TWO CLASSES OF FIXED POLARITY LINEARLY INDEPENDENT ARITHMETIC TRANSFORMS FOR QUATERNARY FUNCTIONS

Cicilia C. Lozano and Bogdan J. Falkowski

School of Electrical and Electronic Engineering
Nanyang Technological University
Block S1, 50 Nanyang Avenue, Singapore 639798
email: cicilia@ntu.edu.sg; efalkowski@ntu.edu.sg

Tadeusz Łuba

Institute of Telecommunications
Warsaw University of Technology
Nowowiejska 15/19, 00-665, Warsaw, Poland
luba@tele.pw.edu.pl

ABSTRACT

Two classes of fixed polarity linearly independent arithmetic transforms (FPQLIA) for quaternary functions are introduced in this paper. These transforms are Kronecker-based and therefore can be calculated efficiently by fast transforms. Their basic definitions and fast flow graphs are shown. Relations between the different FPQLIA transforms are also presented and an algorithm for the optimization of FPQLIA is described which utilizes the given relation to reduce the computational cost. Experimental results for the transforms in terms of the number of nonzero spectral coefficients in the optimal FPQLIA transforms have also been given for several quaternary test functions and compared to the corresponding numbers for the optimal fixed polarity quaternary arithmetic (FPQA) transforms. The results show that for the set of quaternary test functions the numbers for FPQLIA transforms are on average 32% smaller than the ones for the FPQA transforms.

1. INTRODUCTION

Spectral expansions are alternative representations of logic functions/signals in which the information are redistributed and presented differently in terms of spectral coefficients [1]–[3]. The use of spectral representations often allows certain operations or analysis to be performed more efficiently on the data. In general each spectral expansion has an associated pair of forward and inverse transforms which can be used to transform the truth vector representation of a particular function to the equivalent spectral expansion and vice versa. Among the existing spectral expansions, several pairs of spectral transforms can be found where they have the same forward transforms but different inverse transforms due to the different operators used to define the transforms. One example of such pair is the well-known binary Reed-Muller (RM) transform which uses GF(2) operators and Arithmetic transform [1]–[3] which uses standard arithmetic operators. The reason for this is because the use of different operators may lead to advantages for different applications. In particular, transforms employing standard/decimal arithmetic operations have been found to be advantageous for analysis of circuits, verification, testability, as well as reliability analysis. Such arithmetic transforms are also useful for efficient representation and parallel calculation of multi-output functions [1]. Linearly independent (LI) and linearly independent arithmetic (LIA) transforms [4], which can be thought of as the broadest generalization of RM and Arithmetic transforms, respectively, are another example of transform pair whose definitions are identical except for the employed arithmetic operators, and therefore the operational domains. LI and LIA transforms can be thought of as the RM and Arithmetic transforms in which the basis functions are not restricted to be conjunctive of input literals but are allowed to be replaced by any set of linearly independent binary functions. Since RM and Arithmetic transforms are merely special cases of LI and LIA transforms, the performance of LI and LIA transforms are never worse than the RM and Arithmetic transforms, respectively. It has been shown in [4] that for detection of stuck-at and bridging faults some fast LIA transforms outperform the Arithmetic transform in terms of the number of spectral coefficients that need to be tested.

The concept of LI and LIA transforms have been extended for the case of quaternary functions, called quaternary linearly independent (QLI) and quaternary linearly independent arithmetic (QLIA) transforms. Some classes of fast QLI and QLIA transforms have been defined in [5] and [6], where the transforms in [5] are derived from the recursive definitions of four fast binary LI transforms. Similar to the binary case, the broad definition of the QLI and QLIA transforms also cover the various quaternary extensions of the RM and Arithmetic expansions, respectively, such as the fixed polarity RM over GF(4) [7] and quaternary fixed polarity Arithmetic (QFPA) transforms [1], [8]. In this paper, two new classes of fast FPQLIA transforms are introduced where the basic forward transforms for the first 1-variable FPQLIA transform in the two classes are the same as the forward transforms of the first transforms of Class 3 and 2 QLI transforms in [5]. Basic definitions, fast flow graph, and several relations for the FPQLIA transforms are given. An algorithm to find the optimal FPQLIA expansion for a quaternary function is also presented. Finally, experimental results of the FPQLIA transforms are given for several quaternary test functions and compared with the FPQA transforms [8]. The results show that the introduced FPQLIA transforms are more advantageous as they can give more compact representation in terms of smaller number of nonzero spectral coefficients.

2. BASIC DEFINITIONS FOR THE NEW FPQLIA TRANSFORMS

The new transforms that are introduced in this paper belong to the broad class of QLIA transforms. In the following, Definitions 1–3 give the general basic definitions for any QLIA transform whereas Definitions 4–6 describe the notations used in this paper.

Definition 1. Let T_n be a $4^n \times 4^n$ matrix with rows corresponding to minterms and columns corresponding to some n-variable quaternary switching functions. If the set of columns is linearly independent with respect to standard arithmetic algebra, then T_n has only one arithmetic inverse and is said to be a QLIA matrix.

Definition 2. Let T_n be a QLIA matrix as given in Definition 1. Also, let $\vec{F} = [F_0, F_1, \ldots, F_{4^n-1}]^T$ be the truth vector of an n-variable quaternary switching function $f(\overrightarrow{X_n}) = f(X_n, X_{n-1}, \ldots, X_1)$ where T denotes transpose operator. Then, the spectrum of T_n for $f(\overrightarrow{X_n})$, denoted by $\vec{A} = [A_0, A_1, \ldots, A_{4^n-1}]^T$, can be obtained by

$$\vec{A} = T_n^{-1}\vec{F} \quad (1)$$

where T_n^{-1} is the arithmetic inverse of T_n.

Conversely,

$$\vec{F} = T_n\vec{A}. \quad (2)$$

Definition 3. Given a particular QLIA transform T_n, any n-variable quaternary switching function $f(\overrightarrow{X_n})$ can be represented by QLIA polynomial expansion as follows:

$$f(\overrightarrow{X_n}) = \sum_{j=0}^{4^n-1} A_j g_j,$$

where g_j $(0 \le j \le 4^n - 1)$ denotes the n-variable quaternary switching function whose truth vector is equal to the j-th column of T_n and A_j denotes the j-th QLIA spectral coefficient for $f(\overrightarrow{X_n})$ based on T_n. All the additions and multiplications inside the QLIA expansion are performed in standard arithmetic algebra.

Definition 4. Let a matrix T_n be recursively partitioned into sixteen submatrices T_{n-1} of size $4^{n-1} \times 4^{n-1}$ each as follows:

$$T_n = \begin{bmatrix} T_{n-1}^{(1,1)} & \cdots & T_{n-1}^{(1,4)} \\ \cdots & \cdots & \cdots \\ T_{n-1}^{(4,1)} & \cdots & T_{n-1}^{(4,4)} \end{bmatrix}.$$

Then, the operation μ_{EH} on T_n is defined as recursively grouping the submatrices vertically and interchanging them horizontally:

$$\mu_{EH}(T_n) = \begin{bmatrix} T_{n-1}^{(1,4)} & \cdots & T_{n-1}^{(1,1)} \\ \cdots & \cdots & \cdots \\ T_{n-1}^{(4,4)} & \cdots & T_{n-1}^{(4,1)} \end{bmatrix}.$$

Similarly, the operation μ_{EV} on T_n is defined as recursively grouping the submatrices horizontally and interchanging them vertically:

$$\mu_{EV}(T_n) = \begin{bmatrix} T_{n-1}^{(4,1)} & \cdots & T_{n-1}^{(4,4)} \\ \cdots & \cdots & \cdots \\ T_{n-1}^{(1,1)} & \cdots & T_{n-1}^{(1,4)} \end{bmatrix}.$$

Definition 5. A quaternary variable X_i may assume the value of 0, 1, 2, or 3. In polynomial expansions, the literals of X_i are denoted by $X_i^{S_i}$ where $S_i \subseteq \{0, 1, 2, 3\}$ is said to be the true set of the variable X_i and

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

Definition 6. In binary function representation, applying the complement operator to a variable changes the value of the variable from 0 to 1 and vice versa. In multiple-valued logic, similar mapping can be performed through the application of cycle and negation/complement unary operators [1]. Let X_i be a quaternary variable, then the cycle and complement operations on X_i are denoted by X_i^k ($k \in \{0,1,2,3\}$) and $\overline{X}_i$, respectively, where $X_i^k = (X_i + k) \bmod 4$ and $\overline{X}_i = 3 - X_i$.

Two classes of FPQLIA transforms are introduced in this paper. They are classified into Class 1 and Class 2 and their forward transforms are denoted by $T_{1,n}^{\omega}$ and $T_{2,n}^{\omega}$, respectively where $T_{\alpha,n}^{\omega}$ ($\alpha \in \{1,2\}$) is a $4^n \times 4^n$ matrix which is obtained from Kronecker product on n basic matrices $T_{\alpha,1}^{\omega}$ as follows:

$$T_{\alpha,n}^{\omega} = \otimes\prod_{i=n}^{1} T_{\alpha,1}^{\omega_i} = T_{\alpha,1}^{\omega_n} \otimes T_{\alpha,1}^{\omega_{n-1}} \otimes \cdots \otimes T_{\alpha,1}^{\omega_1}. \quad (3)$$

Their inverse transforms are denoted by $(T_{\alpha,n}^{\omega})^{-1}$ and are similarly defined as

$$(T_{\alpha,n}^{\omega})^{-1} = \otimes\prod_{i=n}^{1} (T_{\alpha,1}^{\omega_i})^{-1} = (T_{\alpha,1}^{\omega_n})^{-1} \otimes (T_{\alpha,1}^{\omega_{n-1}})^{-1} \otimes \cdots \otimes (T_{\alpha,1}^{\omega_1})^{-1}. \quad (4)$$

In both (3) and (4), the symbol ω denotes the polarity number of $T_{\alpha,n}^{\omega}$ where ω is the decimal equivalent of the n-digit quaternary number $\omega_n, \omega_{n-1}, \ldots, \omega_1$, i.e., $<\omega>_{10} = <\omega_n, \omega_{n-1}, \ldots, \omega_1>_4$. In addition, $T_{\alpha,1}^{\omega}$ and $(T_{\alpha,1}^{\omega})^{-1}$ are the basic forward and inverse transforms for the FPQLIA of the respective class where they are of size 4×4.

All the $T_{\alpha,1}^{\omega}$ and $(T_{\alpha,1}^{\omega})^{-1}$ matrices are listed in Table 1 together with the fast flow graphs for $(T_{\alpha,1}^{\omega})^{-1}$. Since there are four basic transforms in each class, for an n-variable quaternary function there are 4^n possible FPQLIA expansions of each class where each expansion is uniquely identified by its polarity number ω ($0 \le \omega \le 4^n - 1$). Different FPQLIA expansions may have different number of nonzero elements in their spectral coefficient vectors. The one with the minimum number of nonzero elements is called the optimal FPQLIA expansion and its polarity number is said to be the optimal polarity number, denoted by ω_{opt}.

Among the transforms given in Table 1, it should be mentioned that the transforms $T_{1,1}^0$ and $T_{2,1}^0$ correspond to the forward matrix of the first transforms in Class 3 and 2 fast QLI transforms given in [5], respectively. In addition, they also coincide with the forward transforms of the 2-variable binary fixed polarity arithmetic (FPA) transforms [1]–[3] in polarity zero and three. Due to this, the transform matrices for an n-variable Class 1 (2) FPQLIA transforms in polarity zero are identical to the transform matrices of a $2n$-variable binary FPA transforms in polarity zero ($2^{2n} - 1$).

The FPQLIA expansion representations for a quaternary function $f(\overrightarrow{X_n})$ can be generated using the general equation for QLIA expansion given in Definition 3. Due to the fact that the FPQLIA transforms are built from Kronecker product, their basis functions g_j $(0 \le j \le 4^n - 1)$ have regular structure and can be obtained by:

$$g_{\alpha,n}^{\omega} = [g_0 \;\; g_1 \;\; \cdots \;\; g_{4^n-1}] = \overset{1}{\underset{i=n}{\otimes}} g_{\alpha,1}^{\omega_i}$$

where $g_{\alpha,1}^{\omega_i}$ is the row matrix whose columns are the basis functions of the FPQLIA transform $T_{\alpha,1}^{\omega_i}$ for a single quaternary variable X_i (recall that $<\omega>_{10} = <\omega_n, \omega_{n-1}, \ldots, \omega_1>_4$). Let $g_{\alpha,1}^{\omega_i} = [g_0 \;\; g_1 \;\; g_2 \;\; g_3]$. Then from Table 1 it can be seen that for $\alpha = 1$ and $\omega_i = 0$ $g_{\alpha,1}^{\omega_i} = [1 \;\; X_i^{\{1,3\}} \;\; X_i^{\{2,3\}} \;\; X_i^{\{3\}}]$ whereas for $\alpha = 1$ and ω_i = 1, 2, and 3 the $g_{\alpha,1}^{\omega_i}$ are the same as $g_{\alpha,1}^{0}$ except that cycle operations should be applied on X_i so that X_i is replaced by X_i^1, X_i^2, and X_i^3, respectively. Similarly, for $\alpha = 2$ their corresponding basis function row matrices $g_{\alpha,1}^{\omega_i}$ can also be obtained from the basis function row matrix $g_{1,1}^0$ by replacing X_i with $\overline{X}_i$, $(\overline{X}_i)^1$,

$(\overline{X}_i)^2$, and $(\overline{X}_i)^3$ for ω_i = 0, 1, 2, and 3, respectively. As a result, all FPQLIA expansions have the same product terms except that appropriate cycle and complement operations need to be applied to the variables based on the polarity number of the expansion.

Owing to the fact that all the forward transforms given in Table 1 contain only 0 and 1 elements, the basis functions for the FPQLIA transforms can also be represented as two-variable binary functions. Thus, the given FPQLIA transforms can also be used to generate binary polynomial expansions for a binary function. This can be done by pairing up every two binary input variables in the binary function and encoding them to form quaternary input variables, obtaining the FPQLIA expansion for the resulting quaternary function, and finally replacing the quaternary basis functions in the FPQLIA expansion with the corresponding binary functions to get the binary expansion in terms of the binary input variables. For example, if the encoding 00→0, 01→1, 10→2, and 11→3 is used to map the binary input variables $x_{2i}x_{2i-1}$ values to the quaternary variable X_i values, then the basis function row matrices $g_{1,1}^{\omega_i}$ can be alternatively represented as $[1\ x_{2i-i}\ x_{2i}\ x_{2i}x_{2i-1}]$, $[1\ \bar{x}_{2i-1}\ x_{2i}\oplus x_{2i-1}\ x_{2i}\bar{x}_{2i-1}]$, $[1\ x_{2i-1}\ \bar{x}_{2i}\ \bar{x}_{2i}x_{2i-1}]$, and $[1\ \bar{x}_{2i-1}\ \overline{x_{2i}\oplus x_{2i-1}}\ \bar{x}_{2i}\bar{x}_{2i-1}]$ for ω_i = 0, 1, 2, and 3, respectively. Furthermore, by replacing the binary variables in $g_{1,1}^{\omega_i}$ with their complements we can get the corresponding binary basis function row matrices $g_{2,1}^{\omega_i}$.

3. RELATIONS, ALGORITHM, AND COMPUTATIONAL COSTS

Property 1. All the FPQLIA transform matrices in the same class are related by row permutations to each other. Let ρ_0, ρ_1, ρ_2, and ρ_3 be four permutation matrices of size 4×4 given by $\rho_0=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$, $\rho_1=\begin{bmatrix}0&1&0&0\\0&0&1&0\\0&0&0&1\\1&0&0&0\end{bmatrix}$, $\rho_2=\begin{bmatrix}0&0&1&0\\0&0&0&1\\1&0&0&0\\0&1&0&0\end{bmatrix}$, and $\rho_3=\begin{bmatrix}0&0&0&1\\1&0&0&0\\0&1&0&0\\0&0&1&0\end{bmatrix}$. Then, for $\alpha=1$ and any two polarities ω_a and ω_b $(<\omega_a>_{10}=<\omega_{an},\omega_{an-1},\ldots,\omega_{a1}>_4,\ <\omega_b>_{10}=<\omega_{bn},\omega_{bn-1},\ldots,\omega_{b1}>_4)$

$$T_{\alpha,n}^{\omega_a}=\left(\overset{1}{\underset{i=n}{\otimes}}\left(\rho_{\omega_{ai}}\cdot\left(\rho_{\omega_{bi}}\right)^{-1}\right)\right)\cdot T_{\alpha,n}^{\omega_b}. \quad (5)$$

For $\alpha=2$ $\rho_{\omega_{ai}}$ and $\rho_{\omega_{bi}}$ in (5) need to be replaced by $\rho_{(-\omega_{ai}\bmod 4)}$ and $\rho_{(-\omega_{bi}\bmod 4)}$, respectively.

Property 2. For the same polarity number ω, the Class 1 and Class 2 FPQLIA transforms are related as follows:

$$T_{1,n}^{\omega}=\mu_{EV}\left(T_{2,n}^{\omega}\right) \quad \text{and} \quad \left(T_{1,n}^{\omega}\right)^{-1}=\mu_{EH}\left(\left(T_{2,n}^{\omega}\right)^{-1}\right).$$

Property 3. Let $\vec{A}^{\omega_a}$ and $\vec{A}^{\omega_b}$ be the spectral coefficient vectors of two FPQLIA transforms of the same class but different polarity numbers ω_a and ω_b. Then, the following relation can be derived from (1)–(4):

$$\vec{A}^{\omega_a}=\left(\overset{1}{\underset{i=n}{\otimes}}Z^{(\omega_{ai}-\omega_{bi})\bmod 4}\right)\cdot\vec{A}^{\omega_b} \quad (6)$$

where $Z^0=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$, $Z^1=\begin{bmatrix}1&1&1&1\\0&-1&-1&-1\\0&0&-1&-1\\0&0&2&1\end{bmatrix}$, $Z^2=\begin{bmatrix}1&0&1&0\\0&1&0&1\\0&0&-1&0\\0&0&0&-1\end{bmatrix}$, and $Z^3=\begin{bmatrix}1&1&0&0\\0&-1&1&0\\0&0&1&1\\0&0&-2&-1\end{bmatrix}$.

In spectral techniques, one important problem associated with each class of spectral transforms is on how to obtain the optimal expansion based on all the transforms in the class. For fixed polarity transforms, this optimization problem is translated into the problem of finding the optimal polarity ω_{opt}. The exhaustive way to do it is to generate the spectral coefficient vectors of all possible polarities and identify the ω_{opt}. Although it is possible to calculate each spectral coefficient vector directly by fast transform [1]–[3], [7] of (1), it has been shown that it is more computationally efficient to calculate them using algorithms that utilize the existing relations between the transforms of different polarities [7]. One example of such algorithm for FPQLIA transforms is described below.

In the algorithm all the 4^n polarity numbers are arranged in a list such that any two consecutive polarity numbers in the list have only one different digit in their *n*-digit quaternary number representations. Such a list is called extended dual polarity route where the first polarity number in the list is called the initial polarity number. Given an extended dual polarity route, the algorithm first calculates the spectral coefficient vector for the initial polarity number from truth vector by fast transform of (1) and then continues to calculate the spectrum for the other polarity numbers serially in the order specified by the route. For each non-initial polarity number, the algorithm calculates its spectrum from the spectrum of the previous polarity number in the list using (6). Note that as a result of the requirement that any two consecutive polarity numbers in the route cannot have more than one different digit, the values of $(\omega_{ai}-\omega_{bi})\bmod 4$ in (6) are reduced to 0 for all except one particular *i*. Since Z^0 is simply an identity matrix, Z^0 does not contribute any additional computational cost to calculation of (6) which leads to the smaller computational cost for the algorithm.

From the description of the algorithm above, it is clear that the computational cost of the algorithm is mostly due to the cost of performing (6), which in turn is determined by the value of the non-zero digit of $(\omega_{ai}-\omega_{bi})\bmod 4$. Hence, the computational cost of the algorithm is largely dependent on the used extended dual polarity route. Because of that, the chosen route for the algorithm should be the one that leads to the minimum computational cost. Fig. 1 shows the pseudocode of the algorithm to calculate all the FPQLIA spectral coefficients in a particular class where the route that results in the minimum computational cost is employed.

In order to show that the given algorithm indeed reduces the computational cost of obtaining the optimal FPQLIA expansion compared to directly calculating all individual spectral coefficient vectors by fast transform of (1), in the following their computational costs are derived.

Let us first derive the computational cost for calculating all spectral coefficient vector by fast transform of (1). In fast transform method $\vec{A}$ is calculated by

$$\vec{A}=\prod_{i=n}^{1}\left(I_{n-i}\otimes\left(T_{\alpha,1}^{\omega_i}\right)^{-1}\otimes I_{i-1}\right)\cdot\vec{F}$$

where I_j denotes the identity matrix of size $4^j\times4^j$ and each value of *i* corresponds to one stage in the fast flow graph of the inverse transform. From the fast inverse flow graphs given in Table 1, it can be seen that the computational cost of (1) for calculating any $\left(T_{\alpha,1}^{\omega}\right)^{-1}$ is four subtractions. Hence, for *n* variables the computational cost contributed by each fast flow graph stage is $4\cdot4^{n-1}=4^n$ subtractions. As there are *n* stages in the fast flow graph, it follows

that the total computational cost of (1) by fast transform method is $n \cdot 4^n$ subtractions for one polarity number and $n \cdot 4^{2n}$ subtractions for all 4^n polarity numbers.

Next, let us derive the computational cost for the algorithm in Fig. 1. The cost of calculating the FPQLIA spectra for all 4^n polarity numbers in a particular class using the algorithm can be divided into the cost of calculating the spectrum of the initial polarity number by (1) and the cost of calculating the rest of the spectra by (6). The former has been derived above to be $n \cdot 4^n$ subtractions whereas the latter is dependant upon how many times the $(\omega_{ai} - \omega_{bi}) \bmod 4$ in (6) evaluates to 1, 2, or 3 throughout the algorithm.

It can be observed that inside the algorithm $(\omega_{ai} - \omega_{bi}) \bmod 4$ never evaluates to 1. Instead, its value is always 0, 2, or 3 where exactly one nonzero value of $(\omega_{ai} - \omega_{bi}) \bmod 4$ occurs for each pair of consecutive transforms. In each recursion level the calculation of every cycle of one *Dir*[*loop_var*] (e.g., 0→2→1→3 for *Dir*[*loop_var*] ='a') involves two Z^2 and one Z^3 where throughout the algorithm there are a total of 4^i cycles of *Dir*[*loop_var*] for *loop_var* = *i* ($0 \le i \le n-1$). Since it can be easily derived from Z^2 and Z^3 matrices that for an *n*-variable the computational cost of performing (6) is $2 \cdot 4^{n-1}$ additions if the nonzero value of $(\omega_{ai} - \omega_{bi}) \bmod 4 = 2$ and $4 \cdot 4^{n-1} = 4^n$ additions/subtractions and 4^{n-1} multiplications if the nonzero value of $(\omega_{ai} - \omega_{bi}) \bmod 4 = 3$, the total cost for calculating the spectra of the nonzero polarity numbers (the non-initial polarity numbers) is given by

$$Cost_{alg(nz)} = \left(\sum_{i=0}^{n-1} 4^i\right)\begin{pmatrix}\left(2 \cdot 2 \cdot 4^{n-1} + 4 \cdot 4^{n-1}\right)\text{additions/subtractions} \\ +\,4^{n-1}\text{ multiplications}\end{pmatrix}$$
$$= \frac{1}{3}\left(4^n - 1\right)\begin{pmatrix}8 \cdot 4^{n-1}\text{ additions/subtractions} \\ +\,4^{n-1}\text{ multiplications}\end{pmatrix}.$$

Adding the cost of calculating the polarity zero spectrum, the total calculation cost for the algorithm is

$$Cost_{alg} = \frac{1}{12}\begin{pmatrix}\left(8 \cdot 16^n + (3n-2) \cdot 4^{n+1}\right)\text{additions/subtractions} \\ +\,(16^n - 4^n)\text{ multiplications}\end{pmatrix}$$

which is much smaller than the cost of calculating all 4^n spectra by (1). Note that the number of multiplications in $Cost_{alg}$ arises due to the multiplication by −2 inside Z^3. However, if the multiplications are performed by simple shifting and sign bit change operations, the multiplication number is reduced to zero.

4. EXPERIMENTAL RESULTS

The computation of the spectral coefficient vectors of all possible Class 1 and 2 FPQLIA transforms has been implemented in MATLAB and run for a set of quaternary test functions as shown in Table 2. The quaternary test functions xor5, squar5, con1, ex5, inc, misex1, rd84, z5xp1, 9sym, apex4, clip, ex1010, and z9sym were obtained by modifying the MCNC IWLS 93 binary benchmark files of the same names to represent quaternary functions. The translation from binary to quaternary functions has been done by encoding every two input (output) bits in binary files to an input (output) symbol in the quaternary files. If the number of input and/or output variables is odd, then a zero bit is added behind the binary cubes to make it even. For both inputs and outputs the binary values pairs −−, 00, 01, 10, and 11 are encoded to −, 0, 1, 2, and 3, respectively. With these conversions, the binary benchmark files have become an array of quaternary cubes. The other test functions prod*n*, sum*n*, sqsum*n*, mprod*n*, msum*n*, max*n*, min*n* and avg*n* were written to represent some simple single output *n*-variable quaternary functions. The output of prod*n,* sum*n,* and sqsum*n* is the GF(4) product, sum, and sum of squares of the inputs, respectively. The output of mprod*n* and msum*n* is the modulo 4 product and sum of the inputs, whereas the output of max*n*, min*n* and avg*n* is the maximum, minimum, and integer part of the arithmetic average of the inputs, respectively. In addition to them, test functions count*ni* have also been written where the values of *i* varies from 0 to 3. They represent *n*-variable quaternary functions where the outputs are the quaternary number representations of the number of occurrence of *i* in the inputs. Since for *n* = 3, 4, and 5 the optimal numbers of nonzero spectral coefficients for count*ni* are the same regardless of *i*, in Table 2 these functions are simply represented as count3*i,* count4*i,* and count5*i*.

For comparison purpose, the numbers of nonzero spectral coefficients for optimal QFPA transform have also been given in the rightmost column of Table 2. It can be seen that for all functions in the table either $T_{1,n}^{\omega_{opt}}$, $T_{2,n}^{\omega_{opt}}$, or in most cases both, give better number of nonzero spectral coefficients compared to the optimal QFPA. Also, the number of nonzero spectral coefficients of $T_{1,n}^{\omega_{opt}}$ and $T_{2,n}^{\omega_{opt}}$ are similar for most of the functions but may be quite different for some cases, such as min4 and max4.

Table 2 – Minimum number of nonzero spectral coefficients

Input filename	Number of nonzero spectral coefficients		
	$T_{1,n}^{\omega_{opt}}$	$T_{2,n}^{\omega_{opt}}$	Optimal QFPA
xor5	7	7	18
squar5	21	24	27
con1	14	14	38
ex5	123	116	187
inc	55	49	71
misex1	23	20	40
rd84	55	65	202
z5xp1	42	49	60
9sym	78	78	370
apex4	473	456	481
clip	260	296	383
ex1010	1010	1011	1016
z9sym	78	78	370
prod3	24	24	27
prod4	73	69	73
prod5	218	218	243
sum3	14	15	23
sum4	30	30	76
sum5	62	63	237
sqsum3	39	39	56
sqsum4	111	111	206
sqsum5	351	351	764
mprod3	8	8	10
mprod4	16	16	20
mprod5	32	34	37
msum3	38	39	48
msum4	185	186	219
msum5	704	704	937
avg3	27	20	49
avg4	177	179	215
avg5	704	704	937
min3	21	27	26
max3	28	22	28
min4	47	73	81
max4	73	48	82
count3*i*	3	3	6
count4*i*	5	5	24
count5*i*	11	11	122
Total	5240	5262	7809
Average	137.89	138.47	205.5

```
Void fpqlia(int truth vector[ ])
{ int W[n] = <0, 0, …., 0>, p[ ];
  char Dir[n] = 'aa…a';
  Calculate A⃗^0 from truth vector by (1) and store the result in p[ ];
  For(j = 0 to 4^n – 2)
  { cont = true;
    loop_var = n;
    while (cont==true)
    { loop_var = loop_var —;
      cont = false;
      if(Dir[loop_var]=='a')
      { switch (W[loop_var])
        { case 0: W[loop_var] = 2; break;
          case 1: W[loop_var] = 3; break;
          case 2: W[loop_var] = 1; break;
          case 3: { cont = true; Dir[loop_var] = 'b';}} }
      else if(Dir[loop_var]=='b')
      { switch (W[loop_var])
        { case 0: W[loop_var] = 2; break;
          case 1: W[loop_var] = 0; break;
          case 2: { cont = true; Dir[loop_var] = 'c';}
          case 3: W[loop_var] = 1; break;} }
      else if(Dir[loop_var]=='c')
      { switch (W[loop_var])
        { case 0: W[loop_var] = 3; break;
          case 1: { cont = true; Dir[loop_var] = 'd';}
          case 2: W[loop_var] = 0; break;
          case 3: W[loop_var] = 1; break;} }
      else
      { switch (W[loop_var])
        { case 0: { cont=true; Dir[loop_var] = 'a';}
          case 1: W[loop_var] = 3; break;
          case 2: W[loop_var] = 0; break;
          case 3: W[loop_var] = 2; break;} }}
    Calculate A⃗^W from p[ ] by (6); p[ ] = A⃗^W ;}}
```

Figure 1 – Algorithm to calculate FPQLIA spectral coefficients.

5. CONCLUSIONS

Two new classes of FPLIA transforms have been presented. Their forward and inverse matrices as well as the relations between the different matrices have been given. An algorithm for generating all FPQLIA spectra and obtaining the optimal expansion has also been described based on the existing relations between the FPQLIA spectra. Its computational cost has been derived and shown to be more efficient than calculating each FPQLIA spectrum separately. Experimental results for the FPQLIA transforms have also been presented in the paper where the result shows that they are able to provide the quaternary test functions with more compact representations compared to FPQA transforms.

For $n > 2$ the introduced FPQLIA transforms are built from the Kronecker product of its basic transform matrices. As a result their basis functions have regular structure and can be directly generated from the basis functions of their basic transform matrices. It has been shown that the FPQLIA expansion can be directly transformed into binary arithmetic expansion by replacing each n-variable quaternary basis function with the equivalent $2n$-variable binary basis functions. The resulting expansion is a binary LIA expansion which corresponds to two- or three-level binary circuits. Thus, other than for optimization of quaternary function representation, the new transforms may also be useful for representation and analysis of binary functions. Similar to other arithmetic transforms, the FPQLIA transforms are also useful for efficient representation and parallel calculation of multi-output functions [1].

REFERENCES

[1] M. G. Karpovsky, R. S. Stankovic, and J. T. Astola, *Spectral Logic and its Applications for the Design of Digital Devices*. Hoboken: Wiley-Interscience, 2008.

[2] R. S. Stankovic and J. T. Astola, *Spectral Interpretation of Decision Diagrams*. New York: Springer-Verlag, 2003.

[3] S. N. Yanushkevich, D. M. Miller, V. P. Shmerko, and R. S. Stankovic, *Decision Diagram Techniques for Micro- and Nanoelectronic Design Handbook*. Boca Raton: CRC Press, 2006.

[4] S. Rahardja and B. J. Falkowski, "Application of linearly independent arithmetic transforms in testing of digital circuits," *Electron. Lett.*, vol. 35, pp. 363–364, March 1999.

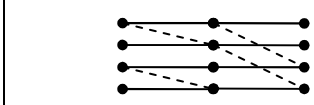
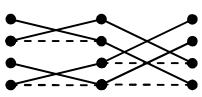
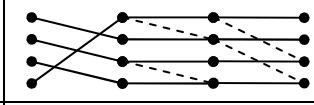
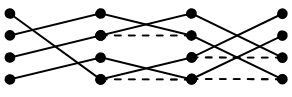
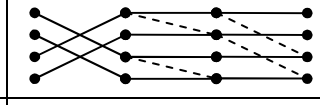
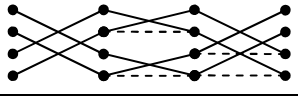
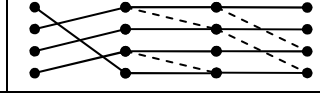
[5] B. J. Falkowski and C. Fu, "Generation and relation of quaternary and binary linearly independent transforms," in *Proc. 36th IEEE Int. Symp. Multiple-Valued Logic*, Singapore, May 2006, CD publication.

[6] B. J. Falkowski and C. Fu, "Classification of fastest quaternary linearly independent arithmetic transforms," in *Proc. 38th IEEE Int. Symp. Multiple-Valued Logic*, Dallas, Texas, USA, May 2008, pp. 169–173.

[7] D. H. Green, "Reed-Muller expansions with fixed and mixed polarities over GF(4)," *IEE Proc. Comput. and Digit. Tech.*, vol. 137, pp. 380–388, Sept. 1990.

[8] C. C. Lozano, B. J. Falkowski, and S. Rahardja, "Algorithms for generation of quaternary fixed polarity arithmetic spectra," in *Proc. 39th IEEE Int. Symp. Circuits and Systems*, Kos, Greece, May 2006, pp. 803–806.

Table 1 – Forward transforms, inverse transforms, and fast inverse flow graphs for 1-variable FPQLIA

Polarity number ω	Class 1			Class 2		
	$T_{1,1}^{\omega}$	$\left(T_{1,1}^{\omega}\right)^{-1}$	Fast flow graph for $\left(T_{1,1}^{\omega}\right)^{-1}$	$T_{2,1}^{\omega}$	$\left(T_{2,1}^{\omega}\right)^{-1}$	Fast flow graph for $\left(T_{2,1}^{\omega}\right)^{-1}$
0	$\begin{bmatrix}1&0&0&0\\1&1&0&0\\1&0&1&0\\1&1&1&1\end{bmatrix}$	$\begin{bmatrix}1&0&0&0\\-1&1&0&0\\-1&0&1&0\\1&-1&-1&1\end{bmatrix}$		$\begin{bmatrix}1&1&1&1\\1&0&1&0\\1&1&0&0\\1&0&0&0\end{bmatrix}$	$\begin{bmatrix}0&0&0&1\\0&0&1&-1\\0&1&0&-1\\1&-1&-1&1\end{bmatrix}$	
1	$\begin{bmatrix}1&1&0&0\\1&0&1&0\\1&1&1&1\\1&0&0&0\end{bmatrix}$	$\begin{bmatrix}0&0&0&1\\1&0&0&-1\\0&1&0&-1\\-1&-1&1&1\end{bmatrix}$		$\begin{bmatrix}1&0&0&0\\1&1&1&1\\1&0&1&0\\1&1&0&0\end{bmatrix}$	$\begin{bmatrix}1&0&0&0\\-1&0&0&1\\-1&0&1&0\\1&1&-1&-1\end{bmatrix}$	
2	$\begin{bmatrix}1&0&1&0\\1&1&1&1\\1&0&0&0\\1&1&0&0\end{bmatrix}$	$\begin{bmatrix}0&0&1&0\\0&0&-1&1\\1&0&-1&0\\-1&1&1&-1\end{bmatrix}$		$\begin{bmatrix}1&1&0&0\\1&0&0&0\\1&1&1&1\\1&0&1&0\end{bmatrix}$	$\begin{bmatrix}0&1&0&0\\1&-1&0&0\\0&-1&0&1\\-1&1&1&-1\end{bmatrix}$	
3	$\begin{bmatrix}1&1&1&1\\1&0&0&0\\1&1&0&0\\1&0&1&0\end{bmatrix}$	$\begin{bmatrix}0&1&0&0\\0&-1&1&0\\0&-1&0&1\\1&1&-1&-1\end{bmatrix}$		$\begin{bmatrix}1&0&1&0\\1&1&0&0\\1&0&0&0\\1&1&1&1\end{bmatrix}$	$\begin{bmatrix}0&0&1&0\\0&1&-1&0\\1&0&-1&0\\-1&-1&1&1\end{bmatrix}$