FACTA UNIVERSITATIS (NIŠ)
SER.: ELEC. ENERG. vol. 20, no. 3, December 2003, 479-498

Linearization of Multi-Output Logic Functions by Ordering of the Autocorrelation Values

Osnat Keren and Ilya Levin

Abstract: The paper deals with the problem of linear decomposition of a system of Boolean functions. A novel analytic method for linearization, by reordering the values of the autocorrelation function, is presented. The computational complexity of the linearization procedure is reduced by performing calculations directly on a subset of autocorrelation values rather than by manipulating the Boolean function in its initial domain. It is proved that unlike other greedy methods, the new technique does not increase the implementation cost. That is, it provides linearized functions with a complexity that is not greater than the complexity of the initial Boolean functions. Experimental results over standard benchmarks and random Boolean functions demonstrate the efficiency of the proposed procedure in terms of the complexity measure and the execution time.

Keywords: Logic synthesis, spectral technique, autocorrelation function, linear transform, disjoint cubes.

1 Introduction

The linear decomposition approach is a well known technique for efficient implementation of Boolean functions. The linearly decomposed system, see Figure 1, consists of a linear function σ followed by a linearized function f_σ, such that

$$z = \sigma(x) \text{ and } y = f_\sigma(z).$$

Reduction in realization cost is obtained by replacing the original set of input variables by another set of linearly independent variables which are linear combinations of the input variables. The σ defines a linear transformation of variables that minimizes the complexity of the corresponding linearized function.

Manuscript received September 14, 2007.

O. Keren is with Bar Ilan University, Israel (email: kereno@eng.biu.ac.il). I. Levin is with Tel-Aviv University, Israel (email: ilia1@post.tau.ac.il).

The linear part of a Boolean function of n inputs is implemented by two-input XOR gates with complexity of order $n^2/log_2(n)$ and thus has negligible effect on overall complexity of the decomposed function [13].

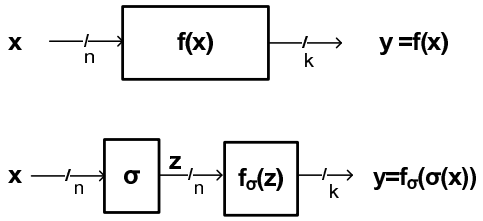

Fig. 1. Realization of a function f: direct realization (top) and realization by using the linear decomposition $f(x) = f_\sigma(\sigma(x))$ (bottom)

In [13] the complexity of the realization is measured by the number μ of adjacent minterms carrying the same output value. For Boolean functions of large number of input variables the realization cost in terms of 2-input gates almost always decreases as the μ increases [22, 25]. The complexity measure μ is the sum of the values of the autocorrelation function of at positions 2^i, $i = 0, \ldots n-1$. Therefore, it is possible to reduce the implementation cost by choosing a linear transformation σ that reorders the autocorrelation values of f so that f_σ has high autocorrelation values at those positions [13].

A Binary Decision Diagram (BDD) represents a Boolean function as an acyclic directed graph. This form of representation is suitable for Boolean functions of large number of input variables and is used by CAD tools for logic synthesis and simulations. The characteristics of the BDD, i.e. the size of the BDD, the number of paths and the average path length, are sensitive to the ordering of the nodes variables [6, 10, 18, 21].

Considerable research has been made for developing both **dynamic** (algorithmic) and **static** (analytic) procedures for minimizing these parameters by reordering variables or by replacing them by a linear combination of input variables. The algorithmic approach involves an heuristic search algorithm, [4, 5, 7, 19, 24] and references therein. In some cases, the dynamic approach may fail to find a better set of variables within a given time limitation. The analytic approach defines a new ordered set of variables by considering the Boolean function properties like the values of the Walsh coefficients [26], the autocorrelation values [14, 17, 23] or the ambiguity of the variables [12]. This paper deals with a static linearization based on ordering the autocorrelation values for functions of large number of input variable for which the known linearization procedures may become impractical due to their high computational complexity.

The autocorrelation function, denoted by $R(\tau)$, can be calculated either accord-

ing to its definition or by using the Wiener-Khinchin theorem.

The efficiency of methods for computing the autocorrelation function depends on the way the function f is specified. In this paper we are interested in functions of large number of inputs which are often represented as a set of cubes. Since any set of cubes can be expanded to a set of disjoint cubes, we assume that the function to be linearized is already given as a set of disjoint cubes. Note that a BDD can be interpreted as a compact disjoint cubes representation where the number of paths in a BDD equals to the number of disjoint cubes. In some cases, the number of disjoint cubes, N, may be exponential in the number of inputs. The technique for manipulating disjoint cubes discussed herein is efficient in terms of processing time and memory consumption only for functions represented by a moderate number of disjoint cubes, namely when $N << 2^n$.

An extensive work has been done in developing efficient methods for calculations of various discrete spectral transforms, including the Walsh transform, of Boolean functions defined by disjoint cubes, see, for instance, [8, 9] and references therein. These methods can be employed to calculate the autocorrelation function following the Wiener-Khinchin theorem. However, the computational complexity of the calculation of $R(\tau)$ based on these methods depends not only on the number of cubes representing the function but also on the number of disjoint cubes representing its spectrum.

Another method for the calculation of $R(\tau)$ for functions represented by disjoint cubes is the tabular technique [1, 2, 20, 29]. The complexity of the method depends on the number of minterms and hence may become impractical in terms of computation time or memory space for function of large number of ON-SET values. In [30] the complexity of the calculation of the autocorrelation values is reduced by performing the calculation entirely in the disjoint cubes domain. A single autocorrelation value is calculated at a time by comparing pairs of disjoint cubes. The computational complexity depends on the number of disjoint cubes and is of order $\mathcal{O}(N^2)$ per autocorrelation value. In [15] the autocorrelation $R(\tau)$ is also calculated by manipulating the initial set of disjoint cubes. However, the calculation there is performed simultaneously for cubes of τ's (rather for a single τ). The autocorrelation function in [15] has a compact representation as a so called arithmetic-SOP. In this paper we use this form of representation to reduce the computational complexity of the proposed linearization procedure.

Due to their high computational complexity [3], static linearization procedures tend to be greedy. The majority of known static linearization methods provide an improvement only **on average**. Namely, at each step a single vector is added to the current set of vectors regardless how it may delimit the set of candidate vectors of future steps.

In [14] a linearization algorithm, called the *K*-procedure, for the reduction of

the size of Binary Decision Diagrams (BDD) was introduced.

At each level, starting from the bottom of the BDD, a linearization is performed by determining the basis of the inertia group for the function represented by the nodes at the upper levels, and then the BDD is paired.

The pairing of the BDD's terminal nodes is an essential step of the K-procedure. This guaranties that the chosen vector is linearly independent in previous vectors. However the complexity of the pairing is exponential with the number of inputs, and therefore the K-procedure may not be applicable for functions of large number of inputs.

A linearizarion algorithm for efficient minimization of Boolean functions represented as a set of disjoint cubes was presented in [30]. The procedure there reduces the computational complexity by: a) use of an heuristic algorithm to define a candidate vector, and b) calculation of the autocorrelation of the chosen candidate vector according to the definition of the autocorrelation function.

Consequently, the procedure can handle functions of large number of inputs. However, the complexity of the linearized function depends on the order of processing the initial set of cubes.

Another linearization algorithm, that works on multi-output functions represented as a set of disjoint cubes, was presented in [15].

The computational complexity of the procedure is polynomial in the number of inputs and in the maximal number of cubes processed per iteration N_{max}.

The drawback of this approach is the value of N_{max}, which may be larger than the initial number of cubes N since a linear transformation of a cube may break it into a number of cubes of a smaller order.

In this paper we introduce a linearization procedure, which is based on ordering the autocorrelation values. The suggested method differs from other known static linearization procedures in:

1. It works directly on a small predefined subset of autocorrelation values and thus has a smaller computational complexity in comparison with existing linearization techniques.
2. Unlike other static greedy algorithms, this procedure **never** increases the implementation cost of the decomposed system of logic functions.

The paper is organized as follows: Section 2 provides mathematical background. The linear decomposition problem and the suggested linearization procedure are presented in Section 3. In section 4 we show that the suggested procedure cannot produce a linearized function of degraded complexity measure. Section 5 includes simulation result of standard benchmarks and random functions. The conclusions summarizing the results are presented in Section 6.

2 Preliminaries

This section provides basic definitions and briefly presents a technique for the calculation of autocorrelation function over disjoint cubes.

2.1 Definitions

A logic unit of n inputs and k outputs can be represented either by a set of k single-output Boolean functions $f : GF(2^n) \rightarrow GF(2)^k$ or by a single multi-output function $f : GF(2^n) \rightarrow GF(2^k)$. In this paper we refer to the function as a multi-output function.

Let $\mathscr{G} = \{0,1,*\}$, and, $p \in \mathscr{G}$. Let a be a Boolean variable we define a^p as

$$a^p = \begin{cases} a & if \quad p = 1 \\ \bar{a} & if \quad p = 0 \\ 1 & if \quad p = * \end{cases} .$$

Let $x = (x_{n-1}, \ldots, x_0) \in GF(2^n)$. A cube $P = (p_{n-1}, \ldots, p_0) \in \mathscr{G}^n$, of order r is a coset comprising the 2^r assignments of x for which the corresponding Boolean product $f_P(x) = \Pi_{i=0}^{n-1} x_i^{p_i}$ equals "1".

The intersection between two cubes P_i and P_j comprises the elements in the intersection of the cosets. Equivalently, the the assignments x for which $f_{P_i}(x) \cdot f_{P_j}(x) = 1$. Two cubes are called disjoint if their intersection is empty.

The representation of a multi-output function f in the cubes domain is a set of N pairs $F = \{(P_i, Y_i)\}_{i=1}^N$ where $P_i \in \mathscr{G}^n$ is a cube and $Y_i \in GF(2^k)$ is the corresponding output. A function is represented as a set of disjoint cubes if any pair of cubes is disjoint. Clearly any non-disjoint set can be expanded into a disjoint set, and therefore without loss of generality, we assume that the system consists of N disjoint (orthogonal) cubes.

The cubes of a multi-output Boolean function can be partitioned into sets having identical output vectors, called characteristic sets. The characteristic set, $F^{(u)}$, $u \in GF(2^k)$, is the set

$$F^{(u)} = \{(P_i, Y_i) | (P_i, Y_i) \in F, Y_i = u\} \tag{1}$$

The Boolean function defined by a characteristic set $F^{(u)}$ is called a characteristic function $f^{(u)}(x)$,

$$f^{(u)}(x) = \begin{cases} 1 & f(x) = u \\ 0 & otherwise \end{cases} .$$

Let $N^{(u)}$ be the number of products $\{P_i\}_{i=1}^{N^{(u)}}$ associated with the characteristic set $F^{(u)}$, $\sum_{u \in GF(2^k)} N^{(u)} = N$. Since F is an orthogonal set of products, so does $F^{(u)}$ and

thus

$$f^{(u)}(x) = \vee_{i=1}^{N^{(u)}} f_{P_i}(x) = \sum_{i=1}^{N^{(u)}} f_{P_i}(x)$$

where $\vee$ stands for OR operation and $\sum$ is an arithmetic summation.

Let $P_i = (p_{n-1}^{(i)}, \ldots, p_0^{(i)})$ and $P_j(p_{n-1}^{(j)}, \ldots, p_0^{(j)}) \in \mathscr{G}^n$ be disjoint cubes. Denote by n_* the number places where $p_k^{(i)} = p_k^{(j)} = *$, for $0 \leq k < n$. The intersection between P_i and P_j (equivalently, $f_{P_i}(x)$ and $f_{P_j}(x)$) is empty. However, there are τ's in $GF(2^n)$ for which the intersection between $f_{P_i}(x)$ and the shifted function $f_{P_j}(x \oplus \tau)$ is not empty. In [15] it was shown that the set of these τ's forms a cube $C_{i,j} = (c_{n-1}^{(i,j)}, \ldots, c_0^{(i,j)}) \in \mathscr{G}^n$ where

$$c_k^{(i,j)} = \begin{cases} 0 & (p_k^{(i)}, p_k^{(j)}) \in \{(0,0),(1,1)\} \\ 1 & (p_k^{(i)}, p_k^{(j)}) \in \{(0,1),(1,0)\} \\ * & \textit{otherwise} \end{cases} \tag{2}$$

and $0 \leq k < n$. The number of common elements in the intersection of $f_{P_i}(x)$ and $f_{P_j}(x \oplus \tau)$ is $V_{i,j} = 2^{n_*}$.

Example 1 *Let $P_1 = (01**1)$ and $P_2 = (00*11)$ be disjoint cubes. The corresponding functions are $f_{P_1} = \bar{x}_4 x_3 x_0$ and $f_{P_2} = \bar{x}_4 \bar{x}_3 x_1 x_0$. The elements in the cosets defined by the cubes P_1 and P_2 are $\{(01001),(01011),(01101),(01111)\}$ and $\{(00011),(00111)\}$, respectively.*

The intersection between the cubes is empty. However, if the cube P_2 is being shifted by $\tau = (01110)$, then,

$$P_2 \oplus (01110) = (01*01)$$

and the intersection between the shifted cube and P_1 has $V_{1,2} = 2^1$ elements. Following Eq. 2, the τ's for which the intersection is not empty are elements of the following cube:

$$C_{1,2} = C_{2,1} = (01**0) \quad \textit{and} \quad V_{1,2} = V_{2,1} = 2^1.$$

Similarly,

$$C_{1,1} = (00**0) \quad \textit{and} \quad V_{1,1} = 2^2,$$
$$C_{2,2} = (00*00) \quad \textit{and} \quad V_{2,2} = 2^1.$$

2.2 Autocorrelation function and complexity measure

The autocorrelation function of a single output Boolean function is defined as

$$R(\tau) = \sum_{x \in GF(2^n)} f(x) f(x \oplus \tau).$$

The definition of the autocorrelation function corresponding to a k-output logic unit depends on how the function is addressed. When the function is referred as a multi-output function $f : GF(2^n) \rightarrow GF(2^k)$, the autocorrelation function is called Total autocorrelation function [13]. The Total autocorrelation function is defined as $R(\tau) = \sum_{u \in GF(2^k)} R^{(u)}(\tau)$ where $R^{(u)}$ is the autocorrelation function corresponding to the characteristic function $f^{(u)}$.

This paper is focused on multi-output functions represented as a set of disjoint cubes. In this case the value of $R^{(u)}(\tau)$ equals to

$$R^{(u)}(\tau) = \sum_{i=0}^{N^{(u)}} \sum_{j=0}^{N^{(u)}} \sum_{x \in GF(2^n)} f_{P_i^{(u)}}(x) f_{P_j^{(u)}}(x \oplus \tau) = \sum_{i=0}^{N^{(u)}} \sum_{j=0}^{N^{(u)}} R_{i,j}^{(u)}(\tau).$$

The autocorrelation function has a compact representation in a PLA-like form [15], i.e. as a set of M pairs

$$R = \{(C_i, V_i)\}_{i=1}^{M}, \tag{3}$$

where $M \leq \sum_u (N^{(u)} + \binom{N^{(u)}}{2}) \leq N^2$, C_i is a cube and $1 \leq V_i \leq 2^n$. Equivalently, the autocorrelation function can be represented by the following **arithmetic** sum

$$R(\tau) = \sum_{i=1}^{M} f_{C_i}(\tau) \cdot V_i. \tag{4}$$

Example 2 *Consider a 4-input 3-output Boolean function* $f : GF(2^4) \rightarrow GF(2^3)$ *specified by the following set of cubes :*

$$F = \left\{ \begin{array}{ccc} ((0100) & , & 0), \\ ((0011) & , & 0), \\ ((1*00) & , & 1), \\ ((0*10) & , & 1), \\ ((0101) & , & 2), \\ ((000*) & , & 2), \\ ((1*1*) & , & 2), \\ ((1*01) & , & 3), \\ ((0111) & , & 4) \end{array} \right\}$$

where a symbol of $GF(2^3)$ *is addressed by its integer value, e.g.* $(011) = 3$. *There are five characteristic functions. The autocorrelation function of* $F^{(0)}$ *is*

$$R^{(0)} = \left\{ \begin{array}{ccc} ((0000) & , & 2^0), \\ ((0000) & , & 2^0), \\ ((0111) & , & 2 \cdot 2^0) \end{array} \right\}.$$

Notice that $R^{(0)}$ comprises two identical cubes (0000). In a conventional PLA, the cubes can be merged by ORing their values, here the cubes are merged by adding their values. Namely,

$$R^{(0)} = \left\{ \begin{array}{ccc} ((0000) & , & 2), \\ ((0111) & , & 2) \end{array} \right\}.$$

The total autocorrelation function consists of the following pairs

$$R = \{\ R^{(0)}, R^{(1)}, R^{(2)}, R^{(3)}, R^{(4)}\ \} = \left\{ \begin{array}{ccc} ((0000) & , & 4), \\ ((0111) & , & 2), \\ ((0*00) & , & 6), \\ ((1*10) & , & 4), \\ ((000*) & , & 2), \\ ((0*0*) & , & 4), \\ ((010*) & , & 2), \\ ((1*1*) & , & 6) \end{array} \right\}.$$

The value of the autocorrelation function $R(\tau)$ for $\tau = (0100)$ is

$$R(0100) = \sum_{i=1}^{8} f_{C_i}(0100)V_i = 0\cdot 4 + 0\cdot 2 + 1\cdot 6 + 0\cdot 4 + 0\cdot 2 + 1\cdot 4 + 1\cdot 2 + 0\cdot 6 = 12$$

Note that in some cases (such as reordering or sub-optimal linearization procedure) only a subset of autocorrelation values is considered. In such cases the storage size can be farther reduced by deleting cubes from the set.

As shown in [13], the complexity measure $\mu(f)$ can be written in terms of the autocorrelation function values : $\mu(f^{(u)}) = \sum_{i=0}^{n-1} R^{(u)}(\delta_i)$ where δ_i stands for the representation of 2^i as a binary vector of length n in base 2, and

$$\mu(f) = \sum_{u \in GF(2^k)} \mu(f^{(u)}) = \sum_{i=0}^{n-1} R(\delta_i). \tag{5}$$

Example 3 *The complexity measure of the function from Example 2 is* $\mu = R(0001) + R(0010) + R(0100) + R(1000) = 6 + 0 + 12 + 0 = 18$

3 Linear Decomposition

3.1 The problem of linearization

As it was mentioned above, the linear decomposition of a Boolean function is superposition of a linear transformation function σ implemented by XOR operations and a non-linear part, f_σ.

The linear transformation function σ can be represented by a nonsingular $(n \times n)$ matrix and thus, $f(x) = f_\sigma(\sigma \odot x)$ where $\odot$ stands for matrix multiplication over $GF(2)$.

The optimization problem is to find for a given function f, a nonsingular matrix σ that permutes the function's values, such that $\mu(f_\sigma)$ is maximal.

In this paper, we interpret the linearization problem as follows:
Determine a nonsingular linear transformation matrix σ that reorders the values of the initial R such that high autocorrelation values are placed at positions 2^i, $i = 0,1,\ldots,n-1$ in R_σ.

Recall that the autocorrelation functions of $f(x)$ and $f_\sigma(x)$ carry the same values but in a different positions [13], i.e.

$$R(\tau) = R_\sigma(\sigma \odot \tau), \text{ or, } R_\sigma(\tau) = R(\sigma^{-1} \odot \tau). \tag{6}$$

Let $\sigma = T^{-1}$, $T = (\tau_{n-1}, \ldots \tau_1, \tau_0)$, then, the complexity measure of the linearized function is

$$\mu(f_\sigma) = \sum_i R_\sigma(\delta_i) = \sum_i R(\tau_i). \tag{7}$$

The matrix σ defines the permutation of the autocorrelation values, i.e. an autocorrelation value at position x in R is relocated to position $\sigma \odot x$ in R_σ. Hence, the autocorrelation values associated with the original vectors δ_i are moved to positions $\sigma \odot \delta_i$ and vectors τ_i of high autocorrelation values are moved to $\sigma \odot \tau_i = \delta_i$.

Clearly, the problem of the construction of a nonsingular matrix σ, and hence nonsingular T is equivalent to the problem of the construction of a set of n base vectors. That is, the initial set of base vectors, $\{\delta_i\}_{i=0}^{n-1}$, that spans the domain of the function is being replaced by another set of base vectors $\{\tau_i\}_{i=0}^{n-1}$ that allows a more compact representation of the function [15].

Example 4 *Consider the function from Example 2. The autocorrelation values of the function are*

$$R = [16,6,0,0,12,6,0,2,0,0,10,6,0,0,10,6].$$

The sum of the autocorrelation values at positions corresponding to positions 2^i is $\mu(f) = 18$. The value of $\mu(f_\sigma)$ is upper bounded by $12+10+10+6 = 38$. However, this value is not achievable by a linear transformation of the input variables. A set of four linearly independent vectors that maximizes $\mu(f_\sigma)$ is the following, $\tau_0 = (0100) = 4$, $\tau_1 = (1010) = 10$, $\tau_2 = (0001) = 1$ and $\tau_3 = (0111) = 7$. This set of τ's defines a non-singular matrix $T = (\tau_3, \tau_2, \tau_1, \tau_0)$ and the optimal linear transformation matrix

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

The permuted autocorrelation function defined by σ is

$$R_\sigma = [16,12,10,10,6,6,6,6,2,0,0,0,0,0,0,0].$$

and $\mu(f_\sigma) = 12+10+6+2 = 30$. Since $\mu(f_\sigma) > \mu(f)$ the linearized function is of a lower complexity than the original function. Table 1 shows the Karnaugh-like maps of the multi-output function and the corresponding linearized function.

Table 1. Karnaugh-like maps of f and f_σ

x_3x_2 / x_1x_0	00	01	11	10
00	2	0	1	1
01	2	2	3	3
11	0	4	2	2
10	1	1	2	2

$f(x_3x_2x_1x_0)$

z_3z_2 / z_1z_0	00	01	11	10
00	2	2	1	4
01	0	2	1	0
11	2	2	1	3
10	2	2	1	3

$f_\sigma(z_3z_2z_1z_0)$

3.2 Linearization algorithm

Known linear decomposition algorithms for Boolean functions of a large number of inputs are greedy and do not guarantee an improvement in the implementation cost.

The linearization procedure described below is still greedy, however it has an inherent property that the linearized function has a μ greater or equal to the μ of the original function (see the performance analysis on Section 4).

Denote by F_0 and R_{F_0} the initial set of disjoint cubes and its corresponding autocorrelation function respectivly.

Let F_i, $1 \le i \le n$, be a local linearly transformed set obtained by applying a local linear transformation matrix σ_i on F_{i-1}, [15], $F_i = \sigma_i(F_{i-1})$. Equivalently, $f_{i-1}(x) = f_i(\sigma_i \odot x)$, or $f_i(x) = f_{i-1}(T_i \odot x)$ where $T_i = \sigma_i^{-1}$. The linearized function obtained by applying iteratively n local transformations is $F_n = \sigma_n(\cdots(\sigma_2(\sigma_1(F_0)))\cdots) = (\sigma_n \odot \cdots \sigma_2 \odot \sigma_1)(F_0) = \sigma(F_0)$.

Let R_{F_i} be the autocorrelation function corresponding to F_i, From Eq. 6 we have,

$$R_{F_i}(\tau) = R_{F_{i-1}}(T_i \odot \tau). \tag{8}$$

Clearly, $R_i(\tau)$ can be computed by applying the local linear transformation matrix σ_i directly on the autocorrelation function $R_{F_{i-1}}$. Namely, let $R_{F_i} = \{C_j^{(i)}, V_j^{(i)}\}_{j=1}^{M_i}$ be the set of M_i cubes that represent the autocorrelation function R_{F_i}, then,

$$R_{F_i} = \sigma(R_{F_{i-1}}) = \{\sigma(C_j^{(i-1)}), V_j^{(i-1)}\}_{j=1}^{M_{i-1}} \tag{9}$$

where $R = R_{F_0} = \{C_j^{(0)}, V_j^{(0)}\}_{j=1}^{M_0}$ and $R_\sigma = R_{F_{n-1}} = \{C_j^{(n-1)}, V_j^{(n-1)}\}_{j=1}^{M_{n-1}}$. Note that in suboptimal linearization procedures, R_{F_0} may be calculated for a smaller set of τ's, i.e. $R_{F_0} \subset R$.

Example 5 *Assume that the predefined set of τ's defined for the function in Example 2 consists of all binary cubes of Hamming weight less or equal to two, i.e.*

$$R_{F_0} = \{C_j^{(0)}, V_j^{(0)}\}_{j=1}^{3} = \left\{ \begin{array}{c} (0*01), 6 \\ (1010), 10 \\ (0100), 12 \end{array} \right\} \subset R.$$

Let $\sigma = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$, *then,* $\sigma(R_{F_0}) = \{\sigma(C_j^{(0)}), V_j^{(0)}\}_{j=1}^{3} = \left\{ \begin{array}{c} (010*), 6 \\ (0010), 10 \\ (0001), 12 \end{array} \right\}$.

The linearization procedure described in Table 2 works on a predefined set of autocorrelation values corresponding to τ's of Hamming weight smaller or equal to *w*. The procedure constructs the linear transformation matrix σ and a set of linearized disjoint cubes representing the linearized function f_σ.

In order to simplify the calculation of σ_i, the matrix T_i is decompose into a permutation matrix $\mathbf{P_i}$ and a linearization matrix $\mathbf{L_i}$, $T_i = \mathbf{P_i} \odot \mathbf{L_i}$ and $\sigma_i = \mathbf{L_i} \odot \mathbf{P_i}$. The matrices $\mathbf{P_i}$ and $\mathbf{L_i}$ are defined in Appendix A.

Table 2. **Linearization procedure**

Set $i = 1$
For all $\tau \in GF(2^n)$, $\|\|\tau\|\| \leq w$ calculate $R(\tau)$.
Set $\sigma = I$
While $i \leq n$
1) If $R(\tau) = 0$ for all candidate τ's then break.
2) Determine τ, $\tau \geq 2^{i-1}$ that maximizes $R(\tau)$.
In case there is more than one τ choose one randomly.
3) Construct the local linear transformation matrix σ_i
(see Appendix A)
4) Apply the local linear transformation on R
5) Apply the local linear transformation on the set of products
6) Update σ, $\sigma = \sigma_i \odot \sigma$
7) Increment i

Note that the local linear transformation may break a cube C into two cubes of a smaller order. However the number of distinct cubes in each step is upper bounded by the restriction on the Hamming weight of the initial set of τ's, i.e. $M_i = |\{C^{(i)}, V^{(i)}\}_{j=1}^{M_i}| \leq \sum_{k=1}^{w} \binom{n}{k}$. Thereby, the complexity of the computation and

Table 3. The autocorrelation function and optimal τ per step without restriction on the Hamming weight of τ.

i	$R_{F_i}(0,1,\ldots 15)$	τ	μ
0-original	[16, 6, 0, 0, 12, 6, 0, 2, 0, 0, 10, 6, 0, 0, 10, 6]	4	18
1	[16, 12, 0, 0, 6, 6, 0, 2, 0, 0, 10, 10, 0, 0, 6, 6]	10	18
2	[16, 12, 10, 10, 6, 6, 6, 6, 0, 0, 0, 0, 0, 0, 0, 2]	4	28
3	[16, 12, 10, 10, 6, 6, 6, 6, 0, 0, 0, 0, 0, 0, 0, 2]	15	28
final	[16, 12, 10, 10, 6, 6, 6, 6, 2, 0, 0, 0, 0, 0, 0, 0]	-	30

representation of the autocorrelation function in a PLA-like form is of order N^2 and the computational complexity of the ordering for $w = 3$ is of order n^4.

Example 6 *Table 3 shows the linearization process of the function from Example 2. It relates to the case where there is no restriction on the Hamming weight of the initial set of τ's, i.e. $w = 4$. The first column is the step number, the second column shows the permuted autocorrelation values and the third and fourth columns are the chosen τ and the μ. Note that the chosen τ written in the i'th row is greater or equal to 2^i.*

Example 7 *The linearization process of f, when the Hamming weight of the initial set of τ's is restricted by $w = 2$, is shown in Figure 2. To simplify the presentation, the set of cubes in Example 5 is expanded into minterms and the minterms in each column are ordered by their corresponding integer value. Figure 2 shows how the autocorrelation values are permuted by applying local transformations on the set of cubes. Each column represents a step. Only the cubes that have integer value $\geq 2^{i-1}$ are considered and may proceed to the next step. The chosen vector at each step is bolded. Due to the restriction $w = 2$ the process stops after three steps. The linearized function has $\mu = 12+10+6+0 = 28$ which is smaller than the maximal possible μ.*

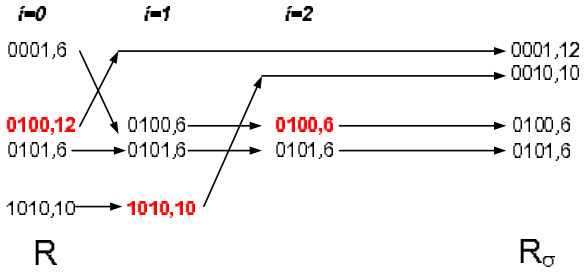

Fig. 2. Local transformations on a subset of cubes $\{C_j^{(i)}, V_j^{(i)}\}_{j=1}^{M_i}$ that represent the autocorrelation functions R_i in Example 7. The initial subset corresponds to $R = R_0$ and comprises cubes of Hamming weight ≤ 2.

4 Performance Analysis

In this section we prove that although the suggested algorithm is sub-optimal and greedy, it cannot derive a linearized function with a complexity measure μ that is smaller than the μ of the original function.

The linearization procedure produces a non-singular linear transformation matrix $\sigma = T^{-1}$, which is a product of n' matrices, $n' \leq n$, $\sigma = \sigma_{n'} \odot \cdots \odot \sigma_2 \odot \sigma_1$. At the i'th step, $(i = 0, \ldots n-1)$, the $(i+1)$'th base vector is determined. By restricting the decimal value of the candidate τ's to be greater or equal to 2^i it is guaranteed that the chosen vector is linearly independent of previous i base vectors $\{\delta_k\}_{k=0}^{i-1}$. Therefore the chosen base vector has to replace one of the vectors $\{\delta_k\}_{k=i}^{n-1}$. Let $\tau = (b_{n-1}, \ldots, b_0)$ be the chosen vector. If the i'th bit of τ, b_{i-1}, is set to "1" then δ_{i-1} is replaced by τ. Otherwise, δ_w, $w \geq i$, is replaced by τ. The index w is determined as follows;
Let J stand for the set $J = \{j | i+1 \leq j \leq n-1,\ b_j = 1\}$, the value of w is

$$w = arg(min_{j \in J} R_i(\delta_j)). \tag{10}$$

In other words, the vector of the smallest autocorrelation value is replaced. For example,

Example 8 *For $i = 4$ and $\tau = (00101011)$ the base vector δ_3 is replaced by τ. If $i = 4$ and $\tau = (00100011)$ then δ_5 is replaced by τ.*

When the i'th bit of τ is set to "1" then the matrix T_i is of the form

$$T_i = \left(\begin{array}{c|c|c} I_{(n-i)\times(n-i)} & \tau & 0 \\ 0 & & I_{(i-1)\times(i-1)} \end{array} \right)$$

and if the i'th bit of τ, is 0, then

$$T_i = \left(\begin{array}{ccc|c|c} I & 0 & 0 & \tau & 0 \\ 0 & 0 & 0 & & 0 \\ 0 & 0 & I & & 0 \\ 0 & 1 & 0 & & 0 \\ 0 & 0 & 0 & & I_{(i-1)\times(i-1)} \end{array} \right).$$

In both cases, the i'th column of T_i is the vector τ that has larger correlation value than any vector in the set $\{\delta_k\}_{k=i}^{n-1}$. The matrix T is the product of the local T_i matrices, $T = T_1 \odot T_2 \odot \cdots \odot T_{n'}$.

Let μ_i stands for the complexity measure of F_i, and let μ_0 and μ_n stand for the complexity measure of the original and transformed functions respectively. Since

$$R_{F_i}(\tau) = R_{F_{i-1}}(T_i \odot \tau),$$

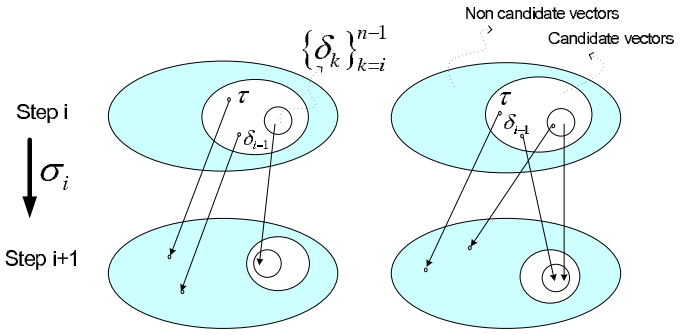

Fig. 3. Behavior of the elements in $GF(2^n)$ in two sequential steps of the procedure

then $\mu_i = \sum_{j=0}^{n-1} R_{F_i}(\delta_j) = \sum_{j=0}^{n-1} R_{F_{i-1}}(\tau_i)$. The following theorems state that the suggested algorithm cannot decrease the μ of the linearized function.

Theorem 1 *For all* $1 \leq i < n$, *the suggested linearization procedure constructs a local matrix* T_i *for which* $\mu_i \geq \mu_{i-1}$.

Proof The proof is by induction.
At the first step, one of the original base vectors, say δ_k, $k \in \{0, \ldots n-1\}$ is replaced by a vector carrying the maximal autocorrelation value. Therefore, $R_{F_0}(\tau) \geq R_{F_0}(\delta_k)$ and

$$\begin{aligned} \mu_1 &= \sum_j R_{F_1}(\delta_j) = \sum_j R_{F_0}(\tau_j) = \sum_{i \neq k} R_{F_0}(\delta_i) + R_{F_0}(\tau) \\ &= \mu_0 + (R_{F_0}(\tau) - R_{F_0}(\delta_k)) \geq \mu_0. \end{aligned} \tag{11}$$

Assume that the inequality holds for the first $(i-1)$ steps. Consider the $\tau = (b_{n-1}, \ldots, b_0)$ of the i'th step. There are two cases: case I, when the i'th bit of τ is equal to "1", and case II, when it equals "0".

Figure 3 illustrates the behavior of the elements in $GF(2^n)$ in two sequential steps of the procedure. The vectors are classified as candidate or non candidate vectors. The non candidate vectors at the i'th step ($(i+1)$'th step) are vectors of integer value $< 2^{i-1}$ ($< 2^i$). The left figure illustrates case I. For this case, it is shown below, that the linearly transformed chosen vector τ and δ_{i-1} become non candidate vectors for the $(i+1)$'th step while the linearly transformed $\{\delta_k\}_{k=i}^{n-1}$ proceed to the next step as candidates. The right figure illustrates case II. In this case, the linearly transformed δ_{i-1} and $\{\delta_k\}_{k=i,k\neq w}^{n-1}$ remain candidates for the $(i+1)$'th step.

Case I. The i'th bit of τ, b_{i-1}, is equal to "1". When the i'th bit is set to "1" then the $\mathbf{P_i}$ matrix is the identity matrix, i.e. $\sigma_i = T_i = \mathbf{L_i}$, and $\mu_i = \mu_{i-1} + (R_{F_{i-1}}(\tau) - R_{F_{i-1}}(\delta_{i-1})) \geq \mu_{i-1}$.

To complete the proof we show that the subset of (the remaining) original base vectors, $\{\delta_k\}_{k=i}^{n-1}$, that were not replaced by vectors of higher autocorrelation values will be considered as candidates in the next step. In other words, we show that these vectors are transformed to binary vectors of decimal value that fulfills the restriction.

The linearly transformed set of the original base vectors is the set $\{\sigma_i \odot \delta_k\}_{k=i}^{n-1}$. From the structure of $\mathbf{L_i}$ we have, $\sigma_i \odot \delta_k = \mathbf{L_i} \odot \delta_k = \delta_k$ for $k \geq i$. Clearly, the decimal values of the elements of the transformed set are greater or equal to 2^i. Hence they remain as candidates for the next step.

Case II. The i'th bit of τ, b_{i-1}, is equal to "0".
In case the i'th bit of τ is zero then δ_w is replaced by τ, i.e. $\mu_i = \mu_{i-1} + (R_{F_{i-1}}(\tau) - R_{F_{i-1}}(\delta_w))$ Since $w \geq i$ then the base vector δ_w has decimal value greater or equal to 2^i has been considered as a candidate. Moreover, it was chosen to be replaced by τ. Namely, $R_{F_{i-1}}(\tau) \geq R_{F_{i-1}}(\delta_w)$ and therefore $\mu_i \geq \mu_{i-1}$.

In this case the original vectors $\{\delta_k\}_{k=i-1}^{n-1}$ excluding the vector δ_w which was replaced by τ are linearly transformed into the vectors $\{\sigma_i \odot \delta_k\}_{k=i-1,k\neq w}^{n-1}$. Indeed, for $k \neq w$ we have,

$$\sigma_i \odot \delta_k = \mathbf{L_i} \odot \mathbf{P_i} \odot \delta_k = \begin{cases} \mathbf{L_i} \odot \delta_k & k \geq i \\ \mathbf{L_i} \odot \delta_w & k = i-1 \end{cases} = \begin{cases} \delta_k & k \geq i \\ \delta_w & k = i-1 \end{cases}$$

Since the transformed set has decimal values greater or equal to 2^i they are available for future use. Moreover, since for all $j \in J$, $R_{F_i}(\delta_w) \leq R_{F_i}(\delta_j)$ the suggested structure of T_i keeps the base vectors of the higher correlation values as candidates.

□
Based on the above,

Theorem 2 *The suggested linearization procedure constructs a linear transformation matrix σ and its corresponding linearized function f_σ for which $\mu(f_\sigma) \geq \mu(f)$.*

5 Experimental Results

In this section we provide simulation results on several benchmarks. The performance of the suggested linearization algorithm is examined in terms of the cost function and the execution time.

Table 4 shows that restricting the Hamming weight of the initial set of τ's to be less or equal to w, does not effect the performance significantly. This justifies the use of $w = 3$.

Table 5 shows for standard benchmark functions the cost function μ of the original and linearized functions. The value of the original function is denoted by

Table 4. Values of the cost function μ as a function of the restriction w on the Hamming weight of τ.

benchmark	n,k	$w=1$	2	3	5	7
sqrt8.pla	8, 4	1164	1284	1268	1286	1286
radd.pla	8,5	824	1304	1304	1304	1304
root.pla	8 ,5	868	932	940	958	958
average		952	1173	1170	1182	1182

μ_{orig}. The μ_{ALG} of the linearized function provided by the suggested linearization procedure is compared to the μ's obtained by [14] and [15]. μ_{upb} is an upper bound on cost function. It is defined as the sum of the n maximal values of the autocorrelation values. This bound is not always achievable.

Table 5. Benchmark results for the complexity measure μ of the original and linearized functions and upper bound on μ

Benchmark	n,k	μ_{orig}	[14]	[15]	μ_{ALG} $w=3$	μ_{upb}
rd53.pla	5,3	50	66	82	82	104
sqr6.pla	6,12	114	114	124	124	196
z4.pla	7,4	320	412	476	476	588
sqn.pla	7,3	292	310	348	346	504
rd73.pla	7,3	308	476	566	566	644
con1.pla	7,2	520	520	524	524	704
5xp1.pla	7,10	512	520	578	574	670
inc.pla	7,9	304	304	324	316	560
misex1.pla	8,7	1472	1536	1664	1664	2048
sqrt8.pla	8,4	1164	1164	1268	1270	1816
radd.pla	8,5	824	1112	1304	1304	1556
root.pla	8,5	868	870	940	948	1712
dist.pla	8,5	638	638	700	690	1058
mlp4.pla	8,8	664	674	734	734	1034
f51m.pla	8,8	884	1076	1244	1204	1536
adr4.pla	8,5	1040	1212	1340	1340	1492
dc2.pla	8,7	820	820	888	894	1310
average		635	695	771	768	1031

Table 6 refers to the benchmark function *s420*. The *s420* represents a Finite State Machine (FSM) that has 19 input variables,16 state variables and two output bits. The FSM is defined by a set of 18 Boolean functions $f^{(i)}$ of 35 variables. In the table: N is the number of disjoint cubes in the representation of $f^{(i)}$ and L_{orig} and L_{lin} stand for the number of literals in SOP representation of the original function and the linearized function as computed by ESPRESSO.

Figure 4 shows the average execution time of the linearization procedures of [14, 15] and the proposed method with $w=3$ as a function of the number of inputs. The execution time was measured on Intel-Centrino, 1.2Ghz, 0.99GB RAM. For

Table 6. FSM s420

i	N	L_{orig}	L_{lin}	% improvement	*sec*
1	24	66	49	25.8	7.8
2	24	49	32	34.7	7.4
3	24	32	15	53.1	7.3
4	26	32	32	reordering	8.1
18	69	151	151	reordering	23.9

the statistics we used random PLA's of four outputs and 50 products. The variance of the measurements was less than 3%. It is clear from Figure 4 that linearization over disjoint cubes ([15] and *ALG*) is more efficient in terms of execution time than linearization based on Wiener-Khinchin theorem (*Kproc*).

Table 7 compares the average execution time of the linearization procedure of [15] and the suggested method (both with $w = 3$) for randomly generated PLAs having 10 to 40 inputs, four outputs 50 products.

Table 7. Average execution-time in seconds for 4-outputs and 50-products PLAs.

inputs	10	15	20	25	30	35	40
[15]	2.64	7.69	21.87	56.38	151.42	339.95	738.03
ALG	0.33	0.69	1.55	3.59	8.37	16.97	31.62

The average execution time for larger number of inputs is given in Figure 5.

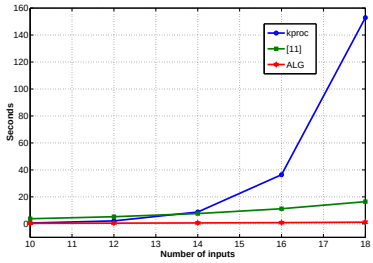

Fig. 4. The average execution time in sec of the K-procedure [14] (labeled as *Kproc*), [15] and the proposed algorithm linearization procedure (*ALG*) as a function of the number of inputs of randomly generated PLAs (4 outputs and 50 products).

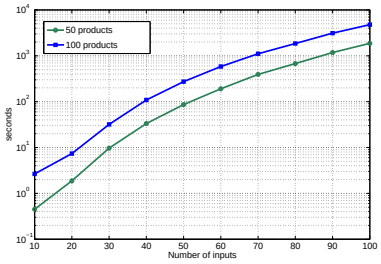

Fig. 5. Average execution time as a function of the number of inputs of a randomly generated PLAs having 4 outputs and 50 and 100 products

6 Conclusion

Linear decomposition is an effective approach for reduction of the implementation cost of a Boolean function. The linearization goal is to construct a linear transfor-

mation function for which the corresponding linearized function is of a minimal implementation cost. The computational complexity of known linearization procedures comes from the calculation of the autocorrelation function for functions of large number of input variables. The present work provides a technique having a small computational complexity. This was achieved by ordering a small predefined set of autocorrelation values. Although the proposed technique is greedy and hence suboptimal, it is proved to derive a linearized function having a μ which is not smaller than the μ of the original function.

The proposed technique is checked by using a set of standard benchmarks. The experimental results clearly demonstrate efficiency of the proposed technique.

Apendix

A Construction of the local linear transformation matrix

The matrices $\boldsymbol{T_i}$ and $\boldsymbol{\sigma}_i$ can be represented as a product of two $(n \times n)$ non-singular matrices $\boldsymbol{P}_i$ and $\boldsymbol{L}_i$, namely,

$$\boldsymbol{T}_i = \boldsymbol{P}_i \odot \boldsymbol{L}_i, \text{ and } \boldsymbol{\sigma}_i = \boldsymbol{L}_i \odot \boldsymbol{P}_i,$$

where $\boldsymbol{P_i}$ is a permutation matrix, ${\boldsymbol{P}_i}^{-1} = {\boldsymbol{P}_i}^T = \boldsymbol{P}_i$, and $\boldsymbol{L}_i$ has ones on its diagonal and a single column of Hamming weight greater or equal one. Clearly, $\boldsymbol{L}_i$ satisfies $\boldsymbol{L}_i^{-1} = \boldsymbol{L}_i$.

Let $\tau = (b_{n-1}, \ldots, b_0)$ be the chosen vector. The permutation matrix $\boldsymbol{P}_i$ and a linearization matrix $\boldsymbol{L}_i$ that are constructed as follows:

Construction of $\boldsymbol{P}_i$:
The structure of the permutation matrix $\boldsymbol{P_i}$ of the i'th step depends on the value of the i'th bit of τ. If it equals "1" then $\boldsymbol{P_i}$ is the identity matrix. Else, $\boldsymbol{P_i}$ is the following permutation matrix:

$$\boldsymbol{P}_i = \begin{pmatrix} \overset{n-1\ldots}{\boldsymbol{I}} & \overset{\ldots w \ldots}{0} & \overset{\ldots}{0} & \overset{\ldots i-1 \ldots}{0} & \overset{\ldots 1,0}{0} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & \boldsymbol{I} & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \boldsymbol{I} \end{pmatrix} \tag{12}$$

Construction of $\boldsymbol{L}_i$:
If the i'th bit of τ is set to "1", then $w = i-1$, otherwise, w is defined according to Eq. 10. Let $I_{(k \times k)}$ be a $(k \times k)$ identity matrix.

$$\boldsymbol{L_i} = \left(\begin{array}{c|c|c} \boldsymbol{I}_{(n-i)\times(n-i)} & \tau - \delta_{i-1} + \delta_w & 0 \\ 0 & & \boldsymbol{I}_{(i-1)\times(i-1)} \end{array} \right).$$

Note that the $+$ and the $-$ are XORs since the calculation is made over $GF(2)$, we use the above notation to emphasize that the vector δ_w is replaced by the vector τ at the *i*'th step.

References

[1] A. E. A. Almaini, P. Thomson and D. Hanson, "Tabular techniques for Reed-Muller logic," *International Journal of Electronics*, vol. 70, no. 1, pp. 23-34, Jan. 1991.

[2] A.E.A. Almaini and L. McKenzie, "Tabular techniques for generating Kronecker expansions," *IEE Proceedings on Computers and Digital Techniques*, vol. 143, no. 4, pp. 205-212, July 1996.

[3] B. Bollig and I. Wegener, "Improving the variable ordering of OBDDs is NP-complete," *IEEE Transactions on Computers*, vol. 45, no. 9, pp. 9931002, Sep. 1996.

[4] J. T. Butler, T. Sasao, and M. Matsuura, "Average Path Length of Binary Decision Diagrams," *IEEE Transactions on Computers*, vol. 54, no. 9 , pp. 1041-1053, Sep. 2005.

[5] R. Drechsler, N. Drechsler, and W. Gunther, "Fast exact minimization of BDDs," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 19, no. 3, pp. 384389, Mar. 2000.

[6] E. V. Dubrova D.M. Miller, "On disjoint covers and ROBDD size," *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 162-164, 1999.

[7] R. Ebendt and R. Drechsler, "Effect of Improved Lower Bounds in Dynamic BDD Reordering," *Transactions on computer-aided design of integrated circuits and systems*, vol. 25, no. 5, pp. 902-909, May 2006.

[8] B.J. Falkowski and S. Kannurao, "Calculation of sign Walsh spectra of Boolean functions from disjoint cubes," *Proc. IEEE International Symposium on Circuits and Systems Circuits and Systems*, vol. 5, pp. 61-64, May 2001.

[9] B.J. Falkowski, I. Schafer and M.A. Perkowski, "Calculation of the Rademacher-Walsh spectrum from a reduced representation of Boolean functions," *Proc. of the Conference on European Design Automation*, pp. 181-186, Sept. 1992.

[10] G. Fey and R. Drechsler, "Minimizing the number of paths in BDDs: Theory and algorithm," *IEEE Trans. on CAD of Integrated Circuits and Systems*, Vol. 25, No. 1, pp. 4-11, 2006.

[11] W. Gunther and R. Drechsler, "Efficient manipulation algorithms for linearly transformed BDDs," *Proc. of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 50-54, 1999.

[12] J. Jain, D. Moundanos, J. Bitner, J.A. Abraham, D.S. Fussell and D.E. Ross, "Efficient variable ordering and partial representation algorithm," *Proc. of the 8th International Conference on VLSI Design*, pp. 81-86, Jan. 1995.

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

[14] M.G. Karpovsky, R.S. Stankovic and J.T. Astola, "Reduction of sizes of decision diagrams by autocorrelation functions," *IEEE Trans. on Computers*, vol. 52, no. 5, pp. 592-606, May 2003.

[15] O. Keren, "Efficient linear decomposition of multi output Boolean functions represented in SOP form," *WSEAS transactions on circuits and systems*, vol. 5, no. 2, pp.219-226, 2006.

[16] O. Keren, I. Levin and R.S. Stankovic, "Linearization of Functions Represented as a Set of Disjoint Cubes at the Autocorrelation Domain," *Proc. of the 7th International Workshop on Boolean Problems*, pp. 137-144, Sept. 2006.

[17] O. Keren, I. Levin and R.S. Stankovic, "Reduction of the Number of Paths in Binary Decision Diagrams by Linear Transformation of Variables," *Proc. of the 7th International Workshop on Boolean Problems*, pp. 79-84, Sept. 2006.

[18] C. Meinel, F. Somenzi, and T. Theobald, "Linear sifting of decision diagrams and its application in synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 5, pp. 521-533 May 2000.

[19] D. M. Miller, R. Drechsler and M. A. Thornton, *Spectral Techniques in VLSI CAD*, Kluwer Academic Pub., 2001.

[20] J. F. Miller, H. Luchian, P. V. G. Bradbeer and P. J. Barclay, "Using a genetic algorithm for optimizing fixed polarity Reed-Muller expansions of boolean functions," *International Journal of Electronics*, vol. 4, no. 76, pp. 601-609, 1994.

[21] S. Nagayama, A. Mishchenko, T. Sasao and J.T. Butler, "Exact and Heuristic minimization of the average path length in decision diagrams," *Journal of Multi-Valued Logic and Soft Computing*, vol. 11, pp. 437.465, 2005.

[22] E.I. Nechiporuk, "On the synthesis of networks using linear transformations of variables," *Dokl. AN SSSR*, vol. 123, no. 4, pp. 610-612, Dec. 1958.

[23] J. Rice, M. Serra, and J.C. Muzio, "The use of autocorrelation coefficients for variable ordering for ROBBDDs," *Proc. of the Fourth Int. Workshop Applications of Reed-Muller Expansion in Circuit Design*, pp. 185-196, Aug. 1999.

[24] R. Rudell, "Dynamic variable ordering for ordered binary decision diagrams," *Proc. International Conference on Computer Aided Design*, pp. 42-47, 1993.

[25] C. E. Shannon, "The Synthesis of Two-Terminal Switching Circuits," *Bell System Technical Journal*, Vol. 28, pp. 59-98, Jan. 1949.

[26] M. Stankovic and S. Stojkovic, "Linear transformation of binary decision diagrams through spectral domain," *Proc. of the 2006 International Workshop on Spectral Methods and Multirate Signal Processing*, Sept. 2006.

[27] R.S. Stankovic, and B.J. Falkovski, "Spectral Interpretation of Fast Tabular Technique for Fixed-Polarity Reed-Muller Expressions," *International Journal of Electronics*, vol. 87, no. 6, pp. 641-648, June 2000.

[28] R. Stankovic and M.G.Karpovsky, "Remarks on calculation of autocorrelation on finite dyadic groups by local transformations of decision diagrams," *Lecture Notes Springer and Verlag*, 2005.

[29] E.C.Tan and H. Yang, "Fast tabular technique for fixed-polarity Reed-Muller logic with inherent parallel processes," *International Journal of Electronics*, vol. 85, no. 4, pp. 511-520, Oct. 1998.

[30] D. Varma and E.A. Trachtenberg, "Design automation tools for efficient implementation of logic functions by decomposition," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 8, pp. 901-916, Aug. 1989.