Multiple valued input generalised Reed–Muller forms

I. Schäfer
M.A. Perkowski

Indexing terms: Multiple valued inputs, Programmable devices, Reed–Muller expansion

Abstract: The concept of canonical multiple valued input generalised Reed–Muller (MIGRM) forms is introduced. The MIGRM is a direct extension of the well known generalised Reed–Muller (GRM) forms to the logic with multiple valued inputs. The concept of the polarity of a GRM form is generalised to the polarity matrix of a MIGRM form. A tabular pattern-matching method is presented for the calculation of a MIGRM form. The MIGRM transform has been implemented for further investigations of such forms and their comparison with other circuit realisations.

1 Introduction

The concept of a fixed-polarity generalised Reed–Muller (GRM) form [1, 2] of an n-input Boolean function has been studied extensively in the literature. One reason for studying such forms is that, for each of the 2^n polarities, they are canonical, which has several applications in both theory and practice. They have been intensively studied for better understanding of the canonical representations of switching functions [2–5]. As is well known, the circuits corresponding to Reed–Muller and GRM forms have excellent design-for-test properties [6–11]. Finally, GRMs have applications in signal coding and image processing [12].

In recent years a logic with multiple valued inputs has been introduced with applications in synthesis of PLAs with decoders and function generators [13–15], multi-level logic synthesis and factorisation [16, 17]. A multiple valued input, binary output function can be readily implemented with currently available digital circuits, while a real multiple valued function cannot. The concept of multiple valued input Exclusive-OR sum-of-products (ESOP) expressions has been presented in Reference 18.

This paper introduces the counterpart of GRMs for the logic with multiple valued inputs and binary outputs. We will call such forms multiple valued input generalised Reed–Muller forms (MIGRMs). The counterpart to the RMs, the restricted multiple valued input generalised Reed–Muller forms (RMIGRMs), have been introduced in Reference 19.

A motivation for the investigation of MIGRM forms is that the concept of the AND-EXOR PLA [20], which is used for the realisation of ESOPs, can be modified to MIGRMs. The AND-EXOR PLA with two-input, four-output input decoders [20] is used for the multiple valued input, binary output exclusive sum-of-products (MIESOP) expansions. Similarly, it can be used for the MIGRMs. Moreover, if the RMIGRM forms [19] are applied, the AND/EXOR needs only two-input, three-output input decoders. Although one gains 25% on one dimension of the PLA, one can lose in the other dimension, as the number of terms is usually larger in a GRM and a RMIGRM or MIGRM than in an ESOP or MIESOP of the same function. But the GRM and MIGRM have the advantage of excellent testability properties which have been proven for the strict Reed–Muller forms [10] and are extendable for the GRMs and MIGRMs.

The existence of new programmable devices such as the Xilinx LCA 3000, the 1020 series from Actel, or the LHS501 from Signetics allows for the direct implementation of the form introduced here. For instance, in Xilinx LCA 3000 devices, every module realisation of a Boolean function of five variables has the same cost and speed. The use of EXORs is then reasonable, as 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 [15]. Moreover, EXOR-based circuits such as MIGRMs, GRMs or ESOPs have much better testability properties than SOPs.

This paper presents the basic research on MIGRMs for completely specified multiple valued input, binary multioutput functions. We introduce the general concept of a tabular pattern matching method to calculate the MIGRM. Section 2 presents the theory of the MIGRM forms, and Section 3 gives the algorithm for the MIGRM transform. Section 4 illustrates the transformation of the multiple valued input, multioutput SOP (sum-of-products) function of a two-bit adder to a MIGRM form.

2 Canonical multiple valued input, binary output generalised Reed–Muller forms

Canonical forms have already been proposed for multiple valued input, multiple valued output functions [21, 22]. The m-Reed–Muller canonical (m-RMC) forms [22] are obtained from the truth vector or the SOP representation and the generalisation of the Boolean difference to multiple valued logic. However, such forms cannot be realised

Paper 9102E (C2), first received 18th July 1991 and in revised form 6th July 1992
The authors are with the Department of Electrical Engineering, Portland State University, PO Box 751, Portland, OR 97207-0751, USA

The work presented in this paper was partially supported by NSF grant MIP-9110772

with available digital circuits. Therefore, the MIGRM introduced here is based on the multiple valued input, binary output logic.

Definition 1: A multiple valued input, completely specified binary output function (*mv* function for short) is a mapping $f(X) = f(X_1, X_2, \ldots, X_n)$: $R_1 \times R_2 \times \cdots \times R_n \rightarrow B$, where X_i is a multiple valued variable that takes the values from the set $R_i = \{0, 1, \ldots, p_i - 1\}$ where p_i is the number of values of the variable, and $B = \{0, 1\}$.

Definition 2: A literal of the multiple valued input variable X_i, denoted by $X_i^{S_i}$, is defined as

$$X_i^{S_i} = \begin{cases} 1 & \text{if } X_i \in S_i \\ 0 & \text{if } X_i \notin S_i \end{cases}$$

A literal of an *mv* variable with a single value will be called a single-valued literal. A product of literals, $X_1^{S_1}$, $X_2^{S_2}, \ldots, X_n^{S_n}$ is referred to as a product term (also called term for short).

The approach presented here to generate a MIGRM makes use of spectral methods similar to the ones introduced in References 19 and 23. First the concept of polarity is introduced for a multiple valued literal. Then it is shown how it is further applied for a product term of multiple valued literals.

2.1 Concept of polarity for a multiple valued variable

The concept of the polarity of a singular multiple valued literal is given by the following theorem.

Theorem 1: Multiple valued literal $X_i^{S_i}$, where $S_i \subseteq R_i$ $[0, 1, \ldots, p_i - 1]$ and $X_i^{S_i} \subseteq R_i$, can be represented by p_i polarity literals $P_i^{T_i}$ with the set of truth values $T_i \subseteq R_i$. The values of the p_i polarity literals form the row vectors of the orthogonal $p_i \times p_i$ polarity matrix P_i.

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

A form according to theorem 1 is canonic because the polarity matrix P is orthogonal.

Example 1: To illustrate theorem 1, all possible sets of truth values $S \subseteq R = \{0, 1, 2\}$ of a three-valued literal X^S are calculated from the chosen 3×3 orthogonal matrix shown in Fig. 1. In Table 1 the calculations of all possible sets of truth values S by exoring the rows of the orthogonal matrix P (Fig. 1) are given, where the rows are denoted by T_1, T_2 and T_3.

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

Fig. 1 *Example of a 3 × 3 orthogonal matrix*

Table 1 : All possible superpositions of 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	$T_1 \oplus T_3$

Definition 3: The matrix P_i introduced in theorem 1 is called the polarity matrix or, for short, the polarity of a multiple valued variable X_i. The rth row vector T_i^r of the matrix P_i is the binary representation of the polarity literals $P_i^{T_i^r}$ (P_i^r for short).

For the representation of the polarity literals P_i^r, nonorthogonal matrices can also be used. Thus, theorem 1 can be generalised to the following lemma.

Lemma 1: Instead of the orthogonal matrix P, defined in theorem 1, any set of vectors T^r can be used for the polarity literals P^r if, by exoring of those literals, every possible set $S \subseteq R = \{0, 1, \ldots, p - 1\}$ of a 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 nonorthogonal matrix. AND-EXOR expressions created with such polarity literals are no longer canonical forms.

The MIESOP expressions [18, 20] are examples of expressions generated by polarity literals as described in lemma 1.

Table 2 summarises the notation presented in definition 3 and theorem 1. In the first row of Table 2, the multiple valued literals $X_i^{S_i}$ of a function $F(X_1, X_2, \ldots, X_n)$ are shown. In the second row, the sets of truth values P_i for those literals are given. Next, the corresponding polarity literals P_i^r are shown, where the r stands for the values represented by the T_i^r vector of the matrix P_i. Finally, Fig. 2 illustrates two polarity matrices P_i consisting of the value vectors T_i^r.

$$P_1 = \begin{bmatrix} T_1^1 \\ \cdots \\ T_1^r \\ \cdots \\ T_1^{Pn} \end{bmatrix} \quad \cdots \quad P_n = \begin{bmatrix} T_n^1 \\ \cdots \\ T_n^r \\ \cdots \\ T_n^{Pn} \end{bmatrix}$$

Fig. 2 *Description of polarity matrices*

2.2 MIGRM forms

Before the MIGRM forms are formally defined, let us consider a simple example of such a form.

Example 2: The two-variable *mv* function $F_1(X_1, X_2) = X_1^{023} X_2^{01}$, where X_1 is four-valued and X_2 is three-valued, is used to show how to calculate the MIGRM

Table 2: Notation for MIGRM

mv literal	$X_1^{S_1}$	$X_2^{S_2}$	$X_i^{S_i}$	$X_n^{S_n}$
set of truth values	$R_1 = (0, 1, \ldots, p_1 - 1)$		$R_i = (0, 1, \ldots, p_i - 1)$	$R_n = (0, 1, \ldots, p_n - 1)$
polarity literals	$P_1^1, \ldots, P_1^r, \ldots, P_1^{p_1}$	$\cdots$	$\cdots$	$P_n^1, \ldots, P_n^r, \ldots, P_n^{p_n}$
polarity matrix	P_1		P_i	P_n

form for the polarity given in Fig. 3. The variable X_2^{01} can be represented by the superposition of the polarity literals $P_2^1 \oplus P_2^3$, where for P_2^1 the subscript 2 indicates the corresponding *mv* literal X_2 and the superscript 1 denotes the values T_2^1 for the *mv* literal X_2. The natural method to perform the transformation would seem to be by an EXOR-term multiplication, i.e.

$$\begin{aligned} X_1^{023}X_2^{01} &= (P_1^1 \oplus P_1^3 \oplus P_1^4)(P_2^1 \oplus P_2^3) \\ &= (1 \oplus P_1^3 \oplus P_1^4)(1 \oplus P_2^3) \\ &= 1 \oplus P_1^3 \oplus P_1^4 \oplus P_2^3 \oplus P_1^3P_2^3 \oplus P_1^4P_2^3 \end{aligned}$$

In example 2, the multiplication is applied to obtain the MIGRM form from the polarity literals. To perform the

$$A_1 = \begin{bmatrix} 1111 \\ 0101 \\ 0011 \\ 0111 \end{bmatrix} = \begin{bmatrix} T_1^1 \\ T_1^2 \\ T_1^3 \\ T_1^4 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 111 \\ 100 \\ 001 \end{bmatrix} = \begin{bmatrix} T_2^1 \\ T_2^2 \\ T_2^3 \end{bmatrix}$$

Fig. 3 *Polarity matrices*

multiplication of the polarity literals shown there, the MIGRM form will be described as a spectrum M. A tabular pattern matching method between the indices of the spectral coefficients and a product term will be introduced to calculate the final MIGRM from the polarity representation of the literals $X_i^{S_i}$.

The following definition extends the concept of the GRM forms for Boolean functions to the concept of MIGRM forms for *mv* functions.

Definition 4: The multiple valued input, binary output generalised Reed–Muller (MIGRM) form of a single output function $F(X_1, X_2, \ldots, X_n)$, where X_i, $i = 1, 2, \ldots, n$ are the *mv* literals, is defined as follows

$$\begin{aligned} F(X_1, X_2, \ldots, X_n) &= a_0 P_1^1 P_2^1 \cdots P_n^1 \\ &\oplus a_1 P_1^1 P_2^1 \cdots P_n^2 \\ &\oplus \cdots \oplus a_{p_n} P_1^1 P_2^1 \cdots P_n^{p_n} \\ &\oplus \cdots \oplus a_t P_1^{p_1} P_2^{p_2} \cdots P_n^{p_n} \qquad (1) \end{aligned}$$

where $a_i \in \{0, 1\}$ and $t = \prod_{i=0}^{n-1} p_i$; the other notation follows the one in Table 2. For a set of functions $F_j(X_1, X_2, \ldots, X_n)$ we obtain

$$\begin{aligned} F_j(X_1, X_2, \ldots, X_n) &= a_{0_j} P_1^1 P_2^1 \cdots P_n^1 \\ &\oplus a_{1_j} P_1^1 P_2^1 \cdots P_n^2 \\ &\oplus \cdots \oplus a_{p_{nj}} P_1^1 P_2^1 \cdots P_n^{p_n} \\ &\oplus \cdots \oplus a_{t_j} P_1^{p_1} P_2^{p_2} \cdots P_n^{p_n} \qquad (2) \end{aligned}$$

Definition 5: The polarity of a MIGRM form is the vector of polarity matrices describing the polarity for each multiple valued literal in the form.

The above definitions can be described with the terminology of spectral techniques as shown in Reference 24 for the GRM form.

Definition 6: The MIGRM given by definition 5 can be represented by the spectrum M, where the index of a spectral coefficient $M_{P_1^r \ldots P_n^r}$ corresponds to the terms of polarity literals $P_1^r \cdots P_n^r$ in eqns. 1 and 2. Those terms represent the standard trivial functions [23] of the MIGRM spectrum.

Example 3: The MIGRM of the function $F_1(X_1, X_2) = X_1^{023}X_2^{01}$ (see example 2) can be represented by its spectrum M. Fig. 4 gives the standard trivial functions of the

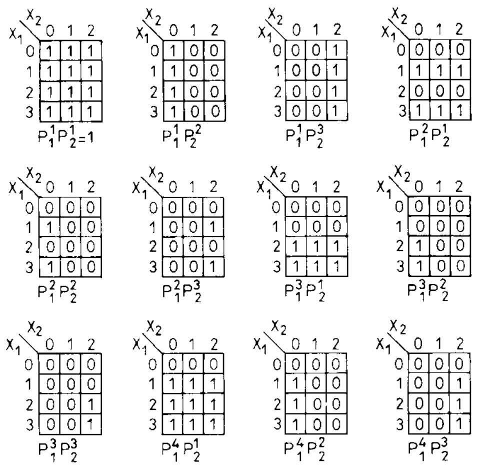

Fig. 4 *Standard trivial functions for polarities of variables X_1 and X_2 specified in Fig. 3*

above product term for the polarity from Fig. 3. For a comparison of the product $X_1^{023}X_2^{01}$ with the standard trivial functions in Fig. 4, the map of this product is shown in Fig. 5.

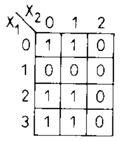

Fig. 5 *Map of function $F(X_1, X_2) = X_1^{023}X_2^{01}$ from examples 2 and 3*

The new expression can be represented now in the form of spectral coefficients (see Table 3), where the indices correspond to the standard trivial functions. The same result as in example 2 has been obtained

$$X_1^{023}X_2^{01} = 1 \oplus P_1^3 \oplus P_1^4 \oplus P_2^3 \oplus P_1^3P_2^3 \oplus P_1^4P_2^3$$

The reader may wish to verify this form by exoring the corresponding maps from Fig. 4 to obtain the map from Fig. 5. The algorithm for calculating this form will be given in Section 3.

The 'coefficient' row of Table 3 gives all possible spectral coefficients, where the MIGRM terms represent all the indices for a general function where the arguments are a four-valued literal X_1 and a three-valued literal X_2.

Table 3: Spectrum of function $F_1(X_1, X_2)$

Coefficient	$M_{P_1^1P_2^1}$	$M_{P_1^1P_2^2}$	$M_{P_1^1P_2^3}$	$M_{P_1^2P_2^1}$	$M_{P_1^2P_2^2}$	$M_{P_1^2P_2^3}$	$M_{P_1^3P_2^1}$	$M_{P_1^3P_2^2}$	$M_{P_1^3P_2^3}$	$M_{P_1^4P_2^1}$	$M_{P_1^4P_2^2}$	$M_{P_1^4P_2^3}$
value	1	0	1	0	0	0	1	0	1	1	0	1

In the 'value' row, a '1' indicates that the term represented by the index of the spectral coefficient in the same column is present in the MIGRM form. In Table 3, the terms $M_{P_1^1P_2^1}$, $M_{P_1^1P_2^3}$, $M_{P_1^3P_2^1}$, and $M_{P_1^4P_2^1}$ are identical to 1, P_2^3, P_1^3, and P_1^4 because $P_1^1 = P_2^1 = 1$.

The final MIGRM form is now obtained by substitution of the polarity literals for their multiple valued literal representation according to the polarity matrices. Thus

$$X_1^{023}X_2^{01} = 1 \oplus P_2^3 \oplus P_1^3 \oplus P_1^3P_2^3 \oplus P_1^4 \oplus P_1^4P_2^3$$
$$= 1 \oplus X_2^2 \oplus X_1^{23} \oplus X_1^{23}X_2^2 \oplus X_1^{123} \oplus X_1^{123}X_2^2$$

Example 4 illustrates the transformation of multioutput functions consisting of more than one product term. The transformation is based on the input function being in ESOP form or disjoint SOP form, where a disjoint SOP form is a form in which the product terms of the function have no overlapping parts in their map representation.

Example 4: The MIGRM form of the set of two functions $F_1(X_1, X_2 = X_1^{023}X_2^{01}$, and

$$F_2(X_1, X_2) = X_1^{023}X_2^{01} + X_1^0X_2^2$$
$$= X_1^{023}X_2^{01} \oplus X_1^0X_2^2$$

for the polarity used in Examples 1 and 2 is

$$F_1 = X_1^{023}X_2^{01}$$
$$= 1 \oplus P_2^3 \oplus P_1^3 \oplus P_1^3P_2^3 \oplus P_1^4 \oplus P_1^4P_2^3$$

as calculated in example 3, where the map of F_1 is shown in Fig. 6. The second function F_2 is composed of the

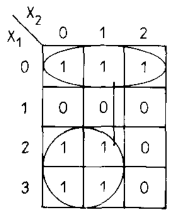

Fig. 6 *Map of function $F_2(X_1, X_2)$*

polarity terms

$$F_2 = X_1^{023}X_2^{01} \oplus X_1^0X_2^2$$
$$= 1 \oplus P_1^3 \oplus P_1^3P_2^3 \oplus P_1^4$$

For a comparison of the function F_2 with the standard trivial functions for the polarities of variables X_1 and X_2 shown in Fig. 5, the map of F_2 is given in Fig. 6. To apply spectral techniques as in example 3, the notation for the coefficients from Table 3 is used.

The output functions F_1 and F_2 are represented by an output term $f_1 f_2$ for each product term. The output term $f_1 f_2$ is given in the right column of Table 4. Now the spectrum for each term is calculated separately, similar to example 4. Instead of using '1' as an entry in the spectrum table, the output term for the product term (Table 4) is taken. As will be shown in Section 3.2, the calculation of the spectrum can be performed in one step for all output functions. However, for case of explanation, in Table 5 we assume that the spectrum for each output function is calculated separately. Thus, the spectrum for each term in each output function can be obtained as shown in example 3, Table 3. Because the product term

Table 4: Representation of the multioutput function $F(X_1, X_2) = F(F_1, F_2)$

Product term	f_1f_2
$X_1^{023}X_2^{01}$	11
$X_1^0X_2^2$	01

$X_1^{023}X_2^{01}$ which is present in F_1 and F_2 is the same as in example 3, the results can be taken directly for the spectrum shown in Table 5. The spectrum for the term $X_1^0X_2^2$ can be calculated analogously. The results are given in Table 5. The first bit of the entries 11 and 10 in the table stands for the output function F_1 and the second one for the output function F_2.

The calculation of the entries in Table 5 can be performed in two different ways:

(i) Each product term is compared with the standard trivial function of each spectral coefficient by the tabular pattern matching method.

(ii) The spectral coefficients are determined directly from the product term. Thus, the entry '—' in Table 5 indicates that no calculation has to be performed for these cells.

The calculation according to (i) is used in our implementation. $P_1^1P_2^1$, $P_1^1P_2^3$, $P_1^3P_2^1$, and $P_1^4P_2^1$ are identical to 1, P_2^3, P_1^3, and P_1^4 because $P_1^1 = P_2^1 = 1$. The row 'EXOR' in Table 5 is obtained by exoring the entries (output functions) in every column. Finally, the last row gives the polarity literals for the spectral coefficients having nonzero entry in the 'EXOR' row. The polarity terms in the result row are the same as obtained for the functions F_1 and F_2 at the beginning of this example.

Now the polarity literals have to be replaced by their multiple valued literals as shown in example 3. Thus, we obtain

$$F_1 = 1 \oplus X_2^2 \oplus X_1^{23} \oplus X_1^{23}X_2^2 \oplus X_1^{123} \oplus X_1^{123}X_2^2$$

as calculated in example 3 and

$$F_2 = 1 \oplus X_1^{23} \oplus X_1^{123} \oplus X_1^{23}X_2^2$$

3 Algorithm for calculation of MIGRM form

The method for the generation of the MIGRM form consists of two basic stages. First, as described in Section 3.1, each multiple valued literal of the *mv* function has to be transformed to a polarity specified by the chosen orthogonal polarity matrix. In the second stage, presented in Section 3.2, the terms consisting of transformed literals are used to calculate the final MIGRM form. The code for the transformation of a multiple valued literal is chosen in such a way that the second stage of the transformation is not dependent on the chosen polarity for the

Table 5: Spectrum of function $F(X_1, X_2)$

Term	$M_{P_1^1P_2^1}$	$M_{P_1^1P_2^2}$	$M_{P_1^1P_2^3}$	$M_{P_1^2P_2^1}$	$M_{P_1^2P_2^2}$	$M_{P_1^2P_2^3}$	$M_{P_1^3P_2^1}$	$M_{P_1^3P_2^2}$	$M_{P_1^3P_2^3}$	$M_{P_1^4P_2^1}$	$M_{P_1^4P_2^2}$	$M_{P_1^4P_2^3}$
$X_1^{023}X_2^{01}$	11	—	11	—	—	—	11	—	11	11	—	11
$X_1^0X_2^2$	—	—	01	—	—	—	—	—	—	—	—	01
EXOR	11	—	10	—	—	—	11	—	11	11	—	10
result	1		$P_1^1P_2^3$				P_1^3		$P_1^3P_2^3$	P_1^4		$P_1^4P_2^3$

literal. This approach requires the introduction of the concept of normalised codes.

3.1 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 normalised code will be illustrated by an example. Table 6 shows the transformations for some possible four-valued literals to the polarity used in the previous examples. The first row gives all the possible combinations of the polarity literals. Let us observe that the first four polarity literals are the rows of matrix P_1 from example 2. In the first column, some possible literals of the variable X 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. $X^0 = P^1 \oplus P^4 = 1 \oplus P^4$, which is denoted by $1111 \oplus 0111 = 1000$). The second row is then created as follows. Its entries for columns P^1, P^2, and P^3 are the binary representations of the polarity literals from the polarity matrix. All other entries in the second row are created by the bit-by-bit EXOR operation on the P^i literals from the column header. Finally the last row of the table gives the normalised code of the spectral coefficients. It represents the combination of the polarity literals P^r. For instance, 0111 means that variables P^2, P^3, and P^4 are used. This row is just a binary encoding of the first row.

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

Example 5: The literal X^0 with the internal representation 1000 can be represented by $P^1 \oplus P^4$ which has the normalised 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 7.

The normalised code has a '1' in its bit representation corresponding to the index of the polarity literal P^r.

Table 7: Code representation of polarity literals

Normalised code	Polarity literals P^r
1000	P^1
0100	P^2
0010	P^3
0001	P^4

The algorithm 1 performs the transformation of a multiple valued literal to the normalised code.

Algorithm 1:

Step 1: Generate all possible EXOR combinations of the polarity literals P^r (the first row of Table 6) for later comparison with the original *mv* literal.

Step 2: Compare the binary representation of the *mv* literal with the binary representation (row 2, Table 6) of the EXOR combination (row 1, Table 6) of Step 1. If these two binary representations are equal, assign to the *mv* literal its normalised code (shown in the last row of Table 6).

3.2 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 MIGRM. Our implementation is based on a tabular pattern matching method, where every product term of the input function is compared with the normalised code of the indices of all spectral coefficients.

The basic steps of the algorithm for the tabular pattern matching are explained in the following example.

Example 6: Let us assume a function $G(X_1, X_2, X_3)$ where the literal X_1, being three-valued, is represented by three polarity literals P_1^1, P_1^2 and P_1^3. The second literal X_2, being four-valued, is represented by the polarity literals P_2^1, P_2^2, P_2^3 and P_2^4, and the three-valued literal X_3 is represented by P_3^1, P_3^2, and P_3^3. The notation of the spectral coefficients for a spectrum representing such a function $G(X_1, X_2, X_3)$ is shown in Table 8. The code shown in Table 8 is obtained by generating all possible combinations of polarity literals 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. $M_{P_1^2 P_1^3}$ cannot occur because both

Table 6: Transformations of some four-valued literals

Literal	P^1	P^2	P^3	P^4	$P^1 \oplus P^2$	$P^1 \oplus P^3$	$P^1 \oplus P^4$	$P^2 \oplus P^3$	$P^2 \oplus P^4$	$P^3 \oplus P^4$
	1111	0101	0011	0111	1010	1100	1000	0110	0010	0100
X^0							1			
X^1										1
X^{01}						1				
X^{12}								1		
X^{012}										
Norm. code	1000	0100	0010	0001	1100	1010	1001	0110	0101	0011

Literal	$P^1 \oplus P^2 \oplus P^3$	$P^1 \oplus P^2 \oplus P^4$	$P^1 \oplus P^3 \oplus P^4$	$P^2 \oplus P^3 \oplus P^4$	$P^1 \oplus P^2 \oplus P^3 \oplus P^4$
	1001	1101	1011	0001	1110
X^0					
X^1					
X^{01}					
X^{12}					
X^{012}					1
Norm. code	1110	1101	1011	0111	1111

Table 8: Complete MIGRM spectrum of function $G(X_1, X_2, X_3)$ and three-valued X_3

M_i	$M_{P_1^1P_2^1P_3^1}$	$M_{P_1^1P_2^1P_3^2}$	$M_{P_1^1P_2^1P_3^3}$	$M_{P_1^1P_2^2P_3^1}$	$M_{P_1^1P_2^2P_3^2}$	$\cdots$	$M_{P_1^3P_2^4P_3^3}$
Code	1001000100	1001000010	1001000001	1000100100	1000100010	$\cdots$	001 0001 001

polarity literals are from the original *mv* literal X_1. As one can observe from the binary representation of the index (second row in Table 8), 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. Because Table 8 has been created for normalised *mv* literals, the normalised code has to be used for the code in Table 8 also.

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 8, is obtained by comparing the normalised codes of all product terms, further called $term_x$ ($x = 0, \ldots, m-1$; where m is the total number of terms) of the Boolean function including variables $X_1, X_2, \ldots, X_n$ with the representation of the spectral coefficients $code_s$ from Table 8. The value M_s for $term_x$ is the representation of the output functions f_x of $term_x$, 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 has to be performed among all m values M_s. The calculation of the values of coefficients M_s is described by

$$M_s^0 = 0$$

$$M_s^{x+1} = M_s^x \oplus (((code_s \,\&\, term_x) \neq \varnothing)\&f_x),$$
$$\text{for } x = 0, \ldots, m-1 \quad (3)$$

where the value of the spectral coefficient is a product term, $code_s$ and $term_x$ are the binary representations of the respective literals, where $((code_s \,\&\, term_x) \neq \varnothing)$ has to be true for the intersection of each literal of $code_s$ and $term_x$. By $\varnothing$ we denote a vector which contains at least one zero. The output product term for $term_x$ is denoted by f_x.

The final MIGRM form consists of the terms obtained by replacing the polarity literals in the indices of the spectral coefficients M_s (row code in Table 8) which have a value that is not '0', with their binary representation.

To summarise, the procedure for obtaining the MIGRM of a multioutput Boolean function can be described by algorithm 2:

Algorithm 2:

Step 1: Transform all the multiple valued input literals of the function to their normalised codes according to the chosen polarities for the variables (Section 2, algorithm 1).

Step 2: Calculate the MIGRM spectrum for the normalised codes as presented in Section 2.2.

Step 3: Replace the polarity literals of non-zero spectral coefficients by their original *mv* literal. The output functions for the terms are given by the values of the spectral coefficients.

As one can observe, each spectral coefficient can be calculated in turn. Thus, it can be stored directly on hard disk. With this approach, the necessity to keep the whole spectrum in the computer memory has been overcome.

4 Practical example

A real-life example is included to illustrate the complete MIGRM transformation. The function used is a two-bit adder (Table 9, 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 9: Truth table of two-bit adder

Binary	Multiple valued		Normalised		Output
x_1, x_2, x_3, x_4	X_1	X_2	X_1	X_2	$f_c f_0 f_1$
0000	1000	1000	1110	1110	000
0001	1000	0100	1110	1111	001
0010	1000	0010	1110	0010	010
0011	1000	0001	1110	1011	011
0100	0100	1000	1111	1110	001
0101	0100	0100	1111	1111	010
0110	0100	0010	1111	0010	011
0111	0100	0001	1111	1011	100
1000	0010	1000	0010	1110	010
1001	0010	0100	0010	1111	011
1010	0010	0010	0010	0010	100
1011	0010	0001	0010	1011	101
1100	0001	1000	1011	1110	011
1101	0001	0100	1011	1111	100
1110	0001	0010	1011	0010	101
1111	0001	0001	1011	1011	110

The input variable assignment is not unique. Thus, there exist many different assignments. In this example, the *mv* literal X_1 is obtained by changing the first two bits of the binary function (x_1, x_2) to a four-valued literal ($X_1^0 = 00$, $X_1^1 = 01$, $X_1^2 = 10$, $X_1^3 = 11$). The second two bits are used to obtain X_2.

To make it easier to follow the steps of the transformation, the same polarity is assumed for both literals. The polarity literals P_i^r and the binary representations of their values (T_i^r) are given in Table 10.

Table 10: Polarity for two-bit adder

Polarity for X_1	Polarity for X_2
P_1^1: 1111	P_2^1: 1111
P_1^2: 0101	P_2^2: 0101
P_1^3: 0010	P_2^3: 0010
P_1^4: 1100	P_2^4: 1100

Now the binary representations of the literals X_1 and X_2 in Table 7 are replaced by their normalised codes (algorithm 1). Because only four different values occur in the literals X_1 and X_2, the normalised codes for those values are shown in Table 11, where $P^r = P_1^r = P_2^r$. The multiple valued literals X_1 and X_2 shown in Table 9 are now substituted with the cube representations of their normalised codes given in Table 11; the result is given in Table 9.

Table 11: Normalised codes for four occurring values

Binary value of literal	Normalised code	Cube representation of normalised code
0001	$P^1 \oplus P^3 \oplus P^4$	1011
0010	P^3	0010
0100	$P^1 \oplus P^2 \oplus P^3 \oplus P^4$	1111
1000	$P^1 \oplus P^2 \oplus P^3$	1110

The normalised code obtained in this procedure is now compared with all the indices of the spectral coefficients of the general spectrum for a function $G(X_1, X_2)$,

Table 12: Spectrum for first three terms from table 9

Term	$M_{p_1^1p_2^1}$	$M_{p_1^1p_2^2}$	$M_{p_1^1p_2^3}$	$M_{p_1^1p_2^4}$	$M_{p_1^2p_2^1}$	$M_{p_1^2p_2^2}$	$M_{p_1^2p_2^3}$	$M_{p_1^2p_2^4}$
	1000 1000	1000 0100	1000 0010	1000 0001	0100 1000	0100 0100	0100 0010	0100 0001
1110 1111	001	001	001	001	001	001	001	001
1110 0010	—	—	010	—	—	—	—	010
1110 1011	011	—	011	011	011	—	011	011
Result	010	001	000	010	010	001	010	000

Term	$M_{p_1^3p_2^1}$	$M_{p_1^3p_2^2}$	$M_{p_1^3p_2^3}$	$M_{p_1^3p_2^4}$	$M_{p_1^4p_2^1}$	$M_{p_1^4p_2^2}$	$M_{p_1^4p_2^3}$	$M_{p_1^4p_2^4}$
	0010 1000	0010 0100	0010 0010	0010 0001	0001 1000	0001 0100	0001 0010	0001 0001
1110 1111	001	001	001	001	—	—	—	—
1110 0010	—	—	010	—	—	—	—	—
1110 1011	011	—	011	011	—	—	—	—
Result	010	001	000	010	000	000	000	000

where X_1 and X_2 are four-valued literals (algorithm 2, step 2). Table 12 illustrates the calculation of the spectrum for the first three terms from Table 9. The Table has the output functions of the terms as entries if, for each term, the intersection of the term and the index of the coefficient is not empty (eqn. 3); otherwise the entry is '—'. For instance, the cell for the intersection of row 1110 1011 and column 0100 1000 is '011' as the intersection of those indices is 0100 1000 where both literals are not empty. The intersection of row 1110 0010 and column 0100 1000 is 0100 0000 so '—' is placed in the corresponding cell. The final coefficients for the partial function in Table 12 are obtained by exoring the entries of the columns. The result of this operation is given in the last row.

The result of the complete function is shown in Table 13. The first column of Table 13 lists all nonzero spectral coefficients M_s that occur in any of its component functions. These coefficients have been obtained by comparing (as in Table 12) the normalised binary representations of their indices (listed in the second column of Table 13) with the normalised code obtained according to Tables 10 and 11. 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 row, by their binary representations (algorithm 2, step 3). According to Table 10, the conversion is: 1000 to 1111, 0100 to 0101, 0010 to 0010, and 0001 to 1100. The output terms $f_c\, f_0\, f_1$ are the values of the spectral coefficients.

Table 13: MIGRM of 2-bit adder

Spectral coefficient	Index	X_1	X_2	$f_c f_0 f_1$
$M_{p_1^1p_2^1}$	1000 1000	1111	1111	100
$M_{p_1^1p_2^2}$	1000 0100	1111	0101	101
$M_{p_1^1p_2^4}$	1000 0001	1111	1100	110
$M_{p_1^2p_2^1}$	0100 1000	0101	1111	101
$M_{p_1^2p_2^2}$	0100 0100	0101	0101	010
$M_{p_1^2p_2^3}$	0100 0010	0101	0010	100
$M_{p_1^2p_2^4}$	0100 0001	0101	1100	100
$M_{p_1^3p_2^2}$	0010 0100	0010	0101	100
$M_{p_1^4p_2^1}$	0001 1000	1100	1111	110
$M_{p_1^4p_2^2}$	0001 0100	1100	0101	100
$M_{p_1^4p_2^4}$	0001 0001	1100	1100	100

The implementation of the above result as an AND-EXOR PLA with two-input, three-output decoders for the chosen polarities is shown in Fig. 7.

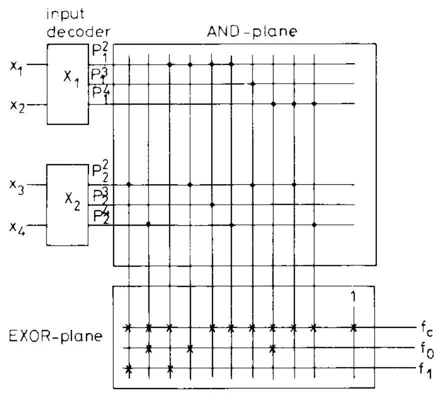

Fig. 7 *AND-EXOR PLA implementation with input decoders of the MIGRM form of the 2-bit adder*

The input decoders are the same because the same polarity has been chosen for both literals, the input decoder for both variables X_1, and X_2 is shown in Fig. 8.

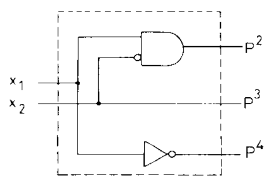

Fig. 8 *Input decoder for variables X_1 and X_2*

5 Evaluation and results

The calculation method presented for the computation of the MIGRM has been implemented in the program GRM-MV (generalised Reed–Muller synthesiser, multiple valued version). The numerical results have confirmed the validity of the MIGRM concept. However, the problem of finding a polarity with the minimal number of product terms without the computation of all possible forms still has to be solved. It is known, that for an n variable Boolean function, there are 2^n GRM forms. For an n variable function where each variable is three-valued, there are 28^n forms, and for each variable being four-valued there are 840^n forms. Therefore, the computation of the form having the minimal number of product terms by going through all forms is very complex. Moreover, the pairing of the Boolean variables to obtain the multiple valued form also has an impact on the number of necessary product terms in the MIGRM.

For the evaluation of the MIGRM in comparison to the GRM, the number of polarities for a four-valued variable has been restricted to 60 randomly selected polarities. Therefore, the total number of polarities considered was 60^n. Table 14 gives the results obtained for some MCNC benchmark functions.

Table 14: Benchmark results for MIGRM

	ESPRESSO	GRM	ESPRESSO 2-bit decoder	MIGRM 2-bit decoder
adr2	11	8	9	5
adr4	75	34	66	34
bw	22	22	23	22
con1	9	17	9	17
inc	30	47	28	40
rd53	31	20	12	15
rd73	127	63	37	40
rd84	255	107	54	54
squar5	25	23	26	25
xor5	16	5	4	4
5xp1	65	61	55	64

The column ESPRESSO lists the number of product terms obtained by the two-level SOP optimizer ESPRESSO [13]. The following column gives the minimal number of product terms in the respective GRM form. For the computation of the MIGRM and MIESOP based on four-valued variables, pairs of Boolean variables have been taken to obtain a four-valued one. The results obtained are given in the last columns of Table 14.

One can observe from the obtained results that the MIGRM gives a reduction of up to 50% in the number of product terms over the GRM. Moreover, the number of product terms of the GRM is the absolute minimum while only a local minimum is obtained for the MIGRM. This is due to the restriction of the number of polarities per variable to 60 instead of 840.

The results obtained give motivation to further investigate in MIGRM forms. However, methods to avoid the computation of all possible polarities similar to the ones introduced for GRMs [5, 25] have to be developed to overcome the high computational complexity of MIGRM forms. Additionally, it has to be investigated which multiple valued decoders are practical for circuit realisations. Then, the search for the minimal MIGRM form can be restricted to the one which is optimal with respect to a certain set of multiple valued decoders.

6 Conclusions

The extension of the general Reed–Muller expansion to multiple valued input, binary multioutput functions has been shown. For this, the concept of code normalisation of single multiple valued literals to perform a final transformation has been developed. The code normalisation is applied to make the transformation of the complete function independent of the polarity chosen. This simplifies and speeds up the main transformation step to the final MIGRM form for the transformed single *mv* literal. For further investigations of the properties of such forms, the MIGRM transformation has been implemented as a computer algorithm. Because an exhaustive search of all $\prod_{i=0}^{n} p(X_i)$ MIGRM forms, where $p(X_i)$ is the number of possible polarities for variable X_i with $i = 1, \ldots, n$, would be too time consuming, further research has to be concentrated on avoiding a complete search and to finding immediately good polarities as has been carried out for GRM forms [5, 25].

As circuits which realise the MIGRM forms are both easily testable, or modifiable to very easily testable circuits, further research into them is important. It must however, be 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 [15] must also be investigated, as a good pairing together with a good choice of variable polarities may significantly improve the cost.

7 References

1 AKERS, S.B.: 'On a theory of Boolean functions', *SIAM J.*, 1959, **7**, pp. 487–498

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

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

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

5 SARABI, A., and PERKOWSKI, M.A.: 'Fast exact and quasi-minimal minimization of highly testable fixed-polarity AND/XOR canonical networks'. Proceedings of the 29th Design Automation Conference, Anaheim, CA, June 1992, pp. 30–35

6 BHATTACHARYA, B.B., GUPTA, B., SARKAR, S., and CHOUDHURY, A.K.: 'Testable design of RMC networks with universal tests for detecting stuck-at and bridging faults', *IEE Proc. E, Comput. Digit. Tech.*, 1985, **132**, (3), pp. 155–161

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

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

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

10 REDDY, S.M.: 'Easily testable realizations for logic functions', *IEEE Trans. Comput.*, 1972, **21**, (11), pp. 1182–1188

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

12 REDDY, B.R.K., and PAI, A.L.: 'Reed–Muller transform image coding', *Comput. Vis., Graph. Image Process.*, 1988, **42**, pp. 48–61

13 RUDELL, R., and SANGIOVANNI-VINCENTELLI, A.: 'Multiple-valued minimization for PLA optimization', *IEEE Trans. Comput.-Aided Des. Integrated Circuits and Syst.*, 1987, **6**, (5), pp. 727–750

14 SASAO, T.: 'On the optimal design of multiple-valued PLA's', *IEEE Trans. Comput.*, 1989, **38**, (4), pp. 582–592

15 SASAO, T.: 'Multiple-valued decomposition of generalized Boolean functions and the complexity of programmable logic arrays', *IEEE Trans. Comput.*, 1981, **30**, (9), pp. 636–643

16 BRAYTON, R.K., RUDELL, R., SANGIOVANNI-VINCENTELLI, A., and WANG, A.R.: 'MIS: multi-level interactive logic optimization system', *IEEE Trans., Comput.-Aided Des. Integrated Circuits and Syst.*, 1989, **6**, (6), pp. 1062–1082

17 SASAO, T.: 'MACDAS: multi-level AND-OR circuit synthesis using two-variable function generators', Proceedings of the 23rd Design Automation Conference, June 1986, pp. 86–93

18 PERKOWSKI, M.A., HELLIWELL, M., and WU, P.: 'Minimization of multiple-valued input multi-output mixed-radix exclusive sums of products for incompletely specified Boolean functions', 19th Int. Symp. on Multiple-Valued Logic, May 1989, pp. 256–263

19 SCHÄFER, I., and PERKOWSKI, M.A.: 'Multiple valued generalized Reed–Muller forms', 21st Int. Symp. on Multiple-Valued Logic, May 1991, pp. 40–48

20 SASAO, T.: 'EXMIN: a simplification algorithm for exclusive-OR-sum-of-products expressions for multiple-valued input two-valued output functions', 20th Int. Symp. on Multiple-Valued Logic, May 1990, pp. 128–135

21 HU, Z.: 'The simple methods for evaluating the coefficients of the canonical RM expansion of multivalued functions', *Int. J. Electron.*, 1987, **63**, (6), pp. 851–856

22 KODANDAPANI, K.L., and SETLUR, R.V.: 'Reed–Muller canonical forms in multivalued logic', *IEEE Trans. Comput.*, 1975, pp. 628–636

23 FALKOWSKI, B.J., and PERKOWSKI, M.A.: 'Walsh type transforms for completely and incompletely specified multiple-valued

input binary functions', 20th Int. Symp. on Multiple-Valued Logic, May 1990, pp. 75–82
24 FALKOWSKI, B.J., and PERKOWSKI, M.A.: 'On the calculation of generalized Reed–Muller canonical expansions from Disjoint cube representation of Boolean functions'. IEEE 33rd Midwest Symp. on Circuits & Systems, August 1990, pp. 1131–1133
25 ALMAINI, A.E.A.: 'Tabular techniques for Reed–Muller logic', *Int. J. Electron.*, 1991, **70**, (1), pp. 23–34