Flexible Optimization of Fixed Polarity Reed-Muller Expansions for Multiple Output Completely and Incompletely Specified Boolean Functions

Chip-Hong Chang, Bogdan J. Falkowski
School of Electrical and Electronic Engineering,
Nanyang Technological University,
Nanyang Avenue, Singapore 2263.

Abstract – A new algorithm that generates optimal fixed polarity Reed-Muller expansions based on user specified optimization criteria is shown. The algorithm accepts reduced representation of Boolean functions in form of an array of cubes and operates on an Algebraic Ternary Decision Tree together with lookup tables of flexible sizes. Allocation of don't care minterms is performed in an non exhaustive way by a heuristic approach based on the properties of Reed-Muller expansions.

I. Introduction

It has long been known that for some applications, the logic circuits using EXOR gates are more economical than the design based on other gates. What is more, circuits built around the EXOR gates, are easily testable. Fault detection of any logical circuit by verification of its Reed-Muller (RM) coefficients was considered in [2]. The problem of finding the minimal RM expansion of a Boolean function is classical in logic synthesis and has received various formulations in terms of coding or graph theory [3]. Such expansions can always be derived from any logical expression of Boolean functions by simple manipulations in either modulo 2 algebra or standard arithmetic [3]. Moreover, there exists transformation matrices for obtaining any RM expansion directly from the truth vector of the function [3]. Later, more attention has been paid to the development of algorithms for the determination of RM expansions of Boolean functions from their disjunctive normal form (DNF). Fisher [6] derived a first algorithm to determine the RM expansions directly from DNF. Some improvements on this approach with a suitable cube notation was indicated in [3, 5, 11, 17].

The most popular criterion for minimization of RM expansions is to obtain the expression with the least number of terms. Saluja and Ong [10] proposed an exhaustive algorithm to compute all the RM expansions sequentially using matrix multiplication. Row wise and column wise construction of the polarity coefficient matrix (PCM) was also considered in [3]. These algorithms have a space complexity given by $O(4^n)$ when the Boolean function is initially represented by a truth vector. Non-exhaustive minimization methods for RM expansions grounded on the concept of the extended truth vector were considered in [3]. Those classes of algorithms have a space complexity of $O(3^n)$ with the same assumption on the initial truth vector representation of a Boolean function [3]. Another method for the construction of PCM of RM expansions without matrix multiplication was shown by Harking [7]. In order to find an optimal FPRM expansion an exhaustive search of all the polarity vectors is necessary. Wu, Chen and Hurst [16] introduced a map method for mapping coefficients of a positive polarity RM expansion to find the minimum FPRM solution without undertaking exhaustive determination of all possible realizations. However, they pointed in their paper that the method is not clear for both analysis and synthesis of all fixed polarity cases. Sarabi and Perkowski [11] presented an exact and a quasi-minimal algorithms for the minimization of FPRM expansions. Similarly to three earlier methods [5, 6, 17] their algorithm is based on the reduced representation in the form of disjoint cubes. To find an optimal expansion the total search is necessary which is performed in Gray code order to minimize the number of required operations. Unfortunately, their algorithms can handle only single output completely specified functions.

In this paper, a hybrid approach combining the advantages of lookup table based methods and efficient data structure in the form of a decision tree for minimization of FPRM expansions for large multiple output completely and incompletely specified Boolean functions is presented. Contrary to all exact algorithms known from the literature, our algorithm for the minimization of a system of Boolean functions, is adaptable to different cost criteria. Based on the size of the tackled problem, our algorithm can use different size of the lookup table and decision diagram in order to trade space complexity for processing time. Since many CAD methods using RM coefficients require only the knowledge of some coefficients, such an option is also available in our algorithm. Hence, the number of operations to be performed is dependent on the size of the underlying function, chosen cost criteria, the size of the lookup table, and the required information : either some spectral coefficient in a chosen polarity are sought for, or the best fixed polarity RM expansion based on the chosen criteria is required. The results of the research summarized here can serve as a basis for assessing the quality of various heuristic minimizers. The introduced algorithm also provides mechanism for handling functions with don't care conditions in a non exhaustive way. Although absolute minimality can not be guaranteed for all incompletely specified functions, it generally obtains either minimal or quasi-minimal RM expansion for most functions.

II. Basic Definitions

Definition 1 : An n-variable Boolean function can be expressed as a *canonical Reed-Muller expansion* [3-11, 14-17] of 2^n terms as follows :

$$F(x_n, x_{n-1}, \cdots, x_1) = \bigoplus_{i=0}^{2^n-1} a_i \prod_{j=1}^{n} x_j^{\bullet k_j} \qquad (1)$$

where $\oplus$ denotes the modulo-2 addition, $a_i \in (0, 1)$ is called a Reed-Muller coefficient and $k_j \in (0, 1)$ is called the power of $\overset{\bullet}{x}_j$ such that $k_n k_{n-1} \ldots k_2 k_1$ is equal to the binary representation of i. When $k_j = 0$, the literal $\overset{\bullet}{x}_j$ is absent in the product term $\prod_{j=1}^{n} \overset{\bullet}{x}_j^{k_j}$, otherwise it is present in the product term. When each literal ($\overset{\bullet}{x}_i$, $i = 1, 2, \ldots, n$) throughout the expression (1) assumes either true or complemented form but not both forms simultaneously, such an expression is known as a *fixed polarity Reed-Muller* (*FPRM*) expansion.

Let x_i^j denote $\bar{x}_i$, x_i or 1 for $j = 0, 1, 2$ respectively. A Reed-Muller product term can also be expressed as $\prod_{i=1}^{n} x_i^{j_i}$ for $j_i \in (0, 1, 2)$. It is convenient to interpret the decimal number j as the ternary n-tuple $\langle j_n j_{n-1} \ldots j_2 j_1 \rangle$ when $j = \sum_{i=1}^{n} j_i \times 3^{i-1}$.

Definition 2 : The *polarity number* ω of an FPRM expansions is an integer computed by taking the decimal equivalent of the n bit straight binary code formed by writing a 0 or a 1 for each literal according to whether this literal is used in a positive or negative form in all the terms. The polarity number ω may be written as a binary n-tuple $\langle \omega_n \omega_{n-1} \ldots \omega_2 \omega_1 \rangle$ instead of the decimal form.

Definition 3 : A *polarity vector* A^{ω} is an ordered set of all 2^n FPRM coefficients $[a_{2^n-1} a_{2^n-2} \ldots a_0]$ in some polarity ω.

Definition 4 : The *polarity coefficient matrix* (PCM) of an n-variable Boolean function F, denoted by $PC(F)$ is a $2^n \times 2^n$ binary matrix, where every row corresponds to a polarity vector A^{ω} in a different polarity ω.

Property 1 : Every element m_{ij} (row i, column j) of the PCM describes the coefficient a_j of the FPRM expansion with polarity $\omega = i$.

Definition 5 : The *Hamming weight* of an integer i, denoted by $H(i)$ is the number of '1' in its binary representation.

Definition 6 : The *weight* of an FPRM expansion considering the number of product terms, denoted by $w_p(\omega)$, is the number of nonzero coefficients in the polarity vector A^{ω}. Similarly, the weight of an FPRM expansion in terms of the number of literals, denoted by $w_l(\omega)$, is the sum of the orders of the nonzero coefficients in the polarity vector A^{ω} . The *order* of a coefficient is Hamming weight of the power k of its product term in (1).

Ordered sets of either all 2^n weights w_p or w_l of $PC(F)$ in ascending order of the polarity number ω are denoted by the column vector W_p or W_l respectively. For convenience, the term weight w without any subscript refers to either w_p or w_l when its use in the context is unambiguous.

Property 2 : The *optimal polarity* of a Boolean function F in an FPRM expansion happens for A^{ω} that has a minimal weight and is equal to the row number of $PC(F)$ with such a minimal weight.

III. Lookup Tables for Weights of FPRM Expansions

Recently, an algorithm has been developed that utilizes only a subset of Walsh coefficients to reveal all the information carried by the PCM of any 3 variable Boolean functions [4]. Each class of the functions is associated with a specific subroutine that computes the optimal polarities, optimal weights, optimal FPRM expansions etc. without resorting to an exhaustive search. Direct extension of the method in [4] to handle larger Boolean functions with the number of variables $n>3$ is unmanageable due to the increasing number of different classes. Nevertheless, exact optimal generation of FPRM expansions for large n can be solved by reducing the PCM into submatrices of smaller dimension such that each submatrix is a PCM of a subfunction obtained by either Shannon's decomposition or Boolean difference with respect to some variables.

Lemma 1 : The PCM of an n-variable completely specified Boolean function F, can be partitioned into four submatrices of order 2^{n-1} as [6, 7] :

$$PC(F) = \begin{bmatrix} PC(f^0) & PC(f^0 \oplus f^1) \\ PC(f^1) & PC(f^0 \oplus f^1) \end{bmatrix} \qquad (2)$$

where f^0 and f^1 are cofactors of Shannon's decomposition of $F(X)$ evaluated at $x_n = 0$ and $x_n = 1$ respectively.

By applying (2) recursively, the PCM of order 2^n can be partitioned into p^2 submatrices of order 2^k, where $p = 2^{n-k}$. Each submatrix is a PCM of a k variable subfunction and the total number of unique subfunctions is equal to 3^{n-k}.

A subfunction f^i, $i = 0, 1, 2$ for an n-variable Boolean function F is defined as : $f^0 = F(0, x_{n-1}, \ldots, x_2, x_1)$, $f^1 = F(1, x_{n-1}, \ldots, x_2, x_1)$ and $f^2 = f^0 \oplus f^1$. This definition can be generalized to more than one variable. In general, a subfunction of k variables, $0 \le k < n$, can be formed by decomposing or taking the Boolean difference of F with respect to some $n-k$ variables. Each subfunction f^{α} of k-variables is associated with a ternary k-tuple $\langle \alpha_k \alpha_{k-1} \ldots \alpha_2 \alpha_1 \rangle$ where $\alpha_i \in (0, 1, 2)$. If $\alpha_i \in (0, 1)$ for all $1 \le i \le n-k$, α is said to be reducible to a binary k-tuple and f^{α} is a cofactor formed from F by substituting all the k variables with α_i in the following manner : x_n with α_n, x_{n-1} with α_{n-1}, etc.. If there exists an $\alpha_i = 2$ for any $1 \le i \le k$, then $f^{\alpha} = f^{\beta} \oplus f^{\gamma}$ where $\beta = \langle \alpha_k \alpha_{k-1} \ldots \alpha_{i+1}\ 0\ \alpha_{i-1} \ldots \alpha_2 \alpha_1 \rangle$ and $\gamma = \langle \alpha_k \alpha_{k-1} \ldots \alpha_{i+1}\ 1\ \alpha_{i-1} \ldots \alpha_2 \alpha_1 \rangle$. It is obvious that two or more different α_i, α_j, etc., $i \ne j$, can be equal to 2 in f^{α}.

Theorem 1 : The weight vectors of an n-variable Boolean function F can be expressed as $[W(f^0)\ W(f^1) \ldots W(f^{2^{n-k}-1})]$ where each subvector $W(f^r)$ of k-variable subfunction f^r in term of the number of products and number of literals are given by :

$$W_p(f^r) = \sum_{j=0}^{2^{n-k}-1} W_p(f^{r*j}) \qquad (3a)$$

$$W_l(f^r) = \sum_{j=0}^{2^{n-k}-1} [W_l(f^{r*j}) + H(j)\, W_p(f^{r*j})] \qquad (3b)$$

where $0 \le k < n$, r and j are decimal equivalents of binary $(n-k)$-tuples $\langle r_{n-k}\ r_{n-k-1} \ldots r_1 \rangle$ and $\langle j_{n-k}\ j_{n-k-1} \ldots j_1 \rangle$. The operator $* : \{0, 1\}^k \times \{0, 1\}^k \to \{0, 1, 2\}^k$ is defined by the following componentwise table of operation :

	second operand		
first operand	*	0	1
	0	0	2
	1	1	2

Proof : The weight vector $W(F)$ of $PC(F)$ for an n-variable Boolean function F can be obtained in a similar recursive way from the weight vectors $W(f^0)$, $W(f^1)$ and $W(f^0 \oplus f^1)$ corresponding to the FPRM expansions of subfunctions f^0, f^1 and f^2 respectively.

$$W_p(F) = \begin{bmatrix} W_p(f^0) + W_p(f^2) \\ W_p(f^1) + W_p(f^2) \end{bmatrix}$$

$$= \begin{bmatrix} W_p(f^{00}) + W_p(f^{02}) + W_p(f^{20}) + W_p(f^{22}) \\ W_p(f^{01}) + W_p(f^{02}) + W_p(f^{21}) + W_p(f^{22}) \\ W_p(f^{10}) + W_p(f^{12}) + W_p(f^{20}) + W_p(f^{22}) \\ W_p(f^{11}) + W_p(f^{12}) + W_p(f^{21}) + W_p(f^{22}) \end{bmatrix} = \text{... etc.}$$

After $n-k$ recursions, there are 2^{n-k} rows each having 2^{n-k} weight vectors W_p to be summed. The first weight vector in the r-*th* ($r=0, 1, \ldots 2^{n-k}-1$) row is the weight vector W_p of the cofactor generated by decomposing F with respect to $\prod_{i=1}^{k} x_{i+1}^{r_i}$. If the cofactor at r-*th* row is denoted by f^r, the subfunction appearing in the j-*th* ($j = 0, 1, \ldots 2^{n-k}-1$) weight vector W_p of the same row is given by f^{r^*j}. Hence (3a) is obtained.

When (2) is expanded recursively $n-k$ times, each submatrix at the r-th row and j-th column corresponds to a subfunction f^{r^*j}. Each such submatrix $PC(f^{r^*j})$ has the dimension of $2^k \times 2^k$ and its elements are located at the row number g and column number h where $g, h = 0, 1, \ldots, 2^k-1$. Any non zero entry in $PC(f^{r^*j})$ corresponds to a Reed-Muller product term $\prod_{i\in v} x_{i+k}^{r_i} \cdot \prod_{i\in u} x_i^{g_i}$ where $u=\{Z: 1\le u \le k \text{ and } h_u = 1\}$, $v=\{Z: 1\le v \le n-k \text{ and } j_v = 1\}$ and Z is the set of integers. The number of literals in the product $\prod_{i\in u} x_i^{g_i}$ has already been accounted for by W_f. Since there are $H(j)$ additional literals in $\prod_{i\in v} x_{i+k}^{r_i}$ of each Reed-Muller product term, the weight vector for the number of literals for any submatrix $PC(f^{r^*j})$ is given by $W_f(f^{r^*j}) + H(j)W_p(f^{r^*j})$. Hence (3b) follows. Q.E.D.

In order to use the subfunction as an index to the lookup tables, each subfunction is assigned a unique integer value whose decimal equivalent is $< m_{2^k-1}\, m_{2^k-2} \cdots m_1\, m_0 >$ where $m_i \in (0, 1)$ is the binary value of the i-th minterm of the k-variable subfunction f^α. We call this integer value the row index and denote it as $\Re(f^\alpha)$. Clearly, $\Re(\overline{f^\alpha}) = 2^{2^k}-1-\Re(f^\alpha)$.

Three lookup tables in the form of two dimensional integer arrays $Exp[R/2][C]$, $P[R][C]$ and $L[R/2][C]$ are created for any k-variable subfunction where $R = 2^{2^k}$ and $C = 2^k$. For $k = 3$, $R = 256$, $C = 8$ and the lookup tables can be generated from the algorithm in [4]. The polarity number ω is used as the column index to all three arrays. $\Re(f^\alpha)$ is used as the row index to the array P and the smallest integer between $\Re(f^\alpha)$ and $\Re(\overline{f^\alpha})$, i.e., $min(\Re(f^\alpha), \Re(\overline{f^\alpha}))$ is used as the row index to arrays Exp and L. Each entry of Exp is an integer whose binary representation is $<a_7\ a_6\ \ldots\ a_1\ a_0>$ where a_i is the $(i+1)$-th coefficient of the polarity vector A^ω of the subfunction f^α or $\overline{f^\alpha}$ whichever has the lower row index. If $\Re(\overline{f^\alpha}) < \Re(f^\alpha)$, $A^\omega(f^\alpha) = 1 \oplus_d Exp[\Re(\overline{f^\alpha})][\omega]$ where $\oplus_d$ is the symbol for dyadic addition between two integers. Each entry in P is the weight w_p of the polarity vector A^ω for the subfunction f^α and each entry in L is the weight w_f of the polarity vector A^ω for the subfunction f^α or $\overline{f^\alpha}$ whichever has the lower row index. Notice that $w_f(\overline{f^\alpha}) = w_f(f^\alpha)$ since $A^\omega(\overline{f^\alpha})$ and $A^\omega(f^\alpha)$ differ only in the constant term a_0.

Corollary 1 : The exact weight of any n-variable completely specified Boolean function in polarity ω is given by :

$$w_p(\omega) = \sum_{j=0}^{M-1} P[\Re(f^{(\omega>>k)^*j})][\omega \bmod 2^k] \quad (4a)$$

$$w_f(\omega) = \sum_{j=0}^{M-1} \{L[\min(\Re(f^{(\omega>>k)^*j}), 2^{2^k}-1-\Re(f^{(\omega>>k)^*j}))][\omega \bmod 2^k] + H(j)\, P[\Re(f^{(\omega>>k)^*j})][\omega \bmod 2^k]\} \quad (4b)$$

where "$\omega >> k$" is the right shift operator of the binary number ω by k positions, $M = 2^{n-k}$ and $\omega \bmod 2^k$ is the remainder of ω divided by 2^k. The proof of (4a) and (4b) is straightforward by replacing r in Theorem 1 by $\omega >> k$.

Corollary 2 :

$$A^\omega = \sum_{j=0}^{M-1} 2^j \{C_j \oplus_d Exp[\min(\Re(f^{(\omega>>k)^*j}), 2^{2^k}-1-\Re(f^{(\omega>>k)^*j}))][\omega \bmod 2^k]\} \quad (5)$$

where $C_j = 1$ if $\Re(f^{(\omega>>k)^*j}) \ge 2^{2^k-1}$ and $C_j = 0$ otherwise.

In Corollary 2, the polarity vector A^ω of the n-variable Boolean function is represented by a decimal integer whose binary equivalent is equal to $< a_{2^n-1}\ a_{2^n-2}\ \ldots\ a_1\ a_0 >$. In fact, A^ω is formed as a concatenation of the binary representations of $C_j \oplus_d Exp\left[\min\left(\Re(f^{(\omega>>k)^*j}), 2^{2^k}-1-\Re(f^{(\omega>>k)^*j})\right)\right][\omega \bmod 2^k]$ for $j = 0, 1, \ldots, M-1$. The proof of Corollary 2 is a trivial extension from the proof of Theorem 1.

Example 1 : Consider the 5-variable Boolean function $F(x_5, x_4, x_3, x_2, x_1) = \Sigma m(8, 10, 11, 16, 17, 19, 23, 24, 26, 27)$. This example is taken from [16]. Using the 3-variable lookup tables, i.e., $k = 3$, the number of Reed-Muller products in polarity $\omega = <01110> = 14$ can be calculated as follows :

$M=2^{5-3}=4$. $\omega>>3=<00001>$. $14 \bmod 8=6$. For $j=0$, $f^{<01>*<00>} = f^{<01>} = <00001101>$; For $j=1$, $f^{<01>*<01>} = f^{<02>} = f^{<00>} \oplus_d f^{<01>} = <00000000> \oplus_d <00001101> = <00001101>$; For $j=2$, $f^{<01>*<10>} = f^{<21>} = f^{<01>} \oplus_d f^{<11>} = <00001101> \oplus_d <00001101> = <00000000>$; For $j=3$, $f^{<01>*<11>} = f^{<22>} = f^{<20>} \oplus_d f^{<21>} = f^{<00>} \oplus_d f^{<10>} \oplus_d f^{<21>} = <00000000> \oplus_d <10001011> \oplus_d <00000000> = <10001011>$. As $\Re(<00001101>)=13$, $\Re(<00000000>)=0$ and $\Re(<10001011>)=139$, by (4a), $w_p = 2P[13][6]+P[0][6]+P[139][6] = 2\times2+0+3 = 7$.

The number of literals for polarity $\omega = 14$ can be calculated by (4b) as follows : Since only $139 > 127$ and $min(139, 255-139) = 116$, $w_f = L[13][6] + H(0)P[13][6] + L[13][6] + H(1)P[13][6] + L[0][6] + H(2)P[0][6] + L[116][6] + H(3)P[139][6] = 4 + 0 + 4 + 2 + 0 + 0 + 5 + 2 \times 3 = 21$.

The FPRM expansion in polarity $\omega = 14$ can be calculated by (5) as follows : $A^{14} = 8(1 \oplus_d Exp[116][6]) + 4Exp[0][6] + 2Exp[13][6] + Exp[13][6] = 8 \times (1 \oplus_d 75) + 4 \times 0 + 2 \times 144 + 144 = [74 \mid 0 \mid 144 \mid 144] = [\,01001010 \mid 00000000 \mid 10010000 \mid 10010000\,] = \bar{x}_3 \oplus \bar{x}_3\bar{x}_2x_1 \oplus \bar{x}_4\bar{x}_3 \oplus \bar{x}_4\bar{x}_3\bar{x}_2x_1 \oplus x_5\bar{x}_4x_1 \oplus x_5\bar{x}_4\bar{x}_2x_1 \oplus x_5\bar{x}_4\bar{x}_3\bar{x}_2$.

As noticed by Fisher [6], the minimization of FPRM expansions can be divided into two steps. The first step is to find an optimal polarity and the second is to realize the FPRM expression of the switching function with this polarity.

Most approaches to the first problem, though exhaustive, are formidable to varying cost functions and usually identify only one out of the many possible optimal polarities. With an appropriate data structure, Corollary 1 provides efficient and flexible means for the identification of all optimal as well as suboptimal polarities based on different criteria if necessary. Once the optimal polarities have been identified, realization of the FPRM expansions at those polarities is straightforward by applying Corollary 2.

IV. Minimization of Multiple Output Functions

In order to obtain an exact global minimization for a system of completely specified functions, common terms for each polarity must be sought. Theorem 2 gives a method for determining the exact weight of the FPRM expansion in any polarity with product terms shared by more than one output counted only once.

Theorem 2 : Let $f_1, f_2, \ldots, f_m$ be the outputs of an n-variable m-output function F, then the weight of the FPRM expansion of F in any polarity ω is given by :

$w(F)=\frac{1}{2^{m-1}}[\Sigma w(f_g)+\Sigma w(f_g \oplus f_h)+\Sigma w(f_g \oplus f_h \oplus f_j)+\ldots+\Sigma w(f_1 \oplus f_2 \oplus \ldots \oplus f_m)]$ (6)

where $g, h, j \in \{1, 2, \ldots, m\}$. $g \neq h$, $h \neq j$, $g \neq j$.

Proof: Consider any arbitrary polarity ω, $0 \leq \omega \leq 2^n-1$. Let the polarity vectors of m outputs be $A_1, A_2, \ldots, A_m$. Any non zero RM coefficient a_i that appears in at least one of $A_1, A_2, \ldots, A_m$ contributes a weight of 1 to $w_p(F)$ or $H(i)$ to $w_f(F)$. Let p_r be the contribution of the non zero coefficient a_r to $w_p(F)$ and l_r be the contribution of a_r to $w_f(F)$. $p_r = 0$ iff $a_r = 0$ in all the polarity vectors $A_1, A_2, \ldots, A_m$ and 1 otherwise. $l_r = 0$ iff $r = 0$ or $a_r = 0$ in all the polarity vectors $A_1, A_2, \ldots, A_m$ and $H(r)$ otherwise. Since $H(0) = 0$, $l_r = H(r)\, p_r$.

Define $y_{r1}, y_{r2}, \ldots, y_{rm} \in (0,1)$ to be the variables representing the logical value of any RM coefficient a_r in $A_1, A_2, \ldots, A_m$ respectively. Then $p_r = y_{r1} \vee y_{r2} \vee \ldots \vee y_{rm}$. It can be shown by induction that for any Boolean variable x_i, $2^{m-1}\left(\bigvee_{i=1}^{m} x_i\right)=\sum_{i=1}^{m} x_i +$ Σ(all combinations of EXORing of two variables) + Σ(all combinations of EXORing of three variables) + ... + Σ(all combinations of EXORing of $m-1$ variables) + (EXORing of m variables), where Σ means arithmetic summation. Thus, $w_p(F)=\sum_{r=0}^{2^n-1} p_r=\frac{1}{2^{m-1}}\{\sum_{r=0}^{2^n-1}\sum_{g=1}^{m} y_{rg}+\sum_{r=0}^{2^n-1}\Sigma(y_{rg} \oplus y_{rh})+\sum_{r=0}^{2^n-1}\Sigma(y_{rg} \oplus y_{rh} \oplus y_{rj})+ \ldots+\sum_{r=0}^{2^n-1} y_{r1} \oplus y_{r2} \oplus \ldots \oplus y_{rm}\}=\frac{1}{2^{m-1}}\{\sum_{g=1}^{m}\sum_{r=0}^{2^n-1} y_{rg}+\Sigma\sum_{r=0}^{2^n-1}(y_{rg} \oplus y_{rh})\Sigma\sum_{r=0}^{2^n-1}(y_{rg} \oplus y_{rh} \oplus y_{rj})+\ldots+\sum_{r=0}^{2^n-1} y_{r1} \oplus y_{r2} \oplus \ldots \oplus y_{rm}\}=\frac{1}{2^{m-1}}\{\sum_{g=1}^{m} w_p(f_g)+\Sigma w_p(f_g \oplus f_h)+ \Sigma w_p(f_g \oplus f_h \oplus f_j)+\ldots+w_p(f_1 \oplus f_2 \oplus \ldots \oplus f_m)\}$. Similarly, $w_f(F)=\sum_{r=0}^{2^n-1} H(r)\, p_r=\frac{1}{2^{m-1}}\{\sum_{g=1}^{m} w_f(f_g)+\Sigma w_f(f_g \oplus f_h)+\Sigma w_f(f_g \oplus f_h \oplus f_j)+\ldots+w_f(f_1 \oplus f_2 \oplus \ldots \oplus f_m)\}$. Hence $w(F)=\frac{1}{2^{m-1}}\{\sum_{g=1}^{m} w(f_g)+\Sigma w(f_g \oplus f_h)+\Sigma w(f_g \oplus f_h \oplus f_j)+\ldots+ w(f_1 \oplus f_2 \oplus \ldots \oplus f_m)\}$ for any polarity. Q.E.D.

Corollary 3 : For a system of m n-variable completely specified Boolean functions F, the weight of the FPRM expansion in polarity ω is given by :

$w_p(\omega) = \frac{1}{2^{m-1}}\{\sum_{j=0}^{k-1}\sum_{s=1}^{2^{n-1}} P[\bigoplus_{s_i=1} \Re(f_i^{(\omega>>k)*j})][\omega \bmod 2^k]\}$ (7a)

$w_f(\omega) = \frac{1}{2^{m-1}}\{\sum_{j=0}^{k-1}\sum_{s=1}^{2^{n-1}} (L[\min(\bigoplus_{s_i=1} \Re(f_i^{(\omega>>k)*j}),\ 2^{2^k}-1-\bigoplus_{s_i=1} \Re(f_i^{(\omega>>k)*j}))]$ $[\omega \bmod 2^k] + H(j)\, P[\bigoplus_{s_i=1} \Re(f_i^{(\omega>>k)*j})][\omega \bmod 2^k])\}$ (7b)

where s is the decimal number and s_i is the i-th ($i = 1, 2, \ldots, m$) bit in binary m-tuple of s. f_i is the i-th output of F.

V. Data Structure and Implementation

In this section, we introduce a new data structure called the *Algebraci Ternary Decision Tree* (ATDT). It is analogous to the generation tree [8] and EXOR Ternary Decision Diagram (ETDD)[13] with the following distinctive features :

1. There are $n-k+1$ levels instead of $n+1$ levels as compared with the generation tree or ETDD. k is the number of variables of the subfunctions for the lookup tables.
2. Each leaf of the tree consists of a pointer to an array of integers as oppose to the use of multi-place decision diagram for multiple output functions.

ATDT is a rooted directed acyclic graph with vertex set V comprising of two basic types of vertices. A non terminal vertex v has as attribute an $index(v) \in \{n-k, n-k-1, \ldots, 1\}$ and three children $low(v)$, $high(v)$ and $\Delta(v) \in V$. A terminal vertex, commonly called a leaf v has as attributes an $index(v)=0$ and a pointer $value(v)$ to an integer array S, where $S[i] \in \{0, 1, \ldots, 2^{2^k}-1\}$ for $1 \leq i \leq m$, is the row index of the i-th output of the m output functions. For readability, we denote the integer $S[i]$ by $value(v) \rightarrow i$ in the description of the algorithms. In addition, the edges connected to $low(v)$, $high(v)$ and $\Delta(v)$ are called the *0*-, *1*- and *2*-edges of v respectively. $f_{low(v)}$ and $f_{high(v)}$ are cofactors generated by decomposing f_v with respect to $\bar{x}_i$ and x_i respectively where $i = index(v) + k$, $f_{\Delta(v)}=f_{low(v)} \oplus f_{high(v)}$.

If the function F is represented by a binary n-tuple with the i-th bit representing the presence or absence of a specific truth minterm m_{i-1}, the subfunction f^0 is a binary $(n-1)$-tuple obtained from the most significant half of F and f^1 is a binary $(n-1)$-tuple obtained from the least significant half while $f^2 = f^0 \oplus f^1$. Continuing in this manner, the subfunctions at each level are represented by binary $(l+k)$-tuples where the index $l = n-k, n-k-1, \ldots, 1, 0$ is decremented level by level.

Each path from the root to the leaf is represented by a ternary $(n-k)$-tuple *path* where the i-th digit of *path* is formed by taking the edge value of the vertex at level i. If *path* is reducible to a binary $(n-k)$-tuple, value(v)$\rightarrow j$ of the leaf v can be obtained directly by decomposing the j-th output function with respect to *path* using the procedure **decompose**(f_j, *path*). Procedure **decompose**(f_j, *path*) computes $\Re(f^{\alpha})$ directly from a reduced representation such as a sum-of-product (SOP) form without requiring the truth table representation of the function f_j. The system of Boolean functions is given initially in the form of an array of cubes, not necessary disjoint. Each cube C has an input part $I(C) = <c_n\, c_{n-1} \ldots c_1>$ and an output part $O(C) = <c_{n+m}\, c_{n+m-1} \ldots c_{n+1}>$ where for $1 \leq i \leq n$, $c_i = 0, 1,$ – corresponding to the presence of a positive, negative or the absence of x_i variable in a SOP term respectively, $c_i = 0, 1$ for $n+1 \leq i \leq n+m$ corresponding to the presence or absence of the term $I(C)$ in the $(i-n)$-th output. After creating the ATDT,

the row index $\Re(f^{(\omega>>k)*j})$ in Equations (4a), (4b) and (5) can be easily obtained by traversing the tree. The Procedure **traverse**(ω, j) returns a leaf v and Hamming weight $H(j)$.

To search for the optimal polarities, an object consisting of two components w and ω is defined. An array of 2^n such objects is generated according to the following procedure. For each $\omega = 0, 1, 2$ to 2^n-1, the algorithm loops through $j = 0, 1, \ldots$ to $M-1$ where $M = 2^{n-k}$, and executes **traverse**(ω, j) to reach the appropriate leaf v. Since $\Re(f_i^{(\omega>>k)*j})$ is equal to $value(v) \rightarrow [i]$, (7a) or (7b) can be applied to calculate the weight w for polarity ω of each object depending on the primary cost criterion. Once all 2^n objects are created, the array of objects are sorted in ascending order of the component w. All polarities of objects, having the same w as the top object, are optimal. The FPRM expansions of each output can be calculated using (5) for these optimal polarities. If there is a secondary cost criterion, further optimization among these polarities is carried out in a similar procedure as before with a reduced search space. If only one of the optimal polarities is required, in place algorithm can be used to avoid sorting of an array of objects.

VI. Incompletely Specified Functions

Let D_f be the set of don't care minterms such that $D_f = \{d_i \in \{0, 1\}^n \mid f_j(d_i) = -$ for any $0 \le i \le 2^n-1, 1 \le j \le m\}$. Our algorithm, like those of Fisher and Varma [6, 17], decouples the effect of don't cares one at a time. Unlike their algorithms where the don't care minterms are stepped through in a fixed decimal order, we improve the final result by assigning the most influential don't care first. The following Property and Proposition are used as the basis for the new algorithm.

Property 3 : Any minterm can be transformed into a RM product in polarity ω by replacing each literal $\dot{x}_i$ by $-$ if $\dot{x}_i = \bar{x}_i$ and $\overline{\omega}_i$ if $\dot{x}_i = x_i$ where $\omega_i \in \{0, 1\}$ and $-$ represents the absence of variable x_i in the RM product.

Proposition 1 : The don't care minterm that forms the largest RM product in the current optimal polarity is assigned first. If the assignment results in a new optimal polarity, the don't care minterm that forms the largest RM product in the new optimal polarity is then assigned, else the one that forms the next larger product in the same polarity is assigned.

The philosophy behind the above proposition is twofold : (1) at least half of the RM coefficients in polarity ω are affected by the assignment of a don't care minterm d_i if $i = \omega$. (2) it is more likely to improve the weight of the current optimal or near optimal polarity expansion than one that is closer to the worse case polarity by assigning a don't care minterm.

In ATDT, among all the paths that lead to a leaf containing some don't care minterm, the path, if any, with its edge values matching the most significant $n-k$ bits of the current optimal polarity is traversed first, followed by the path that differs in only $1, 2, \ldots, n-k$ edge values. If there exists more than one don't care minterms in the designated leaf, the don't care minterm d_i with the lower Hamming weight of $H(i \oplus_d \omega_{min})$ is assigned first. The optimal polarity can change at any point depending on whether the assignment leads to an improvement, thus the order of traversing the tree changes dynamically.

The procedure for dealing with incompletely specified system of functions is given as follows :

1. Assign all don't cares to 0, i.e., $f_j(d_i)=0$ for all $d_i \in D_f$, $1 \le j \le m$. Apply the algorithm given in the previous section to obtain a minimal weight w_{min} and its associated polarity ω_{min}.
2. According to Proposition 1, assign the most influential don't care minterm $d_i \in D_f$, $0 \le i \le 2^n-1$ to 1 and change each row index $\Re(f_j)$ that is affected by such an assignment to $\Re(f_j^u)' = \Re(f_j^u) \oplus_d 2^n$ where u is the decimal equivalent of $<i_k\, i_{k-1} \ldots i_1>$ and f_j is an output that contains the minterm d_i. The leaves containing the pointer to the affected row indices are traversed by Procedure **traverse**(i, t) for $t = 0$ to $2^{n-k}-1$.
3. Calculate the new weight w and its associated polarity ω in place for each polarity ω from 0 to 2^n-1. At any time if $w \ge w_{min}$ within the loops $\sum_{j=0}^{M-1} \sum_{s=0}^{2^n-1}$ (refer to (7a) and (7b)), the computation for that polarity is pruned and the weight for the next polarity is calculated. If $w < w_{min}$ for some polarity ω, assign $w_{min} = w$ and $\omega_{min} = \omega$. Repeat the above process until the weight of all polarities have been calculated.
4. If w_{min} remains unchanged after Step 3, reassign the don't care minterm d_i to 0 for f_j and remove d_i from D_f. Restore the tree before Step 2 by replacing all affected row indices $\Re(f_j^u)'$ back to $\Re(f_j^u)$. If w_{min} has improved, remove d_i from D_f.
5. If D_f is not empty, go to Step 2.

VII. Experimental Results

The algorithm has been implemented on the Sun Sparc IPC workstation with 22 mega bytes of system memory. The current implementation uses 3 variable lookup tables, i.e., $k = 3$ and allows on the calculation of minimal FPRM expansions based on different cost criteria. Theoretically, any cost functions of the form $aw_l + bw_p$ where a, b are integer constants are allowed. For convenience, our algorithm attempts to search for all the optimal FPRM expansions that meet the user specified primary optimization criterion based on the requirements of a specific target device. Among those optimal polarities, further minimization is performed according to the secondary criterion if there is any. The primary and secondary criteria may be any of w_l or w_p. There is also an option to allow the user to specify some upper bounds on w_l and w_p. To the best of our knowledge, an exact algorithm having flexible user specified criteria has been proposed for the first time by us. Our program will search for such polarities that fulfill the requirements and produce FPRM expansions of those polarities. Since the number of vertices in ATDT for the current implementation is equal to $\frac{1}{2}(3^{n-k+1} - 1)$ and each of the 3^{n-k} leaves points to an array of size m, the space complexity of our algorithm is $O(\frac{3+2m}{2} \times 3^{n-k})$. On the average ATDT is traversed 2^m-1 times, thus the time complexity is $O(2^{m-1} \times 3^{n-k+1})$.

A range of benchmark examples have been tested and the results compared favorably with well known previous algorithms. Some two level examples from MCNC benchmarks minimized with w_p as the primary cost criterion and w_l as the secondary criterion have been summarized in Table I. Column *6* provides the number of FPRM expansions fulfilling both criteria. Our results for all completely specified functions agree with the exhaustive search routine of Cannes [1] and for the incompletely specified function inc, the number of terms generated by our algorithm is only one more than that of the exhaustive routine of Cannes. However, our results for all completely specified functions agree with the results given in [1, 15]. Unfortunately, no timing information is given in [1]. In order to compare the execution time with MINGRM, an exact minimizer which deals only with single output completely specified functions [11], the execution time of the same benchmark functions with selected output as published in [11] is reproduced in Table II. Though the results of MINGRM are obtained from a Sequent S27 machine with two processors, our program implemented on a non parallel machine is able to execute and produce optimal results for the complete system of functions much faster than MINGRM which minimizes only one selected output.

VIII. Conclusion

An efficient algorithm for the exact minimization of FPRM expansions of multiple output completely specified Boolean functions has been presented. The data structure employed for exploring the search space of the minimizer is an Algebraic Ternary Decision Tree modified from Ternary Decision Diagram [13]. Base on this data structure, a non exhaustive heuristic method is developed to optimally allocating the don't care outputs for incompletely specifed functions. Besides being exact, the proposed algorithm is also adaptable to various cost functions which is what lack in many exisiting minimizers. Since some ESOP minimization techniques start from a near minimal FPRM expansion [9, 14] followed by the search for mixed polarity reduction, our algorithm can be used as an effective preprocessor for those algorithms to improve both the execution time and quality of the results. The time and space complexity of the algorithm presented can be improved by collapsing the ATDT to a Reduced Algebraic Ternary Decision Diagram with attributed edges [13].

References

[1] L. Csanky, M. A. Perkowski and I. Schaefer, "Canonical restricted mixed-polarity exclusive-OR sums of products and the efficient algorithm for their minimization," *IEE Proc.*, vol. 140, Pt. E, no. 1, pp. 69-77, Jan. 1993.

[2] T. R. Damarla and M. Karpovsky, "Fault detection in combinational networks by Reed-Muller transforms," *IEEE Trans. Comput.*, vol. 38, no. 6, pp. 788-797, Jun. 1989.

[3] P. Davio, J. P. Deschamps and A. Thayse, *Discrete and Switching Functions*. New York: George and McGraw-Hill, 1978.

[4] B. J. Falkowski and C. H. Chang, "Generation of fixed polarity Reed-Muller expansions from subset of Walsh spectral coefficients for completely specified Boolean functions," in *Proc. of 5th Workshop on Spectral Techniques*, Beijing, China, pp. 214-219, Mar. 1994.

[5] B. J. Falkowski and M. A. Perkowski, "On the calculation of generalized Reed-Muller canonical expressions from disjoint cube representation of Boolean functions," in *Proc. 33rd IEEE Midwest Symp. on Circuits and Systems*, Calgary, Alberta, pp. 1131-1134, Aug. 1990.

[6] L. T. Fisher, "Unateness properties of AND-Exclusive-OR logic circuits," *IEEE Trans. Comput.*, vol. 23, no. 2, pp. 166-172, Feb. 1974.

[7] B. Harking, "Efficient algorithm for canonical Reed-Muller expansions of Boolean functions," *IEE Proc.*, vol. 137, Pt. E, no. 5, pp. 366-370, Sep. 1990.

[8] G. Papakonstantinou, "Minimization of modulo-2 sum of products," *IEEE Trans. Comput.*, vol. 28, no. 2, pp. 163-167, Feb. 1979.

[9] J. P. Robinson and C. L. Yeh, "A method for modulo-2 minimization," *IEEE Trans. Comput.*, vol. 21, pp. 800-801, Aug. 1982.

[10] K. K. Saluja and E. H. Ong, "Minimization of Reed-Muller canonic expansion," *IEEE Trans. Comput.*, vol. 28, pp. 535-537, Feb. 1979.

[11] A. Sarabi and M. A. Perkowski, "Fast exact and quasi-minimal minimization of highly testable fixed-polarity AND/XOR canonical networks," in *Proc. 29th ACM/IEEE Design Automation Conf.*, pp. 30-35, Jun. 1992.

[12] T. Sasao, *Logic Synthesis and Optimization*, Boston: Kluwer Academic Publishers, 1993.

[13] T. Sasao, "Optimization of Pseudo-Kronecker expressions using multiple-place decision diagrams," *IEICE Trans. Inf. and Syst.*, vol. E76-D, no. 5, pp. 562-570, May 1993.

[14] A. Tran, "Graphical method for the conversion of minterms to Reed-Muller coefficients and the minimisation of exclusive-OR switching functions," *IEE Proc.*, vol. 134, Pt. E, no. 2, pp. 93-99, Mar. 1987.

[15] C. C. Tsai and M. Marek-Sadowska, "Minimization of fixed-polarity AND/XOR canonical networks," *IEE Proc.*, vol. 141, Pt. E, no. 6, pp. 369-374, Nov. 1994.

[16] X. Wu, X. Chen and S. L. Hurst, "Mapping of Reed-Muller coefficients and the minimisation of exclusive OR-switching functions," *IEE Proc.*, vol. 129, Pt. E, no. 1, pp. 15-20, Jan. 1982.

[17] D. Varma and E. A. Trachtenberg, "Computation of Reed-Muller expansions of incompletely specified Boolean functions from reduced representations," *IEE Proc.*, vol. 138, Pt. E, no. 2, pp. 85-92, Mar. 1991.

TABLE I

Minimization with w_p as Primary Criterion and w_l as Secondary Criterion

	n	m	w_{pmin}	w_{lmin}	# opt FPRM	Time (sec) usr	Time (sec) sys	Examples of optimal polarities
5xp1	7	10	61	224	1	15.3	0.1	0
9sym	9	1	173	636	252	5.5	0.2	85, 86, 89, 90, 91, ...
con1	7	2	17	48	2	0.1	0.1	1, 5
inc	7	9	48	174	-	522.2	0.5	31 with dc_{42} of $f_4 = 1$
misex1	8	7	20	68	8	7.5	0.1	31,63,95,127,159,191,223,225
rd53	5	3	20	45	1	0.0	0.1	0
rd73	7	3	63	189	1	0.3	0.1	0
rd84	8	4	107	352	1	1.5	0.1	0
sao2	10	4	100	707	2	18.8	0.3	820, 868
squar5	5	8	23	56	1	0.2	0.1	0
xor5	5	1	5	5	16	0.0	0.1	0,3,5,6,9,10,12,15,18,20,...
Z9sym	9	1	173	636	252	5.6	0.2	92,93,94,171,172,173,174,...
clip	9	5	206	995	2	8.2	0.2	33,452

TABLE II

Execution Time of MINGRM

FUNCTION	MINGRM TIME (sec)
9sym	1,851.3
con12	1.2
rd532	1.1
rd732	49.6
rd842	364.2
sao22	642.1
sao23	905.9