Fast Exact and Quasi-Minimal Minimization of Highly Testable Fixed-Polarity AND/XOR Canonical Networks

Andisheh Sarabi

Department of Electrical Engineering
Portland State University
Portland, OR 97223

Marek A. Perkowski

Department of Electrical Engineering
Portland State University
Portland, OR 97223

Abstract

The high testability of AND/XOR networks and new technologies that make their use more possible call for new minimization and synthesis tools. This paper introduces the fast Exact and Quasi-minimal algorithms for minimal fixed polarity AND/XOR canonical representation of Boolean functions. The method uses features of array of disjoint cubes representation of functions to identify the minimal networks. These features can drastically reduce the search space and provide high quality heuristics for quasi-minimal representations. Experimental results show that these special AND/XOR networks, on the average, have similar number of terms to Boolean AND/OR networks while there are functions for which AND/XOR circuits are much smaller. The circuits generated here are much more testable.

1 Introduction

AND/XOR realizations of switching functions have certain advantages over traditional AND/OR realizations but, due to the lack of circuit technologies and necessary synthesis tools, they have not been used extensively.

In many applications the AND/XOR realizations of the circuits require less layout area than their AND/OR counterparts [1]. The AND/XOR PLAs often require fewer products than AND/OR PLAs [2]. These networks are also highly testable and many of their applications can be found in arithmetic, encoding, telecommunication, and linear systems.

Although XOR gates already exist in EPLDs (Signetics LHS501, arithmetic PALs, etc.), and standard cell libraries, the full potential of this logic has not been realized in the mainstream logic design and synthesis. One disadvantage, in terms of realization, has been the slow speed and area constraints of the XOR gate. Recent advances in FPGA technology have made significant progress in this regard. FPGAs such as the Xilinx 3000 family, which incorporate a high-speed lookup table approach to implementing any function of up to five input variables, make an XOR gate with 5 inputs as fast as any other gate of the same number of inputs. Another example is that of the 1010/1020 family of devices from Actel. The cascaded multiplexer structure of Actel makes it possible to implement two- and most three-input functions, that include XOR operators. Of special interest here are new programmable technologies that include XOR as one of few logic gates available in them. For instance, the CLi6006 Field Programmable Gate Array from Concurrent Logic, Inc. includes a two-input XOR gate in a small granularity block [3].

Currently, few, if any, minimization and synthesis methods exist which use AND/XOR-based methods for these devices. Two-level minimizers such as ESPRESSO and multi-level minimizers such as MISII, BOLD, or FPGA specific ones such as mis-pga do not include XOR synthesis schemes.

In this paper methods for fast minimization of functions with large numbers of variables and fixed-polarity AND/XOR networks will be presented. These special AND/XOR networks are not only the most easily testable of all general purpose networks, but also have other merits of their own. These canonical forms which are known as Consistent Generalized Reed-Muller *CGRM* [4] forms, are used for general Boolean function representations in such applications as the classification of functions, Ashenhurst and other decomposition methods, and multi-level design. In addition, these forms have been studied for applications in image processing and used as the basis for minimization of other AND/XOR networks [5].

In the following sections after a brief introduction of *CGRM* forms, the testability of AND/XOR networks is discussed. Next, fast realization of *CGRM* expansion for a given polarity is introduced along with several new cube operations. In section 5, an Exact method to find the minimal polarity *CGRM* is presented together with characteristics of arrays of disjoint cubes which can reduce the required search space. In the following section a Quasi-Minimal approach is described and in section 7 some experimental results are presented and discussed.

†This research was partially supported by the NSF grant MIP-9110772.

0738-100X/92 $3.00 © 1992 IEEE

2 Background

While there are only two canonical forms for the Boolean AND/OR networks, the number of possible canonical representations of a function for the AND/XOR networks is very large [4]. One class of AND/XOR networks, the first to be noted in the literature, is that of the *Reed-Muller canonical (RMC)* form [9, 10] and its superset, the *Consistent Generalized Reed-Muller (CGRM)* canonical forms [4].

The *RMC* representation consists of only positive product terms and is given as:

$$f(x_1, x_2, \ldots, x_n) = \bigoplus_{i=0}^{2^n-1} a_i \mu_i \qquad (1)$$

where $a_i \in \{0,1\}$ and $\mu_i = {x_n}^{e_n} {x_{n-1}}^{e_{n-1}} \ldots {x_2}^{e_2} {x_1}^{e_1} = \prod_{j=1}^{n} {x_j}^{e_j}$ where $e_j \in \{0,1\}$ such that $e_n e_{n-1} \ldots e_2 e_1$ is a binary number which equals i. Moreover ${x_i}^0 = 1$ and ${x_i}^1 = x_i$. $\oplus$ denotes the summation over GF(2), the Galois field of two elements.

If the restriction that all the variables should take positive polarity is removed and they are also allowed to take negative polarities, one can have a *Generalized Reed-Muller (GRM)* canonical form. If the variables are, however, restricted to retain the same polarity, either positive or negative, in all product terms, the canonical form will be that of the *Consistent Generalized Reed-Muller (CGRM)* form.

For n variables, there are 2^n possible arrangements of polarities; hence, there are 2^n possible *CGRM* forms for a function. Depending on the polarity of the variables, the number of terms in the expansion varies. As an example for three variables, the function $\bar{x}_1\bar{x}_2\bar{x}_3$ is represented in *RMC* as:

$$\bar{x}_1\bar{x}_2\bar{x}_3 = 1 \oplus x_1 \oplus x_2 \oplus x_1x_2 \oplus x_3 \oplus x_1x_3 \oplus x_2x_3 \oplus x_1x_2x_3. \qquad (2)$$

The same function will just be represented as $\bar{x}_1\bar{x}_2\bar{x}_3$ if negative polarities are chosen for all the three variables. This *CGRM* obviously has less number of terms. The problem of finding those polarities of the variables which result in a *CGRM* with a least number of terms is essentially the minimization problem of interest here.

In order to differentiate between the Boolean product terms and the terms in Reed-Muller forms, the term "monoterm" will be used for the latter.

Definition 1 *A* monoterm *is a product term in Reed-Muller canonical forms.*

The minimization problem can be formulated in terms of the minimal polarity. In this paper, the binary number $\mathbf{p} = p_1 p_2 \ldots p_n$ where $p_i \in \{0,1\}$ such that $\dot{x}_i = {x_i}^{p_i}$ is called the *polarity*. If $p_i = 0$, $\dot{x}_i = \bar{x}_i$ and if $p_i = 1$, $\dot{x}_i = x_i$. For all possible CGRM forms of a switching function, the *minimal polarity* is the polarity which results in a CGRM form with the least number of monoterms. The polarity of each literal in this polarity is called the *polarity literal.*

3 Testability of AND/XOR Networks

The inherent high testability of AND/XOR networks has long been known [11, 12]. It has been shown [11] that all single stuck-at-faults of the *RMC* network, assuming the primary inputs are fault free, can be detected by the set of $(n+4)$ tests which are independent of the function. With faulty primary inputs, the same number was shown to be sufficient to detect stuck-at-faults provided that an extra observable AND gate is added to the network. With addition of extra observable outputs, the same $n + 4$ independent tests can detect all single and multiple stuck-at-faults. It has been shown that the tests for detecting bridging faults of the *RMC* network are also independent of the function. It has to be noted that the test set for the *CGRM* networks is of the same cardinality as one for the *RMC* network. The only difference is that the test vector bits for the variables with negative polarities are the opposites of the ones in the *RMC* network. The distinction between *CGRM* networks and other AND/XOR networks was made by Pradhan [12]. Although the test set for the other AND/XOR networks is also independent of the function, its cardinality is higher than *CGRM* networks.

4 Fast Realization of the CGRM

Following Fisher [6], the problem of *CGRM* minimization of a switching function can be divided into two steps. The first step is to identify the minimal polarity and the second is to realize the *CGRM* expansion of the function with this polarity. In this section a fast method dealing with the second step will be presented. The identification of the minimal polarity is the subject of sections 5 and 6.

One efficient method for realizing a *CGRM* expansion of switching functions is by operating on the set of disjoint cubes which represent the function [6, 7, 8]. In this method, the function is represented by disjoint cubes rather than minterms to reduce the memory requirements. Monoterms representing each cube are expanded and those occurring in an odd number of cubes are retained as the ones representing the function. The fast method introduced here uses the new operations of cube *commonality, difference*, and *symmetric difference* together with a fast Gray-code approach to realize a *CGRM* expansion. Before introducing the method, the monoterms representing each cube, originally reported by Fisher [6] for the case of *CGRM*, are given by Theorem 1:

Theorem 1 *The monoterms originating from a cube for the* RMC *expansion are all the cubes that have their 1s in the same literal positions as the 1s of the original cube and either "0" or "-" in the 0-literal positions of the original cube.*

These monoterms will be referred to as monoterms *representing* the cube from now on.

$\equiv$	0	1	–
0	1	0	–
1	0	1	–
–	–	–	–

Table 1: Equivalence Operator for a Simple Bit

Γ	0	1	-
0	0	1	-
1	1	1	$\emptyset$
-	-	$\emptyset$	-

Table 2: Cube Commonality Operator for a Single Bit

-	0	1	-
0	ϵ	-	1
1	ϵ	ϵ	$\emptyset$
-	ϵ	$\emptyset$	ϵ

Table 3: Cube Difference Operator for a Single Bit

Observation 1 *The monoterms are generated by the ones of the cube and zeros being replaced by DCs and then alternately replacing the positions of zeros with ones and DCs going through all the possible combinations. If the number of zeros in a cube is denoted by N_0, the total number of monoterms representing that cube will be 2^{N_0} as this is the total possible number of combinations. This can be proven based on the fact that $\bar{x}$ can be represented by $1 \oplus x$. Any negative variable then contributes two terms to the expansion.*

Theorem 1 gives the monoterms representing the cube at polarity 11...1 or the *RMC*. In order to obtain the representation for any other polarity, first a matching operation of the original cube and the polarity cube is performed to find the equivalent *RMC* representation. Then Theorem 1 is used to generate the monoterms of this equivalent cube. Finally the matching operation is performed on each monoterm generated to get *CGRM* form of interest. The matching operation here is the bitwise equivalence operation. Table 1 shows this operation for a single bit.

When a function is comprised of more than one cube, the monoterms that represent this function are not the simple union set of the monoterms of each cube but only those that occur in an even number of cubes. These will be referred to as *expansion* monoterms from now on.

To formulate the process of *CGRM* generation, analogies with set theory will be used. For two cubes, the expansion monoterms are those that are not common in the two cubes. For this purpose the *commonality* of two cubes needs to be identified. The cube commonality is a cube representing the monoterms that are common in two cubes and is given below:

Definition 2 *Let C_1 and C_2 be two cubes. The* cube commonality *operator on C_1 and C_2 is defined as follows:*

$$C_1 \Gamma C_2 = \begin{cases} \emptyset & \text{if } \exists i \text{ such that } C_{1i} \Gamma C_{2i} = \emptyset \\ C_3 & \text{otherwise, where } C_{3i} = C_{1i} \Gamma C_{2i} \end{cases} \tag{3}$$

where C_{ki} represents the i-th literal of the cube C_k, and the commonality operator for a single bit is defined in Table 2.

By the same analogy with set theory, the expansion monoterms of two disjoint cubes are the result of the *symmetric difference* of the two. The *difference* of two cubes is the set of all the monoterms which represent the first cube and do not represent the second. This operation which is the counterpart of the *sharp* operation for the case of monoterms is given below:

Definition 3 *Let C_1 and C_2 be two cubes. The* cube difference *operator of C_1 and C_2, $C_1 - C_2$, is:*

$$\begin{cases} C_1 & \text{if } C_{1i} - C_{2i} = \emptyset \text{ for some } i; \\ \emptyset & \text{if } C_{1i} - C_{2i} = \epsilon \text{ for all } i = 1, 2, \ldots, n \\ \bigcup_i & \text{if } (C_{11}, C_{12}, \ldots, \alpha_i, \ldots, C_{1n}) \text{ otherwise,} \\ & \text{where the } \bigcup \text{ is for all those } i \text{ for which} \\ & C_{1i} - C_{2i} = \alpha_i \in \{-, 1\} \end{cases} \tag{4}$$

where C_{ki} is the i-th literal of the cube C_k and the literal difference operator is defined in Table 3.

Definition 4 *Let C_1 and C_2 be two cubes. The* symmetric difference *of the two cubes is:*

$$C_1 \oplus C_2 = (C_1 - C_2) \bigcup (C_2 - C_1). \tag{5}$$

Difference and symmetric difference follow the same properties as in set theory. As an example, the symmetric difference is commutative and associative:

$$C_1 \oplus C_2 = C_2 \oplus C_1; \tag{6}$$

$$(C_1 \oplus C_2) \oplus C_3 = C_1 \oplus (C_2 \oplus C_3) = C_1 \oplus C_2 \oplus C_3. \tag{7}$$

The associativity of symmetric difference, given in Equation (7), makes it possible to extend this operation to more than two cubes. That is, the expansion monoterms of a switching function, given as a set of disjoint cubes, are obtained from the n-argument symmetric difference of all the cubes, i.e. $C_1 \oplus C_2 \oplus \cdots \oplus C_n$.

The process of realization of *CGRM* expansion for a given polarity can now be outlined. First, the function is represented as a set of disjoint cubes. The cubes are then operated by the equivalence operation with the polarity of the *CGRM*. The symmetric difference of all these cubes is found and monoterms representing each of the resulting cubes are generated in a Gray-code order. Finally, the Equivalence operation with the polarity cube is performed on each of these monoterms to give the *CGRM* expansion.

The number of the resulting monoterms can be found from the inclusion-exclusion principle. This number is given by the following theorem:

Theorem 2 *Let $C_1, C_2, \ldots, C_n$ be a set of n disjoint cubes. Let S_k denote the sum of the number of all monoterms common in all possible k cubes. The number of monoterms representing the set of cubes is:*

$$S_1 - 2S_2 + 4S_3 - 8S_4 + \cdots + (-2)^{k-1}S_k + \cdots + (-2)^{n-1}S_n. \quad (8)$$

In the above equation, S_1 denotes the number of all monoterms that are only in one cube, S_2 denotes the number of all monoterms that are common in any two cubes, etc.

5 Exact Minimization

Although identification of the minimal polarity is an NP-hard problem, certain features of the array of disjoint cubes can be used to reduce the required search space. Some of these features, if present in an array, can be used to find the minimal polarity without any search. Others can just reduce the amount of search needed to find the exact solution. In the case that none of the features exist, the whole exhaustive search needs to be performed.

The number of expansion monoterms for a set of disjoint cubes is the difference between the total sum of the number of monoterms representing each cube and the number of monoterms that are subtracted because they occur in an even number of cubes. Both of these numbers change with different polarities. The minimal polarity is the one which results in the optimum balance between these two numbers resulting in the least number of expansion monoterms.

There are certain features of the function that can be used to reduce the search space for identifying the minimal polarity. For the case of functions that are comprised of only one cube, the minimal polarity can be found directly without any search.

Theorem 3 *The minimal polarities for a single cube are the polarities which match all the literals in the cube. The number of such polarities is equal to $2^{N_{DC}}$ where N_{DC} is the number of DC-literals in the cube.*

When a function is comprised of more than one cube, there are other features that if they exist, lead to the search space reduction. Theorem 4 provides one criteria for identifying a minimal polarity literal based on the columns of an array of disjoint cubes, using Theorem 2.

Theorem 4 *Let ${S^1}_i$ denote the sum of S_is of the cubes which have a value of 1 in a given column. Let ${S^0}_i$ denote the sum of S_is of the cubes which have a value of 0 in that column. Let ${S^{1-DC*}}_i$ denote the sum of S_is of the cubes that have both 1s and DCs in the column, assuming the 1 has been changed to a 0. Let ${S^{0-DC}}_i$ denote the sum of S_is of the cubes that have both 0s and DCs in the column. The corresponding minimal polarity literal for a column in the array of disjoint cubes should be changed when*

$$\sum_{k=1}^{n}(-2)^{k-1}S_k^1 + \sum_{k=2}^{n}(-2)^{k-1}S_k^{1-DC*} \quad (9)$$

$$< 1/2\sum_{k=1}^{n}(-2)^{k-1}S_k^0 + \sum_{k=2}^{n}(-2)^{k-1}S_k^{0-DC}.$$

From Theorem 4, it is possible to infer the following theorem:

Theorem 5 *For a column comprised of all 0s or all 1s, the corresponding minimal polarity literal is the same as the value in the column. (If the opposite is chosen, the number of monoterms representing the cubes would be doubled.) For a column comprised of all DC values, either 0 or 1 will be the minimal literal value.*

In the Exact method of minimization, first it is checked if the function is only comprised of one cube or two. Direct solution for these cases is found using Theorems 3, 4, and 5. If there are more cubes involved, first Theorem 5 is used to identify any columns in the array of disjoint cubes for which minimal polarity literal can be found readily. All the other columns are set to the zero polarity and a search for minimal polarity is performed in Gray-code order, changing one column at a time. This method is given formally below:

```
The exact algorithm
CGRMIN_EXACT
{
 If ( the function is comprised of one or two cubes )
 {
   Preset the columns to the minimal polarity;
   Generate the CGRM of the array; }
 else {
   Preset the columns
     to minimal polarity literals if possible, or
     to polarity 0 otherwise;
   for ( all combinations of the columns with
         undetermined polarities )
   {
     Set one column to its opposite polarity using
     Gray-code order;
     Generate the CGRM of the array;
     if ( No. of monoterms decreases )
         min_polarity = polarity_of_current_CGRM;
   }
 Generate the CGRM of min_polarity; }
}
```

6 Heuristic Minimization

In this section, a fast heuristic approach to the minimization problem is introduced. The corresponding heuristics combine the characteristics of the the overall

number of monoterms and the ones subtracting, in order to identify the minimal *CGRM* polarity for a given array of disjoint cubes. Based on these heuristics, a priority of search for different polarities is devised and a minimization algorithm is introduced.

Here, similar to the Exact method, the columns minimal polarities of which can be identified without any search are first pre-set to their minimal polarities. Next, for all the remaining columns in the array of disjoint cubes, the polarity literals are chosen such that there will be the least number of 0s in each column. Referring back to Table 1, the least number of 0s in a column results from a polarity which is equivalent to the value that occurs the most in that column. That is, there will be the least number of non-matching values in that column with its corresponding polarity literal. Since the overall number of monoterms is determined by the number of zeros in the cubes, this polarity is used as a starting point for the minimization. However, the minimum number of 0s in the array does not guarantee the minimum number of expansion monoterms. For many functions it was shown experimentally that this first choice gives the minimal solution. For other cases, however, a search scheme is required. The general heuristics below can be used in any minimization scheme:

- *As the number of monoterms for a cube is 2^{N_0}, where N_0 is the number of 0s in the cube, if a cube has a relatively large number of 0s, some polarity literals should be changed to reduce the contribution of this cube to the number of expansion monoterms.*
- *From (9), if the number of occurrences of a 1 or a 0 in a column with no DC values is much higher than its opposite value, it is more likely that this value will be the minimal polarity literal for that column. A ratio of 4 to 1 for the most occurring and least occurring values almost guarantees this likelihood.*
- *When there are DC values involved in a column, the greater the number of the DC values, the less will be the overall number of monoterms affected if the polarity of that column is changed.*
- *For functions with large number of cubes, the monoterms which are common in two or three cubes are the main subtracting monoterms.*

Here, based on the above heuristics, a priority is calculated for every column minimal polarity of which can not be found without search. This priority is used to guide the search towards the minimal polarity.

Let the *priority* of the column i be represented by Γ_i. Let the number of the 1s in that column be represented by $N^1{}_i$, and the number of 0s by $N^0{}_i$. Then

$$\Gamma_i = \frac{(N^1{}_i)^2}{N^1{}_i + N^0{}_i} \qquad (10)$$

The Quasi-minimal method can now be presented formally as follows:

```
The quasi-minimal algorithm
CGRMIN
{
  Preset the columns
     to minimal polarity literals if possible, else
     to most occurring values in the columns;
        Find the values of column priorities, Γs ;
  Sort Γs in descending order;
  for (k = 1; k <No_of_undetermined_polarities; k++)
  {
     Set the column with priority k to its opposite
        polarity;
     Generate the CGRM of the array;
     if ( No. of monoterms representing the array or
        the overall number of monoterms decreases );
     {
        min_polarity = polarity_of_current_CGRM;
        continue;
     }
     else
        change the column back to its previous polarity;
  }
  Generate the CGRM of min_polarity;
}
```

In this algorithm, the number of searches required for the worst case will be of order $2n$, for a function of n variables which significantly contributes to its speed.

7 Experimental Results

The Exact and Quasi-Minimal methods were tested for 112 MCNC benchmark functions. As most of the benchmarks are multi-output, the BLIF format of the functions was used to generate single output components of these functions for testing. Several of these functions are shown for comparison in Table 4. In this table n stands for the number of variables in the functions and No. for the number of terms.

The times given in the table are the CPU times in seconds on a Sequent S27 - two 386 processors - machine. The table includes the time comparisons of the functions for MINGRM [7], an existing exact *CGRM* minimizer, ESPRESSO, and CGRMIN, the *CGRM* minimizer developed based on the algorithms described earlier.

For the functions tested, the two level AND/OR and *CGRM* were found to be relatively comparable. For the 112 functions overall, while ESPRESSO found 1193 terms, this number for exact *CGRM* was found to be 1382 and the quasi-minimal program gave 1557. Depending on the functions, AND/OR can be a better choice for realization while for others it is the *CGRM* that certainly gives the better alternative. The functions rd532, rd732, and rd844 in Table 4 are examples of the latter; mis20, mis59, and sao22 are examples of the former. EXORCISM [1] gives 102 terms for 9sym, 5 for con12 and rd532, 9 for f51m4, 22 for sao23, and 16 for misex20. It is obvious that classifying functions according to their linearity and using multi-level AND/OR/XOR minimizers will definitely

Name	n	ESPRESSO		MINGRM		CGRMIN	
		No.	sec	No.	sec	No.	sec
5xp11	7	7	0.1	12	5.6	12	0.0
9sym	9	85	3.9	173	1851.3	173	10.3
bw7	5	6	0.0	8	0.4	12	0.0
con12	7	5	0.0	8	1.2	8	0.0
duke8	22	5	0.0	4	*	6	0.0
f51m4	8	10	0.1	7	4.5	9	0.1
rd532	5	16	0.1	5	1.1	5	0.1
rd732	5	64	0.9	7	49.6	7	1.5
rd842	8	128	3.1	8	364.2	8	5.7
sao22	10	20	0.2	52	642.1	61	0.4
sao23	10	22	0.9	47	905.9	59	0.5
mis20	10	7	0.0	62	309.6	66	0.2
mis59	12	4	0.0	26	1166.1	26	0.0
vg28	25	5	0.0	13	*	13	0.0
z42	7	28	0.4	9	12.3	13	0.4

Table 4: Two Level AND/OR Compared to Two Level *CGRM*

lead to more economical realizations of switching functions.

The differences between the minimal and quasi-minimal results demonstrate the quality of heuristics used. For majority of the functions, 66 out of 112 to be precise, the heuristic program found the exact minimum solutions. There were only 6 functions that differed by more than 10 terms; the largest being 12 terms for the function sao23. As indicated in the table, CGRMIN is extremely fast and overall with the results presented, it demonstrates the high quality of the approach.

8 Conclusions

In this paper fast Exact and Quasi-Minimal methods for minimization of fixed polarity AND/XOR canonical forms were presented. First, a fast cube-based method for the generation of these forms was introduced which can handle functions with large numbers of variables. In addition, certain features of switching functions represented as arrays of disjoint cubes were presented. It was shown that if these features are present in the function, they can significantly reduce the minimization task. Based on these features, exact and heuristic algorithms were introduced, implemented, and tested with 112 single output functions created from the MCNC benchmark functions. These methods give faster results than any method published previously. The Quasi-Minimal method proved to be based on high quality heuristics. Out of 112 functions, it gives the exact solution for 66 while only 6 deviate by 10 or more terms, the highest being 12 terms.

The results presented confirm also the merits of the AND/XOR networks. By identifying XOR contents of the Boolean functions, one can create more compact realizations of these functions. This study can be used for classification of functions and multi-level AND/OR/XOR and other XOR realizations. The fast programs introduced can be used to bring more XOR utilization into the realm of logic synthesis. This research will also be expanded into multi-output, incompletely specified, multi-valued functions.

* stands for unavailable data.

References

[1] M. Helliwell and M. A. Perkowski, "A Fast Algorithm to Minimize Multi-Output Mixed-Polarity Generalized Reed-Muller Forms", *Proc. of the 25th ACM/IEEE Design Automation Conference*, pp. 4427-432, 1988.

[2] T. Sasao, and Ph. Besslich, "On the Complexity of MOD-2 Sum PLAs", *IEEE Trans. on Computers*, Vol. 39, No. 2, pp. 262-266, 1990.

[3] "Concurrent Logic, Inc.", *CLi 6000 Series Field Programmable Gate Arrays*, Preliminary Information, Dec 1991, Rev 1.3.

[4] M. Davio, J. P. Deschamps, and A. Thayse, *Discrete and Switching Functions*, Mc-Graw-Hill, 1978.

[5] Ph. W. Besslich and M. W. Riege, "An Efficient Program for Logic Synthesis of Mod-2 Sum Expressions" *Euro ASIC'91*, pp. 136-141, Paris, France, 1991.

[6] L. T. Fisher, "Unateness Properties of AND Exclusive OR Logic Circuits", *IEEE Trans. on Computers*, Vol. C-23, No. 2 pp. 166-172, 1974.

[7] I. Schäfer and M. A. Perkowski, "Multiple-Valued Input Generalized Reed-Muller Forms", *Proc. of ISMVL'91*, pp. 40-48, May 1991.

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

[9] I. S. Reed, "A Class of Multiple-Error-Correcting Codes and Their Decoding Scheme", *IRE Trans. on Inf. Theory*, Vol. PGIT-4, pp. 38-49, 1954.

[10] D. E. Mueller, "Application of Boolean Algebra to Switching Circuit Design and to Error Detection", *IRE Trans. on Elec. Computers*, Vol. EC-3, pp. 6-12, September 1954.

[11] S. M. Reddy, "Easily Testable Realization for Logic Functions", *IEEE Trans. on Computers*, Vol. C-21, No. 11, pp. 1183-1188, 1972.

[12] D. K. Pradhan, "Universal Test Sets for Multiple Fault Detection in AND-EXOR Arrays", *IEEE Trans. on Computers*, Vol. C-27, No. 2, pp. 181-187, 1978.