Università di Pisa
Dipartimento di Informatica

Technical Report: TR-07-01

# An Approximation Algorithm for Fully Testable kEP-SOP Networks

Anna Bernasconi,
Department of Computer Science
University of Pisa
56100 Pisa, Italy
annab@di.unipi.it

Valentina Ciriani, Roberto Cordone
Department of Information Technologies
University of Milano
26013 Crema (CR), Italy
{ciriani, cordone}@dti.unimi.it

January 2, 2007

Address: via F. Buonarroti 2, 56127 Pisa, Italy. Tel: +39 050 2212700 Fax: +39 050 2212726

# An Approximation Algorithm for Fully Testable kEP-SOP Networks

Anna Bernasconi,
Department of Computer Science
University of Pisa
56100 Pisa, Italy
annab@di.unipi.it

Valentina Ciriani, Roberto Cordone
Department of Information Technologies
University of Milano
26013 Crema (CR), Italy
{ciriani, cordone}@dti.unimi.it

January 2, 2007

**Abstract**

Multi-level logic synthesis yields much more compact expressions of a given Boolean function with respect to standard two-level sum of products (SOP) forms. On the other hand, minimizing an expression with more than two-levels can take a large time. In this paper we introduce a novel algebraic four-level expression, named k-EXOR-projected sum of products (kEP-SOP) form, whose synthesis can be performed in polynomial time with an approximation algorithm starting from a minimal SOP. Our experiments show that the resulting networks can be obtained in very short computational time and often exhibit a high quality. We also study the testability of these networks under the Stuck-at-fault model, and show how fully testable circuits can be generated from them by adding at most a constant number of multiplexer gates. Experimental results show the effectiveness of this method both for four-level logic and general multi-level logic synthesis.

## 1 Introduction

A crucial task in logic synthesis is the derivation of high quality networks from an initial specification. The selection of a structure for the final circuit, out of all known models, is critical and the quality of the synthesized circuit depends on different factors, such as area, delay, synthesis time, and testability.

The classical synthesis approach is the two-level logic minimization of *Sum of Products* (SOPs). Big efforts have been done to obtain efficient SOP minimization procedures, which usually involve refined set covering algorithms. The main advantages of SOP synthesis are the small and constant number of levels of the resulting networks (which guarantees a very low delay); their full testability in the *Stuck-at-fault* model; and the availability of very efficient heuristics for their minimization (e.g., Espresso [18, 2], Scherzo [7], SCG [6]). On the other hand, SOP forms are in general not very compact.

In order to obtain networks with smaller area, many multi-level logic expressions have been proposed. Multi-level minimization methods can be divided into two groups, depending on whether a bound on the number of levels in the resulting networks has been fixed or not. Unbounded multi-level minimization algorithms (e.g., SIS [20] and BDD-based circuits [10]) are very fast and the resulting networks are often compact. However, no constraint is given on the number of levels, and therefore the delay of the corresponding networks is not guaranteed to be low. On the contrary,

bounded multi-level forms (e.g., EXSOP [8, 11], OR-AND-OR [9], SPP [5, 17], ESPP [16, 15], EP-SOP [1]), while still keeping a very compact area, result into circuits of constant depth, usually three or four levels. Bounded multi-level forms are therefore quite promising for the high quality of the corresponding networks, but the main problem of these models is the huge minimization time required for their synthesis.

This work is aimed to describe a new bounded multi-level form (with four levels) that can be synthesized in reasonable time, the *k-EXOR-projected sum of products* (kEP-SOP) form. The main idea is to manipulate an already minimal SOP expression in order to derive, in polynomial time, a more compact but still bounded network. Generalizing the strategy described in [1], we project the minimal SOP onto subspaces of the Boolean space $\{0,1\}^n$. These projections reduce the Hamming distances among the cubes appearing in each subspace, so that further merges can be performed using standard SOP heuristics. Due to the high complexity of their synthesis, we minimize kEP-SOP forms with a polynomial time approximation algorithm, whose approximation ratio improves the one derived in [1] for EP-SOP forms.

Our overall method yields a polynomial time approximation algorithm. Recall that both heuristics and approximation algorithms do not guarantee the minimality of their solution, but while we cannot perform any prediction on the result of a heuristic, an approximation algorithm guarantees near-optimum solutions.

We first introduce a 2EP-SOP (k=2) form through an example. Let us consider the Boolean function $f$ that is represented by the optimal SOP $\overline{x}_1x_2\overline{x}_3+x_1\overline{x}_2\overline{x}_3+\overline{x}_1\overline{x}_2x_3+x_1x_2x_3x_4+\overline{x}_4x_5\overline{x}_6+x_4\overline{x}_5\overline{x}_6+\overline{x}_4\overline{x}_5x_6+x_3x_5\overline{x}_6$. We project this SOP onto the spaces $(x_1\oplus x_2)$, $(x_1\oplus\overline{x}_2)$, $(x_4\oplus\overline{x}_5)$, and $(x_4\oplus x_5)$. These four projections return the form $(x_1\oplus x_2)\,\overline{x}_3+(x_1\oplus\overline{x}_2)\,(\overline{x}_2x_3+x_3x_4)+(x_4\oplus x_5)\,\overline{x}_6+(x_4\oplus\overline{x}_5)\,(\overline{x}_5x_6+x_3x_5\overline{x}_6)$. The resulting network is shown in Figure 2.

We can observe that kEP-SOP expressions can be seen as Boolean factorized forms. Factorization of literal terms is a widely studied field in multi-level logic [4, 19]. Most of the proposed methods produce disjoint factorization [13]. In contrast, the factorization of a kEP-SOP form is not disjoint since a literal can stay simultaneously in the projected SOPs and in the corresponding EXORs, as shown in the previous example.

Beside synthesis, testability is a major aspect of the design process. Therefore we study the testability of kEP-SOP networks under the Stuck-at-fault model, and show how, adding at most a constant number of multiplexer gates, fully testable circuits can be generated from them.

Finally we have performed an extensive set of experiments, with the classical benchmark suite of Espresso, in order to validate in practice our theoretical proofs. The experiments show two different results. First, the 1EP-SOP (k=1) forms seem to be sufficient to give compact four-level networks, as 2EP-SOP forms are rarely more compact. Second, for multi-level synthesis (with SIS), 2EP-SOPs and 1EP-SOPs are both useful for obtaining compact networks.

The remainder of this paper is organized as follows. Section 2 recalls some preliminary results from [1]; Section 3 defines kEP-SOP forms. Section 4 presents the approximation algorithm for kEP-SOP synthesis and proves that its solution is nearly optimal. Section 5 studies the testability of kEP-SOPs in the Stuck-at fault model. Finally, Section 6 discusses the experimental results validating the proposed approach.

# 2 Preliminaries

## 2.1 EP-SOP Networks

In this section we recall some basic definitions from [1].

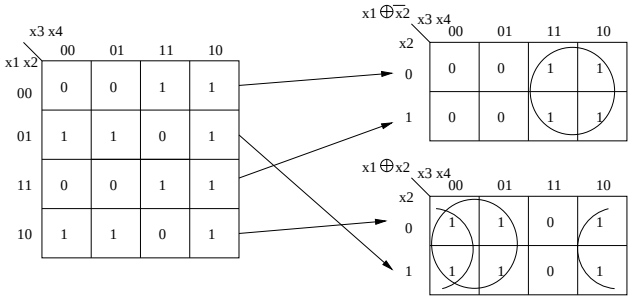



Figure 1: *Karnaugh maps of a function $f$ (left side) and its corresponding projections onto $x_1 \oplus \overline{x}_2$ (right side, top) and $x_1 \oplus x_2$ (right side, bottom).*

Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function depending on $n$ variables $x_1$, $x_2$, ..., $x_n$, and let $\phi$ be a SOP representation of it. Let us consider a couple of variables $x_i$ and $x_j$, where w.l.o.g. $i < j$. The space $\{0,1\}^n$ can be partitioned into two disjoint subspaces: the space defined by the characteristic function $\chi_{\overline{\oplus}} = (x_i \oplus \overline{x}_j)$, i.e., the space where $x_i = x_j$, and its complement defined by the function $\chi_{\oplus} = (x_i \oplus x_j)$, i.e., the space where $x_i \neq x_j$. We can represent the function $f$ as the sum (union) of the two projections of $\phi$, $\phi_{\oplus}$ and $\phi_{\overline{\oplus}}$, onto these two spaces:

$$\xi_{ij} = (x_i \oplus x_j)\phi_{\oplus} + (x_i \oplus \overline{x}_j)\phi_{\overline{\oplus}}\,,$$

The expression $\xi_{ij}$ is called the $(i,j)$-*EP-SOP* of $f$ [1].

**Example 1** *Let us consider the Boolean function $f$ shown on the left side of Figure 1. An optimal SOP representation for $f$ is $\phi = \overline{x}_1x_2\overline{x}_3 + x_1\overline{x}_2\overline{x}_3 + \overline{x}_1\overline{x}_2x_3 + x_1x_2x_3 + x_3\overline{x}_4$. Suppose to project $\phi$ onto the spaces $(x_1 \oplus \overline{x}_2)$ and $(x_1 \oplus x_2)$. The first product $\overline{x}_1x_2\overline{x}_3$ in $\phi$ is projected only onto the space $(x_1 \oplus x_2)$ (where $x_1 \neq x_2$) because it contains both $x_1$ (complemented) and $x_2$; the projected product is $x_2\overline{x}_3$. All other products, but the last one, can be projected in a similar way, onto one of the two spaces. Since the last product $x_3\overline{x}_4$ does not contain $x_1$ and $x_2$, it projects onto both the spaces without any literal removal. The overall procedure returns the form $(x_1 \oplus \overline{x}_2)(\overline{x}_2x_3 + x_2x_3 + x_3\overline{x}_4) + (x_1 \oplus x_2)(x_2\overline{x}_3 + \overline{x}_2\overline{x}_3 + x_3\overline{x}_4)$ . Note that the projected SOP forms are not minimal. Minimizing them we obtain $(x_1 \oplus \overline{x}_2)\,x_3 + (x_1 \oplus x_2)(\overline{x}_3 + \overline{x}_4)$, as shown on the right side of Figure 1.*

As Example 1 shows, the products of a generic SOP $\phi$ can be classified into two subsets: those that are entirely included into one of the two subspaces $x_1 = x_2$ and $x_1 \neq x_2$ and those that intersect both of them, which are called *crossing products* [1] (for example, in Figure 1 the product $x_3\overline{x}_4$). In general, it is questionable whether it is profitable to project a crossing product, since this produces two identical products, which reside into the two subspaces (though sometimes these products can merge with other ones, yielding simplified SOPs). Therefore, we can choose to keep the crossing products in an unprojected SOP, named *remainder*. In this case, the resulting expression is called *EP-SOP with remainder* [1].

For example, for the Boolean function $f$ shown on the left side of Figure 1, if the unique crossing product of $\phi$, $x_3\overline{x}_4$, is inserted in the remainder, the overall projection returns the form: $(x_1 \oplus \overline{x}_2)(\overline{x}_2x_3 + x_2x_3) + (x_1 \oplus x_2)(x_2\overline{x}_3 + \overline{x}_2\overline{x}_3) + x_3\overline{x}_4$. Minimizing the projected SOPs we obtain $(x_1 \oplus \overline{x}_2)\,x_3 + (x_1 \oplus x_2)\,\overline{x}_3 + x_3\overline{x}_4$.

Formally, let $f : \{0,1\}^n \to \{0,1\}$, and let $\phi$ be a SOP representation of $f$. Given a couple of variables $x_i$ and $x_j$, the $(i,j)$-*EP-SOP with remainder* of $f$ is the expression

$$\psi_{ij} = (x_i \oplus x_j)\phi'_{\oplus} + (x_i \oplus \overline{x}_j)\phi'_{\overline{\oplus}} + \rho\,,$$

where $\phi'_{\oplus}$ and $\phi'_{\overline{\oplus}}$ are the two projections of the products of $\phi$ containing both $x_i$ and $x_j$ (possibly complemented) onto the spaces $(x_i \oplus x_j)$ and $(x_i \oplus \overline{x}_j)$, respectively, and $\rho$ is the sum of all crossing products of $\phi$.

# 3 kEP-SOP Forms

In this section we define a new algebraic expression, called *kEP-SOP form*, as a direct generalization of an EP-SOP with remainder. We can observe that the remainder in an EP-SOP depends in general on all the $n$ binary variables, since it contains all crossing products. Our idea is to further project the remainder, by choosing another couple of variables and projecting the crossing products onto the two corresponding subspaces. These new projections could be performed with or without remainder. Of course, as long as one keeps a remainder, one can iterate the procedure.

We can formally define this new expression as follows.

**Definition 1** *Let $f : \{0,1\}^n \to \{0,1\}$, and let $\phi$ be a SOP representation of $f$. Given $k$ couples of variables $(x_{a_i}, x_{b_i})$, with $i = 1, \ldots, k$ and $a_i < b_i$, the* kEP-SOP *of $f$ is the expression*

$$kEP\text{-}SOP(f) = \sum_{i=1}^{k} \left( (x_{a_i} \oplus x_{b_i})\phi^i_{\oplus} + (x_{a_i} \oplus \overline{x}_{b_i})\phi^i_{\overline{\oplus}} \right),$$

*where*

- *for $i = 1, \ldots, k-1$, $\phi^i_{\oplus}$ and $\phi^i_{\overline{\oplus}}$ are the* **projections with remainder** *of $\rho^{i-1}$ (setting, by convention, $\rho^0 = \phi$) onto the spaces $(x_{a_i} \oplus x_{b_i})$ and $(x_{a_i} \oplus \overline{x}_{b_i})$, respectively, and $\rho^i$ is the sum of all crossing products of $\rho^{i-1}$.*
- *$\phi^k_{\oplus}$ and $\phi^k_{\overline{\oplus}}$ are the* **projections without remainder** *of $\rho^{k-1}$ onto the spaces $(x_{a_k} \oplus x_{b_k})$ and $(x_{a_k} \oplus \overline{x}_{b_k})$, respectively.*

Observe that a kEP-SOP with $k = 1$ is an EP-SOP without remainder. Moreover, if in Definition 1 we define $\phi^k_{\oplus}$ and $\phi^k_{\overline{\oplus}}$ as the projections **with** remainder of $\rho^{k-1}$, we obtain a *kEP-SOP with remainder*.

For simplicity, in the following we will consider just the case $k = 2$. As we will point out, all results can be easily extended to the general case

**Definition 2** *Let $f : \{0,1\}^n \to \{0,1\}$, and let $\phi$ be a SOP representation of $f$. Given two non identical couples of variables $(x_i, x_j)$ and $(x_{i'}, x_{j'})$ with $i < j$ and $i' < j'$, the* (i,j,i',j')-2EP-SOP *of $f$ is the expression*

$$\begin{aligned} \zeta_{i,j,i',j'} &= (x_i \oplus x_j)\phi_{\oplus} + (x_i \oplus \overline{x}_j)\phi_{\overline{\oplus}} + \rho = \\ &= (x_i \oplus x_j)\phi_{\oplus} + (x_i \oplus \overline{x}_j)\phi_{\overline{\oplus}} + (x_{i'} \oplus x_{j'})\rho_{\oplus} + (x_{i'} \oplus \overline{x}_{j'})\rho_{\overline{\oplus}}, \end{aligned}$$

*where $\phi_{\oplus}$ and $\phi_{\overline{\oplus}}$ are the* **projections with remainder** *of $\phi$ onto the spaces $(x_i \oplus x_j)$ and $(x_i \oplus \overline{x}_j)$, respectively, $\rho$ is the sum of all crossing products of $\phi$; and $\rho_{\oplus}$ and $\rho_{\overline{\oplus}}$ are the* **projections without remainder** *of $\rho$ onto the spaces $(x_{i'} \oplus x_{j'})$ and $(x_{i'} \oplus \overline{x}_{j'})$, respectively.*

After the projections, we can further minimize the SOPs $\phi_{\oplus}$, $\phi_{\overline{\oplus}}$, $\rho_{\oplus}$, and $\rho_{\overline{\oplus}}$, in order to minimize the 2EP-SOP $\zeta_{iji'j'}$.

**Definition 3** *A* Minimal (i,j,i',j')-2EP-SOP *of $f$ is the expression*

$$\zeta^{(min)}_{i,j,i',j'} = (x_i \oplus x_j)\phi^{(min)}_{\oplus} + (x_i \oplus \overline{x}_j)\phi^{(min)}_{\overline{\oplus}} + (x_{i'} \oplus x_{j'})\rho^{(min)}_{\oplus} + (x_{i'} \oplus \overline{x}_{j'})\rho^{(min)}_{\overline{\oplus}} ,$$

*where $\phi^{(min)}_{\oplus}$ and $\phi^{(min)}_{\overline{\oplus}}$ are two minimal SOP forms representing the projections of $\phi$, and $\rho^{(min)}_{\oplus}$ and $\rho^{(min)}_{\overline{\oplus}}$ are two minimal SOP forms representing the projections of the remainder $\rho$.*

In the previous definitions we have a priori fixed two couples of variables, but we can define a 2EP-SOP expression containing the minimum number of products among all possible minimal 2EP-SOP w.r.t. *any* pair of couples of variables.

Let $|\phi|$ denote the number of products in a SOP $\phi$, and $|\zeta| = |\phi_{\oplus}| + |\phi_{\overline{\oplus}}| + |\rho_{\oplus}| + |\rho_{\overline{\oplus}}|$ the overall number of products in a 2EP-SOP $\zeta$.

**Definition 4** *A* Minimal 2EP-SOP *representation of a Boolean function $f$ is given by a 2EP-SOP expression $\zeta_{MIN}$ such that*

$$|\zeta_{MIN}| = \min_{i,j,i',j'} |\zeta^{(min)}_{i,j,i',j'}| .$$

**Example 2** *Let us consider the Boolean function $f$ represented by the optimal SOP*

$$\phi = \overline{x}_1 x_2 \overline{x}_3 + x_1 \overline{x}_2 \overline{x}_3 + \overline{x}_1 \overline{x}_2 x_3 + x_1 x_2 x_3 x_4 + \overline{x}_4 x_5 \overline{x}_6 + x_4 \overline{x}_5 \overline{x}_6 + \overline{x}_4 \overline{x}_5 x_6 + x_3 x_5 \overline{x}_6 .$$

*We first project $\phi$ onto the spaces $(x_1 \oplus x_2)$ and $(x_1 \oplus \overline{x}_2)$. The first two products $\overline{x}_1 x_2 \overline{x}_3$ and $x_1 \overline{x}_2 \overline{x}_3$ are projected onto the space $(x_1 \oplus x_2)$, while the products $\overline{x}_1 \overline{x}_2 x_3$ and $x_1 x_2 x_3 x_4$ are projected onto the space $(x_1 \oplus \overline{x}_2)$. Since the last four products do not contain both the variables $x_1$ and $x_2$, they are inserted into the remainder: $\rho = \overline{x}_4 x_5 \overline{x}_6 + x_4 \overline{x}_5 \overline{x}_6 + \overline{x}_4 \overline{x}_5 x_6 + x_3 x_5 \overline{x}_6$.*

*Second, we project the remainder $\rho$ onto the spaces $(x_4 \oplus \overline{x}_5)$ and $(x_4 \oplus x_5)$. By Definition 2, the projection of $\rho$ is performed without remainder. Therefore the crossing products are inserted in both spaces, i.e., the unique crossing product $x_3 x_5 \overline{x}_6$ will be inserted in both $(x_4 \oplus \overline{x}_5)$ and $(x_4 \oplus x_5)$ without any literal removal. We then project $\overline{x}_4 x_5 \overline{x}_6$, $x_4 \overline{x}_5 \overline{x}_6$, and $x_3 x_5 \overline{x}_6$ onto $(x_4 \oplus x_5)$, and $\overline{x}_4 \overline{x}_5 x_6$ and $x_3 x_5 \overline{x}_6$ onto the space $(x_4 \oplus \overline{x}_5)$.*

*The overall projections will return the form*

$$\begin{aligned} \zeta_{1,2,4,5} &= (x_1 \oplus x_2)(x_2 \overline{x}_3 + \overline{x}_2 \overline{x}_3) + (x_1 \oplus \overline{x}_2)(\overline{x}_2 x_3 + x_2 x_3 x_4) \\ &+ (x_4 \oplus x_5)(x_5 \overline{x}_6 + \overline{x}_5 \overline{x}_6 + x_3 x_5 \overline{x}_6) + (x_4 \oplus \overline{x}_5)(\overline{x}_5 x_6 + x_3 x_5 \overline{x}_6) . \end{aligned}$$

*Finally, minimizing the projected SOP forms, we obtain the minimal form:*

$$\begin{aligned} \zeta^{(min)}_{1,2,4,5} &= (x_1 \oplus x_2)\overline{x}_3 + (x_1 \oplus \overline{x}_2)(\overline{x}_2 x_3 + x_3 x_4) \\ &+ (x_4 \oplus x_5)\overline{x}_6 + (x_4 \oplus \overline{x}_5)(\overline{x}_5 x_6 + x_3 x_5 \overline{x}_6) . \end{aligned}$$

*The minimal 2EP-SOP network is shown in Figure 2.*

## 4 Synthesis of 2EP-SOP Networks

In [1] it is shown that finding an EP-SOP form minimal w.r.t. a given couple of variables is a hard problem ($NP^{NP}$-hard), even if the input SOP is already minimal. The same holds for the 2EP-SOP minimization problem (and in general for kEP-SOP minimization). Indeed, the problem of finding an EP-SOP form minimal w.r.t. a given couple of variables $x_{i'}$ and $x_{j'}$ can be reduced

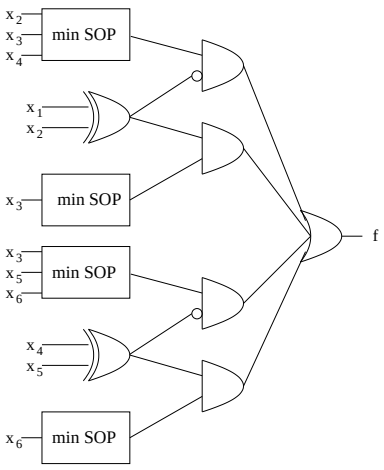



Figure 2: *2EP-SOP network for the function $f$ in Example 2.*

in polynomial time to the problem of finding a minimal $(i, j, i', j')$-2EP-SOP form of $f$. The basic idea of the reduction is that of choosing for the first projections two additional variables $x_i$ and $x_j$ that do not belong to $\{x_1, \ldots, x_n\}$. In this way, the remainder $\rho$ is equal to the entire SOP given in input, and the minimal $(i, j, i', j')$-2EP-SOP of $f$ results in an EP-SOP form minimal w.r.t. $x_{i'}$ and $x_{j'}$.

Due to the complexity of the problem, we must give up exact minimization for fast not exact polynomial algorithms, mirroring what has been done for EP-SOP minimization. There are two possible strategies for not exact minimization: heuristics and approximation algorithms. Both strategies do not guarantee the minimality of their solution, but while we cannot perform any prediction on the result of a heuristic, an approximation algorithm gives guaranteed near-optimum solutions. In a minimization framework, a $p$-approximation algorithm (i.e., an algorithm with approximation ratio $p$) guarantees that the cost $C$ of its solution is such that $C/C^* \leq p$, where $C^*$ is the cost of an optimal solution [14].

In the next two sections we describe a *polynomial approximation algorithm* for the problem of finding the minimal 2EP-SOP representation of a function $f$ starting from a *minimal SOP* $\phi$ for $f$, and we prove that it guarantees a constant approximation ratio.

### 4.1 The Approximation Algorithm

The input to the algorithm is a *minimal SOP* $\phi$ for the function $f$. First of all, we must select the couples of variables $(x_i, x_j)$ and $(x_{i'}, x_{j'})$ for the two projections. Instead of considering all possible pairs of couples of variables, we choose two particular ones driven by the frequencies of their occurrences in the original minimal SOP $\phi$. Once the couples of variables have been chosen, we perform the projections. This can be done in time linear in the size of $\phi$. Finally, the four projected SOPs will be further synthesized with a SOP polynomial heuristic (Espresso not Exact). All steps of the algorithm can be performed in polynomial time. The overall algorithm is described in Figure 3.

**Approximation Algorithm for 2EP-SOP synthesis**

**INPUT:** An optimal SOP form $\phi$ for $f$
**OUTPUT:** A 2EP-SOP form $\zeta$ for $f$

**NOTATION:** let $l$ be a literal and $p$ be a product, $l \in p$ means that $l$ is a literal in $p$, and $l \notin p$ means that $l$ is not a literal in $p$.

**Step 1:** *Selection of the couples* $(x_i, x_j)$ *and* $(x_{i'}, x_{j'})$.
max-freq := 0;
**for any** couple of variables ($x_l$, $x_m$)
  $\phi_{lm}$ := sum of the products in $\phi$ containing both $x_l$ and $x_m$ (possibly complemented);
  $\nu_{lm} := |\phi_{lm}|$;
  $\rho_{lm}$ := sum of the products of $\phi$ that are not in $\phi_{lm}$;
  **for any** couple of variables ($x_p$, $x_q$)
    $\phi_{pq}$ := sum of the products of $\rho_{lm}$ containing both $x_p$ and $x_q$ (possibly complemented);
    $\nu_{pq} := |\phi_{pq}|$;
    **if** (max-freq $< \nu_{lm} + \nu_{pq}$)
      $x_i := x_l$; $x_j := x_m$;
      $x_{i'} := x_p$; $x_{j'} := x_q$;
      max-freq := $\nu_{lm} + \nu_{pq}$;

**Step 2:** *Projections, with remainder* $\rho$, *of* $\phi$ *onto* $(x_i \oplus x_j)$ *and* $(x_i \oplus \overline{x}_j)$.
$\phi_{\oplus} := 0$; $\phi_{\overline{\oplus}} := 0$; $\rho := 0$;
**for any** product $p$ in $\phi$
  **if** $(x_i, x_j \in p)$ $\phi_{\overline{\oplus}} := \phi_{\overline{\oplus}} + x_j q$;
  **else if** $(x_i, \overline{x}_j \in p)$ $\phi_{\oplus} := \phi_{\oplus} + \overline{x}_j q$;
  **else if** $(\overline{x}_i, x_j \in p)$ $\phi_{\oplus} := \phi_{\oplus} + x_j q$;
  **else if** $(\overline{x}_i, \overline{x}_j \in p)$ $\phi_{\overline{\oplus}} := \phi_{\overline{\oplus}} + \overline{x}_j q$;
  **else** $\rho := \rho + p$; //$p$ is a crossing product

**Step 3:** *Projections, without remainder, of* $\rho$ *onto* $(x_{i'} \oplus x_{j'})$ *and* $(x_{i'} \oplus \overline{x}_{j'})$.
$\rho_{\oplus} := 0$; $\rho_{\overline{\oplus}} := 0$;
**for any** product $p$ in $\rho$
  **if** $((x_{i'}, x_{j'} \in p) \vee (x_{i'} \in p \ \wedge \ \overline{x}_{j'}, x_{j'} \notin p) \vee (\overline{x}_{i'}, x_{i'} \notin p \ \wedge \ x_{j'} \in p))$ $\rho_{\overline{\oplus}} := \rho_{\overline{\oplus}} + x_{j'} q$;
  **if** $((x_{i'}, \overline{x}_{j'} \in p) \vee (x_{i'} \in p \ \wedge \ \overline{x}_{j'}, x_{j'} \notin p) \vee (\overline{x}_{i'}, x_{i'} \notin p \ \wedge \ \overline{x}_{j'} \in p))$ $\rho_{\oplus} := \rho_{\oplus} + \overline{x}_{j'} q$;
  **if** $((\overline{x}_{i'}, x_{j'} \in p) \vee (\overline{x}_{i'} \in p \ \wedge \ \overline{x}_{j'}, x_{j'} \notin p) \vee (\overline{x}_{i'}, x_{i'} \notin p \ \wedge \ x_{j'} \in p))$ $\rho_{\oplus} := \rho_{\oplus} + x_{j'} q$;
  **if** $((\overline{x}_{i'}, \overline{x}_{j'} \in p) \vee (\overline{x}_{i'} \in p \ \wedge \ \overline{x}_{j'}, x_{j'} \notin p) \vee (\overline{x}_{i'}, x_{i'} \notin p \ \wedge \ \overline{x}_{j'} \in p))$ $\rho_{\overline{\oplus}} := \rho_{\overline{\oplus}} + \overline{x}_{j'} q$;
  **if** $(x_{i'}, \overline{x}_{i'}, x_{j'}, \overline{x}_{j'} \notin p)$ $\rho_{\oplus} := \rho_{\oplus} + p$; $\rho_{\overline{\oplus}} := \rho_{\overline{\oplus}} + p$;

**Step 4:** *Heuristic SOP minimization of the four projected SOPs computed by Step 2 and Step 3.*
$\phi_{\oplus}^{(min)}$ = EspressoNotExact ($\phi_{\oplus}$);
$\phi_{\overline{\oplus}}^{(min)}$ = EspressoNotExact ($\phi_{\overline{\oplus}}$);
$\rho_{\oplus}^{(min)}$ = EspressoNotExact ($\rho_{\oplus}$);
$\rho_{\overline{\oplus}}^{(min)}$ = EspressoNotExact ($\rho_{\overline{\oplus}}$);
$\zeta := (x_i \oplus x_j)\phi_{\oplus}^{(min)} + (x_i \oplus \overline{x}_j)\phi_{\overline{\oplus}}^{(min)} + (x_{i'} \oplus x_{j'})\rho_{\oplus}^{(min)} + (x_{i'} \oplus \overline{x}_{j'})\rho_{\overline{\oplus}}^{(min)}$;
**return** $\zeta$;

Figure 3: Approximation Algorithm for 2EP-SOP minimization.

### 4.2 Analysis of the Algorithm

We now prove that the proposed synthesis strategy is indeed an approximation algorithm for the 2EP-SOP minimization problem. In order to prove that the cost $|\zeta|$ of our solution is such that $|\zeta|/|\zeta_{MIN}|$ is upper bounded by a constant, where $|\zeta_{MIN}|$ is the cost of an optimal solution, we first find a lower bound for $|\zeta_{MIN}|$, as shown in the following lemma, and then an upper bound for $|\zeta|$, as shown in Theorem 1.

**Lemma 1** *Given a minimal SOP form $\phi$ for a Boolean function $f$, and a minimal 2EP-SOP $\zeta_{MIN}$*

$$|\zeta_{MIN}| \geq \frac{1}{2}|\phi| .$$

**Proof.** Suppose that

$$\zeta_{MIN} = (x_i \oplus x_j)\phi_{\oplus}^{(min)} + (x_i \oplus \overline{x}_j)\phi_{\overline{\oplus}}^{(min)} + (x_{i'} \oplus x_{j'})\rho_{\oplus}^{(min)} + (x_{i'} \oplus \overline{x}_{j'})\rho_{\overline{\oplus}}^{(min)} .$$

We can build a SOP $\sigma$ for $f$ starting from $\zeta_{MIN}$. Let $\phi_{\oplus}^{(min)} = \sum_{h=1}^{|\phi_{\oplus}^{(min)}|} p_h$, $\phi_{\overline{\oplus}}^{(min)} = \sum_{h=1}^{|\phi_{\overline{\oplus}}^{(min)}|} q_h$, $\rho_{\oplus}^{(min)} = \sum_{h=1}^{|\rho_{\oplus}^{(min)}|} r_h$, and $\rho_{\overline{\oplus}}^{(min)} = \sum_{h=1}^{|\rho_{\overline{\oplus}}^{(min)}|} s_h$. Thus

$$\begin{aligned}
\sigma &= (x_i \oplus x_j) \sum_{h=1}^{|\phi_{\oplus}^{(min)}|} p_h + (x_i \oplus \overline{x}_j) \sum_{h=1}^{|\phi_{\overline{\oplus}}^{(min)}|} q_h + (x_{i'} \oplus x_{j'}) \sum_{h=1}^{|\rho_{\oplus}^{(min)}|} r_h + (x_{i'} \oplus \overline{x}_{j'}) \sum_{h=1}^{|\rho_{\overline{\oplus}}^{(min)}|} s_h = \\
&= x_i\overline{x}_j \sum_{h=1}^{|\phi_{\oplus}^{(min)}|} p_h + \overline{x}_i x_j \sum_{h=1}^{|\phi_{\oplus}^{(min)}|} p_h + x_i x_j \sum_{h=1}^{|\phi_{\overline{\oplus}}^{(min)}|} q_h + \overline{x}_i\overline{x}_j \sum_{h=1}^{|\phi_{\overline{\oplus}}^{(min)|}|} q_h + \\
&\quad + x_{i'}\overline{x}_{j'} \sum_{h=1}^{|\rho_{\oplus}^{(min)}|} r_h + \overline{x}_{i'}x_{j'} \sum_{h=1}^{|\rho_{\oplus}^{(min)}|} r_h + x_{i'}x_{j'} \sum_{h=1}^{|\rho_{\overline{\oplus}}^{(min)}|} s_h + \overline{x}_{i'}\overline{x}_{j'} \sum_{h=1}^{|\rho_{\overline{\oplus}}^{(min)|}|} s_h .
\end{aligned}$$

Observe that

$$|\sigma| = 2|\zeta_{MIN}| .$$

Since $\phi$ is minimal, $|\sigma| \geq |\phi|$ and this implies $2|\zeta_{MIN}| \geq |\phi|$. Thus the thesis immediately follows. ■

We now prove that if we project the starting minimal SOP $\phi$ with respect to the two couples of variables selected in Step 1 of the algorithm in Figure 3, we get a solution whose approximation ratio is bounded by a constant.

For a couple of variables $x_i$ and $x_j$, let us denote with $\nu_{ij}$ the number of products in $\phi$ containing both $x_i$ and $x_j$, possibly complemented, and with $\nu_{i'j'}$ the number of products in the remainder $\rho$ containing both $x_{i'}$ and $x_{j'}$, possibly complemented.

**Theorem 1** *Let $\zeta_{MIN}$ be a minimal 2EP-SOP of a Boolean function $f$, and $\phi$ a minimal SOP form for $f$. Let $\zeta$ be the minimal $(i, j, i', j')$-2EP-SOP computed by the algorithm in Figure 3. Then*

$$\frac{|\zeta|}{|\zeta_{MIN}|} \leq 4 - \frac{2(\nu_{ij} + \nu_{i'j'})}{|\phi|} .$$

**Proof.** First observe that, after the projections performed in Step 2 of the algorithm, the number of products in the resulting expression is not changed, and it is exactly equal to $|\phi|$. In particular, $|\phi_{\oplus}| + |\phi_{\overline{\oplus}}| = \nu_{ij}$ and $|\rho| = |\phi| - \nu_{ij}$. With the projections of Step 3, the $|\rho| - \nu_{i'j'} = |\phi| - \nu_{ij} - \nu_{i'j'}$ products of $\rho$ not containing both $x_{i'}$ and $x_{j'}$, are added to $\rho_{\oplus}$ and $\rho_{\overline{\oplus}}$. Therefore

$$\begin{aligned} |\zeta| &= |\phi_{\oplus}| + |\phi_{\overline{\oplus}}| + |\rho_{\oplus}| + |\rho_{\overline{\oplus}}| \\ &= \nu_{ij} + \nu_{i'j'} + 2(|\phi| - \nu_{ij} - \nu_{i'j'}) \\ &= 2|\phi| - \nu_{ij} - \nu_{i'j'} \,. \end{aligned}$$

Finally, by Lemma 1 we get

$$\frac{|\zeta|}{|\zeta_{MIN}|} \leq \frac{2|\phi| - \nu_{ij} - \nu_{i'j'}}{|\phi|/2} = 4 - \frac{2(\nu_{ij} + \nu_{i'j'})}{|\phi|} \,.$$

■

Observe that in the best case $\nu_{ij} + \nu_{i'j'} = |\phi|$, thus the bound becomes

$$\frac{|\zeta|}{|\zeta_{MIN}|} \leq 2 \,.$$

Such a bound could be certainly reached by considering a kEP-SOP representation with a sufficiently large $k$. On the other hand, this could cause the insertion of many EXOR gates into the final network, increasing its overall cost. Comparing this result with the one proved in [1], we have that the bound for 2EP-SOP forms is better then the one given for EP-SOP forms, which is $\frac{|\zeta|}{|\zeta_{MIN}|} \leq 4 - \frac{2\nu_{ij}}{|\phi|}$.

# 5 Testability of 2EP-SOP Networks

Beside synthesis, testability is a major aspect of the design process. In this section the testability of minimal 1EP-SOP (i.e., EP-SOP), with and without remainder, is studied from a theoretical point of view under the Stuck-At Fault Model (SAFM). The results are then generalized to the case of 2EP-SOP networks.

As we will show, minimal EP-SOP networks without remainder turn out to be fully testable under the considered model, while the full testability of minimal EP-SOPs with remainder and of 2EP-SOPs can be guaranteed by adding to the networks a constant number of extra inputs and multiplexer gates.

Let $C$ be any combinational logic circuit, a fault in the SAFM [3] fixes exactly one input or output pin of a node in $C$ to constant value (0 or 1) independently of the values applied to the primary inputs of the circuit. In the following we simply speak of stuck-at-0 (s-a-0) and stuck-at-1 (s-a-1) faults.

The construction of complete test sets requires the determination of the faults which are not testable (i.e., *redundant faults*), even though it is easy to see that in general the detection of redundancies is *coNP-complete*. Redundancies may also invalidate tests for testable faults and often correspond to locations of the circuit where area is wasted [3]. For this reason, synthesis procedures which result in non-redundant circuits are desirable.

A node $v$ in $C$ is called *fully testable*, if there does not exist a redundant fault with fault location $v$. If all nodes in $C$ are fully testable, then $C$ is *fully testable*.

### 5.1 Testability of EP-SOPs Without Remainder

Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function and let $\phi$ be a minimal SOP for $f$. For any two variables $x_i$ and $x_j$, both appearing in $\phi$, let

$$\xi_{ij} = (x_i \oplus x_j)\phi_{\oplus}^{(min)} + (x_i \oplus \overline{x}_j)\phi_{\overline{\oplus}}^{(min)},$$

be the *minimal* $(i,j)$-EP-SOP *without remainder representation* of $f$, where $\phi_{\oplus}^{(min)}$ and $\phi_{\overline{\oplus}}^{(min)}$ are two minimal SOPs representing the projections of $\phi$ onto the spaces $(x_i \oplus x_j)$ and $(x_i \oplus \overline{x}_j)$. The network representation of $\xi_{ij}$ is shown in Figure 4(a).

**Theorem 2** *Minimal $(i,j)$-EP-SOP forms without remainder are fully testable in the SAFM.*

**Proof.** Let $\xi_{ij}$ be a minimal $(i,j)$-EP-SOP form, without remainder, of a Boolean function $f$, and let $C$ be its corresponding circuit representation. To prove the full testability of $C$ we must consider five cases.

1. **An error occurs in the circuit representation of $\phi_{\oplus}^{(min)}$.**

   Since prime and irredundant SOPs are fully testable in the SAFM, and $\phi_{\oplus}^{(min)}$ is minimal (therefore prime and irredundant), we only have to show that the error can be propagated to the output of $C$, in order to be tested.

   This can be done by setting $x_i \neq x_j$. In this way, $C$ computes exactly $\phi_{\oplus}^{(min)}$, and all possible values can be applied to its inputs (remember that $\phi_{\oplus}^{(min)}$ does not depend on $x_i$).

2. **An error occurs in the circuit representation of $\phi_{\overline{\oplus}}^{(min)}$.**

   As in the previous case, we only have to show that the error can be propagated to the output of the network $C$, so that it can be tested. This can be done by setting $x_i = x_j$. In fact, in this way, $C$ computes exactly $\phi_{\overline{\oplus}}^{(min)}$, and all possible values can be applied to its inputs ($\phi_{\overline{\oplus}}^{(min)}$ does not depend on $x_i$).

3. **An error occurs in the EXOR gate.**

   Suppose that a stuck-at-fault occurs at the output of the EXOR gate. In this case, either the network $C$ computes $\phi_{\oplus}^{(min)}$ or it computes $\phi_{\overline{\oplus}}^{(min)}$.

   Suppose the network computes $\phi_{\oplus}^{(min)}$. The error can be tested applying in input a configuration of values in $\{0,1\}^n$ such that $\phi_{\oplus}^{(min)} \neq \phi_{\overline{\oplus}}^{(min)}$, and $x_i = x_j$ (this can be done because $\phi_{\oplus}^{(min)}$ does not depend on $x_i$). In this way, $\xi_{ij} = \phi_{\overline{\oplus}}^{(min)}$, but $C$ computes $\phi_{\oplus}^{(min)} \neq \phi_{\overline{\oplus}}^{(min)}$.

   Observe that we cannot have $\phi_{\oplus}^{(min)} \equiv \phi_{\overline{\oplus}}^{(min)}$, since this would imply that $\xi_{ij} \equiv \phi_{\oplus}^{(min)} \equiv \phi_{\overline{\oplus}}^{(min)}$. In this case, the function $f$ would not depend on the variable $x_i$, in contradiction with the fact that $x_i$ occurs in the minimal SOP $\phi$ of $f$.

   The other case, in which the network $C$ computes $\phi_{\overline{\oplus}}^{(min)}$, can be handled in a similar way.

4. **An error occurs in one of the two AND gates.**

   Suppose that a stuck-at-1 occurs at the output of the AND gate between $\phi_{\oplus}^{(min)}$ and $(x_i \oplus x_j)$. Observe that $\phi_{\oplus}^{(min)} \not\equiv 1$, otherwise $\xi_{ij} = (x_i \oplus x_j) + (x_i \oplus \overline{x}_j)\phi_{\overline{\oplus}}^{(min)}$ and $C$ would not contain

that AND gate. Thus, the error can be tested applying in input to $C$ a configuration of values in $\{0,1\}^n$ such that $\phi_{\oplus}^{(min)}$ takes the value 0, and $x_i = \overline{x}_j$. In this way, $\xi_{ij}$ takes the value 0, but $C$ computes 1.

Now suppose that a stuck-at-0 occurs at the output of the same AND gate. Since $\phi_{\oplus}^{(min)} \not\equiv 0$ (otherwise $\xi_{ij} = (x_i \oplus \overline{x}_j)\phi_{\overline{\oplus}}^{(min)}$ and we would not have that AND gate in the network), the error can be tested applying in input to $C$ a configuration of values in $\{0,1\}^n$ such that $\phi_{\oplus}^{(min)}$ takes the value 1, and $x_i = \overline{x}_j$. In this way, $\xi_{ij}$ takes the value 1, but $C$ computes 0.

The presence of an error in the other AND gate can be tested in a similar way.

5. **An error occurs in the OR gate.**

   Suppose that a stuck-at-1 (0) occurs at the final OR gate. In this case, the network $C$ computes the constant 1 (0) function, and the fault can be easily tested by applying in input to $C$ a configuration of values in $\{0,1\}^n$ such that $f$ is equal to 0 (1).

■

By the proof of Theorem 2, even if the projections $\phi_{\oplus}$ and $\phi_{\overline{\oplus}}$ are just prime and irredundant, and not necessarily minimal, we can state the same result:

**Corollary 1** *An $(i,j)$-EP-SOP without remainder form for a Boolean function $f$ is fully testable in the SAFM, if $f$ depends on both $x_i$ and $x_j$, and if the projected SOP forms $\phi_{\oplus}$ and $\phi_{\overline{\oplus}}$ are prime and irredundant, and not necessarily minimal.*

**Proof.** The thesis follows immediately since prime and irredundant SOP expressions are fully testable in the SAFM. ■

## 5.2 Testability of EP-SOPs With Remainder

Let $f : \{0,1\}^n \rightarrow \{0,1\}$ be a Boolean function and let $\phi$ be a minimal SOP for $f$. For any two variables $x_i$ and $x_j$, both appearing in $\phi$, let

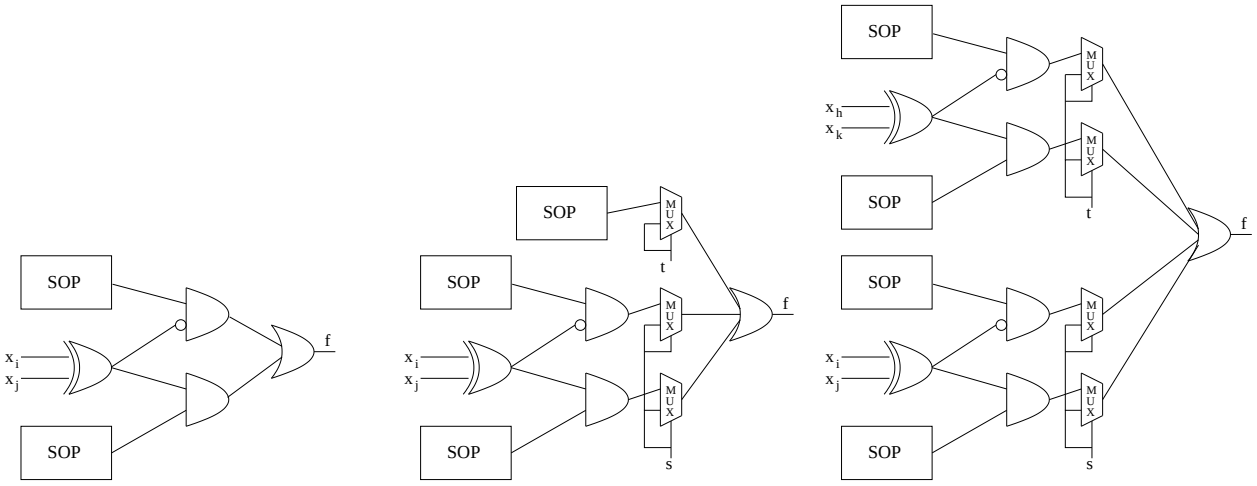
$$\psi_{ij} = (x_i \oplus x_j)\phi_{\oplus}^{(min)} + (x_i \oplus \overline{x}_j)\phi_{\overline{\oplus}}^{(min)} + \rho\,,$$

be the *minimal* $(i,j)$-EP-SOP *with remainder representation* of $f$, where $\phi_{\oplus}^{(min)}$ and $\phi_{\overline{\oplus}}^{(min)}$ are two minimal SOPs representing the projections of the products of $\phi$ containing both $x_i$ and $x_j$ (possibly complemented) onto the spaces $(x_i \oplus x_j)$ and $(x_i \oplus \overline{x}_j)$, respectively, and $\rho$ is the sum of all crossing products of $\phi$.

For an EP-SOP with remainder, the assumption that $f$ does depend on the chosen variables $x_i$ and $x_j$ is not sufficient to guarantee the propagation of all faults to the output of the network, and thus its testability. This is due to the presence of the remainder $\rho$, whose input configurations cannot be set independently from those of $\phi_{\oplus}^{(min)}$, $\phi_{\overline{\oplus}}^{(min)}$, and of the EXOR gate. This problem can be solved by adding some extra gates and inputs: precisely three multiplexers and two additional inputs $s$ and $t$, as shown in Figure 4(b) and described in the following theorem.

**Theorem 3** . *By two additional inputs and three multiplexers, a circuit can be generated from a* minimal $(i,j)$-EP-SOP with remainder *that is fully testable in the SAFM.*

**Proof.** Let $\psi_{ij}$ be a minimal $(i,j)$-EP-SOP with remainder form of a given Boolean function $f$, and let $C$ be the circuit derived from $\psi_{ij}$ adding two new inputs $s$ and $t$ and three multiplexers, as shown Figure 4(b).




(a) Testable EP−SOP without remainder (b) Testable EP−SOP with remainder (c) Testable 2EP−SOP

Figure 4: *Fully testable circuits for EP-SOP forms without and with remainder, and for a 2EP-SOP form.*

To prove the full testability of $C$, we only have to show that any error occurring in it can be propagated to the output, so that it can be tested. This can be accomplished in the following way.

Any stuck-at-fault occurring in the circuits for $\phi_{\oplus}^{(min)}$, $\phi_{\overline{\oplus}}^{(min)}$, in one of the two AND gates, or in the EXOR gate, can be tested by first setting $s = 1$ and $t = 0$, and then applying the appropriate configuration of values in $\{0,1\}^n$ to the other inputs of the network (these configurations can be determined as in the proof of Theorem 2). In fact, by setting $t = 0$, the remainder $\rho$ does not influence the output of the network, allowing us to test the fault.

On the other hand, to test any stuck-at-fault occurring in $\rho$ we must set $s = 0$ and $t = 1$. In this way $C$ simply computes $\rho$, and the testability follows since $\rho$ is prime and irredundant, and all possible values can be applied to its inputs.

Now suppose that a stuck-at-1 (0) occurs at the final OR gate. In this case, $C$ computes the constant 1 (0) function, and the fault can be easily tested by setting $s = 1$, $t = 1$ and by applying in input to $C$ a configuration of values in $\{0,1\}^n$ such that $f$ is equal to 0 (1).

Finally, the presence of a fault in one of the three multiplexers can be tested by applying appropriate values to the additional variables $s$ and $t$, and to the other inputs of $C$. ■

As before, we can state the following corollary.

**Corollary 2** *An $(i,j)$-EP-SOP with remainder form for a Boolean function $f$ is fully testable in the SAFM, if $f$ depends on both $x_i$ and $x_j$, and if $\phi_{\oplus}$, $\phi_{\overline{\oplus}}$ and $\rho$ are prime and irredundant, and not necessarily minimal.*

### 5.3 Testability of 2EP-SOPs and kEP-SOPs

2EP-SOPs can be seen as generalizations of EP-SOPs with remainder. Therefore to guarantee their testability under the SAFM, we must add to the corresponding networks additional inputs and gates.

Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function and let $\phi$ be a minimal SOP for $f$. For any four variables $x_i$, $x_j$, $x_{i'}$, and $x_{j'}$, all appearing in $\phi$, let

$$\zeta_{i,j,i',j'}^{(min)} = (x_i \oplus x_j)\phi_{\oplus}^{(min)} + (x_i \oplus \overline{x}_j)\phi_{\overline{\oplus}}^{(min)} + (x_{i'} \oplus x_{j'})\rho_{\oplus}^{(min)} + (x_{i'} \oplus \overline{x}_{j'})\rho_{\overline{\oplus}}^{(min)},$$

be the *minimal (i,j,i',j')-2EP-SOP representation* of $f$, and let $C$ be the circuit derived from $\zeta^{(min)}_{i,j,i',j'}$ adding two new inputs $s$ and $t$ and four multiplexers, as shown in Figure 4(c).

Using Theorem 2, and generalizing the proof of Theorem 3, one can easily prove the following

**Theorem 4** *By two additional inputs and four multiplexers, a circuit $C$ can be generated from a* minimal $(i, j, i', j')$-2EP-SOP *form that is fully testable in the SAFM.*

As before, the full testability of the network $C$ can be guaranteed even if we replace $\zeta^{(min)}_{i,j,i',j'}$ with the 2EP-SOP computed by the approximation algorithm described in Figure 3. Indeed, this algorithm never chooses variables not occurring in the minimal SOP $\phi$ in input, and the SOPs computed in Step 4 are prime and irredundant. Thus we can state the following corollary.

**Corollary 3** *By two additional inputs and four multiplexers, a circuit $C$ can be generated from the 2EP-SOP form computed by the algorithm in Figure 3 that is fully testable in the SAFM.*

We finally observe that these results can be easily extended to kEP-SOP forms. In this case, the full testability can be obtained adding $k$ extra variables and $2k$ multiplexers to the corresponding networks.

## 6 Experimental results

The polynomial approximation algorithm presented above has been applied to the Espresso benchmark suite [21], running on a Pentium 1.6 GHz processor with 1 GB RAM.

In order to test the practical performance of the approximation algorithm, we have first considered the problem of generating an optimal 1EP-SOP form equivalent to a given optimal SOP form. Since most instances of the Espresso benchmark have multiple outputs, we have split each of them into as many single-output functions as the original outputs. Then, we have removed the trivial instances (zero or one product) and optimized the remaining 2 538 ones by Espresso exact. The number of their input variables ranges from 4 to 128, while the number of products ranges from 2 to 228.

The results show that projecting the given SOP with respect to the most frequent couple of variables pays significantly. In fact, when producing 1EP-SOPs without a remainder, 76% of the times the result is optimal and 88% of the times the gap is below 10%. When producing 1EP-SOPs with a remainder, 64% of the times the result is optimal, 84% of the times the gap is below 10%. The average gap decreases from 7% for the smaller instances (up to 5 products) to 4% for the larger ones (more than 20 products), and it is consistently lower for the instances in the *Math* subset of the Espresso benchmark. Notice that, when different couples of variables have the maximum frequency, the algorithm simply chooses at random one of them for the projection. The results strongly improve if the algorithm evaluates all most frequent couples of variables: in this case, 85% of the 1EP-SOPs without remainder and 71% of the 1EP-SOPs with remainder are optimal. This confirms that the algorithm, besides enjoying an approximation guarantee, has a nearly optimal performance also in practice. The computational times are negligible for both versions described: always less than 1 ms when considering a single couple, a few milliseconds when considering all couples with the maximum frequency.

Then, we have taken into account the standard multiple-output instances of Espresso and compared the starting optimal SOP with the 1EP-SOP and 2EP-SOP produced by the approximation algorithm. As there are multiple outputs, different definitions of "frequency" can be adopted. By *global frequency* of a given couple of variables, we denote the number of products in which it occurs in the whole set of outputs, counting separately the occurrence of the same product in

different outputs. The algorithm adopting this definition determines a single kEP-SOP form with exactly $k$ EXOR gates. By *local frequency* of a given couple of variables, we denote the number of products in which it occurs in each single output. The resulting algorithm performs independent projections for each output, obtaining separate kEP-SOP forms for the outputs that have been projected onto different couples of subspaces. Of course, if different outputs are projected with respect to the same couple of variables, the corresponding EXOR gates are shared, but in the worst case each output could require exactly $k$ EXOR gates. Under both definitions of frequency, the projected SOP forms are synthesized all together with multi-output synthesis. Combining the two definitions of frequency and the two approaches related to the use of the remainder, we obtain four different algorithms, respectively denoted as NG (no remainder and global frequency), NL (no remainder and local frequency), RG (remainder and global frequency), RL (remainder and local frequency).

In order to evaluate the practical performance of the forms generated by these algorithms, we have used a technology mapping (mcnc.genlib) provided by the SIS tool [12] to evaluate the area and the delay corresponding to their physical implementation. Due to the limited space available, we report in Table 1 only a significant subset of the results. The first column reports the name of the instance considered. The following three ones report the area and the delay of the physical implementation of the original SOP form optimized by ESPRESSO EXACT, and the computational time in seconds required to obtain it. The next four columns report the minimum area obtained by one of the four 1EP-SOP forms, the corresponding delay, the algorithm used to generate it and the total computational time in seconds required to generate the four 1EP-SOP forms, that is to factorize the starting SOP and heuristically minimize its projections with all four algorithms. Finally, the last four columns report the same data for the 2EP-SOP forms. Of course, the physical implementation of the EP-SOP forms also include one or more EXOR gates, whose cost cannot be neglected, as our results clearly show. First of all, the EXOR part of the network can be expensive, depending on the technology adopted. Second, some functions benefit from the multi-output minimization: common products can be shared, thus reducing the overall area. Anyway, on about 35% of the instances, the 1EP-SOP form has a lower area than the original SOP form. The gain can be quite striking: it exceeds 50% on instance *adr4* and 40% on many others (e.g., *root*, *z4*). Since the time required to obtain such improvements is quite limited, the evaluation of the 1EP-SOP forms as a possible alternative to the optimal SOP form appears to be an advisable *post-processing strategy*.

Comparing the performances of the four algorithms one to another, we can notice how the number of EXOR gates particularly affects the performance of the algorithms NL and RL, which adopt the local definition of frequency, while the best-performing algorithm seems to be the RG algorithm. When considering 2EP-SOP forms, the results do not improve significantly upon 1EP-SOPs, mainly due to the cost of the additional EXOR gates, which can indeed be relevant, depending on the technology adopted. Most likely, the factorization with respect to the first couple of variables is sufficient, at least for the instances in the ESPRESSO benchmark.

In the end, we have investigated on the relationship between kEP-SOP forms and other multi-level techniques already known in the literature and applied in synthesis tools. In particular, we have applied the multi-level synthesis routines (script.rugged) of SIS to the optimal SOP forms, to the four 1EP-SOP forms and to the four 2EP-SOP forms, in order to determine whether a different, though equivalent, starting form implies a different final result. In a certain number of cases (e.g., on instances *b2*, *exps* and *in1*), SIS was unable to process the optimal SOP form in a limit time of 12 hours. Starting from 1EP-SOP forms, this only happened for instance *in1*, and only for the two 1EP-SOP forms with remainder. Table 2 presents the results. The first column reports the name of the instance, the following one the cost of the multi-level network synthesized by SIS starting

| Function | min SOP | | | min 1EP-SOP | | | | min 2EP-SOP | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Area | Delay | CPU | Area | Delay | Alg. | CPU | Area | Delay | Alg. | CPU |
| **addm4** | 1172 | 47.9 | 0.14 | 906 | 38.3 | RL | 0.23 | 1080 | 43.5 | NL | 0.22 |
| **adr4** | 224 | 19.2 | 0.04 | 105 | 11.1 | RG | 0.13 | 115 | 12.6 | RG | 0.04 |
| **amd** | 1171 | 46.7 | 0.06 | 1022 | 38.0 | RL | 0.17 | 1167 | 43.0 | NL | 0.09 |
| **b4** | 645 | 30.5 | 3.45 | 717 | 34.4 | RG | 0.04 | 762 | 29.8 | RG | 0.04 |
| **br1** | 446 | 32.5 | 0.01 | 353 | 24.5 | NG | 0.08 | 353 | 24.5 | NG | 0.04 |
| **br2** | 352 | 26.6 | 0.01 | 292 | 25.5 | NG | 0.04 | 292 | 25.5 | NG | 0.04 |
| **chkn** | 717 | 43.6 | 0.48 | 758 | 36.1 | RG | 0.12 | 766 | 39.2 | RG | 0.10 |
| **dc2** | 253 | 23.1 | 0.04 | 236 | 19.7 | NL | 0.04 | 267 | 20.4 | RG | 0.04 |
| **exps** | 3932 | 114.5 | 0.50 | 3760 | 112.6 | RG | 0.29 | 3712 | 107.1 | NG | 0.25 |
| **f51m** | 501 | 31.5 | 0.09 | 273 | 19.1 | RL | 0.16 | 336 | 24.1 | RL | 0.06 |
| **in0** | 1214 | 48.3 | 0.10 | 989 | 44.9 | RL | 0.19 | 1209 | 47.0 | NL | 0.15 |
| **in1** | 3876 | 79.8 | 0.23 | 4113 | 81.3 | NG | 0.24 | 4168 | 78.2 | NG | 0.34 |
| **in2** | 1112 | 41.4 | 0.09 | 1000 | 36.7 | NG | 0.10 | 1016 | 41.6 | RG | 0.06 |
| **in5** | 905 | 38.5 | 0.14 | 923 | 40.9 | RG | 0.04 | 931 | 40.0 | RG | 0.04 |
| **intb** | 2170 | 57.3 | 2.96 | 2466 | 57.6 | RG | 2.28 | 2511 | 56.3 | RL | 1.89 |
| **luc** | 806 | 41.0 | 0.01 | 758 | 52.4 | RG | 0.04 | 758 | 52.4 | RG | 0.06 |
| **m1** | 208 | 19.6 | 0.01 | 304 | 21.0 | NG | 0.12 | 313 | 22.8 | NG | 0.04 |
| **m2** | 710 | 37.8 | 0.01 | 833 | 40.9 | NG | 0.04 | 865 | 42.3 | NG | 0.04 |
| **m181** | 166 | 18.4 | 0.60 | 240 | 22.5 | RG | 0.06 | 267 | 19.8 | RL | 0.04 |
| **mlp4** | 734 | 36.4 | 0.31 | 839 | 40.5 | RG | 0.13 | 921 | 44.4 | RG | 0.15 |
| **mp2d** | 362 | 26.0 | 0.25 | 333 | 23.7 | RG | 0.04 | 339 | 24.3 | RG | 0.04 |
| **newcond** | 114 | 17.4 | 0.01 | 119 | 18.2 | RG | 0.04 | 137 | 20.8 | NL | 0.04 |
| **p82** | 239 | 18.4 | 0.01 | 239 | 25.8 | NG | 0.04 | 286 | 28.3 | NG | 0.04 |
| **radd** | 183 | 15.7 | 0.39 | 120 | 15.1 | RG | 0.04 | 120 | 13.2 | RG | 0.04 |
| **rckl** | 341 | 49.7 | 0.04 | 495 | 72.3 | NG | 0.04 | 459 | 30.2 | NG | 0.04 |
| **rd73** | 220 | 25.6 | 0.03 | 264 | 24.1 | RL | 0.12 | 389 | 27.6 | NG | 0.04 |
| **risc** | 228 | 18.7 | 0.01 | 310 | 29.0 | RG | 0.09 | 340 | 33.1 | NG | 0.04 |
| **root** | 592 | 35.5 | 0.35 | 349 | 26.5 | RG | 0.10 | 406 | 29.1 | RG | 0.04 |
| **sqr6** | 278 | 25.5 | 0.06 | 330 | 24.9 | RG | 0.04 | 369 | 27.8 | RG | 0.04 |
| **vg2** | 341 | 18.6 | 0.53 | 468 | 22.5 | RG | 0.17 | 533 | 21.4 | RG | 0.04 |
| **vtx1** | 324 | 21.3 | 0.17 | 365 | 23.4 | RG | 0.04 | 497 | 21.1 | NL | 0.04 |
| **x6dn** | 1054 | 36.8 | 0.18 | 817 | 34.8 | RG | 0.04 | 821 | 33.8 | NG | 0.04 |
| **x9dn** | 384 | 23.0 | 0.20 | 424 | 24.7 | RG | 0.17 | 539 | 23.6 | RL | 0.04 |
| **z4** | 171 | 18.3 | 0.01 | 99 | 14.2 | RG | 0.04 | 102 | 13.7 | RG | 0.04 |

Table 1: Area, delay and synthesis time of SOP, 1EP-SOP and 2EP-SOP forms, computed in SIS after the technology mapping.

from the original SOP form. Then, a pair of columns reports the cost of the best network obtained starting from the four 1EP-SOPs, and its variation with respect to the cost of the network derived from the original SOP. In the end, three columns report the cost of the best network obtained starting from the four 2EP-SOPs, and its variation with respect to the network derived from the original SOP and the network derived from the best 1EP-SOP.

Only few times the final results were identical and one third of the times the final result obtained starting from a 1EP-SOP was better than the one obtained from the optimal SOP form, ranging from 30% better to 30% worse. The results for the 2EP-SOPs are worse on average, but they exhibit some improvements, even with respect to both SOPs and 1EP-SOPs. We can therefore notice that, especially for difficult benchmarks, multi-level synthesis often benefits from starting with a 1EP-SOP or 2EP-SOP form instead of a standard SOP. A reason for this behaviour can be the fact that SIS computes the resulting network by heuristics, which can overlook part of the underlying regularity in the network. Therefore, the form of the input network can significantly change the final output, and our factorization can guide the search towards a more compact multi-level form.

# 7 Conclusion

We have proposed a new four-level algebraic form, called kEP-SOP, based on EXOR projections of minimal SOP forms, generalizing the approach of [1], and we have studied its testability properties.

Due to the high complexity of the exact minimization problem, we have presented a polynomial time approximation algorithm for kEP-SOP synthesis, whose approximation ratio improves the one derived in [1] for EP-SOP forms.

The proposed algorithm has been implemented and tested with interesting results, proving that kEP-SOP forms are a practical alternative to standard SOP synthesis.

| Function | SOP | 1EP-SOP | | 2EP-SOP | | |
|---|---|---|---|---|---|---|
| | Cost | Cost | Gap w.r.t. SOP | Cost | Gap w.r.t. SOP | Gap w.r.t. 1EP-SOP |
| **addm4** | 392 | 372 | -5.10% | 390 | -0.51% | 4.84% |
| **adr4** | 60 | 44 | -26.67% | 44 | -26.67% | 0.00% |
| **alcom** | 136 | 135 | -0.74% | 135 | -0.74% | 0.00% |
| **alu1** | 41 | 41 | 0.00% | 41 | 0.00% | 0.00% |
| **amd** | 265 | 248 | -6.42% | 299 | 12.83% | 20.56% |
| **b4** | 310 | 315 | 1.61% | 317 | 2.26% | 0.63% |
| **b7** | 97 | 101 | 4.12% | 106 | 9.28% | 4.95% |
| **b11** | 97 | 101 | 4.12% | 106 | 9.28% | 4.95% |
| **b12** | 96 | 115 | 19.79% | 124 | 29.17% | 7.83% |
| **br1** | 109 | 112 | 2.75% | 112 | 2.75% | 0.00% |
| **br2** | 81 | 78 | -3.70% | 78 | -3.70% | 0.00% |
| **chkn** | 462 | 463 | 0.22% | 550 | 19.05% | 18.79% |
| **co14** | 78 | 78 | 0.00% | 78 | 0.00% | 0.00% |
| **dc1** | 44 | 44 | 0.00% | 52 | 18.18% | 18.18% |
| **dc2** | 142 | 140 | -1.41% | 134 | -5.63% | -4.29% |
| **f51m** | 109 | 130 | 19.27% | 159 | 45.87% | 22.31% |
| **gary** | 493 | 493 | 0.00% | 587 | 19.07% | 19.07% |
| **in0** | 505 | 490 | -2.97% | 587 | 16.24% | 19.80% |
| **in2** | 327 | 389 | 18.96% | 406 | 24.16% | 4.37% |
| **in5** | 363 | 357 | -1.65% | 352 | -3.03% | -1.40% |
| **in7** | 136 | 136 | 0.00% | 144 | 5.88% | 5.88% |
| **intb** | 306 | 401 | 31.05% | 590 | 92.81% | 47.13% |
| **log8mod** | 137 | 138 | 0.73% | 143 | 4.38% | 3.62% |
| **luc** | 169 | 174 | 2.96% | 170 | 0.59% | -2.30% |
| **m1** | 82 | 84 | 2.44% | 84 | 2.44% | 0.00% |
| **m2** | 227 | 226 | -0.44% | 226 | -0.44% | 0.00% |
| **m3** | 284 | 258 | -9.15% | 294 | 3.52% | 13.95% |
| **m181** | 104 | 119 | 14.42% | 129 | 24.04% | 8.40% |
| **max128** | 317 | 323 | 1.89% | 342 | 7.89% | 5.88% |
| **mlp4** | 322 | 321 | -0.31% | 335 | 4.04% | 4.36% |
| **newapla1** | 36 | 36 | 0.00% | 36 | 0.00% | 0.00% |
| **newapla2** | 28 | 28 | 0.00% | 28 | 0.00% | 0.00% |
| **newapla** | 44 | 55 | 25.00% | 51 | 15.91% | -7.27% |
| **newcond** | 104 | 106 | 1.92% | 110 | 5.77% | 3.77% |
| **newcpla1** | 105 | 116 | 10.48% | 126 | 20.00% | 8.62% |
| **newcpla2** | 68 | 68 | 0.00% | 76 | 11.76% | 11.76% |
| **newcwp** | 22 | 25 | 13.64% | 28 | 27.27% | 12.00% |
| **newill** | 37 | 31 | -16.22% | 31 | -16.22% | 0.00% |
| **newtag** | 18 | 18 | 0.00% | 18 | 0.00% | 0.00% |
| **newtpla1** | 33 | 34 | 3.03% | 34 | 3.03% | 0.00% |
| **newtpla2** | 37 | 38 | 2.70% | 41 | 10.81% | 7.89% |
| **newtpla** | 78 | 78 | 0.00% | 78 | 0.00% | 0.00% |
| **newxcpla1** | 110 | 115 | 4.55% | 129 | 17.27% | 12.17% |
| **p82** | 103 | 101 | -1.94% | 109 | 5.83% | 7.92% |
| **radd** | 45 | 45 | 0.00% | 44 | -2.22% | -2.22% |
| **rckl** | 258 | 286 | 10.85% | 286 | 10.85% | 0.00% |
| **rd53** | 37 | 34 | -8.11% | 43 | 16.22% | 26.47% |
| **rd73** | 78 | 70 | -10.26% | 84 | 7.69% | 20.00% |
| **risc** | 100 | 101 | 1.00% | 108 | 8.00% | 6.93% |
| **root** | 151 | 139 | -7.95% | 156 | 3.31% | 12.23% |
| **sex** | 61 | 65 | 6.56% | 64 | 4.92% | -1.54% |
| **sqr6** | 130 | 146 | 12.31% | 120 | -7.69% | -17.81% |
| **t3** | 91 | 113 | 24.18% | 110 | 20.88% | -2.65% |
| **tms** | 193 | 201 | 4.15% | 206 | 6.74% | 2.49% |
| **vg2** | 106 | 106 | 0.00% | 106 | 0.00% | 0.00% |
| **x6dn** | 391 | 395 | 1.02% | 406 | 3.84% | 2.78% |
| **x9dn** | 146 | 123 | -15.75% | 119 | -18.49% | -3.25% |
| **z4** | 46 | 36 | -21.74% | 36 | -21.74% | 0.00% |

Table 2: Network costs of multilevel SIS networks computed starting from classical SOP forms, 1EP-SOP and 2EP-SOP forms.

**Acknowledgments** The authors would like to thank Fabrizio Luccio, whose precious suggestions have contributed to the development of this work.

## References

[1] A. Bernasconi, V. Ciriani, and R. Cordone. EXOR Projected Sum of Products. In *14th International Conference on Very Large Scale Integration*, 2006. (An extended version is available as Technical Report TR–06-10, Computer Science Department, University of Pisa).

[2] R. Brayton, G. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli. *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer Academic Publishers, 1984.

[3] M. Breuer and A. Friedman. *Diagnosis & Reliable Design of Digital Systems*. Computer Science Press, 1976.

[4] G. Caruso. Near Optimal Factorization of Boolean Functions. *IEEE Transactions on CAD*, 10(8):1072–1078, 1991.

[5] V. Ciriani. *Three-Level Logic Synthesis: Algebraic Approach and Minimization Algorithms*. PhD thesis, Dipartimento di Informatica, University of Pisa, 2003.

[6] R. Cordone, F. Ferrandi, D. Sciuto, and R. Wolfler Calvo. An efficient heuristic approach to solve the unate covering problem. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(12):1377–1388, Dec. 2001.

[7] O. Coudert. Two-Level Logic Minimization: an Overview. *INTEGRATION*, 17:97–140, 1994.

[8] D. Debnath and T. Sasao. Multiple–Valued Minimization to Optimize PLAs with Output EXOR Gates. In *IEEE International Symposium on Multiple-Valued Logic*, pages 99–104, 1999.

[9] D. Debnath and Z. Vranesic. A Fast Algorithm for OR-AND-OR Synthesis. *IEEE Transactions on CAD*, 22(9):1166–1176, 2003.

[10] R. Drechsler, J. Shi, and G. Fey. Synthesis of Fully Testable Circuits from BDDs. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 23(3):440–443, 2004.

[11] E. Dubrova, D. Miller, and J. Muzio. AOXMIN-MV: A Heuristic Algorithm for AND-OR-XOR Minimization. In *4th Int. Workshop on the Applications of the Reed Muller Expansion in circuit Design*, pages 37–54, 1999.

[12] E.M. Sentovich et al. SIS: A system for sequential circuit synthesis. Technical report, 1992.

[13] M. Fujita, Y. Matsunaga, and M. Ciesielski. Multi-Level Logic Optimization. In S. Hassoun and T. Sasao, editors, *Logic Synthesis and Verification*, pages 29–63. Kluwer Academic Publishers, 2002.

[14] M. Garey and D. Johnson. *Computer and Intractability: A Guide to the Theory of NP-completeness*. W.H. Freeman and Company, 1979.

[15] R. Ishikawa, T. Hirayama, G. Koda, and K. Shimizu. New Three-Level Boolean Expression Based on EXOR Gates. *IEICE Transactions on Information and Systems*, (5):1214–1222, 2004.

[16] R. Ishikawa, T. Igarashi, T. Hirayama, and K. Shimizu. Pseudocube-based expressions to enhance testability. In *IEEE Asia-Pacific Conference on Circuits and Systems*, volume 2, pages 305–310, 2002.

[17] F. Luccio and L. Pagli. On a New Boolean Function with Applications. *IEEE Transactions on Computers*, 48(3):296–310, 1999.

[18] R. Rudell and A. Sangiovanni-Vincentelli. Multiple-valued Minimization for PLA Optimization. *IEEE Trans. on CAD of Integrated Circuits and Systems*, CAD-6:727–750, Sept. 1987.

[19] T. Sasao. *Switching Theory for Logic Synthesis*. Kluwer Academic Publishers, 1999.

[20] E. Sentovich, K. Singh, L. Lavagno, C. Moon, A. S. R. Murgai, H. Savoj, P. Stephan, R. Brayton, and A. Sangiovanni-Vincentelli. SIS: A system for sequential circuit synthesis. Technical report, 1992.

[21] S. Yang. Logic synthesis and optimization benchmarks user guide version 3.0. User guide, Microelectronic Center, 1991.