Improving XOR-Dominated Circuits by Exploiting Dependencies between Operands

Ajay K. Verma and **Paolo Ienne**

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

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Ecole Polytechnique Fédérale de Lausanne (EPFL)

Logic Synthesis: Limited Dependency Exploitation

- Logic synthesis tools are extremely good at optimizing the Boolean expressions containing AND, OR and NOT gates.

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

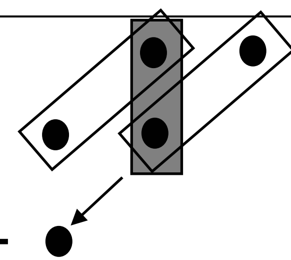
- In case of XOR gate either expand the XOR gate in terms of AND and OR gate, or replace the XOR expression by a new variable.

- Expanding XOR gates might increase the expression size exponentially, and the expansion cannot be restored due to shortcomings of algebraic factoring.

Only dependencies among the operands of AND and OR gates are utilized.

Multiplier: XOR Gates with Correlated Operands

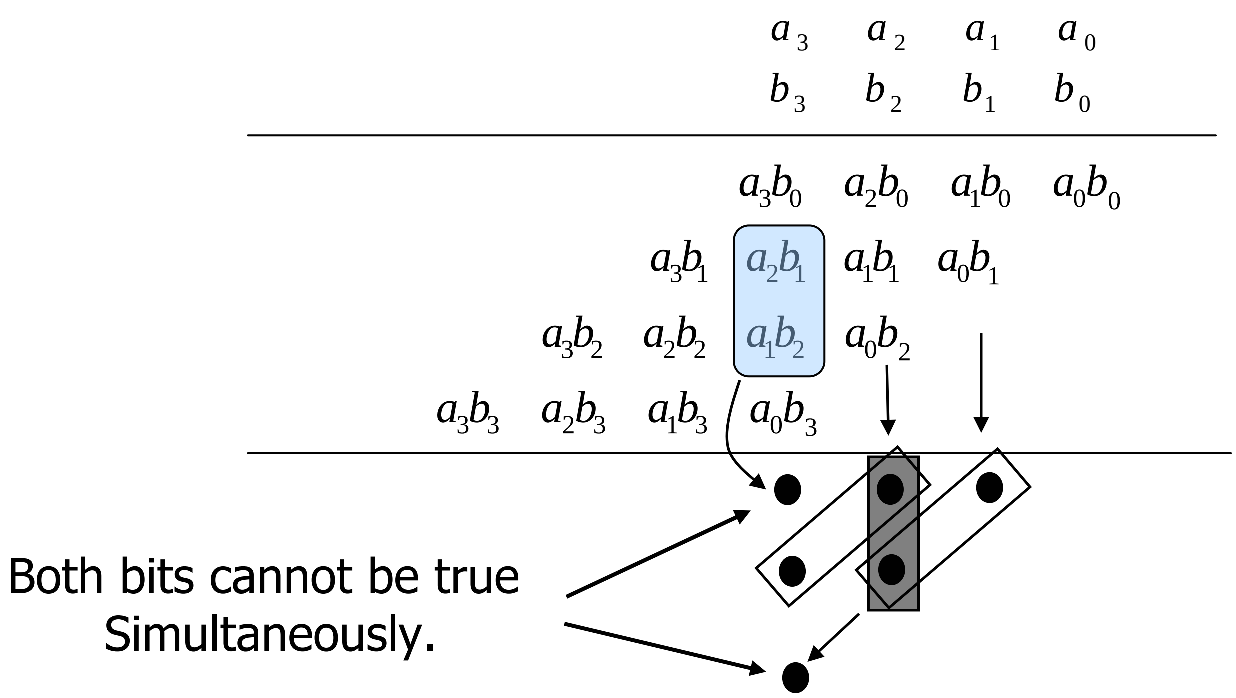
			a_3	a_2	a_1	a_0
			b_3	b_2	b_1	b_0
			a_3b_0	a_2b_0	a_1b_0	a_0b_0
		a_3b_1	a_2b_1	a_1b_1	a_0b_1	
	a_3b_2	a_2b_2	a_1b_2	a_0b_2		
a_3b_3	a_2b_3	a_1b_3	a_0b_3			



$$a_0b_1 \bullet a_1b_0 \to a_1b_1$$

$$c_3 = a_0b_0a_1b_1(a_0b_2 \oplus a_1b_1 \oplus a_2b_0)$$
$$= a_0b_0a_1b_1(\overline{a_2 \oplus b_2})$$

Multiplier: XOR Gates with Correlated Operands



$$a_0b_1a_1b_0\left(a_0b_2 \oplus a_1b_1 \oplus a_2b_0\right) \rightarrow \overline{a_1b_2 \oplus a_2b_1}$$

$$a_0b_1a_1b_0(a_0b_2 \oplus a_1b_1 \oplus a_2b_0) \oplus (a_1b_2 \oplus a_2b_1)$$

Outline

- Related work.
- Problem formulation.
- Basic Idea.
  - Expanding XOR gates selectively.
- Analysis and improvement of basic idea.
  - Broadening of selection criteria.
- Results.
- Conclusions.

Related Work

- Optimization of general XOR-dominated circuits.
  - Optimization of Reed-Muller form [Mishchenko01, …]
  - 2-SPP form optimization [Bernasconi06]
  - BDD-based logic optimization [Sasao93, Yang00, …]
- Optimization of column compressors.
  - Carry-save addition [Wallace64]
  - Optimization using various size counters [Song91]
  - Proper scheduling of counters (TGA) [Oklobdzija96, Stelling98, …]

Why Not Expand All XOR's

$$P = a_0 b_3 a_3 b_0 + a_0 b_3 a_1 b_2 + a_1 b_2 a_3 b_0$$

$$Q = a_2 b_1 (a_0 b_3 \oplus a_1 b_2 \oplus a_3 b_0)$$

$$R = P \oplus Q \longrightarrow \overline{PQ}(P + Q)$$

Before expansion : 0.37 ns (138.2 μm²)
After expansion : 0.26 ns (146.9 μm²)

$$P = a_0 a_1 + a_1 a_2 + a_2 a_3 + a_3 a_4$$

$$Q = b_0 b_1 + b_1 b_2 + b_2 b_3 + b_3 b_4$$

$$R = P \oplus Q$$

Before expansion : 0.22 ns (58.8 μm²)
After expansion : 0.27 ns (221.2 μm²)

Selective Expansion of XOR's is the key.

Problem Statement

Given a circuit consisting of AND, OR, NOT, and XOR gates, find a list of XOR gates which should be expanded to achieve smallest critical path delay after logic synthesis.

- Correlation factor: XOR gates with sufficient correlation between its operands must be expanded in order to reduce delay.

Selective Expansion

$$AB = 0 \rightarrow A \oplus B = A + B$$

$$A + B = 1 \rightarrow A \oplus B = \overline{AB}$$

} Extremely correlated operands

- *Small* expression: An expression that can be computed quickly.
- XOR expansion:

$$A \oplus B = \overline{AB}(A + B)$$

Relative sizes of (AB) and (A+B) are a good measure of correlation between A and B.

Selective Expansion Contd.

$$P = a_0b_3a_3b_0 + a_0b_3a_1b_2 + a_1b_2a_3b_0$$
$$Q = a_2b_1(a_0b_3 \oplus a_1b_2 \oplus a_3b_0)$$
$$R = P \oplus Q$$

Expand

$$PQ = a_0b_0a_1b_1a_2b_2a_3b_3$$

Quickly computable compared to P and Q

Before expansion : 0.37 ns (138.2 μm^2)
After expansion : 0.26 ns (146.9 μm^2)

$$P = a_0a_1 + a_1a_2 + a_2a_3 + a_3a_4$$
$$Q = b_0b_1 + b_1b_2 + b_2b_3 + b_3b_4$$
$$R = P \oplus Q$$

Do not expand

$$P + Q = a_0a_1 + a_1a_2 + a_2a_3 + a_3a_4 + b_0b_1 + b_1b_2 + b_2b_3 + b_3b_4$$
$$PQ = \sum_{i=0}^{i=3}\sum_{j=0}^{j=3} a_ia_{i+1}b_jb_{j+1}$$

Slow computation compared to P and Q

Before expansion : 0.22 ns (58.8 μm^2)
After expansion : 0.27 ns (221.2 μm^2)

Criteria for XOR Expansion

isExpansionUseful (operand A, operand B)

{

$\varepsilon = 1 - \dfrac{\min(D_{AB}, D_{A+B})}{\max(D_A, D_B)}$; // correlation factor.

$\Delta = \dfrac{AR(D_{AB} + D_{A+B})}{AR(D_A + D_B)} - 1$; // area penalty.

// correlation between the operands must be significant as well as
// expansion should not have huge area overhead.

if ($\varepsilon < \varepsilon_{threshold}$ **or** $\Delta > \Delta_{threshold}$)

return *false*;

// area penalty per unit correlation must be small.

if ($\Delta / \varepsilon > \kappa$)

return *false*;

return *true*;

}

Correlation between XOR-Operands and Rest of the Function Can Be Useful

Local correlation: correlation between the operands of XOR.

$$A \oplus B = \overline{AB}\left(A + B\right)$$

Global correlation: correlation between the rest of the expression and the operands of XOR.

$$\left(A \oplus B\right) + C = \left(AB \rightarrow C\right) \cdot \left(A + B + C\right)$$

$$\left(A \oplus B\right) + C = \left(\overline{A} \cdot \overline{B} \rightarrow C\right) \cdot \left(\overline{AB} + C\right)$$

Global Correlation Example

Comparator function

(1) $(A>B) \equiv (a_n > b_n) + (a_n = b_n)(a_{n-1} > b_{n-1}) + (a_n = b_n)(a_{n-1} = b_{n-1})(a_{n-2} > b_{n-2}) + \ldots$

$(A>B) \equiv (a_n \overline{b_n}) + (a_n \oplus \overline{b_n})(a_{n-1}\overline{b_{n-1}}) + (a_n \oplus \overline{b_n})(a_{n-1} \oplus \overline{b_{n-1}})(a_{n-2}\overline{b_{n-2}}) + \ldots$

(2) $(A>B) \equiv (a_n > b_n) + (a_n \geq b_n)(a_{n-1} > b_{n-1}) + (a_n \geq b_n)(a_{n-1} \geq b_{n-1})(a_{n-2} > b_{n-2}) + \ldots$

$(A>B) \equiv (a_n \overline{b_n}) + (a_n + \overline{b_n})(a_{n-1}\overline{b_{n-1}}) + (a_n + \overline{b_n})(a_{n-1} + \overline{b_{n-1}})(a_{n-2}\overline{b_{n-2}}) + \ldots$

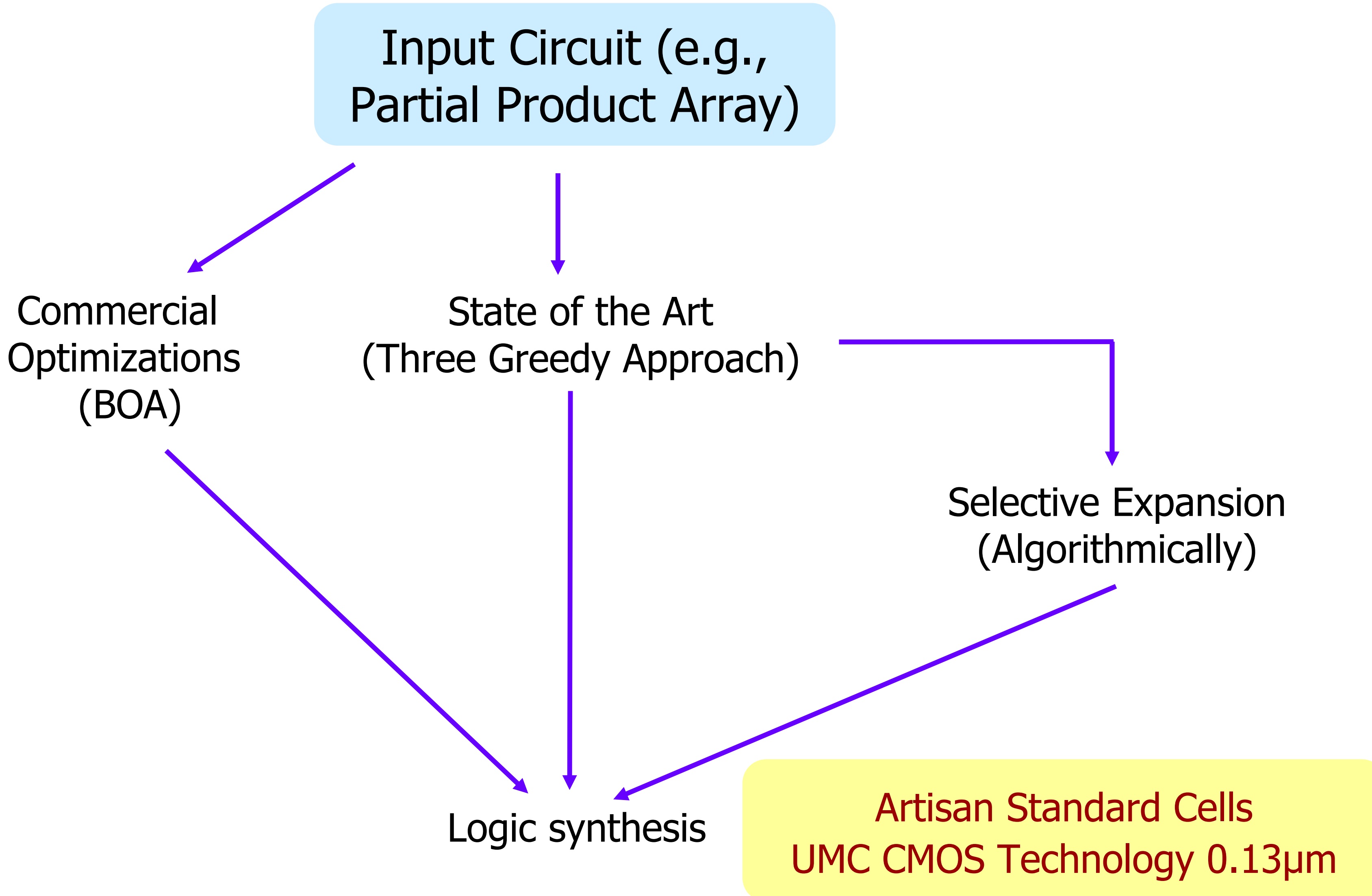
carry $(A + \overline{B} + 1)$ = carry $(A - B)$

Global correlation converts a user friendly implementation into synthesis friendly implementation.

Overall Algorithm

- The expression tree is optimized iteratively by expanding XOR gates based on local and global correlation criteria.
- If there are more than XOR gates which satisfy the expansion criteria, a proper ordering of expansion is decided using a greedy heuristic.
- In order to measure local and global correlation, an estimator function is used to estimate the delay and area values.

Experimental Setup

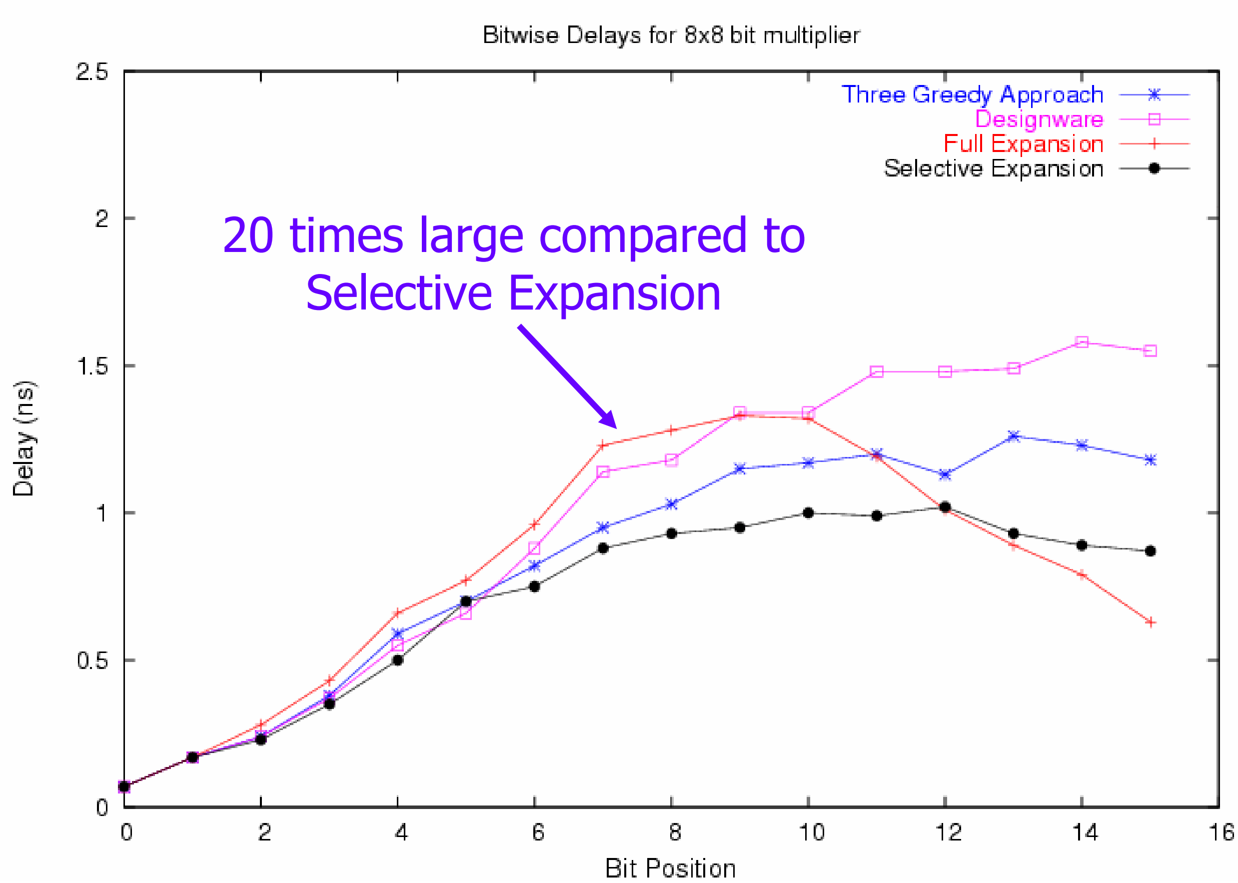


Results

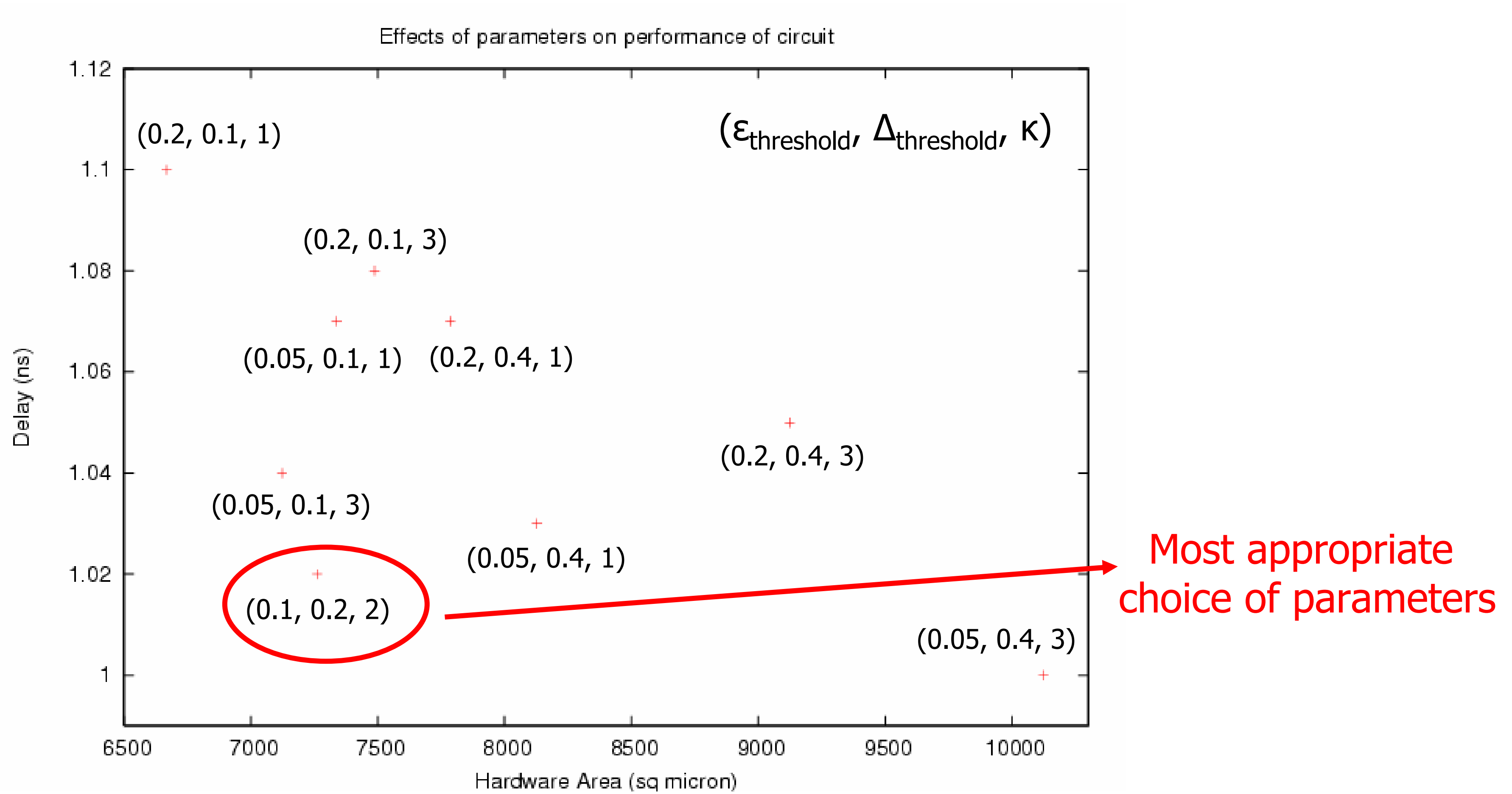
ADPCM Decoder		
Commercial Optimizations (BOA)	6556 μm^2	1.06 ns
Selective Expansion	6934 μm^2	0.90 ns
8 x 8-bit Multiplier		
DesignWare	4488 μm^2	1.60 ns
Three Greedy Approach (TGA)	5996 μm^2	1.28 ns
Selective Expansion	7262 μm^2	1.02 ns
Constant Multiplication (A x 7)		
A x 7	2587 μm^2	0.85 ns
A + 2A + 4A	3155 μm^2	0.72 ns
8A - A	1941 μm^2	0.56 ns
Selective Expansion (A + 2A + 4A as input)	3018 μm^2	0.50 ns
Selective Expansion (8A – A as input)	2822 μm^2	0.52 ns
15-bit Comparator		
Commercial Optimizations	515 μm^2	0.40 ns
Selective Expansion	466 μm^2	0.33 ns

Results Contd.

8x8-bit Multiplier: Delay Comparison



Appropriate choice of Parameters



Delay and hardware area values for different implementations of a 8 x 8-bit multiplier generated by Selective Expansion for different values of $\varepsilon_{threshold}$, $\Delta_{threshold}$, κ.

Results Contd.

$$
\begin{array}{cccccccc}
 & & & & a_3 & a_2 & a_1 & a_0 \\
 & & & & b_3 & b_2 & b_1 & b_0 \\
\hline
 & & & & a_3b_0 & a_2b_0 & a_1b_0 & a_0b_0 \\
 & & & a_3b_1 & a_2b_1 & a_1b_1 & a_0b_1 & \\
 & & a_3b_2 & a_2b_2 & a_1b_2 & a_0b_2 & & \\
 & a_3b_3 & a_2b_3 & a_1b_3 & a_0b_3 & & & \\
\hline
p_7 & p_6 & p_5 & p_4 & p_3 & p_2 & p_1 & p_0
\end{array}
$$

$$p_3 = (a_0b_2a_2b_0 + a_0b_2a_1b_1 + a_1b_1a_2b_0) \oplus a_0b_3 \oplus a_3b_0 \oplus$$

$$\boxed{a_1b_2 \oplus a_2b_1 \oplus a_0b_0a_1b_1(a_0b_2 \oplus a_1b_1 \oplus a_2b_0)}$$

Results Contd.

$$a_1b_2 \oplus a_2b_1 \oplus a_0b_0a_1b_1(a_0b_2 \oplus a_1b_1 \oplus a_2b_0)$$

Optimizations based on simple dependency

$$a_1b_2 \oplus a_2b_1 \oplus a_0b_0a_1b_1(a_2 \oplus \overline{b_2})$$

$$(a_1b_2 \oplus a_2b_1)a_0b_0a_1b_1(a_2 \oplus \overline{b_2}) = 0$$

Local correlation

$$(a_1b_2 \oplus a_2b_1) + a_0b_0a_1b_1(a_2 \oplus \overline{b_2})$$

$$a_0b_0a_1b_1a_2\overline{b_2} \rightarrow (a_1b_2 \oplus a_2b_1)$$

Global correlation

$$(a_1b_2 \oplus a_2b_1) + a_0b_0a_1b_1(a_2 + \overline{b_2})$$

Optimum implementation

$$(a_1b_2 \oplus a_2b_1) + a_0b_0a_1b_1$$

Conclusion

- We have shown that logic synthesis escapes certain kind of optimizations on XOR-dominated circuits.
- We present an algorithm which works as a front end to logic synthesis tool and transforms a given circuit into synthesis friendly circuit.
- Selective Expansion improves the speed of some arithmetic circuits such as 8 x 8-bit multiplier by 20% over state of art techniques.