A FAMILY OF ALL ESSENTIAL RADIX-2 ADDITION/SUBTRACTION MULTI-POLARITY TRANSFORMS: ALGORITHMS AND INTERPRETATIONS IN BOOLEAN DOMAIN

Bogdan J. Falkowski

Marek A. Perkowski

Department of Electrical Engineering,
Portland State University,
P.O. Box 751,
Portland, OR 97207.

ABSTRACT

By investigating some family of elementary order-2 matrices, new transforms of real vectors are introduced. When used for Boolean function transformations, these transforms are one-to-one mappings in a binary / ternary vector space. The concept of different polarities of considered Arithmetic and Adding transforms has been introduced.

1. INTRODUCTION

Encouraged by a multiplicity of applications of Fourier, Walsh and Reed-Muller transforms the authors are investigating new orthogonal transforms that can find applications in Boolean minimization, testing, image coding, cryptography and communication. With respect to the simplicity of the implementation the authors assume that the operations used in the transformation are the ordinary addition and subtraction. One of these transforms is simply the well-known *Hadamard-Walsh transform* [1, 2, 6-8, 13-15, 18] that is applied here to binary and ternary vectors. One of the other considered transforms when applied to binary vectors is known under the name of *Arithmetic transform* [5, 14]. However, this transform has never been applied to ternary vectors. The third transform is *completely new*, is called in this presentation under the name of *Adding transform*, and is applied to ternary and binary vectors.

Considered transforms are obtained by introducing some operations on matrices and considering some family of order-1 matrices. Two new operations on matrices: the row-wise and column-wise joins (concatenations) of two matrices are used in order to create the transforms of radix-2. Later on, the elementary order-2 matrices are expanded by using the standard *tensor product of matrices* known also under the names of *direct* or *Kronecker product* [1, 2, 6, 10, 13, 15, 18].

It has been shown in this paper that when the elementary order-2 matrices are composed of only 0, 1 and - 1 then there are only four essential types of radix-2 transforms (one of them is the identity matrix), since all other permutations of elements 0, 1, and - 1 create the order-2 matrices that can be obtained from the essential types by multiplication with some permutation matrices. Since the identity matrix is a trivial case from the point of view of the transformation then there are only three essential matrices of order-2 that are considered. After expansion of the basic types by using Kronecker product the obtained transforms of higher radices are used to create spectra of binary and ternary vectors.

For each of the three transforms, the interpretation of the meaning of each particular spectral coefficient on Karnaugh map is presented. All mathematical relationships between the number of true, false, and don't care minterms in the areas of Karnaugh maps which correspond to *standard trivial functions* (where the standard trivial function is an area of Karnaugh map corresponding to the given spectral coefficient) are stated for two different codings and all three types of transforms.

In this presentation only ordinary subtraction/addition operations are used. Since the generalized Reed-Muller transforms [3, 4, 5, 11, 12] (with all possible 2^n fixed polarities for n variable Boolean functions) have been found useful in Boolean minimization, design for testability, and image processing, the authors propose here to apply the same idea of fixed polarities for all the three transforms. The concept of different polarities of new transforms is important from the point of view of analysis and synthesis of digital networks - it is already well known, for example, that fixed-polarity Reed-Muller form can have much better implementation for many Boolean functions than standard sum-of-products expression [4]. The same savings from the point of view of the computer memory storing the spectra are valid for the new transforms as well.

The mathematical relationships which exist between the several alternative spectra that may be used to represent any Boolean function (or simply binary or ternary vectors) can be found in another article by the authors [9]. It is possible, for example, to calculate the arithmetic transform of any polarity from Walsh-type of transforms, where the Hadamard-Walsh ordering corresponds to the zero polarity. In the cases of other polarities, the Hadamard-Walsh transform is transformed to the Walsh-type transform still in Hadamard ordering which has, however, reversed signs for all but one row. Hence, there exist together 2^n such combinations, and each of them corresponds to one possible polarity of generalized arithmetic representation of the given Boolean function. The relationships between the considered transforms and the Reed-Muller transform are valid for each polarity. The Reed-Muller expansion of a given polarity can be obtained from either Arithmetic or Adding transforms by replacing in the transform matrix all additions/subtractions operations with a modulo 2 operation and reducing all spectral coefficients modulo 2.

A very important property of the new transforms should also be noticed. In the case of the Reed-Muller transforms there exist more than one expression for an incompletely specified Boolean function [12]. In the case of the new transforms this property is no longer valid - on the contrary, each incompletely specified Boolean function has only a single spectrum. Hence, there is an exact relationship between incompletely specified Boolean functions and their spectra. So, it is always possible for the new transforms to calculate the inverse transforms for incompletely specified Boolean functions. In the case of completely specified Boolean functions all the new transforms as well as the Reed-Muller transform do not lose any information and it is always possible to calculate the inverse transforms.

2. DEFINITIONS OF ESSENTIAL RADIX-2 MATRICES

Some families of matrices will be defined. The building blocks for the definitions are three elementary elements (matrices of orders 1×1) : 0, -1, and $+1$. The following operations on matrices are introduced.

Definition 2.1: A *row-wise join* or *concatenation* of a matrix A of order $n \times m$ and a matrix B of order $n \times m$ is the partitioned matrix C of order $n \times 2m$ such that its first m rows are exactly the same as the rows of matrix A and the rows from $m+1$ to $2m$ are exactly the same as the rows of matrix B. This operator is denoted by the symbol "RWJ".

Definition 2.2: A column-wise join or *concatenation* of a matrix A of order $n \times m$ and a matrix B of order $n \times m$ is the partitioned matrix C of order $2n \times m$ such that its first n columns are exactly the same as the columns of matrix A and the columns from $n+1$ to $2n$ are exactly the same as the columns of matrix B. This operator is denoted by the symbol "CWJ".

Let us apply the operator CWJ to three elementary matrices of orders 1×1 for all possible concatenations of these matrices. There are 9 different matrices of order 2×1 as the result of the application of the CWJ to all three elementary matrices. They are shown in Fig. 1.

Let us now apply the operator RWJ to all possible combinations of matrices from Fig. 1. There exist together 81 different matrices of order 2×2 - some of them are non-orthogonal and are not of interest in this case. All orthogonal matrices can be classified into *four basic types* (denoted by I, II, III, and IV). The first 45 matrices with marked 4 basic types are shown in Fig. 2. The way of the generation of the remaining 36 matrices should be obvious from this picture. In each row of the picture, one of the nine matrices from Fig. 1 is the first matrix on which the RWJ operation is performed with all matrices from Fig. 1. The same 81 matrices could be generated by first applying the operator RWJ to the basic elements and obtaining the matrices of order 1×2 (9 such matrices), and next applying the operator CWJ to the elementary row matrices obtained in the previous step. The latter operation is performed in a way similar to the operation of the generation of the 4×4 matrices obtained by the operator RWJ described previously.

All basic types have been found by observing the following property of these matrices: any matrix (of order 2×2) from the basic type can be obtained from the other matrix of the same type by applying some of the following operations on matrices: mutual transposition of rows, mutual transposition of columns, change of the signs in the whole row, change of the signs in the whole column. Hence, there exist only four elementary types of the matrices of orders 2×2 composed out of the elements 0, + 1, and -1. One of this types, denoted by the type I^*, is the identity matrix, and therefore is not interesting from the point of view of the transformations. Then, three types of orthogonal, radix-2 matrices exist and their application to the transformation of binary and ternary vectors are presented in the sequel. Out of each of the three types, one particular representative has to be chosen. In our case, in order to get some already known transforms, the matrices denoted by * in Fig. 2 have been chosen. The three elementary matrices of orders 2×2 (other than identity) will be denoted by symbols H_2 *(Hadamard transform* [1, 2, 6-9, 13, 15, 18]), AR_2 *(Arithmetic transform* [5, 14]), and AD_2 *(Adding transform).*

The Walsh functions in Hadamard order are generated when the standard Kronecker product of the elementary Hadamard matrix H_2 is performed with itself. Similarly, the Arithmetic transform of higher orders is obtained by successive application of the Kronecker product to the core matrix AR_2. The same is valid for the Adding transform as well - the core matrix being AD_2. When all these three elementary matrices are denoted by the same symbol TR_2, then

$$TR_N = (TR_2)^{[n]}, \tag{1}$$

where $[n]$ in the exponent means the application of the Kronecker product n times, N is the order of the transform matrix, and $n = \log_2 N$.

It will be shown in the sequel, how the obtained transforms are used to create spectra of ternary and binary vectors. Since the detailed description of the properties of Hadamard-Walsh spectrum of Boolean functions has been presented elsewhere in this Proceedings [8], only the application and properties of Arithmetic and Adding transforms will be considered.

3. GENERALIZED ARITHMETIC AND ADDING TRANSFORMS

The Arithmetic transform AR_N has been used for the generation of an *arithmetic canonic expansion* of Boolean functions [5, 14, 17]. In the literature, this expansion has been used only for completely specified Boolean functions. The authors propose *three extensions* of currently used Arithmetic transform. First, it is proposed to use this transform not only for completely specified Boolean functions but for incompletely specified ones as well. Hence, the *Arithmetic transform can be applied not only to binary but also to ternary vectors.* Secondly, *two types of codings of Boolean functions are used.* In the first type, in the case of the completely specified Boolean function, the true minterms of the function are represented by 1 and false minterms by 0. When the second

CH2868-8/90/0000-2913$1.00

coding is used, the true minterms are represented by -1 and the false minterms by 1. In the case of the incompletely specified Boolean functions, in the first coding scheme the don't care minterms are represented by 0.5, and in the second coding scheme by 0. The coding of the true and false minterms for the functions with don't cares is the same as the one for the completely specified Boolean functions. The same types of coding schemes have been used for Hadamard-Walsh spectrum of Boolean functions and the corresponding Walsh spectra are known in the literature under the names of the R spectrum (for the first type of coding, later called the R coding), and the S spectrum (for the second type of coding, called the S coding), accordingly [7, 8, 13]. Thirdly, the *notion of the polarity of the Arithmetic transform is introduced.* Since for the Boolean function having n variables there exist 2^n possible substitutions of a given i^{-th} variable by its complement then there is possible to have an equal number (2^n) of possible expansions in which each variable is in either complemented or not-complemented form. These all possible expansions will be called the *generalized Arithmetic transforms* and are classified by their *polarities*. The latter notion is similar to the one used for Reed-Muller transforms [11, 12] and will be rewritten for our needs.

Definition 3.1: A *polarity* number is calculated by taking the decimal equivalent of the n-bit straight binary code formed by writing a 0 or a 1 for each variable dependently whether this variable is in positive or complemented form, respectively.

Let us illustrate the introduced notions on the following example.

Example 3.1: An example of the calculation of the Arithmetic transform of four variable completely specified Boolean function in the R coding is shown in Fig. 3. The transform is in the zero polarity, and all the variables describing the coefficients of the arithmetic canonical expansion are positive. In the matrix AR from Fig. 3 the rows correspond to the standard trivial functions (explained in more detail in Section 4). The arithmetic canonical expansion for this function corresponding to the vector C in Fig. 3 is as follows:

$$f(X) = x_3 x_1 + x_4 - x_4 x_2 x_1 - x_4 x_3 - x_4 x_3 x_1 + x_4 x_3 x_2 + x_4 x_3 x_2 x_1 \tag{2}$$

The addition symbol in the canonic arithmetic expansion "+" is an arithmetic addition and not Boolean "or". The value of a given minterm can be obtained from the arithmetic expansion of any polarity when the binary values of variables x_4, x_3, x_2, and x_1 equivalent to the minterm are substituted in the expansion, the value of each term in the expansion is calculated logically and the ones that correspond to the terms that are true after the first substitution are arithmetically added or subtracted. This rule is valid for both codings of completely and incompletely specified Boolean functions.

As it can be easily checked, the values of all the minterms of this function can be generated from its canonical arithmetic expansion by replacing the literals x_4, x_3, x_2, and x_1 with the binary code of a given minterm. For instance, the minterm 0000 has the value 0, and the minterm 1111 has the value $1 + 1 - 1 - 1 - 1 + 1 + 1 = 1$.

The other arithmetic canonical expansion can be obtained for this function from the second coding S. The coefficients for the second expansion are shown in Fig. 4 (the vector on the right side of this picture with the arrow AR pointing to it). Since the polarity is zero again, then the variables of the Boolean function occurring in the terms of this arithmetic canonical expansion are exactly the same as in the previous case. And again, it can be easily checked that the value of a minterm of the function in the S coding can be obtained from the second canonical form by substituting the variables x_4, x_3, x_2, and x_1 with the values resulting from the binary code of this minterm.

The Arithmetic transform can be applied to both completely and incompletely specified Boolean functions in both codings S and R. Similarly to the Arithmetic transform, the Adding transform can be applied to both completely and incompletely specified Boolean functions. Two types of codings can be used for this transform as well. Moreover, the Adding transform can have the same polarities as the Arithmetic transform. Before showing the examples of other polarities, and applications of the transforms to incompletely specified Boolean functions let us state the fundamental relationship between both these transforms. *For the zero polarity, both matrices AR_N and AD_N are inverses of each other, i.e.,*

$$(AR_N)^{-1} = AD_N \tag{3}$$

and

$$(AD_N)^{-1} = AR_N \tag{4}$$

The transform matrix for the Adding transform looks similar to the transform matrix for the Arithmetic transform shown in Fig. 3 - the only difference being the fact that all the entries are $+1$ in all the matrix i.e., all -1 in the matrix for the Arithmetic transform should be replaced by $+1$ for the Adding transform and all $+1$ are not changed. The next two examples are only for the polarity zero.

Example 3.2: An example of both Arithmetic and Adding spectra for the same four variable completely specified Boolean function X_R (in the coding R) and X_S (in the coding S) is shown in Fig. 4. It is the same function that was used in Example 3.1. The arrows on this picture show the applications of the Arithmetic AR and Adding AD transforms accordingly. It is also shown that both transforms' matrices are inverses of each other.

Example 3.3: Transformations of the same four variable incompletely specified Boolean functions in two codings by means of the Arithmetic and Adding transforms are shown in Fig. 5. Even for incompletely specified Boolean functions both transforms' matrices are inverses of each other.

It is very important to notice that both Arithmetic and Adding spectra are the *canonical representations* of completely and incompletely specified Boolean functions for any polarity. The latter property of both transforms makes them especially distinct from other related transforms. For example, the Reed-Muller transform that has the same transformation matrix as the Adding transform (for any given polarity this correspondence exists) and only the operations of addition are executed "modulo-2" instead of normal arithmetic addition as in the case of the Adding transform, does not have a canonical form for the transformation of incompletely specified Boolean function [12].

An important relationship exists between the Arithmetic spectral coefficients calculated according to R and S codings for both completely and incompletely specified Boolean functions and for all polarities, accordingly. When arr_I (where I are different natural numbers) denotes the coefficients calculated for the R coding, and ars_I denotes the coefficients calculated for the S coding then for all ar_I but ar_0 the following formula holds:

$$arr_I = -\frac{1}{2}\, ars_I \tag{5}$$

For arr_0 and ars_0 Equation (5) is not valid. Instead, the following formula holds for such a case:

$$arr_0 = \frac{1}{2}\,(1 - ars_0). \tag{6}$$

Let us notice, that the same relationship as Equation (5) is valid for Hadamard-Walsh spectral coefficients [8, 13]. However, Equation (5) does not hold for all coefficients from the Adding spectrum what can be easily checked in Fig. 4. and 5. Equation (6) is valid also for ad_0 spectral coefficients, where arr_0 is replaced by adr_0, and ars_0 is replaced by ads_0, respectively.

Let us now show examples of the generalized Arithmetic transform for the same completely specified Boolean function. Due to the lack of the space only one example of the generalized Arithmetic transform for the polarity 0011 is shown and only for the completely specified Boolean function. The Adding transform can be calculated for this polarity by replacing all -1 by $+1$, and rewriting all $+1$ from the matrix describing the Arithmetic transform. Only one coding R is shown. It should already be obvious from the previous examples, how to calculate the generalized Arithmetic and Adding transforms for any coding and any Boolean function.

Example 3.4: The calculation of the Arithmetic transform in the polarity 0011 for the four variable completely specified Boolean function is shown in Fig. 6. The coefficients of the arithmetic canonical expansion for this polarity have positive and complemented forms as shown in Fig. 6.

Example 3.5: The calculation of the inverse Arithmetic transform for the polarity 0011 for the function from the previous example is shown in Fig. 7.

Let us notice, that for not zero polarity the relationships (3) and (4) are no longer valid. The methods that show how to calculate the forward and inverse Arithmetic and Adding transforms for any polarity without the necessity of inversing the forward transform are shown in [9].

4. LINKS OF ARITHMETIC AND ADDING TRANSFORMS WITH CLASSICAL LOGIC DESIGN

Let us show the real meaning of the Arithmetic and Adding spectral coefficients in classical logical terms. Let symbol a_I denotes the spectral coefficient from either Arithmetic or Adding transform in any coding. The definition of *standard trivial functions* and their relationships to the spectral coefficients (from both Arithmetic and Adding spectra) follows.

Definition 5.1: Each spectral coefficient a_I gives a *correlation value* between the Boolean function F and a *standard trivial function* u_I corresponding to this coefficient. The standard trivial functions for the spectral coefficients are, respectively, for the coefficients a_I (where $I = 0$) - the minterm of the Boolean function corresponding to a given polarity denoted by u_0, for the first order coefficients a_I (where $I = i$, $i \neq 0$) - the minterm of the Boolean function u_0 and one of its neighbors, in turn, denoted by u_i, for the second order coefficients a_I (where $I = ij$, $i \neq 0$, $j \neq 0$) - the minterm of the Boolean function u_0 and three of its neighbors, in turn, denoted by u_{ij}, for the third order coefficients a_I and ($I = ijk$, $i \neq 0$, $j \neq 0$, $k \neq 0$) - the minterm of the Boolean function u_0 and seven of its neighbors, in turn, denoted by u_{ijk}, etc.

Since the formulas for the calculation of spectral coefficients are derived for both spectra then the necessary symbols are introduced together. Moreover, let us expand our considerations for incompletely specified Boolean functions as well. The following symbols will be used. Let a_I be the number of true minterms of Boolean function F, where both the function F and the standard trivial function u_I have the logical values 1; let b_I be the number of true minterms of Boolean function F, where the function F has the logical value 1 and the standard trivial function u_I has the logical value 0; let c_I be the number of don't care minterms of Boolean function F, where the standard trivial function u_I has the logical value 1; let d_I be the number of don't care minterms of Boolean function F, where the standard trivial function u_I has the logical value 0.

The arr_I Arithmetic spectral coefficients for the *completely* specified Boolean function in the coding R, having n variables, can be defined in the following way :

$$arr_0 = a_0, \tag{7}$$

and

$$arr_I = a_I - b_I \qquad \text{when } I \neq 0. \tag{8}$$

The arr_I Arithmetic spectral coefficients for the *incompletely* specified Boolean function in the coding R, having n variables, can be defined in the following way :

$$arr_0 = a_0 + \frac{1}{2}\, c_0, \tag{9}$$

and

$$arr_I = (a_I - b_I) + \frac{1}{2}\,(c_I - d_I), \qquad \text{when } I \neq 0. \tag{10}$$

The formulas for the calculation of the Arithmetic spectral coefficients in the coding S can be found from Equations (5, 6, 7-10).

When the Adding spectrum in the coding R is to be calculated then the formulas for its coefficients are the same as Equations (7-10) - the only difference being the replacement in all these formulas of the sign $-$ onto $+$.

Example 5.1: The standard trivial functions for the same completely specified Boolean function as in Example 3.1 for the polarity 0000 are shown in Fig. 8. The circles denote the areas where the standard trivial functions have the logical values 1 while the triangles denote the areas where the standard trivial functions have the logical values 0. The coefficients of the arithmetic canonical expansion for this polarity have only positive forms which are written below the Karnaugh maps showing the corresponding standard trivial functions for each coefficient. One can easily check that by calculating spectral coefficients according to the above formulas one obtains exactly the same results as previously.

Example 5.2: The standard trivial functions for the function from previous example for the polarity 0011 are shown in Fig. 9. Since it is the same polarity as the one considered previously then one can easily check that again Equations (7-10) give the correct results.

5. CONCLUSION

By using two types of coding, each of the three basic types of considered transforms has two types of spectra for a given Boolean function. Our considerations are confined only to the transforms that are created by Kronecker products of three elementary order-2 matrices. Such a limitation has been applied in order to satisfy the requirements of hardware/software realizations of transforms in recursive data-flow or systolic architectures [2, 6, 16]. This approach enables to create for each of the considered transform the corresponding fast transforms according to Good's formula [2, 6, 18]. the spectral classification of Boolean functions,

Since the Walsh spectral coefficients have received recently a considerable attention for network analysis, synthesis and test purposes then it is interesting to consider applications of the new transforms in these areas as well. For instance, the authors see the possibility of using these transforms for spectral-based testing, layered Boolean network decomposition and adaptive image coding. These are the topics of ongoing research of the authors.

Besides the applications in designing and testing of digital circuits the new transforms can have applications in multidimensional digital signal processing (including image processing) [18]. It is well known that the most simple representation form of images is a binary or ternary vector representation. By applying new transforms the structure of the binary or ternary images can be represented in the compact form.

6. ACKNOWLEDGMENT

This research was financially supported in part by the Faculty Development Grant 90-050-1009-2000 from Portland State University.

7. REFERENCES

[1] S.S. Agaian, *Hadamard Matrices and Their Applications.* Lect. Notes in Mathematics No. 1168, Berlin: Springer-Verlag, 1980.

[2] K.G. Beauchamp, *Applications of Walsh and Related Functions.* New York: Academic Press, 1984.

[3] Ph.W. Besslich, "Efficient Computer Method for EXOR Logic Design," *IEE Proc. E., Comput. & Digital Tech.*, vol. 130, No. 6, pp. 203-206, Nov. 1983.

[4] Ph.W. Besslich, Spectral Processing of Switching Functions Using Signal-Flow Transformations, in *Spectral Techniques and Fault Detection.* (M.G. Karpovsky, ed.), Orlando: Academic Press, 1985.

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

[6] D.F. Elliot, and K.R. Rao, *Fast Transforms: Algorithms, Analysis, Applications.* London: Academic Press, 1982.

[7] B.J. Falkowski, and M.A. Perkowski, "Algorithms for the Calculation of Hadamard-Walsh Spectrum for Completely and Incompletely Specified Boolean Functions," *Accepted for publication in Proc. of IEEE Int. Phoenix Conf. Comput. & Communications (IPCCC)*, Scottsdale, Arizona, March 1990.

[8] B.J. Falkowski, and M.A. Perkowski, "Essential Relations Between Classical and Spectral Approaches to Analysis, Synthesis, and Testing of Completely and Incompletely Specified Boolean Functions," *Accepted for publication in Proc. of IEEE ISCAS*, New Orleans, Louisiana, May 1990.

[9] B.J. Falkowski, and M.A. Perkowski, "On Mutual Relationships Among Arithmetic, Adding, Reed-Muller, and Hadamard-Walsh Multi-Polarity Transforms," *Submitted to IEEE International Conference on Computer Design (ICCD)*, 1990.

[10] A. Graham, *Kronecker Products and Matrix Calculus With Applications.* Chichester: Ellis Horwood, 1981.

[11] D. Green, *Modern Logic Design.* Wokingham: Addison-Wesley, 1986.

[12] D. Green, "Reed-Muller Expansions of Incompletely Specified Functions," *IEE Proc. E., Comput. & Digital Tech.*, vol. 134, No. 5, pp. 228-236, Sept. 1987.

[13] S.L. Hurst, D.M. Miller, and J.C. Muzio, *Spectral Techniques in Digital Logic.* London: Academic Press, 1985.

[14] S.L. Hurst, The Interrelationships Between Fault Signatures Based Upon Counting Techniques, in *Developments in Integrated Circuit Testing*, (D.M. Miller, ed.), London: Academic Press, 1987.

[15] M.G. Karpovsky, *Finite Orthogonal Series in the Design of Digital Devices.* New York: Wiley, 1976.

[16] B.R.K. Reddy, and A.L. Pai, "Reed-Muller Transform Image Coding," *Computer Vision, Graphics, and Image Processing*, vol. 42, pp. 48-61, 1988.

[17] A.J. Rubio-Ayuso, "Algebraic Representation of Boolean Functions," *Int. J. Electronics*, vol. 56, pp. 735-739, May 1984.

[18] L.P. Yaroslavsky, *Digital Picture Processing - An Introduction.* Berlin: Springer-Verlag, 1985.

$$\begin{bmatrix}1\\1\end{bmatrix}\ \begin{bmatrix}1\\-1\end{bmatrix}\ \begin{bmatrix}-1\\1\end{bmatrix}\ \begin{bmatrix}-1\\-1\end{bmatrix}\ \begin{bmatrix}1\\0\end{bmatrix}\ \begin{bmatrix}0\\1\end{bmatrix}\ \begin{bmatrix}0\\-1\end{bmatrix}\ \begin{bmatrix}-1\\0\end{bmatrix}\ \begin{bmatrix}0\\0\end{bmatrix}$$

Fig.1. Nine different matrices of order 2 x 1.

$$\underset{\text{NO}}{\begin{bmatrix}1&1\\1&1\end{bmatrix}}\ \underset{\text{IV}^*}{\begin{bmatrix}1&1\\1&-1\end{bmatrix}}\ \underset{\text{IV}}{\begin{bmatrix}1&-1\\1&1\end{bmatrix}}\ \underset{\text{NO}}{\begin{bmatrix}1&-1\\1&-1\end{bmatrix}}\ \underset{\text{II}^*}{\begin{bmatrix}1&0\\1&1\end{bmatrix}}\ \underset{\text{II}}{\begin{bmatrix}1&1\\1&0\end{bmatrix}}\ \underset{\text{III}}{\begin{bmatrix}1&0\\1&-1\end{bmatrix}}\ \underset{\text{III}}{\begin{bmatrix}1&-1\\1&0\end{bmatrix}}\ \underset{\text{NO}}{\begin{bmatrix}1&0\\1&0\end{bmatrix}}$$

$$\underset{\text{IV}}{\begin{bmatrix}1&1\\-1&1\end{bmatrix}}\ \underset{\text{NO}}{\begin{bmatrix}1&1\\-1&-1\end{bmatrix}}\ \underset{\text{NO}}{\begin{bmatrix}1&-1\\-1&1\end{bmatrix}}\ \underset{\text{IV}}{\begin{bmatrix}1&-1\\-1&-1\end{bmatrix}}\ \underset{\text{III}^*}{\begin{bmatrix}1&0\\-1&1\end{bmatrix}}\ \underset{\text{II}}{\begin{bmatrix}1&0\\-1&0\end{bmatrix}}\ \underset{\text{II}}{\begin{bmatrix}1&0\\-1&-1\end{bmatrix}}\ \underset{\text{III}}{\begin{bmatrix}1&-1\\-1&0\end{bmatrix}}\ \underset{\text{NO}}{\begin{bmatrix}1&0\\-1&0\end{bmatrix}}$$

$$\underset{\text{IV}}{\begin{bmatrix}-1&1\\1&1\end{bmatrix}}\ \underset{\text{NO}}{\begin{bmatrix}-1&1\\1&-1\end{bmatrix}}\ \underset{\text{NO}}{\begin{bmatrix}-1&-1\\1&1\end{bmatrix}}\ \underset{\text{IV}}{\begin{bmatrix}-1&-1\\1&-1\end{bmatrix}}\ \underset{\text{II}}{\begin{bmatrix}-1&0\\1&1\end{bmatrix}}\ \underset{\text{III}}{\begin{bmatrix}-1&1\\1&0\end{bmatrix}}\ \underset{\text{III}}{\begin{bmatrix}-1&0\\1&-1\end{bmatrix}}\ \underset{\text{II}}{\begin{bmatrix}-1&-1\\1&0\end{bmatrix}}\ \underset{\text{NO}}{\begin{bmatrix}-1&0\\1&0\end{bmatrix}}$$

$$\underset{\text{NO}}{\begin{bmatrix}-1&1\\-1&1\end{bmatrix}}\ \underset{\text{IV}}{\begin{bmatrix}-1&1\\-1&-1\end{bmatrix}}\ \underset{\text{IV}}{\begin{bmatrix}-1&-1\\-1&1\end{bmatrix}}\ \underset{\text{NO}}{\begin{bmatrix}-1&-1\\-1&-1\end{bmatrix}}\ \underset{\text{III}}{\begin{bmatrix}-1&0\\-1&1\end{bmatrix}}\ \underset{\text{III}}{\begin{bmatrix}-1&1\\-1&0\end{bmatrix}}\ \underset{\text{II}}{\begin{bmatrix}-1&0\\-1&-1\end{bmatrix}}\ \underset{\text{II}}{\begin{bmatrix}-1&-1\\-1&0\end{bmatrix}}\ \underset{\text{NO}}{\begin{bmatrix}-1&0\\-1&0\end{bmatrix}}$$

$$\underset{\text{II}}{\begin{bmatrix}1&1\\0&1\end{bmatrix}}\ \underset{\text{II}}{\begin{bmatrix}1&1\\0&-1\end{bmatrix}}\ \underset{\text{III}}{\begin{bmatrix}1&-1\\0&1\end{bmatrix}}\ \underset{\text{III}}{\begin{bmatrix}1&-1\\0&-1\end{bmatrix}}\ \underset{\text{I}^*}{\begin{bmatrix}1&0\\0&1\end{bmatrix}}\ \underset{\text{NO}}{\begin{bmatrix}1&1\\0&0\end{bmatrix}}\ \underset{\text{I}}{\begin{bmatrix}1&0\\0&-1\end{bmatrix}}\ \underset{\text{NO}}{\begin{bmatrix}1&-1\\0&0\end{bmatrix}}\ \underset{\text{NO}}{\begin{bmatrix}1&0\\0&0\end{bmatrix}}$$

NO - Not Orthogonal matrix

Fig.2. First 45 possible matrices with marked 4 basic types.

$$\underbrace{\begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
-1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
-1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&-1&-1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
-1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
1&-1&0&0&-1&1&0&0&0&0&0&0&0&0&0&0\\
1&0&-1&0&-1&0&1&0&0&0&0&0&0&0&0&0\\
-1&1&1&-1&1&-1&-1&1&0&0&0&0&0&0&0&0\\
-1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
1&-1&0&0&0&0&0&0&-1&1&0&0&0&0&0&0\\
1&0&-1&0&0&0&0&0&-1&0&1&0&0&0&0&0\\
-1&1&1&-1&0&0&0&0&1&-1&-1&1&0&0&0&0\\
1&0&0&0&-1&0&0&0&-1&0&0&0&1&0&0&0\\
-1&1&0&0&1&-1&0&0&1&-1&0&0&-1&1&0&0\\
-1&0&1&0&1&0&-1&0&1&0&-1&0&-1&0&1&0\\
1&-1&-1&1&-1&1&1&-1&-1&1&1&-1&1&-1&-1&1
\end{bmatrix}}_{AR}
\underbrace{\begin{bmatrix}0\\0\\0\\0\\0\\1\\0\\1\\1\\1\\1\\0\\0\\0\\1\\1\end{bmatrix}}_{X_R}
=
\underbrace{\begin{bmatrix}0\\0\\0\\0\\0\\1\\0\\0\\1\\0\\0\\-1\\-1\\-1\\1\\1\end{bmatrix}}_{C}
\begin{matrix}1\\X_1\\X_2X_1\\X_2X_1\\X_3\\X_3X_1\\X_3X_2\\X_3X_2X_1\\X_4\\X_4X_1\\X_4X_2\\X_4X_2X_1\\X_4X_3\\X_4X_3X_1\\X_4X_3X_2\\X_4X_3X_2X_1\end{matrix}$$

Fig.3. Calculation of Arithmetic transform for completely specified Boolean function.

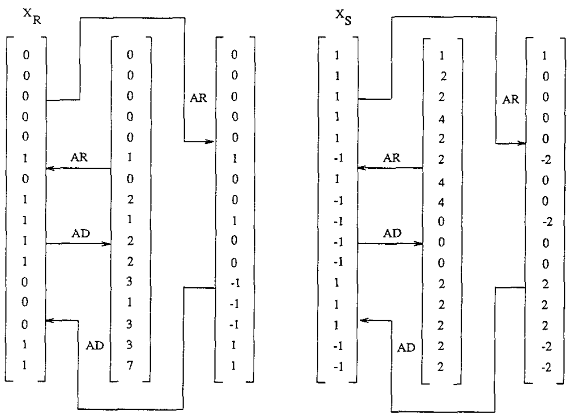

Fig.4. Calculation of Arithmetic and Adding transforms for 4 variable completely specified Boolean function.

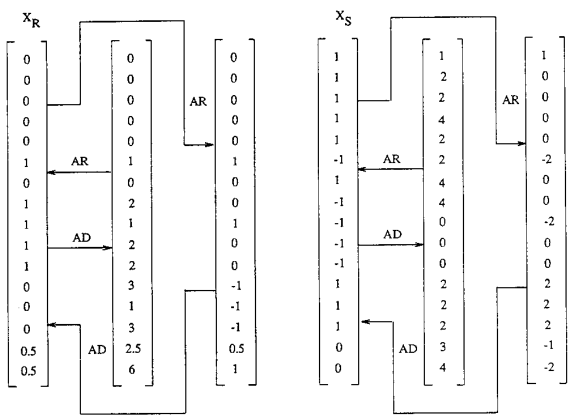

Fig.5. Calculation of Arithmetic and Adding transforms for 4 variable incompletely specified Boolean function.

$$
\overset{AR}{\begin{bmatrix}
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&-1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&-1&0&0&0&0&0&0&0&0&0&0&0&0\\
1&-1&-1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&-1&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&-1&1&0&0&1&-1&0&0&0&0&0&0&0&0\\
0&-1&0&1&0&1&0&-1&0&0&0&0&0&0&0&0\\
-1&1&1&-1&1&-1&-1&1&0&0&0&0&0&0&0&0\\
0&0&0&-1&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&-1&1&0&0&0&0&0&0&1&-1&0&0&0&0\\
0&-1&0&1&0&0&0&0&0&1&0&-1&0&0&0&0\\
-1&1&1&-1&0&0&0&0&1&-1&-1&1&0&0&0&0\\
0&0&0&1&0&0&0&-1&0&0&0&-1&0&0&0&1\\
0&0&1&-1&0&0&-1&1&0&0&-1&1&0&0&1&-1\\
0&1&0&-1&0&-1&0&1&0&-1&0&1&0&1&0&-1\\
1&-1&-1&1&-1&1&1&-1&-1&1&1&-1&1&-1&-1&1
\end{bmatrix}}
\overset{X_R}{\begin{bmatrix}0\\0\\0\\0\\0\\1\\0\\1\\1\\1\\1\\0\\0\\0\\1\\1\end{bmatrix}}
=
\overset{C}{\begin{bmatrix}0\\0\\0\\0\\1\\-1\\0\\0\\0\\1\\1\\-1\\0\\0\\-2\\1\end{bmatrix}}
\begin{matrix}1\\ \bar{x}_1\\ \bar{x}_2\bar{x}_1\\ \bar{x}_2\bar{x}_1\\ x_3\\ x_3\bar{x}_1\\ x_3\bar{x}_2\\ x_3\bar{x}_2\bar{x}_1\\ x_4\\ x_4\bar{x}_1\\ x_4\bar{x}_2\\ x_4\bar{x}_2\bar{x}_1\\ x_4x_3\\ x_4x_3\bar{x}_1\\ x_4x_3\bar{x}_2\\ x_4x_3\bar{x}_2\bar{x}_1\end{matrix}
$$

Fig.6. Calculation of Arithmetic transform for completely specified Boolean function for polarity 0011.

$$
\overset{AR^{-1}}{\begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0\\
1&0&1&0&1&0&1&0&0&0&0&0&0&0&0&0\\
1&1&0&0&1&1&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0\\
1&0&1&0&0&0&0&0&1&0&1&0&0&0&0&0\\
1&1&0&0&0&0&0&0&1&1&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0\\
1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0\\
1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0
\end{bmatrix}}
\overset{C}{\begin{bmatrix}0\\0\\0\\0\\1\\-1\\0\\0\\0\\1\\1\\-1\\0\\0\\-2\\1\end{bmatrix}}
=
\overset{X_R}{\begin{bmatrix}0\\0\\0\\0\\0\\1\\0\\1\\1\\1\\1\\0\\0\\0\\1\\1\end{bmatrix}}
$$

Fig.7. Calculation of an inverse Arithmetic transform for completely specified Boolean function for polarity 0011.

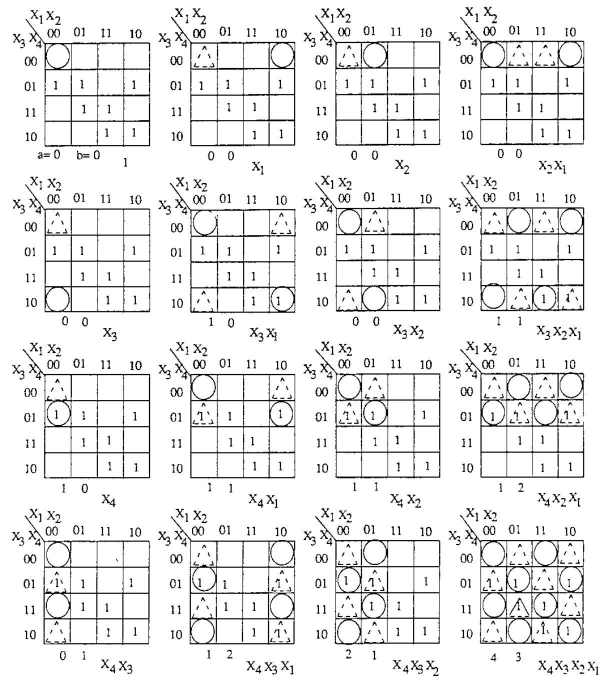

Fig.8. Standard trivial functions corresponding to all arithmetic coefficients for polarity 0000.

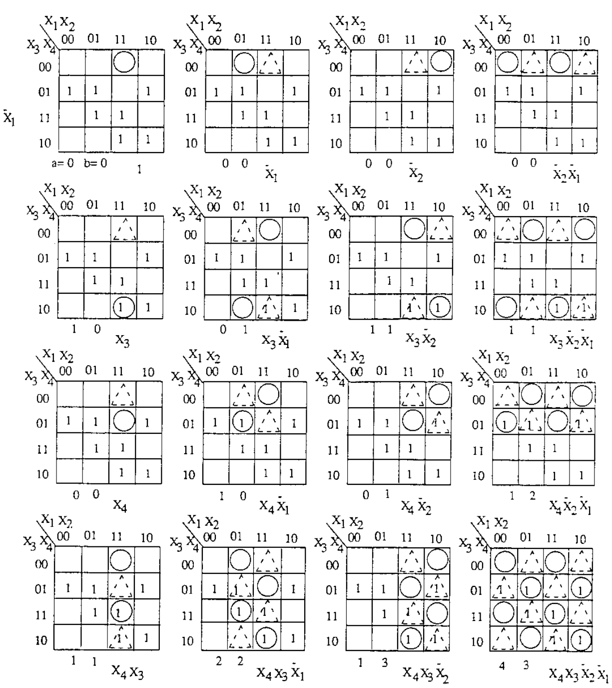

Fig.9. Standard trivial functions corresponding to all arithmetic coefficients for polarity 0011.