Multi-Level Logic with Constant Depth: Recent Research from Italy

Researchers:

Anna Bernasconi (U. Pisa), Valentina Ciriani (U. Milano-Crema) , Roberto Cordone (U. Milano-Crema), Fabrizio Luccio (U. Pisa), Linda Pagli (U. Pisa), Tiziano Villa (U. Verona, speaker)

DIMACS-RUTCOR Workshop on Boolean and Pseudo-Boolean Functions

in Memory of Peter L. Hammer

Rutgers, January 19-22, 2009

2-SPP: synthesis and testing

Three-level logic

- Three level networks of the form (Debnath, Sasao, Dubrova, Perkowski, Miller and Muzio):

$$f = g_1 \circ g_2$$

Where:

- g_i is an SOP form
- $\circ$ is a binary operator:
  - $\circ$ = AND : AND-OR-AND forms
  - $\circ$ = EXOR: AND-OR-EXOR forms (EX-SOP)
- OR-AND-OR (Sasao)
- SPP (Luccio, Pagli): EXOR-AND-OR

SPP forms

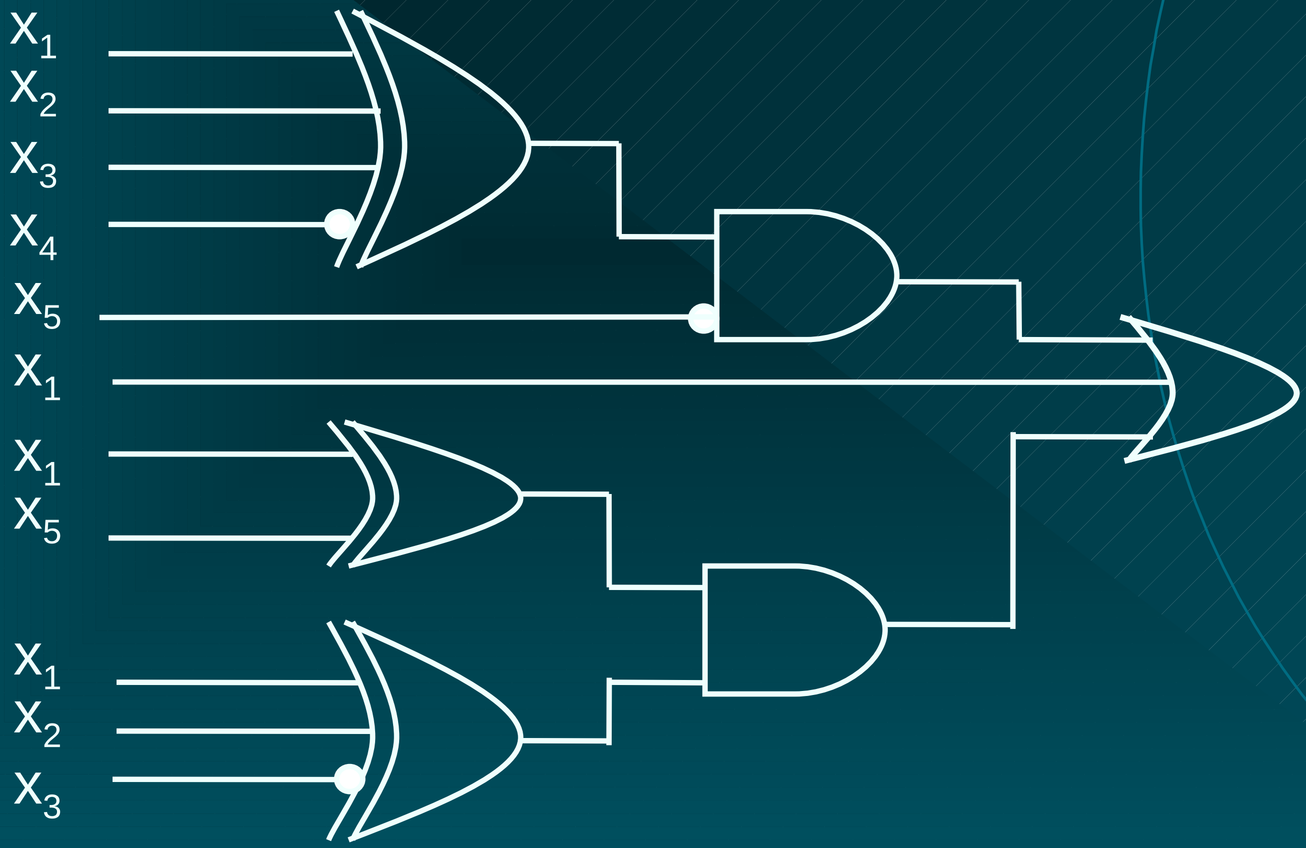
- SPP forms are a direct generalization of SOP forms:

$$\underbrace{(x_1 \oplus x_2 \oplus x_3 \oplus \overline{x}_4)\,\overline{x}_5}_{\text{Pseudoproduct}} + \underbrace{(x_1 \oplus x_2 \oplus \overline{x}_3)\overbrace{(x_1 \oplus x_5)}^{\text{EXOR factor}}}_{\text{Pseudoproduct}} + \underbrace{x_1}_{\text{Pseudoproduct}}$$

- An SPP form is a sum (OR) of pseudoproducts
- ***The SPP problem***: find an SPP form for a function F with the min. number of literals

SPP forms

$$(x_1 \oplus x_2 \oplus x_3 \oplus \overline{x_4})\,\overline{x_5} + (x_1 \oplus x_2 \oplus \overline{x_3})(x_1 \oplus x_5) + x_1$$



SPP forms

Advantages

- *Compact expressions*
- *Good testability of EXORs*
- *Three levels of logic*

Disadvantages

- ***Unbounded fan-in EXORs***
- ***Impractical for many technologies***
- ***Huge minimization time***

Affine spaces

- The affine space A over the vector space $V \subseteq \{0,1\}^n$ (with operator $\oplus$) is:

$$A = \{p \oplus v \mid v \in V\} = p \underbrace{\oplus}_{\text{Translation Point}} \underbrace{V}_{\text{Vector Space}}$$

Affine space

Translation point

Vector space

x1	x2	x3	x4
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

=

1	0	0	0

$\oplus$

x1	x2	x3	x4
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0

A **p** **V**

Pseudocubes

Product = characteristic function of a **cube**

$x_1 \cdot x_4$

X1	X2	X3	X4
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Pseudoproduct = characteristic function of a **pseudocube**

$x_1 \cdot (x_2 \oplus x_3 \oplus \overline{x}_4)$

X1	X2	X3	X4
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Canonical Expressions CEX

- A pseudocube can be represented by different pseudoproducts

P =

X1	X2	X3	X4
0	0	1	1
0	1	1	1
1	0	0	0
1	1	0	0

CEX(P) = $(x_1 \oplus x_3)(x_1 \oplus x_4)$

$(x_1 \oplus x_3)(x_3 \oplus \overline{x}_4)$

$(x_1 \oplus x_4)(x_3 \oplus \overline{x}_4)$

- One of them is called **CEX**

Pseudocubes and Affine Spaces

- **Theorem:**

  Pseudocubes $\Leftrightarrow$ Affine Spaces

- **Corollary:**

  Cubes $\subseteq$ Affine Spaces

- **Pseudocube can be represented by:**
  - **CEX**
  - **Affine Space: $p \oplus V$**

Affine Spaces

X1 X2 \ X3 X4	00	01	11	10
00				
01				
11		●		●
10	●		●	

Pseudoproduct:

$$x_1 \cdot (x_2 \oplus x_3 \oplus \overline{x}_4)$$

Red: canonical variables

Black: non canonical variables

$$\begin{array}{cccc} X1 & X2 & X3 & X4 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{array} = \begin{array}{cccc} 1 & 0 & 0 & 0 \end{array} \oplus \begin{array}{cccc} X1 & X2 & X3 & X4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{array}$$

Cubes as Affine Spaces

X1 X2 \ X3 X4	00	01	11	10
00				
01				
11		●	●	
10		●	●	

Product:

$$X_1 \cdot X_4$$

Red: canonical variables

Black: non canonical variables

X1	X2	X3	X4
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

=

1	0	0	1

$\oplus$

X1	X2	X3	X4
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0

Union of Pseudocubes

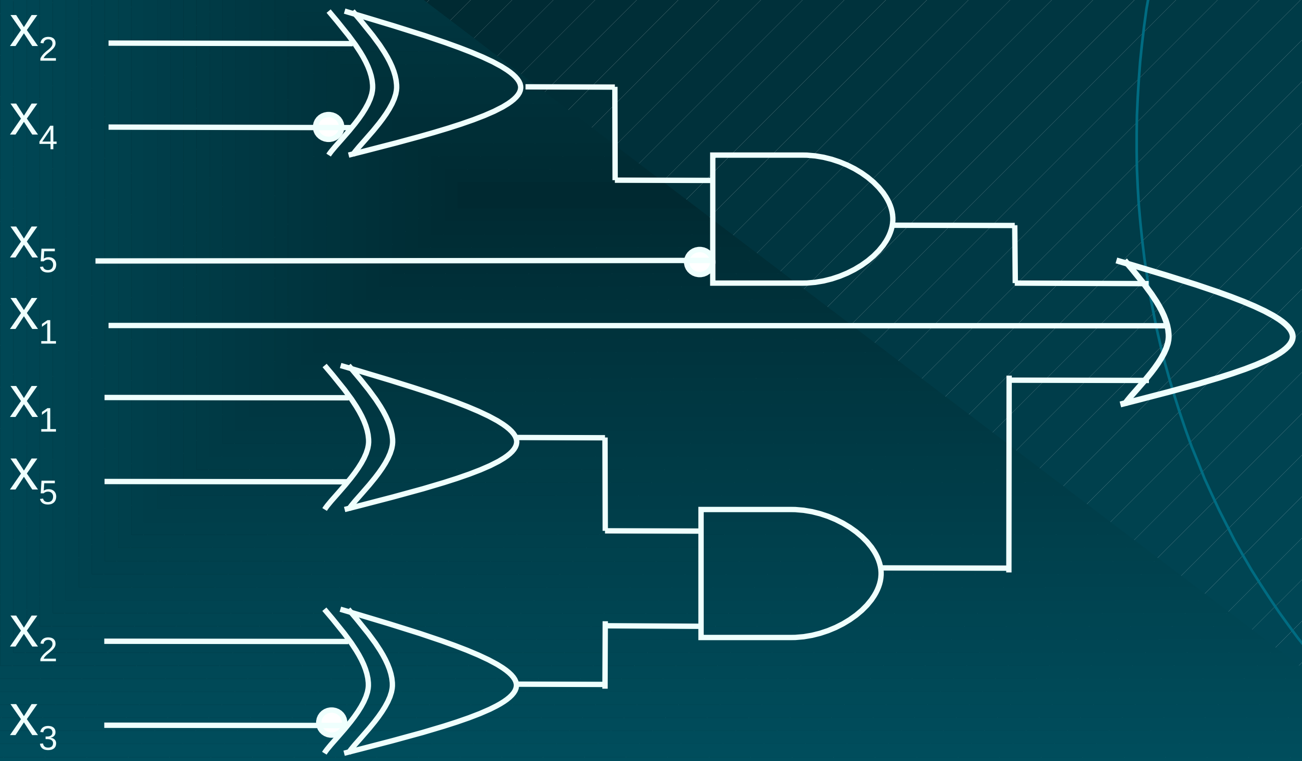
- The union of of two pseudocubes is a pseudocube iff they are affine spaces over the same vector space.
- $A = p \oplus V$, $A' = p' \oplus V$ and $p \oplus p' \notin V$
- Bases of V $\quad v_1, \dots, v_k$

$$A \cup A' = p \oplus V'$$

- Bases of V' $\quad v_1, \dots, v_k, p \oplus p'$

2-SPP forms

$$\underbrace{(x_2 \oplus \overline{x}_4)\,\overline{x}_5}_{\text{2-pseudoproduct}} + (x_2 \oplus \overline{x}_3)\underbrace{(x_1 \oplus x_5)}_{\text{2-EXOR}} + x_1$$



Solving the Disadvantages of SPP

2-SPP forms:

- Are still very compact
  - Only 4% more literals than SPP expressions
- Have a reduced minimization time
  - 92% less time than SPP synthesis
- Are practical for the current technology
  - EXOR gates with fan-in 2 are easy to implement

Parity Function

SPP: $(x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus \ \ldots \ \oplus x_n)$

SOP: is the sum of all the minterms with an odd number of positive literals.

Costs

- SPP: polynomial cost in *n*
- SOP: exponential cost in *n*

2-SPP gives exponential gain

2-SPP: $(x_1 \oplus x_2)(x_3 \oplus x_4) \ldots (x_{n-1} \oplus x_n)$

SOP: is the sum of all the minterms ($2^{n/2}$)

Costs

- 2-SPP: polynomial cost in *n*
- SOP: exponential cost in *n* ($2^{n/2}$)

Cubes

x_1	x_2	x_3	x_4
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1

$x_1 x_2 \backslash x_3 x_4$	00	01	11	10
00		●	●	
01		●	●	
11				
10				

Product:

$\overline{x}_1 \cdot x_4$

2-Pseudocubes

x_1	x_2	x_3	x_4
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0

$x_1 x_2$ \ $x_3 x_4$	00	01	11	10
00		●		●
01		●		●
11				
10				

2-pseudoproduct:

$$\overline{x}_1 \cdot (x_3 \oplus x_4)$$

Representation of 2-pseudocubes

- A cube has an unique representation
- A 2-pseudocube can be represented by different 2-pseudoproducts

$$(x_1 \oplus \overline{x}_2)x_4(x_3 \oplus \overline{x}_5)(x_3 \oplus x_7)\overline{x}_9$$

$$(x_1 \oplus \overline{x}_2)x_4(x_3 \oplus \overline{x}_5)(x_5 \oplus x_7)\overline{x}_9$$

$$(x_1 \oplus \overline{x}_2)x_4(x_3 \oplus x_7)(x_5 \oplus x_7)\overline{x}_9$$

Canonical Representation

$$(x_1 \oplus \overline{x}_2)x_4(x_3 \oplus \overline{x}_5)(x_3 \oplus x_7)\overline{x}_9$$

$$\begin{cases} (x_1 \oplus \overline{x}_2) = 1 \\ x_4 = 1 \\ (x_3 \oplus \overline{x}_5) = 1 \\ (x_3 \oplus x_7) = 1 \\ \overline{x}_9 = 1 \end{cases} = \begin{cases} x_1 = x_2 \\ x_4 = 1 \\ x_3 = x_5 \\ x_3 = \overline{x}_7 \\ \overline{x}_9 = 1 \end{cases}$$

$\{x_1, x_2\}$ $\{1, x_4, \overline{x}_9\}$ $\{x_3, x_5, \overline{x}_7\}$ $\{x_6\}$ $\{x_8\}$

Representation of cubes

$$\bar{x}_2 x_4 \bar{x}_5 x_7 \bar{x}_9$$

$$\begin{cases} \bar{x}_2 = 1 \\ x_4 = 1 \\ \bar{x}_5 = 1 \\ x_7 = 1 \\ \bar{x}_9 = 1 \end{cases}$$

$\{1, \bar{x}_2, x_4, \bar{x}_5, x_7, \bar{x}_9\}$ $\{x_1\}$ $\{x_3\}$ $\{x_6\}$ $\{x_8\}$

Structure of 2-pseudoproducts

- Structure:

are the sets without complementations

$$\{x_1, x_2\} \quad \{1, x_4, \bar{x}_9\} \quad \{x_3, x_5, \bar{x}_7\} \quad \{x_6\} \quad \{x_8\}$$

$$\Big\Downarrow \ \textbf{Structure}$$

$$\{x_1, x_2\} \quad \{1, x_4, x_9\} \quad \{x_3, x_5, x_7\} \quad \{x_6\} \quad \{x_8\}$$

Union of 2-pseudocubes

- A union of two 2-pseudocubes is a 2-pseudocube if
  - The 2-pseudocubes have the same structure
  - The complementations differ in just one set

$$\{x_1, x_2\} \quad \{1, x_4, \overline{x}_9\} \quad \{x_3, x_5, \overline{x}_7\} \quad \{x_6\} \quad \{x_8\}$$

$$\{x_1, x_2\} \quad \{1, x_4, \overline{x}_9\} \quad \{x_3, \overline{x}_5, x_7\} \quad \{x_6\} \quad \{x_8\}$$

Union of 2-pseudocubes

- The set with different complementations is split into two sets:
  - A set containing the variables with the different complementations
  - A set containing the variables with the same complementations

$$\{x_1, x_2\} \quad \{1, x_4, \overline{x}_9\} \quad \{x_3, \overline{x}_5, x_7\} \quad \{x_6\} \quad \{x_8\}$$

$$\cup$$

$$\{x_1, x_2\} \quad \{1, x_4, \overline{x}_9\} \quad \{x_3, x_5, \overline{x}_7\} \quad \{x_6\} \quad \{x_8\}$$

$$=$$

$$\{x_1, x_2\} \quad \{1, x_4, \overline{x}_9\} \quad \{x_3\} \quad \{x_5, \overline{x}_7\} \quad \{x_6\} \quad \{x_8\}$$

2-SPP Minimization Problem

- Boolean function F:
  - single output
  - represented by its ON-set

Problem:

- Find a sum of 2-pseudoproducts that is a characteristic function for F, and is minimal w.r.t. the number of literals/products

2-SPP Synthesis

- Start with the minterms (points of the function)
- Perform the union of 2-pseudocubes in order to find the set of

prime 2-pseudocubes

- Set covering step

Data structure for the union

- We represent each different structure only once
- Partitions with the same structure are grouped together

⇩

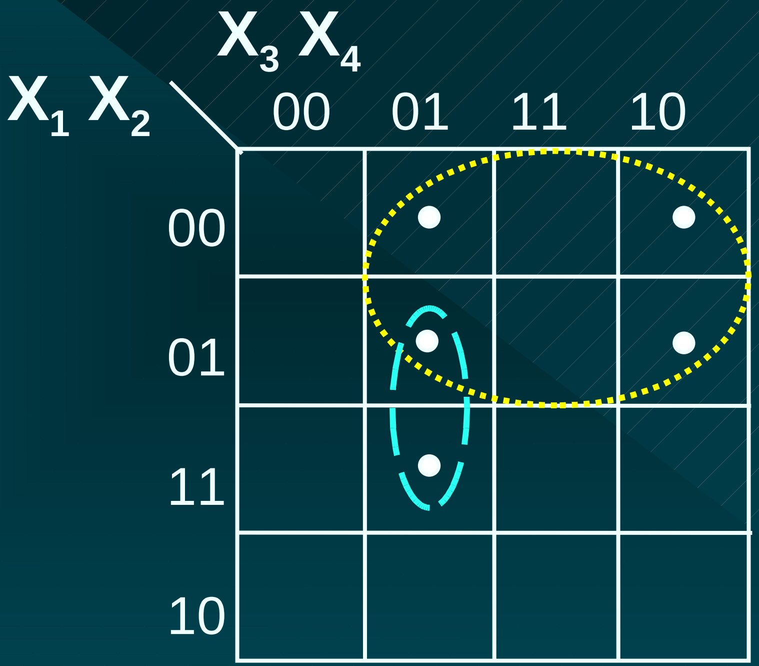
❖ **We perform the union only inside the same group**

Minimal form property

- SPP form: the minimal form depends on the variable ordering
- SOP form: the minimal form does not depend on the variable ordering
- 2-SPP form: the size of the minimal form does not depend on the variable ordering
  - Different 2-pseudoproducts represent the same 2-pseudocube
  - But they have the same cost

A minimization example

$F = \{0001, 0010, 0101, 0110, 1101\}$



An example

the minterms:

0001	0010	0101	0110	1101
$\{1,\overline{x}_1,\overline{x}_2,\overline{x}_3,x_4\}$	$\{1,\overline{x}_1,\overline{x}_2,x_3,\overline{x}_4\}$	$\{1,\overline{x}_1,x_2,\overline{x}_3,x_4\}$	$\{1,\overline{x}_1,x_2,x_3,\overline{x}_4\}$	$\{1,x_1,x_2,\overline{x}_3,x_4\}$

have the same structure: $\{1, x_1, x_2, x_3, x_4\}$

$$\{1,\overline{x}_1,\overline{x}_2,\overline{x}_3,x_4\}\cup\{1,\overline{x}_1,\overline{x}_2,x_3,\overline{x}_4\} = \{1,\overline{x}_1,\overline{x}_2\}\ \{x_3,\overline{x}_4\}$$

$$\{1,\overline{x}_1,\overline{x}_2,\overline{x}_3,x_4\}\cup\{1,\overline{x}_1,x_2,\overline{x}_3,x_4\} = \{1,\overline{x}_1,\overline{x}_3,x_4\}\ \{x_2\}$$

…

An example: the union

Structure: **Sets:**

$\{1, x_1, x_2\}\ \{x_3, x_4\}$ $\{1, \overline{x}_1, \overline{x}_2\}\ \{x_3, \overline{x}_4\}$ and $\{1, \overline{x}_1, x_2\}\ \{x_3, \overline{x}_4\}$

$\{1, x_1, x_3, x_4\}\ \{x_2\}$ $\{1, \overline{x}_1, \overline{x}_3, x_4\}\ \{x_2\}$ and $\{1, \overline{x}_1, x_3, \overline{x}_4\}\ \{x_2\}$

$\{1, x_1\}\ \{x_2, x_3, x_4\}$ $\{1, \overline{x}_1\}\ \{x_2, \overline{x}_3, x_4\}$ and $\{1, \overline{x}_1\}\ \{x_2, x_3, \overline{x}_4\}$

$\{1, x_3, x_4\}\ \{x_1, x_2\}$ $\{1, \overline{x}_3, x_4\}\ \{x_1, x_2\}$

$\{1\}\ \{x_1, x_2, x_3, x_4\}$ $\{1\}\ \{x_1, x_2, \overline{x}_3, x_4\}$

$\{1, x_2, x_3, x_4\}\ \{x_1\}$ $\{1, x_2, \overline{x}_3, x_4\}\ \{x_1\}$

$\{1, x_2\}\ \{x_1, x_3, x_4\}$ $\{1, x_2\}\ \{x_1, \overline{x}_3, x_4\}$

An example

$$\{1, \bar{x}_1, \bar{x}_2\}\ \{x_3, \bar{x}_4\} \quad \cup \quad \{1, \bar{x}_1, x_2\}\ \{x_3, \bar{x}_4\}$$

$$\{1, \bar{x}_1\}\ \{x_2\}\ \{x_3, \bar{x}_4\}$$

$$\{1, \bar{x}_1, \bar{x}_3, x_4\}\ \{x_2\} \quad \cup \quad \{1, \bar{x}_1, x_3, \bar{x}_4\}\ \{x_2\}$$

$$\{1, \bar{x}_1\}\ \{x_2\}\ \{x_3, \bar{x}_4\}$$

$$\{1, \bar{x}_1\}\ \{x_2, \bar{x}_3, x_4\} \quad \cup \quad \{1, \bar{x}_1\}\ \{x_2, x_3, \bar{x}_4\}$$

$$\{1, \bar{x}_1\}\ \{x_2\}\ \{x_3, \bar{x}_4\}$$

An example: set covering

Prime 2-pseudoproducts:

Set covering

$\{1, \overline{x}_3, x_4\}\ \{x_1, x_2\}$

$\{1\}\ \{x_1, x_2, \overline{x}_3, x_4\}$

$\{1, x_2, \overline{x}_3, x_4\}\ \{x_1\}$

$\{1, x_2\}\ \{x_1, \overline{x}_3, x_4\}$

$\{1, \overline{x}_1\}\ \{x_2\}\ \{x_3, \overline{x}_4\}$

$\Longrightarrow$

$\{1, x_2, \overline{x}_3, x_4\}\ \{x_1\}$

$\{1, \overline{x}_1\}\ \{x_2\}\ \{x_3, \overline{x}_4\}$

An example

- 2-SPP minimal form:

$$x_2\overline{x_3}x_4 + \overline{x_1}(x_3 \oplus x_4)$$

- SOP minimal form:

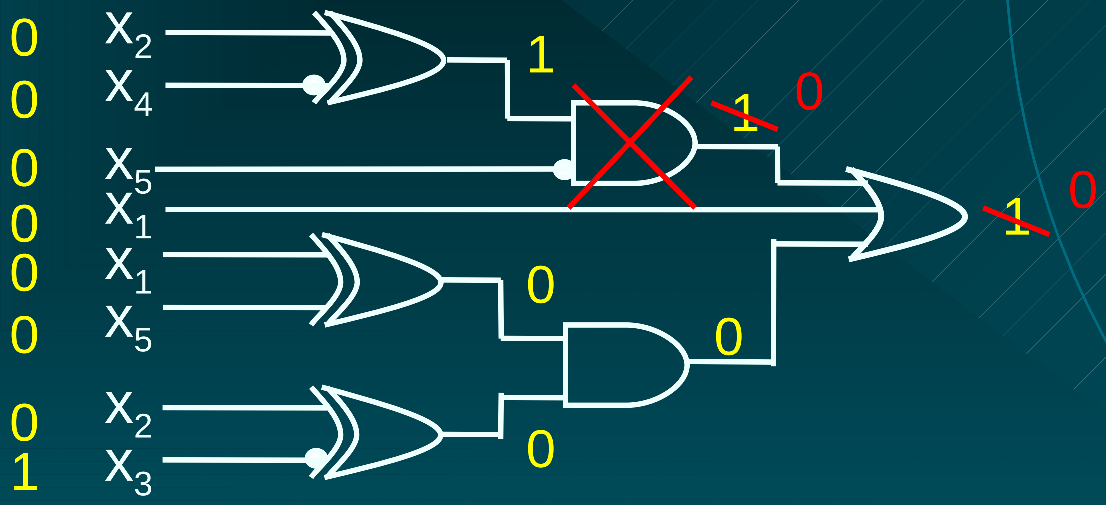
$$x_2\overline{x_3}x_4 + \overline{x_1}x_3\overline{x_4} + \overline{x_1}\overline{x_3}x_4$$

Testability of 2-SPP forms

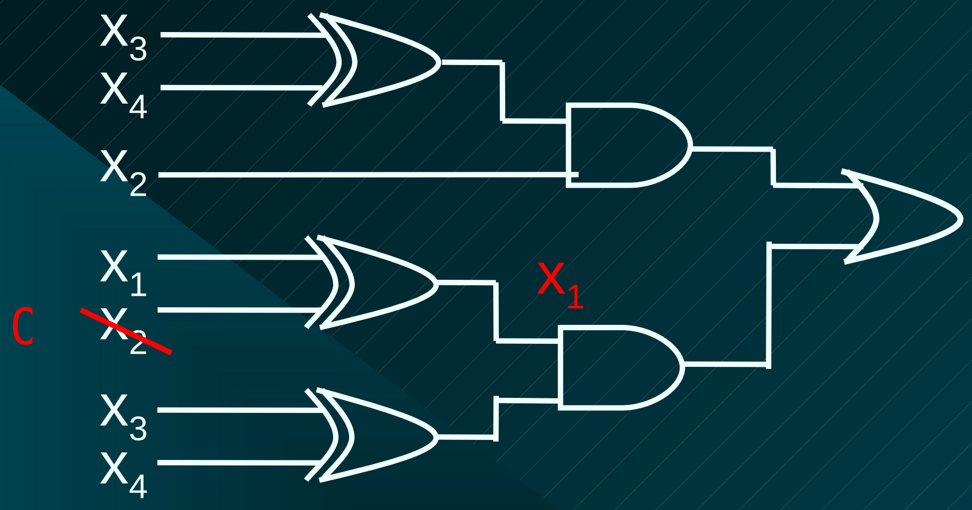
- In collaboration with Rolf Drechsler
- Testability is a major aspect of design process
- Testability of 2-SPP Three-Level Logic Networks.
- Fault models:
  - Stuck at fault
  - Cellular fault

Fault Model

- Fault model: Stuck at fault
  - One input/output of a gate in circuit has a fixed constant value (0 or 1)



Redundancies



$$F = (x_3 \oplus x_4)x_2 + (x_1 \oplus x_2)(x_3 \oplus x_4)$$

$$=$$

$$F_f = (x_3 \oplus x_4)x_2 + x_1(x_3 \oplus x_4)$$

Fully testable networks

- A gate is fully testable if there does not exist redundant fault on it

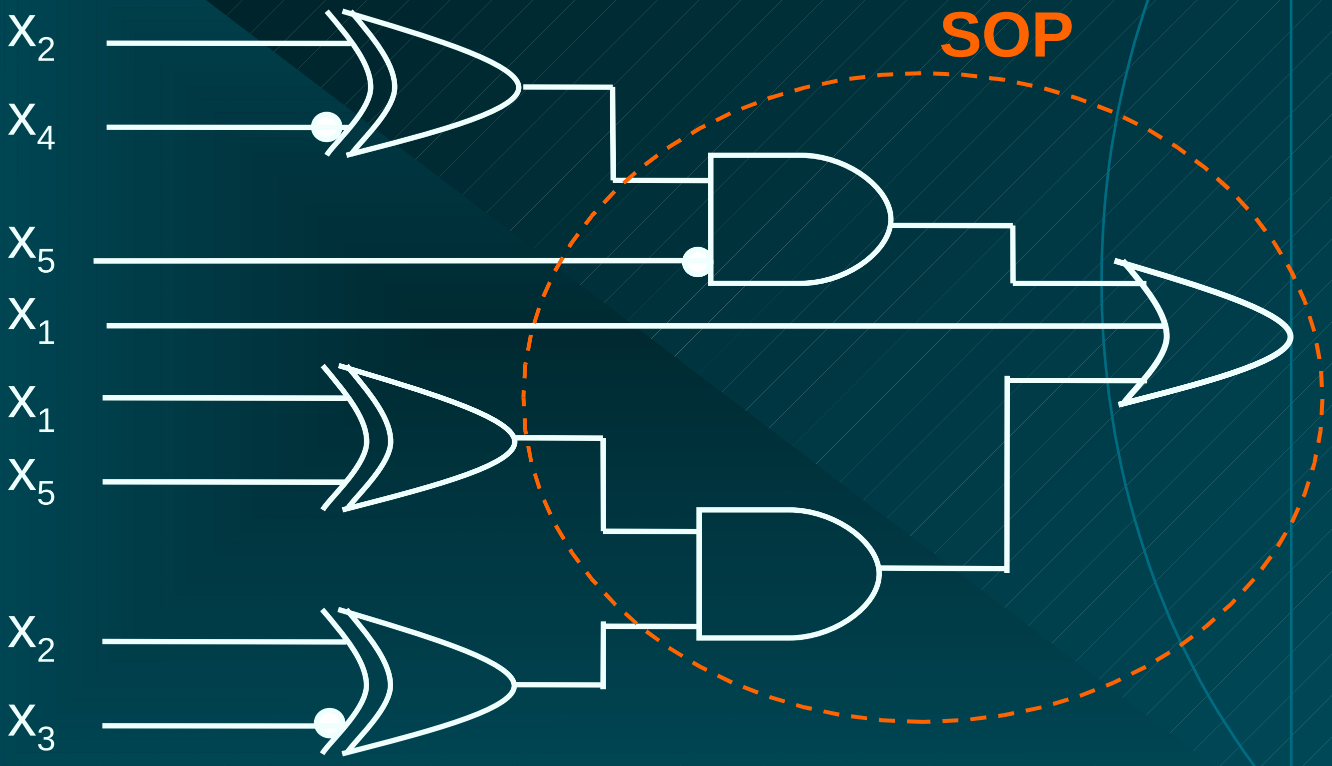
- A circuit is fully testable if all its gates are fully testable.

Our Aim

- Study the testability of 2-SPP networks.
- Are the minimal 2-SPP networks fully testable?
- How can we improve the testability of a network?

2-SPP forms

$$(x_2 \oplus \overline{x}_4)\,\overline{x}_5 + (x_2 \oplus \overline{x}_3)(x_1 \oplus x_5) + x_1$$



Testability

- Prime and irredundant SOP networks are fully testable in the SAFM
- 2-SPP minimal forms contain:
  - EXOR part
  - SOP part
    - prime
    - irredundant
- We must show:
  - EXOR gates are fully testable
  - The inputs to the SOP part can have all possible values

Inputs to the SOP part

$$(x_1 \oplus \overline{x}_2)x_4(x_3 \oplus \overline{x}_5)(x_3 \oplus x_7)\overline{x}_9 =$$

$$(x_1 \oplus \overline{x}_2)x_4(x_3 \oplus \overline{x}_5)(x_3 \oplus x_7)(x_5 \oplus x_7)\overline{x}_9$$

$$\begin{cases} (x_1 \oplus \overline{x}_2) = 1 \\ x_4 = 1 \\ (x_3 \oplus \overline{x}_5) = 1 \\ (x_3 \oplus x_7) = 1 \\ \overline{x}_9 = 1 \end{cases} = \begin{cases} (x_1 \oplus \overline{x}_2) = 1 \\ x_4 = 1 \\ (x_3 \oplus \overline{x}_5) = 1 \\ (x_3 \oplus x_7) = 1 \\ (x_5 \oplus x_7) = 1 \\ \overline{x}_9 = 1 \end{cases}$$

$\boxed{\textbf{System of maximum rank}}$

Testability of 2-SPPs

Main results:

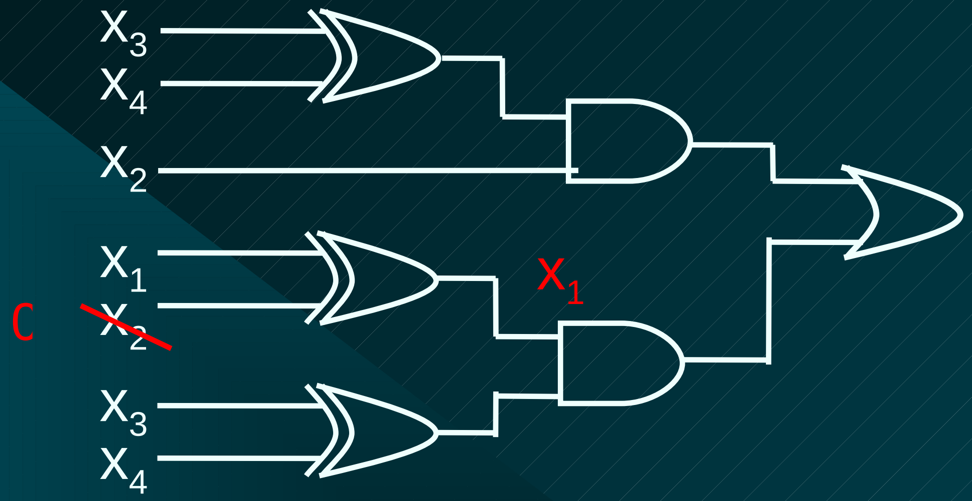
- Theorem: 2-SPP forms minimal w.r.t. the number of *2-pseudoproducts* are

  NOT fully testable

- Theorem: 2-SPP forms minimal w.r.t. the number of *literals* are

  fully testable

Counter-example: Theorem 1



$$F = (x_3 \oplus x_4)x_2 + (x_1 \oplus x_2)(x_3 \oplus x_4)$$

$$=$$

$$F_f = (x_3 \oplus x_4)x_2 + x_1(x_3 \oplus x_4)$$

Theorem 2

Theorem 2: 2-SPP forms minimal w.r.t. the number of *literals* are fully testable

Proof (sketch):

- 2-SPP is a SOP with an upper EXOR level
- The SOP networks are fully testable
- All possible values can be applied to the AND layer (max. rank of the system of EXORs)
- The EXOR gates are fully testable

Improving the testability

- Is the minimality really necessary for testability?
- No
- For SOP forms:
  - Irredundancy (OR)
  - Primality (AND)
- For 2-SPP forms:
  - Irredundancy (OR)
  - AND-Irredundancy (AND)
  - EXOR-Irredundancy (EXOR)

SOP properties

- Irredundancy:
  - A SOP form for a function f is irredundant if deleting any product from it
    - we get a different function
- Primality:
  - A SOP form for a function f is prime if deleting any literal from any product
    - we get a different function

2-SPP properties

- Irredundancy:
  - A 2-SPP form for a function f is irredundant if deleting any 2-pseudoproduct from it
    - we get a different function
- AND-Irredundancy
  - A 2-SPP form for a function f is AND-irredundant if deleting any factor from any 2-pseudoproduct
    - we get a different function

EXOR-Irredundancy

- A 2-SPP form for a function f is EXOR-irredundant if replacing any literal with 0 or 1 in any EXOR factor
  - we get a different function

F = $(x_3 \oplus x_4)x_2 + (x_1 \oplus x_2)(x_3 \oplus x_4)$

$$=$$

$$(x_3 \oplus x_4)x_2 + x_1(x_3 \oplus x_4)$$

Is not EXOR-irredundant!

Minimal 2-SPP forms

- *Definition:* a 2-SPP form is OR-AND-EXOR-irredundant if it satisfies the three properties.
- *Theorem:* OR-AND-EXOR-irredundant 2-SPP forms are fully testable in the SAFM.

- 2-SPP minimal w.r.t. literals:
  - are OR-AND-EXOR- irredundant
- 2-SPP minimal w.r.t. 2-pseudoproducts:
  - are not EXOR- irredundant

Making a network testable

- We try to replace each

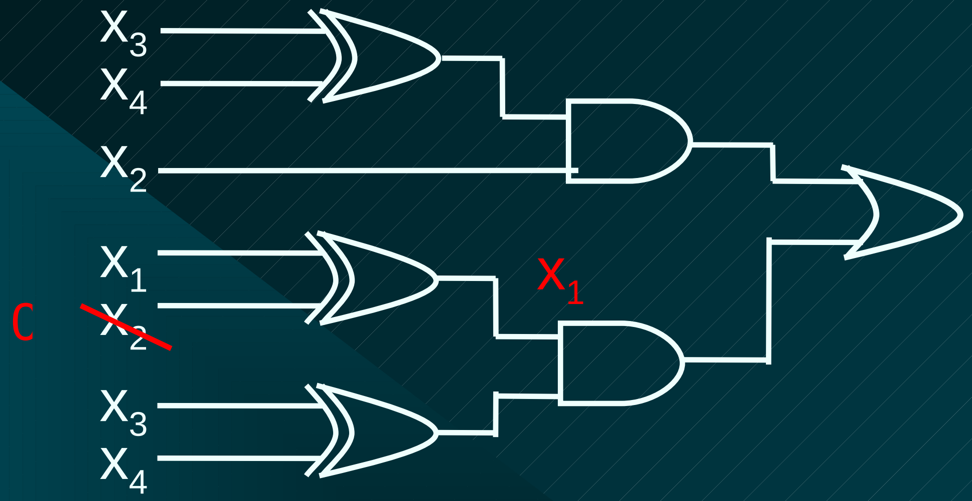
$$(x_i \oplus x_j)\, p$$

with

$x_i\, p$ or $x_j\, p$ or $\overline{x}_i\, p$ or $\overline{x}_j\, p$

without changing the function

Example



$$F = \underbrace{(x_3 \oplus x_4)x_2 + (x_1 \oplus x_2)(x_3 \oplus x_4)}$$

$$F = (x_3 \oplus x_4)x_2 + x_1(x_3 \oplus x_4)$$

Fully testable!

Practical Issues

- The synthesized form could be non-minimal:
  - The set covering phase is not always exact
- We seldom have redundancies in practice
- We can design fully testable non- minimal forms (heuristics)

Metrics

- CMOS:
  - *k* fan-in AND/OR gates cost *k* literals
  - *k* fan-in EXOR gates cost *4(k-1)* literals
    - 2-EXOR gates cost 4 literals:

$$(x_1 \oplus x_2) = \overline{x}_1 x_2 + x_1 \overline{x}_2$$

- FPGA:
  - *k* fan-in AND/OR/EXOR gates cost *k* literals
    - 2-EXOR gates cost 2 literals

Conclusion

- Theoretical results:
  - 2-SPP minimal w.r.t. the number of literals are fully testable
  - 2-SPP minimal w.r.t. the number of 2-pseudoproducts are NOT fully testable
    - But we can make them fully testable
- 2-SPP vs SOP
  - 2-SPP forms are more compact
  - SOP and 2-SPP are fully testable
  - Minimization time for 2-SPP is too high
    - heuristics

EXOR Projected Sum of Products

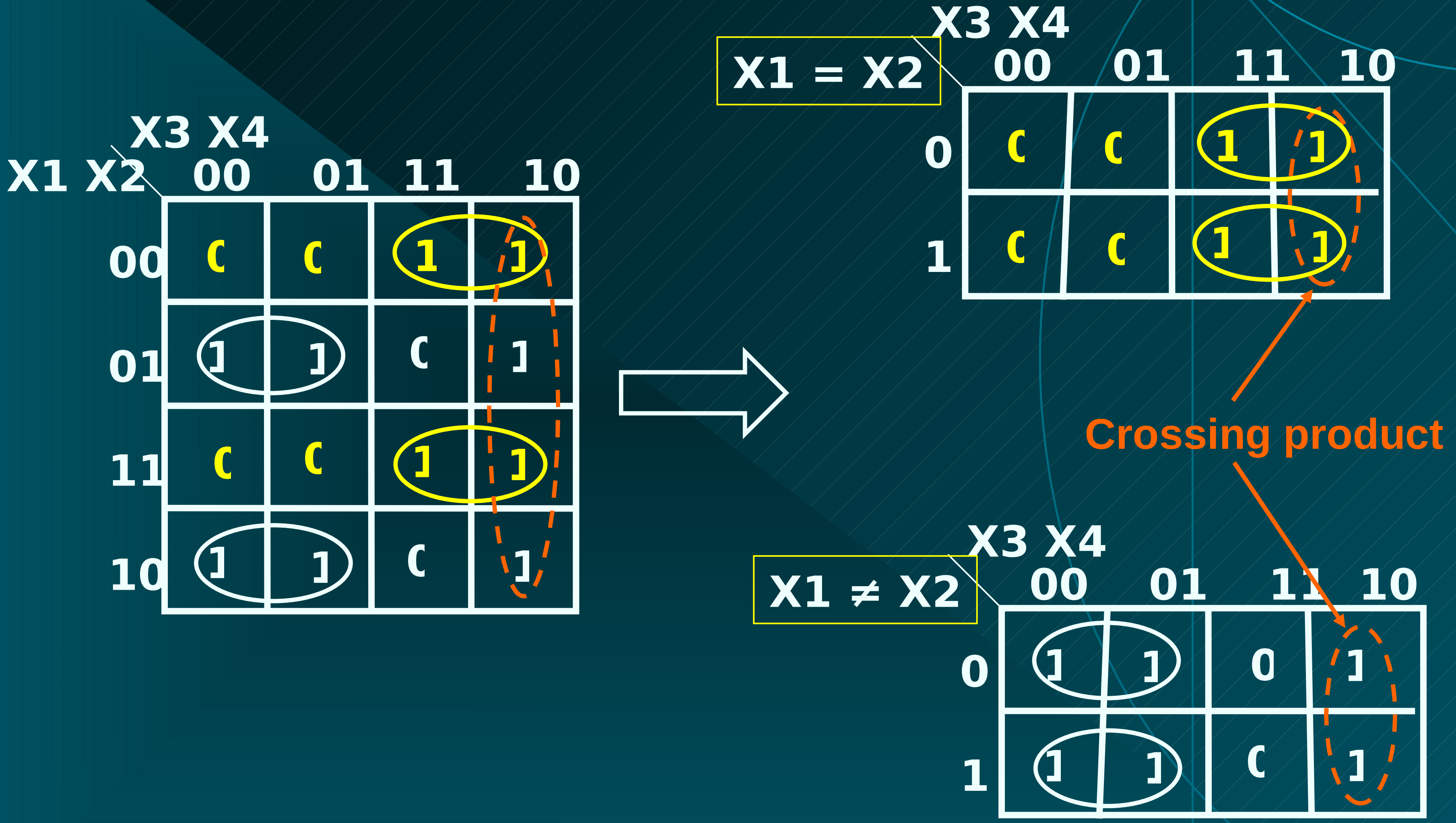
Motivations

- Two level logic (SOP) is the classical approach to logic synthesis
- Three or four level networks
  - are more compact (less area) than SOPs
  - are harder to optimize
- Our purpose is to find a compact form with
  - a bounded number of levels
  - an efficient minimization algorithm

Overview

- Derivation of EP-SOPs (EXOR-Projected Sum of Products) from SOPs
- EP-SOP representation
  - without remainder
  - with remainder
- Projection algorithms
- Minimal EP-SOP forms:
  - Computational complexity (NP^{NP}-hard)
  - Approximation algorithms
- Experimental results

Example SOP vs EP-SOP



X1 X2 \ X3 X4	00	01	11	10
00	0	0	1	1
01	1	1	0	1
11	0	0	1	1
10	1	1	0	1

X1 = X2

X3 X4	00	01	11	10
0	0	0	1	1
1	0	0	1	1

X1 ≠ X2

X3 X4	00	01	11	10
0	1	1	0	1
1	1	1	0	1

Example SOP vs EP-SOP

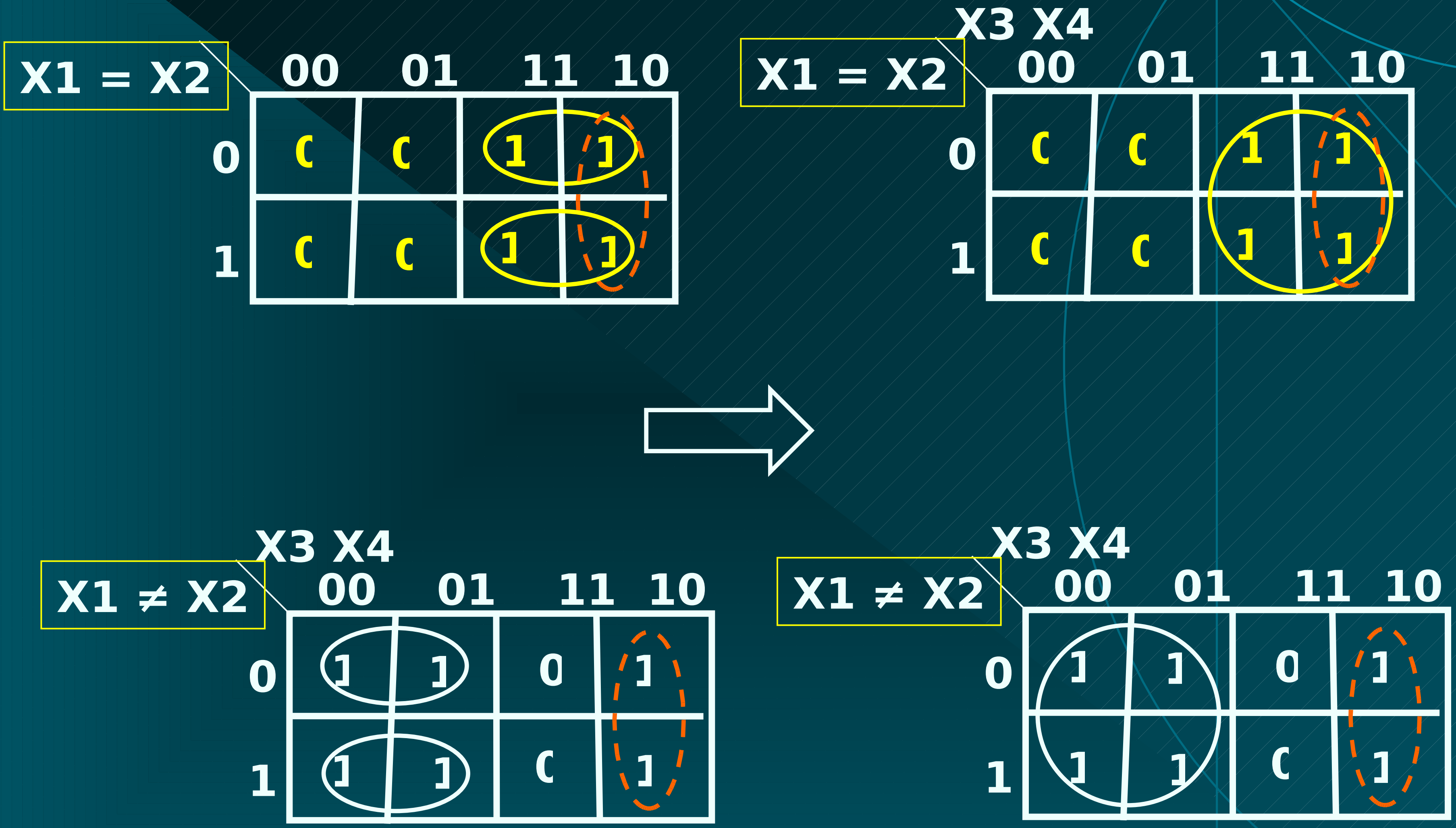
minimal SOP form

$$\bar{x}_1 x_2 \bar{x}_3 + x_1 \bar{x}_2 \bar{x}_3 + \bar{x}_1 \bar{x}_2 x_3 + x_1 x_2 x_3 + x_3 \bar{x}_4$$

EP-SOP form

$$(x_1 \oplus \bar{x}_2)(\bar{x}_2 x_3 + x_2 x_3 + x_3 \bar{x}_4) + (x_1 \oplus x_2)(x_2 \bar{x}_3 + \bar{x}_2 \bar{x}_3 + x_3 \bar{x}_4)$$

Minimization of the EP-SOP



X1 = X2	00	01	11	10
0	0	0	1	1
1	0	0	1	1

X1 = X2 (X3 X4)	00	01	11	10
0	0	0	1	1
1	0	0	1	1

X1 ≠ X2 (X3 X4)	00	01	11	10
0	1	1	0	1
1	1	1	0	1

X1 ≠ X2 (X3 X4)	00	01	11	10
0	1	1	0	1
1	1	1	0	1

Example SOP vs EP-SOP

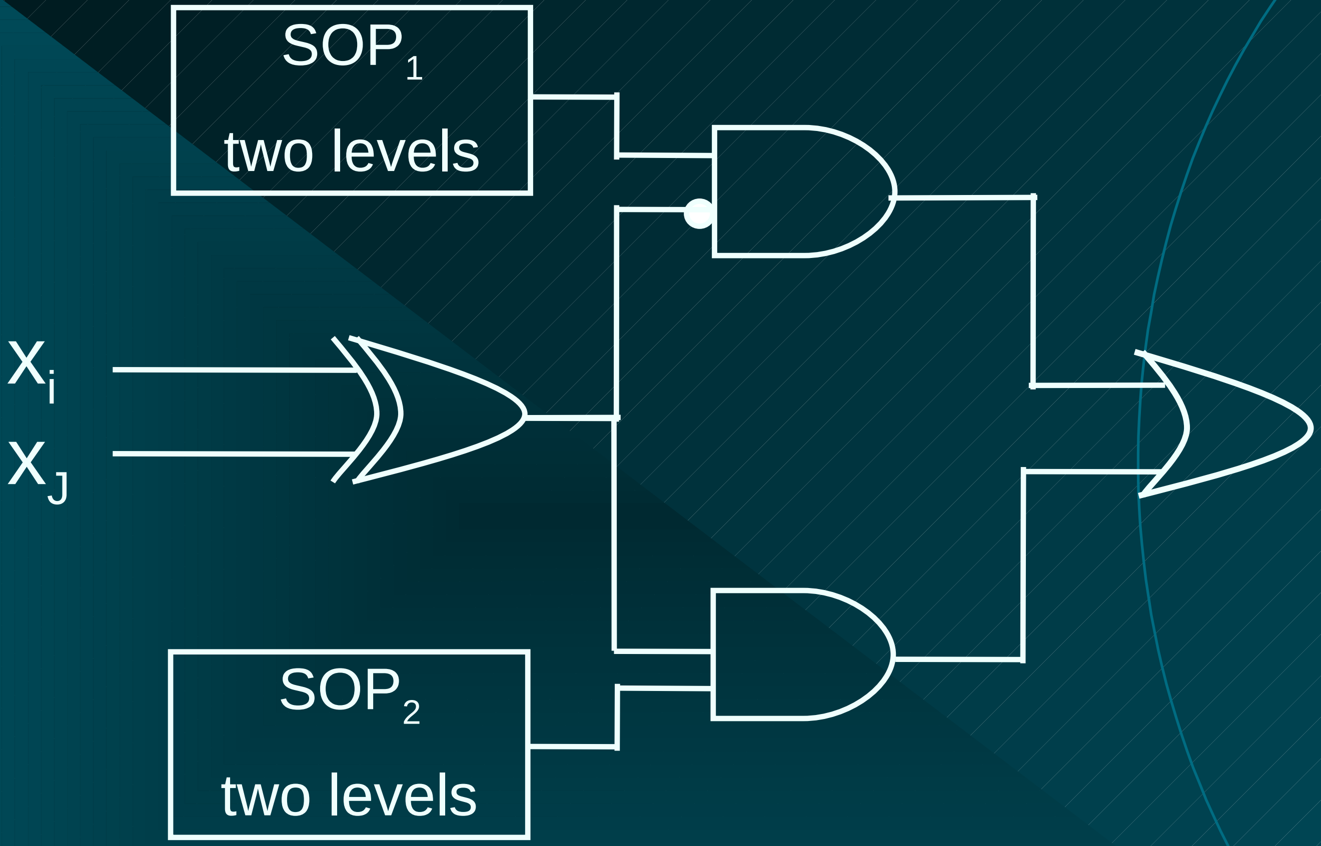
minimal SOP form

$$\bar{x}_1 x_2 \bar{x}_3 + x_1 \bar{x}_2 \bar{x}_3 + \bar{x}_1 \bar{x}_2 x_3 + x_1 x_2 x_3 + x_3 \bar{x}_4$$

minimal EP-SOP form

$$(x_1 \oplus \bar{x}_2) x_3 + (x_1 \oplus x_2)(\bar{x}_3 + x_3 \bar{x}_4)$$

EP-SOP networks



$$(x_i \oplus \overline{x}_j)SOP_1 + (x_i \oplus x_j)SOP_2$$

EP-SOP without remainder

- Given
  - a SOP expression φ
  - a pair of variables x_i and x_j
- The SOP φ is equivalent to

$$\underbrace{(x_i \oplus \overline{x_j})\varphi_{\overline{\oplus}} + (x_i \oplus x_j)\varphi_{\oplus}}_{\textbf{EP-SOP without remainder}}$$

- where:
  - $\varphi_{\overline{\oplus}}$ is the projection of φ in the space $x_i = x_j$
  - $\varphi_{\oplus}$ is the projection of φ in the space $x_i = \overline{x_j}$

EP-SOP without remainder: projection

For each product *p* in in the SOP φ:

- If *p* contains both variables x_i and x_j:
  - it ends up in one of the two SOPs $\varphi_{\oplus}$ and $\varphi_{\overline{\oplus}}$
  - with a literal removal
- If *p* contains one variable or none (crossing):
  - it ends up in both SOPs $\varphi_{\oplus}$ and $\varphi_{\overline{\oplus}}$

Example of projection

min SOP:

$$\bar{x}_1\bar{x}_2x_3 + x_1x_2x_3 + \bar{x}_1x_2\bar{x}_3 + x_1\bar{x}_2\bar{x}_3 + x_3\bar{x}_4$$

EP-SOP:

$$(x_1 \oplus \bar{x}_2)(\bar{x}_2x_3 + x_2x_3 + x_3\bar{x}_4) + (x_1 \oplus x_2)(x_2\bar{x}_3 + \bar{x}_2\bar{x}_3 + x_3\bar{x}_4)$$

The EP-SOP form is not minimal!

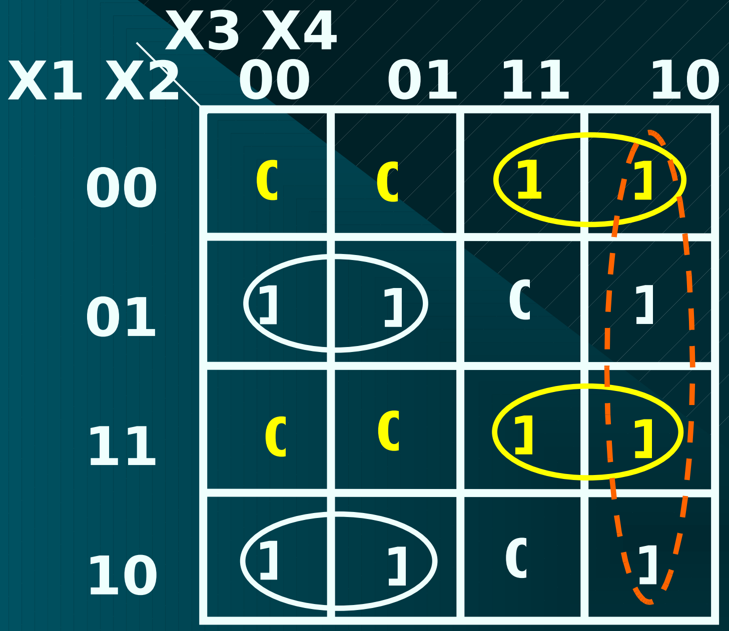
Minimization of the EP-SOP form

EP-SOP:

$$(x_1 \oplus \bar{x}_2)\underbrace{(\bar{x}_2x_3 + x_2x_3 + x_3\bar{x}_4)}_{\boxed{\text{SOP minimization}}} + (x_1 \oplus x_2)\underbrace{(x_2\bar{x}_3 + \bar{x}_2\bar{x}_3 + x_3\bar{x}_4)}_{\boxed{\text{SOP minimization}}}$$

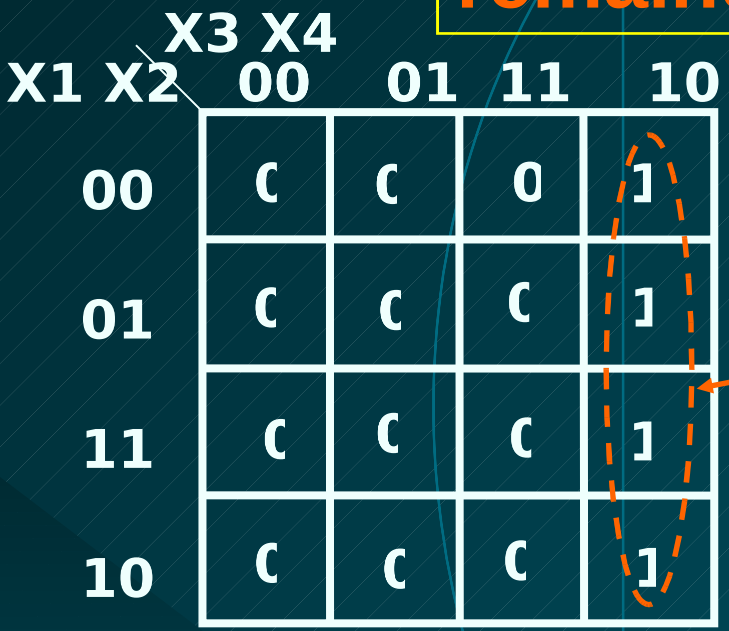
$$(x_1 \oplus \bar{x}_2)x_3 + (x_1 \oplus x_2)(\bar{x}_3 + x_3\bar{x}_4)$$

Example EP-SOP with remainder



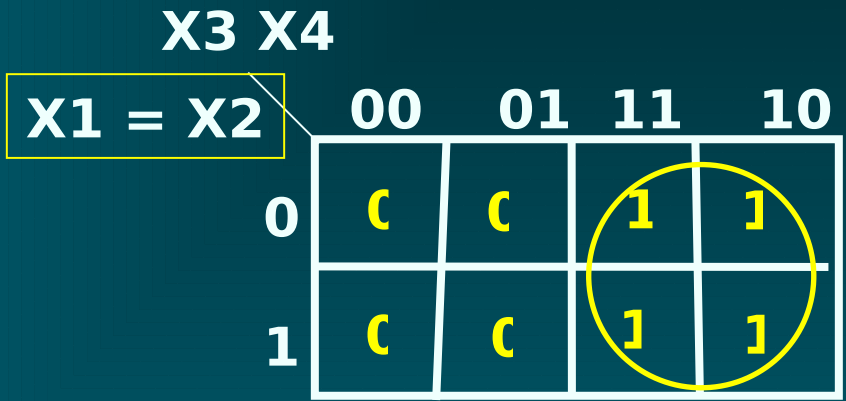
X1 X2 \ X3 X4	00	01	11	10
00	0	0	1	1
01	1	1	0	1
11	0	0	1	1
10	1	1	0	1

remainder

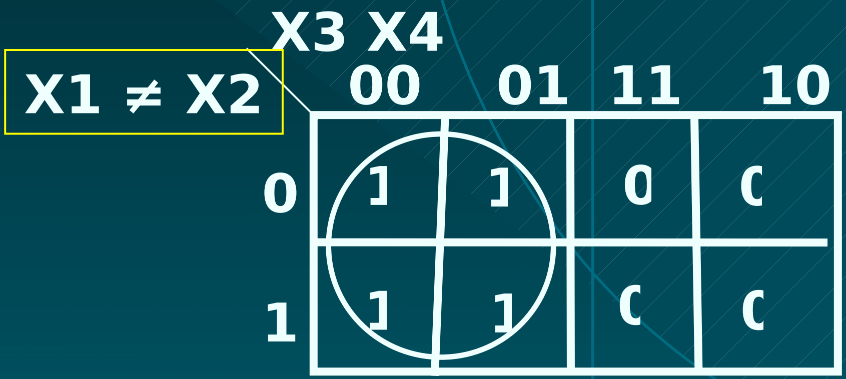


X1 X2 \ X3 X4	00	01	11	10
00	0	0	0	1
01	0	0	0	1
11	0	0	0	1
10	0	0	0	1

Crossing product



X1 = X2 \ X3 X4	00	01	11	10
0	0	0	1	1
1	0	0	1	1



X1 ≠ X2 \ X3 X4	00	01	11	10
0	1	1	0	0
1	1	1	0	0

EP-SOP with remainder

- Consider
  - a SOP expression φ
  - a couple of variables x_i and x_j
- The SOP φ can be written as

remainder

$$\underbrace{(x_i \oplus \overline{x}_j)\varphi_{\overline{\oplus}} + (x_i \oplus x_j)\varphi_{\oplus} + \rho}_{\text{EP-SOP with remainder}}$$

EP-SOP with remainder: projection

Given a SOP φ and two variables x_i and x_j :

For each product *p* in φ

- If *p* contains both variables
  it ends up in one of the two SOPs $\varphi_{\oplus}$ and $\varphi_{\overline{\oplus}}$
- If *p* contains one variable or none (crossing)
  it ends up in the remainder ρ

SOP $\overline{x}_1 x_2 \overline{x}_3 + x_1 \overline{x}_2 \overline{x}_3 + \overline{x}_1 \overline{x}_2 x_3 + x_1 x_2 x_3 + x_3 \overline{x}_4$

EP-SOP $(x_1 \oplus \overline{x}_2) x_3 + (x_1 \oplus x_2) \overline{x}_3 + x_3 \overline{x}_4$

EP-SOP forms

SOP form

$$\bar{x}_1 x_2 \bar{x}_3 + x_1 \bar{x}_2 \bar{x}_3 + \bar{x}_1 \bar{x}_2 x_3 + x_1 x_2 x_3 + x_3 \bar{x}_4$$

EP-SOP form without remainder

$$(x_1 \oplus \bar{x}_2) x_3 + (x_1 \oplus x_2)(\bar{x}_3 + x_3 \bar{x}_4)$$

EP-SOP form with remainder

$$(x_1 \oplus \bar{x}_2) x_3 + (x_1 \oplus x_2) \bar{x}_3 + x_3 \bar{x}_4$$

Minimal forms

SOP and EP-SOP have related sizes

- Does a minimal SOP produce a minimal EP-SOP?
- How to choose x_i and x_j?

Minimal forms

Trivial idea:

- try all variables pairs
- project the SOPs (the projection algorithms are polynomial)
- If φ is an optimal SOP
  - $\varphi_{\oplus}$ and $\varphi_{\overline{\oplus}}$ might be optimal
- Bad news: $\varphi_{\oplus}$ and $\varphi_{\overline{\oplus}}$ are not optimal even if φ is!

Computational complexity

- Even if the original SOP form is minimal, we must further minimize $\varphi_{\oplus}$ and $\varphi_{\overline{\oplus}}$:

$$(x_i \oplus \overline{x}_j)\varphi_{\overline{\oplus}}^{min} + (x_i \oplus x_j)\varphi_{\oplus}^{min}$$

- Minimizing $\varphi_{\oplus}$ and $\varphi_{\overline{\oplus}}$ is as difficult as optimizing a generic SOP form.

- ***Theorem:*** **Even if φ is optimal, minimizing $\varphi_{\oplus}$ and $\varphi_{\overline{\oplus}}$ is an NP^{NP}-hard problem.**

Approximation algorithms

Good news:

- If we choose a good strategy we can produce a near-optimal EP-SOP in polynomial time
- Strategy:
  - Choose the pair of variables appearing in the largest number of products of φ
  - Project φ with respect to that couple
  3. minimize the two projected SOPs with a two-level logic heuristic
- The algorithm is polynomial:
  - $O((n_{var})^2 \cdot n_{prod})$
  - $O(n_{var} \cdot n_{prod})$
  - polynomial (e.g., using Espresso not exact)

Approximation algorithms

Theorem. The resulting number of products is at most:

❖ (4 - 2ν/ |φ|) times the optimum (without remainder)

❖ twice the optimum (with remainder)

even without reoptimizing $\varphi_{\oplus}$ and $\varphi_{\overline{\oplus}}$.

The polynomial reoptimization of the two SOPs can improve the result

Approximation algorithms

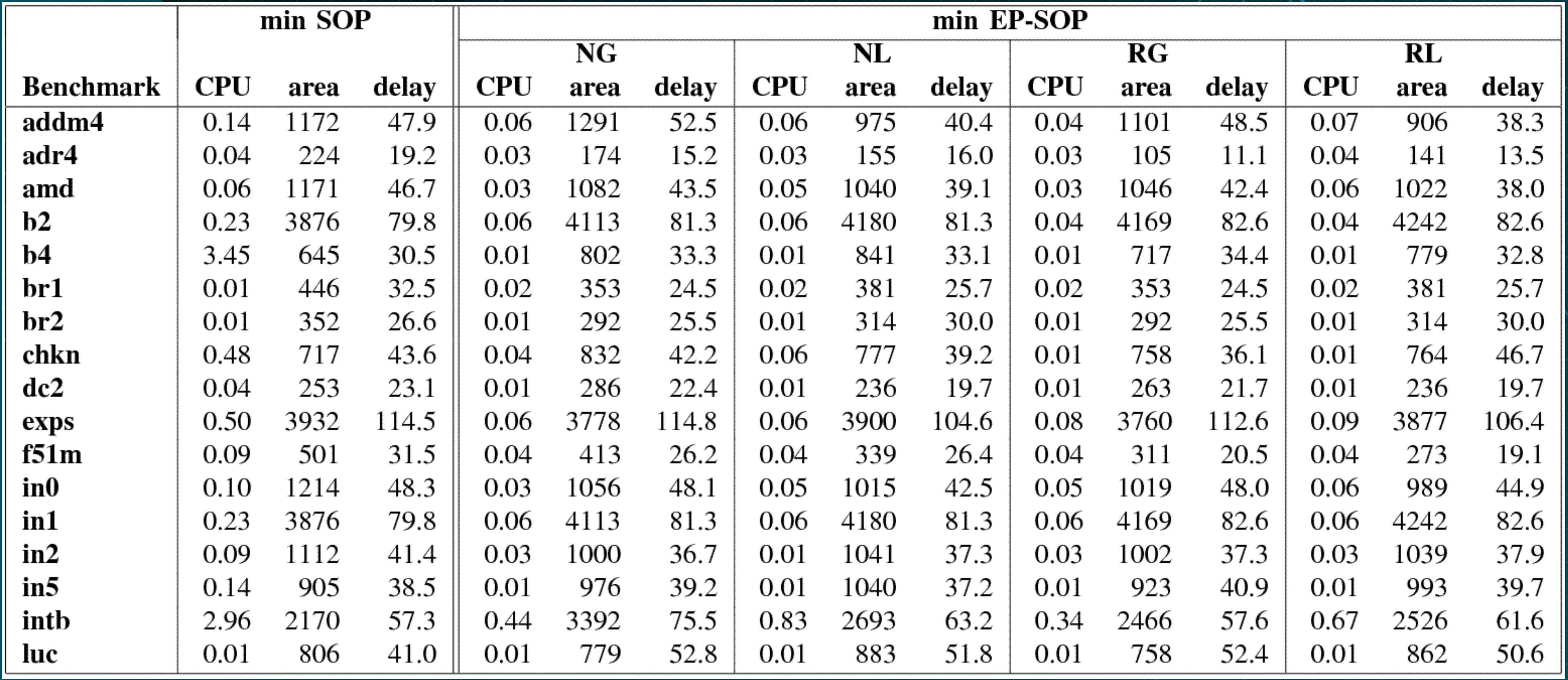
A sketch of the proof:

- The optimal EP-SOP costs at least ½ of the optimal SOP
- Without remainder:
  - the products with both variables appear only once in the projected SOPs
  - the other products appear twice
- With remainder:
  - the products with both variables appear only once in the projected SOPs
  - the other products appear in the remainder

Experimental results (1)

- ESPRESSO benchmark suite
- Four variants of the algorithm
  - without remainder (N) and with remainder (R)
  - with global frequency (G) and local frequency (L) (the same couple of variables for all outputs or a specific couple for each output)
- Physical area and delay computed by SIS
- Pentium 1.6 GHz with 1GB RAM

Experimental results (2)

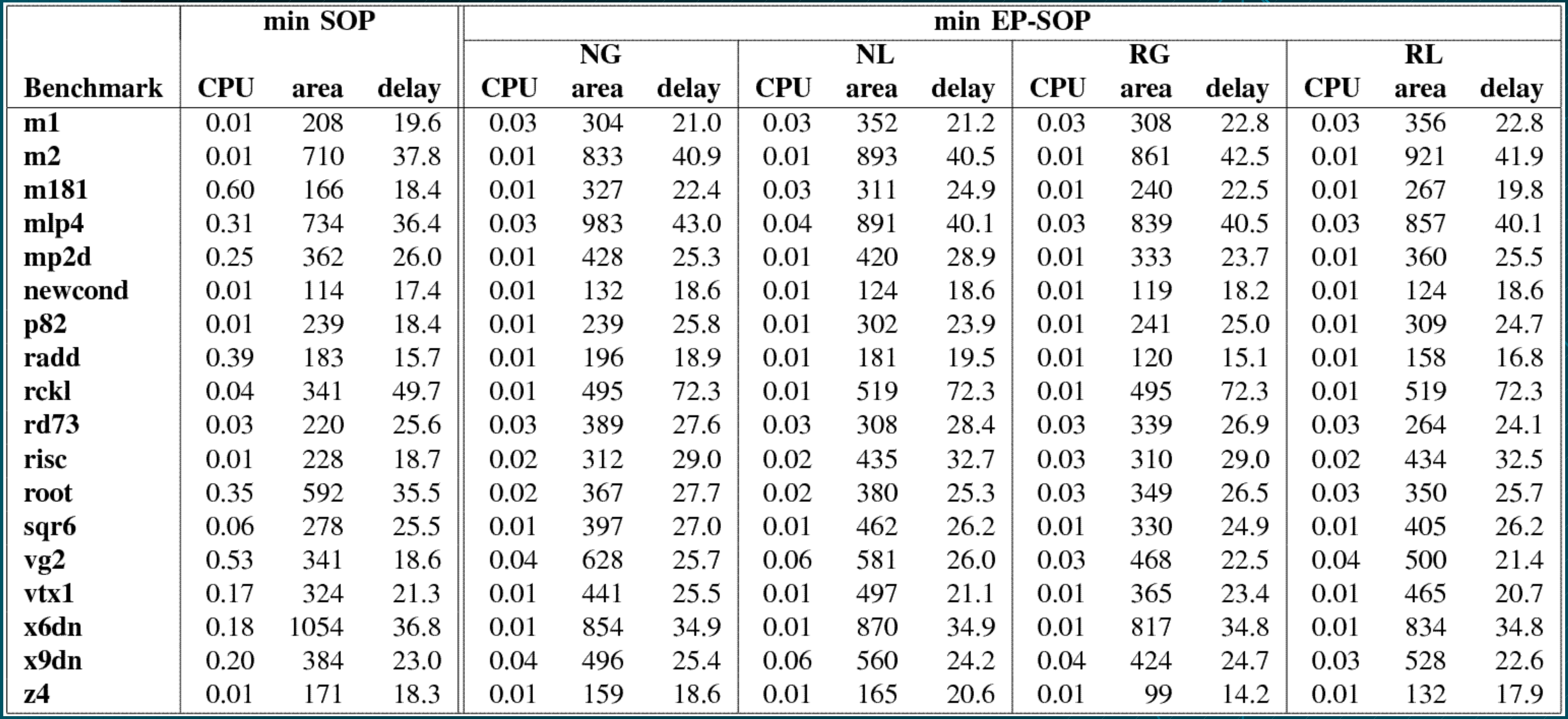


	min SOP			min EP-SOP											
				NG			NL			RG			RL		
Benchmark	CPU	area	delay	CPU	area	delay	CPU	area	delay	CPU	area	delay	CPU	area	delay
addm4	0.14	1172	47.9	0.06	1291	52.5	0.06	975	40.4	0.04	1101	48.5	0.07	906	38.3
adr4	0.04	224	19.2	0.03	174	15.2	0.03	155	16.0	0.03	105	11.1	0.04	141	13.5
amd	0.06	1171	46.7	0.03	1082	43.5	0.05	1040	39.1	0.03	1046	42.4	0.06	1022	38.0
b2	0.23	3876	79.8	0.06	4113	81.3	0.06	4180	81.3	0.04	4169	82.6	0.04	4242	82.6
b4	3.45	645	30.5	0.01	802	33.3	0.01	841	33.1	0.01	717	34.4	0.01	779	32.8
br1	0.01	446	32.5	0.02	353	24.5	0.02	381	25.7	0.02	353	24.5	0.02	381	25.7
br2	0.01	352	26.6	0.01	292	25.5	0.01	314	30.0	0.01	292	25.5	0.01	314	30.0
chkn	0.48	717	43.6	0.04	832	42.2	0.06	777	39.2	0.01	758	36.1	0.01	764	46.7
dc2	0.04	253	23.1	0.01	286	22.4	0.01	236	19.7	0.01	263	21.7	0.01	236	19.7
exps	0.50	3932	114.5	0.06	3778	114.8	0.06	3900	104.6	0.08	3760	112.6	0.09	3877	106.4
f51m	0.09	501	31.5	0.04	413	26.2	0.04	339	26.4	0.04	311	20.5	0.04	273	19.1
in0	0.10	1214	48.3	0.03	1056	48.1	0.05	1015	42.5	0.05	1019	48.0	0.06	989	44.9
in1	0.23	3876	79.8	0.06	4113	81.3	0.06	4180	81.3	0.06	4169	82.6	0.06	4242	82.6
in2	0.09	1112	41.4	0.03	1000	36.7	0.01	1041	37.3	0.03	1002	37.3	0.03	1039	37.9
in5	0.14	905	38.5	0.01	976	39.2	0.01	1040	37.2	0.01	923	40.9	0.01	993	39.7
intb	2.96	2170	57.3	0.44	3392	75.5	0.83	2693	63.2	0.34	2466	57.6	0.67	2526	61.6
luc	0.01	806	41.0	0.01	779	52.8	0.01	883	51.8	0.01	758	52.4	0.01	862	50.6

The area of the XOR gates cannot be neglected (esp. for L)

Nevertheless, in 35% of the cases EP-SOP has a lower area

Experimental results (3)



	min SOP			min EP-SOP											
				NG			NL			RG			RL		
Benchmark	CPU	area	delay	CPU	area	delay	CPU	area	delay	CPU	area	delay	CPU	area	delay
m1	0.01	208	19.6	0.03	304	21.0	0.03	352	21.2	0.03	308	22.8	0.03	356	22.8
m2	0.01	710	37.8	0.01	833	40.9	0.01	893	40.5	0.01	861	42.5	0.01	921	41.9
m181	0.60	166	18.4	0.01	327	22.4	0.03	311	24.9	0.01	240	22.5	0.01	267	19.8
mlp4	0.31	734	36.4	0.03	983	43.0	0.04	891	40.1	0.03	839	40.5	0.03	857	40.1
mp2d	0.25	362	26.0	0.01	428	25.3	0.01	420	28.9	0.01	333	23.7	0.01	360	25.5
newcond	0.01	114	17.4	0.01	132	18.6	0.01	124	18.6	0.01	119	18.2	0.01	124	18.6
p82	0.01	239	18.4	0.01	239	25.8	0.01	302	23.9	0.01	241	25.0	0.01	309	24.7
radd	0.39	183	15.7	0.01	196	18.9	0.01	181	19.5	0.01	120	15.1	0.01	158	16.8
rckl	0.04	341	49.7	0.01	495	72.3	0.01	519	72.3	0.01	495	72.3	0.01	519	72.3
rd73	0.03	220	25.6	0.03	389	27.6	0.03	308	28.4	0.03	339	26.9	0.03	264	24.1
risc	0.01	228	18.7	0.02	312	29.0	0.02	435	32.7	0.03	310	29.0	0.02	434	32.5
root	0.35	592	35.5	0.02	367	27.7	0.02	380	25.3	0.03	349	26.5	0.03	350	25.7
sqr6	0.06	278	25.5	0.01	397	27.0	0.01	462	26.2	0.01	330	24.9	0.01	405	26.2
vg2	0.53	341	18.6	0.04	628	25.7	0.06	581	26.0	0.03	468	22.5	0.04	500	21.4
vtx1	0.17	324	21.3	0.01	441	25.5	0.01	497	21.1	0.01	365	23.4	0.01	465	20.7
x6dn	0.18	1054	36.8	0.01	854	34.9	0.01	870	34.9	0.01	817	34.8	0.01	834	34.8
x9dn	0.20	384	23.0	0.04	496	25.4	0.06	560	24.2	0.04	424	24.7	0.03	528	22.6
z4	0.01	171	18.3	0.01	159	18.6	0.01	165	20.6	0.01	99	14.2	0.01	132	17.9

On average, the best algorithm is RG

The area can reduce by 40%-50% (*adr4, f51m, root, z4*)

Experimental results (4)

- We have compared the results of our heuristics with the optimal EP-SOP:
  - without rest:
    - for the 76% of the benchmarks, the result is optimal
    - for the 88% of the benchmarks, the gap is below 10%
  - with rest:
    - for the 64% of the benchmarks, the result is optimal
    - for the 84% of the benchmarks, the gap is below 10%

Conclusions

- The heuristic algorithm often finds the optimal form
- In 35% of the cases EP-SOP has a lower area
- Projection and reoptimization add a limited time overhead
- This suggests to use EP-SOPs as a fast post-processing step after SOP minimization