Canonical restricted mixed-polarity exclusive-OR sums of products and the efficient algorithm for their minimisation

L. Csanky
M.A. Perkowski
I. Schäfer

Indexing terms: Boolean algebra, Logic, Reed–Muller form, Minimisation

Abstract: The concept of canonical restricted mixed polarity (CRMP) exclusive-OR sum of products forms is introduced. The CRMP forms include the inconsistent canonical Reed–Muller forms and the fixed-polarity Reed–Muller (FPRM) forms as special cases. The set of CRMP forms is included in the set of exclusive-OR sum-of-product (ESOP) expressions. An attempt to characterise minimal CRMP forms for completely specified Boolean functions is presented as well as an insight into the complexity of computation needed to find such a form. Some fundamental properties unique to CRMPs are proven. It is also proven that the upper bound on the number of terms in the CRMP form is smaller than that in the conventional normal forms and equal to that of the ESOPs. A theorem providing a lower bound on the number of CRMP terms is given. Finally, based on these theoretical results, a heuristic algorithm and its implementation to obtain a quasiminimal CRMP form for a multioutput function are presented.

1 Introduction

It has long been the experience of logic designers that exclusive-OR sum-of-product (ESOP) expressions, AND–EXOR, are more economical than the conventional inclusive sum-of-product (SOP), AND–OR, expressions. This was confirmed practically on many examples, especially on arithmetic and telecommunication circuits, [1, 2, 3], and theoretically [4]. New results are also reported in References 5, 6, 7. Reference 8 introduces several AND–EXOR forms being particular cases of ESOPs and presents experimental analysis of their complexity. Although the CRMP forms are mentioned, their complexity is not discussed.

The EXOR gate exists in most VLSI cell libraries but there are hardly any logic synthesis systems able to find optimised netlists that include EXOR gates. Several families of new PLD devices able to realise EXOR gates have been recently marketed: table look-up based field programmable gate arrays (Xilinx LCA 3000) [9], folded NAND (Signetics LHS501) [10], multiplexer-based devices (Actel 1020) [11] and particularly the fine-grain (cellular) field programmable gate arrays from Algotronix [12] and Concurrent Logic (CLI 6000 series) [13]. Such devices either directly include EXOR gates (LHS501, CLI 6000 series) or allow their realisation in 'universal modules'. Since the five-input EXOR gate has the same speed and cost as, for instance, a five-input OR gate [14], logic synthesis algorithms that will take EXOR gates into account on equal terms to AND and OR gates become necessary. Particularly for a design implemented in Xilinx devices, if a fixed polarity Reed–Muller (FPRM) form has less terms than a two-level AND–OR expression, there is no reason why it should not be taken as a base of factorisation [15] since it will very probably produce a circuit with smaller number of blocks and connections. Additionally, it will be more easily [16–20]. In the case of CLi 6000 series, the usage of EXOR gates is a 'must', since methods based on sum of products give results that are very far from the minimum. The fact that the synthesis tools for technologies that include EXOR gates do not use algorithms for FPRMs, ESOPs or other EXOR-based expressions is an anachronism caused by inertia, and the fact that high quality algorithms of speed comparable to Espresso [21] are only recently becoming available.

The problem of finding the minimal exclusive-OR sum of products (ESOP) of a Boolean function is classical in logic synthesis theory, but exact approaches to solve this problem have been proposed for only very small functions [22, 23].

In this paper a canonical form, called canonical restricted mixed polarity exclusive-OR sum of product (CRMP), being a particular case of an ESOP, is discussed. (This form was mentioned in References 8 and 24 but not studied much). The family of fixed-polarity Reed–Muller forms [2, 8, 24, 25, 26] and the family of inconsistent canonic forms introduced in References 27–29 constitute the family of CRMP forms (also called generalised Reed–Muller forms in Reference 24). A CRMP expression is always not more expensive than fixed polarity expressions for the same function. Both these forms have very good testability and universal tests [30].

There are two reasons why we study the CRMP forms. First, they have an interest on their own, and it is interesting to compare them with other AND–EXOR forms discussed by several authors [8, 31]. Secondly, and more importantly, we used the concept of CRMP forms to develop the algorithm for ESOP expressions that, con-

Paper 9218E (C1), first received 6th August 1991 and in revised form 28th August 1992

I. Schäfer and M.A. Perkowski are with the Department of Electrical Engineering and L. Csanky is with the Department of Computer Science all at the Portland State University, PO Box 751, Portland, Oregon, 97207, USA

trary to all exact algorithms known from the literature [22, 23], is not exhaustive. In brief, this is done as follows. We introduce the concept of a sparse function. A sparse function, represented as an ESOP, is an EXOR of clusters (each cluster is an ESOP) of product terms such that if terms t_1 and t_2 are in two different clusters, there is no EXOR-type operation such as xlink or unlink [32] or exorlink [33] that would replace these two terms with one or two other terms. Now there are two possibilities. Either the original ESOP cannot be decomposed to such clusters and a single CRMP is found for it, or it is decomposed to clusters, for each cluster the CRMP is found, and the EXOR of all those CRMPs is returned as the minimal ESOP. We do not know whether this algorithm always finds the exact solution, although in many cases the algorithm can prove the minimality of the solution. For many Boolean functions of not more than five variables that we checked, the ESOP expressions found using this algorithm were actually the exact ESOP expressions, which was confirmed using the exact ESOP minimiser from Reference 23. Using the last one was, however, much more time consuming. Unfortunately, the program from Reference 23 can be used for completely specified functions of not more than five variables so we cannot verify the exactness of larger solutions. Also, one can construct functions for which this new method still requires much search and is slow.

The goal of the research reported here has been to create an exact synthesis program for CRMPs that would allow one to find ESOPs whose quality would be better than that of the previous heuristic minimisers [3], and which would be much faster than the exact minimisers [22, 23], thus allowing further study of exact and quasi-minimum ESOP minimisers. Most of the theoretical results of their proofs given here are new.

2 Canonical restricted mixed polarity forms

First we compare the CRMP form to the FPRM and RM forms.

Definition 2.1: The literal x_i^c is a variable x_i in either positive ($x_i = x_i^1$) or complemented ($\overline{x_i} = x_i^0$) form.

Consider the following form:

$$f(x_1, \ldots, x_n) = g_0 \oplus g_1 x_1^c \oplus \cdots \oplus g_n x_n^c \oplus g_{n+1} x_1^c x_2^c \oplus \cdots \oplus g_{2^n-1} x_1^c x_2^c \cdots x_n^c \quad (1)$$

where $g_i = 0$ or 1, and $x_i^c = x_i$ or x_i.

Definition 2.2: By a fixed-polarity Reed–Muller (FPRM) form one understands a form of eqn. 1 in which a variable is complemented (negative) or not complemented (positive), but cannot stand in both forms [2, 24, 29].

Definition 2.3: The form of eqn. 1 in which each variable is both complemented and not complemented but in which there is exactly one coefficient for each subset of variables of a term will be called a canonical restricted mixed polarity form (CRMP).

It follows that the CRMP form is either a FPRM or an inconsistent canonical form [28]. The CRMP form can be represented as binary vector of length 2^n whose coefficients g_i are calculated respectively to selected polarities of their corresponding subsets of variables in products. Observe that a CRMP expression corresponds to the CRMP form. This expression is an exclusive sum of products in which there exists no more than one term (product) for each possible subset of input variables. Denote by g_t the value of the coefficient that stands near product term t in a CRMP. Assuming a function of two variables a and b, for example, expression a corresponds to forms $[g_1, g_a, g_b, g_{ab}] = [0, 1, 0, 0] = 0 \cdot 1 \oplus 1 \cdot a \oplus 0 \cdot b \oplus 0 \cdot ab$ and $[g_1, g_a, g_{\bar{b}}, g_{a\bar{b}}] = [0, 1, 0, 0]$, while expression $\bar{a}$ corresponds to forms $[g_1, g_{\bar{a}}, g_b, g_{\bar{a}\bar{b}}]$and $[g_1, g_{\bar{a}}, g_{\bar{b}}, g_{\bar{a}b}] = [1, 1, 0, 0]$. Observe that there are as many different CRMP forms as vectors of length 2^2 of coefficients g_t, but many of those forms correspond to the same binary vectors.

It follows from the preceding definitions that the FPRM class is properly included in the CRMP class, and that the CRMP class is properly included in the class of ESOP expressions. Also observe that the CRMPs are canonical forms, while it is easy to show that the ESOPs are not. This property of CRMPs is useful in many respects, including the design for testability.

Definition 2.4: A product of distinct literals is called a term.

Definition 2.5: A Reed–Muller form (RM) [2] is a CRMP form that contains only uncomplemented variables.

A theorem describing the properties of operator $\oplus$ follows without proof, since they are well known [2, 24].

Theorem 2.1: The following identities are true:

$$x_1 \oplus x_2 = x_1\overline{x_2} + \overline{x_1}x_2$$
$$x_1 \oplus x_1 = 0$$
$$x_1 \oplus \overline{x_1} = 1$$
$$x_1 \oplus 0 = x_1$$
$$x_1 \oplus 1 = \overline{x_1}$$
$$x_1 \oplus (x_2 \oplus x_3) = (x_1 \oplus x_2) \oplus x_3$$
$$x_1(x_2 \oplus x_3) = x_1x_2 \oplus x_1x_3$$

Applying the principle of duality to the CRMP form and the identities from theorem 2.1, one obtains the dual CRMP form and dual identities: the system ($\oplus$, $\cdot$) is replaced with the dual system ($\odot$, $+$) (see also Reference 25). All the results of this paper, after applying the principle of duality, hold in the dual system as well. Circuits generated for systems can be implemented using EXOR and NOR gates or EXOR and NAND gates.

The following example illustrates the concepts introduced.

Example 2.1: $1 \oplus x_1 \oplus \overline{x_2} \oplus \bar{x}_1 x_2$ is a CRMP form because there exists only one term for each subset of variables. It is not an FPRM form because there exists a variable, say x_1, that stands both in negated and positive form. $x_1\overline{x_2} \oplus \overline{x_1}x_2 \oplus x_2 \oplus x_1x_2$ is an ESOP form that is not a CRMP form because it includes more than one product with a set of variables $\{x_1, x_2\}$. $\overline{x_2} \oplus \overline{x_1}\,\overline{x_2}$ is an FPRM form since both the variables x_1 and x_2 have constant polarities in the entire form (FPRM of polarity (0, 0)). It is called a negative Reed–Muller form. The FPRM form of polarity (0, 1) is $1 \oplus \overline{x_1} \oplus x_2 \oplus \overline{x_1}x_2$.

It can be easily verified by substitutions $x = 1 \oplus \bar{x}$ and $\bar{x} = 1 \oplus x$ that all these forms are equivalent to a FPRM form of polarity (1, 0) being a single term $x_1\overline{x_2}$, as well as to the RM form $x_1 \oplus x_1x_2$ (which is a FPRM of polarity (1, 1)). This example shows also that a selection of a proper form can minimise the number of terms.

The next example shows that the minimum CRMP form is not necessarily the minimum ESOP solution to a Boolean function.

Example 2.2: Given is a function of three variables $f(x_1, x_2, x_3) = \overline{x_1}\,\overline{x_2}\,\overline{x_3} + x_1x_2x_3$. Since the terms of f are disjoint, $f(x_1, x_2, x_3) = \overline{x_1}\,\overline{x_2}\,\overline{x_3} \oplus x_1x_2x_3$. It can be shown by an exhaustive search that this is the minimal ESOP for function f. The minimal CRMP is: $f(x_1, x_2, x_3) = \overline{x_1}\,\overline{x_3} \oplus x_2\overline{x_3} \oplus x_1x_2$. It has three terms and is not a termwise nor literalwise minimal ESOP solution. We are interested in minimal CRMP solutions. This means that our solutions will be not necessarily the minimum ESOP solutions. This is a kind of function, called sparse function, for which the costs of the exact FPRM, the exact CRMP, and the exact ESOP differ the most. It can be separated to two clusters, $\overline{x_1}\,\overline{x_2}\,\overline{x_3}$ and $x_1x_2x_3$, and for each of them a separate CRMP is found, thus producing the exact ESOP $f(x_1, x_2, x_3) = \overline{x_1}\,\overline{x_2}\,\overline{x_3} \oplus x_1x_2x_3$.

Theorem 2.2: For a Boolean function of n variables there exist $2^{E(n)}$ various CRMP forms, where

$$E(n) = \sum_{i=1}^{i=n} i\binom{n}{i} \qquad (2)$$

Proof: Consider all possible CRMP forms for a Boolean function of n variables. There exist $\binom{n}{i}$ subsets of variables with i variables in a subset. For each such subset there exist two polarities of each variable, which means 2^i polarities for all the variables of the subset. Therefore the number of all possible CRMP forms is

$$\prod_{i=1}^{i=n}(2^i)^{\binom{n}{i}} = \prod_{i=1}^{i=n} 2^{i\binom{n}{i}} = 2^{\sum_{i=1}^{i=n} i\binom{n}{i}} = 2^{E(n)} \qquad (3)$$

It can be proven by mathematical induction that $E(n) = n2^{n-1}$. □

Example 2.3: Let f be a function of $n = 4$ input variables a, b, c, d. There are $\binom{n}{1} = n$ one-variable sets $\{a\}$, $\{b\}$, $\{c\}$, and $\{d\}$. Each set can be in 2^1 polarities. There are 2^4 forms with various polarities of single literals. There are $\binom{n}{2} = 6$ two-variable sets $\{a, b\}$, $\{a, c\}$, $\{a, d\}$, $\{b, c\}$, $\{b, d\}$, and $\{c, d\}$. Each set can be in 2^2 polarities, which makes $(2^2)^6 = 2^{12}$ forms. There are $\binom{n}{3} = 4$ three-variable sets $\{a, b, c\}$, $\{a, b, d\}$, $\{a, c, d\}$, and $\{b, c, d\}$. Each set can be in 2^3 polarities, which makes $(2^3)^4 = 2^{12}$ forms. There exists $\binom{n}{4} = 1$ four-variable set $\{a, b, c, d\}$. Each variable can be in two polarities, which makes 2^4 forms. The total number of forms is therefore $2^4 \cdot 2^{12} \cdot 2^{12} \cdot 2^4 = 2^{32}$. Of course, theorem 2.2 gives the number of all possible forms. The actual number of binary vectors or expressions of these forms is usually much smaller. The number given in eqn. 3 is then the upper bound of the number of binary vectors. For instance, for function $f(a, b) = a$ there are 16 forms being ordered 4-tuples of elements $1/0$, $a/\bar{a}$, $b/\bar{b}$, $ab/a\bar{b}/\bar{a}b/\bar{a}\bar{b}$ but only two binary vectors, and two expressions a and $1 \oplus a$.

For a function of n variables there exists only a single RM form, while there are 2^n FPRM forms, and $2^{n2^{n-1}}$ CRMP forms, which is a very large number. Therefore, the cost of the minimum ESOP form for functions of few variables is usually much closer to the cost of the minimum CRMP form that to the cost of the minimum FPRM form of this function. This reason causes the superiority of the method proposed here over the methods of FPRM minimisation. Moreover, observe that the minimal FPRM form is always not better than that minimal CRMP form.

Next, some properties of the CRMP forms are given.

Definition 2.6: The Boolean difference [24] of function f with respect to variable x_i is denoted by f_{x_i} and defined as

$$f_{x_i} = f(x_1, \ldots, x_i, \ldots, x_n) \oplus f(x_1, \ldots, \overline{x_i}, \ldots, x_n)$$

Definition 2.7: The Boolean difference of function f with respect to term $t = x_i^c, x_j^c, \ldots, x_k^c$ is denoted by f_t and defined as

$$f_t = (\cdots(f_{x_i^c})_{x_j^c}\cdots)_{x_k^c}$$

Definition 2.8: Let t be a term. The term set $S(t)$ of t is $S(t) = \{x_i \mid x_i^c \text{ appears in } t\}$.

This is a set of all variables whose literals occur in term t. For instance, $S(x1, \overline{x_2}, x_3) = \{x1, x_2, x_3\}$.

Definition 2.9: Term t is a prime term with respect to function f iff $f_t \equiv 1$, where $\equiv$ stands for identical equality.

Lemma 2.1: The following properties hold for a Boolean difference:

$$f_{x_i} = f_{\overline{x_i}}$$

$$f_{x_ix_j} = f_{x_jx_i}$$

$$(x_ix_j)_{x_i} = x_j$$

Theorem 2.3: Term t is a prime term with respect to function f iff in any CRMP form of f there exists exactly one term t' such that $S(t) = S(t')$ and there exists no term t'' such that $S(t) \subset S(t'')$.

Proof: To prove in a forward direction assume that t is a prime term with respect to f, so $f_t \equiv 1$. Consider any CRMP form of f and compute f_t by computing t_t^* for each term t^* of that CRMP form and add the results over Galois field (2) together. The following three rules follow from lemma 2.1:

$S(t) = S(t^*)$ iff $t_t^* \equiv 1$

$S(t) \subset S(t^*)$ iff t_t^* is a term

for every other case $t_t^* \equiv 0$

Function f_t (when derived this way) itself is a CRMP form and thus is canonical, $f_t \equiv 1$ is possible in exactly one way, namely that there exists exactly one term t' such that $S(t) = S(t')$ and there exists no term t'' such that $S(t) \subset S(t'')$. To prove the converse choose a CRMP form of f and find term t'. Computer $f_{t'}$ from that from using Lemma 2.1. Clearly $f_{t'} \equiv 1$. Since $S(t) = S(t')$ it follows from the properties of the Boolean difference that $f_t \equiv 1$. Therefore, from definition 2.9 term t is a prime term with respect to function f. □

Some of the implications of this theorem are given by the following properties.

Property 2.1: The minimal CRMP (and FPRM) forms of f will have one term t' such that $S(t) = S(t')$ and there is no term t'' such that $S(t) \subset S(t'')$.

Property 2.2: For all existing terms $\underline{t}$ of a CRMP form of f there is a prime term t of f such that $S(\underline{t}) \subseteq S(t)$.

Property 2.3: Every function in a CRMP form has at least one prime term.

Definition 2.10: Term t is a nonexisting term with respect to function f iff $f_t \equiv 0$.

Theorem 2.4: Term t is a nonexisting term with respect to function f iff in any CRMP form of f there is no term t' such that $S(t) \subseteq S(t')$.

Proof: Similar to that of theorem 2.3.

Definition 2.11: A function $f(x_1, \ldots, x_n)$ is odd iff $f_{x_1, \ldots, x_n}$ is identically 1. A function $f(x_1, \ldots, x_n)$ is even iff $f_{x_1, \ldots, x_n}$ is identically 0.

It can be proven that the following properties are true.

Property 2.4: Every even function f in a CRMP form has at least one nonexisting term.

Property 2.5: If function f is odd it has no nonexisting terms in a CRMP form. For instance, $f_1 = x_1x_2x_3$ and $f_2 = x_1x_2x_3 \oplus x_1x_2$ are odd functions. Function $f_3 = x_1x_2 \oplus x_2x_3 \oplus x_1x_3$ is an even function and $x_1x_2x_3$ is a nonexisting term of this function.

In the next two Sections an upper and lower bounds on the number of existing terms in a CRMP form are proven.

3 Upper bound of existing terms in CRMP form

The goal in this Section is to prove that a termwise upper bound for the CRMP form is less than that for the conventional disjunctive (sum of products) or conjunctive (product of sums) expressions of a Boolean function.

We restate here a well-known theorem concerning the term upper bound for disjunctive or conjunctive expressions.

Theorem 3.1: Any Boolean function f of n variables can be described by at most 2^{n-1} terms in disjunctive or conjunctive expression. Moreover, the value 2^{n-1} is the least upper bound, since there are functions whose description needs exactly 2^{n-1} terms.

The CRMP form can be proven to be more economical in the sense that it has an upper bound with lower number of terms.

Theorem 3.2: Any Boolean function of n variables ($n \geqslant 3$) can be described by at most $\frac{3}{4}(2^{n-1})$ terms in a CRMP form. Furthermore, for any function f given in an RM form, there is an algorithm to find this CRMP form which takes $\frac{3}{4}(2^{n-1})$ steps.

Proof: See Appendix.

The same upper bound was obtained for ESOP forms in Reference 4, which shows that for difficult functions the CRMPs are as good as the ESOPs. An analogous theorem can be stated for the dual CRMP form ($\oplus \Leftrightarrow \odot$, $\cdot \Leftrightarrow +$).

4 Lower bound of existing terms in CRMP form

In this Section a property of the CRMP form is discussed which is related to the concept of the prime implicant.

Definition 4.1: Term t_1 is a proper subcombination of term t_2, iff $S(t_1) \subset S(t_2)$.

For instance, in the function $f = x_1x_2x_3\bar{x} \oplus \overline{x_1}x_3$ term $\overline{x_1}x_3$ is a proper subcombination of term $x_1x_2x_3\overline{x_4}$.

Theorem 4.1: All terms of a Boolean function f of n variables given in a CRMP form which are not subcombinations of other terms in the same CRMP form will exist in any CRMP form of f. Before giving the proof of this theorem we introduce some necessary definitions and lemmas.

Lemma 4.1: A term is a prime term (definition 2.9) if it is not a subcombination of other terms in the same CRMP.

Lemma 4.2: A Boolean function is odd iff its expanded disjunctive normal form (canonic sum of products form) contains an odd number of elementary terms (true minterms). A Boolean function is even iff its expanded disjunctive normal form contains an even number of minterms.

Lemma 4.3:

$$f_{1e}(x_1, \ldots, x_n) + f_{2e}(x_1, \ldots, x_n) = f_{3e}(x_1, \ldots, x_n)$$

$$f_{1e}(x_1, \ldots, x_n) + f_{2o}(x_1, \ldots, x_n) = f_{3o}(x_1, \ldots, x_n)$$

$$f_{1o}(x_1, \ldots, x_n) + f_{2e}(x_1, \ldots, x_n) = f_{3o}(x_1, \ldots, x_n)$$

$$f_{1o}(x_1, \ldots, x_n) + f_{2o}(x_1, \ldots, x_n) = f_{3e}(x_1, \ldots, x_n)$$

where subscript e means an even function, and subscript o means an odd function.

Proof: Consider the expanded disjunctive normal form of the preceding functions. In such a form $+$ can be replaced by $\oplus$ without changing the function. Thus, using lemma 4.2, the number of the remaining terms on the left sides will be (number of terms of f_1) + (number of terms of f_2) $-$ 2(number of common terms). Therefore the rules concerning the parity of ring sums of Boolean functions are identical to those of the natural numbers. □

Lemma 4.4: Assume that the Boolean function f of n variables in CRMP form has one term consisting of k literals and no other terms. Then

$f(x_1, \ldots, x_n)$ if even iff $k < n$

$f(x_1, \ldots, x_n)$ is odd iff $k = n$

Proof: Consider the expanded normal disjunctive form of the function. The number of minterms will be 2^{n-k}, which is always even, unless $n = k$. □

Lemma 4.5: A Boolean function of n variables is odd iff the CRMP form of f contains the term with n literals.

Proof: $\Rightarrow$. If f is odd then its CRMP form must be also odd. By lemma 4.3 the CRMP form must contain an odd number of terms describing odd functions. By lemma 4.4 the only such term is one containing n literals. Thus the term with n literals must exist in the CRMP form of f.
$\Leftarrow$. Conversely, if the CRMP form of f contains the term of n literals, then by lemma 4.4 f is the ring sum of an odd function and some even functions. By lemma 4.3 we conclude that f is odd.

Now we can give the proof of theorem 4.1.

Proof of theorem 4.1

(i) Choose arbitrary prime term containing $k \leqslant n$ variables in the CRMP form of f. From lemma 4.1. It is not a subçombination of other terms of this CRMP.

(ii) Give constant values to all the $n - k$ variables not occurring in the chosen prime term.

(iii) Since the prime term is not a subcombination of other terms by hypothesis, in the resulting k variable function the chosen prime term will be the only one term containing k literals. So, according to lemma 4.5, this segment of f is odd.

(iv) This segment will remain, therefore, odd in every polarity, and its terms will never be combinations of other terms.

(v) Points 1–4 can be repeated for any prime term.

The following corollaries can be derived from theorem 4.1.

Corollary 4.1: The prime terms will exist in the minimal CRMP form too.

Corollary 4.2: All existing terms in any CRMP form of a Boolean function f of n variables are subcombinations of prime terms.

Proof: Trivial.

Corollary 4.3: For a given Boolean function f of n variables the prime terms are entirely determined by f and they do not depend on the CRMP form from which they are determined.

Proof: Let C_{CRMP1} be the class of the term sets $S(t_i)$ of the prime terms t_i in CRMP1 form of f and C_{RMP2} the class of the term sets $S(t_j)$ of the prime terms t_j in CRMP2 form of f. For reasons of symmetry it suffices to show $C_{CRMP1} \subset C_{CRMP2}$. Assume $S(t) \in C_{CRMP1}$. Then by theorem 4.1 there exists a term t' in CRMP2 form such that $S(t) = S(t')$. Assume $S(t') \notin C_{CRMP2}$, i.e. t' not a prime term. Then there exists term t'' such that $S(t') \subset S(t'')$ and $S(t'') \in C_{CRMP2}$. By theorem 4.1 there exists term t'' in CRMP1 form such that $S(t''') = S(t'')$ so that $S(t') \subset S(t''')$ and $S(t') \notin C_{CRMP1}$. This contradiction establishes corollary 4.3.

Corollary 4.4: There exists a Boolean function of n variables for which the minimal CRMP form contains as many as

$$\binom{n}{\frac{n}{2}}$$

terms if n is even, and

$$\binom{n}{\frac{n-1}{2}} = \binom{n}{\frac{n+1}{2}}$$

terms if n is odd.

Proof: If n is even consider the function whose RM form contains all terms with $n/2$ variables and no other terms. Since none of those terms is a subcombination of any of the other terms all these terms are prime terms. If n is odd consider either a function whose RM form contains all terms with $(n - 1)/2$ variables and no other terms or the function whose RM form contains all terms with $(n + 1)/2$ variables and no other terms. It is easy to see that in both cases all terms are prime terms.

This proves that the conjectured upper bound on the number of terms in a termwise minimal CRMP form cannot be further decreased. Corollary 4.4 was given without proof in Reference 28.

5 Prime terms in CRMP minimisation

To explain the CRMP minimising algorithms, we first illustrate the fundamental concepts of prime and nonexisting terms.

Examples: In the functions below the prime terms are underlined and the nonexisting terms are listed.

(i) $1 \oplus x_1 \oplus x_2 \oplus \underline{x_1x_2x_3}$

There are no nonexisting terms.

(ii) $1 \oplus x_1 \oplus \underline{x_1x_2} \oplus \underline{x_1x_3} \oplus \underline{x_2x_3}$

Nonexisting terms: $x_1x_2x_3$

(iii) $1 \oplus \underline{x_1} \oplus x_2 \oplus \underline{x_2x_3}$

Nonexisting terms: $x_1x_2, x_1x_3, x_1x_2x_3$

(iv) $x_1 \oplus x_2 \oplus \underline{x_3} \oplus x_4 \oplus \underline{x_1x_2x_4}$

Nonexisting terms:
$x_1x_3, x_2x_3, x_3x_4, x_1x_2x_3, x_1x_3x_4,$
$x_2x_3x_4, x_1x_2x_3x_4$

(v) $x_1 \oplus x_2x_3 \oplus \underline{x_1x_2x_3x_4}$

Nonexisting terms: none.

(vi) $\underline{x_1} \oplus \underline{x_2x_3} \oplus \underline{x_2x_4} \oplus \underline{x_3x_4}$

Nonexisting terms:

$x_1x_2, x_1x_3, x_1x_4, x_1x_2x_3, x_1x_2x_4, x_1x_3x_4,$
$x_2x_3x_4, x_1x_2x_3x_4.$

If there exist only prime terms in the expression, then this expression is a both termwise and literalwise minimal CRMP form. If one can merge other terms with prime terms so that the resultant form has the same number of terms as the number the prime terms, the resultant form is miminal.

Illustration of some of the preceding cases:

(ii) $f = 1 \oplus x_1 \oplus x_1x_2 \oplus \underline{x_1}x_3 \oplus x_2x_3$ after merging x_1 and x_1x_2 becomes $1 \oplus x_1\overline{x_2} \oplus x_1x_3 \oplus x_2x_3$. Now one can see that variable x_2 occurs in both forms. Also adding $\underline{x_2}$ to the form would permit to merge it with 1 to create $\overline{x_2}$, which in turn could be merged with $x_1\overline{x_2}$. Therefore, the preceding form becomes $\underline{1 \oplus x_1\overline{x_2}} \oplus \underline{x_1x_3}$ $\oplus x_2x_3 \oplus x_2 \oplus x_2 = x_1\overline{x_2} \oplus x_1x_3 \oplus x_2\overline{x_3} \oplus \overline{x_2} = \overline{x_1}\,\overline{x_2}$ $\oplus x_1x_3 \oplus x_2\overline{x_3}$. All terms are prime, so the solution is an exact minimum CRMP form. (In this case it is an exact minimum ESOP as well). Observe that the exhaustive search of References 22 and 23 to find the exact ESOP has been avoided.

(iii) One expects a form with two terms. By merging 1 with $\underline{x_1}$ and $\underline{x_2}$ with x_2x_3 one obtains an exact minimum form $\overline{x_1} \oplus x_2\overline{x_3}$.

(vi) All terms are prime, so the exact minimum form is $x_1 \oplus x_2x_3 \oplus x_2x_4 \oplus x_3x_4$.

Similar mergings do not always lead to minimal CRMP forms, but usually reduce the number of terms. For instance, 1 and x_1 can be merged in (i). Moreover, by knowing the number of prime terms, which is a lower bound of a termwise solution cost, one can evaluate the upper bound of the distance of this solution cost from the minimal cost. The solution (iv) for instance has in the worst case three terms more than the exact minimum solution.

The ideas of prime terms are used in a tree-searching algorithm to find an exact CRMP.

Algorithm 5.1: Calculation of exact CRMP form from FPRM form.

(*a*) Find set PT of all prime terms from an arbitrary FPRM of f.

(*b*) Find set PRT of all product terms corresponding to the prime terms from PT. This is done by changing in all possible ways polarities of all variables in terms from PT (we create for prime term t all such terms t' that $S(t) = S(t')$).

(*c*) Find set PRT1 of product terms that can be created from the product terms of PRT by removing all possible subsets of literals.

(*d*) Using tree search select the smallest subset of groups from PRT1 such that the EXOR of all those groups equals to f. When in some branch a product term t is selected to the solution, all possible product terms t' such that $S(t) = S(t')$ are discarded in all branches of the tree that starts from t.

Observe that the exhaustive search algorithm for exact ESOPs from Reference 23 can be easily adopted to find exact CRMPs. In such a case the algorithm is made much more efficient by not using in H-function groups g_i that correspond to nonexisting terms. During search to find products of g_i to satisfy the decision function H, the groups g_i are selected on the same principles as the product terms in (*d*) of algorithm 5.1

The concept of the prime term makes it possible to define the following 'generic' and nondeterministic, simple algorithm for the minimisation of CRMP forms. The input to the algorithm can be any FPRM form.

Algorithm 5.2: Calculation of quasiminimum CRMP form from FPRM form.

(*a*) $i := 1, f_1 := f$;

(*b*) decompose $f_i = f_{ip} \oplus f_{i+1}$ where f_{ip} is the modulo 2 sum of prime terms with respect to f_i such that

(i) terms t and t' occurring in f_{ip} imply that $S(t') \not\subset S(t)$

(ii) $(f_i)_t \equiv 1$ implies $(f_{ip})_t \equiv 1$

(iii) set $\{S(t) | (f_{i+1})_t \equiv 1\}$ has the minimal cardinality $i := i + 1$.

(*c*) if $f_{i+1} \neq 0$, repeat step 2.

(*d*) print solution: $f = f_{1p} \oplus f_{2p} \oplus \cdots \oplus f_{ip}$.

6 Implementation

Agorithm 5.2 can be implemented in many different ways. Our implementation called Cannes (CANonic Nor Exor Synthesiser) has been based on a depth-first tree-searching algorithm that makes use of the theory introduced and in particular the properties given in corollary 4.2 and 4.3. Those corollaries state that all prime terms are entirely determined by the Boolean function f and do not depend on the CRMP form and that all existing terms in a CRMP form of f are subcombinations of prime terms. Thus, Cannes is based on an algorithm which generates the minimal FPRM form for the prime terms and their subcombinations. The simplified recursive minimisation procedure of Cannes is as follows.

Notation:

List	– complete list of terms describing the function
NewList	– starting List of next recursion
prime term	– prime term of the List
subset	– subset of terms for a prime term
minsubset	– minimal FPRM form of the subset

```
minimise(List) {
  for each prime term of the List {
    // calculate the subset for the prime term
    subset := subset_of (prime term);

    //calculate the minimal FPRM form of the subset
    minsubset := minimal_FPRM (subset);

    // compare number of terms
    if (|minsubset| < |subset|) {
        NewList := List;
        relace subset in NewList by minsubset;
        minimise( NewList ); }}}
```

Currently Cannes checks the FPRM forms for all possible polarities of a subset of variables. However, observe that although our method requires finding the minimal FPRM, and even generates the minimal FPRM several times during the CRMP minimisation, the exact FPRM generation has to be performed on subfunctions that depend only on subsets of variables of the initial function. It is only in the worst case of a FPRM form having a single prime term that all the polarities of the input variables have to searched to find the minimum FPRM. With an amount of search that is comparable to that for a FPRM, we are able to find a form that is not worse than the FPRM.

Replacing the current FPRM minimiser in Cannes with the exact FPRM minimiser from Reference 26 will speed it up significantly. Moreover, we will be able to study the performance of the new heuristic variant of Cannes using the quasiminimal FPRM minimiser from Reference 26. We have done it for single-output functions [30] and the results are very encouraging, but to use the Cannes program presented in this paper, the FPRM minimiser from [26] has to be first generalised to multi-output case.

7 Complete example of execution of Cannes program

For the description of the algorithm the function shown in Fig. 1 is taken. For comparison, the SOP solution given by the well-known minimisation program Espresso [21] and the CRMP solution calculated by Cannes are shown in Fig. 2. First, all possible FPRMs are generated for the input function. A FPRM with the minimal number of terms is selected: ----, 0---, --0--, -0--, ---0-, 00-0-, ----0, -0-00. Fig. 3 illustrates the different steps of the minimisation of this FPRM. One prime term is 00-0-. Its subset is 00-0-, 0---, ----0-, as shown in Fig. 3*a*. The next step is the calculation of the minimal FPRM form of this subset. Because all FPRM forms do not have less cubes than the initial subset, the prime term is a final

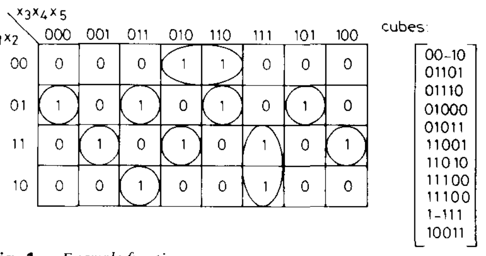

Fig. 1 *Example function*

solution term. Therefore it is stored in the solution array (first cube in the solution array from Fig. 3*b*). In the remaining array the term 0--- is now a prime term without subcombinations. Thus it can be removed and stored in the solution array.

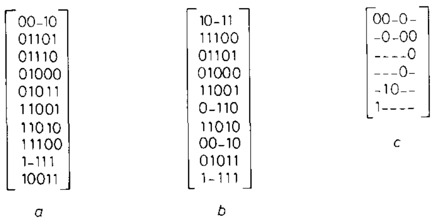

Fig. 2 *Comparison of Espresso with Cannes*
a Input array
b Minimisation by Espresso
c Minimisation by Cannes

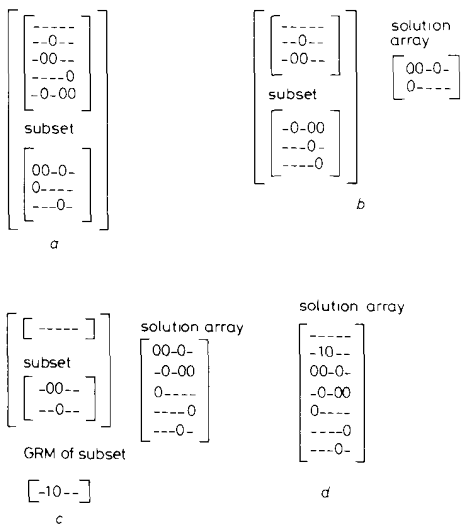

Fig. 3 *Step-by-step minimisation of array*
a First loop
b Second loop
c Third loop
d Fourth loop

The prime term -0-00 with its subcombinations (shown as a subset in Fig. 3*b*) also does not lead to a more minimal solution. Therefore the prime term is also a final solution term and has to be inserted in the solution array. Also the prime terms of the remaining array which have no subcombinations have to be inserted in the solution array (Fig. 3*c*). Again, the next prime term is taken.

Table 1 shows the subset (Fig. 3*c*) of this prime term and the minimal FPRM form.

The final two terms are prime terms and are therefore the solution terms (Fig. 3*d*). As one can observe this solution is the not the final solution show in Fig. 2*c*. The general minimisation according to the algorithm is now

Table 1: Minimal FPRM of subset

Subset	Minimal FPRM
-00-- --0--	-10--

completed. As the last minimisation step it is tried to substituted a possible DC term (----). In our example ----⊕ 0---- = 1---. With this last iteration we obtain the result shown in Fig. 2*c*.

8 Evaluation of results

For comparison, some two-level examples from the MCNC benchmarks have been minimised. Because current Cannes has an exhaustive search routine for the minimal FPRM form, the computation for functions having more than 15 input variables is very time consuming.

Table 2: MCNC benchmark comparison

	Input variables	Output variables	SOP terms	FPRM terms	Cannes terms	Prime terms
5xp1	7	10	65	60	60	1/7
9sym	9	1	86	173	131	8
bw	5	28	22	22	22	1/4
con1	7	2	9	17	12	4
inc	7	9	30	47	43	5
misex1	8	12	12	20	20	3
rd53	5	3	31	20	20	5
rd73	7	3	127	63	63	5
sao2	10	4	58	100	52	1/8
squar5	5	8	25	23	22	4
xor5	5	1	16	5	5	5
Z5xp1	7	10	65	60	60	6

The column prime terms lists the number of prime terms in the minimised form. A $1/m$ indicates that the form contains a minterm and apart from that m prime terms. In the column FPRM the number of terms of a minimal FPRM form, obtained by exhaustive search, is given. The number of SOP terms are the results obtains by Espresso.

9 Discussion of results, conclusion and related work

The concept of CRMP forms and an approach to their minimisation have been introduced. We believe the most important aspect of the theory introduced and of Cannes is that they will be useful to create an exact ESOP minimiser.

The Cannes algorithm has been tested on many example, and on all small functions that can be verified (such as all single output functions of three and four variables), if the function was not sparse, it produced the exact CRMP solution. It is not known whether the Cannes algorithm always produces the exact CRMP for nonsparse function; we have not found a counterexample yet. Cannes was compared with a good quality ESOP minimiser Exorcism-mv-2 [33], and we were able to find some functions for which Exorcism-mv-2 found better solutions than Cannes. However, all those functions were sparse. Therefore, there are two questions open:

(i) Suppose the original function is sparse, is partitioned to all its clusters, and the exact CRMP solution for each cluster is found. Then, is the EXOR of these CRMPs the exact ESOP of the original function?

(ii) Does the algorithm from Section 6 find the exact CFMP form?

The second hypothesis can be easily verified, since we developed the exact CRMP minimising algorithm 5.1 (which is more efficient than the exact ESOP minimiser form Reference 23), and which will be programmed and used in the comparison.

Another property that is useful in the exact algorithm is the concept of prime terms introduced previously. It allows one to decompose a function to disjoint subsets of input variables for which minimisation can be done separately. Also, it allows one to find the upper bound of the distance of the solution found from the exact minimum solution (the lower bound is given by the initial number of prime terms). The concepts of disjoint decomposition and upper bound are both new in the EXOR theory. They have been extended and are used to design a better exact ESOP minimiser. For instance, these concepts aid in fast clustering of a sparse function to non-sparse clusters for which CRMPs are found.

Unfortunately, although we can create whole classes of functions for which the proposed approach will lead to minimum solutions without much search, the MCNC benchmark examples show that on the real-life examples the number of prime terms is much smaller than the number of terms in the minimal solution, which causes these variable-based decompositions to occur rarely, and the upper bound distance evaluations to be too pessimistic.

It is surprising that for many practical examples of multioutput functions of many variables the optimal solution found was an FPRM form. This means that, because of only a few prime terms in such functions, either our algorithm does not produce the exact CRMP solution, or that those functions the exact CRMP is the exact FPRM, which would be a rather unexpected result since the class of CRMPs is much larger than the class of FPRMs.

These interesting questions give motivation to investigate further the exact CRMP and sparse function decomposition algorithms. This gives also an additional motivation to the generalisation of the efficient exact and quasi-minimum FPRM algorithms from Reference 26 for multioutput and incompletely specified functions.

Moreover, use of CRMPs and FPRMs is important with respect to the logic design methods for the field programmable gate arrays [9, 13] and other programmable devices [10, 14]. Our results show that a FPRM [26] or a CRMP form can be not only more easily testable but also smaller than the two-level inclusive form. Similary to SOPs, they can be factorised [15], which further reduces their areas. Since the cost and speed of an EXOR gate and OR gate in such technologies are the same, there is no reason to assume that PLA minimisers such as Espresso will always produce better results. However, Espresso can be still applied to larger functions and is still faster than Cannes.

Since for some functions the results of Cannes and Espresso can vary significantly (see the *rd*73 function), and until software becomes available to automatically find the best AND/OR/EXOR mixture, we suggest that the CAD user will run both the SOP minimiser and the ESOP minimiser for any particular function he minimises, and will make his final implementation decision based on the comparison of their results.

10 References

1 BESSLICH, Ph.W.: 'Efficient computer method for EXOR logic design', *IEE Proc. E*, 1983, **130**, (6), pp. 203–206

2 GREEN, D.H.: 'Modern logic design'. Electronic Systems Engineering Series, 1986

3 HELLIWELL, M., and PERKOWSKI, M.A.: 'A fast algorithm to minimize multi-output mixed-polarity generalized Reed–Muller forms'. Proceedings of 25th ACM/IEEE Conference on *Design automation*, Anaheim, CA, USA, 1988, pp. 427–432

4 SASAO, T., and BESSLICH, PhW.: 'On the complexity of mod-2 sum PLA', *Trans. Electron. Commun. Eng. Jpn.*, 1986, **17**, pp. 1–8

5 SASAO, T.: 'A transformation of multiple-valued input two-valued output functions and its application to simplification of exclusive-or-sum-of-products expressions'. Proceedings of IEEE 21st International Symposium on *Multiple-valued logic*, Victoria, BC, Canada, 1991, pp.270–279

6 SASAO, T.: 'Optimization of multiple-valued AND-EXOR expressions using multiple-place decision diagrams'. Proceedings of IEEE 22nd International Symposium on *Multiple-valued logic*, Sendai, Japan, 1992

7 SASAO, T.: 'EXMIN2: A tool for EXOR logic synthesis'. Proceedings of SASIMI '92 conference, 1992, pp. 46–53

8 BRAND, D., and SASAO, T.: 'On the minimization of and-exor expressions'. Proceedings of 23rd IEEE conference on *Fault-tolerant computing*, 1990, pp. 1–9

9 'Programmable gate array data book'. Xilinx, 1989

10 'PLD Data Manual'. Signetics, 1990

11 'ACT-family field programmable gate array data book'. Actel, 1991

12 'Configurable array logic user manual'. Algotronix, Edinburgh, UK, 1991

13 'CLi6000 series field programmable gate array'. Concurrent_Logic, 1991

14 DETJENS, E.: 'FPGA devices require FPGA-specific synthesis tools', *Computer Design*, 1990, November, p. 124

15 SAUL, J.M.: 'An algorithm for the multi-level minimization of Reed–Muller representation'. Proceedings of IEEE ICCD, 1991, pp. 634–637

16 REDDY, S.M.: 'Easily testable realizations for logic functions', *IEEE Trans.*, 1972, **C-21**, (11), pp. 1183–1188

17 SALUJA, K.K., and REDDY, S.M.: 'Fault detecting test sets for Reed–Muller canonic networks', *IEEE Trans.*, 1975, **C-24**, (10), pp. 995–998

18 KODANDAPANI, K.L.: 'A note on easily testable realizations for logic functions', *IEEE Trans.*, 1974, **C-23**, pp. 332–333

19 PRADHAN, D.K.: 'Universal test sets for multiple fault detection in AND-EXOR arrays', *IEEE Trans.*, 1978, **C-27**, (2), pp. 181–187

20 DAMARLA, T., and KARPOWSKY, M.: 'Detection of stuck-at and bridging faults in Reed–Muller canonical (RMC) networks', *IEE Proc. E*, 1989, **136**, (5), pp. 430–433

21 BRAYTON, R.K., HACHTEL, G.D., McMULLEN, C.T., and SANGIOVANNI-VINCENTELLI, A.L.: 'Logic minimization algorithms for VLSI synthesis' (Kluwer, 1984)

22 PAPAKONSTANTINOU, G.: 'Minimization of modulo-2 sum of products', *IEEE Trans.*, 1979, **C-28**, (2), pp. 163–167

23 PERKOWSKI, M., and CHRZANOWSKA-JESKE, M.: 'An algorithm to minimize mixed-radix exclusive sums of products for incompletely specific Boolean functions'. Proceedings of ISCAS international symposium on *Circuits and systems*, New Orleans, USA, 1990, pp. 1652–1655

24 DAVIO, M., DESCHAMPS, J.P., and THAYSE, A.: 'Discrete and switching functions' (McGraw-Hill, 1978)

25 MUKHOPADHYAY, A., and SCHMITZ, G.: 'Minimization of exclusive-OR and logical equivalence switching circuits', *IEEE Trans.*, 1970, **C-19**, (2), pp. 132–140

26 SARABI, A., and PERKOWSKI, M.A.: 'Fast exact and quasi-minimal minimization of highly testable fixed-polarity AND/XOR canonical networks'. Proceedings 29th conference on *Design automation*, Anaheim, CA, USA, 1992, pp. 30–35

27 CALINGAERT, P.: 'Switching function canonical forms based on commutative and associative binary operations', *AIEE Trans.*, 1961, **79**, pp. 808–814

28 COHN, M.: 'Inconsistent canonical forms of switching functions', *IRE Trans.*, 1962, **EC-11**, p. 284

29 CSANKY, L.: 'On the generalized Reed–Muller canonical form of Boolean functions'. MSc thesis, University of California, Berkeley (from Michigan Repository of MS theses), 1972

30 PERKOWSKI, M.A., CSANKY, L., SARABI, A., and SCHÄFER, I.: 'Minimization of mixed-polarity canonical AND/EXOR forms'. Proceedings of ICCD, 1992 (to be published)

31 GREEN, D.H.: 'Families of Reed–Muller canonical forms', *Int. J. Electron.*, 1991, **63**, (2), pp. 259–280

32 PERKOWSKI, M.A., HELLIWELL, M., and WU, P.: 'Minimization of multiple-valued input multi-output mixed-radix exclusive sums of products for incompletely specified Boolean functions'. Proceedings of 19th International Symposium on *Multiple-valued logic*, 1989, pp. 256–263

33 SONG, N., and PERKOWSKI, M.: 'Exorcism-mv-2: minimization of exclusive sum of products expressions for multiple-valued input incompletely specified functions'. Portland State University, 1992

11 Appendix

To prove theorem 3.2 we first prove the following lemmas.

Lemma 3.1: Any Boolean function f of n variables can be given in the form

$$f(x_1, \ldots, x_n) = g_1(x_1, \ldots, x_{n-1}) \oplus x_n g_2(x_1, \ldots, x_{n-1})$$

where g_1 and g_2 are Boolean functions of $n - 1$ variables.

Proof: Consider function $f(x_1, \ldots, x_n)$ in the RM form. From the Shannon expansion theorem one obtains $f(x_1, \ldots, x_n) = \overline{x_n} f(x_1, \ldots, x_{n-1}, 0) + x_n f(x_1, \ldots, x_{n-1}, 1)$. Since the functions $\overline{x_n} f(x_1, \ldots, x_{n-1}, 0)$ and $x_n f(x_1, \ldots, x_{n-1}, 1)$ are disjoint, $f(x_1, \ldots, x_n) = \overline{x_n} f(x_1, \ldots, x_{n-1}, 0) \oplus x_n f(x_1, \ldots, x_{n-1}, 1)$. Substituting $\overline{x_n}$ by $x_n \oplus 1$ one obtains:

$$\begin{aligned} f(x_1, \ldots, x_n) &\\ &= (x_n \oplus 1) f(x_1, \ldots, x_{n-1}, 0) \\ &\quad \oplus x_n f(x_1, \ldots, x_{n-1}, 1) \\ &= x_n (f(x_1, \ldots, x_{n-1}, 0) \oplus f(x_1, \ldots, x_{n-1}, 1)) \\ &\quad \oplus f(x_1, \ldots, x_{n-1}, 0) \\ &= g_1(x_1, \ldots, x_{n-1}) \oplus x_n g_2(x_1, \ldots, x_{n-1}) \end{aligned}$$

Lemma 3.2: Any Boolean function f of n variables can be given in the form:

$$\begin{aligned} f(x_1, \ldots, x_n) &\\ &= g(x_1, \ldots, x_k) \oplus x_{k+1} g_2(x_1, \ldots, x_k) \\ &\quad \oplus \cdots \oplus x_n g_{n-k+1}(x_1, \ldots, x_k) \\ &\quad \oplus \cdots \oplus x_{k+1} x_{k+2} \cdots x_n g_{2^k}(x_1, \ldots, x_k) \end{aligned}$$

Proof: Apply Lemma 3.1 $(n - k)$ times.

Lemma 3.3: Considering term $x_1 x_2$ with all polarity combinations, terms x_1 and x_2 occur in all possible combinations of existence, which means: none of them, any one of them, or both of them.

Proof:

$$\begin{aligned} x_1 x_2 &= \overline{x_1} x_2 \oplus x_2 \\ &= x_1 \overline{x_2} \oplus x_1 \\ &= \overline{x_1}\,\overline{x_2} \oplus x_2 \oplus x_1 \oplus 1 \end{aligned}$$

Lemma 3.4: Considering term $x_1 x_2 x_3$ will all polarity combinations, terms $x_1 x_2$, $x_1 x_3$ and $x_2 x_3$ occur in all possible combinations of existence.

Proof: Changing the polarity of variables x_1, x_2 only and applying lemma 3.3 we obtain

$$0; \quad x_2 x_3; \quad x_1 x_3; \quad x_1 x_3, x_2 x_3;$$

Changing the polarity of variable x_3 as well, one obtains

$$\begin{aligned} x_1 x_2 x_3 &= x_1 x_2 \overline{x_3} \oplus x_1 x_2 \\ &= x_1 \overline{x_2}\,\overline{x_3} \oplus x_1 x_2 \oplus x_1 x_3 \oplus x_1 \\ &= \overline{x_1} x_2 \overline{x_3} \oplus x_1 x_2 \oplus x_2 x_3 \oplus x_2 \\ &= \overline{x_1}\,\overline{x_2}\,\overline{x_3} \oplus x_1 x_2 \oplus x_1 x_3 \oplus x_2 x_3 \\ &\quad \oplus x_1 \oplus x_2 \oplus x_3 \oplus 1 \end{aligned}$$

□

Lemma 3.5: Any Boolean function f of three variables can be given in a CRMP form with at most three terms.

Proof: It takes at most three steps to find that CRMP form.

Case (i): Term $x_1 x_2 x_3$ exists in function f.

Step #1: Choose polarities for variables x_1, x_2, x_3 so that out of the two-variable terms only term $x_1 x_2$ will exist (possible by lemma 3.4).

Step #2: Choose polarities of variables x_1, x_2 so that terms 1 and/or x_3 exist (possible by lemma 3.3).

Step #3: If both 1 and x_3 exist, merge and get $\overline{x_3}$.

Case (ii): The existing terms containing the most literals are $x_1 x_2$; $x_1 x_3$; $x_2 x_3$.

Step #1: Choose polarities: $x_1 x_2$; $x_1 x_3$; $x_2 \overline{x_3}$.

Step #2: Choose polarities for variables x_1, x_2 so that neither term x_1 nor term x_2 exists (possible by lemma 3.3).

Step #3: If only x_3 exists merge x_3 into $x_1 x_3$ and get $\overline{x_1} x_3$. If only 1 exists, merge 1 into $x_1 x_3$; $x_2 \overline{x_3}$ to get $\overline{x_1} x_3$; $\overline{x_2}\,\overline{x_3}$ (it means, use the transformation: $1 \oplus x_1 x_3 \oplus x_2 \overline{x_3} = x_3 \oplus \overline{x_3} \oplus x_1 x_3 \oplus x_2 \overline{x_3} = \overline{x_1} x_3 \oplus \overline{x_2}\,\overline{x_3}$). If both 1 and x_3 exist, merge them into $x_1 x_3$; $x_2 \overline{x_3}$ to get $x_1 x_3$; $\overline{x_2}\,\overline{x_3}$.

Case (iii): The existing terms containing the most literals are either $x_1 x_2$; $x_1 x_3$ or $x_1 x_2$; $x_2 x_3$ or $x_1 x_3$; $x_2 x_3$. For reason of symmetry it suffices to show only one of those three cases. Let this be $x_1 x_2$; $x_1 x_3$.

Step #1: Choose polarities for variables x_1, x_2 so that neither term x_1 nor term x_2 will exist (possible by lemma 3.3). May be $1 \oplus 1$ has been added to permit mergings.

Step #2: There are still four terms, if both terms 1 and x_3 exist. Merge $1 \oplus x_3$, and get $\overline{x_3}$.

Case (iv): The existing terms containing the most literals are either $x_1 x_2$ or $x_1 x_3$ or $x_2 x_3$. For reasons of symmetry it suffices to show only one of those three cases. Let this be $x_1 x_2$.

Step #1: Choose polarities for variables x_1, x_2 so that neither x_1 nor x_2 will exist (possible by lemma 3.3).

Case (v): The existing terms containing the most literals are x_1; x_2; x_3.

Step #1: If 1 exists merge and get $\overline{x_1}$; x_2; x_3.

□

Now we can prove theorem 3.2.

Proof (theorem 3.2): Given a Boolean function f of n variables in RM form, according to lemma 3.2 write it in the form

$$\begin{aligned} f(x_1, \ldots, x_n) &= g_1(x_1, x_2, x_3) \oplus x_4 g_2(x_1, x_2, x_3) \\ &\quad \oplus x_5 g_3(x_1, x_2, x_3) \oplus x_4 x_5 g_4(x_1, x_2, x_3) \\ &\quad \oplus \cdots \oplus x_4 x_5 x_6 \cdots x_n g_{2^{n-3}}(x_1, x_2, x_3) \end{aligned}$$

Minimise each g_i by lemma 3.5 to obtain their CRMP forms containing at most three terms. This process takes $3 \times 2^{n-3}$ steps. The resulting CRMP form for f will contain $3 \times 2^{n-3} = \frac{3}{4}(2^{n-1})$ terms.