EXORCISM-MV-2: Minimization of Exclusive Sum of Products Expressions for Multiple-Valued Input incompletely Specified Functions

Ning Song, and Marek A. Perkowski

Portland State University, Department of Electrical Engineering
P.O. Box 751, Portland, OR, 97207, tel. (503) 725-5411

Abstract

A new cube operation, exorlink, is presented, as well as its application to the minimization of multiple-valued input, multi-output Exclusive Sums Of Products (ESOPs) for incompletely specified Boolean functions. Exorlink generalizes all cube operations such as crosslink, unlink or X-merge introduced by Helliwell, Perkowski, Sasao and other authors. Our program, EXORCISM-MV-2, gives efficient results for functions that are incompletely specified and have an arbitrary number of values for each of the input variables. This allows to realize a wider class of circuits which implement the multiple-valued input ESOP expressions. Evaluation on benchmark functions is also given and proves the superiority of the program to those known from the literature.

1. Introduction

There is recently an increased interest in the design of logic circuits which use EXOR gates[1 , 2 , 3 , 4 , 5 , 6 , 7 , 8]. Functions realized by such circuits can have fewer gates, fewer connections, and take up less area in VLSI and especially, FPGA realizations. They are also easily testable[9]. Circuits of this type find applications in self-testing schemes, linear machines, arithmetic and communication circuits, encrypting schemes, coding schemes for error control and synchronization, sequence generation for process identification, system testing, etc.

Why are then ESOPs not as popular as their Sum of Products (SOP) counterparts? One of the main reasons it that the problem of the minimization of ESOP circuits is difficult. Papakonstantinou [10] gave exact algorithm for 4 input functions. The algorithm from [6] has theoretically no limits on the number of variables but 4 is its practical limit. Since exact solutions can be practically found only for functions with not more than 5 variables the interest is in approximate solutions. The heuristic computer programs have been presented in[1 , 4 , 5 , 7 , 8]. Efficient programs for sub-families of ESOPs were given in [3]. The paper [4] presented an algorithm based on crosslink operations. The algorithm from [4] was next improved in [5], and also extended for the case of the logic with multiple-valued inputs. Unlink operation has been also added. Unlink operation was efficiently implemented in [8]. Few more operations were also included in an independent realization by Sasao [7], which is the only other author that published on the most general ESOP minimization algorithms for the multiple-valued input logic.

Here, we present a solution to a problem that has not yet been practically solved in the literature: efficient minimization of arbitrary ESOP expressions for multiple-output multiple-valued input incompletely specified functions. This paper describes an approximate method that yields especially good results for the minimization of strongly unspecified multi-output logic functions with multiple-valued inputs and binary-valued outputs. We are the first authors who provide documented results of ESOP minimization for r-bit decoders with $r > 2$. Experimental results for binary, 4-valued inputs, and 8-valued inputs will be shown. For convenience, examples with 4-valued inputs will be presented in most of the cases. Our approach is a generalization and a continuation of the line of research from [4 , 5 , 7] and [8].

In Section 2, definitions and terminologies are given. Our new cube operation -- multiple-valued exorlink operation is introduced in Section 3. In Section 4, our algorithm for ESOP minimization using exorlink is discussed. The experimental results are evaluated in Section 5. The conclusion is given in Section 6.

2. Definitions and Basic Properties

A multiple-valued input, two-valued output, incompletely specified switching function f (*multiple-valued function*, for short) is a mapping $f(X_1, X_2, \cdots, X_n)$: $P_1 \times P_2 \times \cdots \times P_n \rightarrow B$, where X_i is a *multiple-valued variable*, $P_i = \{0, 1, \ldots, p_i - 1\}$ is a *set of truth values* that this variable may assume, and $B = \{0, 1, -\}$ (-

This research was partially supported by the NSF grand MIP9110772

denotes a *don't care value).* This is a generalization of an ordinary n-input switching function f: $B^n \rightarrow$ B.

Definition 1. For any subset $S_i \subseteq P_i$, $X_i^{S_i}$ is a *literal* of X_i representing the function such that

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

A 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* or *product* for short). In a product term, all the operations are AND operations. A product term that includes literals for all variables $X_1, X_2, \ldots, X_n$ is called a *full term.* A sum of products is denoted as a *sum-of-products (SOP)* expression while a product of sums is called a *product-of-sums (POS)* expression. An EXOR of products will be called a *Multiple-Valued Input Exclusive Sum of Products Form (MIESOP* for short).

An r-bit decoder has r inputs and 2^r outputs. Each output represents one value of 2^r-valued logic. For instance, for r=3 each output bit represents one of 8 values of 8-valued logic.

Switching functions with multiple-valued inputs and two-valued outputs, find several applications in logic design, pattern recognition, and other areas. In logic design, they are primarily used for the minimization of PLAs that have 2-bit decoders or function generators on the inputs. A Programmable Logic Array (PLA) with r-bit decoders directly realizes a SOP of a 2^r-valued input two-valued output function [11]. An *EXOR-based PLA* is a PLA in which EXOR gates replace OR gates in the OR plane. An *EXOR-based Programmable Logic Array (EXPLA)* with r-bit decoders directly realizes a MIESOP of a 2^r-valued input two-valued output function [11].

Now let us consider a Boolean function F with *multiple output,* that is, $F(X_1, \ldots, X_{n-1}) = (f_0, \ldots, f_{m-1})$, we define a single-output switching function F on n variables where variables $X_1, \ldots, X_{n-1}$ are multiple-valued and the variable X_n takes the values $\{0, \ldots, m-1\}$. $F(X_1, \ldots, X_{n-1}, X_n=i) = F_i\ (X_1, \ldots, X_{n-1})$ where $F_i(X_1, \ldots, X_{n-1})$ denotes the i-th *projection* of $F(X_1, \ldots, X_{n-1}, X_n)$, that is, f_i. Therefore, only single-output switching functions will be considered and n will denote the number of input variables.

When our method is used for a completely specified function it uses an *array ON of arbitrary disjoint cubes (product terms,* ON-cubes, cubes of minterms). These can be minterms, full terms, or any disjoint product implicants. In the case of an incompletely specified function the function is represented as the *array ON of disjoint ON-cubes* and the *array DC of disjoint DC-cubes (DC-cubes are cubes of don't cares).*

Our primary goal of MIESOP synthesis is to *minimize the number of terms.* For the circuit with the minimum number of terms our secondary goal is to minimize the total number of inputs to AND and EXOR gates. Therefore the *cost function* C to be minimized by our algorithm is as follows:

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

where:

- NT is the total number of terms in the solution,
- NI is the total number of input wires to AND and EXOR gates in the solution,
- NI_{in} is the total number of input wires to AND and EXOR gates in the initial function.

For instance, literal X^{012} as an input to an AND gate requires a single wire for the 2-by-4 decoder realization of logic with 4-valued inputs. X^{01} is realized as $X^{012} X^{013}$. It, therefore, requires two wires. Similarly $X^0 = X^{012} X^{013} X^{023}$ requires three wires. $X^0 Y^1$ requires six wires. $X^0 Y^1 \oplus X^1 Y^2$ have two terms and requires fourteen wires (twelve to the AND gates and two to the EXOR gate).

According to the cost function, if two solutions have different number of terms, the better solution is the one which has less number of terms, because its NT is smaller. If two solutions have the same number of terms, the better solution is the one which has less number of inputs, because its $\frac{NI}{NI_{in}}$ is smaller.

A switching function with multiple-valued inputs is uniquely determined by its *truth table,* or by an expression given in SOP or POS. A multiple-valued input, two-valued output function will be simply called a *function* from now on. Any literal of the form $X_i^{P_i}$ is identically equal 1. Hence, we often write $X_i^{P_i} X_j^{S_j}$ as $X_j^{S_j}$.

Let A, B, C denote any values of multiple-valued input literals, or any functions on them. The following operations hold for multiple-valued input algebra:

1. Associative laws: $A \oplus (B \oplus C) = (A \oplus B) \oplus C$.
2. Commutative laws: $A \oplus B = B \oplus A$.
3. Identities:

   3A. $A \oplus A = 0$;

   3B. $X_i^{S_i} \oplus 1 = X_i^{P_i - S_i}$;

   3C. $X_i^{S_i} \oplus X_i^{P_i - S_i} = X_i^{P_i} = 1$;

   3D. $X_r^{S_i} \oplus X_r^{S_j} = X_r^{(S_i - S_j) \cup (S_j - S_i)}$;

   3E. $X_i^{S_i} X_j^{S_j} \oplus X_i^{R_i} X_j^{R_j}$

   $= X_i^{(S_i - R_i) \cup (R_i - S_i)} X_j^{S_j} \oplus X_i^{R_i} X_j^{(S_j - R_j) \cup (R_j - S_j)}$

   $= X_i^{(S_i - R_i) \cup (R_i - S_i)} X_j^{R_j} \oplus X_i^{S_i} X_j^{(S_j - R_j) \cup (R_j - S_j)}$

Definition 2. The *distance of two cubes* is the number of variables for which the corresponding literals have disjoint sets of truth values (in other words - the number

of variables for which the sets S_i and R_i are *disjoint*).

For instance, the distance of cubes $X^{01}Y^{02}Z^{123}$ and $X^{12}Y^{1}Z^{13}$ in the four-valued algebra is 1, since sets {0, 1} and {1, 2} of variable X are non-disjoint, so as the sets {1, 2, 3} and {1, 3} of variable Z. The sets {0, 2} and {1} of variable Y are disjoint.

Definition 3. The *difference of two cubes* is the number of variables for which the corresponding literals have different sets of truth values (in other words - the number of variables for which the sets S_i and R_i are *different*).

For instance, the difference of cubes $X^{01}Y^{02}Z^{123}$ and $X^{12}Y^{1}Z^{13}$ in the four-valued algebra is 3, since for all three variables, their corresponding literals have different sets of truth values.

3. The Multiple-Valued Exorlink Operation

3.1. The formula.

Basic operation of our system is the operation of *exorlink*. This operation generalizes the *xlinking[4] and unlinking[5] operations* introduced by us, as well as the operations introduced by T. Sasao in [7].

Definition 4. Let $C_S = X_1^{S_1} \ldots X_n^{S_n}$ and $C_R = X_1^{R_1} \ldots X_n^{R_n} \neq C_S$ be two cubes. Let X_{j_i}, i = 1, ..., r be the variables for which $S_{j_i} \neq R_{j_i}$. Let X_{l_k}, k = 1, ..., n - r be a variable for which $S_{l_k} = R_{l_k}$.

The *exorlink* of cubes C_S and C_R is defined by the following formula:

$$C_S \otimes C_R = \bigcup_{i=1}^{r} [\, (X_{j_1}^{S_{j_1}} \cdots X_{j_{i-1}}^{S_{j_{i-1}}} X_{j_i}^{(S_{j_i} - R_{j_i}) \cup (R_{j_i} - S_{j_i})} X_{j_{i+1}}^{R_{j_{i+1}}} \cdots X_{j_r}^{R_{j_r}})(\prod_{k=1}^{n-r} X_{l_k}^{S_{l_k}}) \,]$$

To find the exorlink of a pair of two full terms, for example $A^{01}B^{02}C^{012}D^{2}E^{13}$ and $A^{12}B^{12}C^{2}D^{2}E^{13}$, in a 4-valued function, we write them vertically like this:

$A^{01}\ B^{02}\ C^{012}\ D^{2}\ E^{13}$

$A^{12}\ B^{12}\ C^{2}\ D^{2}\ E^{13}$

$\uparrow\ \ \uparrow\ \ \uparrow$

Each time when the literals have different set of values from full term to full term in the pair it is denoted by an arrow. Let us now consider each arrow separately. The above initial pair of full terms can then be expanded to three *resultant terms*, for variables A, B, and C respectively, as shown below.

For variable A, the term $A^{02}\ B^{12}\ C^{2}\ D^{2}\ E^{13}$ is created as follows:

$A^{01}\ B^{02}\ C^{012}\ D^{2}\ E^{13}$

$A^{12}\ B^{12}\ C^{2}\ D^{2}\ E^{13}$

$A^{02}\ B^{12}\ C^{2}\ D^{2}\ E^{13}$ ← the first resultant term

For variable B, the term $A^{01}\ B^{01}\ C^{2}\ D^{2}\ E^{13}$ is created as follows:

$A^{01}\ B^{02}\ C^{012}\ D^{2}\ E^{13}$

$A^{12}\ B^{12}\ C^{2}\ D^{2}\ E^{13}$

$A^{01}\ B^{01}\ C^{2}\ D^{2}\ E^{13}$ ← the second resultant term

For variable C, the term $A^{01}B^{02}\ C^{01}\ D^{2}E^{13}$ is created as follows:

$A^{01}B^{02}\ C^{012}\ D^{2}E^{13}$

$A^{12}\ B^{12}\ C^{2}\ D^{2}E^{13}$

$A^{01}B^{02}\ C^{01}\ D^{2}E^{13}$ ← the third resultant term

Under each pair of literals of different sets of truth values under consideration (A in the first pair, B in the second, C in the third pair), we write the multiple-valued literal, with the set of truth values being the exclusive-sum of the respective sets from the literals of the cubes. To create the result of exorlink for a resultant term, we copy the part of the term to the left of the arrow from the top full term. The part to the right of the arrow is copied from the bottom full term as shown. The exorlink of the initial pair of full terms is an EXOR of resultant terms. Therefore

$$A^{01}B^{02}C^{012}D^{2}E^{13} \quad \oplus \quad A^{12}B^{12}C^{2}D^{2}E^{13} \quad = \\ A^{02}B^{12}C^{2}D^{2}E^{13} \oplus A^{01}B^{01}C^{2}D^{2}E^{13} \oplus A^{01}B^{02}C^{01}D^{2}E^{13}$$

This procedure can easily be further extended for any two terms in which for every two correspondingly different literals for the same variable: $X_i^{S_j}$ and $X_i^{R_j}$ the sets S_j and R_j are different.

Definition 5. Given cubes C_S and C_R, if the difference of the two cubes is r, we call $C_S \otimes C_R$ *difference r exorlink*. If the distance of the two cubes is d, we call $C_S \otimes C_R$ *distance d exorlink*.

Our approach uses specific exorlink variant rules for difference value from 0 to 3 and distance value from 0 to 3. Since the difference value cannot be smaller than the distance value, there are 10 variant exorlink operations, part of which are described by formulas from section 2. For instance, 3A corresponds to difference 0, distance 0 exorlink; 3B corresponds to difference 1, distance 0 exorlink; 3C corresponds to difference 1, distance 1 exorlink; 3D corresponds to difference 1 exorlink; 3E corresponds to difference 2 exorlink.

Next, the difference 1, difference 2, and difference 3 exorlink are discussed.

3.2. Difference 1 exorlink

Difference 1 exorlink generates one resultant cube. Let us look at the following examples which assume 4-valued input logic:

$X^{013}\ Y^{23}$	$X^{123}\ Y^{23}$	$X^{01}\ Y^{23}$
$X^{23}\ Y^{23}$	$X^{23}\ Y^{23}$	$X^{23}\ Y^{23}$
---	---	---
$X^{012}\ Y^{23}$	$X^{1}\ Y^{23}$	$X^{0123}\ Y^{23} = Y^{23}$

This operation is equivalent to X-MERGE [7].

3.3. Difference 2 exorlink.

Difference 2 exorlink generates two resultant cubes:

$X^{013}\ Y^{13}$	and	$X^{013}\ Y^{13}$
$X^{23}\ Y^{01}$		$X^{23}\ Y^{01}$
---		---
$X^{012}\ Y^{01}$		$X^{013}\ Y^{03}$

So, $X^{013}\ Y^{13} \otimes X^{23}\ Y^{01} = X^{012}\ Y^{01} \oplus X^{013}\ Y^{03}$. It can be observed that difference 2 exorlink of cubes C_S and C_R is different from the difference 2 exorlink of cubes C_R and C_S.

- If $S_i \cap R_i = \phi$ and $S_j \cap R_j = \phi$, the operation is equivalent to distance 2 primary xlink of [5] and X-EXPAND-1 of [7].
- If $S_i \supset R_i$ and $S_j \cap R_j = \phi$, the operation is equivalent to distance 1 secondary xlink of [5] and X-EXPAND-2 of [7].
- If $S_i \subset R_i$ and $S_j \cap R_j = \phi$, the operation is equivalent to RESHAPE of [7].
- If $S_i \supset R_i$ and $S_j \subset R_j$, the operation is equivalent to DUAL-COMPLEMENT of [7].
- If $S_i \subset R_i$ and $S_j \supset R_j$, the operation is equivalent to primary unlink of [5] and X-REDUCE-1 of [7].
- If $S_i \supset R_i$ and $S_j \supset R_j$, the operation is equivalent to secondary unlink of [5] and X-REDUCE-2 of [7].
- If $S_i \cap R_i \neq \phi$ and $S_j \cap R_j \neq \phi$, no compatible operation can be found in the previous literature.

3.4. Difference 3 exorlink

The example given in Section 3.1 is an example of difference 3 exorlink operation. Difference 3 exorlink generates three resultant cubes. So, one more term will be generated each time when this operation is performed. This seems contradictory to our primary goal: minimizing the number of terms. However, it is proved in [2], that increasing the number of terms may help to reduce the number of terms at the later stage and get better results. Our experimental results show that it is beneficial to include this operation in the algorithm. If $S_i \supset R_i$, $S_j \cap R_j = \phi$ and $S_k \cap R_k = \phi$, this operation is equivalent to distance 2 secondary xlink of [5].

4. Algorithm of EXORCISM-MV-2 program

Our program, EXORCISM-MV-2 is a new version of EXORCISM-MV presented in [5]. The algorithm currently used in EXORCISM-MV-2 was created after much experimentation with previous variants. Because our main goal is to minimize the number of cubes, all possible difference 1 exorlinks are executed. The pairs of equal cubes are removed and difference 1 exorlink operations are performed iteratively. Then difference 2 exorlink operations are executed which may provide opportunities for difference 1 exorlink. If no more difference 1 exorlink is possible, difference 3 exorlink operations are performed, which may provide opportunities for difference 1 exorlink.

If no further difference 1 exorlink operation are possible, difference 2 exorlink operations are performed to minimize the connections. For incompletely specified functions, the ON set is minimized first. Then sharp operation [12] is executed between each cube i in the ON set and the cubes in the don't care set. If cube i is empty, it will be removed from the ON set. If no more cubes in the ON set can be sharped out, difference 2 exorlink operations are performed in order to provide further opportunities. Next, each variable of cube i in F is masked, and sharp is performed between cube i and the cubes in the don't care set. Let us denote the resultant cube by j. If the complement of cube j is the same as cube i except the masked variable, difference 1 exorlink operations between cubes j and i are performed. Successful application of these operations decreases the numbers of connections.

Input: Arrays ON and DC of disjoint cubes for a multi-valued input function.

1. F := ON; D := DC. SOLUTION := F, MIN_COST := COST(F). (MIN_COST will be updated in below steps to reflect always the lowest cost of solutions obtained until now. This solution is also stored).
2. Sorting the F array in ascendent order according to the number of literals in each cube. The cube with smallest number of literals is in the top of the array after sorting.
3. From the top to the bottom of the array, check the difference of each pair of cubes. Remove a pair of cubes if they are identical. Perform difference 1 exorlink if possible.
4. If operations in 3 were successful, go to 3.

5. Perform difference 2 exorlink
6. Compute the cost function C.
7. Repeat steps 2 to 6 as long as the reduction of NT of the cost function C is possible.
8. Perform difference 3 exorlink operations.
9. Repeat steps 2 to 6 as long as the reduction of NT of the cost function C is possible.
10. Repeat steps 7 to 8 as long as the reduction of NT of the cost function C is possible.
11. Perform difference 2 exorlink operations.
12. If the option "don't care" is not selected, print the output and stop the program, else go to 13.
13. Perform sharp operation between each cube i in F and all cubes in D.
14. Perform difference 2 exorlink operations on F.
15. Compute the cost function C.
16 Repeat steps 13 to 15 as long as the reduction of cost function C is possible.
17. Mask each variable for each cube i in F and perform sharp operation between it and D (this corresponds to various expansions of this cube). Name the resultant cube j. If the cube i and the complement of cube j are identical, except the masked variable, perform difference 1 exorlink between them.
18. Compute the cost function C.
19. Repeat steps 3-18 as long as the reduction of cost function C is possible.
20. Print the output and stop the program.

5. Evaluation of Results of EXORCISM-MV-2

Exorcism-mv-2 was tested on several benchmarks, as illustrated in Tables 1, 2, 3 and 4. All the benchmarks are run on Sparc Station I. Table 1 presents, for each function, the total number of resultant cubes (i.e. product terms in the solution), the number of inputs to AND gates in the solution, the number of inputs to EXOR gates in the solution, and the user time. Table 2 and Table 3 does the same for 2-bit and 3-bit decoders respectively. Table 4 shows the results for MCNC benchmarks. Table 5 compares our results with those of Exmin [7].

6. Conclusion

An improved cube operation and algorithm for MIESOP minimization along with the efficient program to minimize such forms have been introduced. No algorithms have been published until now for MIESOP solutions to multi-output and incompletely specified functions.

Table 1

	1 bit decoder			
	cube	AND	EXOR	Time
ADR4	31	114	40	9.3
LOG8	95	508	192	332.9
MLP4	62	301	86	47.8
NRM4	67	375	131	64.1
RDM8	31	110	42	10.7
ROT8	37	200	62	8.1
SQR8	113	535	219	348.6
WGT8	58	261	64	75.6

Table 2

	2 bit decoder			
	cube	AND	EXOR	Time
ADR4	11	46	14	9.7
LOG8	85	663	163	831.8
MLP4	51	380	75	158.5
NRM4	53	429	86	97.6
RDM8	26	152	38	19.9
ROT8	26	201	51	14.1
SQR8	93	646	154	822.8
WGT8	22	120	29	83.6

Table 3

	3 bit decoder			
	cube	AND	EXOR	Time
ADR4	10	50	15	10.0
LOG8	41	307	112	875.7
MLP4	29	174	54	163.4
NRM4	26	170	46	104.1
RDM8	19	102	37	22
ROT8	16	102	26	16.6
SQR8	66	474	267	910.9
WGT8	10	78	17	85.2

tions.

Exorcism-mv-2 was tested on many benchmark functions and compared to the top program Exmin [7] (see Table 5 for part of benchmarks). Our program give the same or better solutions on binary and 4-valued completely specified functions. More importantly, Exorcism-mv-2 is able to minimize efficiently arbitrary-valued and incompletely specified functions. Additionally, as in Espresso, the number of variables in our program is unlimited and the only constraint is the number of input cubes that are read, so very large functions can be minimized.

Table 4

	cube	AND	EXOR	Time
b12	28	123	40	4.2
bw	22	81	239	4.6
clip	66	380	111	68.9
con1	9	28	9	0.2
cps	141	1634	1270	2126.5
e64	65	2145	123	179.9
ex5	78	456	404	134.5
f51m	31	113	42	10.8
frg1	115	1300	116	272.9
inc	28	117	57	7.2
misex1	12	48	37	1.2
misex2	27	172	38	7.3
pdc	249	1871	469	2820.9
rd53	15	47	21	0.7
rd73	41	154	51	23.2
rd84	58	259	66	97.5
sao2	28	225	52	27.3
seq	248	3277	1565	3352.3
squar	19	50	28	1.9
table3	166	1848	658	362.6
table5	156	1860	604	476.9
t481	23	174	23	416.6
vg2	184	1807	193	240.9
xor5	5	5	5	0.1
5xp1	32	117	53	10.0
9sym	51	374	51	140.3

Table 5

	Exmin		Exorcism-mv-2	
	1 bit	2 bit	1 bit	2 bit
ADR4	32	12	31	11
LOG8	105	105	95	85
MLP4	66	56	62	51
NRM4	76	62	67	53
RDM8	31	28	31	26
ROT8	37	32	37	26
SQR8	121	121	113	93
WGT8	66	26	58	22

Acknowledgment

The authors would like to thank the Reviewer B for his very careful review of the paper.

References

1. Besslich, Ph. W. and M. W. Riege, "An Efficient Program for Logic Synthesis of Mod-2 Sum Expressions," *EUROASIC*, pp. 136-141, Paris, France, 1991.
2. Brand, D. and T. Sasao, "On the Minimiazation of And-Exor Expressions," *23rd IEEE Conference on Fault Tolerant Computing*, pp. 1-9, 1990.
3. Csanky, L., M. A. Perkowski, and I. Schäfer, "Canonical Restricted Mixed-Polarity Excluxive Sums of Products and the Efficient Algorithm for Their Minimization," *Proc. IEEE Intern. Symp. on Circuits and Systems*, pp. 17-20, San Diego, May 1992.
4. Helliwell, M. and M. A. Perkowski, "A Fast Algorithm to Minimize Multi-Output Mixed-Polarity Generalized Reed-Muller Forms," *Proc. 25th ACM/IEEE Design Automation Conference*, pp. 427-432, Anaheim, CA, June 1988.
5. Perkowski, M. A., M. Helliwell, and P. Wu, "Minimization of Multiple-Valued Input Mutli-Output Mixed-Radix Exclusive Sums of Products for Incompletely Specified Boolean Functions," *19th Int. Symp. on Multiple-Valued Logic*, pp. 256-263, May 1989.
6. Perkowski, M. A. and M. Chrzanowska-Jeske, "An Exact Algorithm to Minimize Mixed-Radix Exclusive Sums of Products for Incompletely Specified Boolean Functions," *Proc. of the ISCAS'90, International Symposium on Circuits and Systems*, pp. 1652-1655, New Orleans, May 1990.
7. 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*, pp. 128-135, May 1990.
8. Saul, J. M., "An Improved Algorithm for the Minimization of Mixed Polarity Reed-Muller Representations," *Int. Conf. on Comput. Design: VLSI in Comput. and Processors*, pp. 372-375, September 1990.
9. Fujiwara, H., "Logic Testing and Design for Testability," in *Computer System Series* , MIT Press, 1986.
10. Papakonstantinou, G., "Minimization of Modulo-2 Sum of Products," *IEEE Trans. on Comput.*, vol. 28, no. 2, pp. 163-167, February 1979.
11. Sasao, T., "Multiple-Valued Decomposition of Generalized Boolean Functions and the Complexity of Programmable Logic Arrays," *IEEE Trans. on Comput.*, vol. 30, no. 9, pp. 635-643, September 1981.
12. Dietmayer, D. L., in *Logic Design of Digital Systems*, Allyn and Bacon, 1978.