IEICE Electronics Express, Vol.1, No.5, 92–97

Calculation of best fixed polarity Reed-Muller transform over GF(5)

Bogdan J. Falkowski,[1] **Cicilia C. Lozano,**[1] **and Susanto Rahardja**[2]

[1] *School of Electrical and Electronic Engineering, Nanyang Technological University, Block S1, 50 Nanyang Avenue, Singapore 639798*

[2] *Institute for Infocomm Research, 21 Heng Mui Keng Terrace, Singapore 119613*

Abstract: In this article, a new algorithm that takes the truth vector of a 5-valued function as its input and proceeds to generate all of the function's fixed polarity Reed-Muller (FPRM) spectral coefficient vectors one by one in a certain sequence is presented. The experimental results for this algorithm are compared with other methods and it was found that it is more efficient than other methods for some functions. Moreover, the presented algorithm requires very low memory storage.

Keywords: finite fields, Galois field(5), Reed-Muller, spectral coefficients

Classification: Science and engineering for electronics

References

[1] J. S. Lee and L. E. Miller, *CDMA Systems Engineering Handbook*, Artech House, Boston, 1998.

[2] T. Sasao, *Switching Theory for Logic Synthesis*, Kluwer Academic, Boston, 1999.

[3] B. J. Falkowski and S. Yan, "Walsh-Hadamard optimization of fixed polarity Reed-Muller transform," *IEICE Electron. Express*, vol. 1, no. 2, pp. 39–45, 2004.

[4] D. Jankovic, R. S. Stankovic, and C. Moraga, "Optimization of Kronecker expressions using the extended dual polarity property," *Proc. 37th Int. Sci. Conf. Inf., Commun. Energy Syst. Technol.*, Nis, Yugoslavia, pp. 749–752, Oct., 2002.

[5] D. Jankovic, R. S. Stankovic, and C. Moraga, "Optimization of GF(4) expressions using the extended dual polarity property," *Proc. 33rd IEEE Int. Symp. Multiple-Valued Logic*, Tokyo, Japan, pp. 50–55, May, 2003.

[6] B. J. Falkowski, C. C. Lozano, and S. Rahardja, "Spectra generation for fixed-polarity Reed-Muller transform over GF(5)," *Proc. 34th IEEE Int. Symp. Multiple-Valued Logic*, Toronto, Canada, May, 2004, accepted for publication.

1 Introduction

Reed-Muller (RM) transform had been successfully applied in many areas such as signal processing, fault detection, and coding techniques, especially

DOI: 10.1587/elex.1.92
Received April 15, 2004
Accepted May 06, 2004
Published June 10, 2004

those concerned with group or block codes for error control [1]. One reason for the wide usage of the RM transform is because it has been found to be advantageous in terms of area, speed, and testability [2]. This is true for both binary and multiple-valued cases.

Fixed polarity RM expansion (FPRME) is the RM expansion in which each variable has the same form throughout it. An n-variable p-valued function can be expressed by p^n different FPRMEs, where each of them is canonical and can be differentiated from each other by its polarity number. The polarity number of the FPRME with smallest number of spectral coefficients or literals is called the optimal polarity number.

Although an exact and non-exhaustive algorithm that generates optimal RM expansion for 3-variable binary functions directly from just few Walsh-Hadamard spectral coefficients had been developed in [3], in general no efficient method had been found that is able to obtain the optimal polarity number without first constructing the FPRMEs for all polarity numbers. Hence, it is important to find a method that is able to generate all polarity FPRMEs efficiently. In this article, such an algorithm is proposed for FPRME over Galois Field (5) (GF(5)), which is used in error-correcting codes for CDMA systems [1].

A method that optimizes Kronecker expressions by introducing the term extended dual polarity was presented in [4]. In [5], the method was extended for the optimization of FPRMEs over GF(4). In this article, an algorithm that uses extended dual polarity property for optimizing FPRMEs over GF(5) is introduced. The new algorithm is simple and can be implemented with low storage requirement. Experimental results for the algorithm are also presented here and their comparisons with other methods show that this algorithm is efficient in generating all possible FPRMEs for a given 5-valued logic function and in finding its optimal FPRME.

2 Basic definitions

Definition 1 Each variable x_l in an FPRME over GF(5) always appears in one of its five possible literals throughout the expansion. Let us denote the five possible literals by $^{<0>}x_l$, $^{<1>}x_l$, $^{<2>}x_l$, $^{<3>}x_l$, and $^{<4>}x_l$ whereby $^{<j_l>}x_l = x_l + j_l$ over GF(5) ($0 \le j_l \le 4$, $1 \le l \le n$). Then the polarity number ω of the expansion is the decimal equivalent of the 5-valued digits $j_1 j_2 j_3 \dots j_n$, $<\omega>_{10} = <j_1 j_2 j_3 \dots j_n>_5$.

Definition 2 The spectral coefficient vector in polarity ω of an n-variable function $f(\vec{x})$ is a collection of all 5^n spectral coefficients of the function's FPRME in polarity ω,

$$C^{\omega} = \left[c_0^{\omega}, c_1^{\omega}, \dots, c_{5^n-1}^{\omega}\right].$$

Let $\vec{F}$ be the truth vector of the function for which the FPRME is to be calculated. Then C^{ω} and $\vec{F}$ are related by the following transform:

$$C^{\omega} = S_n^{<\omega>} \cdot \vec{F}, \tag{1}$$

DOI: 10.1587/elex.1.92
Received April 15, 2004
Accepted May 06, 2004
Published June 10, 2004

where $S_n^{<\omega>} = \otimes \prod_{l=1}^{n} S_1^{<j_l>}$, $(0 \le j_l \le 4, <\omega>_{10}=<j_1 j_2 j_3 \dots j_n>_5)$, '$\otimes$' denotes Kronecker product [2], $S_1^{<0>} = \begin{bmatrix} 1&0&0&0&0 \\ 0&4&2&3&1 \\ 0&4&1&1&4 \\ 0&4&3&2&1 \\ 4&4&4&4&4 \end{bmatrix}$, and $S_1^{<j_l>} = S_1^{<0>}$ with column d ($d = \{0,1,2,3,4\}$) rearranged to column $((d+(4\times j_l))$ over GF(5)).

Definition 3 The FPRME in polarity ω of the function $f(\vec{x})$ is in the form of

$$f(x_1, x_2, \dots, x_n) = \sum_{i=0}^{5^n-1} c_i^{\omega} \left[\prod_{l=1}^{n} \hat{x}_l^{k_l} \right],$$

where $\hat{x}_l = ^{<j_l>}x_l$ is the literal of the l-th variable, $<i>_{10}=<k_1 k_2 \dots k_n>_5$, $k_l \in \{0,1,2,3,4\}$, and $c_i^{\omega} \in \{0,1,2,3,4\}$ is the i-th element of C^{ω}.

Definition 4 The polarity $\omega_a = \omega_{a1}\omega_{a2}\dots\omega_{ar-1}\omega_{ar}\omega_{ar+1}\dots\omega_{an}$ is an extended dual polarity for the polarity $\omega_b = \omega_{b1}\omega_{b2}\dots\omega_{br-1}\omega_{br}\omega_{br+1}\dots\omega_{bn}$ iff $\omega_{as} \neq \omega_{bs}$ for $s = r$ and $\omega_{as} = \omega_{bs}$, otherwise [4].

Definition 5 For an n-variable p-valued function, extended dual polarity route is an ordering of all p^n polarities in which each two successive polarities are extended dual polarities [4]. There are always more than one possible extended dual polarity routes for a given function.

Definition 6 From Eq. (1), the following relationship between any two spectral coefficient vectors with polarity numbers ω_a and ω_b can be derived:

$$C^{\omega_a} = \left(\otimes \prod_{l=1}^{n} \left(S_1^{<\omega_{al}>} \cdot T_1^{<\omega_{bl}>} \right) \right) \cdot C^{\omega_b}. \tag{2}$$

When ω_a and ω_b are extended dual polarities, Eq. (2) is simplified into

$$C^{\omega_a} = \left(I_{r-1} \otimes \left(S_1^{<\omega_{ar}>} \cdot T_1^{<\omega_{br}>} \right) \otimes I_{n-r} \right) \cdot C^{\omega_b}, \tag{3}$$

where I_u is an identity matrix of size $5^u \times 5^u$, $T_1^{<\omega_{bl}>} = \left(S_1^{<\omega_{bl}>} \right)^{-1}$, $T_1^{<\omega_{br}>} = \left(S_1^{<\omega_{br}>} \right)^{-1}$, and ω_a and ω_b are as in Definition 4.

3 Algorithm for calculating all FPRMEs from truth vector

Let us represent each product term $c_i^{\omega} \hat{x}_1^{k_1} \hat{x}_2^{k_2} \dots \hat{x}_n^{k_n}$ in the FPRME by an $(n+1)$-digit 5-valued string '$k_1 k_2 \dots k_n c_i^{\omega}$' and call it "term". Based on Eqs. (1) and (3), a recursive algorithm that generates all polarity FPRMEs from truth vector is derived. The algorithm contains two recursive loops which are given below as Algorithm 1 and Algorithm 2. Given a truth vector, the algorithm first replaces each truth vector element f_i $(0 \le i \le 5^n - 1)$ by an $(n+1)$-digit 5-valued string, i.e. truth vector term '$m_1 m_2 \dots m_n f_i$' where $<m_1 m_2 \dots m_n>_5=<i>_{10}$. The generated truth vector terms are then taken as the input for the first level recursion of Algorithm 1. Algorithm 1 has n recursion levels. At each level, the output is taken as input for the next recursion. The output of the last level recursion of Algorithm 1 is the polarity zero FPRME terms. After Algorithm 1 is done, Algorithm 2 is then executed, which generates all the nonzero polarities expansions terms for the input

DOI: 10.1587/elex.1.92
Received April 15, 2004
Accepted May 06, 2004
Published June 10, 2004

function one by one in a sequence determined by the extended dual polarity route. The terms for each nonzero polarity FPRME are determined based on the previous polarity FPRME terms according to the relation between the current polarity ω_a and the previous polarity ω_b digits. At the end of the algorithm, all 5^n FPRMEs for the given function are obtained. Moreover, the polarity number stored in ω_{best} is the optimal polarity number with $C_{\min}$ nonzero spectral coefficients.

Both Algorithm 1 and Algorithm 2 use matrix M below where the element located at the row y and column z is denoted by M_{yz} $(1 \leq y \leq 3,\ 1 \leq z \leq 4)$.

$$M = \begin{bmatrix} 2 & 4 & 1 & 3 \\ 3 & 1 & 4 & 2 \\ 4 & 3 & 2 & 1 \end{bmatrix}$$

The steps of the algorithms are:

Algorithm 1

Step 1: Generate truth vector terms for the given truth vector.

Step 2: Initialize l to n. Take the truth vector terms as the input.

Step 3: Generate initial terms '$k_1k_2 \ldots k_nc_i^{\omega}$' for the output. The initial terms are all the 5^n terms $(0 \leq i \leq 5^n - 1)$ with zero as the last digit.

Step 4: For each nonzero term (the term with nonzero last digit) of the input, generate its contribution to the output terms according to the rule given in Table I. Note that q in Table I may represent k or m depending on the input terms.

Step 5: Sum up the initial and all contributed terms. The summation is done by replacing terms with identical first n-digit values with a new term. The first n-digits of the new term are equal to those of replaced terms, while the last digit of the new term is the summation of all the replaced terms last digits over GF(5). After the summation the output terms are obtained.

Step 6: $l = l - 1$. If $l = 0$ go to Algorithm 2. Otherwise go back to Step 3 with current output as input.

Algorithm 2

Step 1: Initialization

- Initialize the polarity vector v to '00..0'.
- Initialize the polarity number ω to 0.
- Initialize the variables ω_{best} and $C_{\min}$ to 0 and the number of nonzero terms in polarity zero, respectively.

Step 2: Determine the next polarity vector v in the employed extended dual polarity route and set ω to the decimal equivalent of v.

Step 3: Generate polarity ω terms.

- List initial terms of polarity ω.
- Find the contribution of each nonzero term of previous polarity according to the rules given in Table II.
- Sum up all the terms for the polarity ω.
- If the number of nonzero terms of polarity ω, N is less than current value of $C_{\min}$, set $\omega_{\text{best}} = \omega$ and $C_{\min} = N$.

DOI: 10.1587/elex.1.92
Received April 15, 2004
Accepted May 06, 2004
Published June 10, 2004

Table I. Contributed terms rule for generation of polarity zero terms

Processed term	Contributed terms	Processed term	Contributed terms
$q_1 \ldots q_{l-1}0\, q_{l+1} \ldots q_n c$	$q_1 \ldots q_{l-1}0 q_{l+1} \ldots q_n\, c$	$q_1 \ldots q_{l-1}3\, q_{l+1} \ldots q_n c$	$q_1 \ldots q_{l-1}1 q_{l+1} \ldots q_n\, M_{2c}$
	$q_1 \ldots q_{l-1}4 q_{l+1} \ldots q_n\, M_{3c}$		$q_1 \ldots q_{l-1}2 q_{l+1} \ldots q_n\, c$
$q_1 \ldots q_{l-1}1\, q_{l+1} \ldots q_n c$	$q_1 \ldots q_{l-1}1 q_{l+1} \ldots q_n\, M_{3c}$		$q_1 \ldots q_{l-1}3 q_{l+1} \ldots q_n\, M_{1c}$
	$q_1 \ldots q_{l-1}2 q_{l+1} \ldots q_n\, M_{3c}$		$q_1 \ldots q_{l-1}4 q_{l+1} \ldots q_n\, M_{3c}$
	$q_1 \ldots q_{l-1}3 q_{l+1} \ldots q_n\, M_{3c}$	$q_1 \ldots q_{l-1}4\, q_{l+1} \ldots q_n c$	$q_1 \ldots q_{l-1}1 q_{l+1} \ldots q_n\, c$
	$q_1 \ldots q_{l-1}4 q_{l+1} \ldots q_n\, M_{3c}$		$q_1 \ldots q_{l-1}2 q_{l+1} \ldots q_n\, M_{3c}$
$q_1 \ldots q_{l-1}2\, q_{l+1} \ldots q_n c$	$q_1 \ldots q_{l-1}1 q_{l+1} \ldots q_n\, M_{1c}$		$q_1 \ldots q_{l-1}3 q_{l+1} \ldots q_n\, c$
	$q_1 \ldots q_{l-1}2 q_{l+1} \ldots q_n\, c$		$q_1 \ldots q_{l-1}4 q_{l+1} \ldots q_n\, M_{3c}$
	$q_1 \ldots q_{l-1}3 q_{l+1} \ldots q_n\, M_{2c}$		
	$q_1 \ldots q_{l-1}4 q_{l+1} \ldots q_n M_{3c}$		

Table II. Contribution of processed term for some combinations of ω_{ar} and ω_{br} values

ω_{ar}	ω_{br}	Processed term	Contributed terms	ω_{ar}	ω_{br}	Processed term	Contributed terms
0	4	$k_1 \ldots k_{r-1}0k_{r+1} \ldots k_n c$	$k_1 \ldots k_{r-1}0k_{r+1} \ldots k_n\, c$	0	3	$k_1 \ldots k_{r-1}0k_{r+1} \ldots k_n c$	$k_1 \ldots k_{r-1}0k_{r+1} \ldots k_n\, c$
1	0	$k_1 \ldots k_{r-1}1k_{r+1} \ldots k_n c$	$k_1 \ldots k_{r-1}0k_{r+1} \ldots k_n\, M_{3c}$	1	4	$k_1 \ldots k_{r-1}1k_{r+1} \ldots k_n c$	$k_1 \ldots k_{r-1}0k_{r+1} \ldots k_n\, M_{2c}$
2	1		$k_1 \ldots k_{r-1}1k_{r+1} \ldots k_n\, c$	2	0		$k_1 \ldots k_{r-1}1k_{r+1} \ldots k_n\, c$
3	2	$k_1 \ldots k_{r-1}2k_{r+1} \ldots k_n c$	$k_1 \ldots k_{r-1}0k_{r+1} \ldots k_n\, c$	3	1	$k_1 \ldots k_{r-1}2k_{r+1} \ldots k_n c$	$k_1 \ldots k_{r-1}0k_{r+1} \ldots k_n\, M_{3c}$
4	3		$k_1 \ldots k_{r-1}1k_{r+1} \ldots k_n\, M_{2c}$	4	2		$k_1 \ldots k_{r-1}1k_{r+1} \ldots k_n\, c$
			$k_1 \ldots k_{r-1}2k_{r+1} \ldots k_n\, c$				$k_1 \ldots k_{r-1}2k_{r+1} \ldots k_n\, c$
		$k_1 \ldots k_{r-1}3k_{r+1} \ldots k_n c$	$k_1 \ldots k_{r-1}0k_{r+1} \ldots k_n\, M_{3c}$			$k_1 \ldots k_{r-1}3k_{r+1} \ldots k_n c$	$k_1 \ldots k_{r-1}0k_{r+1} \ldots k_n\, M_{1c}$
			$k_1 \ldots k_{r-1}1k_{r+1} \ldots k_n\, M_{2c}$				$k_1 \ldots k_{r-1}1k_{r+1} \ldots k_n\, M_{1c}$
			$k_1 \ldots k_{r-1}2k_{r+1} \ldots k_n\, M_{1c}$				$k_1 \ldots k_{r-1}2k_{r+1} \ldots k_n\, M_{3c}$
			$k_1 \ldots k_{r-1}3k_{r+1} \ldots k_n\, c$				$k_1 \ldots k_{r-1}3k_{r+1} \ldots k_n\, c$
		$k_1 \ldots k_{r-1}4k_{r+1} \ldots k_n c$	$k_1 \ldots k_{r-1}0k_{r+1} \ldots k_n\, c$			$k_1 \ldots k_{r-1}4k_{r+1} \ldots k_n c$	$k_1 \ldots k_{r-1}0k_{r+1} \ldots k_n\, c$
			$k_1 \ldots k_{r-1}1k_{r+1} \ldots k_n\, c$				$k_1 \ldots k_{r-1}1k_{r+1} \ldots k_n\, M_{2c}$
			$k_1 \ldots k_{r-1}2k_{r+1} \ldots k_n\, c$				$k_1 \ldots k_{r-1}2k_{r+1} \ldots k_n\, M_{3c}$
			$k_1 \ldots k_{r-1}3k_{r+1} \ldots k_n\, c$				$k_1 \ldots k_{r-1}3k_{r+1} \ldots k_n\, M_{1c}$
			$k_1 \ldots k_{r-1}4k_{r+1} \ldots k_n\, c$				$k_1 \ldots k_{r-1}4k_{r+1} \ldots k_n\, c$
0	1	$k_1 \ldots k_{r-1}0k_{r+1} \ldots k_n c$	$k_1 \ldots k_{r-1}0k_{r+1} \ldots k_n\, c$	0	2	$k_1 \ldots k_{r-1}0k_{r+1} \ldots k_n c$	$k_1 \ldots k_{r-1}0k_{r+1} \ldots k_n\, c$
1	2	$k_1 \ldots k_{r-1}1k_{r+1} \ldots k_n c$	$k_1 \ldots k_{r-1}0k_{r+1} \ldots k_n\, c$	1	3	$k_1 \ldots k_{r-1}1k_{r+1} \ldots k_n c$	$k_1 \ldots k_{r-1}0k_{r+1} \ldots k_n\, M_{1c}$
2	3		$k_1 \ldots k_{r-1}1k_{r+1} \ldots k_n\, c$	2	4		$k_1 \ldots k_{r-1}1k_{r+1} \ldots k_n\, c$
3	4	$k_1 \ldots k_{r-1}2k_{r+1} \ldots k_n c$	$k_1 \ldots k_{r-1}0k_{r+1} \ldots k_n\, c$	3	0	$k_1 \ldots k_{r-1}2k_{r+1} \ldots k_n c$	$k_1 \ldots k_{r-1}0k_{r+1} \ldots k_n\, M_{3c}$
4	0		$k_1 \ldots k_{r-1}1k_{r+1} \ldots k_n\, M_{1c}$	4	1		$k_1 \ldots k_{r-1}1k_{r+1} \ldots k_n\, M_{3c}$
			$k_1 \ldots k_{r-1}2k_{r+1} \ldots k_n\, c$				$k_1 \ldots k_{r-1}2k_{r+1} \ldots k_n\, c$
		$k_1 \ldots k_{r-1}3k_{r+1} \ldots k_n c$	$k_1 \ldots k_{r-1}0k_{r+1} \ldots k_n\, c$			$k_1 \ldots k_{r-1}3k_{r+1} \ldots k_n c$	$k_1 \ldots k_{r-1}0k_{r+1} \ldots k_n\, M_{2c}$
			$k_1 \ldots k_{r-1}1k_{r+1} \ldots k_n\, M_{2c}$				$k_1 \ldots k_{r-1}1k_{r+1} \ldots k_n\, M_{1c}$
			$k_1 \ldots k_{r-1}2k_{r+1} \ldots k_n\, M_{2c}$				$k_1 \ldots k_{r-1}2k_{r+1} \ldots k_n\, c$
			$k_1 \ldots k_{r-1}3k_{r+1} \ldots k_n\, c$				$k_1 \ldots k_{r-1}3k_{r+1} \ldots k_n\, c$
		$k_1 \ldots k_{r-1}4k_{r+1} \ldots k_n c$	$k_1 \ldots k_{r-1}0k_{r+1} \ldots k_n\, c$			$k_1 \ldots k_{r-1}4k_{r+1} \ldots k_n c$	$k_1 \ldots k_{r-1}0k_{r+1} \ldots k_n\, c$
			$k_1 \ldots k_{r-1}1k_{r+1} \ldots k_n\, M_{3c}$				$k_1 \ldots k_{r-1}1k_{r+1} \ldots k_n\, M_{1c}$
			$k_1 \ldots k_{r-1}2k_{r+1} \ldots k_n\, c$				$k_1 \ldots k_{r-1}2k_{r+1} \ldots k_n\, M_{3c}$
			$k_1 \ldots k_{r-1}3k_{r+1} \ldots k_n\, M_{3c}$				$k_1 \ldots k_{r-1}3k_{r+1} \ldots k_n\, M_{2c}$
			$k_1 \ldots k_{r-1}4k_{r+1} \ldots k_n\, c$				$k_1 \ldots k_{r-1}4k_{r+1} \ldots k_n\, c$

Step 4: If the terms for all polarities have been generated exit the algorithm. Otherwise go back to Step 2.

4 Experimental results

The calculations of all the FPRMEs for a given 5-valued function using the proposed algorithm (employing the extended dual polarity route in which $\omega_{ar} + \omega_{br}$ is always one or four) as well as by the direct and fast versions of Eq. (2) in lexicographic order [6] have been implemented as C++ programs and run on a 500 MHz, 256 MB RAM Pentium III computer. Their execution results in term of execution time for some 5-valued test files are presented

DOI: 10.1587/elex.1.92
Received April 15, 2004
Accepted May 06, 2004
Published June 10, 2004

Table III. Execution times for some 5-valued test files (in seconds)

Input file name	Number of inputs	Number of outputs	Lexicographic order (direct)	Lexicographic order (fast)	Extended dual polarity
9sym	3	1	3.22	3.29	3.27
apex4	3	7	8.91	8.95	8.49
clip	3	2	4.09	4.08	3.98
con1	3	1	3.23	3.21	3.17
ex1010	4	4	124.60	73.37	22.17
ex5	3	21	21.20	21.29	20.69
inc	3	3	4.95	4.91	4.88
misex1	3	3	5.17	5.04	4.87
rd84	3	2	4.13	4.09	3.92
squar5	2	3	2.20	2.21	2.21
xor5	2	1	1.49	1.49	1.48
z5xp1	3	4	5.84	5.83	5.78
z9sym	3	1	3.20	3.21	3.17

in Table III. The 5-valued test files that are used here are binary MCNC benchmarks that have been modified to represent 5-valued functions. Their generation was described in [6].

It can be seen from Table III that the proposed algorithm is faster, and hence more efficient than the other methods for most of the functions. This is to some extent due to the reduced number of additions resulting from the use of extended dual polarity routes instead of lexicographic order. The other reason for the efficiency of the algorithm comes from the facts that this algorithm only processes nonzero terms instead of all terms as in the other methods and that it does not build any transform matrices. In addition to eliminating the time spent on building transform matrices, the latter reason also means that the algorithm has low storage requirement as it does not need to store the transform matrix, which grows rapidly with an increase in the number of variables. At any time, the only storage space required is for the terms of current and previous polarities (which grows less rapidly with an increase in the number of variables compared to those required to store the transform matrix), v, ω_a, ω_b, ω_{best}, and C_{min}. Furthermore, as the number of input variables becomes larger, the increase in the time required to build the transform matrix is larger than the increase in the time required for determining the contributions of each nonzero term to the output terms. This in turn results in the larger difference between the execution times of the other algorithms and that of the proposed algorithm, which can be noticed from the numbers in Table III.

5 Conclusions

An algorithm that efficiently generates all FPRMEs for 5-valued functions has been presented. The proposed algorithm takes the truth vector of functions defined over GF(5) as its inputs, calculates the polarity zero FPRME from the truth vector, and then continues to generate the rest of the polynomial expansions one by one. The algorithm is simple, requires small amount of storage and is computationally efficient when compared with other methods as shown by experimental results.

DOI: 10.1587/elex.1.92
Received April 15, 2004
Accepted May 06, 2004
Published June 10, 2004