MULTIPLE-VALUED GENERALIZED REED-MULLER FORMS

Ingo Schäfer, Marek A. Perkowski
Department of Electrical Engineering,
Portland State University,
P.O. Box 751,
Portland, OR 97207.
phone (503) 725-3806

Abstract

The paper introduces the concept of canonical Multiple-Valued Generalized Reed-Muller Forms (MIGRM), a direct extension of the well-known Generalized Reed-Muller (GRM) Forms for the logic with multiple-valued inputs. The motivation to study the MIGRMs is the same as for the GRMs: their direct circuit realization as AND-EXOR PLAs with input decoders have excellent testability properties; they have applications to synthesis of other kinds of circuits with EXOR gates, such as the Exclusive Sums of Products (ESOP); they have good image compression abilities and find applications in image processing. Their study is also interesting for better understanding of the structure of canonical expansions of switching functions. For further investigations of such forms and their comparison to other circuit realizations, a computer program to realize the MIGRM transform for Boolean functions has been implemented.

1. Introduction

The concept of a fixed polarity Generalized Reed-Muller (GRM) Form [1,2] of a n-input Boolean function has been studied extensively in the literature. One of the reasons to study such forms is that for each of the 2^n polarities they are canonical, which has several applications in both theory and practice. For instance, when used as an internal representation of data in CAD systems [3,4,5] they allow for more efficient Boolean function processing, useful in various forms of decomposition. They are intensively studied for better understanding of the canonical representations of switching functions [2,6,7]. As is well know, the circuits corresponding to Reed-Muller and GRM forms have excellent design-for-test properties [8-13]. Finally, GRMs have applications in signal coding and image processing [14].

In recent years a logic with multiple-valued inputs has been introduced with applications in state assignment [15], input encoding [16], PLA decomposition [17], synthesis of PLAs with decoders [18,19], multi-level logic synthesis and factorization [20,21]. The spectral approach to analysis and synthesis of switching circuits has been also extended for logic with multiple-valued inputs [22]. Finally, the concept of multiple-valued input ESOP expressions has been recently introduced [23,24] (the forms are canonical while the expressions are not). The canonical and non-canonical Shannon-like tree expansions for multiple-valued input (mv, for short) logic have been also investigated [25].

In this paper the counterpart of GRMs for the logic with multiple-valued inputs will be introduced. We will call such forms Multiple-Valued Input Generalized Reed-Muller Forms (MIGRMs).

One motivation to investigate the MIGRMs is that they are canonical forms, similar to the binary GRMs. Since the binary forms are used for efficient coding of images [14], it is obvious that their multiple-valued input counterparts can not give worse results, since the number of MIGRMs for a given function is much larger than that of the binary GRMs. It is thus more probable to find among them an efficient code for any given image.

A second motivation is that the concept of the AND-EXOR PLA [26], which is used for the realization of ESOPs, can be extended for GRMs. When a GRM is realized in an AND-EXOR PLA, each input is either negated or not, but cannot be in both forms. This decreases the number of columns in the AND plane by 50%. The AND-EXOR PLA with two-input, four-output input decoders [26], which is used for the multiple-valued ESOP forms can be used for the MIGRMs as well. Moreover, decoders with only three outputs are used, which decreases the number of columns in the AND plane by 25% with respect to the multiple-valued AND-EXOR PLA. The number of terms is usually larger in a GRM and a MIGRM than in an ESOP or multiple-valued input ESOP of the same function. However, for some functions the GRM and MIGRM realizations have smaller total areas or input counts.

CH3009-8/91/0000/0040/$01.00 © 1991 IEEE

The third motivation, and perhaps the most important because it has a direct practical importance is, that the excellent testability properties which have been proven for the strict Reed-Muller forms [8] are extentable also for the GRMs and MIGRMs [27].

The existance of new Programmable Devices such as Xilinx LCA 3000, 1020 series from Actel, or LHS501 from Signetics, gives motivation to study circuits with EXOR and other complex gates. For instance in Xilinx LCA 3000 devices every Boolean function of five variables has the same cost and speed. Using EXORs is then reasonable, since they are more powerful than the inclusive gates. It has been proven that the circuits with EXOR gates have lower worst-case complexity than the circuits which use only inclusive gates [28,29].

Circuits with high percentage of EXOR gates lead to area saving solutions for linear Boolean functions, which have several applications in coding and test generation. They are also advantageous in parity circuits, coders, arithmetic circuits, and others. Circuits with high percentage of EXOR gates and input decoders, which realize the multiple-valued ESOPs and their derivatives such as the MIGRMs and Multiple-Valued RM Trees [25], have good testability properties like the GRMs, but have less gates. Therefore, they can be a good design trade-off.

With respect to the above arguments the compiler/optimizer and minimization algorithms for the implementation of multiple-valued programmable logic arrays [18,24,28,30,31] have been created. [32] and [33] discuss efficient minimization algorithms for binary Exclusive Sum of Products (ESOP) expressions which are realized using AND-EXOR PLAs. In [19,26] minimization algorithms for Multiple-Valued Input Binary Output Exclusive Sum of product (MIESOP) expressions have been shown. They correspond to the AND-EXOR PLAs with input decoders. It was stated in [26] that in most cases the AND-EXOR PLAs with input decoders require fewer products than the AND-OR PLAs with full input decoders. Since efficient algorithms still do not exist for the general ESOPs, an attempt is made at efficient logic minimization algorithms for special cases of ESOPs, such as the Restricted and Inconsistent Forms [34,35].

In this paper the research on MIGRMs for completely specified multiple-valued input binary output functions will be shown both theoretically and practically. We introduce a general concept of a *pattern–matching* method to perform the transformation. This method was also used for other spectral transformations of Boolean functions [22]. In section 2 the theory for the MIGRM form and the RMIGRM form, a restriction to it, will be presented. In section 3 the algorithms for the RMIGRM will be described. In section 4 the algorithm for the RMIGRM transform is given. Finally, in section 5 the multiple-valued input, multi-output SOP (sum of products) function of a 2 bit adder is used to illustrate the transformation to a RMIGRM form.

2. Canonical Multiple-Valued Input, Binary Output Generalized Reed-Muller Forms

Canonical forms for multiple-valued input, multiple-valued output functions were already proposed [36,37]. The m-Reed-Muller canonical (m-RMC) forms [36] are obtained from the truth vector or the SOP representation and the generalization of the Boolean difference to multiple-valued logic. Our approach is to generate a multiple-valued input, binary output MIGRM expansion that makes use of spectral methods similar to the ones introduced in [22].

The concept used to change the polarity of a singular multiple-valued literal is given by the following Theorem.

Theorem 1

Any multiple-valued variable $X_i^{S_i}$ with the set of truth values $S \subseteq P_i = \{0, 1, .., p_i\text{-}1\}$ can be represented by p_i multiple-valued variables $V_i^{T_i}$ with the set of truth values $T_i \subseteq P_i$, where the p_i vectors $T_{i,r}$ form the row vectors of an orthogonal $p_i \times p_i$ matrix PM_i.

Proof:

One property of an orthogonal $m \times m$ matrix O with the elements $o_{ij} \in (0,1)$ is, that any vector $U(u_0, u_1, ..., u_{m-1})$ with $u_j \in (0,1)$ can be represented by a superposition (performing of a bit-by-bit EXOR operation) of row vectors O_i of the matrix O (m is an arbitrary natural number). Thus, any set S of truth values $S \subseteq P = \{0,1,..,p\text{-}1\}$ of a mv-literal X^S can be represented by a superposition of the values from the orthogonal set of p truth values $T \subseteq P = \{0, 1, .., p\text{-}1\}$, where T_r is a row vector of the orthogonal $p \times p$ matrix PM.

Example 1 : To illustrate Theorem 1, all possible sets of truth values $S \subseteq P = \{0,1,2\}$ of a three-valued literal X^S are calculated from the chosen 3×3 orthogonal matrix shown in Figure 1.

$$PM = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix}$$

Figure 1. Example of a 3×3 orthogonal matrix.

In the following Table I the calculation of all possible sets of truth values S by exoring the rows of the orthogonal matrix PM (Figure 1) is shown, where the rows are denoted T_1, T_2 and T_3.

Definition 1

The matrix PM_i introduced in Theorem 1 is called the

polarity matrix or for short *polarity* of a multiple-valued variable (mv-variable) X_i. The row vectors T_r of this matrix are the binary representations of the *polarity literals* $V_i^{T_{i,r}}$.

TABLE I
ALL POSSIBLE SUPERPOSITIONS OF THE POLARITY LITERALS

truth values S	binary code	superposition
{2}	001	$T_1 \oplus T_2$
{1}	010	T_2
{1,2}	011	T_1
{0}	100	T_3
{0,2}	101	$T_1 \oplus T_2 \oplus T_3$
{0,1}	110	$T_2 \oplus T_3$
{0,1,2}	111	$T_1 \oplus T_3$

For the representation of the polarity literals T_r non-orthogonal matrices can also be used. Thus, Theorem 1 can be generalized to the following Lemma.

Lemma

Instead of the orthogonal matrix PM defined in Theorem 1 any non-orthogonal matrix can be used. A set of vectors T_r can be used for the polarity literals V^{T_r}, if by exoring of those literals every possible set $S \subseteq P = \{0,1,..,p-1\}$ of a mv-variable X^s can be generated. Thus, there is more than one way to create an mv-literal X_i^S out of the polarity literals given by the non-orthogonal matrix. (This result is no longer a canonical form.)

Table II summarizes the notation presented in the above Definition 1 and Theorem 1:

TABLE II
THE NOTATION FOR THE MIGRM

$X_1^{S_1}$	$X_2^{S_2}$...	$X_i^{S_i}$...	$X_n^{S_n}$
$P_1=(0,1,..,p_1-1)$		$P_i=(0,1,..,p_i-1)$	$P_n=(0,1,..,p_n-1)$
$V_1^1...V_1^r...V_1^{P_1}$	...	...	$V_n^1...V_n^r...V_n^{P_n}$
PM_1		PM_i	PM_n

In the first row of Table II the multiple-valued literals $X_i^{S_i}$ of a function $F(X_1, X_2,...,X_n)$ are shown. Below the sets of truth values P_i for those literals are given. Next the corresponding polarity literals V_i^r are shown, where the r stands for the values represented by the $T_{i,r}$ vector of the matrix PM_i.

Finally, Figure 3 illustrates two polarity matrices PM_i consisting of the value vectors $T_{i,r}$.

$$PM_1 = \begin{bmatrix} T_{11} \\ .... \\ T_{1r} \\ .... \\ T_{1p_n} \end{bmatrix} \qquad PM_n = \begin{bmatrix} T_{n1} \\ .... \\ T_{nr} \\ .... \\ T_{np_n} \end{bmatrix}$$

Figure 3. Description of polarity matrices.

Example 2 : The two variable mv-function $F(X_1,X_2) = X_1^{023} X_2^{01}$ is used to show how to calculate the MIGRM form for the polarity shown in Figure 4. The variable X_2^{01} can be represented by the superposition of the polarity literals $V_2^{T_{2,1}} \oplus V_2^{T_{2,3}}$, which is abbreviated by $V_2^1 \oplus V_2^3$.

$$A_1 = \begin{bmatrix} 1111 \\ 0101 \\ 0011 \\ 0111 \end{bmatrix} = \begin{bmatrix} T_{11} \\ T_{12} \\ T_{13} \\ T_{14} \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 111 \\ 100 \\ 001 \end{bmatrix} = \begin{bmatrix} T_{21} \\ T_{22} \\ T_{23} \end{bmatrix}$$

Figure 4. Polarity matrices.

The natural method to perform the transformation seems to be an exor-term multiplication:

$$X_1^{023} X_2^{01} = (V_1^1 \oplus V_1^3 \oplus V_1^4)(V_2^1 \oplus V_2^3)$$
$$= (1 \oplus V_1^3 \oplus V_1^4)(1 \oplus V_2^3)$$
$$= 1 \oplus V_1^3 \oplus V_1^4 \oplus V_2^3 \oplus V_1^3 V_2^3 \oplus V_1^4 V_2^3$$

The multiplication method realized as above is, however, in this form computationally inefficient.

It will be now be shown how to describe the MIGRM as a spectrum M and apply the pattern matching method between the indices of the spectral coefficients and a multiple-valued term to calculate the final MIGRM from the polarity representation of the literals, $X_i^{S_i}$. It will be shown, that the pattern matching can be used instead of the product multiplication approach from Example 2. We extend now the Reed-Muller form for Boolean functions to the MIGRM form.

Definition 2

The Multiple-valued Input binary output Generalized Reed-Muller (MIGRM) form of a function $F(X_1,X_2,...,X_n)$, where X_i, i=1,2,...,n are the mv-literals, is defined as

follows:

$F(X_1,X_2,...,X_n) =$

$$a_0 \oplus a_1 V_1^1 \oplus ... \oplus a_r V_1^r \oplus ... \oplus a_{p_1} V_1^{p_1}$$
$$\oplus a_{p_1+1} V_2^2 \oplus ... \oplus a_{p_1+\cdots+p_n} V_n^{p_n} \oplus$$
$$a_{u_1} V_1^1 V_2^2 \oplus a_{u_1+1} V_1^1 V_3^1 \oplus ... \oplus a_{u_2-1} V_{n-1}^{p-1} V_n^{p_n} \oplus$$
$$.....$$
$$a_{u_n} V_1^1 ... V_n^1 \oplus ... \oplus a_{u_{n-1}-1} V_1^{p_1} ... V_n^{p_n}$$

where a_{u_i} denotes the first coefficients of the first term of the i+1 order.

Because of having a total number of $\prod_{i=0}^{n} (p_i + 1)$ terms, and a complex notation, the above formula in its general formulation may be difficult to understand. It will be, therefore further illustrated with a spectral description and an complete example in section 6.

Definition 3

The MIGRM defined above can be represented by a spectrum M, where the index of a coefficient $M_{V_i^r}$ corresponds to the indices of the polarity literals V_i^r which form the terms in the formula of Definition 2.

Because of the complexity of the multiple-valued spectrum M, only a simple example of a spectrum is shown below.

Example 3 : The MIGRM of the mv-function $F(X_1, X_2) = X_1^{023} X_2^{01}$ (see Example 2) can be represented in a different way by its spectrum M. Figure 5 gives the standard trivial functions [22] of the above term for the polarity from Figure 4.

For a comparison of the product $X_1^{023} X_2^{01}$ with the standard trivial functions in Figure 5, the Karnaugh map of this product is shown in Figure 6.

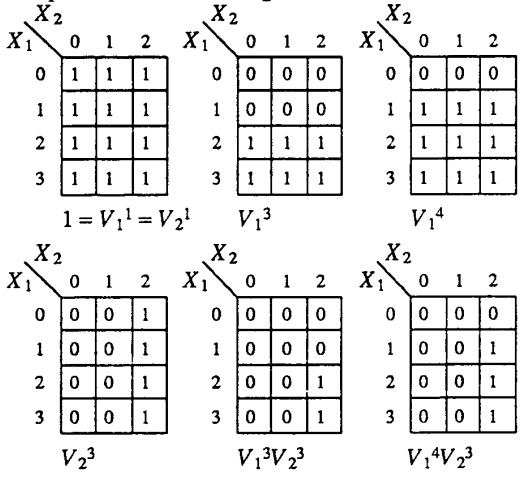

Figure 5. Some standard trivial functions for the polarities of variables X_1 and X_2 specified in Figure 4.

X_1 \ X_2	0	1	2
0	1	1	0
1	0	0	0
2	1	1	0
3	1	1	0

Figure 6. Karnaugh map of function $F(X_1, X_2) = X_1^{023} X_2^{01}$ from Examples 2 and 3.

The new expression can be now represented in the form of spectral coefficients (see Table III), where the indices correspond to the standard trivial functions. The same spectrum as in Example 2 has been obtained:

$$X_1^{023} X_2^{01} = 1 \oplus V_1^3 \oplus V_1^4 \oplus V_2^3 \oplus V_1^3 V_2^3 \oplus V_1^4 V_2^3.$$

The reader may wish to verify this form by exoring corresponding maps from Fig. 5 to obtain the map from Fig. 6. The algorithm to calculate this form will be given in section 5.

The first rows of Table VII give all possible MIGRM terms for the above example. In the second rows a '1' indicates, that the term represented by the index of the spectral coefficient in the same column, is present in the MIGRM form.

TABLE III
THE SPECTRUM OF THE FUNCTION(X_1, X_2)
FROM EXAMPLES 2 AND 3

M_0	$M_{V_1^1}$	$M_{V_1^2}$	$M_{V_1^3}$	$M_{V_1^4}$	$M_{V_2^1}$	$M_{V_2^2}$	$M_{V_2^3}$
1	0	0	1	1	0	0	1

$M_{V_1^1V_2^1}$	$M_{V_1^1V_2^2}$	$M_{V_1^1V_2^3}$	$M_{V_1^2V_2^1}$	$M_{V_1^2V_2^2}$	$M_{V_1^2V_2^3}$
0	0	0	0	0	0

$M_{V_1^3V_2^1}$	$M_{V_1^3V_2^2}$	$M_{V_1^3V_2^3}$	$M_{V_1^4V_2^1}$	$M_{V_1^4V_2^2}$	$M_{V_1^4V_2^3}$
0	0	1	0	0	1

3. Restricted Multiple-Valued Input Generalized Reed-Muller Form (RMIGRM)

Analogously to the binary Reed-Muller expansion where each literal can occur in only one value, we now want to restrict the MIGRM form. The restriction makes use of the availability of the dc term '1', M_0, which is necessary to be able to perform the complete set of operations with AND- and EXOR-gates.

Theorem 2

Any value of a multiple-valued variable $X_i^{S_i}$ with the set of truth values $P_i = \{0,1,..,p_i\text{-}1\}$ can be created by $p_i - 1$ variables V_i^r with the set of truth values $A_i \subseteq P_i$ where one of the truth values P_i is not used in the set of truth values A_i. The set of truth values has to form a orthogonal $(p-1)\times(p-1)$ matrix. To expand this matrix to an orthogonal $p \times p$ matrix PM it is sufficient to add a row corresponding to the dc-literal (literal that contains all values).

Proof:

The proof is similar to the proof of Theorem 1 because again an orthogonal $p \times p$ matrix PM is used. It is trivial, that an orthogonal $(p-1)\times(p-1)$ matrix B can be expanded to an orthogonal $p \times p$ matrix PM by adding a row containing only 1's, and filling up the new positions in the matrix PM with 0's.

The impact of Theorem 2 is that one value of a multiple-valued literal is not used in the orthogonal matrix, except in the dc-literal. One advantage of this restriction is the reduction in the number of the spectral coefficients.

This form will be called the Restricted Multiple-valued Input, multiple-valued output Generalized Reed-Muller (RMIGRM) form.

Example 4 : The mv-function from Example 3 (Table III) can now be represented by the reduced spectrum M shown in Table IV.

TABLE IV
THE SPECTRUM OF THE PRODUCT X_1X_2

product	M_0	$M_{V_1^2}$	$M_{V_1^3}$	$M_{V_1^4}$	$M_{V_2^2}$	$M_{V_2^3}$
$X_1{}^{023}X_2{}^{01}$	1	0	1	1	0	1

$M_{V_1^2V_2^2}$	$M_{V_1^2V_2^3}$	$M_{V_1^3V_2^2}$	$M_{V_1^3V_2^3}$	$M_{V_1^4V_2^2}$	$M_{V_1^4V_2^3}$
0	0	0	1	0	1

The reduced representation in this example has eight coefficients less than in Example 3.

In general the gain of spectral coefficients, that is the ratio of lesser spectral coefficients that have to be generated for the RMIGRM in comparison to the MIGRM, is $\prod_{i=0}^{n} p_i \;/\; \prod_{i=0}^{n} (p_i + 1)$ where n is the number of variables. In the case of all mv variables having the same number of values, the RMIGRM spectrum has $\frac{1}{n}$ times the number of the spectral coefficients of the MIGRM spectrum. Thus, the minimal form obtained by finding the optimal polarity for the RMIGRM spectrum for a given function contains an equal or larger number of terms than the minimal form obtained for a MIGRM spectrum. Hence, from the point of view of circuit minimization, where each term has to be realized with a certain number of gates, the RMIGRM leads to a larger circuit.

4. Algorithm For The RMIGRM Form

The RMIGRM form was chosen for the computer program implementation. Thus, the algorithms shown make use of such forms. An analogous method can be created for the MIGRM form. The method for the generation of the RMIGRM form consists of two basic parts. First, each multiple-valued literal of the mv-function has to be transformed to a polarity specified by on orthogonal matrix. In the second part the terms with transformed literals are used to calculate the final RMIGRM form. The code for the transformation of a multiple-valued literal is chosen in such a way that the second part of the transformation is not dependent on the chosen polarity of the literal. This approach requires the introduction of the concept of *normalized codes*.

4.1 Transformation for one multiple-valued literal.

The basic steps of the algorithm for the transformation of a multiple-valued literal to its representation of polarity literals in the normalized code will be illustrated on an example. Table V (see end of paper) contains the set of transformations of a three-valued literal X_1 for the polarity used in Example 2. The first row gives all the possible combinations of the polarity literals. In the second row the results of the EXOR operation on the respective polarity literals are shown in the binary representation. Let us observe that the first four polarity literals are the rows of matrix PM_1 from Example 2. In the first column all possible literals of the variable X_1 are given. In the respective row for each of these literals the representation by its polarity literals is given, where the binary representation for the value has to be calculated by performing the EXOR operation among the code words that are determined by '1' in the table (i.e. $X_1{}^0 = 1 \oplus V^4$, which is denoted by $1111 \oplus 0111 = 1000$). Finally the last row of the table gives the *normalized code* of the spectral coefficients. It represents the combination of the polarity literals V^r. For instance 0111 means that variables V^2, V^3, and V^4 are taken. This row is just a binary encoding of the first row.

As mentioned above, the code for the RMIGRM representation of one literal is created in such a way, that this *normalized code* can be directly used to perform the transformation of the total mv-function. This can be done by using the same representation of the polarity literals $V^1, V^2, \ldots$ for every chosen polarity. The code determines then of what polarity literal/s the initial literal is composed.

If a literal has to be represented by a combination of the orthogonal subset and the dc coefficient, the new code is calculated by performing a bit-by-bit or-operation between the code of the dc coefficient and the code of the subset combination.

Example 5 : The literal X^0 with the internal representation 1000 is can be represented by $1 \oplus V^4$ which has the normalized code 1001. The new code representing this combination of polarity literals can be derived by the bit-by-bit OR-operation of the code representation of the polarity literals shown in Table VI.

TABLE VI
THE CODE REPRESENTATION FOR THE POLARITY LITERALS

normalized code	polarity literals V^r
1000	$V^1 = 1$
0100	V^2
0010	V^3
0001	V^4

The *normalized code* has a "1" in its bit representation corresponding to the index of the polarity literal V^r.

The procedure implemented to perform the transformation of a multiple-valued literal to the *normalized code* includes the following steps:

1. Generate all possible EXOR combinations of the polarity literals V^r for the later comparison with the original mv-literal (the first row of Table V).
2. Compare the binary representation of the mv-literal (or the inverted binary representation, if the literal contains the not used value of the polarity) with the EXOR combination of step 1. If these two binary representations are equal then assign to the mv-literal its *normalized code* calculated by bit-by-bit ORing of the binary representation of respective V^r (Example 5).

4.2 Transformation of a multiple-valued function.

After the transformation of each original mv-literal, as described above, the whole set of multiple-valued terms of the function has to be changed to the RMIGRM. Again, the basic steps of the algorithm will be explained on an example.

Example 6 : Let us take any function $G(X_1, X_2, X_3)$ where the literal X_1 being three-valued is represented by three polarity literals V_1^1, V_1^2 and V_1^3. The second literal X_2 being four-valued is represented by the polarity literals V_2^1, V_2^2, V_2^3 and V_2^4, the three-valued literal X_3 is represented by V_3^1, V_3^2, and V_3^3. The polarity literals V_1^1, V_2^1, and V_3^1 are dc-literals. The spectral coefficients for a spectrum representing such a function $G(X_1, X_2, X_3)$ are shown in Table VII (see end of paper).

The code shown in Table VII is obtained by generating all possible combinations of polarity literals, except the dc polarity literal, from the distinct original mv-literals. This means that not more than one polarity literal per original mv-literal can occur in the index of a spectral coefficient (i.e. $M_{V_i^x V_i^y}$ can not occur because both polarity literals are from the original mv-literal X_1). As one can observe from the binary representation of the index (second rows in Table VII), called *code* each spectral coefficient consists basically of a combination of three polarity literals, where each polarity literal is taken from a different original mv-literal. However, in the descriptions of the indices of spectral coefficients (first rows of Table VII). Because Table VII has been created for normalized mv-literals, the *normalized code* has to be used to obtain the complete spectrum. In Table VIII the possible combinations of polarity literals of an original mv-literal are listed.

TABLE VIII
THE CODE FOR THE SPECTRUM INDICES

1000	dc-literal or V_i^1
0100	V_i^2
0010	V_i^3
1010	V_i^1 or V_i^3

The final value of the spectral coefficient M_s, where M_s is any of the possible spectral coefficients, determined by a column in Table VII, is obtained by comparing the *normalized codes* of all terms $term_x$ of the Boolean function (x = 0,..., m-1 ; where m is the total number of terms) including variables $X_1, X_2, ..., X_n$ with the normalized code $code_s$ from Table VII. The value M_i for $term_x$ is '1' if the intersection of the $code_s$ and the $term_x$ is not empty. To obtain the final value M_s for the whole function (all its terms), the EXOR operation among all m values M_s has to be performed. The calculation of the values of coefficients M_s is described by the following formula:

$$M_s^0 = 0$$

$$M_s^{x+1} = M_s^x \oplus ((code_s \ \& \ term_x) \neq \phi), \text{ for } x = 0, ..., m\text{-}1$$

where the value of the spectral coefficient is a Boolean variable, $code_s$ and $term_x$ are the binary representation of the respective literals, where $((code_s \ \& \ term_x) \neq \phi)$ has to be true for the intersection of each literal of $code_s$ and $term_x$. By ϕ we denote a vector of zeros.

The final RMIGRM form consists of the terms obtained by replacing the polarity literals in the indices of the spectral coefficients M_s (row cube in Table VII) which have value '1', with their binary representation.

To summarize, the procedure to obtain the RMIGRM of a multi-output Boolean function includes the following stages:

1. Transform all the multiple-valued input literals of the function to their *normalized codes* according to

the chosen polarities.

2. Calculate the RMIGRM spectrum for the *normalized codes*.
3. For each output function take the binary representation for the *polarity* of the non-zero coefficients of the normalized spectrum.
4. Merge the results of each output function to a single table.

5. A Practical Example

To illustrate the complete RMIGRM transformation a real life example is included. The function used is a 2 bit adder (Table IX, where the two four-valued literals X_1 and X_2 represent two binary values each ($X_1 = (x_1, x_2)$, $X_2 = (x_3, x_4)$).

TABLE IX
THE TRUTH TABLE OF THE 2 BIT ADDER

binary function	X_1	X_2	Y
x_1, x_2, x_3, x_4			
0000	1000	1000	000
0001	1000	0100	001
0011	1000	0010	011
0001	1000	0001	001
0100	0100	1000	001
0101	0100	0100	010
0111	0100	0010	100
0110	0100	0001	011
1100	0010	1000	011
1101	0010	0100	100
1111	0010	0010	110
1110	0010	0001	101
1000	0001	1000	010
1001	0001	0100	011
1011	0001	0010	101
1010	0001	0001	100

The mv-literal X_1 is obtained by changing the first two bits of the binary function to a four-valued literal ($X_1{}^0$ = 00, $X_1{}^1$ = 01, $X_1{}^2$ = 11, $X_1{}^3$ = 10). The chosen coding here does not relate to the implementation of input decoders in a real circuit. The second two bits are taken to obtain X_2.

The chosen polarity is given in Table X. To make it easier to follow the steps of the transformation, both literals have the same polarity. The polarity literals V_i^r and the binary representations of their values ($T_{i,r}$) are shown.

TABLE X
THE POLARITY FOR THE 2 BIT ADDER

polarity for X_1	polarity for X_2
$V_1{}^2$: 0110	$V_2{}^2$: 0110
$V_1{}^3$: 0010	$V_2{}^3$: 0010
$V_1{}^4$: 1100	$V_2{}^4$: 1100

As one can observe from Table X, the polarity value of four (0001) was never used among the above polarities. That means that the mv-literals having value four require the dc literal '1'.

Now the binary representations of the literals X_1 and X_2 in Table IX are replaced by their *normalized code*. Because only four different values occur in the literals X_1 and X_2, the *normalized code* for those values is shown in Table XI, where $V^r = V_1{}^r = V_2{}^r$.

TABLE XI
THE NORMALIZED CODES FOR THE FOUR OCCURING VALUES

the binary value of a literal	the normalized code	the cube representation of the normalized code
0001	$1 \oplus V^3 \oplus V^4$	1011
0010	V^3	0010
0100	$V^2 \oplus V^3$	0110
1000	$V^2 \oplus V^3 \oplus V^4$	0111

The literals of variables X_1 and X_2 in Table IX are now substituted with their *normalized codes* from Table XI. For the first four terms of Table IX this is shown in Table XII.

TABLE XII
THE NORMALIZED CODE FOR THE 2 BIT ADDER

old representation		normalized code	
X_1	X_2	X_1	X_2
1000	1000	0111	0111
1000	0100	0111	0110
1000	0010	0111	0010
1000	0001	0111	1011

The complete code obtained in this procedure is now compared with all the indices of the spectral coefficients of the general spectrum for a function $G(X_1, X_2)$, where X_1 and X_2 are four-valued literals. To save space, the following Table XIII (see end of paper) illustrates the calculation of the spectrum only for four terms from Table XII. The table has a "1" as an entry if for each variable the intersection of the index of the term and the index of the coefficient is not empty. For instance, in the cell at the intersection of row 0111 1011 and column 0100 1000 there is "1", since the intersection of those indices is 0100 1000 so that both literals are not empty. The intersection of row 0111 0111 and column 0100 1000 is 0100 0000 so "0" is placed in the corresponding cell.

The next stage is to find the final coefficients for each output function. The first column of Table XIV lists all non-zero spectral coefficients M_s that occur in any of its component functions. These coefficients have been obtained by comparing (as in Table XIII) the binary representations of their indices (listed in the second column of Table XIV) with the *normalized code* obtained according to Tables IX and XI. Finally, the

binary representation for the literals of variables X_1 and X_2 is obtained by replacing the polarity literals of the indices of spectral coefficients from the first column, by their binary representations. According to Table X the conversion is: 1000 to 1111, 1010 to 0010, 1001 to 1100, 1100 to 0110, 0001 to 1100, 0100 to 0110, and 0010 to 0010. In the above procedure, each column (bit) of Y is calculated separately, and next the results are merged for every table's row into a 3-tuple. For instance, this notation means that coefficient $M_{V_1^4}$ is has value "1" in functions 1 and 3, coefficient $M_{V_1^2}$ is has value "1" in function 2, and so on.

TABLE XIV
THE RMIGRM OF THE 2 BIT ADDER

spectral coefficient	index	X_1	X_2	Y
$M_{V_1^4}$	0001 1000	1100	1111	101
$M_{V_1^2}$	0100 1000	0110	1111	010
$M_{V_2^2}$	1000 0100	1111	0110	010
$M_{V_2^4}$	1000 0001	1111	1100	101
$M_{V_2^3}$	1000 0010	1111	0010	110
$M_{V_1^2V_2^2}$	1100 1100	0110	0110	100
$M_{V_1^2V_2^3}$	1100 1010	0110	0010	001
$M_{V_1^3V_2^3}$	1010 1010	0010	0010	110
$M_{V_1^3V_2^2}$	1010 1100	0010	0110	001
$M_{V_1^4V_2^3}$	1001 1010	1100	0010	110
$M_{V_1^4V_2^4}$	1001 1001	1100	1100	001
dc	1000 1000	1111	1111	001

The above method was implemented in our program called GERMS-MV (GEneralized Reed-Muller Synthesizer). The calculation time for this expansion using this program took a 1/10 of a second on a Sun 3/50.

6. Conclusions

The extension of the General Reed-Muller expansion to multiple-valued input, binary output functions has been shown. For this, the concept of code normalization of single multiple-valued literals to perform a final transformation has been developed. The code normalization is used to make the transformation of the complete function independent on the polarity chosen. This simplifies and speeds up the main transformation step to the final RMIGRM form for the transformed single mv-literal. For further investigations on the behavior of such forms the RMIGRM transformation has been implemented on a computer. Because an exhaustive search of all possible polarities to find the minimal RMIGRM solution for any polarity would be too time consuming, therefore further research will be concentrated on avoiding a complete search.

The main reason for the implementation of the RMIGRM algorithm is to use and test it for the generation of the MICRMP form, which like its Boolean counterpart makes use of the MIGRM forms on certain subexpressions of the function [29]. Since circuits which realize the RMIGRM forms are both easily testable or modifiable to very easily testable circuits, further research into them is important. It must be however experimentally found with practical logic benchmarks how much of a circuit cost penalty we pay with respect to the corresponding MIESOPs. The role of input variable pairing [28] must also be investigated, since a good pairing together with a good choice of variable polarities may significantly improve the cost.

7. References

Due to the page limitation the references could not be included. The complete paper with references is available from the authors.

TABLE VII
THE COMPLETE RMIGRM SPECTRUM OF THE FUNCTION
WITH 3-VALUED X_1, 4-VALUED X_2,
AND 3-VALUED X_3

dc	$M_{V_1^2}$	$M_{V_1^3}$	$M_{V_2^2}$	...	$M_{V_3^3}$
100 1000 100	010 1000 100	001 1000 100	100 0100 100	...	100 1000 001

$M_{V_1^2V_2^2}$	$M_{V_1^2V_2^3}$	$M_{V_1^2V_2^4}$	...	$M_{V_2^4V_3^3}$
110 1100 100	110 010 100	110 1001 100	...	100 1001 101

$M_{V_1^2V_2^2V_3^2}$	$M_{V_1^2V_2^2V_3^3}$	...	$M_{V_1^3V_2^4V_3^3}$
110 1100 110	110 1100 101	...	101 1001 101

TABLE V
THE SET OF TRANSFORMATIONS FOR A
MULTIPLE VALUED LITERAL

literal	1	V^2	V^3	V^4	$V^2 \oplus V^3$	$V^2 \oplus V^4$	$V^3 \oplus V^4$	$V^2 \oplus V^3 \oplus V^4$
	1111	0101	0011	0111	0110	0010	0100	0001
X_1^3								1
X_1^2						1		
X_1^{23}			1					
X_1^1							1	
X_1^{13}		1						
X_1^{12}					1			
X_1^{123}				1				
X_1^0	1			1				
X_1^{03}	1				1			
X_1^{02}	1	1						
X_1^{023}	1						1	
X_1^{01}	1		1					
X_1^{013}	1					1		
X_1^{012}	1							1
X_1^{0123}	1							
code	1000	0100	0010	0001	0110	0101	0011	0111

TABLE XIII
THE SPECTRUM FOR TABLE XI

term	dc	$M_{V_1^2}$	$M_{V_1^3}$	$M_{V_1^4}$	$M_{V_2^2}$	$M_{V_2^3}$	$M_{V_2^4}$
	1000 1000	0100 1000	0010 1000	0001 1000	1000 0100	1000 0010	1000 0001
0111 0111	0	0	0	0	0	0	0
0111 0110	0	0	0	0	0	0	0
0111 0010	0	0	0	0	0	0	0
0111 1011	0	1	1	1	0	0	0
	0	1	1	1	0	0	0

term	$M_{V_1^2} M_{V_2^2}$	$M_{V_1^2} M_{V_2^3}$	$M_{V_1^2} M_{V_2^4}$	$M_{V_1^3} M_{V_2^2}$	$M_{V_1^3} M_{V_2^3}$
	0100 0100	0100 0010	0100 0001	0010 0100	0010 0010
0111 0111	1	1	1	1	1
0111 0110	1	1	0	1	1
0111 0010	0	1	0	0	1
0111 1011	0	1	1	0	1
	0	0	0	0	0

term	$M_{V_1^2} M_{V_2^4}$	$M_{V_1^4} M_{V_2^2}$	$M_{V_1^4} M_{V_2^3}$	$M_{V_1^4} M_{V_2^4}$
	0010 0001	0001 0100	0001 0010	0001 0001
0111 0111	1	1	1	1
0111 0110	0	1	1	0
0111 0010	0	0	1	0
0111 1011	1	0	1	1
	0	0	0	0