# On Decomposing Boolean Functions via Extended Cofactoring


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

Valentina Ciriani, Gabriella Trucco
Department of Information Technologies
University of Milano, Italy
{ciriani, trucco}@dti.unimi.it

Tiziano Villa
Department of Computer Science
University of Verona, Italy
tiziano.villa@univr.it



***Abstract*—We investigate restructuring techniques based on decomposition/factorization, with the objective to move critical signals toward the output while minimizing area. A specific application is synthesis for minimum switching activity (or high performance), with minimum area penalty, where decompositions with respect to specific critical variables are needed (the ones of highest switching activity for example). In this paper we describe new types of factorization that extend Shannon cofactoring and are based on projection functions that change the Hamming distance of the original minterms and on appropriate don't care sets, to favor logic minimization of the component blocks. We define two new general forms of decomposition that are special cases of the pattern F = G(H(X),Y). The related implementations, called P-Circuits, show experimentally promising results in area with respect to Shannon cofactoring.**


## I. Introduction

The design of CMOS digital circuits targets optimization objectives like area, delay and power consumption. Achieving low power-consumption is increasingly important due to the spreading of portable electronic devices. In CMOS technology, power consumption is characterized by three components: dynamic, short-circuit, and leakage power dissipation, of which dynamic power dissipation is the predominant one. Dynamic power dissipation is due to the charge and discharge of load capacitances, when the logic value of a gate output toggles; switching a gate may trigger a sequence of signal changes in the gates of its output cone, increasing dynamic power dissipation. So, reducing switching activity reduces dynamic power consumption.

Previous work proposed various transformations to decrease power consumption and delay (for instance [8], [11], [13] for performance, and [1], [10], [12] for low power), whereby the circuit is restructured in various ways, e.g., redeploying signals to avoid critical areas, bypassing large portions of a circuit. For instance, if we know the switching frequency of the input signals, a viable strategy to reduce dynamic power is to move the signals with the highest switching frequency closer to the outputs, in order to reduce the part of the circuit affected by the switching activity of these signals. Similarly for performance, late-arriving signals are moved closer to the outputs to decrease the worst-case delay.

The objective of our research is a systematic investigation of restructuring techniques based on decomposition/factorization, with the objective to move critical signals toward the output and avoid losses in area. A specific application is synthesis for minimum switching activity (or high performance), with minimum area penalty.

Differently from factorization algorithms developed only for area minimization, we look for decompositions with respect to specific critical variables (the ones of highest switching activity for example). This is exactly obtained by Shannon cofactoring, which decomposes with respect to a chosen splitting variable; however, when applying Shannon, the drawback is that too much area redundancy might be introduced because large cubes are split between subspaces, whereas no new cube merging will take place. So one should look for different cofactors and expansions geared towards area minimization.

In this paper we study more general types of factorization that extend straightforward Shannon cofactoring; instead of cofactoring only with respect to single variables as Shannon does, we will cofactor with respect to more complex functions, expanding with respect to the orthogonal basis $\overline{x}_i \oplus p$ (i.e., $x_i = p$), and $x_i \oplus p$ (i.e., $x_i \neq p$), where $p(x)$ is a function defined over all variables except $x_i$. We will study different functions $p(x)$ trading-off quality vs. computation time. Our new factorizations modify the Hamming distance of the onset minterms, so that more logic minimization may be performed, while signals are moved in the circuit closer to the outputs. To favor minimization, the final expansion is defined in such a way to avoid cube fragmentation (e.g., cube splitting for the cubes intersecting both subspaces $x_i = p$ and $x_i \neq p$), by the introduction of appropriate don't care sets for the blocks of the decomposition. This can be seen as a form of generalized cofactoring (that is based on expanding a function over an orthogonal basis), augmented by appropriate don't care sets.

The contribution of this paper is represented by two new general forms of decomposition that are special cases of the type $F = G(H(X), Y)$, with $G$ fixed and $X, Y$ disjoint variables ($X \cap Y = \emptyset$). The paper is organized as follows. Sec. II describes a new theory of decomposition based on generalized cofactoring, which is applied in Sec. III to the synthesis of Boolean functions by a new structure called P-circuits. Experiments and conclusions are reported in Sec. IV.

## II. Decomposition Methods

How to decompose Boolean functions is an on-going research area to explore alternative logic implementations. A


978-3-9810801-5-5/DATE09 © 2009 EDAA

technique to decompose Boolean functions is based on expanding them according to an orthogonal basis (see [6], Ch. 3.15), as in the following definition, where a function $f$ is decomposed according to the basis $(g, \overline{g})$.

*Definition 1:* Let $f = (f_{on}, f_{dc}, f_{off})$ be an incompletely specified function and $g$ be a completely specified function, the generalized cofactor of $f$ with respect to $g$ is the incompletely specified function $co(f, g) = (f_{on}.g, f_{dc} + \overline{g}, f_{off}.g)$.

This definition highlights that in expanding a Boolean function we have two degrees of freedom: choosing the basis (in this case, the function $g$), and choosing one completely specified function included in the incompletely specified function $co(f, g)$. This flexibility can be exploited according to the purpose of the expansion. For instance, when $g = x_i$, we have $co(f, x_i) = (f_{on}.x_i, f_{dc} + \overline{x}_i, f_{off}.x_i)$. Notice that the well-known Shannon co-factor $f_{x_i} = f(x_1, \ldots, (x_i = 1), \ldots, x_n)$ is a completely specified function contained in $co(f, x_i) = (f_{on}.x_i, f_{dc} + \overline{x}_i, f_{off}.x_i)$ (since $f_{on}.x_i \subseteq f_{x_i} \subseteq f_{on}.x_i + f_{dc} + \overline{x}_i = f_{on} + f_{dc} + \overline{x}_i$); moreover, $f_{x_i}$ is the unique cover of $co(f, x_i)$ independent of the variable $x_i$.

We introduce now two types of expansion of a Boolean function that yield decompositions with respect to a chosen variable (as in Shannon cofactoring), but are also area-efficient because they favor minimization in the obtained logic blocks.

Let $f(X) = (f_{on}(X), f_{dc}(X), f_{off}(X))$ be an incompletely specified Boolean function depending on the set $X = \{x_1, x_2, \ldots, x_n\}$ of $n$ binary variables. Let $X^{(i)}$ be the subset of $X$ containing all variables but $x_i$, i.e., $X^{(i)} = X \setminus \{x_i\}$, where $x_i \in X$. Consider now a completely specified Boolean function $p(X^{(i)})$ depending only on the variables in $X^{(i)}$. We introduce two Boolean functional decomposition techniques based on the projections of the function $f$ onto two complementary subsets of the Boolean space $\{0,1\}^n$ defined by the function $p$. More precisely, we note that the space $\{0,1\}^n$ can be partitioned into two sets: one containing the points for which $x_i = p(X^{(i)})$ and the other containing the points for which $x_i \neq p(X^{(i)})$. Observe that the characteristic functions of these two subsets are $(\overline{x}_i \oplus p)$ and $(x_i \oplus p)$, respectively, and that these two sets have equal cardinality. We denote by $f|_{x_i = p}$ and $f|_{x_i \neq p}$ the projections of the points of $f(X)$ onto the two subsets where $x_i = p(X^{(i)})$ and $x_i \neq p(X^{(i)})$, respectively. Note that these two functions only depend on the variables in $X^{(i)}$. The first decomposition technique, already described in [9] and [4], is defined as follows.

*Definition 2:* Let $f(X)$ be an incompletely specified Boolean function, $x_i \in X$, and $p(X^{(i)})$ be a completely specified Boolean function. The $(x_i, p)$-*decomposition* of $f$ is the algebraic expression

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

First of all we observe that each minterm of $f$ is projected onto one and only one subset. Indeed, let $m = m_1 m_2 \cdots m_n$ be a minterm of $f$; if $m_i = p(m_1, \ldots, m_{i-1}, m_{i+1}, \ldots, m_n)$, then $m$ is projected onto the set where $x_i = p(X^{(i)})$, otherwise $m$ is projected onto the complementary set where $x_i \neq p(X^{(i)})$. The projection simply consists in eliminating $m_i$ from $m$. For example, consider the function $f$ shown on the left side of Fig. 1 with $f_{on} = \{0000, 0001, 0010, 0101, 1001, 1010, 1100, 1101\}$ and $f_{dc} = \{0111\}$. Let $p$ be the simple Boolean function $x_2$, and $x_i$ be $x_1$. The Boolean space $\{0,1\}^4$ can be partitioned in the two sets: $x_1 = x_2$ and $x_1 \neq x_2$ each containing $2^3$ points. The projections of $f$ onto these two sets are $f_{on}|_{x_1=x_2} = \{000, 001, 010, 100, 101\}$ , $f_{dc}|_{x_1=x_2} = \emptyset$, and $f_{on}|_{x_1 \neq x_2} = \{101, 001, 010\}$, $f_{dc}|_{x_1 \neq x_2} = \{111\}$.

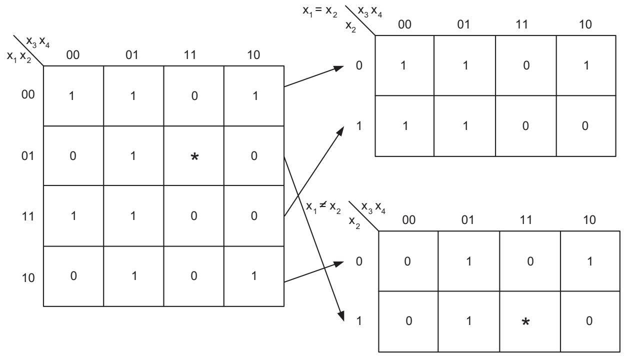


Fig. 1. An example of projection of the incompletely specified function $f$ onto the spaces $x_1 = x_2$ and $x_1 \neq x_2$.

Secondly, observe that these projections do not preserve the Hamming distances among minterms, since we eliminate the variable $x_i$ from each minterm, and two minterms projected onto the same subset could have different values for $x_i$. The Hamming distance is preserved only if the function $p(X^{(i)})$ is a constant, that is when the $(x_i, p)$-decomposition corresponds to the classical Shannon decomposition. The fact that the Hamming distances may change could be useful when $f$ is represented in SOP form, as bigger cubes could be built in the projection sets. For example, consider again the function $f$ shown on the left side of Fig. 1. The points 0000 and 1100 contained in $f_{on}$ have Hamming distance equal to 2, and thus cannot be merged in a cube, while their projections onto the space $f_{on}|_{x_1=x_2}$ (i.e., 000 and 100, respectively) have Hamming distance equal to 1, and they form the cube $\overline{x}_3 \overline{x}_4$.

On the other hand, the cubes intersecting both subsets $x_i = p(X^{(i)})$ and $x_i \neq p(X^{(i)})$ are divided into two smaller subcubes. For instance, in our running example, the cube $\overline{x}_3 x_4$ of function $f_{on}$ is split in the two sets $x_1 = x_2$ and $x_1 \neq x_2$ forming a cube in $f_{on}|_{x_1=x_2}$ and one in $f_{on}|_{x_1 \neq x_2}$, as shown on the right side of Fig. 1.

Observe that the cubes eventually split can contain pairs of minterms, whose projections onto the two sets are identical. In our example, $\overline{x}_3 x_4$ is the cube corresponding to the points $\{0001, 0101, 1001, 1101\}$, where 0001 and 1101 are projected onto $f_{on}|_{x_1=x_2}$ and become 001 and 101, respectively, and 0101 and 1001 are projected onto $f_{on}|_{x_1 \neq x_2}$ and again become 101 and 001, respectively. Therefore, we can characterize the set of these minterms as $I = f|_{x_i=p} \cap f|_{x_i \neq p}$. Note that the points in $I$ do not depend on $x_i$. In our example

$I_{on} = f_{on}|_{x_1=x_2} \cap f_{on}|_{x_1 \neq x_2} = \{001, 010, 101\}$, and $I_{dc} = \emptyset$.

In order to overcome some splitting cubes, we could keep $I$ unprojected, and project only the points in $f|_{x_i=p} \setminus I$ and $f|_{x_i \neq p} \setminus I$, obtaining the expression $f = (\overline{x}_i \oplus p)(f|_{x_i=p} \setminus I) + (x_i \oplus p)(f|_{x_i \neq p} \setminus I) + I$.

However, we are left with another possible drawback: some points of $I$ could also belong to cubes covering points of $f|_{x_i=p}$ and/or $f|_{x_i \neq p}$, and their elimination could cause the fragmentation of these cubes. Thus, eliminating these points from the projected subfunctions would not be always convenient. On the other hand, some points of $I$ are covered only by cubes entirely belonging to $I$. Therefore keeping them both in $I$ and in the projected subfunctions would be useless and expensive. In our example, since $I_{on} = \{001, 010, 101\}$, in $f_{on}|_{x_1=x_2}$ 001 and 101 are useful for forming, together with 000 and 100, the cube $\overline{x}_3$; instead the point 010 is useless and must be covered with an additional cube. The solution of this problem is to project the points belonging to $I$ as don't cares for $f|_{x_i=p}$ and $f|_{x_i \neq p}$, in order to choose only the useful cubes. We therefore propose the following more refined decomposition, using the notation $h = (h_{on}, h_{dc})$ for an incompletely specified function $h$ and its on-set $h_{on}$ and don't care set $h_{dc}$.

*Definition 3:* Let $f(X)$ be an incompletely specified Boolean function, $x_i \in X$, and $p(X^{(i)})$ be a completely specified Boolean function. The $(x_i, p)$-*decomposition with intersection* of $f = (f_{on}, f_{dc})$ is the algebraic expression

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

where

$$\begin{aligned}\tilde{f}|_{x_i=p} &= (f_{on}|_{x_i=p} \setminus I_{on}, f_{dc}|_{x_i=p} \cup I_{on})\,,\\ \tilde{f}|_{x_i \neq p} &= (f_{on}|_{x_i \neq p} \setminus I_{on}, f_{dc}|_{x_i \neq p} \cup I_{on})\,,\\ I &= (I_{on}, I_{dc})\,,\end{aligned}$$

with $I_{on} = f_{on}|_{x_i=p} \cap f_{on}|_{x_i \neq p}$ and $I_{dc} = f_{dc}|_{x_i=p} \cap f_{dc}|_{x_i \neq p}$. For our example, the projections of $f$ become $\tilde{f}|_{x_1=x_2} = (f_{on}|_{x_1=x_2} \setminus I_{on}, f_{dc}|_{x_1=x_2} \cup I_{on}) = (\{000, 100\}, \{001, 010, 101\})$ and $\tilde{f}|_{x_1 \neq x_2} = (f_{on}|_{x_1 \neq x_2} \setminus I_{on}, f_{dc}|_{x_1 \neq x_2} \cup I_{on}) = (\emptyset, \{111\} \cup \{001, 010, 101\})$. The Karnaugh maps of this decomposition are show in Fig. 2. Observe that, fixing the function $p$ and a variable $x$, these decompositions are canonical. We now study these decomposition methods for some choices of the function $p$.

*a) Case $p = 0$.:* As we have already observed, if $p$ is a constant function, then the $(x_i, p)$-decomposition is indeed the classical Shannon decomposition: $f = \overline{x}_i f|_{x_i=0} + x_i f|_{x_i=1}$. Recall that $(\overline{x}_i \oplus 0)$ is equivalent to $\overline{x}_i$, while $(x_i \oplus 0)$ is equivalent to $x_i$. Also observe that choosing $p = 1$ we would get exactly the same form. For the $(x_i, p)$-decomposition with intersection we have the following particular form:

$$f = \overline{x}_i \tilde{f}|_{x_i=0} + x_i \tilde{f}|_{x_i=1} + I\,.$$

Observe that in this particular case, the set $I$ is

$$\begin{aligned}I &= f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)\\ &\cap\ f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)\,.\end{aligned}$$

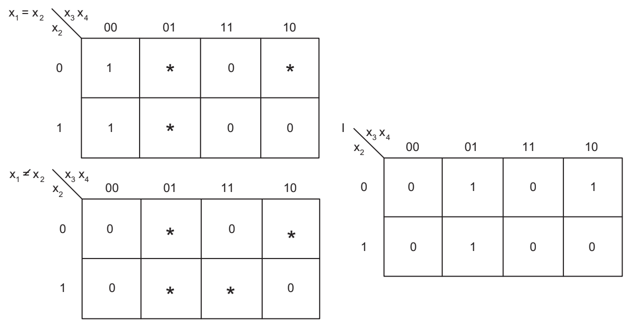


Fig. 2. An example of projection *with intersection* of the function $f$ of Fig. 1 onto the spaces $x_1 = x_2$ and $x_1 \neq x_2$.

This implies the following property.

*Proposition 1:* The characteristic function $\chi_I$ of $I$ is the biggest subfunction of $f$ that does not depend on $x_i$.

*Proof:* Let $\chi_1, \ldots, \chi_k$ be the subfunctions of $f$ that do not depend on $x_i$, and let $\chi$ be their union, i.e., $\chi = \chi_1 + \chi_2 + \ldots + \chi_k$. Observe that $\chi$ is still a subfunction of $f$ and it does not depend on $x_i$. Therefore $\chi$ is the biggest subfunction that does not depend on $x_i$. We must show that $\chi = \chi_I$. First note that $\chi_I$ is one of the functions $\chi_1, \ldots, \chi_k$. Suppose $\chi_I = \chi_j$, $1 \leq j \leq k$. By construction, $\chi_j$ is a subfunction of $\chi$. On the other hand, if $\chi(X) = 1$, then there exists an index $h$ such that $\chi_h(X) = 1$. Since $\chi_h$ does not depend on $x_i$, we have

$$\begin{aligned}&\chi_h(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) =\\ &\quad = \chi_h(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) = 1\,.\end{aligned}$$

Moreover, since $\chi_h$ is a subfunction of $f$, on the same input $X$ we have that

$$\begin{aligned}&f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) =\\ &\quad = f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) = 1\,.\end{aligned}$$

This implies that

$$\begin{aligned}\chi_j(X) &= f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)\\ &\cap\ f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) = 1\,,\end{aligned}$$

which means that $\chi$ is a subfunction of $\chi_j$. As $\chi_j = \chi_I$, we finally have that $\chi = \chi_I$. ■

Note that if $\chi_I$ is equal to $f$, then $f$ does not depend on $x_i$.

We conclude the analysis of this special case observing how the $(x_i, 0)$-decomposition, i.e., the classical Shannon decomposition, and the $(x_i, 0)$-decomposition with intersection show a different behavior when the subfunctions $f|_{x_i=0}$, $f|_{x_i=1}$, $\tilde{f}|_{x_i=0}$, $\tilde{f}|_{x_i=1}$, and the intersection $I$ are represented as sums of products. Consider a minimal sum of products $SOP(f)$ for the function $f$. The number of products in $SOP(f)$ is always less or equal to the overall number of products in the minimal SOP representations for $f|_{x_i=0}$ and $f|_{x_i=1}$. This easily follows from the fact that each product in $SOP(f)$ that does not depend on $x_i$ is split into two products, one belonging to a minimal SOP for $f|_{x_i=0}$ and the other belonging to a minimal SOP for $f|_{x_i=1}$. On the other hand, the $(x_i, 0)$-decomposition

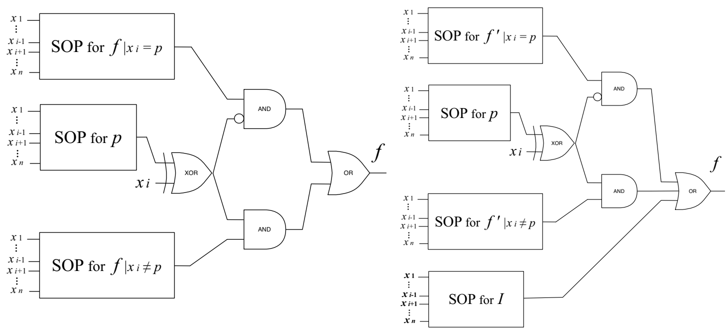


Fig. 3. P-circuit (left) and P-circuit with intersection (right).

with intersection contains the same number of products as $SOP(f)$, and its overall number of literals is less or equal to the number of literals in $SOP(f)$.

*Theorem 1:* An $(x_i, 0)$-decomposition with intersection for a function $f$, where $\tilde{f}|_{x_i=0}$, $\tilde{f}|_{x_i=1}$, and $I$ are represented as minimal sums of products, contains an overall number of products equal to the number of products in a minimal SOP for $f$, and an overall number of literals less or equal to the number of literals in a minimal SOP for$f$.

*Proof:* First observe how we can build minimal SOP representations for $\tilde{f}|_{x_i=0}$, $\tilde{f}|_{x_i=1}$, and $I$ starting from a minimal SOP $SOP(f)$ for $f$. Indeed, the sum of the projections of all products in $SOP(f)$ containing the literal $x_i$ gives a minimal SOP for $\tilde{f}|_{x_i=1}$, the sum of the projections of all products in $SOP(f)$ containing the literal $\overline{x}_i$ gives a minimal SOP for $\tilde{f}|_{x_i=0}$, while all remaining products, that do not depend on $x_i$ or $\overline{x}_i$, give a minimal SOP covering exactly the points in the intersection $I$. The minimality of these SOPs follows from the fact that the $(x_i, 0)$-decomposition with intersection does not change the Hamming distances among the minterms, so that no bigger cubes could be built in the projection sets.

Let us now analyze the overall number of literals in the $(x_i, 0)$-decomposition with intersection built from $SOP(f)$. Let $\ell_{SOP}$ denote the number of literals in $SOP(f)$. The products in the SOP for $I$ are left unchanged, so that their overall number of literals $\ell_I$ is preserved. Suppose that $r$ products in $SOP(f)$ contain $x_i$, and let $\ell_{x_i}$ denote their overall number of literals. The projection of these $r$ products forms a SOP for $\tilde{f}|_{x_i=1}$, whose number of literals is equal to $\ell_{x_i} - r$, as projecting a product simply consists in eliminating $x_i$ from it. Analogously, if $s$ products in $SOP(f)$ contain $\overline{x}_i$, and $\ell_{\overline{x}_i}$ is their overall number of literals, the SOP for $\tilde{f}|_{x_i=0}$ contains $\ell_{\overline{x}_i} - s$ literals. Thus, the $(x_i, 0)$-decomposition with intersection contains exactly $\ell_I + \ell_{x_i} - r + \ell_{\overline{x}_i} - s + 2 = \ell_{SOP} - r - s + 2$ literals, where the two additional literals represent the characteristic functions of the projection sets. ∎

*b) Case $p = x_j$.:* For $p = x_j$, with $j \neq i$, the two decomposition techniques are based on the projection of $f$ onto the two complementary subspaces of $\{0,1\}^n$ where $x_i = x_j$ and $x_i \neq x_j$. For the $(x_i, x_j)$-decomposition we get the following expression $f = (\overline{x}_i \oplus x_j) f|_{x_i=x_j} + (x_i \oplus x_j) f|_{x_i \neq x_j}$, while the $(x_i, x_j)$-decomposition with intersection is given by $f = (\overline{x}_i \oplus x_j)\tilde{f}|_{x_i=x_j} + (x_i \oplus x_j)\tilde{f}|_{x_i \neq x_j} + I$, where

$$\tilde{f}|_{x_i=x_j} = (f_{on}|_{x_i=x_j} \setminus I_{on}, f_{dc}|_{x_i=x_j} \cup I_{on}),$$
$$\tilde{f}|_{x_i \neq x_j} = (f_{on}|_{x_i \neq x_j} \setminus I_{on}, f_{dc}|_{x_i \neq x_j} \cup I_{on}),$$

with $I_{on} = f_{on}|_{x_i=x_j} \cap f_{on}|_{x_i \neq x_j}$ and $I_{dc} = f_{dc}|_{x_i=x_j} \cap f_{dc}|_{x_i \neq x_j}$. These expressions share some similarities with the *EXOR Projected Sum of Products* studied in [3]. In particular, if we represent the subfunctions as sums of products, the $(x_i, x_j)$-decomposition corresponds to an *EP-SOP form*, while the $(x_i, x_j)$-decomposition with intersection is only partially similar to an *EP-SOP with remainder form* [3]. The differences between the two expressions are due to the presence of don't cares in $\tilde{f}|_{x_i=x_j}$ and $\tilde{f}|_{x_i \neq x_j}$, and to the fact that the intersection $I$ does not depend on the variable $x_i$, while the remainder in an EP-SOP may depend on all the $n$ input variables. Also observe that, thanks to the presence of don't cares, the $(x_i, x_j)$-decomposition with intersection has a cost less or equal to that of an EP-SOP with remainder.

*c) Cases $p = x_j \oplus x_k$ and $p = x_j x_k$.:* In general the function $p$ used to split the Boolean space $\{0,1\}^n$ may depend on all input variables, but $x_i$. In this paper we consider only two special cases, based on the use of two simple functions: an EXOR and an AND of two literals. The partition of $\{0,1\}^n$ induced by the EXOR function does not depend on the choice of the variable complementations. Indeed, since $x_j \oplus x_k = \overline{x}_j \oplus \overline{x}_k$, and $\overline{(x_j \oplus x_k)} = \overline{x}_j \oplus x_k = x_j \oplus \overline{x}_k$, the choices $p = x_j \oplus x_k$ and $p = \overline{x}_j \oplus x_k$ give the same partition of the Boolean space. On the contrary, the partition of $\{0,1\}^n$ induced by the AND function changes depending on the choice of the variable complementations, so that four different cases must be considered:

1) $p = x_j x_k$, corresponding to the partition into the sets where $x_i = x_j x_k$ and $x_i \neq x_j x_k$, i.e., $x_i = \overline{x}_j + \overline{x}_k$;
2) $p = x_j \overline{x}_k$, corresponding to the partition into the sets where $x_i = x_j \overline{x}_k$ and $x_i \neq x_j \overline{x}_k$, i.e., $x_i = \overline{x}_j + x_k$;
3) $p = \overline{x}_j x_k$, corresponding to the partition into the sets where $x_i = \overline{x}_j x_k$ and $x_i \neq \overline{x}_j x_k$, i.e., $x_i = x_j + \overline{x}_k$;
4) $p = \overline{x}_j \overline{x}_k$, corresponding to the partition into the sets where $x_i = \overline{x}_j \overline{x}_k$ and $x_i \neq \overline{x}_j \overline{x}_k$, i.e., $x_i = x_j + x_k$.

When the subfunctions are represented as SOPs, the resulting decomposition forms share some similarities with the *Projected Sum of Products (P-SOP)* introduced in [2]. Again, the

**Synthesis of $P$-Circuits**

**INPUT:** Functions $f$ and $p$, and a variable $x_i$
**OUTPUT:** An optimal $P$-circuit for the $(x_i, p)$-decomposition of $f$
**NOTATION:** let $f = (f_{on}, f_{dc})$, i.e., $f_{on}$ is the on-set of $f$, and $f_{dc}$ is the don't care-set of $f$,

$f_{on}^{(=)}$ = $f_{on}|_{x_i=p}$;
$f_{on}^{(\neq)}$ = $f_{on}|_{x_i\neq p}$;
$f_{dc}^{(=)}$ = $f_{dc}|_{x_i=p}$;
$f_{dc}^{(\neq)}$ = $f_{dc}|_{x_i\neq p}$;
$MinSOP^{(=)}$ = $OptSOP(f_{on}^{(=)}, f_{dc}^{(=)})$; // optimal SOP for $f^{(=)}$
$MinSOP^{(\neq)}$ = $OptSOP(f_{on}^{(\neq)}, f_{dc}^{(\neq)})$; // optimal SOP for $f^{(\neq)}$
$MinSOP^{p}$ = $OptSOP(p, \emptyset)$; // optimal SOP for $p$
$P$-circuit = $(\overline{x}_i \oplus MinSOP^p)MinSOP^{(=)}+$
$(x_i \oplus MinSOP^p)MinSOP^{(\neq)}$
**return** $P$-circuit

Fig. 4. Algorithm for the optimization of P-circuits.

two forms are different thanks to the presence of don't cares in the subfunctions, and to the fact that the intersection $I$ does not depend on $x_i$.

## III. P-CIRCUITS

We now show how the decomposition methods described in Section II can be applied to the logic synthesis of Boolean functions. The synthesis idea is simply that of constructing a network for $f$ using networks for the projection function $p$, for the subfunctions $f|_{x_i=p}$, $f|_{x_i\neq p}$, $\tilde{f}|_{x_i=p}$, and $\tilde{f}|_{x_i\neq p}$, and a network for the intersection $I$ as building blocks. Observe that the overall network for $f$ will require an EXOR gate for computing the characteristic functions of the projection subsets, two AND gates for the projections and a final OR gate (see Fig. 3).

The function $p$, the projected subfunctions, and the intersection can be synthesized in any framework of logic minimization. In our experiments we focused on the standard *Sum of Products* synthesis, i.e., we represented $p$, $f|_{x_i=p}$, $f|_{x_i\neq p}$, $\tilde{f}|_{x_i=p}$, $\tilde{f}|_{x_i\neq p}$, and $I$ as sums of products. In this way we derived networks for $f$ which we called *Projected Circuit* and *Projected Circuit with Intersection*, in short *P-Circuits*, see Fig. 3. If the SOPs representing $p$, $f|_{x_i=p}$, $f|_{x_i\neq p}$, $\tilde{f}|_{x_i=p}$, $\tilde{f}|_{x_i\neq p}$, and $I$ are minimal, the corresponding circuits are called *Optimal P-Circuits*. For instance, the function in Figures 1 and 2 has minimal SOP form $\overline{x}_1\overline{x}_2\overline{x}_3 + x_1x_2\overline{x}_3 + \overline{x}_3x_4 + \overline{x}_2x_3\overline{x}_4$, while its corresponding optimal P-circuit is $(\overline{x}_1 \oplus x_2)\overline{x}_3 + \overline{x}_3x_4 + \overline{x}_2x_3\overline{x}_4$. The number of logic levels in a P-circuit varies from four to five: it is equal to four whenever the SOP for $p$ consists in just one product, and it is equal to five otherwise.

If we consider now the power consumption, we can observe in Fig. 3 that $x_i$, i.e., the variable with the highest switching frequency, is connected near the output of the overall logic network, thus triggering a sequence of switching events only for the last four gates. In this way, the contribution of $x_i$ to the total power consumption is limited. Finally, we observe that it is possible to apply recursively this decomposition when more than one variable switches with high frequency.

**Synthesis of $P$-Circuits with intersection**

**INPUT:** Functions $f$ and $p$, and a variable $x_i$
**OUTPUT:** An optimal $P$-circuit for the $(x_i, p)$-decomposition with intersection of $f$
**NOTATION:** let $f = (f_{on}, f_{dc})$, i.e., $f_{on}$ is the on-set of $f$, and $f_{dc}$ is the don't care-set of $f$,

$I_{on}$ = $f_{on}|_{x_i=p} \cap f_{on}|_{x_i\neq p}$;
$I_{dc}$ = $f_{dc}|_{x_i=p} \cap f_{dc}|_{x_i\neq p}$;
$f_{on}^{(=)}$ = $f_{on}|_{x_i=p} \setminus I_{on}$;
$f_{on}^{(\neq)}$ = $f_{on}|_{x_i\neq p} \setminus I_{on}$;
$f_{dc}^{(=)}$ = $f_{dc}|_{x_i=p} \cup I_{on}$;
$f_{dc}^{(\neq)}$ = $f_{dc}|_{x_i\neq p} \cup I_{on}$;
$MinSOP^{(=)}$ = $OptSOP(f_{on}^{(=)}, f_{dc}^{(=)})$; // optimal SOP for $f^{(=)}$
$MinSOP^{(\neq)}$ = $OptSOP(f_{on}^{(\neq)}, f_{dc}^{(\neq)})$; // optimal SOP for $f^{(\neq)}$
$MinSOP^{I}$ = $OptSOP(I_{on}, I_{dc})$; // optimal SOP for $I = (I_{on}, I_{dc})$
$MinSOP^{p}$ = $OptSOP(p, \emptyset)$; // optimal SOP for $p$
$P$-circuit = $(\overline{x}_i \oplus MinSOP^p)MinSOP^{(=)}+$
$(x_i \oplus MinSOP^p)MinSOP^{(\neq)} + MinSOP^I$
**return** $P$-circuit

Fig. 5. Algorithm for the optimization of P-circuits with intersection.

| Without Intersection | | | With Intersection | | | |
|---|---|---|---|---|---|---|
| **VAR** | **XOR** | **AND** | **Constant** | **VAR** | **XOR** | **AND** |
| 32% | 22% | 28% | 79% | 59% | 50% | 58% |

TABLE I
PERCENTAGE OF P-CIRCUITS, OVER ALL THE BENCHMARKS, HAVING SMALLER AREA THAN THE P-CIRCUITS BASED ON SHANNON DECOMPOSITION.

### A. Synthesis Algorithms

We now describe two algorithms for computing optimal P-circuits, without and with intersection. Both algorithms can be implemented using OBDD data structures [7] for Boolean function manipulation, and a classical SOP minimization procedure (e.g., ESPRESSO [5]). Figures 4 and 5 show the algorithms for the optimization of a P-circuit without and with intersection, respectively. The complexity of the algorithms depends from two factors: the complexity of OBDD operations, which is polynomial in the size of the OBDDs for the operands $f$ and $p$, and the complexity of SOP minimization. Exact SOP minimization is exponential in time, but efficient heuristics are available (i.e., ESPRESSO in the heuristic mode).

## IV. EXPERIMENTAL RESULTS

In this section we report experimental results for the two decomposition methods described in the previous sections. The methods have been implemented in C, using the CUDD library for OBDDs to represent Boolean functions. The experiments have been run on a Pentium 1.6GHz CPU with 1 GByte of main memory. The benchmarks are taken from LGSynth93 [14]. We report in the following a significant subset of the functions as representative indicators of our experiments. In order to evaluate the performances of these new synthesis methods, we compare the area of different versions of P-circuits with P-circuits based on the classical Shannon decomposition, i.e., P-circuits representing $(x_i, 0)$-decomposition without intersection (referred as **Shannon** in Table II). In particular we report P-circuits for the following choices of the projection function $p$: 1) $p = 0$, decomposi-

| | Without Intersection | | | | | | | | With Intersection | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Shannon | | VAR | | XOR | | AND | | Constant | | VAR | | XOR | | AND | |
| **Bench** | **Area** | **Time** | **Area** | **Time** | **Area** | **Time** | **Area** | **Time** | **Area** | **Time** | **Area** | **Time** | **Area** | **Time** | **Area** | **Time** |
| **add6** | 908 | 0.65 | **507** | 5.19 | 669 | 24.58 | 524 | 90.84 | 672 | 0.51 | 814 | 4.44 | 759 | 23.70 | 651 | 80.93 |
| **alu2** | 355 | 0.45 | 382 | 0.79 | 416 | 3.60 | 356 | 12.93 | **283** | 0.18 | 308 | 1.03 | 310 | 4.72 | 298 | 16.79 |
| **amd** | 1620 | 0.17 | 1694 | 1.24 | 1800 | 8.65 | 1747 | 30.31 | **1012** | 0.12 | 1085 | 1.55 | 1202 | 10.88 | 1180 | 37.65 |
| **b12** | 227 | 0.11 | 306 | 0.55 | 401 | 4.27 | 340 | 15.90 | **199** | 0.18 | 248 | 0.65 | 367 | 5.25 | 292 | 18.13 |
| **dk17** | 263 | 0.10 | 250 | 0.38 | 291 | 1.82 | **230** | 6.85 | 263 | 0.06 | 250 | 0.46 | 291 | 1.99 | **230** | 7.21 |
| **ex7** | 436 | 0.12 | 463 | 1.04 | 492 | 8.30 | 472 | 29.07 | **327** | 0.09 | 360 | 1.56 | 393 | 10.39 | 364 | 38.51 |
| **f51m** | 497 | 0.09 | 706 | 0.21 | 640 | 0.64 | 528 | 2.24 | **277** | 0.09 | 290 | 0.28 | 314 | 0.85 | 323 | 4.11 |
| **m181** | 227 | 0.42 | 308 | 0.58 | 404 | 4.44 | 341 | 16.39 | **199** | 0.08 | 252 | 0.68 | 341 | 6.65 | 288 | 29.20 |
| **max1024** | 2534 | 0.34 | **2511** | 1.97 | 2973 | 8.74 | 2642 | 30.72 | 2980 | 0.25 | 3043 | 2.12 | 2977 | 10.13 | 2829 | 34.28 |
| **max46** | 297 | 0.03 | 301 | 0.14 | 291 | 0.41 | **286** | 1.75 | 307 | 0.02 | 289 | 0.10 | 294 | 0.46 | 293 | 2.14 |
| **mp2d** | 355 | 0.09 | 435 | 0.61 | 508 | 4.47 | 455 | 16.49 | **276** | 0.16 | 357 | 0.75 | 411 | 6.82 | 359 | 22.56 |
| **p1** | 724 | 0.18 | 781 | 0.96 | 821 | 3.07 | 842 | 10.77 | **711** | 0.20 | 777 | 1.18 | 847 | 3.74 | 818 | 13.66 |
| **root** | 416 | 0.05 | 594 | 0.14 | 393 | 0.50 | **385** | 1.91 | 417 | 0.02 | 536 | 0.17 | 602 | 0.55 | 446 | 1.94 |
| **spla** | **2239** | 0.79 | 2570 | 7.88 | 3142 | 74.99 | 2886 | 273.75 | 2428 | 0.73 | 2761 | 8.82 | 3249 | 84.11 | 3107 | 336.30 |
| **sym10** | 559 | 0.30 | 414 | 0.64 | **309** | 2.92 | 416 | 14.31 | 568 | 0.27 | 529 | 0.96 | 551 | 3.90 | 554 | 16.81 |
| **t1** | 905 | 0.83 | 951 | 3.52 | 1186 | 41.02 | 982 | 155.28 | **463** | 0.61 | 510 | 6.06 | 655 | 78.07 | 585 | 277.38 |
| **t2** | 501 | 0.06 | 589 | 0.65 | 686 | 6.37 | 618 | 22.95 | **358** | 0.05 | 406 | 0.88 | 469 | 9.80 | 416 | 22.33 |
| **test1** | **1465** | 0.34 | 1488 | 1.06 | 1565 | 3.13 | 1510 | 11.43 | 1535 | 0.25 | 1645 | 1.18 | 1583 | 3.66 | 1484 | 13.50 |
| **tial** | 3430 | 5.33 | 3337 | 23.68 | 4062 | 159.84 | 3823 | 557.19 | 3368 | 3.29 | **3319** | 31.12 | 3952 | 215.08 | 3827 | 741.85 |
| **vtx1** | 430 | 0.09 | 445 | 1.89 | 501 | 32.57 | 585 | 107.74 | **390** | 0.14 | 499 | 3.03 | 486 | 50.57 | 524 | 171.45 |
| **x9dn** | 530 | 0.22 | 528 | 2.23 | 595 | 30.62 | 548 | 116.64 | 412 | 0.19 | **401** | 4.26 | 457 | 57.18 | 418 | 217.77 |
| **Z5xp1** | 479 | 0.08 | 593 | 0.12 | 743 | 0.33 | 547 | 1.24 | 324 | 0.03 | 369 | 0.19 | 441 | 0.41 | **302** | 1.29 |
| **Z9sym** | 464 | 0.17 | 288 | 0.33 | **267** | 1.15 | 371 | 6.07 | 379 | 0.17 | 391 | 0.64 | 395 | 1.68 | 393 | 9.28 |

TABLE II

COMPARISON OF AREA AND SYNTHESIS TIME OF P-CIRCUITS REPRESENTING $(x_0, p)$-DECOMPOSITION FORMS FOR DIFFERENT CHOICES OF THE PROJECTION FUNCTION $p$.

tion with intersection (referred as **Constant** in Table II); 2) $p = x_j$, decomposition without and with intersection (**VAR** in Table II); 3) $p = x_j \oplus x_k$, decomposition without and with intersection (**XOR** in Table II); 4) $p = x_j x_k$, decomposition without and with intersection, choosing the complementations of variables giving the best area (**AND** in Table II). After the projection, all SOP components of the $P$-circuits have been synthesized with multi-output synthesis using ESPRESSO in the heuristic mode. Finally, to evaluate the obtained circuits, we ran our benchmarks using the SIS system with the MCNC library for technology mapping and the SIS command `map -W -f 3 -s`. In Table II we compare mapped area and synthesis time (in seconds) of $P$-circuits representing decomposition forms without and with intersection for a subset of the benchmarks. Due to space limitation, the results shown refer only to decompositions with respect to the first input variable, $x_0$, of each benchmark. In all the experiments we considered decompositions with respect to each input variable of each benchmark. The results, summarized in Tab. I, are quite promising. These results support the conclusion that decompositions with intersection provide better results, and that the best choice for the projection function $p$ is the simplest: $p = 0$. Moreover synthesis for $p = 0$ with intersection is very efficient in computational time. When $p$ is not constant, the synthesis is time-consuming, since the algorithm must choose the best combination of variables to be utilized for $p$. Altogether, only $14\%$ of the P-circuits achieve the smallest area when implemented based on the classical Shannon decomposition.

## V. CONCLUSION

In conclusion, we presented a new method to decompose Boolean functions via complex cofactoring. Experimental results show that this decomposition yields more compact circuits than those obtained with Shannon decomposition. This decomposition has the advantage to minimize the dynamic power dissipation with respect to a known input signal switching with high frequency. In future work, we plan to verify this property with a transistor level simulation of the circuits. Widely used data structures (i.e., OBDDs) are based on Shannon decomposition. Thus a future development of this work could be the definition of new data structures based on the proposed decomposition.

## REFERENCES


[1] L. Benini and G. D. Micheli, "Logic Synthesis for Low Power," in *Logic Synthesis and Verification*, S. Hassoun and T. Sasao, Eds. Kluwer Academic Publishers, 2002, pp. 197–223.

[2] 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.

[3] ——, "The Optimization of kEP-SOPs: Computational Complexity, Approximability and Experiments," *ACM Trans. on Design Automation of Electronic Systems*, vol. 13, no. 2, 2008.

[4] J. C. Bioch, "The complexity of modular decomposition of boolean functions," *Discrete Applied Mathematics*, vol. 149, no. 1-3, pp. 1–13, 2005.

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

[6] F. Brown, *Boolean Reasoning*. Kluwer Academic Publishers, 1990.

[7] R. Bryant, "Graph Based Algorithm for Boolean Function Manipulation," *IEEE Trans. on Computers*, vol. 35, no. 9, pp. 667–691, 1986.

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

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

[10] L. Lavagno, P. C. McGeer, A. Saldanha, and A. L. Sangiovanni-Vincentelli, "Timed Shannon Circuits: a Power-Efficient Design Style and Synthesis Tool," in *DAC*, 1995, pp. 254–260.

[11] P. C. McGeer, R. K. Brayton, A. L. Sangiovanni-Vincentelli, and S. Sahni, "Performance Enhancement through the Generalized Bypass Transform," in *ICCAD*, 1991, pp. 184–187.

[12] M. Pedram, "Power Estimation and Optimization at the logic level," *Int. Journal of High Speed Electronics and Systems*, vol. 5, no. 2, pp. 179–202, 1994.

[13] C. Soviani, O. Tardieu, and S. A. Edwards, "Optimizing Sequential Cycles through Shannon Decomposition and retiming," in *DATE*, 2006, pp. 1085–1090.

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