Determining the Number of Paths in Decision Diagrams by Using Autocorrelation Coefficients

Osnat Keren, Ilya Levin, and Radomir S. Stanković

***Abstract*—This paper deals with the number of paths in multiterminal binary decision diagrams (MTBDDs) and shared binary decision diagrams (SBDDs) representing a set of Boolean functions. It is shown that the number of paths in an MTBDD (SBDD) can be uniquely determined by values of specific weighted-autocorrelation coefficients. An analytical expression for the number of paths as a linear function of the values of the weighted-autocorrelation coefficients is presented. Based on this expression, a method of minimization of the number of paths is proposed. The method is based on replacing the initial set of input variables with their linear combinations. By using this method, a deterministic paths-reduction procedure, which provides MTBDDs and SBDDs with a reduced number of paths, is presented. The efficiency of the suggested approach is demonstrated on benchmark functions.**

***Index Terms*—Autocorrelation function, binary decision diagram (BDD), linear transform, logic synthesis, spectral technique.**

I. Introduction

THE EXPONENTIAL scaling in sizes of lithography-based very-large scale integrated technology and the increase in processing power of computer circuits pushed the technology to its fundamental physical limits and opened the door for new nanotechnologies [23], [29]. The micro-miniaturization complicates the task of design, verification, and testing. These tasks become even more complex and time consuming when the logic states are determined by the Coulomb law [3], [29], i.e., when the logic state of a certain cell is determined by the states of its neighboring cells. In such cases, it is not sufficient to verify the correct functionality of the processing elements, since the value of an output signal is not determined solely by the functionality of the processing elements along a single path, rather, it is also affected by the behavior of adjacent processing elements and adjacent wires. Consequently, the design verification and validation requires checking the correct propagation of the signals through the implemented circuit in all possible paths. Therefore, the number of paths can be used as a complexity measure for design verification, validation and testing. This paper deals with minimization of the number of paths. It is focused on circuits whose underlaying structures are decision diagrams.

A binary decision diagram (BDD) is considered as an optional underlying structure in nanotechnology [31]. A BDD is a form of a compact representation of a single Boolean function as an acyclic directed graph. The most popular representations of logic units of n inputs and k outputs are shared BDDs (SBDDs) and multiterminal BDDs (MTBDDs) [28]. The minimization of the number of nodes in BDDs was extensively studied [4], [5], [7], [11], [13], [18]–[20], [26], as well as other characteristics of BDDs, such as the average path length (APL) and the number of paths [2], [8]–[10], [14], [22], [27].

It is well known that the number of BDD paths depends on the order of the input variables. There are functions for which a reduction in the number of nodes may increase the number of paths [6]. Therefore, existing techniques targeting BDD size reduction may not be suitable for paths reduction. In this paper, we discuss methods that are aimed to reduce the number of paths rather than the number of nodes.

Existing techniques for minimization of the number of paths in decision diagrams can be divided into two classes: dynamic approaches and static (analytic) approaches. Dynamic approaches work directly on the BDD; they are based on swapping and modified sifting of the input variables with acceptance criteria for a minimal number of paths [8], [10]. In general, the performance of the dynamic procedure may depend on the number of iterations that the minimization procedure performs. This relationship between the performance of a dynamic minimization procedure and the available resources (memory size and time restriction) does not exist in analytic procedures.

An analytic approach provides the exact solution of the minimization problem. It defines analytically the ordering of the input variables or linear combination of the input variables (also called linearization) by considering the characteristics of a Boolean function such as its autocorrelation function coefficients. This paper presents an analytic technique. To the best of our knowledge, this is the first paper that deals with an analytic approach for the paths-reduction problem.

The main contribution of this paper is the introduction of an explicit expression for the number of paths in decision diagrams as a *linear function* of so-called weighted autocorrelation coefficients associated with the basis vectors that span the finite Galois field $GF(2^n)$. This relation between

Manuscript received December 10, 2009; revised March 26, 2010 and July 15, 2010; accepted July 15, 2010. Date of current version December 17, 2010. This paper was recommended by Associate Editor V. Bertacco.

O. Keren is with the School of Engineering, Bar-Ilan University, Ramat-Gan 52900, Israel (e-mail: kereno@eng.biu.ac.il).

I. Levin is with the School of Education, Tel-Aviv University, Tel-Aviv 69978, Israel (e-mail: ilia1@post.tau.ac.il).

R. S. Stanković is with the Department of Computer Science, Faculty of Electronics, Niš 18000, Serbia (e-mail: radomir.stankovic@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2010.2069290

the number of paths and the weighted autocorrelation coefficients is then used to define an analytic paths-reduction procedure. The suggested procedure minimizes the number of paths in MTBDDs by choosing a set of basis vectors corresponding to the maximal weighted autocorrelation coefficients. The MTBDD with the reduced number of paths is obtained by using the ordered set of n basis vectors which corresponds to input variables or linear combinations of the variables.

The initial idea of this paper and first preliminary experimental results were presented in the *7th International Workshop on Boolean Problems*, September 2006, Freiberg, Germany [16]. This paper presents a comprehensive study of the subject.

This paper is organized as follows. Section II includes mathematical background and defines the weighted autocorrelation function. Section III presents methods for counting the number of paths and shows that the number of paths depends on the value of the weighted autocorrelation coefficients. In Section IV, the relationship between the number of paths and the number of nodes is discussed. Section V describes the paths-reduction procedure for MTBDDs and SBDDs and analyzes the computational complexity, the dependency on the chosen basis vectors, and the dependency on the cost function. Experimental results on standard benchmark functions are presented in Section VI. The conclusions summarizing the results are presented in Section VII.

II. Mathematical Background

In this paper, the domain and the range of the multioutput function of n-input and k-output are defined as the finite fields $GF(2^n)$ and $GF(2^k)$, respectively, and, an assignment of the n inputs is viewed as a binary vector $x = (x_{n-1}, \ldots, x_1, x_0)$. An element of the field $GF(2^n)$ is represented as a linear combination of n basis vectors $\{\delta_i\}_{i=0}^{n-1}$ with the coefficient vector x, where δ_i is the binary vector corresponding to 2^i. The set of δ_i's is called the initial basis of $GF(2^n)$. Clearly, any set of n independent elements of $GF(2^n)$ forms a basis. The truth table of a Boolean function defines the mapping between the coefficient vector that specifies an element in $GF(2^n)$ to an element of $GF(2^k)$. This mapping depends on the set of basis vectors.

Example 1: Consider the field $GF(2^3)$ and two sets of basis vectors: the first set is the initial set of basis vectors $\{\delta_i\}_{i=0}^{2}$ which are the columns of the 3×3 identity matrix I; the second set of vectors is $\{\tau_i\}_{i=0}^{2}$ which are the columns of a nonsingular matrix T

$$T = (\tau_2, \tau_1, \tau_0) = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

An element, say the binary tuple (110) also referred as α_6, corresponds to the following linear combination of the initial basis vectors:

$$\alpha_6 = (\delta_2, \delta_1, \delta_0) \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = I \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

equivalently, this element can be represented as

$$\alpha_6 = T \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}.$$

TABLE I

Truth Tables of $f_I(x_2, x_1, x_0)$ and $f_T(z_2, z_1, z_0)$

x	Element	f_I
000	α_0	00
001	α_1	01
010	α_2	00
011	α_3	01
100	α_4	00
101	α_5	01
110	α_6	10
111	α_7	11

z	Element	f_T
000	α_0	00
001	α_1	01
010	α_6	10
011	α_7	11
100	α_4	00
101	α_5	01
110	α_2	00
111	α_3	01

Therefore, the element α_6 can be addressed in two ways: the first way is by using the coefficient vector $x = (x_2x_1x_0)$ which corresponds to the initial basis, i.e., $x = (110)$; the second way is by using the coefficient vector $z = (z_2z_1z_0)$ which corresponds to the basis defined by the columns of T, i.e., $z = (010)$. Clearly, the matrix T defines a linear transformation between the coefficient vectors $x = Tz$ or $z = T^{-1}x = \sigma x$.

A function from $GF(2^n)$ to $GF(2^k)$ maps the 2^n elements of $GF(2^n)$ to elements of $GF(2^k)$. The mapping between the coefficient vectors depends on the basis vectors that span the field.

Example 2: The left part of Table I shows the truth table of the function $f_I(x_2, x_1, x_0)$ defined by the initial set of basis vectors. The right part of the table corresponds to the function f_T, with the basis vectors defined by the columns of the matrix T (specified in Example 1). These functions define the same mapping from $GF(2^n)$ to $GF(2^k)$. For example, the element α_6 is mapped to the output vector (10), i.e., $f(\alpha_6) = f_I(110) = f_T(010)$.

Definition 1 (Characteristic Function): Let $u \in GF(2^k)$. The characteristic function f_u associated with the symbol u is defined as

$$f_u(x) = \begin{cases} 1 & if \quad f(x) = u, \\ 0 & \text{otherwise} \end{cases}. \tag{1}$$

The autocorrelation function of f_u is

$$R_u(\tau) = \sum_{x \in GF(2^n)} f_u(x) f_u(x \oplus \tau) \quad \tau \in GF(2^n) \tag{2}$$

where the $\oplus$ sign stands for addition in the finite field $GF(2^n)$, that is, a bitwise XOR operation.

Property 1: Let $R_u(\tau)$ be the autocorrelation function of f_u. Denote by $R_{\bar{u}}(\tau)$ be the autocorrelation function of $\bar{f}_u$—the complement of f_u. Then

$$\begin{aligned} R_{\bar{u}}(\tau) &= \sum_{x \in GF(2^n)} \bar{f}_u(x) \bar{f}_u(x \oplus \tau) \\ &= \sum_{x \in GF(2^n)} (1 - f_u(x))(1 - f_u(x \oplus \tau)) \\ &= 2^n - 2R_u(0) + R_u(\tau). \end{aligned}$$

We define a *weighted* autocorrelation function, $R^w(\tau)$, as follows.

Definition 2: The *weighted* autocorrelation function of the function f is

$$R^w(\tau) = \sum_{u \in GF(2^k)} w_u R_u(\tau), \quad \tau \in GF(2^n) \tag{3}$$

where $\{w_u\}_{u \in GF(2^k)}$ is a set of weights.

In this paper, the weights are real numbers, i.e., $w_u \in \mathcal{R}$. The *total-autocorrelation* function, defined in [12], is a special case where all the weights are equal to 1. The total-autocorrelation function was used in [14] for analytical calculation of the APL of an MTBDD. In this paper, we use the *weighted* autocorrelation function to calculate analytically the number of paths. We show that the weights $\{w_u\}_{u\in GF(2^k)}$ can be chosen in a way that the autocorrelation values at a certain τ will reflect the number of paths.

Example 3: Consider the function f_I specified in Table I. There are four distinct output combinations which correspond to four characteristic functions, f_{00}, f_{01}, f_{10}, and f_{11}. Fig. 1(a) shows the characteristic function $f_{01}(x) = f_{01}(x_2, x_1, x_0)$ as a 3-D cube. The eight nodes of the cube correspond to the eight elements in $GF(2^3)$. The bold nodes represent the input vectors for which f_{01} equals one. The *weight* of $f_{01}(x)$, i.e., the number of "ones" in $f_{01}(x)$, equals three. Fig. 1(b) shows the characteristic function f_{01} shifted by $\tau = (110)$, that is

$$f_{01}(x \oplus \tau) = f_{01}(x_2 \oplus 1, x_1 \oplus 1, x_0 \oplus 0) = f_{01}(\bar{x}_2, \bar{x}_1, x_0).$$

The autocorrelation function $R_{01}(\tau)$ at $\tau = (110)$ is

$$\begin{aligned} R_{01}(110) &= \sum_{x\in GF(2^3)} f_{01}(x) f_{01}(x \oplus (110)) \\ &= 0\cdot 0 + 1\cdot 0 + 0\cdot 0 + 1\cdot 1 \\ &\quad +0\cdot 0 + 1\cdot 1 + 0\cdot 0 + 0\cdot 1 \\ &= 2. \end{aligned}$$

The autocorrelation value at $\tau = (110)$ equals the weight of the product function obtained by multiplying the two functions $f_{01}(x)$ and $f_{01}(x \oplus (110))$ [see Fig. 1(c)]. Clearly, the complexity of calculating a single autocorrelation value according to its definition is exponential in the number of inputs. Efficient methods for the calculation of the autocorrelation values with an acceptable complexity are discussed in Section V-E. The autocorrelation values $R_{01}(\tau)$ are shown in Fig. 2. The horizontal axis is the value of τ when represented as an integer number, for example, (110) = 6. Notice that for all $\tau \in GF(2^3)$, the weight of the (product) function $f_{01}(x)f_{01}(x \oplus \tau)$ is smaller or equal to the weight of $f_{01}(x)$. Therefore, $\tau = (000)$ attains the maximal autocorrelation value.

The autocorrelation values for the characteristic functions and the weighted autocorrelation functions if all the weights are equal to 1, are

$$\begin{aligned} R_{00} &= [3, 0, 2, 0, 2, 0, 2, 0] \\ R_{01} &= [3, 0, 2, 0, 2, 0, 2, 0] \\ R_{10} &= [1, 0, 0, 0, 0, 0, 0, 0] \\ R_{11} &= [1, 0, 0, 0, 0, 0, 0, 0] \\ R^w &= [8, 0, 4, 0, 4, 0, 4, 0]. \end{aligned} \quad (4)$$

Notice that $R^w(0) = 2^3$. ■

A multioutput function of k-outputs can be represented by an MTBDD, which is a directed acyclic graph that has at most 2^k terminal nodes (leaves) [31]. Each nonterminal node in the MTBDD is associated with a variable. A variable represents a coefficient of a basis vector. The coefficients of the n basis

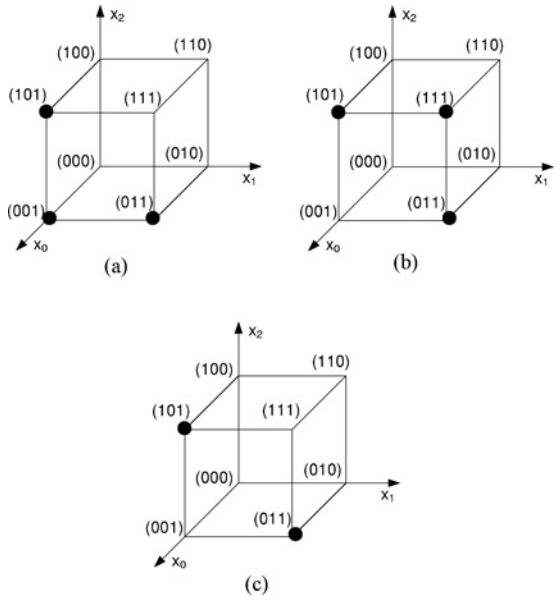

Fig. 1. Characteristic function $f_{01}(x)$ is shown in (a) and the function $f_{01}(x \oplus \tau)$ for $\tau = (110)$ in (b). The product $f_{01}(x) \cdot f_{01}(x \oplus \tau)$ is shown at the bottom of the figure (c).

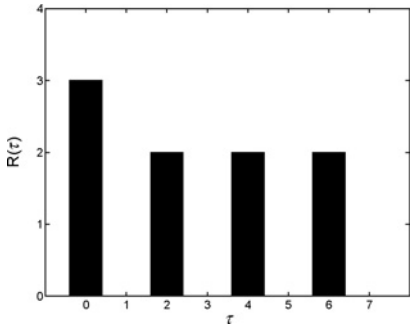

Fig. 2. Autocorrelation function $R_{01}(\tau)$ for Example 3.

vectors appear in some fixed order and once at each path. Usually, the initial basis is used; the root of the tree is x_{n-1} (the coefficient of δ_{n-1}), and a node associated with x_i descends from x_{i+1}. If the order of the basis vectors is predetermined, it is possible to reduce the number of nodes by: 1) eliminating nodes whose outgoing edges point to the same subtree, and 2) sharing all equivalent subtrees. Such reduced ordered MTBDD is called a reduced ordered MTBDD (ROMTBDD). For simplification, when it is clear from the context we omit the prefix, hence ROMTBDD is referred to as MTBDD.

A multiterminal binary decision tree (MTBDT) is derived from a complete MTBDT by eliminating nodes whose edges are pointing to the same subtrees, but without sharing equivalent subtrees that are not rooted from the same node.

An ordered MTBDD whose root variable is x_{n-1} and x_i descends from x_{i+1}, is referred to as an MTBDD with a *natural* ordering of the input variables. It is possible to reduce size, the number of paths, and APL in an MTBDD with natural ordering, by reordering basis vectors or by replacing them with linear combinations (EXOR sum) of subset of basic vectors. The latter operation is called *linearization*.

The following property shows the relationship between the autocorrelation values and the number of nodes that can be eliminated from the bottom of the MTBDT:

Property 2: The number of nodes with a node variable x_0 that can be eliminated from a MTBDT with a natural ordering of the input variables equals $R(\delta_0)/2$.

Proof: A node with a node variable x_0 can be eliminated if its outgoing edges point to leafs that carry equal value. In other words, let $Y \in GF(2^{n-1})$ stand for the realization of the variables $(x_{n-1}, \ldots, x_1)$, then, a node at the end of the path defined by Y can be eliminated if $f(Y, 0) = f(Y, 1)$. The number N_0 of nodes with a node variable x_0 that can be eliminated is

$$\begin{aligned} N_0 &= |\{Y | f(Y,0) = f(Y,1), Y \in GF(2^{n-1})\}| \\ &= \sum_{u \in GF(2^k)} \left| \left\{ Y \;\middle|\; \begin{array}{l} f_u(Y,0) = f_u(Y,1) = 1, \\ Y \in GF(2^{n-1}) \end{array} \right\} \right| \\ &= \frac{1}{2} \sum_{u \in GF(2^k)} \sum_{x \in GF(2^n)} f_u(Y, x_0) f_u(Y, \bar{x}_0) \\ &= \frac{1}{2} \sum_{u \in GF(2^k)} \sum_{x \in GF(2^n)} f_u(x) f_u(x \oplus \delta_0) \\ &= \frac{1}{2} \sum_{u \in GF(2^k)} 1 \cdot R_u(\delta_0) = \frac{1}{2} R^w(\delta_0) \end{aligned}$$

where all the weights, w_u, are equal to one. ■

Example 4: Consider the function specified in Table I. The value of the autocorrelation function at δ_0 equals zero [see (4)]. Therefore, no nodes can be eliminated from the bottom of the corresponding MTBDT.

The *pairing* operation, defined below, will be used in the following sections for analysis and minimization of MTBDDs. Pairing an MTBDD is eliminating the lower level by encoding pairs of leaves that are rooted from the same node. This operation increases the number of distinct paired leaves. A paired MTBDD represents a paired multioutput function of $n-1$ variables.

Definition 3 (Pairing): Let $f(x_{n-1}, \ldots, x_1, x_0)$ be a function from $GF(2^n)$ to $GF(2^k)$. The paired function g: $GF(2^{n-1}) \rightarrow GF(2^k)^2$ with respect to x_0 is $(y_1, y_0) = g(x_{n-1}, \ldots, x_1)$, where $y_1 = f(x_{n-1}, \ldots, x_1, 1)$ and $y_0 = f(x_{n-1}, \ldots, x_1, 0)$.

Recall that the field $GF(2^k)^2$ can be identified with the field $GF(2^{2k})$. Thus, after applying the pairing i times the paired function, denoted by f^i, is $f^i : GF(2^{n-i}) \rightarrow GF(2^{k2^i})$ for $0 \leq i < n$.

Example 5: The pairing of the MTBDD representing $f_I(x_2, x_1, x_0)$ specified in Table I is illustrated in Fig. 3. To simplify the presentation, we show the MTBDT and not the MTBDD. In addition, the outputs of f_I, which are binary vectors, are represented as integers, i.e., $(00) = 0, (01) = 1, (10) = 2$, and $(11) = 3$. The pairing is done in two steps: the initial function $f_I(x_2, x_1, x_0)$ (denoted as f^0) is paired with respect to x_0. This results a paired function $f^1(x_2, x_1)$, whose range is $GF(2^2)^2$, or equivalently, $GF(2^4)$. Then, the function f^1 is being paired with respect to x_2. This results a paired function $f^2(x_2)$ whose range is $GF(2^2)^4$, equivalently, $GF(2^8)$. Notice that additional paring (with respect to x_2) will result in a single leaf which is the truth vector of f_I.

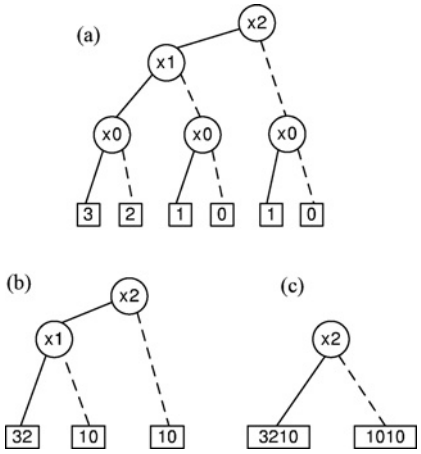

Fig. 3. Pairing of the function $f_I(x_2, x_1, x_0)$ in Table I. (a) MTBDT corresponding to f_I. (b) MTBDT after pairing with respect to x_0. (c) MTBDT after pairing with respect to x_0 and x_1.

III. Determining the Number of Paths in MTBDD

In this section we present methods for counting and reducing the number of paths in a multiterminal BDD. However, the analysis as well as the minimization procedure are also applicable to shared BDDs.

For a fixed order of variables, the number of paths in an MTBDD and in the corresponding MTBDT is the same. Consequently, the number of paths in a given MTBDD equals the number of edges connected to the terminal nodes of the equivalent MTBDT. Recall that a multioutput function of k outputs can also be represented by an SBDD which is a set of k BDDs that share nodes. The number of paths in an SBDD equals the sum of paths in each BDD.

Example 6: Consider the 4-input 4-output logic function presented in [28]

$$\begin{aligned} f_0 &= x_3 x_2 + (\bar{x}_3 + \bar{x}_2) x_1 x_0 \\ f_1 &= x_3 x_2 \\ f_2 &= x_3 x_2 + (\bar{x}_3 + \bar{x}_2) x_1 \\ f_3 &= x_3 x_2 + (\bar{x}_3 + \bar{x}_2) x_0. \end{aligned}$$

Fig. 4 shows the SBDD and the MTBDD of the same logic function. The MTBDD has nine paths whereas the SBDD has 20 paths that are the sum of the number of paths in the four BDDs.

Each node in a binary decision tree has a single ingoing edge. Therefore, the number of paths that pass through a node is equal to the number of paths in the sub-BDT rooted from that node. Consequently, the number of paths in an MTBDD can be calculated recursively starting from the bottom of the MTBDT. In what follows we show how the calculation is performed.

The leaves of the MTBDT corresponding to the paired function f^i represent the output values u, $u \in GF(2^{k2^i})$. The symbol u is a pair (concatenation) of symbols (u_m, u_l), $u_m, u_l \in GF(2^{k2^{i-1}})$. It has the following property.

Property 3: Let the symbol u ($u \in GF(2^{k2^i})$) at the ith level be assigned with a weight c_u^i, where the value of c_u^i is the number of paths in the sub-BDT that the leaf u represents.

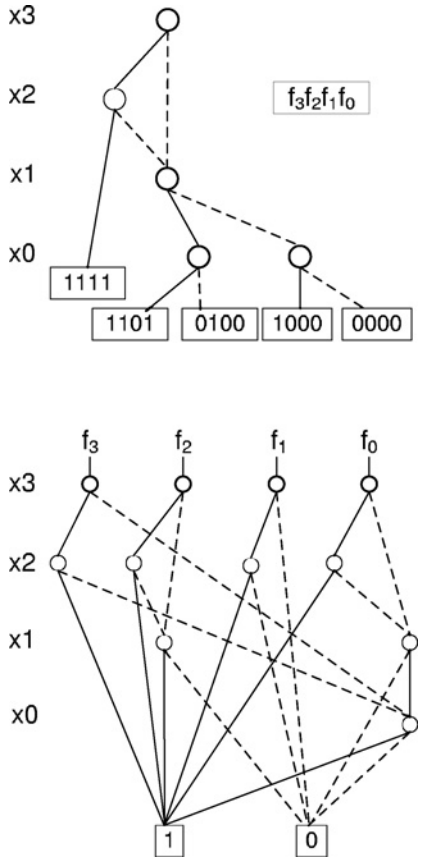

Fig. 4. (Top) MTBDD and (bottom) SBDD of the logic function presented in Example 6.

Let each output value of the original function $f = f^0$ be assigned with a weight $c_u^0 = 1$. Then, for $i > 0$, the weights are

$$c_u^i = \begin{cases} c_{u_m}^{i-1} + c_{u_l}^{i-1} & u_m \neq u_l \\ c_{u_m}^{i-1} & u_m = u_l \end{cases} \tag{5}$$

where $u = (u_m, u_l)$, and u_m and u_l are elements of $GF(2^{k2^{i-1}})$.

This property is illustrated in Fig. 5. Note that the number of distinct values of u's of the paired function f at the level i is at most $\min(2^{n-i}, 2^{k2^i})$. At the upper level, there is a single leaf u, which represents the truth vector F of the function f, namely, $u = (f(2^n - 1), \ldots f(2), f(1), f(0)) = F$. This property can be formulated as follows.

Property 4: The number of paths in an MTBDD equals c_F^n.

Example 7: The number of paths in the MTBDD of f_I (specified in Table I) is calculated as shown in Table II. For each level i ($0 \leq i \leq 3$), the second column of the table shows the truth vector of the paired function f^i, and the right column shows the weight assigned to the leaves in the BDT of the paired function. The first row (level 0) corresponds to the original function, i.e., $f^0 = f_I$. Following Property 4, the number of paths is equal to the weight c_F^3 associated with the single leaf of the third level ($i = 3$), namely, $c_{32101010}^3 = 6$.

In this paper, we apply a different approach for counting the number of paths. This approach allows us to derive an explicit expression for the number of paths as a linear function of the autocorrelation values. Based on the analytic expression, we

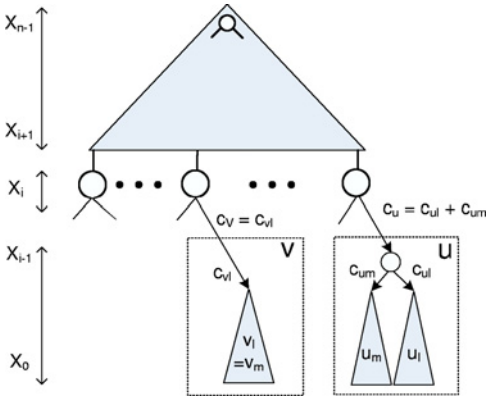

Fig. 5. Illustration—recursive path counting.

TABLE II
STEPS IN THE CALCULATION OF THE NUMBER OF PATHS IN THE MTBDD OF f_I

Level	Truth Vector of f^i : $[f^i(0), \ldots, f^i(2^{n-i}-1)]$	Weights
0	[0, 1, 0, 1, 0, 1, 2, 3]	$c_0^0 = c_1^0 = c_2^0 = c_3^0 = 1$
1	[10, 10, 10, 32]	$c_{10}^1 = c_1^0 + c_0^0 = 2$ $c_{32}^1 = c_3^0 + c_2^0 = 2$
2	[1010, 3210]	$c_{1010}^2 = c_{10}^1 = 2$, $c_{3210}^2 = c_{32}^1 + c_{10}^1 = 4$
3	[32101010]	$c_{32101010}^3 = c_{3210}^2 + c_{1010}^2 = 6$

will introduce a *deterministic* path minimization procedure. The main idea behind this approach is to calculate the number of paths *per level* rather than the number of paths per node. The calculation is performed by using the weighted autocorrelation functions coefficients.

Let f_u^i be a characteristic function of the paired function f^i at the level i. The autocorrelation function of f_u^i is

$$R_u^i(\tau) = \sum_{x \in GF(2^{n-i})} f_u^i(x) f_u^i(x \oplus \tau) \quad \tau \in GF(2^{n-i}).$$

Recall that a finite field $GF(2^w)$ contains the finite field $GF(2)$. Indeed, a field $GF(2^w)$ is an extension of the field $GF(2)$. The element δ_0 is the *unit* element of $GF(2)$ and hence it is also the unit element in $GF(2^w)$. When δ_0 is referred to as an element of $GF(2^w)$, it is represented as a binary vector $(0, \ldots 0, 1)$ of length w. In this paper, we address δ_0 as a basis vector that spans the domain of the function. Being a basis vector, δ_0 is associated with a variable. For example, the variable that determines the coefficient of δ_0 in the paired function $f_u^i(x_{n-1}, \ldots, x_i)$ is x_i.

Definition 4: The number of *accumulated paths* C^i at the ith level is the number of paths in all the sub-MTBDDs that are represented as the leaves of the complete binary tree of the paired function f^i.

Lemma 1: Let N_u^i be the number of ON-set values in f_u^i. Then

$$C^i = \sum_{u \in GF(2^{k2^i})} N_u^i c_u^i.$$

The correctness of this lemma follows from the fact that there are $2^{n-i} = \sum_{u \in GF(2^{k2^i})} N_u^i$ leaves in the complete binary tree of f^i. A leaf that carries the value u is associated with a sub-MTBDDs that has c_u^i paths.

Notice that the number of leaves in the complete binary tree of the initial function f (= f_0) equals $2^n = \sum_{u \in GF(2^k)} N_u^0$. Following the definition of c_u^0 (in Property 3), we have

$$C^0 = \sum_{u \in GF(2^k)} N_u^0 c_u^0 = \sum_{u \in GF(2^k)} N_u^0 \cdot 1 = 2^n.$$

Moreover, at the top level ($i = n$), there is a single leaf that shows the truth vector of the function. This can be formulated as follows.

Property 5: The number of paths in the MTBDD equals

$$C^n = N_F^n c_F^n = c_F^n.$$

Example 8: Consider the paired functions f^i presented in Table II. The number of accumulated paths C^0 equals

$$N_0^0 c_0^0 + N_1^0 c_1^0 + N_2^0 c_2^0 + N_3^0 c_3^0 = 3 \cdot 1 + 3 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 = 2^3.$$

The number of accumulated paths C^1 equals

$$N_{10}^1 c_{10}^1 + N_{32}^1 c_{32}^1 = 3 \cdot 2 + 1 \cdot 2 = 8.$$

Similarly, $C^2 = 1 \cdot 2 + 1 \cdot 4 = 6$, and C^3 which is the number of paths in the MTBDD equals 6.

In what follows, we derive an analytic expression showing the relationship between the weighted autocorrelation values and the number of paths in the MTBDT. We do it in two steps. First, in Theorem 1 we show the relationship between the number of accumulated paths C^i and the autocorrelation value of $R^{c,i-1}$ at δ_0. Then, in Theorem 2 we show that C^n, which is the number of paths in the MTBDD, is a linear function of the weighted autocorrelation coefficients of the paired functions. We start by defining the weights in the weighted autocorrelation.

Definition 5: Let c_u^i be the weights defined by (5). The weighted autocorrelation function $R^{c,i}$ of the paired function f^i is defined as

$$R^{c,i}(\tau) = \sum_{u \in GF(2^{k2^i})} c_u^i R_u^i(\tau) \quad \tau \in GF(2^{n-i}). \tag{6}$$

The following theorem states that the number of accumulated paths at the level i depends on: 1) the number of accumulated paths at the level $i-1$, and 2) the weighted autocorrelation of the paired functions.

Theorem 1: Let $R^{c,i}$ denote the weighted autocorrelation of the paired function f^i with weights $\{c_u^i\}$ as defined by (5). Then, $C^0 = 2^n$ and for all i, $0 < i \leq n$, the number of accumulated paths at the level i is

$$C^i = C^{i-1} - 0.5 R^{c,i-1}(\delta_0). \tag{7}$$

Proof: Following the definition of the accumulated number of paths, we have, $C^i = \sum_{u \in GF(2^{k2^i})} N_u^i c_u^i$. The symbol u is a concatenation of two symbols, i.e., $u = (u_m, u_l)$ where u_m and u_l are elements of $GF(2^{k2^{i-1}})$. Hence

$$C^i = \sum_{u_m} \sum_{u_l, u_l \neq u_m} N_{(u_m,u_l)}^i c_{(u_m,u_l)}^i + \sum_{u_l} N_{(u_l,u_l)}^i c_{(u_l,u_l)}^i.$$

By using Property 3 we get

$$\begin{aligned} C^i &= \sum_{u_m} \sum_{u_l, u_l \neq u_m} N_{(u_m,u_l)}^i (c_{u_m}^{i-1} + c_{u_l}^{i-1}) + \sum_{u_l} N_{(u_l,u_l)}^i c_{u_l}^{i-1} \\ &= \sum_{u_m} c_{u_m}^{i-1} \sum_{u_l} N_{(u_m,u_l)}^i + \sum_{u_l} c_{u_l}^{i-1} \sum_{u_m} N_{(u_m,u_l)}^i \\ &\quad - \sum_{u_m} N_{(u_m,u_m)}^i c_{u_m}^{i-1} \end{aligned}$$

or

$$C^i = \sum_{u_m} c_{u_m}^{i-1} \sum_{u_l} (N_{(u_m,u_l)}^i + N_{(u_l,u_m)}^i) - \sum_{u_m} N_{(u_m,u_m)}^i c_{u_m}^{i-1}.$$

The pairing of the $(i-1)$th level has paired $(N_{(u_m,u_l)}^i + N_{(u_l,u_m)}^i)$ leaves carrying the value u_m with the same number of leaves carrying the value u_l, $(u_l \neq u_m)$. Additionally, $2N_{(u_m,u_m)}^i$ leaves carrying the value u_m were paired into leaves of the form (u_m, u_m). Therefore, the inner sum, $\sum_{u_l} (N_{(u_m,u_l)}^i + N_{(u_l,u_m)}^i)$ in the last equation, which goes over all the possible values of u_l (including the case where u_l equals u_m), is equal to $N_{u_m}^{i-1}$. Hence we have

$$C^i = \sum_{u_m} c_{u_m}^{i-1} N_{u_m}^{i-1} - \sum_{u_m} N_{(u_m,u_m)}^i c_{u_m}^{i-1}.$$

The last equality consists of two sums. The first one equals C^{i-1}. In the second sum we have $N_{(u_m,u_m)}^i$ which is the number of leaves, in the complete binary tree of f^i, that carry the value u_m and are rooted at the same node. From Property 2, the number of such leaves equals $0.5 \sum_u R_u^{i-1}(\delta_0)$. Consequently

$$C^i = C^{i-1} - 0.5 \sum_u R_u^{i-1}(\delta_0) c_u^{i-1} = C^{i-1} - 0.5 R^{c,i-1}(\delta_0). \tag{8}$$

■

Note that since $R^{c,i-1}(\delta_0) \geq 0$, then the accumulated number of paths at the ith level decreases as i increases.

The following Theorem 2 states that the number of paths is a linear function of the weighed autocorrelation coefficients of the paired functions.

Theorem 2: Let $R^{c,i}$ be the weighted autocorrelation of the paired function f^i at level i with weights $\{c_u^i\}$ as defined by (5). Then, the number of paths in the corresponding MTBDD equals

$$C^n = 2^n - 0.5 \sum_{i=0}^{n-1} R^{c,i}(\delta_0).$$

The proof of Theorem 2 follows directly from Theorem 1.

Note that a reordering or a linear transformation of the input variables is equivalent to a permutation of the autocorrelation coefficients. The explicit expression for the number of paths as a function of the weighted autocorrelation coefficients explains why it is possible to reduce the number of paths by performing a reordering and/or a linear transformation. Moreover, Theorem 2 opens the way to define a deterministic paths-reduction procedure that maximizes $\sum_{i=0}^{n-1} R^{c,i}(\delta_0)$ by permuting the weighted autocorrelation coefficients. The paths-reduction procedure (presented in Section V-B) performs

the permutation by replacing the initial basis that spans the domain of the function by another basis the elements of which are linear combinations of elements of the initial basis.

IV. Number of Paths vs. the Number of Nodes

In this section we present two examples. The first example illustrates the use of autocorrelation coefficients for determining analytically the ordering of the variables. In the second example (Example 10), we present a function for which there exists no ordering that can minimize the number of paths and the number of nodes simultaneously. That is, the optimal ordering that minimizes the number of nodes differs from the optimal ordering that minimizes the number of paths.

The relationship between the number of nodes in a BDD and the number of paths was discussed in [6]. It was shown that a reduction in the number of nodes does not necessarily reduce the number of paths.

Theorem 3: [6] For any $n \geq 5$, there exists a function $f(x_{n-1}, \ldots, x_0)$, such that for at least one pair of orderings of the input variables, π_i, π_j, the number of nodes is $N_i > N_j$, but the number of paths is $C_i < C_j$.

For the proof, the authors in [6] constructed the function

$$f(x_{n-1}, \ldots, x_0) = x_0\bar{h} + \bar{x}_1 h + x_0 x_2 + \bar{x}_0 x_1 \bar{x}_2 \tag{9}$$

where $h(x_{n-1} \ldots x_4, x_3) = x_3 \oplus x_4 \oplus \cdots \oplus x_{n-1}$, and they specified two orderings of the input variables that satisfy this property. These two orderings are $\pi_1 = (x_1, x_2, (h), x_0)$, and $\pi_2 = ((h), x_1, x_2, x_0)$, where (h) denotes any ordering of the variables of the function h. The corresponding BDDs have $C_1 < C_2$ and $N_2 < N_1$.

The following example illustrates the use of autocorrelation coefficients for determining the ordering of the variables analytically. We analyze the above function in terms of its autocorrelation coefficients and show that the ordering by decreasing autocorrelation coefficients, referred to as π_3, determines a BDD that has $C_3 < \min(C_1, C_2)$ and $N_3 = \min(N_1, N_2)$.

Example 9: The function f [specified above in (9) in its SOP form] can be presented as a set of five disjoint cubes $\{P_i\}_{i=1}^5$, where P_i is a cube in the variables x_0, x_1, x_2 and h. Namely

$$\begin{aligned} P_1 &= \bar{x}_0\bar{x}_1 h \qquad & P_2 &= \bar{x}_0 x_1 \bar{x}_2 \\ P_3 &= x_0\bar{x}_1 \qquad & P_4 &= x_0 x_1 \bar{x}_2 \bar{h} \\ P_5 &= x_0 x_1 x_2. \end{aligned}$$

Each cube represents a cluster of binary vectors, which form a coset (a coset is a linear subspace shifted by a constant vector). The order of a cube is the dimension of the corresponding linear subspace. For example, the cube P_3 is of order $n-2$ since it represents all the 2^{n-2} binary vectors of length n in which $x_0 = 1$ and $x_1 = 0$. The cubes in our example are of orders $w-1$, w and $1+w$, where $w = n-3$.

The cubes are disjoint and therefore the autocorrelation function can be calculated by using the technique introduced in [15] and [30] or directly on the BDD by using the method presented in [25]. In this example, we used the method presented in [15]. The advantage of the method presented in [15] is that it calculates a number of autocorrelation values simultaneously. The low computational complexity of the method results from the disjointness of the cubes. To simplify the presentation, we illustrate the principles of this method by calculating a single autocorrelation value. The generalization of the method for calculating a set of values simultaneously, is described in details in [15].

TABLE III
BDD's Characteristics for the Orderings π_1, π_2, and π_3

Ordering	Nodes N	Paths C
π_1	$4w+2$	$6 \cdot 2^{w-1} + 2$
π_2	$2w+5$	$10 \cdot 2^{w-1}$
π_3	$2w+5$	$4 \cdot 2^{w-1} + 4$

There are two characteristic functions f_0 and f_1. The corresponding autocorrelation functions are $R_0(\tau)$ and $R_1(\tau)$. The initial weights are $c_0^0 = c_1^0 = 1$. Indeed, it is sufficient to calculate only one autocorrelation function, say $R_1(\tau)$, since from Property 1 we have

$$R^{c,0}(\tau) = 1 \cdot R_0(\tau) + 1 \cdot R_1(\tau) = 2^n - 2R_1(0) + 2R_1(\tau). \tag{10}$$

Define $\hat{P}_i(h, x_2, x_1, x_0) = P(h, x_2, x_1, \bar{x}_0) = P(x \oplus \delta_0)$. Since the cubes are disjoint, we have $P_i P_j = 0$, $\hat{P}_i \hat{P}_j = 0$, and, $\hat{P}_i P_i = 0$. The value of $R_1(\delta_0)$ equals the weight of the function obtained by multiplying $f = \sum_{i=1}^5 P_i$ by the function $f(x \oplus \delta_0) = \sum_{i=1}^5 \hat{P}_i$. The product of the two functions equals

$$P_1\hat{P}_3 + P_2\hat{P}_4 + P_3\hat{P}_1 + P_4\hat{P}_2 = \bar{x}_1 h + x_1 \bar{x}_2 \bar{h}.$$

The weight of the latter function is $2^2 \cdot 2^{w-1} + 2 \cdot 2^{w-1}$. Thus, $R_1(\delta_1) = 6 \cdot 2^{w-1}$.

Similarly, the values of $R_1(\tau)$ for $\tau = 0, \delta_0, \delta_1, \delta_2, \delta_i$ and h are

$$\begin{aligned} R_1(0) &= 2^w + 2^w + 2^{w+1} + 2^{w-1} + 2^w = 11 \cdot 2^{w-1} \\ R_1(\delta_0) &= 2^2 \cdot 2^{w-1} + 2 \cdot 2^{w-1} = 6 \cdot 2^{w-1} \\ R_1(\delta_1) &= 2 \cdot 2^{w-1} + 2 \cdot 2^{w-1} + 2 \cdot 2^w = 8 \cdot 2^{w-1} \\ R_1(\delta_2) &= 2 \cdot 2^{w-1} + 0 + 2^{w+1} + 2 \cdot 2^{w-1} = 8 \cdot 2^{w-1} \\ R_1(\delta_i) &= 2^w + 2 \cdot 2^w + 2^w = 8 \cdot 2^{w-1} \qquad \text{for } i \geq 4 \\ R_1(h) &= \begin{cases} 11 \cdot 2^{w-1} & \omega \text{ is even} \\ 8 \cdot 2^{w-1} & \omega \text{ is odd.} \end{cases} \end{aligned} \tag{11}$$

The basis vector associated with x_0 carries the minimal (weighted) autocorrelation value. From (10) and (11) we have

$$R^{c,0}(\delta_0) < R^{c,0}(\delta_1) = R^{c,0}(\delta_2) \leq R^{c,0}(h).$$

Any analytic reordering algorithm that orders the variables by decreasing autocorrelation coefficients (e.g., [11], [13], [15] and the procedure suggested in Section V-B) may derive (in the worst case) the ordering $\pi_3 = (x_0, x_1, x_2, (h))$. Fig. 6 shows the three BDDs defined by the orderings π_1, π_2, and π_3, and Table III compares them in terms of their size and the number of paths. The ordering π_3 defines a BDD having a smaller number of nodes and a smaller number of paths with respect to the BDDs with orderings π_1 and π_2.

The ordering π_3 defined by using the autocorrelation coefficients leads to a better result, with respect to π_1 and π_2, in

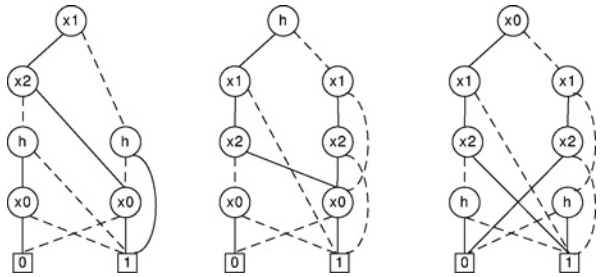

Fig. 6. BDDs of the function in Example 9 with orderings (left) π_1, (center) π_2, and (right) π_3.

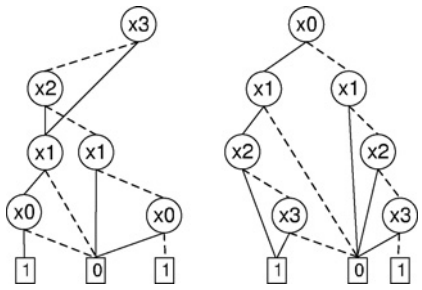

Fig. 7. BDDs in Example 10 with the orderings (left) π_1 and (right) π_2.

both the number of paths and the number of nodes. However, although π_3 reduces both parameters, it does not contradict Theorem 3. The theorem justifies the need for a dedicated paths-reduction procedure, especially when suboptimal minimization procedures are considered, since for some functions the number of nodes and the number of paths may not decrease simultaneously. The following example presents a function for which the minimal number of nodes and the minimal number of paths are obtained by different orderings.

Example 10: Consider the function $f(x_3, x_2, x_1, x_0) = \bar{x}_0\bar{x}_1\bar{x}_2\bar{x}_3 + x_0x_1x_2 + x_0x_1x_3$. The BDDs of the function with the orderings $\pi_1 = (x_3, x_2, x_1, x_0)$ and $\pi_2 = (x_0, x_1, x_2, x_3)$ are shown in Fig. 7. The BDD defined by π_1 has the minimal possible number of nodes, $N = 6$, and has 9 paths. The BDD defined by π_2 has the minimal possible number of paths, $C = 8$, but has 7 nodes. For this function, there exists no permutation of the input variables that minimizes the number of nodes and the number of paths simultaneously.

V. Reduction of the Number of Paths in MTBDD

In this section, we present a procedure for minimizing the number of paths in MTBDD and analyze the performance and computational complexity of the procedure.

A. Optimization Problem

A linear decomposition [12] is a well-known technique that allows a representation of a function f as a superposition of a linear transformation function σ and a nonlinear function, f_σ, such that $f(\mathbf{x}) = f_\sigma(\sigma(\mathbf{x}))$. In this paper, the cost function for the linearization is the number of paths in the MTBDD corresponding to the linearized function f_σ.

Let $T = \sigma^{-1} = (\tau_{n-1}, \ldots \tau_1, \tau_0)$ be a nonsingular matrix. The columns of T form a basis that spans the domain $GF(2^n)$. The

Set $c_u^0 = 1$ for all $u \in GF(2^k)$
Set $i = 0$
a. For all $\tau \in GF(2^{n-i})$, $\tau \neq 0$, calculate the weighted autocorrelation function $R^{c,i}(\tau)$, as determined in Eq. 6. b. Determine τ that maximizes the weighted autocorrelation function. If there is more than one τ, choose one randomly.
c. Replace the node variable at the i'th level by τ and apply pairing on the MTBDD.
d. Update the set of weights following Eq.(5).
e. Set $i = i + 1$ and repeat the procedure until $i = n - 1$, or, until $R^{c,i}(\tau) = 0$ for all calculated τ.
f. Extend the chosen vectors τ into $\hat{\tau}$. The vectors $\hat{\tau}$s are the columns of T.

Fig. 8. Paths-reduction procedure.

weighted autocorrelation function, $R^w_{f_\sigma}(x)$, of the linearized function f_σ, (defined by σ), is $R^w_{f_\sigma}(x) = R^w_f(\sigma^{-1}x) = R^w_f(Tx)$, where $R^w_f(x)$ is the weighted autocorrelation function of $f(x)$, and we have $R^w_{f_\sigma}(\delta_0) = R^w_f(T\delta_0) = R^w_f(\tau_0)$. Therefore, the optimization problem for reducing the number of paths can be formulated as follows:

Construct a set of vectors $\tau_i \in GF(2^{n-i})$, $i = 0, \ldots n-1$, for which $\sum_{i=0}^{n-1} R_f^{c,i}(\tau_i)$ is maximal.

Note that for $i > 0$, the vectors τ_is are not in $GF(2^n)$. In order to construct a set of linearly independent basis vectors that span $GF(2^n)$, we have to extend the τs to $\hat{\tau}_i$s by multiplying their corresponding decimal value by 2^{n-i}, i.e., $\hat{\tau}_i = (\tau_i, 0, 0 \ldots 0)$.

B. Procedure for Minimization of the Number of Paths

The computational complexity of constructing an optimal set of basis vectors (the optimal linear transformation matrix σ), or an optimal permutation of an initial set of basis vectors is NP-hard [1]. For that reason, greedy algorithms are used to solve the minimization problem.

The proposed path-reduction procedure is presented in Fig. 8. The procedure constructs deterministically a set of basis vectors that provides an MTBDD or SBDD with reduced number of paths. At each step of the procedure a vector τ which carries the maximal weighted autocorrelation value is chosen, and the corresponding $\hat{\tau}$ is added to the set of basis vectors instead of one of the initial vectors. In other words, at level i, a vector $\hat{\tau}_i = (\tau_i, 0, 0 \ldots 0)$, $\tau_i \in GF(2^{n-i})$ replaces a basis vector δ_k, $k \geq i$, if its weighted autocorrelation satisfies

$$R^{c,i}(\tau_i) \geq R^{c,i}(\tau)$$

for all $\tau \in GF(2^{n-i})$, $\tau \neq 0$. Hence, the corresponding local linear transformation matrix σ_i is determined deterministically.

The following example illustrates how the linearization procedure works.

Example 11: Consider the benchmark function *9sym*, which has 9 inputs and a single output. The BDD of *9sym* has 220 paths. Since the function is a totally symmetric function, a simple permutation of the input variables cannot improve

the path count. However, it is possible to reduce the number of paths by applying a linear transformation on the input variables. In this example, we restrict the Hamming weight of τ to be less or equal to two. This restriction reduces the number of paths in the linearized part (f_σ) and also in the linear part (σ).

The autocorrelation function of *9sym* with the natural ordering is equal to the weighted autocorrelation function $R^{c,0}(\tau)$. The values of $R^{c,0}$ for $\tau's$ whose Hamming weight is smaller or equal to two, are shown in Fig. 9 (top). The x-axis corresponds to the integer value of τ when referred to as a number in base two. To avoid confusion, the autocorrelation values associated with τ's having Hamming weight greater than two (and hence are not considered as candidate basis vectors) are assigned with the value "-1."

Notice that all the original basis vectors (δ_i) carry the maximal autocorrelation value, which equals 400. Thus, one of them will be chosen randomly (see Step b in the algorithm). Assume that the algorithm chooses δ_0 [i.e., the binary vector (000000001)]. Consequently, at the first step ($i = 0$) the set of the basis vectors is kept unchanged (see Step c in the algorithm). This is equivalent to applying a *local linear transformation matrix* $\hat{\sigma}_0$ ($= \sigma_0$) that equals $I_{9\times 9}$, where I is the identity matrix. The paring operation (see Step d) results in an 8-input function, which is referred to as the *partially* linearized function $f_{\hat{\sigma}_0}$. This function takes four possible values: (00), (01), (10) and (11), with weights $c^1_{00} = 1$, $c^1_{01} = 2$, $c^1_{10} = 2$ and $c^1_{11} = 1$, respectively. The number of paths at this step is unchanged, it equals 220.

At the second step ($i = 1$) the weighted autocorrelation $R^{c,1}(\tau)$, of $f_{\hat{\sigma}_0}$ is calculated. The autocorrelation values are shown in Fig. 9. The maximal autocorrelation value (for $\tau \neq 0$) is equal to 192. Several vectors attain this value, assume that we choose τ = (00000011). Notice that this τ is an eight-bit vector, i.e., it is an element of $GF(2^8)$. In order to map it to an element in $GF(2^9)$ we have to multiply it by 2^{9-8}, i.e., $\hat{\tau}$ = (000000110). Replacing the node variable at the second level by the chosen τ results in a *partially* linearized function $f_{\hat{\sigma}_1} = f_{\sigma_1\sigma_0}$ which has 196 paths. The corresponding local linear transformation matrix equals

$$\hat{\sigma}_1 = \sigma_1\sigma_0 = \sigma_1 = (\delta_8, \delta_7, \ldots \delta_2, \hat{\tau}, \delta_0)^{-1}.$$

Table IV presents for each level i (the first column) the number of paths in the BDD of the *partially* linearized function (the second column). Namely, the number of paths in the BDD corresponding to $f_{\hat{\sigma}_i}$, where $\hat{\sigma}_i = \sigma_i\sigma_{i-1}\cdots\sigma_0$. The third column shows the maximal value of the weighted autocorrelation function of level i, $R^{c,i}(\tau)$, and the fourth column shows the corresponding τ. The last three columns show the number of characteristic functions at level i (denoted as $|u|$) and the minimal and maximal values of the accumulated weight c^i. The first row in the table, labeled as "*orig*," shows the number of paths in the BDD with the natural ordering of the input variables, and, the weighted autocorrelation of the leaf-variable x_0.

Note that after the pairing at the sixth step, all the leaves carry different values (i.e., max $R^{c,6} = 0$). Hence, a further reduction in the number of paths is not possible, and therefore,

TABLE IV
LINEARIZATION OF THE FUNCTION 9SYM

i	C	Max $R^{c,i}(\tau)$ ($\tau \neq 0$)	τ	$\lvert u\rvert$	min c^i	max c^i
Orig	220	400	000 000 001	2		
0	220	400	000 000 001	2	1	1
1	196	192	00 000 011	4	1	2
2	152	116	0 000 110	10	1	3
3	116	80	001 100	12	1	6
4	88	58	11 000	19	1	12
5	88	2	0 001	12	1	24

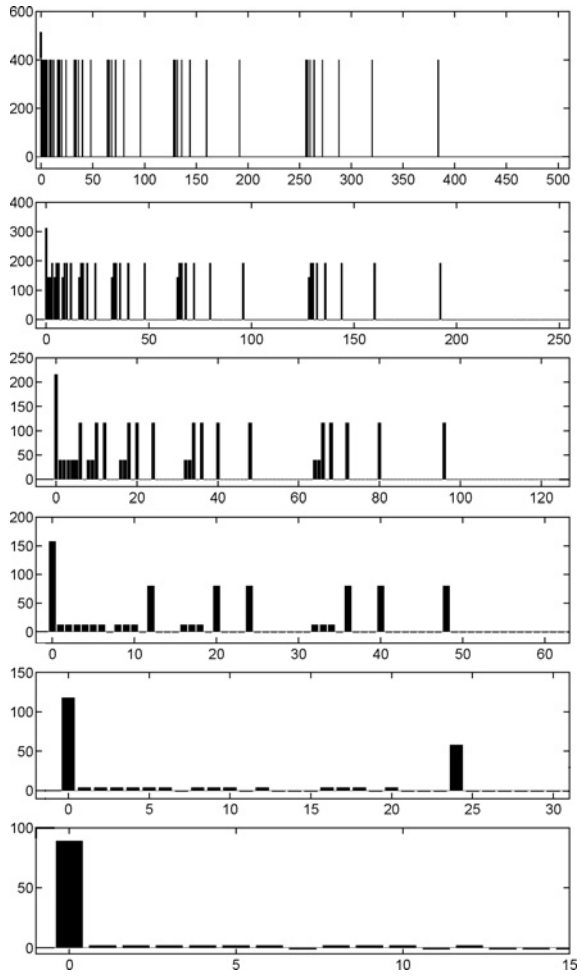

Fig. 9. Weighted autocorrelation functions $R^{c,i}(\tau)$, for Example 11 plotted in increasing order. The weighted autocorrelation of the first level $R^{c,0}$ is placed at the top of the figure, and $R^{c,5}$ is placed at the bottom.

the procedure ends. The final number of paths in the linearized BDD according to Theorem 2 is

$$C = 2^9 - 0.5 \cdot (400 + 192 + 116 + 80 + 58 + 2 + 0 + 0 + 0) = 88.$$

In this example, the linearization reduces the number of paths from 220 to 88.

Notice that the number of paths in the linear part is negligible, as can seen in Fig. 10. The left-hand side of the figure presents a logic scheme of the combinatorial circuit that implements the linear transformation of the input variables $x = (x_8, \ldots, x_0)$ into $z = (z_8, \ldots, z_0)$. The right-hand side of

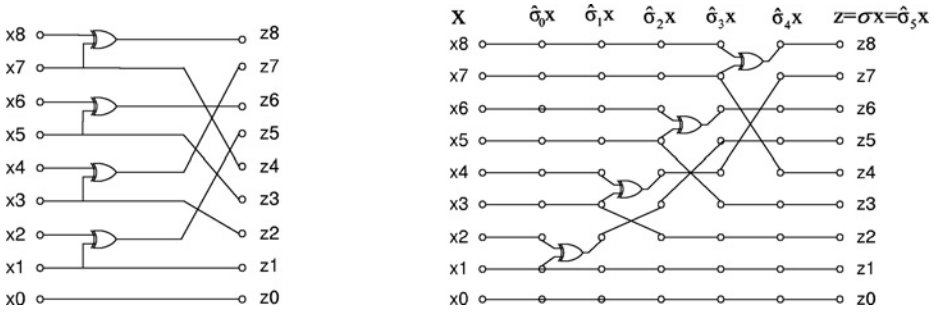

Fig. 10. Implementation of the linear transformation σ in Example 11 as a combinatorial circuit (left). Representation of σ as a concatenation of six local linear transformations (right).

TABLE V
TRUTH TABLE AND WEIGHTED AUTOCORRELATION OF $f(x_2, x_1, x_0) = x_0x_1' + x_0'x_2$

x	0	1	2	3	4	5	6	7
$f(x) = f^0(x_2, x_1, x_0)$	0	1	0	0	1	1	1	0
$R = R^{c,1}$	8	4	4	4	4	4	0	4

the figure presents the linear transformation σ as a concatenation of the above six local linear transformations defined by the τs in Table IV.

C. Number of Paths as a Function of the Chosen Basis Vector

The algorithm is greedy since at the ith iteration it chooses a τ of the maximal weighted autocorrelation value while ignoring how it affects the weighted autocorrelation function of the paired function at level i+1. Therefore, the linearized MTBDD may be suboptimal. The following example illustrates such a case. For simplification, the Hamming weight of the candidate basis vectors was restricted to one, i.e., only permutations of the input variables were considered.

Example 12: Consider the function $f(x_2, x_1, x_0) = x_0\bar{x}_1 + \bar{x}_0x_2$ given in Table V. There are six possible orderings of the input variables. The best two orderings are $\pi_1 = (x_0, x_1, x_2)$ and $\pi_2 = (x_0, x_2, x_1)$ which define a BDD that has $N = 3$ nodes and $C = 4$ paths. The worst orderings are $\pi_3 = (x_1, x_2, x_0)$ and $\pi_4 = (x_2, x_1, x_0)$, they provide $N = 5$ and $C = 6$. The other two orderings, $\pi_5 = (x_1, x_0, x_2)$ and $\pi_6 = (x_2, x_0, x_1)$, have the same number of paths ($C = 6$) but a smaller number of nodes ($N = 4$).

The (weighted) autocorrelation coefficients of $f^0(x_2, x_1, x_0)$ are written in the last row of Table V. Note that the autocorrelation value corresponding to the three initial basis vectors are all equal to 4. Therefore, any one of them may be selected by the procedure as the basis vector for level 0. If the variable x_2 (or x_1) is chosen as the first basis vector, then the algorithm generates the optimal ordering $\pi_1 = (x_0, x_1, x_2)$ (or $\pi_2 = (x_0, x_2, x_1)$). However, if x_0 is selected for the first level ($i = 0$), then the worst ordering is obtained.

In general, when there is more than one τ that attains the maximal weighted autocorrelation value, the minimization results are affected by the random selection of τ. In such cases, an exhaustive search procedure that simultaneously works on several adjacent levels may reduce the degradation caused by the local decisions.

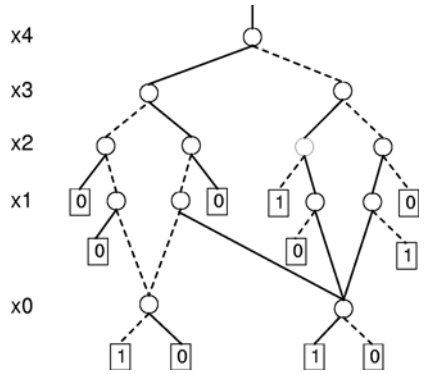

Fig. 11. MTBDD of the function in Example 13.

D. Dependency of Paths-Reduction Results on the Cost Function

In this section, we compare the cost function that we use for path minimization to the cost function used in [13] for reducing the BDD size. A local linear transformation (or a basis vector) is chosen by considering the autocorrelation coefficients of the paired functions. In [13], the authors use the total-autocorrelation function (i.e., $c_u^i = 1$ for all $0 \le i < n$ and $u \in GF(2^{k2^i})$), while the paths-reduction procedure uses the weighted autocorrelation coefficients. When the number of variables is small, the differences between the accumulated weights c_u^i are negligible. Thus, the performance of the suggested algorithm and the procedure presented in [13] may be similar.

The difference between the cost functions is clearly observed in large values of n. However, for some functions, it can affect the minimization results even if the number of variables is small. The following example shows the effect of the accumulated weights on the linerization results for $n = 5$. To simplify the presentation, we restrict the Hamming weight of the basis vectors to 1, which means that the linear transformation matrix σ is restricted to a permutation matrix.

Example 13: Consider the function $f(x_4, x_3, x_2, x_1, x_0)$ shown in Fig. 11. The BDD of f has 13 nodes and 17 paths. The *total-autocorrelation* function of $f = f^0$ (denoted by $R^{\text{Total},0}(\tau)$), and the weighted autocorrelation $R^{c,0}(\tau)$, are equal since the initial weights are all set to one, i.e., $c_0^0 = c_1^0 = 1$. The autocorrelation coefficients for $\tau = \delta_0, \delta_1, \delta_2, \delta_3$ and δ_4 are 22, 22, 14, 18, and 18, respectively. Assume that the first basis vector is δ_0, namely, there is no permutation at

TABLE VI
CHARACTERISTICS OF THE BDDS IN EXAMPLE 13

	Initial	Ordering Using $R^{\text{Total},i}$ [13]	Ordering Using $R^{c,i}$
Order	π_1	π_2	π_3
Paths	17	16	12
Nodes	13	12	10
APL	3.8125	3.5625	3.3125

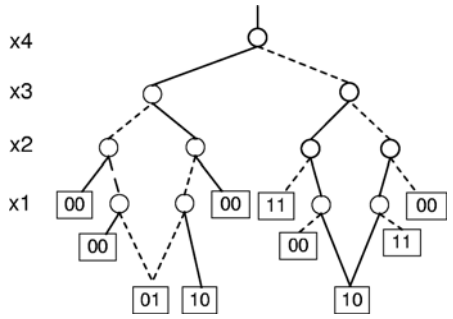

Fig. 12. MTBDD of the paired function $f^1(x_4, x_3, x_2, x_1)$ in Example 13.

level 0. The MTBDD of the paired function $f^1(x_4, x_3, x_2, x_1)$ with the natural ordering $\pi_1 = (x_4, x_3, x_2, x_1, x_0)$ is shown in Fig. 12. The MTBDD has four leaves with accumulated weights $c^1_{00} = c^1_{11} = 1$ and $c^1_{01} = c^1_{10} = 2$. Therefore, the corresponding autocorrelation functions are different

$$R^{\text{Total},1} = [16, 8, 2, 2, 8, 4, 6, 6, 4, 4, 6, 4, 2, 2, 6, 6]$$
$$R^{c,1} = [21, 8, 2, 2, 12, 4, 6, 6, 4, 4, 8, 4, 2, 2, 8, 6].$$

The value of $R^{\text{Total},1}(\tau)$, $\tau \neq 0$, is maximal for $\tau = \delta_0$ and δ_2. If δ_0 is chosen, then the final ordering becomes $\pi_2 = (x_4, x_2, x_3, x_1, x_0)$ and the BDD has 16 paths. However, the weighted autocorrelation attains its maximal value for $\tau = \delta_2$. Therefore, the final ordering obtained by the suggested algorithm is $\pi_3 = (x_4, x_1, x_3, x_2, x_0)$ and the corresponding BDD has 12 paths. Table VI summarizes the characteristics of the BDDs with three orderings.

E. Computational Complexity of the Paths-Reduction Procedure

The paths-reduction procedure suggested in this paper can be considered as an analytic reordering method if the Hamming weight of the candidate basis vectors is restricted to one. The paths-reduction procedure is greedy, and in some cases it may produce a suboptimal solution. However, its computational complexity is acceptable. The time complexity of the procedure is the sum of time complexities of its steps. The most demanding computation in each step is the calculation of the autocorrelation function $R^{c,i}_u$. The value of $R^{c,i}_u(\delta_k)$ can be calculated in several ways:

1) by using the Wiener–Khinchin theorem with complexity $\mathcal{O}((n-i)2^{n-i})$ [12];
2) according to the definition of $R^{c,i}_u$ with complexity $\mathcal{O}((N^i_u)^2)$, where N^i_u is the number of paths in the paired MTBDD that end at the leaf u, or the number of disjoint cubes that carry the value u [15], [25], [30];
3) directly on the BDD [25].

The above approaches were discussed in detail in [25]. The experimental results reported in [25] show that for the considered benchmark functions, the third method calculates the *first-order* autocorrelation coefficients, i.e., $R^{c,i}_u(\delta_k)$, within a reasonable time.

F. Reduction of the Number of Paths in Shared BDD

Consider a multioutput function of n inputs and k outputs represented by a shared BDD. The number of paths in a shared BDD equals the sum of the paths in each of the k BDDs. Therefore, Theorem 2 can be extended to the case of SBDD as follows.

Corollary 1: Let $f^{j,i}$ be the logic function that corresponds to the jth BDD, $0 \leq j \leq k$, at the ith level. Let $R^{c,i}_j$ be the corresponding weighted autocorrelation function. Denote by $\{c^i_u\}$ the accumulated weights as defined by (5). Then, the number of paths in the corresponding SBDD equals

$$C^n = 2^n k - 0.5 \sum_{j=0}^{k-1} \sum_{i=0}^{n-1} R^{c,i}_j(\delta_0). \tag{12}$$

It is clear from the corollary that the number of paths can be reduced by choosing a set of basis vectors that maximizes the sum $\sum_{j=0}^{k-1} R^{c,i}_j(\delta_0)$ of autocorrelation coefficients per level. Namely, the suggested algorithm can be used for SBDD paths-reduction by replacing the computation of the single weighted autocorrelation function by the calculation of the $\sum_{j=0}^{k-1} R^{c,i}_w(\tau)$ for all τ, $\tau \in GF(2^{n-i})$ of Hamming weight less or equal to w.

VI. EXPERIMENTAL RESULTS

In this section, we apply the proposed methods to a number of relatively small MCNC benchmark functions. The corresponding software was developed by using standard packages in MATLAB; however, existing programming packages for handling large switching functions can be equally applied without restrictions. The performance of various algorithms are compared in terms of the number of paths in the corresponding MTBDDs.

The main advantage of suggested path-reduction procedure is that it completes the minimization within n steps, where n is the number of input variables. The efficiency of the procedure (about 50% reduction in the number of paths) is due to the fact that it uses the autocorrelation coefficients in order to determine the set of basis vectors. In order to demonstrate this advantage over dynamic algorithms which do not use the autocorrelation information, we compared the suggested procedure to a "branch-and-bound"-like dynamic iterative procedure. Each iteration the dynamic procedure replaces one (randomly chosen) basis vector by another random vector, if the new vector reduces the number of paths. Fig. 13 (top) shows the probability $p(x)$ that after $N = 300$ iterations the dynamic procedure will construct an MTBDD with x paths. Fig. 13 (bottom) shows the accumulated probability, i.e., the probability that the dynamic procedure will construct an MTBDD with at most x paths. The experiment was conducted on the benchmark function *clip* that has nine input variables.

The candidate basis vector were randomly chosen from a set of $(2^9 - 1)$ vectors. The probabilities were obtained by repeating the experiment 10^5 times. Note that the MTBDD with the natural ordering has 454 paths and that the suggested analytic procedure produced an MTBDD of 204 paths within seven steps.

Tables VII and VIII show the following MTBDD and SBDD characteristics for each benchmark function of n input variables and k output variables.

1) $C_{MT}(natural)$ and $C_S(natural)$ are the number of paths in the MTBDD and SBDD with the natural ordering of the input variables, respectively.
2) $C_{MT}(sifting)$ and $C_S(sifting)$ are the number of paths in the MTBDD and SBDD after reordering them them by performing sifting with a cost function which is the number of paths.
3) $C_{MT}(ordered)$ is the number of paths in an ordered MTBDD. The input variables are ordered by their corresponding autocorrelation values as in Example 9, that is by using the autocorrelation values of the original (i.e., nonpaired) function.
4) $C_{MT}(weight)$ and $C_S(weight)$ are the number of paths of the linearized MTBDD and SBDD using the suggested algorithm with weighted autocorrelation coefficients.

The last column in each table shows the execution time (in seconds) for both MTBDD and SBDD minimization with weighted autocorrelation coefficients.

The experimental results indicate that a reordered MTBDD (i.e., MTBDD with the optimized order of variables) almost always reduces the number of paths with respect to the MTBDD with the natural ordering. In addition, linearization almost always reduces the number of paths with respect to the reordered MTBDD. Note that reordering of the input variables of symmetric functions (like *9sym*) cannot reduce the number of paths. However, for partially symmetric functions (like *add6*) it can reduce the number of paths by finding the optimal order of the symmetry-sets (a symmetry-set is a subset of input variables for which the function is invariant to any permutation of variables in the set; the symmetric-sets are disjoint).

Notably, on average, the number of *nodes* in the linearized BDDs does not increase. The number of nodes in the reordered MTBDD is reduced by 5% with respect to the MTBDD with the natural ordering, whereas the number of paths is reduced by 30.3%. The same occurs with the linearized MTBDD, where the number of nodes was reduced by 15.9% and the number of paths by 50.1%. As for the SBDDs, the number of paths was reduced by 32.2% and the number of nodes was reduced by 22.9%. As was mentioned earlier, the analytic paths-reduction approach has an advantage over dynamic approaches since it works on a more global level. That is, it considers the intrinsic properties of the function and thus its cost function is similar to the cost function of analytic nodes-reduction algorithms. Thus, inherently, the suggested procedure is not likely to increase the number of nodes significantly.

A comparison between the performance of the conventional sifting algorithm (modified so to minimize the number of paths) and the suggested paths-reduction procedure, indicates a clear difference between SBDD and MTBDD paths reduction. For SBDDs, the sifting (when adapted to path reduction) reduces the number of paths by 34.8%, and the linearization provides a reduction of 32.3%. In contrast, for MTBDDs, the sifting reduces the number of paths by almost the same amount (31.2%), while the linearization improves it by 50.1%. The reason for this is the following. Our paths-reduction procedure is based on Theorem 2. Namely, it is designed to minimize the number of paths in a *single diagram* (MTBDD). Since, the problem of the SBDD optimization is similar to the problem of the minimization of the *average* number of paths over several shared diagrams, the performance of the paths-reduction procedure degrades when it is applied to SBDDs.

Notice that the performance of the sifting algorithm and the autocorrelation-based ordering is quite similar (31.2% vs. 30.3%). However, the two algorithms differ in their *computational complexity*. The computational complexity of the autocorrelation-based ordering is much smaller than the computational complexity of the sifting. In the sifting algorithm [26], each variable is moved up and down in the order so that it takes all possible positions. Then, the best position is identified and the variable is returned to that position. In contrast, in the autocorrelation-based ordering, there is no need to place a variable in all possible positions (see Example 9). As was shown in Property 2, it is sufficient to place each variable

TABLE VII
NUMBER OF PATHS IN MTBDDS

Func.	n, k	C_{MT} *ntrl*	C_{MT} *Sifting*	C_{MT} *Ord.*	C_{MT} *Weight*	Time (s)
9sym	9, 1	220	220	220	58	0.79
add6	12, 7	4096	4096	4096	729	6.40
alu1	12, 8	1754	1641	1468	1387	7.32
alu2	10, 8	581	398	407	407	1.25
alu3	10, 8	707	483	478	487	1.27
apla	10, 12	264	127	161	166	1.62
clip	9, 5	454	434	468	204	0.52
dk17	10, 11	377	119	106	107	1.60
dk27	9, 9	86	47	47	47	0.76
misex3c	14, 14	15 288	8634	8924	8882	61.10
sao2	10, 4	237	95	95	89	1.81
Reduction		0	31.2%	30.3 %	50.1%	

TABLE VIII
NUMBER OF PATHS IN SBDDS

Func.	n, k	C_S Natural	C_S Sifting	C_S Weight	Time (s)
9sym	9, 1	220	220	58	0.87
add6	12, 7	2185	1441	367	5.82
alu1	12, 8	39	39	65	6.79
alu2	10, 8	452	176	263	1.16
alu3	10, 8	439	174	263	1.30
apla	10, 12	563	288	386	1.44
clip	9, 5	728	510	336	0.49
dk17	10, 11	306	161	217	1.56
dk27	9, 9	51	42	56	0.78
sao2	10, 4	431	220	236	1.79
Reduction		0 %	34.8%	32.2%	

TABLE IX
NUMBER OF PATHS IN THE MTBDD WITH THE NATURAL ORDERING AND THE LINEARIZED MTBDD WITH LIMITATIONS ON w

Func.	n, k	Natural	$w = 1$	2	3	n
9sym	9, 1	220	220	88	88	58
add6	12, 7	4096	4096	729	729	729
alu1	10, 8	1754	1387	1387	1387	1387
alu2	10, 8	581	407	407	407	407
alu3	10, 8	707	487	487	487	487
clip	9, 5	454	480	204	204	204
misex3c	14, 14	15288	8882	8882	8882	8882
sao2	10, 4	237	95	88	88	88

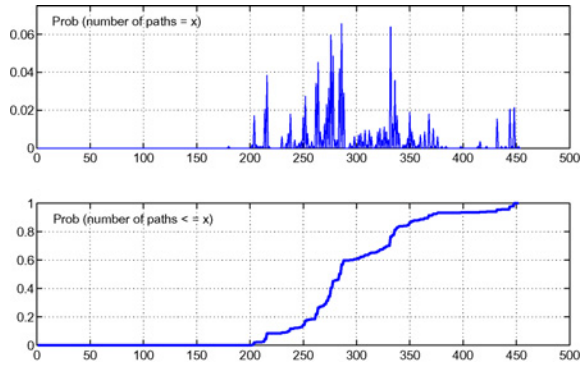

Fig. 13. Clip: the probability $p(x)$ that a dynamic iterative procedure will converge to an MTBDD of x paths within 300 iterations (top). The accumulated probability that a dynamic iterative procedure will converge to an MTBDD of at most x paths within 300 iterations (bottom).

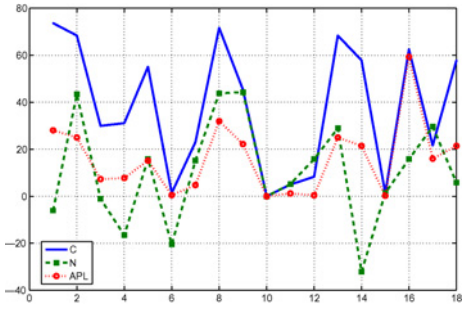

Fig. 14. Improvement (in percentage) in the number of paths (solid line), nodes (dashed line), and APL (dotted line) in the linearized MTBDDs with respect to MTBDDs with the natural ordering of the inputs.

only *at the bottom* of the decision diagram so to extract its autocorrelation value.

Table IX shows the reduction of the number of paths when the Hamming weight of the basis vector is restricted to be less or equal to w. Note that $w = 1$ means reordering of the input variables. It is clear that it is sufficient to work with basis vectors of Hamming weight less than four; thus, the number of paths in the linear part (i.e., σ) can be reduced.

Table X shows the number of nodes (N), paths (C), and the APL of the MTBDD with the natural ordering (denoted by a subscript *ntrl*) and the MTBDD corresponding to the linearized function (*lin*) for several benchmark functions. Fig. 14 shows the reduction in the number of paths, the number of nodes, and the APL of the linearized MTBDD with respect to the MTBDD with the natural ordering. The x-axis

TABLE X
NUMBER OF PATHS, NODES, AND APL IN MTBDDS

No.	Func.	n, k	C *ntrl*	N *ntrl*	APL *ntrl*	C *lin*	N *lin*	APL *lin*
1	9sym	9, 1	**220**	33	7.34	**58**	35	5.28
2	adr4	8, 5	**256**	113	8.00	**81**	64	6.00
3	alu2	10, 8	**581**	264	8.92	**407**	267	8.27
4	alu3	10, 8	**707**	278	9.27	**487**	324	8.55
5	clip	9, 5	**454**	189	8.75	**204**	159	7.42
6	dc2	8, 7	**144**	117	6.09	**142**	141	6.06
7	dist	8, 5	**204**	125	7.54	**157**	106	7.18
8	dk17	10, 11	**377**	160	8.39	**107**	90	5.71
9	dk27	9, 9	**86**	79	6.31	**47**	44	4.91
10	f51m	8, 8	**256**	255	8.00	**256**	255	8.00
11	inc	7, 9	**40**	39	4.98	**38**	37	4.92
12	mlp4	8, 8	**241**	240	7.75	**221**	202	7.72
13	radd	8, 5	**256**	90	8.00	**81**	64	6.00
14	rd73	7, 3	**128**	28	7.00	**54**	37	5.50
15	root	8, 5	**73**	72	5.55	**72**	71	5.54
16	sao2	10, 4	**237**	95	7.10	**89**	80	2.89
17	sqn	7, 3	**88**	81	6.25	**69**	57	5.25
18	z4	7, 4	**128**	52	7.00	**54**	49	5.50

is the benchmark number (from Table X) and the y-axis is the improvement in percentage. Note that when the variance of the accumulated weights (c_u^i) is small, the procedure also provides a significant reduction in the APL.

VII. CONCLUSION

This paper focused on an analytic approach for reducing the number of paths in MTBDDs and SBDDs. Our approach is based on spectral techniques. Specifically, we analyze and use the autocorrelation coefficients for this aim.

We have shown that the number of paths in decision diagrams is a function of the weighted autocorrelation coefficients at the basis vectors that span the domain of the Boolean functions. Hence, it is possible to minimize the number of paths by constructing an ordered set of basis vectors of high weighted autocorrelation value. A greedy bottom-up linearization algorithm that produces an ordered set of basis vectors, a corresponding linear transformation matrix, and its corresponding linearized function was introduced.

Experimental results on standard benchmark functions clearly demonstrate the efficiency of this approach. Unlike known dynamic reordering procedures designed to minimize the number of paths, the suggested algorithm does not significantly increase the number of nodes in the decision diagrams. However, the proposed algorithm can produce a suboptimal solution. We expect that an hybrid approach that combines the advantages of the proposed analytic method and other known dynamic techniques can produce better results.

ACKNOWLEDGMENT

The authors would like to thank the reviewers whose constructive comments were very useful in improving the presentation of this paper. They thank Dr. S. Stojkovic from the Department of Computer Science, Faculty of Electronics,

University of Niš, Niš, Serbia, for useful discussion in organizing and performing experimental results.

REFERENCES

[1] B. Bollig and I. Wegener, "Improving the variable ordering of OBDDs is NP-complete," *IEEE Trans. Comput.*, vol. 45, no. 9, pp. 993–1002, Sep. 1996.

[2] J. T. Butler, T. Sasao, and M. Matsuura, "Average path length of binary decision diagrams," *IEEE Trans. Comput.*, vol. 54, no. 9, pp. 1041–1053, Sep. 2005.

[3] M. Crocker, X. S. Hu, and M. Niemier, "Defect tolerance in QCA-based PLAs," in *Proc. IEEE ACM Int. Symp. NANOARCH*, 2008, pp. 46–53.

[4] R. Drechsler, W. Gunther, and F. Somenzi, "Using lower bounds during dynamic BDD minimization," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 20, no. 1, pp. 51–57, Jan. 2001.

[5] R. Drechsler, N. Drechsler, and W. Gunther, "Fast exact minimization of BDDs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 19, no. 3, pp. 384–389, Mar. 2000.

[6] E. V. Dubrova and D. M. Miller, "On disjoint covers and ROBDD size," in *Proc. IEEE Pacific Rim Conf. Commun., Comput. Signal Process.*, Aug. 1999, pp. 162–164.

[7] R. Ebendt, W. Gunther, and R. Drechsler, "An improved branch and bound algorithm for exact BDD minimization," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 12, pp. 1657–1663, Dec. 2003.

[8] G. Fey and R. Drechsler, "Minimizing the number of paths in BDDs," in *Proc. Symp. Integr. Circuits Syst. Design*, 2002, pp. 359–364.

[9] G. Fey and R. Drechsler, "Minimizing the number of paths in BDDs: Theory and algorithm," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 1, pp. 4–11, Jan. 2006.

[10] M. Hilgemeier N. Drechsler, and R. Drechsler, "Minimizing the number of one-paths in BDDs by an evolutionary algorithm," in *Proc. 2003 Congr. Evol. Comput.*, vol. 3. 2003, pp. 1724–1731.

[11] J. Jain, D. Moundanos, J. Bitner, J. A. Abraham, D. S. Fussell, and D. E. Ross, "Efficient variable ordering and partial representation algorithm," in *Proc. 8th Int. Conf. VLSI Design*, 1995, pp. 81–86.

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

[13] M. G. Karpovsky, R. S. Stanković, and J. T. Astola, "Reduction of sizes of decision diagrams by autocorrelation functions," *IEEE Trans. Comput.*, vol. 52, no. 5, pp. 592–606, May 2003.

[14] O. Keren, "Reduction of average path length in binary decision diagrams by spectral methods," *IEEE Trans. Comput.*, vol. 57, no. 4, pp. 520–531, Apr. 2008.

[15] O. Keren, I. Levin, and R. S. Stanković, "Linearization of functions represented as a set of disjoint cubes at the autocorrelation domain," in *Proc. 7th Int. Workshop Boolean Probl.*, Sep. 2006, pp. 137–144.

[16] O. Keren, I. Levin, and R.S. Stanković, "Reduction of the number of paths in binary decision diagrams by linear transformation of variables," in *Proc. 7th Int. Workshop Boolean Probl.*, Sep. 2006, pp. 79–84.

[17] L. Macchiarulo, L. Benini, and E. Macii, "On-the-fly layout generation for PTL macrocells," in *Proc. Design, Automation Test Europe*, 2001, pp. 546–551.

[18] C. Meinel and A. Slobodova, "Speeding up variable reordering of OBDDs," in *Proc. Int. Conf. Computer Design*, 1997, pp. 338–343.

[19] C. Meinel, F. Somenzi, and T. Theobald, "Linear sifting of decision diagrams and its application in synthesis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 19, no. 5, pp. 521–533, May 2000.

[20] D. M. Miller, R. Drechsler, and M. A. Thornton, *Spectral Techniques in VLSI Comput.-Aided Design*. Boston, MA: Kluwer, 2001.

[21] A. Mishchenko and M. Perkowski, "Fast heuristic minimization of exclusive-sums-of-products," in *Proc. Reed-Muller Workshop*, 2001, pp. 242–250.

[22] S. Nagayama, A. Mishchenko, T. Sasao, and J. T. Butler, "Exact and heuristic minimization of the average path length in decision diagrams," *J. Multi-Valued Logic Soft Comput.*, vol. 11, no. 6, pp. 437–465, 2005.

[23] W. Porod, C. S. Lent, G. Toth, H. Luo, A. Csurgay, Y. E. Huang, and R. W. Liu, "Quantum-dot cellular nonlinear networks: Computing with locally-connected quantum dot arrays," in *Proc. IEEE ISCAS*, vol. 1. 1997, pp. 745–748.

[24] S. Reda, R. Drechsler, and A. Orailoglu, "On the relation between SAT and BDDs for equivalence checking," in *Proc. Int. Symp. Quality Electron. Design*, 2002, pp. 394–399.

[25] J. E. Rice and J. C. Muzio, "Methods for calculating autocorrelation coefficients," in *Proc. 4th IWSB*, 2000, pp. 69–76.

[26] R. Rudell, "Dynamic variable ordering for ordered binary decision diagrams," in *Proc. Int. Conf. Comput.-Aided Design*, 1993, pp. 42–47.

[27] T. Sasao, J. T. Butler, and M. Matsuura, "Average path length as a paradigm for the fast evaluation of functions represented by binary decision diagrams," in *Proc. 1st Int. Symp. New Paradigm VLSI Comput.*, Dec. 2002, pp. 31–36.

[28] T. Sasao, Y. Iguchi, and M. Matsuura, "Comparison of decision diagrams for multiple-output logic functions," in *Proc. IWLS*, Jun. 2002, pp. 379–384.

[29] G. Toth and C. S. Lent, "Quantum computing with quantum-dot cellular automata," *Phys. Rev.*, vol. 63, no. 5, pp. 0523151–0523159, 2001.

[30] D. Varma and E. A. Trachtenberg, "Design automation tools for efficient implementation of logic functions by decomposition," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 8, no. 8, pp. 901–916, Aug. 1989.

[31] S. Yanushkevich, D. M. Miller, V. Shmerko, and R. S. Stanković, *Decision Diagram Techniques for Micro- and Nanoelectronic Design Handbook*. New York/Boca Raton, FL: Taylor and Francis/CRC Press, 2006.

Osnat Keren received the M.S. degree in electrical engineering from the Technion-Israeli Institute of Technology, Haifa, Israel, and the Ph.D. degree from Tel-Aviv University, Tel-Aviv, Israel, in 1988 and 1999, respectively.

From 1988 to 1994, she held a Chip Design and Senior DSP Engineer position with National Semiconductor, Santa Clara, CA, and from 1999 to 2003 she was the Senior Scientist with Millimetrix Broadband Networks, Givat-Hashlosha, Israel. Since 2004, she has been with the School of Engineering, Bar-Ilan University, Ramat-Gan, Israel. Her current research interests include algebraic and combinatorial methods applied to problems of reliability and efficiency of computer and communication systems, logic design and spectral methods for logic testing and synthesis, and coding theory.

Ilya Levin received the M.S. degree in electrical engineering from the Leningrad Transport Engineering Institute, Leningrad, Russia, in 1976, and the Ph.D. degree in computer engineering from the Institute of Electronics, Latvian Academy of Science, Latvia, in 1987.

He is currently an Associate Professor with the School of Education, Tel Aviv University, Tel Aviv, Israel, and is the Head of the Department of Mathematics, Science and Technology Education. His current research interests include logic design, fault tolerant design, and design automation.

Radomir S. Stanković received the B.S. degree in electronic engineering from the Faculty of Electronics, University of Niš, Niš, Serbia, and the M.S. and Ph.D. degrees in applied mathematics from the Faculty of Electrical Engineering, University of Belgrade, Belgrade, Serbia, in 1984 and 1986, respectively.

He is currently a Professor with the Department of Computer Science, Faculty of Electronics, University of Niš. His current research interests include spectral techniques, switching theory and multiple-valued logic, and signal processing.