Approximate Minimization Of Generalized Reed-Muller Forms

Xiaoqiang Zeng, Marek Perkowski, Karen Dill

Portland State University,
Dept. of Electr. Engn.
Portland, Oregon 97207
Tel: 503-725-5411,
Fax: 503-725-4882
e-mail: mperkows@ee.pdx.edu

Andisheh Sarabi

Viewlogic Systems, Inc.
47211 Lakeview Blvd.,
Fremont, CA 94538
Tel: (510) 659-4001x330,
Fax: (510) 659-4010
e-mail: andisheh@wcdf.viewlogic.com

Abstract— Paper presents a new approach to Generalized Reed-Muller form minimization. The concept of adjacent polarity of forms is introduced, that is next used to generate exact and approximate solutions. An algorithm to generate a GRM form from any other form is given. While adjacent polarities are used for local optimization, global minimum is found using the adaptation of Genetic Algorithm approach. For the first time, the problem has been solved for multiple-output, incompletely specified functions.

I. Introduction

Mathematically, the Generalized Reed-Muller (GRM) forms [2, 4, 11] do not exhibit a general structure in the nested hierarchy of families of canonical forms in [7] because most of them are not constructed from Kronecker matrix products [6]. Few computational methods for GRM forms are in [3, 11, 5, 18, 17]. Let us point out that from the practical point of view, the GRM forms($2^{n2^{n-1}}$) are the most interesting because they contain many more forms than KRM forms(3^n) when $n \geq 2$. Hence, generally, the minimal GRM form will be closer to the minimal of ESOP than the minimal KRM form. It was also clearly shown by the numerical results of Sasao [5, 18], that for functions of up to 6 variables, the minimal GRMs found by his exact algorithm are very close to the minimal ESOPs. However, an exhaustive search for a minimal GRM form becomes computationally unfeasible for more variables, and heuristic algorithms should be looked for. In addition, as presented in [20, 16], GRMs are easily testable, and various test generation algorithms for them have been created.

The goal of this research is to create a high quality approximate algorithm for GRMs of many variables for multi-output incompletely specified functions. The paper is organized as follows. Section 2 introduces basic concepts and definitions for GRMs. Then, in section 3, the term-wise complementary expansion diagram is introduced. Using this diagram, one can calculate any GRM expansion with expected polarity from any other GRM expansion. The exact minimum GRM form can be obtained by an exhaustive search through all $2^{n2^{n-1}}$ Gray code ordered polarity vectors. A local heuristic minimization algorithm is also presented in section 3. Instead of depending on the number of input variables, the computation time of this algorithm depends mainly on the complexity of the function, thus it can solve much larger problems. The algorithm in section 4 uses Genetic Algorithm for finding good starting point to a local optimization algorithm. Section 5 presents the handling of incompletely specified functions. Some details of program implementation are discussed in section 6, and numerical results are given in section 7.

II. Basic Concepts of Generalized Reed-Muller Expansion

Literal $\dot{x_i}$ is a variable x_i in either positive ($\dot{x_i} = x_i$) or negative ($\dot{x_i} = \bar{x}_i$) form. Literal x_i is the complementary literal of literal $\bar{x}_i$. A product of distinct literals is called a product term (product or term in short). If terms T_1 and T_2 contain exactly the same literals, they will be called identical.

The expansion of an n-variable function can be written as follows:

$$f(x_1, x_2, ..., x_i, ..., x_n) = b_0 x_1^0 x_2^0 ... x_i^0 ... x_n^0 \oplus b_1 x_1^0 x_2^0 ... x_i^0 ... x_n^1 \oplus ... \oplus b_j x_1^{e_1} x_2^{e_2} ... x_i^{e_i} ... x_n^{e_n} \oplus ... \oplus b_{2^n-1} x_1^1 x_2^1 ... x_i^1 ... x_n^1 \quad (2.1)$$

where $x_i^0 = 1$, $x_i^1 = x_i$, $(i = 1, 2, ..., n)$; $b_j \in \{0,1\}$, (j=0,1,2,..., 2^n-1), $e_i \in \{0,1\}$, and $(j)_{10} = (e_1 e_2 ... e_i ... e_n)_2$.

This is defined as the Canonical Reed-Muller Expansion. Since all literals in this expression are positive, Eqn. (2.1) is also called a *Positive Polarity Reed-Muller (RM) expansion.*

Definition 2.1(a). The GRM forms are created by selecting any set of literals in the Positive Polarity (or zero polarity) RM expansion and replacing these literals with their inverses [4, 11].

From Eqn. (2.1) we observe that the RM expansion has 2^n terms. Each term is a product of a subset of the n variables. This subset can be identified by the "1's" in the corresponding n-bit binary numbers $(e_1e_2...e_n)_2 = (j)_{10}$. Thus, the following definition complies with Definition 2.1(a).

Definition 2.1(b). The form of Eqn. (2.1) in which each variable can be both positive and negative but in which there is exactly one coefficient for each subset of variables of a term will be called a *Generalized Reed-Muller* expansion.

Definition 2.2. By a *Fixed-Polarity Reed-Muller Expansion (FPRM)* one denotes a form of GRM in which the literals of a variable are either positive or negative, and cannot stand in both forms in the same expression.

It follows from the preceding definitions that the FPRM class is properly included in the GRM class.

Definition 2.3. A Reed-Muller expansion (also called *Positive Polarity Reed-Muller* form) is a GRM form that consists of only positive literals.

Eqn. (2.1) can be used as the n-variable Generalized Reed-Muller expansion except for every literal $\dot{x_i}$ is used instead of x_i where for each $\dot{x_i}$ in different terms we have $\dot{x_i} = x_i$ or $\dot{x_i} = \bar{x}_i$. If we use T_j to represent product term $b_j\dot{x_1}\dot{x_2}...\dot{x_i}...\dot{x_n}$, then the GRM expansion can be written as $f = T_0 \oplus T_1 \oplus ... \oplus T_j \oplus ... \oplus T_{2^n-1}$.

Definition 2.4. In a GRM expansion, any literal $\dot{x_i}$ can be expressed as $x_i \oplus \delta$ where $\delta \in \{1,0\}$. Here δ is defined as the polarity of literal $\dot{x_i}$.

$$\delta = \begin{cases} 0 \quad if \ \dot{x_i} = x_i & (positive); \\ 1 \quad if \ \dot{x_i} = \bar{x}_i & (negative); \\ - \quad if \dot{x_i} = x_i \ or \ \dot{x_i} = \bar{x}_i & \end{cases}$$

Example 2.1. A GRM form of a 3-variable function is as follows: $f = f_e = x_2 \oplus \bar{x}_2x_3 \oplus \bar{x}_1\bar{x}_2x_3$. This expression can be written as:

$f_e = 0 \oplus 0 \cdot (x_3 \oplus -) \oplus 1 \cdot (x_2 \oplus 0) \oplus 1 \cdot (x_2 \oplus 1)(x_3 \oplus 0) \oplus 0 \cdot (x_1 \oplus -) \oplus 0 \cdot (x_1 \oplus -)(x_3 \oplus -) \oplus 0 \cdot (x_1 \oplus -)(x_2 \oplus -) \oplus 1 \cdot (x_1 \oplus 1)(x_2 \oplus 1)(x_3 \oplus 0)$
$= T_0 \oplus T_1 \oplus T_2 \oplus T_3 \oplus T_4 \oplus T_5 \oplus T_6 \oplus T_7$

where: $T_0 = 0$, $T_1 = 0 \cdot (x_3 \oplus -)$, $T_2 = 1 \cdot (x_2 \oplus 0)$, $T_3 = 1 \cdot (x_2 \oplus 1)(x_3 \oplus 0)$, $T_4 = 0 \cdot (x_1 \oplus -)$, $T_5 = 0 \cdot (x_1 \oplus -)(x_3 \oplus -)$, $T_6 = 0 \cdot (x_1 \oplus -)(x_2 \oplus -)$, $T_7 = 1 \cdot (x_1 \oplus 1)(x_2 \oplus 1)(x_3 \oplus 0)$.

In the above GRM expansion, if the coefficient of a term T_j is "1", for instance, $T_3 = 1 \cdot (x_2 \oplus 1)(x_3 \oplus 0)$, then the polarity of each literal in this term has a fixed value, either positive or negative but cannot take an arbitrary one. If the coefficient of term T_j is "0", for instance, $T_5 = 0 \cdot (x_1 \oplus -)(x_2 \oplus -)$, then the polarity of each literal in this term can be arbitrary, either negative or positive. We use "-" to denote the polarity of a literal when it can take an arbitrary value.

Definition 2.5. In a complete GRM expansion f_e of an n-variable function, there are $2^n - 1$ occurrences of any literal x_i. We define the collection of the $2^n - 1$ polarities of these literals as the *Polarity Set of Variable* x_i. We denote this polarity set of variable x_i as $\delta_{f_e}(\dot{x_i})$.

Definition 2.6. Let T be a product term of n distinct literals $T = \dot{x_1} \ \dot{x_2} \ ... \ \dot{x_i} \ ... \ \dot{x_n}$. The collection of the n polarities of these n literals is defined as the *Polarity Set of Term T*. We use δ_T to represent this polarity set.

Definition 2.7. A collection of the n polarity sets of the n variables of a GRM expansion f_e is defined as *the Polarity Set of Expansion* f_e. This is denoted as $\delta_{f_e}(\dot{x_1}, \ \dot{x_2}, \ ..., \dot{x_n})$. The Polarity of a GRM expansion can also be represented by *the Term-wise Polarity Set.* For an n-variable function, the term-wise polarity set is: $\{\delta_{T_1}, ...\delta_{T_j}, ...\delta_{T_{2^n-1}}\}$.

Please note that for any GRM form, term T_0 is a constant $T_0 = b_0 \in \{0,1\}$, so for T_0 no polarity set exists.

Definition 2.8. The Coefficient Set of a GRM Expansion f_e is a binary vector of length 2^n, denoted by $f_e(b_0, \ b_1, \ ..., b_{2^n-1})$. This vector *Coef_Set* represents a GRM form under the selected polarities of its corresponding subsets of variables in products.

By the definitions above, we see that any GRM form of function f can be identified by a *Polarity Set* (being a binary vector of length $n2^{n-1}$) and its corresponding *Coefficient Set* (being a binary vector of length 2^n). The polarity set can be defined either *term-wise* or *literal-wise.*

Thus, the coefficient set of the expansion in Example 2.1 is: $f_e(00110001)$. The polarity set of $\dot{x_3}$ is $\delta(x_3) = (-0-0)$. The polarity set of $\dot{x_2}$ is $\delta(x_2) = (01-1)$. The polarity set of $\dot{x_1}$ is $\delta(x_1) = (---1)$. The polarity set of term T_7 is $\delta_{T_7} = (110)$. The polarity set of term T_6 is $\delta_{T_6} = (--)$. The literal-wise polarity set of the expansion is $\delta(\dot{x_1}, \dot{x_2}, \dot{x_3}) = (-0-0)(01-1)(---1)$. The term-wise polarity set of the expansion is:

$\{\delta_{T_1}, \ \delta_{T_2}, \ \delta_{T_3}, \ \delta_{T_4}, \ \delta_{T_5}, \ \delta_{T_6}, \ \delta_{T_7}\}$
$= \{ \ (-), \ (0), \ (10), \ (-), \ (--), \ (--), \ (110) \ \}$

Since in a FPRM expansion, the literals of variable x_i can only take either positive or negative polarity but not both, only one polarity bit is necessary to denote the polarities of the 2^{n-1} x_i literals. The polarity set of a FPRM expansion can be denoted as: $\delta_{\dot{x_1}}\delta_{\dot{x_2}}...\delta_{\dot{x_n}}$ or simply as a *Polarity Vector* δ_{f_e}. It will be called *Polarity,* for short.

Example 2.2: The polarity of FPRM expansion $f = f_e = \bar{x_2} \oplus \bar{x_2}x_3 \oplus \bar{x_1}\bar{x_2}x_3$ is written as $\delta_{\bar{x_1}\bar{x_2}x_3}$ or $\delta_{f_e}(\dot{x_1}\dot{x_2}\dot{x_3}) = (110)$.

Definition 2.9: Let f_{e_1} and f_{e_2} be two GRM expansions of the same function. By *Adjacent Polarity GRM Expansions* we understand expansions f_{e_1} and f_{e_2} such that there is only one bit difference between the polarity sets of these two expansions.

Example 2.3: In the following, f'_e and f''_e are two GRM expansions of the same function f:

$f = f'_e = b'_0 \oplus b'_1\bar{x}_3 \oplus b'_2x_2 \oplus b'_3x_2x_3 \oplus b'_4x_1 \oplus b'_5x_1x_3 \oplus b'_6\bar{x}_1x_2 \oplus b'_7\bar{x}_1\bar{x}_2\bar{x}_3$

and

$f = f''_e = b''_0 \oplus b''_1\bar{x}_3 \oplus b''_2x_2 \oplus b''_3x_2x_3 \oplus b''_4\bar{x_1} \oplus b''_5x_1x_3 \oplus b''_6\bar{x}_1x_2 \oplus b''_7\bar{x}_1\bar{x}_2\bar{x}_3$

The polarity sets of f'_e and f''_e are $\delta_{f'_e}(\dot{x_1}, \dot{x_2}, \dot{x_3}) = (0011)(0001)(1001)$ and

$\delta_{f_e''}(\dot{x}_1, \dot{x}_2, \dot{x}_3)$ =(1011)(0001)(1001), respectively. Since there is only one bit difference between $\delta_{f_e'}$ $(\dot{x}_1, \dot{x}_2, \dot{x}_3)$ and $\delta_{f_e''}$ $(\dot{x}_1, \dot{x}_2, \dot{x}_3)$, f_e' and f_e'' are the Adjacent Polarity GRM Expansions.

Combining Definition 2.2 and Definition 2.9 we obtain the definition of the Adjacent Polarity FPRM Expansions.

Definition 2.10: Let f_{e_1} and f_{e_2} be two FPRM expansions of the same function. If there is only one bit difference between the polarity sets of f_{e_1} and f_{e_2}, we define that f_{e_1} and f_{e_2} are the *Adjacent Polarity FPRM Expansions.*

Example 2.4: Similar to Example 2.3, f_e' and f_e'' are two FPRM expansions of the same function f

$f = f_e' = b_0' \oplus b_1'\bar{x}_3 \oplus b_2'x_2 \oplus b_3'x_2\bar{x}_3 \oplus b_4'\bar{x}_1 \oplus b_5'\bar{x}_1\bar{x}_3 \oplus b_6'\bar{x}_1x_2 \oplus b_7'\bar{x}_1x_2\bar{x}_3$

and

$f = f_e'' = b_0'' \oplus b_1''x_3 \oplus b_2''x_2 \oplus b_3''x_2x_3 \oplus b_4''\bar{x}_1 \oplus b_5''\bar{x}_1x_3 \oplus b_6''\bar{x}_1x_2 \oplus b_7''\bar{x}_1x_2x_3$.

The polarity sets of f_e' and f_e'' are $\delta_{f_e'}(\dot{x}_1, \dot{x}_2, \dot{x}_3)$ = (101) and $\delta_{f_e''}(\dot{x}_1, \dot{x}_2, \dot{x}_3)$ = (100), respectively. Since there is only one different bit between $\delta_{f_e'}(\dot{x}_1, \dot{x}_2, \dot{x}_3)$ and $\delta_{f_e''}(\dot{x}_1, \dot{x}_2, \dot{x}_3)$, f_e' and f_e'' are the Adjacent Polarity FPRM Expansions.

Our primary goal of this paper is to minimize the number of terms in GRM expressions. The cost function to be used here is $cost = NT$. NT is the total number of terms in the solution, $NT = \sum_{j=0}^{2^n-1} b_j$ where b_j is the coefficient of term T_j. The secondary goal is literal cost minimization.

III. The Local Minimization Approach

If we calculate the adjacent polarity GRM expansions one-by-one and search through all the GRM forms, the exact minimum GRM is found. In this section identification of a quasi-minimal GRM form based on Term-wise Complementary Expansion Diagrams is presented.

A. Term-wise Complementary Expansion Diagram

Term-wise Complementary Expansion Diagrams are diagrams that relate different term-wise complementary expansions. In this section these diagrams and some related definitions are presented.

Theorem 3.1 Let f_{1e} and f_{2e} be GRM forms of two different n-variable functions f_1 and f_2. Let the coefficient sets of f_{1e} and f_{2e} be $f_{1e}(b_{10}b_{11}b_{12}...b_{12^n-1})$ and $f_{2e}(b_{20}b_{21}b_{22}...b_{22^n-1})$, respectively. If f_{1e} and f_{2e} have the same polarity, then $f_e = f_{1e} \oplus f_{2e}$ is the GRM form of function $f = f_1 \oplus f_2$ with the same polarity as f_{1e} and f_{2e}. The coefficient set of f_e is: $f_e(b_0b_1...b_{2^n-1}) = f_{1e}(b_{10}b_{11}...b_{12^n-1}) \oplus f_{2e}(b_{20}b_{21}...b_{22^n-1})$
$= f_e(b_{10} \oplus b_{20},\ b_{11} \oplus b_{21}, ..., b_{12^n-1} \oplus b_{22^n-1})$.

Furthermore, let $f_{1e}, f_{2e}, ...,$ *and* f_{me} be the GRM forms of m different n-variable functions with the same polarity. Then the GRM expansion of function $f = f_1 \oplus f_2 \oplus ... \oplus f_m$ can be obtained by $f_e(b_0b_1...b_{2^n-1}) = f_{1e}(b_{10}b_{11}...b_{12^n-1}) \oplus f_{2e}(b_{20}b_{21}...b_{22^n-1}) \oplus ... \oplus f_{me}(b_{m0}b_{m1}...b_{m2^n-1})$

where $b_0 = b_{10} \oplus b_{20} \oplus ... \oplus b_{m0}$, $b_1 = b_{11} \oplus b_{21} \oplus ... \oplus b_{m1}$, $b_2 = b_{12} \oplus b_{22} \oplus ... \oplus b_{m2}$,....., $b_{2^n-1} = b_{12^n-1} \oplus b_{22^n-1} \oplus ... \oplus b_{m2^n-1}$

Example 3.1. Let f_1 and f_2 be two 3-variable functions in GRM forms: $f_1 = f_{1e} = 1 \oplus x_3 \oplus x_1 \oplus \bar{x}_1x_2 \oplus x_1\bar{x}_2x_3$, $f_2 = f_{2e} = x_2 \oplus x_1 \oplus \bar{x}_1x_3 \oplus \bar{x}_1x_2$. The polarity sets and coefficients sets of f_{1e} and f_{2e} are $\delta_{f_{1e}}(\dot{x}_1\dot{x}_2\dot{x}_3) = (0-10)(--01)(0--0)$, $\delta_{f_{2e}}(\dot{x}_1\dot{x}_2\dot{x}_3) = (011-)(0-0-)(- -0-)$ and $f_{1e}(11001011)$, $f_{2e}(00101110)$, respectively. Since in the polarity set a "-" stands for either a "1" or a "0", the common polarity set of the above two expansions is $\delta_{f_e}(\dot{x}_1\dot{x}_2\dot{x}_3) = (0110)(0-01)(0-00)$.

From Theorem 3.1 we obtain the GRM expansion of function $f = f_1 \oplus f_2$ with the polarity of $\delta_{f_e}(\dot{x}_1\dot{x}_2\ \dot{x}_3) = (0110)(0-01)(0-00)$. Hence, $f_e(b_0b_1b_2b_3b_4b_5b_6b_7) = f_{1e}(11001011) \oplus f_{2e}(00101110) = f_e(11100101)$.

Definition 3.1: Let T be a product term of n distinct literals $T = \dot{x}_1\dot{x}_2...\dot{x}_i\ ...\dot{x}_n$. Applying $\dot{x} = 1 \oplus \overline{\dot{x}}$ to all the literals in T, the following FPRM expansion results:

$$\begin{aligned} T &= \dot{x}_1\dot{x}_2...\dot{x}_i...\dot{x}_n \\ &= 1 \oplus \overline{\dot{x}_n} \oplus \overline{\dot{x}_{n-1}} \oplus \overline{\dot{x}_n}\ \overline{\dot{x}_{n-1}} \ \oplus ... \oplus \overline{\dot{x}_1}\ \overline{\dot{x}_2}\ ...\ \overline{\dot{x}_n}. \end{aligned}$$

The polarities of all the variables in the above expansion are the complements of the corresponding variables in product T. This expansion is defined as the *Complementary Expansion of Product T*. The complementary expansion of product T can be readily obtained because it is a FPRM expansion with all the coefficients equal to "1".

Example 3.2. The complementary expansion of product term $x_1x_2\bar{x}_3$ is:

$1 \oplus x_3 \oplus \bar{x}_2 \oplus \bar{x}_2x_3 \oplus \bar{x}_1 \oplus \bar{x}_1x_3 \oplus \bar{x}_1\bar{x}_2 \oplus \bar{x}_1\bar{x}_2x_3$.

Definition 3.2: Let $T = \dot{x}_1\dot{x}_2...\dot{x}_n$ be a term and $k_1,\ k_2,\ ...,\ k_m \in \{1, 2, ..., n\}$. Applying $\dot{x} = 1 \oplus \bar{\dot{x}}$ only to a selected set of literals $\dot{x}_{k_1}$, $\dot{x}_{k_2}$, ..., and $\dot{x}_{k_m}$ in term T results in a FPRM expansion. The polarities of the literals of $\dot{x}_{k_1}$, $\dot{x}_{k_2}$, ..., $\dot{x}_{k_m}$ are complements of the corresponding literals in product T. The polarities of other literals remain the same as those in product T. This expansion is called the *Complementary Expansion of T for Literals* $\dot{x}_{k_1}$, $\dot{x}_{k_2}$, ..., and $\dot{x}_{k_m}$. The complementary expansion of T for literals $\dot{x}_{k_1}$, $\dot{x}_{k_2}$, ..., $\dot{x}_{k_m}$ can be obtained by first calculating the complementary expansion of $\dot{x}_{k_1}\ \dot{x}_{k_2}...\ \dot{x}_{k_m}$ and then multiplying by the other literals.

Example 3.3. The complementary expansion of term $x_1x_2\bar{x}_3$ for $\bar{x}_3$ is: $x_1x_2(1 \oplus x_3) = x_1x_2 \oplus x_1x_2x_3$.

The complementary expansion of term $x_1x_2\bar{x}_3$ for x_2 and $\bar{x}_3$ is: $x_1 \cdot (1 \oplus \bar{x}_2 \oplus x_3 \oplus \bar{x}_2x_3) = x_1 \oplus x_1\bar{x}_2 \oplus x_1x_3 \oplus x_1\bar{x}_2x_3$.

The complementary expansion of T can be illustrated by a subtree where T is the root and the terms in the complementary expansion are represented by the descendents of T. For instance, the complementary expansion

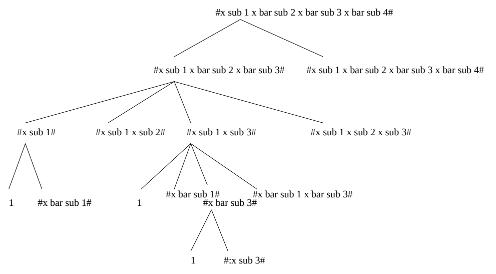

#x sub 1 x bar sub 2 x bar sub 3 x bar sub 4#
#x sub 1 x bar sub 2 x bar sub 3#
#x sub 1 x bar sub 2 x bar sub 3 x bar sub 4#
#x sub 1#
#x sub 1 x sub 2#
#x sub 1 x sub 3#
#x sub 1 x sub 2 x sub 3#
1
#x bar sub 1#
1
#x bar sub 1#
#x bar sub 3#
#x bar sub 1 x bar sub 3#
1
#:x sub 3#

Fig. 1. Complementary Expansion

of $T = x_1x_2\bar{x}_3$ for x_2 and $\bar{x}_3$ can be illustrated by the diagram from Fig. 1.

By the complementary expansion one can always transform a product term with any polarity into a FPRM expansion which includes a term $T' = \dot{x_1}\dot{x_2}...\dot{x_n}$ with expected polarities. If we apply the complementary expansion to all the terms of a GRM form and to all the terms of the resulting complementary expansions, then a GRM form with the expected polarity can be obtained.

Example 3.4. Consider a 4-variable function:

$f = 1 \oplus x_3 \oplus x_3x_4 \oplus x_2x_3 \oplus \bar{x}_1 \oplus \bar{x}_1\bar{x}_3 \oplus x_1x_2x_3 \oplus x_1\bar{x}_2\bar{x}_3\bar{x}_4$ $= T_0 \oplus T_1 \oplus T_2 \oplus ... \oplus T_{15}$

where: $T_0 = 1$, $T_1 = 0\cdot(x_4 \oplus -)$, $T_2 = 1\cdot(x_3 \oplus 0)$, $T_3 = 1\cdot(x_3 \oplus 0)(x_4 \oplus 0)$, $T_4 = 0\cdot(x_2 \oplus -)$, $T_5 = 0\cdot(x_2 \oplus -)(x_4 \oplus -)$, $T_6 = 1\cdot(x_2 \oplus 0)(x_3 \oplus 0)$, $T_7 = 0\cdot(x_2 \oplus -)(x_3 \oplus -)(x_4 \oplus -)$, $T_8 = 1\cdot(x_1 \oplus 1)$, $T_9 = 0\cdot(x_1 \oplus -)(x_4 \oplus -)$, $T_{10} = 1\cdot(x_1 \oplus 1)(x_3 \oplus 1)$, $T_{11} = 0\cdot(x_1 \oplus -)(x_3 \oplus -)(x_4 \oplus -)$, $T_{12} = 0\cdot(x_1 \oplus -)(x_2 \oplus -)$, $T_{13} = 0\cdot(x_1 \oplus -)(x_2 \oplus -)(x_4 \oplus -)$, $T_{14} = 1\cdot(x_1 \oplus 0)(x_2 \oplus 0)(x_3 \oplus 0)$, $T_{15} = 1\cdot(x_1 \oplus 0)(x_2 \oplus 1)(x_3 \oplus 1)(x_4 \oplus 1)$.

Thus, this function can be represented using the coefficient set $f(1011001010100011)$ and a term-wise polarity set:

$$\{\delta_{T_1}(-),\ \delta_{T_2}(0),\ \delta_{T_3}(00),\ \delta_{T_4}(-),\ \delta_{T_5}(--),\ \delta_{T_6}(00),\ \delta_{T_7}(---),\ \delta_{T_8}(1),\ \delta_{T_9}(--),\ \delta_{T_{10}}(11),\ \delta_{T_{11}}(---),\ \delta_{T_{12}}(--),\ \delta_{T_{13}}(---),\ \delta_{T_{14}}(000),\ \delta_{T_{15}}(0111)\ \}$$

Our goal is to calculate the coefficient set of function f when one literal in the above expansion changes its polarity.

For example, if literal $\bar{x}_4$ in term $x_1\bar{x}_2\bar{x}_3\bar{x}_4$ changes its polarity, that is, $\delta_{T_{15}}(0111)$ changes to $\delta_{T_{15}}(0110)$. Let $f = f_1 \oplus f_2$ where $f_1 = 1 \oplus x_3 \oplus x_3x_4 \oplus x_2x_3 \oplus \bar{x}_1 \oplus \bar{x}_1\ \bar{x}_3 \oplus x_1x_2x_3$ and $f_2 = x_1\bar{x}_2\bar{x}_3\bar{x}_4$. Obviously, f_1 complies with the expected polarity and f_2 does not. In order to transform f_2 to a GRM form that complies with the expected polarity, we apply the complementary expansion to f_2 recursively and have:

$f_2 = x_1\bar{x}_2\bar{x}_3\bar{x}_4 = x_1\bar{x}_2\bar{x}_3 \oplus x_1\bar{x}_2\bar{x}_3x_4 = (x_1 \oplus x_1x_2 \oplus x_1x_3 \oplus x_1x_2x_3) \oplus x_1\bar{x}_2\bar{x}_3x_4 = ((\bar{x}_1 \oplus 1) \oplus x_1x_2 \oplus (1 \oplus \bar{x}_1 \oplus \bar{x}_3 \oplus \bar{x}_1\bar{x}_3) \oplus x_1x_2x_3) \oplus x_1\bar{x}_2\bar{x}_3x_4 = ((\bar{x}_1 \oplus 1) \oplus x_1x_2 \oplus (1 \oplus \bar{x}_1 \oplus (x_3 \oplus 1) \oplus \bar{x}_1\bar{x}_3) \oplus x_1x_2x_3) \oplus x_1\bar{x}_2\bar{x}_3x_4$
$= 1 \oplus x_3 \oplus \bar{x}_1\bar{x}_3 \oplus x_1x_2 \oplus x_1x_2x_3 \oplus x_1\bar{x}_2\bar{x}_3x_4$

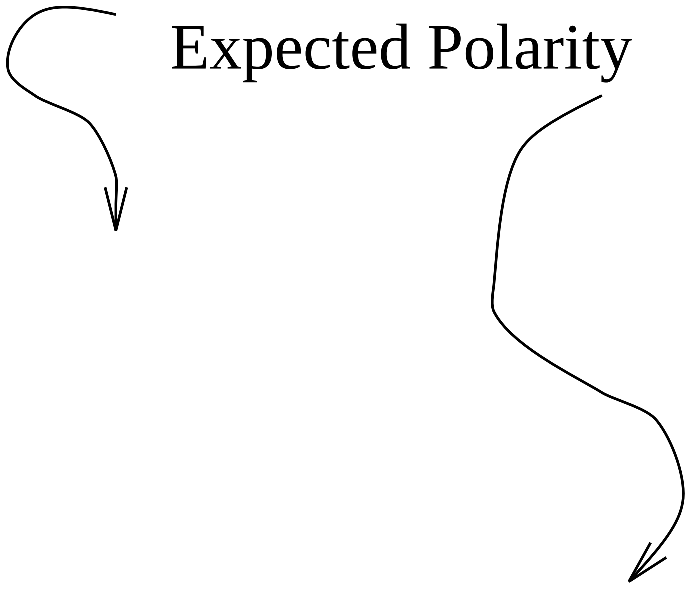


Fig. 2. Decomposition of f_2

The decomposition of f_2 can be demonstrated by a diagram from Fig. 2. Because $f_1 = 1 \oplus x_3 \oplus x_3x_4 \oplus x_2x_3 \oplus \bar{x}_1 \oplus \bar{x}_1\bar{x}_3 \oplus x_1x_2x_3$ we obtain:

$f = f_1 \oplus f_2 = x_3x_4 \oplus x_2x_3 \oplus \bar{x}_1 \oplus x_1x_2 \oplus x_1\bar{x}_2\ \bar{x}_3x_4$. The diagram from Fig. 2 is defined as a *Term-wise Complementary Expansion Diagram.* We can observe that:

- The diagram is a rooted, directed graph with nodes set V containing two types of nodes: terminal nodes and nonterminal nodes.
- All nodes denote terms. Each node has an index as a reference. Here the reference index is the expected polarity set of the term denoted by the node.
- A terminal node denotes a term in which each literal complies with the reference index.
- A non-terminal node denotes a term which has at least one literal that does not comply with the reference index.
- A nonterminal node has at least one descendent that is a terminal node.
- By exoring f_1 and all the terminal nodes in the diagram we obtain the expected adjacent polarity GRM expansion.

Algorithm 3.1
(Calculation of the Adjacent Polarity GRM Expansion)

Let $f_e = T_0 \oplus T_1 \oplus ... \oplus T_j \oplus ... \oplus T_{2^n-1}$ be a GRM expansion of the function f where $T_j = b_j x_1^{e_1} x_2^{e_2} ... x_i^{e_i} ... x_n^{e_n}$. Let the polarity of $\dot{x_i}$ in T_j be inverted without the polarities of other literals changing. The GRM expansion f_e' can be calculated from f_e under the new polarity in the following:

1. Let $f_{2e} := T_j$, $f_{1e} := f_e \oplus f_{2e}$.
2. If $b_j = 0$ then $f_{2e} := 0$, $f'_e := f_e$. Exit.
3. If $b_j \neq 0$, construct the term-wise complementary expansion diagram for f_{2e}. Let $f_{2e} := T_j$ be the root and node i represents a descendent of T_j.
4. Compare node i with the corresponding term in f_{1e}. If the two terms are identical, i is a terminal node. Check other nodes.
5. If two terms are not identical, with literals $\dot{x}_{k_1}$, $\dot{x}_{k_2}$, ..., and $\dot{x}_{k_m}$ differing in their polarities, calculate the complementary expansion of i for $\dot{x}_{k_1}, \dot{x}_{k_2}, ..., \dot{x}_{k_m}$.
6. Expand the diagram until no further expansion is necessary (all descendents are terminal nodes).
7. Exor all terminal nodes results in the GRM form of f_{2e} under the expected polarity.
8. $f'_e := f_{1e} \oplus f_{2e}$ is the GRM expansion of f under the expected polarity.

As we can see, if $b_j = 0$ the algorithm immediately exits, which makes searching thorough many polarities more efficient when the number of terms is low.

By arranging all the GRM polarities of function f in Gray code order and calculating the adjacent polarity expansions one by one, the exact minimum GRM form can be obtained by the exhaustive search of $2^{n2^{n-1}}$ polarity vectors. Hopefully, for most of these codes Algorithm 3.1 immediately exits.

Using k number of term-wise complementary expansion diagrams, one can calculate any GRM form f'_e with expected polarity from any initial GRM form f_e. Here k is the number of terms in f_e that do not comply with the expected polarity.

Example 3.4. Given is a 3-variable function in GRM form

$f = f_e = 1 \oplus x_3 \oplus x_2x_3 \oplus x_1\bar{x}_2 \oplus x_1\bar{x}_2x_3$

and expected polarity as shown in the following:

$f = f'_e = b_0 \oplus b_1\bar{x}_3 \oplus b_2x_2 \oplus b_3x_2x_3 \oplus b_4x_1 \oplus b_5x_1x_3 \oplus b_6\bar{x}_1x_2 \oplus b_7\bar{x}_1\bar{x}_2\bar{x}_3$.

To calculate the coefficients of f'_e, let $f_{1e} = 1 \oplus x_2x_3$, $f_{2e} = x_3$, $f_{3e} = x_1\bar{x}_2$ and $f_{4e} = x_1\bar{x}_2x_3$. As we can see, f_{1e} complies with the expected polarity but f_{2e}, f_{3e} and f_{4e} do not. Construct the term-wise complementary expansion diagrams and calculate the GRM forms for f_{2e}, f_{3e} and f_{4e} separately as in Fig. 3. From the above, we have $f_2 = 1 \oplus \bar{x}_3$, $f_3 = x_2 \oplus x_1 \oplus \bar{x}_1x_2$, and $f_4 = \bar{x}_3 \oplus x_2x_3 \oplus x_1 \oplus \bar{x}_1x_2 \oplus \bar{x}_1\ \bar{x}_2\ \bar{x}_3$. Since $f_1 = 1 \oplus x_2x_3$, thus, $f_T = f_1 \oplus f_2 \oplus f_3 \oplus f_4 = x_2 \oplus \bar{x}_1\bar{x}_2\bar{x}_3$. This is done by Algorithm 3.2.

B. lGRMMIN - The Local Quasi-minimum Algorithm.

This algorithm starts from an initial GRM expansion. This can be a minimized FPRM form or GRMPRM form [23, 24], or any other GRM form. Polarity of a single literal is inverted if this inversion reduces the number of terms. This procedure is carried out iteratively until no further improvement can be achieved.

Given an n-variable function in GRM form:

$f = f_e = T_0 \oplus T_1 \oplus ... \oplus T_j \oplus ... \oplus T_{2^n-1}$

where: $T_j = b_j\dot{x}_1^{e_1}\dot{x}_2^{e_2}...\dot{x}_i^{e_i}...\dot{x}_n^{e_n}$.

The coefficient set of the above GRM form is: $f(b_0b_1...b_j...b_{2^n-1})$. The term-wise polarity set of the above GRM form is: $\{\delta_{T_0}, \delta_{T_1}, ...\delta_{T_j}, ...\delta_{T_{2^n-1}}\}$. The cost of the above expression is: $cost(f_e) = \sum_{j=0}^{2^n-1} b_j$.

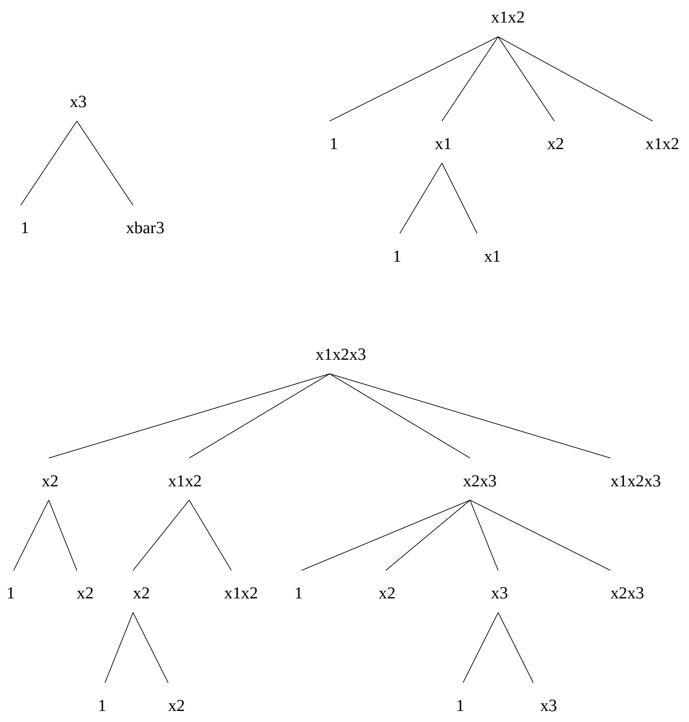


Fig. 3. Term-wise Complementary Expansion Diagram.

If we change the polarity set $\delta_{T_{2^n-1}}$ in Gray code order and keep the other term-wise polarity sets $\delta_{T_0}, \delta_{T_1}, ...\delta_{T_j}, ..., \delta_{T_{2^n-2}}$ unchanged, and calculating all the adjacent polarity expansions, it is possible to find an optimal expansion $f = f'_e$ with the best cost function $cost(f'_e) \leq cost(f_e)$ under this polarity arrangement. The term-wise polarity set of f'_e is $\{\delta_{T_0}, \delta_{T_1}, ...\delta_{T_j}, ...\delta'_{T_{2^n-1}}\}$. If we use f'_e and $\{\delta_{T_0}, \delta_{T_1}, ...\delta_{T_j}, ...\delta'_{T_{2^n-1}}\}$ as the input vector and apply the above procedure to the polarity set $\delta_{T_{2^n-2}}$, again we have another optimal expansion $f = f''_e$ with the cost function $cost(f''_e) \leq cost(f'_e)$. Applying the above procedure to all term-wise polarity sets, finally we obtain the best GRM expression of this pass f_{pass1} with the polarity set $\delta'_{T_0}, \delta'_{T_1}, ..., \delta'_{T_j}, ...\delta'_{T_{2^n-1}}$. If the cost of f_{pass1} is better than the cost of the initial expression, then the above procedure is carried out again. This is done recursively until no further improvement can be achieved. The algorithm is presented as follows.

Algorithm 3.3.
(Quasi-minimization of the GRM expansion)

1. Let f_e be the initial GRM form. Let $j := 2^n - 1$. $f_{min} := f_e$.
2. If $b_j = 0$, no matter how we select the polarities of literals in

T_j, the cost function remains unchanged. $j := j - 1$. Goto step 2.

3. If $b_j \neq 0$. Arrange the term-wise polarity set δ_{T_j} in Gray code order, calculate all GRM forms under each polarity set of term T_j. Select the one which has the best cost function. Let f'_e be the resulting optimal GRM form. Then the term-wise polarity set of f'_e is $\{\delta_{T_0}, \delta_{T_1}, ..., \delta'_{T_j}, ...\delta'_{T_{2^n-1}}\}$
4. Let $f_e := f'_e$, $j := j - 1$. $f_{min} := f'_e$. Goto step 2.
5. After applying the above procedure to all terms we have a GRM form f_{pass1} with the optimal cost of this pass.
6. If $cost(f_{pass1}) < cost(f_{min})$, then $f_{min} := f_{pass1}$, $f_e := f_{pass1}$, goto step 1.
7. If $cost(f_{pass1}) = cost(f_e)$ then exit.

In Algorithm 3.3, when the coefficient of a term is zero, the cost functions of all the GRM expansions under each polarity set of this term will be the same. Thus, the calculation of these expansions is not necessary. Since after each iteration, the number of terms in the optimal expansion is reduced, this optimal expansion is used as the input function to the next iteration. Thus, the computation time is decreased. As an example, the exact minimum GRM expansion for $f = 1 \oplus x_3 \oplus \bar{x}_1\bar{x}_3 \oplus x_1x_2 \oplus x_1x_2x_3 \oplus x_1\bar{x}_2\bar{x}_3x_4$ is $f_e = x_1\bar{x}_2\bar{x}_3\bar{x}_4$. In order to prove exactness of this solution, the exact algorithm has to calculate $2^{n2^{n-1}} = 2^{32}$ GRM expansions. The quasi-minimum algorithm calculates only $2^n = 16$ expansions to find this exact form. Experimental results show that the computation time of the quasi-minimum algorithm depends mainly on the complexity of the number of input variables and the number of terms in the input function.

For an n-variable function, the maximum number of literals in a product term is n. Thus for each iteration at most 2^n GRM expansions are calculated. For an input function with t product terms, the upper bound of expansions calculated is $t \cdot 2^n$. Therefore, the time complexity of Algorithm 3.3 is $O(t \cdot 2^n)$. The algorithm finds a local minimum, and to find a global minimum several good starting points must be generated. This is done by a Genetic Algorithm.

IV. gGRMMIN - The Global Genetic Search

Genetic algorithm is applied with the polarity vectors as *chromosomes,* and the *crossover operation* executed on these polarities. Each polarity is a binary vector, so finding of new polarities consists in creating new vectors from old vectors. Given any number of starting points to the search - the GRM forms, their polarities are found as the first parent generation of the genetic algorithm.

Let us observe, that while in the known applications of Genetic Programming to logic synthesis the crossover operation is performed on functional expressions [9], we apply the crossover to the polarities. Therefore, in contrast to the existing approaches, our technique guarantees that **only functionally correct circuits are created** and their verification is not required nor used as a component of the fitness function. This is because for every created polarity a corresponding Coefficient Set of GRM expression (*Coef_Set*) can be easily calculated as shown in section 3. The only price paid for this improvement is that we have to find coefficients (expressions) for polarities in order to calculate their values of fitness function. The calculation of the fitness function becomes more complex for incompletely specified functions. The fitness function is the term cost and it should be minimized.

Algorithm gGRMMIN.

1. Create the initial *Pool_of_Parents* from polarities of good GRM forms and random polarities. The elements of this set are *GRM_Triplets* = ($Polarity_i$,$Coef\_Set_i$,$Fitness_i$). Initially, only the *Coef_Sets* and *Fitness* values for the initial good GRMs are non-empty.
2. For randomly generated pairs of parent triplets from *Pool_of_Parents* with polarities $Polarity_{j1}$ and $Polarity_{j2}$ create their offspring polarities by the $Crossover(Polarity_{j1},Polarity_{j2})$ operation on parent chromosomes. The lower is the value of the fitness function from the triple, the higher the probability of selecting the polarity as a parent for crossover. Create set *Intermediate_Pool* of all new polarities from crossover.
3. Execute the *Mutation* operation on some randomly selected offspring $Polarity_{child}$ in *Intermediate_Pool*:
   $Polarity_m := Mutation(Polarity_{child})$.
   Add all new polarities to the *Intermediate_Pool.*
4. For each new $Polarity_i$ from the *Intermediate_Pool* calculate its corresponding $Coef\_Set_i$ and $Fitness_i$ (being its term cost). In case of incompletely specified functions this step is done by algorithm iGRMMIN from the next section.
5. Add all the offspring triplets ($Polarity_i$, $Coef\_Set_i$, $Fitness_i$) for polarities from *Intermediate_Pool* to the *Pool_of_Parents.*
6. Remove from *Pool_of_Parents* these GRM-triplets that have the highest fitness function values. In case of a tie, remove those that have the highest values of the literal cost. If there is still not enough triplets to be removed, remove randomly some of those with higher fitness values in order to keep the limited size of the *Pool_of_Parents.*
7. Repeat steps 2 - 6 until the preset time of global optimization expires.

Algorithm $Crossover(Polarity_1, Polarity_2)$.

1. Select a random set of literals x_i that occur often (above a threshold parameter value) in $Coef\_Set(Polarity_1)$ and $Coef\_Set(Polarity_2)$.
2. Find all bits $\{b_j\}$ corresponding to these literals in $Polarity_1$ and $Polarity_2$ of parents.
3. Create two new polarities, $New\_Polarity_i$ and $New\_Polarity_j$ by replacing mutually the values of these bits in the respective term-wise polarity sets of parents, keeping their positions, respectively:
   $$\forall j \; Polarity_1[b_j] \longleftrightarrow Polarity_2[b_j].$$

The *Coef_Set* for the *New_Polarity* is calculated, using Algorithm 3.2, from the polarity of that parent that has the shorter distance to this *New_Polarity.* A multi-output completely specified function with r outputs is represented by a binary vector of single-output $Coef\_Set^i$; $\underline{Coef\_Set} = [\ Coef\_Set^1, Coef\_Set^2, ..., Coef\_Set^r\ ]$, all of the same polarity. It means, when a new polarity, $Polarity_{child}$, is given as an argument, each single-output expression $Coef\_Set^j_{child}$ of the offspring is created by converting the corresponding $Coef\_Set^j_{parent}$ from $Polarity_{parent}$ to $Polarity_{child}$, $j = 1,...,r$. For r-output function the length of Polarity vector is thus $n2^{n-1}$ and the $\underline{Coef\_Set}$ has length $r2^n$. This means, assuming 32-bit word, 318k of memory is necessary to store a single Polarity of 20 variables. The same memory is needed for a $\underline{Coef\_Set}$ of a 10-output, 20-input function.

V. iGRMMIN - Handling of Incompletely Specified Functions.

Few authors, [25, 8, 15, 10, 22] have considered the problem of PPRM minimization for single-output *incompletely specified* functions. However, with an exception of [25], the algorithms are very inefficient for functions that have very many don't cares, since their complexity grows quickly with the number of don't cares. Moreover, all these algorithms cannot be adopted to GRMs, which are quite different from PPRMs.

The algorithm presented below gives good results for multi-output functions with very many don't cares and it scales well with the increase of the numbers of don't cares and output functions. It can be applied to functions represented by sets ON and OFF of either minterms or disjoint cubes.

To calculate the quasi-minimum GRM form for a given polarity, the algorithm creates a table (similar to Quine table) with all GRM coefficients for this polarity as columns and all ON and OFF cubes as rows. The cubes in various output functions are repeated as separate rows, one for each function. The set of rows represents therefore all output functions. ON cubes are marked as active by setting value of flag ON to "1". Cubes with flag ON = "0" are treated as OFF-cubes. Initially the cells of the table are set to "0's". Whenever an active cube matches a coefficient, a "1" is set in a cell on the intersections of this cube's row and this coefficient's column in the table. By the *matching* of a cube and a coefficient (also represented as a cube) we mean a relation when all literals from the coefficient have the same polarity as their corresponding literals in the cube. For instance, coefficient $\bar{b}c$ matches cube $001X1 = \bar{a}\bar{b}ce$. For all columns that have at least one "1" in some of the rows, the cost is calculated. The $Column\_Cost(column_s)$ is defined as:

$Column\_Cost(column_s) = \alpha \cdot (N_0 - N_1) + \frac{1}{N_0+N_1}$, where α is a weighting coefficient. The column C_{best} with the highest $Column\_Cost$ is selected. The coefficient cube $Coef(C_{best})$ corresponding to C_{best} is next exored from these output functions f^j, $j = 1,...,r$, for the rows of which there were symbols "1" in the table, and for which exoring the cube corresponding to this column with function f^j would bring an improvement in function's f^j estimated cost. This is done based on a numbers of "1's" and "0's" in f^j, the literal costs of the cube, and other evaluations. If there are no better groups available, any matching cube is applied. The cost of the cube selected to solution is calculated across all output functions f^i.

The exoring operation converts some of ON-cubes to OFF-cubes, and vice versa. This is done by activating and deactivating ON flag for each cube. Each ON-cube covered by a $Coef(C_{best})$ cube in the header of the selected column C_{best} is converted to an OFF-cube. Each OFF-cube covered by a selected column is converted to an ON-cube (flag ON is set to 1). (In case that disjoint cubes are used instead of minterms in ON and OFF sets the exoring can also split a larger cube to smaller cubes, and set flags ON of these new cubes accordingly, removing the original cube). All cubes selected for output function f^j, $j = 1,...,r$, are triggered for this function. It means, in the first selection, the cube is recorded for this function (by triggering respective bit from 0 to 1), any next selection of the same cube triggers the respective bit in $Coef\_Set$. So that an even number of selections means no selection, and an odd number of selections means a single selection.

For every new ON cube, the contents of table's cells is modified accordingly. The procedure is repeated until no more ON-cubes remain in the table. The *cost* of the *solution_$Coef_Set$* is calculated incrementally with selection of new ON cubes.

Algorithm iGRMMIN is called by algorithm GGRMMIN to find the $Coef\_Set_v$ and $cost_v$ (fitness) of the offspring's polarity $Polarity_v$, and store them together with $Polarity_v$ in the GRM-triplet.

Algorithm iGRMMIN($Polarity_v$, ON, OFF).

1. Create the Table with coefficients of Polarity as columns and all ON cubes of multi-output function $\{f_1, ..., f_r\}$ as rows (repeated for each function in which they stand). Set all cells of the table to zeros.

2. New_ON_Cubes := ON. *solution_$Coef_Set$* := $\emptyset$.

   solution_cost(*solution_$Coef_Set$*) := 0.

   For every new cube from New_ON_Cubes mark with value "1" in Table every intersection of a column that matches this cube.

3. For each column C_i that has at least one "1" calculate $Column\_Cost(C_i)$.

4. Select column C_{best} with the highest value of $Column\_Cost$.

5. Mark for C_{best} those cubes in output functions f^j (marked functions) that have high scores of $Column\_Cost$, and those where there are no other covering groups available for the ON-cubes in a function f^j.

6. For each output function f^j that includes a cube marked in step 5 do:

   $$f^j := f^j \oplus Coef(C_{best}).$$

The exoring creates sets New_ON_Cubes and New_OFF_cubes. Activate and modify sets ON and OFF in the Table accordingly.

7. Update the $solution\_Coef\_Set$ by triggering bit of cube C_{best} in the marked output functions f^j of the $solution\_Coef\_Set$.

   $min\_cost := cost(solution\_Coef\_Set)$

   If this cost is higher than the global minimal cost found until now by gGRMMIN, $global\_min\_cost$ then:

   goto step 2 (with probability κ)

   goto next step with probability (1 - κ)

   Else goto next step. (The probabilistic selection is done to avoid creating too many low quality offspring in the pool, yet to keep some worse chromosomes for further evolution.)

8. If there still exist some cubes in ON-set goto 3.

9. Iterate steps 2 to 8 unless no cost improvement in three iterations.

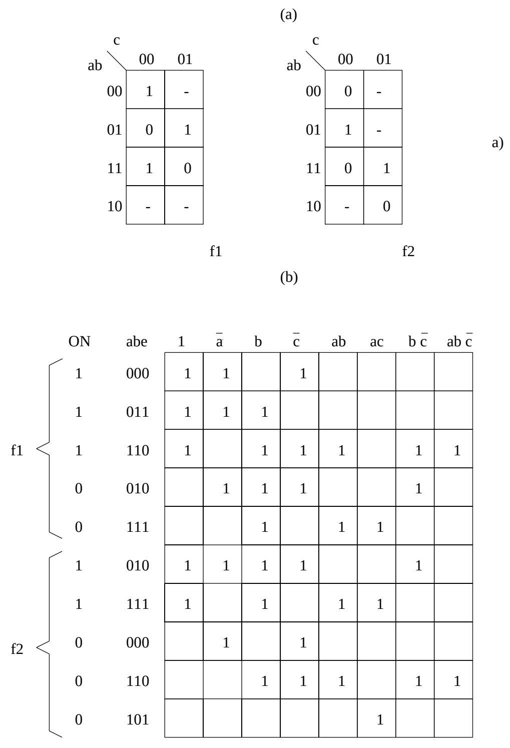
10. Return a GRM-triplet

    $(Polarity_v, solution\_Coef\_Set, min\_cost)$.

Example 5.1. Given is a 2-output function $(f_1(a,b,c),\ f_2(a,b,c))$ from Fig. 4a. Assuming minterm representation of sets ON and OFF and termwise polarity $(1,(\bar{a}),(b),(\bar{c}),(a,b),(a,c),(b,\bar{c}),(a,b,\bar{c}))$, a Table from Fig. 4b is created with coefficients as columns, and minterms of functions f_1 and f_2 as rows. The initial states of flags ON are shown near rows. Assuming $\alpha = 1$, the costs of the coefficients are as follows. For function f_1: cost$(1) = 1\frac{1}{5}$, cost$(\bar{a}) = 1\frac{1}{3}$, cost$(b) = \frac{1}{4}$, cost$(\bar{c}) = 1\frac{1}{3}$, cost$(ab) = \frac{1}{2}$, cost$(ac) = -\frac{1}{2}$, cost$(b\bar{c}) = \frac{1}{2}$, cost$(ab\bar{c}) = 2$. For function f_2: cost$(1) = -\frac{4}{5}$, cost$(\bar{a}) = \frac{1}{2}$, cost$(b) = 1\frac{1}{3}$, cost$(\bar{c}) = -\frac{2}{3}$, cost$(ab) = \frac{1}{2}$, cost$(ac) = \frac{1}{2}$, cost$(b\bar{c}) = \frac{1}{2}$, cost$(ab\bar{c}) = -\frac{1}{2}$. In the first iteration the successive choices are: $ab\bar{c}$ to f_1, $\bar{a}$ to f_1, b to f_2, $ab\bar{c}$ to f_2, $b\bar{c}$ to f_1, $ab\bar{c}$ to f_1 (toggle back to 0). In result, $f_1 = \bar{a} \oplus b\bar{c}$, $f_2 = b \oplus ab\bar{c}$ of total term cost = 4. In the second iteration the choices are: $\bar{a}$ to f_1, $b\bar{c}$ to f_1 and f_2, ab to f_2. In result, $f_1 = \bar{a} \oplus b\bar{c}$, $f_2 = ab \oplus b\bar{c}$ of total term cost = 3. In the third iteration the choices are: $\bar{c}$ to f_1, b to f_1 and f_2, ab to f_1 and f_2, ab to f_2, $ab\bar{c}$ to f_2. In result, $f_1 = \bar{c} \oplus b \oplus ab$, $f_2 = b \oplus ab\bar{c}$ of total term cost = 4.

VI. Pseudo-Code for Program lGRMMIN

Algorithm from section 3 is realized as program lGRMMIN. Before we present the pseudo-code of the lGRMMIN, let us first explain how to obtain the complementary expansion of term T. In section 3.2, it was shown that the complementary expansion of T is a FPRM with all the coefficients equal to "1" and the polarities of variables are the complement of that in T. By constituting the variables in T, we have all the subsets of these variables and each of these subsets represents a term in the complementary expansion. For example, if $T = \dot{a}\,\dot{b}\,\dot{c}$, then the set of constitution is $(1,(\dot{a}),(\dot{b}),(\dot{c}),(\dot{a},\dot{b}),(\dot{a},\dot{c}),(\dot{b},\dot{c}),(\dot{a},\dot{b},\dot{c}))$. In the program the constitution is done by function $consttn()$ which accepts a set of decision variables and return the constitution result: i.e. the descendent nodes.



ab \ c	00	01
00	1	-
01	0	1
11	1	0
10	-	-

ab \ c	00	01
00	0	-
01	1	-
11	0	1
10	-	0

	ON	abe	1	$\bar{a}$	b	$\bar{c}$	ab	ac	b $\bar{c}$	ab $\bar{c}$
f1	1	000	1	1		1				
	1	011	1	1	1					
	1	110	1		1	1	1		1	1
	0	010		1	1	1			1	
	0	111			1		1	1		
f2	1	010	1	1	1	1			1	
	1	111	1		1		1	1		
	0	000		1		1				
	0	110			1	1	1		1	1
	0	101						1		

Fig. 4

Fig. 4. Optimization of GRM for Incomplete Function.

We use two stacks in the construction of the term-wise complementary expansion diagram and computation of the adjacent polarity GRM expansion. $Stack\_A$ is used as a dynamic memory to store the generated descendent nodes, and all the terminal nodes are put in $Stack\_B$. The procedure is:

1. Push the root into $Stack\_A$. This is the term in which the polarity of literal x_i is changed.

2. Pop $Stack\_A$, we have node i. Check i. If i is a terminal node, push i to $Stack\_B$. If i is a nonterminal node, calculate the complementary expansion of i, expand the diagram and push the descendents of i into $Stack\_A$.

3. If $Stack\_A$ is not empty, go to step 2.

4. If *Stack_A* is empty, pop all nodes in *Stack_B* and perform the exor operation with the corresponding terms in f.

The pseudo-code of lGRMMIN program for quasi-minimum GRM follows. The function *check*() accepts a node(polarity array of a term) and then compares with the polarity vector. If they are identical, a NULL is returned. Otherwise a set of decision variables is returned that do not comply with the expected polarities.

```
lGRMMIN()
{
read initial GRM$f_i$ ;
$f_{min} = f_i$;
while(improvement == true){
    for(j=$2^n - 1$; $j \geq 1$; j - -){/*start pass recursion*/
        $f_e = f_{min}$;
        $pv$ = polarity vector;
        $flag$ = number of variables in $T_j$;
          do{
              $k = count()$; /* counting function */
              inverse the $k^{th}$ bit in $pv$;
              push $T_j$ to $Stack\_A$; /* $T_j$ is the root */
              while($Stack\_A \neq$ NULL){
                  $node_i$ = pop($Stack\_A$);
                  $decision\_vars = check(node_i)$;
                  if($decision\_vars \neq$ NULL){
                      $descnnts = contttn(decision\_vars)$;
                      push $descnnts$ to $Stack\_A$;
                  }
                  else
                      push $node_i$ to $Stack\_B$;
                            /* $node_i$ is a terminal node */
                  }
                  while($Stack\_B \neq$ NULL)
                    $f_e = \bar{f}_e \oplus$ pop($Stack\_B$);
                 if(cost($f_e$) < cost($f_{min}$))
                            $f_{min} = f_e$;
             }while($k \neq flag + 1$);
          }
      output $f_{min}$;
   }
}
```

VII. The Experimental Results

So far, only program lGRMMIN has been fully implemented and tested. Over 110 single-output functions generated from MCNC benchmarks have been tested. Some results of qGRMMIN program for single output examples are listed in Table 1. In the table, *inp* means the number of input variables, *Time* is the CPU time in seconds (it does not take into account the time of CGRMIN). The columns give the numbers of terms in the output files of our minimizer qGRMMIN (gGRM), as well as FPRM minimizer CGRMIN [19] (CGR) and ESOP minimizer EXORCISM [21] (XRC), respectively. Interestingly, for the functions tested, the qGRMMIN was able to find better results on some functions than EXORCISM. Obviously, since the GRMs are a sub-family of the ESOP, exact GRMs are not better than exact ESOPs. However, since exact GRMs are close to exact ESOPs, and both qGRMMIN and EXORCISM are approximate programs, in some cases EXORCISM, the top ESOP minimizer, was clearly outperformed. For the 110 functions overall, qGRMMIN had 11 results better than EXORCISM, and for 52 functions the two minimizers gave the same results. The differences were sometimes quite significant. For instance, for function 5xp1 qGRMMIN gave solution with 20 terms while EXORCISM needed 32. This shows that EXORCISM can be much improved, at least on single-output functions.

The results were also never worse than those from exact CGRMIN, the exact minimum for FPRM. Comparing to the two level AND/OR minimizer ESPRESSO [1], for 110 functions overall, while ESPRESSO generated 1336 terms, this number for qGRMMIN was found to be 918.

The scatter plot of the number of input variables versus program qGRMMIN execution time, and the scatter plot of the number of terms in the input function versus qGRMMIN execution time clearly demonstrate that the execution time of the quasi-minimum GRM minimization algorithm depends more on the number of input terms than on the number of input variables [24].

TABLE I
BINARY INPUT EXAMPLES

	inp	qGRM	CGR	XRC	Time
5x01	7	8	12	6	0.63
5x1	7	23	61	33	12.45
5x2	7	15	30	10	10.2
5x3	7	12	19	9	1.12
5xp1	7	20	61	32	12.67
bw14	5	4	8	4	0.05
bw15	5	2	6	3	0.03
bw23	5	5	9	5	0.08
bw6	5	3	6	4	0.05
bw7	5	5	10	5	0.05
9sym	9	129	173	58	15.3
cm152a	11	8	27	8	4.93
con1	7	10	17	9	0.32
f501	8	17	31	11	12.63
misex20	6	19	62	7	77.43
misex21	6	9	16	8	0.22
misex47	11	7	18	4	116.22
misex48	6	8	16	8	0.33
misex62	10	17	50	7	63.37
misex63	10	20	84	7	78.12
misex64	10	6	28	4	28.25
rd53	5	12	20	15	0.35
sao21	10	13	36	10	40.78
sao22	10	19	52	13	70.43
sao23	10	16	47	12	133.67
sao24	10	24	55	11	77.08

VIII. Conclusions

The paper introduced new concepts of the adjacent polarity GRM expansion and term-wise complementary expansion diagrams. These concepts have been next used in algorithms for the calculation of GRM expansions. The exact minimum GRM form can be obtained by an exhaustive search through all GRM forms. Heuristic minimization algorithms have been designed to decrease the time complexity of the exact algorithm. The computation times of the quasi-minimum algorithms depend mainly on the complexity of the input functions, and not on the number of input variables, contrary to most algorithms of this type. Thus, larger problems can be solved. Moreover, to our knowledge, for the first time the Genetic Algorithm has been applied to a problem in logic synthesis represented with polarities and not with functional representations, which makes it much more efficient. A new formulation of minimizing multi-output GRM forms for incompletely specified functions has been introduced in this paper. This approach can be adopted to PPRM, FPRM, KRM and other AND/EXOR canonical forms, and also to any *Linearly Independent (LI) Forms* [14] (formerly called the *Orthogonal Forms*), including the AND/OR/EXOR forms [12, 13]. The testing results of qGRMIN prove that our ESOP minimizer, EXORCISM, which is superior to other ESOP minimizers, can be still much improved, at least on some functions. Although preliminary experimental results of qGRMMIN are encouraging, we still need to complete the implementation.

Acknowledgements

The authors are very grateful to Prof. Tsutomu Sasao for sending helpful papers and for advices.

References

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

[2] M. Cohn, "Inconsistent canonical forms of switching functions," *IRE Trans. Electron. Comput.,* Vol.11, pp. 284-285, 1962.

[3] L. Csanky, M. A. Perkowski, and I. Schaefer, "Canonical restricted mixed-polarity exclusive-OR sums of products and the efficient algorithm for their minimisation," *IEE Proceedings-E,* Vol. 140, No. 1, pp. 69 - 77, Jan. 1993.

[4] M. Davio, J.P. Deschamps, and A. Thayse, *Discrete and Switching Functions,* New York: McGraw-Hill, 1978.

[5] D. Debnath, and T. Sasao, "GRMIN: a heuristic simplification algorithm for generalized Reed-Muller expressions," *in print.*

[6] D.H. Green, "Reed-Muller canonical forms with mixed polarity and their manipulations," *IEE Proc.,* vol. 137, part E, pp. 103-113, Jan. 1990.

[7] D.H. Green, "Families of Reed-Muller canonical forms," *Int. J. Electron.* Vol. 70, pp. 259-280, 1991.

[8] D.H. Green, "Reed-Muller expansions of incompletely specified functions," *IEE Proc. Part E.* Vol. 134, No. 5, pp. 228 - 236, 1987.

[9] J. R. Koza, *Genetic Programming,* The MIT Press, 1992.

[10] L. McKenzie, A.E.A. Almaini, J.F. Miller, and P. Thompson, "Optimization of Reed-Muller Logic Functions," *Intern. J. of Electronics,* Vol. 75, No. 3, pp. 451-466, Sept. 1993.

[11] M.A. Perkowski, L. Csanky, A. Sarabi, and I. Schaefer, "Fast minimization of mixed-polarity AND/EXOR canonical networks," in *Proc. of IEEE ICCD, Cambridge,* MA, pp. 33-35, 1992.

[12] M.A. Perkowski, "A fundamental theorem for EXOR circuits," in *Proc. of IFIP Workshop on Appl. of RM Expansion in Circuit Design,* Hamburg, 1993.

[13] M.A. Perkowski, A. Sarabi, and F.R. Beyl, "XOR canonical forms of switching functions," in *Proc. of IFIP Workshop on Appl. of RM Expansion in Circuit Design,* Hamburg, pp. 147-153, 1993.

[14] M.A. Perkowski, A. Sarabi, and F.R. Beyl, "Fundamental Theorems and Families of Forms for Binary and Multiple-Valued Linearly Independent Logics," *in print*, this Workshop, 1995.

[15] M.W. Riege, and Ph.W. Besslich, "Low-complexity synthesis of incompletely specified multiple-output Mod-2 Sums," *IEE Proc., Part E*, Vol. 139, No. 4, pp. 355-362, Jul. 1992.

[16] T. Sasao, "Easily testable realizations for generalized Reed-Muller expressions." *Proc. IEEE Third Asian Test Symposium*, pp. 157-162, Nov. 1994.

[17] T. Sasao, and D. Debnath, "An exact minimization algorithm for generalized Reed-Muller expressions," *Proc. IEEE Asia-Pacific Conference on Circuits and Systems,* pp. 460 - 465, Dec. 1994.

[18] T. Sasao, and M.A. Perkowski, *EXOR Logic Synthesis*, Kluwer Academic Publishers, *unpublished.*

[19] A. Sarabi, and M.A. Perkowski, " Fast exact and quasi-minimal minimization of highly testable fixed-polarity AND / EXOR canonical networks, " in *Proc. of 29th ACM/IEEE DAC*, pp. 30-35, 1992.

[20] A. Sarabi, and M.A. Perkowski, "Design for testability properties of AND/EXOR networks," in *Proc. of IFIP Workshop on Appl. of RM Expansion in Circuit Design,* Hamburg, 1993.

[21] N. Song, and M.A. Perkowski, "EXORCISM-MV-2: minimization of exclusive sum of products expressions for multiple-valued input incompletely specified functions," *Proc. ISMVL '93*, pp. 132-137, Sacramento, CA, May 24-27, 1993.

[22] D. Varma, and E.A. Trachtenberg, "Computation of Reed-Muller Expansions of Incompletely Specified Boolean Functions from Reduced Representations," *IEE Proc. Part E*, Vol. 138, No. 2, pp. 85 - 92, Mar. 1991.

[23] X. Zeng, H. Wu, M.A. Perkowski, and A. Sarabi, "A new efficient algorithm for finding exact minimal Generalized Partially-Mixed-Polarity Reed-Muller Expansion", *in print*, this Workshop, 1995.

[24] X. Zeng, *Minimization of Generalized Reed-Muller Expansion and Its Sub-classes,* M.S. Thesis, Dept. Electr. Engn., PSU, Dec. 1994.

[25] Z. Zilic, and Z. Vranesic, "A Multiple-Valued Reed-Muller Transform for Incompletely Specified Functions," *unpublished*, 1995.