Università di Pisa
Dipartimento di Informatica

Technical Report: TR-02-03

A New Approach to Three-Level Logic Synthesis

Valentina Ciriani
Department of Computer Science
University of Pisa
Corso Italia, 40
56127 Pisa, Italy
ciriani@di.unipi.it

April 10, 2002

ADDRESS: Corso Italia 40, 56125 Pisa, Italy. TEL: +39 050 2212700 FAX: +39 050 2212726

A New Approach to Three-Level Logic Synthesis

Valentina Ciriani
Department of Computer Science
University of Pisa
Corso Italia, 40
56127 Pisa, Italy
ciriani@di.unipi.it

April 10, 2002

Abstract

Three-level logic SPP forms are OR of AND of EXORs expressions. In the framework of SPP minimization we give a new algebraic definition of SPP expressions using affine spaces. The main problems of SPP model are: the "hard to digest" SPP theory; the time required for minimization, which is still high; and the unbounded fan-in EXOR gates in the form. Consequently, our main results in this paper are: 1) we rephrase the SPP theory using well known algebraic structures (vector and affine spaces) to obtain an easier description of pseudocubes, which play the same role as cubes in standard minimization; 2) we describe a new canonical representation of pseudocubes leading to a more efficient data structure for SPP minimization; 3) we introduce a novel form, called *k-SPP form*, where the number of literal in the EXOR factors is upper bounded by a chosen constant k, and show how to modify the SPP algorithms for computing the minimal k-SPP form efficiently. Finally, we perform an extensive set of experiments on classical benchmarks aimed at validating the new approach.
Index Terms: Three-level logic synthesis, EXOR gates, bounded fan-in , sum of pseudoproducts (SPP) forms, logical design.

1 Introduction

Two-level logic minimization has been one of the most studied problems in logic synthesis. Big efforts have been done to elaborate efficient Sum of Products (or SP) minimization procedures and their related set covering algorithms (see for example [2, 19, 16, 5, 24]). The implicit techniques are able to handle very complex functions in reasonable time. Three-level logic minimization is instead much harder, and the difficulty further increases with the number of levels. On the other hand, the size of a three-level network can be significantly smaller than the size of a two-level network. In many cases three-level logic is a good trade-off among circuit speed, circuit size, and the time needed for the minimization procedure [20]. Note that, in all cases, algorithms for exact minimization have exponential complexity, hence the time to attain minimal forms becomes huge for increasing size of the input.

A relevant three-level minimization algorithm was given in [14] for networks of the form $f = g_1 \circ g_2$ where g_1 and g_2 are SP forms and $\circ$ denotes a *binary* operation. The case of $\circ = EXOR$ (called *EX-SOP*) has been widely studied (see for details [21, 17, 11, 7, 8, 9, 12]). The case of $\circ = AND$ (called *AND-OR-AND*) is described in [10]. Indeed many practical functions (e.g., arithmetic functions) have a more concise expression if we allow the use of EXORs other than classical AND and OR gates. For example, the parity function $F = x_1 \oplus x_2 \oplus \ldots \oplus x_n$ is represented with a SP form composed of 2^{n-1} products. Another well known advantage of EXOR gates is their excellent testability.

An EX-SOP three-level network is one of the simplest EXOR of sum of product architecture, since it contains only a single two-input EXOR gate. An algorithm for exact minimization of EX-SOP networks is described in [7], limited to functions with up to five variables. Some interesting heuristics are described in [9, 12]. An estimation metric which measures whether an input function is suitable for EX-SOP minimization is also developed in [12]. To the best of our knowledge no practical algorithm for exact minimization of EX-SOP forms has been described thus far.

In [13] is introduced a different three-level form called *Sum of Pseudoproducts* (or *SPP*) to express Boolean functions using EXOR gates, as a direct generalization of SP expressions. A *pseudoproduct* is a product (AND) of Exclusive OR (EXOR) factors, and an *SPP form* is a sum (OR) of pseudoproducts. For example, $(x_1 \oplus \overline{x}_2) \cdot x_5 \cdot (x_1 \oplus x_4 \oplus \overline{x}_7) + x_5 \cdot \overline{x}_4 + (x_1 \oplus x_3 \oplus x_4) \cdot (x_3 \oplus x_5 \oplus x_6)$ is an SPP form. Any arbitrary Boolean function can be expressed as a disjunction of pseudoproducts, giving rise to a sum of pseudoproducts form.

The three-level forms studied in the past couldn't be handled efficiently with the same techniques used for the minimization of SP forms. On the contrary, SPP and SP forms have similar definitions and properties which we review here. A product P is an *implicant* of the Boolean function F iff $P \subseteq F$ (Boolean functions are treated as sets of points of $\{0,1\}^n$). An *EXOR factor* is a single literal, or a string of literals connected by EXOR's. A product P of EXOR factors is a *pseudoproduct* of the Boolean function F iff $P \subseteq F$. An implicant (pseudoproduct) P of F is *prime* iff no other implicant (pseudoproduct) P' of F exists such that $P \subseteq P'$. A product (pseudoproduct) is the characteristic function of a set of points denoted *cube* (*pseudocube*). Minimal SP (SPP) covers of a function F are made of prime implicants (pseudoproducts), and SP (SPP) *minimization* is a set covering problem on the set of prime implicants (pseudoproducts) with cost function given by the number of factors or literals. Experimental results show that the average size of SPP forms is approximately half the size of the corresponding SP forms, and SPP expressions are, in general, more compact than EX-SOP and AND-OR-AND forms (see Section 5). In the worst case, SP and SPP forms coincide. In this paper we then study SPP forms for multiple reasons: 1) SPP forms have a strong and concise algebraic expression; 2) SPP forms are a direct generalization of SP forms, consequently we can use similar techniques for both SPP and SP minimization; 3) experimental results show that SPP forms are more compact than two-level and other three-level forms; 4) employing EXOR gates often facilitates circuit testing; 4) to our knowledge, SPP minimization is the only practical exact (e.g., not heuristic) three-level logic synthesis.

As the original Quine-McCluskey algorithm ([18, 15]), the one proposed in [13] for SPP minimization is exponential in nature, and fails in practice in minimizing functions with a too large number of prime pseudoproducts (i.e., more than 15.000). A deeper understanding of the problem, together with the use of ad hoc data structures, has allowed to widely extend the set of functions practically tractable [3]. On the other hand, minimization of SPP forms requires a greater effort than SP optimization, and research on SPP synthesis is still at the beginning. For example we have no implicit minimization methods, and the number of EXOR gates is, in general, unbounded. Yet a number of standard benchmark functions can not be easily handled, and this implies that a deeper understanding of the mathematical properties of SPP forms is worth to be achieved.

The contributes of this paper are twofold. First we give a new algebraic description of pseudocubes and cubes using affine spaces. This understanding of the theory allows us to describe a new canonical representation of pseudocubes leading to a more compact and efficient data structure for the minimization algorithm. Second, we describe a new model of SPP forms with bounded fan-in gates that guarantees practical networks and extremely reduced minimization time.

In particular, in the first part of the paper we introduce affine spaces, and prove that the set of pseudocubes coincide with the set of affine subspaces of $\{0,1\}^n$. We then prove that the union of two pseudocubes is a pseudocube iff they are affine spaces over the same vector space. This property is used to design our minimization algorithms, which are based on a new data structure called *compressed partition trie* (CPT). CPTs , similarly to partition tries introduced in [3], have two main properties: first the pseudocubes are stored in disjoint partitions, each containing pseudocubes over the same vector space; second the pseudocubes and, in particular, their bases are stored compactly (for example, the common basis of different pseudocubes is stored only once).

As in the Quine-McCluskey approach, the generation of prime pseudoproducts is performed in steps by successive unions of pseudocubes, but here only the ones over the same vector space are compared. For this purpose the pseudocubes found in each step are grouped in sets, each of which contains pseudocubes over the same vector space, to reduce the computational work drastically. We then describe a new SPP minimization algorithm based on the CPT data structure. Experimental results in this paper show that the new algorithm is, on the average, 23% faster that the algorithm proposed in [3].

Among three-level networks, SPP forms are particularly compact. However SPP forms have been defined for EXOR gates with unbounded fan-in that seems to be impractical. Indeed in most technologies EXOR gates with many inputs are slow and expensive [25]. It follows that finding SPP-like forms with a fixed maximum number of literal in the EXOR factors is desirable. In the second part of paper we introduce the *k-SPP form* where each EXOR factor contains at most k literals. In a Boolean space $\{0,1\}^n$ described by n variables, let a *k-EXOR factor* ($1 \leq k \leq n$) be an EXOR factor with at most k literals. A *k-pseudoproduct* is a product of k-EXOR factors, and a *k-SPP form* is a sum of k-pseudoproducts.

All SPP algorithms, described till now, follow the same union strategy in determining prime pseudoproducts. In fact, they start from the minterms, and pair pseudoproducts until the set of prime pseudoproducts is found. In this paper we show how to modify any SPP minimization algorithm, based on a union procedure, to efficiently compute k-SPP forms. Note that, we cannot simply generate all the prime pseudoproducts, and then select the prime k-pseudoproducts between them, because the set of prime k-pseudoproducts is not, in general, contained in the set of prime pseudoproducts. Our experimental results show that the size of the k-SPP minimal forms is not significantly greater than the one for unbounded fan-in, but the computational effort drastically decreases. For example 2-SPP synthesis requires **92%** less time than SPP synthesis. Moreover 2-SPP forms have, on the average, only **4%** more literals than SPP forms.

In the next section, we recall the definition of affine spaces and we prove the direct correspondence with the set of pseudocubes. We also describe a new canonical representation of pseudocubes as affine spaces. In Section 3 we describe a new exact SPP minimization algorithm using the canonical representation of pseudocubes. We first describe how to perform the union of two affine spaces over the same vector space. Second we show the *compressed partition tries* (CPT) as the fundamental data structure to represent affine spaces and to efficiently implement the union strategy. In Section 4 we introduce k-SPP forms and their properties, and present an exact minimization algorithm. Section 5 gives some experimental results demonstrating the effectiveness of the new methods.

2 Pseudocubes via Affine Spaces

Pseudocubes, pseudoproducts and SPP forms are presented in [13] as direct generalizations of cubes, products and SP forms. In this section we give an equivalent, but easier, new description of pseudocubes using affine spaces (for an introduction on vector and affine spaces see [4]). We work in a Boolean space $\{0,1\}^n$ described by n variables $x_1, x_2, \ldots, x_n$, where each point is represented as a binary vector of n *components*. Let us now extend the symbol $\oplus$ to denote the elementwise EXOR (or modulo 2 sum) between two vectors. Then $\alpha \oplus \beta$ is the vector obtained from β complementing in it the elements corresponding to the 1's of α. For a vector $\alpha \in \{0,1\}^n$ and a subset $S \subseteq \{0,1\}^n$, let $\alpha \oplus S = \{\alpha \oplus s \,|\, s \in S\}$.

Definition 1 *Let $(V, \oplus)$ be a vector subspace of $\{0,1\}^n$ with the bitwise EXOR operator, $\oplus$, and let $\alpha \in \{0,1\}^n$ be a Boolean point. The set*

$$A = \alpha \oplus V = \{\alpha \oplus v \mid v \in V\}$$

is an affine space *over V with* translation point *α.*

In practice an affine space is a translate of a vector space. For example $V = \{000, 010, 011, 001\}$ is a vector subspace of $\{0,1\}^3$, and $A = 100 \oplus V = \{100, 110, 111, 101\}$ is an affine space over V. Note that α is any vector of A. Indeed in the former example $A = 100 \oplus V = 110 \oplus V = 111 \oplus V = 101 \oplus V = \{100, 110, 111, 101\}$. In other words, there exists an unique vector space V such that $A = \alpha \oplus V$. Furthermore, $(\alpha \oplus V \equiv V) \Leftrightarrow \alpha \in V$. For example $A = 010 \oplus V = \{000, 010, 011, 001\} = V$. Clearly, a vector space is an affine space.

An affine space A over V is *k-dimensional* (or equivalently has dimension k) if $dim(V) = k$. We have:

Definition 2 *A* pseudocube *of degree m is an affine space of dimension m.*

In Section 2.3 we give a proof that this definition of pseudocubes and the definition given in [13] coincide.

2.1 Canonical Representation of Pseudocubes

Since α can be any point of the pseudocube A, and the vector space V can be represented by any of its bases, we need a canonical representation of the affine space A. Therefore we introduce some notation. A set of k points in $\{0,1\}^n$ (e.g., an affine or vector space) can be arranged in a $k \times n$ matrix whose rows correspond to the points and whose columns correspond to the variables $x_1, x_2, \ldots, x_n$ (see for example Fig. 1).

Definition 3 *A matrix of points in $\{0,1\}^n$ is in* binary order *if its rows (points) are sorted as increasing binary numbers.*

Definition 4 *Let A be an affine space, the* canonical translation point *α_A is the minimum point of A in binary order.*

Definition 5 *Let A be an affine spaces over V, a vector space sorted in binary order from index 0 to $2^k - 1$. The* canonical basis *B_A of V (or, equivalently, of A) is the set of points of V with indices $2^0, 2^1, \ldots, 2^{k-1}$.*

Definition 6 *The canonical translation point plus the canonical basis form the* canonical representation *of a pseudocube.*

x1 x2 x3 x4 x5		x1 x2 x3 x4 x5	
$\boxed{0\ 0\ 0\ 0\ 1}$	= a	0 0 0 0 0	
0 0 1 0 0		$\boxed{0\ 0\ 1\ 0\ 1}$	= v_1
0 1 0 1 1		$\boxed{0\ 1\ 0\ 1\ 0}$	= v_2
0 1 1 1 0		0 1 1 1 1	
1 0 0 1 1		$\boxed{1\ 0\ 0\ 1\ 0}$	= v_3
1 0 1 1 0		1 0 1 1 1	
1 1 0 0 1		1 1 0 0 0	
1 1 1 0 0		1 1 1 0 1	
A		V	

Figure 1: A pseudocube represented as an affine space $A = a \oplus V$ and the corresponding vector space V, the points v_1, v_2, v_3 are the canonical basis for V, and $a = \alpha_A$ is the translation point (note that $V = a \oplus A$).

x1 x2 x3 x4 x5		x1 x2 x3 x4 x5	
$\boxed{0\ 0\ 0\ 0\ 1}$	= a	0 0 0 0 0	
0 0 1 0 1		$\boxed{0\ 0\ 1\ 0\ 0}$	= v_1
0 1 0 0 1		$\boxed{0\ 1\ 0\ 0\ 0}$	= v_2
0 1 1 0 1		0 1 1 0 0	
1 0 0 0 1		$\boxed{1\ 0\ 0\ 0\ 0}$	= v_3
1 0 1 0 1		1 0 1 0 0	
1 1 0 0 1		1 1 0 0 0	
1 1 1 0 1		1 1 1 0 0	
A		V	

Figure 2: A cube represented as an affine space $A = a \oplus V$ and the corresponding vector space V.

For example in Figure 1 the affine space A is represented as $A = a \oplus V$, and its canonical basis is v_1, v_2, v_3 (note that A and V are in binary order).

Recall that the *standard basis* of the vector space $\{0,1\}^n$ is the set $\{e_1, \ldots, e_n\}$ where e_i is a vector whose ith component from left is 1 while the others are 0 (i.e., $e_1 = 100 \cdots 00$, $e_2 = 010 \cdots 00$, $\ldots$, $e_n = 000 \cdots 01$). A cube is a particular pseudocube. Indeed a cube is an affine space over a particular vector space:

Proposition 1 *An affine space $A = \alpha_A \oplus V$ is a cube iff the canonical basis of V is a subset of the standard basis of $\{0,1\}^n$.*

Figure 2 shows a cube $A = a \oplus V$, where the canonical basis $\{10000, 01000, 00100\}$ of V is a subset of the standard basis of $\{0,1\}^5$. Note that, in general, the canonical basis of a pseudocube is not a subset of $\{e_1, \ldots, e_n\}$ (see, for example, Fig. 1).

We show here some properties of the canonical bases.

Proposition 2 *Let $A = \alpha_A \oplus V$ be an affine space over the vector space V of canonical basis $v_1, \ldots, v_k$, and $dim(V) = k$. The following properties holds:*

1. *The canonical basis is a basis for the vector space V;*
2. *The canonical basis B_A is the unique basis of V such that for each vector v in B_A the variable corresponding to the first 1 from left in v has value 0 in all the other vectors in B_A;*

Let A and A' be two affine spaces over the same vector space V, they either coincide or are disjoint:

Proposition 3 *Let $A = \alpha \oplus V$ and $A' = \alpha' \oplus V$ be two affine spaces over the vector space V.*

$$A \equiv A' \Leftrightarrow \alpha \oplus \alpha' \in V .$$

If A and A' do not coincide then $A \cap A' = \emptyset$.

Proof. First note that $A = \alpha \oplus V = \alpha' \oplus V = A' \Leftrightarrow \alpha \oplus \alpha' \oplus V = V \Leftrightarrow \alpha \oplus \alpha' \in V$. Finally we have to show that if A and A' do not coincide then they mast be disjoint. By confutation, let α_0 be a common point of A and A', by Definition 1, A and A' are both equal to $\alpha_0 \oplus V$, thus $A \equiv A'$. ∎

Finally, some simple properties of affine space are in the following:

Proposition 4 *Let $A = p \oplus V$ and $A' = p' \oplus V'$ two affine spaces:*

1. *$A \subseteq A' \Leftrightarrow V \subseteq V'$ and $p \oplus p' \in V'$,*
2. *$A = A' \Leftrightarrow V = V'$ and $p \oplus p' \in V'$.*

The Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ with value 1 in the points of a pseudocube A (i.e., the characteristic function of A) is a *pseudoproduct*, and can be expressed as a product of EXOR factors in several different forms, one of which is called the *canonical expression* (briefly *CEX*) of A. For the non trivial definition of the CEX expression see [13]. For example the pseudoproduct $(x_1 \oplus x_2 \oplus \overline{x}_4)(x_3 \oplus x_5)$ is the CEX expression of the pseudocube in Fig. 1. In the following section we describe how to obtain the CEX expression of a pseudocube from its canonical representation.

2.2 CEX expressions

The characteristic function of an affine space can be expressed in various ways as a products of EXORs. In [13] is described a canonical AND-of-EXOR form (CEX) for a given pseudocube.

Let $A = \alpha_A \oplus V$ be a pseudocube with canonical basis $v_1, \ldots, v_k$. For all v_i let x_{j_i} the variable corresponding to its first 1-component from left, x_{j_i} is called *canonical variable* associated to v_i. For example the first 1-component from left of 000101101 is the 4th component, thus the corresponding canonical variable is x_4. The variables that are not canonical for any vector in the canonical basis are called *non canonical.* For example in Fig. 1, the basis is $v_1 = 00101$, $v_2 = 01010$, $v_3 = 10010$, the canonical variables are: x_3 from vector v_1, x_2 from vector v_2, x_1 from vector v_3; the non canonical variables are, finally x_4 and x_5. Recall that, by Proposition 2.2, the first 1-component from left of a vector in the canonical basis corresponds to a 0-component in any other vector of the basis. Therefore the canonical variables have all the possible configurations in V.

Note that in a cube the non canonical variables are constant in A, and are always set to 0 in V (in Fig. 2, x_4 and x_5 are the non canonical variables). Observe that a vector of the basis states the connection between a canonical variable (the one associated with it) and some non canonical variables (the ones that have an 1 in the vector). For example a vector 01001101 states that the non canonical variables x_5, x_6 and x_8 are connected with the canonical variable x_2. In fact in the CEX expression there is an EXOR factor for any non canonical variable, and in the EXOR factor corresponding to the non canonical variable x_i there are all the canonical variables (if any) connected with x_i. A non canonical variables is complemented iff its corresponding component in α_A is 0.

For example let $A = \alpha_A \oplus V$ be a affine space with $B_A = \{001010, 010110, 100000\}$ and $\alpha_A = 000100$. The first vector in B_A shows that the canonical variable x_3 is in the EXOR factor corresponding to the non canonical variable x_5. Vector 010110 shows that the canonical variable x_2 is in the EXOR factors corresponding to the non canonical variables x_4 and x_5. Vector 100000 shows that the canonical variable x_1 does not appear in any EXOR factor. The vector $\alpha_A = 000100$ shows that the non canonical variable x_4 is directed meanwhile x_5 and x_6 are complemented. Therefore the CEX expression is $(x_2 \oplus x_4)(x_2 \oplus x_3 \oplus \overline{x}_5)\overline{x}_6$. It is easy to verify that this AND-of-EXOR expression is the CEX defined in [13], thus it is a characteristic function of the pseudocube A. Recall that the CEX expression has several properties:

1. $CEX(P) = E_1 \cdot E_2 \cdot \ldots \cdot E_k$ where E_i is an EXOR of literals (possibly a single literal),
2. each E_i contains exactly one non canonical variable directed or complemented, this variable is the one with higher index in E_i,
3. each non canonical variable is in an unique E_i (note that the non canonical variables are k in number),
4. the canonical variables (always directed) can be in any EXOR factor E_i or may non appear in the expression.

For example

$$CEX(P) = x_1 \cdot (x_2 \oplus \overline{x}_4) \cdot (x_2 \oplus x_3 \oplus x_6).$$

is a canonical expression where x_1, x_4 and x_6 are the non canonical variables (i.e., the variables with higher index in each EXOR factor), and x_2, x_3 and x_5 are the canonical ones.

2.3 The Equivalence Theorem

In this section we prove that the set of pseudocubes (defined in [13]) and the set of affine spaces (defined here) coincide. Paper [13] shows that a pseudocube P can be represented by an unique CEX(P) expression (up to a fixed variable order). CEX expression is not, in general, the unique product-of-EXOR's expression which represents a given pseudocube. A general property of the algebraic representation of pseudocubes is given in the following

Lemma 1 *In a Boolean space* $\{0,1\}^n$*:*

1. *the EXOR factor of any subset of variables (directed or complemented) represents a pseudocube of degree* $n-1$*;*
2. *the product of* $k \leq n$ *arbitrary EXOR factors represents either an empty set or a pseudocube of degree* $\geq n-k$*.*

Proof. Point 1 of can be easily proved by induction on the number of variables in the EXOR factor. Point 2 then follows from a theorem of [13] which states that the intersection of two pseudocubes of degrees p, r is either empty, or is a pseudocube of degree $\geq p+r-n$. ■

Theorem 1 *The set of pseudocubes in* $\{0,1\}^n$ *defined in [13] coincides with the set of affine subspaces over the vector space* $(\{0,1\}^n, \oplus)$.

Proof. Observe that the points of a pseudocube P (with $CEX(P) = E_1 \cdot E_2 \cdot \ldots \cdot E_k$) are the solutions of the following system of EXOR equations:

$$S_0 = \begin{cases} E_1 = 1 \\ \quad\vdots \\ E_k = 1 \end{cases}$$

The rank of the system is k because each E_i contains a non canonical variable that is not in any other E_j with $j \neq i$. Note that $E_i = (x_{i_1} \oplus \ldots \oplus \widehat{x}_{i_h})$ were $i_h > i_j$ with $j = 1, \ldots, h-1$, and $\widehat{x}$ is either x or $\overline{x}$. We denote $P_i = c_{1,i}x_1 \oplus \ldots \oplus c_{n,i}x_n$, where $c_{j,i} = 1$ iff x_j appears in E_i. It is straightforward to show that $(E_i = 1) \Leftrightarrow (P_i = b_i)$, where b_i is a Boolean variable, and $b_i = 0$ iff x_{i_h} is complemented in E_i. The new equivalent system

$$S = \begin{cases} P_1 = b_1 \\ \quad\vdots \\ P_k = b_k \end{cases}$$

is a system of linear equations in $\{0,1\}^n$ with the EXOR, $\oplus$, operator. The solutions of S are still the points of the pseudocube. As known [4] such a system specifies an affine subspace over the vector space $\{0,1\}^n$ (n is the number of variables of S).

It remains to show that any affine subspace of $\{0,1\}^n$ is a pseudocube. Any affine subspace of $\{0,1\}^n$ with the $\oplus$ operator and dimension $n-k$ is the set of solutions of a system of linear equations of rank k. Furthermore, any system of linear equations of rank k is equivalent to a product of k EXOR factors, which represents, by lemma 1, a pseudocube of degree $\geq n-k$. Indeed, it is easy to verify that if k is the rank of the system then the pseudocube has exactly degree $n-k$. ■

For example the CEX expression

$$CEX(P) = x_1 \cdot (x_2 \oplus \overline{x}_4) \cdot (x_2 \oplus x_3 \oplus x_6).$$

corresponds to the system:

$$\begin{cases} x_1 & = 1 \\ x_2 \oplus \overline{x}_4 & = 1 \\ x_2 \oplus x_3 \oplus x_6 & = 1 \end{cases} = \begin{cases} 1x_1 \oplus 0x_2 \oplus 0x_3 \oplus 0x_4 \oplus 0x_5 \oplus 0x_6 & = 1 \\ 0x_1 \oplus 1x_2 \oplus 0x_3 \oplus 1x_4 \oplus 0x_5 \oplus 0x_6 & = 0 \\ 0x_1 \oplus 1x_2 \oplus 1x_3 \oplus 0x_4 \oplus 0x_5 \oplus 1x_6 & = 1 \end{cases}$$

which represents an affine space $\alpha \oplus V$ where $\alpha = 100001$ and a basis of V is $\{000010, 001001, 010101\}$.

The direct correspondence between pseudocubes and pseudoproducts (described in Section 2.2) allows us to rephrase the SPP minimization problem as: "Given a set of points in the Boolean space $\{0,1\}^n$, find its minimal cover composed of affine spaces". Theorem 1 is also a interesting "bridge" between SPP forms and *Linearized Disjunctive Normal Forms* (or LDNF) introduced in [1]. Indeed, LDNF are SPP forms where the pseudoproducts are not, in general, in CEX form. In [1] the author gives interesting results on upper and lower bounds on the number of literals in minimal LDNFs. Unfortunately, no minimization algorithm is proposed.

3 SPP Minimization Algorithm

In this section we describe a new exact SPP minimization algorithm using the canonical representation of pseudocubes. We first describe how to perform the union of two affine spaces over the same vector space. Second we show the *compressed partition tries* (CPT) as the fundamental data structure to represent affine spaces and to efficiently implement the union strategy. Finally, we give a new SPP minimization algorithm based on CPTs.

3.1 The union algorithm

A well known Boolean property states that the union (or sum) of two products is a product if and only if they are composed by the same literals and differ by exactly one complementation. This also implies that the corresponding subcubes are disjoint. For example $x_1x_3\overline{x}_4x_7 + x_1\overline{x}_3\overline{x}_4x_7 = x_1\overline{x}_4x_7$. By Proposition 3 and some results in [3], the corresponding property of affine spaces is the following:

Proposition 5 *The union of two affine spaces (pseudocubes) $A = \alpha_A \oplus V$ and $A' = \alpha_{A'} \oplus V'$ is an affine space (pseudocube) iff $V \equiv V'$.*

In the following proposition we describe how to perform the union of $A = \alpha_A \oplus V$ and $A' = \alpha_{A'} \oplus V$.

Proposition 6 *Let $A = \alpha \oplus V$ and $A' = \alpha' \oplus V$ be two distinct affine spaces (i.e., $A \neq A'$) over the vector space V of basis $v_1, v_2, \ldots, v_k$. The union $B = A \cup A' = \alpha \oplus \ V' = \alpha' \oplus \ V'$ is an affine space where V' is the $(k+1)$-dimensional vector space of basis $v_1, v_2, \ldots, v_k, \alpha \oplus \alpha'$.*

Proof. By Proposition 3, A and A' are disjoint. First we prove that the vector $\alpha \oplus \alpha'$ is linearly independent by $v_1, v_2, \ldots, v_k$ (i.e., $\alpha \oplus \alpha' \notin V$). By confutation let $\alpha \oplus \alpha'$ be linearly dependent by $v_1, v_2, \ldots, v_k$ (i.e., $\alpha \oplus \alpha' \in V$) in other words there exists $v \in V$ such that $\alpha = \alpha' \oplus v$, and then $A = A'$.

We have that $B = \alpha \oplus V'$ is an affine space of dimension $k+1$. It remains to show that each $\beta \in A \cup A'$ is in B. Let $\beta \in A$, then $\beta = \alpha \oplus v$ with $v = c_1v_1 \oplus \ldots \oplus c_kv_k \in V$, we have that $\beta = \alpha \oplus c_1v_1 \oplus \ldots \oplus c_kv_k \oplus 0(\alpha \oplus \alpha')$ hence $\beta \in B$. Let now $\beta \in A'$, then $\beta = \alpha' \oplus v$ with $v = c_1v_1 \oplus \ldots \oplus c_kv_k \in V$, we have that $\beta = \alpha \oplus c_1v_1 \oplus \ldots \oplus c_kv_k \oplus 1(\alpha \oplus \alpha')$ hence $\beta \in B$ (recall that $\alpha \oplus \alpha = 0$ and $0 \oplus \alpha = \alpha$). Since A and A' are disjoint, and B has dimension $k+1$ the thesis immediately follows. ■

Proposition 6 shows that the union of two distinct affine spaces over the same vector space is very easy to implement. The only practical problem is: if the vectors $v_1, v_2, \ldots, v_k$ are a canonical basis, then the basis $v_1, v_2, \ldots, v_k, \alpha \oplus \alpha'$ is not, in general, canonical. The definition of canonical basis suggests to generate all the points of the union from $v_1, v_2, \ldots, v_k, \alpha \oplus \alpha'$ and reorder them; it is straightforward that this strategy is very expensive (i.e., exponential in the dimension of $A \cup A'$). Fortunately, there is a simple (i.e., linear in the dimension of $A \cup A'$) transformation called here *normalization*, that computes the canonical basis from $v_1, v_2, \ldots, v_k, \alpha \oplus \alpha'$. We first describe the normalization procedure using an example.

Example 1 (Normalization) *Let $A = 0001 \oplus V$ and $A' = 0100 \oplus V$ be two distinct affine spaces over the vector space V of basis $\{0010, 1100\}$. The new vector in the basis of the union $A \cup A'$ is, by Proposition 6, $0001 \oplus 0100 = 0101$. Thus the basis is $\{0010, 0101, 1100\}$. Note that, by Proposition 2.2, this is not the canonical basis; indeed the variable x_2, that corresponds to the first 1 from left of 0101, has value 1 in 1100. To obtain the canonical basis we need only to replace 1100 with $0101 \oplus 1100 = 1001$, the canonical basis is then $\{0010, 0101, 1001\}$.*

The strategy of the normalization algorithm is to identify the vectors in the basis of A that have an 1 corresponding to the first 1-component from left of $\alpha \oplus \alpha'$. The algorithm replaces such vectors with their EXOR with $\alpha \oplus \alpha'$. It is easy to verify that the resulting basis is the canonical one. Given two distinct affine spaces $A = \alpha_A \oplus V$ and $A' = \alpha_{A'} \oplus V$ over the same vector space V of canonical basis $\{v_1, \ldots, v_k\}$, the canonical representation ($B_{A \cup A'}$ and $\alpha_{A \cup A'}$) of the union $A \cup A'$ is given by the following:

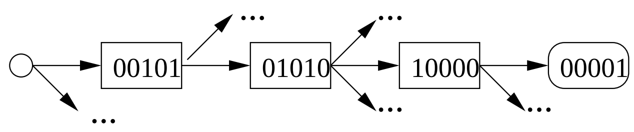

Figure 3: A path in a s-CPT representing the pseudocube A with $\alpha_A = 00001$ (in the leaf) and $B_A = \{00101, 01010, 10000\}$.

Algorithm 1 ***Union*** *(build $A \cup A'$ from $A = \alpha_A \oplus V$ and $A' = \alpha_{A'} \oplus V$)*

1. $\alpha_{A \cup A'} = \min(\alpha_A,\, \alpha_{A'})$
2. $B_{A \cup A'} = Norm(B_A, \alpha_A \oplus \alpha_{A'})$

The function $\min(\alpha_A,\, \alpha_{A'})$ gives the minimum vector, between α_A and $\alpha_{A'}$, in binary order, and $Norm(B, a)$ builds a canonical basis as union of the canonical basis B and the new vector a. The time required by algorithm Union is linear in the size of the basis B_A. Note that Algorithm 1 is simpler to implement and understand than the union algorithm proposed in [3].

3.2 Compact Partition Tries

Partition tries (PT) are the basis of the minimization method described in [3]. In this section we describe a new data structure, called *Compact Partition Trie (CPT)*, that uses canonical bases and transposition points, instead of CEX expressions, for representing pseudocubes, and is more compact than PT.

Given a set of affine spaces in canonical representation we are interested in a data structure that stores each canonical bases only ones. The simpler data structure we can think about is the following, which we call s-CPT (for simple CPT). The internal nodes (except for the root) and the leaves of a s-CPT are labeled with a Boolean vector. The root has no label. In a s-CPT, any path from root to leaf represents an affine space, where the internal nodes are the canonical basis (in binary order), and the leaf is labeled by the canonical transposition point.

For example the path in figure 3 represents the pseudocube A with $\alpha_A = 00001$ (in the leaf) and $B_A = \{00101, 01010, 10000\}$. The main property of s-CPT is:

Property 1 *Any two leaves with the same parent represent affine spaces over the same vector space.*

Therefore, the union can be performed only between leaves with common parents. We call this kind of trees compact partition tries because a set of affine spaces is partitioned into disjoint subsets containing affine spaces over the same vector space (the leaves with the same parent); and because they are particular tries (see [23] for details on tries).

Recall that a PT represents CEX expressions: each internal node of a PT is labeled with a variable in the CEX expression. The canonical representation (with bases) of a pseudocube is in general more compact than the CEX ones. For example the CEX expression of the pseudocube in Fig. 3 is $(x_2 \oplus x_4)(x_3 \oplus x_5)$ that requires four internal nodes (there are 4 variables) plus the leaf, meanwhile the new representation requires only three internal nodes (the basis) and the leaf. On the other hand, there are some pseudocubes that can be represented more compactly with the former representation. In fact, the cube in Fig. 2 has CEX expression $\overline{x}_4 x_5$, when the basis has three vectors. In order to guarantee an always more compact representation we define the CPT data structure by inserting in a s-CPT a new node (called *occurrences node*), which has a component set to 1 iff it corresponds to the first 1 from left in a vector of the basis (i.e., the occurrence node has a component set to 1 iff it corresponds to a canonical variable), and deleting from a s-CPT each vector in the basis with only one component set to 1, since we can represent it using the occurrences node. For example, let $A = \alpha_A \oplus V$ be an affine space with canonical basis $B_A = \{1001, 0100, 0011\}$, which is represented in a CPT with an occurrence node 1110 and the two vectors of the basis 1001 and 0011. Note that the node 0100 of the basis can be easily detected since it is the corresponding vector to the only 1-occurrence in the occurrences vector (the second 1 from left) that does not correspond to any vector in the basis. In facts, the first 1-occurrence in 1110 corresponds to 1001 and the third 1-occurrence corresponds to 0011. For an other example the cube in Fig. 2 can be represented only by the occurrence node 11100 and the leaf 00001, without representing the basis. Observe that, in the case of cubes, this is an equivalent representation of meta-products introduced by Coudert and Madre [6]. Figure 4 shows the corresponding CPT path of the s-CPT in Figure 3. The occurrences node is the first, and the vector 10000 of the basis is

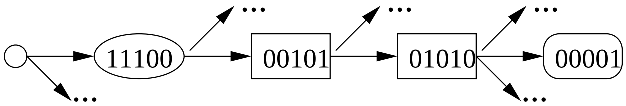

Figure 4: A path in a CPT representing the pseudocube A with $\alpha_A = 00001$ (in the leaf) and $B_A = \{00101, 01010, 10000\}$. The first node is the occurrence node, and the vector 10000 of the basis is not in the path.

no more in the path. Finally, from some properties of CEX expressions (see [13, 3]) we have that CPTs are more compact than PTs:

Proposition 7 *The length of any path, in a CPT, is less than or equal to the length of the corresponding PT path.*

The insertion and the search in a CPT are performed in the natural way. Clearly, Property 1 still holds for CPTs. Compact partition tries will be used to represent pseudoproducts in the minimization procedures. By Proposition 7 we have proved that the CPT data structure is more efficient than PT, furthermore experimental evidences (in Section 5) show that the average minimization time is reduced by 23%.

3.3 The SPP minimization method

We focus our attention to the synthesis of SPP forms with minimal number of literals. As usual pseudoproducts will be built by pairwise composition of pseudoproducts of smaller degree, but we store pseudoproducts in a much more efficient way than before. As usual the number of literals in a CEX expression may depend on the ordering of the variables, therefore minimization holds relatively to a chosen order. The critical novelty of our algorithm is the data structures used in the construction of the prime pseudocubes set, now based on compact partition tries. We then have the exact algorithm:

Algorithm 2 ***SPP minimization***

1. insert the points of the function as leaves of the root in the CPT P_0 (an affine space A containing a single point correspond to α_A);
2. for increasing k, generate a *new* CPT P_{k+1} from the union of all the leaves that have common parent in P_k; discard from P_k a pseudocube if its CEX expression contains h literals, and is combined into one of degree $k+1$ whose expression contains $\leq h$ literals;
3. make a selection with minimal number of literals from among the pseudocubes retained in step 2 (i.e. solve the set covering problem).

The algorithm is the same proposed in [3] with a different data structure. The correctness of the algorithm is straightforward. This algorithm is compared with the one proposed in [3], which uses PT, in Section 5.

3.4 Example of SPP minimization (algorithm 2)

Let $f = \{0000, 0011, 0101, 0110, 1101\}$ be a Boolean function in $\{0,1\}^4$. Figure 5 shows the Karnaugh map of f. In this section we show the SPP minimization procedure on function f. Figure 6 describes the compressed partition tries used for the primes' generation. The first step of algorithm 2 inserts the points $0000, 0011, 0101, 0110, 1101$ in the partition trie P_0, and the corresponding occurrence vector that is 0000 since affine spaces with only one point have no basis. We now perform the second step. Since all the leaves in P_0 have the same parent, they are pairwise unified and inserted in P_1 (see figure 6):

1. the affine space with basis 0011 and canonical translation point 0000 is the union of 0000 and 0011. The occurrences vector is 0010.
2. the affine space with basis 0101 and canonical translation point 0000 is the union of 0000 and 0101. The occurrences vector is 0100.
3. the affine space with basis 0110 and canonical translation point 0000 is the union of 0000 and 0110. The occurrences vector is 0100.

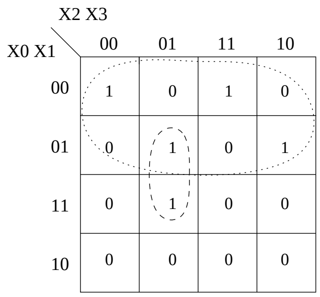

Figure 5: Karnaugh map of a Boolean function f. The sum of the two highlighted pseudocubes represents a minimal SPP form of f: $\overline{x}_0 \cdot (x_1 \oplus x_2 \oplus \overline{x}_3) + x_1 \cdot \overline{x}_2 \cdot x_3$.

4. the affine space with basis 1101 and canonical translation point 0000 is the union of 0000 and 1101. The occurrences vector is 1000.

5. the affine space with basis 0110 and canonical translation point 0011 is the union of 0011 and 0101. The occurrences vector is 1000.

6. the affine space with basis 0101 and canonical translation point 0011 is the union of 0011 and 0110. The occurrences vector is 0100.

7. the affine space with basis 1110 and canonical translation point 0011 is the union of 0011 and 1101. The occurrences vector is 1000.

8. the affine space with basis 0011 and canonical translation point 0101 is the union of 0101 and 0110. The occurrence vector is 0010.

9. the affine space with basis 1000 and canonical translation point 0101 is the union of 0101 and 1101. The occurrences vector is 1000. Note that in the vector of the basis 1000 in not in the CPT because in 1000 there is a unique 1-occurrence.

10. the affine space with basis 1011 and canonical translation point 0011 is the union of 0110 and 1101. The occurrences vector is 1000.

We mark for deletion each affine spaces in P_0. That is, any pseudocube in P_0 contains 4 literals, and is combined into, at least, one pseudocube with CEX expression of ≤ 4 literals. Note that P_1 in figure 6 has only three paths with more than a single leaf. We unify the pseudocubes corresponding to the leaves with the same parent and we obtain the affine space with basis $0011, 0101$ and canonical translation point 0000 from all the three unions. We mark for deletion the pseudocubes unified.

Finally, we discard the market leaves. The set of prime pseudocubes is the union of pseudocubes in the subtree of P_1 marked with a dotted line in figure 6 and the affine space with basis $0011, 0101$ and canonical translation point 0000 (the pseudocube in P_2). The set covering step (step 3) of the algorithm gives the minimal SPP form: $\overline{x}_0(x_1 \oplus x_2 \oplus \overline{x}_3) + x_1\overline{x}_2x_3$.

Note that the number of comparisons to generate P_2 is 3. Meanwhile algorithm in [13] performs $9 \cdot 10/2 = 45$ pairwise comparisons. The total number of internal nodes in the corresponding PTs is 29, meanwhile using CPTs we need only 13 internal nodes. The number of leaves in PT and CPT are clearly equal, since each leaf represents a pseudoproduct.

4 k-SPP forms and minimization

Although many practical logic functions are well-suited for EXOR forms, SPP networks require EXOR gates with unbounded fan-in. In most technologies, EXOR gates with many inputs are slow, expensive and often impractical [25]. The problem with the fan-in of the EXOR gates can be overcome by fixing the maximum number of literals in the EXOR factors. Consequently, in this section we introduce the *k-SPP forms* where each EXOR factor contains at most k literals.

In a Boolean space $\{0,1\}^n$ described by n variables, let a *k-EXOR factor* $(1 \leq k \leq n)$ be an EXOR factor with at most k literals. For example x_0, $(x_0 \oplus x_2 \oplus \overline{x}_3)$ and $(x_2 \oplus x_4 \oplus x_5 \oplus x_7)$ are 4-EXOR factors.

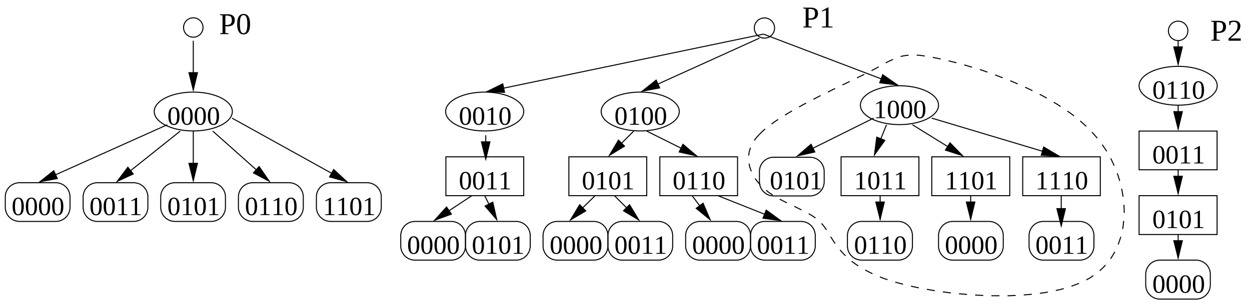

Figure 6: Example of generation of prime pseudoproducts using compact partition tries (P_0, P_1 and P_2).

A *k-pseudoproduct* is a product of k-EXOR factors, and a *k-SPP form* is a sum of k-pseudoproducts. For example, $(x_0 \oplus \overline{x}_1) \cdot x_4 \cdot (x_0 \oplus x_3 \oplus \overline{x}_6) + x_4 \cdot \overline{x}_3 + (x_0 \oplus x_2 \oplus x_3) \cdot (x_2 \oplus x_4 \oplus x_5)$ is a 3-SPP form. A k-pseudoproduct P of a Boolean function F is *prime* iff no other k-pseudoproduct P' of F exists such that $P \subseteq P'$. A *k-pseudocube* is a pseudocube having a k-pseudoproduct as CEX expression. Of course, a k-pseudocube is an affine space. We now characterize k-pseudocubes ($1 \leq k \leq n$) as particular affine spaces. This is a generalization of concepts of cube (i.e., 1-pseudocube) and pseudocube (i.e., n-pseudocube). In particular SP form is a 1-SPP form, and each SPP form in $\{0,1\}^n$ is an n-SPP form.

Let $A = \alpha_A \oplus V$ be an affine space of dimension m. If x_i is a non canonical variable, denote with u_i ($1 \leq u_i \leq m$) the number of 1-occurrences in the column corresponding to x_i in the matrix of B_A. For example let $B_A = \{000011, 011001, 100001\}$ be the basis of A, the non canonical variables are x_3, x_4 and x_6, and $u_3 = 1$, $u_4 = 0$ and $u_6 = 3$. In the CEX expression of A, the EXOR factor corresponding to the non canonical variable x_i contains $u_i + 1$ literals (the canonical variables associated to x_i plus x_i). Therefore we can give a characterization of k-pseudocubes:

Definition 7 *A k-*pseudocube *is an affine space with $u_i + 1 \leq k$ for all u_i.*

The following theorem gives the essential tool for transforming SPP minimization algorithms into efficient k-SPP algorithms:

Theorem 2 *For any k-pseudocube $A = \alpha_A \oplus V$ of dimension d, $1 < d \leq n$, there exist two k-pseudocubes $A_1 = \alpha_{A_1} \oplus V'$ and $A_2 = \alpha_{A_2} \oplus V'$ of dimension $d-1$ such that $A = A_1 \cup A_2$.*

Proof. Let B_A the canonical basis of A. Let $v_i \in B_A$ be a vector of the canonical basis. Note that $B_A \backslash v_i$ is a canonical basis of a subspace of V of dimension $d-1$. Let $\alpha_{A_1} = \alpha_A$, $\alpha_{A_2} = v_i \oplus \alpha_A$, and $V' = B_A \backslash v_i$. Our task is now to prove that $A = A_1 \cup A_2$ with $A_1 = \alpha_{A_1} \oplus V'$ and $A_2 = \alpha_{A_2} \oplus V'$. Following the union strategy we have that the new vector in the basis of the union is $\alpha_{A_1} \oplus \alpha_{A_2} = \alpha_A \oplus \alpha_A \oplus v_i = v_i$, and $B_A \backslash v_i \cup \{v_i\} = B_A$ is the canonical basis of A. Since $v_i \oplus \alpha_A$ and α_A are in A, we have that $\min(\alpha_{A_1}, \alpha_{A_2}) = \min(v_i \oplus \alpha_A, \alpha_A) = \alpha_A$. It remains to show that A_1 and A_2 are k-pseudocubes. The essential observation is that the basis $B_A \backslash v_i \cup \{v_i\}$ is equal to B_A without any normalization. The number of 1-occurrences in each column can not increase in $B_A \backslash v_i$. The thesis immediately follows. ■

Note that a pseudocube $A' \subset A$ can exists such that A is a k-pseudocube, and A' is h-pseudocube with $h > k$. For example we have $CEX(A) = (x_3 \oplus \overline{x}_4) \cdot (x_1 \oplus \overline{x}_6)$ (i.e., A is a 2-pseudocube), $CEX(A') = (x_1 \oplus x_2 \oplus \overline{x}_3) \cdot (x_1 \oplus x_2 \oplus \overline{x}_4) \cdot (x_1 \oplus \overline{x}_6)$ (i.e., A' is a 3-pseudocube), and $A' \subset A$. The bases are $B_A = \{100001, 010000, 001100, 000010\}$, and $B'_A = \{101101, 011100, 000010\}$; the transposition points are $\alpha_A = 000000$, and $\alpha_{A'} = 000000$. However, by theorem 2, there exists two 2-pseudocube $A_1 = \alpha_{A_1} \oplus V'$ and $A_2 = \alpha_{A_2} \oplus V'$ that are sub-pseudocubes (with dimension 3) of A. We now compute the canonical representation of A_1 and A_2 following the construction in the proof of the theorem. Let v_i be the vector 00010 of the basis B_A. The canonical basis of V' is $B_{A_1} = B_A \backslash v_i = \{100001, 010000, 001100\}$. The canonical transposition points are $\alpha_{A_1} = \alpha_A = 000000$ and $\alpha_{A_2} = \alpha_A \oplus v_i = 000010$.

Note that in k-SPP minimization we cannot generate all the prime pseudoproducts, and then select the primes k-pseudoproducts between them, because the set of prime k-pseudoproducts is, in general, not contained in the set of prime pseudoproducts. For instance, let F_p the parity function in B^n (i.e., the set of minterms with a even number of non complemented variables) in the Boolean space $\{0,1\}^n$. The SPP minimal form for F_p is $(x_1 \oplus x_2 \oplus \ldots \oplus x_{n-1} \oplus x_n)$, which is also the unique prime pseudoproduct. Clearly this is not a k-SPP form with $k < n$, and the set of prime k-pseudoproducts is not included in the set $\{(x_1 \oplus x_2 \oplus \ldots \oplus x_{n-1} \oplus x_n)\}$.

Each SPP synthesis algorithm that computes the set of prime pseudoproducts from minterms performing successive unions, can be transformed into a k-SPP minimization algorithm discarding the h-pseudocubes with $h > k$ after each union. By Theorem 2, following this strategy we are not loosing any prime k-pseudoproduct. Therefore we have:

Algorithm 3 ***k-SPP minimization***

1. start from pseudocubes of dimension $i = 0$ (single points);
2. for increasing i:
   (a) generate the pseudocubes of dimension $i + 1$ using a union algorithm (for instance Algorithm 1);
   (b) discard a pseudocube of degree i if its CEX expression contains j literals, and is combined into one of dimension $i + 1$ whose expression contains $\leq j$ literals;
   (c) *discard a pseudocube of dimension $i + 1$ if is not a k-pseudocube*;
3. make a selection with minimal number of literals from among the k-pseudocubes retained in step 2 (i.e. solve the set covering problem).

Proposition 8 *Algorithm 3 computes the exact k-SPP minimal form.*

Proof. The essential observation is that in Step 2.c we discard the h-pseudocubes with $h > k$ after each union, but we are not loosing any k-pseudocube. In fact, by theorem 2 each k-pseudocube (in particular, a *prime* k-pseudocube) P of dimension d is computed as the union of two k-pseudocubes of dimension $d - 1$. Therefore, in the set covering step we select from among all the prime k-pseudocubes, and we have the k-SPP minimal form as output. ■

Since in step 2.c of Algorithm 3 we discard h-pseudocubes with $h > k$, the number of the overall k-pseudocubes generated is, in general, smaller than the number of pseudocubes. Therefore, the algorithm performs a reduced number of unions.

In conclusion a new three level logic form has been presented, that seems to combine the compactness of SPP forms and the technological requirement of EXOR gates with a limited number of inputs. We have described a general method to transform SPP (union-based) minimization algorithms into efficient k-SPP minimization algorithms, without loosing the optimal k-SPP solution. The experimental results, described in the next section, show that the limitations on the fan-in do not significantly affect the compactness of the form, but the minimization time for example of 2-SPP forms is drastically reduced. Therefore k-SPP expressions are particularly compact (even 2-SPP are very small) that can be efficiently generated (2-SPP minimization requires 92% less time that standard SPP synthesis). Observe that k-SPP forms have, beside EXORs of bonded fan-in, a limited number of distinct EXOR gates. Indeed in a k-SPP form we can have at most $s = \sum_{i=2}^{k} \binom{n}{i}$ distinct EXOR factors. Recall that $\sum_{i=2}^{k} \binom{n}{i} \in O(n^k)$ and k is a constant. Note also that in a 2-SPP we can have at most $n(n-1)/2$ different EXOR factors.

5 Experimental results

Our minimization method has been tested on a range of functions taken from the Espresso benchmark suite [26]. Note that the theory of SPP forms is here developed only for single-output functions because it can be easily extended to multi-output functions using the same strategies developed for two-level forms [22].

We first compare the exact minimization method described in Algorithm 2 with the one in [3]. Since the two algorithms generate the same set of pseudoproducts, the time needed for set covering procedures coincide. Table 1 compares the CPU time for the construction of the primes pseudoproducts set for the two minimization methods. CPU times are reported in seconds on a Pentium III 450 machine. These examples show the great increase of speed of algorithm 2 versus the original one. The new algorithm requires in average 23% less time that the one in [3]. Since we used some heuristics in solving the set covering problem, the number of literals and factors in the expressions are upper bounds for the minimal solution.

Table 2 compares SPP (alg. 2) with 2-SPP (alg. 3) forms. Note that, although we allow only 2-fan-in EXOR gates, the size of the minimal forms is not significantly greater than the one for unbounded fan-in, meanwhile the minimization time is drastically reduced.

The result is very interesting: since the prime 2-pseudoproducts are, on the average, **78%** fewer than prime pseudoproducts, 2-SPP minimization requires, on the average, **92%** less time than the SPP algorithm,

function	#Primes	Time alg. of [3]	Time alg. 2	%Gain
addm4	191133	440.17	321.64	27%
ex5	273680	53747.97	35739.16	34%
f51m	5779	81.36	64.84	23%
l8err	17008	28.68	22.51	22%
m4	110198	10960.84	3206.24	71%
newapla1	37	29.10	10.94	62%
newcond	46889	8235.47	7724.55	6%
prom2	341557	104.23	75.04	28%
test1	444407	155.57	147.37	5%

Table 1: CPU times (in seconds) of the two minimization algorithms for the construction of the set of prime pseudoproducts. #Prime is the total number of Primes. #L is the number of literals in the minimal expression. #PP is the number of Primes in the minimal expression. %Gain is $1-Time(\text{Alg. [3]})/Time(\text{Alg. } 2)$ in percentage.

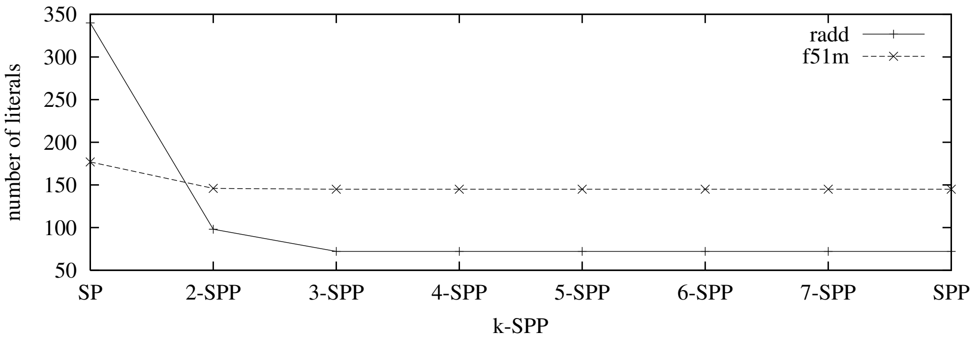

Figure 7: Number of literals in the k-SPP forms of the benchmark functions *radd* and $f51m$.

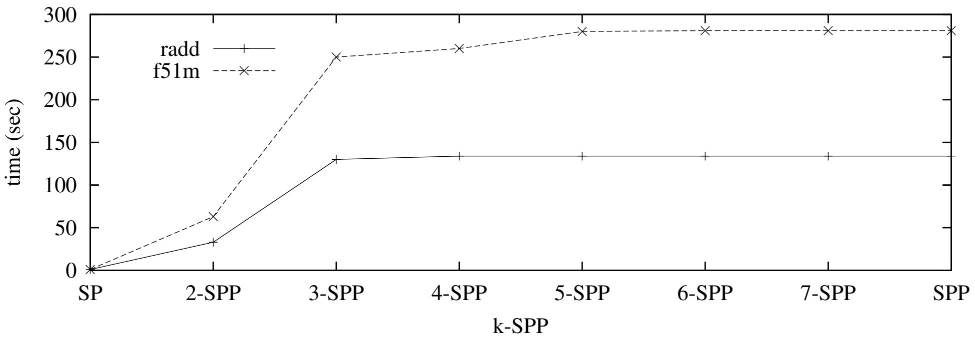

Figure 8: CPU time (in seconds) for the synthesis of the k-SPP forms of the benchmark functions *radd* and $f51m$.

	SPP (alg. 2)				2-SPP (alg. 3)				
function	**#Primes**	**#L**	**#P**	**Time**	**#Primes**	**#L**	**#P**	**Time**	**%Gain**
b11	87	194	45	434	85	194	45	26	94%
ex5	273680	755	254	49086	21335	737	252	1445	97%
m4	110198	645	1224	16527	20195	676	131	815	95%
newcpla1	4933	232	53	6159	834	232	53	117	98%
newtag	8199	17	6	775	458	17	6	29	96%
p1	323922	171	32	10121	22214	180	37	307	97%
p3	*	*	*	*	20951	157	37	350	*
risc	88	199	46	422	86	199	46	26	94%
root	37324	220	39	1974	7196	220	39	247	87%
test1	444407	541	70	15086	71672	580	87	1448	90%
test4	*	*	*	*	239532	900	210	6051	*
sym10	*	*	*	*	26775	302	43	21002	*
Z9sym	*	*	*	*	1260	126	21	345	*

Table 2: Experimental results for SPP vs. the 2-SPP algorithm. #Primes is the total number of prime pseudoproducts (2-pseudoproducts). #L is the number of literals in the SPP (2-SPP) form. #P is the numbers of pseudoproducts (2-pseudoproducts) in the SPP (2-SPP) form. Time is the minimization times (in seconds). %Gain is $1 - Time(2\text{-}SPP)/Time(SPP)$ in percentage. A star * indicates that the SPP algorithm did not terminate after 172800 seconds (2 days).

meanwhile the 2-SPP forms have, on the average, only the **4%** more literals than SPP forms. The number of literals and the CPU time of the SP, k-SPP and SPP forms of the functions *radd* and *f51m* are plotted in figure 7 and 8, respectively. Almost all the functions from the ESPRESSO benchmark suite have analogous plots. The two functions in Figure 7 show that the number of literals of a 2-SPP form and the corresponding SPP form are very closed, meanwhile the SP form is very larger. On the other hand, Figure 8 shows the 2-SPP forms are minimized in reasonable time, meanwhile more-than-2-SPP forms can require much more effort. *Therefore we claim that 2-SPP forms are reasonable lower bounds of the exact SPP forms, and are a good trade off between the compactness of SPP forms and the efficiently of SP minimization. Furthermore 2-SPP forms require a reduced number of EXOR gates, and are practicable for the current technology.*

We have also compared SPP and 2-SPP forms with two widely studied three-level form: EX-SOP [7, 11, 12] and AND-OR-AND [10]. Table 3 shows the number of products (or pseudoproducts) in the minimal SP, SPP, 2-SPP, AND-OR-AND and EX-SOP forms, where the results for EX-SOP and AND-OR-AND are taken from [12] and [10], respectively. This table shows that the size of SPP forms is smaller than the size of EX-SOP forms, while 2-SPP forms can have more products than EX-SOP forms (see function *f51m*). However, in general, the 2-SPP forms are much smaller in size than the corresponding EX-SOP and AND-OR-AND forms (see for example *dist* and *rd37*). The direct comparison between minimization times in table 3, are indeed not significant since the experiments in [11, 12] were run on an Sun Ultra 60 operating on two 360MHz CPU and 768Mb memory, and the experiments for AND-OR-AND were run on a computer with the same configuration except for memory that is 1024Mb. While our experiments are performed on a single 450MHz CPU with 128Mb memory.

6 Conclusions

In this paper we have proved the equivalence between affine spaces and pseudocubes. This new characterization provides a deeper understanding of SPP forms, and a better canonical representation of pseudocubes leading to a more efficient minimization algorithm. This method allows to deal with functions whose number of prime pseudoproducts is too high to be handled by previous algorithms.

The technology requirement of EXOR gates with limited fan-in, and the need of even faster synthesis algorithms give rise to new SPP models with bounded fan-in EXOR gates. Even if we concentrate only on 2-fan-in EXOR gates, the experiments evidence shows that the form is still much compact and the minimization time is drastically reduced.

The new representation of pseudocubes is a direct generalization of the well-known meta-products introduced in [6]. Therefore, our future investigations will be concentrated on implicit minimization algorithms that perform the basic operations directly on BDDs. Since, using BDDs, there is no direct correspondence between the size of the cover of the function and the size of its representation, very large size functions could be minimized efficiently.

	SP		AND-OR-AND		EX-SOP		2-SPP		SPP	
function	**#P**	**Time**	**#P**	**Time**	**#P**	**Time**	**#P**	**Time**	**#P**	**Time**
dist	*109*	1.2	*108*	70	*92*	23	*70*	924	*64*	8196
f51m	*76*	0.90	-	-	*33*	55	*36*	64	*28*	463
life	*84*	0.42	-	-	*60*	200	*18*	120	*18*	262
radd	*61*	0.24	*39*	14	*20*	15	*20*	33	*14*	144
rd53	*19*	0.01	*26*	6	*17*	13	*10*	0.1	*7*	0.2
rd73	*83*	0.11	*79*	312	*62*	494	*32*	24	*22*	114
z4	*45*	0.11	*33*	2	*18*	9	*16*	5	*10*	9

Table 3: Number of products or pseudoproducts ($\#P$) and minimization times (in seconds) for some benchmark functions in the SP, SPP, 2-SPP, AND-OR-AND and EX-SOP forms. A dash - indicates that we do not have the results for AND-OR-AND forms. *Experimental times for EX-SOP and SP forms are from a Sun Ultra 60 operating on two 360MHz CPU and 768Mb memory, and the experiments for AND-OR-AND were run on a computer with the same configuration except for memory that is 1024Mb. While our 2-SPP and SPP experiments are performed on a single 450MHz CPU with 128Mb memory.*

7 Acknowledgment

This work has been supported in part by MURST of Italy. The author would like to thank Fabrizio Luccio, Linda Pagli, Tiziano Villa, and Paolo Ferragina for many useful suggestions; and Marek Perkowski for the fruitful discussion on the EXOR-gates technology. The author is also grateful to Stefano Gliozzi, Tito Ciriani and Stella Catalano for their help in solving the set covering problems. Finally, the author in debt with Anna Bernasconi (who has noticed, for the first time, the relation between affine spaces and pseudocubes) for her useful comments and precious support.

References

[1] A. Aleksanyan. Linearized Disgiuntive Normal Forms of Boolean Functions. In *FCT'87, in LNCS vol. 278*, pages 14–168, 1988.

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

[3] V. Ciriani. Logic Minimization Using Exclusive OR Gates. In *ACM/IEEE 38th DAC*, pages 115–120, 2001.

[4] P. Cohn. *Algebra Vol. 1*. John Wiley & Sons, 1981.

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

[6] O. Coudert and J. C. Madre. A New Method to Compute Prime and Essential Prime Implicants of Boolean Functions. In *MIT Conference on Advanced Research in VLSI and Parallel Systems*, pages 113–128, 1992.

[7] D. Debnath and T. Sasao. Minimization of AND-OR-EXOR Three-Level Networks with AND Gate Sharing. *IEICE Trans. Information and Systems*, E80-D(10):1001–1008, 1997.

[8] D. Debnath and T. Sasao. A Heuristic Algorithm to Design AND-OR-EXOR Three-Level Networks. In *Asia and South Pacific Design Automation Conference*, pages 69–74, 1998.

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

[10] E. Dubrova and P. Ellervee. A Fast Algorithm for Three-Level Logic Optimization. In *Int. Workshop on Logic Synthesis*, pages 251–254, 1999.

[11] E. Dubrova, D. Miller, and J. Muzio. AOXMIN: A Three-Level Heuristic AND-OR-XOR Minimizer for Boolean Functions. In *3rd International Workshop on the Applications of the Reed-Muller Expansion in Circuit Design*, pages 209–218, 1997.

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

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

[14] A. Malik, D. Harrison, and R. Brayton. Three-Level Decomposition with Application to PLDs. In *IEEE ICCD*, pages 628–633, 1991.

[15] E. McCluskey. Minimization of Boolean Functions. *Bell System Technical Journal*, 35:1417–1444, 1956.

[16] P. McGeer, J. Sanghavi, R. Brayton, and A. Sangiovanni-Vincentelli. ESPRESSO-SIGNATURE: A New Exact Minimizer for Logic Functions. *IEEE Transactions on VLSI*, 1(4):432–440, 1993.

[17] M. Perkowski. A New Representation of Strongly Unspecified Switching Functions and its Application to Multi-Level AND/OR/EXOR Synthesis. In *IFIP WG 10.5 Workshop on Applications of the Reed-Muller Expansion*, pages 143–151, 1995.

[18] W. Quine. The Problem of Simplifying Truth Functions. *American Mathematical Monthly*, 59:521–531, 1952.

[19] R. Rudell and A. Sangiovanni-Vincentelli. Multiple Valued Minimization for PLA Optimization. *IEEE Transactions on CAD*, 6(5):727–750, 1987.

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

[21] T. Sasao. A Design Method for AND-OR-EXOR Three Level Networks. In *Int. Workshop on Logic Synthesis*, pages 8:11–8:20, 1995.

[22] T. Sasao. Compact SOP Representations for Multiple-Output Functions: An Encoding Method using Multiple-Valued Logic. In *IEEE International Symposium on Multiple-Valued Logic*, pages 207–212, 2001.

[23] R. Sedgewick and P. Flajolet. *An Introduction to the Analysis of Algorithms.* Addison-Wesley, 1996.

[24] T. Villa, T. Kam, R. Brayton, and A. Sangiovanni-Vincentelli. *Synthesis of Finite State Machines: Logic Optimization.* Kluwer Academic Publishers, 1997.

[25] N. Weste and K. Eshraghian. *Principles of CMOS VLSI Design.* Addison-Wesley Publishing Company, 1993.

[26] S. Yang. Synthesis on Optimization Benchmarks. User guide, Microelectronic Center of North Carolina, 1991. Benchmarks available at ftp://ftp.sunsite.org.uk/computing/general/espresso.tar.Z.