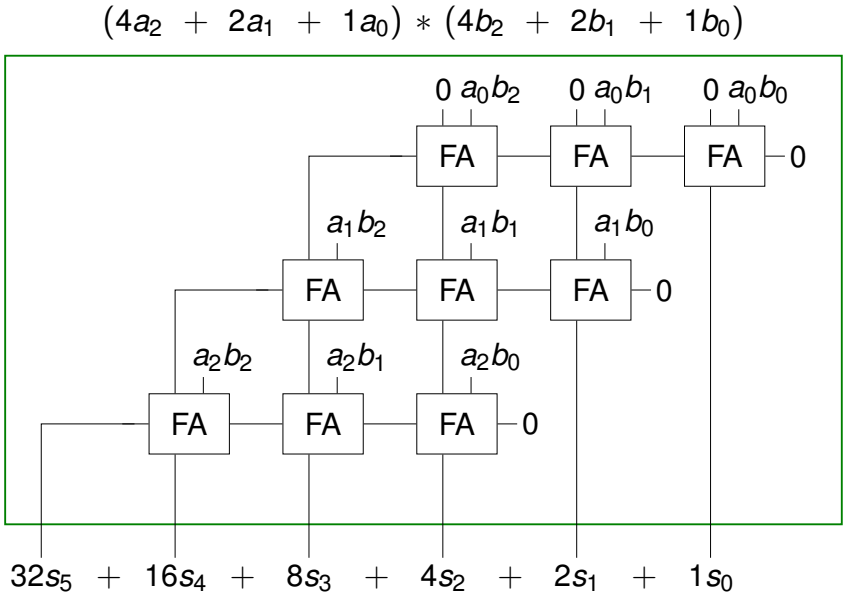
Column-Wise Verification of Multipliers Using Computer Algebra

Daniela Ritirc, Armin Biere, Manuel Kauers
Johannes Kepler University
Linz, Austria

FMCAD 2017
October 02 - 06, 2017
Vienna, Austria

Motivation & Solving Techniques

Given: a (gate level) multiplier circuit C for fixed-size bitwidth n



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

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

Motivation

- verify circuits to avoid issues like Pentium FDIV bug

Solving Techniques

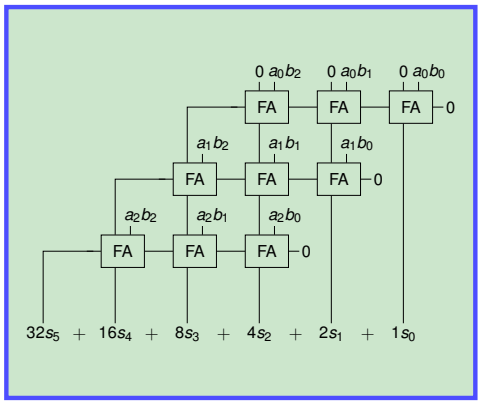
- SAT using CNF encoding
- Binary Moment Diagrams (BMD)
- Algebraic reasoning

Related Work

- **Binary moment diagrams**
  - Y.-A. Chen and R.E. Bryant. Verification of arithmetic circuits with **binary moment diagrams**. In DAC, 1995.
- **Algebraic reasoning**
  - O. Wienand, M. Wedler, D. Stoffel, W. Kunz, and G.-M. Greuel. An algebraic approach for proving data correctness in **arithmetic data paths**. In CAV, 2008.
  - J. Lv, P. Kalla, and F. Enescu. Efficient Gröbner basis reductions for formal verification of **Galois field arithmetic circuits**. In IEEE TCAD, 2013.
  - C. Yu, W. Brown, D. Liu, A. Rossi, and M. Ciesielski. Formal verification of arithmetic circuits by **function extraction**. In IEEE TCAD, 2016.
  - A.A.R. Sayed-Ahmed, D. Große, U. Kühne, M. Soeken, and R. Drechsler. Formal verification of integer multipliers by combining **Gröbner basis with logic reduction**. In DATE, 2016.
- **Proofs**
  - P. Beame and V. Liew. **Towards verifying** nonlinear integer arithmetic. In CAV, 2017.

Basic Idea of Algebraic Approach

Multiplier



Polynomials

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

Specification

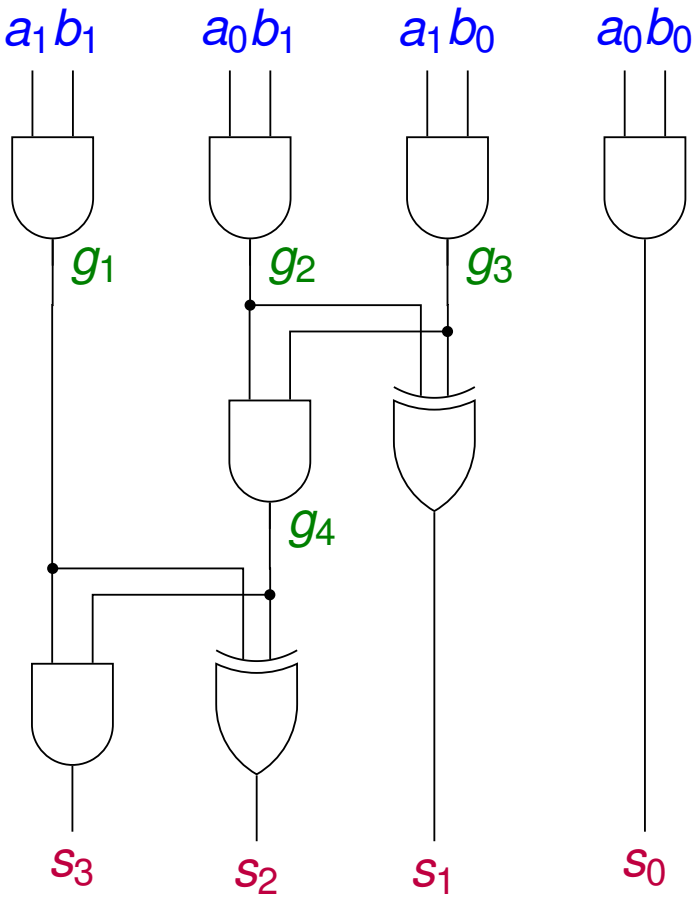
$$\sum_{i=0}^{2n-1} 2^i s_i - \\ \left(\sum_{i=0}^{n-1} 2^i a_i\right)\left(\sum_{i=0}^{n-1} 2^i b_i\right)$$

Membership Test

$= 0$ ✓

$\neq 0$ ✗

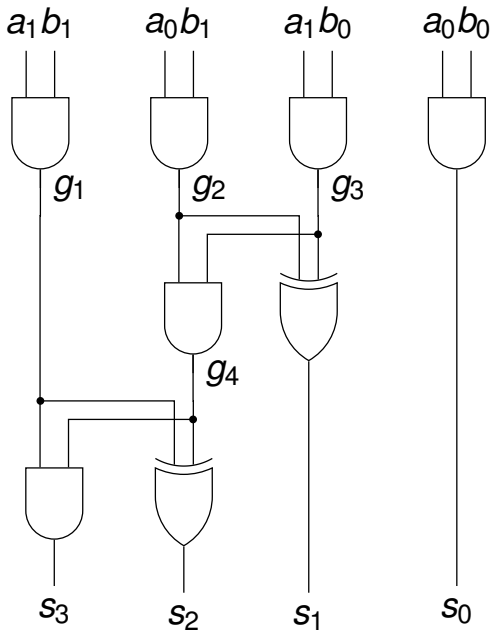
n-bit Multipliers



- inputs: $a_0, \ldots, a_{n-1}$
- inputs: $b_0, \ldots, b_{n-1}$
- outputs: $s_0, \ldots, s_{2n-1}$
- internal: $g_1, \ldots, g_k$

Circuit Polynomials

Let $X = a_0, \ldots, a_{n-1}, b_0, \ldots, b_{n-1}, g_1, \ldots, g_k, s_0, \ldots, s_{2n-1}$.



Definition 1 (Gate polynomial).
Polynomial $p \in \mathbb{Q}[X]$ representing a circuit gate.

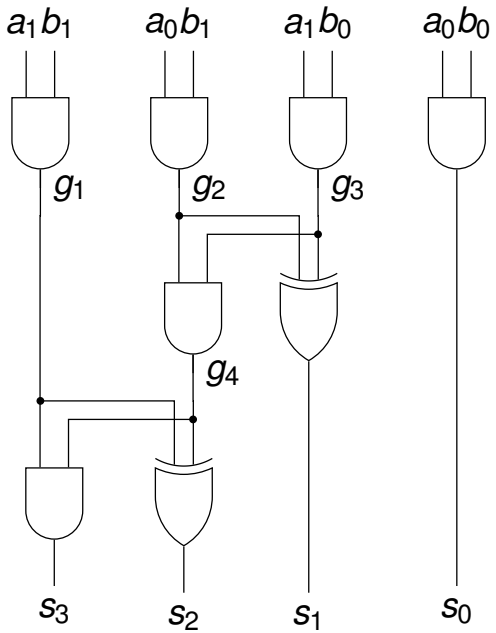
$$
\begin{array}{ll}
s_3 = g_1 \wedge g_4 & -s_3 + g_1 g_4, \\
s_2 = g_1 \oplus g_4 & -s_2 + g_1 + g_4 - 2 g_1 g_4, \\
g_4 = g_2 \wedge g_3 & -g_4 + g_2 g_3, \\
s_1 = g_2 \oplus g_3 & -s_1 + g_2 + g_3 - 2 g_2 g_3, \\
g_1 = a_1 \wedge b_1 & -g_1 + a_1 b_1, \\
g_2 = a_0 \wedge b_1 & -g_2 + a_0 b_1, \\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

Definition 2 (Field polynomial).
Polynomial $p \in \mathbb{Q}[X]$ which models the domain.

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

Circuit Polynomials

Let $X = a_0, \ldots, a_{n-1}, b_0, \ldots, b_{n-1}, g_1, \ldots, g_k, s_0, \ldots, s_{2n-1}$.



Definition 1 (Gate polynomial).
Polynomial $p \in \mathbb{Q}[X]$ representing a circuit gate.

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

Definition 2 (Field polynomial).
Polynomial $p \in \mathbb{Q}[X]$ which models the domain.

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

Ideals associated to Circuits

Definition 3 (Polynomial Circuit Constraints).
A polynomial $p \in \mathbb{Q}[X]$ is a *polynomial circuit constraint (PCC)* if for all

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

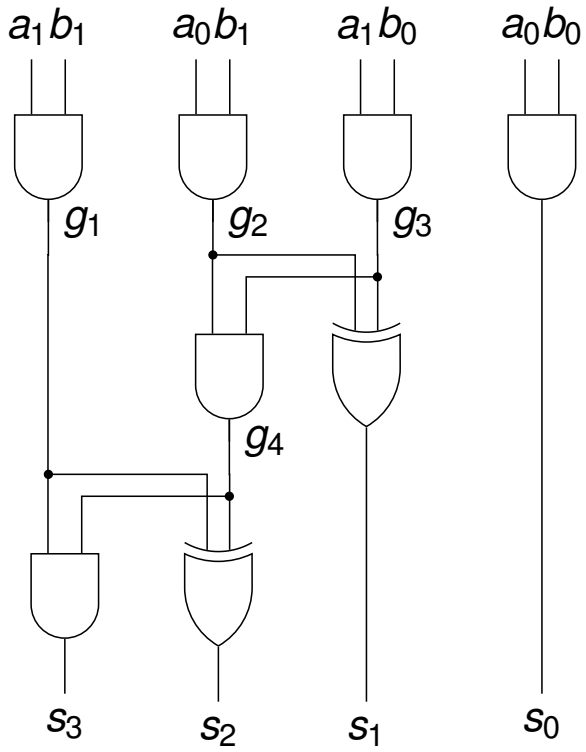
and resulting values $g_1, \ldots, g_k, s_0, \ldots, s_{2n-1}$ implied by the gates of the circuit C the substitution of these values into p gives zero.

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

Definition 4 (Ideal). A nonempty subset $I \subseteq \mathbb{Q}[X]$ is called an *ideal* if

- $\forall\, p, q \in I : p + q \in I$
- $\forall\, p \in \mathbb{Q}[X]\ \forall\, q \in I : pq \in I$

Ideals associated to Circuits



Examples for PCCs:

- $s_0 - a_0 b_0$ and gate
- $a_1^2 - a_1$ a_1 boolean
- $g_2^2 - g_2$ g_2 boolean
- $s_1 g_4$ xor-and constraint
- ...

Definition 5 (Multiplier). A circuit C is called a *multiplier* if

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

GRÖBNER BASIS

Definition 6 (Term order).
An order $\leq$ is fixed on the set of terms compatible with multiplication.

- every ideal I of $\mathbb{Q}[X]$ has a Gröbner basis G with $I = \langle G \rangle$.
- ideal membership test
  - multivariate polynomial division with remainder
  - remainder r contains no term that is a multiple of any of the leading terms of G
- construction algorithm by Buchberger which given an arbitrary basis of an ideal I computes a Gröbner basis of it (doubly exponential)

Gröbner basis

We can deduce at least some elements of $I(C)$:

- G = Gate Polynomials + (Input) Field Polynomials
- Let $J(C) = \langle G \rangle$.
- Term order: output variable of a gate is greater than input variables

Theorem

G is a Gröbner basis for $J(C)$.

Proof idea: Application of Buchberger's Product criterion.

Theorem (Soundness and completeness)

For all acyclic circuits C, we have $J(C) = I(C)$.

Non-Incremental Algorithm

Algorithm 1 (Non-Incremental Checking Algorithm).
Divide 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 until no further reduction is possible, then C is a multiplier iff remainder is zero.

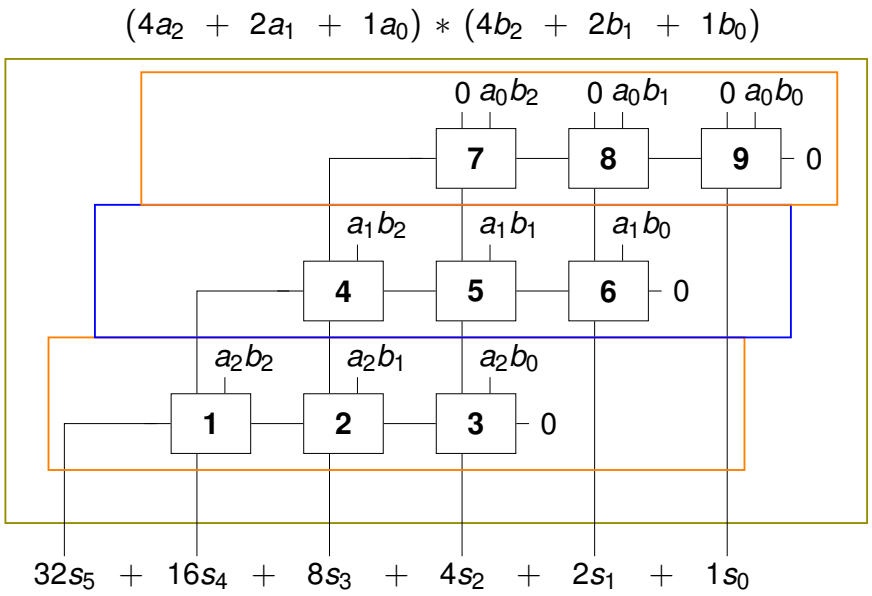
Implications

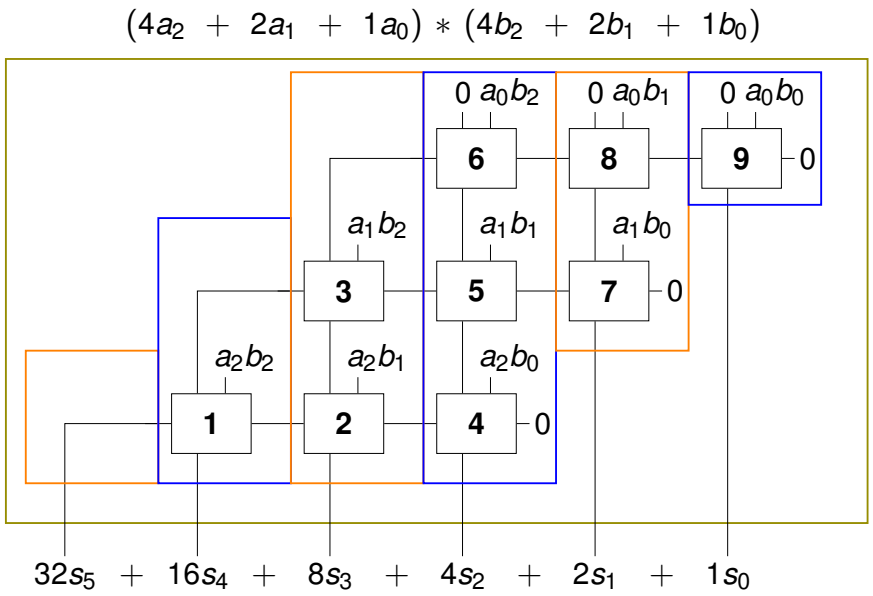
- Leading coefficient -1 of all gate polynomials, computation stays in $\mathbb{Z}$.
- Completeness proof allows to derive input assignment if C is incorrect.
- Still can use rational coefficients $\mathbb{Q}$ (important for Singular).

Generally the size of intermediate results in Algorithm 1 increases drastically:

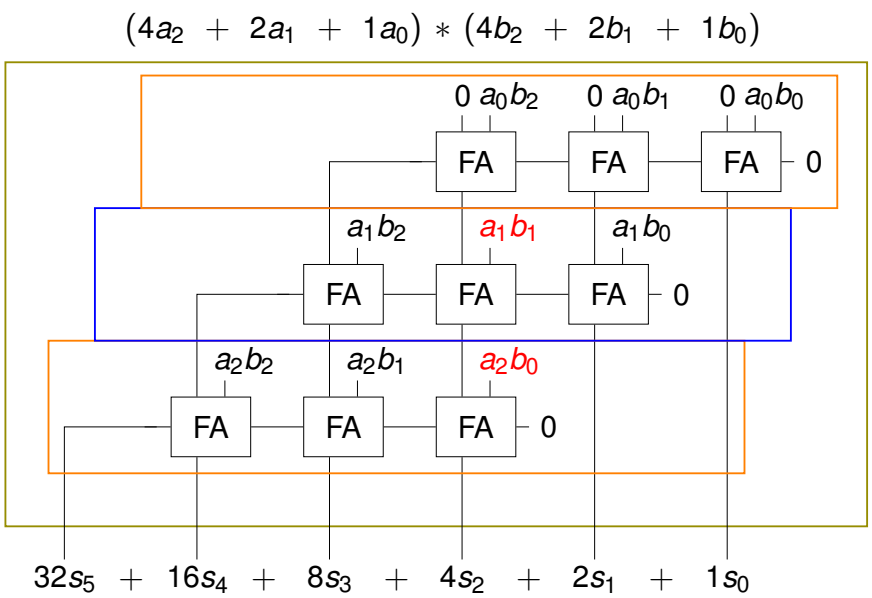
- 8-bit multiplier can not be verified within 20 minutes.
- Tailored heuristics become very important.

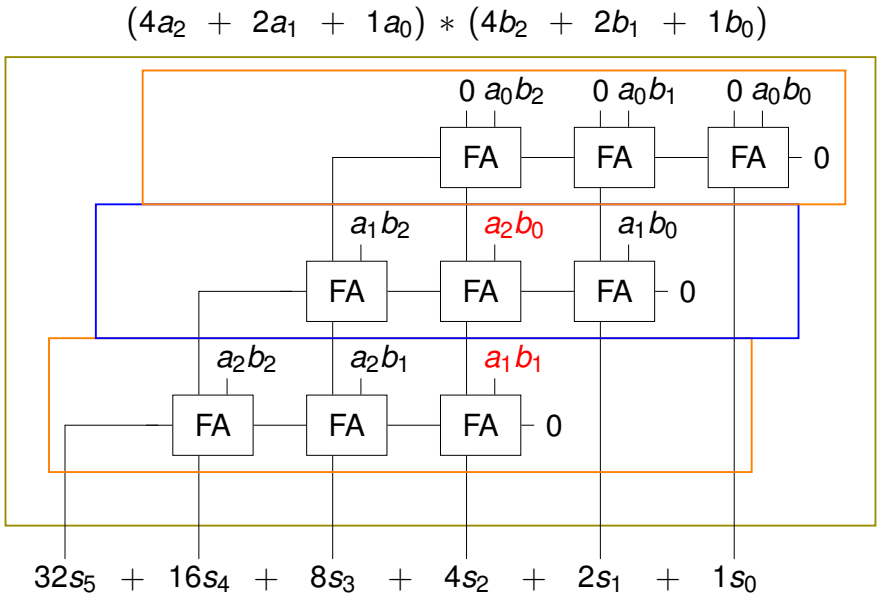
Rows and Columns





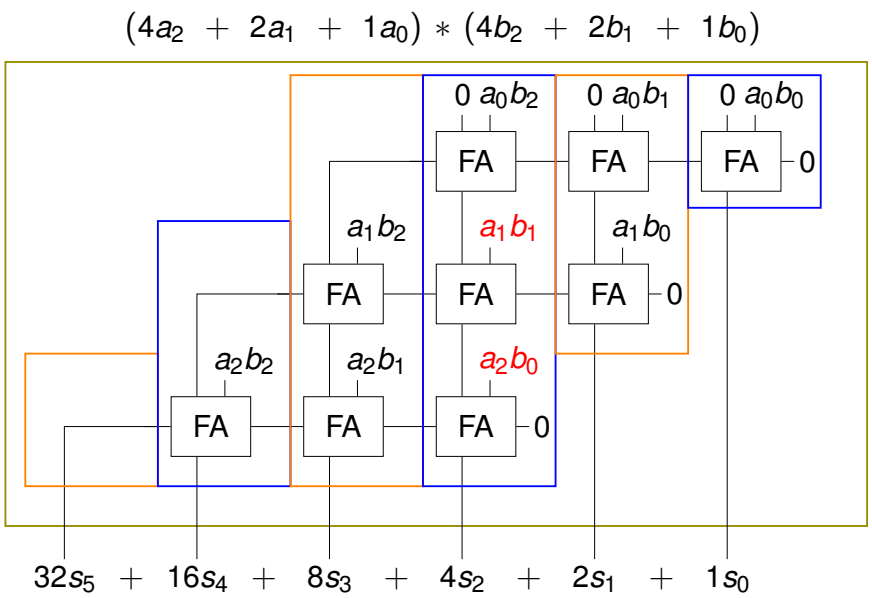
Row-Wise Order

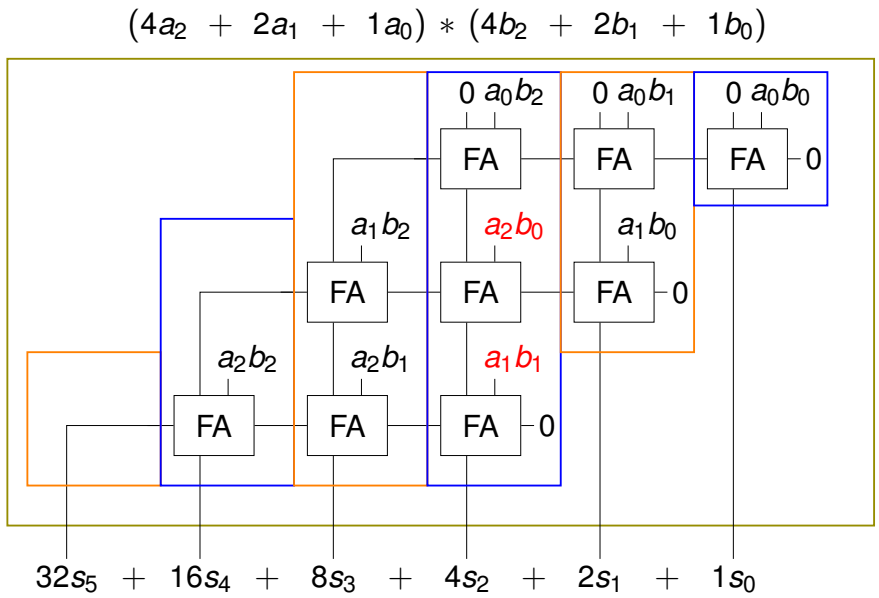




not robust under permutation

Column-Wise Order





robust under permutation

Definition 7 (Partial Products). Let $P_k = \sum_{k=i+j} a_i b_j$.

Slicing

Definition 8 (Input Cone).
For each output bit s_i we determine its input cone

$$I_i = \{\text{gate } g \mid g \text{ is in input cone of output } s_i\}$$

Definition 9 (Slice).
Slices S_i are defined as the difference of consecutive cones I_i:

$$S_0 = I_0 \qquad S_{i+1} = I_{i+1} \setminus \bigcup_{j=0}^{i} S_j$$

Definition 10 (Sliced Gröbner Bases).
Let G_i be the set of polynomial representations of the gates in slice S_i.

Carry Recurrence Relation

Definition 11 (Carry Recurrence Relation).

- A sequence of $2n+1$ polynomials $C_0, \ldots, C_{2n}$ over the variables of C is called a *carry sequence* of *carry polynomials*.
- For $0 \leq i < 2n$, carry polynomial C_i and output s_i let

$$-C_i + 2C_{i+1} + s_i - P_i$$

  denote the *carry recurrence relation* R_i for column i.
- Then R_i *holds* on C if it vanishes in $I(C)$, i.e., $R_i \in I(C)$.

Theorem

Let C be a circuit where all carry recurrence relations hold.
Then C is a multiplier in the sense of Def. 6, iff $C_0 - 2^{2n} C_{2n} \in I(C)$.

Incremental Algorithm

Algorithm 2 (Incremental Checking Algorithm).

input: Circuit C with sliced Gröbner bases G_i
output: Determine whether C is a multiplier

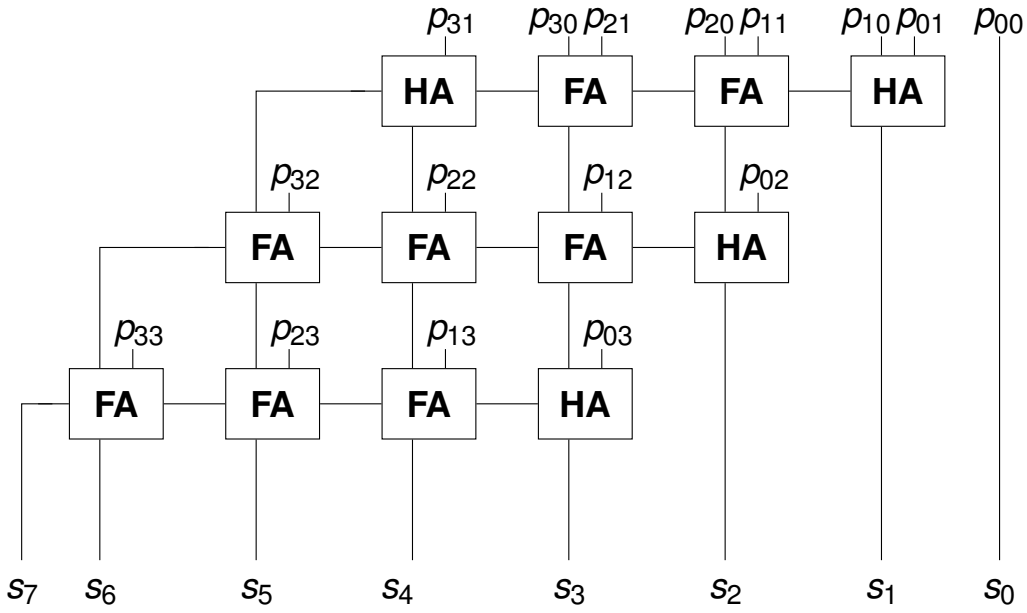
$C_{2n} \leftarrow 0$

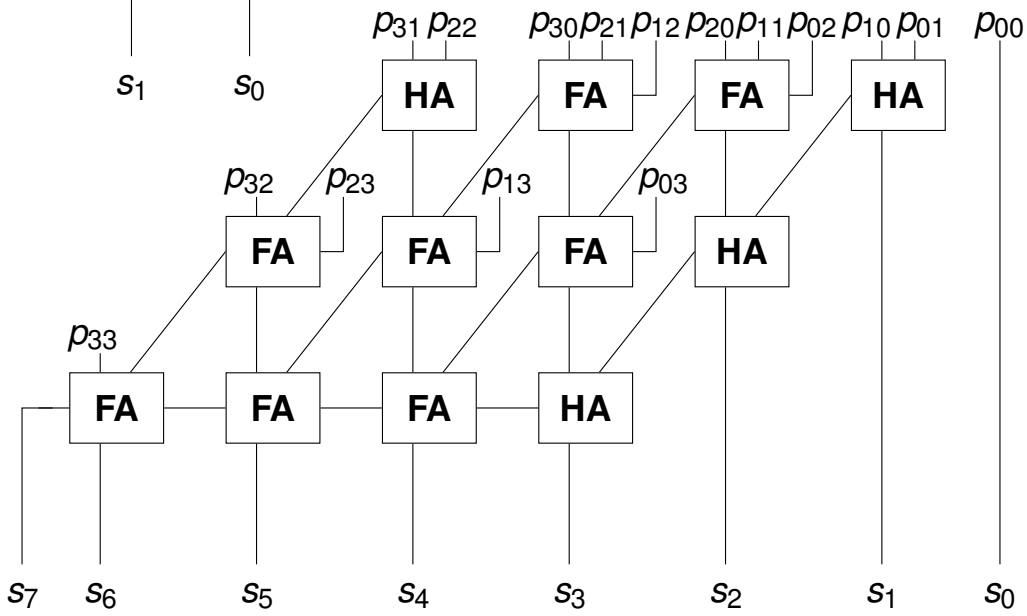
for $i \leftarrow 2n-1$ **to** 0

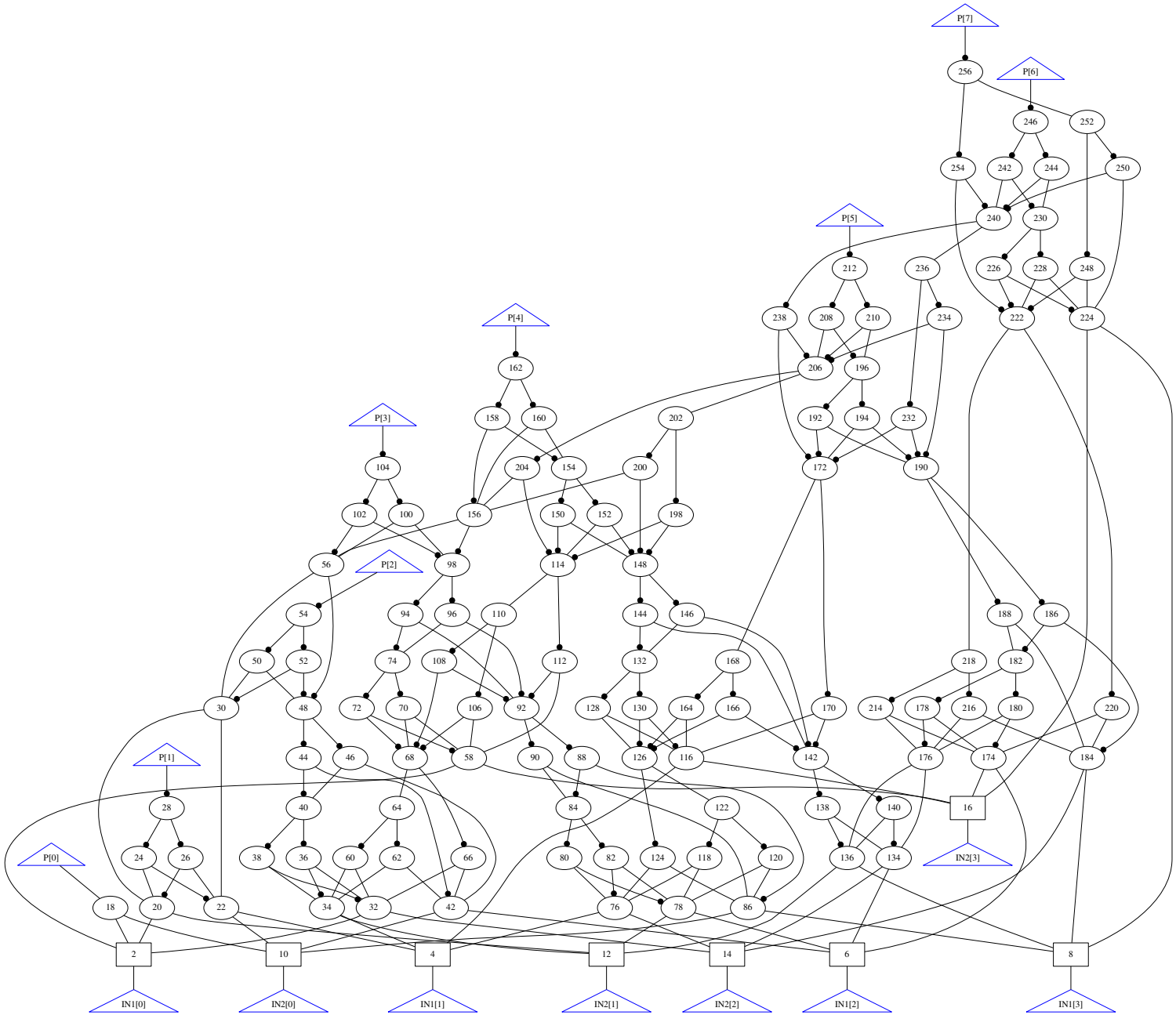
$$C_i \leftarrow \text{Remainder}\,(\,2C_{i+1} + s_i - P_i,\quad G_i \cup F\,)$$

return $C_0 = 0$

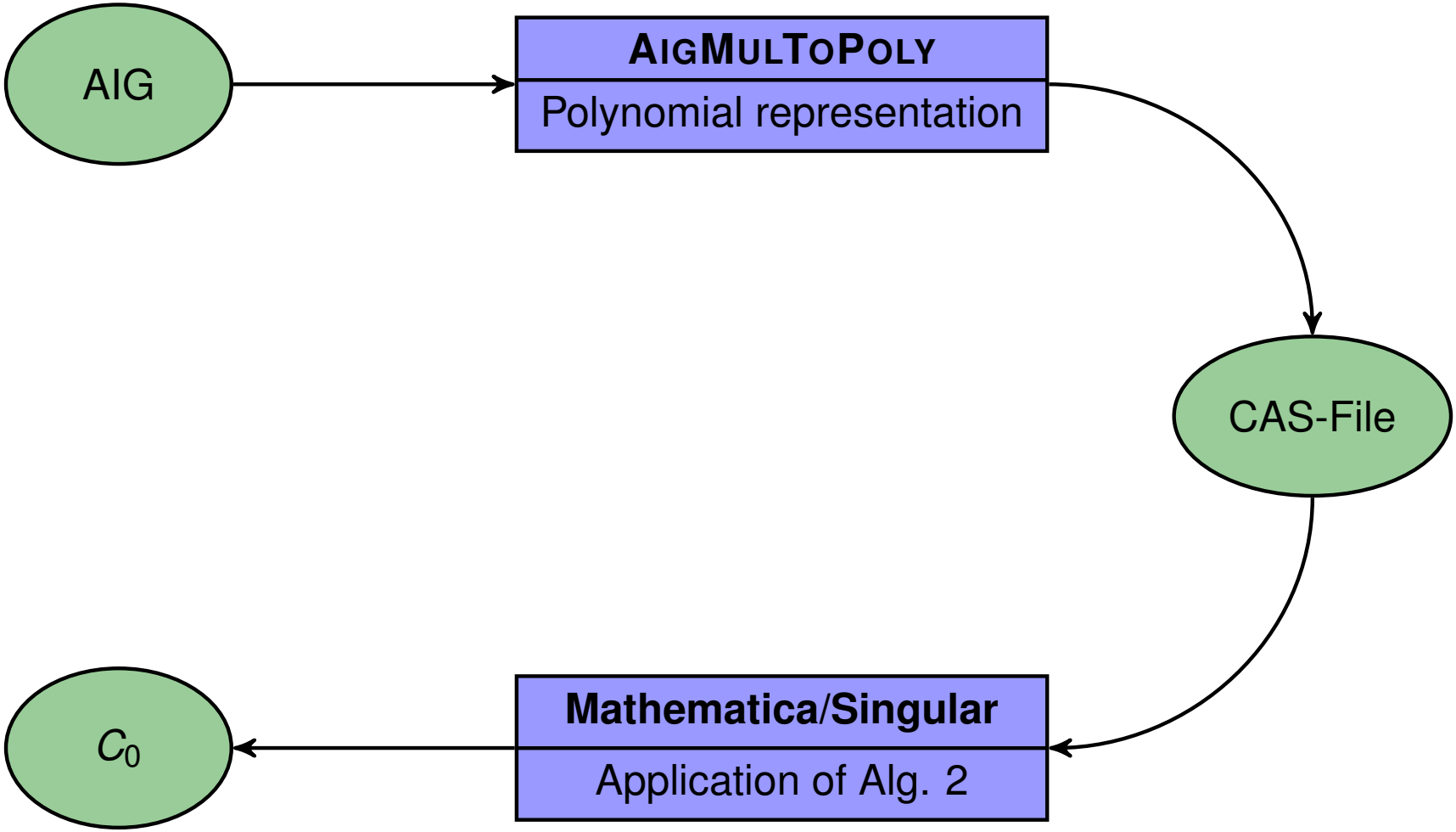
Multipliers






P[7]
256
P[6]
246
252
254
242
244
250
240
230
P[5]
212
236
226
228
248
238
208
210
234
222
224
206
196
P[4]
162
158
160
202
192
194
232
172
190
P[3]
104
204
154
200
102
100
156
150
152
198
56
P[2]
98
114
148
54
94
96
110
144
146
188
186
50
52
74
108
112
132
168
218
182
30
48
72
70
106
92
128
130
164
166
170
214
178
216
180
220
P[1]
44
46
68
58
90
88
126
116
142
176
174
184
28
40
64
84
122
138
140
16
P[0]
24
26
38
36
60
62
66
80
82
124
118
120
136
134
IN2[3]
18
20
22
34
32
42
76
78
86
2
10
4
12
14
6
8
IN1[0]
IN2[0]
IN1[1]
IN2[1]
IN2[2]
IN1[2]
IN1[3]

Engineering



Experiments

multiplier	Bit width	Mathematica			Singular		
		Alg. 1		Alg. 2	Alg. 1		Alg. 2
		col	row		col	row	
btor	16	12	12	4	2	2	1
btor	24	102	101	14	12	13	4
btor	32	531	491	35	53	58	16
btor	40	TO	TO	78	210	219	55
btor	48	TO	TO	156	602	621	145
btor	56	MO	MO	263	MO	MO	226
btor	64	MO	MO	409	MO	MO	MO
sp-ar-rc	8	TO	TO	2	TO	TO	1
sp-ar-rc	16	TO	TO	7	TO	TO	1
sp-ar-rc	32	TO	TO	67	TO	TO	39
sp-ar-rc	64	MO	MO	841	MO	MO	MO

TABLE: time in sec; TO = 1200 sec, MO = 14 GB

Conclusion & Future Work

Conclusion:

- simple and precise mathematical formulation
- new incremental column-based verification approach
- magnitude faster than previous non-incremental approach
- using computer algebra systems

Future Work:

- other word-level operators (shift, division, . . .)
- more complex multipliers
- extend our methods to floating points
- negative numbers

Experimental data, source code, benchmarks, and scripts are available at
`http://fmv.jku.at/cwmulverca`.

Column-Wise Verification of Multipliers Using Computer Algebra

Daniela Ritirc, Armin Biere, Manuel Kauers
Johannes Kepler University
Linz, Austria

FMCAD 2017
October 02 - 06, 2017
Vienna, Austria