Progressive Decomposition: A Heuristic to Structure Arithmetic Circuits

Ajay K. Verma, Philip Brisk and **Paolo Ienne**

Processor Architecture Laboratory (LAP)
& Centre for Advanced Digital Systems (CSDA)

Ecole Polytechnique Fédérale de Lausanne (EPFL)

Logic Synthesis: Unable to Impose Hierarchy and Structure

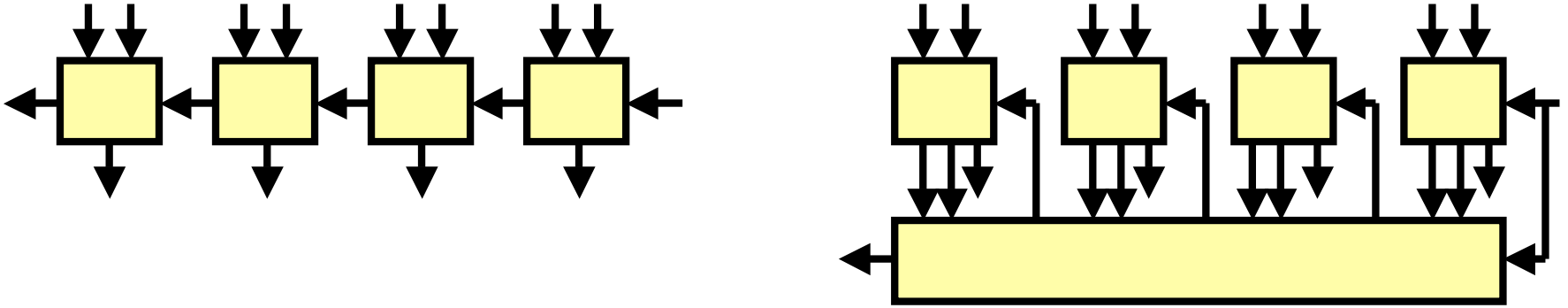
- Logic synthesis tools
  - Local optimization via Boolean minimization
  - Lacking for arithmetic circuits

$$ab(a + b) + ca + \bar{c} \longrightarrow a + \bar{c}$$

- Architectural transformation
  - Not with logic synthesis

Our contribution

Ripple-Carry Adder ⟶ Carry-Lookahead Adder



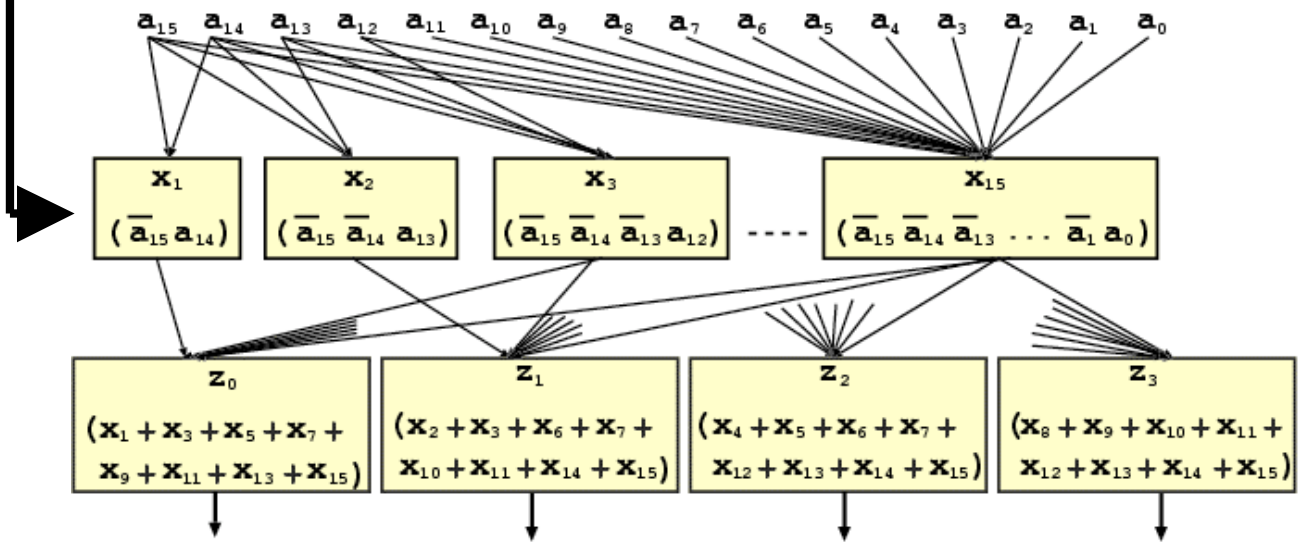
Leading Zero Detector (LZD)

- Finds the position of the most significant non-zero bit

$$x_i = \overline{a}_{15}\overline{a}_{14} \ldots \overline{a_{15-(i-2)}}\,\overline{a_{15-(i-1)}}\,a_{15-i}$$

x_i is TRUE if $(i+1)^{th}$ most-significant bit is the leading non-zero bit

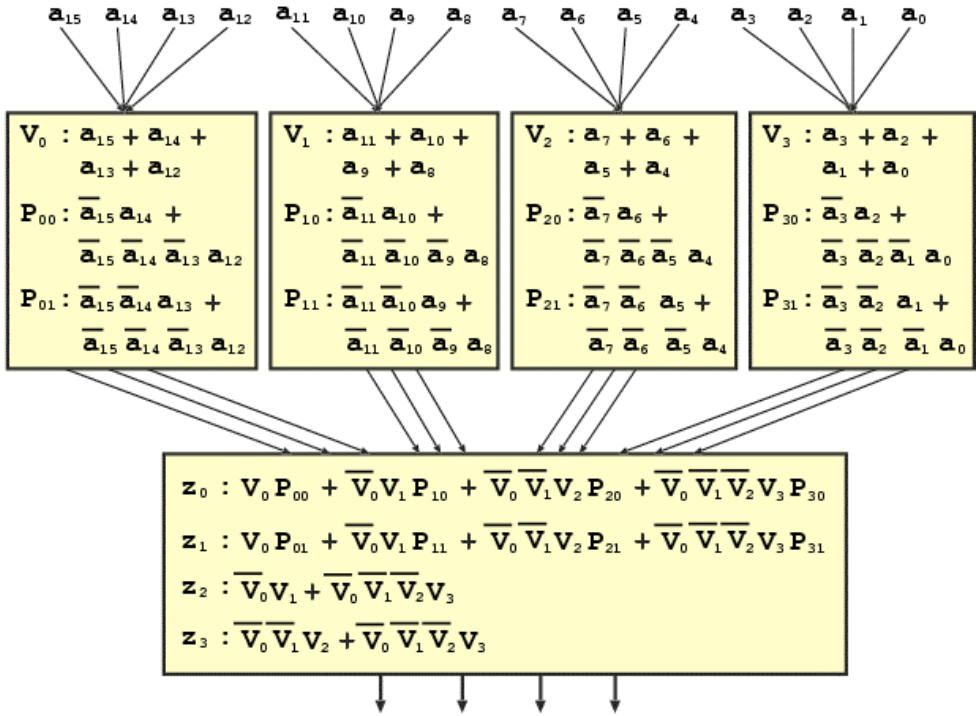
Large fan-in dependencies



Convert x_i to a binary number

LZD: A Better Implementation [Oklobdzija94]

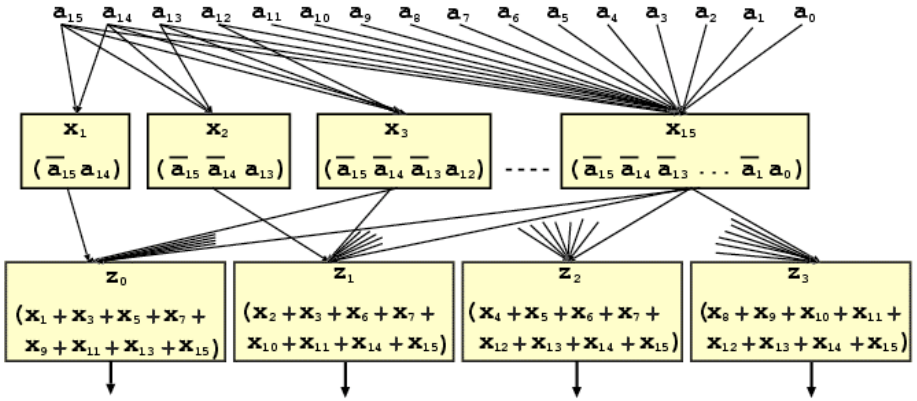
- Divide 16 bits into 4 blocks
  - For each block compute the following:
    - Position of the most significant non-zero bit
    - Control bit V_i is TRUE if at least one bit in this block is one



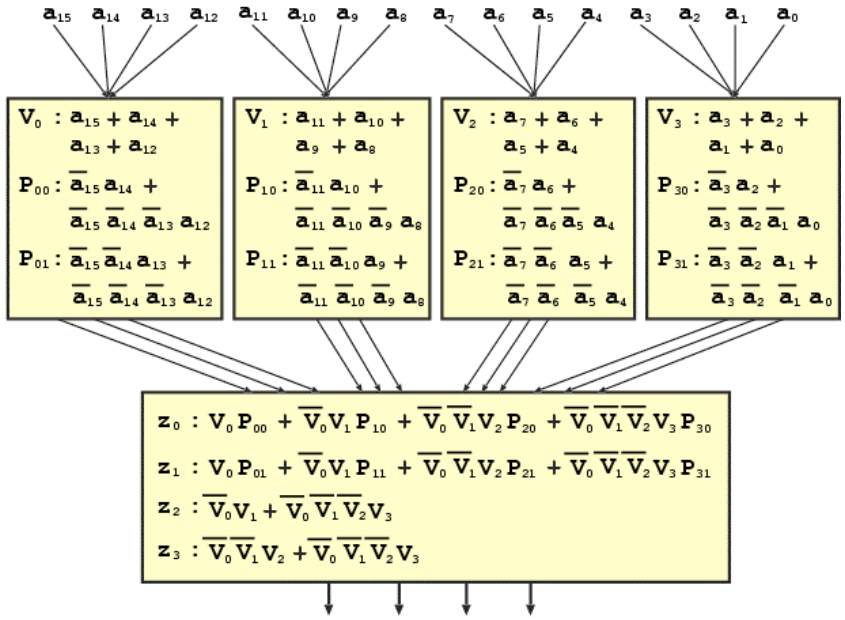
- Similar in principle to carry-lookahead addition

Reduced fan-in dependencies

What is the Bottom Line?



0.36 ns
(426.8 μm²)



0.30 ns
(392.3 μm²)

16% faster, 8% smaller

Outline

- Related Work
- Architectural optimization
  - How to impose hierarchy?
- Properties of Algebra
  - Ring structure of Boolean expressions
- Progressive Decomposition Algorithm
- Results
- Conclusions

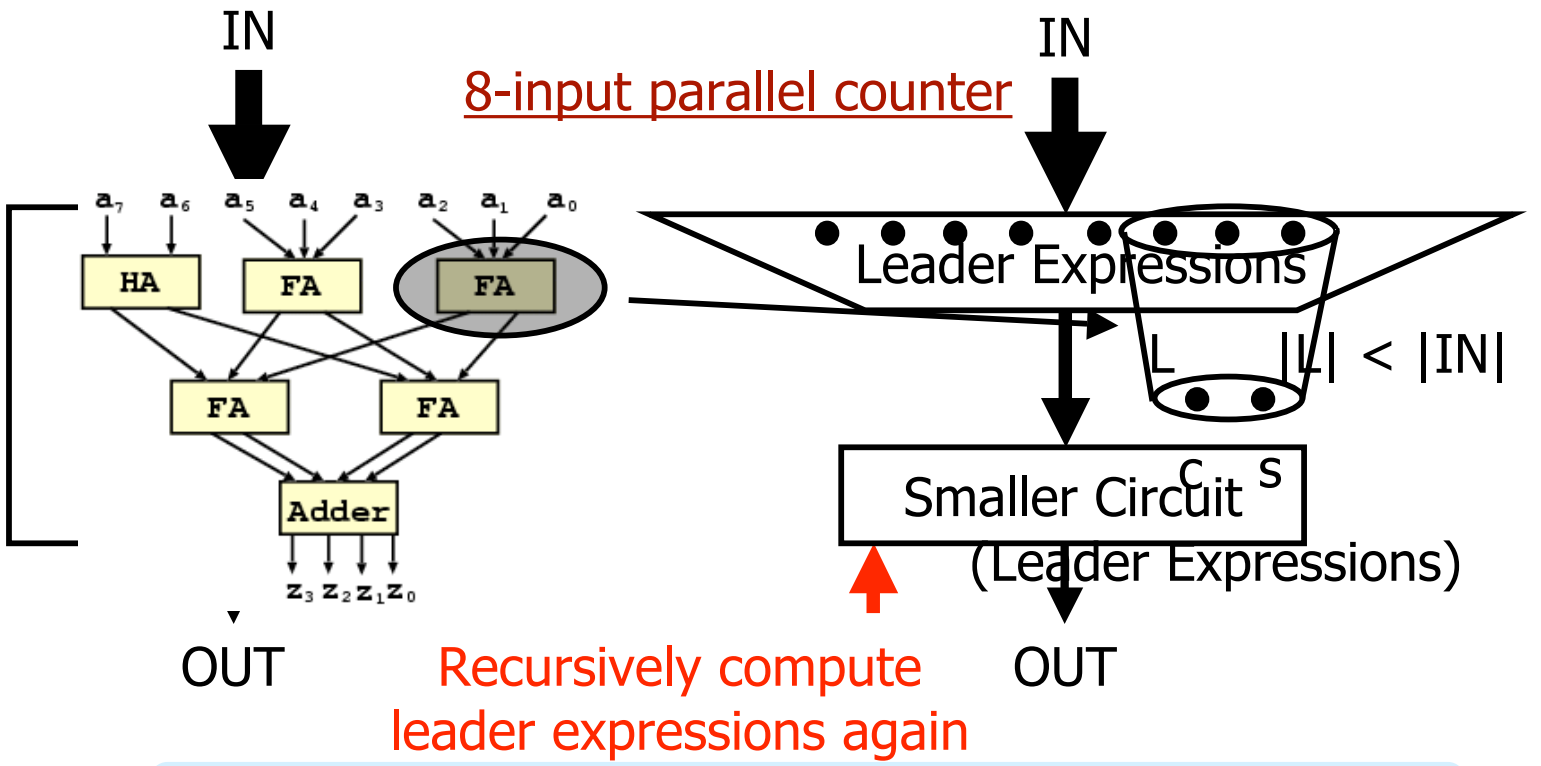
Related Work

- Manual approaches for optimizing circuits of interest
  - The entire field of computer arithmetic
    - Great ideas by really smart people!
- Algorithmic approaches for a particular class of circuits
  - Variable group size CLA adder [Lee91]
  - Irregular partial product compressors [Stelling98]
- Heuristics to optimize general classes of circuits
  - Kernel and co-kernel extraction [Brayton82]
  - Architecture exploration via exhaustive search [Verma06]

Input Condensation

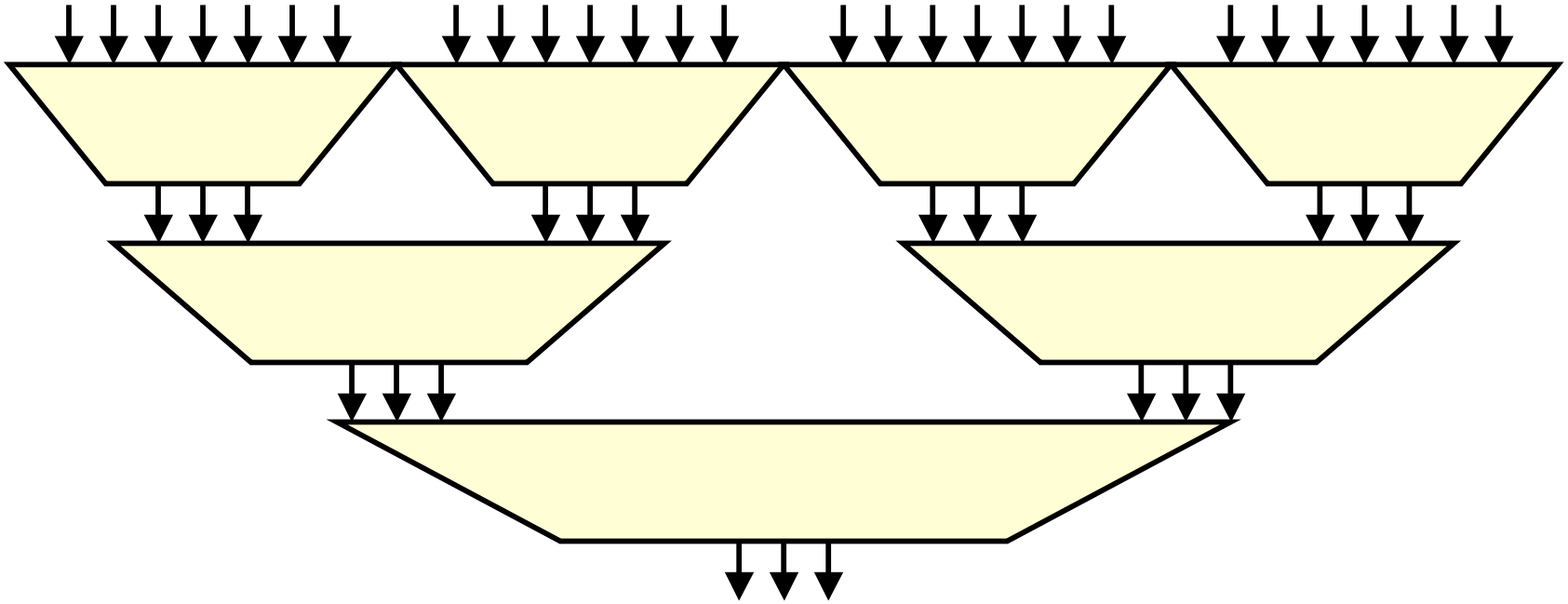
Leader expressions:

- Sufficient to evaluate the whole of an expression
- Once you evaluate them, you can discard the input bits



Compute all leader expressions in parallel

Hierarchical Circuit Construction



Theorem: This approach always generalizes to circuits that have an "effective inductive algorithm"

Use regular expressions as building blocks to impose hierarchy

Reed-Muller Form

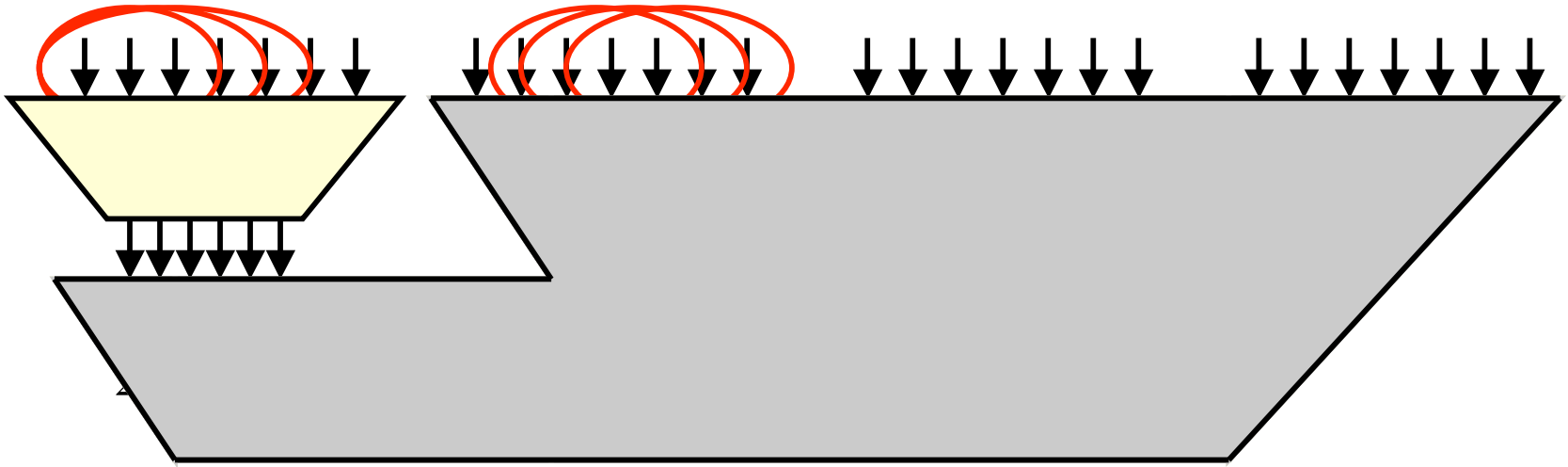
- XOR-of-Product Form
  - Better suits arithmetic circuits
  - Forms a ring under the operations XOR and AND
  - Boolean properties exploited by our algorithm
    - Identities
    - Null Spaces
    - Linear Dependence
- Before

$$X = [a_1 b_1 + (a_1 + b_1) a_0 b_0] \oplus [(a_1 \oplus b_1 \oplus a_0 b_0) c_1 + c_0 (a_0 \oplus b_0)(c_1 + (a_1 \oplus b_1 \oplus a_0 b_0))]$$

- After

$$X = a_1 b_1 \oplus a_0 a_1 b_0 \oplus a_0 b_0 b_1 \oplus a_1 c_1 \oplus b_1 c_1 \oplus a_0 b_0 c_1 \oplus a_0 c_0 c_1 \oplus a_0 a_1 c_0 \oplus a_0 b_1 c_0 \oplus a_0 b_0 c_0 \oplus b_0 c_0 c_1 \oplus a_1 b_0 c_0 \oplus b_0 b_1 c_0$$

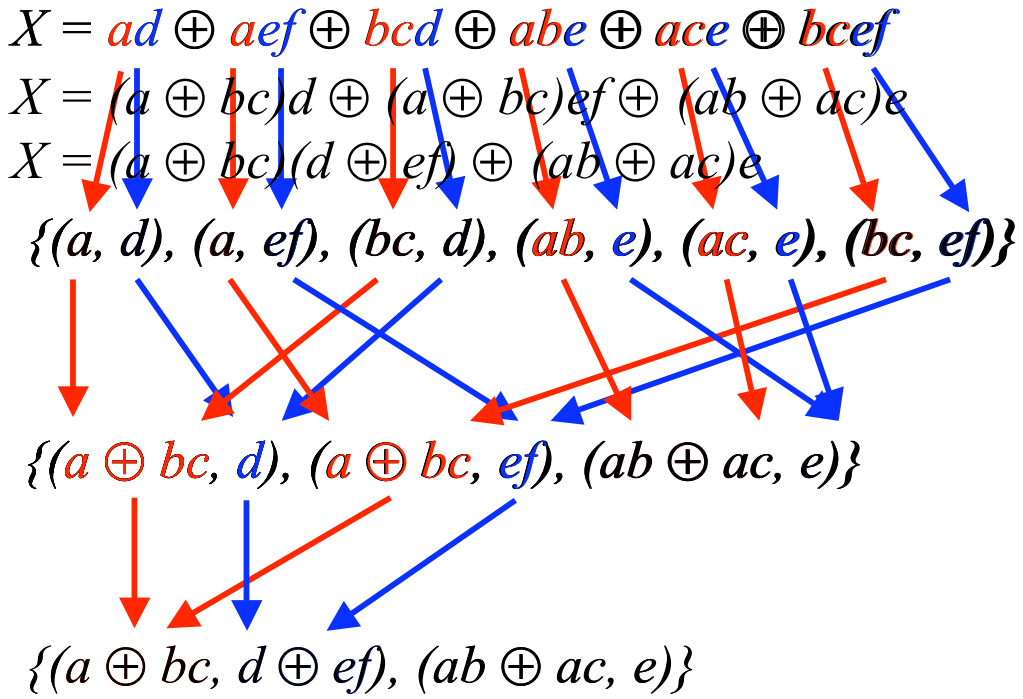
Progressive Decomposition: Algorithm Overview



- Find leader expressions
- Choose a subset of input bits
  - Optimize via Boolean ring properties
    - How many bits?
    - Find identities
- Rewrite circuit in terms of leading expressions

Finding Leader Expressions

- Similar to kernel extraction in algebraic factorization



$L(X, \{a, b, c\}) = ?$

Leader expression of X using inputs {a, b, c}

$\alpha\beta \oplus \beta\gamma \rightarrow (\alpha(\beta \oplus \beta))\gamma$

Combine expressions: $\{(\alpha, \beta), (\beta, \gamma)\} \rightarrow \{((\alpha \oplus \beta), \gamma)\}$

Separate products terms: $\{(\alpha, \beta), (\beta, \gamma)\} \rightarrow \{(\alpha, \beta \oplus \gamma)\}$

chosen pair not chosen input bits

Hierarchy and Circuit Structure

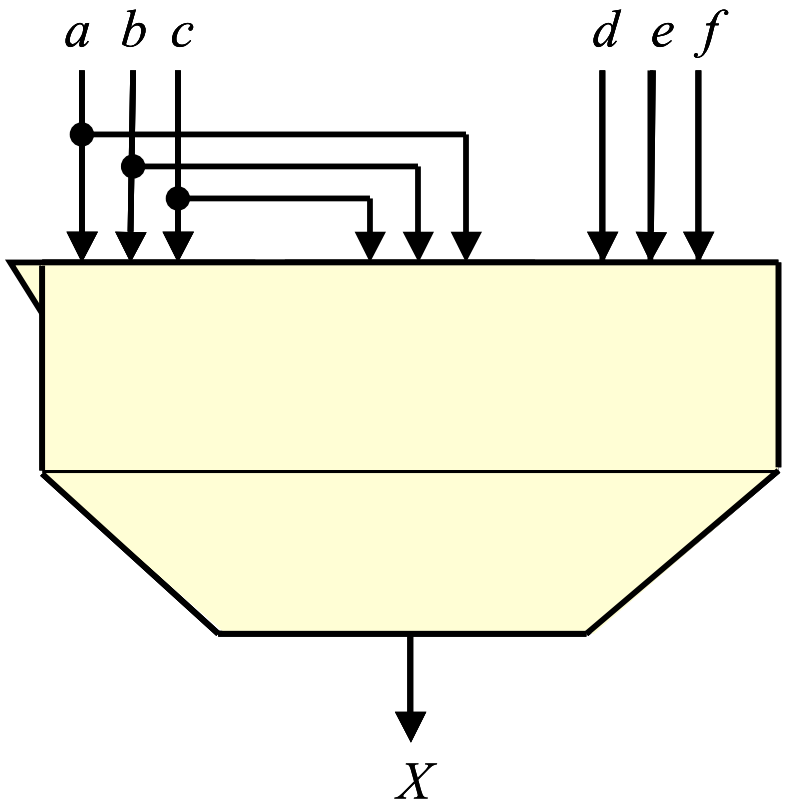
$X = ad \oplus aef \oplus bcd \oplus abe \oplus ace \oplus bcef$

$L(X, \{a, b, c\}) = \{a \oplus bc, ab \oplus ac\}$

$s_1 = a \oplus bc$

$s_2 = ab \oplus ac$

$X = s_1(d \oplus ef) \oplus s_2 e$

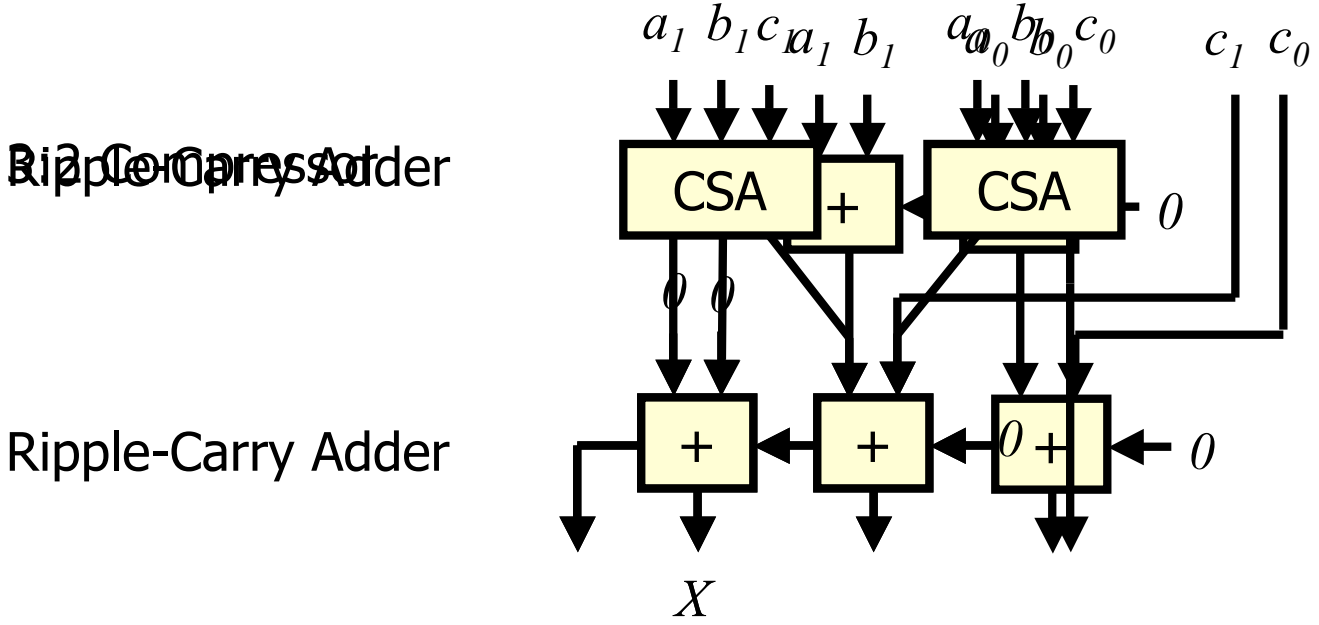


Example: Ternary Adder (3rd Output)

$$X = [a_1 b_1 + (a_1 + b_1) a_0 b_0]$$
$$\oplus\ [(a_1 \oplus b_1 \oplus a_0 b_0) c_1 + c_0 (a_0 \oplus b_0)(c_1 + (a_1 \oplus b_1 \oplus a_0 b_0))]$$

$$L(X, \{a_1, b_1, c_1\}) = \{a_1 \oplus b_1 \oplus c_1,\quad a_1 b_1 \oplus b_1 c_1 \oplus a_1 c_1\}$$

sum carry Carry-save adder



Exploiting the Null Space

- Null space of P, N(P):
  - All expressions that satisfy PX = 0

$X = s_1(t_1 \oplus e) \oplus t_1 s_2 \oplus e s_2$

$X = (c \oplus d)(s_1 \oplus e) \oplus cds_2 \oplus es_2$

$X = s_2(t_1 \oplus e) \oplus t_2(s_1 \oplus e)$

$L(X, \{a, b\}) = \{ab, a \oplus b\}$

$L(X, \{c, d\}) = \{cd, c \oplus d\}$

$L(X, \{s_2, t_2\}) = ?$ $t_1 = cd$ $t_2 = c \oplus d$ $s_1 = ab$ $s_2 = a \oplus b$ $s_1 \in N(s_2)$ $t_1 \in N(t_2)$

$\{(s_2, t_1 \oplus e), (t_2, s_1 \oplus e)\}$

$\{(s_2, s_1 \oplus t_1 \oplus e), (t_2, s_1 \oplus t_1 \oplus e)\}$

$s_1 \in N(s_2)$ $t_1 \in N(t_2)$

$\{(s_2 \oplus t_2, s_1 \oplus t_1 \oplus e)\}$

Combine expressions: $\{(\alpha, \gamma), (\beta, \gamma)\} \rightarrow \{(\alpha \oplus \beta, \gamma)\}$

Linear Independence

- Linear dependence
  - Between leader expressions
  - Or between their corresponding coefficients
  - Rewrite some elements in terms of others

$$\{a \oplus b,\ b \oplus c,\ c \oplus a\} \longrightarrow \{a \oplus b,\ b \oplus c\}$$

$$c \oplus a = (a \oplus b) \oplus (b \oplus c)$$

- LZD
  - Initial basis: $\{V_0,\ P_{00},\ P_{01},\ V_0 \oplus P_{00},\ V_0 \oplus P_{01}\}$
  - Reduces to: $\{V_0,\ P_{00},\ P_{01}\}$

7-bit Majority Function

- Returns 1 if at least 4 bits are 1; 0 otherwise

$$L(X, \{a_1, a_2, a_3, a_4\}) = \{s_1, s_2, s_3, s_4\}$$

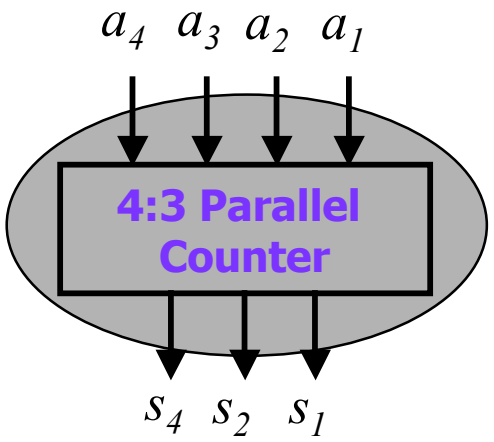
$$s_1 = a_1 \oplus a_2 \oplus a_3 \oplus a_4$$

$$s_2 = a_1a_2 \oplus a_1a_3 \oplus a_1a_4 \oplus a_2a_3 \oplus a_2a_4 \oplus a_3a_4$$

$$s_3 = a_1a_2a_3 \oplus a_1a_2a_4 \oplus a_1a_3a_4 \oplus a_2a_3a_4$$

$$s_4 = a_1a_2a_3a_4$$

$s_1, s_2 \in N(s_4)$ $\quad s_4 \in N(s_1)$ $\quad s_4 \in N(s_2)$



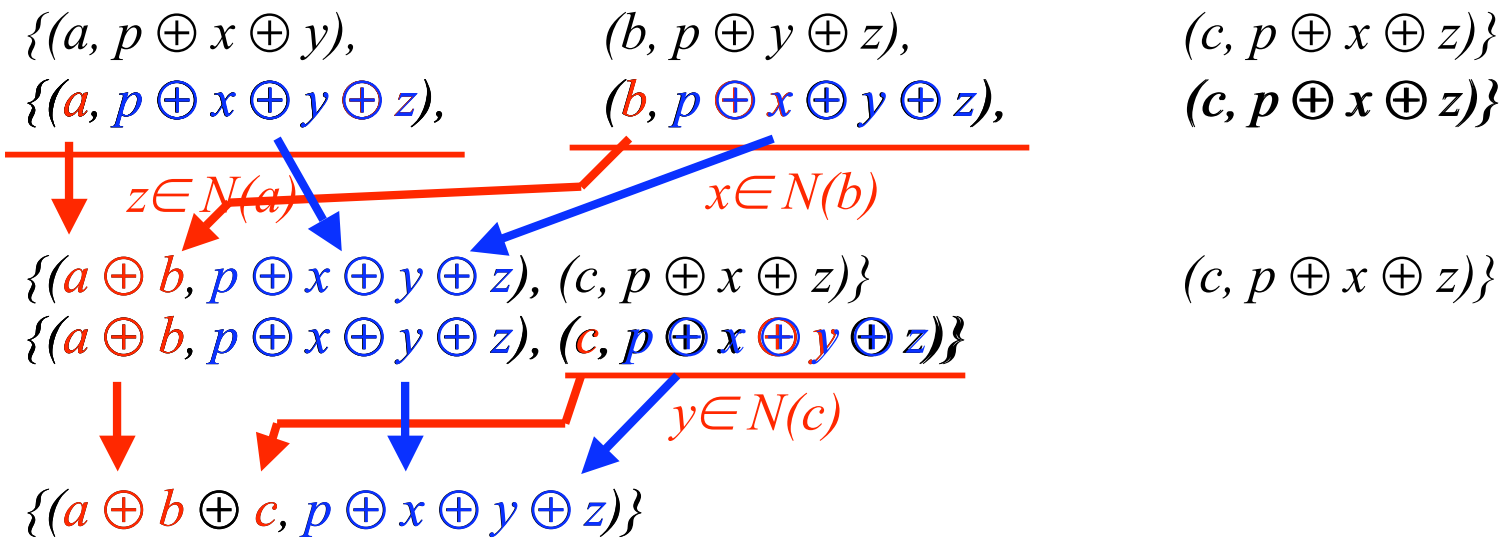
$s_3 = s_1s_2$

$a_4a_3a_2a_1$	s_4	s_3	s_2	s_1
0000	0	0	0	0
0001	0	0	0	1
0010	0	0	1	0
0011	0	0	1	1
0100	0	0	0	1
0101	0	0	1	0
0110	0	0	1	0
0111	0	1	1	1
1000	0	0	0	1
1001	0	0	1	0
1010	0	0	1	0
1011	0	1	1	1
1100	0	0	1	0
1101	0	1	1	1
1110	0	1	1	1
1111	1	0	0	0

Propagation of Null Space Information

$X = ap \oplus bp \oplus cp \oplus ax \oplus ay \oplus by \oplus bz \oplus cx \oplus cz \qquad \{az = 0,\ bx = 0,\ cy = 0\}$

$L(X, \{a, b, c\}) = ?$

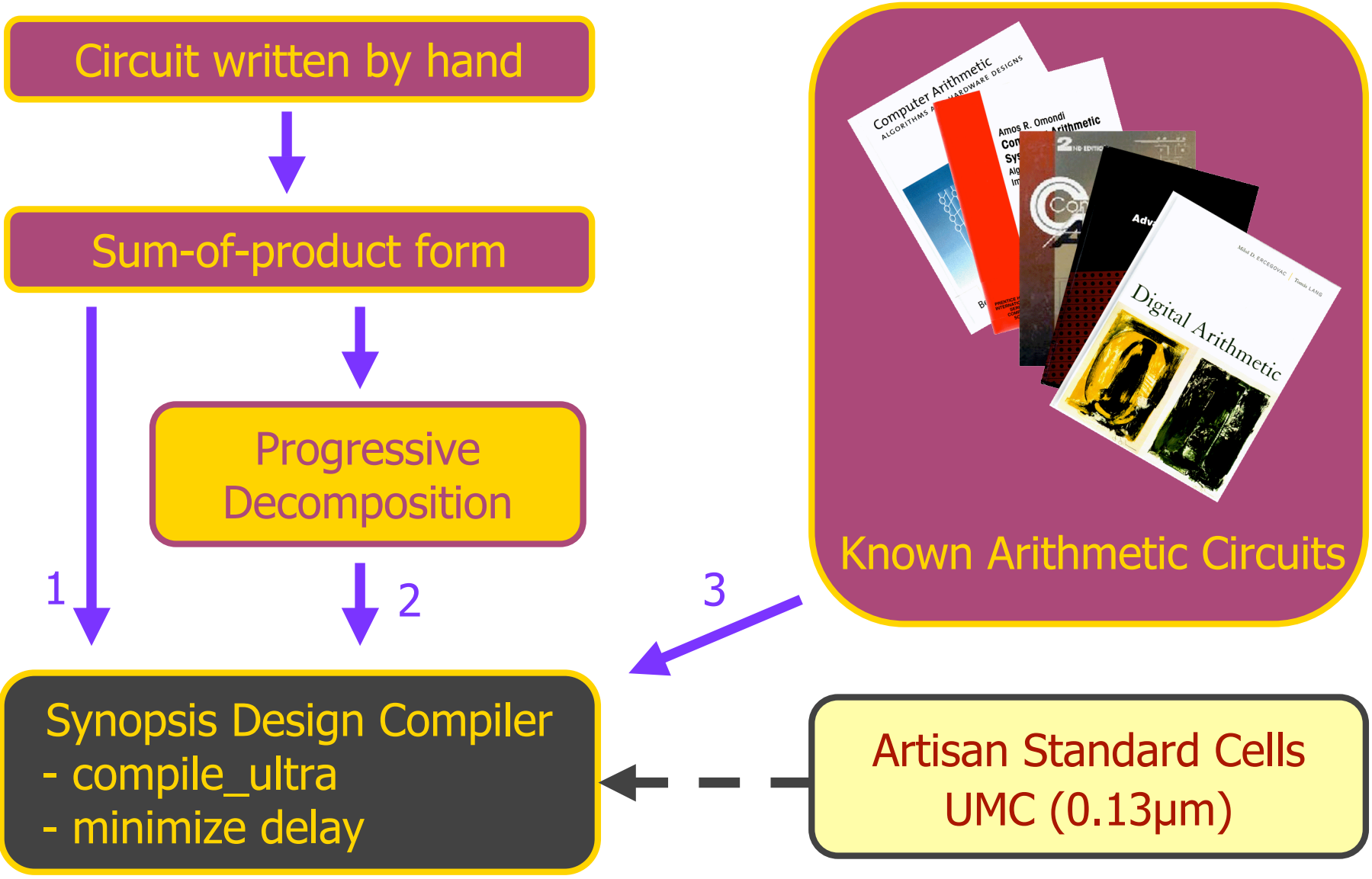


$X = (a \oplus b \oplus c)(p \oplus x \oplus y \oplus z)$

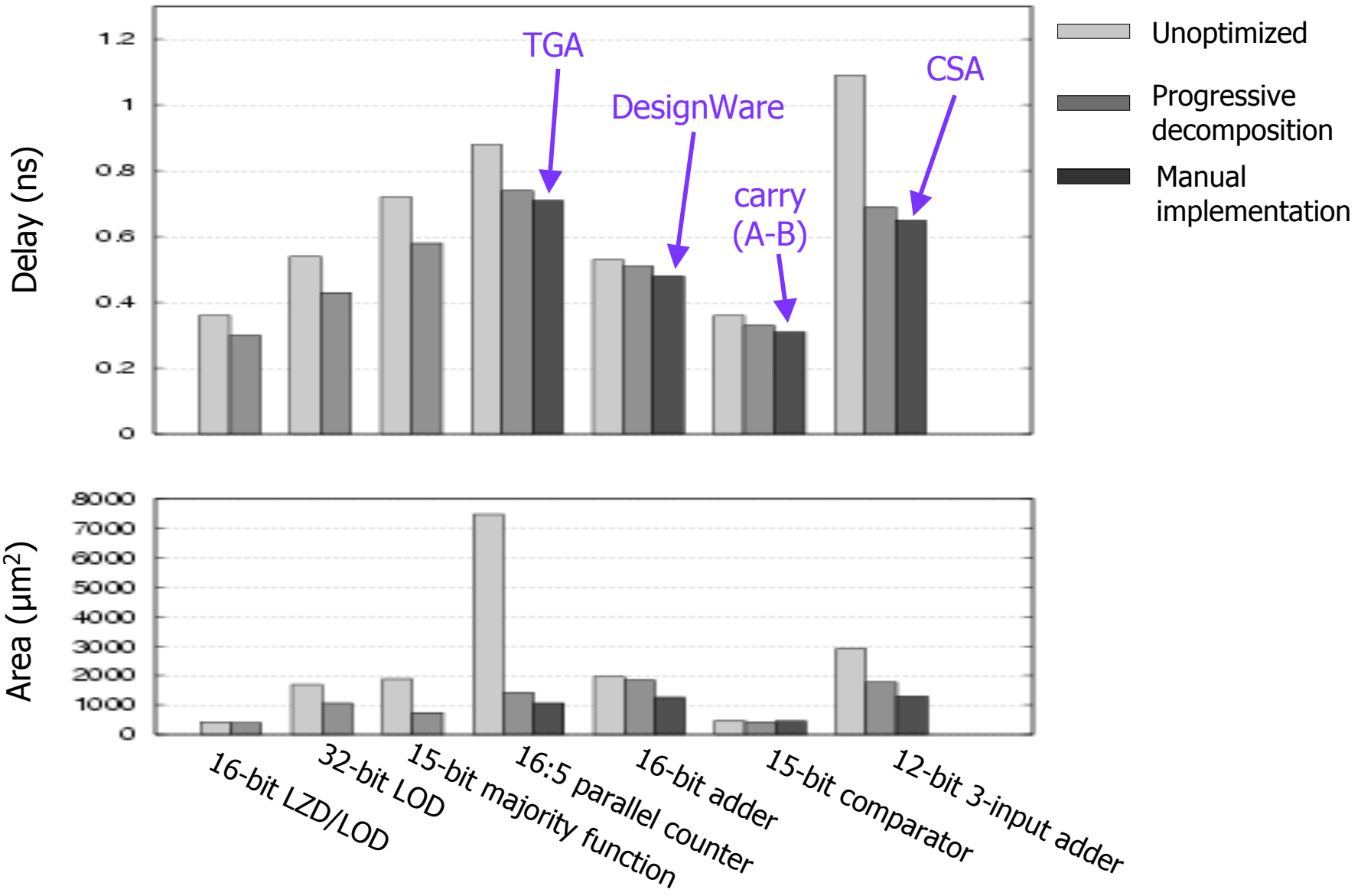
$L(X, \{a, b, c\}) = \{a \oplus b \oplus c\}$

Combine expressions: $\{(\alpha, \gamma), (\beta, \gamma)\} \rightarrow \{(\alpha \oplus \beta, \gamma)\}$

Experimental Setup



Results


Delay (ns)
1.2
1
0.8
0.6
0.4
0.2
0
TGA
DesignWare
carry
(A-B)
CSA
Unoptimized
Progressive decomposition
Manual implementation
Area (μm²)
8000
7000
6000
5000
4000
3000
2000
1000
0
16-bit LZD/LOD
32-bit LOD
15-bit majority function
16:5 parallel counter
16-bit adder
15-bit comparator
12-bit 3-input adder

Conclusion

- Progressive Decomposition Algorithm
  - Arithmetic circuits
    - Previously, hard to optimize
    - Expert ideas can be generalized and automated
  - Automatically infers successful circuit designs from the literature
    - Carry-lookahead adder
    - Structured LZD circuit
    - Carry-save addition
    - Parallel counters
- Long-term goal
  - Replace manual circuit design with automated tools