Boolean Minimization of Projected Sums of Products via Boolean Relations

Anna Bernasconi, Valentina Ciriani, Gabriella Trucco, and Tiziano Villa

Abstract—Projected Sums of Products (PSOPs) are a Generalized Shannon Decomposition (GSD) with remainder that restructures a logic function into three logic blocks corresponding to a logic bi-decomposition plus a reminder generated by a cofactoring function. In this paper we discuss a Boolean synthesis technique for PSOPs, which exploits the fact that the resulting logical structure induces don't care conditions that can be exploited to reduce the problem of area minimization to Boolean relation minimization, with the guarantee that all valid realizations of the circuit are considered. This technique is more general than the algebraic methods investigated so far. Moreover, we characterize the points that are in the remainder with a simple procedure that implies a fast construction of the Boolean relation for important classes of cofactoring functions like the chain of XORs or ANDs. We report experiments confirming the effectiveness in area of the proposed approach based on Boolean relations, with better run times for some cost functions.

Index Terms—Logic synthesis, Boolean decomposition, Boolean relations.

1 Introduction

Two-level Sum of Products (SOP) minimization is the classical approach to logic synthesis [27], [35]. In general, this approach guarantees a very low delay, due to the fixed number of levels, and a reasonable synthesis time, at the expense of a possibly high number of gates in the resulting circuit. In order to obtain networks with a smaller area, multi-level network synthesis has been proposed and widely studied, both in unbounded [3], [4], [23], [30], [37], [39] and bounded [7], [28], [29], [31] models. While circuits with an unbounded number of levels can be very compact, the unrestricted approach can lead to longer delays. A good trade-off between area and delay minimization is represented by the bounded multi-level minimization, where the number of levels (typically, three or four) is fixed before the synthesis step. Sasao statistically showed that three levels of logic are enough to produce a minimal network for most of the Boolean functions; and in many cases three-level logic is a good compromise between circuit delay, circuit area, and minimization time [38].

In this paper, we focus on a bounded-multilevel model based on decomposition. Decomposition is a frequently exploited technique for reducing circuit area while keeping the number of gate levels bounded (see [2], [8], [9], [10], [11], [12], [14], [15], [16], [17], [19], [23]).

The most widely used form of decomposition of a Boolean function $f : \{0,1\}^n \to \{0,1\}$ is bi-decomposition, defined as $f(X,Y,Z) = g(X,Z)$ *op* $h(Y,Z)$, where *op* stands for any binary Boolean operation (see [24], [26], [33], [34], [36], [41]). A classical well-known bi-decomposition is given by the Shannon decomposition of a Boolean function with respect to a Boolean variable x_i: $f = \overline{x}_i f|_{x_i=0} + x_i f|_{x_i \neq 0}$ (see [21]). Furthermore, generalizations of the classical Shannon decomposition (GSD) represent a Boolean function f as $GSD(f) = (\overline{x}_i \oplus p) f|_{x_i=p} + (x_i \oplus p) f|_{x_i \neq p}$, where x_i is a selected input variable and p is a function defined over all variables except x_i (e.g., when p is the constant 0 function we obtain the standard Shannon decomposition), see [20] and [32]. The cofactors $f|_{x_i=p}$ and $f|_{x_i \neq p}$ correspond to the projections of f onto the two subsets with $x_i = p$ and $x_i \neq p$, whose characteristic functions are $(\overline{x}_i \oplus p)$ and $(x_i \oplus p)$, respectively.

We can observe that if we algebraically derive a GSD expression from a SOP for f, each product is either entirely included into one of the subspaces, or it intersects them both. The products that intersect both subspaces are called *crossing products* and are split into the two subspaces. However, splitting a crossing product implies its insertion in both cofactors of the algebraic form and could lead to a waste of circuit area [6].

The *Projected Sums of Products with remainder (PSOP)* form has been introduced in [13], and further discussed in [11] in order to avoid the split of crossing products. This algebraic decomposition introduces a *remainder* R given by the sum of all crossing products, that are then not split and not projected onto the two subspaces. For this reason, a PSOP with remainder is, in general, smaller than a standard GSD.

In this paper we propose a new Boolean definition for PSOPs that generalizes the algebraic one, where the remainder R is not forced to contain all the possible crossing products, but only a subset. Moreover, the PSOP definition and the corresponding minimization technique proposed in this paper are Boolean, i.e., they exploit all the properties of Boolean algebra to simplify the Boolean function f, whereas the synthesis methods proposed in [11], [13] are algebraic, i.e., they rewrite the expressions with the rules of

- *A. Bernasconi is with the Department of Computer Science, Università di Pisa, Italy.*
  E-mail: anna.bernasconi@unipi.it
- *V. Ciriani and G. Trucco are with the Department of Computer Science, Università degli Studi di Milano, Italy.*
  E-mail: valentina.ciriani@unimi.it and gabriella.trucco@unimi.it
- *T.Villa is with the Department of Computer Science, Università di Verona, Italy.*
  E-mail: tiziano.villa@univr.it

Manuscript received ...; revised ...

polynomial algebra on the algebraic expressions of f (e.g., SOP forms). Furthermore, we present a new formulation of the synthesis problem via Boolean relations, proving that the minimal PSOP form is the solution of the corresponding Boolean relation.

In general, while Boolean synthesis techniques yield smaller implementations, algebraic synthesis methods are often less time consuming. In the experimental results section we show that we can find a good trade-off between area reduction and computational time, by using a heuristic method for solving the Boolean relation. We report an average gain in area of the 28%, and an average gain in delay of the 17% with respect to the algebraic method; morever, we were even able to improve on synthesis time, with an average gain of the 30%, by using a Boolean relation minimizer (BREL [5]) in the heuristic mode.

For defining the synthesis problem through a Boolean relation, it is fundamental to characterize and efficiently compute the remainder set R. While in the algebraic context the definition of R is straightforward and depends on the products only, in the Boolean setting R must be defined and built using the on- and dc-set of the function. Since on- and dc-set of a function can have an exponential size, an efficient algorithm for the construction of R is crucial. Moreover, in the Boolean context, the remainder construction heavily depends on the function p used to define the projection subspaces. Therefore, the second main result described in this paper is the characterization of the reminder R when p is a linear function (i.e., an XOR of a subset of variables) or a conjunctive function (an AND of some input variables).

In particular, this paper is an extended version of the conference paper presented in [18] that considers only the restricted case with $p = x_j$ (EP-SOP forms), where the Boolean definition for the remainder set simply exploits the Hamming distances between minterms. Unfortunately, that approach cannot be directly generalized to other functions p. In this work, we deal with this issue, and propose efficient algorithms for the construction of R when p is a linear function or a conjunction of literals.

Finally, in this extended version, the synthesis problem via Boolean relations is formulated for a generic function p.

The paper is organized as follows : Section 2 is a brief review on Boolean relations , Section 3 introduces the new definition of PSOP forms, for a completely specified Boolean function and for an incompletely specified Boolean function , Section 4 describes the PSOP minimization strategy via Boolean relations , in Section 5 the computation of the remainder for p of type XORs or ANDs is discussed , and Section 6 reports the experimental results. Finally, Section 7 concludes the work.

2 Boolean Relations: a brief review

The concept of Boolean relation was introduced in [22] as a more general specification of non-determinism in logic networks, which cannot always be represented using don't cares.

Definition 1 ([22]). A *Boolean relation* is a one-to-many multi-output Boolean mapping $\mathcal{R} : \{0,1\}^n \to \{0,1\}^m$. $\{0,1\}^n$ and $\{0,1\}^m$ are called the *input* and *output sets* of $\mathcal{R}$.

A Boolean relation $\mathcal{R}$ can be considered a generalization of a Boolean function, where a point in the input set $\{0,1\}^n$ can be associated with several points in the output set $\{0,1\}^m$; indeed, because of the one-to-many nature of Boolean relations, there may be several equivalent outputs for a given input. A relation $\mathcal{R}$ is *well-defined* if for all $x \in \{0,1\}^n$, there is $y \in \{0,1\}^m$ such that $(x, y) \in \mathcal{R}$. To any relation $\mathcal{R}$ we can associate a set $\mathcal{F}(\mathcal{R})$ of *compatible* multi-output Boolean functions, i.e. the set of all functions f such that, for all inputs $x \in \{0,1\}^n$, $f(x)$ is contained in the set $\mathcal{R}(x)$ of the outputs related to x. In this case, we write $f \subseteq \mathcal{R}$. The problem of the optimal implementation of a Boolean relation $\mathcal{R}$ is to select, among the possible functions compatible with $\mathcal{R}$, one of minimum cost according to a given metric. More precisely, the *solution* of a Boolean relation $\mathcal{R}$ is a multi-output Boolean function $f \in \mathcal{F}(\mathcal{R})$. The function f is an *optimal solution* of $\mathcal{R}$ according to a given cost function c, if for all $f' \in \mathcal{F}(\mathcal{R})$, $c(f) \leq c(f')$. Several exact and heuristic algorithms have been proposed for solving Boolean relations (see [5] for an overview of these methods, and for bibliographic references). For our minimization problem, we use the algorithm proposed in [5], in both exact and heuristic mode, because of its efficiency and ability to explore a wide space of solutions.

3 PSOP expressions

In this section, we first recall the definition of PSOP forms, introduced in [6], [13] and further discussed in [11]. We then show how to change and rephrase the definition of PSOP form, both for completely and incompletely specified functions, in order to obtain a new general form, more flexible and better suited to be described and minimized via Boolean relations.

3.1 Completely specified Boolean functions

Generalizing the classical Shannon decomposition (see [20] and [32]), any completely specified Boolean function $f : \{0,1\}^n \to \{0,1\}$ can be represented as follows

$$GSD(f) = (\overline{x}_i \oplus p) f|_{x_i=p} + (x_i \oplus p) f|_{x_i \neq p} \,, \qquad (1)$$

where x_i is a selected input variable and p is a function possibly depending on all input variables except x_i. This decomposition partitions the Boolean space $\{0,1\}^n$ into two subsets each containing 2^{n-1} points: the subset of points $(x_1, \ldots, x_i, \ldots, x_n) \in \{0,1\}^n$ where the function p and the variable x_i have the same value 0 or 1, i.e., $x_i = p$, and the subset of points where the value of p and the value of x_i are different, i.e., $x_i \neq p$.

The characteristic functions of these two subsets are $(\overline{x}_i \oplus p)$ and $(x_i \oplus p)$, respectively. The cofactors $f|_{x_i=p}$ and $f|_{x_i \neq p}$ correspond to the projections of f onto the two subsets. Note that we can obtain the classical Shannon decomposition when $p = 0$, i.e., when p is the constant 0 function.

The two cofactors ($f|_{x_i=p}$ and $f|_{x_i \neq p}$) can be equivalently defined as incompletely specified Boolean functions, in the Boolean space $B^n = \{0,1\}^n$, in the following way:

1) $f^{on}|_{x_i=p}$ ($f^{on}|_{x_i \neq p}$) is the on-set of the original function f such that $x_i = p$ (resp. $x_i \neq p$);

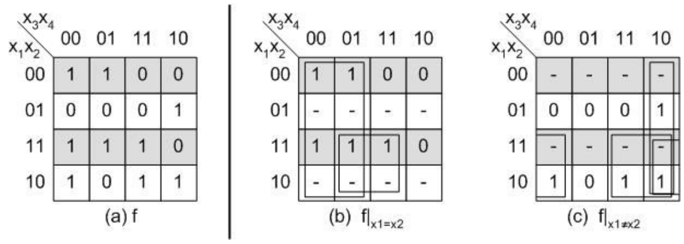

Fig. 1. The projected functions $f|_{x_1=x_2}$, $f|_{x_1\neq x_2}$ of the Boolean function f, together with the corresponding covers $f|_{x_1=x_2} = \overline{x}_3 + x_1x_4$, $f|_{x_1\neq x_2} = x_3\overline{x}_4 + x_1\overline{x}_4 + x_1x_3$.

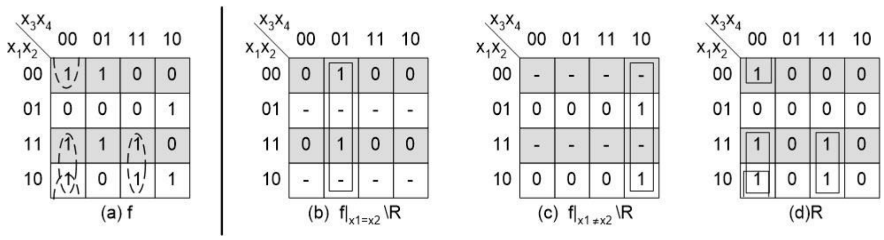

Fig. 2. On the left, a function f with the crossing products. On the right the functions $f|_{x_1=x_2} \setminus R$, $f|_{x_1\neq x_2} \setminus R$, and the remainder R, together with the corresponding covers $f|_{x_1\neq x_2} \setminus R = \overline{x}_3x_4$, $f|_{x_1\neq x_2} \setminus R = x_3\overline{x}_4$, and $f^R = x_1\overline{x}_3\overline{x}_4 + x_1x_3x_4 + \overline{x}_2\overline{x}_3\overline{x}_4$.

2) $f^{dc}|_{x_i=p}$ ($f^{dc}|_{x_i\neq p}$) is the set of points such that $x_i \neq p$ (resp. $x_i = p$), i.e., $f^{dc}|_{x_i=p} = B^n|_{x_i\neq p}$ (resp. $f^{dc}|_{x_i\neq p} = B^n|_{x_i=p}$).

In the present paper, we adopt this alternative definition. For example, consider the function f in Figure 1(a), with $i = 1$ and p defined as the variable x_2, i.e., $p = x_2$. In the figure, the subset of the Boolean space where $x_1 = x_2$ ($x_1 \neq x_2$, resp.) is depicted in gray (white, resp.). The corresponding (non-projected) functions $f|_{x_1=x_2}$ and $f|_{x_1\neq x_2}$ are represented in Figures 1(b) and 1(c), respectively. Note that $f|_{x_1=x_2}$ corresponds to f for the points where $x_1 = x_2$ and contains don't care conditions where $x_1 \neq x_2$. These don't cares can be inserted in $f|_{x_1=x_2}$ since, in Equation 1, this function is multiplied by $(\overline{x}_1 \oplus x_2)$, which evaluates to 0 when $x_1 \neq x_2$. A symmetric observation can be performed for $f|_{x_1\neq x_2}$.

A clear advantage of this representation is that EXOR-based decompositions insert don't care points that help in forming larger cubes. Consider, for instance, the function f in Figure 1(a) and the two distinct cubes $\overline{x}_1\overline{x}_2\overline{x}_3$ and $x_1x_2\overline{x}_3$. In the function $f|_{x_1=x_2}$, the corresponding cubes are merged together in the larger cube $\overline{x}_3$ using don't cares, as shown in Figure 1(b). Therefore, while a minimal SOP form for the function f is $\overline{x}_1\overline{x}_2\overline{x}_3+\overline{x}_1x_2x_3\overline{x}_4+x_1x_2\overline{x}_3+x_1x_3x_4+x_1\overline{x}_2\overline{x}_4$, a minimal GSD form (i.e., Equation (1)) is $GSD(f) = (\overline{x}_1 \oplus x_2)(\overline{x}_3 + x_1x_4) + (x_1 \oplus x_2)(x_3\overline{x}_4 + x_1\overline{x}_4 + x_1x_3)$.

However, there may be a drawback: the cubes of f intersecting both subsets $x_i = p$ and $x_i \neq p$ are split into two subcubes when they are decomposed onto the two subsets. For example, consider again the function f in Figure 1(a), the cube $x_1x_3x_4$ is split into two minterms: $x_1x_2x_3x_4$ in Figure 1(b) and $x_1\overline{x}_2x_3x_4$ in Figure 1(c) that are covered by two different cubes (x_1x_4 and x_1x_3, resp.). Following the terminology introduced in [6], we call these cubes *crossing cubes*, since they cross the two subsets of the Boolean space $x_i = p$ and $x_i \neq p$.

In order to avoid the split of crossing cubes, a slightly different decomposition called *Projected Sums of Products (PSOP)* form, has been introduced in [6] (for a function p consisting in a single variable) and in [13] (for a general function p), and further discussed in [11]. This decomposition allows the use of a non decomposed set R called *remainder*, which contains all crossing cubes occurring in an original SOP representation of the target function f. Indeed, in these previous papers, PSOP forms are algebraically defined starting from a SOP for f. The products of the SOP are first classified into three subsets: 1) those that are entirely included into the subspace $x_i = p$, 2) those that are entirely included into the subspace $x_i \neq p$, and 3) those that intersect both spaces, which are called *crossing products* and form the remainder R. The overall *PSOP with remainder* form is the sum of these three sets of cubes where the first two are factorized with $(\overline{x}_i \oplus p)$ and $(x_i \oplus p)$, respectively. Since R contains the crossing products, PSOPs are in general smaller than classical GSD.

In this paper, instead of considering an algebraic SOP form for f, we start from the Boolean representation of the function. Therefore, we have not a fixed SOP covering for f, but its Boolean description only. For this reason we have to slightly redefine the notion of point belonging to a crossing cube and the definition of remainder.

Definition 2. A point $v \in \{0, 1\}^n$ is a *crossing point* if

1) v belongs to the on-set of the function f;
2) there exists another point u in the on-set of f such that v and u are neighbors, i.e., their Hamming distance is 1;
3) v and u do not belong to the same projection subset.

In other words, a minterm v is a crossing point if and only if we can find a point, in the other projection subset, which can form a cube c with v. The cube c is called *crossing cube*. Considering the same example of Figure 1, the

Karnaugh map in Figure 2(a) shows the points belonging to crossing cubes and the corresponding cubes.

Definition 3. The *remainder* R is the set of all on-set minterms that are crossing points.

In the running example, the reminder R, with a minimal cover, is depicted in Figure 2(d).

An operational definition and the related computational procedure of R depend on the structure of the cofactoring function p, as it will be discussed in more detail in Section 5.

From the previous discussion, the following Boolean definition for a PSOP with remainder would follow:

$$(\overline{x}_i \oplus p)(f|_{x_i=p} \setminus R) + (x_i \oplus p)(f|_{x_i \neq p} \setminus R) + R\,. \qquad (2)$$

The corresponding example is depicted on the right side of Figure 2.

However, the previous Boolean definition in Equation (2) does not account for the full power of Boolean minimization. In fact, any point of f is covered in one and only one subset: either in the remainder R or in one of the two cofactors $f|_{x_i=x_j} \setminus R$ or $f|_{x_i \neq x_j} \setminus R$, whereas to get a minimal decomposition form it is better to allow the flexibility to cover any point of the function by means of *at least* one subset, or, more precisely, to cover any point x of the remainder R: 1) only in the remainder, 2) only in the corresponding cofactor (i.e., $f|_{x_i=p}$ if $x_i = p$, or $f|_{x_i \neq p}$ if $x_i \neq p$), 3) both in the remainder and in the corresponding cofactor. In fact, x may help to form larger cubes in the remainder, in a cofactor, or in both in the remainder and in the cofactors. For instance, in the running example, the point 1100 can be used in $f|_{x_1=x_2}$ to form the cube $\overline{x}_3$ and in the remainder R to form the cube $x_1\overline{x}_3\overline{x}_4$ with the point 1000.

Because of these observations, we sharpen the Boolean definition of PSOP expressions as follows.

Definition 4. A *PSOP decomposition* of a completely specified function f is the expression:

$$\text{PSOP}(f) = (\overline{x}_i \oplus p)\, f^{=} + (x_i \oplus p)\, f^{\neq} + f^R$$

where the sets of points $f^{=}$, $f^{\neq}$, f^R, and f satisfy the following conditions:

1) $(f^{on}|_{x_i=p} \setminus R) \ \subseteq\ f^{=} \ \subseteq\ f^{on}|_{x_i=p} \cup f^{dc}|_{x_i=p}$
2) $(f^{on}|_{x_i \neq p} \setminus R) \ \subseteq\ f^{\neq} \ \subseteq\ f^{on}|_{x_i \neq p} \cup f^{dc}|_{x_i \neq p}$
3) $\emptyset \ \subseteq\ f^R \ \subseteq\ R$
4) $\text{PSOP}(f) = f$.

This definition includes the flexibility to avoid the splitting of the crossing cubes (covering them in the remainder) and to reuse points, already covered in the remainder, to form larger cubes for the cofactors.

The idea of PSOP synthesis is to construct a network for f by choosing appropriately the sets $f^{=}$, $f^{\neq}$, and f^R as building blocks. If we focus on the standard SOP synthesis, we get a *PSOP circuit*

$$\text{PSOP}(f) = (\overline{x}_i \oplus p)\, S(f^{=}) + (x_i \oplus p)\, S(f^{\neq}) + S(f^R)\,,$$

where $S(f^{=}), S(f^{\neq})$, and $S(f^R)$ denote the SOP implementations of $f^{=}$, $f^{\neq}$, and f^R, respectively.

Given the variable x_i and the cofactoring function p, an *optimal PSOP circuit*, $\text{PSOP}^*_{(i,p)}(f)$, is a PSOP circuit with the minimum cost that can be synthesized for f, decomposing the function f with respect to the variable x_i and the function p; while an *optimal PSOP circuit*, $\text{PSOP}^*(f)$, for f is a PSOP circuit with the minimum cost among all possible $\text{PSOP}^*_{(i,p)}(f)$ circuits for f. For example, an optimal $(1, x_2)$ PSOP form, minimal with respect to the number of literals, for the function f in Figure 3(a) is

$$\text{PSOP}^*_{(1,x_2)}(f) = (\overline{x}_1 \oplus x_2)(\overline{x}_3) + (x_1 \oplus x_2)(x_3\overline{x}_4) + (x_1\overline{x}_3\overline{x}_4 + x_1x_3x_4).$$

We note that we are using fewer and larger cubes than the ones used in a standard minimal SOP cover $f = \overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1x_2x_3\overline{x}_4 + x_1x_2\overline{x}_3 + x_1x_3x_4 + x_1\overline{x}_2\overline{x}_4$. Also note that any point of the function is covered at least once, by $f^{=}$, $f^{\neq}$, or f^R. For example, 0000 is covered by $f^{=}$, 1100 is covered by both $f^{=}$ and f^R, and 1000 is covered by f^R (note that 1000 is also covered by $f^{=}$ but not by $(\overline{x}_1 \oplus x_2)f^{=}$ in the final form).

3.2 Incompletely specified Boolean functions

Let $f = \{f^{on}, f^{dc}\}$ be an incompletely specified Boolean function. For the sake of simplicity, suppose that $f^{on} \cap f^{dc} = \emptyset$; otherwise, following the usual semantics, we consider $f^{on} \setminus f^{dc}$ as the on-set of f.

When f is an incompletely specified Boolean function, the definition of the projected don't care set $f^{dc}|_{x_i=p}$ ($f^{dc}|_{x_i \neq p}$) changes as follows: $f^{dc}|_{x_i=p}$ ($f^{dc}|_{x_i \neq p}$) contains the points of f^{dc} such that $x_i = p$ (resp. $x_i \neq p$) together with all points of $\{0,1\}^n$ where $x_i \neq p$ (resp. $x_i = p$), i.e., $f^{dc}|_{x_i=p} = B^n|_{x_i \neq p} \cup f^{dc}|_{x_i=p}$ (resp. $f^{dc}|_{x_i \neq p} = B^n|_{x_i=p} \cup f^{dc}|_{x_i \neq p}$).

For incompletely specified Boolean functions, the definitions of crossing points and R are extended as follows.

Definition 5. A point $v \in \{0,1\}^n$ is a *crossing point* if

1) v belongs to the on-set or a dc-set of the function f;
2) there exists another point u in the on-set or dc-set of f whose Hamming distance from v is 1;
3) v and u do not belong to the same subset $x_i = p$ or $x_i \neq p$.

Definition 6. The *remainder* R ($R \subseteq f^{on} \cup f^{dc}$) is the subset of on-set and dc-set minterms that are crossing points.

The notions of PSOP decomposition and PSOP circuit can be immediately generalized to incompletely specified Boolean functions, noting that the remainder set R now includes all points of f^{on} and all points of f^{dc} that could form a crossing cube.

Definition 7. A *PSOP decomposition* of an incompletely specified function $f = \{f^{on}, f^{dc}\}$ is the expression:

$$\text{PSOP}(f) = (\overline{x}_i \oplus p)\, f^{=} + (x_i \oplus p)\, f^{\neq} + f^R$$

where the sets of points $f^{=}$, $f^{\neq}$, f^R, and f satisfy the following conditions:

1) $(f^{on}|_{x_i=p} \setminus R) \ \subseteq\ f^{=} \ \subseteq\ f^{on}|_{x_i=p} \cup f^{dc}|_{x_i=p}$
2) $(f^{on}|_{x_i \neq p} \setminus R) \ \subseteq\ f^{\neq} \ \subseteq\ f^{on}|_{x_i \neq p} \cup f^{dc}|_{x_i \neq p}$
3) $\emptyset \ \subseteq\ f^R \ \subseteq\ R$
4) $f^{on} \ \subseteq\ \text{PSOP}(f) \ \subseteq\ f^{on} \cup f^{dc}$.

All the observations for completely specified functions still hold in this context.

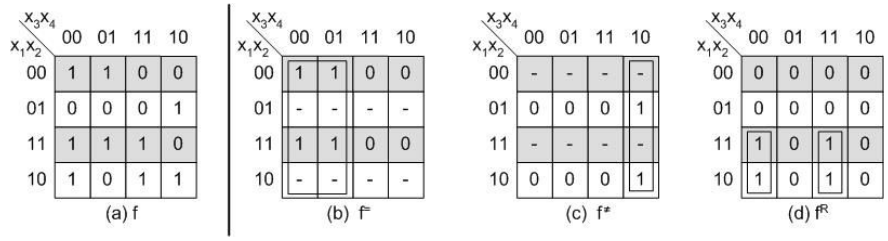

Fig. 3. The functions $f^=$, $f^{\neq}$, and f^R , together with the corresponding covers $f^= = \overline{x}_3$, $f^{\neq} = x_3\overline{x}_4$, and $f^R = x_1\overline{x}_3\overline{x}_4 + x_1x_3x_4$ used in a minimal PSOP expression for f.

We would like to point out that with the new Definitions 4 and 7 we do not need to introduce the two notions of PSOP expressions without and with remainder as done in [6], [13]. Indeed, the synthesis procedure, by means of the construction of the sets $f^=$, $f^{\neq}$, and f^R, will determine the most compact expression, which usually lies in between the two forms, i.e., it singles out only a subset of minterms that could form a crossing cube.

Finally, we observe that PSOP expressions share some similarities with *P-circuits*, a circuit structure studied in [9], [10], [12], [19]. P-circuits are decomposed Boolean expression where the intersection I between the cofactors, i.e., $I = f_{x_i=p} \cap f_{x_i \neq p}$, is used instead of the remainder R. The differences between the two expressions are due to the fact that the intersection I does not depend on the variable x_i, while the remainder R may depend on all the n input variables. Thus, PSOP circuits exhibit an higher level of flexibility exploring a larger optimization space.

4 Minimization of PSOP circuits

Before analyzing the details of the remainder computation, we consider the problem of minimizing a Boolean function in PSOP circuit form. We refer in the following only to the minimization of incompletely specified Boolean functions, as this case subsumes the problem of minimizing a completely specified function, whose don't care set is just the empty set.

Let f be an incompletely specified Boolean function depending on n variables, x_i a selected input variable and p a function possibly depending on all input variables except x_i. Consider the two cofactors $f|_{x_i=p}$ and $f|_{x_i \neq p}$, obtained by decomposing f with respect to the two subsets $x_i = p$ and $x_i \neq p$, and the remainder R. Recall that these three sets contain points in $\{0,1\}^n$. The final PSOP circuit for f is then given by three minimal SOPs representing $f^=$, $f^{\neq}$, and f^R as described in Definition 7.

So, once the remainder set R has been computed for the given cofactoring function p, as explained in the next Section 5, the problem is to find the sets $(f^=, f^{\neq}, f^R)$ that lead to a PSOP circuit of minimal cost, according to a given cost metric.

We now show how this problem can be nicely formalized and efficiently solved using Boolean relations. Our aim is to define a relation $\mathcal{R}_f$ whose set of compatible functions $\mathcal{F}(\mathcal{R}_f)$ corresponds exactly to the set of tuples $f^=$, $f^{\neq}$, and f^R occurring in all PSOP circuit implementations of f, with respect to a given variable x_i and a given cofactoring function p, so that an optimal solution of $\mathcal{R}_f$ defines an optimal (i,p) PSOP circuit, $\text{PSOP}^*_{(i,p)}(f)$, for f.

TABLE 1
Outputs $(f^=, f^{\neq}, f^R)$ for the relation $\mathcal{R}_f$ corresponding to a PSOP circuit implementation

$v \in \{0,1\}^n$	$v \notin R$	$v \in R$
$p = v_i,\ f(v) = 1$	$1-0$	$1--, --1$
$p = v_i,\ f(v) = -$	$--0$	$---$
$p = v_i,\ f(v) = 0$	$0-0$	
$p \neq v_i,\ f(v) = 1$	-10	$-1-, --1$
$p \neq v_i,\ f(v) = -$	$--0$	$---$
$p \neq v_i,\ f(v) = 0$	-00	

We underline that the choice of the cofactoring function p affects only the computation of the remainder R, but does not appear in the definition of the Boolean relation whose construction requires only the knowledge of R, independently of the procedure by which R was obtained.

Let $\mathcal{R}_f : \{0,1\}^n \to \{0,1\}^3$ be a Boolean relation, whose input set is the space spanned by the n input variables, while the output set describes all possible tuples of functions $f^=$, $f^{\neq}$, f^R defining a PSOP circuit for f. To construct correctly the relation $\mathcal{R}_f$, we must consider different cases, for the points in $\{0,1\}^n$ where $x_i = p$, and for the ones where $x_i \neq p$, i.e., for the points in $\{0,1\}^n$ where the value of the cofactoring function p is equal to the value of the i-th bit, and the points where these two values differ, as shown in Table 1.

Let us first consider the points with $x_i = p$. Thus, let $v = (v_1, \ldots, v_i, \ldots, v_n) \in \{0,1\}^n$ be such that the value of $p(v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)$ is equal to the i-th bit v_i.

1) $[f(v) = 1$ and $v \notin R]$ v is a point of the on-set of f that belongs to the on-set of the cofactor $f|_{x_i=p}$. Then, v must be necessarily inserted in $f^=$. Moreover, since in the PSOP expression, $f^{\neq}$ is multiplied by $(x_i \oplus p)$ and this factor evaluates to 0 on v, we can set v as a don't care point for $f^{\neq}$, so that it could be exploited to form larger cubes in that subset. Finally, observe that v does not belong to the remainder R, thus we must have $f^R(v) = 0$. Therefore, we pose $\mathcal{R}_f(v) = 1-0$.
2) $[f(v) = 1$ and $v \in R]$ v is a point of the on-set of f that belongs to the on-set of the cofactor $f|_{x_i=p}$ and that can be part of a crossing cube. Thus, v could be covered only by a cube entirely included in the subset where $x_i = p$, or it could be covered only by a crossing cube, or it could be covered by both cubes. Taking into account the fact that $(x_i \oplus p)$ evaluates to 0 on v so that we can set v as a don't care point for $f^{\neq}$, we then have these

possible output values for the relation: $\mathcal{R}_f(v) = \{1-0, 0-1, 1-1\} = \{1--, --1\}$.

3) $[f(x) = -$ and $v \notin R]$ Since v belongs to $f^{dc}|_{x_i=p}$ and $v \notin R$, we have these possible values for $f^=$, $f^{\neq}$ and f^R: $f^=(v) = -$, $f^{\neq}(v) = -$ as $(x_i \oplus p)$ evaluates to 0 on v, and $f^R(v) = 0$. Thus $\mathcal{R}_f(v) = --0$.
4) $[f(v) = -$ and $v \in R]$ If v is a point of $f^{dc}|_{x_i=p}$ that belongs to the remainder R, we have these possible values for $f^=$, $f^{\neq}$ and f^R: $f^=(v) = -$, $f^{\neq}(v) = -$ as $(x_i \oplus p)$ evaluates to 0 on v, and $f^R(v) = -$. Thus $\mathcal{R}_f(v) = ---$.
5) $[f(v) = 0]$ Since v is a point of $f^{off}|_{x_i=p}$, by construction $v \notin R$ and cannot be covered neither in the subset where $x_i = p$, nor in R. Thus, we pose $\mathcal{R}_f(v) = 0-0$, as we can set v as a don't care point for $f^{\neq}$.

The cases $v = (v_1, \ldots, v_n) \in \{0,1\}^n$ when $v_i \neq p(v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)$ are symmetrical and omitted.

We refer the reader to the next section for some explicit examples of Boolean relations $\mathcal{R}_f$ and of their solutions.

With this formalism, we can rephrase our PSOP minimization problem as the problem of finding an optimal implementation of $\mathcal{R}_f$, that is, of selecting among all possible three-output functions compatible with $\mathcal{R}_f$, the one defining a tuple $f^=$, $f^{\neq}$, and f^R whose overall SOP representation is minimal. In fact, recall that the three output variables of R_f are used to describe the tuple of functions defining a PSOP circuit for f: the first two outputs define $f^=$ and $f^{\neq}$, and the third defines f^R. Thus, each function in $\mathcal{F}(\mathcal{R}_f)$ corresponds to a possible tuple.

Theorem 1. The set $\mathcal{F}(\mathcal{R}_f)$ of all three-output functions compatible with the relation $\mathcal{R}_f$ specifies exactly the set of all tuples $f^=$, $f^{\neq}$, and f^R occurring in all PSOP circuit implementations of f, with respect to a given variable x_i and a given cofactoring function p.

Proof. First of all, observe that any PSOP circuit PSOP(f) defines a three-output function compatible with $\mathcal{R}_f$. The three functions $f^=$, $f^{\neq}$, and f^R represented by the three SOPs in PSOP(f) define the three outputs, and for all $x \in \{0,1\}^n$, $(f^=(x), f^{\neq}(x), f^R(x)) \in \mathcal{R}_f(x)$. Indeed, let $x = (x_1, \ldots, x_n) \in \{0,1\}^n$, and suppose that $x_i = p$ (the case $x_i \neq p$ is symmetrical and omitted). Recall that, by construction, x is always a don't care for $f|_{x_i \neq p}$. Then we have

- if $(f^=(x), f^{\neq}(x), f^R(x)) \in \{000, 010\}$, then $f|_{x_i=p}(x) \in \{0, -\}$; if $f|_{x_i=p}(x) = 0$, then x does not belong to the remainder set R, and $\mathcal{R}_f(x) = 0-0$ contains both 000 and 010; on the other hand, if $f|_{x_i=p}(x) = -$, we have $\mathcal{R}_f(x) = ---$ if $x \in R$, and $\mathcal{R}_f(x) = --0$ if $x \notin R$, and in both cases, 000 and 010 belong to $\mathcal{R}_f(x)$;
- if $(f^=(x), f^{\neq}(x), f^R(x)) \in \{100, 110\}$, then $f|_{x_i=p}(x) \in \{1, -\}$, and the construction of $\mathcal{R}_f$ guarantees that $(f^=(x), f^{\neq}(x), f^R(x)) \in \mathcal{R}_f(x)$, both for $x \in R$ and for $x \notin R$;
- if $(f^=(x), f^{\neq}(x), f^R(x)) \in \{001, 011\}$, then $x \in R$ and $f|_{x_i=p}(x) = -$, thus $(f^=(x), f^{\neq}(x), f^R(x)) \in \mathcal{R}_f(x)$, as $\mathcal{R}_f(x) = ---$;
- if $(f^=(x), f^{\neq}(x), f^R(x)) \in \{101, 111\}$, then $f|_{x_i=p}(x) \in \{1, -\}$, and $x \in R$. If $f|_{x_i=p}(x) = 1$, we have $\mathcal{R}_f(x) = \{1--, --1\}$, while if $f|_{x_i=p}(x) = -$, we have $\mathcal{R}_f(x) = ---$; in both cases, $(f^=(x), f^{\neq}(x), f^R(x)) \in \mathcal{R}_f(x)$.

We now prove that any three-output function compatible with $\mathcal{R}_f$ defines a PSOP circuit for f. First of all, note that the definition of R_f (see Table 1) guarantees that each function in $\mathcal{F}(\mathcal{R}_f)$ assumes value 1 on its first output $f^=$ on all points that belong to the on-set of $f|_{x_i=p}$ but not to the remainder R, and it assumes value 1 on its second output $f^{\neq}$ on all points that belong to the on-set of $f|_{x_i \neq p}$ but not to R. Thus, $f^=$ contains the subset $f^{on}|_{x_i=p} \setminus R$ and $f^{\neq}$ contains $f^{on}|_{x_i \neq p} \setminus R$, as required by Definition 7.

Now, observe that the definition of $\mathcal{R}_f$ also implies that $f^=$ and $f^{\neq}$ could also assume value 1 on the points that belong to the don't care set of the corresponding cofactor ($f^{dc}|_{x_i=p}$ for $f^=$ and $f^{dc}|_{x_i \neq p}$ for $f^{\neq}$), and also on the points of the remainder set R, which is a subset of $f^{on} \cup f^{dc}$. On the other hand, the relation always sets to 0 the values of $f^=$ and $f^{\neq}$ on all points in the off-set of the corresponding cofactor. Therefore, $f^= \subseteq f^{on}|_{x_i=p} \cup f^{dc}|_{x_i=p}$ and $f^{\neq} \subseteq f^{on}|_{x_i \neq p} \cup f^{dc}|_{x_i \neq p}$, as required by Definition 7.

Consider now the third output, defining the set f^R. From the definition of $\mathcal{R}_f$, it follows that for each function in $\mathcal{F}(\mathcal{R}_f)$, f^R can get the value 1 or - only on the points of R, while it is always 0 outside R. Thus, $\emptyset \subseteq f^R \subseteq R$.

To complete the proof we must show that for any function compatible with $\mathcal{R}_f$, the PSOP circuit PSOP(f) constructed using the tuple $(f^=, f^{\neq}, f^R)$ is a PSOP circuit for f, i.e., $f^{on} \subseteq$ PSOP$(f) \subseteq f^{on} \cup f^{dc}$. This follows from the way the relation is defined:

- Let $v \in f^{on}$. In all possible outputs of $\mathcal{R}_f(v)$ we have that either the third output f^R is set to 1, thus v is covered at least by the SOP for f^R, or the third output $f^R \in \{0, -\}$, but the output $f^=$ is set to 1 (if $v_i = p(v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)$) or the output $f^{\neq}$ is set to 1 (if $v_i \neq p(v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)$), and v is certainly covered by the SOP of its corresponding cofactor. Thus $f^{on} \subseteq$ PSOP(f).
- Let $v \in f^{off}$, and suppose that $v_i = p(v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)$. In all possible outputs of $\mathcal{R}_f(v)$ we have that both the first output $f^=$ and the third output f^R are set to 0. Thus, v could only be covered by the SOP for $f^{\neq}$, which is then multiplied by the factor $(\overline{x}_i \oplus p)$ that is 0 on v. Thus, PSOP(f) assumes value 0 on v. The case $v \in f^{off}$, with $v_i \neq p(v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)$ is symmetrical and omitted. This proves that PSOP$(f) \cap f^{off} = \emptyset$.
- Finally, if $v \in f^{dc}$, then v might be covered by a SOP for f^R and/or by the SOP of the corresponding cofactor, or by neither of them. Thus, PSOP$(f) \subseteq f^{on} \cup f^{dc}$.

■

Corollary 1. An optimal solution of the Boolean relation $\mathcal{R}_f$, according to a given cost function μ chosen to evaluate PSOP circuits, defines an optimal (i,p) PSOP circuit, PSOP$^*_{(i,p)}(f)$, for f with respect to the same cost function μ.

Proof. The thesis immediately follows from Theorem 1, as any three-output function compatible with $\mathcal{R}_f$ defines a possible PSOP circuit implementation for f whose cost, under any given cost metric μ, is determined by the cost under μ of the SOP representations of $f^=$, $f^{\neq}$ and f^R. ■

5 Remainder computation

In the previous section we have described a Boolean method, based on Boolean relations, for the synthesis of general PSOP expressions decomposed with respect to any cofactoring function p. Indeed, as shown in Table 1, the high-level structure of the Boolean relation $\mathcal{R}_f$ is valid for any p. However, to define explicitly and solve the relation $\mathcal{R}_f$, we must compute the remainder set R for the given function p. To this aim, in this section, we first characterize and then show how to compute the remainder set for some cofactoring functions. In particular, we will derive an algebraic formula for the remainder set, which can be used directly to compute it.

Let f be an incompletely specified Boolean function, x_i a selected input variable, and p a function depending on a subset of the input variables not including x_i. Recall from Section 3 that the remainder R contains all on-set and dc-set minterms that could form a crossing cube, i.e., a cube intersecting both subsets $x_i = p$ and $x_i \neq p$. In general, by Definition 5, a point $v \in \{0,1\}^n$ belongs to R if and only if there is another point u in the on-set or dc-set of f with Hamming distance 1 such that v and u do not belong to the same subset $x_i = p$ or $x_i \neq p$. The last condition is the one that strictly depends on the chosen function p. We discuss three different cases: *(i)* the simple cofactoring function $p = x_j$; *(ii)* its generalization consisting of a linear combination (EXOR) of two or more distinct variables: $p = x_{j_1} \oplus x_{j_2} \oplus \ldots \oplus x_{j_k}$; and *(iii)* a non linear cofactoring function defined as the AND of two or more distinct variables: $p = \bigwedge_{\ell=1}^{k} x_{j_\ell}$.

5.1 A simple cofactoring function $p = x_j$

First of all we consider a cofactoring function consisting in just one variable x_j, with $j \neq i$. Thus, we consider the decomposition of the form

$$\mathrm{PSOP}(f) = (\overline{x}_i \oplus x_j)\, f^= + (x_i \oplus x_j)\, f^{\neq} + f^R\,,$$

where, according to Definition 7,

1) $(f^{on}|_{x_i=x_j} \setminus R) \subseteq f^= \subseteq f^{on}|_{x_i=x_j} \cup f^{dc}|_{x_i=x_j}$
2) $(f^{on}|_{x_i\neq x_j} \setminus R) \subseteq f^{\neq} \subseteq f^{on}|_{x_i\neq x_j} \cup f^{dc}|_{x_i\neq x_j}$
3) $\emptyset \subseteq f^R \subseteq R$
4) $f^{on} \subseteq \mathrm{PSOP}(f) \subseteq f^{on} \cup f^{dc}$.

As already observed, when a single variable x_j ($j \neq i$) is used as cofactoring function p, PSOP expressions can be considered the Boolean version of the *EXOR-Projected Sums of Products (EP-SOPs)* forms introduced in [6].

Let us suppose for the moment that f is completely specified, i.e., f^{dc} is empty.

For this particular decomposition, we can observe that the remainder is composed by all points $v \in f^{on}$ that can form a cube with the point u obtained complementing in v the i-th or the j-th variable. Indeed, in this case, u and v have Hamming distance 1, and belong to different subsets: if v is such that $v_i \oplus v_j = 0$ (i.e., v belongs to the subset where $x_i = x_j$), then u will be such that $u_i \oplus u_j = v_i \oplus v_j \oplus 1 = 1$ (i.e., u belongs to the subset where $x_i \neq x_j$). We pose the following definition:

Definition 8. Given a point $x \in \{0,1\}^n$, the *k-neighbor* $x^{(k)} \in \{0,1\}^n$ of x is the point obtained complementing the k-th bit of x, for $1 \leq k \leq n$.

Thus, since a minterm of f can be part of a crossing cube if and only if f takes the value 1 on at least one of its i and j-neighbors, we can state the following definition.

Definition 9. The *remainder* R of a completely specified function f with respect to the generalized decomposition onto the subsets $(\overline{x}_i \oplus x_j)$ and $(x_i \oplus x_j)$ is given by

$$R = \{x \in \{0,1\}^n \mid f(x) = 1 \wedge (f(x^{(i)}) = 1 \vee f(x^{(j)}) = 1)\}\,.$$

Figure 3(d) shows the remainder for the function f of the running example from Section 3, with $i = 1$ and $p = x_2$, i.e., the set of points of f that have at least a 1-neighbor or a 2-neighbor.

We now discuss a simple way to actually compute the remainder. Let $f^i|_{x_i=x_j}$ be the function obtained from the cofactor $f^{on}|_{x_i=x_j}$ by deleting all occurrences of x_i in its minterms and $f^i|_{x_i\neq x_j}$ be the function obtained deleting all occurrences of x_i from the minterms of $f^{on}|_{x_i\neq x_j}$. Observe that $f^i|_{x_i=x_j}$ and $f^i|_{x_i\neq x_j}$ are two degenerate functions as they do not depend on x_i. Therefore, each minterm of f corresponds to two minterms in $f^i|_{x_i=x_j}$ or in $f^i|_{x_i\neq x_j}$.

For example, consider the function f in Figure 4(a), $i = 1$, and $p = x_2$. $f^1|_{x_1=x_2}$ and $f^1|_{x_1\neq x_2}$ are shown in Figures 4(b) and 4(c), respectively. The minterm 0000 of f (in Figure 4(a)) corresponds to the minterms 0000 and 1000 in $f^1|_{x_1=x_2}$ (Figure 4(b)), while the minterm 1000 of f corresponds to the minterms 0000 and 1000 in $f^1|_{x_1\neq x_2}$ (Figure 4(c)).

Analogously, let $f^j|_{x_i=x_j}$ and $f^j|_{x_i\neq x_j}$ denote the two degenerate functions obtained from $f^{on}|_{x_i=x_j}$ and $f^{on}|_{x_i\neq x_j}$ by eliminating all occurrences of x_j, for instance see Figures 4(e) and 4(f).

The remainder R can be computed using the algebraic formula provided in the following proposition (see Figure 4 for the running example).

Proposition 1. The *remainder* R of the decomposition of a completely specified Boolean function f with respect to the two subsets where $x_i = x_j$ and $x_i \neq x_j$ is given by

$$R = (f^i|_{x_i=x_j} \cap f^i|_{x_i\neq x_j}) \cup (f^j|_{x_i=x_j} \cap f^j|_{x_i\neq x_j})\,.$$

Proof. The thesis immediately follows observing that the set $(f^i|_{x_i=x_j} \cap f^i|_{x_i\neq x_j})$ identifies all pairs of minterms that differ only on the i-th variable, while $(f^j|_{x_i=x_j} \cap f^j|_{x_i\neq x_j})$ defines all pairs of minterms that differ only on the j-th variable, as in Definition 9. ■

Once the set R has been computed, the Boolean relation $\mathcal{R}_f$ can be constructed and minimized, in order to find an optimal PSOP circuit $\mathrm{PSOP}^*_{(i,x_j)}(f)$ for the target function f.

For instance, for the running example in Figures 3 and 4, the reminder is $R = \{0000, 1000, 1011, 1100, 1111\}$. The corresponding Boolean relation is shown in Table 2. A solution of the Boolean relation, minimal with respect to the number

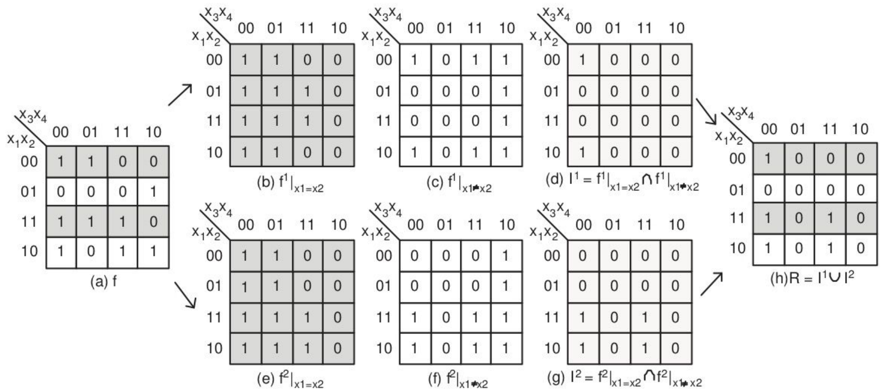

Fig. 4. Construction of the remainder R for the function f of the running example.

TABLE 2
Boolean relation for the example in Figure 3

0000	1 – –, – – 1
0001	1–0
0010	0–0
0011	0–0
0100	–00
0101	–00
0110	–10
0111	–00
1000	– 1 –, – – 1
1001	–00
1010	–10
1011	– 1 –, – – 1
1100	1 – –, – – 1
1101	1–0
1110	0–0
1111	1 – –, – – 1

TABLE 3
A solution for the Boolean relation in Table 2

– – 0 –	100
– – 10	010
1 – 00	001
1 – 11	001

of products, is shown is Table 3, which corresponds to the PSOP expression

$$\begin{aligned}\text{PSOP}^*_{(1,x_2)}(f) &= (\overline{x}_1 \oplus x_2)(\overline{x}_3) + (x_1 \oplus x_2)(x_3\overline{x}_4) \\ &\quad +(x_1\overline{x}_3\overline{x}_4 + x_1x_3x_4)\,.\end{aligned}$$

Now, suppose that the target function f is incompletely specified. Then, we have

$$R = \{x \in f^{on} \cup f^{dc} \,|(x^{(i)} \in f^{on} \cup f^{dc}) \vee (x^{(j)} \in f^{on} \cup f^{dc})\}\,,$$

and, as before, we can give a constructive definition for R, that can be applied to compute this set. Let $f^{on,i}|_{x_i=x_j}$ and $f^{dc,i}|_{x_i=x_j}$ be the sets of points in $\{0,1\}^n$ obtained from $f^{on}|_{x_i=x_j}$ and $f^{dc}|_{x_i=x_j}$ by eliminating all occurrences of x_i, and let $f^{on,i}|_{x_i\neq x_j}$ and $f^{dc,i}|_{x_i\neq x_j}$ be the sets of points obtained deleting all occurrences of x_i from $f^{on}|_{x_i\neq x_j}$ and $f^{dc}|_{x_i\neq x_j}$. Analogously, let $f^{on,j}|_{x_i=x_j}$, $f^{dc,j}|_{x_i=x_j}$, $f^{on,j}|_{x_i\neq x_j}$, and $f^{dc,j}|_{x_i\neq x_j}$ be the sets obtained eliminating the variable x_j from $f^{on}|_{x_i=x_j}$, $f^{dc}|_{x_i=x_j}$, $f^{on}|_{x_i\neq x_j}$, and $f^{dc}|_{x_i\neq x_j}$. Then, we have

Proposition 2. The *remainder* R of the decomposition of an incompletely specified Boolean function f with respect to the two subsets where $x_i = x_j$ and $x_i \neq x_j$ is given by
$R = ((f^{on,i}|_{x_i=x_j} \cup f^{dc,i}|_{x_i=x_j}) \cap (f^{on,i}|_{x_i\neq x_j} \cup f^{dc,i}|_{x_i\neq x_j})) \cup ((f^{on,j}|_{x_i=x_j} \cup f^{dc,j}|_{x_i=x_j}) \cap (f^{on,j}|_{x_i\neq x_j} \cup f^{dc,j}|_{x_i\neq x_j}))\,.$

5.2 Linear cofactoring functions

We now consider a generalization of the cofactoring function $p = x_j$, that is, we define p as a linear combination (EXOR) of two or more distinct variables: $p = x_{j_1} \oplus x_{j_2} \oplus \ldots \oplus x_{j_k}$. Thus, we consider the projection of a target function f onto the two subsets where $x_i = x_{j_1} \oplus x_{j_2} \oplus \ldots \oplus x_{j_k}$ and $x_i \neq x_{j_1} \oplus x_{j_2} \oplus \ldots \oplus x_{j_k}$, respectively, with $x_i \neq x_{j_\ell}$ for all $1 \leq \ell \leq k$, and $k < n$. As before, we first suppose that f is a completely specified function.

To simplify the notation, let us denote the two projection subsets $(x_i \oplus x_{j_1} \oplus \ldots \oplus x_{j_k})$ and $(\overline{x}_i \oplus x_{j_1} \oplus \ldots \oplus x_{j_k})$ as S and S^c, respectively. Moreover, let $I = \{i, j_1, \ldots, j_k\}$ be the set of variable indices that define S and S^c.

In the following proposition, we characterize the remainder set R corresponding to this decomposition.

Proposition 3. The *remainder* R of a completely specified function f with respect to the generalized decomposition onto the subsets S and S^c is given by

$$R = \{v \in f^{on} \mid \bigvee_{\ell \in I} f(v^{(\ell)}) = 1\}\,,$$

where $v^{(\ell)}$ is the ℓ-neighbor of v.

Proof. First of all, observe that for any $v \in \{0,1\}^n$ and any $\ell \in I$, $v \in S$ if and only if its ℓ-neighbor $v^{(\ell)}$, i.e., the minterm obtained complementing the ℓ-th bit of v, does not belong to the set S, that is $v^{(\ell)} \in S^c$. Indeed, if $v \in S$, then $v_i \oplus v_{j_1} \oplus \ldots \oplus v_{j_k} = 1$, and if we complement exactly one of these variables, the EXOR-factor changes value from 1 to 0, so that $v^{(\ell)} \in S^c$, for any $\ell \in I$. On the other hand, the minterm derived complementing in v any other variable, not occurring in the characteristic functions of S and S^c, still belongs to the original subset, implying that for any $t \notin I$, $v \in S$ if and only if $v^{(t)} \in S$.

Now, let $v \in f^{on}$. As previously recalled, $v \in R$ if and only if there exists a minterm $u \in f^{on}$ such that v and u

are neighbors, and v and u are not both in S or both in S^c. Thus, the previous observation implies that $v \in R$ if and only if there exists $\ell \in I$ such that $v^{(\ell)} \in f^{on}$, and the thesis immediately follows. ■

The actual computation of the remainder R can be performed exploiting the characterization of Proposition 3 and generalizing Proposition 1. For any $\ell \in I$, let $f^{on,\ell}|_{x_i=p}$ and $f^{on,\ell}|_{x_i\neq p}$ denote the two degenerate functions obtained from $f^{on}|_{x_i=p}$ and $f^{on}|_{x_i\neq p}$ by eliminating all occurrences of x_ℓ from their minterms.

Proposition 4. The *remainder* R of the decomposition of a completely specified Boolean function f with respect to the two subsets where $x_i = p$ and $x_i \neq p$ is given by

$$R = \bigcup_{\ell \in I} \left(f^{on,\ell}|_{x_i=p} \cap f^{on,\ell}|_{x_i\neq p} \right) .$$

Proof. The thesis immediately follows from Proposition 3 observing that the sets $(f^{on,\ell}|_{x_i=p} \cap f^{on,\ell}|_{x_i\neq p})$ identify all pairs of on-set minterms that differ only on the ℓ-th variable, for any $\ell \in I$. ■

For an example, consider again the Boolean function in Figure 3(a) and suppose that $i = 1$ and $p = x_2 \oplus x_3$. In this case we have that $f^{on}|_{x_1=x_2\oplus x_3} = \{0000, 0001, 0110, 1010, 1011, 1100, 1101\}$ and $f^{on}|_{x_1\neq x_2\oplus x_3} = \{1000, 1111\}$. Since $I = \{1, 2, 3\}$, we have

$$R = \bigcup_{\ell \in \{1,2,3\}} (f^{on,\ell}|_{(\overline{x}_1\oplus x_2\oplus x_3)} \cap f^{on,\ell}|_{(x_1\oplus x_2\oplus x_3)} ,$$

where

$f^{on,1}|_{(\overline{x}_1\oplus x_2\oplus x_3)} \cap f^{on,1}|_{(x_1\oplus x_2\oplus x_3)} = \{-000,\ -001,\ -110,\ -010,\ -011,\ -100,\ -101\} \cap \{-000,\ -111\} = \{-000\} = \{0000, 1000\}$,

$f^{on,2}|_{(\overline{x}_1\oplus x_2\oplus x_3)} \cap f^{on,2}|_{(x_1\oplus x_2\oplus x_3)} = \{0-00, 0-01, 0-10, 1-10, 1-11, 1-00, 1-01\} \cap \{1-00, 1-11\} = \{1-00, 1-11\} = \{1000,\ 1100,\ 1011,\ 1111\}$,

$f^{on,3}|_{(\overline{x}_1\oplus x_2\oplus x_3)} \cap f^{on,3}|_{(x_1\oplus x_2\oplus x_3)} = \{00-0, 00-1, 01-0, 10-0, 10-1, 11-0, 11-1\} \cap \{10-0, 11-1\} = \{10-0, 11-1\} = \{1000,\ 1010,\ 1101,\ 1111\}$.

Consequently, the reminder is

$$R = \{0000, 1000, 1010, 1011, 1100, 1101, 1111\} .$$

In this case the PSOP form, derived from the corresponding Boolean relation, is:

$$\begin{aligned} \mathrm{PSOP}^*_{(1,x_2\oplus x_3)}(f) &= (\overline{x}_1 \oplus x_2 \oplus x_3)(\overline{x}_3 + \overline{x}_4) \\ &\quad + (x_1x_3x_4 + x_1\overline{x}_3\overline{x}_4) . \end{aligned}$$

We can give a constructive definition for the remainder R, that can be applied to compute this set, even in the more general case of an incompletely specified Boolean function $f = \{f^{on}, f^{dc}\}$. Recall that the points potentially included in a crossing cube can now be defined as the points v in f^{on} or in f^{dc}, for which there exists $\ell \in I$ such that $v^{(\ell)} \in f^{on} \cup f^{dc}$.

Proposition 5. The *remainder* R of the decomposition of an incompletely specified Boolean function f with respect to the two subsets where $x_i = p$ and $x_i \neq p$ is given by

$$R = \bigcup_{\ell \in I} \left[\left(f^{on,\ell}|_{x_i=p} \cup f^{dc,\ell}|_{x_i=p} \right) \cap \left(f^{on,\ell}|_{x_i\neq p} \cup f^{dc,\ell}|_{x_i\neq p} \right) \right] ,$$

where, for any $\ell \in I$, $f^{on,\ell}|_{x_i=p}$, $f^{dc,\ell}|_{x_i=p}$, $f^{on,\ell}|_{x_i\neq p}$, and $f^{dc,\ell}|_{x_i\neq p}$ denote the sets obtained eliminating the variable x_ℓ from $f^{on}|_{x_i=p}$, $f^{dc}|_{x_i=p}$, $f^{on}|_{x_i\neq p}$, and $f^{dc}|_{x_i\neq p}$.

5.3 Cofactoring functions based on the AND operation

We now consider an example of non linear cofactoring function, and in particular we study the cofactoring function defined as the AND of two or more distinct variables: $p = \bigwedge_{\ell=1}^{k} x_{j_\ell}$. Thus, we consider the projection of a function f onto the two subsets with characteristic functions $(x_i \oplus \bigwedge_{\ell=1}^{k} x_{j_\ell})$ and $(\overline{x}_i \oplus \bigwedge_{\ell=1}^{k} x_{j_\ell})$, respectively, where $x_i \neq x_{j_\ell}$ for all $1 \leq \ell \leq k$ and $k < n$.

Let S and S^c denote the two projection subsets, and let $J = \{j_1, \ldots, j_k\}$ denote the set of variable indices that define the cofactoring function p.

Proposition 6. The *remainder* R of a completely specified function f with respect to the generalized decomposition onto the subsets S and S^c is given by

$$\begin{aligned} R \;\; = \;\; & \{v \in f^{on} \mid (v^{(i)} \in f^{on}) \;\vee\; (\exists\, t \in J \\ & \text{s.t. } f(v^{(t)}) = 1 \;\;\wedge \bigwedge_{j \in J\setminus\{t\}} v_j = 1)\} . \end{aligned}$$

Proof. To define the remainder set R we must characterize all pairs of on-set minterms that differ for only one bit and do not belong to the same projection subset, as these are the minterms that can form a crossing cube. To this aim, we first observe that for any minterm v, and for all $t \neq i$, $t \notin J$, v and its t-neighbor $v^{(t)}$ always belong to the same subset, either S or S^c, since the characteristic functions of S and S^c do not depend on the t-th input variable. Thus, these minterms do not belong to R. On the other hand, v and its i-neighbor $v^{(i)}$ belong to different subsets; indeed if we complement the i-th variable in the expressions that define the projection subsets S and S^c, their value always changes from 1 to 0 or from 0 to 1. Thus, if both v and $v^{(i)}$ are in f^{on}, they can be part of a crossing cube, and must be inserted in R.

Finally, if we consider any index $t \in J$, then v and $v^{(t)}$ belong to different subsets if and only if all other variables occurring in p are equal to 1. Indeed, whenever at least one of these variables is equal to 0, the value of the expression $x_i \oplus \bigwedge_{\ell=1}^{k} x_{j_\ell}$ evaluated on v becomes equal to v_i, and does not change complementing the t-th variable. Thus, v and $v^{(t)}$ belong to the remainder set R if and only if *(i)* they are both in f^{on}; and *(ii)* for all $j \in J \setminus \{t\}$, $v_j = 1$. ■

The remainder R can be computed exploiting this characterization, as detailed in the following proposition. Let $f^{on,i}|_{x_i=p}$ and $f^{on,i}|_{x_i\neq p}$ denote the two degenerate functions obtained from $f^{on}|_{x_i=p}$ and $f^{on}|_{x_i\neq p}$ by eliminating all occurrences of x_i from their minterms. Moreover, for any $t \in J$, let $f^{on,t}|_{x_i=p,(\forall j\in J\setminus\{t\},x_j=1)}$ and $f^{on,t}|_{x_i\neq p,(\forall j\in J\setminus\{t\},x_j=1)}$ denote the functions obtained from $f^{on}|_{x_i=p}$ and $f^{on}|_{x_i\neq p}$ by eliminating all occurrences of x_t from the minterms where all other variables defining p are equal to 1.

Proposition 7. The *remainder* R of the decomposition of a completely specified Boolean function f with respect to the two subsets where $x_i = p$ and $x_i \neq p$ is given by

$$R = (f^{on,i}|_{x_i=p} \cap f^{on,i}|_{x_i \neq p}) \bigcup_{t \in J} \left(f^{on,t}|_{x_i=p,(\forall j \in J \setminus \{t\}, x_j=1)} \cap f^{on,t}|_{x_i \neq p,(\forall j \in J \setminus \{t\}, x_j=1)} \right).$$

Proof. The thesis immediately follows from Proposition 6. ■

For example, let us consider the Boolean function in Figure 3(a), and suppose that $i = 1$ and $p = x_2 \wedge x_3$. In this case, we have that $f^{on}|_{x_1=x_2\wedge x_3} = \{0000, 0001, 1111\}$, $f^{on}|_{x_1 \neq x_2 \wedge x_3} = \{0110, 1000, 1010, 1011, 1100, 1101\}$, and $R = \{0000, 1000, 1011, 1101, 1111\}$. Since $J = \{2, 3\}$, we have

$$\begin{aligned} R &= (f^{on,1}|_{\overline{x}_1 \oplus (x_2 \wedge x_3)} \cap f^{on,1}|_{x_1 \oplus (x_2 \wedge x_3)}) \\ &\cup \left(f^{on,2}|_{\overline{x}_1 \oplus (x_2 \wedge x_3), x_3=1} \cap f^{on,2}|_{x_1 \oplus (x_2 \wedge x_3), x_3=1}\right) \\ &\cup \left(f^{on,3}|_{\overline{x}_1 \oplus (x_2 \wedge x_3), x_2=1} \cap f^{on,3}|_{x_1 \oplus (x_2 \wedge x_3), x_2=1}\right). \end{aligned}$$

where
$f^{on,1}|_{(\overline{x}_1 \oplus (x_2 \wedge x_3))} \cap f^{on,1}|_{(x_1 \oplus (x_2 \wedge x_3))} = \{-000, -001, -111\} \cap \{-110, -000, -010, -011, -100, -101\} = \{-000\} = \{0000, 1000\}$,
$f^{on,2}|_{\overline{x}_1 \oplus (x_2 \wedge x_3), x_3=1} \cap f^{on,2}|_{x_1 \oplus (x_2 \wedge x_3), x_3=1} = \{1-11\} \cap \{0-10, 1-10, 1-11\} = \{1-11\} = \{1011, 1111\}$,
$f^{on,3}|_{\overline{x}_1 \oplus (x_2 \wedge x_3), x_2=1} \cap f^{on,3}|_{x_1 \oplus (x_2 \wedge x_3), x_2=1} = \{11-1\} \cap \{01-0, 11-0, 11-1\} = \{11-1\} = \{1101, 1111\}$.
Consequently, the reminder is

$$R = \{0000, 1000, 1011, 1101, 1111\}.$$

The PSOP form, computed by minimizing the corresponding Boolean relation, is:

$$\begin{aligned} \text{PSOP}^*_{(1, x_2 \wedge x_3)}(f) &= (\overline{x}_1 \oplus (x_2 \wedge x_3))(\overline{x}_2 \overline{x}_3) \\ &\quad + (x_1 \oplus (x_2 \wedge x_3))(\overline{x}_4) \\ &\quad + (x_1 x_2 x_4 + x_1 x_3 x_4). \end{aligned}$$

Similarly to the previous case studies, the constructive definition of the remainder can be generalized to incompletely specified Boolean functions by including in the set R the don't-care minterms that can be part of a crossing cube (details are omitted).

We finally observe that the case of cofactoring functions based on the OR operation can be studied in a very similar way, as the OR function is the dual of the AND function.

6 Experimental Results

In this section we report the experimental results of the minimization of PSOP circuits based on Boolean relations.

We conducted two different experimental evaluations. The first one, discussed in Section 6.1, compares PSOP circuits, with cofactoring function $p = x_j$, vs. standard SOP forms, and vs. EP-SOP forms with remainder [13]. Our aim is to demonstrate that modeling the PSOP minimization problem using Boolean relations yields significant gains in area and delay, both with respect to classical forms, and to similar bounded-level forms. Recall that the differences between the new proposed PSOP forms and the EP-SOP expressions lies basically in the definition and projection of the remainder set: while in [6], [13], the remainder is defined algebraically and left unprojected in the final form, here the remainder is defined and built exploiting the flexibility of Boolean relations, and can be partially projected in order to derive a smaller expression.

The aim of the second experimental evaluation, discussed in Section 6.2, is to compare the effect of different cofactoring functions, in order to understand what strategy could lead to better results. In more details, we compare area, delay and synthesis time of PSOP forms with the simple cofactoring function $p = x_j$ (described in [18] and in Section 5.1), vs. area, delay and synthesis time of PSOP expressions with cofactoring functions $p = x_j \oplus x_k$ and $p = x_j \wedge x_k$ (defined in Sections 5.2 and 5.3, respectively).

The algorithms have been implemented in C, using the CUDD library for OBDDs to represent Boolean functions, and BREL [5] for the synthesis of Boolean relations since it finds better solutions in shorter runtime than the previously known methods. The experiments have been run on a Linux Intel Core i7, 3.60 GHz CPU with 8 GB of main memory. The benchmarks are taken from LGSynth93 [40], ITC99 [25], and EPFL Benchmarks [1] . Multioutput benchmarks have been synthesized minimizing each single output independently from the others. We report in the following a significant subset of the functions as representative indicators of our experiments. To evaluate the obtained circuits in area and delay, we ran them using the SIS system with the MCNC library for technology mapping and the SIS command `map -W -f 3 -s`.

6.1 Minimization of PSOP with Boolean relations

In the first experiment, we refer only to the simple cofactoring function $p = x_j$, in order to compare the new proposed Boolean approach, vs. the previous algebraic methods discussed in [6], [13].

To determine the two variables x_i and x_j involved in the decomposition, we search the most frequent pair of variables present in an initial SOP representation of the input function. This choice is based on the experimental results previously obtained in [13]. As some benchmarks have multiple outputs, we compute frequency over the whole set of outputs (global frequency), thus employing the same variables for all outputs.

To show the gain in area and delay of PSOP circuits derived using Boolean relations, we compare them vs. plain SOP forms, synthesized using ESPRESSO [35], and vs. the EP-SOP with remainder forms discussed in [13]. These results are summarized in Table 4. The first two columns report the name of the benchmarks and the number of their inputs and outputs. The following ones report, by groups of three, mapped areas, delays and synthesis times in seconds. The first two groups, labeled "PSOP - Exact mode" and "PSOP - Heuristic mode", refer to PSOP circuits with cofactoring function $p = x_j$ synthesized with the new algorithm based on Boolean relations; the first one has a cost function that minimizes the number of literals in an exact mode, and the second one has a cost function that minimizes the number of literals in a heuristic mode. The third group provides the results for plain SOP forms. The last group provides the results for the EP-SOP with remainder forms proposed in [13]. For each benchmark we underline in bold

TABLE 4
Comparison of SOP, Algebraic EP-SOP [13], and the proposed Boolean approach (PSOP) in exact and heuristic mode, for the case $p = x_j$

		PSOP - Exact mode			PSOP - Heuristic mode			SOP - ESPRESSO			EP-SOP - Alg [13]		
Bench	**in/out**	**Area**	**Delay**	**Time**	**Area**	**Delay**	**Time**	**Area**	**Delay**	**Time**	**Area**	**Delay**	**Time**
addm4	9/8	**732**	35.60	9.00	823	38.50	0.05	959	41.9	0.02	1806	65.50	0.96
alu1	12/8	64	8.60	0.15	64	8.60	0.01	**53**	6.8	0.01	60	8.80	0.01
amd	14/24	**786**	30.10	9.19	996	37.80	0.17	986	37.3	0.01	994	43.30	0.03
b12	15/9	**155**	15.80	1.35	187	20.20	0.01	166	16.4	0.01	220	22.50	0.09
co14	14/1	**146**	28.80	0.36	**146**	28.80	0.01	175	34.0	0.01	169	31.90	0.01
in0	15/11	**956**	38.20	12.46	1066	38.80	0.29	1032	54.2	0.01	1050	46.70	0.15
in2	19/10	**967**	38.90	32.27	1102	38.10	0.34	993	40.6	0.01	969	38.50	0.09
in5	24/14	**856**	32.20	7.04	940	35.70	0.06	865	38.5	0.01	935	38.60	0.09
in7	26/10	389	21.90	5.87	405	22.70	0.16	**319**	23.6	0.01	395	27.60	0.04
m181	15/9	**156**	15.80	1.38	197	21.00	0.01	174	16.4	0.01	221	22.50	0.01
m2	8/16	**419**	26.20	1.93	429	28.40	0.01	955	42.5	0.01	815	38.90	0.11
m3	8/16	543	29.80	4.75	**535**	29.00	0.01	1269	51.5	0.01	1148	48.70	0.07
mlp4	8/8	**518**	27.40	6.46	576	28.30	0.02	686	36.4	0.01	1180	50.60	0.18
mp2d	14/14	266	20.30	1.47	265	20.30	0.04	**251**	19.8	0.01	389	23.80	0.03
newtpla	15/5	117	18.50	0.84	117	18.50	0.01	**104**	19.7	0.01	122	18.20	0.01
rckl	32/7	**352**	49.80	1.72	**352**	49.80	0.04	489	72.3	0.01	494	51.50	0.01
t3	12/8	174	15.90	0.24	201	18.80	0.01	**166**	17.1	0.01	175	16.90	0.01
tms	8/16	**459**	30.30	2.30	473	29.40	0.01	647	37.4	0.01	657	38.80	0.05
vg2	25/8	468	22.50	4.78	517	22.80	0.12	**341**	18.6	0.01	633	26.70	0.03
vtx1	27/6	330	23.20	2.44	370	23.80	0.05	**324**	21.3	0.01	364	27.20	0.03
x6dn	39/5	**246**	25.30	2.54	250	20.10	0.02	762	31.2	0.01	851	36.60	0.03
x9dn	27/7	450	25.80	3.29	400	23.80	0.05	**384**	23.0	0.01	420	24.50	0.03

the circuit that exhibits the best area result. Note that the areas reported in the last group of Table 4 (i.e, column 12) refer to areas after the technology mapping evaluated with SIS. On the other hand, the corresponding values reported in [13] refer to areas before technology mapping.

TABLE 5
Gains vs. other minimization techniques

	PSOP - Exact mode			PSOP - Heuristic mode		
	Area	*Delay*	*Time*	*Area*	*Delay*	*Time*
SOP	24%	15%	-223786%	15%	9%	-2643%
EP-SOP alg [13]	35%	23%	-5578%	28%	17%	30%

The results show that modeling the PSOP circuit minimization problem using Boolean relations pays significantly. In fact, PSOP circuits synthesized with Boolean relations turned out to be more compact than the corresponding EP-SOP-circuits in [13] in about 90% of our experiments. The area gain of PSOP circuits synthesized with Boolean relations in exact (heuristic) mode is 35% (28%) on average with respect to EP-SOP circuits in [13], and the gain in the delay is of about 23% (17%). Finally, the PSOP circuits synthesized with Boolean relations in exact (heuristic) mode are smaller than the corresponding SOP forms in about 68% of our experiments, with an average gain in area of 24% (15%), and in delay of 15% (9%).

Comparing the performances of the two new algorithms to the previous results, we notice how the cost function can be critical: minimizing Boolean relations in the exact mode can be very time-expensive (on average, 5578% penalty in computational time with respect to [13]), while with the heuristic mode we obtain the best-performing algorithm (30% gain in computational time, on average, with respect to [13]). For a complete comparison of average gains see Table 5. The algorithm based on Boolean relations in the heuristic mode exhibits a performing behavior with a good trade-off between area and delay minimization and computational time.

6.2 Comparison among different cofactoring functions

The aim of the second experimental evaluation is the comparison among three different cofactoring functions: $p = x_j$, $p = x_j \oplus x_k$ and $p = x_j \wedge x_k$. To determine the three variables x_i, x_j, and x_k involved in the projections onto the subspaces $x_i = x_j \oplus x_k$, $x_i \neq x_j \oplus x_k$, $x_i = x_j \wedge x_k$ and $x_i \neq x_j \wedge x_k$, we search the most frequent triplet of variables present in the initial SOP representation of the input function.

In Table 6 we report mapped area and delay of PSOP circuits implemented using the decomposition and the remainder computation explained in Sections 5.1, 5.2, and 5.3. As before, we have synthesized PSOP circuits using the Boolean relation minimizer BREL both in the exact and heuristic mode.

The first two columns of Table 6 report the name of the benchmarks and the number of their inputs and outputs. The following three groups, of four columns each, report areas and delays, of circuits obtained with exact and heuristic mode, for the three cofactoring functions . For each benchmark we underline in bold the circuit that exhibits the best area result.

The results of this evaluation are summarized in Table 7. As an outcome, the simplest cofactoring function $p = x_j$ provides the best results in area for about 50% of the benchmarks. For the remaining 50%, the cofactoring function $p = x_j \oplus x_k$ provides the best results in area in the 27% of the cases, and the cofactoring function $p = x_j \wedge x_k$ in the 22%. In the remaining 1% of benchmarks, two or all the cofactoring functions yield the same final area.

For the delay, we have a different outcome: the function $p = x_j \wedge x_k$ provides the best delay for 36% of the benchmarks, while the other two functions $p = x_j$ and $p = x_j \oplus x_k$ yield the best results in the 32% and 24% of the cases, respectively; for the remaining 8% of benchmarks none of the three cofactoring functions proved to be strictly better than the others.

Finally, interesting enough, we observe that the computational times for the three different cofactoring fuctions, i.e., $p = x_j$, $p = x_j \oplus x_k$, and $p = x_j \wedge x_k$, are very similar.

Also in this case the results show that modeling the

TABLE 6
Comparison of different cofactoring fuctions, i.e., $p = x_j$, $p = x_j \oplus x_k$, and $p = x_j \wedge x_k$

		$p = x_j$				$p = x_j \oplus x_k$				$p = x_j \wedge x_k$			
		Exact mode		Heuristic mode		Exact mode		Heuristic mode		Exact mode		Heuristic mode	
Bench	**in/out**	*Area*	*Delay*	*Area*	*Delay*	*Area*	*Delay*	*Area*	*Delay*	*Area*	*Delay*	*Area*	*Delay*
addm4	9/8	**732**	35.6	823	38.5	908	36.3	978	40.8	947	40.1	1040	43.9
alu1	12/8	64	8.6	64	8.6	**53**	6.8	**53**	6.8	76	9	80	11
amd	14/24	**786**	30.1	996	37.8	**786**	30.1	988	37.3	812	31.7	946	35.5
apla	10/12	211	19.6	254	22.2	178	17.2	216	23.1	**161**	18.6	182	20.4
b12	15/9	155	15.8	187	20.2	**152**	15.5	189	23.9	185	20.2	228	22.7
co14	14/1	146	28.8	146	28.8	**136**	27.9	**136**	27.9	164	30.4	164	30.4
in0	15/11	**956**	38.2	1066	38.8	1065	43.4	1252	46.9	1004	41.5	1135	42.8
in2	19/10	**967**	38.9	1102	38.1	978	41.5	1600	49.6	1490	48.5	1833	55.5
in5	24/14	856	32.2	940	35.7	**790**	32.6	951	35.8	855	32.3	895	34.7
in7	26/10	389	21.9	405	22.7	**310**	25.6	351	24.7	491	30.6	618	30.5
m181	15/9	156	15.8	197	21	**153**	15.5	199	24.9	184	20.1	226	22.2
m2	8/16	419	26.2	429	28.4	444	29.3	451	29.2	**405**	24.4	425	26.9
m3	8/16	543	29.8	**535**	29	543	30.4	568	30.9	550	29.4	571	28.8
mlp4	8/8	**518**	27.4	576	28.3	520	30.2	643	32	552	31	636	33.2
mp2d	14/14	266	20.3	265	20.3	277	28.1	278	28.5	253	22.7	**238**	16.3
newtpla	15/5	**117**	18.5	**117**	18.5	122	18.5	172	21	119	18.3	169	20.7
rckl	32/7	352	49.8	352	49.8	354	46.2	354	46.2	**350**	49.7	**350**	49.7
t3	12/8	174	15.9	201	18.8	178	15.9	198	18.5	**173**	18.7	189	19.1
tms	8/16	**459**	30.3	473	29.4	483	27.2	517	28.6	466	25.2	485	28.1
vg2	25/8	**468**	22.5	517	22.8	488	26	590	23.2	544	23.5	640	22.3
vtx1	27/6	**330**	23.2	370	23.8	345	22.1	381	23.6	382	24.9	471	25.9
x6dn	39/5	**246**	25.3	250	20.1	704	29.4	729	29.3	697	28.6	724	28.5
x9dn	27/7	450	25.8	**400**	23.8	422	24.1	435	24.5	452	25.8	480	23.7
arbiter	256/129	**346**	16	**346**	16	351	16	351	16	347	15.9	347	15.9
cavlc	10/11	1419	50.3	1545	53	1443	52.5	1608	60.4	**1375**	48	1494	54.4
ctrl	7/26	**283**	31.7	311	34.2	297	32.3	333	35.7	285	23.8	331	27.6
dec	8/256	**2637**	152.3	**2637**	152.3	2642	171.7	2642	171.7	2639	124.4	2639	124.4
int2float	11/7	399	29.4	780	34.8	**397**	28.3	919	40.9	405	28.2	813	38.8
b03	33/34	**192**	17.5	241	14.9	**192**	17.5	399	17.9	205	17.5	401	17.5
b06	10/15	58	13.5	71	15.5	59	12.7	74	19.4	**56**	10.4	65	12.4
b08	29/25	**344**	24.3	430	24.2	**344**	24.3	429	24	459	26.3	684	35.4
b09	28/29	**132**	9.1	186	17.4	**132**	9.1	192	21.8	148	10.4	201	18.9
b10	27/23	474	24.1	601	27.2	**453**	23.2	599	25	**453**	23.2	654	28.7
b11	37/37	2942	64.3	4419	80.8	**2925**	64	3875	76.5	2933	64	4620	82.4
b13	62/63	**686**	18.4	1066	28.4	**686**	18.4	1156	40.2	700	18.4	1146	35.7

TABLE 7
Comparison of different cofactoring functions

	Best result in Area	Best result in Delay
$p = x_j$	50%	32%
$p = x_j \oplus x_k$	27%	24%
$p = x_j \wedge x_k$	22%	36%
ties	1%	8%

PSOP circuit minimization problem using Boolean relations pays significantly. In fact, the PSOP circuits synthesized with Boolean relations in the exact mode are smaller than the corresponding SOP forms in about 76% of our experiments, with an average gain in area of 27% (that reduces to 14% if the Boolean relation minimizer is run in the heuristic mode).

It is an open question how to characterize a-priori Boolean functions in order to predict what cofactors work better on a given input instance. For example, 2-input XOR gates yield good results on logic that contains equality tests and arithmetic patterns, since decomposition using XORs exposes such underlying structures that are common in benchmarks and are not understood by classic SOP minimization. Other patterns (like 3-input XORs) may be less present in the benchmarks used. Further progress on this question is future work: an option is to extract information on a given Boolean function by analyzing its discrete Fourier spectrum.

7 Conclusion and Future Work

In this paper we described a Boolean synthesis technique for PSOPs, a three-level architecture which includes as special cases EPSOPs and other logic forms that found attention in the previous literature. We took advantage of the fact that the structure of the implementation induces don't care conditions that can be exploited to reduce the problem of area minimization to Boolean relation minimization, with the guarantee that all valid realizations of the circuit are considered. We studied the general case of incompletely specified Boolean functions and characterized the remainder of the decomposition with the notion of k-neighbors. We also characterized the points that are in the remainder for important cases of the cofactoring function p, namely linear functions and AND functions

We report experiments showing significant gains in area with respect to the algebraic method, with better run times when we use the heuristic approach for the resolution of the Boolean relation, as summarized in Table 5. More precisely, we obtain an average gain in area of 28%, and an average gain in delay of 17%, with an improvement of synthesis time of about 30%.

Since the problem of finding the best p for a given function f is still open, future work includes a study of the choice of the variables used to define the decomposition, namely the variable x_i and the input variables of the cofactoring function p. We observe that, even though in the proposed model the projected functions and the remainder are represented in SOP form. our approach can be generalized to any other representation, both in the bounded and in the unbounded framework.

Moreover, it is interesting to investigate how to model in our synthesis formulation the simultaneous minimization of multi-output functions, instead of minimizing each single output independently, as done right now.

As a general methodological closing remark, this contribution is part of a systematic exploration of bounded multilevel logic synthesis. Its aim is to investigate architectures with a few levels of logic obtained by generalized Shannon decompositions, enhanced by a variety of Boolean operations that bring out key features of the underlying logic (like linearity by means of XORs). Once this optimization potential will be well understood, it may be embedded inside general tools that explore unbounded multi-level implementations: for instance one could design a PSOP-aware local restructuring procedure to be applied on a complex network. It is a fact that there is a challenging quality vs. scalability trade-off to establish, but this may direct with more insight the optimizations to match logic expressions made available by logic synthesis tools.

References

[1] "The EPFL Combinational Benchmark Suite." [Online]. Available: http://lsi.epfl.ch/benchmarks

[2] L. Amarù, P. Gaillardon, and G. D. Micheli, "Biconditional BDD: A Novel Canonical BDD for Logic Synthesis Targeting XOR-rich Circuits," in *Design Automation and Test in Europe (DATE)*, 2013.

[3] L. G. Amarù, P. Gaillardon, A. Chattopadhyay, and G. D. Micheli, "A Sound and Complete Axiomatization of Majority-n Logic," *IEEE Trans. Computers*, vol. 65, no. 9, pp. 2889–2895, 2016.

[4] L. G. Amarù, P. Gaillardon, and G. D. Micheli, "Majority-Inverter Graph: A New Paradigm for Logic Optimization," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 35, no. 5, pp. 806–819, 2016.

[5] D. Bañeres, J. Cortadella, and M. Kishinevsky, "A Recursive Paradigm to Solve Boolean Relations," *IEEE Transactions on Computers*, vol. 58, no. 4, pp. 512–527, 2009.

[6] A. Bernasconi, V. Ciriani, and R. Cordone, "EXOR Projected Sum of Products," in *IFIP/IEEE VLSI-SoC 2006 - International Conference on Very Large Scale Integration of System-on-Chip,*, 2006.

[7] A. Bernasconi, V. Ciriani, R. Drechsler, and T. Villa, "Logic Minimization and Testability of 2-SPP Networks," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 27, no. 7, pp. 1190–1202, 2008.

[8] A. Bernasconi, V. Ciriani, F. Luccio, and L. Pagli, "Fast Three-Level Logic Minimization Based on Autosymmetry," in *ACM/IEEE 39th Design Automation Conference (DAC)*, 2002, pp. 425–430.

[9] A. Bernasconi, V. Ciriani, G. Trucco, and T. Villa, "On Decomposing Boolean Functions via Extended Cofactoring," in *Design Automation and Test in Europe (DATE)*, 2009, pp. 1464–1469.

[10] ——, "Logic Synthesis by Signal-Driven Decomposition," in *Advanced Techniques in Logic Synthesis, Optimizations and Applications*, K. Gulati, Ed. Springer New York, 2011, pp. 9–29.

[11] ——, "Projected Don't Cares," in *Euromicro Conference on Digital Systems Design: Architectures, Methods and Tools (DSD)*, 2012, pp. 57–64.

[12] ——, "Minimization of P-Circuits using Boolean Relations," in *Design Automation and Test in Europe (DATE)*, 2013.

[13] A. Bernasconi, V. Ciriani, and R. Cordone, "On Projecting Sums of Products," in *11th Euromicro Conference on Digital Systems Design: Architectures, Methods and Tools*, 2008, pp. 787–794.

[14] ——, "The optimization of kEP-SOPs: Computational complexity, approximability and experiments," *ACM Trans. Design Autom. Electr. Syst.*, vol. 13, no. 2, 2008.

[15] A. Bernasconi, V. Ciriani, V. Liberali, G. Trucco, and T. Villa, "Synthesis of P-Circuits for Logic Restructuring," *Integration*, vol. 45, no. 3, pp. 282–293, 2012.

[16] A. Bernasconi, V. Ciriani, F. Luccio, and L. Pagli, "Three-Level Logic Minimization Based on Function Regularities," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 22, no. 8, pp. 1005–1016, 2003.

[17] ——, "Synthesis of autosymmetric functions in a new three-level form," *Theory Comput. Syst.*, vol. 42, no. 4, pp. 450–464, 2008.

[18] A. Bernasconi, V. Ciriani, G. Trucco, and T. Villa, "Minimization of EP-SOPs via Boolean relations," in *IFIP/IEEE VLSI-SoC 2013 - International Conference on Very Large Scale Integration of System-on-Chip,*, 2013, pp. 112–117.

[19] ——, "Using Flexibility in P-Circuits by Boolean Relations," *IEEE Trans. Computers*, vol. 64, no. 12, pp. 3605–3618, 2015.

[20] J. C. Bioch, "The Complexity of Modular Decomposition of Boolean Functions," *Discrete Applied Mathematics*, vol. 149, no. 1-3, pp. 1–13, 2005.

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

[22] R. Brayton and F. Somenzi, "Boolean Relations and the Incomplete Specification of Logic Networks," in *International Conference on Very Large Scale Integration - VLSI*, August 1989.

[23] R. K. Brayton and A. Mishchenko, "ABC: An Academic Industrial-Strength Verification Tool," in *Computer Aided Verification, 22nd International Conference, CAV 2010, Edinburgh, UK, July 15-19, 2010. Proceedings*, 2010, pp. 24–40.

[24] H. Chen, M. Janota, and J. Marques-Silva, "QBF-based Boolean Function Bi-Decomposition," in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE '12. San Jose, CA, USA: EDA Consortium, 2012, pp. 816–819. [Online]. Available: http://dl.acm.org/citation.cfm?id=2492708.2492911

[25] F. Corno, M. Reorda, and G. Squillero, "Rt-level itc'99 benchmarks and first atpg results," *Design Test of Computers, IEEE*, vol. 17, no. 3, Jul 2000.

[26] J. Cortadella, "Timing-Driven Logic Bi-Decomposition," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 22, no. 6, pp. 675–685, 2003.

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

[28] D. Debnath and Z. Vranesic, "A Fast Algorithm for OR-AND-OR Synthesis," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 22, no. 9, pp. 1166–1176, 2003.

[29] 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*, 1999, pp. 37–54.

[30] M. Fujita, Y. Matsunaga, and M. Ciesielski, "Multi-Level Logic Optimization," in *Logic Synthesis and Verification*, S. Hassoun and T. Sasao, Eds. Kluwer Academic Publishers, 2002, pp. 29–63.

[31] R. Ishikawa, T. Hirayama, G. Koda, and K. Shimizu, "New Three-Level Boolean Expression Based on EXOR Gates," *IEICE Transactions on Information and Systems*, no. 5, pp. 1214–1222, 2004.

[32] P. Kerntopf, "New Generalizations of Shannon Decomposition," in *Int. Workshop on Applications of Reed-Muller Expansion in Circuit Design*, 2001, pp. 109–118.

[33] C. Lang and B. Steinbach, "Bi-Decomposition of Function Sets in Multiple-Valued Logic for Circuit Design and Data Mining," *Artificial Intelligence Review*, vol. 20, no. 3, pp. 233–267, Dec 2003. [Online]. Available: https://doi.org/10.1023/B:AIRE.0000006608.31990.cd

[34] R.-R. Lee, J.-H. R. Jiang, and W.-L. Hung, "Bi-Decomposing Large Boolean Functions via Interpolation and Satisfiability Solving," in *Proceedings of the 45th Design Automation Conference, DAC 2008, Anaheim, California, USA, June 8-13, 2008*, 2008, pp. 636–641.

[35] P. McGeer, J. Sanghavi, R. Brayton, and A. Sangiovanni-Vincentelli, "ESPRESSO-SIGNATURE: A New Exact Minimizer for Logic Functions," *IEEE Transactions on VLSI*, vol. 1, no. 4, pp. 432–440, 1993.

[36] A. Mishchenko, B. Steinbach, and M. Perkowski, "An Algorithm for Bi-Decomposition of Logic Functions," in *ACM/IEEE 38th Design Automation Conference (DAC)*, 2001, pp. 103–108.

[37] A. Mishchenko, S. Chatterjee, and R. K. Brayton, "DAG-aware AIG rewriting a fresh look at combinational logic synthesis," in *Proceedings of the 43rd Design Automation Conference, DAC 2006, San Francisco, CA, USA, July 24-28, 2006*, 2006, pp. 532–535.

[38] T. Sasao, "On the Complexity of Three-Level Logic Circuits," in *Int. Workshop on Logic Synthesis*, 1989.

[39] M. Soeken, L. G. Amarù, P. Gaillardon, and G. D. Micheli, "Exact Synthesis of Majority-Inverter Graphs and Its Applications," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 36, no. 11, pp. 1842–1855, 2017.

[40] S. Yang, "Logic Synthesis and Optimization Benchmarks User Guide Version 3.0," Microelectronic Center, User Guide, 1991.

[41] Y. Yao, S. Huang, C. Wang, Y. Wu, and W. Qian, "Approximate Disjoint Bi-Decomposition and Its Application to Approximate Logic Synthesis," *2017 IEEE International Conference on Computer Design (ICCD)*, pp. 517–524, 2017.

Anna Bernasconi received the Laurea degree in Physics from the University of Pavia, Italy, in 1992, and the Ph.D. in Computer Science from the University of Pisa, Italy, in 1998. Her doctoral dissertation Mathematical techniques for the analysis of Boolean functions received the Doctoral Dissertation Thesis Award 1998 from the Italian Chapter of the European Association for Theoretical Computer Science (EATCS). She is currently an Associate Professor with the Department of Computer Science of the University of Pisa, where she teaches fundamental courses in the Computer Science Program. Her research interests include algorithms and complexity, Boolean function complexity, as well as combinational logic synthesis. She has authored or coauthored more than 50 research papers, published in international journals, conference proceedings, books and books chapters.

Valentina Ciriani received the Laurea degree and the Ph.D. degree in Computer Science from the University of Pisa, Italy, in 1998 and 2003, respectively. In 2003 and 2004, she was with the Department Computer Science at University Pisa, Italy as a Ph.D. fellow. From January 2005 to February 2015, she was an assistant professor with the Department Information Technologies and the Department of Computer Science, Università degli Studi di Milano, Italy. She is currently an Associate Professor in Computer Science with the Department of Computer Science, Università degli Studi di Milano. Her research interests include algorithms and data structures, as well as combinational logic synthesis, VLSI design of low power circuits and testing of Boolean circuits. She has authored or coauthored more than 80 research papers, published in international journals, conference proceedings, and books chapters.

Gabriella Trucco graduated in Computer Science in 2002 (Università degli Studi di Milano); in 2005 she received the Ph.D. in Computer Science (Università degli Studi di Milano). From 2006 she is assistant professor in Computer Science at Dipartimento di Informatica, Università degli Studi di Milano. Her research interests include combinational logic synthesis, VLSI design of low power circuits and testing of Boolean circuits, as well as algorithms and data structures for bioinformatics (genome analysis, phylogenetic analysis, and haplotype inference). She has coauthored more than 30 research papers, published in international journals, conference proceedings, and books chapters.

Tiziano Villa received a Laurea degree in Mathematics from the University of Milano, took the Part III of Mathematical Tripos at the University of Cambridge, completed a M.S. in CS at U.C. Berkeley, and was awarded a Ph.D. in EECS in 1995 by U.C. Berkeley. In 1997 he joined as a Research Scientist the PARADES Labs, Rome, Italy. In 2002 he became an Associate Professor at Università di Udine, Italy. Since 2006 he is a Professor with the Department of Computer Science (DI), Università di Verona, Italy. His research interests are in formal methods for electronic design automation, including logic synthesis, formal verification, automata theory and models of computation, discrete-event dynamic systems, supervisory control, cyber-physical and embedded systems. He co-authored the books "Synthesis of FSMs: Functional Optimization" (Kluwer/Springer, 1997, reprint 2010), "Synthesis of FSMs: Logic Optimization" (Kluwer/Springer, 1997, reprint 2012), "The Unknown Component Problem: Theory and Applications" (Springer, 2012), and co-edited the book "Coordination Control of Distributed Systems", Springer, 2015.