Spectral Techniques for Design and Testing of Computer Hardware

Mark G. Karpovsky
Reliable Computing Lab.
Boston University
Boston
USA

Radomir S. Stanković
Dept. of Computer Science
Faculty of Electronics
18 000 Niš
Yugoslavia

Jaakko T. Astola
Tampere Int. Center
for Signal Processing
Tampere University of Technology
Tampere
Finland

Abstract

This paper presents a tutorial review of spectral methods in logic synthesis and testing of digital devices. The methods discussed are based on spectral characteristics of Boolean functions and autocorrelation functions. Then, we proposed a method for optimization of decision diagram representations of systems of Boolean functions through the autocorrelation functions.

1 Introduction

Spectral techniques are a mathematical discipline which may be described as an area of abstract harmonic analysis devoted to the applications in engineering, primarily electrical and computer engineering.

Spectral techniques provide an approach to the problems in these areas which often permits derivation of alternative methods for solving complex tasks efficiently in terms of space and time. Transferring a problem from the original into the spectral domain may provide several advantages. In particular, some numerical calculation tasks difficult to perform in the original domain, may be simple in the spectral domain. The convolution product, that is often used in description and mathematical modeling of linear shift-invariant systems, is a supporting example. Some properties of a signal or a system, which are shadowed or difficult to observe in the original domain, become easy observable in the spectral domain. Examples are determination of cut-off frequencies and sampling rates in signal processing, and detection of decomposability and symmetry properties in logic design, see for example, [1], [9], [25], [26]. The fast calculation algorithms for spectral coefficients further improve performances of the related algorithms. Decision diagrams (DDs) extends applicability of these algorithms to functions defined in a large number of points, [3], [37].

In digital devices design, cost of design is growing very fast with an increasing number of components. Cost of testing is also growing with the increasing density of components and limited number of input and output pins. Poor controllability and poor observability make the problem of testing even more complex. The cost of testing for many products is higher than cost of design. Traditional methods of design and testing require brute force computer search for solving optimization problems. Unlike to that, spectral methods may provide for simple "analytic" solutions.

To support research and related activities in applications, the First Workshop on Spectral Methods was organized in 1983, at Boston University, USA, providing a forum for discussion and

exchange of ideas among at about 30 participants. After that, the workshops on spectral techniques were organized in Montreal, Canada, 1986, Dortmund, Germany, 1989, Beijing, P.R. of China, 1990, and 1994. *SPECLOG 2000*, in Tampere, Finland is a continuation of attempts in organizing, joining efforts, and conducting research in spectral techniques.

The present paper is a contribution to these efforts. It discusses the spectral methods in logic design and testing of digital devices. The results presented are based on the use of spectral characteristics of the functions realized and their autocorrelation functions. Then, we introduce a method for optimization of DD representations for systems of Boolean functions through their integer equivalents and autocorrelation functions.

Presentation in this paper is organized as follows.

In Section 2, we present basic facts from harmonic analysis over finite (Abelian) groups. In Section 3, we discuss applications to spectral synthesis. In Section 4, we present methods for testing of digital devices based on spectral characteristics of functions realized by devices under testing. In Section 5, spectral techniques for robust data compression are reviewed. In Section 6, we introduce a method for optimization of DD representations through the autocorrelation functions. Paper ends with some closing remarks about advantages and limitations of spectral methods and their present use in practice.

2 Spectral Transforms

In this paper, a discrete function $f(x)$ is considered as a mapping $f : G \to P$, $x \in G$, $f(x) \in P$, where G is an Abelian group of order $|G| = N$, and P is the field of complex numbers or a finite (Galois) field. We denote by $P(G)$ the space of all thus defined functions.

Example 1 *The switching functions are a particular example, when $x_i \in \{0,1\}$, thus $G = (\{0,1\}, \oplus) = C_2^m$, where $\oplus$ denotes the addition modulo 2 (EXOR), and $P = GF(2)$.*

Alternatively, m-variable switching functions can be considered as a subset of functions in $C(C_2^m$, where C is the field of complex numbers, if the logic values 0 and 1 for $f(x)$ are interpreted as integers 0, and 1, respectively. In this case, a system of k switching functions can be represented by an integer equivalent function $f(z) = \sum_{i=0}^{k-1} 2^{k-1-i} f_i$.

A spectral transform is defined as a mapping $T : f(x) \to \widehat{f}(w)$, where $x, w \in G$, $f(x), \widehat{f}(w) \in P$.

The mapping T is defined in terms of a set of basic functions. Different spectral transforms are defined by choosing different sets of basic functions [9].

A merit of performing such a mapping T over a given function f is in the property that many problems of design and testing are difficult in the original domain x and simple in the "frequency" domain w.

2.1 Fourier transforms on finite groups

Fourier transform over G is defined by using the group characters as the set of basic functions.

For a given group G, the groups characters of χ_w over the complex field C are defined as a homomorphism $\chi_w : G \to C$ with $|\chi_w| = 1$. Thus,

$$\chi_w(x \oplus y) = \chi_w(x)\chi_w(y).$$

The multiplicative group $\{\chi_w\}$ is isomorphic to the original (additive) Abelian group G. The set of group characters forms a complete orthogonal system, thus, a basis in $C(G)$. Therefore,

$$\frac{1}{N} < \chi_w, \overline{\chi}_\nu >= \delta_{w,\nu},$$

where $< \cdot, \cdot >$ denotes the inner product in $C(G)$.

Then, the Fourier transform in $C(G)$ is defined as follows

$$\widehat{f} = \frac{1}{N} \sum_x f(x) \overline{\chi}_w(x) = \frac{1}{N} < f, \overline{\chi}_w >,$$

where $< ., >$ denotes the inner product in $C(G)$.

The inverse transform is defined by

$$f(x) = \sum_w \widehat{f}(w) \chi_w(x).$$

Example 2 *(Discrete Walsh Transform)*
Let $G = C_2^m$. Then, the group characters are the discrete Walsh functions $\chi_w(x) = Wx(x) = (-1)^{<w,x>}$, where $<,.>$ denotes the inner product.

Note that group of Walsh functions $\{W_w(x)\}$ is isomorphic to the group of linear Boolean functions

$$f(x_1, \ldots, x_m) = \bigoplus_{i=1}^{m} c_i x_i.$$

In matrix notation, Walsh transform is defined by the Walsh matrix

$$\mathbf{W}_m = \bigotimes_{i=1}^{m} \mathbf{W}_1,$$

where $\otimes$ denotes the Kronecker product and the basic Walsh matrix is defined as

$$\mathbf{W}_1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

For example if $m = 3$,

$$\begin{aligned} \mathbf{W}_3 &= \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}. \end{aligned}$$

The Walsh spectrum $\widehat{f}$ represented as a vector $\widehat{\mathbf{f}} = [\widehat{f}(0), \ldots, \widehat{f}(2^n - 1)]^T$, is defined as

$$\widehat{f} = 2^{-m}\mathbf{W}_m\mathbf{f},$$

where $\mathbf{f}$ is the vector of function values. Thus, $\mathbf{f} = [f(0), \ldots, f(2^n - 1)]^T$.

Since the entries of $\mathbf{H}_m$ are in $\{1, -1\}$, the Walsh spectrum can be calculated in terms of additions. A factorization of $\mathbf{H}_m$ in terms of the Kronecker product representable sparse matrices permits derivation of a fast algorithm (FWHT) for calculation of the Walsh transform. Complexity of FWHT (number of additions) is $m2^m$.

The same considerations apply if the groups characters are defined over some field P different from C provided that some relationships between $|G|$ and the characteristics of P are satisfied.

2.2 Related transforms

In some applications, it is convenient to use basic functions different from the Fourier basis consisting of group characters.

For example, if $f(x) \in C(C_2^m)$ can be approximated by a polynomial of a small degree, then the basic functions for a Fourier-like transform may be selected as the discrete Haar functions H_w.

Example 3 *(Discrete Haar functions)*
For $m = 3$, and $G = C_2^3$, the unnormalized discrete Haar functions are defined as rows of the Haar matrix

$$\mathbf{H}_3 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}.$$

In this case, $\widehat{f}$ depends on a local behavior of $f(x)$. Computation of

$$f(x) = \sum_w H_w(x)\widehat{f}(w),$$

in a finite field (Haar-Galois transform) results in a decrease of the memory size.

2.3 Autocorrelation function

Autocorrelation is a very useful concept in spectral methods for analysis and synthesis of networks realizing logic functions.

For a given m-variable switching function f, the autocorelation function B_f is defined as

$$B_f(\tau) = \sum_{x=0}^{2^m-1} f(x)f(x \oplus \tau), \quad \tau \in \{0, \ldots, 2^m - 1\},$$

The Winer-Khinchin theorem states a relationship between the autocorrelation function and Walsh (Fourier) coefficients

$$B_f = 2^m W^{-1}(Wf)^2.$$

It may be remarked that the autocorrelation function is invariant to the shift operator $\oplus$ in terms of which B_f is defined. Due to that, it performs some compression of data in the sense that several functions may have the same autocorrelation function B_f. Fig. 1 illustrates this property of B_f. In this figure, $\varphi_\tau = f(x \oplus \tau)$ is a shifted function for f, WK denotes the Wiener-Khinchin theorem, and C_{B_f} denotes the set of functions having the same autocorrelation function B_f. However, although we are sacrificing a part of data, this compression makes description of problems where the shift is not important in terms of the autocorrelation language very efficient. For example, the autocorrelation is very useful in the applications where we are interested in the equality of some function values, and not in their magnitude.

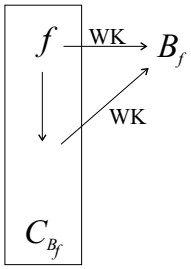

Figure 1: Autocorrelation functions.

For a system of k switching functions $f^{(i)}(x_1, \ldots, x_n)$, $i = 0, \ldots, k-1$, the total autocorrelation function is defined as the sum of autocorelation functions of each function in the system. Thus,

$$B_f(\tau) = \sum_{i=0}^{k-1} B_{f^{(i)}}(\tau).$$

Note that for any $\tau \neq 0$, $B_f \leq B_f(0)$. The set $G_I(f)$ of all values for τ such that $B_f(\tau) = B_f(0) = \sum_{i=0}^{k-1} \sum_{x=0}^{2^m-1} f^{(i)}(x)$ is a group with respect to the EXOR as the group operation which is denoted as the inertia group of the system f.

A generalization of autocorrelation to systems of p-valued m-variable functions is straightforward. For a system of p-valued functions $f(z) = \{f^{(i)}(z^{(0)}, \ldots, z^{(m-1)})\}$, $i = 0, \ldots, k-1$, $z^{[i]} \in \{o, \ldots, p-1\}$, a system of characteristic functions is defined as

$$f_r^{(i)} = \begin{cases} 1, & f^{(i)}(z) = r, \\ 0, & f^{(i)}(z) \neq r. \end{cases}$$

Then, the total autocorrelation function is defined as [9]

$$B_f(\tau) = \sum_{r=0}^{p-1} \sum_{i=0}^{k-1} B_r^{(i)}(\tau).$$

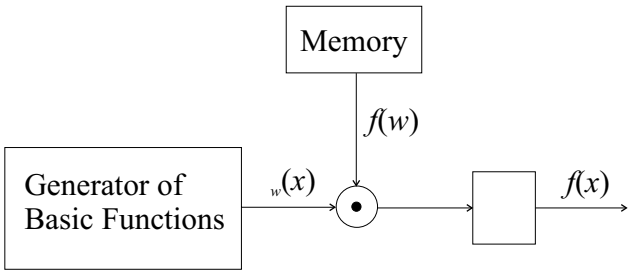

Figure 2: Spectral synthesis.

Further generalizations to functions on arbitrary finite Abelian groups are also straightforward, see for example, [9]. In this case, the Winer-Khinchin theorem is defined with respect to the Fourier transforms defined in terms of group characters. It should be noted that this theorem is correct for the autocorrelation functions of discrete functions and characteristic functions associated to them, however, it does not apply to the related total autocorrelation functions.

3 Spectral Synthesis

Fig. 2 illustrates the method of spectral synthesis by the example of Fourier transforms. It is assumed that a given function f is represented as

$$f(x) = 2^{-m} \sum_{w} \widehat{f}(w)\chi_w(x),$$

Spectral coefficients $\widehat{f}(w)$ for f are stored in a memory and added, after multiplication with basic functions χ_w provided by a suitably designed function generator, which produces f at the output of the system.

Complexity of spectral realization of a given f (memory size) is proportional to the number of non-zero coefficients $\widehat{f}(w)$ which have to be stored in the memory for a hardware implementation of f. The same method applies to any other spectral transform. Advantages of spectral synthesis are

1. Structural design and simple architectures consisting of standard components such as adders, memories, etc.
2. Simple off-line testing and on-line error detection and correction.
3. Adaptability and programmability for a given functional behavior.
4. Efficient optimization techniques.

In particular, spectral synthesis is efficient for digital function generators.

Example 4 *For the function $f(x) = \sin x$, where the argument x and the function values are represented by 16 bits, thus, $x \in C_2^{16}$, $f(x) \in C_2^{16}$, the transition from the traditional synthesis to the spectral approach with Haar functions results in about 25% hardware savings.*

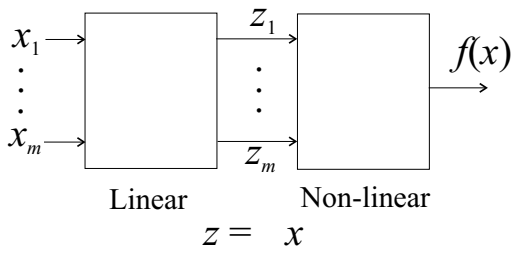

Figure 3: Realization by linearization of f.

3.1 Complexity of realization of Boolean functions

Let $f : C_2^m \to C_2$ be a Boolean function. Denote by $L(f)$ the minimum number of two input gates to implement f.

It is shown that for almost all f, $L(f) \sim \frac{2^m}{m}$ as $m \to \infty$ [21], [33].

Denote by $\mu(f) = |\{(x,y)|x \in C_2^m, y \in C_2^m, d(x,y) = 1, f(x) = f(y)\}|$. It may be stated [34] that for almost all f_1, f_2, as $m \to \infty$,

$$\mu(f_1) \geq \mu(f_2) \to L(f_1) \geq L(f_2).$$

Complexity of realization may be reduced by the linearization and polynomial approximation of Boolean functions.

3.2 Linearization of Boolean functions

Linearization of Boolean functions assumes representing a given system of Boolean functions as the superposition of a system of linear Boolean functions and a residual nonlinear part of minimal complexity. Fig. 3 shows the realization of a given function f based on the linearization. The network produced consists of a serial connection of a linear and a nonlinear blocks. The linear block consists of EXOR circuits only. For an m-variable function, complexity of the linear block increases asymptotically no faster than $m^2/\log_2 m$ as $m \to \infty$ [9], whereas the complexity of the nonlinear block is almost always an exponentially increasing function of m. Therefore, the complexity of the linear block may be ignored in linearization problems.

The linearization of a system of Boolean functions is performed to meet some criterion of the complexity of the realization of $f(x)$. In this paper, we consider the complexity of $f(x)$ as the minimum number of two input gates to implement $f(x)$.

3.2.1 Linearization problem

The linearization problem may be formulated as follows.

For a given $f : C_2^m \to C_2$, find a nonsingular over $GF(2)$ $(m \times m)$ matrix σ such that

$$f(x) = \nu(\sigma x),$$

and $\mu(\nu) \to max_\sigma$.

3.2.2 Solution of the linearization problem

A solution of the linearization problem can be found as follows.

1. Construct by the Wiener-Khinchin theorem and FWHT the autocorrelation function

$$B_f(\tau) = \sum_x f(x)f(x \oplus \tau),$$

2. Find τ_0 such that $B(\tau) = max_{\tau \neq 0} B(\tau)$.

3. Find τ_i, $i = 1, \ldots, m-1$, such that $B(\tau_i) = max_{\tau \notin T_{i-1}} B(\tau)$, where $T_i = \{c_0\tau_0 \oplus c_1\tau_1 \oplus \cdots \oplus c_{i-1}\tau_{i-1}\}$, $c_i \in \{0, 1\}$.

4. Construct

$$\mathbf{T} = \begin{bmatrix} \tau_0 \\ \tau_1 \\ \cdot \\ \cdot \\ \cdot \\ \tau_{m-1} \end{bmatrix},$$

and determine

$$\sigma = \mathbf{T}^{-1}$$

where all the calculations are in $GF(2)$.

Complexity of solving the linearization problem for a given f does not exceed $O(m2^m)$ and may be much smaller than this if we have a compact description of f.

For randomly generated Boolean functions, the linearization results in about 20% reduction in the gate counts. Generalization to multi-valued p-ary logic (p-prime) is possible.

The method of synthesis by linearization will be illustrated by the following example.

Example 5 *Table 1 shows a system of two four-variable Boolean functions $f^{(0)}$ and $f^{(1)}$, and the total autocorrelation function B of this system [9]. Fig. 4 shows a direct AND-EXOR realization of this system with two-input circuits. This realization is chosen for a comparison to the realization by linearization of f, since in the linearization method the linear part is realized by EXOR circuits, and the criterion for minimization is expressed in terms of two-input gates count. The maximum value of $B(\tau) = B(0) = 16$ for the inputs $5 = (0101)$, $10 = (1010)$, and $15 = (1111)$. Thus, as it is pointed out in Section 2.3, the inertia group for this system is $G_I = \{(0000), (0101), (1010), (1111)\}$. As a basis for G_I we take (0101) and (1010), and determine*

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

Then,

$$\sigma = \mathbf{T}^{-1} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

The matrix $\mathbf{T}$ *determines a reordering of the truth-vector* $\mathbf{F}$ *of a four-variable function* f *as*

$$\mathbf{F} = [f(0), f(10), f(5), f(15), f(4), f(14), f(11), f(8).f(2), f(13), f(7), f(12), f(6), f(9), f(3)]^T.$$

Table 1 shows functions $f_\sigma^{(0)}$ *and* $f_\sigma^{(1)}$ *produced by this reordering from* $f^{(0)}$ *and* $f^{(1)}$*. Thus, these functions satisfy the relation*

$$f_\sigma(x) = f(\sigma^{-1}x).$$

Therefore, we determine the intermediate values z_1, z_2, z_3, z_4 *in Fig. 3 from* $\sigma = \mathbf{T}^{-1}$*, as*

$$\begin{aligned} z_1 &= x_1 \oplus x_3, \\ z_2 &= x_2 \oplus x_4 \\ z_3 &= x_4 \\ z_4 &= x_3. \end{aligned}$$

From there,

$$\begin{aligned} f^{(0)} &= z_1 \vee z_2 = (x_1 \oplus x_4) \vee (x_2 \oplus x_3) \\ f^{(1)} &= z_1 z_2 = (x_1 \oplus x_4)(x_2 \oplus x_3), \end{aligned}$$

where $\vee$ *denotes logical OR. It should be noted that both* $f^{(0)}$ *and* $f^{(1)}$ *do not essentially depend on* z_3 *and* z_4*.*

Fig. 5 shows the corresponding AND-EXOR realization of this system of Boolean functions by two-input circuits. Thus, for a comparison with the direct realization in Fig. 4, the logical OR is realized by using AND and EXOR circuits.

3.3 Polynomial approximation of Boolean functions

In the previous section, the realization of a given f was based on linear and nonlinear blocks connected serially, minimizing the complexity of the nonlinear part. Polynomial approximation of systems of switching functions generates networks with the linear and nonlinear blocks connected in parallel, and minimizing a complexity of the nonlinear part. Unlike the method for linearization of switching functions, which is based on the autocorrelation function, the polynomial approximations are based on the efficient use of spectral characteristics in optimization problems.

A switching function is approximated by a product of q linear functions $l_0, \ldots, l_{q-1}$ if

$$f(x) = f_p(x) \prod_{i=0}^{q-1} l_i(x).$$

Table 1: System of Boolean functions.

	$x_1x_2x_3x_4$	$f^{(0)}$	$f^{(1)}$	B	$f_\sigma^{(0)}$	$f_\sigma^{(1)}$
0	0000	0	0	16	0	0
1	0001	1	0	8	0	0
2	0010	1	0	8	0	0
3	0011	1	1	8	0	0
4	0100	1	0	8	1	0
5	0101	0	0	16	1	0
6	0110	1	1	8	1	0
7	0111	1	0	8	1	0
8	1000	1	0	8	1	0
9	1001	1	1	8	1	0
10	1010	0	0	16	1	0
11	1011	1	0	8	1	0
12	1100	1	1	8	1	1
13	1101	1	0	8	1	1
14	1110	1	0	8	1	1
15	1111	0	0	16	1	1

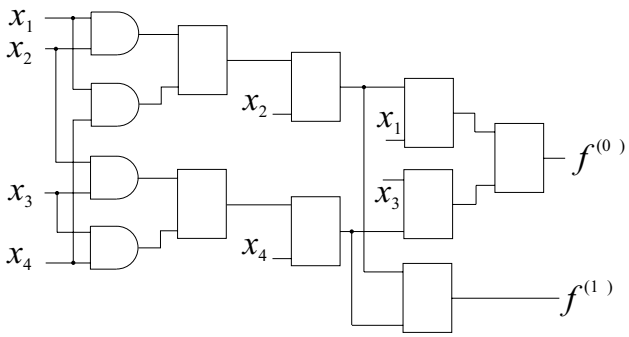

Figure 4: Realization of $f^{(0)}$ and $f^{(0)}$ in Example 5.

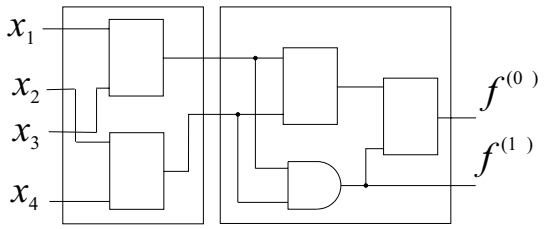

Figure 5: Realization by linearization of $f^{(0)}$ and $f^{(0)}$ in Example 5.

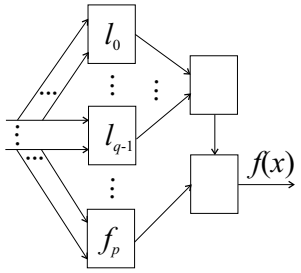

Figure 6: Realization by polynomial approximation of f.

The function $P(x) = \prod_{i=0}^{q-1} l_i(x)$ is a polynomial of degree q denoted as a polynomial approximation of f. Fig. 6 shows the structure of a network which can be used to realize a given f which has a polynomial approximation $P(x)$. In this figure, $\times$ denotes logic multiplication of inputs in the circuit.

An m-tuple $l = (l_0, \ldots, l_{m-1})$ is denoted as a linearity point for a given $f(x_0, \ldots, x_{m-1})$ if

$$l(x) = \bigoplus_{s=0}^{m-1} l_s x_s \oplus l_m,$$

is an approximation of f.

It is shown [9], that a switching function $f(x_0, \ldots, x_{m-1})$ has linearity points $l_0, \ldots, l_{q(f)-1}$ iff

$$|\widehat{f}(l_i)| = 2^{-m} \sum_{x=0}^{2^m-1} f(x), \quad i = 0, \ldots, q(f) - 1,$$

where $\widehat{f}$ is the Walsh spectrum of f and $l_i = \sum_{s=0}^{m-1} l_s^{(i)} 2^s$.

Due to that, a solution of the polynomial approximation can be found as follows.

1. Calculate the Walsh spectrum $\widehat{f}$ of the function f.
2. Construct the group of linearity points l_i for which $|\widehat{f}(l_i)| = 2^{-m} \sum_x f(x)$, and select an arbitrary basis of this group.
3. Determine $P(x)$ as

$$P(x) = \prod_{i=0}^{q(f)-1} \left(\bigoplus_{s=0}^{m-1} l_s^{(i)} x_s \oplus sign(S_f(l_i)) \right) \bmod 2,$$

where

$$sign(x) = \begin{cases} 1, & \text{if } x > 0, \\ 0, & \text{if } x \geq 0. \end{cases}$$

The following example illustrates the method and application of polynomial approximations of Boolean functions.

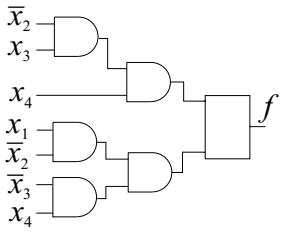

Figure 7: Realization for f in Example 6.

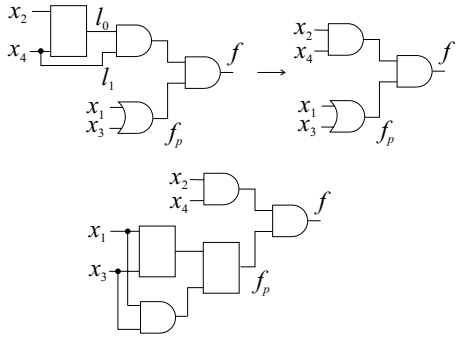

Figure 8: Realization by polynomial approximation for f in Example 6.

Example 6 *Consider a four-variable Boolean function* $f = \overline{x}_2 x_3 x_4 \oplus \overline{x}_1 x_2 x_3 \overline{x}_4$. *The truth-vector for* f *is* $\mathbf{F} = [0001000001010000]^T$, *and the Walsh spectrum for* f *is*

$$\widehat{\mathbf{f}} = [3, -1, 3, -1, -1, -1, -1, -1, -3, 1, -3, 1, 1, 1, 1, 1]^T.$$

Therefore, $|\widehat{f}(0)| = |\widehat{f}(2)| = |\widehat{f}(8)| = |\widehat{f}(10)| = 2^{-4} \sum_x f(x) = 3/16$.

It follows, that the group of linearity points consists of inputs (0000), (0100), , (0101). *As a basis we take* $l_0 = (0001)$, *and* $l_1 = (0101)$. *Since* $sign(\widehat{f}(l_0)) = sign(\widehat{f}(8)) = 0$, *and* $sign(\widehat{f}(l_1)) = sign(\widehat{f}(10)) = 0$, *the optimal approximation is* $P(x) = (x_2 \oplus x_4)x_4$. *The nonlinear part is* $f_p(x) = x_1 \vee x_3$. *Therefore,* $f = x_4(x_2 \oplus x_4)(x_1 \vee x_3)$.

Fig. 7 shows a straightforward realization of f *by two-input circuits. Fig. 8 shows the realization through the polynomial approximation for* f. *We have shown a straightforward realization through polynomial approximation corresponding to the general model in Fig. 6, then a realization by taking advantages that* $x_4(x_2 \oplus x_4) = x_2 x_4 \oplus x_4 = x_4(x_2 \oplus 1) = \overline{x}_2 x_4$, *and an AND-EXOR realization, with logic OR circuit implemented in terms of AND and EXOR circuits.*

4 Spectral Techniques for Functional Testing

With the advent of VLSI and corresponding drastic increase in the density of gates on a chip, high-level functional testing is one of the most viable approaches to the testing of computer hardware.

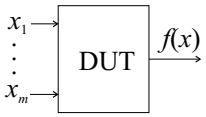

Figure 9: Functional testing.

It may be remarked that testing is the "bottleneck" for computer industry. Interconnection of components in device-under-test (DUT) is too complex or not known for a user, which makes the testing a rather difficult and both space and time complex task.

Fig. 9 shows the basic task in functional testing. The inputs $x_0, \ldots x_{m-1}$ and the output $\tilde{f}$ are accessible. Verify $f = \tilde{f}$, where f is fault free response.

4.1 Linear check tests

Two well known methods of functional testing are signature analysis, see for example, [23], [35], and syndrome check sums [10]. However, these methods have several practical limitation, mainly due to the considerable testing time and the difficulty in estimating the fault coverage. For these reasons, a method denoted as the linear check test sets has been proposed in and further elaborated and used in [4], [12], [13], [19]. The method can be formulated as follows.

For a given $f : C_2^m \to GF(2)$, find a partition $\{B_0, \ldots, B_{r-1}\}$ of C_2^m such that $\forall i, j \ |B_i| = |B_j|$. The subsets B_i are denoted as the test sets and are dependent on the function implemented by DUT. Test sets are determined in such a way that under fault free conditions the sum of the outputs for all inputs within a test set is the same constant K_i for every test set.

The method for error detection consists of the verification of a linear equality check

$$\sum_{\tau \in B_i} f(x \oplus \tau) - K = 0, \tag{1}$$

for every i and every $x \in \{0, \ldots, 2^m - 1\}$.

It follows that a testing procedure can be formulated as follows.

When computing $f(x)$, find i such that $x \in B_i$ and verify $\sum_{x_i \in B_i} f(x) = K_i$.

Good features of the method are that K_i is the only value to be stored for a given f. Test time (test complexity) is $T(f) = |B_i|$. In many cases, the method gives better error detecting and/or error-locating capability, and, in general, requires less testing time.

Example 7 *(m-bit multiplier)*

An m-bit multiplier is a device realizing a function

$$f(x_0, \ldots, x_{m-1}, y_0, \ldots, y_{m-1}) = x \cdot y,$$

where $x = \sum_i x_i 2^{m-1-i}$, *and* $y = \sum_i y_i 2^{m-1-i}$, *with* $x_i, y_i \in \{0, 1\}$.

The domain C_2^{2m} *can be partitioned into the test sets* $\{0^{2m}, 0^m 1^m, 1^m 0^m, 1^{2m}\}$, *where the exponentiation* r^i *is defined as the repetition of the symbol* $r \in \{0, 1\}$, i *times. Therefore, in this case, we have the following check*

$$f(x_0,\ldots,x_{m-1},y_0,\ldots,y_{m-1})+f(x_0,\ldots,x_{m-1},\overline{y}_0,\ldots,\overline{y}_{m-1})$$
$$+f(\overline{x}_0,\ldots,\overline{x}_{m-1},y_0,\ldots,y_{m-1})+f(\overline{x}_0,\ldots,\overline{x}_{m-1},\overline{y}_0,\ldots,\overline{y}_{m-1}) \quad =(2^m-1)^2.$$

For $T(f)=4$, $(x,y),(x,\overline{y}),(\overline{x},y),(\overline{x},\overline{y})$ form one block of the partition for testing.

It was shown in [10], [13], [19], [20], that linear equality checks have very good error-detecting and error-correcting capabilities and may be easily implemented. Very simple equality checks were considered in [19] for many standard computer blocks, and in [13] for programs which evaluate polynomials.

4.2 Linear check for polynomials

Assume that a given $f(x)$ is represented by polynomial, thus, $f(x)=P(x)=\sum_{i=0}^{s}a_ix^i$, where $a_i\in C$, $x=(x_0,\ldots,x_{m-1})$, $x_i\in\{0,1\}$.

For the construction of a check set and a constant K satisfying (1) for $P(x)$, we may use the results from [19], and [13].

Let $V(m,d)$ be a maximal binary linear error-correcting code with the code words of length m and distance d, and let $V^{\perp}(m,d)$ be a dual code to $V(m,d)$. Then,

$$V^{\perp}(m,d)=\{\tau=(\tau_1,\ldots,\tau_m)\in G\}\},$$

with $\bigoplus_{i=1}^{m}\tau_ix_i=0$ for every $x=(x_1,\ldots,x_m]\in V(m,d)$.

Methods for constructing $V(m,d)$,a nd $V^{\perp}(m,d)$ and estimating their cardinalities may be found e.g. in [28]. It was shown in [13] and [19] that if

$$P_d(x)=\sum_{i=0}a_ix^i,$$

with $a_s\neq 0$, and $x\in\{0,\ldots,2^n\}$, then

$$\sum_{\tau\in V^{\perp}(m,d+1)}P_d(x\oplus\tau)-K=0,$$

where

$$K=|V(m,d+1)|^{-1}\sum_{x\in G}P_d(x)=|V(m,d+1)|^{-1}\left(\sum_{i=0}^{d}a_i(i+1)^{-1}\right)\left(\sum_{r=0}^{i}\left(\begin{array}{c}i+1\\ \nu\end{array}\right)\right)2^{(i+1-\nu)m}B_\nu,$$

where B_ν stands for Bernoulli numbers.

A solution of the problem of constructing a linear check test for polynomials can be formulated as follows.

1. Construct a maximal linear code V of length m and distance $s+1$.
2. Select $B_0=V^{\perp}$, where $B_i=coset\ \ of\ \ V^{\perp}$ in C_2^m, and $V^{\perp}$ is the dual code for V.

$$\sum_{\tau\in V^{\perp}}f(x\oplus\tau)=K,$$

where K is the reference value. It follows,

$$f*\delta=K,$$

where $\delta(x)=1$, $x\in V^{\perp}$, $T(f)=\frac{2^m}{|V|}$.

The proof is based on the property that if $degf(x) = s$, then $\widehat{f}(w) = 0$, if $||w|| > s$, where $||w||$ is the Hamming norm of w.

All polynomials of the same degree s have the same check but different reference values K.

Test complexity depends on the degree s and the number m of bits in x.

Example 8 *Consider a function* $f(x) = x^2 - 1$, *where* $m = 3$ *and* $s = 2$. *Therefore,* $V = \{000, 111\}$, $V^{\perp} = B_0 = \{000, 011, 101, 110\}$, *and* $V^{\perp} \oplus 001 = B_1 = \{001, 010, 100, 111\}$. *It follows,*

$$\sum_{\tau \in V^{\perp}} f(x) = \sum_{\tau \in V^{\perp} \oplus 001} = 67,$$

and thus, $T_f = 4$.

4.3 Systems of Linear Orthogonal Checks

Two partitions $B = \{B_0, \ldots, B_{r-1}\}$ and $D = \{D_0, \ldots, D_{t-1}\}$ of C_2^m are orthogonal if $|B_i \cap D_j| = 1$.

Any system of orthogonal linear checks detects up to $2^l - 1$ errors and corrects up to $2^{l-1} - 1$ errors.

4.4 Linear Inequality Checks

It was shown in [10], [13], [19], [20], that equality checks may be efficiently used in the case where $f(x)$ is an integer for every x, and very few non-integer functions have nontrivial equality checks. A generalization of linear check methods to the case of non-integer computations was given in [12]. It were proposed that linear inequality checks

$$\sum_{\tau \in T} f(x \oplus \tau) - K \leq \epsilon, \tag{2}$$

where ϵ is a given small constant, should be used for error detection in numerical computations. In this method, the main task is to construct an optimal inequality with the minimum cardinality $|T|$ of a check set. A solution of that problem was given in [10]. It consists of the following steps.

1. Approximate $f(x)$ by a Chebycheff polynomial (in L_1)

   $$f(x) = P(x) + \Delta(x),$$

   where $\Delta(x) \leq \Delta$ for $\forall x$, and Δ is a given enough small value.

2. Construct $V^{\perp} = B_0$ for $P(x)$, (equality check for P).

   Then $\epsilon = \Delta \cdot |V^{\perp}|$, with $T(f, \epsilon) = T(P, 0)$.

Small increase in ϵ may result in a drastic decrease of test complexity $T(f, \epsilon)$.

Example 9 *Consider a function* $f(y) = y^{-0.5} \sin \pi/2y^{0.5}$, *for* $0 \leq y < 1$. *For* $y = 2^{-23}x$, *and* $x \in \{0, 1, \ldots, 2^{23} - 1\}$, *we have* $T(f, 0) = 2^{23}$.

If $f(y) = P_2(y) + \Delta(y)$, *since* $degP_2(y) = 2$, *then* $\Delta(y) \leq 14 \cdot 10^{-5}$ *for all* y.

For $P_2(y)$, *we have* $T(P_2) = 2^5$, *where* V *is (23, 18, 3) the shortened Hamming code, and* $\epsilon = 14 \cdot 10^{-5} \cdot 2^5 \approx 5 \cdot 10^{-3}$, *and* $T(f, 5 \cdot 10^{-3}) = 25$.

Advantages of the linear inequality checks are

1. The test is independent on the implementation of a program or a device computing the given function $f(x)$.
2. In many practical cases, the checks are very simple and have good error-detecting and error-locating capability.

The main limitations of this approach are

1. Impossibility to use for intermittent faults,
2. Possibility to use when the implemented function $f(x)$ has a good best-absolute-error polynomial approximation.

5 Robust Compression of Test Response

Built-in off-line testing (BIST) becomes in many cases a necessary feature of a VLSI chip. Evidently, the area needed for the checking circuitry and for the storage of reference data should be minimized. This requirement gives rise to testing techniques based on reducing the amount of test response data (response compression) see, for example, [18].

To reduce the storage size, test responses are compressed into an r-bit word called signature. Several compression of test responses techniques (signature analysis techniques) have been reported [15], [23]. We point out few of them using spectral methods [8], [27], [40].

The most popular among different test compression techniques, the linear feedback shift registers (LFSR) the reason being that LFSR can easily be implemented in BIST VLSI design.

In the case of multiple input LFSR compressors, the observed response $\tilde{f} = f \oplus e$, where f is the fault-free response and e is an error, is compressed into a signature $y(\tilde{f})$. Since the mapping $y : \tilde{f} \rightarrow y(\tilde{f})$ is linear, we have $y(\tilde{f}) = y(f) \oplus y(e)$. Therefore, e is masked iff $y(e) = 0$, which we denote as the aliasing of the error. It follows that linear compressors are not robust. In the case of nonlinear compressors, error masking depends on both f and e, that is, e is masked for a given f iff $y(f) = y(f \oplus e)$.

The error masking characteristic function for a compressor y is denoted as $E(f, e) = 1$ iff $y(f \oplus e) = y(f)$, $E(f, e) \in \{0, 1\}$.

An error-masking probability (aliasing probability) is defined as

$$Q = \sum_{f, e \neq 0} E(f, e) Pr(f, e).$$

For complex devices, as processors, it is usually assumed that f and e are independent random variables, and, therefore, $Pr(f, e) = Pr(f)Pr(e)$. Since for an LFSR, $E(f, e) = 1$ iff $y(e) = 0$, then Q for LFSR compressors depends solely on $Pr(e)$. Thus, LFSRs are not robust and their performances depend on distributions of errors $Pr(e)$ which are difficult to estimate (computing these distributions require gate-level or even transistor-level descriptions of DUTs which may be too complex for VLSI devices).

We will describe below a class of robust compressors (quadratic compressors). For these compressors aliasing probabilities do not depend on distributions of errors in DUTs.

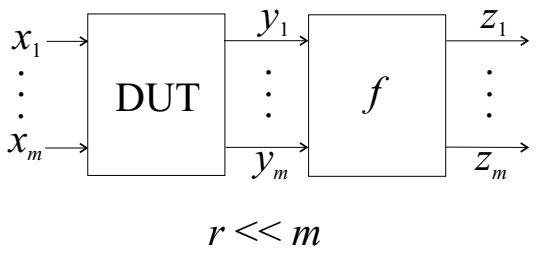

Figure 10: Robust compression of test response.

The concept of quadratic compressors is based on the quadratic nonrepetitive function of $2T$ variables over $GF(2^k)$, the Galois field of 2^k elements

$$y(\widetilde{f}) = \widetilde{f}_0\widetilde{f}_1 \oplus \cdots \oplus \widetilde{f}_{2T-2}\widetilde{f}_{2t-1},$$

where $\widetilde{f}_i \in GF(2^k)$ and $\widetilde{f}$ is a sequence of test responses consisting of $2kT$ bits. We consider $\widetilde{f}$ as a sequence of k-bit symbols $\{\widetilde{f}_i \in GF(2^k),\ i = 0, \ldots, 2T-1$, where $\widetilde{f} = f \oplus e = \{f_i \oplus e_i, f_i, e_i \in GF(2^k)\}$, $i = 0, \ldots, 2T-1\}$.

The k-bit function $y(\widetilde{f})$ is computed by multiplying two k-bit blocks $\widetilde{f}_i$ and $\widetilde{f}_{i+1}$ and accumulating the sum.

It is shown in [17] that quadratic compressors are optimal, that is, for any $r \leq k$ bits representing a signature and chosen form k.bit $y(\widetilde{f})$ we have $Q(e) = 2^{-r}$ for all $e \neq 0$. Therefore, an average performance measure (aliasing probability) for quadratic compressor Q is independent of the statistic of errors and these compressors are robust which provides for an equal protection (equal error masking probability) against all errors

$$P_{AL}(e) = \begin{cases} 1, & e \in Kern(f), \\ 0, & e \notin Kern(f), \end{cases}$$

Thus, compressors based on a non-repetitive quadratic forms over Z_2^r are robust with equal error detection for all errors ($P_{AL}(e) = 2^r$, $e \neq 0$).

Complexity of quadratic compressors is $O(m \cdot r)$.

5.1 Robust quadratic compressor, case $r = 1$, $m = 2s$

We will consider now a special case of robust quadratic compressors with one output line ($r = 1$). In this case, the function implemented by the quadratic compressor can be described as

$$f(y_1, \ldots, y_{2s}) = y_1y_2 \oplus y_3y_4 \oplus \cdots y_{2s-1}y_{2s},$$

where $y_i \in \{0, 1\}$.

We note that f is a bent function [22], i.e., f, represented as a binary vector of length 2^m, is at the maximal Hamming distance from any linear Boolean function of m variables. We note also that

non-repetitive quadratic forms implemented by robust compressors have some properties which are similar to "white noise", i.e., their Walsh autocorrelations are flat

$$B_f(e) = \sum_y f(y)f(y \oplus e) = Const.$$

for $e \neq 0$. Denote

$$\begin{aligned} Y_1 &= (y_1, \ldots, y_r), \\ Y_2 &= (y_{r+1}, \ldots, y_{2r}), \\ &\vdots \\ Y_{2s} &= (y_{2r-1}, \ldots, y_{2r}). \end{aligned}$$

$$f = Y_1Y_2 \oplus Y_3Y_4 \oplus \cdots \oplus Y_{2s-1}Y_{2s},$$

where $f \in Z_2^r$. Then,

1. f is robust for any r,
2. $P_{AL}(e) = 2^{-r}$ for any $e \neq 0$,
3. f is bent (flat autocorrelation) ,

   Let $f_i(y) = 1$ iff $f(y) = 1$.

$$\begin{aligned} B_i(e) &= \sum_y f_i(y)f_i(y \oplus e), \\ B(e) &= \sum_i B_i(e) = Const. \end{aligned}$$

for $e \neq 0$.

6 Reduction of Sizes of DD Representations

Decision diagrams (DDs) are a data structure permitting efficient representation of discrete functions defined on groups of large orders [32]. DDs are derived by the reduction of decision trees (DTs). The reduction is performed by sharing the isomorphic subtrees and deleting the redundant information from the DT. The reduction procedure is formalized through the reduction rules [32] adapted to the range of functions represented.

Different DDs are defined for representation of different classes of discrete functions, [32], [36], [38]. Binary decision diagrams (BDDs) are the basic concept in DD representations [32], and are used to represent Boolean functions [32]. The systems of Boolean functions are represented by Shared BDDs (SBDDs) which are derived by joining isomorphic parts in BDDs for each function in the system. Multi-terminal binary DDs (MTBDDs) [3] are a generalization of BDDs used to represent functions in $C(C_m^2)$. They can represent systems of Boolean functions represented by the corresponding integer equivalent functions $f(z)$.

In many applications, the efficiency of the use of DD representations is determined by the size of the DD defined as the number of nodes in the DD for a given f.

In this section, we show that the linearization method may be useful in DD representations since gives an exact algorithm for linear transformation of DDs. Then, we present a procedure for minimization of MTBDDs for systems of Boolean functions based on the total autocorrelation functions defined below.

6.1 Reduction of sizes of DDs by linearization of Boolean functions

BDDs are derived by the recursive application of the Shannon expansion $f = \overline{x}_i f_0 \oplus x_i f_1$, where f_0, and f_1 are co-factors of f for $x_i = 0$ and $x_i 01$, respectively, to all the variables in f. Linearly transformed BDDs (LT-BDDs) are a generalization derived by performing the expansions with respect to a linear combination of subsets of variables. The chief problem in LT-BDD representations is to determine a suitable linear combination of variables for a given function f. Few heuristic algorithms have been proposed to solve this problem, see for example, [6], [7], [**?**]. Algorithms for efficient manipulation with LT-BDDs are presented in [**?**]. Therefore, the determination of linear combinations of variables in LT-BDDs is an interesting and important task, since LT-BDDs permit for some functions the exponential reduction of the size compared to the BDDs.

The linearization method for Boolean functions presented in Section 3.2 provides an exact algorithm for linear transformation of BDDs and Shared BDDs for systems of Boolean functions [32]. This statement will be explained and illustrated by the following example.

Example 10 *Fig. 11 shows SBDD for the system of Boolean functions $f^{(0)}(x)$ and $f^{(1)}(x)$ defined in Table 1. This SBDD represents the given system in the form of expressions*

$$\begin{aligned} f^{(0)} &= \overline{x}_1\overline{x}_2\overline{x}_3x_4 \oplus \overline{x}_1\overline{x}_2x_3 \oplus \overline{x}_1x_2x_3 \oplus \overline{x}_1x_2\overline{x}_3\overline{x}_4 \oplus x_1\overline{x}_2\overline{x}_3x_4 \oplus x_1x_2x_3\overline{x}_4 \oplus x_1\overline{x}_2\overline{x}_3 \oplus x_1x_2\overline{x}_3, \\ f^{(1)} &= \overline{x}_1\overline{x}_2x_3x_4 \oplus \overline{x}_1x_2x_3\overline{x}_4 \oplus x_1\overline{x}_2\overline{x}_3x_4 \oplus x_1x_2\overline{x}_3\overline{x}_4. \end{aligned}$$

As is shown in Example 5, after linearization, this system is converted into the system $f_\sigma^{(0)}(z)$ and $f_\sigma^{(1)}(z)$, in terms of new variables z_i, $i = 1, 2, 3, 4$ expressed as linear combination of original variables x_i, $i = 1, 2, 3, 4$. It follows that the given system can be represented by a SBDD derived by decomposition in terms of linear combination of variables. Fig. 12 shows SBDD for the given system derived by the linearization method. This SBDD represents the given system in the form of expressions

$$\begin{aligned} f^{(0)} &= (x_1 \oplus x_3) \oplus (\overline{x_1 \oplus x_3})(x_2 \oplus x_4), \\ f^{(1)} &= (x_1 \oplus x_3)(x_2 \oplus x_4). \end{aligned}$$

Compared to the present methods for linear transformation of DDs, an advantage is that the linearization method through total autocorrelation functions provides an exact algorithm to determine linear combination of variables in terms of which the decomposition is performed. At the same time, the method can be used for systems of Boolean functions.

6.2 Reduction of sizes of DDs through the total autocorrelation functions

In what follows, it is assumed that a given system is represented by the integer equivalent function $f(z)$. Since, the reduction of $size(MTBDD(f_z))$ is an NP-complete problem [2], the procedure

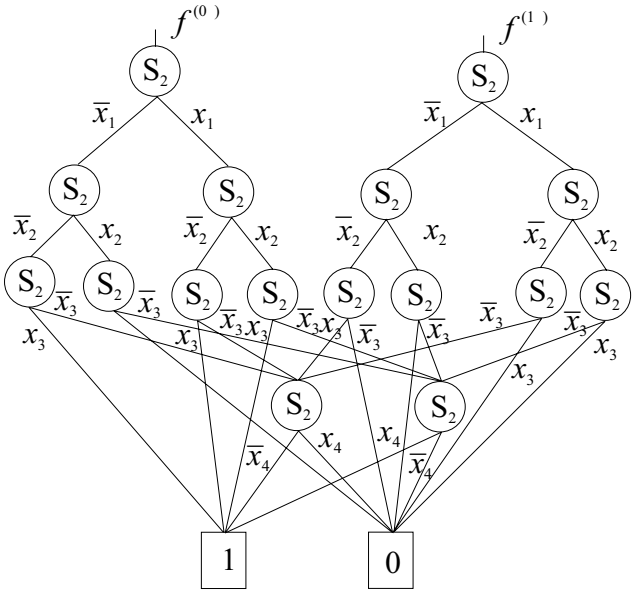

Figure 11: STBDD for the system of functions.

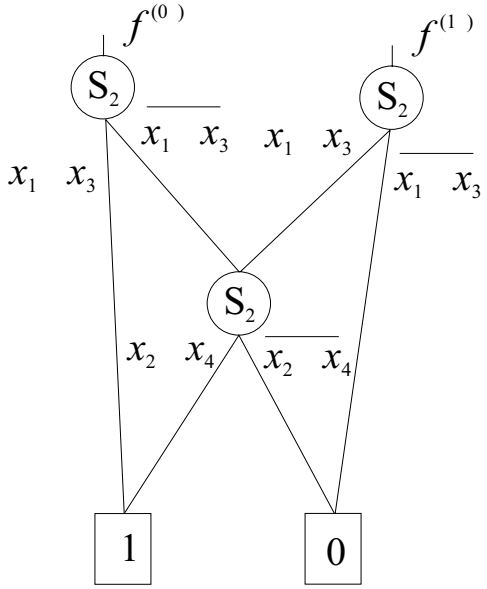

Figure 12: STBDD for the system of functions derived by the linearization method.

described below provides for near minimal solutions. The procedure performs minimization of a MTBDD for a given $f(x_1, \ldots x_n)$ level by level by starting from the bottom level corresponding to the variable x_n. It guarantees the maximal number of equal pairs of values of $f(x)$. Each pair is assigned to a non-terminal node at the level to which x_n is assigned. Thus, for each equal pair we can reduce a node at the lowest level in the MTBDD. Then, we encode pairs of values of $f(x)$ and repeat the procedure at thus produced MTBDD for $n-1$ variables. Under the assumption that we minimized the number of nodes at the previous level, we get a minimal number of nodes a the level $n-1$. The criterion for the minimization is determined by the maximum value of the total autocorrelation function for $f(x)$ and subsequently, by the maximum values of the total autocorrelation functions for integer-valued functions obtained by encoding pairs of values at each level during the implementation of the procedure. The maximum values for total autocorrelation functions may be achieved for different inputs τ. Therefore, the procedure depends on the choice of τ at each level in the sense that for different choices of τ different reduction possibilities at the upper levels may be achieved. At each level i, for a chosen input τ, we perform the reordering of function values of the corresponding function of i variables derived by encoding equal pairs of function values at the preceding levels in the MTBDD. This reordering is determined by the matrix relation

$$\sigma \odot \tau = [0, \ldots, 1]^T,$$

where $\odot$ denotes the multiplication in $GF(2)$, and σ is any $(i \times i)$ matrix over $GF(2)$ satisfying this requirement. Since, the requirement may be achieved for different matrices σ, the method depends also on the choice of a particular matrix σ. However, this is a usual feature of nearly optimal solutions of NP-hard problems.

The implementation of the method is formalized through a procedure denoted in what follows as the K-procedure. Example 11 illustrates the application of the method.

Unlike the method presented in [29], the method presented in this paper can be used for both single output and systems of Boolean functions, and extends the class of permutation matrices for input vectors compared to that allowed in optimization of DDs by variables ordering. Therefore, the method proposed in many cases produces MTBDDs with smaller or at least equal sizes compared to the methods using the variables ordering. However, there are some counter examples. Example 12, Example 13, and Example 14 illustrate that feature as well as dependency on different choices for τ and σ. However, since we use an extended set of permutation matrices for the permutation of elements of the truth-vector, the method proposed may produce solutions which can not be achieved by the variable reordering. Example 15 illustrates this statement.

K-procedure

1. Assign to a given multi-output function $f_0, \ldots, f_{k-1}$, an integer equivalent function $f(z) = \sum_{i=0}^{k-1-i} 2^i f_i$.
2. Denote by R the range of $f(z)$ assigned to f. For each different value $i \in R$, determine characteristic functions

$$f_i(z) = \begin{cases} 1, & \text{if } f(z) = i, \\ 0, & otherwise, \end{cases}$$

3. Calculate the autocorrelation functions B_{f_i} for each $f_i(z)$, and the total autocorrelation function

$$B_f(\tau) = \sum_{i=0}^{k-1} B_{f_i}(\tau).$$

4. Determine the n-tuple of input variables $\tau = (x_1, \ldots, x_n)$, where B_f takes the maximum value, excepting the value $B_f(0)$. If there few possibilities, choose any of them arbitrarily.

5. Determine a matrix σ from the requirement

$$\sigma \odot \tau = (0, \ldots, 0, 1)^T,$$

where $\odot$ denotes the multiplication over $GF(2)$.

6. Determine a function

$$f_\sigma(\sigma \odot x) = f(x).$$

That means, reorder values in a vector $\mathbf{F}$ representing values of f by the mapping $x = (x_1, \ldots, x_n) \to x_\sigma(x_1, \ldots, x_n)$, where $x_\sigma = \sigma^{-1} \odot x$.

7. In a vector $\mathbf{F}_\sigma$ representing the values of f_σ, perform the coding of pairs of adjacent values by assigning the same symbol to the identical pairs. Denote the resulting function of $(n-1)$ variables by $\mathbf{Q}_{n-1}$.

8. Repeat the previous procedure for $i = n-1$ to some k until there are identical pairs in $\mathbf{Q}_k$.

9. Determine MTBDD for f_{σ_k}.

It is possible to show that the K-procedure produces the maximal number of identical pairs or isomorphic subtrees at the each level in the MTBDD provided that the maximal number of identical pairs at the previous level is generated.

Remark 1 *(Lower bound of nodes in MTBDD produced by K-procedure)*
We first minimize the number of non-terminal nodes at the level to which x_n is assigned. Under this condition, we minimize the number of non-terminal nodes at the level corresponding to x_{n-1}, and we continue the procedure until the root node. Each pair of equal values permits reduction of a subtree whose size depends on the level we are processing. The total amount of nodes in the resulting MTBDD(f_σ) is upper bound by

$$L \leq 2^n - 1 - \frac{1}{2} \sum_{i=1}^{n} B_{max}^{(n-i-1)} 2^{i-1}.$$

where B_{max}^k is the maximum value of the total autocorrelation function at the level k.

Remark 2 *For each pair of values determined by K-procedure, a node in the MTBDD($f_{\sigma_i^{-1}}$) may be reduced. It follows, that K-procedure produces the minimal number of different nodes at each level in the MTBDD($f_{\sigma_i^{-1}}$). However, since the pairing of nodes at the i-th level is performed by the total autocorrelation function for Q_i, this is not necessarily the exact minimum of nodes in the MTBDD(f), which may be achieved by an ordering of elements of $\mathbf{F}$ optimal in the sense that produces the MTBDD(f) of the minimum size.*

Remark 3 *A reordering of elements in* $\mathbf{F}$ *can be represented through a permutation matrix* $\mathbf{P}$. *We denote by* $\mathbf{P}_{dv}$, $\mathbf{P}_{FreeBDD}$, *and* $\mathbf{P}_K$, *the set of permutation matrices used in optimization of MTBDDs by variables ordering, in FreeBDDs, and in MTBDDs for* f_σ *determined by K-procedure. Then,*

$$P_{dv} \subset_{FreeBDD} \subset P_K.$$

We explain the K-procedure by the following example.

Example 11 *Table 2 shows an two-output function* $f_0 * f_1$ *of four variables. This function is represented by the integer equivalent function* $f = 2f_0 + f_1$. *The maximum value for the autocorrelation function* B_f *is 8 which corresponds to the n-tuple* $\tau_{max}(1111)$.

Fig. 13 shows MTBDT(f). Fig. 14 shows the corresponding MTBDD(f). We determine a matrix σ_4 *from the requirement*

$$\sigma_4 \odot \tau_{max} = \sigma_4 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

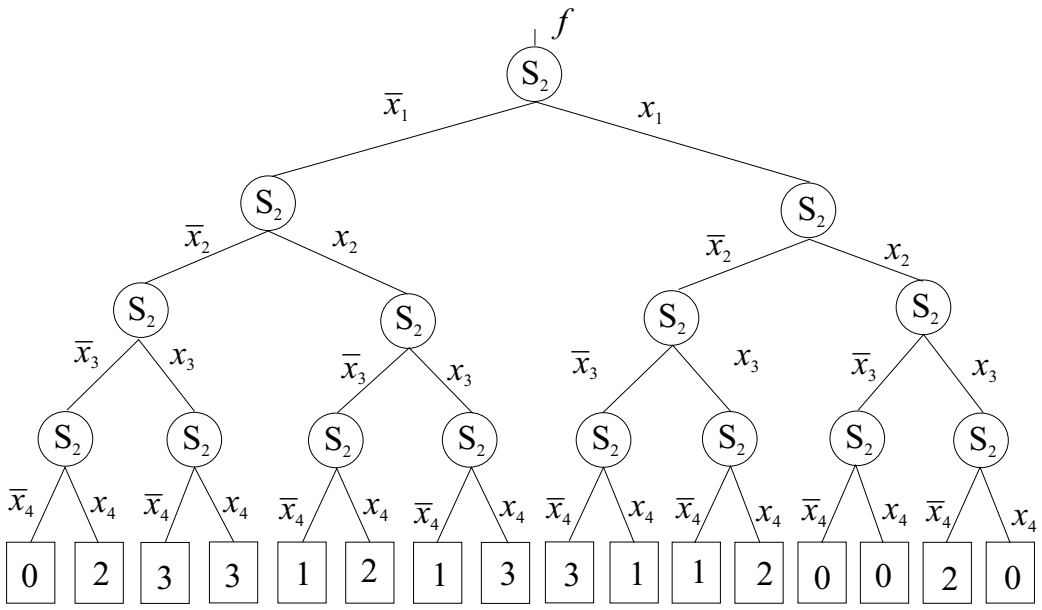

Figure 13: MTBDT for f.

Therefore, among the matrices satisfying this requirement, we arbitrarily choose

$$\sigma_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}.$$

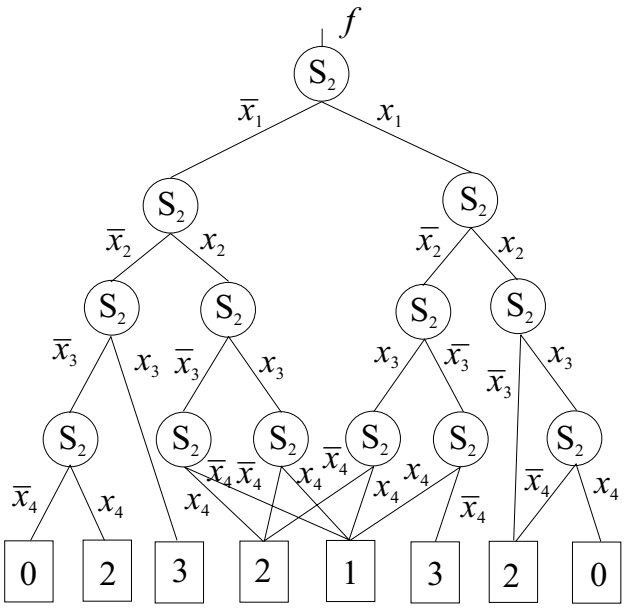

Figure 14: MTBDD for f.

We determine the inverse matrix for σ_4 over $GF(2)$

$$\sigma_4^{-1} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}.$$

Table 3 shows the mapping of vectors of variables in f by using σ_4^{-1}. Then, we determine

$$\sigma_4^{-1}(\mathbf{F}) = [f(0), f(15), f(12), f(3), f(6), f(9), f(10), f(5), f(11), f(4), f(7), f(8), f(13), f(2), f(1), f(14)]^T.$$

For f in Table 2,
$\sigma_4^{-1}(\mathbf{F}) = [00|03|11|12|21|33|03|22]^T$.
We perform the coding of pairs of function values in $\sigma_4(\mathbf{F})$ as follows
$\mathbf{Q} = [0, 4, 1, 5, 6, 3, 4, 2]^T$, *where* $00 = 0$, $03 = 4$, $11 = 1$, $12 = 5$, $21 = 6$, $33 = 3$, $22 = 2$. *Fig. 15 shows MTBDT($f_{\sigma_4^{-1}}$). Fig. 16 shows MTBDT($f_{\sigma_4^{-1}}$) with encoded pairs of equal values for constant nodes. We determine the characteristic functions for each value 0,1,2,3,4,5,6 in $\mathbf{Q}$. They are defined as $f_i = 1$ if $f_i = i$, and 0 otherwise. There is a single non-trivial characteristic function f_4. It is given by*

$$f_4 = [01000010]^T,$$

and its autocorrelation function is given by

$$B_{f_6} = [20000002]^T.$$

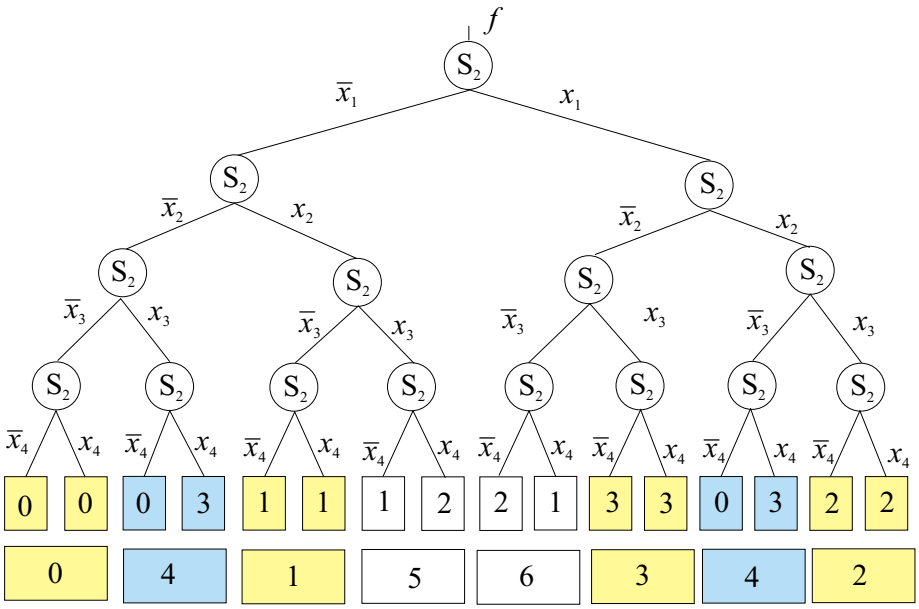

Figure 15: MTBDT for $f_{\sigma_4^{-1}}$.

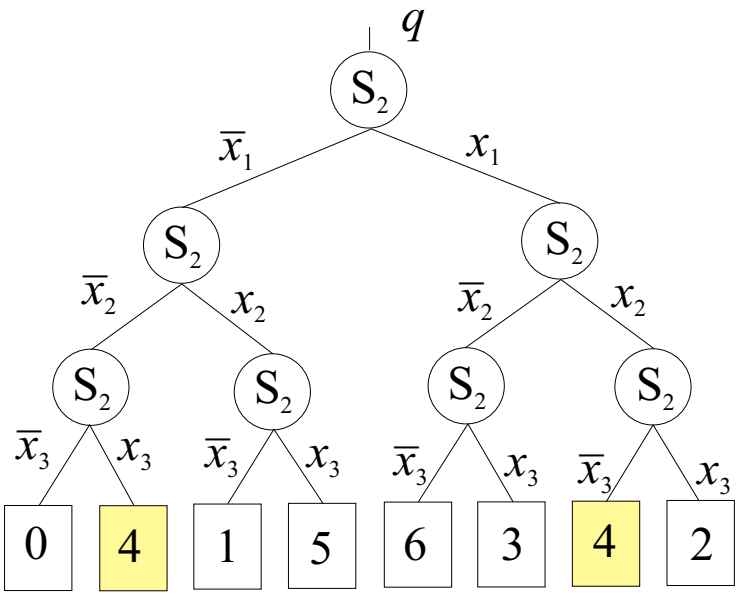

Figure 16: MTBDT for $f_{\sigma_4^{-1}}$ with encoded pair of function values.

The maximum value of the autocorrelation function B_{f_4} is 2 at the n-tuple of variables (111). *We determine a matrix σ_3 from the requirement*

$$\sigma_3 \odot \tau_{max} = \sigma_3 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

Therefore, $\sigma_3 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$. *The inverse matrix is* $\sigma_3^{-1} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$.

Table 4 shows the mapping of vectors of variables in $\mathbf{Q}$ *by using* σ_3^{-1}.

We determine

$$\mathbf{Q}_{\sigma_3^{-1}} = [q(0), q(7), q(1), q(6), q(5), q(2), q(4), q(3)]^T.$$

For f in Table 2, $\mathbf{Q}_{\sigma_3^{-1}} = [0, 2, 4, 4, 3, 1, 6, 5]^T$.

Fig. 17 shows MTBDT($\sigma_3^{-1}(q)$), where q is a function determined by the vector $\mathbf{Q}$. *Fig. 18 shows the corresponding MTBDD($\sigma_3^{-1}(q)$).*

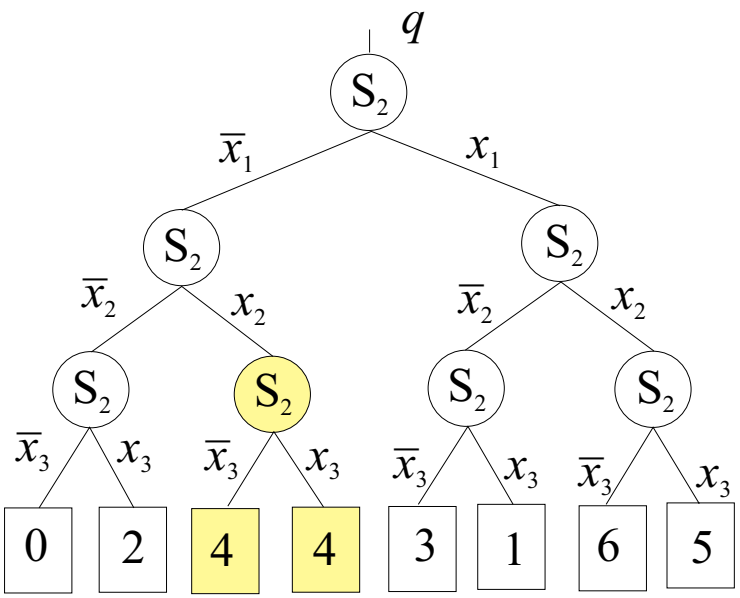

Figure 17: MTBDT for $f_{\sigma_3^{-1}}(\sigma_4^{-1})$.

Therefore,

$$\mathbf{F}_{\sigma_3^{-1}}(\sigma_4^{-1}) = [f(0), f(15), f(1), f(14), f(12), f(3), f(13), f(2), f(7), f(8), f(6), f(9), f(11), f(4), f(10), f(5)]^T.$$

For f in Table 2, $\mathbf{F}_{\sigma_3^{-1}}(\sigma_4^{-1}) = [00|22|03|03|33|11|21|12]^T$.

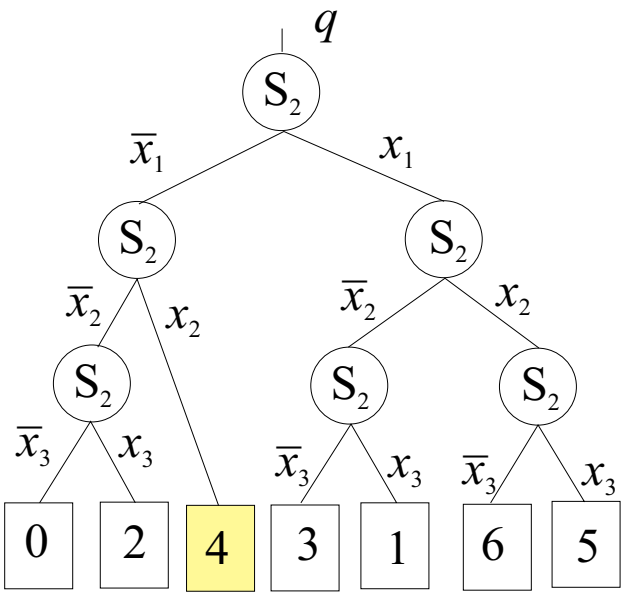

Figure 18: MTBDD for $f_{\sigma_3^{-1}}(\sigma_4^{-1})$.

Fig. 19 shows MTBDT(f_σ) defined by the truth-vector $\mathbf{F}_{\sigma_3^{-1}}(\sigma_4^{-1})$. *Fig. 20 shows the corresponding MTBDD(f).*

Note that the recursive application of σ_4^{-1} *and* σ_3^{-1} *to* f *is identical to the application of a composite mapping*

$$\sigma_{4,3}^{-1} = \sigma_4^{-1} \odot \sigma_{3,1}^{-1},$$

where

$$\sigma_{3,1} = \begin{bmatrix} \sigma_3^{-1} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix},$$

where $\mathbf{0}$ *is* (3×1) *zero matrix.*

Therefore,

$$\sigma_{4,3}^{-1} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} \odot \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}.$$

Table 5 shows the mapping of vectors of variables in f *by using* $\sigma_{4,3}^{-1}$. *It produces the identical permutation of values in* $\mathbf{F}$ *as a recursive application of* σ_4^{-1}, *and* σ_3^{-1} *to* f, *respectively.*

In this example, the size of the MTBDD(f) was reduced from 13 to 9 non-terminal nodes by using the proposed method.

Table 2: Function f and f_σ.

	Function		Characteristic functions				Autocorrelation functions				
x, w	f_0, f_1	$f(x)$	$f_0(z)$	$f_1(z)$	$f_2(z)$	$f_3(z)$	$B_0(z)$	$B_1(z)$	$B_2(z)$	$B_3(z)$	$B(z)$
0	00	0	1	0	0	0	4	4	4	4	16
1	10	2	0	0	1	0	2	0	0	2	4
2	11	3	0	0	0	1	2	2	0	0	4
3	11	3	0	0	0	1	2	2	0	0	4
4	01	1	0	1	0	0	0	0	2	2	4
5	10	2	0	0	1	0	0	0	2	2	4
6	01	1	0	1	0	0	0	0	0	0	0
7	11	3	0	0	0	1	0	0	0	0	0
8	11	3	0	0	0	1	0	0	0	0	0
9	01	1	0	1	0	0	0	0	0	0	0
10	01	1	0	1	0	0	0	0	2	2	4
11	10	2	0	0	1	0	0	0	2	2	4
12	00	0	1	0	0	0	2	2	0	0	4
13	00	0	1	0	0	0	2	2	0	0	4
14	10	2	0	0	1	0	0	2	2	2	4
15	00	0	1	0	0	0	2	2	2	2	8

6.3 Complexity of Determination of f from MTBDD(f_σ)

Remark 4 *The K-procedure performs the decomposition of f with respect to the expansion rule*

$$f = (x_i \oplus \cdots \oplus x_n) f_0 \oplus (\overline{x_i \oplus \cdots \oplus x_n}) f_1,$$

where f_0 and f_1 are co-factor of f for $x_i \oplus \cdots \oplus x_n = 0$, and 1, respectively.

The following example illustrates dependency of the solutions achieved by the method proposed on the choice of the permutation matrices σ and inputs τ where the total autocorrelation functions take the maximum values.

Example 12 *(Dependency on τ)*
Consider a four-variable Boolean function f given by the truth-vector $\mathbf{F} = [0010100010010100]^T$. For this function size$(MTBDD(f)) = 9$.

The maximum value of the autocorrelation function $B_f(\tau) = 14$ for the inputs $\tau = 6$, 9, and 15. For $\tau = 15$ and $\sigma_4(15) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$, we determine $\sigma_4^{-1}(15) = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$. The elements of the truth-vector for f are reordered as $\mathbf{F}_{\sigma_4^{-1}(15)} = [0000000011011100]^T$. We perform encoding of pairs of adjacent values as $\mathbf{Q}_{\sigma_4^{-1}(15)} = [00001210]^T$, where 00=0, 11=1, and 01=2. The maximum value of the total autocorrelation function $B_{\mathbf{Q}_{\sigma_4^{-1}(15)}} = 6$ for the input $2 = (010)$. For $\sigma_3(2) = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$,

Table 3: Mapping of function values by σ_4^{-1}.

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	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
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	0	0	1	1	0	1	0	0	1	1	0	0	1
0	1	1	0	1	0	0	1	0	1	1	0	1	0	0	1
0	1	0	1	1	0	1	0	1	0	1	0	0	1	0	1
0	1	0	1	0	1	0	1	1	0	1	0	1	0	1	0
0	15	12	3	6	9	10	5	11	4	7	8	13	2	1	14

Table 4: Mapping of function values by σ_3^{-1}.

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7
0	0	0	0	1	1	1	1
0	0	1	1	0	0	1	1
0	1	0	1	0	1	0	1
0	1	2	3	4	5	6	7
0	1	0	1	1	0	1	0
0	1	0	1	0	1	0	1
0	1	1	0	1	0	0	1
0	7	1	6	5	2	4	3

Table 5: Mapping of function values by $\sigma_4^{-1} \odot \sigma_3^{-1}$.

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	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
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	0	1	1	0	1	0	0	1	0	1	1	0	1	0
0	1	0	1	1	0	1	0	1	0	1	0	0	1	0	1
0	1	0	1	0	1	0	0	1	0	1	0	1	0	1	0
0	1	1	0	0	1	1	0	1	0	0	1	1	0	0	1
0	15	1	14	12	3	13	2	7	8	6	9	11	4	10	5

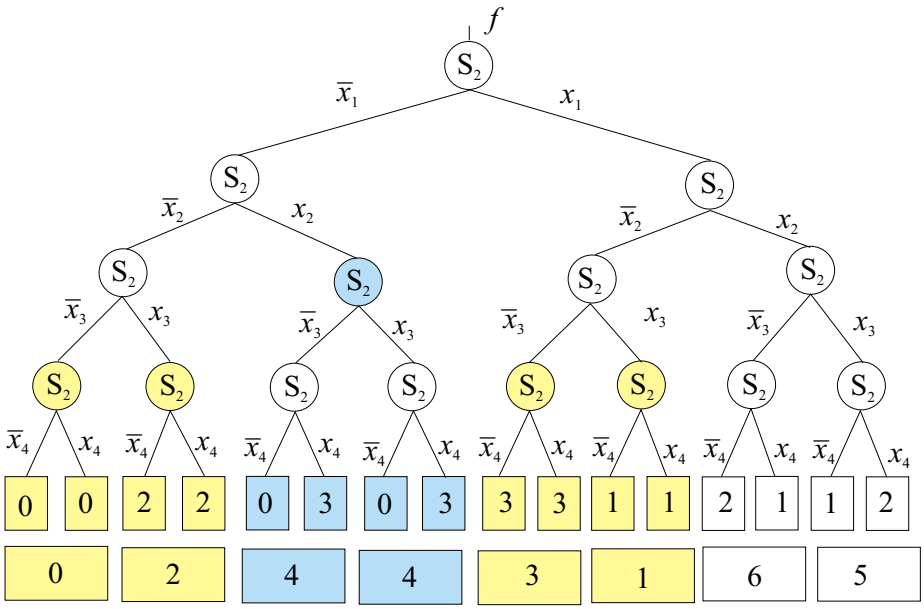

Figure 19: MTBDT for $f_{\sigma_4^{-1}\odot\sigma_3^{-1}}$.

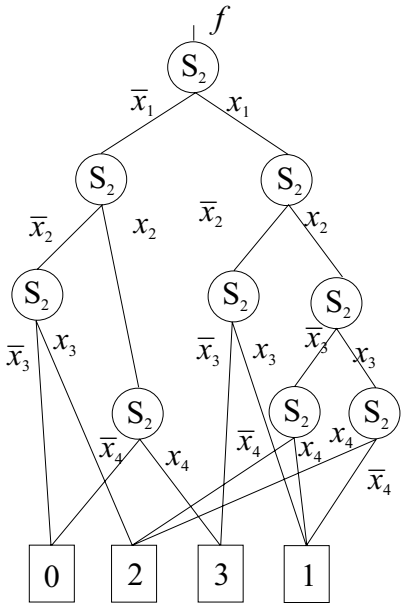

Figure 20: MTBDD for $f_{\sigma_4^{-1}\odot\sigma_3^{-1}}$.

it follows $\sigma_3^{-1}(2) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$, *and the corresponding reordering is* $\mathbf{Q}_{\sigma_4^{-1}(15)\sigma_3^{-1}(2)} = [00001120]^T$, *from where* $\mathbf{F}_{\sigma_4^{-1}(1)\sigma_3^{-1}(2)} = [0000000011110100]^T$. *For thus determined* f_σ, $size(MTBDD(f_\sigma) = 4$.

If for the maximum value of the autocorrelation function $B_f(\tau)$, *we choose the input* $\tau = 6$, *then for* $\sigma_4(6) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$, *we determine* $\sigma_4^{-1}(6) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$. *Thus, we reorder the elements of* $\mathbf{F}$ *for the given* f *as* $\mathbf{F}_{\sigma_4^{-1}(6)} = [0000110000101100]^T$.

For encoding $\mathbf{Q}_{\sigma_4^{-1}(6)} = [00100210]^T$, *where 10=2, the maximum values of the total autocorrelation function of* $\mathbf{Q}_{\sigma_4^{-1}(6)}$ *is 6 for the input* $4 = (100)$.

For $\sigma_3(4) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$, *we determine* $\sigma_3^{-1}(4) = \sigma_3(4)$. *Therefore, the corresponding reordering is* $\mathbf{Q}_{\sigma_4^{-1}(6)\sigma_3^{-1}(4)} = [00110200]^T$, *which produces* $\mathbf{F}_{\sigma_4^{-1}(6)\sigma_3^{-1}(4)} = [0000111100100000]^T$. *For thus determined* f_σ, $size(MTBDD(f_\sigma)) = 5$.

Example 13 *(Dependency on* σ*)*
For f *in the previous example, if we chose for the maximum value of* B_f *the input* $\tau = 15$ *and the matrix* $\sigma_{4,r}(15) = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$, *which requires* $\sigma_{4,r}^{-1}(15) = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$, *then we determine the reordering* $\mathbf{F}_{\sigma_{4,r}^{-1}(15)} = [0000001111100000]^T$. *For encoding* $\mathbf{Q}_{\sigma_{4,r}^{-1}(15)} = [00011200]^T$, *the maximum value of the total autocorrelation function is* $B_{\mathbf{Q}_{\sigma_{4,r}^{-1}(15)}} = 6$ *for the input* $7 = (111)$. *For* $\sigma_3(7) = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$, *we get* $\sigma_3^{-1}(7) = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$, *which induces a reordering* $\mathbf{Q}_{\sigma_{4,r}^{-1}(15)\sigma_3^{-1}(7)} = [00002011]^T$. *From there,* $\mathbf{F}_{\sigma_{4,r}^{-1}(15)\sigma_3} = [000000001011]^T$. *For thus determined* f_σ, $size(MTBDD(f_\sigma) = 5$.

However, if we chose $\sigma_{3,r}(7) = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$, *and the corresponding* $\sigma_{3,r}^{-1}(7) = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}$, *we get the reordering* $\mathbf{Q}_{\sigma_{4,r}^{-1}(15)\sigma_{3,r}^{-1}(7)} = [00110020]^T$, *which produces* $\mathbf{F}_{\sigma_{4,r}^{-1}\sigma_{3,r}^{-1}(7)} = [0000111100001000]^T$. *For thus determined* f_σ, $size(MTBDD(f_\sigma)) = 5$.

The reason for the increased size is in the property that $\sigma_{3,r}(7)$, *unlike* $\sigma_3(7)$, *did not pair together sequences of four 0. This pairing in* $MTBDD(f_\sigma)$ *means assignment of identical subvectors of order four to the same logic value for* x_1. *In this case, that is the negative literal* $\overline{x}_i$. *Due to that, the subtree rooted in the node pointed by* $\overline{x}_1$*in the MTBDD is reduced to a single constant node.*

The method proposed provides reduction of smallest subtrees, since produce pairs of function values. The larger subtrees, which corresponds to the equal subvectors of orders 2^k, $k > 1$ are not taken into account. The method fails in the case when we chose a permutation matrix σ which

does not provide a grouping of isomorphic smallest subtrees into a larger subtree. The Example 14 illustrates that feature of the method. However, searching for the suitable σ matrices, reduces this approach to the reduction of the sizes of DDs to the brute force methods in optimization of DDs by variables ordering.

Example 14 *Consider a function* $f = \overline{x}_2\overline{x}_4 + x_2\overline{x}_3x_4 + x_1x_2x_4 + \overline{x}_1x_3\overline{x}_4$. *The truth-vector for this function is given by* $\mathbf{F} = [1010011010100101]^T$. $size(MTBDD(f)) = 6$ *for thus ordered truth vector for* f.

The maximum value of $B_f = 12$, *which means that we may generate six pairs at the level for* x_4. *Then, the method proposed in this paper, for a few different choices of* τ *and* σ *produces MTBDDs of sizes equal to 7. However, the variable ordering produces the MTBDDs of sizes 5, 6, and 7 [29].*

However, if we first perform encoding $\mathbf{Q} = [22322233]^T$, *where* $10 = 2$, *and* $01 = 3$, *and then apply the method proposed, we get a MTBDD of size 5, by always taking the smallest value for* τ. *This follows form the property that in* $\mathbf{Q}$, *we have five pairs denoted by 2 and three pairs denoted by 3, which permits an immediate reduction of subtrees consisting of three non-terminal nodes.*

A good feature of the method proposed in this paper, is the following. The method uses an extended set of allowed permutation matrices for the inputs, compared to that used in DD optimization by variable ordering. The price for such extension is minor, since the values for f can be easily determined from f_σ assigned to f. Therefore, the method proposed permits to derive efficient solutions which can not be achieved by the variable ordering methods. The Example 15 illustrates that feature of the method.

Example 15 *Consider a function* $f = x_1x_2x_3 + x_2x_3x_4$. *Thus, the truth-vector for this function is* $\mathbf{F} = [0000000100000011]^T$. *The optimization by variable ordering can produce MTBDDs of size 5. However, the method proposed in this paper, produces a MTBDD of size 4 in the following way. The maximum value for* $B_f = 14$ *for the inputs* $\tau = 1, 8, 9$. *For simplicity, we choose* $\tau = 1$, *which implies* $\sigma_4(1)$ *is the identity matrix of order 4, and perform the encoding as* $\mathbf{Q} = [00020001]^T$. *The maximum of the total autocorrelation function for* $\mathbf{Q}$ *is 6 and it is achieved for the input* $40(100]$. *For a matrix* $\sigma_3(4) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$, *which is self-inverse over* $GF(2)$, *we get the reordering* $\mathbf{Q}_{\sigma_3^{-1}} = [00000021]^T$, *which produces* $\mathbf{F}_{\sigma_4^{-1}\sigma_3^{-1}} = [0000000000000111]^T$. *For thus determined* f_σ, $size(MTBDD(f)) = 4$.

7 Closing Remarks

Considerations presented and discussed in this paper permits to state that spectral methods

1. provide for elegant analytical solutions for many problems related to design and testing,
2. can be used efficiently as complementary tools for the existing methods,
3. are efficient when there are compact representations of function describing devices (e.g., digital function generators)
4. are technologically independent.

Among reasons that spectral methods are not still not so widely accepted as they should be, we want to point the following

1. Ignorance, computer engineers do not know groups, fields, Fourier transforms,
2. Area of applicability of spectral methods is difficult to define.
3. Development of spectral CAD tools in very expensive.

References

[1] Agaian, S., Astola, J., Egiazarian, K., *Binary Polynomial Transforms and Nonlinear Digital Filters*, Marcel Dekker, 1995.

[2] Bollig, B. Wegener, I., "Improving the variable ordering of OBDDs is NP-complete", *IEEE Trans. Comput.*, Vol. C-45, No. 9, 1996, 993-1002.

[3] Clarke. E, M., M.C., Millan, K.L., Zhao, X., Fujita, M., "Spectral transforms for extremely large Boolean functions", in Kebschull, U., Schubert, E., Rosenstiel, W., Eds., *Proc. IFIP WG 10.5 Workshop on Applications of the Reed-Muller Expression in circuit Design*, Hamburg, Germany, September 16-17, 1993, 86-90. Workshop Reed-Muller'93, 86-90.

[4] Goel, N., Karpovsky, M.G., "Functional testing of computer hardware based on minimization of magnitude of undetected errors," *IEE J. Comput. Digital Tech.*, Vol. 129, No. 5, September 1982, 169-181.

[5] Günther, W., Drechsler, R., "BDD minimization by linear transforms", in *Advanced Computer Systems*, 1998, 525-532.

[6] Günther, W., Drechsler, R., "Efficient manipupation algorithms for linearly transformed BDDs", *Proc. 4th Int.Workshop on Applications of Reed-Muller Expansion in Corcuit Design*, Victoria, Canada, May 20-21, 1999, 225-232.

[7] Günther, W., Drechsler, R., "Minimization of BDDs using linear transformations based on evolutionary techniques", *Proc. Int. Symp. Circuit and Systems*, 1999.

[8] Hsiao, T.C., Seth, S.C., "Analysis of the use of Rademacher-Walsh spectrum in compact testing", *IEEE Trans, Computers*, Vol. C-33, 1984, 934-937.

[9] Karpovsky, M.G., *Finite Orthogonal Series in the Design of Digital Devices*, John Wiley, 1976.

[10] Karpovsky, M.G., "Error detection in digital devices and computer programs with the aid of linear requrrent equations over finite groups", *IEEE Trans. Computers*, Vol. C-26, 1977, 208-218.

[11] Karpovsky, M.G., "Error detection for polynomial computations", *IEE J. Comput. Digital Tech.*, Vol. 2, 1979, 48-56.

[12] Karpovsky, M.G., "Testing for numerical computations," *IEEE Proc.*, Vol. 127, pt. E, No. 2, March 1980, 69-77.

[13] Karpovksy, M.G., "Error detection for polynomial computations," *IEE J. Comput. Digital Tech.*, C-26, No. 6, June 1980, 523-528.

[14] Karpovsky, M.G., "Detection and location of error by linear inequality checks," *Proc. IEE*, Vol. 129, No. 3, May 1982, 86-92.

[15] Karpovsky, M.G., Levitin, L.B., Vainstein, F.S., "Diagnosis by singature analysis of test responses", *IEEE Trans. Computers*, Vol. C-43, 1994, 141-153.

[16] Karpovsky, M.G., Nagvajara, P., "Optimal codes for the minimax criterion on error detection," *IEEE Trans. on Information Theory*, November 1989.

[17] Karpovsky, M.G., Nagvajara, P., "Optimal robust compression of test responses," *IEEE Trans. on Computers*, Vol. 39, No. 1, 1990, 138-141.

[18] Karpovsky, M.G., Roziner, T., Moraga, C., "Error detection in multiprocessor systems and array processors," *IEEE Trans. on Computers*, Vol. 44, No. 3, March 1995, 383-394.

[19] Karpovsky, M.G., Trachtenberg, E.A., "Linear checking equations and error correcting capability for computation channels", *P. IFIP Congres*, North Holland Publ. Co., 1977.

[20] Karpovsky, M.G., Trachtenberg, E.A., "Fourier transforms over finite groups for error detection and error correction in computation channels", *Inf, and Control*, 1979, 40, 335-358.

[21] Lupanov, O.B., "On a method of network synthesis", *Izv. Vuzov, Radiofizika*, No. 1, 1958, 43-45.

[22] MacWilliams, F.J., Sloane, N.J.A., *Theory of Error-Correcting Codes*, North Holland Publishing, 1978.

[23] McCluskey, E.J., *Digital Test Principles*, Stanford Logical Systems Insitute, 1991.

[24] Meinel, C., Somenzi, F., Theobald, T., "Linear shifting of decision diagrams", *Prpc. Design Automation Conference*, 1997, 202-207.

[25] Moraga. C., "Advances in Spectral Techniques", Berichte zur angewandten Inforamtik, Universität Dortmund, 1998.2, ISSN 0946-2341.

[26] Moraga, C., Heider, R., "Tutorial review on applications of the Walsh transform in switching theory", *Proc. First Int. Workshop on Transforms and Filter Banks*, TICSP Series # 1, June 1998, 494-512.

[27] Muzio. J.C., Miller, D.M., "Spectral techniques for fault detection", *Proc. 12th Int. Symp. Fault Tolerant Computing*, 1982, 297-302.

[28] Peterson, W.W., Weldon, E.J., *Error Correcting Codes*, MIT Press, Cambridge, Mass., 2nd edn. 1972.

[29] Rice, J., Serra, M., Muzio, J.C., "The use of autocorrelation coefficients for variable ordering for ROBDDs", *Proc. 4th Int. Workshop on Applications of Reed-Muller Expansion in Circuit Design*, Victoria, Canada, August 20-21, 1999, 185-196.

[30] Roziner, T., Karpovsky, M.G., "Multidimensional Fourier transforms by systolic architectures," *Journal of VLSI Signal Processing*, No. 4, 1992, 343-354.

[31] Sasao, T., *Switching Theory for Logic Synthesis* Kluwer Academic Publishers, 1999.

[32] Sasao, T., Fujita, M., (eds.), *Representations of Discrete Functions*, Kluwer, 1996.

[33] Shannon, C.E., "The synthesis of two-terminal switching circuits", *Bell System Tech. J.*, 28, No. 1, 1949.

[34] Sholomov, L.A., "Complexity criteria for Boolean functions", *Problemy Kibernetiki*, No. 17, 1966, (in Russian).

[35] Smith, J.E., "Measures of the effectiveness of fault signature analysis", *IEEE Trans. Computers*, Vol. C-29, 1980, 510-514.

[36] Stanković, R.S., *Spectral Transform Decision Diagrams in Simple Questions and Simple Answers*, Nauka, Belgrade, 1998.

[37] Stanković, R.S., Falkowski, B.J., "FFT and decision diagrams based methods for calculation of spectral transforms", *Proc. IEEE Int. Conf. on Informatics, Communications and Signal Processing*, Singapore 1997, Vol. 1, 241-245.

[38] Stanković, R.S., Sasao. T., "Decision diagrams for representation of discrete functions: uniform interpretation and classification", *Proc. ASP-DAC'98*, Yokohama, Japan, February 13-17, 1998.

[39] Stanković, R.S., Sasao, T., Moraga, C., "Spectral transform decision diagrams" in: [32], 55-92.

[40] Susskind, A.K., "Testing by verifying Walsh coeffcients", *Proc. 11th Int. Symp. Fault Tolerant Computing*, 1981, 206-208.