This paper reports a study that has established the possibility of reducing computational complexity while improving the productivity of simplification of Boolean functions in the class of perfect implied normal forms (PINF-1 and PINF-2) using a method of figurative transformations.

The method of figurative transformations has been expanded to cover the process of simplifying the functions of the implicative basis by using the developed algebra of the implicative basis in the form of rules that simplify the PINF-1 and PINF-2 functions of the implicative basis. A special feature in simplifying the functions of the implicative basis on the binary structures of 2-(n, b)-designs) is the use of analogs of perfect disjunctive normal forms (PDNF) and perfect conjunctive normal forms (PCNF) of Boolean functions. The specified forms of the functions define transformation rules for the functions of the implicative basis on binary structures.

It is shown that the perfect implicative normal form of n-place function of the implicative basis can be represented by the binary sets or a matrix. Logical operations over the structure of the matrix ensure the result from simplifying the functions of the implicative basis. This makes it possible to focus the minimization principle within the truth table of the assigned function and avoid auxiliary objects such as Carnot map, Weich charts, etc.

The method under consideration makes it possible:

– to reduce the algorithmic complexity of PINF-1 and PINF-2 simplification;

– to improve the performance of simplifying the functions of the implied basis by 100–200 %;

– to visualize the process of PINF-1 or PINF-2 minimization.

There is reason to argue that minimizing the functions of the implicative basis using a method of figurative transformations brings the task of PINF-1 and PINF-2 minimization to the level of well-researched problems within the class of disjunctive-conjunctive normal forms of Boolean functions

Keywords: method of figurative transformations, minimization of functions of the implicative basis, implication function, PINF-1, PINF-2

UDC 681.325
DOI: 10.15587/1729-4061.2020.220094

DEVISING A METHOD OF FIGURATIVE TRANSFORMATIONS FOR MINIMIZING BOOLEAN FUNCTIONS IN THE IMPLICATIVE BASIS

M. Solomko
PhD, Associate Professor
Department of Computer Engineering
National University of Water and Environmental Engineering
Soborna str., 11, Rivne, Ukraine, 33028
E-mail: doctrinas@ukr.net
Iu. Batyshkina
PhD, Associate Professor*
E-mail: yuliia.batyshkina@rshu.edu.ua
I. Voitovych
Doctor of Pedagogical Sciences, Professor*
E-mail: ihor.voitovych@rshu.edu.ua
L. Zubyk
PhD, Associate Professor
Department of Software Systems and Technologies
Taras Shevchenko National University of Kyiv
Volodymyrska str., 60, Kyiv, Ukraine, 01033
E-mail: labrob@ukr.net
S. Babych
PhD, Associate Professor*
E-mail: stepaniia.babych@rshu.edu.ua
K. Muzychuk
PhD, Associate Professor*
E-mail: kateryna.muzychuk@rshu.edu.ua
*Department of Information and Communication Technologies and Methods of Teaching Informatics
Rivne State University of Humanities
St. Bandery str., 12, Rivne, Ukraine, 33028

Received date 09.10.2020
Accepted date 20.11.2020
Published date 25.12.2020

1. Introduction

The technology of designing Boolean functions included in the logical basis can resort to the implementation based on certain physical phenomena. For example, the properties of semiconductors are matched by the Pierce (Webb) and Schaeffer functions while magnetic phenomena could be employed to realize the implicative basis.

Implication (from Latin *implico* – closely related) is a logical operation that corresponds to the relation «if..., then...» (IF-THEN) when two statements A and B form a conditional statement «if A, then B». Implication often denotes the conditional statement itself, as well as its formalized analogs, such as logical computing formulae, which contain an implication sign (for example, «$\supset$», or «$\rightarrow$») and take the form of $A \supset B$ ($A \rightarrow B$), where A and B are the formulae for logical computation. The implication operation is used to describe linguistic patterns using the algebra of predicates, as well as to record the rules of formal grammar [1].

The logical function $f=x_1 \rightarrow x_2$ (direct implication of x_1 to x_2) is a disjunctive –

$$f = x_1 \to x_2 = \overline{x_1} + x_2, \quad (1)$$

therefore, a value of the function «false» is derived only when the x_1 argument accepts the value «true», and the x_2 argument takes the value «false» (Table 1).

Table 1

Truth table of function $f = x_1 \to x_2$

x_1	x_2	$\overline{x_1}$	$f = x_1 \to x_2 = \overline{x_1} + x_2$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	0	1

The logical scheme that implements function (1) is shown in Fig. 1.

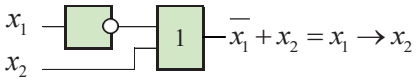

Fig. 1. A logical scheme implementing the function of direct implication $f = x_1 \to x_2$

Implication function (1) can take an algebraic form:

$$f(x_1, x_2) = \begin{cases} 1, & \text{if } x_1 \le x_2, \\ 0, & \text{if } x_1 > x_2. \end{cases} \quad (2)$$

On the sets of variables at which function (2) returns «1», the value of bits in the column «x_1» either matches the value of the bits in the column «x_2», or is less than it (Table 2).

Table 2

Truth table on the sets of variables where the function $f = x_1 \to x_2$ returns «1»

x_1	x_2	$\overline{x_1}$	$f = x_1 \to x_2 = \overline{x_1} + x_2$
0	0	1	1
0	1	1	1
1	1	0	1

This relationship between the bits in the columns «x_1» and «x_2» means that the column «x_1» is an integral part of the column «x_2». Thus, the term «implicates» means «is a part of» (x_1 is a part of x_2) [2].

Minimal logical bases involving the implication are {→, NOT}, {→, 0}, {→, ⊕ }, {→, ←}. Note that {←, NOT}, {←, 1} are also the bases.

The functions of the canonical basis {NOT, OR, AND} are represented by implication as follows:

$$\overline{x} = x \to 0; \quad (3)$$

$$x_1 + x_2 = \overline{x_1} \to x_2; \quad (4)$$

or

$$x_1 + x_2 = \overline{x_1} \to x_2 = (x_1 \to 0) \to x_2;$$

$$x_1 \cdot x_2 = \overline{\overline{x_1} + \overline{x_2}} = \overline{x_1 \to \overline{x_2}}; \quad (5)$$

or

$$x_1 \cdot x_2 = \overline{\overline{x_1} + \overline{x_2}} = \overline{x_1 \to \overline{x_2}} = \overline{x_1 \to (x_2 \to 0)} = \\ = \big(x_1 \to (x_2 \to 0)\big) \to 0.$$

Implication provides for a functionally complete basis – each Boolean function can be implemented by pairing the elements NOT, or NOT-OR (Fig. 2).

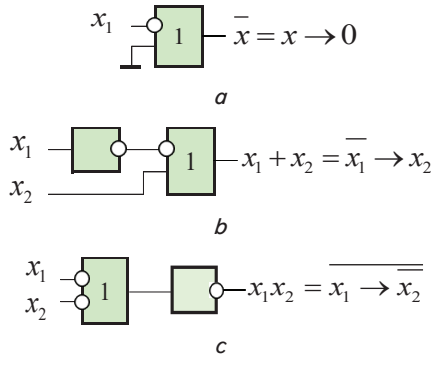

Fig. 2. Implementation of the elements of canonical basis {NOT, OR, AND} on the elements NOY, NOT-OR: *a* – inverter, *b* – disjunction, *c* – conjunction

The functional completeness of the switching function system ensures the possibility to represent an arbitrary functional dependence on the assigned number of arguments by using the minimum number of basic functions (operations). These functions (operations) collectively have the property of functional completeness, and, therefore, possess the ability to synthesize a combination scheme that reproduces the functional dependence by employing a minimum number of the types of logical elements. However, that does not resolve the issue of an optimal combination scheme. As demonstrated by the practical design of logical schemes, combining elemental bases belonging to several functionally complete systems (for example, {OR-NOT}, {AND-NOT}, {AND, OR, NOT} systems) makes it possible to build optimal combination schemes (in terms of hardware complexity and performance). Using the elemental basis of only one functionally complete switching function system does not ensure, in a general case, deriving an optimal combination scheme [2].

The process of minimizing logical functions occupies an important position within the technology of designing digital components. In this regard, it is still an actual task to ensure the adequate compliance of a developed product with the specified cost specifications, the simplification and the warranty of obtaining the optimal result from minimizing different representations of logical functions.

Since the implicative basis belongs to the field of logical function optimization [3], it is a relevant task to undertake research to improve, in particular, the following aspects:

– methods to simplify the functions of the implicative basis;

– the minimization of logical schemes based on the implication functions;

– the reliability of an optimal result from minimizing the implicative basis.

2. Literature review and problem statement

The logical shutter IMPLY and a memristor-based logical scheme are described in work [4]. Memristor devices can be

used as logical schemes. In this logical memory structure, each memristor is used as a computational logical element or as a trigger at different stages of the computational process. The logical state of the device is determined by the memristor resistance. The specified structure can be integrated into the memristor crossbar, which is commonly used for memory. Paper [4] reports a procedure of its design. Based on the methodology reviewed, designing an 8-bit full binary code adder is presented as a case study.

Memristors change their resistance under the control of voltage and can retain their value after a voltage discharge. The small size of memristors makes them useful for designing supercompact memory systems. Memristors execute logical primitives and can, therefore, be used to implement logical functions employing different logical design styles. Article [5] also examines the technology of making memristors, model schemes, the methods of logical function implementation, as well as various computational procedures, which can be used in computational components.

A method to generate the sequences of control signals, which makes it possible to calculate an arbitrary n-input 1-output Boolean function using only two working memristors, is reported in work [6]. This approach is based on the use of a recursive Boolean formula that provides a path for implementing a Boolean function over a functionally complete base {imply, FALSE}, where *imply* is the two-input Boolean implication function $x \to y = \bar{x} + y$.

Such characteristics of the memristor as high speed, low power, and passive memory preservation make it suitable for use in several areas (neural systems, digital and analog circuits, memory blocks). The memristor has different properties that allow for a number of industries to apply it, taking advantage of the desired advantages. The memristor «Implication», which implements the IMPLY logic, enables all logical operations that are possible in structures composed only of memristors. The focus of paper [7] is the full adder of binary codes whose logical structure is based on memristors and which uses only implicative logic. Article [7] suggests an algorithm that ensures fewer steps to perform the adder logic, as well as fewer memristors.

Any logical operation can be implemented in the implicative memristor schemes characterized by low energy consumption and the size at the nanometer level. Paper [8] reports a method for converting And-Inverter Graphs (AIGs) for logical functions in the network based on implication. The optimized copying process is used to reduce the latency and area of memory chains. The experimental studies involved a set of tests, which includes 33 functions with input variables ranging from 3 to 41. The experimental results are compared with the results from the original algorithm and another graph-based representation method (MIG). It is demonstrated that the improved algorithm can produce better delay rates for most test set functions.

The logic of implication is one of the main technologies for memristors. Work [9] reports an optimal design of a complete adder of binary codes based on the memristor using implicative logic. The design considered requires 27 memristors and less area compared to typical 8-bit complete adders based on CMOS technology. The authors also described a complete adder that requires only 184 computational steps. It is noted that the adder performance increased by 20 %.

The physical implementation of memristors or memristor devices that combine the electrical properties of the memory element and resistor was first introduced by Leon Chua in 1971. Such devices are characterized by one or more state variables that determine the resistance of the switch depending on its voltage history. Study [10] shows that this family of nonlinear devices with a dynamic memory can also be used for logical operations. It is demonstrated that the devices examined can perform material implication (IMP), which is a fundamental logical operation of Boolean logic over two variables p and q, such that p IMP q is equivalent to (NOT p) OR q. Thus, when included in the appropriate scheme, memory switches can execute logical operations corresponding to the state for which the same devices simultaneously perform shutter functions (logic) and gates functions (memory) that use resistance instead of voltage or charge as a variable of the physical condition.

The quantity of memristors needed to execute the assigned logical operation is discussed in work [11]. It is demonstrated that memristors are naturally suitable for performing the logic of implication, instead of Boolean logic. It is also noted that the memristor can be used as a logical shutter and trigger. While functionally complete, the logic of implication can be used to calculate any Boolean function. However, performing logical implication with data-containing devices requires additional memristors to store intermediate results. Study [11] reports an effective technique for calculating any with a large number of memristors. The length of the corresponding computational sequence is also taken into consideration.

New methods of logical synthesis for incomplete multilevel binary chains using memristor-based implication elements are considered in work [12]. The first method checks the assumption of the use of only two working memristors. An algorithm minimizes the number of implicative (IMPLY) gateways, which corresponds to minimizing the number of pulses or delay time. The first method is tested with other synthesis techniques such as modified SOP and Exclusive-Or Sum of Products (ESOP) with a minimum number of working memristors. The authors analyzed the task of reducing the number of IMPLY gateways by adding more operating memristors. Sequence diagrams and a new designation, similar to that used in inverse logic, have been implemented.

Paper [13] demonstrated that all Boolean functions can be computed by using two memristors. This requires a recursive connective form to introduce a Boolean function. The procedure for the synthesis of the corresponding computational sequence is also presented. The result is important for minimizing complex logical chains regarding the number of memristors used.

The literary sources considered above [4–13] represent the implementation of the fourth basic element of scheme equipment – the memristor, which is an addition to the resistor, capacitor, and inductiveness.

At the suggestion by Leon Chua [14], there is a fourth basic element of electrical chains – along with inductivity, capacitor, and resistor, which should link the charge to changes in the magnetic field by the following ratio:

$$d\varphi = Mdq.$$

It was shown [14] that when the memristivity M is a constant magnitude, then the memristor performs like a common resistor. However, if the memristivity M is the function of charge q, the correlation between the voltage on the memristor terminals and the charge that passed through the element is determined from the following formula:

$$v(t) = M(q)i(t) = M\left(\int_{-\infty}^{t} i(\tau)d\tau\right)i(t).$$

At each point in time, the behavior of the memristor is similar to that of the resistor, whose actual resistance value depends on the time integral of the current that passes the device. The history of device operation determines its properties at every particular point in time. Thus, the term «memristor» means «a resistor with memory».

According to researchers from Hewlett-Packard, memristors are most effective when using logic based on an implication operation [10].

Parallel connection of two memristors implements a material implication function [15]. Along with the universal elements AND-NOT and OR-NOT, the implication function, together with the constant zero function, creates a functionally complete basis (Table 3) [16].

This makes it possible to perform all 16 switching functions of two variables. However, until now, such a basis has not been used in computing equipment [16].

Table 3

Computational versatility of IMP (implication) and FALSE operations: 16 binary Boolean operations over two logical quantities

Operation	Truth table				Equivalent operation
p	1	1	0	0	$=p$
q	1	0	1	0	$=q$
TRUE	1	1	1	1	$=p$ IMP p
p OR q	1	1	1	0	$=(p$ IMP 0) IMP q
q IMP p	1	1	0	1	$=q$ IMP p
p	1	1	0	0	$=(p$ IMP 0) IMP 0
p IMP q	1	0	1	1	$=p$ IMP q
q	1	0	1	0	$=(q$ IMP 0) IMP 0
p EQUAL q	1	0	0	1	$=((p$ IMP $q)$ IMP $((q$ IMP $p)$ IMP 0)) IMP 0
p AND q	1	0	0	0	$=(p$ IMP $(q$ IMP 0)) IMP 0
p NAND q	0	1	1	1	$=p$ IMP $(q$ IMP 0)
p XOR q	0	1	1	0	$=(p$ IMP $q)$ IMP $((q$ IMP $p)$ IMP 0)
NOT q	0	1	0	1	$=q$ IMP 0
p NIMP q	0	1	0	0	$=(p$ IMP $q)$ IMP 0
NOT p	0	0	1	1	$=p$ IMP 0
q NIMP p	0	0	1	0	$=(q$ IMP $p)$ IMP 0
p NOR q	0	0	0	1	$=((p$ IMP 0) IMP $q)$ IMP 0
FALSE	0	0	0	0	$=0$

Literary sources [4, 7, 9, 11] confirm that a functionally complete implicative basis is not applied to minimize logical functions. Here, the result of minimization is given by the Boolean basis. Only after this minimization, special algorithms replace the elements of the {AND, OR, NOT} basis with the elements of the basis {→, NOT}, or {→, 0}.

For the proper use of a functionally complete implicative basis, algebra is required as part of the rules for simplifying implicative functions.

A method of figurative transformations ensures the minimization of logical functions directly in the implicative basis. Thus, the considered algorithms and methods of minimization of switching functions [4, 7, 9, 11] and the method of figurative transformations follow different approaches, and, therefore, imply different perspectives on the technological possibility of minimizing functions in the implicative basis. In particular, it is promising to use algebra as part of the rules for the equivalent transformation of the functions of the implicative basis {→, NOT}, {→, 0}, which would extend the applicability of an analytical method.

In this regard, there is reason to believe that the procedure for minimizing switching functions, which is represented by the algorithms and minimization methods reported in [4, 7, 9, 11], is insufficient for theoretical research into the optimal minimization of the functions of the implicative basis. This predetermines the need to employ equivalent figurative transformations of implicative functions, specifically for the bases {→, NOT}, {→, 0}. In a practical context, such an approach makes it possible to expand the capabilities of digital component design technology based on the implicative bases {→, NOT}, {→, 0}.

3. The aim and objectives of the study

The aim of this study is to expand a method of figurative transformations to minimize Boolean functions in the class of perfect implicative normal forms (PINF-1, PINF-1.1, PINF-2 and PINF-2.1). This would make it possible to simplify, improve the productivity of minimizing the functions of the implicative basis, by constructing algebraic rules of logical transformations. To accomplish the aim, the following tasks have been set:

– to establish the adequacy of using a method of figurative transformations to minimize the Boolean functions of the implicative basis, in particular, to establish the hermeneutics of logical operations in the implicative basis;

– to create an algebra of the implicative basis in terms of the necessary rules for minimizing Boolean functions;

– to analyze the effectiveness of minimizing the functions of the implicative basis by the method of figurative transformations and examples of function minimization in the implicative basis in order to compare.

4. Perfect implicative normal forms of Boolean functions

All definitions for the logic algebra functions in the {I, OR, NOT} basis have their analogs for the implicative basis {→ , NOT} (Table 4). Replacing the basis {I, OR, NOT} with the basis {→ , NOT} is possible according to formulae (6) to (8).

Table 4

Thesauruses of logical bases

No. of entry	Thesaurus of the basis {AND, OR, NOT}	Thesaurus of the basis {→, NOT}
1	Perfect disjunctive normal form (PDNF)	Perfect implicative normal form –2 (PINF-2)
		Perfect implicative normal form –2.1 (PINF-2.1)
2	Perfect conjunctive normal form (PCNF)	Perfect implicative normal form –1 (PINF-1)
		Perfect implicative normal form –1.1 (PINF-1.1)
3	Minimal disjunctive normal form (MDNF)	Minimal implicative normal form –2 (MINF-2)
		Minimal implicative normal form –2.1 (MINF-2.1)
4	Minimal conjunctive normal form (MCNF)	Minimal implicative normal form –1 (MINF-1)
		Minimal implicative normal form –1.1 (MINF-1.1)

PINF-1 and PINF-2 derivation is demonstrated by examples 1 and 2.

4. 1. PINF-1

The perfect disjunctive normal form (PDNF) and the perfect conjunctive normal form (PCNF) of Boolean functions can be expressed through functions other than conjunction and objection, or disjunction and objection. One can represent the PDNF or PCNF of Boolean functions by using, for example, objection and implication.

Theorem 1 [17]. Any function in the algebra of logic, except identical to unity, can be represented in the following form:

$$f\left(x_1,x_2,...,x_n\right)=\underset{0}{\&}\left(\overline{\overline{x_1^{\delta_1}\to x_2^{\overline{\delta_2}}}\to...\to x_{n-1}^{\overline{\delta_{n-1}}}}\to x_n^{\overline{\delta_n}}\right). \quad (6)$$

The proof of theorem 1 can be found in [17].

Implicative form (6) is analogous to PCNF. We assign to notation (6) a classification of the *perfect implicative normal form* – 1 (PINF-1) of the Boolean function.

To represent the Boolean function in PINF-1, all x_i arguments except x_1 must be entered into the PINF-1 function terms with objection if $x_i^{\delta_i}=1$, without objection – otherwise. For x_1, the conditions for entering the PINF-1 function terms are opposite.

Any binary set corresponds to the term of the PINF-1 function:

$$\overline{\overline{x_1^{\delta_1}\to x_2^{\overline{\delta_2}}}\to...\to x_{n-1}^{\overline{\delta_{n-1}}}}\to x_n^{\overline{\delta_n}}$$

and, on the contrary, – the term of the PINF-1 function:

$$\overline{\overline{x_1^{\delta_1}\to x_2^{\overline{\delta_2}}}\to...\to x_{n-1}^{\overline{\delta_{n-1}}}}\to x_n^{\overline{\delta_n}}$$

corresponds to the binary set (tuple). For example, the set <1100> corresponds to the term PINF-1:

$$\overline{\overline{x_1\to\overline{x_2}\to x_3}\to x_4},$$

and the term PINF-1:

$$\overline{\overline{\overline{x_1\to\overline{x_2}\to x_3}\to x_4}\to\overline{x_5}}$$

corresponds to the set <01001> set.

Example 1. It is required to represent the function $f(x_1, x_2, x_3, x_4)$ (Table 5) in the form of PINF-1.

Table 5

Truth table of the logical function $f(x_1, x_2, x_3, x_4)$

x_1	x_2	x_3	x_4	$f(x_1,x_2,x_3,x_4)$	x_1	x_2	x_3	x_4	$f(x_1,x_2,x_3,x_4)$
0	0	0	0	1	1	0	0	0	1
0	0	0	1	0	1	0	0	1	1
0	0	1	0	1	1	0	1	0	1
0	0	1	1	1	1	0	1	1	0
0	1	0	0	1	1	1	0	0	1
0	1	0	1	0	1	1	0	1	1
0	1	1	0	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	0

We shall construct the terms φ_i of the implicative function for the sets in Table 5, on which $f(x_1, x_2, x_3, x_4)=0$:

$$\varphi_1=\overline{\overline{\overline{x_1\to x_2}\to x_3}\to\overline{x_4}};\quad \varphi_2=\overline{\overline{\overline{x_1\to\overline{x_2}}\to x_3}\to\overline{x_4}};$$

$$\varphi_3=\overline{\overline{\overline{x_1\to x_2}\to\overline{x_3}}\to\overline{x_4}};\quad \varphi_4=\overline{\overline{\overline{x_1\to\overline{x_2}}\to\overline{x_3}}\to\overline{x_4}}.$$

Then notation (7):

$$\begin{aligned}f\left(x_1,x_2,x_3,x_4\right)=&\left(\overline{\overline{\overline{x_1\to x_2}\to x_3}\to\overline{x_4}}\right)\&\\&\&\left(\overline{\overline{\overline{x_1\to\overline{x_2}}\to x_3}\to\overline{x_4}}\right)\&\\&\&\left(\overline{\overline{\overline{x_1\to x_2}\to\overline{x_3}}\to\overline{x_4}}\right)\&\\&\&\left(\overline{\overline{\overline{x_1\to\overline{x_2}}\to\overline{x_3}}\to\overline{x_4}}\right).\end{aligned} \quad (7)$$

would represent the function f (x_1, x_2, x_3, x_4) in the form of PINF-1.

4. 2. Binary equivalent of PINF-1

In the method of figurative transformations, it is advisable to use a binary analog of the assigned Boolean function, including the function of the implicative basis.

Since PINF-1 is analogous to the PCNF function of the Boolean basis, PINF-1 is simplified by the Nelson method [18]. The binary equivalent of variables in the PINF-1 function $f(x_1, x_2, x_3, x_4)$ (11) can be represented in two ways:

$$\begin{aligned}F_{\text{PINF-1}}=&\left(\overline{\overline{x_1\to x_2}\to x_3}\to x_4\right)\left(\overline{\overline{x_1\to\overline{x_2}}\to x_3}\to x_4\right)\times\\&\times\left(\overline{\overline{x_1\to\overline{x_2}}\to x_3}\to\overline{x_4}\right)\left(\overline{\overline{x_1\to\overline{x_2}\to\overline{x_3}}\to x_4}\right)\times\\&\times\left(\overline{\overline{x_1\to\overline{x_2}\to\overline{x_3}}\to\overline{x_4}}\right)\left(\overline{\overline{x_1\to x_2}\to x_3}\to x_4\right)\times\\&\times\left(\overline{x_1\to\overline{x_2}\to x_3}\to x_4\right)\left(\overline{\overline{x_1\to\overline{x_2}\to x_3}\to\overline{x_4}}\right)\times\\&\times\left(\overline{x_1\to\overline{x_2}\to\overline{x_3}}\to x_4\right)\left(\overline{\overline{x_1\to\overline{x_2}\to\overline{x_3}}\to\overline{x_4}}\right).\end{aligned} \quad (8)$$

Variant 1. It is required to represent the binary equivalent of function (8) by its truth table (Table 6) followed by inverting the binary variable values.

Table 6

Truth table of the function $f(x_1, x_2, x_3, x_4)$

No.	x_1	x_2	x_3	x_4	f	No.	x_1	x_2	x_3	x_4	f
0	0	0	0	0	0	8	1	0	0	0	0
1	0	0	0	1	1	9	1	0	0	1	1
2	0	0	1	0	1	10	1	0	1	0	1
3	0	0	1	1	1	11	1	0	1	1	1
4	0	1	0	0	0	12	1	1	0	0	0
5	0	1	0	1	0	13	1	1	0	1	0
6	0	1	1	0	0	14	1	1	1	0	0
7	0	1	1	1	0	15	1	1	1	1	0

The binary equivalent of function (8), according to the first variant of the representation will take the following form:

$$F_{\text{PINF-1}}=\left|\begin{array}{c|cccc}0&0&0&0&0\\4&0&1&0&0\\5&0&1&0&1\\6&0&1&1&0\\7&0&1&1&1\\8&1&0&0&0\\12&1&1&0&0\\13&1&1&0&1\\14&1&1&1&0\\15&1&1&1&1\end{array}\right|=\left|\begin{array}{cccc}1&1&1&1\\1&0&1&1\\1&0&1&0\\1&0&0&1\\1&0&0&0\\0&1&1&1\\0&0&1&1\\0&0&1&0\\0&0&0&1\\0&0&0&0\end{array}\right|. \quad (9)$$

Variant 2. The binary equivalent variables accept a unity value if the variable x_i other than x_1 is represented in the terms of function (8) in direct code. Conversely, the binary equivalent variable takes zero if the variable x_i other than x_1 is represented by the inverse code in the terms of function (8). For x_1 in the terms of function (8), the conditions for entering the binary equivalent are opposite. The binary equivalent of the variable x_1 takes a unity value if the variable x_1 is represented in the inverse code. Conversely, the binary equivalent variable takes zero if the variable x_1 in the terms of function (8) is represented in direct code (Table 7).

Table 7

Matching the variables x_i and x_1 in the PINF-1 of the Boolean function to the second variant of binary equivalent

Variables in PINF-1 of the function x_i	Variables of binary equivalent
x_i	1
$\overline{x_i}$	0
$\overline{x_1}$	1
x_1	0

The binary equivalent of function (8), according to the second variant of the representation, will take the following form:

$$F_{\text{PINF-1}} = \left|\begin{array}{c|cccc} 0 & 1 & 1 & 1 & 1 \\ 4 & 1 & 0 & 1 & 1 \\ 5 & 1 & 0 & 1 & 0 \\ 6 & 1 & 0 & 0 & 1 \\ 7 & 1 & 0 & 0 & 0 \\ 8 & 0 & 1 & 1 & 1 \\ 12 & 0 & 0 & 1 & 1 \\ 13 & 0 & 0 & 1 & 0 \\ 14 & 0 & 0 & 0 & 1 \\ 15 & 0 & 0 & 0 & 0 \end{array}\right|. \quad (10)$$

The binary equivalents (9) and (10) in the PINF-1 of Boolean function (8) are the same.

4. 3. PINF-1.1

In (6), the conjunction can be replaced with an implicative objection based on formula (5):

$$x_1 x_2 = \overline{x_1 \to \overline{x_2}}.$$

After applying formula (5), function $f(x_1, x_2, x_3, x_4)$ (7) from Example 1 will take the following form:

$$f(x_1,x_2,x_3,x_4) = \overline{\varphi_1 \to \left[\varphi_2 \to \left(\varphi_3 \to \overline{\varphi_4}\right)\right]},$$

or

$$f(x_1,x_2,x_3,x_4) = \overline{\left(\overline{\overline{x_1 \to x_2} \to x_3 \to \overline{x_4}}\right) \to \left[\begin{array}{l} \left(\overline{\overline{x_1 \to \overline{x_2}} \to x_3 \to \overline{x_4}}\right) \to \\ \to \left(\left(\overline{\overline{x_1 \to x_2} \to \overline{x_3} \to \overline{x_4}}\right) \to \right. \\ \left. \to \left(\overline{\overline{x_1 \to \overline{x_2} \to \overline{x_3} \to \overline{x_4}}}\right)\right) \end{array}\right]}.$$

Statement 1. Any function in the algebra of logic, except identical to unity, can be represented in the following form:

$$f(x_1,x_2,...,x_n) = \overline{\varphi_1 \to \left[\varphi_2 \to ... \to \left(\varphi_{n-1} \to \overline{\varphi_n}\right)\right]},$$

where

$$\varphi_i = \left(\overline{\overline{x_1^{\delta_1} \to x_2^{\overline{\delta_2}}} \to ... \to x_{n-1}^{\overline{\delta_{n-1}}}} \to x_n^{\overline{\delta_n}}\right). \quad (11)$$

We assign to notation (11) classification of the perfect implicative normal form – 1.1 (PINF-1.1) of the Boolean function.

To represent the Boolean function in PINF-1.1, all x_i arguments except x_1 must be entered into the terms of PINF-1.1 function with the objection if $x_i^{\delta_i} = 1$, without objection – otherwise. For x_1, the conditions for entering the terms of PINF-1.1 function are the opposite.

4. 4. PINF-2

The analog of PDNF is the second form of the implicative notation of a Boolean function (PINF-2).

Theorem 2 [17]. Any function in the algebra of logic, except identical to zero, can be represented in the following form:

$$f(x_1,x_2,...,x_n) = \overline{\underset{1}{\vee}\left(\overline{\overline{x_1^{\delta_1} \to x_2^{\overline{\delta_2}}} \to ... \to x_{n-1}^{\overline{\delta_{n-1}}}} \to x_n^{\overline{\delta_n}}\right)}. \quad (12)$$

The proof of theorem 2 can be found in [17].

Here we note that the functions:

$$\varphi_i = \overline{\overline{x_1^{\delta_1} \to x_2^{\overline{\delta_2}}} \to ... x_n^{\overline{\delta_n}}}$$

are configured so that each such function returns «1» on the set corresponding to the set $<\delta_1 \overline{\delta_2} ... \overline{\delta_n}>$, and returns «0» on the remaining sets.

We assign to notation (12) classification of the perfect implicative normal form – 2 (PINF-2) of a Boolean function.

The entry of arguments x_i and x_1 in the terms of PINF-2 function is similar to the arguments entering the terms of PINF-1.

Any binary set corresponds to the term PINF-2 of the function:

$$\overline{\overline{x_1^{\delta_1} \to x_2^{\overline{\delta_2}}} \to ... \to x_{n-1}^{\overline{\delta_{n-1}}}} \to x_n^{\overline{\delta_n}}$$

and, on the contrary, the term of the PINF-2 function:

$$\overline{\overline{x_1^{\delta_1} \to x_2^{\overline{\delta_2}}} \to ... \to x_{n-1}^{\overline{\delta_{n-1}}}} \to x_n^{\overline{\delta_n}}$$

corresponds to the binary set (tuple). For example, the <1100> set corresponds to the term of PINF-2:

$$\overline{\overline{x_1 \to \overline{x_2}} \to x_3} \to x_4,$$

and the term of PINF-2:

$$\overline{\overline{x_1 \to \overline{x_2}} \to x_3} \to x_4 \to \overline{x_5}$$

corresponds to the set <01001>.

Example 2. It is required to represent the function $f(x_1, x_2, x_3)$ (Table 8) in the form of PINF-2.

Table 8

Truth table of the logical function $f(x_1, x_2, x_3)$

No.	x_1	x_2	x_3	$f(x_1, x_2, x_3)$	No.	x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0	0	4	1	0	0	1
1	0	0	1	0	5	1	0	1	1
2	0	1	0	0	6	1	1	0	0
3	0	1	1	0	7	1	1	1	0

We shall construct the terms φ_i of the implicative function for the sets in Table 8 on which $f(x_1, x_2, x_3)=1$:

$$\varphi_1=\overline{\overline{x_1\to x_2}\to x_3};$$

$$\varphi_2=\overline{\overline{x_1\to x_2}\to \overline{x_3}}.$$

Then notation (13):

$$f(x_1,x_2,x_3)=\left(\overline{\overline{x_1\to x_2}\to x_3}\right)+\left(\overline{\overline{x_1\to x_2}\to \overline{x_3}}\right) \qquad (13)$$

would represent the function $f(x_1, x_2, x_3)$ in the form of PINF-2.

4. 5. Binary equivalent of PINF-2

Since PINF-2 is analogous to PDNF of the functions of the Boolean basis, PINF-2 is simplified according to the rules of simplification for PDNF [18, 19].

The binary equivalent variables accept a unity value if the variable x_i other than x_1 is represented in the terms of function (14) in the inverse code, and, conversely,

$$f(x_1,x_2,x_3)=\left(\overline{\overline{x_1\to x_2}\to x_3}\right)+\left(\overline{\overline{x_1\to x_2}\to \overline{x_3}}\right) \qquad (14)$$

the binary equivalent variable accepts zero if the variable x_i except x_1 is represented in direct code. For x_1 in the terms of function (17), the conditions for entering the binary equivalent are opposite. The binary equivalent of the variable x_1 accepts if the variable x_1 is represented in direct code. Conversely, the binary equivalent of the variable x_1 takes zero if the variable x_1 in the terms of function (14) is represented in the inverse code (Table 9).

Table 9

Matching the variables x_i and x_1 in the PINF-2 of the Boolean function to the binary equivalent

Variables in PINF-2 of the function x_i	Variables in binary equivalent
x_i	0
$\overline{x_i}$	1
$\overline{x_1}$	0
x_1	1

The binary equivalent of function (14) will take the following form:

$$F_{\text{PINF-2}}=\left|\begin{array}{c|ccc} 4 & 1 & 0 & 0 \\ 5 & 1 & 0 & 1 \end{array}\right|. \qquad (15)$$

Thus, logical function (14) is represented by binary matrix (15).

4. 6. PINF-2.1

In (12), the disjunction can be replaced with an implicative objection based on formula (4):

$$x_1+x_2=\overline{x_1}\to x_2.$$

After applying formula (4), the function $f(x_1, x_2, x_3)$ (13) from Example 2 will take the following form:

$$f(x_1,x_2,x_3)=\left(\overline{x_1\to x_2}\to x_3\right)\to\left(\overline{\overline{x_1\to x_2}\to \overline{x_3}}\right).$$

Statement 2. Any function in the algebra of logic, except identical to zero, can be represented in the following form:

$$f(x_1,x_2,...,x_n)=\varphi_1\to\left[\varphi_2\to...\to\left(\varphi_{n-1}\to\overline{\varphi_n}\right)\right],$$

where

$$\varphi_i=\left(\overline{\overline{x_1^{\delta_1}\to x_2^{\overline{\delta_2}}}\to...\to x_{n-1}^{\overline{\delta_{n-1}}}}\to x_n^{\overline{\delta_n}}\right). \qquad (16)$$

We assign to notation (16) classification of the perfect implicative normal form – 2.1 (PINF-2.1) of a Boolean function.

To represent the Boolean function in the form of PINF-2.1, the entry of all x_i and x_1 arguments into the terms of the PINF-2.1 function is similar to the arguments entering the terms of PINF-1.

Example 3. It is required to represent the Boolean expression $x_1(x_2+x_3)+\overline{x_2}x_3$ in the implicative basis {→, NOT}.

Solution:

$$x_1(x_2+x_3)+\overline{x_2}x_3=\overline{x_1(x_2+x_3)}\to\overline{x_2}x_3=$$
$$=\overline{\overline{x_1+(x_2+x_3)}}\to\overline{\overline{x_2}+\overline{x_3}}=$$
$$=\overline{x_1}+\overline{(x_2+x_3)}\to\overline{x_2+\overline{x_3}}=$$
$$=\left(x_1\to\overline{(x_2+x_3)}\right)\to\left(\overline{\overline{x_2}\to\overline{x_3}}\right)=$$
$$=\left(x_1\to\overline{(\overline{x_2}\to x_3)}\right)\to\left(\overline{\overline{x_2}\to\overline{x_3}}\right).$$

The expression $x_1(x_2+x_3)+\overline{x_2}x_3$ is represented in the basis {→, NOT}.

5. Axioms and transformations in the implicative basis

For implication, the following axioms hold:

$$\left.\begin{array}{l} x\to x=1; \\ x\to\overline{x}=\overline{x}; \\ x\to 1=1; \\ x\to 0=\overline{x}; \\ x_1\to x_2\to x_1=x_1. \end{array}\right\} \qquad (17)$$

The validity of the represented axioms is proven by truth tables.

It follows from (17) that only the permutation law in a modified form holds for implication:

$$x_1\to x_2=\overline{x_2}\to\overline{x_1}.$$

The associative law is not applicable to implication.

The rules of implication execution ensure the following transformations of algebraic expressions (Table 10).

Table 10

Equivalent transformations in the implicative basis

No. of entry	Transformations in the implicative basis
1	$x_1 \leftrightarrow x_2 = (x_1 \to x_2)(x_2 \to x_1) = (\overline{x_1} + x_2)(\overline{x_2} + x_1) = (x_1 x_2) + (\overline{x_1}\ \overline{x_2})$
2	$x_1 \to x_2 = 1 \oplus x_1 \oplus x_1 x_2$
3	$x_1 \leftarrow x_2 = x_1 \oplus x_1 x_2$
4	$x_2 \leftarrow x_1 = \overline{x_1} x_2$
5	$x_1 \leftarrow x_2 = x_1 \overline{x_2}$
6	$x_2 \to x_1 = \overline{x_1}\ \overline{x_2} + x_1 \overline{x_2} + x_1 x_2$
7	$x_1 \to x_2 = \overline{x_1}\ \overline{x_2} + \overline{x_1} x_2 + x_1 x_2$
8	$x_1\ NOR\ x_2 = ((x_2 \to 0) \to x_1) \to 0$
9	$x_1\ NAND\ x_2 = x_2 \to (x_1 \to 0)$
10	$x_1\ XOR\ x_2 = \{((x_1 \to 0) \to x_2) \to \to [(x_1 \to (x_2 \to 0)) \to 0]\} \to 0$

The transformations in Table 10 underlie the algebraic apparatus of equivalent inter-basis transitions and simplification of logical functions.

6. Results of minimizing functions of the implicative basis using a method of figurative transformations

The equivalent figurative transformations during the minimization of the functions of the implicative basis produce the following result:

– they determine the hermeneutics of logical operations on binary structures of the functions of the implicative basis;

– they form the algebra of the implicative basis in terms of the simplification of PINF-1, PINF-1.1, PINF-2, and PINF-2.1 of Boolean functions.

6. 1. The hermeneutics of logical operations in the implicative basis

In the implicative basis, the hermeneutics of logical operations are similar to the hermeneutics of the bases considered earlier [18].

To represent perfect implicative normal forms, for example, PINF-1 of the n-place Boolean functions, by a binary equivalent or a matrix, one needs to replace the variables with inversion $\overline{x}_n$ with 1_n, and the variables without inversion x_n with 0_n (chapter 4.2), where n is the numeric index that determines the bit size of the variable symbol «1» or «0» in the terms of the function of the implicative basis. For the variable x_1 in the terms of the function of the implicative basis, the conditions for entering the binary equivalent are opposite (chapter 4.2).

A perfect implicative normal form of the 3-place function of the implicative basis:

$$F = \left(\overline{\overline{x_1} \to x_2} \to x_3\right)\left(\overline{\overline{x_1} \to x_2} \to \overline{x_3}\right), \tag{18}$$

can be represented by binary sets (tuples):

$$F = (0_1 0_2 0_3)(0_1 0_2 1_3), \tag{19}$$

or the matrix:

$$F = \begin{vmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{vmatrix}. \tag{20}$$

We denote matrix (20) an instance of the class of binary matrices of the functions of the implicative basis.

The hermeneutics of logical operations for matrix (20) is that matrix (20) yields the terms of the PINF-1 function of the implicative basis and the operation of conjunction over them. This hermeneutic is advisable to use when deriving the result of logical operations in the class of binary matrices of the functions of the implicative basis.

6. 2. Equivalent transformations of the Boolean functions of the implicative basis into PINF-1

In a general case, when minimizing the Boolean functions of the implicative basis by a method of figurative transformations, the following rules in the algebra of logic are possible.

Gluing the variable 2-place terms in a PINF-1 function can be carried out by the following transformation:

$$\left(\overline{x_1} \to x_2\right)\left(\overline{x_1} \to \overline{x_2}\right) = x_1 = 1 \to x_1. \tag{21}$$

The equivalent transformations for the rule of gluing the variable 2-place terms in PINF-1 (21) have an illustration of the combinatorial representation (22) where the binary equivalent of function (24) is represented by the second variant of its construction (chapter 4. 2).

$$\begin{vmatrix} 1 & 1 \\ 1 & 0 \end{vmatrix} = |\ 1 \quad | = x_1 = 1 \to x_1. \tag{22}$$

The resulting minimal Boolean function in PINF-1 takes the following form:

$$f_{\text{MINF-1}} = 1 \to x_1.$$

Gluing the variable 3-place terms in PINF-1 can be carried out by the following transformation:

$$\left(\overline{\overline{x_1} \to x_2} \to x_3\right)\left(\overline{\overline{x_1} \to \overline{x_2}} \to x_3\right) = \overline{x_1} \to x_3. \tag{23}$$

The equivalent transformations for the rule of gluing the variable 3-place terms in PINF-1 (23) have a representation illustration (24) where the binary equivalent of function (23) is represented by the second variant of its construction (chapter 4. 2).

$$\begin{vmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \end{vmatrix} = |\ 1 \quad 1\ | = x_1 + x_3 = \overline{x_1} \to x_3. \tag{24}$$

Gluing the variable 4-place terms in PINF-1 can be carried out by the following transformation:

$$\left(\overline{\overline{x_1 \to x_2} \to x_3} \to \overline{x_4}\right)\left(\overline{\overline{x_1 \to \overline{x_2}} \to x_3} \to \overline{x_4}\right) =$$
$$= \overline{\overline{x_1} \to x_3} \to \overline{x_4}. \quad (25)$$

The equivalent transformations for the rule of gluing the variable 4-place terms in PINF-1 (25) have a representation illustration (26):

$$\left|\begin{matrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{matrix}\right| = |\ 1 \quad\quad 1 \quad 0\ | =$$
$$= x_1 + x_3 + \overline{x_4} = \overline{\overline{x_1} \to x_3} \to \overline{x_4}. \quad (26)$$

The rule of super-gluing the variables.

For 4-place PINF-1 terms, the rule of super-gluing the variables [17] may take the following form:

$$\left(\overline{\overline{x_1 \to x_2} \to x_3} \to x_4\right)\left(\overline{\overline{x_1 \to x_2} \to x_3} \to \overline{x_4}\right) \times$$
$$\times\left(\overline{\overline{x_1 \to x_2} \to \overline{x_3}} \to x_4\right)\left(\overline{\overline{x_1 \to x_2} \to \overline{x_3}} \to \overline{x_4}\right) =$$
$$= \left(\overline{x_1 \to x_2} \to x_3 + x_4\right)\left(\overline{x_1 \to x_2} \to x_3 + \overline{x_4}\right) \times$$
$$\times\left(\overline{x_1 \to x_2} \to \overline{x_3} + x_4\right)\left(\overline{x_1 \to x_2} \to \overline{x_3} + \overline{x_4}\right) =$$
$$= \left(x_1 \to x_2 + x_3 + x_4\right)\left(x_1 \to x_2 + x_3 + \overline{x_4}\right) \times$$
$$\times\left(x_1 \to x_2 + \overline{x_3} + x_4\right)\left(x_1 \to x_2 + \overline{x_3} + \overline{x_4}\right) =$$
$$= \left(\overline{x_1} + x_2 + x_3 + x_4\right)\left(\overline{x_1} + x_2 + x_3 + \overline{x_4}\right) \times$$
$$\times\left(\overline{x_1} + x_2 + \overline{x_3} + x_4\right)\left(\overline{x_1} + x_2 + \overline{x_3} + \overline{x_4}\right) =$$
$$= \left(\overline{x_1} + x_2 + x_3\right)\left(\overline{x_1} + x_2 + x_3\right) \times$$
$$\times\left(\overline{x_1} + x_2 + \overline{x_3}\right)\left(\overline{x_1} + x_2 + \overline{x_3}\right) =$$
$$= \left(\overline{x_1} + x_2\right)\left(\overline{x_1} + x_2\right)\left(\overline{x_1} + x_2\right)\left(\overline{x_1} + x_2\right) =$$
$$= \overline{x_1} + x_2 = x_1 \to x_2. \quad (27)$$

The equivalent transformations for the rule of super-gluing the variable 4-place terms in PINF-1 (27) have a representation illustration (28):

$$\left|\begin{matrix} 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{matrix}\right| = \overline{x_1} + x_2 = x_1 \to x_2. \quad (28)$$

Rule (28) employs a 2-(2, 4)-design [19].

The rule of incomplete super-gluing of variables.

Combinatoric properties of an incomplete combinatoric system with repeated 2-(n, x/b)-design [19] ensure a rule of incomplete super-gluing of variables in the implicative basis.

For 2-place PINF-1 terms, the rule of incomplete super-gluing of variables may take the following form, for example:

$$f(x_1, x_2) = \left(\overline{x_1} \to x_2\right)\left(x_1 \to x_2\right)\left(x_1 \to \overline{x_2}\right) =$$
$$= \left(x_1 + x_2\right)\left(\overline{x_1} + x_2\right)\left(\overline{x_1} + \overline{x_2}\right) =$$
$$= x_2\left(\overline{x_1} + \overline{x_2}\right) = \overline{x_1} x_2 = \overline{x_1 + \overline{x_2}} =$$
$$= \overline{\overline{x_1} \to \overline{x_2}}. \quad (29)$$

The equivalent transformations for the rule of incomplete super-gluing of the variable 2-place PINF-1 terms (29) have a representation illustration (30) where the binary equivalent of function (32) is represented by the second variant of its construction (chapter 4. 2):

$$\left|\begin{matrix} 1 & 1 \\ 0 & 1 \\ 0 & 0 \end{matrix}\right| = \left|\begin{matrix} & 1 \\ 0 & 0 \end{matrix}\right| = \left|\begin{matrix} & 1 \\ 0 & \end{matrix}\right| =$$
$$= \overline{x_1} x_2 = \overline{x_1 + \overline{x_2}} = \overline{\overline{x_1} \to \overline{x_2}}. \quad (30)$$

Rule (30) employs a 2-(2, 3/4)-design [19].

For 3-place PINF-1 terms, the rule of incomplete super-gluing of variables may take the following form, for example:

$$\left(\overline{x_1 \to \overline{x_2}} \to \overline{x_3}\right)\left(\overline{x_1 \to \overline{x_2}} \to x_3\right) \times$$
$$\times\left(\overline{\overline{x_1} \to \overline{x_2}} \to \overline{x_3}\right)\left(\overline{\overline{x_1} \to \overline{x_2}} \to x_3\right) \times$$
$$\times\left(\overline{x_1 \to x_2} \to \overline{x_3}\right)\left(\overline{x_1 \to x_2} \to x_3\right)\left(\overline{\overline{x_1} \to x_2} \to \overline{x_3}\right) =$$
$$= \left(x_1 \to \overline{x_2} + \overline{x_3}\right)\left(x_1 \to \overline{x_2} + x_3\right) \times$$
$$\times\left(\overline{x_1} \to \overline{x_2} + \overline{x_3}\right)\left(\overline{x_1} \to \overline{x_2} + x_3\right) \times$$
$$\times\left(x_1 \to x_2 + \overline{x_3}\right)\left(x_1 \to x_2 + x_3\right)\left(\overline{x_1} \to x_2 + \overline{x_3}\right) =$$
$$= \left(\overline{x_1} + \overline{x_2} + \overline{x_3}\right)\left(\overline{x_1} + \overline{x_2} + x_3\right) \times$$
$$\times\left(x_1 + \overline{x_2} + \overline{x_3}\right)\left(x_1 + \overline{x_2} + x_3\right) \times$$
$$\times\left(\overline{x_1} + x_2 + \overline{x_3}\right)\left(\overline{x_1} + x_2 + x_3\right)\left(x_1 + x_2 + \overline{x_3}\right) =$$
$$= \left(\overline{x_1} + \overline{x_2}\right)\left(\overline{x_1} + \overline{x_2}\right)\left(x_1 + \overline{x_2}\right)\left(x_1 + \overline{x_2}\right) \times$$
$$\times\left(\overline{x_1} + x_2\right)\left(\overline{x_1} + x_2\right)\left(x_1 + x_2 + \overline{x_3}\right) =$$
$$= \left(\overline{x_1} + \overline{x_2}\right)\left(x_1 + \overline{x_2}\right)\left(\overline{x_1} + x_2\right)\left(x_1 + x_2 + \overline{x_3}\right) =$$
$$= \overline{x_2}\left(\overline{x_1} + x_2\right)\left(x_2 + \overline{x_3}\right) = \overline{x_1}\ \overline{x_2}\left(x_2 + \overline{x_3}\right) =$$
$$= \overline{x_1}\ \overline{x_2}\ \overline{x_3} = \overline{x_1 + x_2 + x_3} =$$
$$= \overline{\overline{x_1 + x_2} \to x_3} = \overline{\overline{x_1} \to x_2 \to x_3}. \quad (31)$$

The equivalent transformations for the rule of incomplete super-gluing of variable 3-place PINF-1 terms (31) have a representation illustration (32) where the binary equivalent of function (31) is represented by the second variant of its construction (chapter 4. 2):

$$\left|\begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{matrix}\right| = \left|\begin{matrix} & 0 & \\ 0 & 1 & \\ 1 & 1 & 0 \end{matrix}\right| =$$
$$= \left|\begin{matrix} & 0 & \\ 0 & 1 & \\ & 1 & 0 \end{matrix}\right| = \left|\begin{matrix} & 0 & \\ 0 & & \\ & & 0 \end{matrix}\right| =$$
$$= \overline{x_1}\ \overline{x_2}\ \overline{x_3} = \overline{x_1 + x_2 + x_3} = \overline{\overline{x_1 + x_2} \to x_3} =$$
$$= \overline{\overline{x_1} \to x_2 \to x_3}. \quad (32)$$

Rule (32) employs a 2-(3, 7/8)-design [19].

Generalized gluing of variables in the implicative basis can be carried out by the following transformation:

$$\left(x_1 \to \overline{x_2}\right)\left(x_1 \to x_3\right)\left(x_2 \to \overline{x_3}\right) =$$
$$= \left(x_1 \to x_3\right)\left(x_2 \to \overline{x_3}\right). \quad (33)$$

The equivalent transformations for the rule of generalized gluing of variables (33) have a representation illustration (34) where the binary equivalent of expression (33) is represented by the second variant of its construction (chapter 4. 2):

$$\begin{vmatrix} 0 & 0 & \\ 0 & & 1 \\ & 0 & 0 \end{vmatrix} = \begin{vmatrix} 0 & & 1 \\ & 0 & 0 \end{vmatrix} =$$
$$= \left(\overline{x_1} + x_3\right)\left(\overline{x_2} + \overline{x_3}\right) = \left(x_1 \to x_3\right)\left(x_2 \to \overline{x_3}\right). \quad (34)$$

Another variant of the generalized rule of gluing the variables in PINF-1:

$$\left(x_1 \to x_3\right)\left(x_2 \to \overline{x_3}\right) =$$
$$= \left(x_1 \to \overline{x_2}\right)\left(x_1 \to x_3\right)\left(x_2 \to \overline{x_3}\right). \quad (35)$$

$$\begin{vmatrix} 0 & & 1 \\ & 0 & 0 \end{vmatrix} = \begin{vmatrix} 0 & 0 & \\ 0 & & 1 \\ & 0 & 0 \end{vmatrix} =$$
$$= \left(\overline{x_1} + \overline{x_2}\right)\left(\overline{x_1} + x_3\right)\left(\overline{x_2} + \overline{x_3}\right) =$$
$$= \left(x_1 \to \overline{x_2}\right)\left(x_1 \to x_3\right)\left(x_2 \to \overline{x_3}\right).$$

A variable absorption rule is reduced to the following transformations:

$$1.\ \left(\overline{x_1} \to 0\right)\left(\overline{x_1} \to x_2\right) = \overline{x_1} \to 0. \quad (36)$$

The equivalent transformations for the rule of the PINF-1 variables absorption (36) have a representation illustration (37).

$$\left(\overline{x_1} \to 0\right)\left(\overline{x_1} \to x_2\right) = x_1\left(x_1 + x_2\right) =$$
$$= \begin{vmatrix} 1 & \\ 1 & 1 \end{vmatrix} = \begin{vmatrix} 1 & \end{vmatrix} = x_1 = \overline{x_1} \to 0. \quad (37)$$

$$2.\ \left(x_1 \to 0\right)\left(x_1 \to x_2\right) = x_1 \to 0. \quad (38)$$

$$3.\ \left(x_1 \to x_2\right)\left(\overline{x_1 \to x_2} \to x_3\right) = x_1 \to x_2. \quad (39)$$

$$\begin{vmatrix} 0 & 1 & \\ 0 & 1 & 1 \end{vmatrix} = \begin{vmatrix} 0 & 1 & \end{vmatrix} = \overline{x_1} + x_2 = x_1 \to x_2.$$

$$4.\ \left(x_1 \to x_2\right)\left(\overline{\overline{x_1 \to x_2} \to x_3} \to x_4\right) = x_1 \to x_2. \quad (40)$$

$$\begin{vmatrix} 0 & 1 & & \\ 0 & 1 & 1 & 1 \end{vmatrix} = \begin{vmatrix} 0 & 1 & & \end{vmatrix} = \overline{x_1} + x_2 = x_1 \to x_2.$$

The *rule of semi-gluing the variables* in the implicative basis can be carried out with the help of the following transformations:

$$\left(\overline{x_1} \to x_2\right)\left(\overline{x_1 \to x_2} \to x_3\right) =$$
$$= \left(\overline{x_1} \to x_2\right)\left(\overline{x_2} \to x_3\right). \quad (41)$$

The rule of semi-gluing the variables (41) has a representation illustration (42):

$$\begin{vmatrix} 1 & 1 & \\ 0 & 1 & 1 \end{vmatrix} = \begin{vmatrix} 1 & 1 & \\ & 1 & 1 \end{vmatrix} =$$
$$= \left(x_1 + x_2\right)\left(x_2 + x_3\right) = \left(\overline{x_1} \to x_2\right)\left(\overline{x_2} \to x_3\right). \quad (42)$$

The rule:

$$\left(\overline{x_1} \to 0\right)\left(x_1 \to x_2\right) = \overline{x_1 \to \overline{x_2}}, \quad (43)$$

is proven by the following transformations:

$$\left(\overline{x_1} \to 0\right)\left(x_1 \to x_2\right) = x_1\left(\overline{x_1} + x_2\right) =$$
$$= x_1 x_2 = \overline{\overline{x_1} + \overline{x_2}} = \overline{x_1 \to \overline{x_2}}.$$

$$x_1\left(\overline{x_1} + x_2\right) = \begin{vmatrix} 1 & \\ 0 & 1 \end{vmatrix} = \begin{vmatrix} 1 & \\ & 1 \end{vmatrix} =$$
$$= x_1 x_2 = \overline{\overline{x_1} + \overline{x_2}} = \overline{x_1 \to \overline{x_2}}. \quad (44)$$

Converting the result of figurative transformations (44) – x_1x_2 to representing it by the implicative basis employs de Morgan formula.

Example 4. It is required to simplify the logical function $f(x_1, x_2, x_3, x_4)$ (Table 11) in a perfect implicative normal form –1 (PINF-1).

Table 11

Truth table for the function $f(x_1, x_2, x_3, x_4)$

No.	x_1	x_2	x_3	x_4	$f(x_1, x_2, x_3, x_4)$	No.	x_1	x_2	x_3	x_4	$f(x_1, x_2, x_3, x_4)$
0	0	0	0	0	1	8	1	0	0	0	1
1	0	0	0	1	1	9	1	0	0	1	0
2	0	0	1	0	0	10	1	0	1	0	0
3	0	0	1	1	0	11	1	0	1	1	1
4	0	1	0	0	0	12	1	1	0	0	0
5	0	1	0	1	0	13	1	1	0	1	0
6	0	1	1	0	1	14	1	1	1	0	1
7	0	1	1	1	0	15	1	1	1	1	1

Apply the first variant of the binary equivalent of the PINF-1 function $f(x_1, x_2, x_3, x_4)$ (chapter 4. 2). The $f(x_1, x_2, x_3, x_4)$ PINF-1 is minimized by the following figurative transformations:

$$F_{\text{MINF-1}}=\left|\begin{array}{c|cccc} 2 & 0 & 0 & 1 & 0\\ 3 & 0 & 0 & 1 & 1\\ 4 & 0 & 1 & 0 & 0\\ 5 & 0 & 1 & 0 & 1\\ 7 & 0 & 1 & 1 & 1\\ 9 & 1 & 0 & 0 & 1\\ 10 & 1 & 0 & 1 & 0\\ 12 & 1 & 1 & 0 & 0\\ 13 & 1 & 1 & 0 & 1 \end{array}\right|=\left|\begin{array}{cccc} 1 & 1 & 0 & 1\\ 1 & 1 & 0 & 0\\ 1 & 0 & 1 & 1\\ 1 & 0 & 1 & 0\\ 1 & 0 & 0 & 0\\ 0 & 1 & 1 & 0\\ 0 & 1 & 0 & 1\\ 0 & 0 & 1 & 1\\ 0 & 0 & 1 & 0 \end{array}\right|=$$

$$=\left|\begin{array}{cccc} & 1 & 0 & 1\\ 1 & & 0 & 0\\ & 0 & 1 & \\ 0 & 1 & 1 & 0 \end{array}\right|=\left|\begin{array}{cccc} & 1 & 0 & 1\\ 1 & & 0 & 0\\ & 0 & 1 & \\ 0 & & 1 & 0 \end{array}\right|=$$

$$=\left(x_2+\overline{x_3}+x_4\right)\left(x_1+\overline{x_3}+\overline{x_4}\right)\times$$

$$\times\left(\overline{x_2}+x_3\right)\left(\overline{x_1}+x_3+\overline{x_4}\right)=$$

$$=\left(\overline{x_2+\overline{x_3}}\to x_4\right)\left(\overline{x_1+\overline{x_3}}\to\overline{x_4}\right)\times$$

$$\times\left(x_2\to x_3\right)\left(\overline{\overline{x_1}+x_3}\to\overline{x_4}\right)=$$

$$=\left(\overline{\overline{x_2}\to\overline{x_3}}\to x_4\right)\left(\overline{\overline{x_1}\to\overline{x_3}}\to\overline{x_4}\right)\times$$

$$\times\left(x_2\to x_3\right)\left(\overline{x_1\to x_3}\to\overline{x_4}\right).$$

The function $f(x_1,x_2,x_3,x_4)$ MINF-1 (Table 11):

$$F_{\text{MINF-1}}=\left(\overline{\overline{x_2}\to\overline{x_3}}\to x_4\right)\left(\overline{\overline{x_1}\to\overline{x_3}}\to\overline{x_4}\right)\times$$

$$\times\left(x_2\to x_3\right)\left(\overline{x_1\to x_3}\to\overline{x_4}\right). \qquad (45)$$

6. 3. Equivalent transformations of the Boolean functions of the implicative basis to PINF-1.1

Before executing the operation of gluing the variable 4-place PINF-1.1 terms, we shall glue the variable 4-place PINF-1 terms:

$$f\left(x_1,x_2,x_3,x_4\right)=$$

$$=\left(\overline{\overline{x_1}\to x_2}\to x_3\to\overline{x_4}\right)\left(\overline{\overline{x_1}\to\overline{x_2}}\to x_3\to\overline{x_4}\right)\times$$

$$\times\left(\overline{x_1\to x_2}\to\overline{x_3}\to\overline{x_4}\right)\left(\overline{x_1\to\overline{x_2}}\to\overline{x_3}\to\overline{x_4}\right)=$$

$$=\left(\overline{\overline{x_1}\to x_2}\to x_3+\overline{x_4}\right)\left(\overline{\overline{x_1}\to\overline{x_2}}\to x_3+\overline{x_4}\right)\times$$

$$\times\left(\overline{x_1\to x_2}\to\overline{x_3}+\overline{x_4}\right)\left(\overline{x_1\to\overline{x_2}}\to\overline{x_3}+\overline{x_4}\right)=$$

$$=\left(\overline{x_1}\to x_2+x_3+\overline{x_4}\right)\left(\overline{x_1}\to\overline{x_2}+x_3+\overline{x_4}\right)\times$$

$$\times\left(x_1\to x_2+\overline{x_3}+\overline{x_4}\right)\left(x_1\to\overline{x_2}+\overline{x_3}+\overline{x_4}\right)=$$

$$=\left(x_1+x_2+x_3+\overline{x_4}\right)\left(x_1+\overline{x_2}+x_3+\overline{x_4}\right)\times$$

$$\times\left(\overline{x_1}+x_2+\overline{x_3}+\overline{x_4}\right)\left(\overline{x_1}+\overline{x_2}+\overline{x_3}+\overline{x_4}\right)=$$

$$=\left(x_1+x_3+\overline{x_4}\right)\left(\overline{x_1}+\overline{x_3}+\overline{x_4}\right)=$$

$$=\left(\overline{x_1\to x_3}\to\overline{x_4}\right)\left(\overline{x_1\to x_3}\to\overline{x_4}\right). \qquad (46)$$

In (46), we shall replace conjunction with an implicative objection based on formula (5):

$$x_1x_2=\overline{x_1\to\overline{x_2}},$$

next, we shall perform the operation by gluing the variable 4-place PINF-1.1 terms.

$$f\left(x_1,x_2,x_3,x_4\right)=$$

$$=\overline{\left(\overline{\overline{x_1}\to x_2}\to x_3\to\overline{x_4}\right)\to\left[\begin{array}{l}\left(\overline{\overline{x_1}\to\overline{x_2}}\to x_3\to\overline{x_4}\right)\to\\ \to\left(\begin{array}{l}\left(\overline{x_1\to x_2}\to\overline{x_3}\to\overline{x_4}\right)\to\\ \to\left(\overline{x_1\to\overline{x_2}}\to\overline{x_3}\to\overline{x_4}\right)\end{array}\right)\end{array}\right]}=$$

$$=\overline{\left(\overline{\overline{x_1}\to x_2}\to x_3\to\overline{x_4}\right)\to\left[\begin{array}{l}\left(\overline{\overline{x_1}\to\overline{x_2}}\to x_3\to\overline{x_4}\right)\to\\ \to\left(\begin{array}{l}\overline{x_1\to x_2\to\overline{x_3}\to\overline{x_4}}+\\ +\left(\overline{x_1\to\overline{x_2}\to\overline{x_3}\to\overline{x_4}}\right)\end{array}\right)\end{array}\right]}=$$

$$=\overline{\left(\overline{\overline{x_1}\to x_2}\to x_3\to\overline{x_4}\right)\to\left[\begin{array}{l}\left(\overline{\overline{x_1}\to\overline{x_2}}\to x_3\to\overline{x_4}\right)+\\ +\overline{x_1\to x_2\to\overline{x_3}\to\overline{x_4}}+\\ +\overline{x_1\to\overline{x_2}\to\overline{x_3}\to\overline{x_4}}\end{array}\right]}=$$

$$=\overline{\overline{x_1\to x_2\to x_3\to\overline{x_4}}+\overline{x_1\to x_2\to x_3\to\overline{x_4}}+}=$$

$$=\overline{+\overline{x_1\to x_2\to\overline{x_3}\to\overline{x_4}}+\overline{x_1\to\overline{x_2}\to\overline{x_3}\to\overline{x_4}}}$$

$$=\overline{\overline{x_1+x_2\to x_3\to\overline{x_4}}+\overline{x_1+x_2\to x_3\to\overline{x_4}}+}=$$

$$=\overline{+\overline{x_1+x_2\to\overline{x_3}\to\overline{x_4}}+\overline{x_1+x_2\to\overline{x_3}\to\overline{x_4}}}$$

$$=\overline{\overline{x_1+x_2+x_3\to\overline{x_4}}+\overline{x_1+x_2+x_3\to\overline{x_4}}+}=$$

$$=\overline{+\overline{x_1+x_2+\overline{x_3}\to\overline{x_4}}+\overline{x_1+\overline{x_2}+\overline{x_3}\to\overline{x_4}}}$$

$$=\overline{\overline{x_1+x_2+x_3+\overline{x_4}}+\overline{x_1+x_2+\overline{x_3}+\overline{x_4}}+}=$$

$$=\overline{+\overline{x_1+x_2+\overline{x_3}+\overline{x_4}}+\overline{x_1+\overline{x_2}+\overline{x_3}+\overline{x_4}}}$$

$$=\overline{x_1\,\overline{x_2}\,\overline{x_3}x_4}+\overline{x_1x_2\,\overline{x_3}\,\overline{x_4}}+\overline{x_1\overline{x_2}x_3\overline{x_4}}+x_1x_2x_3x_4=$$

$$=\overline{\overline{x_1}\,\overline{x_3}x_4}+x_1x_3x_4=\overline{\overline{x_1}\,\overline{x_3}x_4}\to x_1x_3x_4=$$

$$=\overline{\left(x_1+x_3+\overline{x_4}\right)\to\left(\overline{x_1}+\overline{x_3}+\overline{x_4}\right)}=$$

$$=\overline{\left(\overline{x_1}\to x_3+\overline{x_4}\right)\to\left(\overline{x_1\to\overline{x_3}}+\overline{x_4}\right)}=$$

$$=\overline{\left(\overline{x_1\to x_3}\to\overline{x_4}\right)\to\left(\overline{x_1\to\overline{x_3}}\to\overline{x_4}\right)}. \qquad (47)$$

Transformation (47) has a representation illustration (48):

$$f\left(x_1,x_2,x_3,x_4\right)=\left|\begin{array}{cccc} 0 & 0 & 0 & 1\\ 0 & 1 & 0 & 1\\ 1 & 0 & 1 & 1\\ 1 & 1 & 1 & 1 \end{array}\right|=$$

$$=\left|\begin{array}{cccc} 1 & 1 & 1 & 0\\ 1 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 \end{array}\right|=\left|\begin{array}{cccc} 1 & & 1 & 0\\ 0 & & 0 & 0 \end{array}\right|=$$

$$=\left(x_1+x_3+\overline{x_4}\right)\left(\overline{x_1}+\overline{x_3}+\overline{x_4}\right)=$$

$$=\overline{\left(\overline{x_1\to x_3}\to\overline{x_4}\right)\to\left(\overline{x_1\to\overline{x_3}}\to\overline{x_4}\right)}. \qquad (48)$$

Figurative transformations (48) have their natural hermeneutics: the terms of the implicative basis function are combined by the implication operation «→», the last term is inverted, the entire resulting expression is inverted.

6. 4. Equivalent transformations of the Boolean functions of the implicative basis into PINF-2

Gluing the variable 4-place PINF-2 terms.

$$\left(\overline{\overline{\overline{x_1 \to x_2} \to x_3} \to \overline{x_4}}\right) + \left(\overline{\overline{\overline{x_1 \to \overline{x_2}} \to x_3} \to \overline{x_4}}\right) = \\ = \overline{\overline{\overline{x_1 \to x_3} \to \overline{x_4}}}. \quad (49)$$

The equivalent transformations for the rule of gluing the variable 4-place PINF-2 terms (49) have a representation illustration (50):

$$\left|\begin{matrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{matrix}\right| = |\ 0 \quad 0 \quad 1\ | = \overline{x_1}\,\overline{x_3} x_4 = \\ = \overline{x_1 + x_3 + \overline{x_4}} = \overline{\overline{\overline{x_1 \to x_3} \to \overline{x_4}}}. \quad (50)$$

Since PINF-2 is analogous to the PDNF function of the Boolean basis, figurative transformations in the binary matrix of implicative function (45) are carried out according to the rules of PDNF [18, 19].

The rule of super-gluing the variables.

For 4-place PINF-2 terms, the rule of super-gluing the variables [20] may take the following form, for example:

$$\left(\overline{\overline{\overline{x_1 \to x_2} \to x_3} \to x_4}\right) + \left(\overline{\overline{\overline{x_1 \to x_2} \to x_3} \to \overline{x}_4}\right) + \\ + \left(\overline{\overline{\overline{x_1 \to x_2} \to \overline{x}_3} \to x_4}\right) + \left(\overline{\overline{\overline{x_1 \to x_2} \to \overline{x}_3} \to \overline{x}_4}\right) = \\ = \left(\overline{\overline{x_1 \to x_2} \to x_3 + x_4}\right) + \left(\overline{\overline{x_1 \to x_2} \to x_3 + \overline{x}_4}\right) + \\ + \left(\overline{\overline{x_1 \to x_2} \to \overline{x}_3 + x_4}\right) + \left(\overline{\overline{x_1 \to x_2} \to \overline{x}_3 + \overline{x}_4}\right) = \\ = \left(\overline{x_1 \to x_2 + x_3 + x_4}\right) + \left(\overline{x_1 \to x_2 + x_3 + \overline{x}_4}\right) + \\ + \left(\overline{x_1 \to x_2 + \overline{x}_3 + x_4}\right) + \left(\overline{x_1 \to x_2 + \overline{x}_3 + \overline{x}_4}\right) = \\ = \left(\overline{\overline{x_1} + x_2 + x_3 + x_4}\right) + \left(\overline{\overline{x_1} + x_2 + x_3 + \overline{x_4}}\right) + \\ + \left(\overline{\overline{x_1} + x_2 + \overline{x_3} + x_4}\right) + \left(\overline{\overline{x_1} + x_2 + \overline{x_3} + \overline{x_4}}\right) = \\ = x_1\overline{x_2}\,\overline{x_3}\,\overline{x_4} + x_1\overline{x_2}\,\overline{x_3} x_4 + x_1\overline{x_2} x_3\overline{x_4} + x_1\overline{x_2} x_3 x_4 = \\ = x_1\overline{x_2}\left(\overline{x_3}\,\overline{x_4} + \overline{x_3} x_4 + x_3\overline{x_4} + x_3 x_4\right) = \\ = x_1\overline{x_2} = \overline{\overline{x_1} + x_2} = \overline{x_1 \to x_2}. \quad (51)$$

The equivalent transformations for the rule of super-gluing the variable 4-place PINF-2 terms (51) have a representation illustration (52):

$$\left|\begin{matrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{matrix}\right| = x_1\overline{x_2} = \overline{\overline{x_1} + x_2} = \overline{x_1 \to x_2}. \quad (52)$$

Rule (52) employs a 2-(2, 4)-design [19].

PINF-2 is analogous to the PDNF function of the Boolean basis, so the equivalent transformations in the binary matrix of implicative function (52) are carried out according to the rules of PDNF [18, 19].

Other logical operations over PINF-2 of the Boolean functions of the implicative basis are carried out in a similar way to the considered operations (50), (52).

Since the combinatorial structure of the truth tables for the logical functions of the implicative basis produces more information about orthogonality, contiguity, unambiguousness of the blocks in a truth table, the use of combinatorial representations to search for the objects of equivalent transformation in simplifying the functions of the implicative basis is effective.

6. 5. Equivalent transformations of the Boolean functions of the implicative basis in PINF-2.1

Before the operation of gluing the variable 3-place PINF-2.1 terms, we shall glue the variable 3-place PINF-2 terms:

$$f(x_1, x_2, x_3) = \\ = \left(\overline{\overline{x_1 \to x_2} \to x_3}\right) + \left(\overline{\overline{x_1 \to x_2} \to \overline{x_3}}\right) = \\ = \left(\overline{x_1 \to x_2 + x_3}\right) + \left(\overline{x_1 \to x_2 + \overline{x_3}}\right) = \\ = \overline{\overline{x_1} + x_2 + x_3} + \overline{\overline{x_1} + x_2 + \overline{x_3}} = \\ = x_1\overline{x_2}\,\overline{x_3} + x_1\overline{x_2} x_3 = \\ = x_1\overline{x_2} = \\ = \overline{\overline{x_1} + x_2} = \\ = \overline{x_1 \to x_2}. \quad (53)$$

$$f(x_1, x_2, x_3) = \left|\begin{matrix} 1 & 0 & 0 \\ 1 & 0 & 1 \end{matrix}\right| = |\ 1 \quad 0 \quad | = \\ = x_1\overline{x_2} = \overline{\overline{x_1} + x_2} = \overline{x_1 \to x_2}.$$

In (53), we shall replace disjunction with implication based on formula (5):

$$x_1 + x_2 = \overline{x_1} \to x_2,$$

then we shall glue the variable 3-place PINF-2.1 terms.

$$f(x_1, x_2, x_3) = \\ = \left(\overline{x_1 \to x_2} \to x_3\right) \to \left(\overline{\overline{x_1 \to x_2} \to \overline{x_3}}\right) = \\ = \overline{\overline{\overline{x_1 \to x_2} \to x_3}} + \overline{\overline{\overline{x_1 \to x_2} \to \overline{x_3}}} = \\ = \overline{x_1 \to x_2 + x_3} + \overline{x_1 \to x_2 + \overline{x_3}} = \\ = \overline{\overline{x_1} + x_2 + x_3} + \overline{\overline{x_1} + x_2 + \overline{x_3}} = \\ = x_1\overline{x_2}\,\overline{x_3} + x_1\overline{x_2} x_3 = \\ = x_1\overline{x_2} = \\ = \overline{\overline{x_1} + x_2} = \\ = \overline{x_1 \to x_2}. \quad (54)$$

The results of gluing the 3-place PINF-2 (53) and PINF-2.1 (54) terms coincide.

Transformation (54) has a representation illustration (55):

$$f(x_1,x_2,x_3)=\begin{vmatrix}1&0&0\\1&0&1\end{vmatrix}=|\,1\quad 0\quad\ |=$$
$$=x_1\overline{x_2}=\overline{\overline{x_1}+x_2}=\overline{x_1\to x_2}. \qquad (55)$$

Example 5. It is required to minimize the system of equations for a 1-bit complete binary code adder (Table 12) in the implicative basis [7].

Table 12

Truth table for a 1-bit adder of binary codes

No.	x_i	y_i	p_{i-1}	s_i	p_i
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

When contemplating Table 12, we see that everywhere, except for the sets <0,0,0> and <1,1,1>, there is a ratio $s_i=\overline{p}_i$. Let us compile a truth table that would include four arguments x_i, y_i, p_{i-1}, p_i and one function s_i (Table 13) [20].

Table 13

Truth table with four arguments x_i, y_i, p_{i-1}, p_i and one function s_i

No.	x_i	y_i	p_{i-1}	p_i	s_i
0	0	0	0	0	0
1	0	0	0	1	*
2	0	0	1	0	1
3	0	0	1	1	*
4	0	1	0	0	1
5	0	1	0	1	*
6	0	1	1	0	*
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	*
10	1	0	1	0	*
11	1	0	1	1	0
12	1	1	0	0	*
13	1	1	0	1	0
14	1	1	1	0	*
15	1	1	1	1	1

Finalize the function s_i (Table 14).

Table 14

Truth table of the finalized function s_i

No.	x_i	y_i	p_{i-1}	p_i	s_i
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	1
3	0	0	1	1	
4	0	1	0	0	1
5	0	1	0	1	
6	0	1	1	0	1
7	0	1	1	1	
8	1	0	0	0	1
9	1	0	0	1	
10	1	0	1	0	1
11	1	0	1	1	
12	1	1	0	0	1
13	1	1	0	1	
14	1	1	1	0	1
15	1	1	1	1	1

A figurative transformation method is used to minimize the finalized function s_i, by replacing the following variables: x_i-x_1, y_i-x_2, $p_{i-1}-x_3$, p_i-x_4.

$$s_i=\begin{array}{c|cccc}&x_1&x_2&x_3&x_4\\2&0&0&1&0\\4&0&1&0&0\\6&0&1&1&0\\8&1&0&0&0\\10&1&0&1&0\\12&1&1&0&0\\14&1&1&1&0\\15&1&1&1&1\end{array}=\begin{vmatrix}&&1&0\\&1&0&0\\1&0&0&0\\1&1&1&1\end{vmatrix}=$$
$$=\begin{vmatrix}&&1&0\\&1&0&0\\1&&0&0\\1&1&1&1\end{vmatrix}=\begin{vmatrix}&&1&0\\&1&&0\\1&&&0\\1&1&1&1\end{vmatrix}=\begin{vmatrix}&&1&0\\&1&&0\\1&&&0\\1&1&1&\end{vmatrix}=$$
$$=(x_1+x_2+x_3)\overline{x_4}+x_1x_2x_3=$$
$$=\overline{(x_1+x_2+x_3)\overline{x_4}}\to x_1x_2x_3=$$
$$=\overline{(x_1+x_2+x_3)\overline{x_4}}\to\left(\overline{\overline{x_1}+\overline{x_2}+\overline{x_3}}\right)=$$
$$=\left(\overline{x_1+x_2+x_3}+x_4\right)\to\left(\overline{\overline{\overline{x_1}+\overline{x_2}}\to\overline{x_3}}\right)=$$
$$=\left((x_1+x_2+x_3)\to x_4\right)\to\left(\overline{\overline{x_1\to\overline{x_2}}\to\overline{x_3}}\right)=$$
$$=\left(\left(\overline{x_1+x_2}\to x_3\right)\to x_4\right)\to\left(\overline{\overline{x_1\to\overline{x_2}}\to\overline{x_3}}\right)=$$
$$=\left(\left(\overline{\overline{x_1}\to x_2}\to x_3\right)\to x_4\right)\to\left(\overline{\overline{x_1\to\overline{x_2}}\to\overline{x_3}}\right).$$

The minimal redefined function s_i in the implicative basis (Table 14):

$$s_i=\left(\left(\overline{\overline{x_1}\to x_2}\to x_3\right)\to x_4\right)\to\overline{\overline{x_1\to\overline{x_2}}\to\overline{x_3}}. \qquad (56)$$

The x_4 (p_i) variable for implicative function (56) is fictitious because the Boolean derivative of the redefined function s_i for the x_4 variable is zero.

The minimal redefined function s_i in the Boolean basis takes the following form:

$$s_i = (x_1 + x_2 + x_3)\overline{x_4} + x_1x_2x_3.$$

Computing the Boolean derivative from the function s_i for the variable x_4 takes the following form:

$$\frac{\partial s_i}{\partial x_4} = \left((x_1 + x_2 + x_3)\bar{1} + x_1x_2x_3\right) \oplus$$
$$\oplus\left((x_1 + x_2 + x_3)\bar{0} + x_1x_2x_3\right) =$$
$$= \left((x_1 + x_2 + x_3)0 + x_1x_2x_3\right) \oplus$$
$$\oplus\left((x_1 + x_2 + x_3)1 + x_1x_2x_3\right) =$$
$$= x_1x_2x_3 \oplus (x_1 + x_2 + x_3 + x_1x_2x_3) =$$
$$= x_1x_2x_3 \oplus (x_1 + x_2 + x_3) =$$
$$= x_1x_2x_3 \cdot \overline{(x_1 + x_2 + x_3)} + \overline{x_1x_2x_3} \cdot (x_1 + x_2 + x_3) =$$
$$= x_1x_2x_3 \cdot \overline{x_1}\ \overline{x_2}\ \overline{x_3} + (\overline{x_1} + \overline{x_2} + \overline{x_3}) \cdot (x_1 + x_2 + x_3) =$$
$$= (\overline{x_1} + \overline{x_2} + \overline{x_3}) \cdot (x_1 + x_2 + x_3) = 0.$$

Since the x_4 variable of the redefined function s_i is fictitious, the implementation of the function s_i will take the following form:

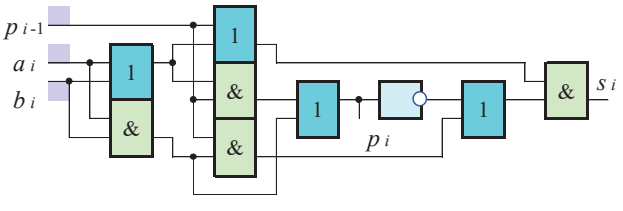

Fig. 3. 1-bit adder of binary codes

The p_i function in the implicative basis takes the following form:

$$p_i = a_ib_i + (a_i + b_i)p_{i-1} = \overline{a_ib_i} \to \left((a_i + b_i)p_{i-1}\right) =$$
$$= \left(\overline{a_i} + \overline{b_i}\right) \to \left((a_i + b_i)p_{i-1}\right) =$$
$$= \left(a_i \to \overline{b_i}\right) \to \left((a_i + b_i)p_{i-1}\right) =$$
$$= \left(a_i \to \overline{b_i}\right) \to \left(\overline{\overline{(a_i + b_i)} + \overline{p_{i-1}}}\right) =$$
$$= \left(a_i \to \overline{b_i}\right) \to \overline{\left((a_i + b_i) \to \overline{p_{i-1}}\right)} =$$
$$= \left(a_i \to \overline{b_i}\right) \to \overline{\left(\overline{(a_i \to b_i)} \to \overline{p_{i-1}}\right)}.$$

The S and C_{OUT} functions of the 1-bit adder in the implicative basis [7] take the following form:

$$S = \left[\left(\overline{a} \to b\right) \to \left(\left(a \to \overline{b}\right) \to c\right)\right] \to \left(\overline{(a \oplus b)} \to \overline{c}\right); \qquad (57)$$

$$C_{out} = \left[\left(\overline{a} \to b\right) \to \left(\overline{\left(a \to \overline{b}\right) \to c}\right)\right]. \qquad (58)$$

The equation of sum s_i (56), compared to equation (57), contains one literal less.

Example 6. It is required, by using the equivalent transformations for the Boolean function in the implicative basis $f = \left(\overline{x_1} \to \overline{x_2}\right) \to \left(x_2x_3 \to \overline{x_1}x_3\right)$, to find a minimum DNF [21].

Solution:

$$f = \left(\overline{x_1} \to \overline{x_2}\right) \to \left(x_2x_3 \to \overline{x_1}x_3\right) =$$
$$= \left(x_1 + \overline{x_2}\right) \to \left(\overline{x_2}\,\overline{x_3} + \overline{x_1}x_3\right) =$$
$$= \left(\overline{x_1 + \overline{x_2}}\right) + \left(\left(\overline{x_2} + \overline{x_3}\right) + \overline{x_1}x_3\right) = \overline{x_1}x_2 + \overline{x_2} + \overline{x_3} + \overline{x_1}x_3 =$$
$$= \overline{x_1}x_2 + \overline{x_1}x_3 + \overline{x_2}\left(x_1 + \overline{x_1}\right) + \overline{x_3}\left(x_1 + \overline{x_1}\right) =$$
$$= \overline{x_1}x_2 + \overline{x_1}x_3 + x_1\overline{x_2} + \overline{x_1}\ \overline{x_2} + x_1\overline{x_3} + \overline{x_1}\ \overline{x_3} =$$
$$= \overline{x_1}\left(x_2 + \overline{x_2}\right) + \overline{x_1}\left(x_3 + \overline{x_3}\right) + x_1\overline{x_2} + x_1\overline{x_3} =$$
$$= \overline{x_1} + x_1\left(\overline{x_2} + \overline{x_3}\right) = \left(x_1 + \overline{x_1}\right)\left(\overline{x_1} + \overline{x_2} + \overline{x_3}\right) = \overline{x_1} + \overline{x_2} + \overline{x_3}.$$

A search for the minimal DNF [21] using figurative transformations takes the following form:

$$f = \left(\overline{x_1} \to \overline{x_2}\right) \to \left(x_2x_3 \to \overline{x_1}x_3\right) =$$
$$= \left(x_1 + \overline{x_2}\right) \to \left(\overline{x_2x_3} + \overline{x_1}x_3\right) =$$
$$= \left(\overline{x_1 + \overline{x_2}}\right) + \left(\left(\overline{x_2} + \overline{x_3}\right) + \overline{x_1}x_3\right) = \overline{x_1}x_2 + \overline{x_2} + \overline{x_3} + \overline{x_1}x_3 =$$
$$= \begin{vmatrix} 0 & 1 & & \\ & 0 & & \\ & & 0 & \\ 0 & & & 1 \end{vmatrix} = \begin{vmatrix} 0 & & & \\ & 0 & & \\ & & 0 & \\ 0 & & & \end{vmatrix} = \begin{vmatrix} 0 & & \\ & 0 & \\ & & 0 \end{vmatrix} = \overline{x_1} + \overline{x_2} + \overline{x_3}.$$

The result of the simplification from the two methods is the same but the method of figurative transformations is easier.

8. Discussion of results of minimizing the functions of the implicative basis by a method of figurative transformations

The mathematical apparatus of figurative transformations is considered in works [3, 18, 19, 22], and others. Here we describe a verbal and figurative representation of information, the protocols of figurative transformations, new logical operations, an attribute of the minimal logical function, the advantages of minimizing Boolean functions on the full truth table, controlling properties of the method, the algorithm of an analytical method and its automation, the extension of the method of figurative transformations to cover logical bases. The hermeneutics of logical operations over binary structures provides the didactic simplification of Boolean functions, including for a class of perfect implicative normal forms.

The devised algebra, as part of the rules for simplifying implicative functions with an illustration of figurative transformations of logical procedures, makes it possible to implement a method of figurative transformations to minimize Boolean functions to the implicative basis.

Thus, the method becomes an alternative to the technology of designing computational components based on the implicative functions since, without the algebra, the simplification of logical functions in the implicative basis remains a function optimization in the Boolean basis. It is only after this minimization of the logical functions that it becomes possible for special algorithms to replace the elements of the main basis {AND, OR, NOT} with the elements of the implicative basis {→, NOT}, or {→, 0}. However, this approach is

characterized by verbal procedures that may fail to detect logical operations and thus reduce the possibilities of the analytical method. The equivalent transformations involving combinatorial images have in their properties a greater information capacity, and, therefore, are able to effectively replace the verbal procedures of algebraic transformations.

The object of solving the task of Boolean functions simplification in the implicative basis by the method of figurative transformations is the binary structures with repetition, which are the truth tables of the assigned functions. That makes it possible to do without auxiliary objects, such as Carnot maps, Mahony maps, Weich charts, acyclic graph, non-directed graph, covering tables, cubes, etc.

When simplifying logical formulae, it is not always obvious which laws in the algebra of logic should be applied in one step or another. Visual combinatoric structures of binary matrices and the unification of original procedures to some extent make it possible to resolve this issue.

A special feature of the considered method to simplify logical expressions in PINF-1, PINF-1.1, PINF-2, and PINF-2.1 is the use of analogs of the perfect forms of the DNF and CNF representation of Boolean functions. The specified forms of Boolean functions determine the rules of transformation on binary structures of the functions of the implicative basis.

The visual structure of figurative transformations makes it possible to manually simplify the functions of the implicative basis (using a mathematical editor, such as Math Type v. 7.0) approximately within ten input variables.

The use of the method of figurative transformations to minimize the functions of the implicative basis brings the task to simplify PINF-1, PINF-1.1, PINF-2, and PINF-2.1 to the level of a well-researched task in the class of the disjunctive and conjunctive normal forms (DCNF) of Boolean functions, as well as in the class of the perfect normal forms of functions in Schaeffer algebra (PNFS-1 and PNFS-2).

The algebra created to transform functions in the implicative basis is represented by the following logical operations (Table 15).

The limitations of applying the figurative transformation method are those cases where the switch function is represented in a mixed basis. In this scenario, the function must be represented with one logical basis.

Table 15

Logical operations in the implicative basis

No. of entry	Logical operation designation	Reference number in the text	Representation form
1	Gluing the variables	(21), (23), (25), (46)	PINF-1
2	Super-gluing the variables	(27)	PINF-1
3	Incomplete super-gluing the variables	(29), (31)	PINF-1
4	Generalized gluing the variables	(33), (35)	PINF-1
5	Absorption of variables	(36), (38), (39), (40)	PINF-1
6	Semi-gluing the variables	(41)	PINF-1
7	Rule without a name	(43)	PINF-1
8	Gluing the variables	(47)	PINF-1.1
9	Gluing the variables	(49), (53)	PINF-2
10	Super-gluing the variables	(51)	PINF-2
11	Gluing the variables	(54)	PINF-2.1

The weakness of the considered method is the limited practical application of equivalent figurative transformations for the process of minimizing the functions of the implicative basis, followed by the manufacture of appropriate computational components. The negative internal factors of the method are associated with additional time costs for establishing protocols for simplifying the functions of the implicative basis, followed by the creation of a library of rules for the algebra of logic that have an illustration of the corresponding figurative transformations. The prospect of further research may be, for example, the use of the method to minimize Boolean functions in the mixed basis class.

9. Conclusions

1. It has been established that simplification of the Boolean functions of the implicative basis using a figurative transformation method is based on a principal diagram with repetition, which is the truth table of the assigned function. This makes it possible to focus the principle of simplification within a truth table of the function and thus do without auxiliary objects such as covering tables, Carnot maps, Weich charts, acyclic graph, cubic representation, etc.

A perfect normal form of the n-place implicative basis function can be represented by binary sets (19) or matrix (20), which, in this case, would represent the terms of implication function and a conjunction operation over them. Such hermeneutics should be used effectively in the simplification of logical functions and when deriving the result of logical operations in the class of binary matrices of the functions of the implicative basis.

2. To properly simplify the functions of the implicative basis by a method of figurative transformations, we have devised the algebra of the implicative basis regarding the rules to simplify PINF-1, PINF-1.1, PINF-2, and PINF-2.1 of the Boolean functions of the implicative basis. Creating an algebra of the implicative basis solves the task of function minimization in the implicative basis.

3. The equivalent transformations involving combinatorial images, which, by their properties, have a greater information capacity, can effectively replace the verbal procedures of algebraic transformations.

The algebra born as part of the rules for simplifying the functions of the implicative basis provides for the direct transformation of logical expressions. And the method of figurative transformations, by using the visual combinator structures of binary matrices and the unification of original procedures, ensures proper transformation of logical expressions and functions. In turn, verbal procedures have a smaller information capacity, require active monitoring, which creates a beginning for not detecting the logical operations (for example, generalized gluing of variables, super-gluing of variables, incomplete super-gluing of variables, semi-gluing of variables), and, therefore, reduce the possibilities of the analytical method.

Memristors are most effective when using logic based on an implication operation. Parallel connection of two memristors

realizes the function of material implication. Together with the universal elements AND-NOT and OR-NOT, the implication function and the constant zero function form a functionally complete basis. Thus, by applying it, one can perform all 16 switch functions of two variables. The algebra created as part of the rules for simplifying implicative functions with the illustrations of figurative transformations of logical procedures makes it possible to expand the application of a functionally complete implicative basis in the fields of computing where the said basis is not currently used to the fullest extent.

By using the structure of a crossbar with memristors, one can execute an implication operation, and, based on it, other logical operations. Anything that can be computed on silicon can be implemented with the help of memristors.

References

1. Bulkin, V. (2014). Modelling of the relation of implication with use of the directed relational networks. Eastern-European Journal of Enterprise Technologies, 6 (4 (72)), 30–37. doi: https://doi.org/10.15587/1729-4061.2014.30567
2. Dychka, I. A., Tarasenko, V. P., Onai, M. V. (2019). Osnovy prykladnoi teoriyi tsyfrovykh avtomativ. Kyiv: KPI im. Ihoria Sikorskoho, 508. Available at: https://core.ac.uk/download/pdf/323531874.pdf
3. Riznyk, V., Solomko, M., Tadeyev, P., Nazaruk, V., Zubyk, L., Voloshyn, V. (2020). The algorithm for minimizing Boolean functions using a method of the optimal combination of the sequence of figurative transformations. Eastern-European Journal of Enterprise Technologies, 3 (4 (105)), 43–60. doi: https://doi.org/10.15587/1729-4061.2020.206308
4. Kvatinsky, S., Satat, G., Wald, N., Friedman, E. G., Kolodny, A., Weiser, U. C. (2014). Memristor-Based Material Implication (IMPLY) Logic: Design Principles and Methodologies. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 22 (10), 2054–2066. doi: https://doi.org/10.1109/tvlsi.2013.2282132
5. Shirinzadeh, S., Datta, K., Drechsler, R. (2018). Logic Design Using Memristors: An Emerging Technology. 2018 IEEE 48th International Symposium on Multiple-Valued Logic (ISMVL). doi: https://doi.org/10.1109/ismvl.2018.00029
6. Teodorovic, P., Vukobratovic, B., Struharik, R., Dautovic, S., Nauka, F. tehnickih, Sad, N. (2012). Sequence generator for computing arbitrary *n*-input Boolean function using two memristors. 2012 20th Telecommunications Forum (TELFOR). doi: https://doi.org/10.1109/telfor.2012.6419391
7. Rohani, S. G., TaheriNejad, N. (2017). An improved algorithm for IMPLY logic based memristive Full-adder. 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE). doi: https://doi.org/10.1109/ccece.2017.7946813
8. Wang, X., Han, J., Yang, Y., Li, Y. (2019). An Improved Mapping and Optimization Method for Implication-based Memristive Circuits Using And-Inverter Graph. Journal of Physics: Conference Series, 1237, 032026. doi: https://doi.org/10.1088/1742-6596/1237/3/032026
9. Teimoory, M., Amirsoleimani, A., Shamsi, J., Ahmadi, A., Alirezaee, S., Ahmadi, M. (2014). Optimized implementation of memristor-based full adder by material implication logic. 2014 21st IEEE International Conference on Electronics, Circuits and Systems (ICECS). doi: https://doi.org/10.1109/icecs.2014.7050047
10. Borghetti, J., Snider, G. S., Kuekes, P. J., Yang, J. J., Stewart, D. R., Williams, R. S. (2010). «Memristive» switches enable «stateful» logic operations via material implication. Nature, 464 (7290), 873–876. doi: https://doi.org/10.1038/nature08940
11. Lehtonen, E., Laiho, M. (2009). Stateful implication logic with memristors. 2009 IEEE/ACM International Symposium on Nanoscale Architectures. doi: https://doi.org/10.1109/nanoarch.2009.5226356
12. Raghuvanshi, A., Perkowski, M. (2014). Logic synthesis and a generalized notation for memristor-realized material implication gates. 2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). doi: https://doi.org/10.1109/iccad.2014.7001393
13. Lehtonen, E., Poikonen, J. H., Laiho, M. (2010). Two memristors suffice to compute all Boolean functions. Electronics Letters, 46 (3), 230. doi: https://doi.org/10.1049/el.2010.3407
14. Chua, L. (1971). Memristor-The missing circuit element. IEEE Transactions on Circuit Theory, 18 (5), 507–519. doi: https://doi.org/10.1109/tct.1971.1083337
15. Krivulya, G., Syrevich, E., Vlasov, I., Pavlov, O. (2013). Osobennosti primeneniya nanomemristornoy logiki dlya proektirovaniya tsifrovyh sistem. Visnyk Khersonskoho natsionalnoho tekhnichnoho universytetu, 1 (46), 280–286. Available at: http://nbuv.gov.ua/UJRN/Vkhdtu_2013_1_54
16. Eliseev, N. (2010). Memristors and Crossbars: Nanotechnologies for Processors. Electronics: Science, Technology, Business, 8, 84–89. Available at: https://www.electronics.ru/files/article_pdf/0/article_149_323.pdf
17. Pospelov, D. A. (1974). Logicheskie metody analiza i sinteza shem. Moscow: Energiya, 368. Available at: http://urss.ru/cgi-bin/db.pl?lang=Ru&blang=ru&page=Book&id=25326
18. Riznyk, V., Solomko, M. (2018). Minimization of conjunctive normal forms of boolean functions by combinatorial method. Technology Audit and Production Reserves, 5 (2 (43)), 42–55. doi: https://doi.org/10.15587/2312-8372.2018.146312
19. Riznyk, V., Solomko, M. (2017). Application of super-sticking algebraic operation of variables for Boolean functions minimization by combinatorial method. Technology Audit and Production Reserves, 6 (2 (38)), 60–76. doi: https://doi.org/10.15587/2312-8372.2017.118336
20. Nikishechkin, A. P. (2019). Diskretnaya matematika i diskretnye sistemy upravleniya. Moscow: Yurayt, 298. Available at: https://urait.ru/book/diskretnaya-matematika-i-diskretnye-sistemy-upravleniya-442305
21. Primery resheniy: minimizatsiya DNF. Available at: https://www.matburo.ru/ex_dm.php?p1=bfmin
22. Riznyk, V., Solomko, M. (2018). Research of 5-bit boolean functions minimization protocols by combinatorial method. Technology Audit and Production Reserves, 4 (2 (42)), 41–52. doi: https://doi.org/10.15587/2312-8372.2018.140351