A FAST ALGORITHM TO MINIMIZE MULTI-OUTPUT MIXED-POLARITY GENERALIZED REED-MULLER FORMS

Martin Helliwell
Marek Perkowski
Portland State University
Department of Electrical Engineering
P.O. Box 751
Portland, OR, 97207.
tel. (503) 464-3806x23

ABSTRACT

A very fast computer program that accepts a Boolan function as an array of multi-output disjoint cubes and returns a mixed-polarity Generalized Reed-Muller Form is presented. Such circuits often have gates and interconnections than classical sum-of-product realizations and are easily testable. The program was tested on many examples from literature as well as on many large arithmetic functions with up to 8 inputs, 8 outputs and 255 minterms. On all the examples from the literature the solutions were either the same or better than those generated by other methods. The algorithm is based on a new cube operation, called xlinking, that generalizes known operations of merger, exclusion and other logic operations specified by previous authors.

1. INTRODUCTION

In recent years there is a growing interest in design of logic circuits with EXOR gates. Particular interest is in the minimization of the Generalized Reed-Muller Forms (GRM), with fixed or mixed polarity of variables [3], [13], [14], [19], [30], [37], [41], [44]. Functions realized with such circuits can have less gates, less connections and smaller VLSI realization [3], [14]. What is even more important, such circuits are easily testable, they are also used in self-testing circuits [2], [4], [15], [16], [24], [28], [29], [32], [35], [36], [40]. As Besslich writes [3] : "secondly, the testability of circuits is significantly improved [35]. The gains from this second advantage may even exceed possible disadvantages in such cases where the EXOR realization is more costly than the equivalent vertex (sum of product) form. Applications have so far not become very popular because of the practical difficulties in the design procedure". GRM circuits find applications in linear machines, arithmetic and communication circuits, encrypting schemes, coding schemes for error control and synchronization, sequence generation for process identification, system testing and other applications.

The problem of minimization of such circuits is very important, but it was traditionally treated as extremely difficult. Since optimal solutions can be found only for functions with not more than 5 variables [30], the interest is in approximate solutions. Although few authors [3], [14], [37] have implemented computer programs, no results have been published that would help to evaluate the speed and quality of their approaches.

The approach presented here attempts to do for GRM forms the same that Espresso [39] has done for standard PLA minimization:

- to create a practical, heuristic method, that would generate optimal or nearly optimal results for most practical size function examples from engineering practice,
- to document the program execution times and properties on many benchmark examples,
- to provide any interested party with the source code.

In this paper a program EXORCISM (EXOR CIrcuit Speedy Minimizer) will be described. It will be incorporated into large VLSI design automation system DIADES [31], which includes several logic minimization procedures that correspond to various kinds of logic and layout realizations and which gives the user the choice of selecting the appropriate minimization approach. We solve here a problem that has not yet been solved in the literature: mixed-polarity minimization of multiple-output incompletely specified functions.

2. WHY IS THE USE OF GRM BENEFICIAL

It is well-known that the set {0, 1} with operations · (AND) and ⊕ (EXOR) is a ring Z_2 and a field. Therefore the following operations hold:

1. Associative laws:
   $a\oplus(b\oplus c)=(a\oplus b)\oplus c$
   $a(bc)=(ab)c$
2. Distributive laws:
   $a(b\oplus c)=ab\oplus ac$
3. Commutative laws:
   $a\oplus b=b\oplus a$
   $ab=ba$
4. Identities:
   $a\oplus 0=a$
   $a\oplus 1=\bar{a}$
   $a1=a$
   $a0=0$
   $a\oplus a=0$
   $a\oplus\bar{a}=1$

A Reed-Muller form is nothing more than an EXOR-sum of products (called also product terms or terms for short), where every literal has positive polarity. It can be found very easily by starting with a list of minterms and replacing each negative literal with the positive literal plus one:

$$a\bar{b}c = a(b\oplus 1)c = abc\oplus ac$$

$$a\bar{b}c\bar{d} = a(b\oplus 1)c(d\oplus 1) = abc(d\oplus 1)\oplus ac(d\oplus 1) = abcd\oplus abc\oplus acd\oplus ac$$

In Generalized Reed-Muller form, each literal may be positive or negative, but must remain in the same polarity through the entire expression. It can be found in a very similar way to the Reed-Muller form.

The mixed GRM form is one in which the literals can be of both polarities at the same time in different terms. The **degree of a term** is the number of literals in it.

Our primary goal is to minimize the number of terms (inputs to EXOR gates). For the circuit with the minimum number of terms our secondary goal is to minimize the total number of inputs. Therefore the cost function to be minimized is:

$$C = NT + \frac{NI}{NI_{in}}$$

where:
- NT is the total number of terms in the solution,
- NI is the total number of inputs in the solution,

- NI_{in} is the total number of inputs in the initial function.

We became interested in synthesis with EXOR gates when Jeff Fox from (then) SILC group in GTE Labs pointed out to the second author that many important applications of circuits with EXORs exist in telecommunication industry and there is no program currently available for design of such circuits.

Later, we found many other circuits, especially code converters and arithmetic circuits, for which the designs with EXORs were simple (see section 5 on numerical results below). Moreover, we learned about the role of EXOR gates in the design for testability [20]. Reed-Muller canonical forms were considered by Reddy [35] as candidates for easily testable circuits with **function independent** testing. For an arbitrary function f of n input variables, by adding one extra AND gate to the Reed-Muller canonical realization of f, with inputs from all those primary inputs that are connected to an even number of AND gates, one can detect all single stuck-at faults by applying only (n+4) test patterns, independent of the function f [35], [15]. Function f can be tested also for all single stuck-at faults with $(n + 2n_e + 4)$ tests where n_e is the number of input variables appearing in an even number of terms. This research has been expanded in [24], [25], [40], [32], [29] and others.

The mixed-polarity GRMs that are discussed here also have very good testability properties, but no systematic study similar to that of [35] is known to the authors.

GRM forms, as well as forms that arise from their factorization can be realized with standard cell library approach, or with multi-input NOR gate, EXOR gate and XNOR gate cell generators. Another approach would be to expand the concept of PLA to the EXOR-based PLA where the EXOR-plane replaces the OR-plane. We are currently working on possible layout of such PLA and PAL circuits.

In interesting concept, not yet to our knowledge realized by industry, would be to fabricate the off-the-shelf EXOR-based PAL, PLA and EPLD user-programmable devices. Minimization of Boolean functions for programming of such devices was now made possible because of availability of EXORCISM. The design for testability would be thus greatly simplified on this design level.

3. THE XLINK OPERATION.

Basic operation of our system is the operation of **xlinking** that generalizes several operations known from the previous papers.

Let us first observe that any two minterms m1 and m2 in a Karnaugh map can be linked by a chain of groups, where:
- each group has two adjacent K-map cells,
- the first group includes m1, the last group includes m2,
- any two subsequent groups of the chain include a single common minterm.

It can be easily proven that the EXOR of minterms m1 and m2 is equal to the EXOR of all groups from the chain. Figure 1 shows an example of a chain from minterm 0100 to minterm 1011. Let us observe that usually there are many such chains from m1 to m2.

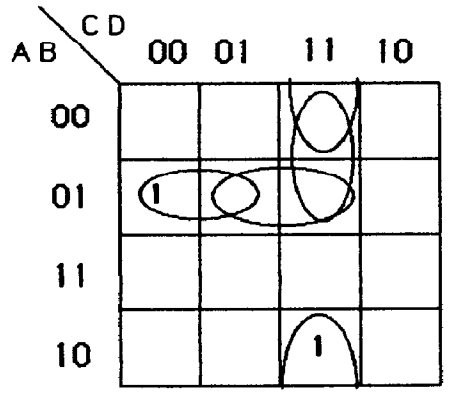


Fig. 1

The first principle of our method to minimize GRM forms is based on the idea that any two minterms can be expanded into an EXOR sum of one or more terms containing fewer literals. It is helpful to put a dash in the place of the missing literal.

For example:

$$A\,B\,\bar{C}\,\bar{D} \oplus A\,\bar{B}\,C\,\bar{D} = A - C\,\bar{D} \oplus A\,B - \bar{D},$$

$$\bar{A}\,B\,\bar{C}\,D \oplus \bar{A}\,\bar{B}\,\bar{C}\,D = \bar{A} - \bar{C}\,D$$

(this is a counterpart of the well known merging rule),

$$\bar{A}\,B\,\bar{C}\,\bar{D} \oplus A\,\bar{B}\,C\,D = -\bar{B}\,C\,D \oplus \bar{A} - C\,D \oplus \bar{A}\,B - D \oplus \bar{A}\,B\,\bar{C} -$$

(see Fig. 1).

As we see, the chaining operation expands the well-known Boolean rules of merging and exclusion used in the Quine-McCluskey or other logic design algorithms. It finds one of the shortest paths between two nodes of a Boolean hypercube. The number of generated terms equals the Hamming distance of these nodes.

Clearly a systematic way of finding the chain expansions is needed. Below we will give a procedure for finding them. The application of this procedure will be called **xlinking** (pronounced crosslinking). The result of the procedure will be called the **xlink** (crosslink) of the two original minterms. The mintenms and the product terms are represented as ternary cubes (cubes with bits 0, 1, -) in the computer. Positional notation that permits for both 2-valued and m-valued minimization is used [42].

To find the xlink of a pair of two minterms, for example $A\,\bar{B}\,C\,D\,\bar{E}$ and $A\,B\,\bar{C}\,D\,\bar{E}$ we write them vertically like this:

$$A\,\bar{B}\,C\,D\,\bar{E}$$

$$A\,B\,\bar{C}\,D\,\bar{E}$$

Each time when the polarity of the literal changes from minterm to minterm in the pair it is denoted by an arrow. Each arrow will give rise to one term of the xlink. Let us now consider each arrow separately. The above initial pair of minterms can then be expanded to two **secondary pairs**, for variables B and C respectively, as shown below.

For variable B the term $A - \bar{C}\,D\,\bar{E}$ is created as follows:

$A\ \bar{B}\ C\,D\,\bar{E}$ } the first secondary pair
$A\ B\ \bar{C}\,D\,\bar{E}$ }
$A\ -\ \bar{C}\,D\,\bar{E}$

For variable C the term $A\,\bar{B} - D\,\bar{E}$ is created as follows:

$A\,\bar{B}\ C\ D\,\bar{E}$ } the second secondary pair
$A\,B\ \bar{C}\ D\,\bar{E}$ }
$A\,\bar{B}\ -\ D\,\bar{E}$

Under each pair of literals of different polarities under consideration (B in the first pair, C in the second pair) we write the don't care symbol (dash). To create the result of xlink for a secondary pair **we copy the part of the term to the left of the dash from the top minterm. The part to the right of the dash is copied from the bottom minterm, as shown.** The xlink of the initial pair of minterms is an EXOR of xlink terms of the secondary pairs for each variable of different polarities.

In our case it can then be seen that:

$$A\,\bar{B}\,C\,D\,\bar{E} \oplus A\,B\,\bar{C}\,D\,\bar{E} = A - \bar{C}\,D\,\bar{E} \oplus A\,\bar{B} - D\,\bar{E}.$$

This operation is illustrated in the Karnaugh map from Fig. 2.

This procedure can be easily extended for any two terms that have dashes in the same positions. For example:

$$\bar{A} - C\,D - \oplus \bar{A} - C\,\bar{D} - = \bar{A} - C - -.$$

This is also illustrated in Fig. 2. The terms (cubes) that have dashes in the same positions will be called **xlinkable terms (cubes)**. For instance

$A\ B\ -\ C\ -$ and $\bar{A}\ \bar{B}\ -\ \bar{C}\ -$

are xlinkable and

$A\ -\ B\ C\ -$ and $\bar{A}\ \bar{B}\ -\ \bar{C}\ -$

are not.

This type of xlinking for any two xlinkable terms will be called **primary xlinking.**

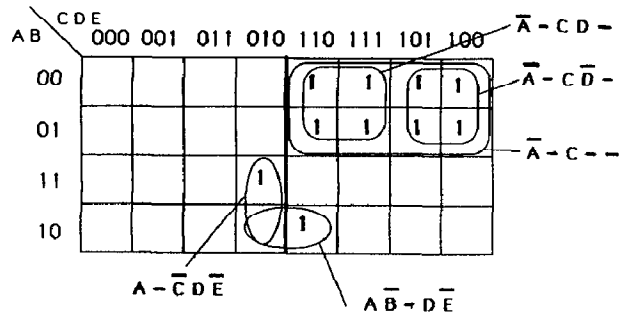


Fig. 2.

Let us now introduce another type of xlinking. This new operation permits two terms of different degrees to be xlinked. As we have seen, the primary xlinking reduces the degree of the terms. By use of the primary xlinking, together with the laws:

1) $x \oplus 0 = x$,

2) $x \oplus x = 0$,

3) $\oplus$ is commutative and associative,

we will be able to formulate a **secondary xlinking** that xlinks terms of degrees differing by one.

Secondary xlinkable terms are two terms that satisfy these conditions:

- their degrees differ by one,
- the term with the higher degree contains all the variables from the other term.

For example, the terms $\bar{A}\ B\ -\ D\ E$ and $\bar{A}\ \bar{B}\ C\ D\ \bar{E}$ are secondary xlinkable, since the term

$\bar{A}\ \bar{B}\ C\ D\ \bar{E}$

of degree 5 has all the variables of the terms

$\bar{A}\ B\ -\ D\ E$

of degree 4.

Let us now consider an example of secondary xlinking. It uses the primary xlinking and the above three laws.

The secondary xlinking will be applied to:

$\bar{A}\ B\ -\ D\ E \oplus \bar{A}\ \bar{B}\ C\ D\ \bar{E}$.

First, let us see that variable C is lacking in $\bar{A}\ B\ -\ D\ E$ and occurs as C in $\bar{A}\ \bar{B}\ C\ D\ \bar{E}$. Let us then create a term, that is adjacent with respect to variable C to the term $\bar{A}\ \bar{B}\ C\ D\ \bar{E}$: this will be a term $\bar{A}\ \bar{B}\ \bar{C}\ D\ \bar{E}$.

Now we EXOR the previous EXOR sum with the zero term:

$\bar{A}\ B\ -\ D\ E \oplus \bar{A}\ \bar{B}\ C\ D\ \bar{E} \oplus (\bar{A}\ \bar{B}\ \bar{C}\ D\ \bar{E} \oplus \bar{A}\ \bar{B}\ \bar{C}\ D\ \bar{E})$.

Next we apply the fact that exoring is associative:

$\bar{A}\ B\ -\ D\ E \oplus (\bar{A}\ \bar{B}\ C\ D\ \bar{E} \oplus \bar{A}\ \bar{B}\ \bar{C}\ D\ \bar{E}) \oplus \bar{A}\ \bar{B}\ \bar{C}\ D\ \bar{E}$.

We xlink the terms in the parantheses:

$\bar{A}\ B\ -\ D\ E \oplus \bar{A}\ \bar{B}\ -\ D\ \bar{E} \oplus \bar{A}\ \bar{B}\ \bar{C}\ D\ \bar{E}$.

We xlink the first two terms:

$\bar{A}\ -\ -\ D\ \bar{E} \oplus \bar{A}\ B\ -\ D\ - \oplus \bar{A}\ \bar{B}\ \bar{C}\ D\ \bar{E}$.

The result shows that the lower degree term has been changed to terms of even lower degree, and the upper degree term has been changed to another term of the same degree.

The above operation is illustrated in the Karnaugh map of Fig. 3.

Remember that each cell covered by an even number of groups is a "zero" cell and one covered by an odd number of groups is a "one" cell (minterm).

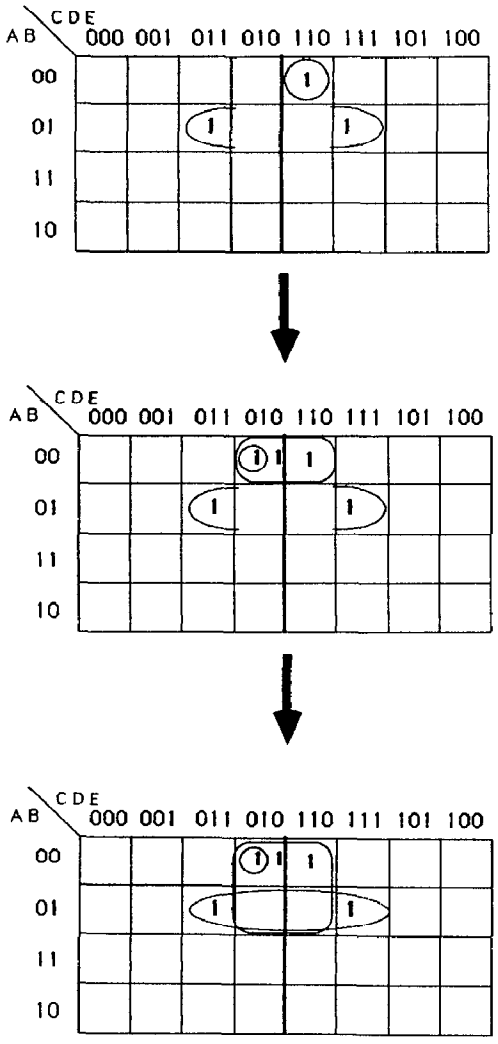


Fig. 3.

The above sequence of transformations was shown for the sake of explanation, but the execution of the secondary xlink is very straightforward in our implementation of the xlink operation. Its execution is therefore speedy.

4. ALGORITHM.

The algorithm currently used in our program is quite simple. The idea is to carry out all primary and secondary xlinks possible, in some reasonable order, giving priority to xlinking least distant groups first. Distance here refers to the Hamming distance of two groups.

Simply stated the algorithm is:

1. Do all primary xlinks with preference given to closer groups.
2. Do all secondary xlinks with preference given to closer groups.
3. If any xlinking was done in 2, then go to 1

It is important in steps one and two of the algorithm to do xlinking on terms of high degree first. This allows for the results of the first xlinks to be compared for xlinking to groups of the same degree before those groups are xlinked to groups of lower degree.

When performing primary or secondary xlinks the program finds close groups by taking the first group in the list and comparing it to every other group and choosing the closest. After performing that xlink it moves to the second group in the list, and so on.

After performing secondary xlinks it is required to check again for primary xlinks, because the secondary xlinks may contain primary xlinkable groups.

By way of example, take the function $f = \sum(1,3,7,10,12,13,15)$:

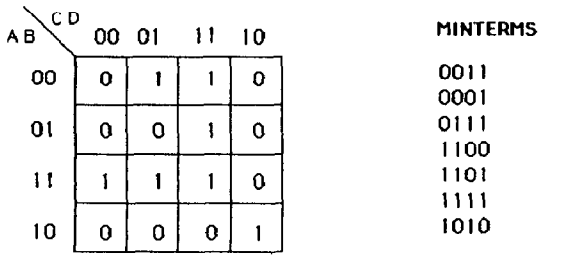
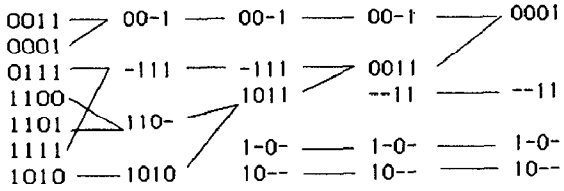


Fig. 4.



The first column is a list of minterms, the second is the results of the primary xlinking in step 1 of the algorithm. The group 1010, being unxlinkable, was just carried to the next step. Columns 3, 4, and 5 show the results of the secondary xlinking in step 2 of the algorithm. Step 3 checks to see if any more primary xlinking is possible - since none is possible, the algorithm terminates.

This program handles multioutput functions in a very simple way. Each output is minimized with the stated xlinking algorithm, producing a list of terms. After each output is minimized, each term from that and all previous outputs is included, one by one with each succeeding output - to see if it helps the minimization. If it is beneficial, then it is included as a term in the final minimization, if not, it is discarded and the next term from a previous output is tried.

A better system for handling multioutput functions is to use each term generated in a previous output as a "don't care" minterm or group in each succeeding output. The idea here is that any "don't care" groups included during minimization can be re-included in the circuit at little cost since the product already exists in a previous output. Remember that if a group is included twice, it is the same as if the group was not included at all (a $\oplus$ a=0). Because of the even/odd properties of GRM forms the existence of even a single don't care permits term minimization. Don't cares can be therefore used to more extent than in the classical minimization and xlinking can be further modified to take this into account.

5. NUMERICAL RESULTS.

In this section we will discuss the numerical results of our algorithm. All results were obtained from an IBM AT with a 10MHz clock and 640K RAM. The program was written in C. Running times and other data for each example are given in Table 1.

Example 1. This is Example 1 from [10]. $f(x_2, x_1, x_0) = \sum(0, 2, 6, 7)$. The same solution, $f = \overline{x_2}\,\overline{x_0} \oplus x_2 x_1$, as in [Davi71] was generated.

Example 2. This is Example 2 from [9]. $f(x_2, x_1, x_0) = \sum(0, 3, 5, 6, 7)$. The solution from [9],

$f = x_0 \oplus \overline{x_1}\,\overline{x_2} \oplus \overline{x_0} x_1 x_2$

has 3 terms and 8 gate inputs. Our solution,

$f = x_0 \oplus x_1 \oplus \overline{x_2} \oplus x_0 x_1 x_2$

has 4 terms but only 7 gate inputs. The solution from [9] could be obtained with our algorithm if a different sorting procedure were applied.

Example 3. This is Example 1 from [13]. $f(x_1, x_2, x_3) = \sum(0, 1, 3, 5, 7)$. The same solution as in [13] was generated:

$f = x_3 \oplus \overline{x_1}\,\overline{x_2}\,\overline{x_3}$.

Example 4. This example is from [42], [13], [3] and [25].

ex. no.	Number of inputs	Number of outputs	Number of minterms	Terms after minimization	Number of ANDs gates	Number of gate inputs	Time (sec)	Improvement in terms
1	3	1	4	2	2	6	*	50%
2	3	1	5	4	1	7	*	20%
3	3	1	5	2	1	5	*	60%
4	5	1	10	4	4	18	*	60%
5	3	1	4	3	0	3	*	25%
6	4	4	15	4	0	9	3	73%
7	4	1	8	4	0	4	*	50%
8	4	4	15	4	0	6	3	73%
9	4	3	15	3	0	9	5	80%
10	3	1	4	3	3	9	*	25%
11	5	1	10	4	4	18	3	60%
12	6	6	49	19	19	83	48	62%
13	4	3	15	8	4	18	*	47%
14	8	5	255	34	26	135	840	87%
15	6	3	63	8	5	24	10	88%
16	8	4	255	22	19	97	240	91%
17	3	6	7	7	5	18	*	0%
18	6	12	63	40	35	152	180	37%
19	8	8	255	119	119	654	7800	54%

* time less than 2 sec.

Table 1.

$f(x_1, x_2, x_3, x_4, x_5) = x_2\overline{x_3}x_4 + x_2\overline{x_3}\,\overline{x_5} + x_1\overline{x_2}\,\overline{x_3}\,\overline{x_4} + x_1\overline{x_2}x_4 x_5$. The same solution as in [13] was obtained:

$f = x_1\overline{x_2}x_4x_5 \oplus x_2\overline{x_3} \oplus x_1\overline{x_2}\,\overline{x_3}\,\overline{x_4} \oplus x_2\overline{x_3}\,\overline{x_4}x_5$.

This is the best of all solutions found in the literature.

Example 5. This is Example 1 from [32].

$f(x_0, x_1, x_2) = \sum(1, 2, 4, 7)$. The solution from [Rama65], $f = x_0 \oplus x_1 \oplus x_2$ has the same term cost as our solution:

$f = \overline{x_0} \oplus x_1 \oplus \overline{x_2}$.

The same solution as in [32] would be generated by our algorithm with a different sorting routine.

Example 6. This Grey code - to - binary conversion from Fig. 5.26a page 160 of [18]. The same solution:

$B_4 = G_4$,
$B_3 = G_3 \oplus G_4$,
$B_2 = G_2 \oplus G_3 \oplus G_4$,
$b_1 = G_1 \oplus G_2 \oplus G_3 \oplus G_4$,

as in [18] has been generated.

Example 7. This is a simple parity-check encoder from Fig. 5.27 page 161 of [18]. The same solution:

$I_1' = I_1$,
$I_2' = I_2$,
$I_3' = I_3$,
$I_4' = I_4$,
$P = I_1 \oplus I_2 \oplus I_3 \oplus I_4$,

as in [18] has been generated.

Example 8. This is binary-to-grey code conversion from Fig. 5.26, page 161 of [18]. The same solution:

$G_4 = B_4$,
$G_3 = B_3 \oplus B_4$,
$G_2 = B_2 \oplus B_3$,
$G_1 = B_1 \oplus B_2$,

as in [18] was generated.

Example 9. This is an encoder for 7-bit Hamming code from Fig. 5.28 page 162 of [Gree86]. The solution from [18] is:

$P_1 = I_4 \oplus I_1 \oplus I_2$,
$P_2 = I_4 \oplus I_1 \oplus I_3$,
$P_3 = I_4 \oplus I_2 \oplus I_3$,

Our solution is,

$P_1 = I_4 \oplus \overline{I_1} \oplus \overline{I_2}$,
$P_2 = I_4 \oplus \overline{I_1} \oplus \overline{I_3}$,
$P_3 = I_4 \oplus I_2 \oplus I_3$,

Example 10. This is Example 1 from [42]. $f = X_1X_2 + X_1X_3 + X_2X_3$. The same solution, $f = X_1X_2 \oplus X_1X_3 \oplus X_2X_3$ as in [42] was obtained.

Example 11. This is Example 2 from [42]. $f(X_1, X_2, X_3, X_4, X_5) = \sum(8, 10, 11, 16, 17, 19, 23, 24, 26, 27)$. The solution from [42],

$f = \overline{X_3} \oplus \overline{X_3}\,\overline{X_4}X_5 \oplus \overline{X_2}\,\overline{X_3} \oplus \overline{X_2}\,\overline{X_3}\,\overline{X_4}X_5 \oplus X_1\overline{X_2}X_5 \oplus X_1X_2X_4X_5 \oplus X_1X_2X_3X_4$

has 7 terms, 6 AND gates and 27 gate inputs. Our solution,

$f = X_2\overline{X_3} \oplus X_2\overline{X_3}\,\overline{X_4}X_5 \oplus X_4X_5X_1\overline{X_2} \oplus X_1\overline{X_2}\,\overline{X_3}\,\overline{X_4}$

has 4 terms, 4 AND gates and 18 gate inputs.

Example 12. This is a combinational arithmetic multiplier (f = x * y) with 6 inputs and 6 outputs. The solution is in Fig. 5. The number of terms is the same as in the multiple-value-inputs PLA (with 2-input decoders) of Sasao [40].

```
Number of inputs: 6
Number of outputs: 6
Number of products: 49
1-1111 110000
1111-1 110000
11-11- 100000
110110 010000
-1111- 011000
11--11 010000
1--1-- 010000
-11011 001000
110-11 001000
1-11-1 001000
-1-1-- 001000
1---1- 001000
-11-11 000100
--11-- 000100
-1--1- 000100
1----1 000100
--1-1- 000010
-1---1 000010
--1--1 000001
Number of terms: 19
```

Fig. 5.

```
Number of inputs: 8
Number of outputs: 4
Number of products: 255
00------ 1110
-------- 1001
-000---- 0110
1------- 0100
-10000-- 0011
0--000-- 0010
0-0----- 0010
-1------ 0010
1-001000 0001
-110-000 0001
---00000 0001
0---1000 0001
1--0-000 0001
0-0--000 0001
00---000 0001
-0101--- 0001
0--000-- 0001
0-110--- 0001
00--0--- 0001
-1-0---- 0001
1--1---- 0001
1-1----- 0001
Number of terms: 22
```

Fig. 6.

```
Number of inputs: 6
Number of outputs: 12
Number of products: 63
10111- 111000000000
11---- 100000000000
110--- 010000000000
1----- 010001000000
-10011 001111000000
010-11 001110000000
11-1-- 001000000000
-01--- 001000000000
0-1--- 001000000000
101011 000100000000
1-111- 000100000000
11-01- 000100000000
110-1- 000100000000
-110-- 000100000000
1--1-- 000100000000
-1---- 000100000000
1-0111 000010000000
10-111 000010000000
1011-1 000010000000
-11-1- 000010100000
```

```
1-1--1 000010010000
--01-- 000010000000
-0-1-- 000010000000
1---1- 000010000000
-11101 000001000000
--110- 000001000000
-1--10 000001000000
1----0 000001000000
--1--- 000001000000
--11-1 000000100000
-01-1- 000000100000
---11- 000000100000
-1---1 000000100000
---110 000000010000
0-1--1 000000010000
---1-- 000000010000
----01 000000001000
---0-1 000000001000
----10 000000000100
-----1 000000000001
Number of terms: 40

23:33:41.16 C:\C>
```

Fig. 7.

Example 13. This is a 2 + 2 adder (f = x + y) with 4 inputs and 3 outputs.

Example 14. This is a 4+4 adder with 8 inputs and 5 outputs.

Example 15. This is a 6-bit square rooter (f = int(sqr(x))) with 6 inputs and 3 outputs.

Example 16. This is an 8-bit square rooter (f = int(sqr(x))) with 8 inputs and 4 outputs. Solution in Fig. 6.

Example 17. This is a 3 input, 6 output square circuit (f = x * x).

Example 18. This is a 6x6 square circuit (f = x * x) with 6 inputs and 12 outputs. See Fig. 7 for solution.

Example 19. This is a 4x4 multiplier with 8 inputs and 8 outputs.

Our mean improvement was 54.84% on functions with mean value of inputs equal 4.74, mean value of outputs equal 3.47, and the mean value of terms equal 59. Papakonstantinou [30] reports an optimal program that had an improvement of 55% on 100 randomly generated single output functions of five variables with a mean value of 13 terms. The methods from Even [12] and Bioul [5] were reported to give an improvement of 51%.

6. CONCLUSION

A fast computer algorithm for minimization of mixed-variable GRM has been presented. There are no data in literature on performance of such algorithms, and essentially very few papers have been published on this topic. As illustrated by the examples, the algorithm gives very good results, and is fast, even implemented on a personal computer.

The algorithm permits for synthesis of multi-output incompletely specified functions - there are currently no papers on this topic in the literature.

It can be easily upgraded to synthesize fixed polarity forms (by using first the presented algorithm and next transforms $\overline{a} = 1 \oplus a$ or $a = \overline{a} \oplus 1$ for selected variables followed by obvious simplifications). Such RM forms have some advantage as far as the minimization of the number of function's inputs is concerned, which can be related to testability.

The algorithm is now being improved by adding more passes and tree branching with various heuristic evaluation functions. Also, the next passes will include shrinking operations ($A \oplus B = \overline{A}\,B \oplus A\,\overline{B}$) followed by xlinking operations. The results can be also improved in some cases if in the last pass the operations: $\overline{a} = 1 \oplus a$ and $a \oplus a = 0$ are iterated (this would for instance improve the solution from Example 9 by removing negations of variables). We see also many other possible improvements of the algorithm, and we plan to experiment with many variants in order to generate absolutely optimum solutions in all or nearly all examples known to us. Since the algorithm is fast, we are able to sacrifice some speed by making it even more optimal.

We will consider also applying a cost function that would more accurately estimate the real layout cost.

The advantage of our method is that it starts from an arbitrary array of disjoint terms (disjoint cubes) and not necessarily from a canonical Reed-Muller form like most the well-known algorithms.

Another advantage of the approach presented here is that our methods can be easily extended to the description of functions with multiple-valued inputs (multiple-valued EXOR forms are described in [17], clasical PLAs with multiple-valued inputs in [42], a new concept of GRM Forms with multiple-valued inputs, that generalizes ideas from [42] and this paper will be described in our outcoming paper).

Currently we see a multitude of fruitful research areas that have been initiated by the research reported here:

1. *Factorization of GRM polynomials* (extensions of the Brayton's methods [6], as well as classical polynomial factorization methods are possible),
2. Design for testability of multi-output functions of the above categories. Testability analysis of mixed-polarity GRM functions extending the known results. Analysis for factorized networks.

4. Application of the spectral analysis methods [23] to the above and more general problems. Decomposition into linear and nonlinear part. Decomposition into functions with simple GRM realizations.
5. Mixed forms, composed of EXOR, NOR and NAND gates, created to concurrently minimize the area and optimize the testability. Creating an expert system by generalizing xlink operations and extending the local transforming operations set (see EXPO [31]). We observed that often the functions that give good results with EXORs produce poor results with classical approaches. Some sort of an evaluation function has to be created that would permit to evaluate the most appropriate realization method for any decomposed subfunction.
6. Joining EXORCISM as a subroutine to MIS, EXPO [31] or other multi-paradigm program for multi-level logic minimization.
7. Efficient realization of many input EXOR gates for EXOR-based PLA layout.

7. LITERATURE

[1] Akers, S.B.: "On a theory of Boolean functions", J. SIAM, Vol. 7., pp. 487-498, December 1959.

[2] Bennetts, R.G., Lewin, D.: "Fault diagnosis of digital systems - a review", Comput. J., Vol. 14., pp. 199-206, 1971.

[3] Beslich, Ph.W.: "Efficient Computer Method for EXOR Logic Design", Proc. IEE, Vol. 130, Part E, CDT, No. 6., pp. 203-206, 1983.

[4] Bhavsar, D., Heckelman, R.W.: "Self-testing by polynomial division", Proc. 1981 IEEE Test Conference, pp. 208-216, 1981.

[5] Bioul, G., Davio, D., Deschamps, J.P.: "Minimization of ring-sum expansions of Boolean functions", Philips Research Report, Vol. 28, pp. 17-36, 1973.

[6] Brayton, R.K., Camposano, R., De Micheli, G., Otten, R.H.J.M., Van Eijndhoven, J.: "The Yorktown Silicon Compiler System", Chapter 7 in Gajski, D., (ed), Silicon Compilation, 1987.

[7] Calingaert, P.: "Switching function canonical forms based on commutative and associative binary operations", AIEE Trans. Vol. 79, pp. 808-814, January 1961.

[8] Carter, W.C., Wadia, A.B., Jessep, D.C.Jr.: "Implementation of Checkable Acyclic Automata by Morphic Boolean Functions", Proc. of the Symp. on Comp. and Automata, Polytechnic Institute of Brooklyn, p. 645, 1971.

[9] Cohn, M.: "Inconsistent canonical forms of switching functions", IRE Trans. Electron. Comput., Vol. EC-11, p. 284, April 1962.

[10] Davio, M.: "Ring-Sum Expansions of Boolean Functions", Presented at the Symposium on Computers and Automata, Polytechnic Institute of Brooklyn, April 13-15, 1971. Computers and Automata, pp. 411-418.

[11] Davio, M., Deschamps, J.P., Thayse, A.: "Discrete and Switching Functions", McGraw-Hill Book Co., Inc., New York, 1978.

[12] Even, S., Kohavi, I., Paz, A.: "On minimal modulo-2 sum of products for switching functions", IEEE Trans. on Electron. Computers, pp. 671-674, October 1967.

[13] Fleisher, H., Tavel, M., Yeager, J.: "Exclusive-OR representations of Boolean functions", IBM J. Res. Develop., vol. 27, pp. 412-416, July 1983.

[14] Fleisher, H., Tavel, M., Yeager, J.: "A Computer Algorithm for Minimizing Reed-Muller Canonical Forms", IEEE Trans. on Computers, Vol. C-36, No. 2, February 1987.

[15] Fujiwara, H.: "Logic Testing and Design for Testability", Computer System Series, The MIT Press, 1986.

[16] Green, D.H., Dimond, K.R.: "Polynomial representation of nonlinear feedback shift-registers", Proc. IEE, 117, No. 1., pp. 56-60, 1970.

[17] Green, D.H., Taylor, I.S.: "Multiple-valued switching circuit design by means of generalized Reed-Muller expansions", Digital Processes, No. 2, pp. 63-81, 1976.

[18] Green, D.H., Edkins, M.: "Synthesis procedures for switching circuits represented in generalized Reed-Muller form over a finite field", Computer and Digital Techniques, Vol. 1., No. 1., pp. 22-35, 1978.

[19] Green, D.: "Modern Logic Design", Electronic Systems Engineering Series, 1986.

[20] Hayes, J.P.: "On modifying logic networks to improve their diagnosability", IEEE Trans. Comput. Vol. C-23, No. 1, pp. 56-62, 1974.

[21] Hurst, S.I.: "Logical processing of digital signals", Edward Arnold, London: Crane-Russak, N.Y., 1978.

[22] Katzan, H., Jr.: "The Standard Data Encryption Algorithm", New York: Petrocelli, pp. 62-68, 1977.

[23] Karpovsky, M.: "Finite Orthogonal Series in the Design of Digital Devices. Analysis, Synthesis, Optimization ", J. Wiley & Sons, Israel Univ. Press, 1976.

[24] Kodandapani, K.L.: "A note on easily testable realizations for logical functions", IEEE Trans. Comp., Vol. C-23, pp. 332-333, 1974.

[25] Kodandapani, K.L., Setlur, R.V.: "A note on minimum Reed-Muller canonic forms of switching functions", IEEE Trans. Comp., Vol. C-26, pp. 310-313, 1977.

[26] Lechner, R.J.: "Transformations Among Switching-Function Canonical Forms", IEEE Trans. Electr. Comput. Vol. EC-12, No. 2., pp. 129-130, April 1963.

[27] Mukhophadhyay, A., Schmitz, G.: "Minimisation of exclusive-OR and logical equivalence switching circuits", IEEE Trans. Comp., Comp. Vol. C-19, No. 2. , pp. 132-140, February 1970.

[28] Muller, D.E.: "Application of Boolean algebra to switching circuit design and to error detection", IRE Trans. Electron. Comp., Vol EC-3, pp. 6-12, September 1954.

[29] Page, E.W.: "Minimally testable Reed-Muller canonical forms", IEEE Trans. Comput., Vol. C-29, No. 8., pp. 746-750, 1980.

[30] Papakonstantinou, G.: "Minimization of modulo-2 sum of products", IEEE Trans. on Computers., Vol. C-28, pp. 163-167, February 1979.

[31] Perkowski, M.: "Digital Design Automation System DIADES. Documentation", Dept. EE, Portland State University, Portland, OR, 1988.

[32] Pradhan, D.K.: "Universal test sets for multiple fault detection in AND-EXOR arrays", IEEE Trans. on Comput., Vol. C-27, No.2., pp. 181-187, 1978.

[33] Pradhan, D.K.: "Fault-Tolerant Computing. Theory and Techniques. Vol. I." Prentice-Hall, 1987.

[34] Ramamoorthy, C.V.: "Procedures for minimization of "exclusive or" and "logical equivalence" switching circuits", IEEE Symp. on Switching Circuit Theory and Logical Design, pp. 143-149, October 1965.

[35] Reddy, S.M.: "Easily testable realization for logic functions", IEEE Trans. Comput., Vol. C-21, pp. 1183-1188, Nov. 1972.

[36] Reed, I.S.: "A class of multiple-error-correcting codes and their decoding scheme", IRE Trans. Inf.Th. Vol. PGIT-4, pp. 38-49, 1954.

[37] Robinson, J.P., Yeh, C.L.: "A Method for modulo-2 Minimization", IEEE Trans. on Computers, Vol. C-31, pp. 800-801, August 1982.

[38] Roth, P.: "Computer Logic, Testing and Verification", Rockville, MD: Computer Science, 1980.

[39] Rudell, R.: "Multiple-Valued Logic Minimization for PLA Synthesis", M.S. Report, June 5, 1986. University of California, Berkeley California 94720.

[40] Saluja, K.K., Reddy, S.M.: "Fault detecting test sets for Reed-Muller canonic networks", IEEE Trans. Comput., Vol. C-24, No. 10., pp. 995-998, 1975.

[41] Saluja, K.K., Ong, E.H.: "Minimization of Reed-Muller canonic expansion", IEEE Trans. Comput., Vol. C-28, pp. 163-167, February 1979.

[42] Sasao, T.: "Input Variable Assignment and Output Phase Optimization of PLA's", IEEE Trans. on Comp., Vol. C-33, No. 10, pp. 879-894, October 1984.

[43] Schmokler, M.S.: "Mod-2 Sums of Products", IEEE Trans on Comp., Vol. C-18, No. 10., October 1969.

[44] Wu, X., Chen, X., Hurst, S.L.: "Mapping of Reed-Muller coefficients and the minimisation of Exclusive-OR switching functions", Proc. IEE, part E, vol. 129, pp. 15-20, January 1982.