ISSN 1064-2307, Journal of Computer and Systems Sciences International, 2016, Vol. 55, No. 2, pp. 189–210. © Pleiades Publishing, Ltd., 2016.
Original Russian Text © N.A. Avdeev, P.N. Bibilo, 2016, published in Izvestiya Akademii Nauk. Teoriya i Sistemy Upravleniya, 2016, No. 2, pp. 29–50.

COMPUTER METHODS

Experimental Comparison of Decomposition Methods for Systems of Boolean Function

N. A. Avdeev and P. N. Bibilo

United Institute of Computer Science Problems, National Academy of Sciences of Belarus, ul. Surganova 6, Minsk, 220012 Belarus

e-mail: bibilo@newman.bas-net.by

Received June 17, 2014; in final form, September 22, 2015

Abstract—In this paper, we describe the results of the experimental comparison of programs that implement various decomposition methods for disjunctive normal forms of systems of completely defined Boolean functions. The complexity of a system of disjunctive normal forms is expressed in two ways: by the area of a programmable logic array that implements a system of disjunctive normal forms, or by the number of vertices of a binary decision diagram, which represents a system of Boolean functions. The complexity of the functional expansion of a system's functions is determined as the sum of the complexities of the subsystem of the functions included in this expansion. The estimates of the complexity are oriented on the synthesis of combinational circuits based on the programmed logical arrays and the library's logical elements.

DOI: 10.1134/S1064230716010044

INTRODUCTION

The decomposition (functional expansion) of Boolean functions has long been known [1]; usually, it is oriented on the decrease of the number of arguments of functions obtained as a result of the expansion. Due to the decrease of the number of input variables (i.e., the decrease of the dimensions of the problems of optimization and logical synthesis), design procedures for the obtained minorant functions (blocks) of the functional expansion can be used more efficiently. If the initial systems of functions are assigned as disjunctive normal forms (two-level ones AND/OR representations of systems of functions) on the general set of elementary conjunctions, then the problem of decomposing a system of disjunctive normal forms (DNFs) of Boolean functions in terms of parameters (the number of arguments, functions, conjunctions) can be interpreted as the decomposition of one large programmable logic array (PLA) into a network of smaller PLAs. Usually, one PLA is replaced by a two-cascade network of several smaller PLAs with the aim of reducing the area of the chip of an ordered VLSI; however, such a replacement can also be performed to satisfy the electric parameters of a PLA.

In this work, we use the flow of known examples to compare programs for the decomposition of DNF systems of Boolean functions; here, these programs implement various decomposition methods. By the use of a large number of practical examples, it is shown that not only the use of the joint minimization of functions in the DNF class but also their decomposition can ensure a reduction of the area of PLA circuits.

The decomposition of Boolean functions is also used in synthesizing circuits from a library of logical elements. With synthesizing on such a basis, the initial (optimized at the stage of technologically independent optimization) systems are often not minimized DNF systems of functions, but multilevel representations: binary decision diagrams (BDDs) constructed based on Shannon's expansion. Such representations are more efficient in synthesizing circuits from library elements, because the minimization of BDDs often brings about circuits of a lesser complexity. The decomposition can also be useful on such a basis; however, in this case (by synthesizing in terms of the decomposed systems of functions), it is desirable to test both versions of the representation of the obtained subsystems of the functions: in the form of BDDs and in the form of DNFs.

1. FORMS OF ASSIGNING SYSTEMS OF BOOLEAN FUNCTIONS

Boolean functions are binary (0, 1) functions $f(\mathbf{x}) = f(x_1, x_2, \ldots, x_n)$ of binary (Boolean) variables $x_1, x_2, \ldots, x_n$. The vector Boolean function $\mathbf{f}(\mathbf{x})$, $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ signifies an ordered system of Boolean

Table 1. The DNF system of Boolean functions

$T^{\mathbf{x}}$						$B^{\mathbf{f}}$		
x_1	x_2	x_3	x_4	x_5	x_6	f^1	f^2	f^3
1	0	–	1	0	1	0	1	0
1	1	1	–	1	0	0	1	0
0	–	0	–	1	–	0	1	0
1	0	1	–	1	–	0	0	1
1	0	1	1	–	1	0	0	1
1	1	0	1	–	1	0	0	1
0	–	–	1	0	1	1	0	0
–	1	0	–	1	–	1	0	1

functions $\mathbf{f}(\mathbf{x}) = (f^1(\mathbf{x}), ..., f^m(\mathbf{x}))$. Here, the matrix form of representing Boolean functions is widely known in the literature. This form consists of a ternary matrix $T^{\mathbf{x}}$ of assigning elementary conjunctions in the form of ternary vectors and a Boolean matrix $B^{\mathbf{f}}$ of occurrences in the DNF of component functions of a system. The representation of a vector Boolean function by a pair of matrices $\langle T^{\mathbf{x}}, B^{\mathbf{f}}\rangle$ is also called the matrix form of a DNF system of Boolean functions, or simply *a DNF system* [2].

An example of a completely defined vector function $\mathbf{f}(\mathbf{x}) = (f^1(\mathbf{x}), f^2(\mathbf{x}), f^3(\mathbf{x}))$, where the component functions f^1, f^2, f^3 are assigned as the DNF

$$f^1 = \bar{x}_1x_4\bar{x}_5x_6 \vee x_2\bar{x}_3x_5;$$

$$f^2 = x_1\bar{x}_2x_4\bar{x}_5x_6 \vee x_1x_2x_3x_5\bar{x}_6 \vee \bar{x}_1\bar{x}_3x_5;$$

$$f^3 = x_1\bar{x}_2x_3x_5 \vee x_1\bar{x}_2x_3x_4x_6 \vee x_1x_2\bar{x}_3x_4x_6 \vee x_2\bar{x}_3x_5,$$

is presented in Table 1. In this table, the ternary vector (– 1 0 – 1 –) (the last row of the matrix $T^{\mathbf{x}}$) assigns the elementary conjunction $x_2\bar{x}_3x_5$ included in the DNF of the functions f^1 and f^3: the literals x_2 and x_5 are represented by ones in the ternary vector; the literal $\bar{x}_3$ is represented by zero; and the missing literals of the variables x_1, x_4, and x_6 are represented by the "–" symbol.

As the second form of representing systems of Boolean functions in this work, algebraic *multilevel representations based on Shannon's expansion* are used.

Shannon's expansion of a completely defined Boolean function $f(\mathbf{x})$ in terms of a variable x_i is represented by

$$f(\mathbf{x}) = \bar{x}_i f(x_1, ..., x_{i-1}, 0, x_{i+1}, ..., x_n) \vee x_i f(x_1, ..., x_{i-1}, 1, x_{i+1}, ..., x_n). \quad (1.1)$$

The functions in the right side of (1.1) are called expansion coefficients. They are obtained from the function $f(\mathbf{x})$ by the substitution of 0 or 1 (respectively) for the variable x_i. A BDD assigns in the form of a graph a sequence of Shannon's expansions of the initial function and the obtained coefficients of the expansion in terms of a certain permutation of the variables. Minimizing the complexity of the BDD rests on the fact that during the expansion, identical expansion coefficients can appear not only for one but also for several (or even for all) component functions. The *joint BDD-minimizing* in this work refers to the optimization of multilevel representations of systems of Boolean functions corresponding to the reduced ordered BDDs (ROBDDs). The detailed description of ordered BDDs is presented in [3] and of ROBDDs, in [4].

In order to construct a BDD for the vector function $\mathbf{f}(\mathbf{x}) = (f^1(\mathbf{x}), f^2(\mathbf{x}), f^3(\mathbf{x}))$ (see Table 1) in terms of the permutation $\langle x_1, x_2, x_3, x_5, x_4, x_6\rangle$, we perform Shannon's expansion of the component functions $f^1(\mathbf{x})$, $f^2(\mathbf{x})$, and $f^3(\mathbf{x})$ in terms of the variable x_1 and compare the obtained coefficients (the minorant functions). We have

$$s^1 = f^1_{x_1=1} = f^1(1, x_2, x_3, x_4, x_5, x_6) = x_2\bar{x}_3x_5;$$

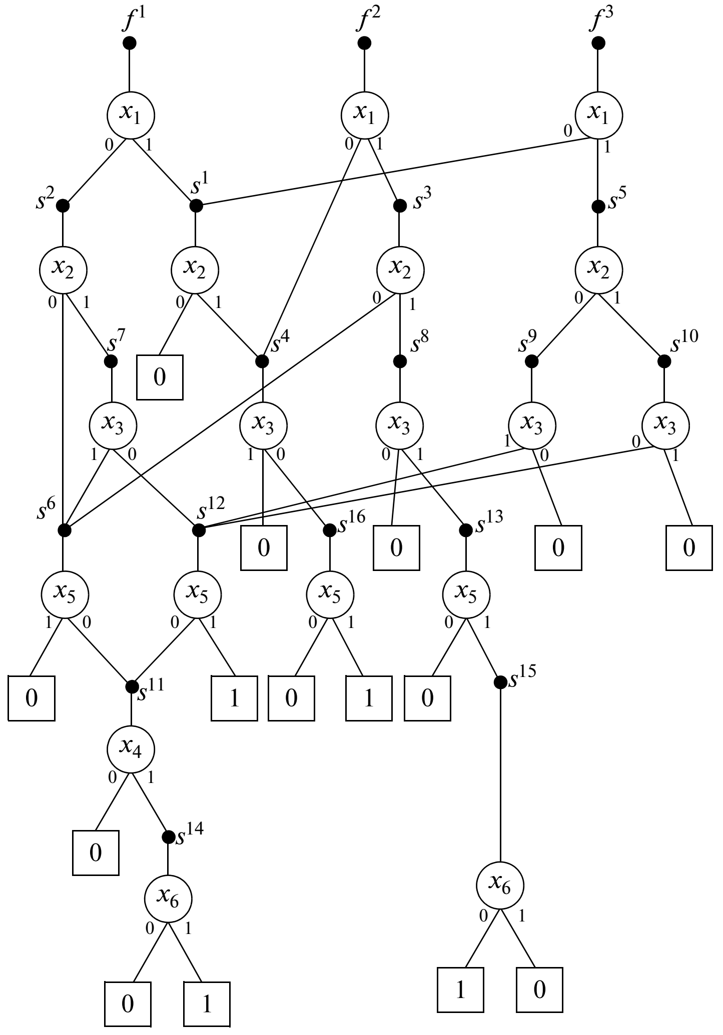

Fig. 1. Binary decision diagram.

$$s^2 = f^1_{x_1=0} = f^1(0, x_2, x_3, x_4, x_5, x_6) = x_4\bar{x}_5x_6 \vee x_2\bar{x}_3x_5;$$

$$s^3 = f^2_{x_1=1} = f^2(1, x_2, x_3, x_4, x_5, x_6) = \bar{x}_2x_4\bar{x}_5x_6 \vee x_2x_3x_5\bar{x}_6;$$

$$s^4 = f^2_{x_1=1} = f^1(0, x_2, x_3, x_4, x_5, x_6) = \bar{x}_3\bar{x}_5;$$

$$s^5 = f^3_{x_1=1} = f^3(1, x_2, x_3, x_4, x_5, x_6) = \bar{x}_2x_3x_5 \vee \bar{x}_2x_3x_4x_6 \vee x_2\bar{x}_3x_4x_6 \vee x_2\bar{x}_3x_5;$$

$$s^1 = f^3_{x_1=0} = f^3(0, x_2, x_3, x_4, x_5, x_6) = x_2\bar{x}_3x_5.$$

The first five different coefficients s^1, ..., s^5 of the multilevel representation are shown on the graph of the BDD (see Fig. 1). We expand the found coefficients in terms of the variable x_2 and obtain the following coefficients (that are not degenerate to constants): s^6, s^7, s^8, s^9, s^{10}. Then, the expansion is performed in terms of the variable x_3, etc. As the result of Shannon's sequential expansions, we obtain the multilevel representation corresponding to the BDD shown in Fig. 1:

$$f^1 = \bar{x}_1s^2 \vee x_1s^1, \quad f^2 = \bar{x}_1s^4 \vee x_1s^3, \quad f^3 = \bar{x}_1s^1 \vee x_1s^5,$$

$$s_1 = x_2s^4, \quad s^2 = \bar{x}_2s^6 \vee x_2s^7, \quad s^3 = \bar{x}_2s^6 \vee x_2s^8,$$

$$s^4 = \bar{x}_3s^{16}, \quad s^5 = \bar{x}_2s^9 \vee x_2s^{10}, \quad s^6 = \bar{x}_5s^{11}, \quad s^7 = \bar{x}_3s^{12} \vee x_3s^6,$$

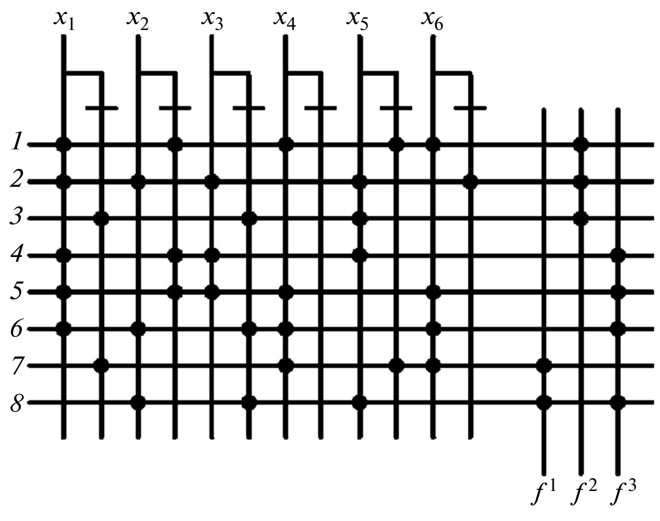

Fig. 2. Programmable logic array.

$$s^8 = x_3 s^{13},\quad s^9 = x_3 s^{12},\quad s^{10} = \bar{x}_3 s^{12},\quad s^{11} = x_4 s^{14},\quad s^{12} = \bar{x}_5 s^{11} \vee x_5,$$

$$s^{13} = x_5 s^{15},\quad s^{14} = x_6,\quad s^{15} = \bar{x}_6,\quad s^{16} = x_5.$$

The complexity $S^{\mathbf{f}}_{BDD}$ of the BDD (Fig. 1), which represents the vector function $\mathbf{f}(\mathbf{x})$, is estimated by using the number of vertices indicated by symbols of functions. In estimating the complexity of BDDs, we do not take into account vertices—variables and leaf vertices [5]. Such an estimate of the complexity coincides with the "size BDD" estimate accepted in the literature. For example, the complexity of the BDD shown in Fig. 1 $S^{\mathbf{f}}_{BDD} = 19$. The main problem in constructing a BDD of the lesser complexity is finding the variables' permutation by which the BDD is constructed.

For each form (a DNF, a BDD) of representation of a system of functions, we shall use a particular estimate of the complexity.

2. ESTIMATES OF THE COMPLEXITY OF REPRESENTATIONS FOR SYSTEMS OF BOOLEAN FUNCTIONS

We estimate the complexity of *the matrix form of a DNF system* of Boolean functions by the area of the PLA circuit, whereas we estimate the complexity of superpositions of DNF systems by the total area of PLAs that implement vector functions of a considered superposition. It is shown in the extensive literature that such an approach is competent in synthesizing circuits from PLAs, because a PLA structure is adequate for a DNF system implemented on it.

A PLA is intended to implement a DNF system of Boolean functions $\mathbf{f}(\mathbf{x})$ assigned on k general elementary conjunctions and consists of two series-connected subcircuits. Signals corresponding to the variables x_i $(i = \overline{1, n})$ of the implemented DNF system are sent to input buses of the PLA. On the matrix first-level subcircuit (the AND matrix), k general elementary conjunctions of the DNF system are implemented; the second-level subcircuit (the OR matrix), which has m output buses, is intended to implement disjunctions of the elementary conjunctions [2].

The area S_{PLA} of a PLA circuit is determined at the stage of the logical design in arbitrary units of area (bits) of the matrix structure by the formula

$$S_{\text{PLM}} = (2nk + mk)\ (\text{bit}), \tag{2.1}$$

where n is the number of inputs of the circuit, m is the number of outputs of the circuit, and k is the number of intermediate buses of the PLA (the number of elementary conjunctions in the implemented DNF system of Boolean functions).

For example, the PLA (Fig. 2) which implements the DNF system (see Table 2) has the following parameters: the number of inputs $n = 6$, the number of outputs $m = 3$, and the number of intermediate buses $k = 8$. The area of this PLA $S_{\text{PLA}} = 120$.

Table 2. The intermediary functions h_1^1 and h_2^1 constructed by the BDD function f^1

Paths from root vertex f^1	DNF	x_1	x_2	x_3	x_5	Code	
						h_1^1	h_2^1
To vertex 0	Q_1^1	1	0	–	–	0	0
		1	1	1	–	0	0
		1	1	0	0	0	0
		0	0	–	1	0	0
		0	1	1	1	0	0
To vertex 1	Q_2^1	1	1	0	1	0	1
		0	1	0	1	0	1
To vertex s^{11}	Q_3^1	0	0	–	0	1	0
		0	1	1	0	1	0
		0	1	0	0	1	0

Networks of PLAs implement superpositions of DNF systems. *The area* of a PLM network will mean the sum of the areas of PLA elements that form the network. Below we describe the examples of PLA networks and determine their areas.

We estimate the complexity of multilevel representations based on Shannon's expansion by the complexity of a BDD. The complexity of a circuit composed of a library's logical elements is expressed at the stage of logical design by the sum of the areas of the elements that form the circuit. Industrial synthesizers of logic circuits produce (after synthesizing a circuit) the value of this parameter, which is usually called "area." Examples of libraries of logical elements are presented in [6].

3. DECOMPOSITION OF SYSTEMS OF BOOLEAN FUNCTIONS

Assume that Y/Z is a partition of the set $X = \{x_1, \ldots, x_n\}$ of variables of the vector Boolean function $\mathbf{f}(\mathbf{x})$ into two disjoint subsets $Y = \{y_1, \ldots, y_r\}$ and $Z = \{z_1, \ldots, z_{n-r}\}$ $(2 \le r \le n-1,\ n \ge 3)$. Suppose $\mathbf{y} = (y_1, \ldots, y_r)$ is a vector obtained by the ordering of variables from the set $Y = \{y_1, \ldots, y_r\}$ and $\mathbf{z} = (z_1, \ldots, z_{n-r})$ is a vector obtained by the ordering of variables from the set $Z = \{z_1, \ldots, z_{n-r}\}$.

The separate decomposition of a system of Boolean functions $\mathbf{f}(\mathbf{x})$ in terms of the assigned partition Y/Z of the set of variables X will refer to the process of constructing the functional expansions (the superpositions)

$$\begin{cases} f^1(\mathbf{x}) = f^1(\mathbf{y},\mathbf{z}) = g^1(\mathbf{h}^1(\mathbf{y}),\mathbf{z}), \\ \ldots \\ f^m(\mathbf{x}) = f^m(\mathbf{y},\mathbf{z}) = g^m(\mathbf{h}^m(\mathbf{y}),\mathbf{z}), \end{cases} \tag{3.1}$$

where $\mathbf{h}^j(\mathbf{y}) = (h_1^j(\mathbf{y}), \ldots, h_{p_j}^j(\mathbf{y}))$ $(j = \overline{1,m})$; in this case, for each component function f^j, the number p_j of intermediary variables (components of the vector function $\mathbf{h}^j(\mathbf{y})$) is minimal.

The joint decomposition of a system of Boolean functions $\mathbf{f}(\mathbf{x})$ in terms of the assigned partition Y/Z of the set of variables X will refer to the process of constructing the functional expansion

$$\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{y},\mathbf{z}) = \mathbf{g}(\mathbf{h}(\mathbf{y}),\mathbf{z}), \tag{3.2}$$

where $\mathbf{h}(\mathbf{y}) = (h_1(\mathbf{y}), \ldots, h_p(\mathbf{y}))$ is a vector function having the minimum number of components p.

Decomposition (3.2) of the vector function is called joint [5], because all the component functions f^j of the decomposed vector Boolean function $\mathbf{f}(\mathbf{x})$ have common (jointly used) intermediary minorant functions $h_1(\mathbf{y}), \ldots, h_p(\mathbf{y})$.

The considered forms of decomposition are called the *disjoint* decomposition in the literature, because subsets Y and Z are disjoint $(Y \cap Z = \varnothing)$, as opposed to the case of the *overlapping* decomposition, corresponding to the case where $Y \cap Z \neq \varnothing$. The conditions $p_j < r$ and $n - r > \log_2 p_j$ separate the class of *nontrivial* expansions (3.1) of the function $f^j(\mathbf{x})$, while the condition $p_j < r$ separates the class of *useful* expansions, i.e., expansions that allows reducing the number r of the recoded arguments p_j by using the new variables $h_1^j, \ldots, h_{p_j}^j$.

The use of decomposition in synthesizing logic circuits rests initially on searching for the simple partition of one Boolean function f into two minorant functions g and h ($p = 1$) with the number of arguments that is smaller than n [1]; this significantly simplifies optimizing and subsequent synthesizing. However, soon it appears that the cases of the partition of functions into two minorant functions are rather rare in occurrence; here, for randomly chosen functions, by increasing the number n of arguments, the probability of the partition of functions tends to zero. Therefore, most attention was concentrated on the decomposition in terms of the two-block partition of variables for obtaining useful expansions. The first works devoted to this subject matter rest on assigning functions by truth tables [7, 8]. Note that the decomposition of incompletely defined (partial) functions [9] and systems of functions [10] has also been studied; for such functions, minimizing the number of intermediary variables is reduced to the coloring problem of a relationship graph of the incompatibility of the coefficients for Shannon's expansion in terms of the subset of arguments Y. Methods for constructing various types of a joint decomposition of the functions' systems are also considered in the literature [10].

Significant impetus to the subject matter of the decomposition is given by PLAs (cascade matrix circuits) whose structure and operation are adequate for DNF systems of functions implemented on these circuits [2]. Implementing a DNF system by a PLA network of a limited dimension gives rise to the orthogonalization and generalized orthogonalization methods [10], the method for identity mapping in the space of intermediary variables [2], the method for covering of ternary matrices [11, 12], etc. The known decomposition methods of functions' systems differ in the used mathematical apparatus. The decomposition of a vector function with the minimum number of intermediary functions is considered in the literature for the case of assigning vector functions by matrix forms (DNF systems) [10]; here, operations on matrix forms of DNFs are used. The decomposition of BDD representations of vector functions is described in [5]. Other formal apparatuses are also used, namely, spectral representations [13], logic equations [14], matrix logic equations [2], and finite predicates [15]. Reviews of the results obtained in this field are presented in [11, 14, 16]. The iteratively applied decomposition of form (3.1) and (3.2) can be used not only for optimizing but also for the technological mapping performed in obtaining structures of field programmable gate arrays (FPDAs) [17].

Wide acceptance of BDDs instead of DNFs in designing logic circuits allows taking a different look at many optimization problems, namely, synthesizing, construction of tests, and verification; BDDs represent efficient means for solving many tasks, including the decomposition of functions. The main problem in the decomposition is searching for a partition of variables by which the decomposition is performed. It is shown in [5] that the use of the BDD apparatus in the decomposition of a system of functions simplifies solving the problem of searching for the separation of variables. In the present paper, decomposition is concerned with the large-block partition of DNF systems of completely defined functions, as opposed to the multilevel synthesis line [18] associated with the small-block representation of elements of logic networks.

However, in spite of the quite large number of the known decomposition methods, there do not exist many works devoted to the experimental comparison of programs that implement the proposed methods and algorithms. The dimensions of logical-synthesis problems in various technological bases have increased; therefore, the problematic of developing efficient methods for the decomposition and experimental comparison of the corresponding programs of decomposition of functions' systems represented in various forms (DNFs, BDDs, parenthesis forms) is topical. In the context of increasing the dimensions of synthesis problems and the emergence of new methods and algorithms for multicriteria optimization of functional descriptions of combinational circuits, it is necessary to develop means for the intellectualization of computer-aided design systems [19, 20]. The means for the intellectualization can rest on expert systems and heuristic optimization-algorithms [21], as well as on estimates of the complexity of combinational circuits in various bases (the Boolean basis, the Zhegalkin-polynomials' basis, the basis of multiplexers with various numbers of control inputs, etc.) [22, 23]. In this case, decomposition can be useful for partitioning circuits of a large dimension into functional blocks, whereas efficiently searching for the basis and estimating the complexity of implementation can be efficient for the functional blocks of a relatively small dimension.

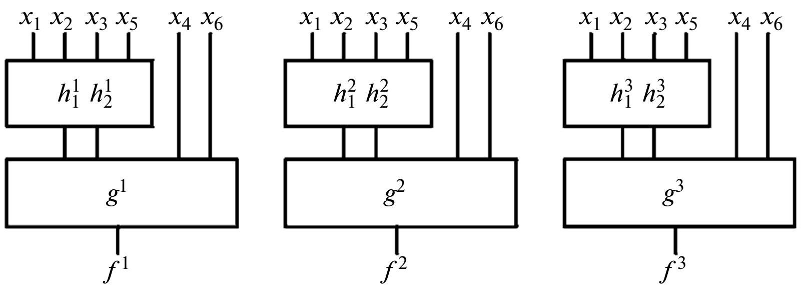
x_1 x_2 x_3 x_5 x_4 x_6
h_1^1 h_2^1
g^1
f^1
x_1 x_2 x_3 x_5 x_4 x_6
h_1^2 h_2^2
g^2
f^2
x_1 x_2 x_3 x_5 x_4 x_6
h_1^3 h_2^3
g^3
f^3

Fig. 3. Separate decomposition.

4. EXAMPLES OF SEPARATE AND JOINT DECOMPOSITION

Example of Separate Decomposition

The separate decompositions of functions $f^j(\mathbf{x}) = f^j(\mathbf{y},\mathbf{z})$ $(j = \overline{1,m})$ are each performed by Shannon's shortest expansion constructed in terms of the subset of arguments Y. Shannon's classic expansion of the function $f^j(\mathbf{y},\mathbf{z})$ in terms of the subset $Y = \{y_1, \ldots, y_r\}$ has the form

$$f^j(\mathbf{y},\mathbf{z}) = \bar{y}_1\bar{y}_2\ldots\bar{y}_r f^j_{\mathbf{y}_0^*}(\mathbf{z}) \vee \bar{y}_1\bar{y}_2\ldots y_r f^j_{\mathbf{y}_1^*}(\mathbf{z}) \vee \ldots \vee y_1 y_2\ldots y_r f^j_{\mathbf{y}^*_{2^r-1}}(\mathbf{z}). \tag{4.1}$$

Here, the expansion coefficient $f^j_{\mathbf{y}_i^*}(\mathbf{z})$ is a completely defined Boolean function representing the result of the substitution of values of vector $\mathbf{y}$ into the function $f^j(\mathbf{y},\mathbf{z})$ for the variables of this vector. The values in question are various binary vectors $\mathbf{y}_i^*$ $(i = \overline{0,\ 2^r - 1})$. We group the terms of expansion (4.1) with identical coefficients $f_q^j(\mathbf{z})$ together in one class (set) and obtain Shannon's shortest expansion

$$f^j(\mathbf{y},\mathbf{z}) = Q_1^j f_1^j(\mathbf{z}) \vee Q_2^j f_2^j(\mathbf{z}) \vee \ldots \vee Q_{k_j}^j f_{k_j}^j(\mathbf{z}). \tag{4.2}$$

The perfect DNFs Q_q^j $(q = \overline{1,k_j})$ are pairwise orthogonal; they can be minimized and represented as DNFs. We recall that two DNFs Q_a^j and Q_b^j $(a \neq b;\ a,b = \overline{1,k_j})$ are orthogonal if and only if $Q_a^j Q_b^j = 0$. It is easy to find the DNFs Q_q^j by the BDD [5]; then, they are encoded with a minimum-length Boolean code. As a result, intermediary functions are obtained. The output functions $g^j(\mathbf{h}^j(\mathbf{y}),\mathbf{z})$ are constructed by replacing Q_q^j with codes that are values of the vector functions $\mathbf{h}^j(\mathbf{y})$ on Q_q^j.

Using the separate decomposition of the vector function $\mathbf{f}(\mathbf{x}) = (f^1(\mathbf{x}), f^2(\mathbf{x}), f^3(\mathbf{x}))$ assigned as the BDD (see Fig. 1) in terms of the partition of the variables $Y = \{x_1, x_2, x_3, x_5\}$ and $Z = \{x_4, x_6\}$, we can obtain six intermediary functions at the first layer (the input block) of the circuit (see Fig. 3). In order to construct Shannon's shortest expansion in terms of the subset of arguments $Y = \{x_1, x_2, x_3, x_5\}$, we use the BDD (Fig. 1). In the example, Shannon's shortest expansions of component functions have the form

$$f^1(\mathbf{y},\mathbf{z}) = (Q_1^1 \ \&\ 0) \vee (Q_2^1 \ \&\ 1) \vee (Q_3^1 \ \&\ s^{11}(x_4,x_6)), \tag{4.3}$$

$$f^2(\mathbf{y},\mathbf{z}) = (Q_1^2 \ \&\ 0) \vee (Q_2^2 \ \&\ 1) \vee (Q_3^2 \ \&\ s^{11}(x_4,x_6)) \vee (Q_4^2 \ \&\ s^{15}(x_4,x_6)), \tag{4.4}$$

$$f^3(\mathbf{y},\mathbf{z}) = (Q_1^3 \ \&\ 0) \vee (Q_2^3 \ \&\ 1) \vee (Q_3^3 \ \&\ s^{11}(x_4,x_6)), \tag{4.5}$$

where $s^{11}(x_4,x_6) = x_4x_6$ and $s^{15}(x_4,x_6) = \bar{x}_6$. The intermediary functions are assigned in Tables 2–4. We insert the codes (h_1^j, h_2^j) of the DNFs Q_q^j into formulas (4.3)–(4.5), perform obvious logical rearrangements, and obtain the DNF of the output functions:

$$f^1(\mathbf{y},\mathbf{z}) = g^1 = \bar{h}_1^1 h_1^2 \vee h_1^1 \bar{h}_1^2 x_4 x_6,$$

$$f^2(\mathbf{y},\mathbf{z}) = g^2 = \bar{h}_1^2 h_2^2 \vee h^2 \bar{h}_2^2 x_4 x_6 \vee h_1^2 h_2^2 \bar{x}_6,$$

Table 3. The intermediary functions h_1^2 and h_2^2 constructed by the BDD function f^2

Paths from root vertex f^2	DNF	x_1	x_2	x_3	x_5	Code	
						h_1^2	h_2^2
To vertex 0	Q_1^2	0	–	1	–	0	0
		0	–	0	0	0	0
		1	1	0	–	0	0
		1	1	1	0	0	0
		1	0	–	1	0	0
To vertex 1	Q_2^2	0	–	0	1	0	1
To vertex s^{11}	Q_3^2	1	0	–	0	1	0
To vertex s^{15}	Q_4^2	1	1	1	1	1	1

Table 4. The intermediary functions h_1^3 and h_2^3 constructed by the BDD function f^3

Paths from root vertex f^3	DNF	x_1	x_2	x_3	x_5	Code	
						h_1^3	h_2^3
To vertex 0	Q_1^3	0	0	–	–	0	0
		1	0	0	–	0	0
		1	1	1	–	0	0
		0	1	1	–	0	0
		0	1	0	0	0	0
To vertex 1	Q_2^3	0	1	0	1	0	1
		1	0	1	1	0	1
		1	1	0	1	0	1
To vertex s^{11}	Q_3^3	1	0	1	0	1	0
		1	1	0	0	1	0

$$f^3(\mathbf{y},\mathbf{z}) = g^3 = \bar{h}_1^3 h_2^3 \vee h_1^3 \bar{h}_2^3 x_4 x_6.$$

In synthesizing circuits on PLAs, the input functions h (the output functions g) are jointly minimized in the DNF class in order to decrease the number of intermediate buses of a PLA. The input functions minimized in the DNF class are assigned in Table 5 and the minimized output functions depending on the variables x_4, x_6, as well as on intermediary variables are assigned in Table 6. It is easy to see that with the separate decomposition each of the functions $g^1(h_1^1, h_2^1, x_4, x_6) = f^1$, $g^2(h_1^2, h_2^2, x_4, x_6) = f^2$, and $g^3(h_1^3, h_2^3, x_4, x_6) = f^3$ depends on its subset of intermediary variables.

If we implement all functions of the first (and the second) cascade on one PLA, we obtain a network of two PLAs (see Fig. 4). Let us determine the areas S_{PLA1} and S_{PLA2} for this network: $S_{\text{PLA1}} = 2 \times 4 \times 8 + 6 \times 8 = 112$, $S_{\text{PLA2}} = 2 \times 8 \times 7 + 3 \times 7 = 123$, and $S_{\text{PLA1}} + S_{\text{PLA2}} = 112 + 123 = 235$.

Hence, in this example, the total area of two PLAs of the resulting network is larger than the area of the initial PLA; the reduction of the area in connection with the separate decomposition is rare, because the intermediary functions are constructed independently for each component function. The separate decomposition in the considered example allows decreasing the number of arguments of the implemented functions; however, with the decomposition, the total complexity of the PLA implementation of two blocks—input and output—increased. Nevertheless, for other examples (of a practical dimension) that represent DNF systems, separate decomposition can not only reduce the number of input variables but

Table 5. Minimized DNF system of functions of the input block in the case of the separate decomposition

x_1	x_2	x_3	x_5	h_1^1	h_2^1	h_1^2	h_2^2	h_1^3	h_2^3
1	0	1	0	0	0	0	0	1	0
1	1	0	0	0	0	0	0	1	0
1	0	1	1	0	0	0	0	0	1
0	–	0	1	0	0	0	1	0	0
1	1	1	1	0	0	1	1	0	0
–	1	0	1	0	1	0	0	0	1
1	0	–	0	0	0	1	0	0	0
0	–	–	0	1	0	0	0	0	0

Table 6. Minimized DNF system of functions of the output block in the case of the separate decomposition

h_1^1	h_2^1	h_1^2	h_2^2	h_1^3	h_2^3	x_4	x_6	f^1	f^2	f^3
1	0	–	–	–	–	1	1	1	0	0
0	1	–	–	–	–	–	–	1	0	0
–	–	1	0	–	–	1	1	0	1	0
–	–	–	1	–	–	–	0	0	1	0
–	–	0	1	–	–	–	–	0	1	0
–	–	–	–	1	0	1	1	0	0	1
–	–	–	–	0	1	–	–	0	0	1

also decrease the total area of the two PLAs. Note that further in the experiments with the separate decomposition, all functions of the first cascade and output functions are implemented on one input PLA and one output PLA, respectively.

Example of Joint Decomposition

Using the joint decomposition of the vector function $\mathbf{f}(\mathbf{x}) = (f^1(\mathbf{x}), f^2(\mathbf{x}), f^3(\mathbf{x}))$ (see Table 1) in terms of the partition of the variables $Y = \{x_4, x_5, x_6\}$ and $Z = \{x_1, x_2, x_3\}$, we can obtain two intermediary functions h_1 and h_2 at the first layer (the input block) of the circuit (Fig. 5).

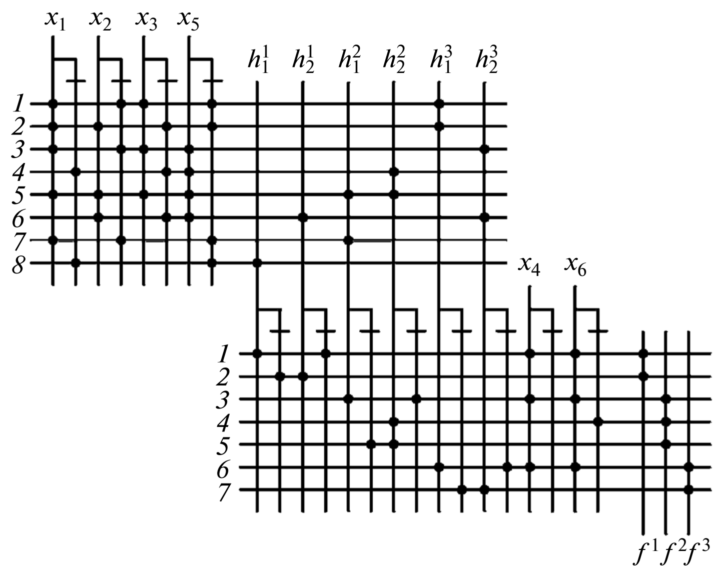

Fig. 4. Implementation of separate decomposition by a PLA network.

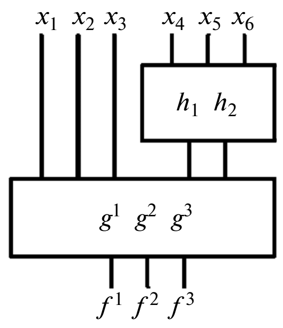

Fig. 5. Joint decomposition.

We briefly recall the main stages of the joint decomposition of a DNF system of Boolean functions [5]. In order to perform the joint decomposition, we construct (in one form or another) Shannon's shortest expansion of the vector function:

$$\mathbf{f}(\mathbf{y},\mathbf{z}) = Q_1\mathbf{f}_1(\mathbf{z}) \vee Q_2\mathbf{f}_2(\mathbf{z}) \vee \ldots \vee Q_k\mathbf{f}_k(\mathbf{z}). \tag{4.6}$$

In order to construct this expansion, *the orthogonalization* of the matrix DNF form in terms of the partition Y/Z is performed: elementary conjunctions (intervals, ternary vectors) corresponding to variables of the subset Y are overlapped, the domains of the overlapping are represented as the DNFs Π_i^y and each elementary conjunction is represented as the disjunction of some of them. For the considered example, the elementary conjunctions k_i^y are presented in Table 7 and the DNFs Π_i^y, in Table 8. In Table 9, the units in the corresponding row specify DNFs Π_i^y whose disjunction produces the elementary conjunction k_i^y. The analogous procedure is also performed for the subset of variables Z: the elementary conjunctions k_i^z are presented in Table 10; the DNFs Π_i^z, in Table 11; and the disjunctive expansion of the elementary con-

Table 7. Conjunctions on the variables of the set Y

x_4	x_5	x_6	k_i^y
1	0	1	k_1^y
–	1	0	k_2^y
–	1	–	k_3^y
1	–	1	k_4^y

Table 8. Representation of products of conjunctions as the Π_i^y DNF system

Π_i^y	DNF system		
	x_4	x_5	x_6
Π_1^y	0	0	0
	0	0	1
	1	0	0
Π_2^y	0	1	1
Π_3^y	1	0	1
Π_4^y	0	1	0
	1	1	0
Π_5^y	1	1	1

Table 9. Disjunctive expansion of elementary conjunctions k_i^y

Π_1^y	Π_2^y	Π_3^y	Π_4^y	Π_5^y	k_i^y
0	0	1	0	0	k_1^y
0	0	0	1	0	k_2^y
0	1	0	1	1	k_3^y
0	0	1	0	1	k_4^y

Table 10. Conjunctions on the variables of the set Z

x_1	x_2	x_3	k_i^z
1	0	–	k_1^z
1	1	1	k_2^z
0	–	0	k_3^z
1	0	1	k_4^z
1	1	0	k_5^z
0	–	–	k_6^z
–	1	0	k_7^z

Table 11. Representation of products of conjunctions as the Π_i^z DNF system

Π_i^z	DNF system		
	x_1	x_2	x_3
Π_1^z	1	0	0
Π_2^z	1	1	1
Π_3^z	0	0	1
	0	1	1
Π_4^z	0	0	0
Π_5^z	1	1	0
Π_6^z	0	1	0
Π_7^z	1	0	1

junctions k_i^z into the DNFs Π_i^z, in Table 12. As a result, the matrix form (see Table 1) is transformed into the orthogonalized matrix form (Table 13) of assigning a system of functions on the DNFs Π_i^y and Π_i^z.

Suppose $D^y = k_1^y \vee k_2^y \vee k_3^y \vee k_4^y = \Pi_2^y \vee \Pi_3^y \vee \Pi_4^y \vee \Pi_5^y$ and $D^z = k_1^z \vee k_2^z \vee k_3^z \vee k_4^z \vee k_5^z \vee k_6^z \vee k_7^z = \Pi_2^z \vee \Pi_3^z \vee \Pi_4^z \vee \Pi_5^z \vee \Pi_6^z \vee \Pi_7^z$; then, $\Pi_1^y = \bar{D}^y$ and $\Pi_1^z = \bar{D}^z$.

If the system of functions $\mathbf{f}(\mathbf{x})$ is assigned over the entire Boolean space of variables of the vector $\mathbf{x}$, then by the orthogonalized form, one can easily construct coefficients of Shannon's expansion of this system in the form of Table 14 called a *compact* table in [11, 12].

In the considered example, Shannon's shortest expansion in terms of the subset of variables $Y = \{x_4, x_5, x_6\}$ has the form

$$\mathbf{f}(\mathbf{y},\mathbf{z}) = Q_1\mathbf{f}_1(\mathbf{z}) \vee Q_2\mathbf{f}_2(\mathbf{z}) \vee Q_3\mathbf{f}_3(\mathbf{z}) \vee Q_4\mathbf{f}_4(\mathbf{z}), \quad (4.7)$$

Table 12. Disjunctive expansion of elementary conjunctions k_i^z

Π_1^z	Π_2^z	Π_3^z	Π_4^z	Π_5^z	Π_6^z	Π_7^z	k_i^z
1	0	0	0	0	0	1	k_1^z
0	1	0	0	0	0	0	k_2^z
0	0	0	1	0	1	0	k_3^z
0	0	0	0	0	0	1	k_4^z
0	0	0	0	1	0	0	k_5^z
0	0	1	1	0	1	0	k_6^z
0	0	0	0	1	1	0	k_7^z

Table 13. Orthogonalized form of a DNF system

Elementary conjunction	$\Pi_i^y\Pi_i^z$	f^1	f^2	f^3
$k_1^y k_1^z$	$\Pi_3^y\Pi_1^z$	0	1	0
	$\Pi_3^y\Pi_7^z$			
$k_2^y k_2^z$	$\Pi_4^y\Pi_2^z$	0	1	0
$k_3^y k_3^z$	$\Pi_2^y\Pi_4^z$	0	1	0
	$\Pi_2^y\Pi_6^z$			
	$\Pi_4^y\Pi_4^z$			
	$\Pi_4^y\Pi_6^z$			
	$\Pi_5^y\Pi_4^z$			
	$\Pi_5^y\Pi_6^z$			
$k_3^y k_4^z$	$\Pi_2^y\Pi_7^z$	0	0	1
	$\Pi_4^y\Pi_7^z$			
	$\Pi_5^y\Pi_7^z$			
$k_4^y k_4^z$	$\Pi_3^y\Pi_7^z$	0	0	1
	$\Pi_5^y\Pi_7^z$			
$k_4^y k_5^z$	$\Pi_3^y\Pi_5^z$	0	0	1
	$\Pi_5^y\Pi_7^z$			
$k_1^y k_6^z$	$\Pi_3^y\Pi_3^z$	1	0	0
	$\Pi_3^y\Pi_4^z$			
	$\Pi_3^y\Pi_6^z$			
$k_3^y k_7^z$	$\Pi_2^y\Pi_5^z$	1	0	1
	$\Pi_2^y\Pi_6^z$			
	$\Pi_4^y\Pi_5^z$			
	$\Pi_4^y\Pi_6^z$			
	$\Pi_5^y\Pi_5^z$			
	$\Pi_5^y\Pi_6^z$			

because the coefficients $\mathbf{f}_{\Pi_2^y} = \mathbf{f}_{\Pi_5^y}$. Therefore, $Q_1 = \Pi_1^y$, $Q_2 = \Pi_2^y \vee \Pi_5^y$, $Q_3 = \Pi_3^y$, and $Q_4 = \Pi_4^y$. Intermediary functions in the joint decomposition are constructed by coding the factors Q_i by using the minimum by the Boolean length or ternary code; here, the codes of Q_i must be pairwise orthogonal [14]. In the con-

Table 14. Coefficients of Shannon's expansion

Π_i^z	$\mathbf{f}_{\Pi_1^y} = \mathbf{f}_1(\mathbf{z})$	$\mathbf{f}_{\Pi_2^y} = \mathbf{f}_2(\mathbf{z})$	$\mathbf{f}_{\Pi_3^y} = \mathbf{f}_3(\mathbf{z})$	$\mathbf{f}_{\Pi_4^y} = \mathbf{f}_4(\mathbf{z})$	$\mathbf{f}_{\Pi_5^y} = \mathbf{f}_2(\mathbf{z})$
Π_1^z	000	000	010	000	000
Π_2^z	000	000	000	010	000
Π_3^z	000	000	100	000	000
Π_4^z	000	010	100	010	010
Π_5^z	000	101	001	101	101
Π_6^z	000	111	100	111	111
Π_7^z	000	001	011	001	001

Table 15. Intermediary functions in the case of a joint decomposition

Π_i^y	Code	
	h_1	h_2
Q_1	1	0
Q_2	0	0
Q_3	1	1
Q_4	0	1

sidered example, we have four factors Q_i $(i = \overline{1,4})$; hence, the code's length (h_1, h_2) is equal to two. Coding the factors Q_i by using the pairwise orthogonal Boolean codes is demonstrated in Table 15.

Different decomposition methods of DNF systems vary in the ways of representing the orthogonalized form and the ways of coding the factors Q_i, or the ways of coding the conjunctions k_i^y and k_i^z of the initial interval form. The most frequently used type of coding is trivial coding analogous to the type presented in the considered example (see Table 15).

In the identity mapping method [2], the columns of Table 9 are recoded by using the special disjunctive code; here, the intermediary functions h are assigned in the form of DNF systems on the conjunctions k_i^y. Such coding increases the number of conjunctions for the output functions g. A specially agreed coding used to reduce the number of conjunctions in the DNF of the output functions is proposed in [15].

The heuristics for the choice of coding aimed to reduce the number of conjunctions on the DNF of the output functions g are used in the method of coding the DNFs Q_i by placing codes at the vertices of the hypercube: if the compact table has few distinctions in its columns of values for coefficients, then it is appropriate to assign identical codes to these coefficients [24].

The output functions g^1, g^2, and g^3 are constructed by replacing the DNF Q_i with the codes from Table 15:

$$f^1 = g^1 = Q_1 f_1^1(\mathbf{z}) \vee Q_2 f_2^1(\mathbf{z}) \vee Q_3 f_3^1(\mathbf{z}) \vee Q_4 f_4^1(\mathbf{z})$$

$$= h_1\bar{h}_2\&0 \vee \bar{h}_1\bar{h}_2(\Pi_5^z \vee \Pi_6^z) \vee h_1h_2(\Pi_3^z \vee \Pi_4^z \vee \Pi_6^z) \vee \bar{h}_1h_2(\Pi_5^z \vee \Pi_6^z)$$

$$= \bar{h}_1\bar{h}_2(x_1x_2\bar{x}_3 \vee \bar{x}_1x_2\bar{x}_3) \vee h_1h_2(\bar{x}_1\bar{x}_2x_3 \vee \bar{x}_1x_2x_3 \vee \bar{x}_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1x_2\bar{x}_3) \vee \bar{h}_1h_2(x_1x_2\bar{x}_3 \vee \bar{x}_1x_2\bar{x}_3)$$

$$= \bar{h}_1\bar{h}_2x_1x_2\bar{x}_3 \vee \bar{h}_1\bar{h}_2\bar{x}_1x_2\bar{x}_3 \vee h_1h_2\bar{x}_1\bar{x}_2x_3 \vee h_1h_2\bar{x}_1x_2x_3 \vee h_1h_2\bar{x}_1\bar{x}_2\bar{x}_3 \vee h_1h_2\bar{x}_1x_2\bar{x}_3 \vee \bar{h}_1h_2x_1x_2\bar{x}_3 \vee \bar{h}_1h_2\bar{x}_1x_2\bar{x}_3.$$

$$f^2 = g^2 = Q_1 f_1^2(\mathbf{z}) \vee Q_2 f_2^2(\mathbf{z}) \vee Q_3 f_3^2(\mathbf{z}) \vee Q_4 f_4^2(\mathbf{z})$$

$$= h_1\bar{h}_2\&0 \vee \bar{h}_1\bar{h}_2(\Pi_4^z \vee \Pi_6^z) \vee h_1h_2(\Pi_1^z \vee \Pi_7^z) \vee \bar{h}_1h_2(\Pi_2^z \vee \Pi_4^z \vee \Pi_6^z)$$

$$= \bar{h}_1\bar{h}_2(\bar{x}_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1x_2\bar{x}_3) \vee h_1h_2(x_1\bar{x}_2\bar{x}_3 \vee x_1\bar{x}_2x_3) \vee \bar{h}_1h_2(x_1x_2x_3 \vee \bar{x}_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1x_2\bar{x}_3)$$

$$= \bar{h}_1\bar{h}_2\bar{x}_1\bar{x}_2\bar{x}_3 \vee \bar{h}_1\bar{h}_2\bar{x}_1x_2\bar{x}_3 \vee h_1h_2x_1\bar{x}_2\bar{x}_3 \vee h_1h_2x_1\bar{x}_2x_3 \vee \bar{h}_1h_2x_1x_2x_3 \vee \bar{h}_1h_2\bar{x}_1\bar{x}_2\bar{x}_3 \vee \bar{h}_1h_2\bar{x}_1x_2\bar{x}_3.$$

Table 16. Not minimized functions of the output block in the case of a joint decomposition

h_1	h_2	x_1	x_2	x_3	g^1	g^2	g^3
0	0	0	0	0	0	1	0
0	0	1	1	0	1	0	1
0	0	1	0	1	0	0	1
0	0	0	1	0	1	1	1
1	1	1	0	0	0	1	0
1	1	0	0	1	1	0	0
1	1	0	1	1	1	0	0
1	1	0	0	0	1	0	0
1	1	1	1	0	0	0	1
1	1	1	0	1	0	1	1
1	1	0	1	0	1	0	0
0	1	1	1	1	0	1	0
0	1	0	0	0	0	1	0
0	1	1	1	0	1	0	1
0	1	1	0	1	0	0	1
0	1	0	1	0	1	1	1

$$\begin{aligned} f^3 = g^3 &= Q_1 f_1^3(\mathbf{z}) \vee Q_2 f_2^3(\mathbf{z}) \vee Q_3 f_3^3(\mathbf{z}) \vee Q_4 f_4^3(\mathbf{z}) \\ &= h_1\bar{h}_2\&0 \vee \bar{h}_1\bar{h}_2(\Pi_5^z \vee \Pi_6^z \vee \Pi_7^z) \vee h_1h_2(\Pi_5^z \vee \Pi_7^z) \vee \bar{h}_1h_2(\Pi_5^z \vee \Pi_6^z \vee \Pi_7^z) \\ &= \bar{h}_1\bar{h}_2(x_1x_2\bar{x}_3 \vee \bar{x}_1x_2\bar{x}_3 \vee x_1\bar{x}_2x_3) \vee h_1h_2(x_1\bar{x}_2\bar{x}_3 \vee x_1\bar{x}_2x_3) \vee \bar{h}_1h_2(x_1x_2\bar{x}_3 \vee \bar{x}_1x_2\bar{x}_3 \vee x_1\bar{x}_2x_3) \\ &= \bar{h}_1\bar{h}_2x_1x_2\bar{x}_3 \vee \bar{h}_1\bar{h}_2\bar{x}_1x_2\bar{x}_3 \vee \bar{h}_1\bar{h}_2x_1\bar{x}_2x_3 \vee h_1h_2x_1x_2\bar{x}_3 \vee h_1h_2x_1\bar{x}_2x_3 \vee \bar{h}_1h_2x_1x_2\bar{x}_3 \vee \bar{h}_1h_2\bar{x}_1x_2\bar{x}_3 \vee \bar{h}_1h_2x_1\bar{x}_2x_3. \end{aligned}$$

The matrix form of the vector function $\mathbf{g} = (g^1, g^2, g^3)$ is presented in Table 16. We perform the joint minimization of the system of functions (Table 16) and obtain the output block (see Fig. 6) implemented on the PLA having eight intermediate buses. With the joint decomposition, we have two intermediary vari-

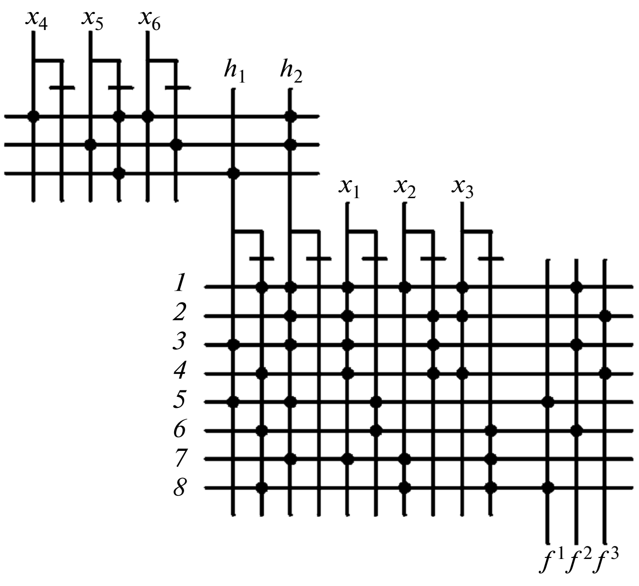

Fig. 6. Implementation of joint decomposition by a PLA network.

ables such that each of them is used as an intermediary variable for each of the component functions f^1, f^2, and f^3 of the decomposed vector function. Note that $h_1 = \bar{x}_5$.

Let us determine the area of the PLA network (see Fig. 6) implementing the joint decomposition: $S_{\text{PLA1}} = 2 \times 3 \times 3 + 2 \times 3 = 24$, $S_{\text{PLA2}} = 2 \times 7 \times 8 + 3 \times 8 = 136$, and $S_{\text{PLA1}} + S_{\text{PLA2}} = 24 + 136 = 160$. This area is considerably smaller in relation to the case of the separate decomposition, but at the same time it is larger than the area of the initial PLA. We often have a different situation for practical examples: the area of a PLA network is smaller than the area of a decomposed PLA, i.e., of a decomposed DNF system.

This illustrative example demonstrates various forms of decomposition in terms of various partitions of the set of arguments. Decomposition is not necessarily oriented only to reducing an area: often decomposition is associated with satisfying the limitations on the number of input and output variables and on the number of intermediate buses. Various methods for such decompositions are considered in [2].

5. PROGRAMS OF MINIMIZATION AND DECOMPOSITION OF SYSTEMS OF BOOLEAN FUNCTIONS

5.1. The ESPRESSO IIC program for *joint DNF-minimizing* systems of the Boolean functions $\mathbf{f}(\mathbf{x})$ in the DNF class is a widely known minimization program; monograph [25] is devoted to it.

The input and resulting data of this program are text matrix representations of the initial and minimized DNF systems of Boolean functions, respectively.

5.2. The SEPT_BDD program for *the separate decomposition* of DNF systems rests on the apparatus of the BDD representation of systems of Boolean functions and implements the method described in [5]. The initial data are DNF systems and the results are superpositions of functions of form (3.1); here, functions of the input and output blocks are represented as a DNF system and as a BDD, respectively.

For the programs of the joint decomposition that are considered below, the initial data are DNF systems and the results are superpositions of form (3.2); in this case, each of the vector functions $\mathbf{h}(\mathbf{y})$ (the input block) and $\mathbf{g}(\mathbf{h}(\mathbf{y}), \mathbf{z})$ (the output block) is represented as a DNF system.

5.3. The DECU_BDD program for *the joint decomposition* of DNF systems rests on the apparatus of the BDD representation of systems of Boolean functions and is described in [5].

Algorithms for choosing the variables' partition that are implemented in the SEPT_BDD and DECU_BDD programs are considered in [5].

5.4. The DEC_FT program for *the joint decomposition* of matrix representations of DNF systems of Boolean functions implements the decomposition method based on compact tables and is described in [12].

Suppose $|Y|$ is the power of set Y and $\lceil a \rceil$ is the least integer that is bigger than or equal to a. The DEC_FT program finds in the experiments the best partition Y/Z such that $|Y| = \lceil n/2 \rceil = l$ whereby the least possible number of intermediary variables p in the joint decomposition is attained. Here, all partitions C_n^l are not searched through: with the use of heuristics, one partition Y/Z is chosen, this partition is taken to be the best, and the decomposition is performed by it.

5.5. The DEC_HIE program for *the joint decomposition* of matrix representations of DNF systems of Boolean functions is described in [26]. The program receives intermediary variables by the method for placing codes (values of the intermediary variables) at the vertices of the hypercube [24].

The DEC_HIE program searches through all $2^n - n - 2$ nontrivial two-block partitions of the set of variables and finds the best version of decomposition (3.2); the version is estimated by the formula $O(2^r/r + 2^{n-r+p}/(n - r + p))$, where n is the number of arguments of a functions' system, r is the number of arguments in block Y, and p is the number of intermediary variables. Note that with this estimation, the number of conjunctions on which one assigns functions of the system that appear in the constructed expansions is ignored. This program is used for systems of functions that depend on $n \le 20$ variables.

5.6. The TIE_BDD program for *joint BDD-minimizing* systems of Boolean functions implements algorithm [27] for minimizing multilevel representations of a system of Boolean functions based on Shannon's expansion. The initial data are DNF systems and the results are multilevel formula-based BDD representations. In conducting the experiments, this program constructs BDDs by nothing more than 5000 randomly taken permutations of variables and chooses the simplest BDD from the considered versions. The TIE_BDD program was used during the experiments in synthesizing circuits from library logic elements for optimizing BDD representations of the functions' systems.

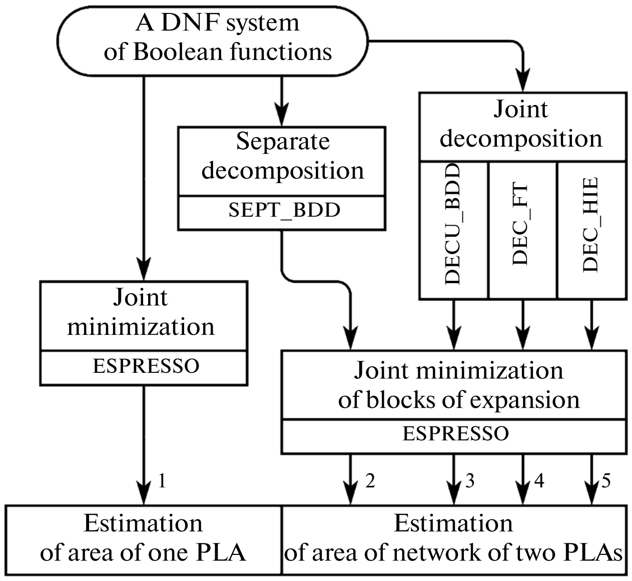
A DNF system of Boolean functions
Separate decomposition
SEPT_BDD
Joint decomposition
DECU_BDD
DEC_FT
DEC_HIE
Joint minimization
ESPRESSO
Joint minimization of blocks of expansion
ESPRESSO
1
2
3
4
5
Estimation of area of one PLA
Estimation of area of network of two PLAs

Fig. 7. Organization of experiments 1–5.

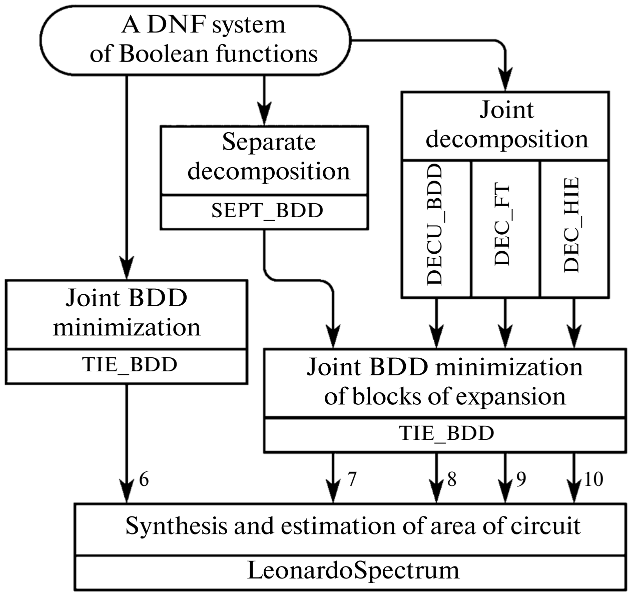

Fig. 8. Organization of experiments 6–10.

6. FORMULATION OF PROBLEMS OF EXPERIMENTAL RESEARCH

Problem 1. By the flow of practical examples of DNF systems of Boolean functions, compare the efficiency of using the programs of the joint minimization of functions and the programs of decomposition for decreasing the area of PLAs according to the block diagram of the experiments (Fig. 7).

Problem 2. By the flow of practical examples of DNF systems of Boolean functions, compare the efficiency of using the BDD representations of systems of functions and the programs of decomposition in synthesizing circuits from library logical elements according to the block diagram of the experiments (Fig. 8).

In order to solve Problem 1 and Problem 2, we conduct Experiments 1–5 and Experiments 6–10, respectively.

7. EXPERIMENTS

As the initial data for the experiments, 62 examples of DNF (PLA) systems from the Berkeley PLA Test Set library of examples are taken [28]. The system of functions in Experiments 1–5 is minimized and decomposed by the ESPRESSO program and the decomposition programs (Section 5), respectively. In addition, the decomposability (the existence of a useful functional expansion) for the collection of practical examples and the efficiency of various algorithms for choosing the Y/Z partition are investigated.

The result of the operation of all the compared decomposition programs represents the superposition of functions' systems that is implemented by two-element PLA networks.

E x p e r i m e n t 1 (ESPRESSO). The joint minimization of the initial functions' system of the ESPRESSO program and estimation of the area of one PLA that implements the minimized system of DNF functions.

E x p e r i m e n t 2 (SEPT_BDD+ESPRESSO). The separate decomposition of the initial functions' system by using the SEPT_BDD program [5], which implements the decomposition method based on a BDD, and then the integration of functions of the input cascade into one system and their joint minimization by using the ESPRESSO program; here, all functions of the output cascade are also jointly minimized. The analogous minimization after the decomposition is performed in Experiments 3–5.

E x p e r i m e n t 3 (DECU_BDD+ESPRESSO). The joint decomposition of the initial functions' system by using the DECU_BDD program [5], which implements the decomposition method based on a BDD.

E x p e r i m e n t 4 (DEC_FT+ESPRESSO). The joint decomposition of the initial functions' system by using the DEC_FT program [12], which implements the method based on the basis compact-tables' method [11].

E x p e r i m e n t 5 (DEC_HIE+ESPRESSO). The joint decomposition of the initial functions' system by using the DEC_HIE program [26], which implements the method based on the compact-tables' method [11].

E x p e r i m e n t s 6–10 are conducted to estimate the efficiency of using BDD representations in the synthesis of circuits from library logic elements for the library of designing Russian CMOS VLSIs; the contents of the library are presented in [29]. Each circuit from library elements is synthesized five times in the Leonardo Spectrum synthesizer (version 2011a.4) [6]. The synthesis in Experiment 6 is performed in terms of the BDD representation of an entire circuit, while the synthesis in Experiments 7–10 is carried out in terms of the decomposed representation of a circuit as two blocks; here, each block (input and output) is assigned by logic equations corresponding to its BDD representation.

The results of all experiments are verified with the use of the FormalPro system [30] of formal verification. In order to verify, the initial examples and the results of the decomposition (the superposition of the functions' systems) were transformed into descriptions represented in VHDL [6].

8. DISCUSSING THE RESULTS OF THE EXPERIMENTS

The results of Experiments 1–5 are presented in Table 17, where n is the number of input variables, m is the number of implemented Boolean functions, and k is the number of general elementary conjunctions in the DNFs of the functions of the initial system. The best decisions in Table 17 (and in Table 18) are marked with asterisks.

Minimizing the area. The main result of Experiments 1–5 is that in two-thirds of the examples (40 examples out of 56), the decomposition allows one to decrease the area of a PLA network better than the ESPRESSO program. This program outperforms the decomposition in one-third of cases; i.e., it allows one to implement the minimized functions' system on one PLA of the area, which is smaller than the area of two PLAs obtained by the decomposition. For six examples, namely, ALU1, NEWALPA, NEWAPLA1, NEWCPLA1, NEWCPLA2, and NEWTPLA, neither minimization nor decomposition allowed one to decrease the area of one PLA or a network of two PLAs. These examples are not presented in Table 17.

Choosing the partition of variables. The heuristic choice of one partition is unreasonable: the number of decisions that allow decreasing the area of the PLAs obtained in Experiment 4 is significantly smaller than the number of decisions obtained in Experiment 3 by the heuristic algorithm for searching for Y/Z partitions of variables of the set X with various powers of the set Y. The DECU_BDD program had found the variables' partitions by which useful decompositions were constructed for 38 examples. The DEC_FT program found such partitions for 30 examples. The heuristic search for a larger number of partitions by the DECU_BDD program allows one to find a larger number of decomposable examples than the DEC_FT program.

Table 17. Results of experiments 1–5

Initial data					Experiments				
Name of example	*n*	*m*	*k*	Area of initial PLA	1	2	3	4	5
					Area of one PLA	Area of two PLAs			
ADD6	12	7	1092	33852	11005	4246	8816	6187	*3637
ADDM4	9	8	512	13312	5200	7096	5394	*5170	6762
ADR4	8	5	256	5376	1575	978	1230	1623	*919
B12	15	9	431	16809	*1677	2229	1971	3105	1701
B2	16	17	110	5390	*5194	34240	7123	6468	–
B9	16	5	123	4551	4403	*3478	10580	11577	–
BR1	12	8	34	1088	*608	1139	612	842	748
BR2	12	8	35	1120	*416	846	416	656	536
CLPL	11	5	20	540	540	692	*291	1068	397
CO14	14	1	47	1363	406	*280	287	*280	504
DC2	8	7	58	1334	*897	1367	1103	1248	1240
DIST	8	5	256	5376	2583	3257	2574	2610	*2230
EX7	16	5	123	4551	4403	*3478	8020	11577	–
F51M	8	8	256	6144	1848	*1463	2094	1896	–
Gary	15	11	442	18122	*4387	14744	5765	6788	8391
IN0	15	11	138	5658	*4387	14744	6023	5536	8878
IN1	16	17	110	5390	*5194	34240	9778	6468	–
IN2	19	10	137	6576	6528	11538	8912	*6480	–
INTB	15	7	664	24568	23347	22957	42688	52236	*21597
LIFE	9	1	512	9728	1596	770	*607	1021	711
LOG8MOD	8	5	47	987	798	978	*738	897	761
M1	6	12	32	768	*456	1196	468	501	506
M181	15	9	430	16770	*1638	2265	1971	3555	1701
M2	8	16	96	3072	1504	3524	*1426	1552	1487
M3	8	16	128	4096	2112	4408	2116	2092	*1907
M4	8	16	256	8192	*3360	9374	3437	3440	3710
MAX1024	10	6	1024	26624	7124	8964	7348	*7095	7294
MAX46	9	1	46	874	874	*792	963	973	805
MAX512	9	6	512	12288	3480	4458	3516	*3384	3492
MLP4	8	8	256	6144	*3072	4184	3270	3168	–
MP2D	14	14	123	5166	*1302	1872	1854	4060	1530
NEWAPLA2	6	7	7	133	133	280	*124	187	135
NEWBYTE	5	8	8	144	144	336	*133	–	–
NEWCOND	11	2	31	744	744	*580	604	1060	676
NEWILL	8	1	8	136	136	*131	188	193	156
NEWTAG	8	1	8	136	136	*80	85	115	98
NEWTPLA	15	5	23	805	805	824	1405	1155	*701
NEWTPLA1	10	2	4	88	88	80	*73	88	150
P82	5	14	24	576	*504	3168	546	516	–
RADD	8	5	120	2520	1575	978	1239	1223	*881
RD53	5	3	32	416	403	298	291	265	*199
RD73	7	3	147	2499	2159	*673	836	836	821
ROOT	8	5	256	5376	1197	1579	1230	1245	*1059
RYY6	16	1	112	3696	3696	*423	*423	840	–
SEX	9	14	23	736	*672	1995	1277	1144	734
Soar	83	94	529	137540	*91780	99260	766920	–	–
SQN	7	3	96	1632	646	1034	683	673	*392
SQR6	6	12	64	1536	*1176	3468	1266	1203	–
SYM10	10	1	837	17577	4410	1015	1362	1070	*817
T3	12	8	152	4864	*1056	1382	1267	2375	1068
TIAL	14	8	640	23040	20916	59813	42822	57396	*19977
vtx1	27	6	110	6600	6600	*3930	7352	25710	–
x9dn	27	7	120	7320	7320	*4898	9673	21305	–
Z4	7	4	128	2304	1062	644	*440	1089	*440
Z5XP1	7	10	128	3072	*1560	3276	1863	1587	–
Z9SYM	9	1	420	7980	1634	838	794	794	*716
Number of the best decisions.				**0**	**19**	**12**	**9**	**5**	**14**

Table 18. Results of experiments 6–10

Name of example	Experiments									
	6		7		8		9		10	
	S^{f}_{BDD}	S_{circ}	S^{h+g}_{BDD}	S_{circ}	S^{h+g}_{BDD}	S_{circ}	S^{h+g}_{BDD}	S_{circ}	S^{h+g}_{BDD}	S_{circ}
ADD6	*60	20222	65	***18414**	86	32587	67	24502	66	21689
ADDM4	*189	89269	316	141754	237	97047	195	***86747**	468	212034
ADR4	*29	*7661	39	10931	45	13180	35	*7661	39	11740
ALU1	*18	*7109	33	*7109	74	31471	142	65107	139	56391
B12	*56	*16617	87	25194	107	34886	189	86412	84	24898
B2	*560	*197967	1079	370752	831	268738	683	259682	736	259766
B9	*71	*24770	95	30338	221	86557	390	168287	220	80246
BR1	*78	*27911	95	32464	89	31449	112	39880	106	32978
BR2	*72	*21634	90	26868	80	24435	94	35729	86	25969
CLPL	*16	*2929	33	7806	19	*2929	44	12572	20	*2929
CO14	*27	12449	32	13353	32	12895	32	13353	35	*11779
DC2	*62	*21946	89	33982	71	25132	103	45644	116	43502
DIST	*146	68601	211	92572	149	69956	152	***67501**	190	89793
EX7	*71	*24770	95	30338	195	76792	390	168287	220	80246
F51M	64	25344	*63	***23637**	71	31812	70	26131		
gary	*303	*98911	345	122258	447	155861	482	200517	424	170251
IN0	*303	*98911	345	122258	418	147864	442	170759	444	177868
IN1	*560	*197967	1079	370752	1154	353962	683	259682	688	243695
IN2	*257	*92410	420	128586	339	109535	408	149550	733	257238
INTB	*617	*227848	731	285344	891	377342	1204	563184	814	337902
LIFE	*26	14977	30	***13961**	39	16617	37	18353	35	15574
LOG8MOD	*62	*25953	74	32933	71	29741	79	34959	65	29602
M1	*47	18715	60	20004	52	***18554**	58	23330	58	20825
M181	*57	*17225	87	24714	107	35600	205	87796	84	24379
M2	*129	61095	140	54851	130	60515	133	55448	134	59438
M3	*141	60325	191	75944	173	78332	169	75358	155	***59818**
M4	*206	82266	314	127486	214	83081	212	***82232**	292	129138
MAX1024	*297	*140862	414	182031	315	146564	304	142050	521	237513
MAX46	*74	36878	81	***35980**	86	37587	96	41621	83	41376
MAX512	*175	*71446	227	98855	189	83761	181	*71446	265	120801
MLP4	*136	*71854	199	81953	176	84197	142	*71854		
MP2D	*71	*17968	85	19011	110	20819	239	101997	107	36125
NEWAPLA	53	*10189	62	13292	*51	13760	84	31109	56	14078
NEWAPLA1	*23	*6869	29	*6869	28	8119	35	9062	39	11300
NEWAPLA2	*20	*4944	24	*4944	22	*4944	26	8030	25	5569
NEWBYTE	21	*5569	25	*5569	*19	*5569	25	*5569		
NEWCOND	*29	*13414	45	14541	40	14687	79	35321	47	15657
NEWCPLA1	*82	28045	106	28598	97	32431	102	34256	90	*24831
NEWCPLA2	*60	*17438	96	27671	68	22582	70	18141	65	17778
NEWILL	*16	5312	18	***5028**	26	5134	27	7985	26	7840
NEWTAG	*10	*2126	12	*2126	12	*2126	16	3036	14	*2126
NEWTPLA	*52	*14229	64	16863	81	25919	96	33681	68	19156

Table 18. (Contd.)

Name of example	Experiments									
	6		7		8		9		10	
	$S^{\mathbf{f}}_{BDD}$	S_{circ}	$S^{\mathbf{h+g}}_{BDD}$	S_{circ}	$S^{\mathbf{h+g}}_{BDD}$	S_{circ}	$S^{\mathbf{h+g}}_{BDD}$	S_{circ}	$S^{\mathbf{h+g}}_{BDD}$	S_{circ}
NEWTPLA1	*13	3577	16	***3342**	15	3577	20	3510	24	3566
NEWTPLA2	*30	7310	37	9296	34	10970	31	9068	34	*6808
P82	*59	*19971	100	32213	68	22270	63	*19971		
RADD	*29	*8465	39	10083	47	14614	38	9157	35	10496
RD53	23	9843	28	7912	26	7399	26	7483	*21	***5312**
RD73	43	15925	44	13766	*39	13046	*39	*12734	*39	13046
ROOT	*75	*26717	103	39172	81	28692	81	*26717	105	42241
RYY6	*17	4224	19	***3197**	19	3298	26	5814	26	5563
SEX	*47	*13928	70	16539	65	16762	72	25065	52	14106
soar	*648	*167484	727	188063	4978	2268666				
SQN	50	23743	88	36298	58	24887	55	23124	37	***12990**
SQR6	*70	*27069	114	40550	76	30099	75	*27069	0	0
SYM10	*38	*19882	46	24825	49	18966	46	24279	43	20088
T3	*56	*19223	71	20055	75	22387	135	52965	68	26505
TIAL	*708	*295952	757	350441	909	369708	1319	690358	914	362109
vtx1	*87	*20216	106	24011	186	66145	550	199580		
x9dn	*91	*22342	111	23899	238	89224	447	169548		
Z4	*26	*6992	34	9519	*26	7405	31	*6992	*26	7405
Z5XP1	*67	*26499	103	40589	81	30690	72	26862		
Z9SYM	*33	18018	35	***14614**	44	16333	44	17544	37	17767
Number of the best decisions	**53**	**41**	**5**	**13**	**9**	**6**	**2**	**12**	**3**	**7**

The exhaustive search of all nontrivial partitions in Experiment 5 shows that we have useful expansions for all examples from the considered collection. The DEC_HIE program finds the partitions of variables that produce useful expansions for 53 examples; here, decomposition is not performed by this program for 9 examples. The DECU_BDD program finds 4 decomposable examples of the above 9 examples, which are not considered. Hence, the decomposability for 57 examples of 62, which represent the entire collection of practical examples, is corroborated. This fact demonstrates the advisability of developing decomposition methods for synthesizing circuits.

Decreasing the area under decomposition that is not useful. In some cases, decomposition ($r = p$) that is not useful makes it possible to decrease the area of a PLA: this is an example of the efficient symbolic recoding of the set of input variables by one multivalued variable. For instance, one cannot decrease the number of intermediary variables by using the DEC_FT program for the Max512, max1024, and addm4 examples; nevertheless, the resultant area of the PLA network decreased compared to the area of the initial PLA.

Dimension of problems. Using the decomposition based on the BDDs, one can solve problems whose dimension is larger than the dimension of problems that can be solved by using the decomposition based on the compact-tables' method. In some examples, the DEC_FT and DEC_HIE programs cannot perform the decomposition, because in order to represent intermediate information, large computer memory and a long time are required for searching all Y/Z partitions.

The results of experiments 6–10 are presented in Table 18, where $S^{\mathbf{f}}_{BDD}$ is the complexity of the BDD that implements the vector function $\mathbf{f}$; $S^{\mathbf{h+g}}_{BDD} = S^{\mathbf{h}}_{BDD} + S^{\mathbf{g}}_{BDD}$ is the sum of the complexities of two BDDs that represent the vector functions $\mathbf{h}$ and $\mathbf{g}$, respectively; and S_{circ} is the complexity (the sum of the elements' areas) of the circuit composed of library elements.

The analysis of experiments 6–10 shows that for the considered collection of practical examples, program-implemented algorithms of a large-block decomposition are weakly effective as a means for preliminary optimization in synthesizing circuits from library logic elements. The global technologically independent BDD-minimizing benefits the majority of cases by synthesizing on such a basis. The use of various kinds of decomposition makes it possible to obtain circuits that are better for the areas than in the case of preliminary BDD-minimizing only in 15 of the 62 examples of circuits (these results are indicated in bold in Table 18). Nevertheless, it is necessary to develop the means of decomposition, because the dimensionality of the solved problems increases; here, in the practice of designing, the number of examples of circuits of large dimensionality also increases (in the cases where industrial synthesizers construct inefficient circuits or cannot synthesize circuits from library elements).

Estimates of the complexity of the system of functions that are expressed in the number of the vertices of a BDD are feasible, because synthesizing a circuit from the representation corresponding to a BDD of the minimum-attained complexity makes it possible, as a rule, to obtain circuits of smaller areas. It is notable that for several examples (ADR4, RADD, RD53, RD73, ROOT, and Z9SYM) the TIE_BDD program found BDDs of minimum complexity (the complexities of the BDDs presented in [31] for these examples can be compared), although for finding the optimal permutation this program had tested only 5000 randomly generated permutations of variables. Note that the number of all permutations is $n!$ for a function that depends on n arguments. For the Z9SYM example, we have $n = 9$ and the number of all permutations $9! = 362880$. The results of solving these examples show that in the process of a random search one can stumble on the global optimum.

CONCLUSIONS

Decomposition can be efficiently used in practice to decrease the area of circuits in the basis of PLAs. By decreasing the number of arguments of the minorant functions, the decomposition allows decreasing the number of sought permutations in finding the best permutation of arguments, which ensures the least complexity of a BDD. The representation of the systems of functions as BDDs allows solving decomposition problems of larger dimensionality due to the more compact representation of BDDs compared to orthogonalized matrix representations of the systems of functions.

An interesting line of the future experimental research is studying the efficiency of the use of decomposition for solving problems of large dimensionality (hundreds of variables and functions) such that the global BDD optimization of an entire circuit and the subsequent synthesis are very laborious, while the local optimization of blocks obtained after the decomposition can be efficiently performed. Examples of such experiments for circuits of large dimensionalities are presented in [32].

REFERENCES

1. G. N. Povarov, "On functional separability of Boolean functions," Dokl. Akad. Nauk SSSR **94**, 801–803 (1954).
2. A. D. Zakrevskii, *Logic Synthesis of Cascade Circuits* (Nauka, Moscow, 1981) [in Russian].
3. R. E. Bryant and C. Meinel, "Ordered binary decision diagrams," in *Logic Synthesis and Verification*, Ed. by S. Hassoun, T. Sasao, and R. K. Brayton (Kluwer Academic, Dordrecht, 2002).
4. D. E. Knuth, *The Art of Computer Programming*, Vol. 4A: *Combinatorial Algorithms* (Addison-Wesley Professional, Reading, MA, 2011), Pt. 1.
5. P. N. Bibilo and P. V. Leonchik, "Decomposition of systems of Boolean functions determined by binary decision diagrams," J. Comput. Syst. Sci. Int. **50**, 609 (2011).
6. P. N. Bibilo and N. A. Avdeev, *VHDL. Effective Use in Design of Digital Systems* (SOLON-Press, Moscow, 2006) [in Russian].
7. R. L. Ashenhurst, "The decomposition of switching functions," Ann. Comput. Labor. Harvard Univ. **29**, 74–116 (1959).
8. H. A. Curtis, *A New Approach to the Design of Switching Circuit* (Van Nostrand, Princeton, 1962).
9. J. P. Roth and R. M. Karp, "Minimization over Boolean graphs," IBM J. Res. Dev. **6**, 227–238 (1962).
10. P. N. Bibilo and S. V. Enin, *Synthesis of Combinational Circuits by Methods of Functional Decomposition* (Nauka Tekhnika, Minsk, 1987) [in Russian].
11. Yu. V. Pottosin and E. A. Shestakov, *Tabular Methods for Decomposition of Systems of Completely Specified Boolean Functions* (Belarus. Navuka, Minsk, 2006) [in Russian].
12. D. V. Sadnikov, "Development of tabular method for decomposition of systems of completely specified Boolean functions," Informatika, No. 2, 79–85 (2005).

13. P. Porwik and R. S. Stankovic, “Dedicated spectral method of Boolean function decomposition,” Int. J. Appl. Math. Comput. Sci. **16**, 271–278 (2006).
14. P. N. Bibilo, *Decomposition of Boolean Functions Based on Logic Equations Solutions* (Belarus. Navuka, Minsk, 2009) [in Russian].
15. E. I. Gol’dberg, “Programmable logic matrix decomposition,” Preprint No. 6 (Inst. Tekhn. Kibernet. AN BSSR, Minsk, 1991).
16. M. A. Perkowski and S. Grugiel, “A survey of literature on function decomposition: Version IV,” Technical report (Portland State University, Department of Electrical Engineering, Portland, USA, 1995).
17. T. Sasao, “FPGA design by generalized functional decomposition,” in *Representations of Discrete Functions*, Ed. by T. Sasao and M. Fujita (Kluwer Academic, Dordrecht, 1996).
18. M. Fujita, Y. Matsunara, and M. Ciesielski, “Multi-level logic optimization,” in *Logic Synthesis and Verification*, Ed. by S. Hassoun, T. Sasao, and R. K. Brayton (Kluwer Academic, Dordrecht, 2002).
19. I. F. Cheburakhin and V. I. Tsurkov, “Syntheses discrete logical device information handling on base of the theories agent,” Mekhatron., Avtomatiz., Upravl., No. 3, 27–34 (2011).
20. I. F. Cheburakhin and V. I. Tsurkov, “Special relational database for optimization and automations of the syntheses combinational automaton,” Mekhatron., Avtomatiz., Upravl., No. 9, 7–13 (2010).
21. P. N. Bibilo and V. I. Romanov, *Logical Design of Discrete Devices with the Use of Production-Frame Model of Knowledge Representation* (Belarus. navuka, Minsk, 2011) [in Russian].
22. I. F. Cheburakhin and V. I. Tsurkov, “Optimization and automation of synthesis of symmetric combinational automatic devices on the basis of base matrix crystals,” Mekhatron., Avtomatiz., Upravl., No. 7, 19–29 (2009).
23. I. F. Cheburakhin and V. I. Tsurkov, “Logic control and processing of the information in mechatronics systems,” Mekhatron., Avtomatiz., Upravl., No. 5, 33–37 (2010).
24. A. S. Taghavi and Yu. V. Pottosin, “Improved decomposition for a system of completely specified Boolean functions,” Int. J. Inform. Technol. Comput. Sci. **6**, 25–32 (2013).
25. K. R. Brayton, G. D. Hactel, C. T. McMullen, and A. L. Sangiovanni-Vincentelli, *Logic Minimization Algorithm for VLSI Synthesis* (Kluwer Academic, Dordrecht, 1984).
26. A. S. Kh. Tagavi and Yu. V. Pottosin, “Study of separability properties of Boolean function systems,” Informatika, No. 4 (2013).
27. P. N. Bibilo and P. V. Leonchik, “An algorithm for constructing diagrams of binary choice for the system of completely specified Boolean functions,” Upravl. Sist. Mashiny, No. 6 (2009).
28. C. Jeong, “Computer-aided design of digital systems,” Department of Computer Science. http://www1.cs.columbia.edu/~cs6861/sis/espresso-examples/ex.
29. P. N. Bibilo and N. A. Kirienko, “Estimating energy consumption in logical CMOS circuits based on their switching activity,” Russ. Microelectron. **41**, 59 (2012).
30. A. Lokhov, “VLSI functional verification in light of Mentor Graphics,” Elektron.: Nauka, Tekhnol., Biznes, No. 1 (2004).
31. N. Ishiura, H. Sawada, and S. Yajima, “Minimization of binary decision diagrams based on exchanges of variables,” in *Proceedings of the IEEE International Conference on Computer-Aided Design ICCAD-1991* (Santa Clara, CA, USA, 1991).
32. P. N. Bibilo, *Application of Binary Choice Diagrams during Logic Circuits Synthesis* (Belarus. Navuka, Minsk, 2014) [in Russian].

Translated by L. Kartvelishvili