Università di Pisa
Dipartimento di Informatica

Technical Report: TR-01-27

Fast Three-Level Logic Minimization Based on Autosymmetry

Anna Bernasconi, Valentina Ciriani, Fabrizio Luccio, Linda Pagli
Dipartimento di Informatica
University of Pisa
Corso Italia, 40, 56125 Pisa, Italy

December 7, 2001

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

Fast Three-Level Logic Minimization Based on Autosymmetry

Anna Bernasconi, Valentina Ciriani, Fabrizio Luccio, Linda Pagli
Dipartimento di Informatica
University of Pisa
Corso Italia, 40, 56125 Pisa, Italy

December 7, 2001

Abstract

In the framework of SPP minimization, a three level logic synthesis technique developed in recent years, we exploit the "regularity" of Boolean functions to decrease minimization time. Our main results are: 1) the regularity of a Boolean function f of n variables is expressed by its *autosymmetry degree* k (with $0 \leq k \leq n$), where $k = 0$ means no regularity (that is, we are not able to provide any advantage over standard synthesis); 2) for $k \geq 1$ the function is *autosymmetric*, and a new function f_k is identified in polynomial time; f_k is "equivalent" to, but smaller than f, and depends on $n - k$ variables only; 3) given a minimal SPP form for f_k, a minimal SPP form for f is built in linear time; 4) experimental results show that 61% of the functions in the classical ESPRESSO benchmark suite are autosymmetric, and the SPP minimization time for them is critically reduced; we can also solve cases otherwise practically intractable. We finally discuss the role and meaning of autosymmetry.

1 Introduction

The standard synthesis of Boolean functions is performed with Sum of Products (SP) minimization procedures, leading to two level circuits. More-than-two level minimization is much harder, but the size of the circuits can significantly decrease. In many cases three-level logic is a good trade-off among circuit speed, circuit size, and the time needed for the minimization procedure. 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.

Two level minimization is well developed [10, 4, 9]. Several techniques for three level minimization have been proposed for different algebraic expressions. Among them EX-SOP forms, where two SP forms are connected in EXOR [5, 6]; and Sum of Pseudoproducts (SPP) forms, consisting of an OR of pseudoproducts, where a pseudoproduct is the AND of EXOR factors [8]. For example $(x_0 \cdot x_1 + x_0 \cdot \overline{x}_2) \oplus (x_2 \cdot x_3 + x_1 \cdot x_4)$ is an EX-SOP form, and $(x_0 \oplus \overline{x}_1) \cdot x_2 + (x_0 \oplus x_2 \oplus x_3) \cdot (x_2 \oplus \overline{x}_4) + x_1 \cdot x_4$ is an SPP form. Experimental results show that the average size of SPP forms is approximately half the size of the corresponding SP, and SPP forms are also smaller than EX-SOP [2]. As a limit case each EXOR factor reduces to a single literal in SPP, and the SP and SPP forms coincide.

In this work we refer to SPP minimization. Initially this can be seen as a generalization of SP minimization, and in fact an extension of the Quine-McCluskey algorithm was given in [8] for SPP. In particular the pseudoproducts to be considered can be limited to the subclass of *prime* pseudoproducts, that play the same role of prime implicants in SP. The algorithm for SPP, however, was more cumbersome than the former, thus failing in practice in minimizing very large functions. 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 [2]. Still a number of standard benchmark functions can be hardly handled with this technique.

In fact the aim of this paper is to exploit the "regularity" of any given Boolean function, in order to decrease the time needed for its logical synthesis. Our main results are: 1) the regularity of a

X0 X1 \ X2 X3	00	01	11	10
00	0	1	0	1
01	1	1	1	1
11	1	1	1	1
10	1	0	1	0

Figure 1: An autosymmetric function f with $k = 2$, in a Karnaugh map of four variables. The reduced function f_2 is the set of points in the dotted line, and depends only on the variables x_1 and x_3.

Boolean function f of n variables can be expressed by an *autosymmetry degree* k (with $0 \leq k \leq n$), which is computed in polynomial time. $k = 0$ means no regularity, that is we are not able to provide any advantage over standard synthesis; 2) for $k \geq 1$ the function f is said to be *autosymmetric*, and a new function f_k is identified in polynomial time. In a sense f_k is "equivalent" to, but smaller than f, and depends on $n - k$ variables only. A function f with $k = 2$, and its corresponding function f_2, are shown in figure 1; 3) given a minimal SPP form for f_k, a minimal SPP form for f is built in linear time; 4) experimental results show that 61% of the functions in the classical ESPRESSO benchmark suite are autosymmetric: the SPP minimization time for them is critically reduced, and we can also solve cases otherwise intractable. Indeed, although autosymmetric functions form a subset of all possible Boolean functions, a great amount of standard functions of practical interest fall in this class. In the last section we speculate on the possible causes of this fact. Observe that even if we can study an autosymmetric function f in a $n - k$ dimensional space, f depends, in general, on all the n input variables, i.e., f is non degenerated.

In the next section 2 we recall the basic definitions and results of SPP theory, and present a companion algebraic formulation later exploited for testing autosymmetry. In section 3 we discuss the properties of autosymmetric functions, and how the problem of determining their minimal SPP forms can be studied on a reduced number of variables. In section 4 we show how autosymmetry can be tested in polynomial time, and derive a new minimization algorithm that includes such a test in the initial phase. In section 5 we present an extensive set of experimental results which validate the proposed approach, also proving that the number of benchmarks practically tractable is significantly increased. A final discussion on the role of autosymmetry is developed in section 6.

2 The Underlying Theory

The basic definitions and properties that we now recall were stated in [8] and extended in [2]. We work in a Boolean space $\{0,1\}^n$ described by n variables $x_0, x_1, \ldots, x_{n-1}$, where each point is represented as a binary vector of n elements. A set of k points can be arranged in a $k \times n$ matrix whose rows correspond to the points and whose columns correspond to the variables. Figure 2 represents a set of eight points in a space of six variables. A Boolean function $f : \{0,1\}^n \to \{0,1\}$ can be specified with an algebraic expression where the variables are connected through Boolean operators, or as the set of points for which $f = 1$. $|f|$ denotes the number of such points.

From [8] we take the following. Let $\boldsymbol{u}$ be a (Boolean) vector; $\overline{\boldsymbol{u}}$ be the elementwise complementation of $\boldsymbol{u}$; and $\widehat{\boldsymbol{u}}$ denote $\boldsymbol{u}$ or $\overline{\boldsymbol{u}}$. The *constant* vectors $\mathbf{0}$ and $\mathbf{1}$ are made up of all 0's or all 1's, respectively. Vector $\boldsymbol{uv}$ is the concatenation of $\boldsymbol{u}$ and $\boldsymbol{v}$. A vector $\boldsymbol{u}$ of 2^m elements, $m \geq 0$, is *normal* if $m = 0$, or $m > 0$ and $\boldsymbol{u} = \boldsymbol{v}\widehat{\boldsymbol{v}}$ with $\boldsymbol{v}$ (hence $\widehat{\boldsymbol{v}}$) normal. For example all the columns in

	$\mathbf{c}_0$	$\mathbf{c}_1$	$\mathbf{c}_2$	$\mathbf{c}_3$	$\mathbf{c}_4$	$\mathbf{c}_5$
$\mathbf{r}_0$	0	1	0	0	0	1
$\mathbf{r}_1$	0	1	0	0	1	0
$\mathbf{r}_2$	0	1	1	1	0	1
$\mathbf{r}_3$	0	1	1	1	1	0
$\mathbf{r}_4$	1	1	0	1	0	0
$\mathbf{r}_5$	1	1	0	1	1	1
$\mathbf{r}_6$	1	1	1	0	0	0
$\mathbf{r}_7$	1	1	1	0	1	1

Figure 2: A canonical matrix representing a pseudocube P in $\{0,1\}^6$. The canonical columns are $\boldsymbol{c_0}$, $\boldsymbol{c_2}$, and $\boldsymbol{c_4}$.

the matrix of figure 2 are normal vectors.

A matrix M with 2^m rows is *normal* if all its rows are different and all its columns are normal. A normal matrix is *canonical* if its rows, interpreted as binary numbers, are arranged in increasing order (the matrix of figure 2 is canonical). A normal vector is *k-canonical*, $0 \leq k < m$, if is composed of an alternating sequence of groups of 2^k 0's and 2^k 1's. In figure 2, $\boldsymbol{c}_0$ is 2-canonical, $\boldsymbol{c}_2$ is 1-canonical, and $\boldsymbol{c}_4$ is 0-canonical. A canonical matrix M contains m columns $\boldsymbol{c}_{i_0}, \ldots, \boldsymbol{c}_{i_{m-1}}$ of increasing indices, called the *canonical columns* of M, such that $\boldsymbol{c}_{i_j}$ is $(m-j-1)$-canonical for $0 \leq j \leq m-1$. The other columns are the *noncanonical* ones. If M represents a set of points in a Boolean space (where the rows are the points and the column c_i corresponds to the variable x_i), canonical and noncanonical columns correspond to *canonical* and *noncanonical variables*, respectively. We have:

Definition 1 (From [8]) *A* pseudocube *of degree m is a set of 2^m points whose matrix is canonical up to a row permutation.*

The matrix of figure 2 represents a pseudocube of 2^3 points in $\{0,1\}^6$. Note that a subcube in $\{0,1\}^n$ is a special case of pseudocube where the noncanonical columns are constant.

The function with value 1 in the points of a pseudocube P 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 P. For the pseudocube P of figure 2 we have:

$$CEX(P) = x_1 \cdot (x_0 \oplus x_2 \oplus \overline{x}_3) \cdot (x_0 \oplus x_4 \oplus x_5). \tag{1}$$

Refer to [8] for the nontrivial rule for generating *CEX(P)*. We simply recall that each EXOR factor of the expression contains exactly one noncanonical variable directed or complemented, namely the one with greatest index (x_1, x_3, and x_5 in the example), and each noncanonical variable is contained only in one EXOR factor; all the other variables in the expression are canonical (x_0, x_2, and x_4 in the example); and some canonical variables may not appear in the expression. Note that the minimal SP form for the above function is $\overline{x}_0 x_1 \overline{x}_2 \overline{x}_3 \overline{x}_4 x_5 + \overline{x}_0 x_1 \overline{x}_2 \overline{x}_3 x_4 \overline{x}_5 + \overline{x}_0 x_1 x_2 x_3 \overline{x}_4 x_5 + \overline{x}_0 x_1 x_2 x_3 x_4 \overline{x}_5 + x_0 x_1 \overline{x}_2 x_3 \overline{x}_4 \overline{x}_5 + x_0 x_1 \overline{x}_2 x_3 x_4 x_5 + x_0 x_1 x_2 \overline{x}_3 \overline{x}_4 \overline{x}_5 + x_0 x_1 x_2 \overline{x}_3 x_4 x_5$, much larger than *CEX(P)*.

If P is in fact a subcube, each EXOR factor in *CEX(P)* reduces to a single noncanonical variable, the canonical variables do not appear, and the whole expression reduces to the well known *product* expression e.g. used for implicants in SP forms.

A general property of the algebraic representation of pseudocubes is given in the following:

Theorem 1 *In a Boolean space $\{0,1\}^n$:*
(i) the EXOR factor of any subset of variables (directed or complemented) represents a pseudocube of degree $n-1$;
(ii) the product of $k \leq n$ arbitrary EXOR factors represents either an empty set or a pseudocube of degree $\geq n-k$.

Point (i) of the theorem can be easily proved by induction on the number of variables in the EXOR factor. Point (ii) then follows from a theorem of [8] which states that the intersection of two pseudocubes of degrees p, r is either empty, or is a pseudocube of degree $\geq p + r - n$.

For the example of figure 2, the EXOR factors x_1, $x_0 \oplus x_2 \oplus \overline{x}_3$, and $x_0 \oplus x_4 \oplus x_5$ in the expression 1 represent three pseudocubes of 2^5 points, and their product represents the pseudocube P of 2^3 points.

Note now that an equality of the form $\text{EXOR}_1 \cdot \text{EXOR}_2 \cdot \ldots \cdot \text{EXOR}_k = 1$, satisfied by all the points of a pseudocube, can be equivalently written as a system of k linear equations:

$$EXOR_i = 1, \quad 1 \leq i \leq k \tag{2}$$

that is an instance of a general linear system $A\boldsymbol{x} = \mathbf{1}$ with $\boldsymbol{x} = \{x_0, \ldots, x_{n-1}\}$, A is a $k \times n$ matrix of coefficients 0, 1, and the sum is substituted with EXOR. As known [3, 1] such a system specifies an *affine subspace* of the linear space $\{0,1\}^n$. Then, from the existence of *CEX(P)* for any pseudocube P, and from theorem 1, we have:

Corollary 1 *In a Boolean space $\{0,1\}^n$ there is a one-to-one correspondence between affine subspaces and pseudocubes.*

This corollary allows to inherit all the properties of affine subspaces into pseudocube theory. In particular, a pseudocube containing the point (vector) $\mathbf{0}$ corresponds to a *linear* subspace. We also have:

Definition 2 (From [2]) *The* structure *of a pseudocube P, denoted by STR(P), is CEX(P) without complementations.*

For the pseudocube P of figure 2 we have: $STR(P) = x_1 \cdot (x_0 \oplus x_2 \oplus x_3) \cdot (x_0 \oplus x_4 \oplus x_5)$. Let us now extend the symbol $\oplus$ to denote the elementwise EXOR between two vectors. Then $\boldsymbol{\alpha} \oplus \boldsymbol{\beta}$ is the vector obtained from $\boldsymbol{\beta}$ complementing in it the elements corresponding to the 1's of $\boldsymbol{\alpha}$. For a vector $\boldsymbol{\alpha} \in \{0,1\}^n$ and a subset $S \subseteq \{0,1\}^n$, let $\boldsymbol{\alpha} \oplus S = \{ \boldsymbol{\alpha} \oplus \boldsymbol{s} \mid \boldsymbol{s} \in S\}$. We have:

Theorem 2 (From various results in [2]) *For any pseudocube $P \subseteq \{0,1\}^n$ and any vector $\boldsymbol{\alpha} \in \{0,1\}^n$, the subset $Q = \boldsymbol{\alpha} \oplus P$ is a pseudocube with STR(Q) = STR(P).*

Finally recall that an arbitrary function can be expressed as an OR of pseudoproducts, giving rise to an SPP form. For example if we add the two rows (points) $\mathbf{r}_8 = 1\,1\,0\,1\,0\,1$ and $\mathbf{r}_9 = 1\,1\,0\,1\,1\,0$ to the matrix of figure 2, we have a new function f that can be seen as the union of two partially overlapping pseudocubes: namely P (already studied), and Q associated to the rows $\mathbf{r}_4, \mathbf{r}_5, \mathbf{r}_8, \mathbf{r}_9$. Note that Q is in fact a cube, and we have $CEX(Q) = x_0 x_1 \overline{x}_2 x_3$. In conclusion f can be expressed in SPP form as:

$$CEX(P) + CEX(Q) = x_1(x_0 \oplus x_2 \oplus \overline{x}_3)(x_0 \oplus x_4 \oplus x_5) + x_0 x_1 \overline{x}_2 x_3. \tag{3}$$

The minimal SP form for f contains 40 literals, while the SPP expression (3) contains 11 literals. Passing from SP to SPP, however, amounts to passing from a two-level to a three-level circuit. This fact has always to be taken into account and will not be further repeated.

3 Autosymmetric Functions

The class of *autosymmetric functions* introduced in [8] seems to be particularly suitable for SPP minimization. The present work is addressed to these functions, for which we give an alternative definition.

Definition 3 *A Boolean function f in $\{0,1\}^n$ is* closed under $\boldsymbol{\alpha}$, *with $\boldsymbol{\alpha} \in \{0,1\}^n$, if for each $\boldsymbol{w} \in \{0,1\}^n$, $\boldsymbol{w} \oplus \boldsymbol{\alpha} \in f$ if and only if $\boldsymbol{w} \in f$.*

Each function is obviously closed under the zero vector $\mathbf{0}$. As proved in [8], if a function f is closed under two different vectors $\boldsymbol{\alpha}_1$, $\boldsymbol{\alpha}_2 \in \{0,1\}^n$, it is also closed under $\boldsymbol{\alpha}_1 \oplus \boldsymbol{\alpha}_2$. Therefore the set L_f of all the vectors $\boldsymbol{\beta}$ such that f is closed under $\boldsymbol{\beta}$ is a linear subspace of $\{0,1\}^n$, see for example [3]. (In fact, combining in EXOR, in all possible ways, k linearly independent vectors $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_k$, we form a subspace of 2^k vectors that is closed under $\oplus$, and contains the vector $\mathbf{0}$ generated as $\boldsymbol{\alpha}_i \oplus \boldsymbol{\alpha}_i$). L_f is called the *linear space* of f, and k is its *dimension.* By corollary 1, L_f is a pseudocube, and we will refer to $CEX(L_f)$ and $STR(L_f)$.

Definition 4 *A Boolean function f is k-*autosymmetric, *or equivalently f has* autosymmetry degree *k, $0 \leq k \leq n$, if its linear space L_f has dimension k.*

We have:

Theorem 3 *Let f be a k-autosymmetric function. There exist ℓ vectors $\boldsymbol{w}^1$, $\boldsymbol{w}^2, \ldots, \boldsymbol{w}^\ell \in f$, with $\ell = |f|/2^k$, such that*

$$f = \bigcup_{i=1}^{\ell} (\boldsymbol{w}^i \oplus L_f) \tag{4}$$

and for each i, j, $i \neq j$, $(\boldsymbol{w}^i \oplus L_f) \cap (\boldsymbol{w}^j \oplus L_f) = \emptyset$.

Proof. Let $\boldsymbol{w}^1$ be any vector in f. By definition 3, $\boldsymbol{w}^1 \oplus L_f \subseteq f$. Consider the set $f^1 = f \setminus (\boldsymbol{w}^1 \oplus L_f)$. If $f^1 = \emptyset$, then $f = \boldsymbol{w}^1 \oplus L_f$ and f is a pseudocube of degree k with $|f| = 2^k$. Otherwise, let $\boldsymbol{w}^2$ be any vector of f^1. Again by definition 3, $\boldsymbol{w}^2 \oplus L_f \subseteq f$. Observe that $(\boldsymbol{w}^1 \oplus L_f) \cap (\boldsymbol{w}^2 \oplus L_f) = \emptyset$ (by contradiction: let $\boldsymbol{v} \in (\boldsymbol{w}^1 \oplus L_f) \cap (\boldsymbol{w}^2 \oplus L_f)$, then $\boldsymbol{v} = \boldsymbol{w}^1 \oplus \boldsymbol{\alpha} = \boldsymbol{w}^2 \oplus \boldsymbol{\beta}$, with $\boldsymbol{\alpha}, \boldsymbol{\beta} \in L_f$; then $\boldsymbol{w}^2 = \boldsymbol{w}^1 \oplus \boldsymbol{\alpha} \oplus \boldsymbol{\beta}$, that is $\boldsymbol{w}^2 \in (\boldsymbol{w}^1 \oplus L_f)$ which is a contradiction). Therefore we have:

$$f = (\boldsymbol{w}^1 \oplus L_f) \cup (\boldsymbol{w}^2 \oplus L_f) \cup f \setminus ((\boldsymbol{w}^1 \oplus L_f) \cup (\boldsymbol{w}^2 \oplus L_f)),$$

and using the same argument on the set $f \setminus ((\boldsymbol{w}^1 \oplus L_f) \cup (\boldsymbol{w}^2 \oplus L_f))$ the theorem easily follows. □

Form the proof above we see that the number of points of a k-autosymmetric function is a multiple of 2^k. Indeed, each affine subspace $\boldsymbol{w}^i \oplus L_f$ contains 2^k points. Recalling that L_f is a pseudocube, and from theorems 2 and 3 we immediately have:

Corollary 2 *A k-autosymmetric function f is a disjoint union of $|f|/2^k$ pseudocubes $\boldsymbol{w}^i \oplus L_f$ of degree k all having the same structure $STR(L_f)$, and the same canonical variables of L_f.*

This corollary has two immediate consequences. First we can extend the definition of canonical variables to autosymmetric functions. Namely, the canonical (respectively: noncanonical) variables of L_f are designated as the canonical (respectively: noncanonical) variables of f. Second, note that the 2^k points of each pseudocube $\boldsymbol{w}^i \oplus L_f$ contain all the 2^k combinations of values of the canonical variables, hence in exactly one of these points all such values are 0. We then have:

Corollary 3 *The vectors $\boldsymbol{w}^1, \ldots, \boldsymbol{w}^\ell$ of theorem 3 can be chosen as the points of f where all the canonical variables have value 0. Moreover, for $1 \leq i \leq \ell$, $CEX(\boldsymbol{w}^i \oplus L_f)$ is obtained from $STR(L_f)$ complementing the noncanonical variables with value 0 in $\boldsymbol{w}^i$.*

Example 1 *Consider the function* $f = \{00001, 00100, 00110, 01000, 01010, 01101, 10001, 10011, 10100, 11000, 11101, 11111\}$. *It can be easily verified that the linear space of* f *is:*

$$L_f = \{00000, 01100, 10101, 11001\},$$

where each vector can be obtained as the EXOR of the other two. f *is then 2-autosymmetric. We have:*

$$STR(L_f) = (x_0 \oplus x_1 \oplus x_2)x_3(x_0 \oplus x_4)\,. \tag{5}$$

where x_0, x_1, *and,* x_2, x_3, x_4, *are the canonical and noncanonical variables, respectively. From corollary 3 we have* $\boldsymbol{w}^1 = 00001$, $\boldsymbol{w}^2 = 00100$, $\boldsymbol{w}^3 = 00110$, *hence:*

$$f = (00001 \oplus L_f) \cup (00100 \oplus L_f) \cup (00110 \oplus L_f)\,. \tag{6}$$

From this we derive the following (nonminimal) SPP form for f, *where each term* $\boldsymbol{w}^i \oplus L_f$ *of expression (6) is transformed into an EXOR factor derived from (5) complementing the noncanonical variables with value 0 in* $\boldsymbol{w}^i$:

$$\begin{array}{l}(x_0 \oplus x_1 \oplus \overline{x}_2)\overline{x}_3(x_0 \oplus x_4) + (x_0 \oplus x_1 \oplus x_2)\overline{x}_3(x_0 \oplus \overline{x}_4)\\ +(x_0 \oplus x_1 \oplus x_2)x_3(x_0 \oplus \overline{x}_4)\,.\end{array} \tag{7}$$

From the above properties of autosymmetric functions we observe:

- Any function is at least 0-autosymmetric, since is closed under $\mathbf{0}$.
- A function is $(n-1)$-autosymmetric if and only if is a pseudocube of degree $n-1$.
- A function is n-autosymmetric if and only if is a constant.
- Pseudocubes of degree k are the only k-autosymmetric functions with only one term in the union of expression (4).

We will study any k-autosymmetric function f with $k \geq 1$ through a simpler function f_k. We state:

Definition 5 *For a* $(k \geq 1)$*-autosymmetric function* f, *the* restriction f_k *consists of the* $|f|/2^k$ *points of* f *contained in the subspace* $\{0,1\}^{n-k}$ *where all the canonical variables of* f *have value 0.*

Note that f_k depends only on the $n-k$ noncanonical variables of f.

Once L_f has been computed (see next section), the canonical variables of f are known, and f_k can be immediately determined applying definition 5. For instance, for the function f with $k = 2$ of example 1, f_2 depends on the noncanonical variables x_2, x_3, x_4. To build f_2 we take the subset $\{00001, 00100, 00110\}$ of the points of f for which the canonical variables x_0, x_1 have value 0, and then project these points into the $\{0,1\}^3$ subspace relative to x_2, x_3, x_4, where we have $f_2 = \{001, 100, 110\}$. We can then represent f_2 as:

$$\overline{x}_2\overline{x}_3x_4 + x_2\overline{x}_3\overline{x}_4 + x_2x_3\overline{x}_4\,.$$

Note that the same expression is obtained setting $x_0 = x_1 = 0$ in expression (7).

We now show that the SPP-minimal form of any autosymmetric function can be easily derived from the SPP-minimal form of its restriction. Note that finding the latter form is easier because the restriction has less variables and points. In the example above f depends on 5 variables and has 12 points, while f_2 depends only on 3 variables and has only 3 points (represented by the minterms $\overline{x}_2\overline{x}_3x_4$, $x_2\overline{x}_3\overline{x}_4$ and $x_2x_3\overline{x}_4$). The following lemma is an easy extension of Theorem 6 of [8].

Lemma 1 *A* $(k \geq 1)$*-autosymmetric function and its restriction have the same number of pseudoproducts in their minimal SPP forms.*

In fact we can prove a stronger property, namely not only f_k and f have the same number of pseudoproducts in their minimal forms, but a minimal form for f can be easily derived from a minimal form for f_k. Let $x_{z_0}, \ldots, x_{z_{n-k-1}}$ be the noncanonical variables of f, and let $STR(L_f) = p_0 p_1 \cdots p_{n-k-1}$, where p_i is the EXOR factor containing x_{z_i}, $1 \leq i \leq n-k-1$ (recall that each noncanonical variable is in an unique EXOR factor). We have:

Theorem 4 *If $SPP(f_k)$ is a minimal SPP form for f_k, then the form $SPP(f)$ obtained by substituting in $SPP(f_k)$ each variable x_{z_i} with the EXOR factor p_i is a minimal SPP form for f.*[1]

Proof. By lemma 1, the number of pseudoproducts in $SPP(f)$ is minimum, then we have only to prove that this form covers exactly all the points of f.

When we transform f into f_k, we select a the vector $\boldsymbol{u}^i$ with all canonical variables set to zero from each affine subspace $\boldsymbol{w}^i \oplus L_f$. Call $\boldsymbol{v}^i$ the vector $\boldsymbol{u}^i$ without the canonical variables, i.e., its projection onto a subspace $\{0,1\}^{n-k}$. When we apply the linear substitutions $x_{z_0} \leftarrow p_0, \ldots, x_{z_{n-k-1}} \leftarrow p_{n-k-1}$, we force any pseudoproduct that covers $\boldsymbol{v}^i$ in f_k to cover all the points in $\boldsymbol{w}^i \oplus L_f$ in f, and the thesis immediately follows. □

4 The Minimization Algorithm

In the previous section we have shown that each Boolean function f is k-autosymmetric, for $k \geq 0$. If $k \geq 1$, f will be simply called *autosymmetric*. For minimization purposes we have an increasing advantage for increasing $k \geq 1$, as minimizing a k-autosymmetric function with n variables and ℓ points reduces to minimizing a different function with $n-k$ variables and $\ell/2^k$ points. Even for $k = 1$ we have to cover only one half of the original points.

Fortunately, for a given function f, finding the associated linear space L_f and computing the autosymmetry degree k is an easy task, because the required algorithm is polynomial in the number n of variables and in the number of points of f. By definition 3, a function f is closed under $\boldsymbol{\alpha}$ if for any $\boldsymbol{u} \in f$ there exists $\boldsymbol{v} \in f$ such that $\boldsymbol{v} = \boldsymbol{u} \oplus \boldsymbol{\alpha}$. Thus, for all $\boldsymbol{u} \in f$, there exists a vector $\boldsymbol{v} \in f$ such that we can express $\boldsymbol{\alpha}$ as $\boldsymbol{\alpha} = \boldsymbol{u} \oplus \boldsymbol{v}$. In other words, $\boldsymbol{\alpha}$ must be searched within the vectors of the set $\{\boldsymbol{u} \oplus \boldsymbol{v} \mid \boldsymbol{u}, \boldsymbol{v} \in f\}$. More precisely we have:

Theorem 5 *Let f be a Boolean function. Then $L_f = \bigcap_{\boldsymbol{u} \in f} (\boldsymbol{u} \oplus f)$.*

Proof. Let $\boldsymbol{\alpha} \in \bigcap_{\boldsymbol{u} \in f} (\boldsymbol{u} \oplus f)$. Then

$$\begin{aligned} \boldsymbol{\alpha} \in \bigcap_{\boldsymbol{u} \in f} (\boldsymbol{u} \oplus f) &\Leftrightarrow \forall \boldsymbol{u} \in f, \quad \boldsymbol{\alpha} \in \boldsymbol{u} \oplus f \\ &\Leftrightarrow \forall \boldsymbol{u} \in f, \quad \exists \boldsymbol{v} \in f \mid \boldsymbol{\alpha} = \boldsymbol{u} \oplus \boldsymbol{v} \\ &\Leftrightarrow \boldsymbol{\alpha} \in L_f \,. \end{aligned}$$

□

Based on theorem 5 we state:

Algorithm 1 Construction of $\boldsymbol{L_f}$ (build L_f and find the autosymmetry degree k of a given function f)

1. for all $\boldsymbol{u} \in f$ build the set $\boldsymbol{u} \oplus f$;
2. build the set $L_f = \bigcap_{\boldsymbol{u} \in f} (\boldsymbol{u} \oplus f)$;
3. compute $k = \log |L_f|$.

[1] The resulting expressions may be reduced using some properties of EXOR, in particular $x \oplus x = 0$ and $0 \oplus x = x$.

The time complexity of algorithm 1 is $\Theta(|f|^2 n)$, because we must build a set $\boldsymbol{u} \oplus f$ for all $\boldsymbol{u} \in f$, and the construction of each such a set requires $\Theta(|f|\, n)$ time.

Any SPP minimization algorithm can be easily extended for exploiting autosymmetry. For a given function f we first compute L_f and k with algorithm 1. If $k = 0$ (i.e., f is not autosymmetric) we proceed with regular minimization, otherwise we compute the restriction f_k of f, minimize it, and finally derive a minimal form $SPP(f)$ from $SPP(f_k)$. We propose the following A (for autosymmetry) minimization algorithm.

Algorithm 2 A-Minimization (build a minimal SPP form of a given function f)

1. build L_f and compute the value of k by algorithm 1;

2. **if** $k = 0$ **then** minimize f with any SPP synthesis algorithm

3. **else**

  (a) determine the canonical variables of L_f and compute the restriction f_k as indicated in section 3;
  (b) compute $STR(L_f) = p_0 p_1 \cdots p_{n-k-1}$;
  (c) compute the minimal form $SPP(f_k)$ for f_k with any SPP synthesis algorithm;
  (d) build $SPP(f)$ by substituting in $SPP(f_k)$ each noncanonical variable x_{z_i} with the EXOR factor p_i of $STR(L_f)$.

By the theory developed in the previous section, algorithm 2 is correct. Note that the algorithm builds an SPP form minimal with respect to the number of pseudoproducts. To obtain the minimal SPP form with respect to the number of literals we must slightly rearrange steps *3(c)* and *3(d)*, executing the substitutions of all p_i for x_{z_i} in the prime pseudoproducts of f_k, before selecting such pseudoproducts in the set covering problem implicit in the minimization algorithm.

Example 2 *Minimization of the function f of example 1, using algorithm 2.*

- *Derive L_f and k by algorithm 1. For this purpose, for all $\boldsymbol{u} \in f$ compute the set $\boldsymbol{u} \oplus f$. (For example, for the point* 00100 *we obtain the set:* $00100 \oplus f = \{00101,\ 00000,\ 00010,\ 01100,\ 01110,\ 01001,\ 10101,\ 10111,\ 10000,\ 11100,\ 11001,\ 11011\}$*). The intersection of all the sets $\boldsymbol{u} \oplus f$ gives the linear space* $L_f = \{00000, 01100, 10101, 11001\}$. *We then have* $k = 2$.

- *Since* $k = 2$ *we proceed with the* **else** *branch of algorithm 2. L_f has noncanonical variables x_2, x_3, x_4, hence f_2 is restricted to these variables and we have:* $f_2 = \{001, 100, 110\}$.

- *The minimization problem now consists of finding a minimal SPP cover of the points of f_2. Applying the algorithm of [2] we have the minimal form:*

$$SPP(f_2) = x_2\overline{x}_4 + \overline{x}_3(x_2 \oplus x_4)\,.$$

- *Compute:* $STR(L_f) = (x_0 \oplus x_1 \oplus x_2)x_3(x_0 \oplus x_4)$.

- *Derive the minimal SPP form for f by substituting x_2, x_3 and x_4 in $SPP(f_2)$ with the EXOR factors of $STR(L_f)$, respectively $(x_0 \oplus x_1 \oplus x_2)$, x_3 and $(x_0 \oplus x_4)$. With some immediate algebraic simplifications we obtain:* [2]

$$SPP(f) = (x_0 \oplus x_1 \oplus x_2)(x_0 \oplus \overline{x}_4) + \overline{x}_3(x_1 \oplus x_2 \oplus x_4)\,.$$

[2]E.g., in the first term of $SPP(f_k)$ we have: $\overline{x}_4[x_4 \leftarrow (x_0 \oplus x_4)] = \overline{(x_0 \oplus x_4)} = (x_0 \oplus \overline{x}_4)$. In the last term of $SPP(f_k)$ we have: $(x_2 \oplus x_4)[x_2 \leftarrow (x_0 \oplus x_1 \oplus x_2), x_4 \leftarrow (x_0 \oplus x_4)] = (x_0 \oplus x_1 \oplus x_2) \oplus (x_0 \oplus x_4) = (x_0 \oplus x_0 \oplus x_1 \oplus x_2 \oplus x_4) = (x_1 \oplus x_2 \oplus x_4)$.

k=	0	1	2	3	4	5	6	7	8	9	10	11	12	Tot
#funct	365	116	72	95	41	43	39	27	16	24	58	28	18	942
%funct	38,9	12,3	7,6	10,1	4,3	4,6	4,1	2,9	1,7	2,4	6,2	3,0	1,9	100
#AC	264	82	55	54	23	35	21	10	2	0	0	0	0	546
#A	0	5	4	5	9	7	16	9	14	24	58	28	18	197
#*	101	29	13	36	9	1	2	8	0	0	0	0	0	199

Table 1: Distribution of k-autosymmetric functions in the ESPRESSO benchmark suite. **#funct** are the total numbers of functions (single outputs) for any value of k, and **%funct** are the corresponding percentages. The other rows report the number of k-autosymmetric functions synthesized with the new algorithm A, and with the previous best algorithm C (**#AC**); synthesized with algorithm A only (**#A**), since algorithm C did not terminate; not synthesized at all since both algorithms did not terminate (**#***).

k=	0	1	2	3	4	5	6	7	8
$\mathbf{Avg(T_A/T_C)}$	1,01	0,63	0,42	0,37	0,32	0,27	0,07	0,004	0,0001

Table 2: Time gain using algorithm A versus algorithm C, for the 546 functions of row **#AC** of table 1 ($k = 0$ to 8). $\mathbf{T_A}$ and $\mathbf{T_C}$ are the times required by the two algorithms on the same function. $\mathbf{Avg(T_A/T_C)}$ is the average of the ratio $\mathbf{T_A/T_C}$ over all the functions with the same value of k.

5 Experimental Results

The new minimization algorithm 2, also called algorithm A, has been tested on a large set of functions without don't cares, taken from the ESPRESSO benchmark suite [11] (the different outputs of each function have been synthesized separately). The performance of algorithm A has been compared with the performance of the best previous algorithm, that is the one proposed in [2], in the following indicated as algorithm C (after Ciriani). In fact, also the minimization of function f_k in algorithm A (step 3(c)) has been implemented with algorithm C.

For all the functions considered we have computed the values of the autosymmetry degree k with algorithm 1, obtaining the results shown in the first two rows of table 1. Surprisingly the overall percentage of autosymmetric functions ($k \geq 1$) is over the 61%. We have then attempted to run algorithms A and C for all such functions, recording the CPU times whenever the computation terminated in less than 172800 seconds (2 days) on a Pentium III 450 machine. Results on program termination are given in the last three rows of table 1.

Table 2 shows the average reduction of computing time using algorithm A instead of C, for all the benchmark functions for which both algorithms terminated (i.e., for the 546 functions of row **#AC** of table 1). Note the improvement introduced by the new algorithm for all the autosymmetric functions in the set, and how such an improvement drastically increases for increasing k. For $k = 0$, instead, we have $Avg(T_A/T_C) > 1$, and actually the ratio (T_A/T_C) is slightly greater than 1 for each such a function. This is because algorithm A computes L_f in any case, then calls algorithm C. The resulting slowdown is however always negligible because L_f is computed in polynomial time by algorithm 1 (see previous section), while algorithm C is exponential in nature. For all the functions in the table the forms obtained with the old and the new method coincide.

Finally table 3 shows the CPU times for a small subset of the above functions with $k \geq 1$, and other relevant minimization parameters for them.

6 The Role of Autosymmetry

To understand the role of autosymmetric functions we must compare them with the set of all possible functions. The total number of Boolean functions of n variables is $N_F = 2^{2^n}$, corresponding to all the ways a subset of points can be chosen in $\{0,1\}^n$. This is a huge number, however, due to the

function	k	#L	#PP	$\mathbf{T_A}$	$\mathbf{T_C}$
max512(0)	1	8	2	20,82	2618,02
prom2(0)	1	25	6	4,12	41,14
Z9sym(0)	1	102	17	310,17	*
log8mod(0)	2	15	5	0,77	19,17
newtpla2(2)	2	17	4	1,71	236,89
gary(1)	3	76	8	13,24	*
radd(2)	3	16	3	0,22	5,43
addm4(6)	4	18	4	0,39	79,86
intb(0)	5	96	12	11420,80	*
radd(1)	5	7	2	0,11	7,41
addm4(7)	6	6	2	0,27	123,63
adr4(4)	7	2	1	0,06	25,54
intb(5)	7	38	8	431,60	*
newtpla(4)	8	7	1	0,48	2893,04
mp2d(13)	9	5	5	149,29	*
opa(17)	10	37	7	503,01	*
alcom(5)	11	6	3	119,67	*

Table 3: Detailed results for a subset of autosymmetric functions. The last two columns report CPU times in seconds on a Pentium III 450 machine for algorithms A and C (a star indicates non termination after 172800 seconds). The results are relative to single outputs, e.g. max512(0) corresponds to the first output of max512. **#L** and **#PP** are the numbers of literals and of prime pseudoproducts in the minimal expression.

randomness of the above generating process very many of such functions do not correspond to any significant circuit. Autosymmetric functions are just a subset of the above. By a counting argument (omitted here for brevity) we can in fact prove that autosymmetric functions are $N_A = (2^n - 1)2^{2^{n-1}}$. Therefore, for increasing n, autosymmetric functions constitute a vanishing fractions of all the functions, as N_A/N_F goes to zero for n going to infinity. Still the question remains on how many significant functions are autosymmetric.

All what we can say at the moment is that most of the major benchmark functions are indeed autosymmetric, as shown in the previous section. The more so when n is small and the values of N_F and N_A are not too distant. The reason, we might argue, is that a function encoding a real life problem must exhibit a regular structure that can be reflected in some degree of autosymmetry. This regularity may also allow to define an autosymmetric function f independently on the number of variables, and then to state a rule for deriving a minimal form for f valid for any n. Well known functions as, for example, the ones counting the parity of n bits, or giving the next-state values for an n bit Gray code, can be easily expressed in minimal form for an arbitrary number of variables just because they are autosymmetric (for the parity see [8]; for Gray codes elementary considerations suffice).

It might be relevant to examine the relation between autosymmetric functions, and functions which are simply *symmetric*. That is, functions which are invariant under any permutation of their variables (see for example [7]). The total number of symmetric functions is $N_S = 2^{n+1}$, hence $N_S << N_A$, but they are not a subset of the autosymmetric ones. In fact a symmetric function may be autosymmetric (e.g., the parity function), but there are symmetric functions that are not autosymmetric (e.g., any symmetric function with an odd number of points). At the moment we have no interesting results in this direction.

We observe that the *information content* of an autosymmetric function f is represented by its reduction f_k together with the linear transformation $x_{z_0} \leftarrow p_0, \ldots, x_{z_{n-k-1}} \leftarrow p_{n-k-1}$ (section 3), so that the core of the synthesis problem is the minimization of f_k (section 4). This suggests a formal generalization. For a given function $g \in \{0,1\}^m \to \{0,1\}$ we can define the *autosymmetry* *class* $\mathcal{C}_g$ as the class of all the autosymmetric functions $f \in \{0,1\}^t \to \{0,1\}, t \geq m$, such that $f_k \equiv g$. Since the information content of any given function f can be easily found, and a minimal SPP form for f can then be derived from $SPP(f_k)$, minimizing the function f_k corresponds to minimize the entire class $\mathcal{C}_g = \mathcal{C}_{f_k}$. Exploiting the full potential of such an approach is currently a matter of study.

Finally another interesting research direction is the generalization of the autosymmetry property to functions with don't care set. Since the autosymmetry is strictly related to the points of a function f, the major goal would be selecting a subset of don't cares as points of f to maximize the autosymmetry degree of the function. However this seems not to be an easy task.

References

[1] A. Aleksanyan. Realization of Boolean Functions by Disjunctions of Products of Linear Forms. *Dokl. Akad. Nauk SSSR (Russian), (translation in Soviet Math. Dokl. 39(1), 1989)*, 304(4):781–784, 1989.

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

[3] P. Cohn. *Algebra Vol. 1.* John Wiley & Sons, 1997.

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

[5] D. Debnath and T. Sasao. An Optimization of AND-OR-EXOR Three-Level Networks. In *Asia and South Pacific Design Automation Conference*, pages 545–550, 1997.

[6] 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.

[7] Z. Kohavi. *Switching and Finite Automata Theory.* McGraw Hill, 1970.

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

[9] 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.

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

[11] 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.