

Experimental Study of Algorithms for Minimization of Binary Decision Diagrams Using Algebraic Representations of Cofactors

P. N. Bibilo^{a,*} and V. I. Romanov^{a,**}

^a United Institute of Informatics Problems, National Academy of Sciences of Belarus,
Minsk, 220012 Belarus

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

**e-mail: rom@newman.bas-net.by

Received January 9, 2023; revised February 16, 2023; accepted March 21, 2023

Abstract—Binary decision diagram (BDD) is used for technology-independent optimization, performed as the first stage in the synthesis of logic circuits in the design of application-specific integrated circuits (ASICs). BDD is an acyclic graph defining a Boolean function or a system of Boolean functions. Each vertex of this graph is associated with the complete or reduced Shannon expansion formula. Binary decision diagrams with mutually inverse subfunctions (cofactors) are considered. We have developed algorithms for finding algebraic representations of cofactors of the same BDD level in the form of a disjunction or conjunction of other inverse or non-inverse cofactors of the same BDD level. The algorithms allow reducing the number of literals by replacing the Shannon expansion formulas of a system of Boolean functions. It is proposed to use the developed algorithms for an additional logic optimization of the constructed BDD representations of systems of Boolean functions. The experimental results of application of the corresponding programs in synthesizing the logic circuits in the design library of custom VLSI CMOS circuits are presented.

Keywords: system of Boolean functions, disjunctive normal form, binary decision diagram, Shannon expansion, digital logic synthesis, VHDL, VLSI

DOI: 10.1134/S0361768823040035

1. INTRODUCTION

Automation of logic design of functional units of combination logic entering the custom digital VLSIs (very large-scale integrated circuits) still is a topical scientific problem, because the dimensions of the solved problems increase, there appear more rigorous requirements to the energy consumption of logic circuits, and the libraries of logic elements change. Logic design of combination circuits in design libraries of custom VLSIs is traditionally divided into two stages: logic optimization and technological mapping, at which the optimized logic functional descriptions of the designed logic circuits are covered by functional descriptions of logic elements. Logic optimization, often called technology-independent optimization, usually does not involve the used basis (library) of logic elements, but tends to decrease the number of literals and the number of two-operand operations of disjunction and conjunction in the minimized algebraic representations of systems of completely specified Boolean functions. In the design practice it was noticed long ago [1, p. 44] that a reduction in the number of literals and logic operations in functional descriptions positively impacts the reduction in the number of elements of the technological library (basis

of synthesis) in the circuit, that is, positively impacts the reduction in the total number of transistors entering all elements of the circuit. Decreasing the number of transistors allows lowering the energy consumption of circuits.

Today, logic minimization is made by means of the methods of optimization of multilevel Boolean functions on the basis of Shannon expansion; graphical forms of such representations, frequently used as data structures, are called binary decision diagrams [2–7]. In work [8] we proposed a method for additional minimization of BDD representations of systems of Boolean functions based on search of disjunctive, conjunctive, and modular expansions of subfunctions entering the optimized BDD representations. The method allows replacing the formulas of Shannon expansion by two-operand formulas of disjunction $g_3 = g_1 \vee g_2$, conjunction $g_3 = g_1 \& g_2$, or modular formulas of type $g_3 = g_1 \oplus g_2$ (where g_1 , g_2 , and g_3 are cofactors of the same BDD level). Such replacement reduces the number of literals and logic operations in the resulting logic formulas, which are used to perform technological mapping, that is, the final stage of circuit synthesis. In the current work the method from [8] is elaborated to algorithms and programs whose efficiency is stud-

Table 1. Cofactors of second BDD level (Fig. 1)

$x_1 x_2$	g_1	g_2	g_3	g_4	g_5	g_6	\bar{g}_1	\bar{g}_2	\bar{g}_3	\bar{g}_4	\bar{g}_5	\bar{g}_6
0 0	0	0	0	1	0	1	1	1	1	0	1	0
0 1	1	1	0	0	0	0	0	0	1	1	1	1
1 0	1	0	1	0	0	0	0	1	0	1	1	1
1 1	0	0	0	0	1	1	1	1	1	1	0	0

ied on flows of examples entering the well-known example library used to check and compare the programs of logic optimization and synthesis of logic circuits. For the expansions of cofactors we use only disjunctive and conjunctive expansions, and the cofactors in such expansions may be in both inverse and non-inverse form.

2. BINARY DECISION DIAGRAMS AND ALGEBRAIC EXPANSIONS OF COFACTORS

The *Shannon expansion* of a completely specified Boolean function $f = f(\mathbf{x})$, $\mathbf{x} = (x_1, x_2, \dots, x_n)$, in variable x_i is the representation

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

The functions $f_0 = f(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n)$ and $f_1 = f(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n)$ in the right-hand side of Eq. (1) are referred to as *cofactors* of expansion in variable x_i . They are obtained from the function f by substitution of the constants 0 and 1, respectively, instead of the variable x_i . Each of the cofactors f_0 and f_1 can be expanded in one variable from the set $\{x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n\}$. The process of expansion of cofactors terminates either when all n variables are used for expansion or when all cofactors degenerate to constants 0, 1. At each step of expansion we perform comparison by equality of the obtained cofactors.

Let $\mathbf{f}(\mathbf{x})$ be an ordered system of completely specified component Boolean functions $f_i(\mathbf{x})$, $i = 1, \dots, m$, (the vector Boolean function $\mathbf{f} = (f_1, \dots, f_m)$): $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_m(\mathbf{x}))$, $\mathbf{x} = (x_1, x_2, \dots, x_n)$. Under the BDD representation of a vector Boolean function $\mathbf{f}(\mathbf{x})$ we understand the oriented acyclic graph prescribing sequential Shannon expansions of all component functions $f_i(\mathbf{x})$, $i = 1, \dots, m$, in all its variables x_1, \dots, x_n for the same given order (permutation) of variables in which we perform expansions. The BDDs considered in this paper correspond to shared reduced ordered BDDs (ROBDDs) [5, p. 18]. In joint ROBDDs the functional vertices of BDDs correspond to the cofactors common to the functions of the system; here, the functional vertices are just implied (identified with vertices-variables). Description of OBDDs (ordered BDDs) is given in [2, p. 90; 5, p. 16], whereas descrip-

tion of ROBDDs is given in [3, p. 243]. Below, BDDs will be understood as joint ROBDDs for the systems of functions (vector functions).

The BDD representation corresponds to the formulas of Shannon expansion, and each functional vertex of BDD [6, p. 18] corresponds to its own formula in which both cofactors take part in non-inverse form. Using the BDD representation, we can find specification of each of the component functions $f_i(\mathbf{x})$ in form of two orthogonalized disjunctive normal forms (DNFs): one of such DNFs prescribes the region $M_{f_i}^1$ of unit values of the function $f_i(\mathbf{x})$, whereas the other DNF prescribes the region $M_{f_i}^0$ of zero values. Similarly, we can find the representations in form of DNF for each of the cofactors. These transitions are thoroughly described in [6, p. 37]. Figure 1a shows a BDD which corresponds to the following coupled equations of Shannon expansions:

$$f_1 = \bar{x}_3 g_1 \vee x_3 x_1; \quad f_2 = \bar{x}_3 g_2 \vee x_3 g_4; \\ f_3 = \bar{x}_3 g_5 \vee x_3 g_3;$$

$$f_4 = \bar{x}_3 \vee x_3 g_6; \quad g_1 = \bar{x}_1 s_1 \vee x_1 s_2; \\ g_2 = \bar{x}_1 s_1; \quad g_3 = x_1 s_2; \quad g_4 = \bar{x}_1 s_2;$$

$$g_5 = x_1 s_1; \quad g_6 = \bar{x}_1 s_2 \vee x_1 s_1; \quad s_1 = x_2; \quad s_2 = \bar{x}_2.$$

The cofactors g_1, \dots, g_6 are determined at the second BDD level and depend on two variables x_1, x_2 ; the cofactors at the first level depend on the variable x_2 : $s_1 = x_2, s_2 = \bar{x}_2$ (see Fig. 1a). At the bottom BDD level there always are constants 0, 1 that can be doubled for simplifying the depiction of the BDD graph in the figure. Figure 1b shows the graph of the same BDD in the notation accepted in the foreign literature: dashed curves in Fig. 1 correspond to labels 0, whereas solid ones correspond to labels 1.

Using the BDD graph, we can easily find the cofactors appearing at the same BDD level, and from the prescription of cofactor (or the function of the original system) in form of DNF we can proceed to its prescription in the form of completed DNF (PDNF) or truth table. We consider the cofactors $g_1, g_2, g_3, g_4, g_5, g_6$ of the second BDD level (Fig. 1a), eliminate the intermediate variables, and obtain the completed DNFs:

$$g_1 = \bar{x}_1 s_1 \vee x_1 s_2 = \bar{x}_1 x_2 \vee x_1 \bar{x}_2; \\ g_2 = \bar{x}_1 s_1 = \bar{x}_1 x_2; \quad g_3 = x_1 s_2 = x_1 \bar{x}_2; \\ g_4 = \bar{x}_1 s_2 = \bar{x}_1 \bar{x}_2; \quad g_5 = x_1 s_1 = x_1 x_2; \\ g_6 = \bar{x}_1 s_2 \vee x_1 s_1 = \bar{x}_1 \bar{x}_2 \vee x_1 x_2.$$

We prescribe the CDNF of cofactors and their inverses by the truth table (Table 1).

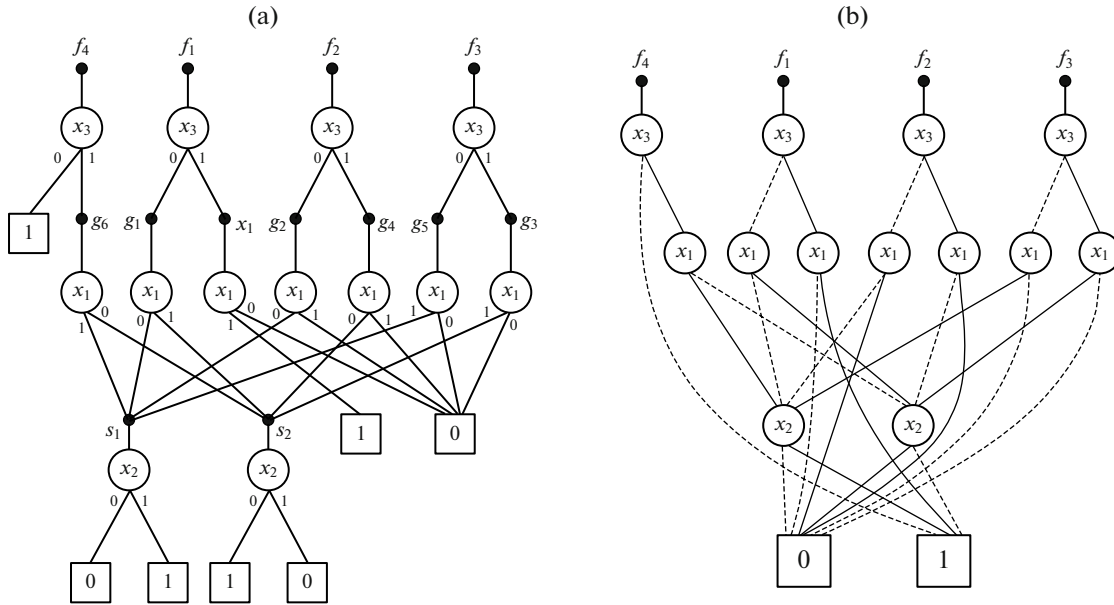


Fig. 1. (a) BDD with labeled cofactors and (b) BDD in traditional image.

Using the truth tables, we can easily check the representations of cofactors of the same BDD level in form of *algebraic expansions*: *disjunctive*,

$$g_p = g_i \vee g_j, \tag{2}$$

or *conjunctive*,

$$g_r = g_i \& g_j \tag{3}$$

expansion. We can also use inversions \bar{g}_i, \bar{g}_j of cofactors g_i, g_j in expansions of form (2), (3). For the cofactors that have four literals in their notation (these are the cofactors g_1, g_6), we write the equations of their algebraic expansions via other cofactors:

$$\begin{aligned} g_1 &= \bar{g}_4 \& \bar{g}_5; & g_1 &= g_2 \vee g_3; \\ g_6 &= \bar{g}_2 \& \bar{g}_3; & g_6 &= g_4 \vee g_5 \end{aligned}$$

and represented the obtained equations in form of orgraph (Fig. 2).

We can easily see (Fig. 3) that the cofactors $g_1 = \bar{x}_1 s_1 \vee x_1 s_2$ and $g_6 = \bar{x}_1 s_2 \vee x_1 s_1$ can be expressed by the formulas of algebraic expansions through two cofactors g_2, g_3 : $g_1 = g_2 \vee g_3$; $g_6 = \bar{g}_2 \& \bar{g}_3$. Replacement of the formulas $g_1 = \bar{x}_1 s_1 \vee x_1 s_2, g_6 = \bar{x}_1 s_2 \vee x_1 s_1$ by simpler ones allows reducing the number of literals and logic operations in the functional description of BDD (Fig. 1a).

Note that the cofactors g_1 and g_6 are mutually inverse; therefore, below, as original formulas, we use the formulas of Shannon expansions corresponding to BDDs. Binary decision diagram with inverse cofactors (BDDI) is below understood as oriented acyclic graph prescribing sequential Shannon expansions of a Boolean function $f(x_1, \dots, x_n)$ in all its variables $x_1, \dots,$

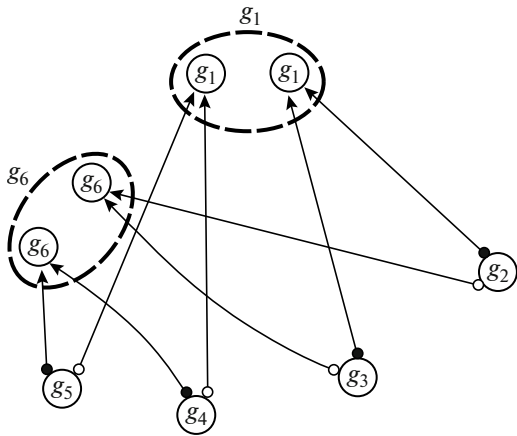


Fig. 2. Prescription of equations in form of orgraph.

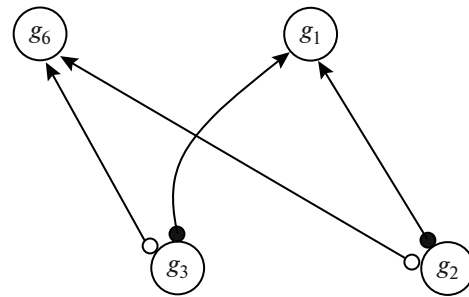
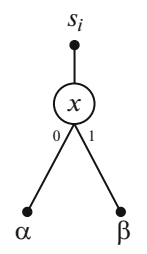
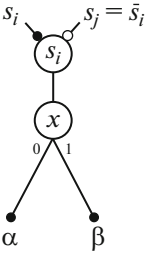


Fig. 3. Reduced acyclic orgraph.

Table 2. Correspondence of graphic BDDI elements and logic formulas

Graphical element	Formula
	$s_i = \bar{x}\alpha \vee x\beta$
	$s_i = \bar{x}\alpha \vee x\beta$ $s_j = \bar{s}_i$

x_n for a given order (permutation) of variables by which we perform expansions, under the condition of finding the pairs of mutually inverse cofactors [9]. The BDDI graphs contain four types of vertices: root vertices corresponding to the functions; functional vertices corresponding to the pairs of expanded mutually inverse cofactors (one of the elements of the pair can be absent if there is no such cofactor among the cofactors at the given BDDI level); vertices-variables; and leaf vertices corresponding to constants 0, 1. A functional vertex of BDDI (Table 2) implements one cofactor s_i or two cofactors: s_i and its inversion \bar{s}_i . The formula $s_i = \bar{x}\alpha \vee x\beta$ in the right column in Table 2 is formula (1) of Shannon expansion of the cofactor s_i in the variable x : $\alpha = f_0$, $\beta = f_1$.

Note that in the BDD the functional vertex implements a single cofactor; mutually inverse cofactors are represented by a pair of vertices. The BDDIs correspond to joint ROBDDs in which the pair of mutually inverse vertices-cofactors is given by a single functional vertex. In the equations of Shannon expansions corresponding to BDDIs, the cofactors may be either with the inverse sign or in the

In what follows, we propose the algorithms of finding the maximal sets of algebraic expansions of form (2), (3) of the cofactors appearing at the same BDDI level. The main issue is as follows: at prescription of the algebraic equations in form of orgraph there may arise cycles, which is unacceptable in circuit implementation of the combination logic. In the considered simple example of orgraph (Fig. 2) illustrating prescription of equations, there are no cycles. Transition to the resulting acyclic orgraph (Fig. 3), in which each cofactor is represented by a single algebraic expansion, is a complex combinatorial problem for graphs con-

taining hundreds and thousands of vertices and in which there may be a vast number of cycles that must be eliminated. Preliminary determining the pairs of mutually inverse cofactors (using BDDIs instead of BDDs) reduces the combinatorial exhaustive search in finding the variants of algebraic expansions of cofactors; we will explain it further.

3. ALGORITHM 1 FOR FINDING ALGEBRAIC EXPANSIONS OF COFACTORS IN BDDI

Initial data for the proposed algorithm 1 for finding algebraic representations of cofactors are the BDDI graph, representing an original vector completely specified Boolean function, and logic equations (the formulas of Shannon expansion), corresponding to the functional vertices of the BDDI and prescribing the multilevel description of the component functions. These equations can be easily written using a BDDI graph.

The proposed algorithm includes the stages executed for the cofactors of each BDDI level, except for the root, leaf, and first one. At the first BDDI levels, there are the cofactors obtained as a result of expansion in the last variable in a given permutation of variables in which we perform the Shannon expansion. The cofactors of the first BDDI level depend on one variable; the cofactors of the second level depend on at most two variables; etc. In our algorithm, the BDDI levels are considered from top to bottom, that is, from the cofactors obtained at expansion in the first variable to the cofactors of the second BDD level. The algorithm is focused on replacing the largest number of formulas of Shannon expansion by the formulas of either disjunction or conjunction, which allows reducing the total number of literals in the multilevel prescription of a system of Boolean functions. For the considered BDDI level we solve the problem of determining the largest number of cofactors that are representable in form of disjunction or conjunction of other cofactors of the given level (earlier, such representations were called algebraic expansions).

Stage 1. Divide the set of equations into the BDDI levels.

Stage 2. For each BDDI level (except for the root, leaf, and first one), find the maximal set of implementable equations (implementable cofactors).

Execution of stage 2 reduces to the execution of the following steps.

Step 2.1. Eliminate the intermediate variables and construct the truth table of cofactors of the given (considered) BDDI level.

Step 2.2. Add the inversions of cofactors to the truth table.

Table 3. Equations for BDDI (Fig. 4)

Number of BDDI level	Equations
5	$f_1 = \bar{x}_1 h_1 \vee x_1 h_2; f_2 = \bar{x}_1 h_2 \vee x_1 h_3; f_3 = \bar{x}_1 h_4 \vee x_1 h_5; f_4 = \bar{x}_1 h_4 \vee x_1 h_6; f_5 = \bar{x}_1 h_6 \vee x_1 h_7;$
4	$h_1 = \bar{x}_2 \bar{g}_1 \vee x_2 \bar{g}_3; h_2 = \bar{x}_2 g_2 \vee x_2 \bar{g}_8; h_3 = \bar{x}_4 g_5 \vee \bar{x}_2 \bar{g}_5; h_4 = \bar{x}_2 g_6 \vee x_2 \bar{g}_7; h_5 = \bar{x}_2 g_6 \vee x_2 g_{10}; h_6 = \bar{x}_2 g_4 \vee x_2 g_9;$ $h_7 = \bar{x}_2 g_8 \vee x_2 g_{10};$
3	$g_1 = \bar{x}_4 s_2 \vee x_4 s_3; g_2 = \bar{x}_4 \bar{x}_5 \vee x_4 s_1; g_3 = \bar{x}_4 s_0 \vee x_4 s_1; g_4 = \bar{x}_4 x_5 \vee x_4 s_2; g_5 = \bar{x}_4 s_0 \vee x_4 x_5; g_6 = \bar{x}_4 \bar{s}_0 \vee x_4 s_4;$ $g_7 = \bar{x}_4 \bar{x}_5 \vee x_4 x_5; g_8 = \bar{x}_4 s_3 \vee x_4 \bar{s}_1; g_9 = \bar{x}_4 s_2 \vee x_4 \bar{x}_5; g_{10} = \bar{x}_4 x_3 \vee x_4 x_5;$
2	$s_0 = \bar{x}_5 x_3; s_1 = x_5 \bar{x}_3; s_2 = \bar{x}_5 \bar{x}_3; s_3 = x_5 x_3; s_4 = \bar{x}_5 \bar{x}_3 \vee x_5 x_3.$
1	Literals x_3, \bar{x}_3
0	Constants 0, 1

Step 2.3. Find all variants of algebraic representations of non-inverse cofactors in form of two-operand disjunctions or conjunctions of other cofactors (or their inversions) of the considered BDDI level. As a result, a single cofactor, we call it *representable*, can be represented by different equations.

Step 2.4. Find the set of implementable cofactors.

From the set of representable cofactors we separate the subset of *implementable* cofactors with maximum cardinality; for this purpose, from the set of equations of the algebraic representation of one cofactor we choose one equation or choose no equation. This choice is determined by solution to problem 1 that is formulated in terms of the graph theory.

Stage 3. Correct the original multilevel BDDI representation of the system of Boolean functions.

The original set of formulas of Shannon expansions varies: the formulas of Shannon expansion of the implementable cofactors are replaced by the given disjunction or conjunction formulas.

Let us estimate the number of searched-through variants at step 2.3. Suppose that k is the number of non-inverse cofactors g_1, \dots, g_k . Then we need to consider $12 \times C_k^3$ (C_k^3 is the number of combinations of k cofactors by triples, $k \geq 3$) variants of testing the disjunctive expansions: the number of all distinct unordered triples of cofactors g_p, g_i , and g_j is C_k^3 , and for each triple $\{g_p, g_i, g_j\}$ we need to test 12 disjunctive expansions,

$$\begin{aligned} g_p &= \bar{g}_i \vee g_j; & g_p &= g_i \vee \bar{g}_j; \\ g_p &= \bar{g}_i \vee \bar{g}_j; & g_p &= g_i \vee g_j; \\ g_i &= \bar{g}_p \vee g_j; & g_i &= g_p \vee \bar{g}_j; \\ g_i &= \bar{g}_p \vee \bar{g}_j; & g_i &= g_p \vee g_j; \\ g_j &= \bar{g}_i \vee g_p; & g_j &= g_i \vee \bar{g}_p; \\ g_j &= \bar{g}_i \vee \bar{g}_p; & g_j &= g_i \vee g_p. \end{aligned}$$

Similarly, for each of C_k^3 triples of cofactors we need to search through 12 variants for constructing the conjunctive expansions.

Now, we can easily explain that using the BDDI instead of BDD reduces the combinatorial exhaustive search of variants in determining the formulas of disjunctive and conjunctive expansions. For instance, if it is not preliminary established that cofactors g_p and g_s are mutually inverse, then, in finding the expansions and determining the formula $g_p = g_i \vee g_j$, for the inversion $\bar{g}_p = g_s$ we always can find the dual formula $g_s = \bar{g}_i \& \bar{g}_j$. Thus, for each formula of the algebraic expansion of the non-inverse cofactor, we will find the dual formula for the inverse cofactor and the search through the triples of cofactors appears to be more labor-intensive.

4. DETERMINATION OF THE SET OF IMPLEMENTABLE COFACTORS

Let us construct an orgraph G specifying the formulas of algebraic representations of cofactors (we call them logic equations or simply equations). The vertices of the orgraph G correspond to the cofactors in the direct form that are mentioned in equations. Let us denote this set of cofactors by K_Y and the set of cofactors for which we have not constructed the equations by K_Z . The edges (oriented edges) corresponding to the operands of algebraically represented cofactor (result of logic operation) are labeled by the same number, the number of equation. Each equation is given by a subgraph with three vertices and two labeled edges: the vertex corresponding to a representable cofactor has two input edges (with one and the same label, number of equation). The edges arising from the vertex of a representable cofactor (Fig. 3) can correspond to the direct form of cofactor (filled circle) or to the inverse form of cofactor (unfilled circle).

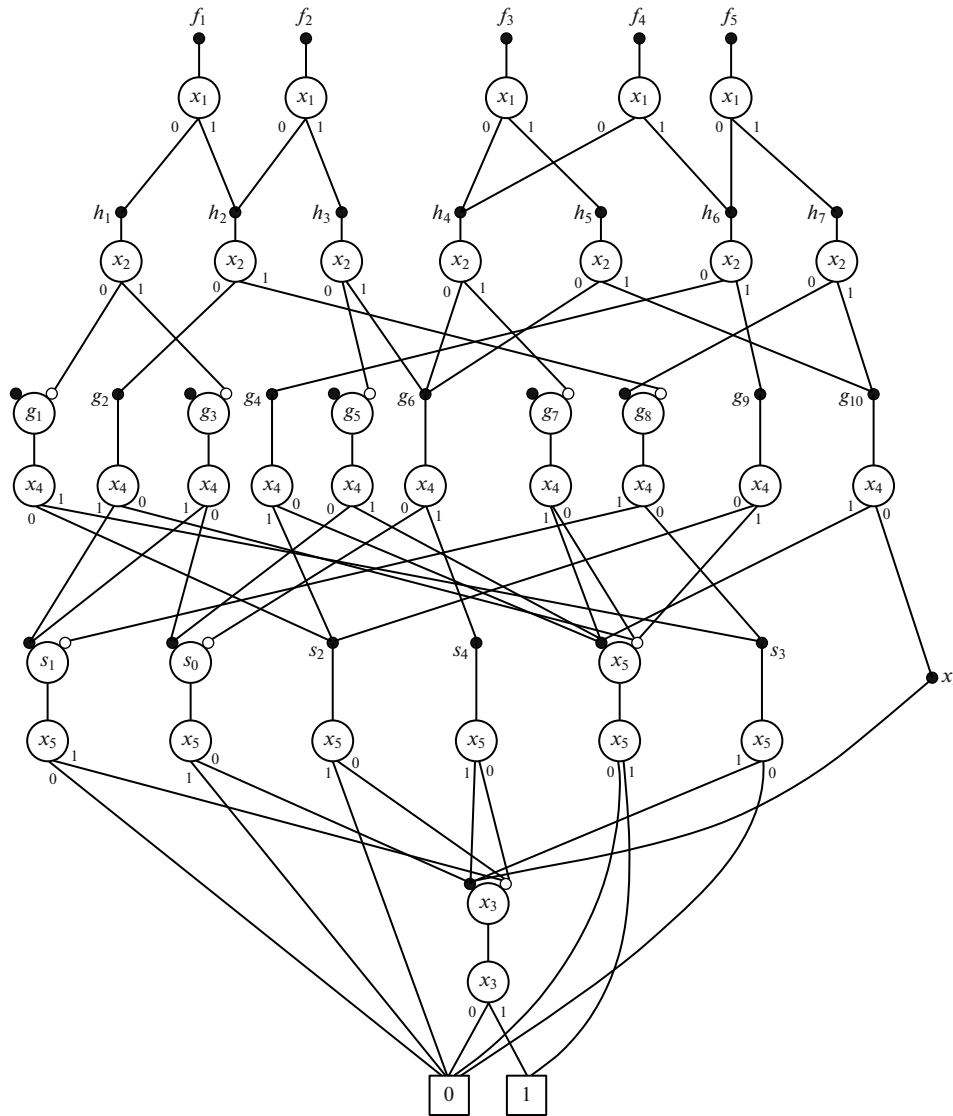


Fig. 4. BDDI graph.

Let us introduce the notion of *cluster* (subset of the vertices of the graph G): the cluster includes the vertices corresponding to the equations with the same cofactor in the left-hand side. A cluster is labeled by the name of the same cofactors.

Using the orgraph G , we can reduce the problem of determining the set of implementable cofactors to the following problem.

Problem 1. In the orgraph G we need to leave just a single vertex in each cluster (condition A) and remove such subset of pairs of equally labeled edges from the graph G that the orgraph G becomes acyclic (condition B) and contains the largest number of nonisolated vertices with just a pair of incoming equally labeled edges (condition C).

The meaning of leaving a single vertex in the cluster (condition A) is clear: it suffices to represent the cofactor by a single equation in the circuit implementation. We demonstrate the requirement of the absence of cycles (condition B) on example of two equations $g_6 = g_1 \vee g_4$ and $g_1 = g_6 \& g_7$ from the below considered example. These equations create a cycle in the orgraph G , that is, the following logic contradiction in the requirement of algebraic (and circuit) representation of cofactors. To express g_6 as a conjunction $g_6 = g_1 \vee g_4$, we need a circuit implementing the cofactor g_1 . However, in order to obtain g_1 in form of $g_1 = g_6 \& g_7$, we need a circuit implementing g_6 . Condition C requires that the largest possible number of cofactors are represented in form of conjunction or disjunction.

Table 4. Cofactors and their inversions of third BDDI level (Fig. 4)

$x_3 x_4 x_5$	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}	\bar{g}_1	\bar{g}_2	\bar{g}_3	\bar{g}_4	\bar{g}_5	\bar{g}_6	\bar{g}_7	\bar{g}_8	\bar{g}_9	\bar{g}_{10}
0 0 0	1	1	0	0	0	1	1	0	1	0	0	0	1	1	1	0	0	1	0	1
0 0 1	0	0	0	1	0	1	0	0	0	0	1	1	1	0	1	0	1	1	1	1
0 1 0	0	0	0	1	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1
0 1 1	0	1	1	0	1	0	1	0	0	1	1	0	0	1	0	1	0	1	1	0
1 0 0	0	1	1	0	1	0	1	0	0	1	1	0	0	1	0	1	0	1	1	0
1 0 1	0	0	0	1	0	1	0	1	0	1	1	1	1	0	1	0	1	0	1	0
1 1 0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	1	1	1	0	0	1
1 1 1	1	0	0	0	1	1	1	1	0	1	0	1	1	1	0	0	0	0	1	0

After solving problem 1, we divide the set K_Y of vertices of the orgraph G into three mutually nonintersecting subsets. The vertices with no incoming edges, but with an emerging edge correspond to the unimplementable cofactors. The vertices with one pair of incoming edges (with the same label) correspond to the implementable cofactors. The vertices appeared to be isolated, for instance, at removing the edges to satisfy condition B , will not participate in the equations are included in the set K_Z of also unimplementable cofactors. An implementable cofactor will be written as an equation to the resulting multilevel representation of the system of Boolean functions.

Algorithm for Solving Problem 1

Stage A1. Selection of one equation in each cluster.

Step A1.1. Arrange the clusters in ascending (non-descending) order by the number of vertices in them and consider the clusters in this order.

Step A1.2. Leave just a single vertex in each cluster.

Put the current set T of vertices empty: $T = \emptyset$. Each remaining vertex g_r in the cluster that have the incoming equally labeled edges incident to the vertices g_i and g_j fulfills the set T by the elements g_i, g_j , and g_r .

To choose the vertex left in the first cluster, we use heuristic 1.

Heuristic 1. Each vertex g_r of the cluster with incoming edges from the vertices g_i and g_j is evaluated by the total number W_{q_r} of edges arising from the vertices g_i and g_j . In the cluster we leave the vertex g_s corresponding to the maximal number W_{q_r} .

The vertices that have not remained in the cluster are removed from the graph together with the edges incoming in them. We introduce three vertices into the set T : the vertex g_s and the two vertices incident to the edges incoming to the vertex g_s .

For the following clusters at consideration we use heuristic 2.

Heuristic 2. In a cluster we leave the vertex that has two incoming edges from the vertices of the set T ; if

there are no such vertices, then we leave the vertex that has one incoming edge from the vertices of the set T . If there are no such vertices for which heuristic 2 is fulfilled, then we use heuristic 1 to leave a vertex in a cluster. The vertices not remaining in the cluster are removed from the graph together with the edges incoming to them.

Step A1.2 and stage A1 are accomplished when a single vertex remains in each cluster.

Stage A2. Reduce the orgraph G to acyclic graph.

We iteratively execute steps A2.1–A2.3 until the orgraph becomes acyclic.

Step A2.1. Test the orgraph at consideration for the absence of cycles.

At this step we perform construction of the set of simple cycles of the graph. For this purpose, we execute exhaustive search over all the vertices of the graph. For the next chosen vertex we carry out depth-first search to construct all possible paths in the graph starting from this vertex. Such procedure is a modification of the software implementation [10] in the C# language of the well-known Johnson algorithm [11].

If cycles are revealed as a result of testing, then we execute step A2.2. If the orgraph has no cycles, then we proceed to step A2.4.

Step A2.2. Find the equation removed from the graph, that is, the pair of edges with the same labels corresponding to the equation and incoming to the same vertex of the graph.

Each pair of edges with the same label (number of equation) p is evaluated by the number C_p of cycles that can be broken when this pair of edges is removed from the graph.

Step A2.3. Remove the pair of edges (equation) with label p that corresponds to the maximal value of C_p . Proceed to step A2.1.

Step A2.4. End.

Table 5. Equations of algebraic expansions of cofactors of third BDDI level

Number of equation	Equation	Cluster	Weight of cluster
1	$g_1 = g_6 \& \bar{g}_4$	g_1	3
2	$g_1 = g_6 \& g_7$		
3	$g_1 = g_7 \& \bar{g}_3$		
4	$g_2 = \bar{g}_4 \& \bar{g}_8$	g_2	2
5	$g_2 = g_7 \& \bar{g}_8$		
6	$g_3 = \bar{g}_6 \& \bar{g}_8$	g_3	14
7	$g_3 = \bar{g}_6 \& \bar{g}_9$		
8	$g_3 = g_{10} \& g_6$		
9	$g_3 = g_{10} \& \bar{g}_8$		
10	$g_3 = g_2 \& \bar{g}_1$		
11	$g_3 = g_2 \& \bar{g}_6$		
12	$g_3 = g_2 \& \bar{g}_9$		
13	$g_3 = g_2 \& g_{10}$		
14	$g_3 = g_2 \& g_{10}$		
15	$g_3 = g_5 \& \bar{g}_1$		
16	$g_3 = g_5 \& \bar{g}_6$		
17	$g_3 = g_5 \& \bar{g}_8$		
18	$g_3 = g_7 \& \bar{g}_1$		
19	$g_3 = g_7 \& \bar{g}_6$		
20	$g_4 = g_6 \& \bar{g}_1$	g_4	2
21	$g_4 = g_6 \& \bar{g}_7$		
22	$g_5 = \bar{g}_4 \& \bar{g}_9$	g_5	4
23	$g_5 = g_{10} \& \bar{g}_4$		
24	$g_5 = g_7 \& \bar{g}_9$		
25	$g_5 = g_7 \& g_{10}$		
26	$g_6 = g_1 \vee g_4$	g_6	1
27	$g_7 = g_1 \vee g_2$	g_7	4
28	$g_7 = g_1 \vee g_3$		
29	$g_7 = g_1 \vee g_5$		
30	$g_7 = g_2 \vee g_5$		

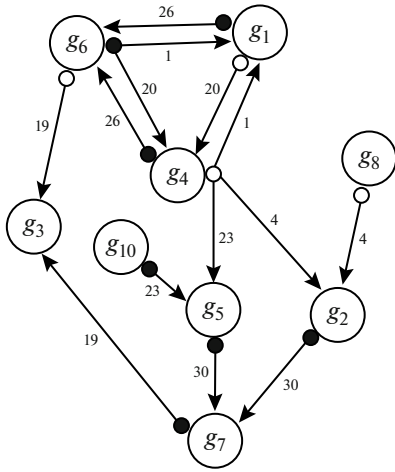


Fig. 5. Orgraph G after cluster elimination.

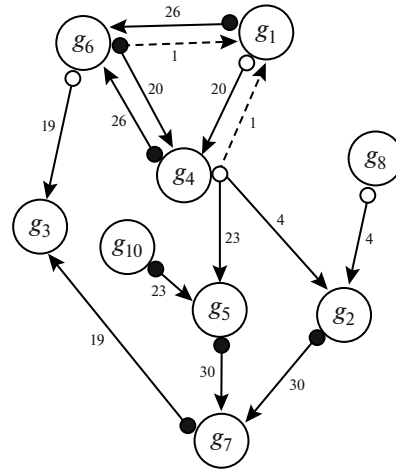


Fig. 6. Step 1 of converting a graph to acyclic one (dashed edges corresponding to equation 1 are removed).

5. EXAMPLE OF EXECUTION OF ALGORITHM 1

Let us illustrate the proposed algorithm 1 on example of the BDDI depicted in Fig. 4. We consider the third BDDI level whose cofactors depend on the variables $x_3, x_4,$ and x_5 .

Stage 1. Partition the set of equations into the BDDI levels.

The equation for each of the BDDI levels are given in Table 3.

Stage 2. Find the set of implementable cofactors.

Steps 2.1–2.3. Using program [12] of elimination of intermediate variables, construct the truth table of the cofactors of the third BDDI level and find the inversions of the cofactors (Table 4). After that, perform the complete combinatorial exhaustive search over the variants of determining the disjunctive and conjunctive expansions of non-inverse cofactors from Table 4. Write the obtained algebraic formulas in Table 5.

Step 2.4.

Stage A1 of solving problem 1. Choose equations in clusters (cluster reduction).

Step A1.1. The order of cluster consideration follows increasing (nondecreasing) cluster weights: $\langle g_6, g_2, g_4, g_1, g_5, g_7, g_3 \rangle$.

Step A1.2.

1. Consider the cluster g_6 ; leave the equation $g_6 = g_1 \vee g_4$; the set $T = \{g_1, g_4, g_6\}$.

2. Consider the cluster g_2 ; leave the equation $g_2 = \bar{g}_4 \& \bar{g}_8$; implement heuristic 1; the set T is fulfilled by the vertices from the preserved equation: $T = \{g_1, g_2, g_4, g_6, g_8\}$.

3. Consider the cluster g_4 ; the maximum weight $W_{g_4} = 18$ is for the vertex g_4 for the equation $g_4 = g_6 \& \bar{g}_1$;

keep the equation $g_4 = g_6 \& \bar{g}_1$; implement heuristic 1; the set $T = \{g_1, g_2, g_4, g_6, g_8\}$ is left unchanged.

Similarly, we preserve the equations in the other clusters $g_1, g_5, g_7,$ and g_3 . The result of execution of stage A1 is the orgraph (Fig. 5) and the corresponding equations (Table 6).

Stage A2 of solving problem 1. Reduce the orgraph G to an acyclic graph.

As an original, we consider the orgraph (Fig. 5).

Iteration 1 (there will be iterations 2 and 3 below).

At *step A2.1* we find all simple cycles: $\{g_1, g_6, g_1\}, \{g_1, g_4, g_1\}, \{g_1, g_4, g_6, g_1\}, \{g_1, g_6, g_4, g_1\},$ and $\{g_4, g_6, g_4\}$.

At *step A2.2*, for each of the equations (pairs of identically labeled edges) 1, 20, 26 we count the number of broken cycles if we can remove this pair of edges

Table 6. Equations implemented in graph G after reducing the cofactors

Number of equation	Equation	Cluster
1	$g_1 = g_6 \& \bar{g}_4$	g_1
4	$g_2 = \bar{g}_4 \& \bar{g}_8$	g_2
19	$g_3 = g_7 \& \bar{g}_6$	g_3
20	$g_4 = g_6 \& \bar{g}_1$	g_4
23	$g_5 = g_{10} \& \bar{g}_4$	g_5
26	$g_6 = g_1 \vee g_4$	g_6
30	$g_7 = g_2 \vee g_5$	g_7

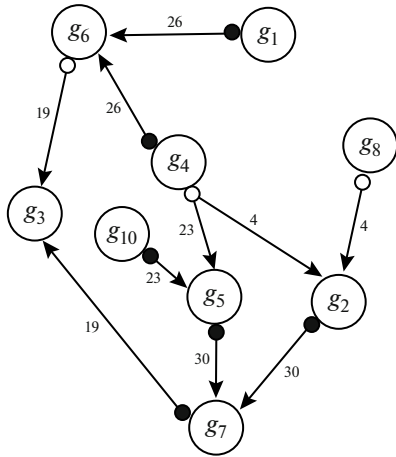


Fig. 7. Resulting acyclic orgraph prescribing the implementable cofactors.

from the graph. It is easy to see that removal of each of the pairs of edges lead to breakage of four contours.

At step A2.3 we choose the pair of edges with label 1 (Fig. 6) that are removed from the graph.

Iteration 2. At step A2.1 we find one simple cycle {g4, g6, g4} that forms two edges with labels 20, 26.

At step A2.2 we find out that removal of any of the pairs of edges with labels 20, 26 breaks the only simple cycle {g4, g6, g4}.

At step A2.3 we remove the pair of edges with label 20 and obtain the graph (Fig. 7).

Iteration 3 is not executed until the end, because at step A2.1 we find out that the graph (Fig. 7) is acyclic.

The result of execution of stage A2 is the equations for the found implementable cofactors (Table 7). In Fig. 8 we show the logic circuit corresponding to the found set of implementable cofactors; it has no feedback, because it corresponds to the oriented acyclic graph.

Table 7. Equations corresponding to the implementable cofactors

Number of equation	Equation	Cluster
4	$g_2 = \bar{g}_4 \ \& \ \bar{g}_8$	g_2
19	$g_3 = g_7 \ \& \ \bar{g}_6$	g_3
23	$g_5 = g_{10} \ \& \ \bar{g}_4$	g_5
26	$g_6 = g_1 \ \vee \ g_4$	g_6
30	$g_7 = g_2 \ \vee \ g_5$	g_7

Stage 3. Correct the original multilevel BDDI representation of Boolean function.

At Stage 3 of algorithm 1, in the functional description (Table 3) the equations of Shannon expansion for the cofactors $g_2, g_3, g_5, g_6,$ and g_7 are replaced by the formulas of algebraic expansions from Table 7, and we obtain the functional description

$$\begin{aligned}
 f_1 &= \bar{x}_1 h_1 \vee x_1 h_2; & f_2 &= \bar{x}_1 h_2 \vee x_1 h_3; & f_3 &= \bar{x}_1 h_4 \vee x_1 h_5; \\
 f_4 &= \bar{x}_1 h_4 \vee x_1 h_6; & f_5 &= \bar{x}_1 h_6 \vee x_1 h_7; \\
 h_1 &= \bar{x}_2 \bar{g}_1 \vee x_2 \bar{g}_3; & h_2 &= \bar{x}_2 g_2 \vee x_2 \bar{g}_8; \\
 h_3 &= \bar{x}_2 \bar{g}_5 \vee x_2 g_6; & h_4 &= \bar{x}_2 g_6 \vee x_2 \bar{g}_7; \\
 h_5 &= \bar{x}_2 g_6 \vee x_2 g_{10}; & h_6 &= \bar{x}_2 g_4 \vee x_2 g_9; \\
 h_7 &= \bar{x}_2 g_8 \vee x_2 g_{10}; & g_1 &= \bar{x}_4 s_2 \vee x_4 s_3; \\
 g_2 &= \bar{g}_4 \ \& \ \bar{g}_8; & g_3 &= g_7 \ \& \ \bar{g}_6; & g_4 &= \bar{x}_4 x_5 \vee x_4 s_2; \\
 g_5 &= g_{10} \ \& \ \bar{g}_4; & g_6 &= g_1 \ \vee \ g_4; & g_7 &= g_2 \ \vee \ g_5; \\
 g_8 &= \bar{x}_4 s_3 \vee x_4 \bar{s}_1; & g_9 &= \bar{x}_4 s_2 \vee x_4 \bar{x}_5; \\
 g_{10} &= \bar{x}_4 x_3 \vee x_4 x_5; & s_1 &= x_5 \bar{x}_3; \\
 s_2 &= \bar{x}_5 \bar{x}_3; & s_3 &= x_5 x_3.
 \end{aligned}
 \tag{4}$$

After formulas replacement it turns out that the equations

$$s_0 = \bar{x}_5 x_3; \quad s_4 = \bar{x}_5 \bar{x}_3 \vee x_5 x_3$$

are unused (redundant) and absent in the set of coupled formulas (4). As a result of applying the proposed method of logic optimization, in the functional BDDI description the number of literals reduces and we have obtained the results of logic synthesis best in terms of area (Table 8). Information about the used library of logic elements will be given below in description of the computational experiment.

6. ALGORITHMS 2–4 FOR FINDING ALGEBRAIC EXPANSIONS OF COFACTORS IN BDDI

To describe the differences of algorithms 2–4 from the described algorithm 1, we note that in algorithm 1 all cofactors are represented as algebraic expansions, that is, the cofactors with two, three, or four literals in their notation.

We call the formulas of Shannon expansion including just two literals *short* and the formulas consisting of three or four literals *long*. The equations of type (2), (3) that replace the formulas of Shannon expansion are referred to as the substituted equations, or *substitutions*.

Algorithm 2 differs from algorithm 1 in that, at stage 3 of the correction of the original multilevel BDDI representation, the original short equations are not

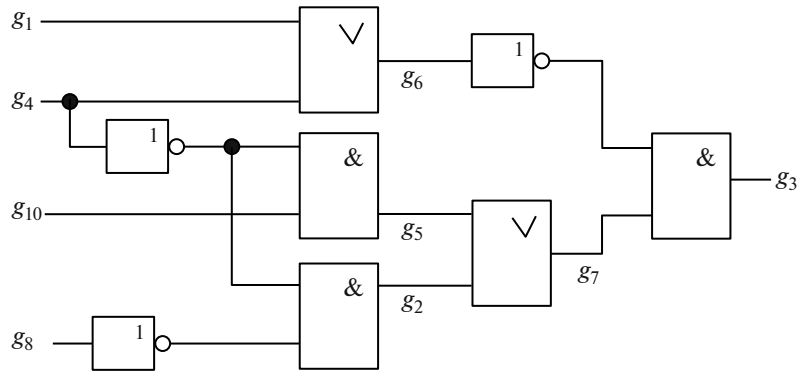


Fig. 8. Logic circuit corresponding to resulting acyclic orgraph.

replaced by the found disjunctive or conjunctive expansions.

Algorithm 3 differs from algorithm 1 in that the algebraic disjunctive and conjunctive expansions are sought only for the cofactors represented only by long equations, whereas for the sought expansions we use the cofactors represented in the original notation by both long and short equations.

Algorithm 4 differs from algorithm 3 by an auxiliary step: if in the found resulting formulas of algebraic expansions

$$g_p = \bar{g}_i \vee \bar{g}_j; \quad g_r = \bar{g}_i \& \bar{g}_j;$$

we use the two inverse cofactors, then, by the axioms of Boolean algebra (de Morgan formulas), we write equivalent substitutions in the solution, that is, we write the formulas

$$g_p = \neg(g_i \& g_j); \quad g_r = \neg(g_i \vee g_j),$$

containing a single inversion operator.

7. PROGRAM IMPLEMENTATION

The software implementation of the above presented algorithms was carried out in the C++ language within the cross-platform Qt environment [13]; here, we intensely applied the previously developed libraries for operations with Boolean objects [14] and for processing the descriptions written in the SF language [15, p. 51]. For purposes of practical use, the

general organization of the program is performed according to the rules for developing the design procedures in the FLC-2 system [16].

The program is parametrically adjustable. Its parameters are as follows:

-i <src> is the path to the original file of circuit description represented in form of BDD file, the system of logic equations written in the SF language.

-o <tar> is the path to the resulting file optimized by the number of literals.

-r <config> is the path to the configuration file, the file of INI format of the Windows system containing the sections with the collection of key parameters establishing the modes of program operation:

```
[BDD_OPT]
. . .
inversion=2
donothandleshort=2
. . .
[DATA]
RPT_Name=e:/FLC2/workDir/protokol.txt
```

Here, the parameter `inversion` can take on the values from the set $\{0,1,2\}$: 0 means that we do not use the inverse functions for selection of optimizing substitutions; 1 is the equivalent use of both direct and inverse functions; 2 stands for that, in addition to use of inverse functions, we transit to the inverse representation for substitutions:

$$g_p = \bar{g}_i \vee \bar{g}_j; \rightarrow g_p = \neg(g_i \& g_j);$$

Table 8. Results of logic synthesis

Functional description	Number of literals	Logic circuit		
		number of logic elements	area S_{ASIC}	delay τ , ns
Original (formulas, Table 2)	100	45	16031	2.89
Result (formulas (4))	84	37	13855	3.36

Table 9. Parameters of original systems of functions and BDDI

Circuit	DNF system			BDDI		
	n	m	k	number S_{BDDI} of equations	number of pairs of mutually inverse cofactors	total number of literals P
ADD6	12	7	1 092	27	10	91
ADDM4	9	8	512	66	41	590
ADR4	8	5	256	17	6	57
ALU1	12	8	19	16	0	50
B12	15	9	431	66	3	209
B9	16	5	123	73	14	253
BR1	12	8	34	119	4	319
BR2	12	8	35	85	2	216
DC2	8	7	58	58	9	179
DIST	8	5	256	115	32	424
EX7	16	5	123	73	14	253
F51M	8	8	256	37	13	124
GARY	15	11	214	317	17	964
IN0	15	11	138	317	17	964
IN1	16	17	110	756	20	2552
IN2	19	10	137	261	25	861
INTB	15	7	664	681	105	2465
LIFE	9	1	512	36	5	133
LOG8MOD	8	5	47	62	10	215
M1	6	12	32	48	7	134
M181	15	9	430	67	3	212
M2	8	16	96	116	28	362
M3	8	16	128	130	30	413
M4	8	16	256	175	46	604
MAX1024	10	6	1 024	333	118	1242
MAX46	9	1	46	72	0	248
MAX512	9	6	512	186	56	686
MLP4	8	8	256	147	37	528
MP2D	14	14	123	78	2	219
NEWAPLA	12	10	17	44	4	112
NEWAPLA1	12	7	10	24	0	55
NEWCPA1	9	16	38	80	8	212
NEWILL	8	1	8	15	0	47
NEWTAG	8	1	8	8	0	23
NEWTPLA	15	5	23	56	4	147
P82	5	14	24	53	5	162

Table 9. (Contd.)

Circuit	DNF system			BDDI		
	n	m	k	number S_{BDDI} of equations	number of pairs of mutually inverse cofactors	total number of literals P
RADD	8	5	120	17	6	57
RD53	5	3	32	15	5	53
RD73	7	3	147	29	9	107
ROOT	8	5	256	56	15	175
SEX	9	14	23	51	0	142
SQN	7	3	96	46	5	161
SQR6	6	12	64	62	11	199
T3	12	8	152	87	5	248
TIAL	14	8	640	582	126	2 099
VTX1	27	6	110	100	2	298
X9DN	27	7	120	102	2	305
Z4	7	4	128	15	5	51
Z5XP1	7	10	128	40	11	131

$$g_r = \bar{g}_i \vee \bar{g}_j; \rightarrow g_r = \neg(g_i \& g_j).$$

The parameter `donothandleshort` adjusts the rules of consideration of the equation from the aspect of searching the substitution of cofactors and can also take on the values from the set $\{0, 1, 2\}$: 0 means that all equations of the considered system will undergo testing on replacement, 1 stands for dropout the possible substitutions capable of reducing the number of literals in the equation, which will be performed at the stage of generation of the resulting file; 2 is for dropout at the stage of searching the substitutions.

The parameter `RPT_Name` from the `DATA` section determines the path to the file of session protocol containing information about errors in the accepted initial data.

The considered program has several limitations. The fixed ones among them are the following: the number n of arguments of the implemented system of Boolean functions $n \leq 27$; the number of possible representations of an individual cofactor (no more than 96 variants); and the number of edges of the graph G (no more than 30000). However, achieving these values was not observed in practice because of the required amount of performed calculations. We had the cases of normal operation of the program for dimensions of the adjacency list of graph description of more than 15000 elements (edges of the graph) and

approximately 90 steps of execution of the algorithm for transforming the orgraph to the acyclic form.

8. COMPUTATIONAL EXPERIMENT

We conducted the experiment on testing the efficiency of the programs implementing the proposed algorithms 1–4 for reduction in the area and increase in the performance of the combination logic units implemented as part of custom digital VLSIs (ASIC).

The initial descriptions of the first collection of examples of combination logic were the DNF systems of Boolean functions written in the SF language [15, p. 53] borrowed from library [17] of circuit examples presented in the PLA format; here, the descriptions were converted from the PLA format to the SF format of the FLC-2 system of logic optimization [16].

The second collection of examples was the systems of Boolean functions prescribing the SF descriptions of truth tables of modular adders [18].

For each of examples of the systems of completely specified Boolean functions, we carried out BDDI minimization using the `BDD_Builder` program [9], also included in the FLC-2. The obtained circuit implementations were called reference for the experiment. The BDDI descriptions and the additionally minimized BDDI descriptions (additional optimization was carried out by means of programs implementing algorithms 1–4 of algebraic expansion of cofactors

Table 10. Results of experiment for the first collection of examples

Circuit	Original BDDIs (reference solutions)		Algorithm 1		Algorithm 2		Algorithm 3		Algorithm 4	
	S_{ASIC}	τ , ns	S_{ASIC}	τ , ns	S_{ASIC}	τ , ns	S_{ASIC}	τ , ns	S_{ASIC}	τ , ns
ADD6	12806	8.03	*11952	*7.72	12806	8.03	12583	9.05	12583	9.05
ADDM4	80782	7.84	79643	8.01	*73974	8.17	78672	11.35	81602	*7.03
ADR4	8074	4.90	*7589	*4.82	8074	4.90	*7851	5.60	*7851	5.06
ALU1	7109	1.12	7109	1.12	7109	1.12	7109	1.12	7109	1.12
B12	18358	3.59	21327	3.77	21293	*3.46	21293	*3.46	18492	*3.40
B9	26081	4.91	26204	*3.89	26081	4.91	*25701	4.65	25701	4.65
BR1	23843	6.36	27867	*5.65	24530	*5.06	26193	*5.46	28547	*5.26
BR2	21371	6.34	21790	*5.57	*20032	*4.43	*20032	*4.43	21371	6.34
DC2	23302	4.28	22845	5.34	20780	4.88	*19976	4.30	*19067	4.73
DIST	60085	6.08	55627	8.34	*54159	6.74	56062	7.14	54807	*6.01
EX7	26081	4.91	26204	*3.89	26081	4.91	*25701	*4.65	*25701	*4.65
F51M	18353	7.67	17622	*5.39	*16824	*5.45	17298	*3.21	*16037	*3.54
GARY	94648	6.74	107264	7.14	96032	6.81	101299	*6.68	*93978	6.43
IN0	94648	6.74	107264	7.14	96032	6.81	98409	*6.50	*93978	6.43
IN1	192655	8.26	235638	14.12	216493	10.05	227262	13.16	221805	11.62
IN2	75414	6.73	71619	11.23	*69471	9.13	70102	10.01	*73851	10.75
INTB	272555	8.67	*241168	13.90	*237758	13.26	*248561	13.50	*248316	15.73
LIFE	18146	4.60	*15256	7.44	*16450	7.26	*15150	7.36	*17359	7.37
LOG8MOD	24022	4.26	25160	*4.05	24440	4.89	24161	4.48	24457	4.76
M1	15312	3.46	17605	*2.63	16121	*2.63	16857	*2.47	16037	*2.77
M181	18849	3.48	20356	3.73	20914	*3.42	20914	*3.42	*18576	3.75
M2	45086	5.20	*45058	*4.87	*43362	5.39	*42776	*5.19	*42787	*5.07
M3	52580	4.49	54656	6.62	*52435	4.92	*51754	5.81	*49947	5.33
M4	78181	5.87	78405	*5.27	78879	6.19	*77473	6.18	*76502	*4.84
MAX1024	146888	7.18	153606	11.02	150777	10.30	148567	10.50	*144076	9.79
MAX46	36125	4.89	*34892	5.61	*32035	4.93	*35344	4.89	*35344	4.89
MAX512	84643	5.98	*84537	7.29	*79827	7.72	*81228	6.29	*79582	6.46
MLP4	68439	5.60	70224	6.28	*66586	6.70	68628	9.17	*68210	6.98
MP2D	17471	3.56	20239	6.15	19351	3.48	18771	4.37	17767	3.61
NEWAPLA	11087	3.84	*10184	3.90	11087	3.84	*10184	3.90	11087	3.84
NEWAPLA1	6869	3.35	*6702	3.63	6869	3.35	6869	3.35	6869	3.35
NEWCPLA1	20585	3.87	22158	4.53	20596	4.58	*19206	4.02	22951	5.14
NEWILL	5122	3.22	5736	4.76	5736	4.76	5736	4.76	5122	3.22
NEWTAG	2126	1.90	2126	1.90	2126	1.90	2126	1.90	2126	1.90
NEWTPLA	11316	3.30	13515	3.85	11316	3.30	13264	4.04	11316	3.30
P82	19988	2.93	22292	3.23	*19965	*2.75	19982	*2.76	*19982	*2.76

Table 10. (Contd.)

Circuit	Original BDDIs (reference solutions)		Algorithm 1		Algorithm 2		Algorithm 3		Algorithm 4	
	S_{ASIC}	τ , ns	S_{ASIC}	τ , ns	S_{ASIC}	τ , ns	S_{ASIC}	τ , ns	S_{ASIC}	τ , ns
RADD	8074	4.90	*7589	*4.82	8074	4.90	7851	5.60	*7851	5.60
RD53	7321	3.38	7779	*3.22	7321	3.38	7321	3.38	7321	3.38
RD73	18090	4.55	*14212	*4.49	15222	5.30	15669	4.61	*15669	4.61
ROOT	26109	4.78	26455	*4.68	*24580	*4.23	*22532	*4.08	*23581	5.30
SEX	12566	3.69	15524	*3.56	12605	*2.08	12605	*2.08	12605	*2.08
SQN	22303	3.33	*20077	4.87	*19642	5.04	*19245	*3.10	*18922	5.57
SQR6	28737	5.72	*28157	*4.55	*28023	*4.33	*27827	*3.77	*27035	*3.72
T3	17276	3.59	19268	5.85	17454	6.03	17454	6.03	17917	4.16
TIAL	255531	8.75	*221509	15.25	*212286	12.65	*209217	12.15	*211633	14.72
VTX1	26996	5.96	*25930	6.40	*24223	*5.94	*24223	*5.94	*24552	6.16
X9DN	26996	6.00	25701	*5.81	*25155	*5.84	*25155	*5.84	*26812	6.65
Z4	6640	4.25	*6339	4.32	6640	4.25	6417	4.71	*6417	4.71
Z5XP1	18442	4.64	*18252	4.30	18442	4.53	*16400	*3.61	*16400	*3.61
Number of the best solutions (bold face)	16	22	9	12	14	11	15	14	18	16
Number of improvements of reference solutions (number of asterisks)			17	18	19	12	21	18	28	13

proposed in this work) were converted to VHDL descriptions and fed to the input of the LeonardoSpectrum synthesizer [19, p. 241]. For each of the examples, the logic circuit synthesis was conducted with the same options of synthesis control and for the same target synthesis library. The target library was the library of designing custom digital CMOS VLSIs; the content of the library is presented in [20].

The parameters of the systems of functions and BDDI descriptions for the first collection of examples are given in Table 9. We established that the GARY and IN0 examples prescribe the same system of functions in form of different DNF systems, that is, DNFs with different sets of elementary conjunctions. For the second collection of examples, the label of example Mod_ i corresponds to the adder modulo i .

Below, in tables presenting the results of our experiments, we use the following denotations for the parameters of the systems of functions $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_m(\mathbf{x}))$, $\mathbf{x} = (x_1, \dots, x_n)$: n is the number of variables x_1, \dots, x_n ; m is the number of functions; k is the number of common elementary conjunctions entering the DNFs of all component functions $f_j(\mathbf{x}), j = 1, \dots, m$.

The resulting logic circuits were evaluated by two parameters: S_{ASIC} is the total area of all elements of the logic circuit and τ is the circuit delay (in ns). The results of our experiment are listed in Tables 10–12. The best solutions (the circuits of smaller area or the circuits with shorter delay) are labeled by bold face, and asterisk is used to denote the solutions that improve the initial reference solutions, which, as we have already said, are the logic circuits constructed by the initial BDDIs. For each of algorithms 1–4, Table 11 presents the numbers of eliminated (redundant) equations after execution of the corresponding algorithm. The LeonardoSpectrum synthesizer outputs this information if at synthesis there are unused signals revealed in the functional descriptions by which we perform circuit synthesis. If, for any example for each of algorithms 1–4, the number of eliminated equations is zero, then information on such example is absent. For instance, we do not provide the information for the first example Add6 in Table 11, because there are no redundant equations for any of the algorithms.

Table 11. Results of experiment for the first collection of examples

Circuit	Number of eliminated equations		
	Algorithm 1	Algorithm 2	Algorithms 3, 4
ADDM4	1	1	1
B12	1	0	0
BR1	4	3	2
BR2	6	2	2
DC2	4	2	1
DIST	5	3	3
GARY	14	6	6
IN0	14	6	6
IN1	73	60	57
IN2	22	20	20
INTB	90	93	96
LIFE	4	4	4
M1	5	2	2
M181	1	0	0
M3	2	0	0
M4	2	2	2
MAX1024	3	3	5
MAX512	1	1	2
MLP4	1	1	1
MP2D	2	2	2
NEWAPLA	2	0	0
NEWCPLA1	2	3	0
P82	2	0	0
RD73	1	0	0
SEX	3	1	3
SQN	1	1	1
T3	11	7	7
TIAL	59	75	75
VTX1	2	2	2
X9DN	2	2	2
Z5XP1	1	0	0

Let us analyze the experimental results. The first collection of examples consisted of 59 functional descriptions of circuits. For 10 examples of low-dimensional circuits—the circuits CLPL, CO14, NEWAPLA2, NEWBYTE, NEWCOND, NEWTPLA1, NEWTPLA2, RYY6, SYM10, and Z9SYM—application of algorithms 1–4 did not vary the refer-

ence solution; therefore, the initial data and the results of computational experiment for these examples are not provided in Tables 9–11.

For the three examples B12, LOG8MOD, and MP2D, application of any of algorithms 1–4 for changing the functional reference BDDI descriptions led to an insignificant increase in the area of the corresponding logic circuit. This is related with the fact that the LeonardoSpectrum synthesizer has its own built-in tools of logic optimization and technological mapping. Technological mapping consists in covering the optimized logic equations by functional descriptions of logic elements entering the design library. Variation in the functional description led to other results of technological mapping. For 33 examples of circuits, application of the proposed algorithms allowed improving their area. Reduction in the area for high-dimensional circuits can be of practical significance: for the INTB circuit the area reduction is 13%, for the Tial circuit it is 18%. In general, algorithms 3 and 4 are more efficient compared to algorithms 1 and 2. Algorithm 4 allows improving the reference solutions (by area) for 27 circuits, and algorithm 3 allows increasing the performance for 18 circuits. Algorithms 3 and 4 can be recommended for practical use in the first wave. It is also not excluded that application of algorithms 1 and 2 can also improve the reference solution.

For the second collection of examples (modular adders), application of algorithms allowed reducing the areas for nine circuits and decreasing the delays for five circuits of 14 ones. The area of the Mod_61 circuit reduced by 28%, and the area of the Mod_37 circuit decreased by 23% (Table 12).

The program experiment also showed that, for reduction in the circuit area, always more efficient are disjunctive and conjunctive expansions of cofactors with the use of their inversions, compared to similar expansions without inversions of cofactors.

9. CONCLUSIONS

Our experiments [9] showed that determination of pairs of mutually inverse cofactors at BDDI optimization considerably improves the results of subsequent synthesis, compared to the BDD optimization executed without finding the inverse cofactors.

In the paper we described software-implemented algorithms of additional minimization of algebraic multilevel BDDI representations of systems of completely specified functions on the basis of disjunctive and conjunctive representations of subfunctions appearing at one BDDI level. Such additional logic optimization by the number of literals in the resulting logic equations prescribing the BDDIs leads to sim-

Table 12. Results of experiment for the second collection of examples

Name	Original BDDIs (reference solutions)		Algorithm 1		Algorithm 2		Algorithm 3		Algorithm 4	
	S_{ASIC}	τ , ns	S_{ASIC}	τ , ns	S_{ASIC}	τ , ns	S_{ASIC}	τ , ns	S_{ASIC}	τ , ns
Mod_5	13520	2.68	*12923	2.86	13520	2.68	13593	3.88	13520	2.68
Mod_7	16946	3.51	20696	4.50	16946	3.51	19469	*3.39	18777	4.00
Mod_9	32358	6.21	*30640	*5.64	32358	6.21	*30082	6.44	*30679	*5.01
Mod_15	41080	5.83	44768	7.00	41080	5.83	44205	6.93	44428	7.34
Mod_17	54137	6.02	57697	7.62	54137	6.02	*52703	*5.98	*48462	8.15
Mod_19	49483	6.93	58311	9.00	49483	9.93	51593	7.80	*48752	*5.95
Mod_23	58746	6.75	*58171	*6.17	58746	6.75	60822	7.81	62624	8.75
Mod_25	68043	6.60	75492	9.00	68043	6.60	75570	8.53	73913	8.35
Mod_27	78600	6.77	92422	10.28	78600	6.77	85240	8.96	87059	9.24
Mod_29	77902	6.51	*77149	10.02	77902	6.51	*72919	10.39	*71469	11.68
Mod_31	73003	7.68	*72652	8.96	73003	7.68	*72864	8.08	*69795	8.77
Mod_37	113352	6.60	*94296	9.04	113352	6.60	*87483	9.89	*91395	11.18
Mod_59	149628	9.35	159030	16.62	149628	9.35	150264	15.73	151542	13.57
Mod_61	129863	8.54	*102466	13.36	129863	8.54	*100261	13.83	*93549	18.04
Number of the best solutions (bold face)	5	10	2	1	4	8	2	2	5	3
Number of improvements of reference solutions (number of asterisks)			7	2	0	0	6	2	7	2

pler functional descriptions by which logic circuit synthesis is carried out. Decrease in the circuit area, reduced to the decrease in the number of transistors in circuits, allows also reducing the energy consumption of circuits. However, reduction in the number of literals at algebraic expansions that allow frequently decreasing the area of circuits made of library CMOS elements can lead to both reduction and increase in the time delays of circuits.

CONFLICT OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

1. Brayton, R.K., Hachtel, G.D., and Sangiovanni-Vincentelli, A.L., Synthesis of multi-level combinational logic circuits, *Tr. Inst. Inzh. Elektron. Radiotekh.*, 1990, vol. 78, no. 2, pp. 38–83.
2. Meinel, C. and Theobald, T., *Algorithms and Data Structures in VLSI Design: OBDD – Foundations and Applications*, Berlin, Heidelberg: Springer-Verlag, 1998.
3. Knuth, D.E., Combinatorial algorithms, in *The Art of Computer Programming*, Pearson Education, Inc., 2011, vol. 4A.
4. Yang, S. and Ciesielski, M., BDS: a BDD-based logic optimization system, *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 2002, vol. 21, no. 7, pp. 866–876.
5. Ebendt, R., Fey, G., and Drechsler, R., *Advanced BDD Optimization*, Springer, 2005.
6. Bibilo, P.N., *Primenenie diagramm dvoichnogo vybora pri sinteze logicheskikh skhem* (Binary Decision Diagrams for Synthesizing Logic Circuits), Minsk: Belaruskaya navuka, 2014.
7. Kubica, M. and Kania, D., SMTBDD: new form of BDD for logic synthesis, *Int. J. Electron. Telecommun.*, 2016, vol. 62, no. 1, pp. 33–41.
8. Bibilo, P.N. and Romanov, V.I., Minimization of binary decision diagrams for systems of completely defined Boolean functions by using Shannon expansions and algebraic representations of cofactors, *Informatika*, 2021, vol. 18, no. 2, pp. 7–32.
9. Bibilo, P.N. and Lankevich, Yu.Yu., Zhegalkin polynomials for minimizing multilevel representations of Boolean functions systems based on the Shannon expansion, *Programm. Inzheneriya*, 2017, vol. 8, no. 8, pp. 369–384.
10. Search for elementary cycles in a graph. <https://vscode.ru/prog-lessons/poisk-elementarnyih-tsiklov-v-grafe.html>. Accessed 05.082021.

11. Johnson, D.B., Finding all the elementary circuits of a directed graph, *SIAM J. Comput.*, 1975, vol. 4, no. 1, pp. 77–84.
12. Toropov, N.R., Multi-level combinational network transformation into a two-level one, in *Logicheskoe proektirovanie* (Logic Design), Minsk: United Institute of Informatics Problems of the National Academy of Sciences of Belarus, 2000, issue 5, pp. 4–14.
13. Shlee, M., *Qt 5.10. Professional Programming in C++*, St. Petersburg: BKhV-Peterburg, 2018.
14. Romanov, V.I., Software tools for solving logic-combinatorial problems, *Informatika*, 2005, no. 4, pp. 114–123.
15. Bibilo, P.N. and Romanov, V.I., *Logicheskoe proektirovanie diskretnykh ustroystv s ispol'zovaniem produktsionno-freimovoi modeli predstavleniya znaniy* (Logical Design of Discrete Devices by Using a Production-Frame Knowledge Representation Model), Minsk: Belaruskaya navuka, 2011.
16. Bibilo, P.N. and Romanov, V.I., The system of logical optimization of functional and structural descriptions for digital devices based on production-frame knowledge representation model, in *Problemy razrabotki perspektivnykh mikro- i nanoelektronnykh sistem. – 2020. sb. trudov* (Promising Micro- and Nano-Electronic Systems: Design Problems. Collection of Scientific Papers 2020), Stempkovskii, A.L., Ed., Moscow: Institute for Design Problems in Microelectronics RAS, 2020, no. 4.
17. The Tests in the Monograph «Logic Minimization Algorithms for VLSI Synthesis». <http://www1.cs.columbia.edu/~cs6861/sis/espresso-examples/ex>. Accessed 20.11.2020.
18. Balaka, E.S., Tel'pukhov, D.V., Osinin, I.P., and Gorodetskii, D.A., Comparative study and analysis of hardware implementation methods for adders in absolute values, *7Universum: Tekh. Nauki: Elektron. Nauch. Zh.*, 2016, no. 1 (23). <http://7universum.com/ru/tech/archive/item/2887>.
19. Bibilo, P.N., *Sistemy proektirovaniya integral'nykh skhem na osnove yazyka VHDL. StateCAD, ModelSim, LeonardoSpectrum* (Integrated Circuit Design Systems Based on the VHDL Language: StateCAD, ModelSim, LeonardoSpectrum), Moscow: SOLON-Press, 2005.
20. Avdeev, N.A. and Bibilo, P.N., Automated Design for Digital Operational Units with Low Power Consumption, *Programm. Inzheneriya*, 2021, no. 2, pp. 63–73.

Translated by E. Oborin