1. Modular Reasoning
2. Combine SAT and Computer Algebra
3. Preprocessing Techniques
4. Tool: AMULET
5. Experiments
Multiplier Specification

Unsigned integers:

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

Unsigned integers:

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

Signed integers:

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

Unsigned integers:

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

Signed integers:

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

Truncated multiplication of integers:

\[ T_n = \sum_{i=0}^{2n-1} 2^i s_i - \left( \sum_{i=0}^{n-1} 2^i a_i \right) \left( \sum_{i=0}^{n-1} 2^i b_i \right) \in \mathbb{Z}_{2^n}[X] \]
From Circuits to Polynomials

**AND-Gate** \( f \land g = y \)

\[
\begin{array}{c|cc|c}
 & f & g & y \\
\hline
0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 \\
\end{array}
\]

**XOR-Gate** \( f \oplus g = y \)

\[
\begin{array}{c|cc|c}
 & f & g & y \\
\hline
0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 \\
\end{array}
\]

![Logic Gates Diagram](image)
From Circuits to Polynomials

**AND-Gate**  \[ f \land g = y \]

\[
\begin{array}{ccc}
  f & g & y \\
  0 & 0 & 0 \\
  0 & 1 & 0 \\
  1 & 0 & 0 \\
  1 & 1 & 1 \\
\end{array}
\]

\[ y = fg \]

**XOR-Gate**  \[ f \oplus g = y \]

\[
\begin{array}{ccc}
  f & g & y \\
  0 & 0 & 0 \\
  0 & 1 & 1 \\
  1 & 0 & 1 \\
  1 & 1 & 0 \\
\end{array}
\]

\[ y = f + g - 2fg \]
From Circuits to Polynomials

**AND-Gate**

\[ f \land g = y \]

<table>
<thead>
<tr>
<th>( f )</th>
<th>( g )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ 0 = -y + fg \]

**XOR-Gate**

\[ f \oplus g = y \]

<table>
<thead>
<tr>
<th>( f )</th>
<th>( g )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ 0 = -y + f + g - 2fg \]
From Circuits to Polynomials

**AND-Gate**

\[ f \land g = y \]

<table>
<thead>
<tr>
<th>f</th>
<th>g</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[-y + fg\]

**XOR-Gate**

\[ f \oplus g = y \]

<table>
<thead>
<tr>
<th>f</th>
<th>g</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[-y + f + g - 2fg\]
Gate polynomials $G(C)$.

$s_3 = g_1 \land g_4 \quad -s_3 + g_1 g_4,$
Gate polynomials $G(C)$.

\[
\begin{align*}
s_3 &= g_1 \land g_4 \quad - s_3 + g_1 g_4, \\
s_2 &= g_1 \oplus g_4 \quad - s_2 + g_1 + g_4 - 2g_1 g_4,
\end{align*}
\]
Gate polynomials $G(C)$.

$s_3 = g_1 \land g_4$ \quad $- s_3 + g_1 g_4$,
$s_2 = g_1 \oplus g_4$ \quad $- s_2 + g_1 + g_4 - 2g_1 g_4$,
$g_4 = g_2 \land g_3$ \quad $- g_4 + g_2 g_3$,
From Circuits to Polynomials

**Gate polynomials** $G(C)$.

$s_3 = g_1 \land g_4$  \quad - $s_3 + g_1 g_4$,
$s_2 = g_1 \oplus g_4$  \quad - $s_2 + g_1 + g_4 - 2g_1 g_4$,
$g_4 = g_2 \land g_3$  \quad - $g_4 + g_2 g_3$,
$s_1 = g_2 \oplus g_3$  \quad - $s_1 + g_2 + g_3 - 2g_2 g_3$,
From Circuits to Polynomials

Gate polynomials $G(C)$.

$s_3 = g_1 \land g_4$ \hspace{1cm} - $s_3 + g_1 g_4,$
$s_2 = g_1 \oplus g_4$ \hspace{1cm} - $s_2 + g_1 + g_4 - 2g_1 g_4,$
$g_4 = g_2 \land g_3$ \hspace{1cm} - $g_4 + g_2 g_3,$
$s_1 = g_2 \oplus g_3$ \hspace{1cm} - $s_1 + g_2 + g_3 - 2g_2 g_3,$
$g_1 = a_1 \land b_1$ \hspace{1cm} - $g_1 + a_1 b_1,$
Gate polynomials $G(C)$.

$$s_3 = g_1 \land g_4 \quad - s_3 + g_1 g_4,$$
$$s_2 = g_1 \oplus g_4 \quad - s_2 + g_1 + g_4 - 2g_1 g_4,$$
$$g_4 = g_2 \land g_3 \quad - g_4 + g_2 g_3,$$
$$s_1 = g_2 \oplus g_3 \quad - s_1 + g_2 + g_3 - 2g_2 g_3,$$
$$g_1 = a_1 \land b_1 \quad - g_1 + a_1 b_1,$$
$$g_2 = a_0 \land b_1 \quad - g_2 + a_0 b_1,$$
Gate polynomials $G(C)$.

- $s_3 = g_1 \land g_4$, $s_3 = g_1 g_4$,
- $s_2 = g_1 \oplus g_4$, $s_2 = g_1 + g_4 - 2g_1 g_4$,
- $g_4 = g_2 \land g_3$, $g_4 = g_2 g_3$,
- $s_1 = g_2 \oplus g_3$, $s_1 = g_2 + g_3 - 2g_2 g_3$,
- $g_1 = a_1 \land b_1$, $g_1 = a_1 b_1$,
- $g_2 = a_0 \land b_1$, $g_2 = a_0 b_1$,
- $g_3 = a_1 \land b_0$, $g_3 = a_1 b_0$,
Gate polynomials $G(C)$.

\[
\begin{align*}
    s_3 &= g_1 \land g_4, & - s_3 + g_1 g_4, \\
    s_2 &= g_1 \oplus g_4, & - s_2 + g_1 + g_4 - 2g_1 g_4, \\
    g_4 &= g_2 \land g_3, & - g_4 + g_2 g_3, \\
    s_1 &= g_2 \oplus g_3, & - s_1 + g_2 + g_3 - 2g_2 g_3, \\
    g_1 &= a_1 \land b_1, & - g_1 + a_1 b_1, \\
    g_2 &= a_0 \land b_1, & - g_2 + a_0 b_1, \\
    g_3 &= a_1 \land b_0, & - g_3 + a_1 b_0, \\
    s_0 &= a_0 \land b_0, & - s_0 + a_0 b_0,
\end{align*}
\]
From Circuits to Polynomials

Gate polynomials \( G(C) \).

\[
\begin{align*}
    s_3 & = g_1 \land g_4 & - & s_3 + g_1 g_4, \\
    s_2 & = g_1 \oplus g_4 & - & s_2 + g_1 + g_4 - 2g_1 g_4, \\
    g_4 & = g_2 \land g_3 & - & g_4 + g_2 g_3, \\
    s_1 & = g_2 \oplus g_3 & - & s_1 + g_2 + g_3 - 2g_2 g_3, \\
    g_1 & = a_1 \land b_1 & - & g_1 + a_1 b_1, \\
    g_2 & = a_0 \land b_1 & - & g_2 + a_0 b_1, \\
    g_3 & = a_1 \land b_0 & - & g_3 + a_1 b_0, \\
    s_0 & = a_0 \land b_0 & - & s_0 + a_0 b_0,
\end{align*}
\]

Boolean value constraints \( B_0(C) \).

\[
\begin{align*}
    a_1, a_0 & \in \mathbb{B} & a_1 (1 - a_1), & a_0 (1 - a_0), \\
    b_1, b_0 & \in \mathbb{B} & b_1 (1 - b_1), & b_0 (1 - b_0)
\end{align*}
\]
Ideals

Ideal. Let $R$ be a ring. A nonempty subset $I \subseteq R[X]$ is called an ideal if

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

Ideal membership problem. Given a polynomial $q \in R[X]$ and a (finite) set of polynomials $P \subseteq R[X]$, decide whether $q \in \langle P \rangle$, where $\langle P \rangle$ is the smallest ideal containing all elements of $P$, also known as the ideal generated by $P$.

\[ U_n \in \langle G(C) \cup B_0(C) \rangle \subseteq \mathbb{Z}[X] \]
\[ S_n \in \langle G(C) \cup B_0(C) \rangle \subseteq \mathbb{Z}[X] \]
\[ T_n \in \langle G(C) \cup B_0(C) \rangle \subseteq \mathbb{Z}_{2n}[X] \]
Ideals

**Ideal.** Let $R$ be a ring. A nonempty subset $I \subseteq R[X]$ is called an ideal if

$$\forall p, q \in I : p + q \in I$$

and

$$\forall p \in R[X] \forall q \in I : pq \in I$$

**Ideal membership problem.** Given a polynomial $q \in R[X]$ and a (finite) set of polynomials $P \subseteq R[X]$, decide whether $q \in \langle P \rangle$, where $\langle P \rangle$ is the smallest ideal containing all elements of $P$, also known as the ideal *generated by* $P$.

**UMLT.** Let $P \subseteq R[X]$. If for a certain term order, all leading terms of $P$ only consist of a single variable with exponent 1 and are unique and further $\text{lc}(p) \in R^\times$ for all $p \in P$, then we say $P$ has *unique monic leading terms.*
Soundness and completeness

- \( P \vdash_R q \iff q \in \langle P \rangle + \langle B_0(P) \rangle \)
- \( P \models_R q \iff \forall \phi : \forall p \in P : \phi(p) = 0 \Rightarrow \phi(q) = 0 \)

**Theorem (Soundness)**

Let \( P \subseteq R[X] \) be a finite set of polynomials with UMLT and \( q \in R[X] \), then

\[ P \vdash_R q \Rightarrow P \models_R q. \]

**Theorem (Completeness)**

Let \( P \subseteq R[X] \) be a finite set of polynomials with UMLT. Then for every \( q \in R[X] \) we have

\[ P \models_R q \Rightarrow P \vdash_R q. \]
Modular reasoning

**Previous work:** \( \mathbb{Q}[X] \)
- unsigned integers
- contains \( \mathbb{Z}[X] \)
- \( \mathbb{Q} \) is a field
- Gröbner basis theory

**Now:** \( \mathbb{Z}_l[X] \) for \( l \in \mathbb{N} \)
- truncated multiplication
- elimination of monomials
Modular reasoning

Unsigned integers:

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

Signed integers:

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

Truncated multiplication of integers:

\[ T_n = \sum_{i=0}^{n-1} 2^i s_i - \sum_{i=0}^{n-1} \sum_{j=0}^{n-1-i} 2^{i+j} a_i b_j \in \mathbb{Z}_{2^n}[X] \]
Modular reasoning

**Previous work:** $\mathbb{Q}[X]$
- unsigned integers
- contains $\mathbb{Z}[X]$
- $\mathbb{Q}$ is a field
- Gröbner basis theory

**Now:** $\mathbb{Z}_l[X]$ for $l \in \mathbb{N}$
- truncated multiplication
- elimination of monomials

$\mathbb{Z}[X]$
- $\mathbb{Z}$ is a principal ideal domain
- D-Gröbner basis theory
D-Gröbner basis

- Gröbner bases theory, where coefficient domains $D$ are PIDs.
- Offers decision procedure (D-reduction) for ideal membership problem in $D[X]$.

Let $q \in D[X]$ and $P \subseteq D[X]$:

$q \in \langle P \rangle \iff q \xrightarrow{P} 0$

- UMLT property: D-reduction amounts to substitution.
- Every ideal of $D[X]$ has a D-Gröbner basis.
- There is an (expensive) algorithm which, given an arbitrary basis of an ideal, computes a D-Gröbner basis.
D-Gröbner basis applied to circuit verification

**Theorem**

Let $R$ be a PID and let $G(C) \cup B_0(C) \subseteq R[X]$ have UMLT. Then $G(C) \cup B_0(C)$ is a D-Gröbner basis of $\langle G(C) \cup B_0(C) \rangle \subseteq R[X]$. 
D-Gröbner basis applied to circuit verification

Theorem

Let $R$ be a PID and let $G(C) \cup B_0(C) \subseteq R[X]$ have UMLT. Then $G(C) \cup B_0(C)$ is a D-Gröbner basis of $\langle G(C) \cup B_0(C) \rangle \subseteq R[X]$.

- We want $R = \mathbb{Z}_l$ for $l \in \mathbb{N}$.
- $\mathbb{Z}_l$ is not a PID.
- $\mathbb{Z}$ is a PID.
D-Gröbner basis applied to circuit verification

Theorem

Let $R$ be a PID and let $G(C) \cup B_0(C) \subseteq R[X]$ have UMLT. Then $G(C) \cup B_0(C)$ is a D-Gröbner basis of $\langle G(C) \cup B_0(C) \rangle \subseteq R[X]$.

- We want $R = \mathbb{Z}_l$ for $l \in \mathbb{N}$.
- $\mathbb{Z}_l$ is not a PID.
- $\mathbb{Z}$ is a PID.

Lemma

Let $l \in \mathbb{N}$ and let $G(C) \cup B_0(C) \subseteq \mathbb{Z}[X]$ have UMLT. Then $G(C) \cup B_0(C) \cup \{l\}$ is a D-Gröbner basis of $\langle G(C) \cup B_0(C) \rangle + \langle l \rangle \subseteq \mathbb{Z}[X]$. 
Correspondence lemma

Lemma

Let $l \in \mathbb{N}$ and let $I \subseteq \mathbb{Z}[X]$ be an ideal. There is a bijective correspondence from

$$q \in I + \langle l \rangle \subseteq \mathbb{Z}[X] \quad \text{to} \quad [q] \in \{ [p] \mid p \in I \} \subseteq \mathbb{Z}[X]/\langle l \rangle,$$

where $[q]$ is the equivalence class of $q$.

Furthermore $\mathbb{Z}[X]/\langle l \rangle \cong \mathbb{Z}_l[X]$. 
Verification

Verification Algorithm

D-reduce specification polynomial

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

by elements of \(G(C) \cup B_0(C) \cup \{l\}\) until no further reduction is possible, then \(C\) is a multiplier iff remainder is zero.
Verification

Verification Algorithm

D-reduce specification polynomial \[ \sum_{i=0}^{2n-1} 2^i s_i - \left( \sum_{i=0}^{n-1} 2^i a_i \right) \left( \sum_{i=0}^{n-1} 2^i b_i \right) \] by elements of \( G(C) \cup B_0(C) \cup \{l\} \) until no further reduction is possible, then \( C \) is a multiplier iff remainder is zero.

Computational Problems

- The number of monomials in the intermediate results increases drastically.
- 8-bit multiplier can not be verified within 20 minutes.
Simple Multiplier with Ripple-Carry Adder
Complex Multiplier with Generate-and-Propagate Adder
Complex Multiplier with Generate-and-Propagate Adder
Complex Multiplier with Generate-and-Propagate Adder
OR Gates

\[ o = o_2 \lor x_0 \quad -o + o_2 + x_0 - o_2 x_0, \]
\[ o_2 = o_1 \lor x_1 \quad -o_2 + o_1 + x_1 - o_1 x_1, \]
\[ o_1 = x_3 \lor x_2 \quad -o_1 + x_3 + x_2 - x_3 x_2. \]
OR Gates

\[ o = o_2 \lor x_0 \quad -o + o_2 + x_0 - o_2 x_0, \]
\[ o_2 = o_1 \lor x_1 \quad -o_2 + o_1 + x_1 - o_1 x_1, \]
\[ o_1 = x_3 \lor x_2 \quad -o_1 + x_3 + x_2 - x_3 x_2. \]

\[ o = x_0 + x_1 - x_0 x_1 + x_2 - x_0 x_2 - x_1 x_2 + x_0 x_1 x_2 + x_3 - x_0 x_3 - x_1 x_3 + x_0 x_1 x_3 - x_2 x_3 + x_0 x_2 x_3 + x_1 x_2 x_3 - x_0 x_1 x_2 x_3 \]

\[ 15 = 2^4 - 1 \text{ monomials} \]
Adder Substitution

Partial Product Generation

Partial Product Accumulation

Final Stage Adder

Adder substitution

Partial Product Generation

Partial Product Accumulation

Ripple Carry Adder

\[ a_{n-1}, \ldots, a_0 \]

\[ b_{n-1}, \ldots, b_0 \]

\[ x_m y_m, \ldots, x_0 y_0 c_in \]

\[ c_{m+1}, s'_m, \ldots, s'_0 \]

\[ s_{2n-1}, s_{2n-2}, \ldots, s_{k+1}, s_k \ldots s_0 \]
Adder Substitution

\[ a_{n-1}, \ldots, a_0 \quad b_{n-1}, \ldots, b_0 \]

Partial Product Generation

\[ x_m y_m \quad \ldots \quad x_0 y_0 c_{in} \]

Partial Product Accumulation

\[ s_{2n-1} s_{2n-2} \quad \ldots \quad s_{k+1} s_k \ldots s_0 \]

Final Stage Adder

Adder substitution

\[ a_{n-1}, \ldots, a_0 \quad b_{n-1}, \ldots, b_0 \]

Partial Product Generation

\[ x_m y_m \quad \ldots \quad x_0 y_0 c_{in} \]

Partial Product Accumulation

\[ s_{2n-1} s_{2n-2} \quad \ldots \quad s_{k+1} s_k \ldots s_0 \]

Ripple Carry Adder

SAT
Adder Substitution

Partial Product Generation

Partial Product Accumulation

Final Stage Adder

Adder substitution

Computer Algebra

Partial Product Generation

Partial Product Accumulation

Ripple Carry Adder

SAT

\( a_{n-1}, \ldots, a_0 \)
\( b_{n-1}, \ldots, b_0 \)

\( x_m, y_m, \ldots, x_0, y_0, c_{in} \)

\( x_m, y_m, \ldots, x_0, y_0, c_{in} \)

\( c_{m+1}, s'_m, \ldots, s'_0 \)

\( s_{2n-1}, s_{2n-2}, \ldots, s_{k+1}, s_k \cdots s_0 \)

\( s_{2n-1}, s_{2n-2}, \ldots, s_{k+1}, s_k \cdots s_0 \)
Adder Substitution

\[ s'_i = p_i \oplus c_i \]

\[ p_i = x_i \oplus y_i \]

\[ c_i = (x_{i-1} \land y_{i-1}) \lor (c_{i-1} \land p_{i-1}) \]
Adder Substitution

Algorithm: Identifying GP adders in AMULET

Input: Circuit $C$ in AIG format
Output: Determine whether $C$ might contain a GP adder

1. $j \leftarrow 2n - 2$, $\tau \leftarrow 1$
2. while $\tau$ and $j \geq 0$ do
3. \hspace{1em} $\tau, c_j, p_j \leftarrow$ Check-if-XOR-and-Identify-$p_j$-and-$c_j$ ($s_j$);
4. \hspace{1em} $x_j, y_j \leftarrow$ Declare-Adder-Inputs ($p_j, \tau$);
5. \hspace{1em} $j \leftarrow j - 1$
6. end
7. $c_{in} \leftarrow c_j$
8. for $i \leftarrow j$ to $2n - 1$ do
9. \hspace{1em} $m \leftarrow$ Follow-and-Mark-Paths($s_i$);
10. end
11. return $m = 0$
AIG is translated to CNF.

CNF is given to SAT solver.

To show correctness SAT solver needs to return UNSAT.
Generated Circuit

- Partial Product Generation
- Partial Product Accumulation
- FSA
- RCA
- Miter

\[ s_2 \quad s_1 \quad s_0 \quad s'_2 \quad s'_1 \quad s'_0 \]

\[ 1 \]

\[ x_{m}, y_{m}, \ldots, x_0, y_0 \]

Partial Product Accumulation

Computer algebra

- SAT

\[ a_{n-1}, \ldots, a_0 \]

\[ b_{n-1}, \ldots, b_0 \]
Algorithm: Verification flow in AMULET

Input: Substituted circuit $C$ in AIG format
Output: Determine whether $C$ is a multiplier

1. for $i \leftarrow 0$ to $2n - 1$ do
2. \hspace{1em} $S_i \leftarrow$ Define-Cone-of-Influence($i$);
3. \hspace{1em} Order ($S_i$);
4. \hspace{1em} Search-for-Booth-Encoding ($S_i$);
5. \hspace{1em} Local-Elimination ($S_i$);
6. end
7. Global-Elimination ();
8. $C_0 \leftarrow$ Incremental-Reduction ();
9. return $C_0 = 0$
Variable Elimination

Local Elimination

- iterate over each slice
- eliminate leading variable if it only occurs in one other polynomial inside the slice
- repeat until all leading variables are either contained in other slices or occur in multiple polynomials
- subsumes “Adder-Rewriting”, “XOR-Rewriting”, “Common-Rewriting”.

Global Elimination

- eliminate marked variables found in “Search-for-Booth-Encoding()”
- need to consider all slices
Incremental Reduction

Computer Algebra System

- Too general for our purpose.

Reduction Engine: AMULET

- Designed to make use of UMLT property.
- D-reduction amounts to substitution.
- On-the-fly reduction of Boolean value constraints.
- On-the-fly generation of proof certificates in PAC format\(^1\).

---

Is the circuit really really correct?
Proofs

Problem:
- Can we trust our own implementation?

Specifications

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

Verification

\[= 0 \sqrt{\text{✓}}\]

\[\neq 0 \times\]

Correct?
Proofs

Multiplier

Polynomials

\[ B = \{ \]
\[ x - a_0 \ast b_0, \]
\[ y - a_1 \ast b_1, \]
\[ s_0 - x \ast y, \]
\[ \ldots \} \]

Verification

\[ = 0 \checkmark \]
\[ \neq 0 \times \]

Correct?

Problem:
- Can we trust our own implementation?

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

Specification

\[ \sum_{i=0}^{2n-1} 2^i s_i - \frac{1}{2} \sum_{i=0}^{n-1} 2^i a_i (\sum_{i=0}^{n-1} 2^i b_i) \]
Proofs

Practical algebraic calculus: Sequence \( P = (p_1, \ldots, p_n) \), where each \( p_k \) is obtained by:

\[
\begin{align*}
\text{d} + & : p_i, p_j, p_i + p_j; \\
& \quad p_i, p_j \text{ appearing earlier in the proof or are contained in } G(C) \\
& \quad \text{and } p_i + p_j \text{ being reduced by } B(X)
\end{align*}
\]

\[
\begin{align*}
\text{d} \ast & : p_i, q, qp_i; \\
& \quad p_i \text{ appearing earlier in the proof or is contained in } G(C) \\
& \quad \text{and } q \in R[X] \text{ being arbitrary and } qp_i \text{ being reduced by } B(X)
\end{align*}
\]

Optional deletion information \( \text{d} \).
Example - PAC

\[ G(C) = \{ -b + 1 - a, \\ -c + ab \} \]

\[ f = c \]

d * : -b+1-a, a, -a*b;
d + : -a*b, -c+a*b, -c;
d * : -c, -1, c;
Tool Flow

Verify

AMULET substitution

 AMULET verify

 CADICAL SAT solver

 x | ✓

Certify

AMULET substitution

 AMULET certify

 CADICAL SAT solver

 .cnf

 x | ✓

 x | ✓

Check

DRAT-TRIM
Proof checker

 PACTRIM
Proof checker

 .spec

 x | ✓

 x | ✓
## Verification Time

<table>
<thead>
<tr>
<th>architecture</th>
<th>$n$</th>
<th>SPEC</th>
<th>nosub</th>
<th>nomod</th>
<th>noelim</th>
<th>Verify sub</th>
<th>Verify cnf</th>
<th>Verify aig</th>
<th>Verify tot</th>
<th>MGD DAC19</th>
<th>CSYY TCAD19</th>
<th>KBK FMSD19</th>
</tr>
</thead>
<tbody>
<tr>
<td>sp-ar-rc</td>
<td>64</td>
<td>u</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>NA$_2$</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>sp-dt-lf</td>
<td>64</td>
<td>u</td>
<td>TO</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>31</td>
<td>NA$_3$</td>
<td>TO</td>
</tr>
<tr>
<td>sp-wt-cl</td>
<td>64</td>
<td>u</td>
<td>TO</td>
<td>TO</td>
<td>3</td>
<td>0</td>
<td>9</td>
<td>1</td>
<td>11</td>
<td>96</td>
<td>NA$_3$</td>
<td>TO</td>
</tr>
<tr>
<td>sp-bd-ks</td>
<td>64</td>
<td>u</td>
<td>TO</td>
<td>TO</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>162</td>
<td>NA$_3$</td>
<td>TO</td>
</tr>
<tr>
<td>sp-ar-ck</td>
<td>64</td>
<td>u</td>
<td>TO</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>143</td>
<td>NA$_3$</td>
<td>TO</td>
</tr>
<tr>
<td>bp-ar-rc</td>
<td>64</td>
<td>u</td>
<td>1</td>
<td>TO</td>
<td>118</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>53</td>
<td>NA$_3$</td>
<td>TO</td>
</tr>
<tr>
<td>bp-ct-bk</td>
<td>64</td>
<td>u</td>
<td>TO</td>
<td>TO</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>119</td>
<td>NA$_3$</td>
<td>TO</td>
</tr>
<tr>
<td>bp-os-cu</td>
<td>64</td>
<td>u</td>
<td>2</td>
<td>TO</td>
<td>TO</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>95</td>
<td>NA$_3$</td>
<td>TO</td>
</tr>
<tr>
<td>bp-wt-cs</td>
<td>64</td>
<td>u</td>
<td>1</td>
<td>TO</td>
<td>114</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>75</td>
<td>NA$_3$</td>
<td>TO</td>
</tr>
<tr>
<td>sp-ar-rc</td>
<td>64</td>
<td>s</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>NA$_1$</td>
<td>0</td>
<td>NA$_1$</td>
</tr>
<tr>
<td>bp-wt-cl</td>
<td>64</td>
<td>s</td>
<td>TO</td>
<td>3</td>
<td>109</td>
<td>0</td>
<td>10</td>
<td>1</td>
<td>11</td>
<td>NA$_1$</td>
<td>NA$_3$</td>
<td>NA$_1$</td>
</tr>
<tr>
<td>btor</td>
<td>64</td>
<td>t</td>
<td>0</td>
<td>NA$_3$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>NA$_1$</td>
<td>NA$_1$</td>
<td>NA$_1$</td>
</tr>
</tbody>
</table>

- **time in sec**
- NA$_1$: tool not applicable to type SPEC
- NA$_2$: tool not yet available
- NA$_3$: incompleteness
- TO: 3600 sec
## Verification and Certification Time

<table>
<thead>
<tr>
<th>architecture</th>
<th>$n$</th>
<th>SPEC</th>
<th>Verify</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>proof size</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>sub</td>
<td>cnf</td>
<td>aig</td>
<td>tot</td>
<td>sub</td>
<td>cnf</td>
<td>aig</td>
<td>tot</td>
<td>cnf</td>
<td>aig</td>
<td>tot</td>
<td>cnf</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sp-ar-rc</td>
<td>64</td>
<td>u</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>sp-dt-lf</td>
<td>64</td>
<td>u</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>sp-wt-cl</td>
<td>64</td>
<td>u</td>
<td>0</td>
<td>9</td>
<td>1</td>
<td>11</td>
<td>0</td>
<td>9</td>
<td>2</td>
<td>12</td>
<td>7</td>
<td>3</td>
<td>10</td>
<td>21</td>
</tr>
<tr>
<td>sp-bd-ks</td>
<td>64</td>
<td>u</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>sp-ar-ck</td>
<td>64</td>
<td>u</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>bp-ar-rc</td>
<td>64</td>
<td>u</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>bp-ct-bk</td>
<td>64</td>
<td>u</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>bp-os-cu</td>
<td>64</td>
<td>u</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>bp-wt-cs</td>
<td>64</td>
<td>u</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>sp-ar-rc</td>
<td>64</td>
<td>s</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>bp-wt-cl</td>
<td>64</td>
<td>s</td>
<td>0</td>
<td>10</td>
<td>1</td>
<td>11</td>
<td>0</td>
<td>10</td>
<td>2</td>
<td>12</td>
<td>7</td>
<td>3</td>
<td>10</td>
<td>22</td>
</tr>
<tr>
<td>btor</td>
<td>64</td>
<td>t</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

*Time in sec*
## Verification and Certification Time of Large Multipliers

<table>
<thead>
<tr>
<th>architecture</th>
<th>$n$</th>
<th>Verify</th>
<th>Certify</th>
<th>Check</th>
<th>total</th>
<th>input</th>
<th>proof size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>sub</td>
<td>cnf</td>
<td>aig</td>
<td>tot</td>
<td>sub</td>
<td>cnf</td>
</tr>
<tr>
<td>btor</td>
<td>512</td>
<td>0 0 16</td>
<td>16</td>
<td></td>
<td></td>
<td>0 0</td>
<td>23</td>
</tr>
<tr>
<td>kjvnkv</td>
<td>512</td>
<td>0 0 13</td>
<td>13</td>
<td></td>
<td></td>
<td>0 0</td>
<td>15</td>
</tr>
<tr>
<td>sp-ar-rc</td>
<td>512</td>
<td>0 0 13</td>
<td>13</td>
<td></td>
<td></td>
<td>0 0</td>
<td>16</td>
</tr>
<tr>
<td>sp-dt-If</td>
<td>512</td>
<td>1 1 25</td>
<td>26</td>
<td></td>
<td></td>
<td>0 0</td>
<td>26</td>
</tr>
<tr>
<td>sp-wt-bk</td>
<td>512</td>
<td>1 0 26</td>
<td>27</td>
<td></td>
<td></td>
<td>0 0</td>
<td>26</td>
</tr>
<tr>
<td>btor</td>
<td>1024</td>
<td>2 0 177</td>
<td>179</td>
<td></td>
<td></td>
<td>2 0</td>
<td>219</td>
</tr>
<tr>
<td>kjvnkv</td>
<td>1024</td>
<td>2 0 91</td>
<td>93</td>
<td></td>
<td></td>
<td>2 0</td>
<td>172</td>
</tr>
<tr>
<td>btor</td>
<td>2048</td>
<td>17 0 1 493</td>
<td>1 510</td>
<td></td>
<td></td>
<td>17 0</td>
<td>2 552</td>
</tr>
<tr>
<td>kjvnkv</td>
<td>2048</td>
<td>18 0 1 129</td>
<td>1 147</td>
<td></td>
<td></td>
<td>18 0</td>
<td>2 077</td>
</tr>
</tbody>
</table>

**time in min**
Conclusion

- SAT and Computer Algebra
- Adder substitution
- Polynomial reasoning over more general rings
- Modular reasoning
- AMULET
- Variable elimination
- Combination of these ideas scales up to 2048 bits
Is the circuit really really really correct?

We don’t know yet. Our proof checker is not certified (yet).
COMBINING SAT AND COMPUTER ALGEBRA
TO SUCCESSFULLY VERIFY LARGE MULTIPLIER CIRCUITS

Daniela Kaufmann, Armin Biere and Manuel Kauers
Johannes Kepler University
Linz, Austria

PLunch Talk
December 6, 2019
Pittsburgh, PA, USA