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:
  \( \circ = \text{AND} : \text{AND-OR-AND forms} \)
  \( \circ = \text{EXOR} : \text{AND-OR-EXOR forms (EX-SOP)} \)

• OR-AND-OR (Sasao)

• SPP (Luccio, Pagli): EXOR-AND-OR
**SPP forms**

- SPP forms are a direct generalization of SOP forms:
  \[
  (x_1 \oplus x_2 \oplus x_3 \oplus \overline{x_4}) \overline{x_5} + (x_1 \oplus x_2 \oplus \overline{x_3})(x_1 \oplus x_5) + x_1
  \]

- **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:

$$A = \{ p \oplus v | v \in V \} = p \oplus V$$

Affine space

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Translation point

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$$

Vector space

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Pseudocubes

**Product** = characteristic function of a cube

\[ x_1 \cdot x_4 \]

**Pseudoprodouct** = characteristic function of a pseudocube

\[ x_1 \cdot (x_2 \oplus x_3 \oplus \overline{x_4}) \]
A pseudocube can be represented by different pseudoproducts.

One of them is called \text{CEX}.

\[
P = \begin{array}{cccc}
X1 & X2 & X3 & X4 \\
0 & 0 & 1 & 1 \\
0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 \\
\end{array}
\]

\[
\text{CEX}(P) = (x_1 \oplus x_3)(x_1 \oplus x_4) \\
(x_1 \oplus x_3)(x_3 \oplus \overline{x_4}) \\
(x_1 \oplus x_4)(x_3 \oplus \overline{x_4})
\]
Pseudocubes and Affine Spaces

- **Theorem:**
  \[ \text{Pseudocubes} \iff \text{Affine Spaces} \]

- **Corollary:**
  \[ \text{Cubes} \subseteq \text{Affine Spaces} \]

- **Pseudocube can be represented by:**
  - CEX
  - Affine Space: \( p \oplus V \)
Affine Spaces

Pseudoproduct:
\[ x_1 \cdot (x_2 \oplus x_3 \oplus \bar{x}_4) \]

Red: canonical variables
Black: non canonical variables
Cubes as Affine Spaces

Product:

\[ X_1 \cdot X_4 \]

Red: canonical variables

Black: non canonical variables

\[
\begin{array}{cccc}
X_1 & X_2 & X_3 & X_4 \\
1 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{cccc}
X_1 & X_2 & X_3 & X_4 \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 \\
\end{array}
\]

\[
\begin{array}{cccc}
1 & 0 & 0 & 1 \\
\end{array}
\]
The union of two pseudocubes is a pseudocube iff they are affine spaces over the same vector space.

\[ A = p \oplus V, \quad A' = p' \oplus V \quad \text{and} \quad p \oplus p' \not\in V \]

Bases of \( V \) \( v_1, \ldots, v_k \)

\[ A \cup A' = p \oplus V' \]

Bases of \( V' \) \( v_1, \ldots, v_k, p \oplus p' \)
2-SPP forms

\[(x_2 \oplus \overline{x}_4) \overline{x}_5 + (x_2 \oplus \overline{x}_3)(x_1 \oplus x_5) + x_1\]

2-pseudoproduct

2-EXOR
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

Product: \( \overline{X_1} \cdot X_4 \)
2-Pseudocubes

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

\[(x_1 \oplus \bar{x}_2)x_4(x_3 \oplus \bar{x}_5)(x_3 \oplus x_7)\bar{x}_9\]

\[(x_1 \oplus \bar{x}_2)x_4(x_3 \oplus \bar{x}_5)(x_5 \oplus x_7)\bar{x}_9\]

\[(x_1 \oplus \bar{x}_2)x_4(x_3 \oplus x_7)(x_5 \oplus x_7)\bar{x}_9\]
Canonical Representation

\[(x_1 \oplus \bar{x}_2)x_4(x_3 \oplus \bar{x}_5)(x_3 \oplus x_7)\bar{x}_9\]

\[
\begin{align*}
(x_1 \oplus \bar{x}_2) &= 1 \\
x_4 &= 1 \\
(x_3 \oplus \bar{x}_5) &= 1 \\
(x_3 \oplus x_7) &= 1 \\
\bar{x}_9 &= 1
\end{align*}
\]

\[
\begin{align*}
x_1 &= x_2 \\
x_4 &= 1 \\
x_3 &= x_5 \\
x_3 &= \bar{x}_7 \\
\bar{x}_9 &= 1
\end{align*}
\]

\{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\}
Representation of cubes

\( x_2 x_4 x_5 x_7 x_9 \)

\[
\begin{align*}
\overline{x}_2 &= 1 \\
x_4 &= 1 \\
\overline{x}_5 &= 1 \\
x_7 &= 1 \\
\overline{x}_9 &= 1
\end{align*}
\]

\( \{1, x_2, x_4, \overline{x}_5, x_7, \overline{x}_9\} \quad \{x_1\} \quad \{x_3\} \quad \{x_6\} \quad \{x_8\} \)
Structure of 2-pseudoproducts

• **Structure:**

are the sets without complementations

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

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

\[
\{x_1, x_2\} \cup \{1, x_4, \overline{x}_9\} \cup \{x_3, \overline{x}_5, x_7\} \cup \{x_6\} \cup \{x_8\} = \{x_1, x_2\} \cup \{1, x_4, \overline{x}_9\} \cup \{x_3\} \cup \{x_5, \overline{x}_7\} \cup \{x_6\} \cup \{x_8\}
\]
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

\textit{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:

\begin{align*}
0001 & \quad 0010 & \quad 0101 & \quad 0110 & \quad 1101 \\
\{1, \bar{x}_1, \bar{x}_2, x_3, x_4\} & \quad \{1, \bar{x}_1, \bar{x}_2, x_3, \bar{x}_4\} & \quad \{1, \bar{x}_1, x_2, \bar{x}_3, x_4\} & \quad \{1, \bar{x}_1, x_2, x_3, \bar{x}_4\} & \quad \{1, x_1, x_2, \bar{x}_3, x_4\}
\end{align*}

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

\[
\begin{align*}
\{1, \bar{x}_1, \bar{x}_2, \bar{x}_3, x_4\} \cup \{1, \bar{x}_1, \bar{x}_2, x_3, \bar{x}_4\} &= \{1, \bar{x}_1, \bar{x}_2\} \{x_3, \bar{x}_4\} \\
\{1, \bar{x}_1, \bar{x}_2, \bar{x}_3, x_4\} \cup \{1, \bar{x}_1, x_2, \bar{x}_3, x_4\} &= \{1, \bar{x}_1, \bar{x}_3, x_4\} \{x_2\} \\
\end{align*}
\]

\[\ldots\]
An example: the union

Structure:

\{1, x_1, x_2\} \{x_3, x_4\}
\{1, x_1, x_3, x_4\} \{x_2\}
\{1, x_1\} \{x_2, x_3, x_4\}
\{1, x_3, x_4\} \{x_1, x_2\}
\{1\} \{x_1, x_2, x_3, x_4\}
\{1, x_2, x_3, x_4\} \{x_1\}
\{1, x_2\} \{x_1, x_3, x_4\}

Sets:

\{1, \overline{x}_1, \overline{x}_2\} \{x_3, \overline{x}_4\} \text{ and } \{1, \overline{x}_1, x_2\} \{x_3, \overline{x}_4\}
\{1, \overline{x}_1, \overline{x}_3, x_4\} \{x_2\} \text{ and } \{1, \overline{x}_1, x_3, \overline{x}_4\} \{x_2\}
\{1, \overline{x}_1\} \{x_2, \overline{x}_3, x_4\} \text{ and } \{1, \overline{x}_1\} \{x_2, x_3, \overline{x}_4\}
\{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\}
An example

\{1, \overline{x}_1, \overline{x}_2\} \{x_3, \overline{x}_4\} \cup \{1, \overline{x}_1, x_2\} \{x_3, \overline{x}_4\}

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

\{1, \overline{x}_1, \overline{x}_3, x_4\} \{x_2\} \cup \{1, \overline{x}_1, x_3, \overline{x}_4\} \{x_2\}

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

\{1, \overline{x}_1\} \{x_2, \overline{x}_3, x_4\} \cup \{1, \overline{x}_1\} \{x_2, x_3, \overline{x}_4\}

\{1, \overline{x}_1\} \{x_2\} \{x_3, \overline{x}_4\}
An example: set covering

Prime 2-pseudoprodutcs:

\{1, x_3, x_4\} \{x_1, x_2\}
\{1\} \{x_1, x_2, \overline{x_3}, x_4\}
\{1, x_2\} \{x_1, \overline{x_3}, x_4\}
\{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:**
  \[ x_2 \overline{x}_3 x_4 + \overline{x}_1 x_3 \overline{x}_4 + \overline{x}_1 x_3 x_4 \]
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

\[ F = (X_3 \oplus X_4)X_2 + (X_1 \oplus X_2)(X_3 \oplus X_4) \]

\[ F_f = (X_3 \oplus X_4)X_2 + X_1(X_3 \oplus X_4) \]
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**

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

\[
\begin{align*}
(x_1 \oplus \overline{x}_2) &= 1 \\
x_4 &= 1 \\
(x_3 \oplus \overline{x}_5) &= 1 \\
(x_3 \oplus x_7) &= 1 \\
\overline{x}_9 &= 1
\end{align*}
\]

\[
\begin{align*}
(x_1 \oplus \overline{x}_2) &= 1 \\
x_4 &= 1 \\
(x_3 \oplus \overline{x}_5) &= 1 \\
(x_3 \oplus x_7) &= 1 \\
(x_5 \oplus x_7) &= 1 \\
\overline{x}_9 &= 1
\end{align*}
\]

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

\[ \begin{align*}
F &= (X_3 \oplus X_4)X_2 + (X_1 \oplus X_2)(X_3 \oplus X_4) \\
F_f &= (X_3 \oplus X_4)X_2 + X_1(X_3 \oplus X_4)
\end{align*} \]
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
  \[x_i \, p \quad \text{or} \quad x_j \, p \quad \text{or} \quad \overline{x_i} \, p \quad \text{or} \quad \overline{x_j} \, p\]
  without changing the function
Example

\[ F = (X_3 \oplus X_4)X_2 + (X_1 \oplus X_2)(X_3 \oplus X_4) \]

\[ F = (X_3 \oplus X_4)X_2 + X_1(X_3 \oplus X_4) \]

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 ($\text{NP}^{\text{NP}}$-hard)
  • Approximation algorithms

• Experimental results
Example  SOP vs EP-SOP

X1 = X2

X1 ≠ X2

Crossing product
Example SOP vs EP-SOP

**minimal SOP form**

\[ X_1X_2\bar{X}_3 + X_1\bar{X}_2\bar{X}_3 + \bar{X}_1X_2X_3 + X_1X_2X_3 + X_3\bar{X}_4 \]

**EP-SOP form**

\[ (X_1 \oplus \bar{X}_2)(\bar{X}_2X_3 + X_2X_3 + X_3\bar{X}_4) + (X_1 \oplus X_2)(X_2\bar{X}_3 + \bar{X}_2X_3 + X_3\bar{X}_4) \]
Minimization of the EP-SOP

X1 = X2

X3 X4

00 01 11 10
0 1 1 1
1 1 1 1

X3 X4

00 01 11 10
0 1 1 1
1 1 1 1

X1 ≠ X2
Example  SOP vs EP-SOP

**minimal SOP form**

\[ \overline{x_1}x_2\overline{x_3} + x_1\overline{x}_2\overline{x_3} + \overline{x_1}x_2x_3 + x_1x_2x_3 + x_3\overline{x_4} \]

**minimal EP-SOP form**

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

\[(x_i \oplus x_j)\text{SOP}_1 + (x_i \oplus x_j)\text{SOP}_2\]
Given

- a SOP expression $\varphi$
- a pair of variables $x_i$ and $x_j$

The SOP $\varphi$ is equivalent to

$$\left( x_i \oplus \overline{x_j} \right) \varphi_{\oplus} + \left( x_i \oplus x_j \right) \varphi_{\oplus}$$

where:
- $\varphi_{\oplus}$ is the projection of $\varphi$ in the space $X_i = X_j$
- $\varphi_{\oplus}$ is the projection of $\varphi$ in the space $X_i = \overline{X_j}$
EP-SOP without remainder: projection

For each product $p$ in the SOP $\varphi$:

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

min SOP:

\[ \overline{x_1}x_2x_3 + x_1x_2x_3 + \overline{x_1}x_2x_3 + x_1\overline{x}_2x_3 + x_3\overline{x}_4 \]

EP-SOP:

\[ (x_1 \oplus \overline{x}_2)(\overline{x}_2x_3 + x_2x_3 + x_3\overline{x}_4) + (x_1 \oplus x_2)(x_2\overline{x}_3 + \overline{x}_2x_3 + x_3\overline{x}_4) \]

The EP-SOP form is not minimal!
Minimization of the EP-SOP form

EP-SOP:

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

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

Crossing product

remainder
Consider

• a SOP expression $\varphi$

• a couple of variables $x_i$ and $x_j$

• The SOP $\varphi$ can be written as

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

EP-SOP with remainder
Given a SOP $\varphi$ and two variables $x_i$ and $x_j$:

For each product $p$ in $\varphi$

- If $p$ contains both variables it ends up in one of the two SOPs $\varphi_\oplus$ and $\varphi_\odot$
- If $p$ contains one variable or none (crossing) it ends up in the remainder $\rho$

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

**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
\[ \overline{x_1} x_2 x_3 + \overline{x_1} x_2 x_3 + \overline{x_1} x_2 x_3 + \overline{x_1} x_2 x_3 + x_3 x_4 \]

EP-SOP form without remainder
\[ (x_1 \oplus \overline{x_2})x_3 + (x_1 \oplus x_2)(\overline{x_3} + x_3 x_4) \]

EP-SOP form with remainder
\[ (x_1 \oplus \overline{x_2})x_3 + (x_1 \oplus x_2)\overline{x_3} + x_3 x_4 \]
Minimal forms

SOP and EP-SOP have related sizes

- Does a minimal SOP produce a minimal EP-SOP?
- How to choose $x_i$ and $x_j$?
Trivial idea:

- try all variables pairs
- project the SOPs (the projection algorithms are polynomial)
- If $\phi$ is an optimal SOP
  - $\phi \oplus$ and $\phi \ominus$ might be optimal
- Bad news: $\phi \oplus$ and $\phi \ominus$ are not optimal even if $\phi$ is!
**Computational complexity**

- Even if the original SOP form is minimal, we must further minimize \( \varphi_\oplus \) and \( \varphi_\ominus \):

\[
(x_i \oplus x_j) \varphi_\ominus^{\min} + (x_i \oplus \overline{x_j}) \varphi_\oplus^{\min}
\]

- Minimizing \( \varphi_\oplus \) and \( \varphi_\ominus \) is as difficult as optimizing a generic SOP form.

- **Theorem:** Even if \( \varphi \) is optimal, minimizing \( \varphi_\oplus \) and \( \varphi_\ominus \) is an \( \text{NP}^{\text{NP}} \)-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 $\phi$
- Project $\phi$ with respect to that couple
- Minimize the two projected SOPs with a two-level logic heuristic

The algorithm is polynomial:

- $O((n_{\text{var}})^2 \cdot n_{\text{prod}})$
- $O(n_{\text{var}} \cdot n_{\text{prod}})$
- Polynomial (e.g., using Espresso not exact)
Approximation algorithms

**Theorem.** The resulting number of products is at most:

- (4 - 2\( \nu \)/ |\( \varphi \)|) times the optimum (without remainder)
- twice the optimum (with remainder)

Even without reoptimizing \( \varphi_\oplus \) and \( \varphi_\ominus \).

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 $\frac{1}{2}$ 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)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>min SOP</th>
<th>min EP-SOP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU</td>
<td>area</td>
</tr>
<tr>
<td>addm4</td>
<td>0.14</td>
<td>1172</td>
</tr>
<tr>
<td>adr4</td>
<td>0.04</td>
<td>224</td>
</tr>
<tr>
<td>amd</td>
<td>0.06</td>
<td>1171</td>
</tr>
<tr>
<td>b2</td>
<td>0.23</td>
<td>3876</td>
</tr>
<tr>
<td>b4</td>
<td>3.45</td>
<td>645</td>
</tr>
<tr>
<td>br1</td>
<td>0.01</td>
<td>446</td>
</tr>
<tr>
<td>br2</td>
<td>0.01</td>
<td>352</td>
</tr>
<tr>
<td>chk</td>
<td>0.48</td>
<td>717</td>
</tr>
<tr>
<td>dc2</td>
<td>0.04</td>
<td>253</td>
</tr>
<tr>
<td>exps</td>
<td>0.50</td>
<td>3932</td>
</tr>
<tr>
<td>f51m</td>
<td>0.09</td>
<td>501</td>
</tr>
<tr>
<td>in0</td>
<td>0.10</td>
<td>1214</td>
</tr>
<tr>
<td>in1</td>
<td>0.23</td>
<td>3876</td>
</tr>
<tr>
<td>in2</td>
<td>0.09</td>
<td>1112</td>
</tr>
<tr>
<td>in5</td>
<td>0.14</td>
<td>905</td>
</tr>
<tr>
<td>intb</td>
<td>2.96</td>
<td>2170</td>
</tr>
<tr>
<td>luc</td>
<td>0.01</td>
<td>806</td>
</tr>
</tbody>
</table>

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)

On average, the **best algorithm is RG**

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