

Chapter 2

Logic Synthesis by Signal-Driven Decomposition

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

Abstract This chapter investigates some restructuring techniques based on decomposition and 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 order to reduce the power consumption of the circuit, the number of gates that are affected by the switching activity of critical signals is maintained constant. This chapter describes new types of factorization that extend Shannon cofactoring and are based on projection functions that change the Hamming distance among the original minterms to favor logic minimization of the component blocks. Moreover, the proposed algorithms generate and exploit don't care conditions in order to further minimize the final circuit. The related implementations, called P-circuits, show experimentally promising results in area with respect to classical Shannon cofactoring.

2.1 Introduction

In recent years, power has become an important factor during the design phase. This trend is primarily due to the remarkable growth of personal computing devices, embedded systems, and wireless communications systems that demand high-speed computation and complex functionality with low power consumption. In these applications, average power consumption is a critical design concern.

Low-power design methodologies must consider power at all stages of the design process. At the logic synthesis level, logic transformations proved to be an effective technique to reduce power consumption by restructuring a mapped circuit through permissible signal substitution or perturbation [1]. A fundamental step in VLSI design is logic synthesis of high-quality circuits matching a given specification. The

A. Bernasconi (✉)

Department of Computer Science, Università di Pisa, Pisa, Italy

e-mail: annab@di.unipi.it

Based on [5], pp.1464–1469, 20–24 April 2009 © [2009] IEEE.

performance of the circuit can be expressed in terms of several factors, such as area, delay, power consumption, and testability properties. Unfortunately, these factors often contradict each other, in the sense that it is very difficult to design circuits that guarantee very good performances with respect to all of them. In fact, power consumption is often studied as a single minimization objective without taking into account important factors such as area and delay.

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 [11, 14, 16] for performance and [1, 13, 15] 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 aim 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 a Boolean function with respect to a chosen splitting variable; however, when applying Shannon cofactoring, the drawback is that too much area redundancy might be introduced because large cubes are split between two disjoint subspaces, whereas no new cube merging will take place as the Hamming distance among the projected minterms do not change.

In this chapter we investigate thoroughly the more general factorization introduced in [5], a decomposition that extends straightforward Shannon cofactoring; instead of cofactoring a function f only with respect to single variables as Shannon does, we cofactor with respect to more complex functions, expanding f with respect to the orthogonal basis $\bar{x}_i \oplus p$ (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 study different functions $p(x)$ trading-off quality vs. computation time. Our factorizations modify the Hamming distance among the on-set minterms, so that more logic minimization may be performed on the projection of f onto the two disjoint subspaces $x_i = p$ and $x_i \neq p$, while signals are moved in the circuit closer to the output. We then introduce and study another form of decomposition, called *decomposition with intersection*, where a function f is projected onto three overlapping subspaces of the Boolean

space $\{0, 1\}^n$ in order to favor area minimization avoiding cube fragmentation (e.g., cube splitting for the cubes intersecting both subspaces $x_i = p$ and $x_i \neq p$). More precisely, we partition the on-set minterms of f into three sets: $f|_{x_i=p}$ and $f|_{x_i \neq p}$, representing the projections of f onto the two disjoint subspaces $x_i = p$ and $x_i \neq p$, and a third set $I = f|_{x_i=p} \cap f|_{x_i \neq p}$, which contains all minterms of f whose projections onto $x_i = p$ and $x_i \neq p$ are identical. Observe that each point in I corresponds to two different points of f that could be merged in a cube, but are split into the two spaces $x_i = p$ and $x_i \neq p$. Thus, we can avoid cube fragmentation keeping the points in I unprojected. Moreover, given that the points in the intersection I must be covered, we can project them as *don't cares* in the two spaces $f|_{x_i=p}$ and $f|_{x_i \neq p}$ to ease the minimization of $f|_{x_i=p} \setminus I$ and $f|_{x_i \neq p} \setminus I$. Observe that, while classical don't care sets are specified by the user or are derived from the surrounding environment, our don't cares are dynamically constructed during the synthesis phase.

The circuits synthesized according to these decompositions are called *Projected Circuits*, or *P-circuits*, without and with intersection. We provide minimization algorithms to compute optimal P-circuits and argue how augmenting P-circuits with at most a pair of multiplexers guarantees full testability under the single stuck-at-fault model. We also show that the proposed decomposition technique can be extended and applied to move all critical signals, and not just one, toward the output, still avoiding losses in area.

The chapter is organized as follows. Section 2.2 describes the new theory of decomposition based on generalized cofactoring, which is applied in Section 2.3 to the synthesis of Boolean functions as *P-circuits*. Section 2.4 extends the decomposition from single to multiple variables. Experiments and conclusions are reported in Sections 2.5 and 2.6, respectively.

2.2 Decomposition Methods

How to decompose Boolean functions is an ongoing research area to explore alternative logic implementations. A technique to decompose Boolean functions is based on expanding them according to an orthogonal basis (see, for example [8], section 3.15), as in the following definition, where a function f is decomposed according to the basis (g, \bar{g}) .

Definition 2.1 Let $f = (f_{\text{on}}, f_{\text{dc}}, f_{\text{off}})$ 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 $\text{co}(f, g) = (f_{\text{on}} \cdot g, f_{\text{dc}} + \bar{g}, f_{\text{off}} \cdot g)$.

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 $\text{co}(f, g)$. This flexibility can be exploited according to the purpose of the expansion. For instance, when $g = x_i$, we have $\text{co}(f, x_i) = (f_{\text{on}} \cdot x_i, f_{\text{dc}} + \bar{x}_i, f_{\text{off}} \cdot x_i)$. Notice that the well-known Shannon cofactor $f_{x_i} = f(x_1, \dots, (x_i = 1), \dots, x_n)$ is a

completely specified function contained in $\text{co}(f, x_i) = (f_{\text{on}} \cdot x_i, f_{\text{dc}} + \bar{x}_i, f_{\text{off}} \cdot x_i)$ (since $f_{\text{on}} \cdot x_i \subseteq f_{x_i} \subseteq f_{\text{on}} \cdot x_i + f_{\text{dc}} + \bar{x}_i = f_{\text{on}} + f_{\text{dc}} + \bar{x}_i$); moreover, f_{x_i} is the unique cover of $\text{co}(f, x_i)$ independent from 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 of the logic blocks so obtained. Let $f(X) = (f_{\text{on}}(X), f_{\text{dc}}(X), f_{\text{off}}(X))$ be an incompletely specified function depending on the set $X = \{x_1, x_2, \dots, x_n\}$ 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 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 $(\bar{x}_i \oplus p)$ 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 [12] and [6], is defined as follows.

Definition 2.2 Let $f(X)$ be an incompletely specified function, $x_i \in X$, and $p(X^{(i)})$ be a completely specified function. The (x_i, p) -decomposition of f is the algebraic expression

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

First of all we observe that each minterm of f is projected onto one and only one subset. Indeed, let $m = m_1 m_2 \dots m_n$ be a minterm of f ; if $m_i = p(m_1, \dots, m_{i-1}, m_{i+1}, \dots, m_n)$, 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. 2.1 with $f_{\text{on}} = \{0000, 0001, 0010, 0101, 1001, 1010, 1100, 1101\}$ and $f_{\text{dc}} = \{0111\}$. Let p be the simple Boolean function x_2 , and x_i be x_1 . The Boolean space $\{0, 1\}^4$ can be partitioned into 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 $f_{\text{on}}|_{x_1=x_2} = \{000, 001, 010, 100, 101\}$, $f_{\text{dc}}|_{x_1=x_2} = \emptyset$, and $f_{\text{on}}|_{x_1 \neq x_2} = \{101, 001, 010\}$, $f_{\text{dc}}|_{x_1 \neq x_2} = \{111\}$.

Second, observe that these projections do not preserve the Hamming distance 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 distance may change could be useful when f is

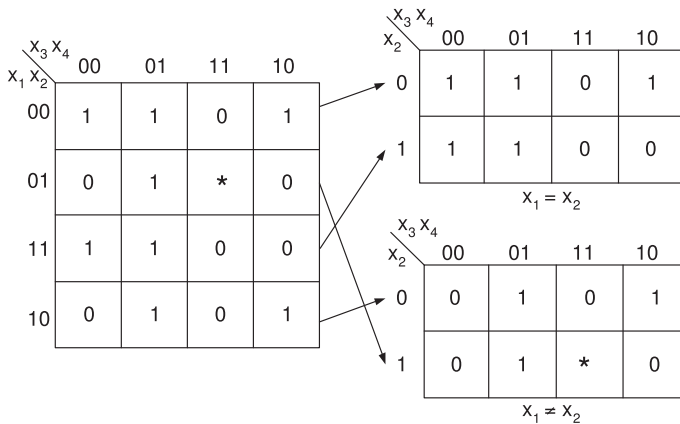


Fig. 2.1 An example of projection of the incompletely specified function f onto the spaces $x_1 = x_2$ and $x_1 \neq x_2$

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. 2.1. 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 $f_{\text{on}}|_{x_1=x_2}$ (i.e., 000 and 100, respectively) have Hamming distance equal to 1, and they form the cube $\bar{x}_3\bar{x}_4$.

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 \bar{x}_3x_4 of function f_{on} is split into the two sets $x_1 = x_2$ and $x_1 \neq x_2$ forming a cube in $f_{\text{on}}|_{x_1=x_2}$ and one in $f_{\text{on}}|_{x_1 \neq x_2}$, as shown on the right side of Fig. 2.1.

Observe that the cubes that end up to be split may contain pairs of minterms, whose projections onto the two sets are identical. In our example, \bar{x}_3x_4 is the cube corresponding to the points {0001, 0101, 1001, 1101}, where 0001 and 1101 are projected onto $f_{\text{on}}|_{x_1=x_2}$ and become 001 and 101, respectively, and 0101 and 1001 are projected onto $f_{\text{on}}|_{x_1 \neq x_2}$ 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 $I_{\text{on}} = f_{\text{on}}|_{x_1=x_2} \cap f_{\text{on}}|_{x_1 \neq x_2} = \{001, 010, 101\}$, and $I_{\text{dc}} = \emptyset$.

In order to overcome the splitting of some 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 $f = (\bar{x}_i \oplus p)(f|_{x_i=p} \setminus I) + (x_i \oplus p)(f|_{x_i \neq p} \setminus I) + I$.

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 contained in I . Therefore keeping them both in I and in the projected subfunctions would be useless and expensive. In our example, since $I_{\text{on}} = \{001, 010, 101\}$, in $f_{\text{on}}|_{x_1=x_2}$ the points 001 and 101

are useful for forming, together with 000 and 100, the cube \bar{x}_3 ; instead the point 010 is useless and must be covered with an additional cube. The solution to 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 points. We therefore propose the following more refined second decomposition technique, using the notation $h = (h_{\text{on}}, h_{\text{dc}})$ for an incompletely specified function h and its on-set h_{on} and don't care set h_{dc} .

Definition 2.3 Let $f(X)$ be an incompletely specified function, $x_i \in X$, and $p(X^{(i)})$ be a completely specified function. The (x_i, p) -decomposition with intersection of $f = (f_{\text{on}}, f_{\text{dc}})$ is the algebraic expression

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

where

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

with $I_{\text{on}} = f_{\text{on}}|_{x_i=p} \cap f_{\text{on}}|_{x_i \neq p}$ and $I_{\text{dc}} = f_{\text{dc}}|_{x_i=p} \cap f_{\text{dc}}|_{x_i \neq p}$.

For our example, the projections of f become $\tilde{f}|_{x_1=x_2} = (f_{\text{on}}|_{x_1=x_2} \setminus I_{\text{on}}, f_{\text{dc}}|_{x_1=x_2} \cup I_{\text{on}}) = (\{000, 100\}, \{001, 010, 101\})$ and $\tilde{f}|_{x_1 \neq x_2} = (f_{\text{on}}|_{x_1 \neq x_2} \setminus I_{\text{on}}, f_{\text{dc}}|_{x_1 \neq x_2} \cup I_{\text{on}}) = (\emptyset, \{111\} \cup \{001, 010, 101\})$. The Karnaugh maps of this decomposition are shown in Fig. 2.2.

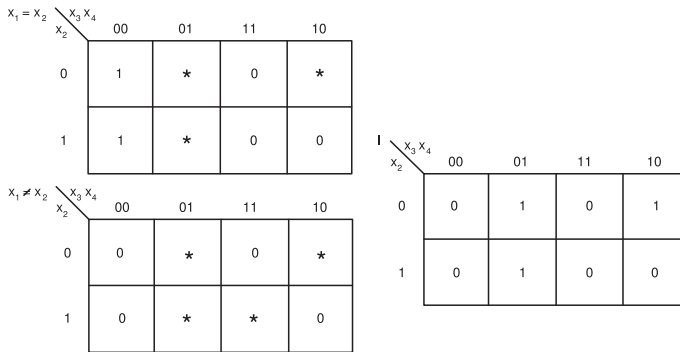


Fig. 2.2 An example of projection *with intersection* of the function f of Fig. 2.1 onto the spaces $x_1 = x_2$, $x_1 \neq x_2$, and I

Observe that, xing the function p and a variable x , these decompositions are canonical. We now study these decomposition methods for some choices of the function p .

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: $f = \bar{x}_i f|_{x_i=0} + x_i f|_{x_i=1}$. Recall that $(\bar{x}_i \oplus 0)$ is equivalent to \bar{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:

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

Observe that in this particular case, the set I is

$$I = f(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n) \cap f(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n).$$

This implies the following property.

Proposition 2.1 *The characteristic function χ_I of I is the biggest subfunction of f that does not depend on x_i .*

Proof Let χ_1, \dots, χ_k be the subfunctions of f that do not depend on x_i , and let χ be their union, i.e., $\chi = \chi_1 + \chi_2 + \dots + \chi_k$. Observe that χ is still a subfunction of f and it does not depend on x_i . Therefore χ is the biggest subfunction that does not depend on x_i . We must show that $\chi = \chi_I$. First note that χ_I is one of the functions χ_1, \dots, χ_k . Suppose $\chi_I = \chi_j$, with $1 \leq j \leq k$. By construction, χ_j is a subfunction of χ . On the other hand, if $\chi(X) = 1$, then there exists an index h such that $\chi_h(X) = 1$. Since χ_h does not depend on x_i , we have

$$\chi_h(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n) = \chi_h(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n) = 1.$$

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

$$f(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n) = f(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n) = 1.$$

This implies that

$$\chi_j = f(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n) \cap f(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n) = 1,$$

which means that χ is a subfunction of χ_j . As $\chi_j = \chi_I$, we nally have that $\chi = \chi_I$.

Note that if χ_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 sum of products. Consider a minimal sum of products $SOP(f)$ for the function f . The number of products in $SOP(f)$ is always less than 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 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 2.1 *An $(x_i, 0)$ -decomposition with intersection for a Boolean 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 \bar{x}_i gives a minimal SOP for $\tilde{f}|_{x_i=0}$, while all remaining products, that do not depend on x_i or \bar{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 distance among the minterms, so that no bigger cubes can 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 ℓ_{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 ℓ_I is preserved. Suppose that r products in $SOP(f)$ contain x_i , and let ℓ_{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 \bar{x}_i , and $\ell_{\bar{x}_i}$ is their overall number of literals, the SOP for $\tilde{f}|_{x_i=0}$ contains $\ell_{\bar{x}_i} - s$ literals. Thus, the $(x_i, 0)$ -decomposition with intersection contains exactly $\ell_I + \ell_{x_i} - r + \ell_{\bar{x}_i} - s + 2 = \ell_{SOP} - r - s + 2$ literals, where the two additional literals represent the characteristic functions of the projection sets.

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 expression $f = (\bar{x}_i \oplus x_j)f|_{x_i=x_j} + (x_i \oplus x_j)f|_{x_i \neq x_j}$, while the (x_i, x_j) -decomposition with intersection is given by $f = (\bar{x}_i \oplus x_j)\tilde{f}|_{x_i=x_j} + (x_i \oplus x_j)\tilde{f}|_{x_i \neq x_j} + I$, where

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

with $I_{\text{on}} = f_{\text{on}}|_{x_i=x_j} \cap f_{\text{on}}|_{x_i \neq x_j}$ and $I_{\text{dc}} = f_{\text{dc}}|_{x_i=x_j} \cap f_{\text{dc}}|_{x_i \neq x_j}$. 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 the cost of an EP-SOP with remainder.

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

2.3 P-Circuits

We now show how the decomposition methods described in Section 2.2 can be applied to the logic synthesis of Boolean functions. The idea for synthesis is simply

to construct a network for f using as building blocks 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 . 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 nal OR gate.

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. 2.3. 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 Figs. 2.1 and 2.2 has minimal SOP form $\bar{x}_1\bar{x}_2\bar{x}_3 + x_1x_2\bar{x}_3 + \bar{x}_3x_4 + \bar{x}_2x_3\bar{x}_4$, while its corresponding optimal P-circuit is $(\bar{x}_1 \oplus x_2)\bar{x}_3 + \bar{x}_3x_4 + \bar{x}_2x_3\bar{x}_4$.

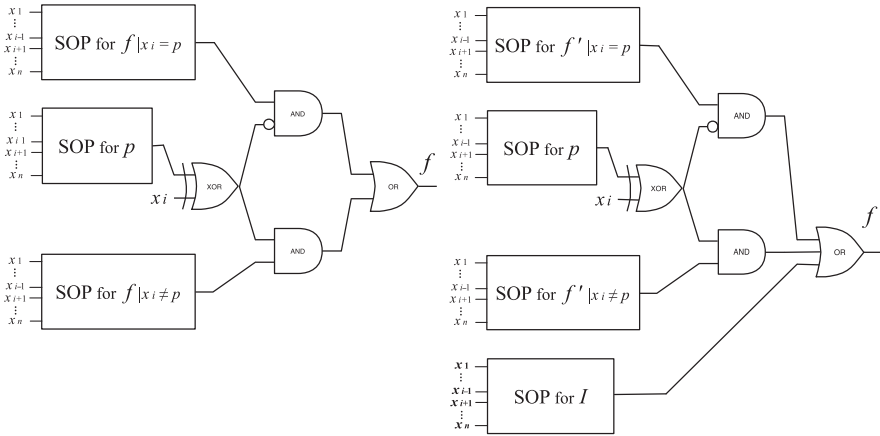


Fig. 2.3 P-circuit (left) and P-circuit with intersection (right)

The number of logic levels in a P-circuit varies from four to five: it is equal to four when 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. 2.3 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 this decomposition when more than one variable switches with high frequency as shown in Section 2.4.

2.3.1 Synthesis Algorithms

We now describe two algorithms for computing optimal P-circuits, with and without intersection. Both algorithms can be implemented using OBDD data structures [9] for Boolean function manipulation and a classical SOP minimization procedure (e.g., ESPRESSO [7]).

The heuristic that `nds` a P-circuit with intersection (in Fig. 2.4) `rst` computes the projections of the on-set and dc-set of f onto $x_i \neq p$ and $x_i = p$ and their intersections I_{on} and I_{dc} . The on-set of the intersection, I_{on} , is subtracted from the two on-sets ($f_{on}|_{x_i \neq p}$ and $f_{on}|_{x_i = p}$), and it is inserted in the two dc-sets ($f_{dc}|_{x_i \neq p}$ and $f_{dc}|_{x_i = p}$). This step guarantees that only the useful points of the intersection are covered in the SOP form of $f|_{x_i \neq p}$ and $f|_{x_i = p}$. Finally, the algorithm synthesizes the projected functions and the intersection with a SOP minimizer, and a P-circuit is then returned. The algorithm that computes a P-circuit without intersection is similar to the former but does not take into account the intersection, as shown in Fig. 2.5.

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 ,

$$I_{on} = f_{on}|_{x_i = p} \cap f_{on}|_{x_i \neq p};$$

$$I_{dc} = f_{dc}|_{x_i = p} \cap f_{dc}|_{x_i \neq p};$$

$$f_{on}^{(=)} = f_{on}|_{x_i = p} \setminus I_{on};$$

$$f_{on}^{(\neq)} = f_{on}|_{x_i \neq p} \setminus I_{on};$$

$$f_{dc}^{(=)} = f_{dc}|_{x_i = p} \cup I_{on};$$

$$f_{dc}^{(\neq)} = f_{dc}|_{x_i \neq p} \cup I_{on};$$

$$MinSOP^{(=)} = OptSOP(f_{on}^{(=)}, f_{dc}^{(=)}); // \text{optimal SOP for } f^{(=)}$$

$$MinSOP^{(\neq)} = OptSOP(f_{on}^{(\neq)}, f_{dc}^{(\neq)}); // \text{optimal SOP for } f^{(\neq)}$$

$$MinSOP^I = OptSOP(I_{on}, I_{dc}); // \text{optimal SOP for } I = (I_{on}, I_{dc})$$

$$MinSOP^p = OptSOP(p, \emptyset); // \text{optimal SOP for } p$$

$$P\text{-circuit} = (\bar{x}_i \oplus MinSOP^p)MinSOP^{(=)} + (x_i \oplus MinSOP^p)MinSOP^{(\neq)} + MinSOP^I$$

return P-circuit

Fig. 2.4 Algorithm for the optimization of P-circuits with intersection

The complexity of the algorithms depends on 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 superexponential in time, but efficient heuristics are available (i.e., ESPRESSO in the heuristic mode).

The algorithms compute correct covers as proved in the following theorem.

Theorem 2.2 (Correctness) *Algorithms in Figs. 2.4 and 2.5 compute a P-circuit C that covers the input function f .*

Proof Overloading the notation, let us denote with C the Boolean function that corresponds to the circuit C . In both cases we have to show that $f_{on} \subseteq C \subseteq f_{on} \cup$

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 ,

$$f_{on}^{(=)} = f_{on} \mid_{x_i=p};$$

$$f_{on}^{(\neq)} = f_{on} \mid_{x_i \neq p};$$

$$f_{dc}^{(=)} = f_{dc} \mid_{x_i=p};$$

$$f_{dc}^{(\neq)} = f_{dc} \mid_{x_i \neq p};$$

$$MinSOP^{(=)} = OptSOP(f_{on}^{(=)}, f_{dc}^{(=)}); // \text{optimal SOP for } f^{(=)}$$

$$MinSOP^{(\neq)} = OptSOP(f_{on}^{(\neq)}, f_{dc}^{(\neq)}); // \text{optimal SOP for } f^{(\neq)}$$

$$MinSOP^p = OptSOP(p, \emptyset); // \text{optimal SOP for } p$$

$$P\text{-circuit} = (\bar{x}_i \oplus MinSOP^p) MinSOP^{(=)} + (x_i \oplus MinSOP^p) MinSOP^{(\neq)}$$

return P -circuit

Fig. 2.5 Algorithm for the optimization of P -circuits without intersection

f_{dc} . We first consider the algorithm in Fig. 2.4 for the (x_i, p) -decomposition with intersection of f that outputs the circuit C .

Let $y \in f_{on}$ be the minterm $y = y_1, y_2, \dots, y_n$, we show that $y \in C$. We have two cases: (1) if $y_i = p(y_1, \dots, y_{i-1}, y_{i+1}, \dots, y_n)$ we have that, for the synthesis algorithm, y is covered by $(\bar{x}_i \oplus MinSOP^p) MinSOP^{(=)}$ or by $MinSOP^I$; (2) if $y_i \neq p(y_1, \dots, y_{i-1}, y_{i+1}, \dots, y_n)$ we have that, for the synthesis algorithm, y is covered by $(x_i \oplus MinSOP^p) MinSOP^{(\neq)}$ or by $MinSOP^I$. Thus y is in C .

From the other side, let y be a point of C , we have to show that y is also in $f_{on} \cup f_{dc}$. We have two cases: (1) if y is covered by $MinSOP^I$, then y is in both $f|_{x_i=p}$ and $f|_{x_i \neq p}$, and – given that $MinSOP^I$ is synthesized with ESPRESSO – y is in $f_{on} \cup f_{dc}$; (2) if y is not covered by $MinSOP^I$, then it is covered by $(\bar{x}_i \oplus MinSOP^p) MinSOP^{(=)}$ or $(x_i \oplus MinSOP^p) MinSOP^{(\neq)}$. In both cases y must be in a projected space that is synthesized with ESPRESSO.

Consider now the algorithm in Fig. 2.5 for the computation of a (x_i, p) -decomposition without intersection. In this case the intersection is not computed thus each point of the function is simply projected onto one of the projecting spaces. The thesis immediately follows.

Considering the Stuck-At Fault Model (SAFM), we now briefly discuss the testability of P -circuits in the case where p is a constant function (i.e., $p = 0$). A fault in the Stuck-At Fault Model is exactly one input or one output pin of a node in a combinatorial logic circuit C to constant value (0 or 1) independently of the values applied to the primary inputs of the circuit. A node v in C is called *fully testable*, if there does not exist a redundant fault with fault location v . If all nodes in C are fully testable, then C is *fully testable*.

Theorem 2.3 *From a given P -circuit we can obtain a circuit that is fully testable in the SAFM by adding at most two more inputs and two multiplexers.*

Proof The proof of this theorem follows directly from the testability proof in [4] where the decomposed functions are synthesized in 2SPP form [10] instead of SOP forms. 2SPP expressions are direct generalizations of SOP forms where we can

use EXORs of two literals instead of simple literals as inputs to the AND gates. We note that the testability theorem in [4] still holds for any form that is prime and irredundant. Since the SOP forms that we use for the synthesis of P-circuits have this property, the thesis immediately follows. In the case of P-circuits with intersection, the testable circuit that we obtain contains two MUXs before the inputs of the final OR gate. One is between the outputs of the decomposed parts and the second is after the output of the intersection. The MUXs are used to test the three single blocks of the circuit separately. In the case of P-circuits without intersection, just one MUX (between the outputs of the decomposed parts before the OR gate) is needed. In this case the proof still holds since we can consider a P-circuit without intersection as a special case of a P-circuit with intersection where the intersection is empty.

2.4 Multivariable Decomposition

In this section we show how our new decomposition technique can be extended from one to more variables, so that it could be applied to move all critical signals, and not just one, toward the output, still avoiding losses in area. A first naive solution for extending our technique could be to apply recursively the decompositions, i.e.,

- compute a decomposition of the function under study with respect to the variable with highest switching frequency among the variables in the set $X = \{x_1, \dots, x_n\}$, say x_i ;
- apply the same procedure to the functions $f|_{x_i=p}$ and $f|_{x_i \neq p}$ or to the functions $\tilde{f}|_{x_i=p}$, $\tilde{f}|_{x_i \neq p}$ and I (in case of decomposition with intersection), with respect to the variable with highest switching frequency in the set $X \setminus \{x_i\}$;
- if needed, recursively repeat the same procedure on the subfunctions derived in the previous decomposition step.

Observe that with this naive approach, the number of levels increases by three at each decomposition step. Moreover, the critical signals have different distances from the final output gate and their switching activity affects different portions of the circuit. In particular, the first variable selected is the one closest to the output, affecting only the last four gates.

In order to keep the number of levels constant and independent from the number of decomposition steps and to move all critical signals equally close to the output, so that the number of gates affected by their switching activity can be maintained constant, a different solution should be adopted. This solution is based on a “parallel decomposition” in which the points of the function are simultaneously partitioned and projected onto the 2^k subspaces defined by the k critical variables. For ease of exposition, we explain in detail only the case $k = 2$. The general case $k > 2$ can be easily derived from it, but at the expense of a quite heavy notation.

Definition 2.4 Let $f(X)$ be an incompletely specified Boolean function, $x_i, x_j \in X$, and p_i and p_j be two completely specified Boolean functions depending on all variables in $X \setminus \{x_i, x_j\}$. The $[(x_i, p_i), (x_j, p_j)]$ -decomposition of f is the algebraic expression

$$f = (\bar{x}_i \oplus p_i)(\bar{x}_j \oplus p_j)f|_{\substack{x_i=p_i \\ x_j=p_j}} + (\bar{x}_i \oplus p_i)(x_j \oplus p_j)f|_{\substack{x_i=p_i \\ x_j \neq p_j}} \\ + (x_i \oplus p_i)(\bar{x}_j \oplus p_j)f|_{\substack{x_i \neq p_i \\ x_j=p_j}} + (x_i \oplus p_i)(x_j \oplus p_j)f|_{\substack{x_i \neq p_i \\ x_j \neq p_j}}.$$

The extension of the notion of decomposition with intersection will require the introduction of ve new subfunctions representing the overall intersection among the four projections of f , and the four intersections between the projections of $f|_{x_i=p_i}$ and $f|_{x_i \neq p_i}$ w.r.t. x_j , and between the projections of $f|_{x_j=p_j}$ and $f|_{x_j \neq p_j}$ w.r.t. x_i , respectively. As for the decomposition w.r.t. one variable, the intersection sets will be added as don't cares to the projected subfunctions, in order to possibly improve their minimal SOP forms.

Definition 2.5 Let $f(X)$ be an incompletely specified Boolean function, $x_i, x_j \in X$, and p_i and p_j be two completely specified Boolean functions depending on all variables in $X \setminus \{x_i, x_j\}$. The $[(x_i, p_i), (x_j, p_j)]$ -decomposition with intersection of $f = (f_{\text{on}}, f_{\text{dc}})$ is the algebraic expression

$$f = (\bar{x}_i \oplus p_i)(\bar{x}_j \oplus p_j)\tilde{f}|_{\substack{x_i=p_i \\ x_j=p_j}} + (\bar{x}_i \oplus p_i)(x_j \oplus p_j)\tilde{f}|_{\substack{x_i=p_i \\ x_j \neq p_j}} + \\ (x_i \oplus p_i)(\bar{x}_j \oplus p_j)\tilde{f}|_{\substack{x_i \neq p_i \\ x_j=p_j}} + (x_i \oplus p_i)(x_j \oplus p_j)\tilde{f}|_{\substack{x_i \neq p_i \\ x_j \neq p_j}} + \\ (\bar{x}_i \oplus p_i)I^{(i,=)} + (x_i \oplus p_i)I^{(i,\neq)} + (\bar{x}_j \oplus p_j)I^{(j,=)} + (x_j \oplus p_j)I^{(j,\neq)} + I,$$

where

$$\begin{aligned} \tilde{f}|_{\substack{x_i=p_i \\ x_j=p_j}} &= (f_{\text{on}}|_{\substack{x_i=p_i \\ x_j=p_j}} \setminus (I_{\text{on}}^{(i,=)} \cup I_{\text{on}}^{(j,=)} \cup I_{\text{on}}), f_{\text{dc}}|_{\substack{x_i=p_i \\ x_j=p_j}} \cup I_{\text{on}}^{(i,=)} \cup I_{\text{on}}^{(j,=)} \cup I_{\text{on}}), \\ \tilde{f}|_{\substack{x_i=p_i \\ x_j \neq p_j}} &= (f_{\text{on}}|_{\substack{x_i=p_i \\ x_j \neq p_j}} \setminus (I_{\text{on}}^{(i,=)} \cup I_{\text{on}}^{(j,\neq)} \cup I_{\text{on}}), f_{\text{dc}}|_{\substack{x_i=p_i \\ x_j \neq p_j}} \cup I_{\text{on}}^{(i,=)} \cup I_{\text{on}}^{(j,\neq)} \cup I_{\text{on}}), \\ \tilde{f}|_{\substack{x_i \neq p_i \\ x_j=p_j}} &= (f_{\text{on}}|_{\substack{x_i \neq p_i \\ x_j=p_j}} \setminus (I_{\text{on}}^{(i,\neq)} \cup I_{\text{on}}^{(j,=)} \cup I_{\text{on}}), f_{\text{dc}}|_{\substack{x_i \neq p_i \\ x_j=p_j}} \cup I_{\text{on}}^{(i,\neq)} \cup I_{\text{on}}^{(j,=)} \cup I_{\text{on}}), \\ \tilde{f}|_{\substack{x_i \neq p_i \\ x_j \neq p_j}} &= (f_{\text{on}}|_{\substack{x_i \neq p_i \\ x_j \neq p_j}} \setminus (I_{\text{on}}^{(i,\neq)} \cup I_{\text{on}}^{(j,\neq)} \cup I_{\text{on}}), f_{\text{dc}}|_{\substack{x_i \neq p_i \\ x_j \neq p_j}} \cup I_{\text{on}}^{(i,\neq)} \cup I_{\text{on}}^{(j,\neq)} \cup I_{\text{on}}), \end{aligned}$$

with

$$\begin{aligned} I_{\text{on}} &= f_{\text{on}}|_{\substack{x_i=p_i \\ x_j=p_j}} \cap f_{\text{on}}|_{\substack{x_i=p_i \\ x_j \neq p_j}} \cap f_{\text{on}}|_{\substack{x_i \neq p_i \\ x_j=p_j}} \cap f_{\text{on}}|_{\substack{x_i \neq p_i \\ x_j \neq p_j}}, \\ I_{\text{dc}} &= f_{\text{dc}}|_{\substack{x_i=p_i \\ x_j=p_j}} \cap f_{\text{dc}}|_{\substack{x_i=p_i \\ x_j \neq p_j}} \cap f_{\text{dc}}|_{\substack{x_i \neq p_i \\ x_j=p_j}} \cap f_{\text{dc}}|_{\substack{x_i \neq p_i \\ x_j \neq p_j}}, \\ I_{\text{on}}^{(i,=)} &= (f_{\text{on}}|_{\substack{x_i=p_i \\ x_j=p_j}} \cap f_{\text{on}}|_{\substack{x_i=p_i \\ x_j \neq p_j}}) \setminus I_{\text{on}}, & I_{\text{dc}}^{(i,=)} &= (f_{\text{dc}}|_{\substack{x_i=p_i \\ x_j=p_j}} \cap f_{\text{dc}}|_{\substack{x_i=p_i \\ x_j \neq p_j}}) \cup I_{\text{on}}, \\ I_{\text{on}}^{(i,\neq)} &= (f_{\text{on}}|_{\substack{x_i \neq p_i \\ x_j=p_j}} \cap f_{\text{on}}|_{\substack{x_i \neq p_i \\ x_j \neq p_j}}) \setminus I_{\text{on}}, & I_{\text{dc}}^{(i,\neq)} &= (f_{\text{dc}}|_{\substack{x_i \neq p_i \\ x_j=p_j}} \cap f_{\text{dc}}|_{\substack{x_i \neq p_i \\ x_j \neq p_j}}) \cup I_{\text{on}}, \end{aligned}$$

$$\begin{aligned}
I_{\text{on}}^{(j,=)} &= (f_{\text{on}}|_{\substack{x_i=p_i \\ x_j=p_j}} \cap f_{\text{on}}|_{\substack{x_i \neq p_i \\ x_j=p_j}}) \setminus I_{\text{on}}, & I_{\text{dc}}^{(j,=)} &= (f_{\text{dc}}|_{\substack{x_i=p_i \\ x_j=p_j}} \cap f_{\text{dc}}|_{\substack{x_i \neq p_i \\ x_j=p_j}}) \cup I_{\text{on}}, \\
I_{\text{on}}^{(j,\neq)} &= (f_{\text{on}}|_{\substack{x_i=p_i \\ x_j \neq p_j}} \cap f_{\text{on}}|_{\substack{x_i \neq p_i \\ x_j \neq p_j}}) \setminus I_{\text{on}}, & I_{\text{dc}}^{(j,\neq)} &= (f_{\text{dc}}|_{\substack{x_i=p_i \\ x_j \neq p_j}} \cap f_{\text{dc}}|_{\substack{x_i \neq p_i \\ x_j \neq p_j}}) \cup I_{\text{on}}.
\end{aligned}$$

Observe that we do not consider the intersections between $f|_{\substack{x_i=p_i \\ x_j=p_j}}$ and $f|_{\substack{x_i \neq p_i \\ x_j \neq p_j}}$ and between $f|_{\substack{x_i=p_i \\ x_j \neq p_j}}$ and $f|_{\substack{x_i \neq p_i \\ x_j=p_j}}$ as these two pairs of projections belong to non-adjacent subspaces and therefore there are no cubes split between them.

When the functions p_i and p_j , the four projected subfunctions, and the intersection sets are represented as minimal SOP forms, these two algebraic expressions give rise to P-circuits without and with intersection, both of depth 5, exactly as in the case of the decomposition w.r.t. a single variable. Moreover, the two critical signals x_i and x_j are equally close to the output and their switching activity affects only a constant number of gates, as p_i , p_j and the intersection sets do not depend on them.

The two circuits can be synthesized generalizing the algorithms shown in Figs. 2.4 and 2.5 in a straightforward way.

For example, consider the function f shown on the left side of Fig. 2.1. Let $p_i = 0$, $p_j = 0$, and x_i and x_j be x_1 and x_2 , respectively. The Boolean space $\{0, 1\}^4$ can be partitioned into the four sets: $(x_1 = 0, x_2 = 0)$, $(x_1 = 0, x_2 = 1)$, $(x_1 = 1, x_2 = 0)$, and $(x_1 = 1, x_2 = 1)$, each containing 2^2 points. The projections of f onto these four sets are

$$\begin{aligned}
f_{\text{on}}|_{\substack{x_1=0 \\ x_2=0}} &= \{00, 01, 10\} & f_{\text{dc}}|_{\substack{x_1=0 \\ x_2=0}} &= \emptyset \\
f_{\text{on}}|_{\substack{x_1=0 \\ x_2 \neq 0}} &= \{01\} & f_{\text{dc}}|_{\substack{x_1=0 \\ x_2 \neq 0}} &= \{11\} \\
f_{\text{on}}|_{\substack{x_1 \neq 0 \\ x_2=0}} &= \{01, 10\} & f_{\text{dc}}|_{\substack{x_1 \neq 0 \\ x_2=0}} &= \emptyset \\
f_{\text{on}}|_{\substack{x_1 \neq 0 \\ x_2 \neq 0}} &= \{00, 01\} & f_{\text{dc}}|_{\substack{x_1 \neq 0 \\ x_2 \neq 0}} &= \emptyset
\end{aligned}$$

The $[(x_1, 0), (x_2, 0)]$ -decomposition of f thus determines the optimal P-circuit $\bar{x}_1 \bar{x}_2 (\bar{x}_3 + \bar{x}_4) + \bar{x}_1 x_2 x_4 + x_1 \bar{x}_2 (\bar{x}_3 x_4 + x_3 \bar{x}_4) + x_1 x_2 \bar{x}_3$, containing 16 literals.

Let us now consider the $[(x_1, 0), (x_2, 0)]$ -decomposition with intersection. The intersection sets are $I_{\text{on}} = \{01\}$, $I_{\text{dc}} = \emptyset$, $I_{\text{on}}^{(i,=)} = I_{\text{on}}^{(i,\neq)} = I_{\text{on}}^{(j,=)} = \emptyset$, $I_{\text{on}}^{(j,\neq)} = \{10\}$, $I_{\text{dc}}^{(i,=)} = I_{\text{dc}}^{(i,\neq)} = I_{\text{dc}}^{(j,=)} = I_{\text{dc}}^{(j,\neq)} = \{01\}$, and the projections become

$$\begin{aligned}
\tilde{f}_{\text{on}}|_{\substack{x_1=0 \\ x_2=0}} &= \{00\} & \tilde{f}_{\text{dc}}|_{\substack{x_1=0 \\ x_2=0}} &= \{01, 10\} \\
\tilde{f}_{\text{on}}|_{\substack{x_1=0 \\ x_2 \neq 0}} &= \emptyset & \tilde{f}_{\text{dc}}|_{\substack{x_1=0 \\ x_2 \neq 0}} &= \{01, 11\} \\
\tilde{f}_{\text{on}}|_{\substack{x_1 \neq 0 \\ x_2=0}} &= \emptyset & \tilde{f}_{\text{dc}}|_{\substack{x_1 \neq 0 \\ x_2=0}} &= \{01, 10\} \\
\tilde{f}_{\text{on}}|_{\substack{x_1 \neq 0 \\ x_2 \neq 0}} &= \{00\} & \tilde{f}_{\text{dc}}|_{\substack{x_1 \neq 0 \\ x_2 \neq 0}} &= \{01\}
\end{aligned}$$

The corresponding P-circuit with intersection is now $\bar{x}_1 \bar{x}_2 \bar{x}_3 + x_1 x_2 \bar{x}_3 + \bar{x}_2 x_3 \bar{x}_4 + \bar{x}_3 x_4$, with 11 literals.

2.5 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 1.6 GHz CPU with 1 GB of main memory. The benchmarks are taken from LGSynth93 [17]. 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 area and delay 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 2.1). In particular we have considered P-circuits for the following choices of the projection function p :

- $p = 0$, decomposition with intersection (referred as **Constant** in Table 2.2);
- $p = x_j$, decomposition without and with intersection (referred as **VAR** in Tables 2.1 and 2.2);
- $p = x_j \oplus x_k$, decomposition without and with intersection (referred as **XOR** in Tables 2.1 and 2.2);
- $p = x_j x_k$, decomposition without and with intersection, choosing the complementations of variables giving the best area (referred as **AND** in Tables 2.1 and 2.2).

After the projection, all SOP components of the P-circuits have been synthesized with multioutput 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 Tables 2.1 and 2.2 we compare the mapped area and the synthesis time (in seconds) of P-circuits representing decomposition forms without intersection (Table 2.1) and with intersection (Table 2.2) 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 the overall set of experiments we have considered decompositions with respect to each input variable of each benchmark.

The results, reported in Fig. 2.6, are quite interesting: about 79% of P-circuits based on the $(x_i, 0)$ -decomposition with intersection have an area smaller than the P-circuits based on the classical Shannon decomposition, i.e., on the $(x_i, 0)$ -decomposition without intersection; analogously, 32 and 59% of the P-circuits benefit from the (x_i, x_j) -decomposition without and with intersection, respectively; 22 and 50% of the circuits benefit from the $(x_i, x_j \oplus x_k)$ -decomposition without and with intersection, respectively; and 28 and 58% of the circuits benefit from the $(x_i, x_j x_k)$ -decomposition without and with intersection, respectively. 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, as shown in Fig. 2.7; in fact, about 80% of P-circuits based on the $(x_i, 0)$ -decomposition with intersection have a synthesis time smaller than the synthesis

Table 2.1 Comparison of area and synthesis time of P-circuits representing (x_0, p) -decomposition forms for different choices of the projection function p without intersection

(x_0, p) -Decomposition without intersection								
Benchmark (in/out)	Shannon		VAR		XOR		AND	
	Area	Time	Area	Time	Area	Time	Area	Time
add6 (12/7)	908	0.65	507	5.19	669	24.58	524	90.84
adr4 (8/5)	284	0.05	172	0.14	223	0.45	237	1.76
alu2 (10/8)	355	0.45	382	0.79	416	3.60	356	12.93
alu3 (10/8)	256	0.34	330	0.67	402	2.54	354	9.22
amd (14/24)	162	0.17	1694	1.24	1800	8.65	1747	30.31
apla (10/1)	379	0.12	371	0.58	467	3.19	398	9.11
b9 (16/5)	436	0.15	463	1.08	492	8.30	472	29.36
b12 (15/9)	227	0.11	306	0.55	401	4.27	340	15.90
br1 (12/8)	347	0.05	381	0.19	435	0.88	418	3.53
br2 (12/8)	281	0.03	314	0.18	377	0.97	337	3.30
dc2 (8/7)	249	0.05	279	0.13	337	0.40	276	1.45
dist (8/7)	891	0.11	1266	0.34	1202	0.95	946	3.77
dk17 (10/11)	263	0.10	250	0.38	291	1.82	230	6.85
dk48 (15/17)	263	0.23	284	1.65	288	14.73	276	46.21
ex7 (16/5)	436	0.12	463	1.04	492	8.30	472	29.07
exp (8/18)	824	0.09	873	0.30	947	0.96	1011	3.15
f51m (8/8)	497	0.09	706	0.21	640	0.64	528	2.24
inc (7/9)	237	0.05	287	0.11	364	0.25	316	1.17
l8err (8/8)	301	0.08	328	0.30	356	0.70	311	2.43
life (9/1)	267	0.06	252	0.21	298	0.60	267	2.56
m181 (15/9)	227	0.42	308	0.58	404	4.44	341	16.39
m2 (8/16)	808	0.08	919	0.21	1282	0.55	1043	1.93
m3 (8/16)	1042	0.08	1392	0.24	1638	0.71	1184	2.92
m4 (8/16)	2766	0.19	3286	0.96	2846	2.10	2271	6.93
max1024 (10/6)	2534	0.34	2511	1.97	2973	8.74	2642	30.72
max128 (7/24)	2373	0.08	2711	0.35	3219	0.91	2391	3.14
max512 (9/6)	1470	0.15	1607	0.64	1116	2.27	1227	8.09
mlp4 (8/8)	1113	0.15	1031	0.36	1292	1.13	997	4.12
mp2d (14/14)	355	0.09	435	0.61	508	4.47	455	16.49
p1 (8/18)	724	0.18	781	0.96	821	3.07	842	10.77
p3 (8/14)	587	0.22	524	0.52	559	1.64	548	5.90
p82 (5/14)	244	0.02	321	0.06	394	0.11	370	0.33
rd73 (7/3)	312	0.05	437	0.60	355	9.12	388	35.82
root (8/5)	416	0.05	594	0.14	393	0.50	385	1.91
spla (8/5)	2239	0.79	2570	7.88	3142	74.99	2886	273.75
sqr6 (6/12)	443	0.05	656	6.05	561	43.76	532	170.62
sym10 (10/1)	559	0.30	414	0.64	309	2.92	416	14.31
t1 (21/23)	905	0.83	951	3.52	1186	41.02	982	155.28
t2 (17/16)	501	0.06	589	0.65	686	6.37	618	22.95
t3 (12/8)	156	0.14	212	0.74	275	5.21	236	20.77
tial (14/8)	3430	5.33	3337	23.68	4062	159.84	3823	557.19
tms (8/16)	670	0.03	787	23.00	904	161.92	737	548.21
vtx1 (27/6)	430	0.09	445	1.89	501	32.57	585	107.74
x9dn (27/7)	530	0.22	528	2.23	595	30.62	548	116.64
Z5xp1 (7/10)	479	0.08	593	0.12	743	0.33	547	1.24
Z9sym (9/1)	464	0.17	288	0.33	267	1.15	371	6.07

Table 2.2 Comparison of area and synthesis time of P-circuits representing (x_0, p) -decomposition forms for different choices of the projection function p with intersection

(x_0, p) -Decomposition with intersection								
Benchmark (in/out)	Constant		VAR		XOR		AND	
	Area	Time	Area	Time	Area	Time	Area	Time
add6 (12/7)	672	0.51	814	4.44	759	23.70	651	80.93
adr4 (8/5)	203	0.03	125	0.18	161	0.40	175	1.58
alu2 (10/8)	283	0.18	308	1.03	310	4.72	298	16.79
alu3 (10/8)	263	0.16	276	0.42	295	1.67	283	5.91
amd (14/24)	1012	0.12	1085	1.55	1202	10.88	1180	37.65
apla (10/1)	379	0.08	371	0.38	470	1.42	398	6.11
b9 (16/5)	327	0.20	360	0.81	393	5.17	364	19.74
b12 (15/9)	199	0.18	248	0.65	367	5.25	292	18.13
br1 (12/8)	347	0.02	381	0.18	435	0.87	418	3.47
br2 (12/8)	281	0.01	314	0.18	377	0.86	337	3.16
dc2 (8/7)	238	0.02	281	0.14	355	0.28	268	1.46
dist (8/7)	1036	0.09	1507	0.26	1373	0.69	1048	3.24
dk17 (10/11)	263	0.06	250	0.46	291	1.99	230	7.21
dk48 (15/17)	263	0.17	284	0.69	288	4.34	276	17.89
ex7 (16/5)	327	0.09	360	1.56	393	10.39	364	38.51
exp (8/18)	838	0.05	877	0.22	930	0.66	1035	3.01
f51m (8/8)	277	0.09	290	0.28	314	0.85	323	4.11
inc (7/9)	270	0.02	134	0.10	372	0.22	348	0.85
l8err (8/8)	355	0.03	337	0.18	450	0.68	354	2.48
life (9/1)	197	0.05	227	0.12	216	0.43	224	2.03
m181 (15/9)	199	0.08	252	0.68	341	6.65	288	29.20
m2 (8/16)	808	0.05	919	0.24	1282	0.48	1043	2.23
m3 (8/16)	1042	0.05	1392	0.26	1638	0.76	1184	3.53
m4 (8/16)	2163	0.14	2981	0.38	3683	1.22	2496	4.77
max1024 (10/6)	2980	0.25	3043	2.12	2977	10.13	2829	34.28
max128 (7/24)	2155	0.06	2259	0.23	2704	0.58	1975	1.97
max512 (9/6)	1346	0.12	1533	0.39	1351	1.45	1265	5.02
mlp4 (8/8)	908	0.08	917	0.30	1081	0.98	938	3.34
mp2d (14/14)	276	0.16	357	0.75	411	6.82	359	22.56
p1 (8/18)	711	0.20	777	1.18	847	3.74	818	13.66
p3 (8/14)	520	0.12	552	0.37	554	0.79	504	2.68
p82 (5/14)	229	0.02	313	0.06	372	0.10	343	0.31
rd73 (7/3)	332	0.02	577	0.69	496	8.78	464	33.63
root (8/5)	417	0.02	536	0.17	602	0.55	446	1.94
spla (8/5)	2428	0.73	2761	8.82	3249	84.11	3107	336.30
sqr6 (6/12)	333	1.59	429	4.49	437	40.35	370	124.48
sym10 (10/1)	568	0.27	529	0.96	551	3.90	554	16.81
t1 (21/23)	463	0.61	510	6.06	655	78.07	585	277.38
t2 (17/16)	358	0.05	406	0.88	469	9.80	416	22.33
t3 (12/8)	218	0.08	270	0.74	336	3.77	295	14.03
tial (14/8)	3368	3.29	3319	31.12	3952	215.08	3827	741.85
tms (8/16)	670	3.00	787	14.06	904	103.07	737	317.65
vtx1 (27/6)	390	0.14	499	3.03	486	50.57	524	171.45
x9dn (27/7)	412	0.19	401	4.26	457	57.18	418	217.77
Z5xp1 (7/10)	324	0.03	369	0.19	441	0.41	302	1.29
Z9sym (9/1)	379	0.17	391	0.64	395	1.68	393	9.28

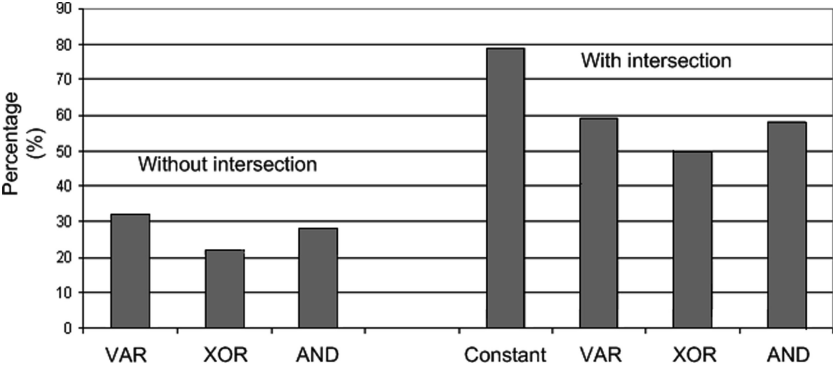


Fig. 2.6 Percentage of P-circuits, over all the benchmarks, having smaller area than the P-circuits based on Shannon decomposition

time of P-circuits based on the classical Shannon decomposition. When p is not constant, synthesis is time consuming, since the algorithm must choose the best combination of variables for p . In particular, 3 and 5% of the P-circuits benefit from the (x_i, x_j) -decomposition without and with intersection, respectively; 2% of the circuits benefit from the $(x_i, x_j \oplus x_k)$ -decomposition both without and with intersection; and only 1% of the circuits benefit from the $(x_i, x_j x_k)$ -decomposition both without and with intersection. Altogether, only 14% of the P-circuits achieve the smallest area when implemented according to the classical Shannon decomposition. The subset of results shown in Tables 2.1 and 2.2 reflects these percentages.

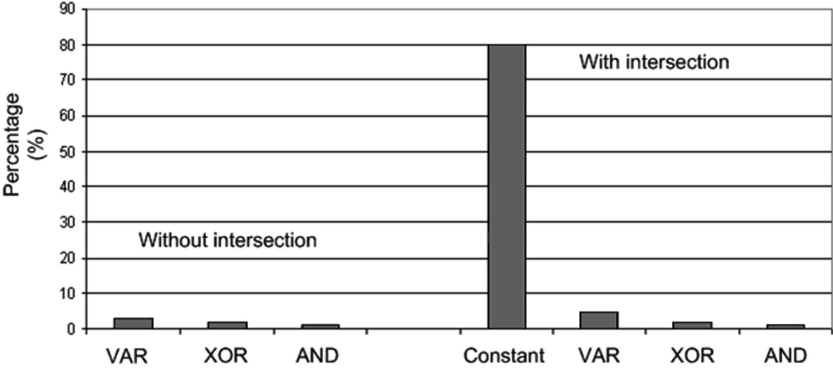


Fig. 2.7 Percentage of P-circuits, over all the benchmarks, having smaller synthesis time than the P-circuits based on Shannon decomposition

2.6 Conclusion

In conclusion, we presented a new method to decompose Boolean functions via complex cofactoring in the presence of signals with high switching activity. Experimental results show that this decomposition yields circuits more compact 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.

Acknowledgments Tiziano Villa gratefully acknowledges partial support from the COCONUT EU project FP7-2007-IST-1-217069, and the CON4COORD EU Project FP7-ICT-2007.3.7.(c) grant agreement nr. INFOS-ICT-223844.

References

1. Benini, L., Micheli, G.D.: Logic synthesis for low power. In: S. Hassoun, T. Sasao (eds.) *Logic Synthesis and Verification*, pp. 197–223. Kluwer Academic Publishers Norwell, MA, USA (2002)
2. Bernasconi, A., Ciriani, V., Cordone, R.: On projecting sums of products. In: 11th Euromicro Conference on Digital Systems Design: Architectures, Methods and Tools. Parma, Italy (2008)
3. Bernasconi, A., Ciriani, V., Cordone, R.: The optimization of kEP-SOPs: Computational complexity, approximability and experiments. *ACM Transactions on Design Automation of Electronic Systems* **13**(2), 1–31 (2008)
4. Bernasconi, A., Ciriani, V., Trucco, G., Villa, T.: Logic Minimization and Testability of 2SPPP-Circuits. In: Euromicro Conference on Digital Systems Design (DSD). Patras, Greece (2009)
5. Bernasconi, A., Ciriani, V., Trucco, G., Villa, T.: On decomposing Boolean functions via extended cofactoring. In: *Design Automation and Test in Europe*. Nice, France (2009)
6. Bioch, J.C.: The complexity of modular decomposition of Boolean functions. *Discrete Applied Mathematics* **149**(1–3), 1–13 (2005)
7. Brayton, R., Hachtel, G., McMullen, C., Sangiovanni-Vincentelli, A.L.: *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer Academic Publishers Norwell, MA, USA (1984)
8. Brown, F.: *Boolean Reasoning*. Kluwer Academic Publishers, Boston (1990)
9. Bryant, R.: Graph based algorithm for Boolean function manipulation. *IEEE Transactions on Computers* **35**(9), 667–691 (1986)
10. Ciriani, V.: Synthesis of SPP three-level logic networks using affine spaces. *IEEE Transactions on CAD of Integrated Circuits and Systems* **22**(10), 1310–1323 (2003)
11. Cortadella, J.: Timing-driven logic bi-decomposition. *IEEE Transactions on CAD of Integrated Circuits and Systems* **22**(6), 675–685 (2003)
12. Kerntopf, P.: New generalizations of Shannon decomposition. In: *International Workshop on Applications of Reed-Muller Expansion in Circuit Design*, pp. 109–118. Starkville, Mississippi, USA (2001)
13. Lavagno, L., McGeer, P.C., Saldanha, A., Sangiovanni-Vincentelli, A.L.: Timed Shannon circuits: A power-efficient design style and synthesis tool. In: 32nd ACM/IEEE Conference on Design automation, pp. 254–260. (1995)

14. McGeer, P.C., Brayton, R.K., Sangiovanni-Vincentelli, A.L., Sahni, S.: Performance enhancement through the generalized bypass transform. In: ICCAD, pp. 184–187. Santa Clara, CA, USA (1991)
15. Pedram, M.: Power estimation and optimization at the logic level. *International Journal of High Speed Electronics and Systems* **5**(2), 179–202 (1994)
16. Soviani, C., Tardieu, O., Edwards, S.A.: Optimizing sequential cycles through Shannon decomposition and retiming. In: DATE '06: Proceedings of the conference on Design, Automation and Test in Europe, pp. 1085–1090. European Design and Automation Association, 3001 Leuven, Belgium, Belgium (2006)
17. Yang, S.: Logic synthesis and optimization benchmarks user guide version 3.0. User Guide, Microelectronics Center of North Carolina (1991)



<http://www.springer.com/978-1-4419-7517-1>

Advanced Techniques in Logic Synthesis, Optimizations
and Applications

Gulati, K. (Ed.)

2011, XXI, 423 p., Hardcover

ISBN: 978-1-4419-7517-1