CONSTRUCTION OF PLANAR BDDS BY USING LINEARIZATION AND DECOMPOSITION[1]

Ilya Levin
School of Engineering
Bar-Ilan University
Ramat Gan 52900
Israel

Radomir S. Stankovic
Dept. of Computer
Science Faculty of
Electronics
Beogradska
18 000 Nis
Serbia

Mark G. Karpovsky
Dept. of Electrical and
Computer Engineering
8 Saint Marry's Street
Boston University
Boston, Ma 02215 USA

Jaakko T. Astola
Tampere Int. Center
for Signal Processing
Tampere University of
Technology
FIN-33101, Finland

ABSTRACT

In VLSI design, crossings of interconnections occupy space and cause delay. In particular, it is desirable to have planar networks for FPGA synthesis and sub-micron LSIs, since delays in the interconnections and crossings are comparable to the delays for logic circuits. Binary decision diagrams (BDDs) provide a simple technology mapping, and planar DDs result in planar networks.

Two different approaches for constructing the planar BDDs are presented in the paper. The first approach is oriented to constructing a linearly transformed BDD and based on calculation of Walsh transform spectral coefficients. The second approach is based on constructing a newly introduced concurrent BDD (CBDD).

The paper presents corresponding algorithms of constructing planar BDDs for both of the approaches. It is shown that integration of the linearization technique into the decomposition technique, and vice versa, leads to optimization.

1. INTRODUCTION

Binary Decision Diagrams (BDDs) are a standard part of many CAD systems in logic design, signal processing, and other fields where efficient, in terms of space and time, manipulation of a BDD representation for a given function *f* is usually estimated by the number of non-terminal nodes in the BDD for *f*, denoted as the size of BDD(*f*). The size of a BDD is very sensitive to the order of variables, ranging from the polynomial to the exponential complexity for the same function for different orders of variables. Therefore, majority of approaches to the reduction of sizes of BDDs are related to development of efficient algorithms for reordering of variables, see for example, [1], [2]. Linearly transformed BDDs are defined by allowing linear combinations of variables [3].

Another important criterion in using the BDD based techniques is planarity of the BDD.

Networks without crossings are advantageous in synthesis with Field Programmable Gate Arrays (FPGAs) [4], since crossings produce considerable delays, and delays in interconnections is one of the most important problems for FPGAs. Furthermore, planar networks are desirable in sub-micron LSIs, since delays in the interconnections and crossings are comparable to the delays in logic circuits. Planar DDs [5], [6] result in planar networks.

Notice that a planar decision diagram can be derived by the reduction of a decision tree if sharing of isomorphic subtrees is restricted to subtrees rooted at neighboring nodes at the same level in the Decision Tree (DT). Restrictions of the order of labels and values of constant nodes can be removed to enlarge the class of functions with planar diagrams. However, in what follows, these restrictions will be applied to the decision diagrams considered.

Planar decision diagrams have been studied in a number of works ([7], [8], [9]). In particular, [8] has derived necessary conditions for planarity in decision diagrams of certain functions. In [9], these results have been extended by completely characterizing symmetric functions with planar decision diagrams, and with the motivation that such functions are an important set of functions being an indispensable part of arithmetic circuits. In [10], [11], Linear Decision Diagrams (LDDs) that are planar by definition have been proposed as models for efficient computation of multiple-valued functions. These decision diagrams are based on the corresponding representations of logic functions by arithmetic polynomials [12].

Complexity of a BDD representation, for a given function *f*, is usually estimated by the number of non-terminal nodes. Reordering and linear transformations of variables are common methods to reduce the size of BDDs.

In this paper, we consider construction Binary Decision Diagrams by using two different approaches: a) linear transformation through Walsh transform coefficients calculation; and b) decomposition into a network of smaller BDDs operating concurrently.

Notice that BDD are aimed at representing Boolean, which means two-valued functions. Therefore, constant nodes in BDDs show two different values, usually encoded by 0 and 1. However, to extend decision diagram representations to integer and complex-valued functions, the Multi-terminal binary decision diagrams (MTBDDs) are introduced by allowing integers and complex numbers

[1] The work was supported by BSF under Grant 2002259, by the Academy of Finland, Finnish Center of Excellence Program, Grant No. 44876, and the EXSITE Project No. 51520.

for constant nodes [13]. It is clear that MTBDDs actually reduce to BDDs when representing two-valued functions. The difference is basically in the interpretation of values of constant nodes as Boolean values or integers 0 and 1. Since we wish to exploit advantages of spectral methods, where a Boolean function is converted into its integer-valued spectrum, we use the term MTBDDs uniformly for diagrams representing either functions or their spectra. Moreover, for spectral processing of logic functions we may prefer encoding (1,-1) instead of (0, 1) to make the functions more compatible with the basic functions in the transforms.

We show that in many cases, the linear transformation based technique provides planar BDD implementations with a relatively small number of nodes. We have noted, however, that in two important situations the technique is inefficient. The first situation is when no linear transformation exists to obtain the planar BDD with a minimized number of nodes and, consequently, the corresponding planar solution requires significant overhead. The second problematic situation relates to the case when a given function is defined by its sum-of-products form with a large number of variables. In such a case, the linear transformation based technique is simply inapplicable due to complexity of the calculations.

We show that both of the above problems can be resolved by using a newly introduced decomposition technique. This technique is based on implementing a BDD in the form of a number of component BDDs operating concurrently. The decomposition allows applying the linearization procedure to each of the component BDDs separately and, consequently, achieving the planar solution. The main idea of the paper is to combine the linearization and the decomposition approaches for synthesis of planar BDDs with the minimized number of nodes.

The paper is organized as follows. The second Section of the paper is devoted to construction of a planar BDD from a regular BDD by using a linear transformation. The decomposition approach based on constructing a network of separate concurrent BDDs is described in Section 3. Section 4 shows integration of the decomposition technique into the linearization method for a BDD-form of representation of a given function. In turn, Section 5 illustrates integration of the linearization into the decomposition technique for a PLA-form of representation of a given function. Conclusions are given in Section 6.

2. CONSTRUCTING PLANAR BDDS BY USING LINEARIZATION

The method to construct Linearly transformed BDDs (LT-BDD) by Walsh coefficients exploits the property that when a given function f is decomposed with respect to the EXOR sum τ of variables corresponding to the maximum value of the Walsh coefficient as $f = \overline{\tau} f_0 \oplus \tau f_1$, the cofactors f_0 and f_1 tend to be simple. Therefore, it may be expected that the LT-BDDs can be represented by BDDs with small number of nodes. Moreover, it was shown in [14] that the linearization may lead to a planar implementation of BDDs.

Definition 1. A Binary Decision Diagram is planar if there is no crossings of edges connecting non-terminal nodes, under assumption that edges labeled by x_i and $\overline{x}_i$ emerge to the left and to the right of a node, respectively, constant node 0 is to the left of the constant node 1, and all edges are directed downwards throughout their length, which precludes arcs that extend around the root node or around constant nodes.

Due to the relationship between Walsh functions and linear switching functions, the following algorithm to construct LT-BDDs can be used.

Algorithm 1 *(Construction of LT-BDD)*

1. Given: an n-variable switching function f in the $\{1,-1\}$ *encoding. Calculate the Walsh spectrum.*

2. Find the Walsh coefficient $S_{f,\max}(w)$ *of the maximum absolute value, except the coefficient for w* = 0. *Declare w* = w_{max} *and write its binary representation, i.e.,* $w_{\max} = (w_1, \ldots, w_n)$.

3. Determine a linear function $\tau = \oplus_r x_r$,*, where r goes over the indices of* $w_j = 1$.

4. Determine co-factors for f with respect to τ, i.e., determine the subfunction of $n-1$ *variables where τ* =0 *and* 1*, respectively.*

5. Create a node whose outgoing edges point to the cofactors f_0 *and* f_1 *and label its edges by* $\overline{\tau}$ $\overline{\tau}$ *and* τ *, respectively.*

6. Repeat the Steps 1 *to* 5 *for co-factors, for i* = 1*, to n.*

Assignment of variables to the edges in LT-BDDs can be performed by using the following algorithm.

Algorithm 2 *(Assignment of labels to the edges)*

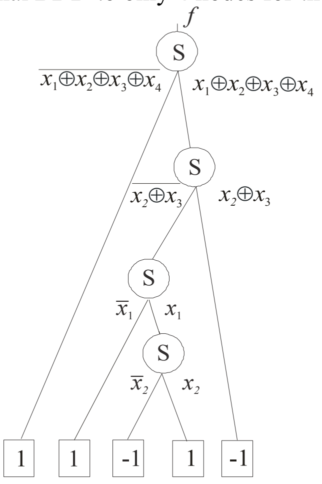
1. If at a node, at the i-th level, decomposition is performed with respect to a linear combination $x_k \oplus x_q \oplus \ldots \oplus x_r$,

where $k < q < \ldots < r$ *, relate the variable* x_r *to the level i and eliminate it from further considerations.*

2. Repeat the Step 1 to all the levels in the BDD starting from the root node.

Example 1 For a four-variable function f, defined as $\mathbf{F} = [1,1,-1,1,-1,1,1,1,-1,1,1,-1,1,-1,1,1]^T$ the Walsh spectrum is $S_f = [6,-2,-2,-2,-2,-2,6,-2,2,-6,2,2,2,2,2,10]^T$.

and the coefficient with the maximum absolute value is $S_{f1}(15) = 10$, and since $w_{\max} = (1111)$, we determine a linear function $\tau_1 = x_1 \oplus x_2 \oplus x_3 \oplus x_4$, where each x_i corresponds to the appearance of coordinate i with value 1 in the binary representation for $w_{\max}$. The co-factors of f with respect to τ_1 are $F_0 = [1,1,1,1,1,1,1,1]^T$, and $F_1 = [1,-1,-1,1,-1,-1,-1,1]^T$. We create a node where outgoing edges point to subfunctions F_0 and F_1 and edges are labeled by $\overline{x_1 \oplus x_2 \oplus x_3 \oplus x_4}$ and $x_1 \oplus x_2 \oplus x_3 \oplus x_4$ respectively. The variable x_4 is eliminated from further considerations. Since F_0 is a constant function, the edge $x_1 \oplus x_2 \oplus x_3 \oplus x_4$ points to the constant node 1 directly. The Walsh spectrum for $\mathbf{F}_1$ is $S_{f_1} = [-2,-2,-2,6,2,2,2,2]^T$.
The maximum Walsh coefficient for $\mathbf{F}_1$ is 6 for $w_{\max} = (011)$, thus we determine $\tau_2 = x_2 \oplus x_3$, and perform decomposition of f_1 into co-factors of two variables $\mathbf{F}_{1,0} = [1,1,-1,1]^T$ and $\mathbf{F}_{1,1} = [-1,-1,-1,-1]^T$. Since $\mathbf{F}_{1,1}$ is a constant function, we proceed with decomposition of $\mathbf{F}_{1,0}$. This co-factor depends on variables x_1 and x_2 only, since x_3 was the last variable in τ_2. The Walsh spectrum of it is $S_{f_{1,0}} = [2,-2,2,2]^T$. Since all the coefficients in this spectrum have the same absolute value, we can chose, for example, $\tau_3 = x_1$ and we do not have to continue the decomposition. In this way, we derive the LT-BDD for f as shown in Fig. 1. This LT-BDD represents f as $f = (x_1 \oplus x_2 \oplus x_3 \oplus x_4)\left(\left(\overline{x_2 \oplus x_3}\right)x_1\bar{x}_2 \oplus (x_2 \oplus x_3)\right)$. Thus, the proposed linearization by Walsh coefficients resulted in the reduction of a size of the BDD for f from 9 nodes for the original BDD to only 4 nodes for the LT-BDD of Fig.



f
S
$\overline{x_1 \oplus x_2 \oplus x_3 \oplus x_4}$
$x_1 \oplus x_2 \oplus x_3 \oplus x_4$
S
$\overline{x_2 \oplus x_3}$
$x_2 \oplus x_3$
S
$\bar{x}_1$
x_1
S
$\bar{x}_2$
x_2
1
1
-1
1
-1

1.

Figure 1. LT-BDD for *f* from Example 1

Table 1.

f	*n*	*BDD*	*BDDv*	*BDDr*	*LT-BDD*
5xp-1	7	14	11	35	13
5xp-4	7	16	11	19	15
9sym	9	33	33	33	57
apex4-10	9	95	91	193	148
clip-1	9	37	34	67	35
clip-2	9	58	42	156	45
clip-3	9	73	32	196	51
clip-4	9	76	36	169	27
clip-5	9	36	36	36	22
con-1	7	12	11	11	16
con-2	7	8	7	8	8
ex1010-2	10	155	148	390	156
ex1010-8	10	154	147	377	209
rd73-3	7	16	16	16	16
rd-84-1	8	32	25	54	82
rd-84-2	8	25	22	169	15
rd-84-4	8	19	19	22	59
sao2-1	10	46	32	75	32
sao2-2	10	48	34	85	31
squar5-1	5	5	5	6	5
z5xp1-3	7	20	14	22	12
z5xp1-6	7	15	9	21	9
av.		45	36	96	46

Table 1 (taken form [14]) compares the sizes of BDDs for the initial (BDD) and optimal (BDDv) order of variables, planar BDDs (BDDr), and planar LT-BDDs produced by decomposition with respect to Walsh coefficients (LT-BDD). With this comparison we estimate impact of keeping decision diagrams planar and efficiency of the linearization of planar BDDs by Walsh coefficients under this restriction as in the definition of planar DDs. As can be seen, on the average, LT-BDDs are for 48% smaller than planar BDD, and for 2% and 26% larger than BDDs for initial and optimal order of variables. It should be noticed that, compared to planar BDDs, planar LT-BDDs, when smaller, are considerably smaller (about 70% on average), and when larger, they are just for about 20% larger.
It should be noted that the above technique can directly be applied not only to regular BDDs but also to multi terminal

BDDs (MTBDDs). We will use the MTBDDs as the main subject of inquiry in the remaining part of the paper.

3. CONCURRENT MTBDD

There are two important cases when the above-described linear transformation technique is ineffective for constructing planar BDDs. They are the following.

1. The initial function is defined by a regular BDD, a planar BDD is then obtained there-from, but no linear transformation exists to minimize the number of nodes of the planar BDD. Consequently, the corresponding planar solution has significant overhead.
2. The initial function is defined by its non-disjoined sum-of-products form (PLA) with a large number of variables. In such cases, the linear transformation based technique is inapplicable due to complexity.

In this paper, we propose achieving the property of planarity of BDDs by using a new approach. This new approach is based on decomposing the MTBDD into a number of concurrently operating component MTBDDs.

For this aim, we introduce a *concurrent* binary decision diagram. The *concurrent MTBDD* (CMTBDD) can be constructed by combining several MTBDDs using the parallel and the series operations as follows:

1) Series connection: replacing one terminal node of one MTBDD with another MTBDD.

2) Parallel connection: connecting inputs of the roots of two or more MTBDDs.

The above two operations allow us to define a CMTBDD recursively as follows:

1. A single decision node is an CMTBDD.
2. Series connection of two CMTBDD is an CMTBDD.
3. Parallel connection of two CMTBDD is an CMTBDD.

This section introduces a notion of D-polynomial, which provides a convenient mathematical model for the manipulation of the CMTBDD. It will be shown that the parallel and series connections of the CMTBDD can be modeled as a product and substitution of D-polynomials representing those CMTBDD.

Consider an *n*-input, *m*-output completely specified Boolean function $F : X^n \to Z^m$, where $X \in \{0,1,*\}$ and $Z \in \{0,1\}$. Let F be initially represented in minimized (prime and irredundant) sum-of-product (SOP) form, where each output Z_i is written as a logical sum of product terms: $Z_i = \sum_{I(i)} \alpha_j$; where $I(i)$ is an index set of implicants associated with output Z_i. Implicants can be shared between different outputs. We will refer to all α_j as α-functions.

Let Y_i be the terminal node associated with a product term α_i. Y_0 will denote a dummy output function, which does not produce any output.

Definition 2. *D-polynomial* is a polynomial defined over a set of terminal nodes $Y_i \sum_i \alpha_i Y_i + \alpha_0 Y_0$, whose coefficients α_i satisfy the following conditions

a) $\sum_i \alpha_i + \alpha_0 = 1$ (completeness) and

b) $\alpha_i \,\&\, \alpha_j = 0, \forall i \neq j$ (orthogonality).

D-binomial is a special case of D-polynomial, with exactly two orthogonal terms, $D = \alpha_1 Y_1 + \alpha_0 Y_0$

Using the terminology of logic synthesis, explicit α-functions α_i represent the *ON-set* of the function, while the implicit coefficients α_0 represent the *OFF-set.*

Thanks to the orthogonality between α-functions, a D-polynomial can be naturally implemented by a binary decision tree whose nodes represent input variables x_i, and whose leaf cells represent the corresponding terminal node Y_1. It is clear that, without any restriction, D-polynomials can be associated with specific MTBDDs. Obviously, such a MTBDD is planar.

A D-polynomial D_i can be interpreted as follows. If α_1^i evaluates to 1, then $D_i = Y_j$. If all of explicit functions α_1^i are equal to 0, then $D_i = Y_0$ which means that no output is produced.

Define a product of two D-polynomials.

Definition 3. Let $D_1 = \sum_i \alpha_i^1 Y_i + \alpha_0^1 Y_0$, and $D_2 = \sum_i \alpha_i^2 Y_i + \alpha_0^2 Y_0$. The *product* of D_1 and D_2, denoted as $D_1 \circ D_2$.is defined as follows $D_1 \circ D_2 = \sum_{ij} \left(\alpha_i^1 \,\&\, \alpha_j^2\right) Y_i Y_j$,

over each pair of terms from D_1 and D_2, including the implicit terms $\alpha_0^1 Y_0$ and $\alpha_0^2 Y_0$. Here $\alpha_i^1 \,\&\, \alpha_j^2$ is a logic product of the corresponding α-functions and $Y_i Y_j$ is a concatenation of the respective terminal nodes. Interpretation: when $\alpha_i^1 \,\&\, \alpha_j^2$ evaluates to 1, both Y_i and Y_j are computed concurrently.

The following two theorems, given here without the proof for the limited space, are important in study and applications of D-polynomials.

Theorem 1. An arbitrary D-polynomial D_i can always be represented as a product of D-binomials $D_1 = \sum_i \alpha_i^1 Y_i + \alpha_0^1 Y_0 = \prod_j \left(\alpha_i^1 Y_i + \alpha_0^1 Y_0\right)$

where $\alpha_0^j = \overline{\alpha_0^i}$, and $\prod_j \alpha_{i0}^j = \alpha_0^i$.

We will call expression (2) a *binomial form* of the D-polynomial. The terms of this expression will be called *terminal binomials.*

An important conclusion from the above theorem is that the product of D-polynomials can be always presented as a product of the corresponding terminal binomials. Subsequently, the terminal binomials can be multiplied to obtain higher level D-polynomials. Obviously, there are several ways to group (multiply) terminal binomials to form a D-polynomial. Different grouping of terminal binomials yields different MTBDDs, resulting in different implementations of a Boolean function. This fact forms the basis of the proposed decomposition approach; an implementation of planar MTBDDs with minimum number of nodes.

Theorem 2. A multiple-output Boolean function can always be implemented as a product of D-polynomials.

Conclusion: CMTBDD corresponding to such an implementation can be realized as a parallel connection of MTBDDs corresponding to the individual D-polynomials.

4. COMPLEMENTING LINEAR TRANSFORMATION WITH DECOMPOSITION

In this section, we discuss a case when the linearization technique discussed in Section 2 is inapplicable or ineffective for synthesis of a planar MTBDD. This situation is possible when no linear transform exists that allows transforming an initial MTBDD into the planar form with the minimized number of nodes. In this case we recommend checking whether the MTBDD can be firstly decomposed into a number of CMTBDDs such, that at least one of these CMTBDDs is transformable into the planar form.

The example given below illustrates that the answer to this question can be positive.

Example 3. Let *D₀* be the *D*-polynomial:
$D_0 = \bar{x}_1\bar{x}_2\bar{x}_3Y_{14} + \bar{x}_1\bar{x}_2x_3Y_{15} + x_1\bar{x}_2\bar{x}_3Y_{24} + x_1\bar{x}_2x_3Y_{25} +$ Let's
$\bar{x}_1x_2\bar{x}_3Y_{25} + \bar{x}_1x_2x_3Y_{26} + x_1x_2\bar{x}_3Y_{35} + x_1x_2x_3Y_{36}$
try to apply the linear transformation described in Section 2. For the application of the Walsh spectrum based procedure, we encode
$Y_{14} = 1, Y_{15} = 2, Y_{24} = 3, Y_{25} = 4, Y_{26} = 5, Y_{35} = 6, Y_{36} = 7$.
Thus, these coefficients are represented by a vector $Y = [1, 2, 4, 5, 3, 4, 6, 7]^T$. It is clear, that in this case the linearization procedure is inefficient since there just one pair of equal values and it belongs to different subtrees.

Let us now apply the decomposition described in Section 3. According to the decomposition technique the initial D-polynomial D_0 can be presented as the product of two D-polynomials D_1, D_2 as follows:
$D_0 = D_1 \circ D_2 = \left(x_1x_2Y_3 + \bar{x}_1x_2Y_2 + x_1\bar{x}_2Y_2 + \bar{x}_1\bar{x}_2Y_1\right)$ Here
$\circ \left(x_2x_3Y_6 + \bar{x}_2x_3Y_5 + x_2\bar{x}_3Y_5 + \bar{x}_2\bar{x}_3Y_4\right)$
$Y_{ij} = Y_iY_j$, which is a concatenation of Y_i and Y_j. Let us now apply the linearization procedure to D-polynomials D_1, D_2 separately. For D_1 we have: $x_2 = x_2 \oplus x_1$. For D_2 we have: $x_2 = x_2 \oplus x_3$. Finally, for the initial D-polynomial D_0 we have the implementation in the form of concurrent LT-BDD as it is shown in Fig. 2.

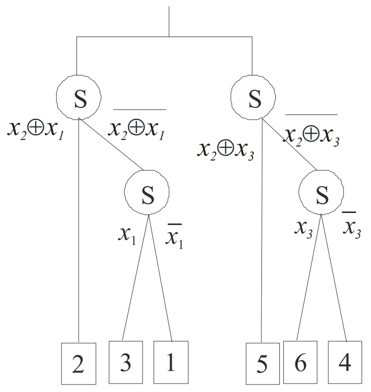


Figure 2. Implementation of the exemplary concurrent LT-BDD for Example 3.

As can be seen from the example, the total number of nodes is reduced considerably by using the decomposition. The total number of nodes in the example is only 4, while in the initial MTBDD implementation we had 7 nodes and the linearization technique was inapplicable.

At this stage, the content of this section should be considered as a motivation example. We don't yet provide a ready-made method for the BDD decomposition. Investigation of signs for such decomposition and development of the suitable method are in the main stream of our present research.

5. COMPLEMENTING LINEARIZATION WITH DECOMPOSITION

The present section deals with the case when a given function is defined in a form of non-disjoint sum of products and when the total number of input variables is large. In other words, the function is defined by a sparse PLA form, where the majority of the cells of the AND array are "don't cares". It is clear, that the linear transformation technique is inapplicable in such cases due to complexity of the task.

In this section we propose a decomposition technique that is based on the concurrent BDD representation and D-polynomials algebra observed in Section 3. The proposed method is based on decomposition of the given function into the network of concurrent BDDs, followed by linearization of the resulting component BDDs independently.

According to proposed approach, the set of logic blocks forming the component BDDs is extracted from a set of product terms representing the ON-set of the given function. It is followed by a hierarchical decomposition of the blocks into a common header and a set of block fragments. When forming the block fragments the same terminal nodes are likely to be included into the same block fragments to provide effective linearization on the next step of the technique. The remaining set of product terms, not included in the block, is called a *remainder*. (Fig. 3 shows a structure of the exemplary MTBDD obtained by the decomposition.)

The header is a fragment (subset of rows and columns) of the PLA table composed of *prefix* variables. The header is selected in such a way as to satisfy the following condition: it must contain at least one column without "don't cares" at each step of the Shannon expansion (along the prefix variables). This strategy helps to guarantee the planarity of each of the component MTBDDs and to limit the number of nodes in the resulting MTBDD. By construction, the block header is a logic function whose ON-set is a superset of the ON-sets of logic functions associated with the individual blocks. It will be implemented as an MTBDD whose internal nodes are associated with the prefix variables. The terminal nodes of the MTBDD represent the block fragments, each to be implemented as a separate MTBDD. The method will be illustrated using the following example adopted from the *misex1* MCNC benchmark set.

Example 4. Consider a logic function described by the following PLA notation with a set of input variables $X = \{x_1, \ldots, x_8\}$, and a set of binary output variables. Output vectors are "1-hot" coded and presented in the PLA table as decimal numbers.

1) 0111----	1	17) 010-----	5
2) 1010----	1	18) 0-11----	5
3) 010-----	2	19) 0-10----	5
4) 0011----	2	20) 0-00----	5
5) 1001----	2	21) 001----	5
6) 001--1--	2	22) 010-----	6
7) 0-00--1-	2	23) 0-11----	6
8) 01-1----	3	24)1001----	6
9) 1001----	3	25)1010----	6
10) 010-1---	3	26) 001--1--	6
11) 0010-0--	3	27) 0-00--1-	6
12) 0000--0-	3	28) 01-1----	7
13) 1010----	4	29) 1001----	7
14) -010--1-	2	30) 1010----	7
15) 010----0	4	31) 010-0---	7
16) 01000---	4	32) -010-0--	6

The product terms of the table are divided into *Block-set* and *Remainder* as shown in Figure 3.

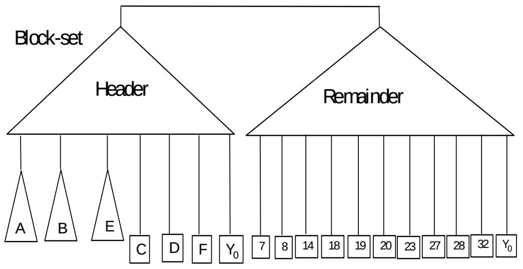


Figure 3. General structure of concurrent MTBDD for Example 4.

Numbers of product terms included in Block-set are: 1-6, 9-13, 15-17, 21, 22, 24-26, 30-32.

The Block-set consists of a header and block-sets A-F. Prefixes associated with blocks A - F of Block-set form the ON-set of the header function:

A: 010-	D: 1001
B: 001-	E: 0000
C: 1010	F: 0111

The block-set can hence be written as the following D-polynomial:

$$D_B = \bar{x}_1 x_2 \bar{x}_3 A + \bar{x}_1 \bar{x}_2 x_3 B + x_1 \bar{x}_2 x_3 \bar{x}_4 C + \\ x_1 \bar{x}_2 \bar{x}_3 x_4 D + \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 E + \bar{x}_1 x_2 x_3 x_4 F ,$$

where A, B, C, D, E, and F represent the block fragments. The entire logic function can be written as a product of D_B and the D-polynomial representing the Remainder. The header can be implemented directly as a planar MTBDD.

Notice that some of the block fragments (in our case those associated with blocks C, D, F) represent the final product terms of the given function. This is because the block fragments obtained by extraction of the respective prefixes are composed of trivial product terms that contain only "don't cares". In our example, node C is associated with product terms 2, 13, 25, 30, node D with product terms 5, 9, 21, 24, 29, and node F with product term 1. The non-trivial blocks (A, B, E), can be implemented directly as a planar MTBDD. The MTBDD of the header can be similarly derived. The algorithm can continue by recursively decomposing the block sets and the remainders as needed (until no planar headers can be found).

The Remainder includes the following product terms.

7) 0-00---1-	2	20) 0-00-----	5
8) 01-1-----	3	23) 0-11-----	6
14) -010---1-	2	27) 0-00---1-	6
18) 0-11-----	5	28) 01-1-----	7
19) 0-10-----	5	29) -010---1-	6

In Figure 3, rectangular nodes represent the product terms; in the Remainder they are labeled with the corresponding product term numbers. The triangular nodes represent the block fragments that are yet to be implemented.

To introduce the linear transform technique to component MTBDDs of the exemplary concurrent MTBDD, we apply the linearization to the Remainder (Fig. 4). Notice that linearization of the Block set is unnecessary due to its simplicity and planarity after the decomposition.

The Remainder represents a function of four variables x_1, x_3, x_4, x_7. For performing the linearization we will handle the header of the Remainder. Consequently, x_7 has to be eliminated from the future considerations as an internal variable of a component MTBDD (not belonging to the header).

The header represents a function of three variables x_1, x_3, x_4. For the header we define two new terminal nodes 1 and 2 (see Fig. 4). Names of these "new" terminal nodes are highlighted by the bold-italic font to distinguish them from the original terminal nodes. The function of the header is expressed by vector $R = [2, 2, 0, 0, 0, 1, 1, 0]^T$ (the order of the variables is: $x_4\ x_1, x_3$). We transform this function by replacing x_3 with linear combination $x_3 = x_1 \oplus x_3$. This transformation reorders elements of the vector R into a new vector $R_\sigma = [2, 2, 0, 0, 0, 1, 0, 1]^T$. The corresponding LT-MTBDD of the header is shown in Fig. 5. In our example, the above transformation results in reduction of the number of the header's nodes from 5 to 3.

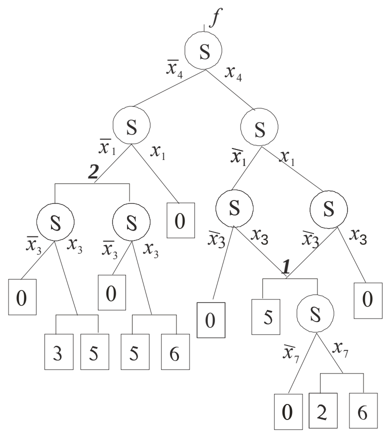


Figure 4. CMTBDD of the Remainder.

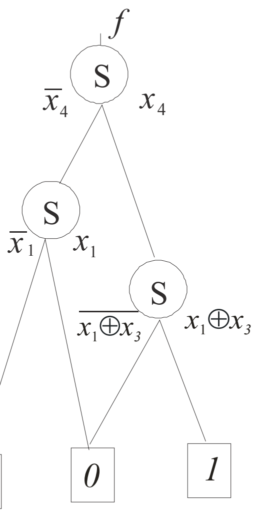


Figure 5. LT-MTBDD of the header of the Remainder

6. CONCLUSIONS

We discussed the problem of constructing planar MTBDDs. A novel technique based on calculating Walsh coefficients has been presented. This technique allows achieving planarity of MTBDD and a relatively small number of nodes in the resulting planar MTBDD. However, there are cases when this technique appears to be ineffective. We investigated two of such cases. In the first of them, the linearization technique is inapplicable since a linear transform leading to simplification of the planar MTBDD does not exist. In the second case, the given function is defined in its PLA form and consequently cannot be linearly transformed due to complexity. Both these cases can be handled by using a newly introduced decomposition technique.

The main results of the paper can be summarized as follows.

1. Linearization of an MTBDD with restriction of the sharing neighboring nodes on the same level provides planarity of the MTBDD.
2. A planar and compact MTBDD can be obtained from a given MTBDD, if the latter is decomposed into two or more component MTBDDs, and they are then separately linearized.
3. A decomposition technique is proposed for obtaining planar MTBDDs from a given function defined by its PLA form. For obtaining compact MTBDDs, the decomposition is completed by the mentioned linearization algorithm.
4. Algebra of D-polynomials is proposed for description of concurrent MTBDDs, and serves the base of the proposed decomposition technique.

REFERENCES

[1] M. Fujita, Y. Kukimoto, R. K. Brayton, "BDD minimization by truth table permutation", *Proc. Int. Symp. on Circuits and Systems, ISCAS'96*, *May 12-15, 1996, Vol. 4, 596-599.*

[2] Rudell, R., "Dynamic variable ordering for ordered binary decision diagrams", *Proc. IEEE Conf. Computer Aided Design*, *Santa Clara, CA, 1993, 42-47.*

[3] W. Gunther, R. Drechsler, "Linear transformations and exact minimization of BDDs", *Proc. 8th Great lake Symp. on VLSI*, *February 19-21, 1998, 325-330.*

[4] R. Murgai, R.K. Brayton, A.L. Sangiovanni-Vincentelli, *Logic Synthesis for Field-Programmable Gate Arrays*, Kluwer Academic Publishers, 1995.

[5] J. T. Butler, J. L. Nowlin, T. Sasao, "Planarity in ROMDD's of Multiple-Valued Symmetric Functions", *26th IEEE International Symposium on Multiple-Valued Logic*, Santiago de Compostela, Spain, May 29-31, 1996, 236-241.

[6] T. Sasao, J. T. Butler, "Planar Multiple-Valued Decision Diagrams", *25th IEEE International Symposium on Multiple-Valued Logic*, Bloomington, Indiana, May 23-25, 1995, 28-35.

[7] J. T. Butler, J. L. Nowlin, T. Sasao, "Planarity in ROMDD's of Multiple-Valued Symmetric Functions", *26th IEEE International Symposium on Multiple-Valued Logic*, Santiago de Compostela, *Spain, May 29-31, 1996, 236-241.*

[8] T. Sasao, J. T. Butler, "Planar Multiple-Valued Decision Diagrams", *25th IEEE International Symposium on Multiple-Valued Logic*, *Bloomington, Indiana, May 23-25, 1995, 28-35.*

[9] T. Sasao and J. T. Butler, "Planar multiple-valued decision diagrams", *Multiple-Valued Logic*, *Vol. 1, No. 1 , 1996, 39-46.*

[10] Tomaszewska, A.M., Yanushkevich, S.N., Shmerko, V.P., "The word-level models for efficient computation of multiple-valued functions, LWL based model", *Prof. 32nd Int. Symp. On Multiple-Valued* Logic, Boston, Massachusetts, USA, May 15-18, 2003, 209-215.

[11] Yanushkevich, S.N., Dziurzanski, P., Shmerko, V.P., "The word level models for efficient computation of multiple-valued functions, LAR based model", *Proc. 32nd Int. Symp. on Multiple-Valued* Logic, Boston, Massachusetts, USA, May 15-18, 2003, 202-208.

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

[13] E.M. Clarke, K.L. Mc Millan, X. Zhao, M. Fujita, "Spectral transforms for extremely large Boolean functions", in *Proc. Reed-Muller Workshop, RM-1993*, *September 16-17, 1993, 86-90.*

[14] M.G. Karpovsky, R. Stankovic and J. Astola, "Construction of Linearly Transformed Planar BDDs by Walsh Coefficients", *Proc. ISCAS, 2004.*