VLSI DESIGN
2000, Vol. 11, No. 2, pp. 85–105
Reprints available directly from the publisher
Photocopying permitted by license only

Spectral Interpretation and Applications of Decision Diagrams

BOGDAN J. FALKOWSKI[a,*] and RADOMIR S. STANKOVIĆ[b]

[a]*Nanyang Technological University, School of Electrical and Electronic Engineering, Block S1, Nanyang Avenue, Singapore 639798;* [b]*Braće Taskovića 17/29, 18 000 Niš, Yugoslavia*

(Received 5 May 1998; In final form 15 March 1999)

Different Decision Diagrams (DDs) for representation of discrete functions are discussed. DDs can be derived by applying some reduction rules to Decision Trees (DTs) which in turn are graphical representations of some function expressions for discrete functions. Various DTs and their generalizations based on lesser known AND-EXOR rather than AND-OR expressions are surveyed. Finally, the concept of spectral interpretation of DDs and some of their applications and ways of calculation are also presented.

Keywords: Logic design, decision diagrams, spectral techniques, AND-EXOR expressions, bit-level decision diagrams, word-level decision diagrams

1. INTRODUCTION

Manipulations and calculations with discrete functions are fundamental tasks in Computer Science and Engineering. Many problems in digital system design and testing can be expressed as a sequence of operations on discrete functions. The performance of CAD systems used in solving various problems in this area strongly depends on the efficiency of representation of discrete functions.

Decision diagrams (DDs) [1, 5] have proved very convenient data structure for discrete function representations, permitting manipulations and calculations with large discrete functions efficiently in terms of time and space. In many applications, as for example those involving large matrices, conventional algorithms are significantly improved by using DDs [11]. In logic design, such applications relate to the basic problems of the design, verification and testing of logical networks [13, 72, 79]. Therefore, DDs based packages are to become, or have already been, a standard part of many CAD tools in logic design.

This paper surveys basic concepts in theory and applications of DDs representations. Presentation is based on the following principle which, at the same time, determined the organization of the paper.

*Corresponding author. Tel.: (65) 790-4521, Fax: (65) 791-2687, e-mail: efalkowski@ntu.edu.sg

An easy way towards studying, understanding and discussing applications of DDs is reached by realization that DDs are derived by the reduction of the corresponding decision trees (DTs), which are graphical representations of some functional expansions for discrete functions. Therefore, such expansions and related DTs are first briefly discussed (Sections 2 and 3). The spectral interpretation of DTs (Section 4) permits uniform consideration of various DDs (Section 5) and explains their efficiency in solving various problems in logic design and other areas as well as in spectral transform calculation (Sections 6–8).

2. AND-EXOR EXPRESSIONS

In an algebraic approach to switching theory and logic design, switching functions are often considered as elements of the vector space $GF_2(C_2^n)$ of functions on finite dyadic groups[1] into the Galois field $GF(2)$. Such a consideration appears convenient for different reasons, ranging from efficiency of EXOR representations [76], their suitability for testing and other advantages offered by EXOR-based designs [72], up to the possibility to extend the theory to multiple-valued (MV) functions. In many cases, such extensions can hardly be done through the Boolean algebra-based approaches [56].

Various classes of AND-EXOR expressions are defined in this vector space, starting from the earlier works by Zhegalkin [105, 106], Reed [70] and Muller [55], up to the recent ones [19–22, 25, 40, 58, 59, 64, 67, 73, 81, 100]. Spectral transforms interpretations for some of these expressions were also given and considered, in earlier works [2, 32, 33, 40, 41, 84] up to the recent ones [34, 44, 58, 78, 81, 85, 101]. In these interpretations, AND-EXOR representations are considered as Fourier series-like expansions in $GF_2(C_2^n)$ [31, 73].

3. DECISION TREES

The AND-EXOR expressions are closely related to the decision diagrams representations of switching functions. Moreover, it may be said that DDs are graphical representations of the corresponding AND-EXOR expansions [75, 78], since both representations are based upon the following function decomposition in $GF_2(C_2^n)$:

$$f = \bar{x}_i f_0 \oplus x_i f_1, \tag{1}$$

$$f = f_0 \oplus x_i f_2, \tag{2}$$

$$f = f_1 \oplus \bar{x}_i f_2, \tag{3}$$

where $f_0 = f(x_i = 0)$, $f_1 = f(x_i = 1)$, and $f_2 = f_0 \oplus f_1$.

The relations (1), (2) and (3) are called the Shannon (S), positive Davio (pD), and negative Davio (nD) expansions, respectively.

3.1. Shannon DTs

A recursive application of (1) to all variables of a function f leads to the well known complete disjunctive form. The Shannon trees [78], also called the binary decision trees (BDTs) [5], are graphical representations of this expression [79].

Example 1 If we expand a function $f(x_1, x_2, x_3)$ by using the Shannon expansion with respect to x_1, then

$$f = \bar{x}_1 f_0 \oplus x_1 f_1.$$

By expanding f_0 and f_1 with respect to x_2, we have

$$f_0 = \bar{x}_2 f_{00} \oplus x_2 f_{01}, \quad f_1 = \bar{x}_2 f_{10} \oplus x_2 f_{11}.$$

The S-expansion of f_{00}, f_{01}, f_{10} and f_{11} with respect to x_3 gives:

$$f_{00} = \bar{x}_3 f_{000} \oplus x_3 f_{001}, \quad f_{01} = \bar{x}_3 f_{010} \oplus x_3 f_{011},$$
$$f_{10} = \bar{x}_3 f_{100} \oplus x_3 f_{101}, \quad f_{11} = \bar{x}_3 f_{110} \oplus x_3 f_{111}.$$

[1] The finite dyadic group of order 2^n, C_2^n consists of the set of binary n-tuples $(x_1, \ldots, x_n)$, $x_i \in \{0, 1\}$, under the operation of componentwise modulo 2 addition, usually denoted as EXOR and written as $\oplus$. This group can be represented as the direct product of basic cyclic groups of order 2, $C_2 = (\{0, 1\}, \oplus)$.

The Shannon decision tree (SDT) in Figure 1 illustrates this procedure. This SDT shows the complete disjunctive form of f

$$f = \bar{x}_1\bar{x}_2\bar{x}_3 f_{000} \oplus \bar{x}_1\bar{x}_2 x_3 f_{001} \oplus \bar{x}_1 x_2 x_3 f_{011} \oplus x_1\bar{x}_2\bar{x}_3 f_{100} \oplus x_1\bar{x}_2 x_3 f_{101} \oplus x_1 x_2 x_3 f_{111}. \quad (4)$$

Each node has the label S, since the Shannon expansion was used. The nodes corresponding to the i-th variable form the i-th level in the DD.

3.2. Spectral Interpretation of SDTs

In the matrix notation, (1) can be written as

$$f = [\bar{x}_i \quad x_i]\mathbf{B}_i(1)\begin{bmatrix} f_0 \\ f_1 \end{bmatrix}, \quad (5)$$

where

$$\mathbf{B}_i(1) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Recursive application of the same decomposition rule to all variables in f can be expressed through the Kronecker product.

For $f \in GF_2(C_2^n)$, given by the truth-vector $\mathbf{F} = [f(0), \ldots, f(2^n - 1)]^T$, by the recursive application of (1), we get

$$f = \left(\bigotimes_{i=1}^{n} [\bar{x}_i \quad x_i] \right) \mathbf{B}(n)\mathbf{F}, \quad (6)$$

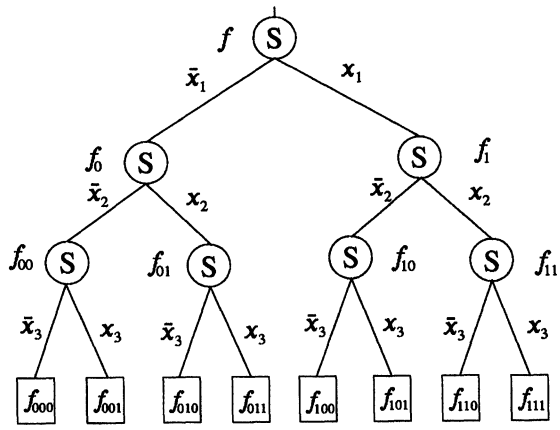

FIGURE 1 Shannon decision tree for $n = 3$.

where $\otimes$ denotes the Kronecker product and

$$\mathbf{B}(n) = \bigotimes_{i=1}^{n} \mathbf{B}_i(1).$$

Relation (6) can be interpreted as a Fourier series-like expansion of f with respect to the so-called trivial basis consisting of functions defined by minterms, *i.e.*, by columns of $\mathbf{B}(n)$. Shannon DT is the graphical representation of the decomposition of f with respect to this basis.

3.3. Davio DTs

If we use (2) recursively to a function f, then each n-variable function f can be represented as

$$f = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus \cdots \oplus a_n x_n \oplus a_{12} x_1 x_2 \oplus a_{13} x_1 x_3 \oplus \cdots \oplus a_{n-1n} x_{n-1} x_n \oplus \cdots \oplus a_{12\ldots n} x_1 x_2 \cdots x_n, \quad (7)$$

which is called the Zhegalkin polynomial [105, 106], or the positive polarity Reed–Muller expression (PPRM) of f [34, 79].

Example 2 If we expand $f(x_1, x_2, x_3)$ by using the pD-expansion with respect to x_1, we have

$$f = f_0 \oplus x_1 f_2.$$

When we expand f_0 and f_2 in x_2, we have

$$f_0 = f_{00} \oplus x_2 f_{02}, \quad f_2 = f_{20} \oplus x_2 f_{22}.$$

Further, when we use the same expansion with respect to x_3, we have

$$f_{00} = f_{000} \oplus x_3 f_{002}, \quad f_{02} = f_{020} \oplus x_3 f_{022},$$
$$f_{20} = f_{200} \oplus x_3 f_{202}, \quad f_{22} = f_{220} \oplus x_3 f_{222}.$$

The expansion tree in Figure 2 illustrates this procedure and, therefore, is called the positive Davio decision tree (pDDT). Thus, it represents the PPRM of f

$$f = f_{000} \oplus x_3 f_{002} \oplus x_2 f_{020} \oplus x_2 x_3 f_{022} \oplus x_1 f_{200} \oplus x_1 x_3 f_{202} \oplus x_1 x_2 f_{220} \oplus x_1 x_2 x_3 f_{222}. \quad (8)$$

Each node has the label pD, which shows that the positive Davio expansion was used in drawing this tree.

The negative Davio decision tree (nDDT) can be defined in the same way by using the relation (3).

3.4. Spectral Interpretation of DDTs

In the matrix notation, (2) can be written as

$$f = [1 \quad x_i]\mathbf{R}_i(1)\begin{bmatrix} f_0 \\ f_1 \end{bmatrix}, \tag{9}$$

where

$$\mathbf{R}_i(1) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}.$$

After recursive application to all the variables in $f \in GF_2(C_2^n)$,

$$f = \left(\bigotimes_{i=1}^{n} [1 \quad x_i] \right) \mathbf{R}(n)\mathbf{F}, \tag{10}$$

where

$$\mathbf{R}(n) = \bigotimes_{i=1}^{n} \mathbf{R}_i(1).$$

The set of coefficients A_f in PPRM, written as a vector $\mathbf{A}_f(n) = [A_f(0), \ldots, A_f(2^n - 1)]^T$, is defined by

$$\mathbf{A}_f(n) = \mathbf{R}(n)\mathbf{F}.$$

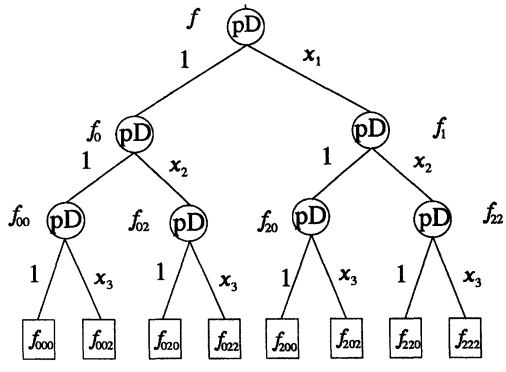

FIGURE 2 Positive Davio decision tree for $n = 3$.

Relation (10) is the matrix representation of PPRM, since $\mathbf{R}(n)$ is the positive polarity Reed–Muller matrix. The columns of this matrix define the Reed–Muller functions representing a basis in $GF_2(C_2^n)$. Therefore, pDDT is graphical representation of decomposition of f with respect to the Reed–Muller basis. In pDDT, each path from the root node up to a constant node corresponds to a Reed–Muller function. The values of constant nodes are the Reed–Muller spectral coefficients of f. The same interpretation extends to nDDTs. The negative literal for x_i requires permutation of columns in $\mathbf{R}_i(1)$ [34, 84].

3.5. Generalizations

3.5.1. Bit-level DTs

In (10), for each variable x_i, $i = 1, \ldots, n$, if we use either a positive literal (x_i) throughout or a negative literal ($\bar{x}_i$) throughout, then we have a fixed polarity Reed–Muller expression (FPRM). FPRMs are represented by fixed polarity Reed–Muller DTs (FPRMDTs) [75]. In FPRMDTs, the same decomposition is used at all the nodes corresponding to the same variable x_i. If either the positive or negative Davio expansion for each node is used, the corresponding tree represents a pseudo Reed–Muller expression (PSDRM) [75].

Example 3 Figure 3 shows PSDRM of a three-variable function where f, f_0, f_{02} and f_{21} use the positive Davio expansions, while f_2, f_{20} and f_{22} use the negative Davio expansions as is shown by the corresponding labels pD and nD of the nodes. Thus, the tree in Figure 3 shows PSDRM

$$\begin{aligned} f = {} & f_{001} \oplus \bar{x}_3 f_{002} \oplus x_2 f_{020} \oplus x_2 x_3 f_{022} \\ & \oplus x_1 f_{210} \oplus x_1 x_3 f_{212} \oplus x_1 \bar{x}_2 f_{221} \\ & \oplus x_1 \bar{x}_2 \bar{x}_3 f_{222}. \end{aligned}$$

Kronecker DTs (KDTs) [16] are defined by freely choosing for each variable among the Shannon, positive Davio, and negative Davio expansion rules. Pseudo–Kronecker DTs (PKDTs) are defined by freely choosing the

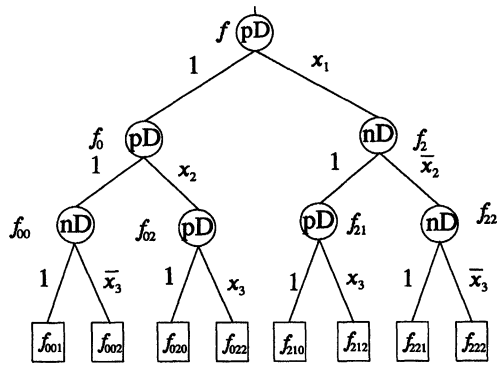

FIGURE 3 PSDRMDT in Example 3.

decomposition rule for each variable irrespective to other nodes in the DT. KDTs and PKDTs represent Kronecker and pseudo–Kronecker expansions for switching functions [75, 79].

DTs mentioned till now are usually denoted as Bit-Level DTs, since the values of constant nodes are the logical values 0 and 1. The matrix notation and spectral interpretation extends as well to these DTs.

Example 4 Denote by $\mathbf{Q}(3)$ a matrix whose columns are the basic functions $\{1, \bar{x}_3, x_2, x_2x_3, x_1, x_1x_3, x_1\bar{x}_2, x_1\bar{x}_2\bar{x}_3\}$ in terms of which PSDRMDT in Figure 3 is defined. If the basic functions are represented as columns of a matrix $\mathbf{Q}(3)$, then the values of constant nodes in this PSDRMDT can be interpreted as spectral coefficients of f calculated with respect to the Q-transform defined by the matrix $\mathbf{Q}^{-1}(3)$ inverse to $\mathbf{Q}(3)$ over $GF(2)$. Thus, in this example

$$\mathbf{Q}(3) = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}.$$

Multiple-output switching functions are represented by Shared BDDs, containing a separate root node for each output [51] and their generalizations [36].

3.6. Further Generalizations of Bit-Level DDs

In bit-level Kronecker DDs, at each node either S, pD, or nD expansion is freely chosen. These expansions can be expressed by (2×2) matrices.

Quaternary DDs (QDDs) [77] are introduced by allowing at each node an expansion described by an arbitrary (4×4) matrix. Thus, QDDs are used to represent functions of four or multiple of two variables.

In [60], a generalization is performed by allowing both (2×2) and $(r \times r)$ matrices, $r < n$, in the same DD. The case when all the matrices are of the same order $r = 4$ is included as a particular example. A restriction assumed in [60] is that columns of used $(r \times r)$ matrices must be described by products of either positive or negative literals of a subset of r switching variables. These DDs are denoted as Generalized Kronecker DDs (GKDDs). The function expressions derived from these DDs involve Kronecker expressions and generalized Reed–Muller expressions as particular cases.

Expressed in terms of variables, these generalizations of AND-EXOR related DDs can be described as follows. In QDDs, expansion rules are performed over four-valued variables X_i recoding pairs of switching variables (x_i, x_j). In [60], an extension to subsets of an arbitrary number of variables is allowed, including the singleton set consisting of a single variable.

In a group theoretic approach to DDs, these methods can be uniformly considered as DDs based on different decompositions of the domain group of order 2^n. In QDDs, all the subgroups are of order four. In Generalized Kronecker DDs, the subgroups are of an arbitrary order. In these and other DDs, as for example, lattice DDs [61], the permitted groups are restricted to Abelian groups.

The use of such decompositions assume nodes with several outgoing edges. A bottleneck is that in many cases, the reduction of the size increases the width of the DD. Extension to DDs on non-Abelian groups permits simultaneous reduction of both size and width of the DDs [88, 90].

Free BDDs [4] are a generalization of BDDs allowing a different order of variables in the paths from the root node to the constant nodes. Different, not quite arbitrary, order of variables is allowed, since the canonicity of the representation should be kept, and the recursive structure of a decision tree should be appreciated. The same way of generalization extends to any DD. For example, we can define Free QDDs, as the Free Generalized Kronecker DDs and other related DDs are described in [61].

DDs defined with respect to arbitrary sets of linearly independent functions over $GF(2)$ are introduced in [58] and further elaborated in a series of papers by this author and his associates ([62] and references given there).

3.6.1. Word-level DTs

Word-level DTs are an extension of DTs for switching functions permitting representations of integer-valued and complex-valued functions [79]. Thus, they extend area of application of DDs representations.

We denote the space of such functions by $C(C_2^n)$ and consider the integer-valued functions as a subset of complex-valued functions. Switching and multiple-valued functions are considered as integer-valued functions. Their logical values are formally identified with corresponding integers. Thus, these functions can be also represented by world-level DDs. In some applications, such representations may be more efficient than bit-level DDs. Further, word-level DDs permit representation of multiple-output switching functions through the integer-valued equivalents f_z. For example, a multiple-output switching function $f_0 * f_1 * \cdots * f_{m-1}$ is uniquely represented by the integer-valued function [41]

$$f_z = \sum_{i=0}^{m-1} 2^i f_i.$$

In logic design and related areas, the main merit of word-level DDs is that for some classes of switching functions they provide more compact representations than bit-level DDs.

Some of word-level DTs are defined by using integer counterparts of the basic functions used in definition of the bit-level DTs. For example, Multi-terminal binary DTs [9] are a generalization of BDTs defined through decomposition of f with respect to the basis represented by $\mathbf{B}(n)$, but with logical values 0 and 1 replaced by the integers 0 and 1, respectively. Binary moment trees (BMTs) [6] are DTs defined in terms of the integer Reed–Muller basis derived in the same way from $\mathbf{R}(n)$. Therefore, the decomposition rule applied at the nodes of BMTs is described by the matrix $\mathbf{A}_i(1) = \left[\begin{smallmatrix} 1 & 0 \\ -1 & 1 \end{smallmatrix}\right]$, which is the inverse of $\mathbf{R}_i(1)$ over the complex field C. This observation shows a way to assign a word-level DD to each bit-level DD, as mentioned above.

In a bit-level DD, the values of basic functions represented by columns of the matrix $\mathbf{Q}$ are interpreted as integers. Then, the matrix $\mathbf{Q}^{-1}$ is calculated as the inverse of $\mathbf{Q}$ over C [91, 94]. The matrix $\mathbf{A}_i(1)$ is the basic transform matrix in the arithmetic representations of switching functions. Arithmetic transform DDs (ACDDs) are defined with respect to the same expansion rule as BMDs, but differ from BMDs in the reduction rules [94].

As in the case of bit-level DDs, different expansion rules produce different DDs. For example, Walsh DTs (WDTs) [94] are defined in terms of the decomposition rule described by the matrix $\mathbf{W}_i(1) = \left[\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right]$, that is derived from $\mathbf{A}_i(1)$ by $(0,1) \rightarrow (1,-1)$ coding of switching variables.

Besides integer counterparts of bit-level DDs, various word-level DDs can be introduced by using different spectral transforms. For example, Complex Hadamard DTs (CHDTs) [25, 27, 67], are defined by using decomposition rules described by the matrix $\mathbf{H}_i(1) = \left[\begin{smallmatrix} 1 & i \\ -i & -1 \end{smallmatrix}\right]$, used in the definition of one of the possible complex Hadamard transforms (CHTs) [26, 28, 65].

Even though the family of CHTs and corresponding DDs were introduced only recently [28, 65], there have been some applications of

CHTs in binary logic design discussed below which justify the usage of complex valued transforms and corresponding spectra formed from Gaussian integers rather than the standard integers. It should be also noticed that when a CHT matrix with half spectrum property is selected [26, 28, 65], then only half of spectral transform coefficients are needed to be calculated and operated on. For such a case, there is no overhead in calculation half spectrum performing Gaussian integers arithmetic *versus* calculation of full spectrum by integer valued arithmetic for standard integer-valued transforms. For example, the CHT with half spectrum property were found very advantageous in classification and synthesis/analysis of logic circuits [29, 67, 68], as well as in some signal processing problems [69].

To perform such a classification, only spectral summation and examination of real and imaginary parts of half of the complex coefficients are necessary. This classification method is more efficient than that based on the Walsh transform, since it does not require any computation once half of the complex Hadamard spectrum and its summation are obtained. The complex spectral coefficients are manipulated according to the lookup flow diagram to identify classes of linearly separable and NPN equivalent functions. Some classes of functions can be identified extremely fast by checking values of only few selected spectral coefficients.

Another important application of CHT developed recently was identification of Boolean symmetries. Detection of the symmetries has been an important issue as its knowledge leads to a more proficient realization of the function. The problem of symmetries is important in many applications, for example in Boolean matching, cell library design, testing, computing arithmetic and related functions using threshold circuits [13]. Different approaches have been tried for the identification of Boolean symmetries. They may be grouped into classical and spectral approaches. Classical approaches usually require decomposition charts, truth-tables, and similar large data structures [13], while spectral approaches are based on arithmetic manipulations on some subsets of spectral coefficients [41, 42, 49]. Since there are efficient methods for calculating the Walsh spectrum of Boolean functions from decision diagrams or disjoint cubes [10, 30], then, the second approach, in general is more viable from the implementation point of view. When CHT is used in identification of symmetries in Boolean functions, only few spectral coefficients are required (in the worst case the half of the spectrum is needed) [66]. These few complex coefficients may be calculated directly from representations of Boolean functions (such as DDs or disjoint cubes), and represented very efficiently with their own Complex Decision DDs. It was shown in [65] that CHT represents a mapping of 4-valued integers into the unit circle of complex plane. As such, it is well suited to processing not only binary, but also MV functions. Analysis in [65] has shown that some CHTs may be considered as systems of complex Walsh functions [99], while others become q-valued Vilenkin–Chrestenson functions for $q=2$ or 4 [41, 99, 104]. It is then obvious that CHTs can be used in different applications of Complex Walsh functions and Vilenkin–Chrestenson functions in processing of MV functions, especially for the case of 4-valued functions. Much work has already been done for Vilenkin–Chrestenson transform, for example characterization of ternary threshold function [53], development of measures of the dependence of MV functions on linear logic functions [54], and disjoint spectral translation that allows extending the possibility of low complexity realization to a large class of MV functions [52]. Similar results can be obtained for these CHTs that are different from Vilenkin–Chrestenson functions. The methods should be computationally more effective as Vilenkin–Chrestenson functions do not possess the half spectrum property.

Various DTs are defined by using different sets of basics functions. Further generalizations are achieved by introducing the additive [45, 102], or multiplicative coefficients [6], or both additive and

multiplicative coefficients [38, 46, 102]. The use of nodes with greater number of outgoing edges [77] permitted extension of DDs representations to MV [49, 50, 86], and other discrete functions on not-necessarily Abelian groups [88]. See [96] for a unified interpretation and classification of various DDs.

A merit of introduction of different DDs is that they may be useful in some applications where other DDs are inconvenient. In Section 4, an example and a brief discussion of that subject is given.

3.7. Spectral Interpretation of DTs

Spectral interpretation shown for BDTs (Subsection 3.2), PRMDTs (Subsection 3.4), and PKDTs (Subsection 3.5.1), and used in Subsection 3.6.1 to introduce examples of some particular DTs, extends to other DTs [12, 17, 86, 94, 95]. In this interpretation, a given function f is assigned to a DT through decomposition with respect to a set Q of basic functions. Each path from the root node to constant nodes corresponds to a basic function that is determined by the product of labels at the edges (Example 4). As noted in this example, if these functions are represented as columns of a matrix $\mathbf{Q}$, then the values of constant nodes are the Q-spectrum S_{Qf} of f defined by

$$\mathbf{S}_{Qf} = \mathbf{Q}^{-1}\mathbf{F}, \tag{11}$$

where $\mathbf{Q}^{-1}$ is the matrix inverse to $\mathbf{Q}$ and $\mathbf{F}$ is the truth-vector of f.

With this interpretation, two basic points on DTs representations of discrete functions read as follows:

1. Given a function f. To represent it by a DT defined in terms of a basis Q, we calculate the Q-spectrum S_{Qf} of f, *i.e.*, we perform the direct Q-transform of f.
2. Given a DT defined in terms of a basis Q. To read f represented by this DT, we perform the inverse Q-transform by starting from the constant nodes in the DT.

Composition of any transform defined by a matrix $\mathbf{Q}$, with the identical transform defined by $\mathbf{B}(n)$ equals $\mathbf{Q}$. Therefore, if we perform the identical transform over DT by starting from the constant nodes, then we read at the root node the Q-spectrum of f. In DTs, that means the nodes corresponding to the Q-transform are formally replaced by the Shannon nodes and, consequently, the labels at the edges are replaced by these used in the Shannon DTs. This consideration permits to derive the following remark illustrated in Figures 4 and 5 for $n=3$.

Remark 1 A DT defined in terms of a basis Q represents f and, at the same time, the Q-spectrum of f, depending on the interpretation of nodes and labels at the edges in the DT.

The following example illustrates the Remark 1. For simplicity, the example is given for $n=2$.

Example 5 Figure 6 shows WDT for two-variable switching function $f(x_1, x_2)$ represented by the truth-vectors $\mathbf{F}=[f(0), f(1), f(2), f(3)]^T$. In this WDT, the values of constant nodes are the Walsh spectral coefficients. The Walsh coefficients repre-

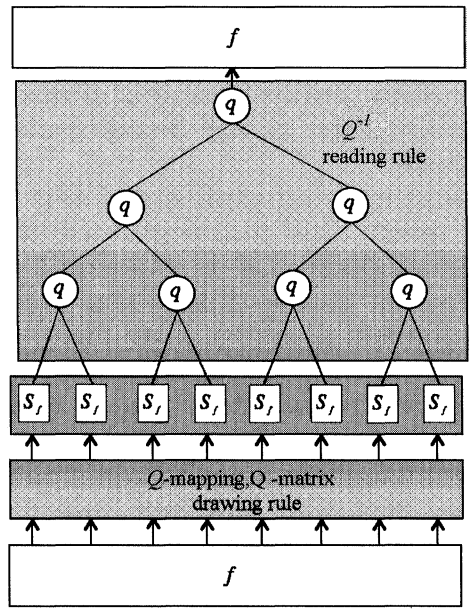

FIGURE 4 Reading f from the DT.

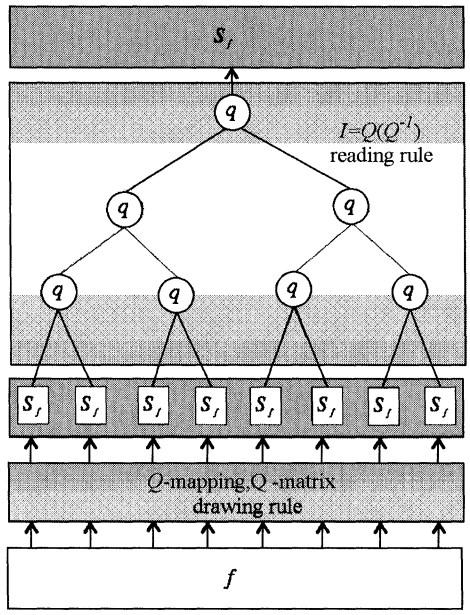

FIGURE 5 Reading S_{Qf} from the DT.

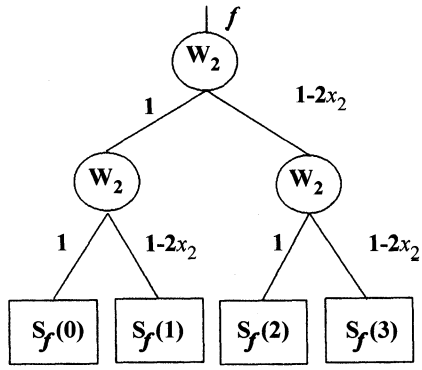

FIGURE 6 WDT for two variable functions.

sented as a vector $\mathbf{S}_f=[S_f(0), S_f(1), S_f(2), S_f(3)]^T$, are calculated as

$$\mathbf{S}_f = \mathbf{WF} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} f(00) \\ f(01) \\ f(10) \\ f(11) \end{bmatrix}$$

$$= \begin{bmatrix} f(00) + f(01) + f(10) + f(11) \\ f(00) - f(01) + f(10) - f(11) \\ f(00) + f(01) - f(10) - f(11) \\ f(00) - f(01) - f(10) + f(11) \end{bmatrix}.$$

In WDDs, the expansion rule used in the nodes is derived from the basic Walsh matrix $\mathbf{W}(1)$

$$f = 1 \cdot f_0 + (1 - 2x_i)f_1.$$

This rule determines the labels at the edges as 1 and $(1-2x_i)$. The basic Walsh matrix is a self-inverse matrix with the constant 1/2. Thus, by following the labels at the edges starting from the constant nodes, which are the Walsh coefficients, we perform the inverse Walsh transform. Thus, we read f at the root node of WDT.

$$\begin{aligned} f =& \frac{1}{4}((1 \cdot S_f(00) + (1 - 2x_2)S_f(01)) \\ &+ (1 - 2x_1)(1 \cdot S_f(10) + (1 - 2x_2)S_f(11))) \\ =& \frac{1}{4}(S_f(00) + (1 - 2x_2)S_f(01) \\ &+ (1 - 2x_1)S_f(10) \\ &+ (1 - 2x_1)(1 - 2x_2)S_f(11)). \end{aligned}$$

Assume that in the nodes of WDT, we perform the rule derived from the inverse Walsh transform. Thus, we perform the identical mapping expressed as composition of mappings $\mathbf{W}^{-1}(1)\ \mathbf{W}(1)=\mathbf{I}(1)$, since the constant nodes are determined by using the rule derived from the direct Walsh transform. In WDT, that means the Walsh nodes are formally replaced by the Shannon nodes. Correspondingly the labels at the edges are replaced by the labels $1 \rightarrow \bar{x}_i$, and $(1-2x_i) \rightarrow x_i$. Thus, we read the Walsh spectrum S_f at the root node in WDT for f

$$S_f = \bar{x}_1\bar{x}_2S_f(0) + \bar{x}_1x_2S_f(1) + x_1\bar{x}_2S_f(2) + x_1x_2S_f(3).$$

4. DECISION DIAGRAMS

Decision diagrams (DDs) are derived by the reduction of DTs. Reduction is performed by deleting or sharing redundant nodes in the DT. Depending on the decomposition rules used at the nodes (choice of the matrices $\mathbf{Q}_i$), reduction is done by using BDD reduction rules [75], *zero-suppressed BDD reduction rules* (ZBDD) [51], or

generalized BDD reduction rules [94]. A DD is reduced (RDD) if further reduction with the same reduction rules is impossible. The DD is ordered if the variables of f assigned to the levels in the DD appear in a fixed order.

A DD is characterized by the number of levels (the depth of the DD), maximal number of non-terminal nodes per level (the width of the DD), and the total number of nodes (the size of the DD).

Complexity of a DD expressed through these parameters determines applicability of the DDs and often is a limiting factor in practical applications within given hardware and software resources. This is at the same time a justification for consideration of many different DDs.

For example, note that representation of n-bit multipliers by BDDs is impossible for a large n within a reasonable node limit of for example 100,000 nodes [15]. However, ACDDs and WDDs are efficient in that case.

Example 6 The output of a 2-bit multiplier can be represented by an integer-valued function f_z determined by summing binary-valued outputs with the coefficients 2^i. Thus, f_z is given by the vector $\mathbf{F}_z = [0,0,0,4,0,0,2,6,0,2,0,6,1,3,3,9]^T$.

In the matrix notation, the arithmetic spectrum A_f for f_z is calculated as

$$\mathbf{A}_z = \mathbf{A}(2)\mathbf{F}_z, \quad \mathbf{A}(2) = \bigotimes_{i=1}^{2} \mathbf{A}_i(1).$$

Thus, $\mathbf{A}_z = [0,0,0,4,0,0,2,0,0,2,0,0,1,0,0,0]^T$. Figure 7 shows MTBDD and Figure 8 ACDD for 2-bit multiplier. In this example, MTBDD and ACDD require 20 and 15 nodes, respectively.

The number of nodes to represent an n-bit multiplier approximates to $O(4^n)$ for MTBDDs, and $O(n^2)$ for ACDDs and WDDs [94].

Some attempts to solve the problem of size reduction are done by introducing DDs with attributed edges. In particular, further reduction of the size of DDs for multipliers is possible by using DDs with attributed edges as for example, $*$BMDs [6] and K$*$BMD [15].

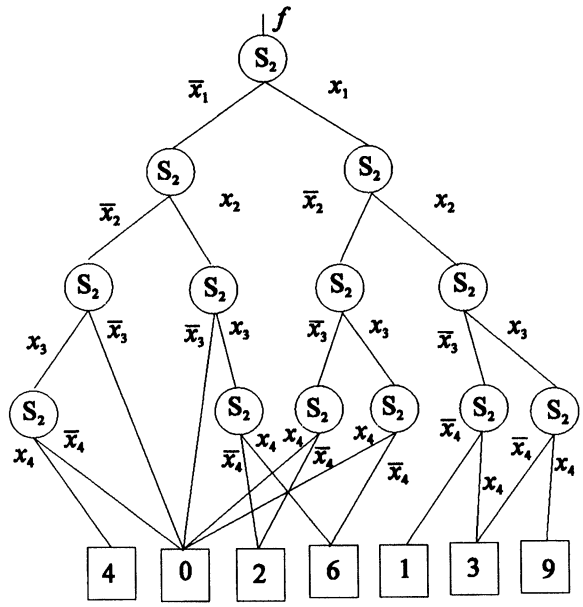

FIGURE 7 MTBDD for 2-bit multiplier.

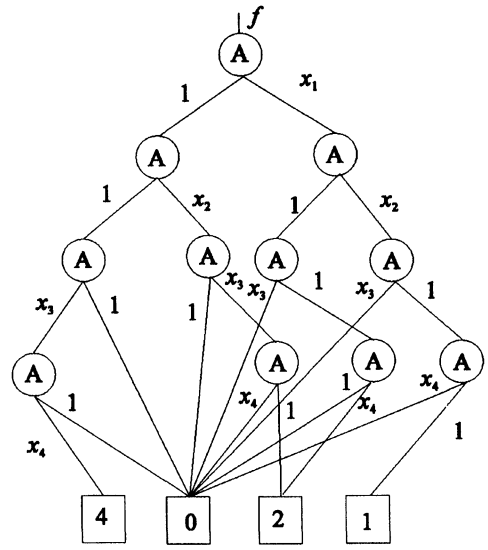

FIGURE 8 ACDD for 2-bit multiplier.

A BMT for f is reduced into BMD for f if there are arithmetic spectral coefficients for f of equal values, since they are values of constant nodes in the BMT. $*$BMTs are a generalization of the concept by using identical factors in the values of arithmetic spectral coefficients for f to achieve

more compact DDs. The concept will be explained by the following example.

Example 7 Arithmetic spectrum for f given by the vector $\mathbf{F}=[8,-12,10,-6,20,24,37,45]^T$ is $\mathbf{A}_f=[8,-20,2,4,12,24,15,0]^T$. These values can be factored as

$$2\cdot 2\cdot 2, 2\cdot 2\cdot(-5), 2\cdot 1\cdot 1, 2\cdot 1\cdot 2, 3\cdot 4\cdot 1,$$
$$3\cdot 4\cdot 2, 3\cdot 1\cdot 5, 3\cdot 1\cdot 0.$$

To produce an $*$BMD for f the first and second factors are moved to the first and the second level in the BMD. The third factors are kept as the values of constant nodes. The final $*$BMD is shown in Figure 9.

$*$BMDs are DDs with multiplicative attributes assigned to the edges. The decomposition rule applied to the nodes in $*$BMDs are defined by $w_i\mathbf{A}_i(1)$, where w_i is the multiplicative weight coefficient assigned to the nodes at the i-th level in the $*$BMD for f.

Edge-valued binary DDs (EVBDDs) [45] are also DDs related to the arithmetic transform [87]. In EVBDDs, the additive weight coefficients are used.

K$*$BMDs are a generalization of $*$BMDs, where the role of the arithmetic transform is replaced by the various Kronecker transforms [15]. In K$*$BMDs, both additive and multiplicative weight coefficients at the edges are used. The same is done in Factored EVBDDs (FEVBDDs) [45]. We refer to [17], and [91] for further details and examples of these DDs.

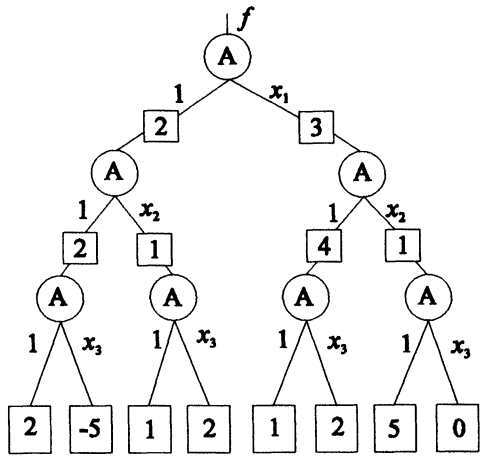

FIGURE 9 $*$BMT for f in Example 7.

It is stated in [15] that the size of a K$*$BMD for n-bit multipliers is considerably reduced and approximates to $O(2n)$. However, in the analysis of complexity of DDs, with attributed edges, it is needed to take into account the amount of other related information that should be stored to uniquely describe a DD, for example, both nodes and attributes to the edges should be stored. In DDs with different decomposition rules at the nodes, another problem is determination of optimal combination of the decomposition rules for the nodes.

In this setting, Fourier DDs on non-Abelian groups (FNADDs) can be interesting [88, 90]. They use the same decomposition rule for all the nodes. The decomposition rule is derived from the Fourier transform on groups and advantages are taken from the properties of group representations of non-Abelian groups. The same as in the case of other DDs with increased number of outgoing edges, FNADDs permit reduction of levels in the DD, compared to DDs with two outgoing edges. In DDs on Abelian groups, reduction of the number of levels by using nodes with several outgoing edges, as for example in QDDs, and GKDDs, often increases the width of the DD. By taking the advantages in the properties of Fourier transform on non-Abelian groups, FNADDs permit reduction of both width and size of the DD [88, 90]. Table I compares sizes and widths of Shared BDDs [51] and FNADDs [93] for some

TABLE I SBDDs and FNADDs for benchmark functions

	SBDD		FNADD				
f	Size	Width	ntn	cv	Size	Width	Group
5xp1	90	25	39	128	167	18	$C_2Q_2^2$
bw	116	37	9	25	34	4	C_4Q_2
con1	20	5	13	12	25	6	$C_2Q_2^2$
rd53	25	6	7	14	21	3	C_4Q_2
rd73	45	10	23	30	53	6	$W_2Q_2^2$
xor5	11	2	5	6	11	2	C_4Q_2

benchmark functions. For FNADDs, the number of non-terminal nodes and constant nodes are given separately. We also show the decomposition of the domain group for f, since the depth of a FNADD is equal to the number of subgroups.

In all DDs, the size strongly depends on the order of variables in the represented function f. That means the size of a DD depends on the assignment of nodes per levels in the DD. Different order of variables in f produces DDs of different sizes. Thus, in all DDs, irrespective to the used sets of basic functions, the optimization by changing the variables ordering may be performed.

Free BDDs are a generalization of the optimization of ordered DDs by variables ordering. The change in variables ordering can be described as the application of the permutation matrices to values of constant nodes in the decision trees [91]. Thus, we perform permutation over truth-vectors for functions or some spectra derived by application of some transform matrices to the truth-vectors. Therefore, in optimization of DDs by variables ordering, we have composition of at least two mappings, described by successive application of the spectral transform matrix and the permutation matrix. In Free DDs, the set of allowed permutation matrices is extended when compared to that used in optimization of DDs by variables ordering [91]. In some Free DDs, several permutation matrices may be successively applied to get the smallest size DD.

In QDDs, another way of optimization of DDs by manipulation with variables ordering is performed. It consists in variables pairing during determination of pairs of recoded variables. It can be described by permutation matrices. However, the permutation matrices are now applied not to the function values or the related spectra, but to the variables. If we express that in terms of permutation matrices applied to function values, that would be a subset of permutation matrices allowed in Free QDDs. See [91] for more details on the relationships among permutation matrices and DDs.

A problem is that we do not know how to predict the order of variables for a given f to get the corresponding DD of the smallest size. In DDs based on the decomposition of the domain group into the product of subgroups of arbitrary orders, the solution of that problem is easy [80]. The subgroups of larger orders should have the variables at the bottom of the DD. If some subgroups are non-Abelian, they also should have variables at the bottom levels in the DD. However, if the subgroups are all of the same order, we recognize that as the variables ordering problem in DDs.

5. DDs BASED LOGIC DESIGN

Design of architectures for realization of f derived from DDs representations of f was studied by several authors [48, 71, 79, 103] and discussions and references given there. These realization architectures are based upon the multiplexers, Reed–Muller modules or suitable FPGAs [81, 77]. Some recent results in that area are given in [37]. These methods can be easy explained by recalling the spectral interpretation of DTs.

The Remark 1 extends to DDs, since the reduction rules are formulated in such a way that they do not reduce the information content in a DD. The impact of deleted nodes can be simply expressed and taken into account through the *cross points* [94].

DEFINITION 1 The cross point in a RDD is a point where a branch longer than 1 crosses a level in DD[2].

With this definition, DDs based design methods can be shortly expressed by the following two rules. It is assumed that an n-variable function f is

[2]A level in the DD consists of the nodes corresponding to a particular variable $x_i, i=1,\ldots,n$. A branch connecting a node at $(i-1)$-th level with a node at $(i+1)$-th level is of the length 2, Note that taking of the cross points into account does not extends the RDD into a complete tree.

represented by a RDD defined with respect to a basis Q. Thus, we use at the nodes of the DD the decompositions described by the sub-matrices $\mathbf{Q}_i(1)$. It is also assumed that the modules that may realize operations defined by $\mathbf{Q}_i(1)$ and $\mathbf{Q}_i^{-1}(1)$ are provided.

Design from DDs:

1. To realize f attach a corresponding Q^{-1}-module to each node and to each cross point in the quasi-reduced QDD.
2. To calculate Q-spectrum of f, attach a multiplexer to each node in the quasi-reduced QDD.

Note It is clear that the Shannon modules, *i.e.*, 1 variable control multiplexers, attached to the cross points corresponding to the deleted Shannon nodes can be also deleted. The same applies to the Reed–Muller modules which have both inputs connected to the constant 0. This is possible, since whatever constant 0 or 1 at both inputs in such a multiplexer produce the same constant at the output. The same is for the constant 0 at both inputs in a two-input Reed–Muller module.

To do the optimization, a quasi-reduced QDD attached to different AND-EXOR expression can be considered and the expression consisting of the smallest number of nodes can be chosen. The procedure can be applied to DDs for different orders of the variables.

Example 8 Figure 10 shows the reduced ordered BDD of the function $f(x_1, x_2, x_3, x_4)$ given by the truth-vector $\mathbf{F}=[1001101001011100]^T$ considered in [77]. The multiplexer-based realization architecture for this function is shown in Figure 11. The procedure produces the network equal to that shown in [77].

Example 9 The realization of the Reed–Muller spectrum S_f of f can be done by the network shown in Figure 12 derived by attaching the Reed–Muller modules to the nodes and cross points in the ROBDD.

Note that, if switching variables and their complements can be used directly as the data

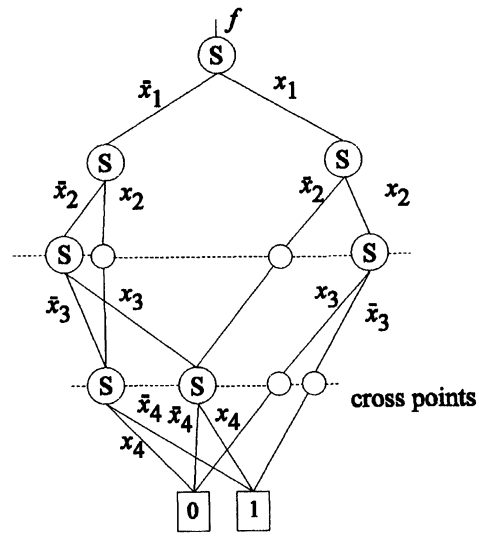

FIGURE 10 ROBDD of f from Example 8.

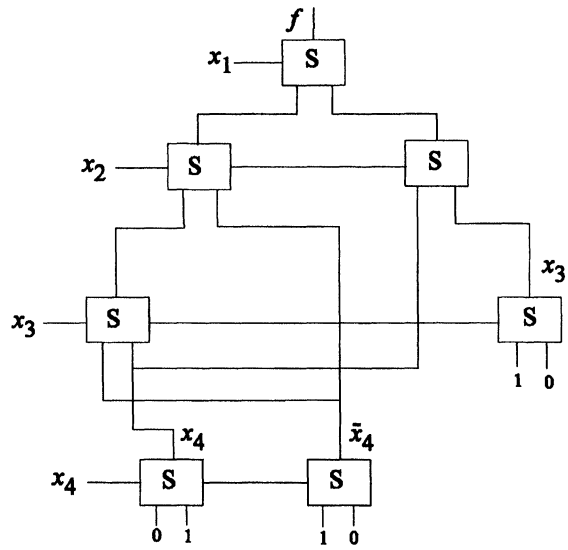

FIGURE 11 Realization of f from Example 8.

inputs of a module, some considerable savings can be achieved, since the modules having constant inputs can be deleted as shown in Figures 10 and 11 at the outputs of these modules.

Main disadvantage of the presented DDs based design methods is due to the propagation delay, since the depth of the produced circuit is equal to the number of variables. Therefore, the design methods for small depth circuits are proposed for

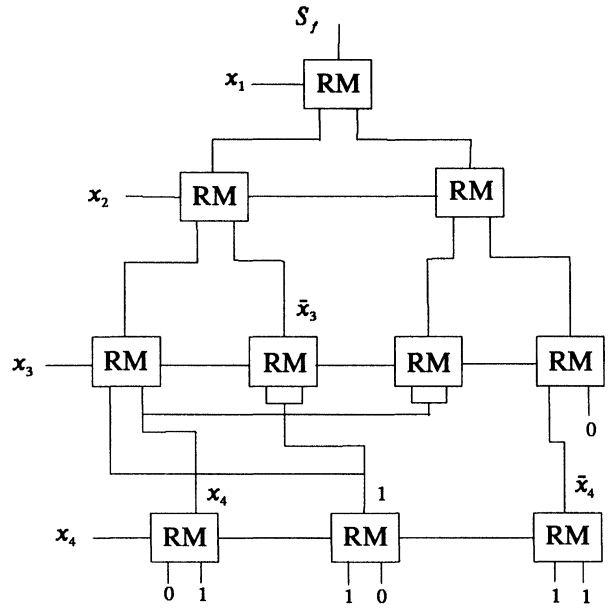

FIGURE 12 Realization of S_f of f from Example 8.

BDDs [39], and KDDs [38]. Some other solutions use the functional decomposition [7, 74].

Design of circuits form DDs is efficiently used in FPGA synthesis [15, 77, 81]. Such realizations often express high testability properties [3, 14].

Lattice DDs [63] are a generalization of BDDs adapted to the synthesis with regular layout networks [61]. Such networks usually consists of identical blocks with connections of the equal length between blocks. The regular structure of a lattice is expressed through the regular structure of a lattice DD, which can be simply transferred into the regular layout network. Extensions to ternary and quaternary lattice DDs are presented in [63]. Their application to the synthesis of fuzzy logic and analog circuit design is also discussed in [63].

6. CALCULATION OF SPECTRAL TRANSFORMS OVER DDS

BDDs can be efficiently used for calculation of spectral transforms on finite dyadic groups [8–10, 19–22, 24, 47, 65, 92]. The method is derived first for Kronecker product representable transform matrices. In that case, it consists in realization of FFT-like algorithms based on the Good-Thomas factorization [35], however, performed over BDD of f instead by using the fast flow-graph of FFT-like algorithm. For a given spectral transform, calculations in a FFT-like algorithm are performed over a fast-flow graph of the same regular structure for all functions of a fixed number of variables. Therefore, complexity of the algorithm is fixed and does not depend on properties of the data function. For example, for n-variable switching functions the time complexity of FFT-like spectral transform methods approximates to $O(n2^n)$, while the space complexity is $O(2^n)$ assuming that in-place computations are used [8, 57, 92].

The BDDs based procedures perform basic "butterfly" operations of the corresponding FFT-like algorithm at non-terminal nodes and cross points in the BDD of f [92]. This way, when implementing FFT over BDDs some particular properties of f are exploited which permits

reduction of nodes in the BDD of f. Hence savings both in time and space complexity are achieved, since

1. calculation is transferred into vector operations at the nodes of the BDD,
2. repeated calculations over identical parts of the truth-vector of f are avoided, since these parts are removed from the BDD.

Example 10 Figure 13 shows BDD for a switching function $f(x_1, x_2, x_3)$ given by the truth-vector $\mathbf{F} = [1,0,0,1,0,1,1,1]^T$. To calculate the Walsh spectrum for f, we perform at each node and the cross point in this BDD, the operations described by the basic Walsh transform matrix $\mathbf{W}(1)$. In the matrix notation, the calculation procedure can be described as follows.

The constant nodes are processed first by performing the matrix $\mathbf{W}(1)$ at the nodes and cross points at the level corresponding to x_3. Therefore,

$$\mathbf{W}_{S_{3,0}} = \begin{bmatrix} 1+0 \\ 1-0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

$$\mathbf{W}_{S_{3,1}} = \begin{bmatrix} 0+1 \\ 0-1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix},$$

$$\mathbf{W}_{C_{3,0}} = \begin{bmatrix} 1+1 \\ 1-1 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}.$$

Performing $\mathbf{W}(1)$ at the nodes at the level corresponding to x_2, *i.e.*, over the subvectors

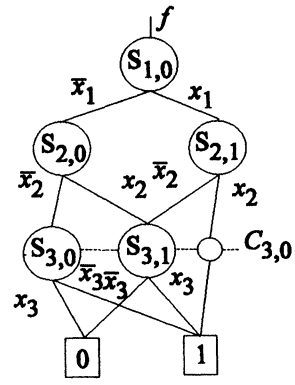

FIGURE 13 BDD for f in Example 10.

where points the outgoing edges of these nodes, we get

$$\mathbf{W}_{S_{2,0}} = \begin{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \end{bmatrix} \\ \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 \\ -1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \\ 2 \end{bmatrix},$$

$$\mathbf{W}_{S_{2,1}} = \begin{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 2 \\ 0 \end{bmatrix} \\ \begin{bmatrix} 1 \\ -1 \end{bmatrix} - \begin{bmatrix} 2 \\ 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 3 \\ -1 \\ -1 \\ -1 \end{bmatrix}.$$

Performing $\mathbf{W}(1)$ at the root node we get the Walsh spectrum of f as follows.

$$\mathbf{S}_f(w) = \mathbf{W}_{S_{1,0}} = \begin{bmatrix} \begin{bmatrix} 2 \\ 0 \\ 0 \\ 2 \end{bmatrix} + \begin{bmatrix} 3 \\ -1 \\ -1 \\ -1 \end{bmatrix} \\ \begin{bmatrix} 2 \\ 0 \\ 0 \\ 2 \end{bmatrix} - \begin{bmatrix} 3 \\ -1 \\ -1 \\ -1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 5 \\ -1 \\ -1 \\ 1 \\ -1 \\ 1 \\ 1 \\ 3 \end{bmatrix}.$$

As shown in Figure 14, the procedure generates the MTBDD for the Walsh spectrum of f.

Thus formulated procedures permit processing of large functions. Tables II and III, taken from [11], show CPU-times for the Reed–Muller and Walsh transform of some benchmark functions calculated through BDDs. Calculations are performed over DEC-5000 workstation. It should be noted that DDs based spectral transform calculation procedures for functions with a number of variables satisfactory large for many applications, may be performed with acceptable efficiency by using simple, easily accessible, hardware, such as Programmable Logic Devices or different FPGAs. At the same time, the DDs based methods may be extended to other spectral transforms with recursive, but not entirely Kronecker product representable matrices [8, 18, 21, 22, 83].

In some cases, further savings may be achieved by exploiting particular properties of a transform.

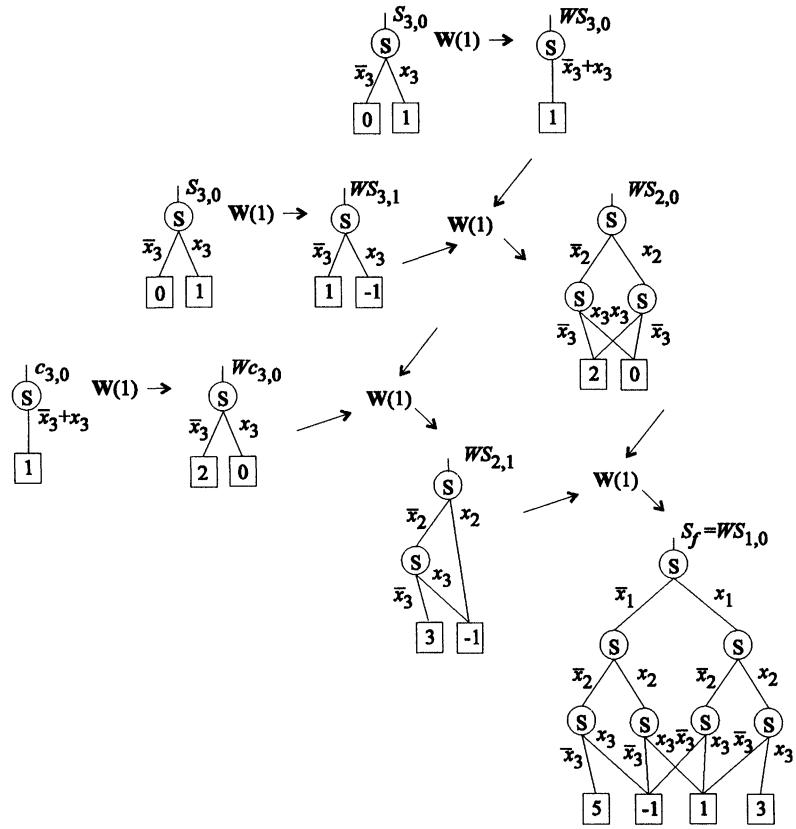

FIGURE 14 Generation of the MTBDD for the Walsh spectrum of f in Example 10.

TABLE II CPU-times for the Reed–Muller transform

f	BDD of f	MTBDD of S_f	Time [sec]
c1908	3607	27748	184
c3540	520	4679	8.2
c5315	1397	2647	25
add50	2307	249	2.3
add100	301	499	11

TABLE III CPU-times for the Walsh transform

f	BDD of f	MTBDD of S_f	Time [sec]
c1908	3607	1850	44
c3540	520	15985	171
c5315	1397	7069	328
add50	2307	7456	23
add100	301	29906	128

For example, the Haar transform is a local discrete transform, and thanks to that, the calculation of the Haar spectrum of f reduces to the processing of the first elements in the subvectors represented by the nodes of BDD of f [83]. Therefore, the calculation complexity of this procedure is directly proportional to the size of the BDD of f. Thus, in Table IV showing the calculation times for the Haar spectrum of some benchmark functions, the number of nodes for the Haar spectrum is not given. In this respect, the

TABLE IV CPU-times for the Haar transform

f	In	Out	BDD of f	Time [10^{-3} sec]
alu4	14	8	1352	2.25
duke2	22	29	976	1.44
misex2	25	18	140	0.18
vg2	25	8	1059	1.58
sao2	10	4	154	0.21
e64	65	65	1446	2.05

procedure corresponds to the in-place computations in FFT. Thanks to the linearity of the Haar transform, the calculations may be performed over shared BDDs. The method assumes that integer-valued function values are represented by binary sequences. Thus, the integer-valued function is transferred into a multiple-output switching function and represented by a shared BDD. The nodes in this shared BDD are processed in the same way as BDDs for single output function are processed in calculation of the Haar transform. The sum of outgoing Haar spectra for separated outputs with coefficients 2^i, $1 < i < n$, are the required Haar spectrum.

This way some other savings may be achieved resulting in a very fast procedure. Calculations reported in Table III have been done without any optimization of shared BDDs by variable ordering. Calculations are performed on a 133 MHz Pentium PC with 32Mbytes of RAM. Time to build up the BDD of f is not counted.

In a similar way, the spectral transforms for switching functions can be calculated over Edge-valued DDs [45], and their generalizations [46, 102]. Through Multiple-Place DDs [82] and their integer-valued counterparts, DDs based calculation methods can be extended to spectral transforms for MV and complex-valued functions on any finite, not necessarily Abelian group [49, 97], and other related algebraic structures [98].

7. CONCLUSION

This paper presented a systematic approach to different decision diagrams used in many applications of Computer Science and Engineering. Spectral interpretation of arbitrary Decision Trees was also shown. Different methods to realize logical circuits directly from Decision Diagrams were discussed. Finally, advantages of calculating spectral transforms over Decision Diagrams rather than with the usage of FFT-like algorithms have been underlined and experimental data for the calculating three main discrete transforms used in logic design have been given. Presented classification and results are important not only for applications in logic design but also everywhere when processing of large discrete functions and matrices is necessary [11, 73, 79]. Some of such applications are in functional verification [6, 19, 46], simulation [36], functional decomposition [7, 73], and technology mapping [43, 73, 77, 79], integer programming [45, 73], and artificial intelligence [79].

Acknowledgment

We thank the referees for many valuable comments and suggestions which were very useful in improving presentation in this paper.

References

[1] Ackers, S. B., "Binary decision diagrams", *IEEE Trans. on Computers*, June, 1978, **C-27**(6), 509–516.

[2] Aizenberg, N. N. and Trofimljuk, O. T. (1981). "Conjunctive transforms for discrete signals and their applications of tests and the detection of monotone functions", *Kibernetika*, No. 1, K, 128–139 (in Russian).

[3] Becker, B. and Drechsler, R., "Synthesis for testability: Circuits derived from Ordered Kronecker functional decision diagrams", *Proc. European Design and Test Conference*, March, 1995, p. 592.

[4] Bern, J., Gergov, C. J., Meinel, C. and Svobodova, A. (1994). "Boolean manipulation with Free BDDs first experimental results", *Proc. European Design and Test Conference'94*, pp. 200–207.

[5] Bryant, R. E. (1986). "Graph-based algorithms for Boolean functions manipulation", *IEEE Trans. Computers*, **C-35**(8), 667–691.

[6] Bryant, R. E. and Chen, Y.-A., "*Verification of arithmetic functions with binary moment decision diagrams*", May, 1994, CMU-CS-94-160.

[7] Bullmann, J. and Kebschull, U., "Multiple-domain logic synthesis", In: Sasao, T. and Fujita, M., *Representations of Discrete Functions*, Kluwer Academic Publishers, 1996, pp. 211–232.

[8] Chang, C. H. and Falkowski, B. J., "In-place transformation of Generalized Complex Hadamard spectra through algebraic decision diagrams", *Proc. IEEE Int. Conf. on Information, Communications and Signal Processing* (1st ICICs), Singapore, September, 1997, **1**, 256–260.

[9] Clarke, E. M., Zhao, X., Fujita, M., Matsunaga, Y. and McGeer, R., "Fast Walsh transform computation with Binary Decision Diagram", *Proc. IFIP WG 10.5 Workshop on Applications of the Reed–Muller Expansion in Circuit Design*, September, 1993, Hamburg, Germany, pp. 82–85.

[10] Clarke, E. M., McMillan, K. L., Zhao, X., Fujita, M. and Yang, J., "Spectral transforms for large Boolean functions with applications to technology mapping", *Proc. 30th ACM/IEEE Design Automation Conference*, Dallas, Texas, June, 1993, pp. 54–60.

[11] Clarke, E. M., Fujita, M. and Zhao, X., "*Multi-terminal decision diagrams and hybrid decision diagrams*", In: [79], pp. 93–108.

[12] Clarke, E. M., Fujita, M. and Weinle, W., "Hybrid Spectral Transform Diagrams", *Proc. IEEE Int. Conf. on Information, Communications and Signal Processing*, (1st ICICS), Singapore, September, 1997, **1**, 251–255.

[13] De Michelli, G., *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, New York, 1994.

[14] Drechsler, R. and Becker, B., "Rapid prototyping of fully testable multi-level AND/EXOR networks", *IFIP WG 10.5 Workshop on Application of the Reed–Muller Expansion in Circuit Design*, September, 1993, pp. 126–133.

[15] Drechsler, R. and Becker, B., "*OKFDDs – algorithms applications and extensions*", in [79], pp. 163–190.

[16] Drechsler, R., Sarabi, A., Theobald, M., Becker, B. and Perkowski, M. A. (1994). "Efficient representations and manipulation of switching functions based on Ordered Kronecker Functional Decision Diagrams", *Proc. Design Automation Conf.*, pp. 415–419.

[17] Drechsler, R., Stanković, R. S. and Sasao, T., "Spectral transforms and Word-level decision diagrams", *Proc. 6th Workshop on Synthesis and System Integration of Mixed Technologies, SASIMI-97*, Osaka, Japan, December, 1997, pp. 39–44.

[18] Falkowski, B. J., "Recursive relationships, fast transforms, generalizations, and VLSI iterative architecture for Gray code ordered Walsh functions", *IEE Proc. Computers, and Digital Devices*, September, 1995, **142**(5), 325–331.

[19] Falkowski, B. J. and Chang, C. H., "Efficient algorithms for the calculation of Walsh spectrum from OBDD and synthesis of OBDD from Walsh spectrum for incompletely specified Boolean functions", *Proc. 37th Midwest Symp. on Circuits and Systems*, Lafayette, Louisiana, USA, August, 1994, pp. 393–396.

[20] Falkowski, B. J. and Chang, C. H., "Efficient algorithm for the calculation of arithmetic spectrum from OBDD and synthesis of OBDD from arithmetic spectrum for incompletely specified Boolean functions", *Proc. 27th IEEE Int. Symp. on Circuits and Systems ISCAS94*, London, United Kingdom, May, 1994, **1**, 197–200.

[21] Falkowski, B. J. and Chang, C. H., "Efficient algorithm for forward and inverse transformations between Haar spectrum and Binary Decision Diagrams", *Proc. 13th Int. Phoenix Conf. on Computers and Communication*, Phoenix, Arizona, USA, April, 1994, pp. 497–503.

[22] Falkowski, B. J. and Chang, C. H., "Forward and inverse transformations between Haar spectra and ordered binary decision diagrams of Boolean functions", *IEEE Transactions on Computers*, November, 1997, **46**(11), 1272–1279.

[23] Falkowski, B. J. and Chang, C. H. (1998). "Mutual conversions between generalized Arithmetic expressions and Free binary decision diagrams", *IEE Proc. Circuits, Devices and Systems*, UK, **145**(4), 219–228.

[24] Falkowski, B. J. and Chang, C. H., "Calculation of arithmetic spectra from Free binary decision diagrams", *Proc. IEEE Int. Symp. on Circuits and Systems* (30th ISCAS), Hong Kong, June, 1997, pp. 1764–1767.

[25] Falkowski, B. J. and Rahardja, S., "Complex Decision Diagrams", *IEE Electronics Letters*, United Kingdom, February, 1996, **32**(4), 290–291.

[26] Falkowski, B. J. and Rahardja, S., "Composite complex Hadamard spectra of Boolean functions", *Proc. 29th IEEE Int. Symp. on Circuits and Systems*, Atlanta, Georgia, USA, May, 1996, **4**, 392–395.

[27] Falkowski, B. J. and Rahardja, S., "Complex spectral decision diagrams", *Proc. 26th IEEE Int. Symp. on Multiple-Valued Logic*, 29–31 May, 1996, Santiago de Campostela, Spain, pp. 255–260.

[28] Falkowski, B. J. and Rahardja, S., "Properties and applications of Unified Complex Hadamard transforms", *Proc. 27th IEEE Int. Symp. on Multiple-Valued Logic*, Antigonish, Nova Scotia, Canada, May, 1997, pp. 131–136.

[29] Falkowski, B. J. and Rahardja, S., "Complex spectral transforms for classification of switching functions", *Proc. 7th Int. Symp. on IC Technology, Systems and Applications*, Singapore, September, 1997, pp. 505–508.

[30] Falkowski, B. J., Schaefer, I. and Perkowski, M. A. (1992). "Effective computer methods for the calculation of Rademacher–Walsh spectrum for completely and incompletely specified Boolean functions", *IEEE Trans. on CAD*, **11**(10), 1207–1226.

[31] Falkowski, B. J. and Stanković, R. S., "Decision diagrams for representations of discrete functions in VLSI computer-aided design systems", invited paper, *Proc. of Int. Symp. on IC Technology, Systems and Applications* (7th ISIC), Singapore, September, 1997, pp. 86–93.

[32] Garaev, M. U. and Faradzhev, R. G. (1965). "On an analog of Fourier expansion over Galois fields and its applications to problem of generalized sequential machines", *Izv. Akad. Nauk Azerb. SSR*, Ser. Fiz. -Techn. i Math. Nauk, No. 6, pp. 1–68 (in Russian).

[33] Gibbs, J. E. (1977). "Instant Fourier transform", *IEE Electronics Letters*, **13**(5), 122–123.

[34] Green, D. (1991). "Families of Reed–Muller canonical forms", *Int. J. of Electronics*, **70-2**, 259–280.

[35] Good, I. J. (1958). "The interaction algorithm and practical Fourier analysis", *J. Roy. Statist. Soc.*, ser. B, **20**, 361–372; Addendum, 1960, **22**, 372–375.

[36] Hasan Babu, H. Md. and Sasao, T., "A method to represent multiple-output switching functions by using binary decision diagrams", *Proc. 6th Workshop on Synthesis and System Integration of Mixed Technologies, SASIMI'96*, November, 1996, pp. 212–217.

[37] Hasan Babu, H. Md. and Sasao, T., "Design of multiple-output networks using time domain multiplexing and shared multi-terminal multiple-valued decision diagrams", *Proc. 28th Int. Symp. on Multiple-Valued Logic*, Fukuoka, Japan, May, 1998.

[38] Hengster, H., Drechsler, R., Eckrich, S., Pfeiffer, T. and Becker, B. (1996). "AND/EXOR based synthesis of testable KFDD-circuits with small depth", *Proc. of the 5th Asian Test Symp.*, pp. 148–154.

[39] Ishiura, N., "Synthesis of multi-level logic circuits from binary decision diagrams", *Proc. 6th Workshop on Synthesis and System Integration of Mixed Technologies, SASIMI'92*, pp. 74–83.

[40] Karpovsky, M. G. and Moskalev, É. S., *Spectral Methods in Analysis and Synthesis of Discrete Devices*, Energiya, Leningrad, 1973 (in Russian).

[41] Karpovsky, M. G., *Finite Orthogonal Series in the Design of Digital Devices*, Wiley and JUP, New York and Jerusalem, 1976.

[42] Karpovsky, M. G. (Ed.), *Spectral Techniques and Fault Detection*, Academic Press, Orlando, 1985.

[43] Kozlowski, T., Dagless, E. L. and Saul, J. M., "Unified decision diagrams, a representation for mixed AND-OR/EXOR combinational networks", *Proc. IFIP WG10.5 Workshop on Applications of the Reed–Muller Expansions in Circuit Design, Reed–Muller'95*, August, 1995, Makuhari, Chiba, Japan, pp. 177–184.

[44] Kukharev, G. A., Shmerko, V. P. and Yanushkievich, S. N., *Technique of Binary Data Parallel Processing for VLSI*, Vyshaja shcola, Minsk, 1991, Belarus.

[45] Lai, Y. F., Pedram, M. and Vrudhula, S. B. K. (1994). "EVBDD-based algorithms for integer linear programming, spectral transformation, and functional decomposition", *IEEE Trans. on CAD*, **13**(8), 959–975.

[46] Lai, Y.-T. and Sastry, S. (1992). "Edge-valued binary decision diagrams for multi-level hierarchical verification", *Proc. of 29th Des. Automation Conf.*, pp. 608–613.

[47] Malyugin, V. D., Stanković, R. S. and Stanković, M., "Calculations of the coefficients of polynomial representations of switching functions through binary decision diagrams", *Proc. Preventive Engineering and Information Technologies*, Niš, Yugoslavia, December, 1994, pp. 10.1–10.4.

[48] McKenzie, L., Xu, L. and Almaini, A., "Graphical representations of generalized Reed–Muller Expansions", In: Kebschull, U., Schubert, E. and Rosenstiel, W. Eds., *Proc. IFIP WG 10.5 Workshop on Applications of the Reed–Muller Expansion in Circuit Design*, September, 1993, Hamburg, Germany, pp. 181–187.

[49] Miller, D. M., "Spectral transformation of multiple-valued decision diagrams", *Proc. 24th Int. Symp. on Multiple-Valued Logic*, Boston, USA, May, 1994, pp. 89–96.

[50] Miller, D. M. and Muranaka, N., "Multiple-valued decision diagrams with symmetric variable nodes", *Proc. 26th Int. Symp. on Multiple-Valued Logic*, May, 1996, Santiago de Campostela, Spain, pp. 242–247.

[51] Minato, S., "Graph-based representations of discrete functions", In: Sasao, T. and Fujita, M. (Eds.), *Representations of Discrete Functions*, Kluwer Academic Publishers, 1996, pp. 1–28.

[52] Moraga, C. (1978). "Introducing disjoint spectral translation in spectral multiple-valued logic design", *IEE Electronics Letters*, **14**(8), 241–243.

[53] Moraga, C. (1979). "Characterization of ternary threshold functions using a partial spectrum", *IEE Electronics Letters*, **15**(24), 803–805.

[54] Moraga, C. (1983). "Parameter spectrum in spectral multiple-valued logic design", *IEE Electronics Letters*, **19**(6), 199–200.

[55] Muller, D. E. (1954). "Application of Boolean algebra to switching circuits design and to error detection", *IRE Trans. Electron. Comp.*, **EC-3**, 6–12.

[56] Muzio, J. C. and Wesselkamper, T. C., *Multiple-Valued Switching Theory*, Adam Hilger, Bristol, 1986.

[57] Nussbaumer, H. J., *Fast Fourier Transform and Convolution Algorithms*, Springer-Verlag, Berlin, 1981.

[58] Perkowski, M. A., "The generalized orthonormal expansions of functions with multiple-valued input and some of its applications", *Proc. Int. Symp. on Multiple-Valued Logic*, May, 1992, Sendai, Japan, pp. 442–450.

[59] Perkowski, M. A., "Fundamental theorem for EXOR circuits", In: *Proc. IFIP WG 10.5 Workshop on Applications of the Reed–Muller Expansion in Circuit Design*, September, 1993, Hamburg, Germany, pp. 52–60.

[60] Perkowski, M. A., Jozwiak, L. and Drechsler, R., "A canonical AND/EXOR form that includes both the generalized Reed–Muller forms and Kronecker forms", *Proc. IFIP WG 10.5 Workshop on Applications of the Reed–Muller Expansion in Circuit Design Reed–Muller'97*, September, 1997, Oxford, England, pp. 219–233.

[61] Perkowski, M. A., Jozwiak, L. and Drechsler, R., "New hierarchies of AND/EXOR trees, decision diagrams, lattice diagrams, canonical forms, and regular layouts", *Proc. IFIP WG 10.5 Workshop on Applications of the Reed–Muller Expansion in Circuit Design Reed–Muller'97*, September, 1997, Oxford, England, pp. 115–132.

[62] Perkowski, M. A., Jozwiak, L., Drechsler, R. and Falkowski, B. J., "Ordered and shared, linearly-independent, variable-pair decision diagrams", *Proc. Int. Conf. on Information, Communication and Signal Processing, ICICS'97*, Singapore, September, 1997, pp. 261–265.

[63] Perkowski, M. A., Pierzchala, E. and Drechsler, R., "Ternary and quaternary lattice diagrams for linearly-independent logic, multiple-valued logic, and analog synthesis", *Proc. Int. Conf. on Information, Communication and Signal Processing, ICICS'97*, Singapore, September, 1997, pp. 269–273.

[64] Perkowski, M. A., Sarabi, A. and Beyl, E. R., "XOR canonical forms of switching functions", In: Kebschull, U., Schubert, E. and Rosenstiel, W. Eds., *Proc. IFIP WG 10.5 Workshop on Applications of the Reed–Muller Expansion in Circuit Design*, September, 1993, Hamburg, Germany, pp. 27–32.

[65] Rahardja, S. and Falkowski, B. J., "Family of Complex Hadamard transforms: Relationships with other transforms and Complex Composite spectra", *Proc. 27th IEEE Int. Symp. on Multiple-Valued Logic*, Antigonish, Nova Scotia, Canada, May, 1997, pp. 125–130.

[66] Rahardja, S. and Falkowski, B. J., "Detection of Boolean symmetries using complex Hadamard transformation", *Proc. 30th IEEE Int. Symp. on Circuits and Systems*, Hong Kong, June, 1997, 2116–2119.

[67] Rahardja, S. and Falkowski, B. J., "Classification and graph based representations of switching functions using a novel spectral technique", *Int. Journal of Electronics*, December, 1997, **83**(6), 731–742.

[68] Rahardja, S. and Falkowski, B. J., "Symmetry conditions of Boolean functions in Complex Hadamard transform", *IEE Electronics Letters*, **34**(17), August, 1998, 1634–1635.

[69] Rahardja, S. and Falkowski, B. J., "Digital signal processing with Complex Hadamard transform", *Proc. 4th Int. Conf. on Signal Processing*, Beijing, China, October, 1998, pp. 533–536.

[70] Reed, S. M. (1954). "A class of multiple error correcting codes and their decoding scheme", *IRE Trans. Inf. Th.*, Vol. PGIT-4, pp. 38–49.

[71] Sarabi, A., Ho, P. F., Iravani, K., Daasch, W. R. and Perkowski, M. A., "Minimal multi-level realization of switching functions based on Kronecker functional decision diagrams", *Proc. Int. Workshop on Logic Synthesis*, Lake Tahoe, Calif., USA, 1993, pp. 3a1–3a6.

[72] Sasao, T. (Ed.), *Logic Synthesis and Optimization*, Kluwer, 1993.
[73] Sasao, T., "AND-EXOR expressions and their optimizations", In: Sasao., T. (Ed.), *Logic Synthesis and Optimization*, Kluwer Academic Publishers, 1993, pp. 287–312.
[74] Sasao, T., "FPGA design by generalized functional decomposition", In: Sasao, T. (Ed.), *Logic Synthesis and Optimization*, Kluwer Academic Publishers, 1993, pp. 233–258.
[75] Sasao, T., "Representations of logic functions by using EXOR operators", In: Sasao, T. and Fujita, M. (Eds.), *Representations of Discrete Functions*, Kluwer Academic Publishers, 1996, pp. 29–54.
[76] Sasao, T. and Besslich, Ph. W. (1990). "On the complexity of MOD-2 sum PLA's", *IEEE Trans. Comput.*, **33**(2), 262–266.
[77] Sasao, T. and Butler, J. T., "A design method for look-up table type FPGA by pseudo–Kronecker expansions", *Proc. 24th Int. Symp. on Multiple-Valued Logic*, Boston, Massachusetts, May, 1994, pp. 97–106.
[78] Sasao, T. and Debnath, D., "An exact minimization algorithm for generalized Reed–Muller expansions", *Proc. IEEE Asia-Pacific Conference on Circuits and Systems*, December, 1994, Taipei, Taiwan, pp. 460–465.
[79] Sasao, T. and Fujita, M. (Eds.), *Representations of Discrete Functions*, Kluwer, 1996.
[80] Sawada, H., Yamashita, S. and Nagoya, A., "Restructuring logic representations with easily detectable simple disjoint decompositions", *Proc. Design and Automation and Test in Europe*, February, 1998, pp. 755–759.
[81] Schaefer, I., Perkowski, M. A. and Wu, H., "Multilevel logic synthesis for cellular FPGAs based on orthogonal expansions", In: Kebschull, U., Schubert, E. and Rosenstiel, W. Eds., *Proc. IFIP WG 10.5 Workshop on Applications of the Reed–Muller Expansion in Circuit Design*, September, 1993, Hamburg, Germany, pp. 42–51.
[82] Srinivasan, A., Kam, T., Malik, Sh. and Brayant, R. K. (1990). "Algorithms for discrete function manipulation", *Proc. Int. Conf. on CAD*, pp. 92–95.
[83] Stanković, M., Janković, D. and Stanković, R. S. (1996). "Efficient algorithms for Haar spectrum calculation", *Scientific Review*, No. 21–22, pp. 171–182.
[84] Stanković, R. S. (1982). "A note on the relation between Reed–Muller expansions and Walsh transform", *IEEE Transactions on Electromagnetic Compatibility*, **EMC-24**(1), 68–70.
[85] Stanković, R. S., "Some remarks on Fourier transforms and differential operators for digital functions", *Proc. 22nd Int. Symp. on Multiple-Valued Logic*, May, 1992, Sendai, Japan, pp. 365–370.
[86] Stanković, R. S., "Functional decision diagrams for multiple-valued functions", *Proc. 25th Int. Symp. on Multiple-Valued Logic*, Bloomington, Indiana, USA, May, 1995, pp. 284–289.
[87] Stanković, R. S., "Some remarks about spectral transform interpretation of MTBDDs and EVBDDs", *ASP-DAC'95*, Makuhari Messe, Chiba, Japan, August, 1995, pp. 385–390.
[88] Stanković, R. S. (1996). "Fourier decision diagrams for optimization of decision diagrams representations of discrete functions", *Proc. Workshop on Post Binary-ultra Large Scale Integration*, Santiago de Campostela, Spain, pp. 8–12.
[89] Stanković, R. S., "*Fast Fourier transform on finite non-Abelian groups*", In: [99], pp. 405–420.
[90] Stanković, R. S., "Fourier decision diagrams on finite non-Abelian groups with preprocessing", *Proc. 27th Int. Symp. on Multiple-Valued Logic*, Antigonish, Nova Scotia, Canada, May, 1997, pp. 281–286.
[91] Stanković, R. S., *Spectral Transform Decision Diagrams in Simple Questions and Simple Answers*, Nauka, Belgrade, 1998.
[92] Stanković, R. S. and Falkowski, B. J., "FFT and decision diagrams based methods for calculation of spectral transforms", *Proc. IEEE Int. Conf. on Information, Communications and Signal Processing* (1st ICICs), Singapore, September, 1997, **1**, 241–245.
[93] Stanković, R. S., Milenović, D. and Janković, D., "Quaternion groups *versus* dyadic groups in representations and processing of switching functions", *Proc. 29th Int. Symp. on Multiple-Valued Logic*, Freiburg im Bereisgau, Germany, May, 1999, pp. 18–23
[94] Stanković, R. S., Sasao, T. and Moraga, C., "*Spectral transform decision diagrams*", In: [79], pp. 55–92.
[95] Stanković, R. S. and Sasao, T., "Spectral interpretation of TDDs", *Proc. 6th Workshop on Synthesis and System Integration of Mixed Technologies, SASIMI-97*, Osaka, Japan, December, 1997, pp. 45–50.
[96] Stanković, R. S. and Sasao, T., "Decision diagrams for discrete functions: classification and unified interpretation", *Proc. Asian and South Pacific Design Automation Conference, ASP-DAC'98* Yokohama, Japan, February, 1998, pp. 439–446.
[97] Stanković, R. S., Stanković, M., Moraga, C. and Sasao, T., "Calculation of Vilenkin–Chrestenson transform coefficients of multiple-valued functions through multiple-place decision diagrams", *Proc. 5th Int. Workshop on Spectral Techniques*, March, 1994, Beijing, China, pp. 107–116.
[98] Stanković, R. S., Stanković, Moraga, C. and Sasao, T., "Calculation of Reed–Muller-Fourier coefficients of multiple-valued functions through multiple-place decision diagrams", *Proc. 24th Int. Symp. on Multiple-Valued Logic*, Boston, USA, May, 1994, pp. 82–88.
[99] Stanković, R. S., Stojić, M. R. and Stanković, M. S. (Eds.), *Recent Developments in Abstract Harmonic Analysis with Applications in Signal Processing*, Nauka, Belgrade, 1996.
[100] Varma, D. and Trachtenberg, E. A. (1989). "Design automation tools for efficient implementation of logic functions by decomposition", *IEEE Trans. on CAD*, **8**(8), 901–916.
[101] Varma, D. and Trachtenberg, E. A., "Efficient spectral techniques for logic synthesis", In: Sasao, T. (Ed.), *Logic Synthesis and Optimization*, Kluwer Academic Publishers, Boston, 1993, pp. 215–232.
[102] Vrudhula, S. B. K., Pedram, M. and Lai, Y.-T., "*Edge valued binary decision diagrams*", In: [79], pp. 109–132.
[103] Wu, H., Perkowski, M. A. and Zhuang, N., "Synthesis of multiplexer directed-acyclic-graph network with application to FPGA and BDDs", *Proc. Int. Workshop on Logic Synthesis*, Tahoe City, USA, May, 1993, 8d/1–8.
[104] Zalmanzon, L. A., *Fourier, Walsh and Haar Transforms and Their Applications in Control, Communication and Other Fields*, Nauka, Moscow, 1989 (in Russian).
[105] Zhegalkin, I. I. (1927). "On the techniques of calculating sentences in symbolic logic", *Math. Sb.*, **34**, 9–28 (in Russian).
[106] Zhegalkin, I. I. (1928). "Arithmetic representations for symbolic logic", *Math. Sb.*, **35**, 311–377 (in Russian).

Authors' Biographies

Bogdan J. Falkowski received the M.S.E.E. degree from the Technical University of Warsaw, Poland, and the Ph.D. degree form Portland State University, Oregon, USA.

His industrial experience includes research and development positions at several companies from 1978 to 1986. He then joined the Electrical Engineering Department at Portland State University. Currently, he is an Associate Professor with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore which he joined in 1992.

His research interests include VLSI systems and design, synthesis and optimization of switching circuits, multiple-valued systems, design of algorithms, design automation, digital signal and image processing. He has published 3 book chapters and over 140 articles in the refereed journals and conferences.

Dr. Falkowski is a Senior Member of IEEE and a Member of IEEE Computer Society and IEEE Circuits and Systems Society. He is a member of Eta Kappa Nu and Pi Beta Upsilon.

Radomir S. Stanković received B.E. degree in Electronic Engineering from Faculty of Electronics, University of Niš, in 1976, and M.Sc., and Ph.D. degrees in Applied Mathematics from Faculty of Electrical Engineering, University of Belgrade, in 1984, and 1986, respectively.

He was with High School of Electrotechnic, Niš, from 1976 to 1987. From 1987 to date he is with Faculty of Electronics, Niš. Presently, he is a Professor teaching Logic Design.

His research interests include switching theory and multiple-valued logic, signal processing and spectral techniques.

He served as the co-editor and editor of two editorials and the author of couple of monographs in spectral techniques.