Classification and Properties of Fast Linearly Independent Logic Transformations

Bogdan J. Falkowski, *Senior Member, IEEE,* and Susanto Rahardja, *Member, IEEE*

***Abstract*—The existence of numerous number of linearly independent (LI) transformations in GF(2) algebra finds application in the design of exclusive-or based polynomial expansions. For a chosen LI matrix transformation, such expansion gives a canonical representation of an arbitrary completely specified logical function. In this paper, family of LI transformations is introduced which possesses fast forward and inverse butterfly diagrams. These transforms are recursively defined and grouped into classes where consistent formulas relating forward and inverse transform matrices are obtained. The classification is further extended into various LI transforms with horizontal and vertical permutations. The possibility of easy implementation of polynomial expansions based on classified LI logic transformations in the form of readily available fine grain FPGA's and EPLD's is also illustrated.**

***Index Terms*—Fast transforms, linearly independent logic, logic polynomial expansions, Reed–Muller transform.**

I. Introduction

THE IDEA to represent switching circuits in different algebra has evolved in recent years. Reed–Muller transform utilizes algebra of GF(2) and any switching function may be completely realized by the modulo-2 sum-of-products expression which is known as the *complement-free ring-sum* [3]. For many important functions that are nonunate (e.g., parity checkers, adders and multipliers), Reed–Muller realizations are advantageous when area, speed, and testability are of main concern [23]. For such linear functions, they allow surprisingly complex designs to be implemented using very few product terms. When each variable throughout Reed–Muller expansion assumes either true or complemented form, such an expression is known as *Fixed Polarity Reed–Muller Expansion* (FPRME). It has been shown in the literature that for some switching functions, more efficient implementations can be obtained when each variable in Reed–Muller expansion can be both in affirmation and negation. The latter expansion is called *Mixed Polarity Reed–Muller Expansion* (MPRME).

The most general concept of a binary *Linearly Independent (LI) Logic* was introduced in [15], and expanded to a multiple-valued case in [17]. For its various properties and special cases see [8], [9], [12]–[17]. A theorem from [15] generalizes all binary logic circuits that are realized in GF(2) algebra. It has been shown there that for any LI set of basis switching functions of n variables represented as a $2^n \times 2^n$ matrix M_n, there exists a canonical three-level realization $f = A_0 g_0 \oplus \cdots \oplus A_i g_i \oplus \cdots \oplus A_{2^n-1} g_{2^n-1}, 0 \le i \le 2^n - 1$, where functions g_i are the given LI basis functions and coefficients A_i are determined by multiplying matrix M_n^{-1} by the truth vector of the function f. With such definition, FPRME is just a special case of the new LI logic. Due to the fact that the selected LI functions can appear in both true and complemented forms, many MPRME can also be derived from LI Logic. Since the number of all LI functions is very large even for the case of few variables, efficient ways of finding those transform matrices that have fast algorithms is of great importance.

In the current paper, ways of generation of fast transforms for binary 2^n-dimensional LI transformation matrices are introduced. These LI Transforms are shown which may be created efficiently in the form of transform matrices for $n = 2$. Since the recursive equations for expanding the transform matrices are provided, then the results presented for $n = 2$ can be easily extended to transform matrices with higher dimensions giving the best sets of basis functions for such cases as well. Hence this paper introduces many new families of basis functions for 2^n-dimensional LI transformation matrices which should be used in finding their polynomial expansions and resulting hardware implementations rather than more general but computationally inefficient approach based on the matrix operations in earlier papers [15]–[17]. It is obvious that the fast transforms exist only for some basis functions and they constitute a small fraction of all possible LI basis function for $n > 2$. Such fast transform matrices are classified under specific mathematical relations between its forward and inverse transforms. Finally, those LI transform matrices which require permutations for both the Forward and Inverse transforms to obtain fast algorithms are also shown. Similarly to arbitrary LI logic transformations [12]–[17], polynomial expansions obtained from fast LI Transforms can always be implemented in the form of fine grain FPGA's or EPLD devices. Hence, the fast LI transform matrices presented in this paper should be used as the basis for LI expansions, rather than the arbitrary LI transform matrices that in most cases do not have recursive fast decomposition and require computationally expensive matrix inversion and multiplication to obtain the coefficients.

II. General Definitions of GF(2) Linearly Independent Logic

Definition 1: Let M_n be a $2^n \times 2^n$ matrix with columns corresponding to minterms and rows corresponding to some switching functions of n variables. If the sets of rows are

Manuscript received May 22, 1995; revised February 22, 1996. This paper was recommended by Associate Editor E. G. Friedman.

The authors are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Republic of Singapore.

Publisher Item Identifier S 1057-7130(97)03641-0.

TABLE I

No.	Symbol	Switching function
1	$g_{(1)}$	x_2x_1
2	$g_{(2)}$	$x_2\overline{x_1}$
3	$g_{(3)}$	x_2
4	$g_{(4)}$	$\overline{x_2}x_1$
5	$g_{(5)}$	x_1
6	$g_{(6)}$	$x_2 \oplus x_1$
7	$g_{(7)}$	$x_2 \vee x_1$
8	$g_{(8)}$	$\overline{x_2}\ \overline{x_1}$
9	$g_{(9)}$	$\overline{x_2 \oplus x_1}$
10	$g_{(10)}$	$\overline{x_1}$
11	$g_{(11)}$	$x_2 \vee \overline{x_1}$
12	$g_{(12)}$	$\overline{x_2}$
13	$g_{(13)}$	$\overline{x_2} \vee x_1$
14	$g_{(14)}$	$\overline{x_2x_1}$
15	$g_{(15)}$	1

linearly independent with respect to XOR operations (i.e., rows are bit-by-bit XORed), then M_n has only one inverse in GF(2) and is said to be *linearly independent*.

Lemma 1 [16]: Let $k = \mathrm{GF}(q)$ be the Galois Field with q elements. The order of the group of all nonsingular m-by-m matrices with entries in the field k is

$$q^{(1/2)m(m-1)} \prod_{i=1}^{m} (q^i - 1). \tag{1}$$

Lemma 2 [16]: Let $\vec{x_n} = \{x_n, x_{n-1}, \cdots x_2, x_1\}$ and $f(\vec{x_n})$ be a switching function of n variables. The number of all possible XOR canonical representations of the functions is

$$\frac{2^{(2^n-1)(2^n-1)}}{2^n!} \prod_{i=1}^{2^n} (2^i - 1). \tag{2}$$

The derivation of a family of recursive LI logic commences from basic 4×4 matrices, which may be recursively defined. This ensures the existence of fast algorithms. For such basic matrices, combinations of canonical representations of 2-variable switching functions are adequate for the generation of all existing *recursive* XOR canonical forms for an arbitrary n-variable switching functions. In order not to loose any information about the transformed functions, two different columns in LI matrices cannot be identical, also the column with zero entries is not allowed. Table I gives the set of switching functions for $n = 2$.

The LI Transform based on Definition 1 and Lemma 1 can be described by the following general formulas performed in modulo-2 algebra:

$$M_n \cdot \vec{A} = \vec{F} \tag{3}$$

$$M_n^{-1} \cdot \vec{F} = \vec{A} \tag{4}$$

where $\vec{F} = [F_0, F_1, \cdots, F_{2^n-1}]^T$ is a column vector defining the truth vector of a switching function $f(\vec{x_n})$ in a natural binary ordering, M_n is an LI matrix of order $N = 2^n$ defined by any LI set of n-variable switching functions and $\vec{A} = [A_0, A_1, \cdots, A_{2^n-1}]^T$ is the coefficient column vector for the particular transform matrix M_n with modulo-2 inverse M_n^{-1}.

In particular, (3) may be expressed as

$$f(\vec{x_n}) = \sum_{i=0}^{2^n-1} A_i g_i \tag{5}$$

where g_i is any set of n-variable switching functions such that the matrix

$$M_n = [\vec{g_0}, \vec{g_1}, \cdots, \vec{g_{2^n-1}}]$$

where $\vec{g_i}$ represents the truth vector of the switching functions $0 \le i \le 2^n - 1$ and the symbol Σ is the addition in modulo-2.

Example 1: Let with $f(\vec{x_2}) = \overline{x_2x_1}$ with $\vec{F} = [1,1,1,0]^T$. Let the matrix

$$M_2 = [\vec{g_0}, \vec{g_1}, \vec{g_2}, \vec{g_3}] = [\vec{g_{(6)}}, \vec{g_{(11)}}, \vec{g_{(2)}}, \vec{g_{(3)}}]$$

where

$$M_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad M_2^{-1} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

i.e., the basis LI functions are

$$g_0 = x_1 \oplus x_2, \quad g_1 = \overline{x_1} \vee x_2, \quad g_2 = x_2\overline{x_1} \quad \text{and} \quad g_3 = x_2.$$

The inverse of M_2 may be evaluated using Gaussian Elimination. By (4)

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}.$$

By (5)

$$\begin{aligned} f(\vec{x_2}) &= A_0g_0 \oplus A_1g_1 \oplus A_2g_2 \oplus A_3g_3 \\ &= (x_1 \oplus x_2) \oplus (\overline{x_1} \vee x_2) \oplus x_2. \end{aligned}$$

To obtain the value of the original function for minterm 0 by using its LI Polynomial Expansion, we have

$$f(\vec{0_2}) = f(0,0) = (0 \oplus 0) \oplus (\overline{0} \vee 0) \oplus 0 = 0 \oplus 1 \oplus 0 = 1.$$

Similarly all other values of the original function can be obtained from this expansion.

In the continuation, the following notation for matrices will be used repetitively.

- The matrix M_n is a $2^n \times 2^n$ square matrix such that it is recursively defined by

$$M_n = \begin{bmatrix} M_{n-1}^{(1)} & M_{n-1}^{(2)} \\ M_{n-1}^{(3)} & M_{n-1}^{(4)} \end{bmatrix} \tag{6}$$

  where each submatrix $M_{n-1}^{(i)}, i \in \{1, 2, 3, 4\}$, has a dimension of $2^{n-1} \times 2^{n-1}$, contains one recursive

equation which is either O_{n-1}, X_{n-1} or M_{n-1}, where O_{n-1} is a $2^{n-1} \times 2^{n-1}$ matrix with all its elements 0, $X_{n-1} = I_{n-1}$ or J_{n-1}, I_{n-1} is a $2^{n-1} \times 2^{n-1}$ identity matrix and J_{n-1} a $2^{n-1} \times 2^{n-1}$ reverse-identity matrix, i.e., elements in the reverse-diagonal positions are 1 and 0 at others.

- The matrix M_n is a $2^n \times 2^n$ square matrix such that it can be partitioned into two vertical submatrices, each with dimension $2^n \times 2^{n-1}$, i.e.,

$$M_n = [M_V^{(1)} \quad M_V^{(2)}] \tag{7}$$

or partitioned into two horizontal submatrices, each with dimension $2^{n-1} \times 2^n$, i.e.,

$$M_n = \begin{bmatrix} M_H^{(1)} \\ M_H^{(2)} \end{bmatrix} \tag{8}$$

where the subscript V or H denote the respective Vertical or Horizontal partitioning of the original matrix M_n.

Definition 2: Let M_n be a nonsingular square matrix which is partitioned into four appropriate $2^{n-1} \times 2^{n-1}$ dimensional submatrices as shown in (6).

The α_0 operator on the matrix M_n is defined as interchanging the diagonal submatrices

$$\alpha_0[M_n] = \begin{bmatrix} M_{n-1}^{(4)} & M_{n-1}^{(2)} \\ M_{n-1}^{(3)} & M_{n-1}^{(1)} \end{bmatrix}. \tag{9}$$

Definition 3: Let M_n be a nonsingular square matrix which is partitioned into four appropriate $2^{n-1} \times 2^{n-1}$ dimensional submatrices as shown in (6).

The α_1 operator on the matrix M_n is defined as interchanging the reverse-diagonal submatrices

$$\alpha_1[M_n] = \begin{bmatrix} M_{n-1}^{(1)} & M_{n-1}^{(3)} \\ M_{n-1}^{(2)} & M_{n-1}^{(4)} \end{bmatrix}. \tag{10}$$

Definition 4: Let M_n be a nonsingular square $2^n \times 2^n$ matrix with submatrices built of basic nonsingular matrices of dimension $2^{n-1} \times 2^{n-1}$ as shown in (6), where $M_{n-1}^{(i)} \in \{O_{n-1}, X_{n-1}, M_{n-1}\}$. The matrix M_n consists of only one O_{n-1} submatrix, at least one M_{n-1} submatrix not diagonally opposite to O_{n-1}, plus two other submatrices which are either X_{n-1} or M_{n-1}, and the two submatrices contain any combinations of the submatrices X_{n-1} or M_{n-1}. The β operator on M_n is defined as selecting the submatrix diagonally opposite to O_{n-1} and interchange it by the following operations:

$$X_{n-1} \leftrightarrow M_{n-1} \quad \text{if } M_{n-1} \text{ and } X_{n-1} \text{ are submatrices in } M_n$$

or

$$M_{n-1} \leftrightarrow X_{n-1} \quad \text{if } X_{n-1} \text{ is a submatrix in } M_n. \tag{11}$$

Example 2: Let a nonsingular matrix be defined recursively as

$$M_n = \begin{bmatrix} J_{n-1} & O_{n-1} \\ J_{n-1} & M_{n-1} \end{bmatrix}.$$

Then

$$M_1 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \quad \text{and} \quad M_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}.$$

The β operation on M_n, by (11) of Definition 4, is

$$\beta[M_n] = \begin{bmatrix} J_{n-1} & O_{n-1} \\ M_{n-1} & M_{n-1} \end{bmatrix}$$

i.e.,

$$\beta[M_1] = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \quad \text{and} \quad \beta[M_2] = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

Definition 5: Let M_n be any nonsingular square $2^n \times 2^n$ matrix such that M_n is defined by (7). The μ_H operator on M_n is defined as interchanging the two partitioned $2^n \times 2^{n-1}$ submatrices

$$\mu_H[M_n] = [M_V^{(2)} \quad M_V^{(1)}]. \tag{12}$$

Definition 6: Let M_n be any nonsingular square $2^n \times 2^n$ matrix such that M_n is defined by (8). The μ_V operator on M_n is defined as interchanging the two partitioned $2^{n-1} \times 2^n$ submatrices

$$\mu_V[M_n] = \begin{bmatrix} M_H^{(2)} \\ M_H^{(1)} \end{bmatrix}. \tag{13}$$

Definition 7: Let M_n be any nonsingular square $2^n \times 2^n$ matrix such that M_n is *recursively* defined by (6) and (7), i.e.,

$$M_n = [M_V^{(1)} \quad M_V^{(2)}] = \begin{bmatrix} M_{n-1}^{(1)} & M_{n-1}^{(2)} \\ M_{n-1}^{(3)} & M_{n-1}^{(4)} \end{bmatrix}.$$

The μ_{EH} operator on M_n is defined as grouping the recursive equations in the submatrices vertically and interchanging them in the submatrices horizontally

$$\mu_{EH}[M_n] = \begin{bmatrix} M_{n-1}^{(2)} & M_{n-1}^{(1)} \\ M_{n-1}^{(4)} & M_{n-1}^{(3)} \end{bmatrix} \equiv [M_V^{(2)} \quad M_V^{(1)}]. \tag{14}$$

Definition 8: Let M_n be any nonsingular square $2^n \times 2^n$ matrix such that M_n is *recursively* defined by (6) and (8), i.e.,

$$M_n = \begin{bmatrix} M_H^{(1)} \\ M_H^{(2)} \end{bmatrix} = \begin{bmatrix} M_{n-1}^{(1)} & M_{n-1}^{(2)} \\ M_{n-1}^{(3)} & M_{n-1}^{(4)}. \end{bmatrix}.$$

The μ_{EV} operator on M_n is defined as grouping the recursive equations in the submatrices horizontally and interchanging the equations in the submatrices vertically

$$\mu_{EV}[M_n] = \begin{bmatrix} M_{n-1}^{(3)} & M_{n-1}^{(4)} \\ M_{n-1}^{(1)} & M_{n-1}^{(2)} \end{bmatrix} \equiv \begin{bmatrix} M_H^{(2)} \\ M_H^{(1)} \end{bmatrix}. \tag{15}$$

III. Family of Fast Transforms for Linearly Independent Logic

From (2) of Lemma 2, it can be shown that there are altogether 840 LI matrices derived from combinations of 2-variable switching functions. Table I shows the list of such 2-variable switching functions that can be used as column entries in LI matrices. Combinations of four basic switching functions which satisfy condition in Definition 1 will form a basic LI matrix over GF(2). The main interest is on those LI matrices which can be expressed in recursive mathematical equations and possess both fast forward and inverse transforms. This, as mentioned earlier, will lead to existence of fast algorithms and recursive butterfly structures for any number of variables, n. For transform matrices that are based on GF(2) algebra, there are altogether 44 LI matrices, in which the forward and inverse transform matrices may be constructed effortlessly. Such results have been obtained computationally. The first 28 transform matrices are mentioned by the same authors in [8]. The fast algorithms of those LI transform matrices are efficiently constructed. In this paper, the family of LI Transform matrices which can be formulated recursively, is discussed in detail. Mathematical operations are developed, allowing transformation of these 44 LI matrices into another classes of LI matrices which require horizontal or vertical permutations so as to possess similar property of recursiveness.

There are only 44 LI Transform matrices which can be recursively defined. They are constructed by the basis recursive submatrices O_{n-1}, M_{n-1}, and X_{n-1}, where at most one O_{n-1} submatrix and two X_{n-1} submatrices could appear in the recursive definitions. In addition, there must be at least one M_{n-1} in the recursive equation. Figs. 1–4 list out these LI transforms which do not require any horizontal or vertical permutations. In Fig. 1, those LI transforms that have identical fast forward and inverse transforms, i.e., $M_n = M_{n-1}$, are shown. This is categorized as Class A of LI Logic. In this class of LI Logic, the submatrices consist of no reverse-identity matrix J_{n-1}. Moreover, the submatrix O_{n-1} lies in either $M_{n-1}^{(2)}$ or $M_{n-1}^{(3)}$ submatrices of (6) and $M_{n-1}^{(1)} = M_{n-1}^{(4)}$. As it can be seen, Fixed Polarity Reed–Muller transform belongs to this class of LI Logic. Moving the submatrix O_{n-1} into either $M_{n-1}^{(1)}$ or $M_{n-1}^{(4)}$ will bring the LI Transform matrices into Class B. There are altogether 6 LI transforms in Class B. In this class, the submatrices consist of no identity matrix I_{n-1}, and the forward and inverse transform matrices are related by the following Property 1, with $M_{n-1}^{(2)} = M_{n-1}^{(3)}$.

Property 1: Let $M_B = M_n$ be one of the nonsingular square matrices in Class B. Then, the inverse of M_B is given by

$$M_B^{-1} = \alpha_0[M_B]. \tag{16}$$

Different combinations and permutations of identity matrix I_{n-1} with M_{n-1} and reverse-identity matrix J_{n-1} with M_{n-1} yield another class of LI Transforms. In this class, the position of the zero submatrix O_{n-1} lies in either $M_{n-1}^{(2)}$ and $M_{n-1}^{(3)}$. This categorizes Class C of LI Logic and for matrices from this class, Property 2 is satisfied.

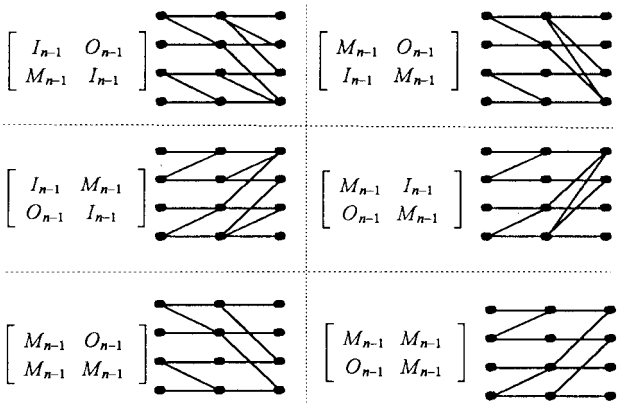

Fig. 1. Class A.

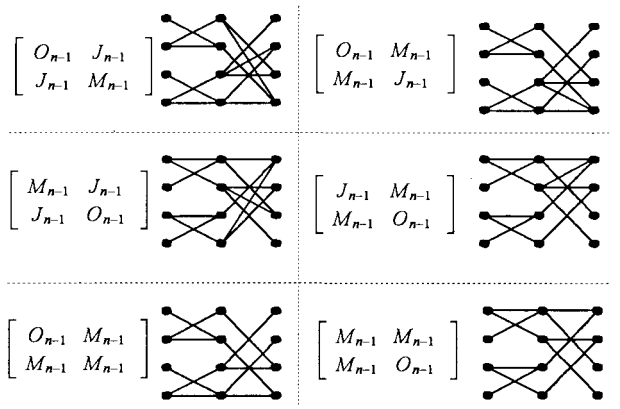

Fig. 2. Class B.

Property 2: Let $M_C = M_n$ be one of the nonsingular square matrices in Class C. Then, the inverse of M_C is given by

$$M_C^{-1} = \beta[M_C]. \tag{17}$$

Fig. 3 shows the list of 16 LI Transforms in Class C. Shifting the zero submatrix O_{n-1} into either $M_{n-1}^{(1)}$ or $M_{n-1}^{(4)}$, and keeping the respective combinations and permutations of identity matrix I_{n-1} with M_{n-1} and reverse-identity matrix J_{n-1} with M_{n-1} from Class C produces Class D of LI Logic. The following property of Class D may be derived.

Property 3: Let $M_D = M_n$ be one of the nonsingular square matrices in Class D. Then, the inverse of M_D is given by

$$M_D^{-1} = \alpha_0\alpha_1\beta[M_D]. \tag{18}$$

Property 4: Let M_C and M_D represent any nonsingular square matrix in Class C and Class D, respectively. Then,

$$\mu_{EH}[M_C] \in M_D \tag{19}$$

and

$$\mu_{EV}[M_C] \in M_D. \tag{20}$$

The μ_{EH} or μ_{EV} operations on the matrix is a one to one and onto mapping. Hence, there are similarly 16 LI Transforms in Class D. The following mathematical relationship may be derived.

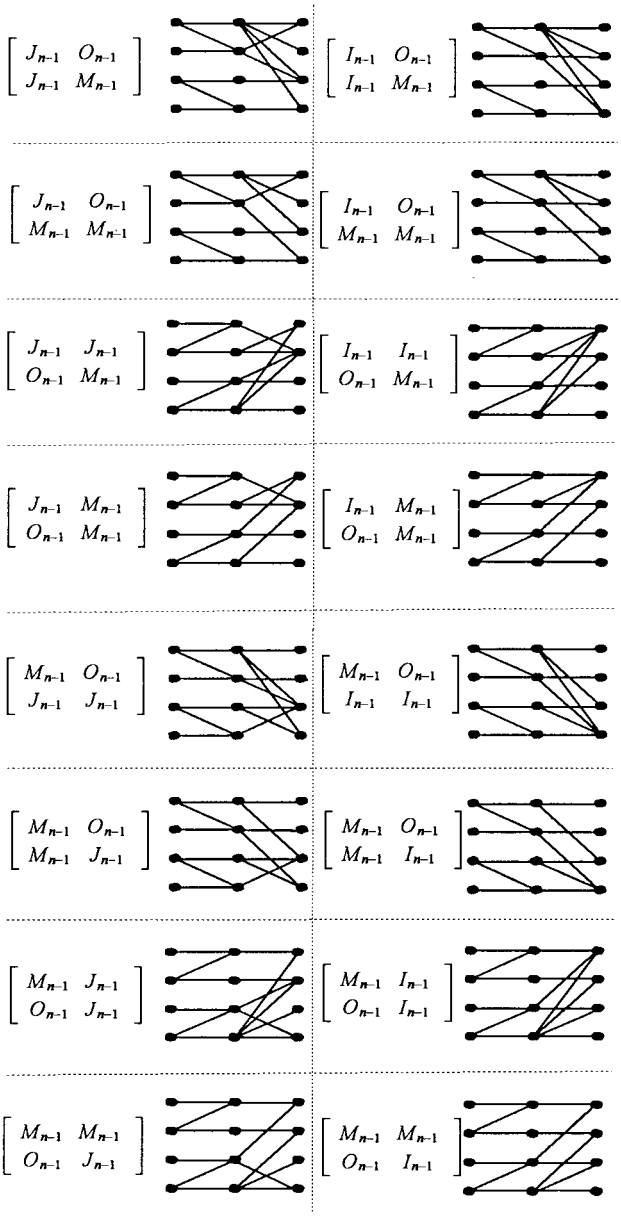

Fig. 3. Class C.

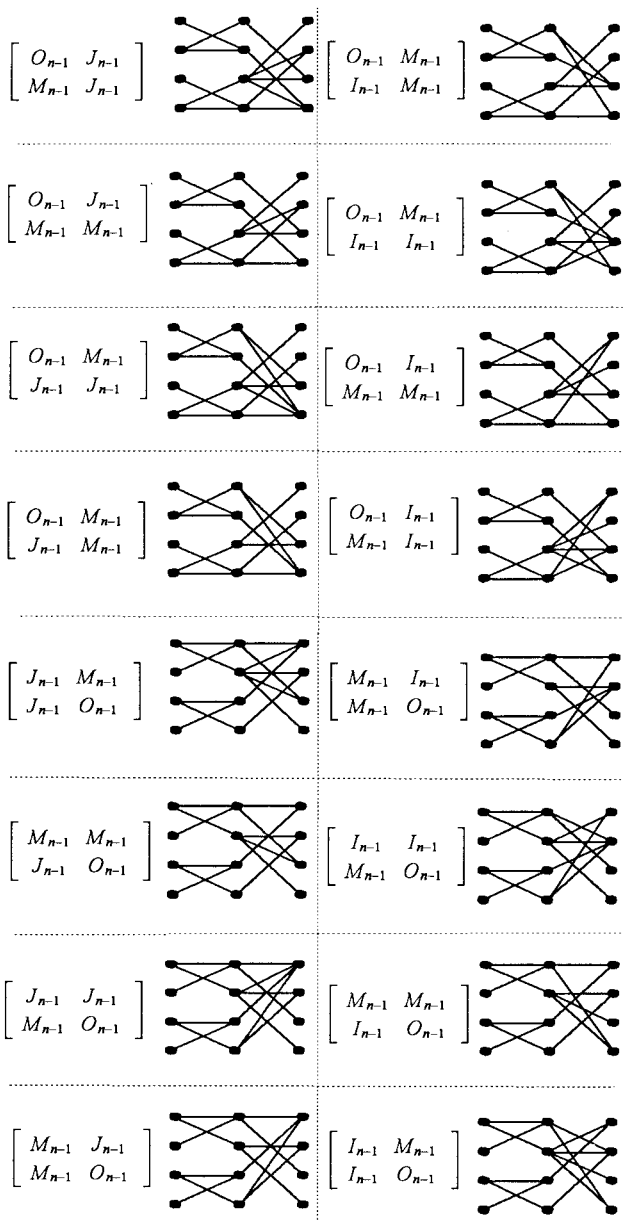

Fig. 4. Class D.

Property 5: Let M_C and M_D represent any nonsingular square matrix in Class C and Class D, respectively. If $M_D = \mu_{EH}[M_C]$ or $M_D = \mu_{EV}[M_C]$, then from Properties 2–4,

$$M_D^{-1} = \mu_{EV}[M_C^{-1}] \tag{21}$$

or

$$M_D^{-1} = \mu_{EH}[M_C^{-1}] \tag{22}$$

respectively.

Proof: Let $M_C = M_n$ and $M_D = \mu_{EH}[M_n]$. From (18)

$$M_D^{-1} = \alpha_0\alpha_1\beta[M_D] = \alpha_0\alpha_1\beta[\mu_{EH}[M_n]].$$

Interchanging β and μ_{EH}

$$M_D^{-1} = \alpha_0\alpha_1\mu_{EH}[\beta[M_n]].$$

From (17) of Property 2,

$$M_D^{-1} = \alpha_0\alpha_1\mu_{EH}[M_C^{-1}] = \mu_{EV}[M_C^{-1}].$$

Equation (22) of Property 5 may be proved similarly. QED.

Classes A, B, C, and D of LI Logic form the basic LI Transforms which possess fast algorithms in forward and inverse transform. They do not require any horizontal or vertical permutations for the existence of their recursiveness. Those 2 permutations, horizontal and vertical, are introduced earlier by Definitions 5, 7 and Definitions 6, 8, respectively. The LI transforms are classified according to their mathematical relationship of forward transform matrices with their respective inverse transform matrices. As shown from all the basic LI transforms, they are constructed from four basic submatrices, namely the zero submatrix O_{n-1}, the identity submatrix I_{n-1}, the reverse-identity submatrix J_{n-1} and the recursive submatrix M_{n-1} in which general rules are applied to them, following their classifications. The general computational complexity of each LI transforms depends solely on the construction of the matrices from the basic

submatrices. Broadly speaking, LI transforms with matrices constructed from only the recursive submatrix M_{n-1} and the zero submatrix O_{n-1} yield least computational costs. This implies that fixed polarity Reed–Muller transform belongs to this category of LI transforms with lowest computational complexity. Replacement of any submatrix M_{n-1} with X_{n-1}, regardless whether it is an identity matrix I_{n-1} or reverse-identity matrix J_{n-1}, increases the computational costs of the transform. For $n=2$, it can be shown that LI transforms with one or two submatrices of either I_{n-1} or J_{n-1} will increase the computational costs of the transform by 1 or 2 modulo-2 additions, respectively. However, it does not imply that LI transforms having one or two submatrices of either I_{n-1} or J_{n-1} are worse than the Reed–Muller transforms from the implementation point of view, when LI polynomial expansions for some logical functions are considered what is shown in the following example.

Example 3: Let

$$f(\vec{x_3}) = \overline{x_3}\,\overline{x_2} \vee x_3\overline{x_2} \vee x_3\overline{x_1}$$

with

$$\vec{F} = [1,1,0,0,1,1,1,0]^T.$$

Let LI transform matrix M_n of order 2^n belong to Class C, with recursive equation defined as

$$M_n = \begin{bmatrix} J_{n-1} & O_{n-1} \\ J_{n-1} & M_{n-1} \end{bmatrix}.$$

Applying μ_{EV} operation to M_n brings this matrix to Class D, according to (20) of Property 4 i.e.,

$$M_D = \mu_{EV}[M_C] = \begin{bmatrix} J_{n-1} & M_{n-1} \\ J_{n-1} & O_{n-1} \end{bmatrix}.$$

For the transform matrix of order 2^3,

$$M_3 = M_C = \begin{bmatrix} 0&0&0&1&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 0&0&0&1&0&1&0&0 \\ 0&0&1&0&1&0&0&0 \\ 0&1&0&0&0&1&1&0 \\ 1&0&0&0&1&0&1&1 \end{bmatrix}.$$

From (17) of Property 2 and by (4)

$$\vec{A_C} = M_C^{-1} \cdot \vec{F} = \beta[M_C] \cdot \vec{F} = \begin{bmatrix} 0&0&0&1&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&1&0&0 \\ 1&0&0&0&1&0&0&0 \\ 1&0&1&0&1&0&1&0 \\ 1&1&1&1&1&1&1&1 \end{bmatrix} \cdot \begin{bmatrix} 1\\1\\0\\0\\1\\1\\1\\0 \end{bmatrix} = \begin{bmatrix} 0\\0\\1\\1\\0\\0\\1\\1 \end{bmatrix}.$$

From (22) of Property 5

$$\vec{A_D} = M_D^{-1} \cdot \vec{F} = \mu_{EH}[M_C^{-1}] \cdot \vec{F} = \begin{bmatrix} 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&1&0&0&0&1 \\ 0&0&1&0&0&0&1&0 \\ 0&1&0&1&0&1&0&1 \\ 1&1&1&1&1&1&1&1 \end{bmatrix} \cdot \begin{bmatrix} 1\\1\\0\\0\\1\\1\\1\\0 \end{bmatrix} = \begin{bmatrix} 0\\1\\1\\1\\0\\1\\0\\1 \end{bmatrix}.$$

It is clear that for the particular truth vector, the transform matrix in Class C results in a simpler LI Spectrum with more number of zeros than that of Class D. An LI transform matrix from Class B which has the same computational costs as the two transform matrices selected earlier is chosen for comparison of the spectra. The recursive transform matrices are

$$M_B = \begin{bmatrix} O_{n-1} & J_{n-1} \\ J_{n-1} & M_{n-1} \end{bmatrix} \quad \text{and} \quad M_A = \begin{bmatrix} M_{n-1} & O_{n-1} \\ M_{n-1} & M_{n-1} \end{bmatrix}$$

respectively. The resulting LI Logic spectra for the same switching function are

$$\vec{A_B} = [0,0,0,0,0,0,1,1]^T \quad \text{and}$$
$$\vec{A_A} = [1,0,1,0,0,0,1,1]^T$$

respectively. Comparing $\vec{A_A}, \vec{A_B}, \vec{A_C}$, and $\vec{A_D}$, it can be seen that the optimal LI Logic spectrum results from the transform matrix in Class B which has the highest computational cost since it possesses only one M_{n-1} in the recursive transform matrix. The known Reed–Muller transform together with transform matrix from Class C yield the second best spectra with four nonzero spectral coefficients. However, it has the least computational costs among the four compared transform matrices. From (5), the LI Expansion based on matrix

$$M_B == [\vec{g_0}, \vec{g_1}, \vec{g_2}, \vec{g_3}, \vec{g_4}, \vec{g_5}, \vec{g_6}, \vec{g_7}]$$

where

$$g_0 = x_3x_2x_1, \quad g_1 = x_3, x_2\overline{x_1}, \quad g_2 = x_3\overline{x_2}x_1$$
$$g_3 = x_3\overline{x_2}\,\overline{x_1}, \quad g_4 = x_2x_1, \quad g_5 = x_2\overline{x_1}$$
$$g_6 = x_3x_1 \vee \overline{x_2}x_1 \quad \text{and} \quad g_7 = x_2x_2 \vee \overline{x_2}\,\overline{x_1}$$

is

$$f(\vec{x_3}) = g_6 \oplus g_7 = (x_3x_1 \vee \overline{x_2}x_1) \oplus (x_3x_2 \vee \overline{x_2}\,\overline{x_1}).$$

The resultant LI Expansion may be implemented easily by existing 20X8 PAL device [11]. The device, in general, has D flip-flops driven by XOR gates. The inputs of these XOR gates are fed by two sum-of-products arrays with two product terms each, allowing the resultant LI Expansion to be implemented directly. Some LI transform matrices may require more than two product terms, which is not appropriate for the 20X8 architecture. However, other types of devices are available to fit this demand, for example, the Cypress 330 [11] allows many product terms. The same expansion may also be easily implemented by Concurrent Logic CFA6006 fine grain Field

Programmable Gate Array [2]. The basic symmetrical cell for this FPGA has two logic functions: NAND and XOR which permits to implement arbitrary switching circuits represented by their LI polynomial expansions.

From the basic LI transform matrices, another category of LI transform matrices may be derived. In this new category, horizontal or vertical permutations are required before making the matrices recursive.

Property 6: Let M_n be any nonsingular square $2^n \times 2^n$ matrix with modulo-2 inverse M_n^{-1}. Then

$$[\mu_H[M_n]]^{-1} = \mu_V[M_n^{-1}] \tag{23}$$

$$[\mu_V[M_n]]^{-1} = \mu_H[M_n^{-1}]. \tag{24}$$

Definition 8: Let M_Y be any nonsingular, square, recursive matrix belonging to Class Y of LI Logic where $\mathrm{A} \cup \mathrm{B} \cup \mathrm{C} \cup \mathrm{D} = \mathrm{Y}$, i.e., $M_A \cup M_B \cup M_C \cup M_D = M_Y$. Then if $M_n = \mu_H[M_Y], M_n$ belongs to a class named $\mathrm{Y_H}$ where $\mathrm{Y_H} = \mathrm{A_H} \cup \mathrm{B_H} \cup \mathrm{C_H} \cup \mathrm{D_H}$, and $\mathrm{A_H}, \mathrm{B_H}, \mathrm{C_H}$, and $\mathrm{D_H}$, are classes in which its respective members are derived from the μ_H operation on the LI transform matrices in Class Y. If M_Y^{-1} defines the modulo-2 inverse of the matrix M_Y then from Property 6

$$M_n^{-1} = \mu_V[M_Y^{-1}]. \tag{25}$$

Similarly, if $M_n = \mu_V[M_Y]$, then

$$M_n^{-1} = \mu_H[M_Y^{-1}] \tag{26}$$

and M_n belongs to a class named $\mathrm{Y_V}$ where $\mathrm{Y_V} = \mathrm{A_V} \cup \mathrm{B_V} \cup \mathrm{C_V} \cup \mathrm{D_V}$, and $\mathrm{A_V}, \mathrm{B_V}, \mathrm{C_V}$, and $\mathrm{D_V}$, are classes in which its respective members are derived from the μ_V operation on the LI transform matrices in Class Y.

Definitions 5 and 6 transform all the basic classes of LI transforms having fast transforms into two categories, for one of which the horizontal permutation needs to be done in the forward transform and for the other one, the vertical permutation needs to be performed instead.

Example 4: In this example, the same 3-variable switching function from Example 3 is used. From Definition 4 and 5, the corresponding transform matrices which require vertical permutation at the Forward transforms are M_{AV}, M_{BV}, M_{CV} and M_{DV}, respectively. For example,

$$M_{DV} = \mu_V[M_D] = \mu_V \begin{bmatrix} 0&0&0&1&0&1&1&1 \\ 0&0&1&0&1&0&1&0 \\ 0&1&0&0&0&1&0&0 \\ 1&0&0&0&1&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \end{bmatrix}$$

$$= \begin{bmatrix} 0&0&0&1&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 0&0&0&1&0&1&1&1 \\ 0&0&1&0&1&0&1&0 \\ 0&1&0&0&0&1&0&0 \\ 1&0&0&0&1&0&0&0 \end{bmatrix}$$

and the corresponding modulo-2 inverse of M_{DV} is

$$M_{DV}^{-1} = \mu_H[M_D^{-1}] = \mu_H \begin{bmatrix} 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&1&0&0&0&1 \\ 0&0&1&0&0&0&1&0 \\ 0&1&0&1&0&1&0&1 \\ 1&1&1&1&1&1&1&1 \end{bmatrix}$$

$$= \begin{bmatrix} 0&0&0&1 & 0&0&0&0 \\ 0&0&1&0 & 0&0&0&0 \\ 0&1&0&0 & 0&0&0&0 \\ 1&0&0&0 & 0&0&0&0 \\ 0&0&0&1 & 0&0&0&1 \\ 0&0&1&0 & 0&0&1&0 \\ 0&1&0&1 & 0&1&0&1 \\ 1&1&1&1 & 1&1&1&1 \end{bmatrix}.$$

The corresponding LI Logic spectra are evaluated similarly. In summary

$$\vec{A_{AV}} = [1,0,0,1,0,0,1,1]^T$$

$$\vec{A_{BV}} = [0,0,0,0,0,1,1,1]^T$$

$$\vec{A_{CV}} = [0,1,1,1,0,0,1,1]^T$$

$$\vec{A_{DV}} = [0,0,1,1,0,1,0,1]^T.$$

From the spectra, it can be seen that the coefficient vector from M_{BV} transform matrix yields the optimal spectral coefficients. Comparing with Example 3, $\vec{A_B}$ still gives the best spectral coefficients with maximum number of zero coefficients. It should be noted that the corresponding vertical or horizontal permutation of each class has identical complexity owing to the same number of 1's and 0's in each permuted row.

So far, different LI transform matrices have been introduced that may be classified into 3 large classes. Class Y presents those LI transforms which require no permutations. Class $\mathrm{Y_H}$ and $\mathrm{Y_V}$ include such LI transforms that respectively require horizontal and vertical permutation at the forward transform matrices, and in addition, vertical and horizontal permutation at the inverse transform matrices. Another smaller classes of LI transforms exist, which have distinct properties from the above 3 large classes of LI transforms. In these classes, the transform matrices require either both horizontal or both vertical permutations at forward and inverse transform matrices. This is categorized as Class E. Class E divides into two subclasses, named $\mathrm{E_H}$ and $\mathrm{E_V}$. Fig. 5 shows the fast transforms for Class E.

Definition 9: Let M_n be one of the nonsingular square matrices in Class E. Let M_{EH} and M_{EV} be two LI transform matrices belonging to Classes $\mathrm{E_H}$ and $\mathrm{E_V}$, respectively. Then by definition, $M_{EH} = \mu_H[M_n], M_{EV} = \mu_V[M_n]$, and

$$M_{EH}^{-1} = \mu_H[\alpha_1(M_n)] \tag{27}$$

and

$$M_{EV}^{-1} = \mu_V[\alpha_1(M_n)]. \tag{28}$$

In this class of LI transform matrices, two fast forward algorithms exist which are shown in Fig. 5. The inverse fast

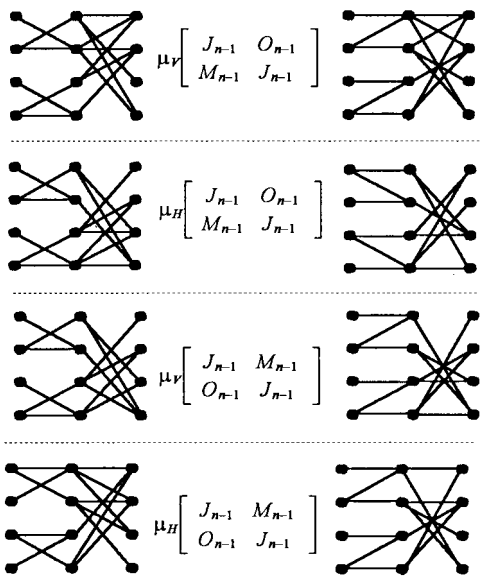

Fig. 5. Class E_H and E_V.

transforms are easily derived by *vertically flipping* the fast forward transforms. With this property, the same fast forward transform may be used to calculate inverse transform by reversing the order of the spectral coefficients.

Property 7: Let $\vec{F}$ define the column vector representing the truth vector of an n-variable switching function and $\vec{A}$ its spectral coefficients in natural binary ordering. Suppose M_E represents the LI transform matrices in Class E such that by (4) $\vec{A} = M_E^{-1} \cdot \vec{F}$. Defining the matrix operator $\underline{R}$ to a column vector as reversing the position of its elements, i.e.,

$$\underline{R}(\vec{F}) = [f_{2^n-1}, \cdots, F_{2^n-2}, F_o]^T \quad \text{and}$$
$$\underline{R}(\vec{A}) = [A_{2^n-1}, A_{2^n-2}, \cdots, A_0]^T$$

then

$$\underline{R}(\vec{A}) = M_E \cdot [\underline{R}(\vec{F})]. \tag{29}$$

This implies

$$\underline{R}(\vec{F}) = M_E^{-1} \cdot [\underline{R}(\vec{A})]. \tag{30}$$

Equation (30) shows that LI transform matrices in Class E have fast forward transform such that the same fast algorithm could be used to evaluate the respective inverse by simply reversing the input and output truth vectors. Equation (30) applies to LI transform matrices in Class B too. In general, if the forward and inverse transform matrices are related by α_0 and α_1 matrix operators with no β operator and the same μ_H or μ_V operations are used at the forward or inverse transform matrices, then the inverse fast transforms are easily derived by *vertically flipping* the respective fast forward transforms, and (29) and (30) are applicable. The operators in Class B or Class E LI transform matrices involve only either α_0 or α_1 matrix operator but not both. However, the existence of either identical diagonal submatrices or reverse-diagonal submatrices will satisfy the condition of forward and inverse transform matrices related by α_0 and α_1 matrix operators. This property is advantageous in terms of hardware architecture since the same implementations of butterfly diagram can be used for the calculation of both forward and inverse transforms.

Another class of transform matrices exists which has similar properties to LI transform matrices in Class A but requires identical permutations, either μ_H or μ_V at the forward or inverse transform matrices. This is categorized as Class F. The transform matrices in Class F contain no submatrices J_{n-1}.

Definition 10: Let M_n be any nonsingular square matrices from Class F, then M_{FH} and M_{FV} belong to two LI transform matrices in Classes $\mathrm{F_H}$ and $\mathrm{F_V}$ accordingly, where $M_{FH} = \mu_H[M_n], M_{FV} = \mu_V[M_n]$. Then,

$$M_{FH}^{-1} = \mu_H[M_n] \equiv M_{FH} \tag{31}$$

and

$$M_{FV}^{-1} = \mu_V[M_n] \equiv M_{FV}. \tag{32}$$

LI transform matrices in Class F have identical computational costs to that of Class E, since the recursive matrices possess only one submatrix M_{n-1}. Fig. 6 shows the fast forward and Inverse transforms for LI transforms in Class F. Generally, two different fast transforms exist for each LI transform matrix belonging to Classes E and F. This is due to the operations of μ_H or μ_V to the recursive equations. Finally, another class which requires only one permutation either at the forward transform matrices or at the inverse transform matrices but not at both, is introduced. In this class of LI transforms, the matrices contain all four different basic submatrices of $M_{n-1}, I_{n-1}, J_{n-1}$, and O_{n-1}. This is categorized as Class G. Fig. 7 lists the fast forward transforms for Class G. The mathematical relations between the Forward and Inverse Transform matrices are given in each subclass of G. Some of the LI transform matrices of G are related by the matrix operations α_0, α_1 or (α_0 and α_1). Among all the presented LI transform matrices, Class G is most impractical from a computational point of view. For some logical functions, this class can however have the simplest spectrum and the resulting hardware implementation and therefore is also discussed. The LI transforms from Class G possess more computationally expensive fast forward algorithms and their Inverses, though may be mathematically defined, have no relations with the respective fast forward butterflies. Moreover, only for $n = 2$, the fast transform requires six modulo-2 additions which is one of the highest computational costs among all other classes of LI transform matrices. This is the same as for all other LI transform matrices which posses only one submatrix M_{n-1}.

IV. Conclusion

The suitability of an LI transform in a given application depends not only on the choice of its basis functions but also on the existence of efficient ways of its calculation. When the concept of LI Logic was introduced, the author used the conventional means of calculating the LI transformations by matrix inversion and multiplication [15]–[17]. In this paper, recursive forward and inverse equations and butterfly structures are used to decompose matrix multiplications into simple pairwise operations so that on average the number of

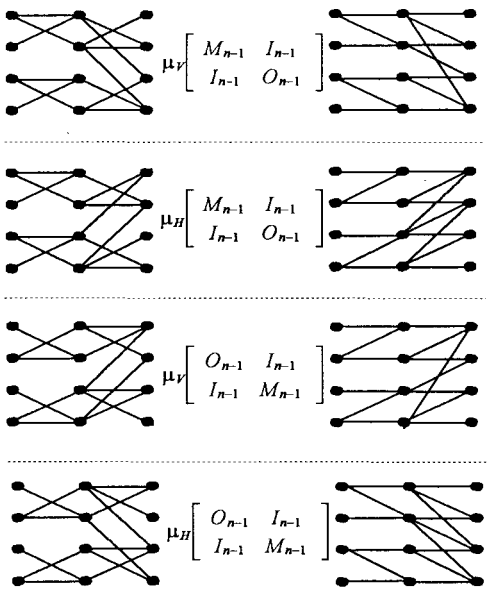

Fig. 6. Class F_H and F_V.

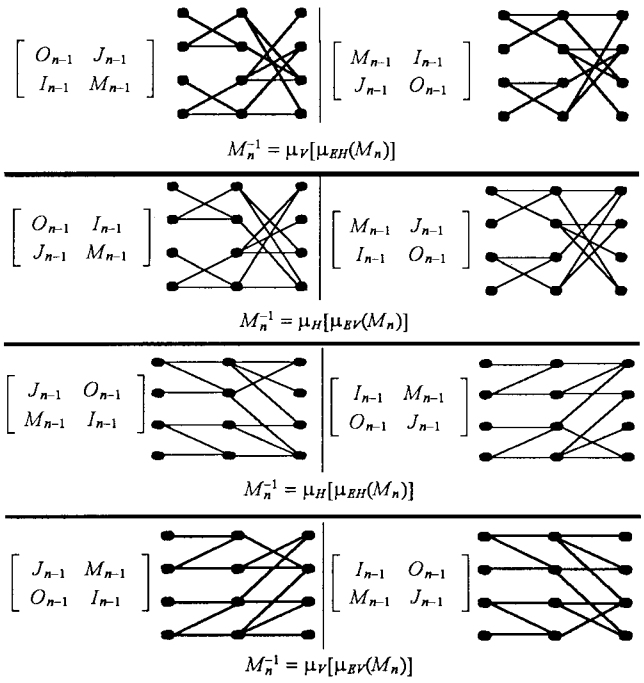

Fig. 7. Class G.

modulo-2 operations is minimized from N^2 to $N \log_2 N$ for an Nth-order transformation matrix.

Minimization of AND-OR and AND-XOR expressions has already been well established in logic synthesis [3], [23], [24]. However some new technologies allow PLA's with optional XOR elements in their outputs [11] what increased research interests in optimizing also different logical structures such as the AND-OR-XOR expression. The LI polynomial expansions may also be efficiently used in mapping to fine grain and cellular automata types of FPGA's (such as those from Concurrent Logic (now ATMEL), Crosspoint, Motorola, Plessey, Toshiba, Algotronix (now Xilinx), National Semiconductor, Pilkington) what was earlier discussed in [16] and [17] and shown by our example. The theory presented in this paper allows to find and calculate the polynomial expansions by using fast transforms. The design of the next generations of fine grain architectures should be influenced by the introduced efficient bases of LI logic transformations so that the resulting hardware implementation could be calculated efficiently by fast transforms and use minimal number of basic cells with a compact routing. As the result, it would have also the advantages of high speed and area minimization. Hence the practical applications of presented developments on the direction of future IC technology are enormous.

For a selected class of LI logic transformations, the corresponding polynomial expansion of an arbitrary logical function is canonical. The proper selection of the transform matrix can greatly reduce the final implementation of a given function not only in the form of available EPLD devices and fine grain FPGA's but also as custom made FPGA's. One of the future related research topics is to associate presented classification of LI Logic with some known classification of logical functions and use it for matching. The ability to select in advance those computationally effective fast LI transformations will result in simple implementation of some classes of logical functions. The good starting point in this new research would be to use spectral classification of logical functions based on Walsh functions and known relations between Walsh and Reed–Muller spectra [3]. This is to find the relations between LI transforms and expansions and spectral classification based either on standard or generalized Walsh functions [6], [7].

The same concept of LI Logic can be applied easily to logical functions with multiple-valued inputs (the transform matrix M is the same for two-valued and multiple-valued cases) [14], [20]–[22] and all the presented derivations are valid for such instances as well. A unified approach to the generation of butterfly structures for family of LI matrices can be also of interest for researchers developing efficient multiresolution digital signal processing systems using unconventional applications of butterfly LI decomposition techniques [1], [4], [5], [10], [18], [19], [25].

REFERENCES

[1] N. Ahmed and K. R. Rao, *Orthogonal Transforms for Digital Signal Processing*. Berlin, Germany: Springer-Verlag, 1975.
[2] Concurrent Logic Inc., *CLI 6000 Series Field Programmable Gate Arrays*, Advanced Information, Dec. 1993, Rev. 2.1.
[3] P. Davio, J. P. Deschamps, and A. Thayse, *Discrete and Switching Functions*. New York: McGraw-Hill, 1978.
[4] A. Drygajlo, "Butterfly orthogonal structure for fast transforms, filter banks and wavelets," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, vol. 5, San Francisco, CA, Mar. 1992, pp. 81-84.
[5] B. J. Falkowski and M. A. Perkowski, "A family of all essential radix-2 addition/subtraction multi-polarity Transforms: Algorithms and interpretations in Boolean domain," in *Proc. 23rd IEEE Int. Symp. Circuits Syst.*, New Orleans, LA, pp. 2913–2916, May 1990.
[6] B. J. Falkowski, "Properties and ways of calculation of multi-polarity generalized Walsh transforms," *IEEE Trans. Circuits Syst. II*, vol. 41, pp. 380–391, June 1994.
[7] ——, "Recursive relationships, fast transforms, generalizations and VLSI iterative architecture for Gray code ordered Walsh functions," *Proc. Inst. Elect. Eng. Comput. Digital Techniques*, vol. 142, no. 5, pp. 325–331, Sept 1995.
[8] B. J. Falkowski and S. Rahardja, "Fast transforms for orthogonal logic," in *Proc. 28th IEEE Int. Symp. Circuits Syst.*, Seattle, WA, May 1995, pp. 2164–2167.

[9] _____, "Family of fast transforms for GF(2) orthogonal logic," *IFIP WG10.5 Workshop on Applications of The Reed-Muller Expansion in Circuit Design*, Makuhari, Chiba, Japan, pp. 273–280, Aug. 1995.
[10] P. S. Moharir, *Pattern-Recognition Transforms*. New York: Wiley, 1992.
[11] D. Pellerin and M. Holley, *Practical Design Using Programmable Logic*. Englewood Cliffs, NJ: Prentice-Hall, 1991.
[12] M. A. Perkowski, P. Dysko, and B. J. Falkowski, "Two learning methods for a tree-search combinatorial optimizer," in *Proc. 9th IEEE Int. Conf. Comput. Commun.*, Scottsdale, AZ, Mar. 1990, pp. 606–613.
[13] M. A. Perkowski and P. Johnson, "Canonical multi-valued input Reed-Muller trees and forms," in *Proc. 3rd NASA Symp. VLSI Design*, Moscow, ID, 1991, pp. 11.3.1–11.3.13.
[14] M. A. Perkowski, "The generalized orthonomal expansion of functions with multiple-valued inputs and some of its applications," in *Proc. 22nd IEEE Int. Symp. Multiple-Valued Logic*, Sendai, Japan. May 1992, pp. 442–450.
[15] _____, "A fundamental theorem for EXOR circuits," in *Proc. IFIP WG 10.5 Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, U. Kebschull, E. Schubert, W. Rosenstiel, Eds., Hamburg, Germany, Sept. 1993, pp. 52–60.
[16] M. A. Perkowski, A. Sarabi, and F. R. Beyl, "XOR canonical forms of switching functions," in *Proc. IFIP WG 10.5 Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, U. Kebschull, E. Schubert, W. Rosenstiel, Eds., Hamburg, Germany, Sept. 1993, pp. 27–32.
[17] _____, "Fundamental theorems and families of forms for binary and multiple-valued linearly independent logic," *IFIG WG 10.5 Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, Makuhari, Chiba, Japan, Aug. 1995, pp. 288–299.
[18] K. R. Rao, M. A. Narasimhan, and K. Revuluri, "Image data processing by Hadamard-Haar transform," *IEEE Trans. Comput.*, vol. C-24, pp. 888–896, Sept. 1975.
[19] _____, "A family of discrete Haar transforms," *Comput. Elect. Eng.*, vol. 2, pp. 367–388, 1975.
[20] T. Sasao, "A transformation of multiple-valued input two-valued output functions and its application to simplification of exclusive-or sum-of-products expressions," in *Proc. 21st IEEE Int. Symp. Multiple-Valued Logic*, Victoria, Canada, May 1991, pp. 270–279.
[21] _____, "Optimization of multiple-valued AND-EXOR expressions using multiple-place decision diagrams," in *Proc. 22nd IEEE Int. Symp. Multiple-Valued Logic*, Sendai, Japan, May 1992, pp. 451–458.
[22] _____, "Optimization of Pseudo-Kronecker expressions using multiple-place decision diagrams," *IEICE Trans. Inform. Syst.*, vol. E76-D, no. 5, pp. 562–570, May 1993.
[23] _____, *Logic Synthesis and Optimization*. Boston, MA: Kluwer Academic, 1993.
[24] W. G. Schneeweiss, *Boolean function with Engineering Applications and Computer Programs*. Berlin, Germany: Springer-Verlag, 1989.
[25] L. P. Yaroslavsky, *Digital Picture Processing-An Introduction*. Berlin, Germany: Springer-Verlag, 1985.

Bogdan J. Falkowski (S'88–M'90–SM'94) received the M.S.E.E. degree from the Technical University of Warsaw, Poland, and the Ph.D. degree from Portland State University, Portland, OR.

His industrial experience includes research and development positions at several companies from 1978 to 1986. He joined the Electrical Engineering Department of Portland State University in 1986. In 1992, he became a Senior Lecturer with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His research interests include VLSI systems and design, synthesis and optimization of switching circuits, multiple-valued systems, testing, design of algorithms, design automation, digital signal and image processing. During the last six years, he has published more than 60 articles in the design of digital circuits with the use of spectral methods.

Dr. Falkowski is a member of Eta Kappa Nu and Pi Beta Upsilon.

Susanto Rahardja (M'97) received the B.Eng. degree in electrical engineering in 1991 from the National University of Singapore (NUS), the M.Eng. degree in digital communication and microwave circuits, and the Ph.D. degree in the area of logic synthesis and signal processing from Nanyang Technological University (NTU), Singapore, in 1993 and 1997, respectively.

He is presently working as a Research Fellow with the Centre for Signal Processing, NTU. He has more than 25 articles in international journals and conferences. His research interests include binary logic synthesis, digital communication systems, digital signal processing, microwave circuits, and multiple-valued logic synthesis.