設計自動化 72－20
(1994. 10. 28)

一般化リード・マラー論理式の最小化について

笹尾 勤　デブナス デバトシュ
九州工業大学情報工学部電子情報工学科
〒820 飯塚市大字川津 680-4

あらまし 一般化リード・マラー論理式 (GRM) とは, 正極性リード・マラー論理式 (PPRM) において各リテラルの極性の反転を許したものである. n 変数の GRM は, 高々$2^{n2^{n-1}}$個存在し最小 GRM は, そのうち積項数が最小のものである. 本論文では, GRM の幾つかの性質を示し, BDD を用いた GRM の最小化法を示す. また, 7 つの論理式 (FPRM, KRO, PSDRM, PSDKRO, GRM, ESOP, SOP) に対して 5 変数以下の全ての NP 代表関数を最小化した場合の積項数の表を示す. この結果, GRM は, SOP に比べて積項数が少なくてよいことを示す.

和文キーワード 排他的論理和, 二分決定グラフ, リード・マラー展開, 論理式最小化, 論理回路の複雑度.

On a Minimization of Generalized Reed-Muller Expressions

Tsutomu SASAO and Debatosh DEBNATH
Department of Computer Science and Electronics
Kyushu Institute of Technology
680-4 Kawazu, Iizuka 820, Japan

Abstract A generalized Reed-Muller expression (GRM) is obtained by negating some of the literals in a positive polarity Reed-Muller expression (PPRM). There are at most $2^{n2^{n-1}}$ different GRMs for an n-variable function. A minimum GRM is one with the fewest products. This paper presents some properties and a minimization algorithm for GRMs. The minimization algorithm is based on binary decision diagrams. Up to five variables, all the representative functions of NP-equivalence class were generated, and minimized. A table compare the number of products necessary to represent 5-variable functions for 7 classes of expressions: FPRMs, KROs, PSDRMs, PS-DKROs, GRMs, ESOPs, and SOPs. We also show that GRMs require, on the average, fewer products than SOPs.

英文 key words EXOR, Binary decision diagrams, Reed-Muller expression, Logic minimization, Complexity of logic networks.

1 Introduction

Conventional logic design is based on AND and OR gates. However, exclusive-OR (EXOR) based designs have certain advantages. The first is that arithmetic and telecommunication circuits are efficiently realized with EXOR gates [16]. Examples of such circuits are adders and parity checkers. The second advantage is that the circuits can be made easily testable by using EXOR gates. Various classes exist in AND-EXOR expressions [6, 9, 15]. Among them, positive polarity Reed-Muller expressions (PPRMs) are well known: a PPRM, an exclusive-OR sum-of-products with positive literals, uniquely represents an arbitrary logic function of n variables. Networks based on PPRMs are easily testable [11, 12], but they require more products than ones based on other expressions. Generalized Reed-Muller expressions (GRMs)[4] are generalization of PPRMs. They were studied many years ago [2], but no practical applications have been shown. Recently, we have developed easily testable realizations for GRMs [18]. Because GRMs require many fewer products than PPRMs and have very good testability, the optimization of GRMs have practical importance. As for the optimization of GRMs, only a few papers have been published [3, 10]. This paper presents some properties and an exact minimization algorithm for GRMs. GRM based design is useful in field programmable gate arrays (FPGAs), where ORs and EXORs have the same costs.

2 Definitions and Basic Properties

2.1 PPRM, FPRM, and GRM

Definition 2.1 *An expression for f is said to be* **minimum** *if it has the least number of product terms.*

The following Lemma is the basis of the EXOR-based expansion:

Lemma 2.1 *An arbitrary logic function $f(x_1, x_2, \ldots, x_n)$ can be expanded as*

$$f = \bar{x}_1 f_0 \oplus x_1 f_1, \qquad (2.1)$$

$$f = f_0 \oplus x_1 f_2, \qquad (2.2)$$

$$f = f_1 \oplus \bar{x}_1 f_2, \qquad (2.3)$$

where $f_0 = f(0, x_2, \ldots, x_n)$, $f_1 = f(1, x_2, \ldots, x_n)$, and $f_2 = f_0 \oplus f_1$.

(2.1), (2.2), and (2.3) are called the *Shannon expansion*, the *positive Davio expansion*, and the *negative Davio expansion*, respectively. If we use (2.2) recursively to a function f, then we have the following:

Lemma 2.2
An arbitrary n-variable function $f(x_1, x_2, \ldots, x_n)$ can be represented as

$$f = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus \cdots \oplus a_n x_n \oplus a_{12} x_1 x_2 \oplus a_{13} x_1 x_3 \oplus \cdots \oplus a_{n-1\,n} x_{n-1} x_n \oplus \cdots\cdots\cdots\cdots\cdots \oplus a_{12\cdots n} x_1 x_2 x_3 \cdots x_n. \qquad (2.4)$$

(2.4) is called a *positive polarity Reed-Muller expression* (PPRM). For a given function f, the coefficients $a_0, a_1, a_2, \ldots, a_{12\cdots n}$ are uniquely determined. Thus, the PPRM is a canonical representation. This unique representation is also the minimum. The number of products in (2.4) is at most 2^n, and all the literals are positive (uncomplemented).

In (2.4), for each variable x_i $(i = 1, 2, \ldots, n)$, if we use either a positive literal (x_i) throughout or a negative literal $(\bar{x}_i)$ throughout, then we have a *fixed polarity Reed-Muller expression* (FPRM). For each variable x_i, there are two ways of choosing the polarities: positive (x_i) or negative $(\bar{x}_i)$. Thus, 2^n different set of polarities exist for an n-variable function. For a given function and a given set of polarities, a unique set of coefficients $(a_0, a_1, \ldots, a_{12\cdots n})$ exists. Thus, an FPRM is a canonical representation.

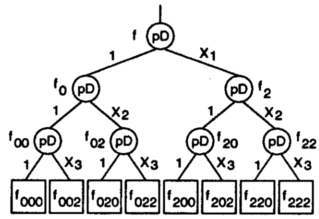

Figure 2.1: Representation of a logic function using positive Davio expansion.

In (2.4), if we can freely choose the polarity for each literal, then we have a *generalized Reed-Muller expression* (GRM). Unlike FPRMs, both x_i and $\bar{x}_i$ can appear in a GRM. There are $n2^{n-1}$ literals in (2.4), so $2^{n2^{n-1}}$ different set of polarities exist for an n-variable function. For a given set of polarities, a unique set of coefficients $(a_0, a_1, \ldots, a_{12\cdots n})$ exists. Thus, a GRM is a canonical representation for a logic function. Properties were analyzed in [3] for GRMs and an exact minimization algorithm was shown. However, this algorithm can simplify functions with only a few input variables. In the next section, we will develop a more efficient minimization algorithm for GRMs.

2.2 KRO, PSDRM, PSDKRO and ESOP

Before studying the minimization method for GRMs, it is convenient to define other classes of AND-EXOR expressions.

Suppose that we are given a three-variable function $f(x_1, x_2, x_3)$. When we expand f by using the positive Davio expansion with respect to x_1, we have

$$f = f_0 \oplus x_1 f_2.$$

Next, when we expand f_0 and f_2 in the similar way with respect to x_2, we have

$$f_0 = f_{00} \oplus x_2 f_{02}, \quad f_2 = f_{20} \oplus x_2 f_{22}.$$

Furthermore, when we use similar expansions with respect to x_3, we have

$$f_{00} = f_{000} \oplus x_3 f_{002}, \quad f_{02} = f_{020} \oplus x_3 f_{022},$$

$$f_{20} = f_{200} \oplus x_3 f_{202}, \quad f_{22} = f_{220} \oplus x_3 f_{222}.$$

The expansion tree in Fig. 2.1 illustrates this process. A path from the root node to a terminal node represents a product of an expression, where a label of an edge shows the literal for the corresponding variable. For example, the path from the root node to f_{000} represents the product $1 \cdot 1 \cdot 1 \cdot f_{000} = f_{000}$, and the path to f_{222} represents $x_1 \cdot x_2 \cdot x_3 \cdot f_{222}$. Thus, the tree in Fig. 2.1 shows the PPRM:

$$f = f_{000} \oplus x_3 f_{002} \oplus x_2 f_{020} \oplus x_2 x_3 f_{022} \oplus x_1 f_{200} \oplus x_1 x_3 f_{202} \oplus x_1 x_2 f_{220} \oplus x_1 x_2 x_3 f_{222}.$$

Each node has a label pD, which shows the positive Davio expansion. In Fig. 2.1, only the positive Davio expansions are used. However, if we use either the positive or the negative Davio expansion for each variable, then we have a more general tree. Such a tree represents an FPRM. If we use either the positive or the negative Davio expansion for each node, then we have a more general tree. Such a tree represents a *pseudo Reed-Muller expression* (PSDRM). For example, in Fig. 2.2, f, f_0, f_{02}, and f_{21} use the positive Davio expansions, while f_2, f_{00}, and f_{22} use the negative Davio expansions. Nodes with label nD denotes the negative Davio expansion. Note that the tree in Fig. 2.2 shows the PSDRM:

$$f = f_{001} \oplus \bar{x}_3 f_{002} \oplus x_2 f_{020} \oplus x_2 x_3 f_{022} \oplus x_1 f_{210} \oplus x_1 x_3 f_{212} \oplus x_1 \bar{x}_2 f_{221} \oplus x_1 \bar{x}_2 \bar{x}_3 f_{222}.$$

There are 7 nodes in the tree, and each node represents either the positive Davio (pD) or the negative Davio (nD) expansion.

In the case of n-variable functions, trees for pseudo Reed-Muller expansions have $2^n - 1$ nodes. Because each

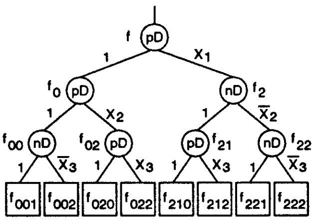

Figure 2.2: Representation of a logic function using pseudo Reed-Muller expansion.

node can represent either the positive or the negative Davio expansion, there are 2^{2^n-1} different PSDRMs. From the definitions, clearly FPRMs are the special class of PSDRMs. Furthermore, PSDRMs are special class of GRMs.

In Fig. 2.1, if we can use either the Shannon, the positive Davio or the negative Davio expansion for each variable, then we have another class of trees. Such a tree represents a *Kronecker expression* (KRO). For each variable, we can select one of the three expansions. Thus, the number of the different KROs for an n-variable function is 3^n.

In Fig. 2.1, if we use either the Shannon, the positive Davio, or the negative Davio expansion for each node, then we have yet another class of trees. Such a tree represents a *pseudo Kronecker expression* (PSDKRO). In the tree of an n-variable function, there are $2^n - 1$ nodes. For each node, we can select one of the three expansions. Thus, the number of the different expansions for n-variable functions is 3^{2^n-1}. By definitions, clearly FPRMs form a special class of KROs, and KROs form a special class of PSDKROs.

Arbitrary product terms combined by EXORs is called an *Exclusive-or Sum-of-Products Expression* (ESOP). The ESOP is the most general AND-EXOR expression.

Example 2.1

1. $x_1x_2x_3 \oplus x_1x_2$ *is a PPRM.*
2. $x_1x_2\bar{x}_3 \oplus x_2\bar{x}_3$ *is an FPRM, but not a PPRM (x_3 has negative literals).*
3. $x_1\bar{x}_2\bar{x}_3 \oplus x_3$ *is a PSDRM, but not an FPRM (x_3 has both positive and negative literals).*
4. $x_1 \oplus x_2 \oplus \bar{x}_1\bar{x}_2$ *is a GRM, but not a PSDRM (it cannot be generated by an expansion tree for a PSDRM).*

From the above arguments, we have the following:

Theorem 2.1 *Suppose that $\mathcal{PPRM}$, $\mathcal{FPRM}$, $\mathcal{PSDRM}$, $\mathcal{KRO}$, $\mathcal{PSDKRO}$, $\mathcal{GRM}$ and $\mathcal{ESOP}$ denote the corresponding set of expressions. Then, the following relations hold:*

$$\mathcal{PPRM} \subset \mathcal{FPRM} \subset \mathcal{PSDRM} \subset \mathcal{GRM} \subset \mathcal{ESOP},$$

$$\mathcal{FPRM} \subset \mathcal{KRO} \subset \mathcal{PSDKRO} \subset \mathcal{ESOP},$$

$$\mathcal{PSDRM} \subset \mathcal{PSDKRO}.$$

Table 2.1 shows the number of 5-variable functions requiring t products for seven classes of expressions, where SOP denotes *sum-of-products expressions*. On the average, GRMs require 6.230 products while SOPs require 7.463 products.

Definition 2.2 *Let $\eta(PPRM : n)$, $\eta(GRM : n)$, and $\eta(SOP : n)$ denote the average number of product needed in the minimal representation of n-variable functions by PPRMs, GRMs, and SOPs, respectively.*

Theorem 2.2

$$\eta(PPRM : n) = 16.000 \cdot 2^{n-5},$$

$$\eta(GRM : n) \leq 6.230 \cdot 2^{n-5} \quad (n \geq 5),$$

$$\eta(SOP : n) \leq 7.463 \cdot 2^{n-5} \quad (n \geq 5).$$

This theorem shows that GRMs require, on the average, less than a half of the products for PPRMs. Table 2.1 also shows that GRMs require fewer products than SOPs. Thus, we have the following:

Conjecture 2.1 $\eta(GRM : n) \leq \eta(SOP : n)$.

3 Some Properties of GRMs

Definition 3.1 *Let p be a product. The set of variables in p is denoted by $V(p) = \{x_i \mid x_i \text{ or } \bar{x}_i \text{ appears in } p\}$.*

Example 3.1 $V(x_1\bar{x}_2\bar{x}_4) = \{x_1, x_2, x_4\}$.

Definition 3.2 *Let G be a GRM. A product p is said to have a* **maximal variable set** *if $V(p) \not\subset V(p')$, for all other products p' in G.*

Example 3.2 *Let a GRM be $G = x_1\bar{x}_2 \oplus \bar{x}_1x_3 \oplus x_1\bar{x}_2x_3 \oplus \bar{x}_4$. Then, $V(x_1\bar{x}_2) = \{x_1, x_2\}$, $V(\bar{x}_1x_3) = \{x_1, x_3\}$, $V(x_1\bar{x}_2x_3) = \{x_1, x_2, x_3\}$, and $V(\bar{x}_4) = \{x_4\}$. Thus, $x_1\bar{x}_2x_3$ and $\bar{x}_4$ have maximal variable sets.*

Definition 3.3 *Let x be a variable and $\alpha \in \{0, 1, 2\}$. x^α is a* **literal** *of x such that*

$$x^\alpha = \begin{cases} \bar{x} & \text{if } \alpha = 0, \\ x & \text{if } \alpha = 1, \\ 1 & \text{if } \alpha = 2. \end{cases}$$

Lemma 3.1 *An arbitrary PPRM can be represented by an expression*

$$F = \sum\nolimits^{\oplus} h(\beta)x_1^{\beta_1}x_2^{\beta_2}\cdots x_n^{\beta_n},$$

where $\beta = (\beta_1, \beta_2, \ldots, \beta_n)$, $\beta_i \in \{1, 2\} (i = 1, 2, \ldots, n)$, and $h(\beta) \in \{0, 1\}$.

Example 3.3 *Consider a PPRM $F = x_1 \oplus x_1x_2 \oplus x_3$. It can be represented as $F = x_1^1x_2^1x_3^2 \oplus x_1^1x_2^2x_3^2 \oplus x_1^2x_2^2x_3^1$.*

Lemma 3.2 *An arbitrary GRM can be represented by an expression*

$$G = \sum\nolimits^{\oplus} g(\alpha)x_1^{\alpha_1}x_2^{\alpha_2}\cdots x_n^{\alpha_n},$$

where $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$, $\alpha_i \in \{0, 1, 2\} (i = 1, 2, \ldots, n)$, and $g(\alpha) \in \{0, 1\}$.

Example 3.4 *Consider a GRM $G = \bar{x}_1 \oplus x_1\bar{x}_2 \oplus \bar{x}_3$. It can be represented as $G = x_1^0x_2^2x_3^2 \oplus x_1^1x_2^0x_3^2 \oplus x_1^2x_2^2x_3^0$.*

Lemma 3.3 *Let the PPRM for f be*

$$F = \sum\nolimits^{\oplus} h(\beta)x_1^{\beta_1}x_2^{\beta_2}\cdots x_n^{\beta_n},$$

where $\beta = (\beta_1, \beta_2, \ldots, \beta_n)$, $\beta_i \in \{1, 2\} (i = 1, 2, \ldots, n)$, and $h(\beta) \in \{0, 1\}$. Also let a GRM for f be

$$G = \sum\nolimits^{\oplus} g(\alpha)x_1^{\alpha_1}x_2^{\alpha_2}\cdots x_n^{\alpha_n},$$

where $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$, $\alpha_i \in \{0, 1, 2\} (i = 1, 2, \ldots, n)$, and $g(\alpha) \in \{0, 1\}$. If F has a product $p = x_1^{\beta_1}x_2^{\beta_2}\cdots x_n^{\beta_n}$ with a maximal variable set, then G has a product $q = x_1^{\alpha_1}x_2^{\alpha_2}\cdots x_n^{\alpha_n}$, where

$$\alpha_i = \begin{cases} 0 \text{ or } 1 & \text{if } \beta_i = 1, \\ 2 & \text{if } \beta_i = 2, \end{cases}$$

and q has the maximal variable set in G.

Corollary 3.1 *If all the products in the PPRM of a function f have a maximal variable set, then a minimum GRM for f contains the same number of products as the PPRM.*

Corollary 3.2 *The PPRM in Corollary 3.1 is also a minimum GRM for f.*

Example 3.5 *Let the PPRM for a function f be $F = x_1 \oplus x_2x_3$. Because both of the products have a maximal variable set, a minimum GRM has two products. Thus, F is also a minimum GRM for f.*

Table 2.1: Number of 5-variable functions requiring t products.

t	FPRM	KRO	PSDRM	PSDKRO	GRM	ESOP	SOP
0	1	1	1	1	1	1	1
1	243	243	243	243	243	243	243
2	6932	24948	24452	24948	24452	24948	20676
3	79820	354780	1232740	1346220	1283820	1351836	818080
4	575930	2508570	28573890	36945666	36127630	39365190	16049780
5	3228162	12029418	274624058	414798570	489868278	545193342	154729080
6	14327120	55321704	1128518304	1525655736	2243146768	2398267764	698983656
7	49694224	187202664	1783419504	1827539820	1494589544	1299295404	1397400512
8	138496600	418029660	931834556	480633264	29183904	11460744	1254064246
9	319912340	804890520	133019772	6301476	677056	7824	571481516
10	587707228	1006381476	12800352	1419120	65600		160200992
11	877839192	1053603288	859480	276048			34140992
12	955078352	544903200	58088	26136			6160176
13	803257168	195821712	1280	0			827120
14	393502216	13630680	480	0			84800
15	130238200	256608	48	0			5312
16	19114960	0	48	48			114
17	1816640	7776					
18	88032	0					
19	3680	0					
20	208	0					
21	48	48					
av	11.566	10.066	6.877	6.541	6.230	6.162	7.463

av : average

Corollary 3.3 *Let p_1 be a product in the PPRM for f which has a maximal variable set. Then*

1. *Any GRM for f contains a product p_2 such that $V(p_2) = V(p_1)$.*
2. *Any GRM for f does not contain a product p_3 such that $V(p_3) \supset V(p_1)$ and $V(p_3) \neq V(p_1)$.*

Example 3.6 *Let the PPRM for f be $F = x_1 \oplus x_2x_3$. GRMs for f are $G_1 = x_1 \oplus x_2x_3$, $G_2 = x_1 \oplus x_3 \oplus \bar{x}_2x_3$, $G_3 = x_1 \oplus x_2 \oplus x_2\bar{x}_3$, $G_4 = \bar{x}_1 \oplus x_2 \oplus x_3 \oplus \bar{x}_2\bar{x}_3$, etc. Note that, in the PPRM for f, the products x_1 and x_2x_3 have maximal variable sets. Thus, all the GRMs for f contain the products with the form $x_1^{b_1}$ and $x_2^{b_2}x_3^{b_3}$. GRMs for f do not contain the products with the form $x_1^{b_1}x_2^{b_2}$, $x_1^{b_1}x_3^{b_3}$, nor $x_1^{b_1}x_2^{b_2}x_3^{b_3}$, where b's are binary constants.*

4 Basic Idea for Minimization

Definition 4.1 *x^a is called a literal of x, where $a \in \{0,1\}$.*

$$x^a = \begin{cases} \bar{x} & \text{if } a = 0, \\ x & \text{if } a = 1. \end{cases}$$

Lemma 4.1 *Let $a \in \{0,1\}$, then $x^a = x \oplus a \oplus 1 = \bar{x} \oplus a$, and*

$$x^a = \begin{cases} \bar{a} & \text{if } x = 0, \\ a & \text{if } x = 1. \end{cases}$$

Definition 4.2 *Let $f(x_1, x_2, \ldots, x_n)$ be a function of n variables. The* **Boolean difference** *of f with respect to x_i is*

$$\frac{df}{dx_i} = f(x_1, x_2, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)$$
$$\oplus f(x_1, x_2, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n).$$

Lemma 4.2 *For an arbitrary function $f(x_1, x_2, \ldots, x_n)$:*

$$\frac{df}{dx_i} = \frac{df}{d\bar{x}_i}, \quad \frac{d^2f}{dx_i dx_j} = \frac{d^2f}{dx_j dx_i}.$$

If g does not depend on x_i, then

$$\frac{dg}{dx_i} = 0, \quad \frac{d(x_i g)}{dx_i} = g.$$

In order to obtain the minimum GRM of a given function, we have to solve a system of logic equations. Such a system is given by

$$f_i(y_1, y_2, \ldots, y_t) = g_i(y_1, y_2, \ldots, y_t),$$

where, $i = 1, 2, \ldots, k$. However, these equations are converted into one equation as follows:

Lemma 4.3 *$f_i = g_i$ holds for all i ($i = 0, \ldots, k$) iff $GR = 1$, where $GR = \bigwedge_{i=0}^{k}(f_i \oplus g_i \oplus 1)$.*

4.1 A Naive Method for Optimization

An arbitrary two-variable function can be represented by a GRM:

$$f(x_1, x_2) = a_{00} \oplus a_{01}x_2^{b_1} \oplus a_{10}x_1^{b_2} \oplus a_{11}x_1^{b_3}x_2^{b_4}, \quad (4.1)$$

where the a's and b's are binary constants. By setting (x_1, x_2) to $(0,0), (0,1), (1,0)$ and $(1,1)$ in (4.1), we have

$$f(0,0) = a_{00} \oplus a_{01}\bar{b}_1 \oplus a_{10}\bar{b}_2 \oplus a_{11}\bar{b}_3\bar{b}_4, \quad (4.2)$$
$$f(0,1) = a_{00} \oplus a_{01}b_1 \oplus a_{10}\bar{b}_2 \oplus a_{11}\bar{b}_3b_4, \quad (4.3)$$
$$f(1,0) = a_{00} \oplus a_{01}\bar{b}_1 \oplus a_{10}b_2 \oplus a_{11}b_3\bar{b}_4, \quad (4.4)$$
$$f(1,1) = a_{00} \oplus a_{01}b_1 \oplus a_{10}b_2 \oplus a_{11}b_3b_4. \quad (4.5)$$

From (4.2)~(4.5) and by Lemma 4.3, we have

$$GR(f) = \psi(0,0) \cdot \psi(0,1) \cdot \psi(1,0) \cdot \psi(1,1) = 1, \quad (4.6)$$

where

$$\psi(0,0) = f(0,0) \oplus a_{00} \oplus a_{01}\bar{b}_1 \oplus a_{10}\bar{b}_2 \oplus a_{11}\bar{b}_3\bar{b}_4 \oplus 1,$$
$$\psi(0,1) = f(0,1) \oplus a_{00} \oplus a_{01}b_1 \oplus a_{10}\bar{b}_2 \oplus a_{11}\bar{b}_3b_4 \oplus 1,$$
$$\psi(1,0) = f(1,0) \oplus a_{00} \oplus a_{01}\bar{b}_1 \oplus a_{10}b_2 \oplus a_{11}b_3\bar{b}_4 \oplus 1,$$
$$\psi(1,1) = f(1,1) \oplus a_{00} \oplus a_{01}b_1 \oplus a_{10}b_2 \oplus a_{11}b_3b_4 \oplus 1.$$

Thus, the assignment of a's and b's that satisfy $GR(f)$ in (4.6) also satisfies (4.1). The minimum GRM is one that has the fewest products, i.e., a GRM with the sum of a's minimum. The number of b's in (4.1) is four. Thus, a minimum GRM can be found out of $2^4 (= 16)$ different GRMs. However, the expression in (4.6) is very complex, and it is not easy to obtain the minimum solution.

4.2 An Efficient Method of Optimization

This method is more complex than the previous method, but it is more efficient. In (4.1), by obtaining the Boolean difference, and by setting $(x_1, x_2) = (0,0)$, we have

$$\frac{d(df)}{dx_1 dx_2} = a_{11}, \quad \frac{df}{dx_1} = a_{10} \oplus a_{11}\bar{b}_4,$$
$$\frac{df}{dx_2} = a_{01} \oplus a_{11}\bar{b}_3. \quad (4.7)$$

On the other hand, consider the PPRM for the function:

$$f(x_1, x_2) = c_{00} \oplus c_{01}x_2 \oplus c_{10}x_1 \oplus c_{11}x_1x_2. \quad (4.8)$$

By obtaining the Boolean difference of (4.8), and by setting $(x_1, x_2) = (0,0)$, we have

$$\frac{d(df)}{dx_1 dx_2} = c_{11}, \quad \frac{df}{dx_1} = c_{10}, \quad \frac{df}{dx_2} = c_{01}. \quad (4.9)$$

From (4.7) and (4.9), we have

$$c_{11} = a_{11}, \quad c_{10} = a_{10} \oplus a_{11}\bar{b}_4, \quad c_{01} = a_{01} \oplus a_{11}\bar{b}_3. \tag{4.10}$$

In (4.1) and (4.8), by setting $(x_1, x_2) = (0, 0)$, we have

$$c_{00} = a_{00} \oplus a_{01}\bar{b}_1 \oplus a_{10}\bar{b}_2 \oplus a_{11}\bar{b}_3\bar{b}_4 \tag{4.11}$$

From (4.10), (4.11) and Lemma 4.3, we have

$$GR(f) = \phi(0,0) \cdot \phi(0,1) \cdot \phi(1,0) \cdot \phi(1,1) = 1, \tag{4.12}$$

where

$$\phi(0,0) = c_{00} \oplus a_{00} \oplus a_{01}\bar{b}_1 \oplus a_{10}\bar{b}_2 \oplus a_{11}\bar{b}_3\bar{b}_4 \oplus 1,$$
$$\phi(0,1) = c_{01} \oplus a_{01} \oplus a_{11}\bar{b}_3 \oplus 1,$$
$$\phi(1,0) = c_{10} \oplus a_{10} \oplus a_{11}\bar{b}_4 \oplus 1,$$
$$\phi(1,1) = c_{11} \oplus a_{11} \oplus 1.$$

Note that ϕ's of (4.12) is simpler than ψ's of (4.6): ϕ's contain fewer EXOR and AND operators than ψ's. In Section 5, we will formulate a method to solve $GR(f)$ by generating its binary decision diagrams (BDDs)[1]. In that case, at first it would be necessary to compute the BDDs of all the ψ's of (4.6) or all the ϕ's of (4.12). Because ϕ's are simpler than ψ's, computation of BDDs for $GR(f)$ using (4.12) is more efficient than using (4.6).

4.3 Three-variable case

An arbitrary 3-variable function f can be represented by a GRM:

$$f(x_1, x_2, x_3) = a_{000} \oplus a_{001}x_3^{b_1} \oplus a_{010}x_2^{b_2} \oplus a_{011}x_2^{b_3}x_3^{b_4}$$
$$\oplus\, a_{100}x_1^{b_5} \oplus a_{101}x_1^{b_6}x_3^{b_7} \oplus a_{110}x_1^{b_8}x_2^{b_9}$$
$$\oplus\, a_{111}x_1^{b_{10}}x_2^{b_{11}}x_3^{b_{12}}, \tag{4.13}$$

where a's and b's are binary constants.

On the other hand, the PPRM for the function f is:

$$f(x_1, x_2, x_3) = c_{000} \oplus c_{001}x_3 \oplus c_{010}x_2 \oplus c_{100}x_1$$
$$\oplus\, c_{011}x_2x_3 \oplus c_{101}x_1x_3 \oplus c_{110}x_1x_2$$
$$\oplus\, c_{111}x_1x_2x_3, \tag{4.14}$$

where c's are binary constants.

Similarly to the two-variable case, we have

$$GR(f) = \phi(1,1,1) \cdot \phi(1,1,0) \cdot \phi(1,0,1) \cdot \phi(0,1,1)$$
$$\cdot\, \phi(1,0,0) \cdot \phi(0,1,0) \cdot \phi(0,0,1) \cdot \phi(0,0,0)$$
$$= 1, \tag{4.15}$$

where

$$\phi(1,1,1) = c_{111} \oplus a_{111} \oplus 1,$$
$$\phi(1,1,0) = c_{110} \oplus a_{110} \oplus a_{111}\bar{b}_{12} \oplus 1,$$
$$\phi(1,0,1) = c_{101} \oplus a_{101} \oplus a_{111}\bar{b}_{11} \oplus 1,$$
$$\phi(0,1,1) = c_{011} \oplus a_{011} \oplus a_{111}\bar{b}_{10} \oplus 1,$$
$$\phi(1,0,0) = c_{100} \oplus a_{100} \oplus a_{101}\bar{b}_7 \oplus a_{110}\bar{b}_9 \oplus a_{111}\bar{b}_{11}\bar{b}_{12} \oplus 1,$$
$$\phi(0,1,0) = c_{010} \oplus a_{010} \oplus a_{110}\bar{b}_4 \oplus a_{011}\bar{b}_9 \oplus a_{111}\bar{b}_{10}\bar{b}_{12} \oplus 1,$$
$$\phi(0,0,1) = c_{001} \oplus a_{001} \oplus a_{011}\bar{b}_5 \oplus a_{101}\bar{b}_7 \oplus a_{111}\bar{b}_{10}\bar{b}_{11} \oplus 1,$$
$$\phi(0,0,0) = c_{000} \oplus a_{000} \oplus a_{001}\bar{b}_1 \oplus a_{010}\bar{b}_2 \oplus a_{011}\bar{b}_3\bar{b}_4$$
$$\oplus\, a_{100}\bar{b}_5 \oplus a_{101}\bar{b}_6\bar{b}_7 \oplus a_{110}\bar{b}_8\bar{b}_9 \oplus a_{111}\bar{b}_{10}\bar{b}_{11}\bar{b}_{12} \oplus 1.$$

4.4 n-variable case

Similar to the two and three-variable cases, we can make 2^n different equations, and can get the expression for $GR(f)$ for an n-variable function. An assignment of a's and b's that satisfies $GR(f)$ corresponds to a GRM for the given function f. For n-variable case, a GRM similar to (4.1) contain 2^n a's and $n2^{n-1}$ b's, thus, the total number of variables in $GR(f)$ is $2^n + n2^{n-1} = (n+2)2^{n-1}$. The minimum GRM corresponds to the assignment of a's and b's that makes the sum of a's minimum.

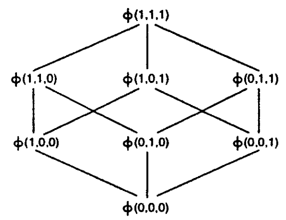

Figure 5.1: Computation of $GR(f)$.

5 An Algorithm using BDDs

5.1 Minimization using BDDs

Consider the binary decision diagram (BDD) for $GR(f)$, where the edges for uncomplemented a's have distance one, and other edges (i.e., edges for $\bar{a}$'s, b's and $\bar{b}$'s) have distance zero. Then, each path in the BDD from the root node to the terminal 1 corresponds to an assignment of a's and b's satisfying (4.15). And the shortest path from the root node to the terminal 1 corresponds to a minimum GRM. Theoretically, it is possible to obtain a minimum GRM by using the BDD for $GR(f)$. However, a naive method using the BDD often requires excessive memory and computation time. To reduce the size of the BDDs and the computation time, we use various techniques, which will be shown in Section 5.2-5.5.

5.2 Threshold Function

$GR(f)$ represents all possible GRMs for a given function. However, we need only one minimum GRM. Suppose that we have a near minimal GRM for f, and let t_0 be the number of products in it. Then, we only need to find a GRM for f that has less than t_0 products. If such a GRM does not exist, then the near minimal GRM is also an exact minimum GRM for f.

Definition 5.1 *Let $a_i \in \{0,1\}$ for $i = 0, 1, \ldots, 2^n - 1$, and t be a positive integer. A function*

$$TH(a_0, a_1, \ldots, a_{2^n-1} : t) = \begin{cases} 1 & \textit{if } \sum_{i=0}^{2^n-1} a_i < t, \\ 0 & \textit{otherwise.} \end{cases}$$

$TH(a_0, a_1, \ldots, a_{2^n-1} : t)$ is used to represent the set of GRMs with less than t products.

5.3 Computation of GR(f)

A naive method for computing $GR(f)$ requires excessive memory and computation time. In the case of three-variable functions, we use the following method:

$$\phi(1,1,1) \leftarrow \phi(1,1,1) \cdot TH(a_0, a_1, \ldots, a_7 : t),$$
$$\phi(0,1,1) \leftarrow \phi(0,1,1) \cdot \phi(1,1,1),$$
$$\phi(1,0,1) \leftarrow \phi(1,0,1) \cdot \phi(1,1,1),$$
$$\phi(1,1,0) \leftarrow \phi(1,1,0) \cdot \phi(1,1,1),$$
$$\phi(0,0,1) \leftarrow \phi(0,0,1) \cdot \phi(0,1,1) \cdot \phi(1,0,1),$$
$$\phi(0,1,0) \leftarrow \phi(0,1,0) \cdot \phi(0,1,1) \cdot \phi(1,1,0),$$
$$\phi(1,0,0) \leftarrow \phi(1,0,0) \cdot \phi(1,0,1) \cdot \phi(1,1,0),$$
$$\phi(0,0,0) \leftarrow \phi(0,0,0) \cdot \phi(0,0,1) \cdot \phi(0,1,0) \cdot \phi(1,0,0),$$
$$GR(f) \leftarrow \phi(0,0,0).$$

This method drastically reduces the computation time as well as the memory requirement for generating the BDD for $GR(f)$. Fig. 5.1 illustrates this multiplication method. Extension to the n-variable case is straightforward.

5.4 Variable Ordering in the BDDs

The ordering of the variables in the BDDs influences the memory requirement as well as computation time. In the case of $GR(f)$ for three-variable functions (4.15), we use the following ordering: $a_{111} < b_{10} < b_{11} < b_{12} < a_{110} < b_8 < b_9 < a_{101} < b_6 < b_7 < a_{011} < b_3 < b_4 < a_{100} < b_5 < a_{010} < b_2 < a_{001} < b_1 < a_{000}$, where a_{111} is the nearest to the root node. Extension to the n-variable case is straightforward.

5.5 Maximal Variable Sets

Corollary 3.3 shows the products that will never appear in the GRMs for a given function. In generating BDDs for $GR(f)$, we do not use the variables (a's and b's) corresponding to such products.

5.6 Minimization Algorithms

Algorithm 5.1 *(Exact Minimum GRM)*

1. *Obtain a near minimal GRM by Algorithm 5.2, and let t_0 be the number of products.*
2. *Construct the BDD for $TH(a_0, a_1, \ldots, a_{2^n-1} : t_0)$.*
3. *Construct the BDD for $TH(a_0, a_1, \ldots, a_{2^n-1} : t_0) \cdot GR(f)$.*
4. *Find a shortest path to the terminal one for the BDD computed in 3.*
5. *Obtain the corresponding GRM.*

Algorithm 5.2 *(Near minimal GRM)*

1. *Obtain a minimal PSDRM for f by the similar algorithm to [15], and let t_1 be the number of products.*
2. *Decompose the function f into 2^{n-k} sub-functions as*

$$f(x_1, x_2, \ldots, x_n) = \sum\nolimits_{\oplus} x_1^{\beta_1} x_2^{\beta_2} \cdots x_{n-k}^{\beta_{n-k}} g(x_{n-k+1}, x_{n-k+2}, \ldots, x_n : \beta_1, \beta_2, \ldots, \beta_{n-k}), \quad (5.1)$$

where β's are 1 or 2, and $g(x_{n-k+1}, x_{n-k+2}, \ldots, x_n : \beta_1, \beta_2, \ldots, \beta_{n-k})$ represents a k-variable sub-function. For each sub-function, obtain the GRM by using the table of exact minimum GRMs of k-variables ($k = 3, 4$ or 5). Let t_2 be the number of products in (5.1).

3. *Obtain the GRM with $\min\{t_1, t_2\}$ products.*

6 Experimental Results

We developed a minimization program, which extensively uses BDDs. The computation time of the program depends on the size of the BDDs, and the size of the BDDs depends on the number of inputs and the number of the products in the near minimal GRMs obtained from Algorithm 5.2. The program can minimize GRMs for all the functions up to five variables, and some functions with more inputs. We generated all the 1,228,158 representative functions for NP-equivalence classes of five or fewer variables, and minimized each function. On the average, a five-variable function could be minimized in 25 seconds by an Hewlett Packard Model 715/50 workstation with 64 megabytes main memory. We also developed minimization programs for FPRMs, KROs, PSDRMs, PSDKROs, ESOPs and SOPs. Tables 2.1 shows the number of five-variable functions requiring t products, respectively. For five-variable functions, on the average, GRMs require 6.230 products while SOPs require 7.463 products. Thus, we verified that Conjecture 2.1 is correct for $n = 4$ and 5.

7 Conclusion and Comments

In this paper, we presented seven classes of AND-EXOR expressions: PPRM, FPRM, KRO, PSDRM, PSDKRO, GRM and ESOP. Among these classes, GRMs have easily testable realizations and require fewer products than other classes of the expressions except for ESOPs. Thus, the optimization problem for GRMs is important, especially in FPGAs, where the EXORs have the same costs as ORs. We presented some properties of GRMs, and showed an exact minimization algorithm. The minimization program can minimize GRMs for all the functions up to five variables, and some functions with more inputs. We have completed the table of minimum GRMs with up to five-variable functions. Thus, the minimum GRMs with up to five variables can be found in a table look-up method. The table of minimum GRMs is also useful in a heuristic optimization program for GRMs with six or more inputs.

Acknowledgments

This work was supported in part by a Grant in Aid for Scientific Research of the Ministry of Education, Science and Culture of Japan. Prof. J. T. Butler carefully reviewed the manuscript. Prof. N. Koda provided the table of representative functions of five variables.

References

[1] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. Comput.*, vol. C-35, No. 8, pp. 677-691, Aug. 1986.

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

[3] L. Csanky, M. A. Perkowski, and I. Schäfer, "Canonical restricted mixed-polarity exclusive-OR sums of products and the efficient algorithm for their minimization," *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*, McGraw-Hill International, New York, 1978.

[5] H. Fujiwara, *Logic Testing and Design for Testability*, The MIT Press, Cambridge, 1985.

[6] D. Green, *Modern Logic Design*, Addison-Wesley Publishing Company, Wokingham, England, 1986.

[7] A. Mukhopadhyay and G. Schmitz, "Minimization of Exclusive OR and logical Equivalence of switching circuits," *IEEE Trans. Comput.*, vol. C-19, pp. 132-140, Feb. 1970.

[8] G. Papakonstantinou, "Minimization of modulo-2 sum of products," *IEEE Trans. Comput.*, vol. C-28, pp. 163-167, Feb. 1979.

[9] M. A. Perkowski, "The generalized orthonormal expansions of functions with multiple-valued input and some of its applications," *Proc. ISMVL-92*, pp. 442-450, May, 1992.

[10] M. A. Perkowski, L. Csanky, A. Sarabi, and I. Schäfer, "Fast minimization of mixed-polarity AND-XOR canonical networks," *Proc. ICCD-92*, pp. 33-36, Oct. 1992.

[11] S. M. Reddy, "Easily testable realizations for logic functions," *IEEE Trans. Comput.*, vol. C-21, No. 11, pp. 1183-1188, Nov. 1972.

[12] K. K. Saluja and S. M. Reddy, "Fault detecting test sets for Reed-Muller Canonic Networks," *IEEE Trans. Comput.*, vol. C-24, No. 1, pp. 995-998, Oct. 1975.

[13] A. Sarabi and M. A. Perkowski, "Fast exact and quasi-minimal minimization of highly testable fixed polarity AND/XOR canonical networks," *Proc. DAC-1992*, pp. 30-35, June 1992.

[14] T. Sasao and P. Besslich, "On the complexity of MOD-2 Sum PLA's," *IEEE Trans. Comput.*, vol. 39, No. 2, pp. 262-266, Feb. 1990.

[15] T. Sasao, "AND-EXOR expressions and their optimization," in (Sasao e.d.) *Logic Synthesis and Optimization*, Kluwer Academic Publishers, 1993.

[16] T. Sasao, "EXMIN2: A simplification algorithm for exclusive-OR-Sum-of-products expressions for multiple-valued input two-valued output functions," *IEEE Trans. CAD.*, vol. 12, No. 5, pp. 621-632, May 1993.

[17] T. Sasao, "An exact minimization of AND-EXOR expressions using BDDs," *Proc. IFIP 10.5 Workshop on Applications of the Reed-Muller expansion in Circuit Design*, Sept. 1993.

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