IEEE Transactions on computers, epub ahead of print 04 March 2015

D.O.I: 10.1109/TC.2015.2409849

Using Flexibility in P-Circuits by Boolean Relations

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

Abstract—In this paper we study the problem of characterizing and exploiting the complete flexibility of a special logic architecture, called P-circuits, which realize a Boolean function by projecting it onto overlapping subsets given by a generalized Shannon decomposition. P-circuits are used to restructure logic by pushing some signals towards the outputs. The algorithms proposed so far for exploiting the structural flexibility of P-circuits do not guarantee to find the best implementation, because they cast the problem as the *minimization of an incompletely specified function*. Instead, here we show that to explore all solutions we must set up the problem as the *minimization of a Boolean relation*, because there are don't care conditions that cannot be expressed by single cubes. Finally we report the results obtained using a minimizer of Boolean relations, which improve in a major way with respect to the previously literature.

Index Terms—Logic synthesis, Boolean decomposition, Boolean relations

1 Introduction

Logic optimization of digital circuits explores different realizations of a logic circuit to improve design parameters like area, speed, power consumption, and others. This goal is achieved exploiting both the flexibility allowed by the specification (external don't care conditions), and the internal or structural flexibility due to the structure of the implementation. Therefore, given a chosen logic architecture, the goal of logic synthesis is from one side to characterize completely the implementations consistent with the specification and the architecture, and from other side to explore all of them choosing the best with respect to a given cost function.

For instance, in the model of multi-level synthesis *á la SIS* [13], based on Boolean networks whose nodes are single-output PLAs, two sources of flexibility were defined to characterize the complete flexibility: satisfiability don't cares, SDCs, to model limited controllability, and observability don't cares, ODCs, to model limited observability. Then, algorithms were proposed to compute part or all of them, and the flexibility was exploited running two-level minimization on the single nodes of the Boolean network.

In this paper we revisit the problem of characterizing and exploiting the complete flexibility of a special logic architecture, called P-circuits. They are extended forms of Shannon cofactoring, investigated in [3], [4], [6], where the expansion is with respect to an orthogonal basis $\overline{x}_i \oplus p$ (i.e., $x_i = p$), and $x_i \oplus p$ (i.e., $x_i \neq p$), where p is a function defined over all variables except for a critical variable x_i. P-circuits were introduced as a specialized form of decomposition with respect to specific critical signals that should be pushed closer to the outputs, e.g., signals with the highest switching activity (to decrease power consumption), or late-arriving ones (to decrease worst-case delay).

More in detail, let $f_{x_i=p}$ and $f_{x_i\neq p}$ be the projections of a function f onto $x_i = p$ and $x_i \neq p$, and let $I = f_{x_i=p} \cap f_{x_i\neq p}$ be the points common to the two projections. A completely defined function f can be decomposed giving three Boolean functions combined by a disjunction: $f = (\overline{x}_i \oplus p)f^{=} + (x_i \oplus p)f^{\neq} + f^I$, where $f^{=} \subseteq f_{x_i=p}$, $f^{\neq} \subseteq f_{x_i\neq p}$, and $f^I \subseteq I$. The circuits synthesized according to this structure are called P-circuits when the blocks $f^{=}$, $f^{\neq}$ and f^I are realized by sums-of-products (as shown in Figure 1). Shannon decomposition is the special case when $p = 0$, $f^{=} = f_{x_i=p}$, $f^{\neq} = f_{x_i\neq p}$, and $f^I = \emptyset$.

The structural flexibility arises in P-circuits as follows: if a point q of f is covered in one set ($f^{=}$, $f^{\neq}$, or f^I), q may be considered as a don't care in the other sets containing it. This can be seen as a special case of the following disjunctive paradigm: when the outputs of some Boolean functions are combined by a disjunction, if a single function produces the output value 1 for some inputs, we do not care about the output values produced by the remaining functions for such inputs; therefore under these inputs the representation of the other functions can be modified and optimized according to a chosen cost function [14].

The computation of the flexibility of P-circuits has been already studied in the literature [3], [4], [6], but the algorithms proposed so far for its exploitation may fail to find the best implementation, because they cast the optimization problem as the *minimization of an incompletely specified function*. Instead, in this paper we show that to explore all solutions we must set up the problem as the *minimization of a Boolean relation*, because there are combinations 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. {valentina.ciriani, gabriella.trucco}@unimi.it*
- *T. Villa is with the Department of Computer Science, Università degli Studi di Verona, Italy. tiziano.villa@univr.it*

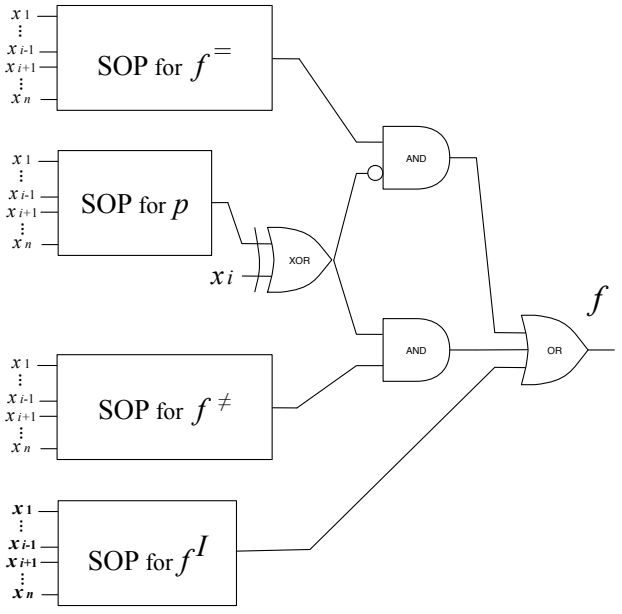


Fig. 1. A P-circuit for the function f decomposed with respect to the critical variable x_i and the function p.

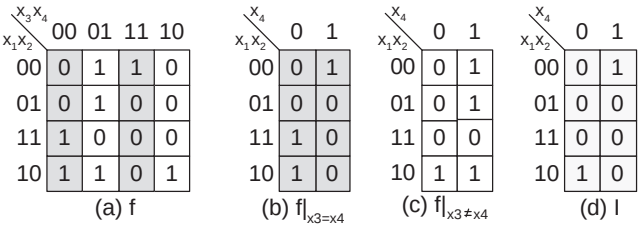


Fig. 2. A function f (a), the corresponding projected functions $f|_{x_3=x_4}$ and $f|_{x_3\neq x_4}$ (b) and (c), and their intersection I (d).

don't care conditions that cannot be expressed by single cubes. Therefore here we model the optimal P-circuit decomposition problem by means of Boolean relations, and show that it is a complete characterization of the structural flexibility of P-circuits. Finally we report the results obtained using a minimizer of Boolean relations, which are a major improvement with respect to the previously published experiments.

This paper is an extended version of the conference paper presented in [5] and is organized as follows. Section 2 introduces P-circuits for completely and incompletely specified functions. Section 3 describes how to minimize P-circuits using Boolean relations. Experiments on a set of benchmarks are reported in Section 4, and Section 5 concludes the paper.

2 P-CIRCUITS

A P-circuit is a network where the dependence on a given variable x_i (e.g., the variable with more switching activity or with higher delay) is projected away from the rest of the circuit.

In this section we first briefly review the P-circuits based on SOP minimization [3], [4], [6]. We then give a new formulation of the associated minimization problem, both for completely and incompletely specified functions, better suited to be formalized via Boolean relations.

We first give some preliminary definitions. A *completely specified Boolean function* f is a function $f : \{0,1\}^n \rightarrow \{0,1\}$. A completely specified Boolean function can also be interpreted as the set of points $x \in \{0,1\}^n$ such that $f(x) = 1$. An *incompletely specified Boolean function* is a function $f : \{0,1\}^n \rightarrow \{0,1,-\}$, where $-$ is called the don't care value of the function. An incompletely specified function can be described by three sets of points: the *on-set*, the *off-set* and the *don't care set*, which characterize the points in $\{0,1\}^n$ with images 0, 1, and $-$, respectively. Given the Boolean space $\{0,1\}^n$ described by the set $\{x_1,\ldots,x_n\}$ of n binary variables, a *literal* is a variable or its complement; a *cube* is conjunction (or product) of a set of literals, and a *minterm* is a cube when it represents only one point, i.e., when it is a conjunction of n literals. Finally, a *multiple-output Boolean function* f is a function $f : \{0,1\}^n \rightarrow \{0,1,-\}^m$; it can be considered also as a vector of Boolean functions $\{f_1, f_2, \ldots, f_m\}$.

2.1 Completely Specified Functions

Let f be a completely specified Boolean function depending on the set $\{x_1,\ldots,x_n\}$ of n binary variables. The classical Shannon decomposition $f = \overline{x}_i f|_{\overline{x}_i} + x_i f|_{x_i}$, and the more general EXOR-based decomposition [7] [11]

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

(where p is a function non-depending on x_i)[1] are suitable for keeping x_i disjoint from the rest of the circuit, but are not oriented to area minimization. In fact, $f|_{x_i=p}$, and $f|_{x_i\neq p}$ do not depend on the variable x_i, but the cubes of f intersecting both subsets $x_i = p$ and $x_i \neq p$ may be split into two smaller subcubes when they are projected onto $f|_{x_i=p}$, and $f|_{x_i\neq p}$, respectively. For example, consider the function f in Figure 2 (a), the variable $x_i = x_3$ and the simple function $p(x_1, x_2, x_4) = x_4$. The projection of f with respect to $x_3 = x_4$ and $x_3 \neq x_4$ are depicted in Figures 2 (b) and (c), respectively. We note that the cube $\overline{x}_1\overline{x}_2x_4$ of f is split into two separate minterms $\overline{x}_1\overline{x}_2x_4$ and $\overline{x}_1\overline{x}_2x_4$ in the two projected functions $f|_{x_3=x_4}$ and $f|_{x_3\neq x_4}$ (Figures 2 (b) and (c)).

To overcome this problem, in P-circuits synthesis, we keep unprojected some of the points of the original function. For this purpose, let $I = f|_{x_i=p} \cap f|_{x_i\neq p}$ be the intersection of the two cofactors $f|_{x_i=p}$ and $f|_{x_i\neq p}$. Note that the intersection I contains the cubes whose products do not contain x_i and that cross the two sets. Therefore, in order to overcome the splitting of these crossing cubes, we could keep I unprojected, and project only the minterms in

1. Note that the Shannon decomposition is a particular EXOR-based decomposition where p=0.

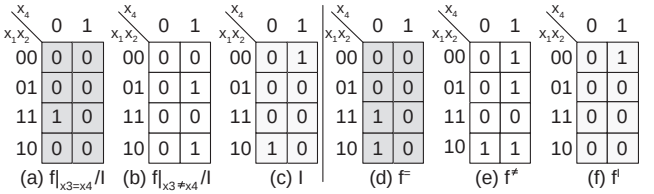


Fig. 3. Running example.

$f|_{x_i=p} \setminus I$ and $f|_{x_i \neq p} \setminus I$, obtaining the expression

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

Note that p, $f|_{x_i=p} \setminus I$, $f|_{x_i \neq p} \setminus I$ and I do not depend on x_i. Following the previous example, see Figures 3 (a), (b), and (c).

However, the points that are in I could be exploited to form bigger cubes in the projected sets. For example, the point 001 of I would be useful for forming the cube $\overline{x}_1 x_4$ in $f|_{x_3 \neq x_4} \setminus I$, see Figure 3 (b), indeed such cube was originally in $f|_{x_3 \neq x_4}$, see Figure 2 (c). Therefore, if a point is in I and is useful for a better minimization of the projected parts, it can be kept both in the projection and in the intersection (see, for example, the point 001 in Figures 3 (e) and (f)). Moreover, if a point is covered in both the projected sets, it is not necessary to cover it in the intersection. In our example, the point 100 of I is used for forming bigger cubes in both the projections (Figures 3 (d) and (e)), thus it can be removed from the intersection (see Figure 3 (f)).

In the following, we indicate a SOP circuit implementing a Boolean function f with $S(f)$, and we denote optimal circuits (in the sense of two-level minimization) with a star, as $S^*(f)$.

From the previous observations, we can infer that the projected sub-circuits should cover at least $f|_{x_i=p} \setminus I$ and $f|_{x_i \neq p} \setminus I$, and must be contained in $f|_{x_i=p}$ and $f|_{x_i \neq p}$, respectively. Moreover, the part of the circuit that is not projected should be contained in the intersection I.

In summary, we rephrase the definition of a P-circuit given in [6] as follows.

Definition 1: A *P-circuit* of a completely specified function f is the circuit $P(f)$ denoted by the expression:

$$P(f) = (\overline{x}_i \oplus S(p))\, S(f^{=}) + (x_i \oplus S(p))\, S(f^{\neq}) + S(f^I)$$

where

1) $(f|_{x_i=p} \setminus I) \subseteq f^{=} \subseteq f|_{x_i=p}$
2) $(f|_{x_i \neq p} \setminus I) \subseteq f^{\neq} \subseteq f|_{x_i \neq p}$
3) $\emptyset \subseteq f^I \subseteq I$
4) $P(f) = f\,.$

Therefore, the synthesis idea of P-circuits is to construct a network for f by appropriately choosing the sets $f^{=}$, $f^{\neq}$, and f^I as building blocks (as shown in Figure 1).

The cofactors and the intersection can be synthesized in any framework of logic minimization. Here we focus on the standard SOP synthesis, since SOP forms are widely used and also because we exploit the Boolean relation minimizer BREL [2] that is built for classical SOP synthesis. As far as we know, no Boolean relation minimizer is available for other logic forms. Thus, we represent $f^{=}$, $f^{\neq}$, and f^I as sums of products.

For example, the expression

$$P(f) = (x_3 \oplus x_4)(x_1\overline{x}_2 + \overline{x}_1 x_4) + (\overline{x}_3 \oplus x_4)(x_1\overline{x}_4) + \overline{x}_1\overline{x}_2 x_4$$

is a P-circuit for the function f defined in Figure 2 (a). $P(f)$ is derived from the functions $f^{=}$, $f^{\neq}$, and f^I in Figures 3 (d), (e), and (f).

An *optimal P-circuit* $P^*(f)$ for the function f is a P-circuit with minimum cost that can be synthesized for f. The P-circuit described in the previous example is an optimal P-circuit with respect to the number of literals in the SOP forms.

2.2 Incompletely Specified Functions

Consider now an incompletely specified Boolean function $f = \{f^{on}, f^{dc}\}$. 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. Let I be the intersection of the projections of f onto the two sets $x_i = p$ and $x_i \neq p$:

$$I = (f^{on}|_{x_i=p} \cup f^{dc}|_{x_i=p}) \cap (f^{on}|_{x_i \neq p} \cup f^{dc}|_{x_i \neq p}).$$

We can generalize the definition of P-circuit in the following way.

Definition 2: A *P-circuit* of an incompletely specified function $f = \{f^{on}, f^{dc}\}$ is the circuit $P(f)$ denoted by the expression:

$$P(f) = (\overline{x}_i \oplus S(p))\, S(f^{=}) + (x_i \oplus S(p))\, S(f^{\neq}) + S(f^I)$$

where

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

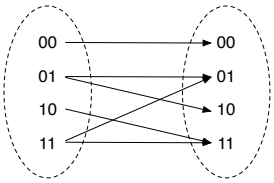
3 Minimization of P-circuits using Boolean Relations

The problem of minimizing a Boolean function in P-circuit form can be characterized completely by using Boolean relations, as explained in the following subsections, first for completely specified and then for incompletely specified single-output functions. We will then discuss some optimization issues and how to deal with multi-output functions.

3.1 Boolean Relations

In this subsection we recall the theory of Boolean relations, and give a brief overview of the recursive paradigm to solve Boolean relations presented in [10], [1], [2].

The concept of Boolean relations was introduced in [8] as a more general scheme for the incomplete specification of logic networks. In fact, the conditions under which a



x_1x_2	y_1y_2
00	$\{00\}$
01	$\{01, 10\}$
10	$\{11\}$
11	$\{01, 11\}$

Fig. 4. An example of a Boolean relation in graphical (left) and tabular (right) form.

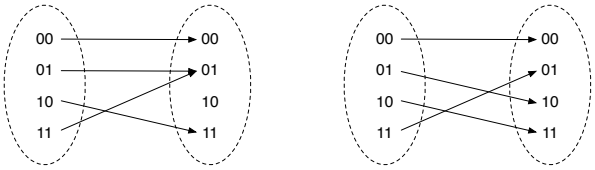


Fig. 5. Two Boolean functions compatible with the relation of Figure 4.

multi-output logic network can be simplified cannot always be completely represented using don't cares.

Definition 3 ([8]): A *Boolean relation* is a one-to-many multi-output Boolean mapping $\mathcal{R} : \{0,1\}^n \rightarrow \{0,1\}^m$. $\{0,1\}^n$ 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. Also, notice that $\mathcal{R}$ is a subset of the Cartesian product $\{0,1\}^n \times \{0,1\}^m$.

A relation $\mathcal{R}$ is *well-defined* if for all $x \in \{0,1\}^n$, there is $y \in \{0,1\}^m$ such that $(x, y) \in \mathcal{R}$. A well defined Boolean relation is *functional* if every input x is associated with one and only one output y.

Figure 4 shows an example of a Boolean relation with two input and two output variables, in tabular and in graphical representation. This relation is not a functional one, indeed the inputs 01 and 11 are associated to two outputs each: $\{01, 10\}$ and $\{01, 11\}$ respectively. Also observe that while the outputs associated to 11 could be expressed by the single output $\{-1\}$ introducing a don't care into the range of output variables, the two outputs related to 01 cannot be expressed using don't cares. As even this simple example shows, Boolean relations capture more flexibility than incompletely specified Boolean functions.

To any relation $\mathcal{R}$ we can associate a set of *compatible* multi-output Boolean functions, i.e. the set of all functions f such that, for all input $x \in \{0,1\}^n$, $f(x)$ is contained in the set $\mathcal{R}(x)$ of the outputs related to x. In this case, we write $f \subseteq \mathcal{R}$.

Definition 4 ([2]): The set of multi-output Boolean functions compatible with a Boolean relation $\mathcal{R} : \{0,1\}^n \rightarrow \{0,1\}^m$, is defined as $\mathcal{F}(\mathcal{R}) = \{f : \{0,1\}^n \rightarrow \{0,1\}^m \mid f \subseteq \mathcal{R} \text{ and } f \text{ is a function}\}$.

Two Boolean functions compatible with the Boolean relation of Figure 4 are shown in Figure 5.

The problem of the optimal implementation of a Boolean relation $\mathcal{R}$ is the one of selecting, among the possible functions compatible with $\mathcal{R}$, one of minimum cost according to a given metric. More precisely

Definition 5 ([2]): 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 $f' \in \mathcal{F}(\mathcal{R})$, $c(f) \leq c(f')$.

Many problems in logic synthesis can be formalized using Boolean relations, and Boolean decomposition is one of them. For instance, consider the decomposition of a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ using a function $g : \{0,1\}^m \rightarrow \{0,1\}$:

$$f(x_1, \ldots, x_n) = g(f_1((x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)) .$$

The relation $\mathcal{R}$ that represents all possible decompositions of f using g can be defined as follows: for all $x \in \{0,1\}^n$, $\mathcal{R}(x) = \{y \in \{0,1\}^m \mid g(y) = f(x)\}$. In other words, each input $x \in \{0,1\}^n$ where $f(x) = 1$, will be associated to the set of outputs $y \in \{0,1\}^m$, where g takes the value 1; each input $x \in \{0,1\}^n$ where $f(x) = 0$, will be associated to the set of outputs $y \in \{0,1\}^m$, where g is 0. A solver of Boolean relations will explore the set of possible solutions of the relation $\mathcal{R}$ and will return an optimal solution according to the minimization objective.

Several exact and heuristic algorithms have been proposed for solving Boolean relations (see [2] for an overview of these methods, and for bibliographic references). For our particular minimization problem, we used the recursive algorithm proposed in [2] because of its ability in finding better solutions in shorter runtime than the previously known methods. In fact, this recursive solver is very efficient and able to explore a wide space of solutions. Moreover it can be used both in an exact and heuristic approximate mode and its cost function can be tuned for different parameters related to the area or the delay in computing the relation.

The idea of the recursive approach is that of reducing the problem of solving a Boolean relation to a sequence of Boolean minimization problems applied to multi-output incompletely specified functions. Each of these functions is an overapproximation of the original Boolean relation, obtained by expanding the inputs whose outputs cannot be expressed using don't cares to cover more points of the output set. For example, the relation of Figure 4 can be overapproximated by the function

x_1x_2	y_1y_2
00	00
01	$--$
10	11
11	-1

obtained by expanding the set $\mathcal{R}(01) = \{01, 10\}$ of the outputs related to 01 to $\{00, 01, 10, 11\} = --$.

The function that overapproximates the relation is then minimized using a standard minimization method. If the minimized function is compatible with the relation, the algorithm reports the results. Otherwise, if the function contains conflicts, the algorithm continues with a successive refinement of the original relation into two smaller relations containing fewer outputs that cannot be expressed using don't cares. These smaller relations, obtained by selecting an input where there is an incompatibility and partitioning its outputs, are solved recursively, and the best compatible solution is selected out of the explored solutions. For instance, the minimal SOP cover of the function f that overapproximates the relation $\mathcal{R}$ of Figure 4 would be (in PLA form):

$$\begin{array}{ll} 0- & 00 \\ 1- & 11 \end{array}$$

corresponding to a function not compatible with $\mathcal{R}$, as $f(01) = 00$ but $00 \notin \mathcal{R}(01)$. Thus, the algorithm splits $\mathcal{R}$ into the two smaller relations

x_1x_2	y_1y_2
00	00
01	01
10	11
11	-1

x_1x_2	y_1y_2
00	00
01	10
10	11
11	-1

both corresponding to incompletely specified Boolean functions. Once the two functions have been minimized, the best cover is selected and reported as solution of the original relation $\mathcal{R}$.

3.2 Completely Specified Functions

Let f be a completely specified Boolean function depending on n variables, x_i one input variable, and p a function depending on all input variables of f but x_i. Consider the two cofactors $f|_{x_i=p}$ and $f|_{x_i\neq p}$, obtained by projecting f onto the sets $x_i = p$ and $x_i \neq p$, and their intersection $I = f|_{x_i=p} \cap f|_{x_i\neq p}$.

As recalled in Section 2, in order to minimize f in P-circuit form, we must find a refinement of the starting partition of the minterms of f into the three disjoint sets $f|_{x_i=p} \setminus I$, $f|_{x_i\neq p} \setminus I$ and I. This refinement, which leads to the three new sets $f^{=}$, $f^{\neq}$, and f^I, consists in *(i)* adding back some points of I to $f|_{x_i=p} \setminus I$ or $f|_{x_i\neq p} \setminus I$, precisely the points that may help in obtaining larger cubes in their SOP representation, and *(ii)* subtracting from I some of the points in the intersection of the refined cofactors (i.e., covered by both cofactors), in order to find a more compact SOP representation for the intersection set.

The final P-circuit for f is then given by three minimal SOPs representing $f^{=}$, $f^{\neq}$, and f^I, appropriately connected to a minimal SOP for p. Therefore, the problem is finding the sets $(f^{=}, f^{\neq}, f^I)$ that lead to a P-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 $\mathcal{R}$ whose set of compatible functions $\mathcal{F}(\mathcal{R})$ corresponds exactly to the set of all possible P-circuits for f, so that an optimal solution of $\mathcal{R}$ is an optimal P-circuit $P^*(f)$ for f.

Let $\mathcal{R}_f : \{0,1\}^{n-1} \rightarrow \{0,1\}^3$ be a Boolean relation, whose input set is the space spanned by the variables $x_1 \ldots x_{i-1}x_{i+1} \ldots x_n$, while the output set describes all possible tuples of functions $f^{=}$, $f^{\neq}$, f^I defining a P-circuit for f.

From Definition 1, we can observe that the sets $f|_{x_i=p} \setminus I$ and $f|_{x_i\neq p} \setminus I$ must be always contained in $f^{=}$ and $f^{\neq}$, respectively. Thus, all points in $f|_{x_i=p} \setminus I$ are associated to the output 100, and all points in $f|_{x_i\neq p} \setminus I$ are associated to the output 010 of $\mathcal{R}_f$.

Moreover, we can also note that the possible P-circuit implementations of f differ in the distribution of the points of I among the three sets $f^{=}$, $f^{\neq}$, f^I. In particular, each minterm in I can be associated to one of the following outputs of $\mathcal{R}_f$:

- 001, if it belongs only to f^I, that is, it has been kept in I, and not added to $f^{=}$ and $f^{\neq}$;
- 101, if it has been added to $f^{=}$ and kept in I;
- 011, if it has been added to $f^{\neq}$ and kept in I;
- 111 if it has been added to $f^{=}$ and $f^{\neq}$ and kept in I;
- 110 if it has been added to $f^{=}$ and $f^{\neq}$ and removed from I.

The possible outputs corresponding to the points in I can be represented in a compact way using don't cares: $--1$ and $11-$. Indeed, if a point is covered in the intersection set, then it becomes a don't care point in the two projection sets $x_i = p$ and $x_i \neq p$, while if it is covered in both projection sets, it becomes a don't care for I.

Since all points of I must be covered either in f^I, or both in $f^{=}$ and $f^{\neq}$, all other possible output choices for I would not define a P-circuit for f.

The relation $\mathcal{R}_f$ can therefore be defined as follows:

$x_1 \ldots x_{i-1}x_{i+1} \ldots x_n$	$\mathcal{R}_f$
points in $f\|_{x_i=p} \setminus I$	$\{100\}$
points in $f\|_{x_i\neq p} \setminus I$	$\{010\}$
points in I	$\{--1, 11-\}$
all other points	$\{000\}$

Observe that $\mathcal{R}_f$ is truly a relation, as it is not functional on the points of I.

With this formalism, we can rephrase our minimization problem as finding an optimal implementation of $\mathcal{R}_f$, that is, as selecting among all possible three-output functions compatible with $\mathcal{R}_f$, each corresponding to a tuple $f^{=}$, $f^{\neq}$, and f^I, the one whose overall SOP representation is minimal. A formal proof of this statement is given in Section 3.4.

The Algorithm in Figure 6 shows the overall synthesis method via Boolean relations. In the algorithm, after the construction of the Boolean relation R_f, the procedure $OPTRelation(R_f)$ is called. $OPTRelation(R_f)$ finds a minimal solution of the Boolean relation R_f with respect

P-circuit synthesis of completely specified functions via Boolean relations

INPUT: Completely specified function f, p, and a variable x_i.
OUTPUT: A P-circuit network representation for f w.r.t. p and x_i.
NOTATION: f is the on-set of the function f.
The relation R_f is a set of couples $\langle input\_cube, \{output\_cubes\}\rangle$.

$R_f = \emptyset$;
$I = f|_{x_i=p} \cap f|_{x_i \neq p}$;
for each cube $c \in I$
  $R_f = R_f \cup \langle c, \{--1, 11-\}\rangle$;
$f^{(=)} = f|_{x_i=p} \setminus I$;
for each cube $c \in f^{(=)}$
  $R_f = R_f \cup \langle c, \{100\}\rangle$;
$f^{(\neq)} = f|_{x_i \neq p} \setminus I$;
for each cube $c \in f^{(\neq)}$
  $R_f = R_f \cup \langle c, \{010\}\rangle$;
$f_{off} = \{0,1\}^{n-1} \setminus (f|_{x_i=p} \cup f|_{x_i \neq p})$;
for each cube $c \in f_{off}$
  $R_f = R_f \cup \langle c, \{000\}\rangle$;
$(f^{=}_{opt}, f^{\neq}_{opt}, f^{I}_{opt}) = OPTRelation(R_f)$;
return $(\overline{x}_i \oplus p) f^{=}_{opt} + (x_i \oplus p) f^{\neq}_{opt} + f^{I}_{opt}$;

Fig. 6. Algorithm for the synthesis of P-circuits of completely specified functions.

to the number of literals in the SOP forms for $f_{=}$, $f_{\neq}$, and f_I. The solutions $(f^{=}_{opt}, f^{\neq}_{opt}, f^{I}_{opt})$ are given in their minimal SOP form. Thus the Algorithm returns the minimal P-circuit solution.

Example 1: Consider the function in Figure 2, and its projections onto the sets $x_3 = x_4$ and $x_3 \neq x_4$. The intersection between the two cofactors $f|_{x_3=x_4} = \{001, 110, 100\}$ and $f|_{x_3 \neq x_4} = \{001, 011, 100, 101\}$ contains two points $\{001, 100\}$. Thus we have

$$I = \{001, 100\}$$
$$f|_{x_3=x_4} \setminus I = \{110\}$$
$$f|_{x_3 \neq x_4} \setminus I = \{011, 101\}$$

The corresponding relation is

$x_1x_2x_4$	$\mathcal{R}_f$
001	$\{11-, --1\}$
011	$\{010\}$
100	$\{11-, --1\}$
101	$\{010\}$
110	$\{100\}$

where all missing points are associated to the output 000. It can be easily verified that among all functions compatible with $\mathcal{R}_f$, the one whose overall SOP representation is minimal (with respect to the number of literals) is

$x_1x_2x_4$	$f^{=} f^{\neq} f^{I}$
001	0 1 1
011	0 1 0
100	1 1 0
101	0 1 0
110	1 0 0

that defines precisely the sets $f^{=}$, $f^{\neq}$ and f^{I} shown in Figures 3 (d), (e), and (f). Thus, the Algorithm in Figure 6 outputs the P-citcuit:

$$(x_3 \oplus x_4)(x_1\overline{x}_2 + \overline{x}_1x_4) + (\overline{x}_3 \oplus x_4)(x_1\overline{x}_4) + \overline{x}_1\overline{x}_2x_4\,.$$

$f\|_{xi=p}$ \ $f\|_{xi\neq p}$	0	1	-
0	000	010	0-0
1	100	--1 11-	--1 1--
-	-00	--1 -1-	---

Fig. 7. Outputs for the relation $\mathcal{R}_f$ when f is an incompletely specified function.

P-circuit synthesis of incompletely specified functions via Boolean relations

INPUT: Incompletely specified function f, p, and a variable x_i.
OUTPUT: A P-circuit network representation for f w.r.t. p and x_i.
NOTATION: let $f = (f_{on}, f_{dc})$, i.e., f_{on} is the on-set of f, and f_{dc} is the DC-set of f.
The relation R_f is a set of couples $\langle input\_cube, \{output\_cubes\}\rangle$.

$R_f = \emptyset$;
$I_{on,on} = f_{on}|_{x_i=p} \cap f_{on}|_{x_i \neq p}$;
for each cube $c \in I_{on,on}$
  $R_f = R_f \cup \langle c, \{--1, 11-\}\rangle$;
$I_{dc,dc} = f_{dc}|_{x_i=p} \cap f_{dc}|_{x_i \neq p}$;
for each cube $c \in I_{dc,dc}$
  $R_f = R_f \cup \langle c, \{---\}\rangle$;
$I_{on,dc} = f_{on}|_{x_i=p} \cap f_{dc}|_{x_i \neq p}$;
for each cube $c \in I_{on,dc}$
  $R_f = R_f \cup \langle c, \{--1, 1--\}\rangle$;
$I_{dc,on} = f_{dc}|_{x_i=p} \cap f_{on}|_{x_i \neq p}$;
for each cube $c \in I_{dc,on}$
  $R_f = R_f \cup \langle c, \{--1, -1-\}\rangle$;
$f^{(=)}_{on} = f_{on}|_{x_i=p} \setminus (I_{on,on} \cup I_{on,dc})$;
for each cube $c \in f^{(=)}_{on}$
  $R_f = R_f \cup \langle c, \{100\}\rangle$;
$f^{(\neq)}_{on} = f_{on}|_{x_i \neq p} \setminus (I_{on,on} \cup I_{dc,on})$;
for each cube $c \in f^{(\neq)}_{on}$
  $R_f = R_f \cup \langle c, \{010\}\rangle$;
$f^{(=)}_{dc} = f_{dc}|_{x_i=p} \setminus (I_{dc,dc} \cup I_{dc,on})$;
for each cube $c \in f^{(=)}_{dc}$
  $R_f = R_f \cup \langle c, \{-00\}\rangle$;
$f^{(\neq)}_{dc} = f_{dc}|_{x_i \neq p} \setminus (I_{dc,dc} \cup I_{on,dc})$;
for each cube $c \in f^{(\neq)}_{dc}$
  $R_f = R_f \cup \langle c, \{0-0\}\rangle$;
$f_{off} = \{0,1\}^{n-1} \setminus (f_{on}|_{x_i=p} \cup f_{on}|_{x_i \neq p} \cup f_{dc}|_{x_i=p} \cup f_{dc}|_{x_i \neq p})$;
for each cube $c \in f_{off}$
  $R_f = R_f \cup \langle c, \{000\}\rangle$;
$(f^{=}_{opt}, f^{\neq}_{opt}, f^{I}_{opt}) = OPTRelation(R_f)$;
return $(\overline{x}_i \oplus p) f^{=}_{opt} + (x_i \oplus p) f^{\neq}_{opt} + f^{I}_{opt}$;

Fig. 8. Algorithm for the synthesis of P-circuits of incompletely specified functions.

3.3 Incompletely Specified Functions

Let us now consider the P-circuit minimization of incompletely specified functions. The construction of $\mathcal{R}_f$ must consider more cases. In fact, the possible P-circuit implementations depend not only on the distribution of the points of I among the three sets $f^{=}$, $f^{\neq}$, f^{I}, but also on how the don't care points of the two cofactors are used to build larger cubes in the two projection spaces $x_i = p$ and

$x_i \neq p$, and possibly in the intersection I.

To correctly define the relation $\mathcal{R}_f$ we must partition the input set $\{0,1\}^{n-1}$ into nine subsets, each corresponding to a combination of the values $\{0,1,-\}$ assumed by the two cofactors $f|_{x_i=p}$ and $f|_{x_i\neq p}$, as shown in Figure 7. Given $x \in \{0,1\}^{n-1}$, we have the following cases:

1) $[f|_{x_i=p}(x) = 0,\ f|_{x_i\neq p}(x) = 0]$
   x is a point of both off-sets, thus $\mathcal{R}_f(x) = \{000\}$.
2) $[f|_{x_i=p}(x) = 0,\ f|_{x_i\neq p}(x) = 1]$
   Definition 2 immediately implies that $\mathcal{R}_f(x) = \{010\}$.
3) $[f|_{x_i=p}(x) = 1,\ f|_{x_i\neq p}(x) = 0]$
   As before, Definition 2 implies that $\mathcal{R}_f(x) = \{100\}$.
4) $[f|_{x_i=p}(x) = -,\ f|_{x_i\neq p}(x) = 0]$
   The don't cares of $f|_{x_i=p}$, corresponding to zeros of the other cofactor, can be exploited to get a smaller SOP form for $f^{=}$, thus we pose $\mathcal{R}_f(x) = \{-00\}$.
5) $[f|_{x_i=p}(x) = 0,\ f|_{x_i\neq p}(x) = -]$
   As before, the don't cares of $f|_{x_i\neq p}$, corresponding to zeros of $f|_{x_i=p}$, can be exploited to get a smaller SOP form for $f^{\neq}$, thus we pose $\mathcal{R}_f(x) = \{0-0\}$.
6) $[f|_{x_i=p}(x) = 1,\ f|_{x_i\neq p}(x) = 1]$
   The points in the intersection between the on-sets of the two cofactors can be treated exactly as done for completely specified functions, so we have $\mathcal{R}_f(x) = \{--1, 11-\}$.
7) $[f|_{x_i=p}(x) = 1,\ f|_{x_i\neq p}(x) = -]$
   For each point x in the on-set of $f|_{x_i=p}$ and in the don't care set of $f|_{x_i\neq p}$ we have two choices: either it is covered in the projection set $x_i = p$, thus becoming a don't care point in both the intersection and the other projection set $x_i \neq p$, or it can be covered in the intersection and be a don't care in both projection sets. Thus, we pose $\mathcal{R}_f(x) = \{1--, --1\}$.
8) $[f|_{x_i=p}(x) = -,\ f|_{x_i\neq p}(x) = 1]$
   Analogously, for each point x in don't care set of $f|_{x_i=p}$ and in the on-set of $f|_{x_i\neq p}$, we pose $\mathcal{R}_f(x) = \{-1-, --1\}$.
9) $[f|_{x_i=p}(x) = -,\ f|_{x_i\neq p}(x) = -]$
   Finally, don't cares of both cofactors can be used as don't cares in all the sets, i.e., $\mathcal{R}_f(x) = \{---\}$.

As for completely specified functions, we can rephrase the synthesis of an incompletely specified function f in P-circuit form as the problem of finding an optimal implementation of $\mathcal{R}_f$, i.e., of selecting among all possible three-output functions compatible with $\mathcal{R}_f$, the one defining a tuple $f^{=}$, $f^{\neq}$, and f^{I} whose overall SOP representation is minimal (as shown in the Algorithm in Figure 8). A formal proof of this statement is given in the next Section 3.4.

3.4 Optimality and Complexity Issues

We now formally prove that 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^{I} defining a P-circuit for a Boolean function f, so that an optimal solution of $\mathcal{R}_f$ is an optimal P-circuit $P^*(f)$ for f. We consider only the more general problem of minimizing an incompletely specified function in P-circuit form, since it subsumes the problem of minimizing a completely specified function, whose don't care set is just the empty set.

In the following, we first show the results for $p = 0$, and then we discuss the case of a general function p.

3.4.1 $\mathbf{p = 0}$

Let f be an incompletely specified Boolean function, x_i an input variable, and R_f the Boolean relation constructed as explained before. For simplicity consider now the case $p = 0$, where p does not need any minimization (note that the experimental results in [3] show that the best choice for the projection function p is the simplest $p = 0$).

Recall that the three output variables of R_f are used to describe the tuple of functions $f^{=}$, $f^{\neq}$, and f^{I} defining a P-circuit for f: the first two outputs define $f^{=}$ and $f^{\neq}$, and the third defines f^{I}. Thus, each function in $\mathcal{F}(\mathcal{R}_f)$ corresponds to a possible tuple.

Theorem 1: For $p = 0$, the set $\mathcal{F}(\mathcal{R}_f)$ of three-output functions compatible with the relation $\mathcal{R}_f$ defines exactly the set of tuples $f^{=}$, $f^{\neq}$, and f^{I} occurring in all possible P-circuit implementations of f.

Proof: First of all observe that any P-circuit $P(f)$ defines a three-output function compatible with R_f in an obvious way. The three functions $f^{=}$, $f^{\neq}$, and f^{I} represented by the three SOPs in $P(f)$ define the three outputs, and we can easily verify that for all $x \in \{0,1\}^{n-1}$, $(f^{=}(x), f^{\neq}(x), f^{I}(x)) \in \mathcal{R}_f(x)$. Indeed, for any $x \in \{0,1\}^{n-1}$

- if $(f^{=}(x), f^{\neq}(x), f^{I}(x)) = 000$, then $f|_{x_i=p}(x) \in \{0,-\}$ and $f|_{x_i\neq p}(x) \in \{0,-\}$, and the construction of $\mathcal{R}_f$ guarantees that, in each of the possible four cases, $000 \in \mathcal{R}_f(x)$ (see Figure 7);
- if $(f^{=}(x), f^{\neq}(x), f^{I}(x)) = 100$, then $f|_{x_i=p}(x) \in \{1,-\}$ and $f|_{x_i\neq p}(x) \in \{0,-\}$, and in all cases $100 \in \mathcal{R}_f(x)$, by construction (see Figure 7);
- if $(f^{=}(x), f^{\neq}(x), f^{I}(x)) = 010$, then $f|_{x_i=p}(x) \in \{0,-\}$ and $f|_{x_i\neq p}(x) \in \{1,-\}$ and in all cases $010 \in \mathcal{R}_f(x)$ (see Figure 7);
- if $(f^{=}(x), f^{\neq}(x), f^{I}(x)) \in \{001, 110, 101, 011, 111\}$, then $x \in I$, i.e., $f|_{x_i\neq p}(x) \in \{1,-\}$ and $f|_{x_i=p}(x) \in \{1,-\}$, and in all cases the corresponding output $(f^{=}(x), f^{\neq}(x), f^{I}(x)) \in \mathcal{R}_f(x)$ (see Figure 7).

We now prove that any three-output function compatible with $\mathcal{R}_f$ defines a P-circuit for f.

Consider the first two outputs. The definition of R_f (cases (2) and (3)), 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 $f|_{x_i=p}$ and to the off-set of $f|_{x_i\neq p}$, while the second output $f^{\neq}$ is equal to 1 on all points that are in the on-set of $f|_{x_i\neq p}$ and in the off-set of $f|_{x_i=p}$. Thus, $f^{=}$ contains the subset $f^{on}|_{x_i=p} \setminus I$ and $f^{\neq}$ contains $f^{on}|_{x_i\neq p} \setminus I$, where $I = (f^{on}|_{x_i=p} \cup f^{dc}|_{x_i=p}) \cap (f^{on}|_{x_i\neq p} \cup f^{dc}|_{x_i\neq p})$, as required by Definition 2.

Now, observe that cases (4) and (5) in the definition of $\mathcal{R}_f$ imply that $f^{=}$ and $f^{\neq}$ could also assume value 1 on all points that belong to the don't care set of the corresponding

cofactor ($f|_{x_i=p}$ for $f^=$ and $f|_{x_i\neq p}$ for $f^\neq$) and to the off-set of the other cofactor.

Furthermore, because of cases (6), (7), (8), and (9) in the definition of $\mathcal{R}_f$, $f^=$ and $f^\neq$ might assume value 1 also on the points in the intersection I. Finally, note that 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 2.

Consider now the third output, defining the set f^I. From the definition of $\mathcal{R}_f$, it follows that for each function in $\mathcal{F}(\mathcal{R}_f)$, f^I can get the value 1 only on the points of I, while is always 0 outside I. Moreover, on each point of I there is always the possibility of choosing between a 1 and a don't care value, so that $\emptyset \subseteq f^I \subseteq I$.

To complete the proof we must show that for any function compatible with $\mathcal{R}_f$, the P-circuit $P(f)$ constructed using the tuple $(f^=, f^\neq, f^I)$ is a P-circuit for f, i.e., $f^{on} \subseteq P(f) \subseteq f^{on} \cup f^{dc}$. This follows from the way the relation is defined on the points of I. Recall that each point x in the intersection I corresponds to a pair of points of the original space $\{0,1\}^n$, one projected onto $x_i = p$ and the other onto $x_i \neq p$, where f is equal to 1 or to a don't care. $\mathcal{R}_f$ is such that

- if $x \in I$ corresponds to two minterms in f^{on}, then in all possible outputs $\mathcal{R}_f(x)$, we have that either the third output f^I is set to 1, thus x is covered by the SOP $S(f^I)$, or both the first and the second output $f^=$ and $f^\neq$ are set to 1, and x is covered by the corresponding SOPs in both projection sets;
- for each point $x \in I$ corresponding to a minterm of one cofactor and to a don't care of the other cofactor, we have one of two cases: either we set to 1 $f^=$ (if $x \in f^{on}|_{x_i=p}$) or $f^\neq$ (if $x \in f^{on}|_{x_i\neq p}$), and therefore x is covered by $S(f^=)$ or $S(f^\neq)$, or the third output is set to 1 and therefore x is covered by $S(f^I)$;
- if $x \in I$ corresponds to a pair of don't cares of f, then R_f sets it to a don't care in all sets, so that it might be covered by $S(f^I)$, or $S(f^=)$, or $S(f^\neq)$, or none of them.

□

We finally prove that under the hypothesis that the projection function p is the constant function $p = 0$, an optimal solution of $\mathcal{R}_f$ provides an optimal P-circuit for f.

Corollary 1: For $p = 0$, the optimal solution of the Boolean relation $\mathcal{R}_f$, according to the cost function μ chosen to evaluate the circuits $P(f)$, defines an optimal P-circuit $P^*(f)$ for the function 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 P-circuit implementation for f. Indeed, with this choice of p we have that $P(f) = \overline{x}_i\, S(f^=) + x_i\, S(f^\neq) + S(f^I)$, and its cost, under any given cost metric μ, is determined by the cost under μ of the SOP representations of $f^=$, $f^\neq$ and f^I. □

From Theorem 1 and Corollary 1 we directly have the following corollary.

Corollary 2: For $p = 0$, the circuit computed by the algorithm in Figure 8, optimized according to a given cost function μ, is an optimal P-circuit $P^*(f)$ for the function f with respect to the same cost function μ.

Finally, we can prove that, when $p = 0$, the complexity of modeling P-circuit synthesis using Boolean relations is super-linear in the truth table representation of the input function f.

Theorem 2: Given a truth table representation of a completely specified Boolean function f defined over n binary variables, the construction of the Boolean relation $\mathcal{R}_f$, for $p = 0$, takes time and space upper-bounded by $O(k \log k)$, when the input dimension is $k = 2^n$.

Proof: The truth table representation of a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ consists in $k = 2^n$ bits. Suppose to generate all the $k = 2^n$ Boolean points in $\{0,1\}^n$ each containing n bits. This operation costs $O(n2^n) = O(k \log k)$. The computational time for deriving the projections $f|_{x_i=0}$ and $f|_{x_i\neq 0}$ is then linear in k, since we have to derive the points in the on-set of f that have $x_i = 0$ and $x_i = 1$, respectively.

The computation of the intersection $I = f|_{x_i=0} \cap f|_{x_i\neq 0}$ can be performed in time $O(k \log k)$ with the following strategy: 1) we sort the points in $f|_{x_i=0}$ and $f|_{x_i\neq 0}$ using a linear time sorting algorithm (e.g., radix sort [9]) in time $O((n-1)2^{n-1})$ (recall that $f|_{x_i=0}$ and $f|_{x_i\neq 0}$ are sets containing vectors of $n-1$ bits); 2) we can perform the intersection of the two sorted sets in time $O((n-1)2^{n-1})$ using a strategy similar to the merge procedure in the Mergesort [9]. Similarly, the computation of $f|_{x_i=0} \setminus I$ and $f|_{x_i\neq 0} \setminus I$ costs $O((n-1)2^{n-1})$, since $f|_{x_i=0}$, $f|_{x_i\neq 0}$ and I are sorted.

Given $f|_{x_i=0} \setminus I$, $f|_{x_i\neq 0} \setminus I$ and I, the construction of the relation R_f is then $O(n2^n)$. In fact, as shown by the algorithm in Figure 6, to each cube in $f|_{x_i=0} \setminus I$, $f|_{x_i\neq 0} \setminus I$, and I corresponds an entry in the Boolean relation; since we have $O(2^n)$ cubes of $O(n)$ variables, this phase costs $O(n2^n)$. The overall procedure takes time and space upper-bounded by $O(n2^n) = O(k \log k)$. □

Note that we can use OBDDs for computing $f|_{x_i=0} \setminus I$, $f|_{x_i\neq 0} \setminus I$, and I. This will give a computational complexity that depends on the sizes of the OBDDs.

We have the same result for incompletely specified functions with a similar proof.

3.4.2 General p

Consider now the case where we project with respect to a general p. Now if we want to construct an optimal P-circuit we must take also into account the SOP $S(p)$ for p. For this purpose, we reformulate the problem with a four output relation $\mathcal{R}_f^p$ where the output set describes all the possible tuples of functions $f^=$, $f^\neq$, f^I, and p. Thus we must double the cases enumerated for the relation $\mathcal{R}_f$. In fact, each case must consider the points that are also in p and the points that do not intersect p: the former points must have a 1 as a final output, while the latter ones must have a 0. For example, if we consider a completely specified function f, we have the following relation $\mathcal{R}_f^p$:

$x_1 \dots x_{i-1} x_{i+1} \dots x_n$	$\mathcal{R}_f^p$
points in $(f\|_{x_i=p} \setminus I) \cap p$	$\{1001\}$
points in $(f\|_{x_i=p} \setminus I) \setminus p$	$\{1000\}$
points in $(f\|_{x_i\neq p} \setminus I) \cap p$	$\{0101\}$
points in $(f\|_{x_i\neq p} \setminus I) \setminus p$	$\{0100\}$
points in $I \cap p$	$\{--11, 11-1\}$
points in $I \setminus p$	$\{--10, 11-0\}$
points in $(\{0,1\}^{n-1} \setminus (f\|_{x_i=p} \cup f\|_{x_i\neq p} \cup I)) \cap p$	$\{0001\}$
all other points	$\{0000\}$

Theorem 1 and Corollary 1 can then be easily generalized to the relation $\mathcal{R}_f^p$.

3.5 Multioutput Functions

In this section we discuss briefly how to handle a multi-output function $f : \{0,1\}^n \rightarrow \{0,1,-\}^m$.

There are two possible approaches. The first one consists simply in minimizing each output independently from the others, thus solving m P-circuit minimization problems: we construct the m Boolean relations related to the m outputs of f, and find an optimal solution for each of them, independently.

The other possibility is to handle the m outputs all together, defining a unique Boolean relation $\mathcal{R}_f$ with input set $\{0,1\}^{n-1}$, and output set $\{0,1\}^{3\,m}$, featuring three outputs of $\mathcal{R}_f$ for each output of f. Thus, the problem of minimizing f in P-circuit form is reduced to the problem of finding an optimal implementation of $\mathcal{R}_f$, according to the chosen cost metric.

However, this second approach presents some difficulties. In fact, a solution of the relation $\mathcal{R}_f$ defines correctly a P-circuit for the function f only if $\mathcal{R}_f$ is represented either in truth table form, or in an algebraic form whose products define a partition of $\{0,1\}^{n-1}$, i.e., the corresponding cubes are disjoint. In fact, when a minterm is represented by two different products, the solution of the relation would choose one of the possible associated outputs.

For example, consider a two-output function. In order to correctly partition the input points for the relation, we have to consider points that are in the first output and not in the second, then points that are in the first and in the second, and so on, including all the possible combinations of cases (e.g., points in $f^{=}$, $f^{\neq}$, etc.), as done for p in the definition of the relation $\mathcal{R}_f^p$. Therefore, the construction of the relation $\mathcal{R}_f$ could become exponential in time and space. Table 1 shows the relation $\mathcal{R}_{f^1,f^2} : \{0,1\}^{n-1} \rightarrow \{0,1\}^6$ for a completely specified multiple output function f with two outputs: f^1 and f^2, where $I^1 = f^1|_{x_i=p} \cap f^1|_{x_i\neq p}$ and $I^2 = f^2|_{x_i=p} \cap f^2|_{x_i\neq p}$.

Theorem 3: Given a SOP representation of a multioutput completely specified Boolean function $f : \{0,1\}^n \rightarrow \{0,1,-\}^m$, the construction of the Boolean relation $\mathcal{R}_f$ takes time upper-bounded by $O(mn2^n2^m)$.

Proof: For each output f^j (with $1 \leq j \leq m$) we construct $f^j|_{x_i=0} \setminus I^j$, $f^j|_{x_i\neq 0} \setminus I^j$, and I^j. From the proof of Theorem 2 we have that the complexity of this phase is $O(mn2^n)$. Then, we have to construct the relation's output set. Each minterm of the function f may belong to the on-set of each output. Thus, we have to perform all the possible intersections of the m outputs. For this purpose, we have to compute all the possible subsets of the set $\{f^1, f^2, \dots, f^m\}$, whose number is 2^m. Each intersection involves $O(m)$ outputs and each intersection between a couple of outputs costs $O(n2^n)$. The overall cost of each intersection is then $O(mn2^n)$ in time. Since we have to compute 2^m subsets, i.e., we have to perform 2^m intersections costing each $O(mn2^n)$, the intersection phase costs $O(mn2^n2^m)$. The overall cost is then upper-bounded by $O(mn2^n2^m)$.

□

For this reason we chose to consider the outputs separately.

Note that if the input is a truth table, i.e., has dimension $m2^n$, the costs $O(mn2^n2^m)$ of the construction is super-linear in n but it is still exponential in m.

Moreover, the following example illustrates another related issue, due to how output cubes are interpreted by the Boolean relation minimizer BREL [2], which we use for our experiments. Consider the relation

$$\begin{array}{cc} 0- & \{1010, 0001\} \\ -1 & \{0100\} \\ 10 & \{----\} \end{array}$$

where the minterm 01 is represented by both products $0-$ and -1. If we minimize $\mathcal{R}$, we get the following function

$$\begin{array}{cc} -1 & 0100 \\ -0 & 0001 \end{array}$$

which is compatible with $\mathcal{R}$, and therefore represents a correct solution for the relation.

However, if the relation $\mathcal{R}$ is a PLA specification of a four-output function f, according to the semantic of a classical SOP miminizer as ESPRESSO [12], output 1 prevails on output 0 in case of intersections among products. In this interpretation, the found solution does not represent correctly a SOP expression for f, since for the input 01 we get 0100, while the correct output should be 1110 or 0101. Observe that to obtain an interpretation *á la* ESPRESSO, we should change the starting relation defining the input mintems separately as follows:

$$\begin{array}{cc} 00 & \{1010, 0001\} \\ 01 & \{1110, 0101\} \\ 11 & \{0100\} \\ 10 & \{----\} \end{array}$$

4 Experimental Results

In this section we report the experimental results for the P-circuits derived by the synthesis of Boolean relations.

We evaluate and report the area, delay, and power dissipation for the cases where p is a constant (i.e., $p = 0$) and p is a variable (i.e., $p = VAR$).

The algorithms have been implemented in C, using the CUDD library for OBDDs to represent Boolean functions, and BREL [2] for the synthesis of Boolean relations since, as far as we know, it finds better solutions in shorter runtime than the previous methods.

TABLE 1
Boolean relation $\mathcal{R}_{f^1,f^2}$ for a completely specified multiple output function f with two outputs: f^1 and f^2.

$x_1 \dots x_{i-1} x_{i+1} \dots x_n$	$\mathcal{R}_{f^1,f^2}$
points in $(f^1\|_{x_i=p} \setminus I^1) \cap (f^2\|_{x_i=p} \setminus I^2)$	$\{100100\}$
points in $(f^1\|_{x_i=p} \setminus I^1) \cap (f^2\|_{x_i\neq p} \setminus I^2)$	$\{100010\}$
points in $(f^1\|_{x_i=p} \setminus I^1) \cap I^2$	$\{100--1, 10011-\}$
points in $(f^1\|_{x_i=p} \setminus I^1) \cap (\{0,1\}^{n-1} \setminus (f^2\|_{x_i=p} \cup f^2\|_{x_i\neq p}))$	$\{100000\}$
points in $(f^1\|_{x_i\neq p} \setminus I^1) \cap (f^2\|_{x_i=p} \setminus I^2)$	$\{010100\}$
points in $(f^1\|_{x_i\neq p} \setminus I^1) \cap (f^2\|_{x_i\neq p} \setminus I^2)$	$\{010010\}$
points in $(f^1\|_{x_i\neq p} \setminus I^1) \cap I^2$	$\{010--1, 01011-\}$
points in $(f^1\|_{x_i\neq p} \setminus I^1) \cap (\{0,1\}^{n-1} \setminus (f^2\|_{x_i=p} \cup f^2\|_{x_i\neq p}))$	$\{010000\}$
points in $I^1 \cap (f^2\|_{x_i=p} \setminus I^2)$	$\{--1100, 11-100\}$
points in $I^1 \cap (f^2\|_{x_i\neq p} \setminus I^2)$	$\{--1010, 11-010\}$
points in $I^1 \cap I^2$	$\{--1--1, --111-, 11---1, 11-11-\}$
points in $I^1 \cap (\{0,1\}^{n-1} \setminus (f^2\|_{x_i=p} \cup f^2\|_{x_i\neq p}))$	$\{--1000, 11-000\}$
points in $(\{0,1\}^{n-1} \setminus (f^1\|_{x_i=p} \cup f^1\|_{x_i\neq p})) \cap (f^2\|_{x_i=p} \setminus I^2)$	$\{000100\}$
points in $(\{0,1\}^{n-1} \setminus (f^1\|_{x_i=p} \cup f^1\|_{x_i\neq p})) \cap (f^2\|_{x_i\neq p} \setminus I^2)$	$\{000010\}$
points in $(\{0,1\}^{n-1} \setminus (f^1\|_{x_i=p} \cup f^1\|_{x_i\neq p})) \cap I^2$	$\{000--1, 00011-\}$
points in $(\{0,1\}^{n-1} \setminus (f^1\|_{x_i=p} \cup f^1\|_{x_i\neq p})) \cap (\{0,1\}^{n-1} \setminus (f^2\|_{x_i=p} \cup f^2\|_{x_i\neq p}))$	$\{000000\}$

TABLE 2
Synthesis time (in seconds), mapped area and delay of P-circuits, S-circuits, and SOP forms (case $p = 0$).

Benchmark (in/out)	**P-circuit μ_L**			**P-circuit μ_{BDD}**			**P-circuit [6]**			**S-circuit**			**SOP**		
	Time	**Area**	**Delay**	**Time**	**Area**	**Delay**	**Time**	**Area**	**Delay**	**Time**	**Area**	**Delay**	**Time**	**Area**	**Delay**
b10 (15/11)	9.75	**850**	39.1	0.18	938	38.7	0.05	1067	42.0	0.07	1113	47.5	0.01	881	45.2
b2 (16/17)	103.82	**3591**	68.2	1.68	3757	69.9	0.66	4479	82.8	10.19	6281	115.4	0.01	3957	73.0
b3 (32/20)	168.10	**949**	28.4	0.13	1002	29.5	0.93	1134	39.0	0.82	1284	40.1	0.03	1095	44.7
ex1010 (10/10)	50.65	**2923**	81.0	0.61	4170	96.1	0.39	4271	102.3	0.58	4818	113.5	0.31	4254	95.1
ex4 (128/28)	18.51	**800**	21.8	1.15	1099	27.3	0.22	806	25.0	0.40	1520	29.6	0.10	801	25.0
exam (10/10)	5.78	**408**	30.0	0.14	475	28.8	0.08	655	29.0	0.12	617	32.7	0.07	526	31.8
exep (30/63)	1.89	1305	30.5	0.03	1315	28.5	0.04	1347	31.6	0.07	1321	30.5	0.01	**1275**	36.1
gary (15/11)	11.98	**1000**	41.1	0.19	1096	44.9	0.04	1189	44.9	0.04	1304	50.3	0.01	1030	53.9
ibm (48/17)	13.33	741	30.1	0.58	1117	35.4	0.04	741	30.1	0.08	1368	42.9	0.01	**700**	29.0
in3 (35/29)	4.32	**929**	34.5	0.15	968	36.2	0.06	1157	35.8	0.04	1153	38.6	0.01	1111	34.1
in4 (32/20)	167.37	**1002**	28.5	0.13	1055	29.6	0.77	1192	38.0	0.56	1331	40.2	0.02	1077	44.8
jbp (36/57)	4.42	**1059**	33.8	0.16	1310	35.4	0.05	1193	38.6	1.38	1690	40.0	0.02	1115	35.9
m4 (8/16)	7.20	**783**	36.6	0.01	849	39.5	0.02	2274	71.4	0.02	2766	85.3	0.03	1778	54.2
mainpla (27/54)	338.55	**15118**	205.1	6.58	15610	217.5	12.17	26531	345.9	337.04	24421	335.2	0.08	25529	371.0
max1024 (10/6)	33.96	**1408**	47.3	0.02	1554	58.4	0.08	2755	70.6	0.07	2534	75.2	0.14	1690	53.7
misg (56/23)	1.38	153	18.2	0.04	167	18.2	0.01	153	18.2	0.04	348	23.0	0.01	**152**	18.2
mish (94/43)	0.54	**168**	10.0	0.07	207	10.0	0.09	**168**	10.0	0.29	427	13.9	0.01	**168**	9.4
misj (35/14)	0.52	**81**	9.7	0.03	125	9.1	0.01	**81**	8.9	0.01	184	13.9	0.01	**81**	8.9
pdc (16/40)	1.83	**683**	26.8	0.03	722	26.4	0.40	1555	46.8	0.53	826	34.7	0.44	1633	44.0
prom2 (9/21)	18.61	**4612**	115.1	0.01	4905	117.0	0.42	7462	171.4	0.24	7226	175.1	0.18	6775	185.1
spla (16/46)	3.24	**1826**	50.6	0.01	1935	50.7	0.17	2284	59.9	0.15	2239	53.8	0.21	2470	68.6
ts10 (22/16)	43.40	942	36.5	0.50	1410	44.9	0.17	942	36.5	0.16	1806	66.1	0.01	**901**	35.2
vg2 (25/8)	5.15	577	22.8	0.06	632	23.9	0.01	603	23.2	0.01	546	25.6	0.01	**341**	18.6
x6dn (39/5)	10.64	**760**	30.8	0.26	788	31.6	0.01	874	35.9	0.01	885	35.1	0.01	762	31.2
x7dn (66/15)	64.15	2378	51.1	0.01	2478	51.8	0.51	2501	63.8	0.54	2371	49.9	0.12	**1544**	46.1
x9dn (27/7)	3.30	412	24.1	0.04	425	24.5	0.02	412	24.1	0.02	530	27.9	0.01	**384**	23.0
xparc (41/73)	84.08	**9175**	121.2	0.32	9280	126.9	0.51	13478	187.3	0.53	13357	180.4	0.14	12678	168.8

The experiments have been run on a Linux Intel Core i7, 3.40 GHz CPU with 8 GB of main memory. The benchmarks are taken from LGSynth93 [15]. We report in the following a significant subset of the functions as representative indicators of our experiments.

TABLE 3
Average gain of P-circuits based on Boolean relations (case $p = 0$).

Average gain	**P-circuit μ_L**			**P-circuit μ_{BDD}**		
	Time	**Area**	**Delay**	**Time**	**Area**	**Delay**
w.r.t. S-circuit	-383%	37%	29%	95%	30%	25%
w.r.t. P-circuit [6]	-4214%	33%	24%	56%	25%	20%
w.r.t. SOP	-39412%	26%	19%	-304%	18%	14%

4.1 Area minimization

To show the gain in area of P-circuits derived using Boolean relations, we compare our results with the algorithm based on don't cares reported in [6].

To highlight the importance of projecting only part of the intersection, we report the results regarding circuits decomposed with standard Shannon decomposition (therefore without any intersection), called here *S-circuits*. We compare our results also with plain SOP forms, whose minimization does not even take into account the critical variable x_i.

The SOP circuits and the projections of the SOP components of the S-circuits have been synthesized using ESPRESSO in the heuristic mode. To evaluate the obtained circuits, we ran them using the SIS system with the MCNC library for technology mapping and the SIS command `map`

TABLE 4
Synthesis time (in seconds), mapped area and delay of P-circuits (case $p = VAR$).

Benchmark (in/out)	P-circuit μ_L			P-circuit μ_{BDD}			P-circuit [6]			S-circuit			SOP		
	Time	Area	Delay	Time	Area	Delay	Time	Area	Delay	Time	Area	Delay	Time	Area	Delay
b10 (15/11)	9,78	**829**	39,70	0,15	914	38,50	0,05	1067	42,0	0,07	1113	47,5	0,00	881	45,2
b2 (16/17)	128,78	4864	84,80	1,74	5096	97,20	0,66	4479	82,8	10,19	6281	115,4	0,01	**3957**	73,0
b3 (32/20)	166,63	**963**	32,30	0,12	1015	34,40	0,93	1134	39,0	0,82	1284	40,1	0,03	1095	44,7
ex1010 (10/10)	48,37	**2977**	76,00	0,58	4083	93,50	0,39	4271	102,3	0,58	4818	113,5	0,31	4254	95,1
ex4 (128/28)	18,73	872	24,00	1,17	1210	30,20	0,22	806	25,0	0,40	1520	29,6	0,10	**801**	25,0
exam (10/10)	6,15	**479**	28,20	0,12	542	31,70	0,08	655	29,0	0,12	617	32,7	0,07	526	31,8
exep (30/63)	1,80	1329	45,80	0,04	1352	45,90	0,04	1347	31,6	0,07	1321	30,5	0,01	**1275**	36,1
gary (15/11)	12,12	**960**	41,10	0,16	1064	40,20	0,04	1189	44,9	0,04	1304	50,3	0,00	1030	53,9
ibm (48/17)	13,64	780	31,40	0,63	1145	35,50	0,04	741	30,1	0,08	1368	42,9	0,00	**700**	29,0
in3 (35/29)	4,43	**942**	44,60	0,13	997	45,60	0,06	1157	35,8	0,04	1153	38,6	0,00	1111	34,1
in4 (32/20)	166,34	**1017**	32,40	0,12	1069	34,50	0,77	1192	38,0	0,56	1331	40,2	0,02	1077	44,8
jbp (36/57)	4,15	1165	41,40	0,14	1438	55,40	0,05	1193	38,6	1,38	1690	40,0	0,02	**1115**	35,9
m4 (8/16)	6,91	**809**	33,90	0,03	876	36,30	0,02	2274	71,4	0,02	2766	85,3	0,03	1778	54,2
mainpla (27/54)	314,85	**14355**	225,40	5,51	14801	247,70	12,17	26531	345,9	337,04	24421	335,2	0,08	25529	371,0
max1024 (10/6)	37,79	**1396**	57,10	0,01	1563	60,90	0,08	2755	70,6	0,07	2534	75,2	0,14	1690	53,7
misg (56/23)	1,34	159	18,20	0,04	173	18,20	0,00	153	18,2	0,04	348	23,0	0,00	**152**	18,2
mish (94/43)	0,53	178	10,80	0,05	218	15,10	0,09	**168**	10,0	0,29	427	13,9	0,00	**168**	9,4
misj (35/14)	0,51	87	9,70	0,01	130	13,30	0,00	**81**	8,9	0,00	184	13,9	0,00	**81**	8,9
pdc (16/40)	0,82	**670**	36,00	0,02	690	35,50	0,40	1555	46,8	0,53	826	34,7	0,44	1633	44,0
prom2 (9/21)	18,26	**4694**	115,80	0,00	5029	120,40	0,42	7462	171,4	0,24	7226	175,1	0,18	6775	185,1
spla (16/46)	2,84	**1766**	50,70	0,00	1860	50,00	0,17	2284	59,9	0,15	2239	53,8	0,21	2470	68,6
ts10 (22/16)	43,63	1046	38,80	0,47	1503	46,40	0,17	942	36,5	0,16	1806	66,1	0,00	**901**	35,2
vg2 (25/8)	4,34	541	21,30	0,03	625	21,90	0,00	603	23,2	0,01	546	25,6	0,00	**341**	18,6
x6dn (39/5)	10,19	763	31,60	0,24	792	32,00	0,01	874	35,9	0,00	885	35,1	0,00	**762**	31,2
x7dn (66/15)	58,77	2376	56,00	0,00	2490	51,80	0,51	2501	63,8	0,54	2371	49,9	0,12	**1544**	46,1
x9dn (27/7)	3,32	450	25,80	0,04	400	23,80	0,02	412	24,1	0,02	530	27,9	0,00	**384**	23,0
xparc (41/73)	68,68	**9165**	115,80	0,25	9212	120,50	0,51	13478	187,3	0,53	13357	180,4	0,14	12678	168,8

TABLE 5
Average gain of P-circuits based on Boolean relations (case $p = VAR$).

Average gain	P-circuit μ_L			P-circuit μ_{BDD}		
	Time	Area	Delay	Time	Area	Delay
w.r.t. S-circuit	-395%	33%	20%	95%	26%	15%
w.r.t. P-circuit [6]	-4326%	28%	14%	59%	21%	8%
w.r.t. SOP	-40440%	21%	8%	-277%	13%	2%

TABLE 6
$p = 0$ vs. $p = VAR$: percentage of benchmarks where the choice $p = 0$ exhibits better results than $p = VAR$.

P-circuit μ_L			P-circuit μ_{BDD}		
Time	Area	Delay	Time	Area	Delay
45%	78%	78%	76%	77%	79%

`-W -f 3 -s`.

In Table 2 we compare synthesis time (in seconds), mapped area and delay of P-circuits, S-circuits, and SOP forms for a significant subset of the benchmarks (i.e. a subset that has on average the same characteristics of the entire set of benchmarks), considering $p = 0$ and $x_i = x_0$. The first column reports the name of the benchmarks and the number of their inputs and outputs. The following ones report, by groups of three, the synthesis times in seconds, and the areas and delays estimated by SIS. The first two groups, labeled P-circuit μ_L and P-circuit μ_{BDD}, refer to P-circuits synthesized with the new algorithm based on Boolean relations with cost function μ_L that minimizes the number of literals, and μ_{BDD} that minimizes the size of the BDDs used for representing the relations. The next group refers to P-circuits synthesized with the minimization strategy proposed in [6]. The last two groups provide the results for S-circuits and SOP forms. For each benchmark we underline in bold the circuit that exhibits better area results.

The results show that modeling the P-circuit minimization problem using Boolean relations pays significantly. In fact, P-circuits synthesized with Boolean relations (P-circuit μ_L and P-circuit μ_{BDD}) turned out to be more compact than the corresponding P-circuits proposed in [6] in about 92% and 78% of our experiments, respectively.

The area gain of P-circuits synthesized with Boolean relations and cost function μ_L (μ_{BDD}) is 33% (25%) in the average w.r.t. P-circuits in [6]. Several benchmarks show gains above the 60% (see, for example *m4* with 66% of gain for μ_L and 63% for μ_{BDD}).

The area gain w.r.t. SOP forms is also very interesting, since P-circuits are designed for dealing with critical variables and thus area is of secondary importance. The P-circuits synthesized with Boolean relations and cost function μ_L (μ_{BDD}) are more compact than the corresponding SOP forms in about 72% (60%) of our experiments, with an average gain in area of 26% (18%).

Comparing the performances of the two new algorithms with respect to the P-circuit algorithm proposed in [6], we notice how the cost function can be critical: μ_L can be quite time-expensive (4214% penalty in computational time, on average, w.r.t. [6]), while μ_{BDD} is the best-performing algorithm (56% gain in computational time, on average, w.r.t. [6]).

Moreover, the delay gain of P-circuits synthesized with Boolean relations and cost function μ_L (μ_{BDD}) is 24% (20%) in the average w.r.t. P-circuits in [6]. This is a good result, considering that we did not perform a synthesis to reduce explicitly the delay. It is reasonable to assume that

TABLE 7
Power dissipation (in Watt) of P-circuits (case $p = 0$), S-circuits, and SOP forms.

Benchmark	in/out	P-circuit μ_L	P-circuit μ_{BDD}	P-circuit [6]	S-circuit	SOP
b3	32/20	5.04E-03	**1.18E-04**	4.40E-03	5.61E-03	5.70E-03
bench1	9/9	7.17E-03	5.27E-03	**4.73E-03**	5.64E-03	7.31E-03
ex1010	10/10	1.48E-02	**7.02E-03**	9.25E-03	1.11E-02	1.23E-02
ibm	48/17	**3.49E-03**	1.69E-02	3.52E-03	7.21E-03	3.91E-03
in3	35/29	6.43E-03	4.99E-03	**4.94E-03**	6.46E-03	5.21E-03
in4	32/20	5.39E-03	6.24E-03	**4.70E-03**	5.80E-03	5.82E-03
jbp	36/57	5.53E-03	5.63E-03	**4.93E-03**	1.15E-02	6.08E-03
max1024	10/6	7.91E-03	7.89E-03	**7.10E-03**	7.46E-03	9.81E-03
misg	56/23	3.29E-03	8.30E-03	1.51E-03	1.04E-02	**1.44E-03**
misj	35/14	1.74E-03	3.98E-03	7.72E-04	6.14E-03	**7.04E-04**
pdc	16/40	6.22E-03	**3.14E-03**	6.26E-03	6.43E-03	7.70E-03
prom2	9/21	1.43E-02	**6.39E-03**	1.31E-02	1.40E-02	1.08E-02
test4	8/30	2.05E-02	1.53E-02	**1.13E-02**	1.92E-02	1.72E-02
ts10	22/16	3.61E-03	2.00E-02	3.56E-03	5.47E-03	**3.17E-03**
x7dn	66/15	1.20E-02	**5.08E-03**	1.05E-02	1.30E-02	1.05E-02
xparc	41/73	8.20E-03	1.39E-02	**3.80E-03**	8.71E-03	3.99E-03

the delay gain is due to the reduction of the circuit area.

For a complete comparison of average gains see Table 3.

Table 4 and Table 5 report the experimental results where p is a variable (i.e., $p = x_1$), with $x_i = x_0$. Also in this case we can observe that the algorithm based on Boolean relations yields better results than the other minimization methods. In fact, P-circuits synthesized with Boolean relations (P-circuit μ_L and P-circuit μ_{BDD}) turned out to be more compact than the corresponding P-circuits proposed in [6] in about 66% and 58% of our experiments, with an average gain, in area, of 28% and 21%, respectively. Moreover, P-circuits synthesized with Boolean relations and cost function μ_L (μ_{BDD}) are more compact than the corresponding SOP forms in about 55% (47%) of our experiments, with an average gain, in area, of 21% (13%).

As expected ([3], [4]), comparing the two sets of results (i.e. $p = 0$ and $p = VAR$), we observe that the best choice for the projection function p is often $p = 0$. In Table 6 we report the percentages of benchmarks where the choice $p = 0$ obtains better results w.r.t. the choice $p = VAR$, both for P-circuit μ_L and for P-circuit μ_{BDD}.

In summary we can observe that the algorithm based on Boolean relations with cost function μ_{BDD} shows a good trade-off between area minimization and computational time.

4.2 Power dissipation

To evaluate power dissipation, the circuits were synthesized using the following equal-delay gates in a 180 nm CMOS technology: inverter, 2,3,4-input NAND, 2,3,4-input NOR and 2-input XOR gates. Since in CMOS technology at 180 nm the switching power is the predominant component of the overall power consumption, our model takes into account only this component.

The circuits were analyzed with two different average values of input transition rates, giving a high-density switching activity for only one known signal (in our case x_0) and a low-density activity for all other signals. The number of input logic transitions for the high-density switching activity signal is approximately one hundred times larger than the number of input logic transitions for the low-density switching activity signals.

The circuits have been simulated at transistor level using SPECTRE on a 1600 MHz Pentium 4 workstation. The values of i_{DD} and i_{SS} currents were sampled and stored for post-processing. In all the experiments we considered for each benchmark decompositions with respect to the input variable with the highest switching frequency.

In Table 7 we compare power dissipation (in Watt) of P-circuits (case $p = 0$), S-circuits, and SOP forms for a subset of the functions as representative indicators of our experiments. The first two columns report the name of the benchmarks and the number of their inputs and outputs. The following ones report the estimated power consumption. In particular, the third and fourth columns, labeled P-circuit μ_L and P-circuit μ_{BDD}, respectively, refer to P-circuits synthesized with the new algorithm based on Boolean relations with cost function μ_L that minimizes the number of literals, and cost function μ_{BDD} that minimizes the size of the BDDs used for representing the relation. The next group refers to P-circuits synthesized with the minimization strategy proposed in [6]. The last two groups provide the results for S-circuits and SOP forms. For each benchmark we underline in bold the circuit that exhibits better results.

Table 8 summarizes the results obtained using Boolean relations. The power consumption of P-circuits synthesized with Boolean relations (P-circuit μ_L and P-circuit μ_{BDD}) is lower than the power consumption of the corresponding P-circuits proposed in [6] in about 13% of our experiments. This is a reasonable good result, since it is obtained by the comparison of the proposed approach (P-circuit with Boolean relation) against an approach (P-circuit [6]) that is itself an improvement with respect to other synthesis techniques (SOP and S-circuits). The percentage of improvement goes up if we compare P-circuits synthesized with Boolean relations against SOP and S-circuits.

In summary, if we compare Tables 3 and 8, we observe that synthesis with cost function μ_L allows to obtain, on average, high gains in power and area/delay, at the expense

TABLE 8
Comparison of power dissipation (case $p = 0$).

	P-circuit μ_L	**P-circuit** μ_{BDD}
w.r.t. S-circuit	65%	61%
w.r.t. P-circuit [6]	13%	13%
w.r.t. SOP	44%	62%

of a significant increase in computational time. On the other hand, the P-circuits synthesized with the algorithm based on Boolean relations with cost function μ_{BDD} have a satisfactory gain in power, area and delay against a moderate increase of computational time. Therefore, the P-circuit synthesis based on the cost function μ_{BDD} is a good trade-off between power dissipation, computational time and area/delay minimization.

5 Conclusion and Future Work

In this paper we studied the problem of characterizing the complete flexibility of P-circuits and exploiting it. The previous algorithms did not necessarily find the best implementation, because they modeled the problem as minimizing an incompletely specified Boolean function. Here we showed that to explore all possible solutions one must set up and minimize a Boolean relation, because there are combinations of don't care conditions that cannot be expressed by cubes. The construction of the Boolean relation is super-linear in the input. In the experiments we report major improvements with respect to the previously published results.

Future work includes two main lines of investigation: evaluating the impact of more general cofactoring p functions, and handling multiouput functions trading-off quality of results vs. scalability. About the former, currently we experimented mainly with $p = 0$ and $p = VAR$, but we lack a systematic understanding of what p are the best and how to choose them. About the latter, we performed the experiments only with single-output disjoint minimization of multioutput functions, because modeling with Boolean relations the complete flexibility of multioutput functions looks computationally too expensive, as shown in the related section. However, we would like to assess the quality gap between the two extreme algorithmic choices, and come up with Boolean relations that model only partially the sharing of the multi-output functions, to balance the objectives of more aggressive optimization vs. runtime. Moreover, how to integrate the proposed decomposition technique with other transformations of the Boolean network and its nodes requires more theoretical and experimental work to come up with a fine-tuned optimization script for multi-level synthesis. This is an interesting future direction.

References

[1] D. Bañeres, J. Cortadella, and M. Kishinevsky, "A Recursive Paradigm to Solve Boolean Relations," in *Proc. ACM/IEEE Design Automation Conference*, 2004, pp. 416–421.

[2] ——, "A Recursive Paradigm to Solve Boolean Relations," *IEEE Transactions on Computers*, vol. 58, no. 4, pp. 512–527, 2009.

[3] 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.

[4] ——, "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.

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

[6] 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.

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

[8] 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.

[9] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*. MIT Press, 2009.

[10] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, E. Pastor, and A. Yakovlev, "Decomposition and Technology Mapping of Speed-Independent Circuits using Boolean Relations," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 18, no. 9, pp. 1221–1236, 1999.

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

[12] 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.

[13] E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. Brayton, and A. Sangiovanni-Vincentelli, "SIS: A System for Sequential Circuit Synthesis," Tech. Rep. No. UCB/ERL M92/41, Berkeley, CA, Tech. Rep., 1992.

[14] B. Sterin, A. Mishchenko, N. Een, and R. Brayton, "Logic Synthesis for Disjunctions of Boolean Functions," in *IEEE/ACM 21st International Workshop on Logic & Synthesis (IWLS)*, 2012, pp. 73–77.

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