The object of research are models of optimal logic circuits based on universal Peirce-Webb functions. The problem solved is the efficiency of the technique for simplifying the Peirce-Webb functions. The extension of the non-standard system to the simplification of Peirce-Webb functions makes it possible to discover new rules of equivalent transformations of Boolean functions, and to complete the simplification procedure in one step. A feature of the simplification of functions in the Peirce-Webb basis by a non-standard system is fixing the digital project at the level of abstraction, followed by the application of the mechanism of logical synthesis to generate the corresponding equivalent at the level of gates of the logic circuit. The result of the transformation of the terms of the binary matrix in the end is some combinatorial system, metadata that can explain other data, for example, determine the minimum function for another logical basis.

The interpretation of the result consists in the use of combinatorial properties of binary structures of functions in the Peirce-Webb basis and binary structures of functions in the basic basis. These properties do not depend on the selected logical basis, which makes it possible to carry out equivalent transformations on binary matrices of Peirce-Webb functions according to the rules of the algebra of the main basis.

It has been experimentally confirmed that a non-standard system enables:

– to reduce the algorithmic complexity of simplifying the Peirce-Webb functions;

– to increase the performance of the simplification of Peirce-Webb functions by 200–300 %;

– to demonstrate the visibility of the process of simplifying functions.

In terms of application, the non-standard system of simplifying the Peirce-Webb functions could ensure the transfer of innovations to material production: from conducting fundamental research, expanding the capabilities of digital component design technology to organizing serial or mass production of novelties

Keywords: DNF simplification, Peirce-Webb function, Peirce-Webb basis, non-standard system, logical circuit, AND-NOT, OR-NOT gates

UDC 681.325
DOI: 10.15587/1729-4061.2024.312968

IMPLEMENTATION OF A NON-STANDARD SYSTEM FOR SIMPLIFYING PEIRCE-WEBB FUNCTIONS

Mykhailo Solomko
Corresponding author
PhD, Associate Professor
Department of Computer Engineering*
E-mail: doctrinas@ukr.net
Petro Tadeyev
PhD, Doctor of Pedagogical Sciences, Professor
Department of Higher Mathematics*
Mykola Antoniuk
PhD, Associate Professor
Department of Information and Communication Technologies and Methods of Teaching Informatics**
Yuliia Mala
PhD
Department of Computer Science and Software Engineering
University of Customs and Finance
Volodymyr Vernadskyi, 2/4, Dnipro, Ukraine, 49000
Stepaniia Babych
PhD, Associate Professor
Department of Information Technologies and Modeling**
Yakiv Ivashchuk
PhD
Department of Higher Mathematics*
*National University of Water and Environmental Engineering
Soborna str., 11, Rivne, Ukraine, 33028
**Rivne State University of Humanities
S. Bandery str., 12, Rivne, Ukraine, 33028

Received date 12.07.2024
Accepted date 23.09.2024
Published date 30.10.2024

How to Cite: *Solomko, M., Tadeyev, P., Antoniuk, M., Mala, Y., Babych, S., Ivashchuk, Y. (2024). Implementation of a non-standard system for simplifying Peirce-Webb functions. Eastern-European Journal of Enterprise Technologies, 5 (4 (131)), 6–32. https://doi.org/10.15587/1729-4061.2024.312968*

1. Introduction

In practice, many logic circuits are implemented using OR-NOT or AND-NOT gates due to the fact that the main gates in some families of logic, such as transistor-transistor logic (TTL) and complementary metal oxide semiconductor (CMOS), are AND-NOT and OR-NOT gates. The AND and OR gates also exist in these families but their selection is much smaller, and they are usually more expensive, have longer signal delays, and may have higher thermal dissipation capacity. The implementation of the AND function by AND-NOT gates is performed by connecting two AND-NOT gates in a cascade, the first of which performs the AND-NOT operation, and the second is used as an inverter. The technique of implementing Boolean functions on AND-NOT gates does not necessarily ensure their minimal implementation. However, in some cases, factorization makes it possible to create an AND-NOT circuit with fewer gates.

These logic elements require fewer transistors (for example, in NMOS logic, a NAND gate is simpler than an AND or OR logic element) [1–3]. However, developers naturally use representations based on the logic base {AND, OR, NOT} rather than based on {NOR, NAND}. In addition, almost all known methods for minimizing logical circuits, starting with

Carnot maps and ending with Espresso algorithms, also give results based on {AND, OR, NOT} [4, 5]. Only after such minimization, special algorithms replace elements of the basis {AND, OR, NOT} with elements of the basis {NOR, NAND}.

Logical elements AND-NOT and OR-NOT are used for hardware implementation of any logic circuit. For this reason, AND-NOT and OR-NOT logic elements are called universal gates.

The Peirce-Webb basis uses OR-NOT logical elements and is functionally complete for the space of Boolean functions from 2 variables. This means that each Boolean function can be implemented using a combination of elements OR-NOT (Fig. 1) or AND-NOT.

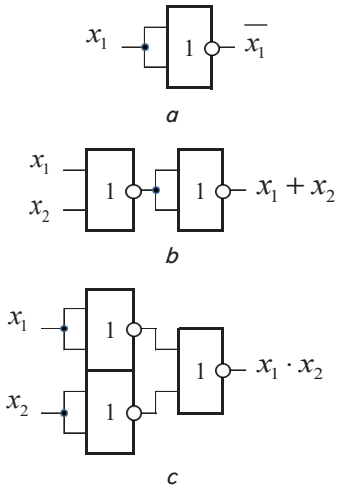

Fig. 1. Implementation of elements of the basic basis {NOT, AND, OR} on OR-NOT elements: *a* – inverter; *b* – conjunction; *c* – disjunction

Logical NOR represents a dyadic logical operation 2-OR-NOT (Fig. 2).

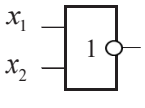

Fig. 2. Logic element 2-OR-NOT

The symbol ↓ is used to indicate the "Peirce arrow" operation. The operation "logical NOR" is the negation of the disjunction $f = (x_1 \downarrow x_2) = \overline{x_1 + x_2} = \overline{x_1} \cdot \overline{x_2}$, therefore, the value of the operation "logical NOR" will be "true" only when both arguments x_1 and x_2 take the value "false" (Table 1).

Table 1

Truth table of the "logical NOR" operation

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

The Peirce-Webb basis belongs to the field of optimization of logical functions. In this regard, it is still relevant to conduct studies aimed, in particular, at improving such factors as:

– methods for simplifying Boolean functions in the Peirce-Webb basis;

– minimization of logic circuits based on Peirce-Webb functions;

– reliability of the optimal function minimization result;

– the cost of the process of simplifying logical functions.

2. Literature review and problem statement

Fast transformation of the multi-level logic of the basis {AND, OR, NOT} to the functionally equivalent scheme of the basis {OR-NOT, AND-NOT} is considered in paper [6]. It is shown that the problem can be solved by replacing logical AND and OR elements with OR-NOT or AND-NOT elements, but this requires in some cases the introduction of additional inverters or the distribution of logic elements. The paper proposes algorithms for fast transformation of the circuit from the basis {AND, OR, NOT} to the basis {OR-NOT, AND-NOT}, while minimizing the number of inverters. The presented algorithms make it possible to turn any multi-level scheme into a chain, which is a combination of logical elements OR-NOT, AND-NOT or both types of universal logical elements. Almost all known methods for minimizing Boolean functions, including Espresso, give results in the same logical AND-OR-NOT basis. Only after such a simplification procedure, a library of logic elements is connected, including OR-NOT and AND-NOT gates for circuit synthesis. For two-level minimization in the form of a sum of products or a product of sums, such a mapping is trivial. But in FPGA schemes consisting of logical blocks of multi-level implementation of Boolean functions, optimizing the transformation of a multi-level scheme from an AND-OR-NOT basis to an OR-NOT-AND-NOT basis is not an easy task. It should be noted that during the transition from the main basis to the Schaeffer and Peirce-Webb bases using the method discussed in [6], the number of operations for the transition would increase non-linearly. In turn, the number of logic elements in digital technology devices would increase linearly depending on the number of variables that are applied to the input of the microcircuit.

The implementation of logical functions using OR-NOT or AND-NOT gates is considered in [7]. The replacement of NOT, OR and AND logic elements in the scheme with AND-NOT logic elements is demonstrated. The uniqueness of replacing logical elements from the basis {AND, OR, NOT} with elements from the basis {OR-NOT, AND-NOT} is ensured by de Morgan transformations. The procedure for replacing logical elements is completed by optimizing the scheme, which is reduced to replacing a connection with an even number of inversions in the form of a NOT function with a conductor segment. It is worth noting that the replacement of NOT, OR and AND logic elements in the circuit with AND-NOT logic elements gives a linear increase in the number of logic elements in digital technology devices, depending on the number of variables that are applied to the input of the microcircuit.

The optimization of simple combinational universal logic elements using the Minitab19 programming environment and Boolean algebra is reported in [8]. It is noted that the replacement of AND and OR logic elements on the OR-NOT or AND-NOT gate in some cases requires the introduction of additional inverters or the separation of gates. The result of minimization is simplified expressions of functions in the form of sum of products (SOP) or product of sums (POS). The SOP form can be transformed into an AND-NOT expression, but such a transformation does not provide an optimal circuit, neither in the range of gates nor in the range of

logic levels. Paper [8] also presents a method of genetic coding for the synthesis of combinational logic universal circuits, when universal AND-NOT logic gates are used instead of the main set of gates, i.e. AND, OR, NOT, XOR. Experimental results show that the considered technique gives better results, compared to the modernization of the SOP form to the AND-NOT expression. It should be noted that the genetic algorithm refers to heuristic methods that provide approximate simplification and minimization of Boolean functions. A genetic algorithm could provide accurate minimization with a large number of iterations. The problem, however, is the stopping of the algorithm, which is currently unsolved.

One of the tasks of the synthesis and analysis of the operation of combinational devices is the representation of Boolean functions in different bases. The absolute completeness of the use and implementation of various transformations of Boolean functions is impossible without the implementation of transformations into universal Schaeffer (AND-NOT) or Peirce-Webb (OR-NOT) bases. The Logic library of the Maple computer algebra system makes it possible to use only AND-NOT (Scheffer basis) and OR-NOT (Peirce-Webb basis) operations to build a digital device model. Therefore, there is a need to extend the functionality of the Maple computer algebra system logic library by representing Boolean functions in Schaeffer and/or Peirce-Webb bases. In this regard, work [9] reports the development of procedures that allow disjunctive or conjunctive normal forms of the main basis to be represented in the Schaeffer basis or in the Peirce-Webb basis. The development was carried out taking into account the existing functions and data processing procedures. The paper presents the technology of expanding the Logic library of the Maple computer algebra system with developed procedures that implement the transformation of a given or obtained Boolean function in the Schaeffer or Peirce-Webb bases. It is advisable to provide the Maple computer algebra system with the use of distributive laws of the 1st and 2nd kind, which will thus form a reserve for optimal simplification of functions in the Schaeffer (AND-NOT) and Peirce-Webb (OR-NOT) bases.

In paper [10], the transition from the basic basis of arbitrary logical functions to the bases of Schaeffer and Peirce is investigated. To this end, recurrent dependences of the representation of disjunctive and conjunctive monomials in the specified bases were first established and generalizations were made to arbitrary logical formulas in the form of disjunctive and conjunctive normal forms of representation. Estimates of the number of operations for logical formulas during the transition to the Schaeffer and Peirce bases are obtained, which determines the practical value of the results. It is worth noting that before the transition of the Boolean function of the main basis to the Schaeffer and Peirce bases, it is possible to apply distributive laws of the 1st and 2nd kind (factorization). This is possible both with the disjunctive normal form (DNF) representation and with the conjunctive normal form (CNF) representation of the function, which is a reserve for optimizing the number of logical operations and literals for the considered problems.

The algorithm for implementing the Boolean value of functions using only logical elements of conjunction negation (NAND) or only disjunction negation (NOR) is reported in [11]. The algorithm for converting the logical structure of the Boolean basis to the structure of the monobasis is performed step by step. First, the Boolean function represented by the logic elements {AND, OR, NOT}, using de Morgan's laws in various forms, is transformed so that only NAND elements or only NOR elements are used in the circuit. Next, redundant inverters are removed. In the case of two inverters in series (inverted output going directly into the inverted input), both inverters are removed because they cancel each other out. At the last stage, the remaining inverters are replaced with equivalent NAND or NOR elements. It is worth noting that during the transition from the main basis to the Schaeffer and Peirce-Webb bases using the method discussed in [11], the number of operations for the transition would grow non-linearly. In turn, the number of logical elements in digital technology devices would increase linearly depending on the number of variables that are applied to the input of the microcircuit.

Optimization of combinational logic circuits using NAND elements and genetic programming is considered in paper [12]. Optimization of a logic circuit implementing a Boolean function can be performed according to various criteria. It can be optimization of the complexity of the circuit, the number of logic levels, the number of semiconductor devices in the circuit, etc. In [12], the authors describe an approach that uses genetic programming to optimize a given Boolean function with respect to the above criteria. Instead of a set of logical elements {AND, OR, NOT, XOR}, universal NAND elements were used, which ensured better speed and compactness of the circuit. Conventional methods for minimizing logical structures give simplified expressions in two standard forms: sum of products (SOP) or product of sums (POS). The SOP form can be converted to a NAND expression by a procedure, but the conversion does not lead to an optimal circuit, either in terms of the number of logic elements or the number of logic levels. The results of experimental studies showed that the method of genetic programming, which is reported in [12], gives better results, compared to the transformation of the SOP form into a NAND expression, in terms of the number of gates, logic levels, and semiconductor devices of the circuit. It is worth noting that genetic programming refers to heuristic methods that provide approximate simplification and minimization of Boolean functions. The genetic programming method can demonstrate exact minimization similar to the results of the analytical method, however, with a large number of iterations. The problem here is stopping the algorithm, which currently appears to be unsolved.

From our review of the literature [6–12], it follows that the task can be solved by replacing logical elements AND and OR with OR-NOT or AND-NOT elements. This, however, requires in some cases the introduction of additional inverters or the distribution of logic elements. The unambiguity of replacing logical elements from the basis {AND, OR, NOT} with elements from the basis {OR-NOT, AND-NOT} is ensured by de Morgan transformations [7, 11]. To replace logical elements AND and OR on the gate AND-NOT or OR-NOT, in [8, 9] automation of calculations is applied. Another method of replacing logical AND and OR elements with OR-NOT or AND-NOT elements consists in using recurrent dependences of the representation of disjunctive and conjunctive monomials in the specified bases [10].

The methods for simplifying Boolean functions in the Peirce-Webb basis, discussed in [6–9, 11, 12], with the exception of [10], represent the first experience of solving problems of this type. The task is solved by replacing logical AND and OR elements with OR-NOT or AND-NOT elements. However, making the transition from the basic basis to the Schaeffer or Peirce-Webb basis at the level of logical elements gives a non-linear increase in the number of operations

for this transition. In turn, the number of logical elements in digital technology devices would increase linearly depending on the number of variables that are applied to the input of the microcircuit. Recurrent dependences give intermediate operations, which increases the complexity of the algorithm for replacing elements from the basis {AND, OR, NOT} with elements from the basis {OR-NOT, AND-NOT}.

The non-linear increase in the number of transition operations from the basis {AND, OR, NOT} to the basis {OR-NOT, AND-NOT} and the linear increase of logical elements in digital devices may affect the innovative capacity of the technology of simplifying the Peirce-Webb functions in the future, which is a problem.

Changes to the technology of simplifying Boolean functions in the Peirce-Webb basis in the form of the objection of methods, which are considered in works [6–12], serve to overcome the specified problem.

To this end, it is necessary:

– to devise new rules for equivalent transformation of Boolean functions in the Peirce-Webb basis;

– to provide a technology for simplifying the Peirce-Webb functions in one step;

– to carry out the minimization of Peirce-Webb functions on the complete truth table;

– to apply, when simplifying the Peirce-Webb functions, the distributive law of the 2nd kind;

– to take into account the simplification of the Peirce-Webb functions in the Reed-Müller basis.

The non-standard system is based on the combinatorial properties of binary structures PNFPW-1, PNFPW-2 functions in the Peirce-Webb basis and DDNF, DKNF functions of the main basis. These properties do not depend on the selected logical basis, which makes it possible to carry out equivalent transformations on the binary matrices PNFPW-1, PNFPW-2 according to the rules of the algebra of the main basis. The result of the transformation of the terms of the binary matrix in the end is some combinatorial system, metadata that can explain other data, for example, determine the minimum function for another logical basis.

Thus, algorithms and methods for replacing logical AND and OR elements with AND-NOT or OR-NOT elements, created software tools for them [6–12] and a non-standard system for simplifying Boolean functions have excellent approaches (principles). And so, they imply different prospects regarding the possibility of optimal simplification of Boolean functions in the Peirce-Webb basis.

And this is a reason to believe that the software-technological base, which is represented by algorithms and methods for replacing the logical elements of the basic basis AND and OR with elements of the universal basis OR-NOT, AND-NOT for solving a Boolean problem [6–12], is insufficient for conducting theoretical studies regarding the optimal simplification of the Peirce-Webb functions. This predetermines the need to carry out research using a non-standard system for simplifying Boolean functions in the Peirce-Webb basis.

In terms of application, the non-standard system of simplifying the Peirce-Webb functions could ensure the development of the innovation process and the transfer of innovations into material production. From decision-making, conducting fundamental research, expanding the capabilities of digital component design technology based on universal Peirce-Webb functions, creating prototypes, testing them, to organizing serial or mass production of novelties and their implementation.

3. The aim and objectives of the study

The aim of our work is to extend a non-standard system for simplifying the Peirce-Webb functions, in particular the first perfect normal form of the function in the Peirce-Webb basis (PNFPW-1) and the second perfect normal form of the function in the Peirce-Webb basis (PNFPW-2). This could make it possible to expand the algebra of equivalent transformations, to increase the productivity of the simplification of the specified functions.

To achieve the set goal, the following tasks must be solved:

– to determine the hermeneutics of logical transformations on the binary structures of Peirce-Webb functions;

– to establish the rules for equivalent transformation of Peirce-Webb functions in exceptional situations;

– to expand the algebra of equivalent transformations to simplify Boolean functions in the Peirce-Webb basis;

– to analyze the expediency of minimizing the Peirce-Webb functions on the complete truth table;

– to conduct a comparative analysis of the results of the simplification of Boolean functions by a non-standard system in the Peirce-Webb basis and the simplification of functions by a matrix algorithm, algebraic bi-decomposition, and the method of bitwise partitioning of the set of output conjunctures, in order to compare the cost of implementing minimal functions.

4. The study materials and methods

The object of research are models of optimal logic circuits based on universal Peirce-Webb functions.

It should be expected that the regular and constant application of the visual-matrix form of the analytical method for simplifying Boolean functions in the Peirce-Webb basis will allow for the following:

– to fix digital projects at the level of abstraction, followed by the use of the logic synthesis mechanism to generate the corresponding equivalent at the gate level;

– to form new rules for equivalent transformation of Boolean functions in the Peirce-Webb basis;

– to complete the procedure for simplifying the Peirce-Webb functions in one step;

– to simplify the Peirce-Webb functions in the Reed-Müller basis.

The term "Binary combinatorial systems with repeated 2-(n, b)-design, 2-(n, x/b)-design" is shortened to "combinatorial systems 2-(n, b)-design, 2-(n, x/b)-design", "2-(n, b)-design, 2-(n, x/b)-design systems".

FAL – functions of the algebra of logic.

Any logical function in the Peirce-Webb algebra can be represented in the canonical form – OR-NOT/OR-NOT (OR-NOT/OR-NOT). OR-NOT/OR-NOT form is 2-level (normal) – both inner and outer are OR-NOT functions.

The canonical form of the OR-NOT/OR-NOT logical function of n variables consists of Peirce-Webb terms of the n-th rank, combined by the OR-NOT operation.

The Peirce-Webb term of the n-th rank takes the following generalized form [13]:

$$\overline{x_n^{\bar{\sigma}_n} \vee x_{n-1}^{\bar{\sigma}_{n-1}} \vee ... \vee x_n^{\bar{\sigma}_1}} \text{ or } x_n^{\bar{\sigma}_n} \downarrow x_{n-1}^{\bar{\sigma}_{n-1}} \downarrow ... \downarrow x_n^{\bar{\sigma}_1},$$

where

$$x_i^{\bar{\sigma}_i} = \begin{cases} \bar{x}_i, \text{ if } \sigma_i = 1, \\ x_i, \text{ if } \sigma_i = 0, \end{cases} i = 2,3,...,n, <\sigma_n \sigma_{n-1} ... \sigma_1> - \text{binary set.}$$

Any binary set corresponds to a Peirce-Webb term $\overline{x_n^{\bar{\sigma}_n} \vee x_{n-1}^{\bar{\sigma}_{n-1}} \vee ... \vee x_n^{\bar{\sigma}_1}}$ or a Peirce-Webb function $x_n^{\bar{\sigma}_n} \downarrow x_{n-1}^{\bar{\sigma}_{n-1}} \downarrow \downarrow ... \downarrow x_n^{\bar{\sigma}_1}$ and, conversely, a binary set (tuple) corresponds to a Peirce-Webb term $\overline{x_n^{\bar{\sigma}_n} \vee x_{n-1}^{\bar{\sigma}_{n-1}} \vee ... \vee x_n^{\bar{\sigma}_1}}$ or a Peirce-Webb function $x_n^{\bar{\sigma}_n} \downarrow x_{n-1}^{\bar{\sigma}_{n-1}} \downarrow ... \downarrow x_n^{\bar{\sigma}_1}$. For example, a set <0101> corresponds to a Peirce-Webb term $\overline{x_4 \vee \overline{x_3} \vee x_2 \vee \overline{x_1}}$ or a Peirce-Webb function $x_4 \downarrow \overline{x_3} \downarrow \underline{x_2} \downarrow \overline{x_1}$, and a set <011> corresponds to a Peirce-Webb term $\overline{x_3 \vee \overline{x_2} \vee \overline{x_1}}$ or a Peirce-Webb function $x_3 \downarrow \overline{x_2} \downarrow \overline{x_1}$.

Any function in the algebra of logic can be represented in the form of a perfect disjunctive normal form (PDNF):

$$f(x_1, x_2, ..., x_n) = \vee_1 x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdot ... \cdot x_n^{\alpha_n} = \vee_1 F_i(x_1, ..., x_n),$$

where $F_i(x_1, ..., x_n) = x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdot ... \cdot x_n^{\alpha_n}$ – characteristic function, $i = 1,2, ..., k$ – numbers of sets on which the function returns 1.

Any function in the algebra of logic can also be represented in the form of perfect conjunctive normal form (PCNF):

$$f(x_1, x_2, ..., x_n) = \wedge_0 \left(\overline{x}_1^{\alpha_1} \vee \overline{x}_2^{\alpha_2} \vee ... \vee \overline{x}_n^{\alpha_n} \right) = \wedge_0 \overline{F}_i(x_1, ..., x_n),$$

where $F_i(x_1, ..., x_n) = x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdot ... \cdot x_n^{\alpha_n} = \left(\overline{x}_1^{\alpha_1} \vee \overline{x}_2^{\alpha_2} \vee ... \vee \overline{x}_n^{\alpha_n} \right)$ (on the basis of de Morgan's formula).

Thus:

$$f(x_1, x_2, ..., x_n) = \vee_1 F_i(x_1, ..., x_n) = \wedge_0 \overline{F}_i(x_1, ..., x_n).$$

For a function of n variables, the maximum number of characteristic functions is 2^n.

It follows from de Morgan's formulas for a function of two variables that $x_1 \downarrow x_2 = \overline{x_1 x_2}$ or $\overline{x_1} \downarrow \overline{x_2} = x_1 x_2$. Analogous expressions can be written for a function of n variables. Then:

$$x_1^{\alpha_1} x_2^{\alpha_2} ... x_n^{\alpha_n} = \overline{x}_1^{\alpha_1} \downarrow \overline{x}_2^{\alpha_2} \downarrow ... \downarrow \overline{x}_n^{\alpha_n}$$

or

$$F_i(x_1, ..., x_n) = \overline{x}_1^{\alpha_1} \downarrow \overline{x}_2^{\alpha_2} \downarrow ... \downarrow \overline{x}_n^{\alpha_n}$$

under condition:

$$i = \alpha_1 2^{n-1} + \alpha_2 2^{n-2} + ... + \alpha_n 2^0.$$

Since $F_1 \vee F_2 \vee ... \vee F_k = \overline{F_1 \downarrow F_2 \downarrow ... \downarrow F_k}$, the first perfect normal form of writing the function in the Peirce-Webb basis (PNFPW-1) is obtained from the expression for PDNF:

$$f(x_1, x_2, ..., x_n) = \overline{\downarrow_1 F_i(x_1, ..., x_n)}.$$

Similarly, on the basis of the relation $\overline{F_1} \wedge \overline{F_2} \wedge ... \wedge \overline{F_k} = F_1 \downarrow F_2 \downarrow ... \downarrow F_k$, from the expression for PCNF, the second perfect normal form of writing functions in the Peirce-Webb basis (PNFPW-2) is obtained:

$$f(x_1, x_2, ..., x_n) = \downarrow_0 F_i(x_1, ..., x_n).$$

In both forms of the Peirce-Webb basis, the functions $F_i(x_1, ..., x_n)$ are found in the same way:

$$F_i(x_1, ..., x_n) = \overline{x}_1^{\alpha_1} \downarrow \overline{x}_2^{\alpha_2} \downarrow ... \downarrow \overline{x}_n^{\alpha_n}.$$

Therefore, all definitions for functions in the algebra of logic in the {AND, OR, NOT} basis have their analogs for the Peirce-Webb basis {OR-NOT} (Table 2). Replacing the basis {AND, OR, NOT} with the basis {OR-NOT} is possible on the basis of de Morgan's formulas:

$$x_1 \cdot x_2 = \overline{\overline{x_1} + \overline{x_2}};$$

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

The rules of transition from the tabular form of the function to, for example, PNFPW-2 are as follows:

1. Select all sets of variables on which the function returns 0 in the truth table.

2. For each set of variables for which the function returns 0, write expressions of the type $\overline{x}_1^{\alpha_1} \downarrow \overline{x}_2^{\alpha_2} \downarrow ... \downarrow \overline{x}_n^{\alpha_n}$. If the argument x_i is included in the given set of variables as 0, then x_i is written without changes, if the argument x_i is included in the given set of variables as 1, then x_i is written with a negation sign $(\overline{x_i})$.

3. All obtained expressions of the characteristic functions $F_i(x_1, ..., x_n)$ are combined by the k-binary Peirce-Webb operation, where k is the number of sets of variables in the truth table on which the function returns a zero value.

Let us represent the PNFPW-1 and PNFPW-2 functions for the Boolean function $f(x_1, x_2, x_3)$ (Table 3).

PNFPW-1:

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

PNFPW-2:

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

Table 2

Thesauri of logical bases

No. of entry	{AND, OR, NOT} basis thesaurus	{OR-NOT} basis thesaurus
1	Implicant	Inversant
2	A simple implicant	A simple inversant
3	Perfect disjunctive normal form (PDNF)	Peirce–Webb-1 perfect normal form (PNFPW-1)
4	Perfect conjunctive normal form (PCNF)	Peirce–Webb-2 perfect normal form (PNFPW-2)
5	Minimal disjunctive normal form (MDNF)	Pierce–Webb-1 minimal normal form (MNFPW-1)
6	Minimal conjunctive normal form (MCNF)	Pierce–Webb-2 minimal normal form (MNFPW-2)

Table 3

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

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

The Peirce-Webb function is given by the expression $x_1 \downarrow x_2 = \overline{x_1 + x_2} = \overline{x_1}\,\overline{x_2}$. The Peirce-Webb algebra has axioms (Table 4), which are verified using truth tables.

Table 4

Axioms of Peirce-Webb algebra and transformation formulas

No. of entry	Logical NOR, 2-OR-NOT	No. of entry	Logical NOR, 2-OR-NOT
1	$x \downarrow x = \bar{x}$	6	$\bar{x} \downarrow 0 = x$
2	$x \downarrow \bar{x} = 0$	7	$x_1 \downarrow x_2 = \overline{\bar{x}_1 / \bar{x}_2}$
3	$x \downarrow 1 = 0$	8	$\overline{x_1} \downarrow \overline{x_2} = \overline{x_1 / x_2}$
4	$x \downarrow 0 = \bar{x}$	9	$x_1 / x_2 = \overline{\bar{x}_1 \downarrow \bar{x}_2}$
5	$\bar{x} \downarrow 0 = x$	10	$\overline{x_1} / \overline{x_2} = \overline{x_1 \downarrow x_2}$

Based on the axioms (Table 4), it follows that only the commutative law is valid for the logical NOR function:

$$x_1 \downarrow x_2 = x_2 \downarrow x_1.$$

By analogy with dyadic functions, n-digit Peirce-Webb functions (W_n) are also considered (Table 5):

$$W_n = x_1 \downarrow x_2 \downarrow x_3 \downarrow \ldots \downarrow x_n.$$

Table 5

Truth table of the n-digit Peirce-Webb function (W_n)

x_1	x_2	x_3	.	.	.	x_{n-1}	x_n	W_n
0	0	0	–	–	–	0	0	1
0	0	0	–	–	–	0	1	0
0	0	0	–	–	–	1	0	0
–	–	–	–	–	–	–	–	–
–	–	–	–	–	–	–	–	–
1	1	1	–	–	–	0	0	0
1	1	1	–	–	–	1	0	0
1	1	1	–	–	–	1	1	0

When $n=2$, from Table 5 we get a dyadic Peirce-Webb function, and at $n=1$ the Peirce-Webb function degenerates into a negation function. Therefore, the expression $x \downarrow x$ can be represented as $\bar{x}$:

$$\bar{x} = \overline{x + x} = x \downarrow x.$$

Any dyadic logical operation can be represented using a logical NOR, for example:

$x \downarrow x \equiv \bar{x}$ – denial;

$(x_1 \downarrow x_1) \downarrow (x_2 \downarrow x_2) \equiv \overline{x_1} \downarrow \overline{x_2} \equiv x_1 x_2$ – conjunction;

$(x_1 \downarrow x_2) \downarrow (x_1 \downarrow x_2) \equiv \overline{x_1 \downarrow x_2} \equiv x_1 + x_2$ – disjunction;

$((x_1 \downarrow x_1) \downarrow x_2) \downarrow ((x_1 \downarrow x_1) \downarrow x_2) \equiv x_1 \to x_2$ – implication.

In this way, logical NOR operation can be used by itself, without any other logical functions, as part of a logical formal system (making this function functionally complete).

For electronics, this means that the implementation of all the variety of signal conversion schemes that represent logic values involves one universal element called the "2-OR-NOT operation" (2-in NOR). On the other hand, this approach increases the complexity of schemes implementing logical expressions. This reduces their reliability, and also increases signal transit time, as well as reduces the performance of the digital device.

The following relations are valid for n-digit Peirce-Webb functions [14]:

$$x \downarrow x \downarrow x \downarrow \ldots \downarrow x = \bar{x};$$

$$x \downarrow x \downarrow \ldots \downarrow x \downarrow \bar{x} \downarrow \ldots \downarrow \bar{x} = 0;$$

$$x_1 \downarrow \ldots \downarrow x_l \downarrow 1 \downarrow \ldots \downarrow 1 = 0;$$

$$x_1 \downarrow \ldots \downarrow x_l \downarrow 0 \downarrow \ldots \downarrow 0 = x_1 \downarrow \ldots \downarrow x_l;$$

$$x_1 \downarrow x_2 \downarrow \ldots \downarrow x_n = \overline{x_1 \vee x_2 \vee \ldots \vee x_n} = \bar{x}_1 \bar{x}_2 \ldots \bar{x}_n.$$

At the end, relations are presented that demonstrate the connection between the n-digit Peirce-Webb and Schaeffer functions, which are analogs of de Morgan's formulas:

$$\left.\begin{aligned} x_1 / x_2 / \ldots / x_n &= \overline{\bar{x}_1 \downarrow \bar{x}_2 \downarrow \ldots \downarrow \bar{x}_n}; \\ x_1 \downarrow x_2 \downarrow \ldots \downarrow x_n &= \overline{\bar{x}_1 / \bar{x}_2 / \ldots / \bar{x}_n}. \end{aligned}\right\}$$

The validity of all the above ratios can be established using FAL truth tables.

5. Results of the simplification of Peirce-Webb functions by a non-standard system

5. 1. Extension of the hermeneutics of logical operations to binary structures of Peirce-Webb functions

In order to represent the perfect normal form of the n-digit Peirce-Webb function as a binary equivalent, the rules of chapter 4 must be followed. To this end, variables with inversion $\bar{x}_n$ are replaced by 1_n, and variables without inversion x_n are replaced by 0_n, where n is a numerical index that determines the bitness of the symbol-variable "1" or "0" in terms of the Peirce-Webb function. Following the specified procedure with the values of the variables belonging to the perfect normal form of the function of the basic basis, for example, the conjunctive normal form (PCNF) of the function:

$$f(x_1, x_2, x_3) = (\overline{x_1} + \overline{x_2} + \overline{x_3})(\overline{x_1} + \overline{x_2} + x_3)$$

it is possible to obtain the binary equivalent of the perfect normal form of the Peirce-Webb function (PNFPW-2):

$$F = (1_1 \downarrow 1_2 \downarrow 1_3) \downarrow (1_1 \downarrow 1_2 \downarrow 0_3),$$

or the matrix:

$$F = \begin{vmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \end{vmatrix}. \qquad (1)$$

Matrix (1) is an instance of the class of binary matrices of Peirce-Webb functions. The algebraic analog of the binary representation of the Peirce-Webb function (1) is the expression:

$$F=\left(x_1\downarrow x_2\downarrow x_3\right)\downarrow\left(x_1\downarrow x_2\downarrow\overline{x_3}\right).$$

The hermeneutics of logical operations for matrix (1) is that the matrix (1) gives the terms of the second perfect normal form of the function in the Peirce-Webb basis (PNFPW-2) and the "Logical NOR" operation for them. A similar hermeneutic is possible for the terms of the first perfect normal form of the function in the Peirce-Webb basis (PNFPW-1). The specified hermeneutics should be used when deriving the results of equivalent transformations in the class of binary matrices of Peirce-Webb functions.

Example 1. For the function $f(x_1, x_2, x_3)$ (Table 6), find PNFPW-1, MNFPV-1, PNFPW-2, MNFPV-2.

Table 6

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

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

Solution.

PNFPW-1 of the function $f(x_1, x_2, x_3)$ (Table 6) takes the following form:

$$F_{\text{DNFPW}-1}=\overline{\left(x_1\downarrow x_2\downarrow\overline{x_3}\right)\downarrow\left(x_1\downarrow\overline{x_2}\downarrow\overline{x_3}\right)\downarrow}$$
$$\overline{\downarrow\left(\overline{x_1}\downarrow\overline{x_2}\downarrow x_3\right)\downarrow\left(\overline{x_1}\downarrow\overline{x_2}\downarrow\overline{x_3}\right)}.$$

The MNFPV-1 derivation of the function $f(x_1, x_2, x_3)$ (Table 6) has an illustration of mapping:

$$F_{\text{MNFPW}-1}=$$
$$=\begin{vmatrix}1&1&0\\1&0&0\\0&0&1\\0&0&0\end{vmatrix}=\begin{vmatrix}1&&0\\0&0&\end{vmatrix}=\overline{\left(x_1\downarrow\overline{x_3}\right)\downarrow\left(\overline{x_1}\downarrow\overline{x_2}\right)}. \qquad (2)$$

Note. The first matrix of expression (2) reflects the structure of the PNFPW-1 function $f(x_1, x_2, x_3)$ (Table 6). It is important to note that the combinatorial properties of binary structures PNFPW-1, PNFPW-2 functions in the Peirce-Webb basis and PDNF, PCNF functions of the main basis do not depend on the selected logical basis (Boolean basis or Peirce-Webb basis). This makes it possible to carry out equivalent transformations in the first matrix of expression (2) according to the rules of the algebra of the main basis.

PNFPW-2 of the function $f(x_1, x_2, x_3)$ (Table 6) takes the following form:

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

The MNFPW-2 derivation of the function $f(x_1, x_2, x_3)$ (Table 6) has an illustration of mapping:

$$F_{\text{MNFPW}-2}=\begin{vmatrix}1&1&1\\1&0&1\\0&1&1\\0&1&0\end{vmatrix}=\begin{vmatrix}1&&1\\0&1&\end{vmatrix}=$$
$$=\overline{\left(x_1\downarrow x_3\right)}\overline{\left(\overline{x_1}\downarrow x_2\right)}=\left(x_1\downarrow x_3\right)\downarrow\left(\overline{x_1}\downarrow x_2\right)$$

or

$$F_{\text{MNFPW}-2}=\begin{vmatrix}1&1&1\\1&0&1\\0&1&1\\0&1&0\end{vmatrix}=\begin{vmatrix}1&&1\\0&1&\end{vmatrix}=\left(x_1\downarrow x_3\right)\downarrow\left(\overline{x_1}\downarrow x_2\right). \qquad (3)$$

The first matrix of expression (3) reflects the PNFPW-2 structure of the function $f(x_1, x_2, x_3)$ (Table 6). It is important to note that equivalent transformations in the first matrix of expression (3) are carried out according to the rules of the logic algebra of the main basis. In particular, the gluing of variables was carried out according to rule (18).

5. 2. Exceptional situations during the simplification of Peirce-Webb functions

The first exceptional situation. If the simplified Peirce-Webb function has term(s) which, among the other n-digit terms present, consists of a single literal (that is, the term has maximum rank), then the literal representing the term is inverted.

In the Peirce-Webb basis, one literal is given in the form:

$$x\downarrow x=\overline{x}\text{ or }\overline{x}\downarrow\overline{x}=x. \qquad (4)$$

Expression (4) represents the negation operation implemented by logic elements as in Fig. 3.

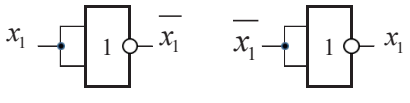

Fig. 3. Logic elements for the negation operation and one literal in the Peirce-Webb basis

Example 2. Obtain MNFPW-1 for the Boolean function $f(x_1, x_2, x_3, x_4)$ given in the canonical form:

$$f\left(x_1, x_2, x_3, x_4\right)=\Sigma(2,3,4,5,6,7,12,13,14,15). \qquad (5)$$

Solution.

Simplification of function (5) is carried out in the Peirce-Webb basis. To this end, the values of the function variables are written to the visual matrix form according to the rules of chapter 4:

$F_{\text{MDFPW}-1}=$

No.	x_1	x_2	x_3	x_4
2	1	1	0	1
3	1	1	0	0
4	1	0	1	1
5	1	0	1	0
6	1	0	0	1
7	1	0	0	0
12	0	0	1	1
13	0	0	1	0
14	0	0	0	1
15	0	0	0	0

$$=\begin{array}{|cccc|}\hline x_1&x_2&x_3&x_4\\\hline 1&&0&\\&0&&\\\hline\end{array}=\overline{\left(x_1\downarrow\overline{x_3}\right)\downarrow x_2}. \qquad (6)$$

In the first matrix of expression (6), the following actions are performed: to blocks 4, 5, 6, 7, 12, 13, 14, 15, which contain the complete combinatorial system 2-(3, 8)-design and to blocks 2, 3, 6, 7, which contain a complete combinatorial system of 2-(2, 4)-design, super-gluing operations of variables are applied [15].

The sixth "1001" and the seventh "1000" sets of variables are common to the two systems – $\Sigma m(4, 5, 6, 7, 12, 13, 14, 15)$ and $\Sigma m(2, 3, 6, 7)$ and two operations of super gluing of variables, the localization of which is determined by combinatorial systems 2-(3, 8)-design and 2-(2, 4)-design [16, 17].

The first exception is applied to the variable x_2. As a result, the minimum normal form of the Boolean function in the Peirce-Webb basis (MNFPW-1) was obtained in one step:

$$F_{\text{MNFPW-1}} = \overline{\left(x_1 \downarrow \overline{x_3}\right) \downarrow x_2}.$$

Example 3. Simplify the Boolean function specified in PNFPW-1.

$$F_{\text{DNFPW-1}} =$$
$$= \overline{\left(\overline{x_1} \downarrow \overline{x_2} \downarrow \overline{x_3}\right) \downarrow \left(\overline{x_1} \downarrow \overline{x_2} \downarrow x_3\right) \downarrow}$$
$$\overline{\downarrow \left(\overline{x_1} \downarrow x_2 \downarrow x_3\right) \downarrow \left(x_1 \downarrow \overline{x_2} \downarrow \overline{x_3}\right) \downarrow}$$
$$\overline{\downarrow \left(x_1 \downarrow \overline{x_2} \downarrow x_3\right) \downarrow \left(x_1 \downarrow x_2 \downarrow x_3\right)}.$$

Solution:

$F_{\text{MNFPW-1}} =$

No.	x_1	x_2	x_3
7	0	0	0
6	0	0	1
4	0	1	1
3	1	0	0
2	1	0	1
0	1	1	1

$=$

x_1	x_2	x_3
	0	
		1

$=$

$$= \overline{\overline{x_2} \downarrow \overline{x_3}} = \overline{x_2 \downarrow \overline{x_3}}. \quad (7)$$

In the first matrix of expression (7), the following actions are performed: the operation of super-gluing the variables is applied to blocks 0, 2, 4, 6 and to blocks 2, 3, 6, 7, each of which contains a complete combinatorial system 2-(2, 4)-design [15].

The second "101" and the sixth "001" sets of variables are common to two systems – $\Sigma m(0, 2, 4, 6)$ and $\Sigma m(2, 3, 6, 7)$ and two operations of super-gluing the variables, the localization of which is determined by combinatorial 2-(2, 4)-design systems [16, 17].

The first exception is applied to the variables x_2 and x_3. As a result, the minimum normal form of the Peirce-Webb function (MNFPW-1) was obtained in one step:

$$F_{\text{MNFPW-1}} = \overline{x_2 \downarrow \overline{x_3}}. \quad (8)$$

The second exceptional situation. If the result of the simplification of the Peirce-Webb function is only one Peirce-Webb term containing several literals, then a general inversion over all literals is taken (Fig. 4).

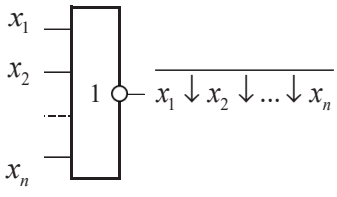

Fig. 4. Logic element *n*-OR-NOT in the Peirce-Webb basis

Example 4. Simplify the Boolean function defined in PNFPW-1.

$$F_{\text{DNFPW-1}} = \overline{\left(\overline{x_1} \downarrow \overline{x_2} \downarrow \overline{x_3}\right) \downarrow \left(\overline{x_1} \downarrow \overline{x_2} \downarrow x_3\right)}.$$

Solution:

$$F_{\text{MNFPW-1}} = \left|\begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{matrix}\right| = |\ 0 \quad 0 \quad | = \overline{\overline{\overline{x_1} \downarrow \overline{x_2}}} = \overline{x_1} \downarrow \overline{x_2}. \quad (9)$$

To derive the result of (9), the second exceptional situation is taken into account.

Example 5. For the function from example 2, simplify PNFPW-2.

$$F_{\text{DNFPW-2}} = \left(x_1 \downarrow x_2 \downarrow \overline{x_3}\right) \downarrow \left(\overline{x_1} \downarrow x_2 \downarrow \overline{x_3}\right).$$

Solution:

$F_{\text{MNFPW-2}} =$

$=$

x_1	x_2	x_3
1	1	0
0	1	0

$=$

x_1	x_2	x_3
	1	0

$$= \overline{x_2 \downarrow \overline{x_3}}. \quad (10)$$

To derive the result of (10), the second exceptional situation is taken into account. It should be noted that $F_{\text{MNFPW-1}}$ (8) and $F_{\text{MNFPW-2}}$ (10) coincide.

Example 6. Simplify PNFPW-2 for the Boolean function $f(x_1, x_2, x_3, x_4)$ given by the truth table (Table 7):

Table 7

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

No. of entry	x_1	x_2	x_3	x_4	$f(x_1, x_2, x_3, x_4)$
2	0	0	1	0	0
6	0	1	1	0	0
7	0	1	1	1	0
9	1	0	0	1	0
11	1	0	1	1	0

Solution.

The "logical NOR" function does not obey the law of associativity. This must be taken into account when moving from *n*-digit operations to dyadic operations. Such a transition can be made using the following ratios:

$$x_1 \downarrow x_2 \downarrow x_3 = x_1 \downarrow \overline{x_2 \downarrow x_3} =$$
$$= \overline{x_1 \downarrow x_2} \downarrow x_3 \neq \left(x_1 \downarrow x_2\right) \downarrow x_3,$$

$$x_1 \downarrow x_2 \downarrow x_3 \downarrow x_4 = \overline{x_1 \downarrow x_2} \downarrow \overline{x_3 \downarrow x_4},$$

the validity of which is checked by a truth table.

Simplification of the function $f(x_1, x_2, x_3, x_4)$ (Table 7) is carried out in the Peirce-Webb basis. To this end, the values of the function variables are written to the visual matrix form according to the rules from chapter 4:

$$F_{\text{MNFPW-2}} = \left|\begin{array}{c|cccc} 2 & 1 & 1 & 0 & 1 \\ 6 & 1 & 0 & 0 & 1 \\ 7 & 1 & 0 & 0 & 0 \\ 9 & 0 & 1 & 1 & 0 \\ 11 & 0 & 1 & 0 & 0 \end{array}\right| = \left|\begin{array}{cccc} 1 & & 0 & 1 \\ 1 & 0 & 0 & \\ 0 & 1 & & 0 \end{array}\right| =$$

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

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

$$= \left(x_1 \downarrow \overline{\overline{x_3} \downarrow x_4}\right) \downarrow \overline{\left(x_1 \downarrow \overline{\overline{x_2} \downarrow \overline{x_3}}\right) \downarrow \left(\overline{x_1} \downarrow \overline{x_2 \downarrow \overline{x_4}}\right)}. \quad (11)$$

MNFPW-2 function (11) consists of dyadic operations "logical NOR" and takes the following form:

$$F_{\text{MNFPW-2}} = \left(x_1 \downarrow \overline{\overline{x_3} \downarrow x_4}\right) \downarrow \overline{\left(x_1 \downarrow \overline{\overline{x_2} \downarrow \overline{x_3}}\right) \downarrow \left(\overline{x_1} \downarrow \overline{x_2 \downarrow \overline{x_4}}\right)}. \quad (12)$$

Verification of MNFPW-2 (12) is carried out in the main basis given in Table 8.

Table 8 demonstrates that MNFPW-2 $\left(x_1 \downarrow \overline{\overline{x_3} \downarrow x_4}\right) \downarrow \overline{\left(x_1 \downarrow \overline{\overline{x_2} \downarrow \overline{x_3}}\right) \downarrow \left(\overline{x_1} \downarrow \overline{x_2 \downarrow \overline{x_4}}\right)}$ satisfies the terms of the initial function of the main basis (Table 7).

The third exceptional situation. If the result of simplifying the function is only one term, and one that contains only one literal, then the MNFPW takes the following from:

$$f_{\text{MNFPW}} = \overline{\overline{x_n}} = x_n.$$

That is, in the third exceptional situation, the result of the simplification of the Peirce-Webb function does not change.

In the Peirce-Webb basis, a literal with a double inversion corresponds to a logic element in Fig. 5.

Rules (14), (20), (28) serve as an example of the simplification of the Peirce-Webb functions for the occurrence of the third exceptional situation.

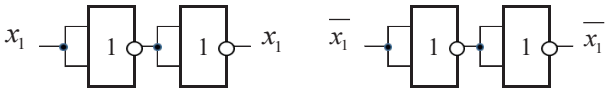

Fig. 5. Logic elements for a literal with a double inversion in the Peirce-Webb basis

5. 3. Equivalent transformations in the Peirce-Webb basis

During the simplification of Boolean functions in the Peirce-Webb basis by a non-standard system, the following rules of logic algebra are possible.

Gluing the variable dyadic PNFPW-1 terms:

$$\overline{\left(x_1 \downarrow x_2\right) \downarrow \left(\overline{x}_1 \downarrow x_2\right)} = x_2. \quad (13)$$

Proof (13):

$$\overline{\left(\overline{x_1} \downarrow \overline{x_2}\right) \downarrow \left(x_1 \downarrow \overline{x_2}\right)} = \overline{\left(x_1 \cdot x_2\right) \downarrow \left(\overline{x_1} \cdot x_2\right)} =$$

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

Equivalent transformations for the rule of gluing the variable dyadic PNFPW-1 terms (13) have an illustration of mapping:

$$\left|\begin{array}{cc} 1 & 1 \\ 0 & 1 \end{array}\right| = \left|\begin{array}{c} \\ 1 \end{array}\right| = x_2. \quad (14)$$

To derive the result in (14), the third exceptional situation is taken into account.

Gluing the variable 3-digit PNFPW-1 terms:

$$\overline{\left(x_1 \downarrow x_2 \downarrow x_3\right) \downarrow \left(x_1 \downarrow x_2 \downarrow \overline{x_3}\right)} = \overline{x_1 \downarrow x_2}. \quad (15)$$

Proof of expression (15):

$$\overline{\left(x_1 \downarrow x_2 \downarrow x_3\right) \downarrow \left(x_1 \downarrow x_2 \downarrow \overline{x_3}\right)} =$$

$$= \overline{\left(\overline{x_1}\,\overline{x_2}\,\overline{x_3}\right) \downarrow \left(\overline{x_1}\,\overline{x_2} x_3\right)} = \overline{\left(\overline{\overline{x_1}\,\overline{x_2}\,\overline{x_3}}\right)\left(\overline{\overline{x_1}\,\overline{x_2} x_3}\right)} =$$

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

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

$$= x_1 \downarrow x_2 =$$

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

Table 8

Verification of MNFPW-2 – $\left(x_1 \downarrow \overline{\overline{x_3} \downarrow x_4}\right) \downarrow \overline{\left(x_1 \downarrow \overline{\overline{x_2} \downarrow \overline{x_3}}\right) \downarrow \left(\overline{x_1} \downarrow \overline{x_2 \downarrow \overline{x_4}}\right)}$

No.	x_1	x_2	x_3	x_4	$F_{\text{DNFPW-2}}$	$\left(x_1 \downarrow \overline{\overline{x_3} \downarrow x_4}\right) \downarrow \overline{\left(x_1 \downarrow \overline{\overline{x_2} \downarrow \overline{x_3}}\right) \downarrow \left(\overline{x_1} \downarrow \overline{x_2 \downarrow \overline{x_4}}\right)}$	$F_{\text{MNFPW-2}}$
2	0	0	1	0	0	$\left(0_1 \downarrow \overline{1_3} \downarrow 0_4\right) \downarrow \overline{\left(0_1 \downarrow \overline{0_2} \downarrow \overline{1_3}\right) \downarrow \left(\overline{0_1} \downarrow \overline{0_2 \downarrow \overline{0_4}}\right)}$	0
6	0	1	1	0	0	$\left(0_1 \downarrow \overline{1_3} \downarrow 0_4\right) \downarrow \overline{\left(0_1 \downarrow \overline{1_2} \downarrow \overline{1_3}\right) \downarrow \left(\overline{0_1} \downarrow \overline{1_2 \downarrow \overline{0_4}}\right)}$	0
7	0	1	1	1	0	$\left(0_1 \downarrow \overline{1_3} \downarrow 1_4\right) \downarrow \overline{\left(0_1 \downarrow \overline{1_2} \downarrow \overline{1_3}\right) \downarrow \left(\overline{0_1} \downarrow \overline{1_2 \downarrow \overline{1_4}}\right)}$	0
9	1	0	0	1	0	$\left(1_1 \downarrow \overline{0_3} \downarrow 1_4\right) \downarrow \overline{\left(1_1 \downarrow \overline{0_2} \downarrow \overline{0_3}\right) \downarrow \left(\overline{1_1} \downarrow \overline{0_2 \downarrow \overline{1_4}}\right)}$	0
11	1	0	1	1	0	$\left(1_1 \downarrow \overline{1_3} \downarrow 1_4\right) \downarrow \overline{\left(1_1 \downarrow \overline{0_2} \downarrow \overline{1_3}\right) \downarrow \left(\overline{1_1} \downarrow \overline{0_2 \downarrow \overline{1_4}}\right)}$	0

Equivalent transformations for the rule of gluing the variable 3-digit PNFPW-1 terms (15) in the Peirce-Webb basis have an illustration of mapping:

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

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

$$= \left|\begin{array}{cc} 1 & 1 \end{array}\right| = \overline{x_1 \downarrow x_2}. \quad (16)$$

To derive results in (15) and (16), the second exceptional situation is taken into account.

Gluing the variable 3-digit PNFPW-2 terms.

Recall that to obtain PNFPW-2 in each Peirce-Webb term, the variables on which it depends are combined by the "logical NOR" operation. If the variable takes a zero value in the main basis, then for the Peirce-Webb term it is taken in the direct code, otherwise – in the inverse code (rules from chapter 4).

PNFPW-2 3-digit terms:

$$\left(x_1 \downarrow x_2 \downarrow x_3\right)\downarrow\left(x_1 \downarrow x_2 \downarrow \overline{x_3}\right), \tag{17}$$

correspond to PNFPW-2 binary sets:

$$\left(1_1 \downarrow 1_2 \downarrow 1_3\right)\downarrow\left(1_1 \downarrow 1_2 \downarrow 0_3\right),$$

or PNFPW-2 matrix:

$$\begin{vmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \end{vmatrix}.$$

The operation of gluing the variable 3-digit PNFPW-2 terms in the Peirce-Webb basis takes the following form:

$$\begin{vmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \end{vmatrix} = \begin{vmatrix} 1 & 1 \end{vmatrix} = x_1 + x_2 + \overline{x_1 \downarrow x_2}. \tag{18}$$

To derive the result in (18), the second exceptional situation is taken into account.

Verification of the result (18) is carried out in the main basis (Table 9).

Table 9

Verification of the result $-\ x_1 \downarrow x_2$

x_1	x_2	x_3	PNFPW-2	$\overline{x_1 \downarrow x_2}$	Verification
0	0	0	0	$\overline{0_1 \downarrow 0_2}$	0
0	0	1	0	$\overline{0_1 \downarrow 0_2}$	0

Table 9 demonstrates that the result of gluing the variables $\overline{x_1 \downarrow x_2}$ satisfies the given terms PNFPW-2 (17).

The logical operation of super-gluing the variables.

The combinatorial properties of binary structures PNFPW-1 or PNFPW-2 and PDNF of the Boolean function of the basic basis do not change depending on the logical basis. This makes it possible to implement the logical operation of super-gluing the variables in the Peirce-Webb basis. For 3-digit PNFPW-1 terms, the operation of super-gluing the variables may take the following form, for example:

$$\overline{\left(x_1 \downarrow x_2 \downarrow x_3\right)\downarrow\left(\overline{x_1} \downarrow x_2 \downarrow x_3\right)\downarrow} \;\overline{\downarrow\left(x_1 \downarrow x_2 \downarrow \overline{x_3}\right)\downarrow\left(\overline{x_1} \downarrow x_2 \downarrow \overline{x_3}\right)} = x_2. \tag{19}$$

Proof:

$$\overline{\left(x_1 \downarrow x_2 \downarrow x_3\right)\downarrow\left(\overline{x_1} \downarrow x_2 \downarrow x_3\right)\downarrow\left(x_1 \downarrow x_2 \downarrow \overline{x_3}\right)\downarrow\left(\overline{x_1} \downarrow x_2 \downarrow \overline{x_3}\right)} =$$
$$= \overline{\left(\overline{x_1}\cdot\overline{x_2}\cdot\overline{x_3}\right)\downarrow\left(x_1\cdot\overline{x_2}\cdot\overline{x_3}\right)\downarrow\left(\overline{x_1}\cdot\overline{x_2}\cdot x_3\right)\downarrow\left(x_1\cdot\overline{x_2}\cdot x_3\right)} =$$
$$= \overline{\overline{\left(\overline{x_1}\cdot\overline{x_2}\cdot\overline{x_3}\right)}\cdot\overline{\left(x_1\cdot\overline{x_2}\cdot\overline{x_3}\right)}\cdot\overline{\left(\overline{x_1}\cdot\overline{x_2}\cdot x_3\right)}\cdot\overline{\left(x_1\cdot\overline{x_2}\cdot x_3\right)}} =$$
$$= \overline{\left(x_1 + x_2 + x_3\right)\left(\overline{x_1} + x_2 + x_3\right)\left(x_1 + x_2 + \overline{x_3}\right)\left(\overline{x_1} + x_2 + \overline{x_3}\right)} =$$
$$= \overline{\left(x_1\overline{x_1} + x_1x_2 + x_1x_3 + \overline{x_1}x_2 + x_2x_2 + x_2x_3 + \overline{x_1}x_3 + x_2x_3 + x_3x_3\right)\times}$$
$$\overline{\times\left(x_1\overline{x_1} + x_1x_2 + x_1\overline{x_3} + \overline{x_1}x_2 + x_2x_2 + x_2\overline{x_3} + \overline{x_1}x_3 + x_2x_3 + x_3\overline{x_3}\right)} =$$
$$= \overline{\left(x_1x_2 + x_1x_3 + \overline{x_1}x_2 + x_2 + x_2x_3 + \overline{x_1}x_3 + x_2x_3 + x_3\right)\times}$$
$$\overline{\times\left(x_1x_2 + x_1\overline{x_3} + \overline{x_1}x_2 + x_2 + x_2\overline{x_3} + \overline{x_1}x_3 + x_2\overline{x_3} + \overline{x_3}\right)} = \overline{\left(x_2 + x_3\right)\left(x_2 + \overline{x_3}\right)} = \overline{\overline{x_2}} = \overline{\overline{x_2}} = x_2.$$

The operation of super-gluing the variables for 3-digit PNFPW-1 terms (19) in the Peirce-Webb basis has an illustration of mapping:

$$\begin{vmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{vmatrix} = x_2. \tag{20}$$

To derive the result in (20), the third exceptional situation is taken into account.

For 4-digit PNFPW-1 terms, the operation of super-gluing the variables may take the following form, for example:

$$\overline{\left(x_1 \downarrow x_2 \downarrow \overline{x_3} \downarrow x_4\right)\downarrow\left(x_1 \downarrow x_2 \downarrow \overline{x_3} \downarrow \overline{x_4}\right)\downarrow} \;\overline{\downarrow\left(x_1 \downarrow \overline{x_2} \downarrow \overline{x_3} \downarrow x_4\right)\downarrow\left(x_1 \downarrow \overline{x_2} \downarrow \overline{x_3} \downarrow \overline{x_4}\right)} = \overline{x_1 \downarrow \overline{x_3}}. \tag{21}$$

Proof:

$$\overline{\left(x_1 \downarrow x_2 \downarrow \overline{x_3} \downarrow x_4\right)\downarrow\left(x_1 \downarrow x_2 \downarrow \overline{x_3} \downarrow \overline{x_4}\right)\downarrow} \;\overline{\downarrow\left(x_1 \downarrow \overline{x_2} \downarrow \overline{x_3} \downarrow x_4\right)\downarrow\left(x_1 \downarrow \overline{x_2} \downarrow \overline{x_3} \downarrow \overline{x_4}\right)} =$$
$$= \overline{\left(\overline{x_1}\cdot\overline{x_2}\cdot x_3\cdot\overline{x_4}\right)\downarrow\left(\overline{x_1}\cdot\overline{x_2}\cdot x_3\cdot x_4\right)\downarrow} \;\overline{\downarrow\left(\overline{x_1}\cdot x_2\cdot x_3\cdot\overline{x_4}\right)\downarrow\left(\overline{x_1}\cdot x_2\cdot x_3\cdot x_4\right)} =$$
$$= \overline{\overline{\left(\overline{x_1}\cdot\overline{x_2}\cdot x_3\cdot\overline{x_4}\right)}\cdot\overline{\left(\overline{x_1}\cdot\overline{x_2}\cdot x_3\cdot x_4\right)}\times} \;\overline{\times\overline{\left(\overline{x_1}\cdot x_2\cdot x_3\cdot\overline{x_4}\right)}\cdot\overline{\left(\overline{x_1}\cdot x_2\cdot x_3\cdot x_4\right)}} =$$
$$= \overline{\left(x_1 + x_2 + \overline{x_3} + x_4\right)\left(x_1 + x_2 + \overline{x_3} + \overline{x_4}\right)\times} \;\overline{\times\left(x_1 + \overline{x_2} + \overline{x_3} + x_4\right)\left(x_1 + \overline{x_2} + \overline{x_3} + \overline{x_4}\right)} =$$
$$= \overline{\left(\begin{array}{l} x_1x_1 + x_1x_2 + x_1\overline{x_3} + x_1\overline{x_4} + x_1x_2 + x_2x_2 + x_2\overline{x_3} + x_2\overline{x_4} + \\ + x_1\overline{x_3} + x_2\overline{x_3} + \overline{x_3}\,\overline{x_3} + \overline{x_3}\,\overline{x_4} + x_1x_4 + x_2x_4 + \overline{x_3}x_4 + \overline{x_4}x_4 \end{array}\right)\times}$$
$$\overline{\times\left(\begin{array}{l} x_1x_1 + x_1\overline{x_2} + x_1\overline{x_3} + x_1\overline{x_4} + x_1\overline{x_2} + \overline{x_2}\,\overline{x_2} + \overline{x_2}\,\overline{x_3} + \overline{x_2}\,\overline{x_4} + \\ + x_1\overline{x_3} + \overline{x_2}\,\overline{x_3} + \overline{x_3}\,\overline{x_3} + \overline{x_3}\,\overline{x_4} + x_1x_4 + \overline{x_2}x_4 + \overline{x_3}x_4 + \overline{x_4}x_4 \end{array}\right)} =$$
$$= \overline{\left(x_1 + x_2 + \overline{x_3}\right)\left(x_1 + \overline{x_2} + \overline{x_3}\right)} =$$
$$= \overline{\left(x_1 + \overline{x_3}\right)} = \overline{\overline{x_1}x_3} = x_1 \downarrow \overline{x_3} = \overline{x_1 \downarrow \overline{x_3}}.$$

The operation of super-gluing the variables for 4-digit PNFPW-1 terms (21) in the Peirce-Webb basis has an illustration of mapping:

$$\begin{vmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{vmatrix} = \overline{x_1 \downarrow \overline{x_3}}. \tag{22}$$

To derive the result in (22), the second exceptional situation is taken into account.

Incomplete super-gluing of variables.

The combinatorial properties of the incomplete combinatorial system with the repeated 2-$(n, x/b)$-design in the main basis [15] provide

the rule of incomplete super-gluing of variables in the Peirce-Webb basis.

For dyadic PNFPW-1 terms, the rule of incomplete super-gluing of variables may take the following form, for example:

$$\overline{\left(\overline{x_1}\downarrow\overline{x_2}\right)\downarrow\left(x_1\downarrow\overline{x_2}\right)\downarrow\left(\overline{x_1}\downarrow x_2\right)}=\overline{\overline{x_1}\downarrow\overline{x_2}}. \quad (23)$$

Proof:

$$\begin{aligned}&\overline{\left(\overline{x_1}\downarrow\overline{x_2}\right)\downarrow\left(x_1\downarrow\overline{x_2}\right)\downarrow\left(\overline{x_1}\downarrow x_2\right)}=\\&=\overline{x_1x_2\downarrow\overline{x_1}x_2\downarrow x_1\overline{x_2}}=\\&=\overline{x_1x_2}\cdot\overline{\overline{x_1}x_2}\cdot\overline{x_1\overline{x_2}}=\\&=\left(\overline{x_1}+\overline{x_2}\right)\left(x_1+\overline{x_2}\right)\left(\overline{x_1}+x_2\right)=\\&=\overline{x_1}\,\overline{x_2}=\overline{\overline{x_1}\downarrow\overline{x_2}}.\end{aligned}$$

The operation of incomplete super-gluing of variables for dyadic PNFPW-1 terms (23) in the Peirce-Webb basis has an illustration of mapping:

$$\left|\begin{matrix}0&0\\1&0\\0&1\end{matrix}\right|=\left|\begin{matrix}&0\\0&\end{matrix}\right|=\overline{\overline{x_1}+\overline{x_2}}=x_1x_2=\overline{\overline{x_1}\downarrow\overline{x_2}}. \quad (24)$$

The first matrix of expression (24) represents the unbalanced combinatorial system of 2-(2, 3/4)-design [14]. To derive the result in (24), the second exceptional situation is taken into account.

The operation of incomplete super-gluing of variables for 3-digit PNFPW-1 terms in the Peirce-Webb basis may take the following form, for example:

No.	x_1	x_2	x_3
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0

$$=\left|\begin{matrix}0&&\\&0&\\&&0\end{matrix}\right|=\overline{\overline{x_1}+\overline{x_2}+\overline{x_3}}=$$

$$=x_1x_2x_3=\overline{\overline{x_1}\downarrow\overline{x_2}\downarrow\overline{x_3}}=\overline{\overline{x_1}\downarrow\left(\overline{\overline{x_2}\downarrow\overline{x_3}}\right)}. \quad (25)$$

In the first matrix of expression (25), the following actions are performed: to blocks 0, 1, 2, 3; 0,1,4,5 and to blocks 0, 2, 4, 6, each of which contains a complete combinatorial system of 2-(2, 4)-design, the operation of super-gluing the variables is applied.

Blocks of variables "000", "001", "010", "100" are common to three systems – $\Sigma m(0, 1, 2, 3)$, $\Sigma m(0, 1, 4, 5)$ and $\Sigma m(0, 2, 4, 6)$ and three operations of super-gluing the variables, the localization of which is determined by combinatorial 2-(2, 4)-design systems [16, 17].

The first matrix of expression (25) represents a 2-(3, 7/8)-design combinatorial system [15]. To derive the result in (25), the second exceptional situation is taken into account.

Generalized gluing of variables is carried out using the transformation:

$$\overline{\left(\overline{x_1}\downarrow\overline{x_2}\right)\downarrow\left(\overline{x_1}\downarrow\overline{x_3}\right)\downarrow\left(\overline{x_2}\downarrow x_3\right)}=\overline{\left(\overline{x_1}\downarrow\overline{x_3}\right)\downarrow\left(\overline{x_2}\downarrow x_3\right)}. \quad (26)$$

Proof:

$$\begin{aligned}&\overline{\left(\overline{x_1}\downarrow\overline{x_2}\right)\downarrow\left(\overline{x_1}\downarrow\overline{x_3}\right)\downarrow\left(\overline{x_2}\downarrow x_3\right)}=\\&=\overline{x_1x_2\downarrow x_1x_3\downarrow x_2\overline{x_3}}=\\&=\overline{\overline{x_1x_2}\cdot\overline{x_1x_3}\cdot\overline{x_2x_3}}=\\&=\overline{\left(\overline{x_1}+\overline{x_2}\right)\left(\overline{x_1}+\overline{x_3}\right)\left(\overline{x_2}+x_3\right)}=\\&=\left(\overline{\overline{x_1}+\overline{x_2}}\right)+\left(\overline{\overline{x_1}+\overline{x_3}}\right)+\left(\overline{\overline{x_2}+x_3}\right)=\\&=x_1x_2+x_1x_3+x_2\overline{x_3}=\\&=x_1x_2x_3+x_1x_2\overline{x_3}+x_1x_3+x_2\overline{x_3}=\\&=x_1x_3+x_2\overline{x_3}=\\&=\overline{x_1}\downarrow\overline{x_3}+\overline{x_2}\downarrow x_3=\\&=\overline{\overline{x_1}\downarrow\overline{x_3}\downarrow\overline{x_2}\downarrow x_3}.\end{aligned}$$

Equivalent transformations for the rule of generalized gluing of variables (26) in the Peirce-Webb basis have an illustration of mapping:

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

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

Another variant of the operation of generalized gluing of variables for PNFPW-1:

$$\overline{\left(\overline{x_1}\downarrow\overline{x_3}\right)\downarrow\left(\overline{x_2}\downarrow x_3\right)}=$$

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

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

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

The logical operation of absorbing the variables is reduced to transformations:

$$1.\ x_1\downarrow\left(\overline{x_1}\downarrow x_2\right)=\overline{x_1}. \quad (27)$$

Proof:

$$\begin{aligned}&x_1\downarrow\left(\overline{x_1}\downarrow x_2\right)=x_1\downarrow x_1\overline{x_2}=\\&=\overline{x_1}\,\overline{x_1\overline{x_2}}=\overline{x_1}\left(\overline{x_1}+x_2\right)=\overline{x_1}.\end{aligned}$$

The left-hand side of expression (27) represents the Peirce-Webb function having one term with one variable. According to the first exceptional situation, when entering such a function into the matrix, the variable representing the term must be inverted:

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

During the derivation of the result in (28), the third exceptional situation is applied:

2. $\overline{x_1} \downarrow (x_1 \downarrow x_2) = x_1$.

Proof:

$$\overline{x_1} \downarrow (x_1 \downarrow x_2) = \overline{x_1} \downarrow \overline{x_1} \cdot \overline{x_2} = \overline{\overline{x_1}\,\overline{(\overline{x_1} \cdot \overline{x_2})}} = x_1(x_1 + x_2) = x_1.$$

3. $(x_1 \downarrow x_2) \downarrow (x_1 \downarrow x_2 \downarrow x_3) = \overline{x_1 \downarrow x_2}$.

Proof:

$$(x_1 \downarrow x_2) \downarrow (x_1 \downarrow x_2 \downarrow x_3) = \overline{x_1}\,\overline{x_2} \downarrow \overline{x_1}\,\overline{x_2}\,\overline{x_3} =$$
$$= \overline{(\overline{x_1} \cdot \overline{x_2})} \cdot \overline{(\overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3})} = (x_1 + x_2) \cdot (x_1 + x_2 + x_3) =$$
$$= x_1 + x_2 = \overline{x_1 \downarrow x_2}.$$

$$\begin{vmatrix} 1 & 1 & \\ 1 & 1 & 1 \end{vmatrix} = \begin{vmatrix} 1 & 1 & \\ & & \end{vmatrix} = \overline{x_1 \downarrow x_2}. \qquad (29)$$

In the derivation of the result in (29), the second exceptional situation is applied:

4. $(x_1 \downarrow x_2) \downarrow (x_1 \downarrow x_2 \downarrow x_3 \downarrow x_4) = \overline{x_1 \downarrow x_2}$.

Proof:

$$(x_1 \downarrow x_2) \downarrow (x_1 \downarrow x_2 \downarrow x_3 \downarrow x_4) =$$
$$= \overline{x_1} \cdot \overline{x_2} \downarrow \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4} =$$
$$= \overline{(\overline{x_1} \cdot \overline{x_2})} \cdot \overline{(\overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4})} =$$
$$= (x_1 + x_2) \cdot (x_1 + x_2 + x_3 + x_4) =$$
$$= x_1 + x_2 = \overline{x_1 \downarrow x_2}.$$

$$\begin{vmatrix} 1 & 1 & & \\ 1 & 1 & 1 & 1 \end{vmatrix} = \begin{vmatrix} 1 & 1 & & \\ & & & \end{vmatrix} = \overline{x_1 \downarrow x_2}. \qquad (30)$$

In the derivation of the result in (30), the second exceptional situation is applied.

The logical operation of semi-gluing the variables is reduced to the following transformations:

$$(\overline{x_1} \downarrow x_2) \downarrow (x_1 \downarrow x_2 \downarrow x_3) = \overline{x_2 \downarrow (x_1 \downarrow \overline{x_3})}, \qquad (31)$$

Proof of rule (31) for PNFPW-2:

$$(\overline{x_1} \downarrow x_2) \downarrow (x_1 \downarrow x_2 \downarrow x_3) = x_1 \cdot \overline{x_2} \downarrow \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} =$$
$$= \overline{(x_1 \cdot \overline{x_2})} \cdot \overline{(\overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3})} = (\overline{x_1} + x_2)(x_1 + x_2 + x_3) =$$
$$= x_1\overline{x_1} + \overline{x_1}x_2 + \overline{x_1}x_3 + x_1x_2 + x_2x_2 + x_2x_3 =$$
$$= x_2 + \overline{x_1}x_3 = x_2 + x_1 \downarrow \overline{x_3} =$$
$$= \overline{\overline{x_2} \downarrow (x_1 \downarrow \overline{x_3})}.$$

The rule of semi-gluing the variables (31) has an illustration of mapping:

$$\begin{vmatrix} 0 & 1 & \\ 1 & 1 & 1 \end{vmatrix} = \begin{vmatrix} 0 & 1 & \\ & 1 & 1 \end{vmatrix} =$$
$$= (\overline{x_1} + x_2)(x_2 + x_3) = \overline{x_1}x_2 + \overline{x_1}x_3 + x_2x_2 + x_2x_3 =$$
$$= x_2 + \overline{x_1}x_3 = x_2 + x_1 \downarrow \overline{x_3} =$$
$$= \overline{\overline{x_2} \downarrow (x_1 \downarrow \overline{x_3})}. \qquad (32)$$

During the derivation of the result in (32), the first exceptional situation is applied.

Proof of the rule:

$$x_1 \downarrow (x_1 \downarrow x_2) = x_1 \downarrow \overline{x_2}. \qquad (33)$$

for PNFPW-2, the following transformations were carried out:

$$x_1 \downarrow (x_1 \downarrow x_2) = x_1 \downarrow \overline{x_1} \cdot \overline{x_2} = \overline{x_1} \cdot \overline{(\overline{x_1} \cdot \overline{x_2})} =$$
$$= \overline{x_1}(x_1 + x_2) = \overline{x_1}x_2 = x_1 \downarrow \overline{x_2}.$$

Rule (33) has a mapping illustration:

$$x_1 \downarrow (x_1 \downarrow x_2) = \begin{vmatrix} 0 & \\ 1 & 1 \end{vmatrix} = \begin{vmatrix} 0 & \\ & 1 \end{vmatrix} = x_1 \downarrow \overline{x_2}. \qquad (34)$$

When deriving the result in (34), the first exceptional situation is applied.

Example 7. Use a non-standard system to simplify the function $f(x_1, x_2, x_3, x_4)$ in the Peirce-Webb basis given by the Veitch diagram (Fig. 6) [18].

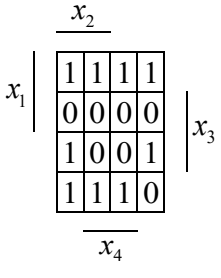

Fig. 6. The function $f(x_1, x_2, x_3, x_4)$, given by the Veitch diagram

Solution.

The simplification of the PCNF function (Fig. 6) takes the following form:

$f_{\text{MCNF}} =$

No.	x_1	x_2	x_3	x_4
0	0	0	0	0
3	0	0	1	1
7	0	1	1	1
10	1	0	1	0
11	1	0	1	1
14	1	1	1	0
15	1	1	1	1

$=$

x_1	x_2	x_3	x_4
1	1	1	1
1	1	0	0
1	0	0	0
0	1	0	1
0	1	0	0
0	0	0	1
0	0	0	0

$=$

x_1	x_2	x_3	x_4
1	1	1	1
		0	0
0		0	

$=$

$$= (\overline{x_1} + \overline{x_3})(\overline{x_3} + \overline{x_4})(x_1 + x_2 + x_3 + x_4). \qquad (35)$$

According to Nelson's method, in the first matrix of expression (35), the variables are inverted [19]. In the second matrix of expression (35), the following actions are performed: the operation of supergluing the variables is applied to blocks 3, 7, 11, 15 and to blocks 10, 11, 14, 15, since these blocks form intervals of the Boolean space containing combinatorial 2 -(2, 4)-design systems. The results of logical

operations of super-gluing the variables are written to the third matrix.

The eleventh block of variables "0100" and the fifteenth "0000" are common to two systems – $\Sigma m(3, 7, 11, 15)$ and $\Sigma m(10, 11, 12, 13)$ and two operations of supergluing the variables, the location of which is determined by combinatorial 2-(2, 4)-design systems [16, 17].

As a result, the minimal function in CNF was obtained in one step.

MCNF (35) coincides with [18].

For MCNF (35), PNFPW-2 is written, using the distributive law of the 2nd kind (distributiveness of disjunction relative to conjunction), factorization in CNF:

$$x_1+(x_2\cdot x_3)=(x_1+x_2)(x_1+x_3).$$

$$\begin{aligned}
&F_{\text{CNFPW-2}}=\\
&=(\overline{x_1}+\overline{x_3})(\overline{x_3}+\overline{x_4})(x_1+x_2+x_3+x_4)=\\
&=(\overline{x_3}+(\overline{x_1})(\overline{x_4}))(x_1+x_2+x_3+x_4)=\\
&=(\overline{x_3}+\overline{x_1}\overline{x_4})(x_1+x_2+x_3+x_4)=\\
&=(\overline{x_3}+(x_1\downarrow x_4))\overline{(x_1\downarrow x_2\downarrow x_3\downarrow x_4)}=\\
&=\overline{(\overline{x_3}\downarrow(x_1\downarrow x_4))}\overline{(x_1\downarrow x_2\downarrow x_3\downarrow x_4)}=\\
&=(\overline{x_3}\downarrow(x_1\downarrow x_4))\downarrow(x_1\downarrow x_2\downarrow x_3\downarrow x_4)=\\
&=(\overline{x_3}\downarrow(x_1\downarrow x_4))\downarrow((\overline{x_1\downarrow x_2})\downarrow(\overline{x_3\downarrow x_4})).
\end{aligned}$$

PNFPW-2 takes the following form:

$$F_{\text{DNFPW-2}}=(\overline{x_3}\downarrow(x_1\downarrow x_4))\downarrow((\overline{x_1\downarrow x_2})\downarrow(\overline{x_3\downarrow x_4})). \qquad (36)$$

The realization price (36) $k_\theta / k_l / k_{in}=4/7/3$, where k_θ, k_l, k_{in} are the number of n-digit "logical NOR" operations, literals, and inversions, respectively. The price of the implementation of function $F_{\text{DNFPW-2}}=\overline{(\overline{x_1}\downarrow\overline{x_3})\downarrow(\overline{x_3}\downarrow\overline{x_4})}\downarrow$ $\downarrow(\overline{x_1\downarrow x_2})\downarrow(\overline{x_3\downarrow x_4})$ [18] is $k_\theta / k_l / k_{in}=4/8/7$.

5. 4. Simplification of Peirce-Webb functions on the complete truth table

Simplification of DNF or CNF of Boolean functions is carried out on the corresponding sets of variables in the truth table. It is important to note that in order to obtain an optimal result, from the point of view of practical implementation in digital technology, it is advisable to perform minimization on two normal forms – DNF and CNF. To this end, you need to use the complete truth table of the given function. The minimum function should be chosen based on the results of minimization of two normal forms – DNF and CNF.

Example 8. Using a non-standard system, simplify the function $f(x_1, x_2, x_3, x_4)$ on the complete truth table given in the canonical form [20]:

$$F(x_1,x_2,x_3,x_4)=\Sigma(2,3,4,6,7,8,9,10,11,15). \qquad (37)$$

In expression (37), Σ defines the minterms at which the function $f(x_1, x_2, x_3, x_4)$ returns "1" at the output.

The minimum function is chosen based on the results of the simplification of two perfect normal forms – PNFPW-1 and PNFPW-2.

Solution.

The PDNF simplification of function (37) takes the following form:

$F_{\text{MDNF}}=$

No.	x_4	x_3	x_2	x_1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
15	1	1	1	1

$=$

x_4	x_3	x_2	x_1
0		1	
0	1		0
1	0		
		1	1

$=$

$$=\overline{x_4}x_2+\overline{x_4}x_3\overline{x_1}+x_4\overline{x_3}+x_2x_1. \qquad (38)$$

The price of realization of MDNF (38) $k_\theta / k_l / k_{in}=4/9/4$, where k_θ, k_l, k_{in} are the number of conjunct terms, literals, and inversions, respectively.

For MDNF (38), PNFPW-1 is recorded:

$$\begin{aligned}
&F_{\text{MNFPW-1}}=\\
&=\overline{x_4}x_2+\overline{x_4}x_3\overline{x_1}+x_4\overline{x_3}+x_2x_1=\\
&=\overline{x_4}(x_2+x_3\overline{x_1})+x_4\overline{x_3}+x_2x_1=\\
&=\overline{x_4}(x_2+(\overline{x_3}\downarrow x_1))+(\overline{x_4}\downarrow x_3)+(\overline{x_2}\downarrow\overline{x_1})=\\
&=\overline{x_4}(\overline{x_2\downarrow(\overline{x_3}\downarrow x_1)})+(\overline{x_4}\downarrow x_3)+(\overline{x_2}\downarrow\overline{x_1})=\\
&=(x_4\downarrow(x_2\downarrow(\overline{x_3}\downarrow x_1)))+(\overline{x_4}\downarrow x_3)+(\overline{x_2}\downarrow\overline{x_1})=\\
&=(x_4\downarrow(x_2\downarrow(\overline{x_3}\downarrow x_1)))+(\overline{x_4}\downarrow x_3)+(\overline{x_2}\downarrow\overline{x_1})=\\
&=\overline{(x_4\downarrow(x_2\downarrow(\overline{x_3}\downarrow x_1)))\downarrow(\overline{x_4}\downarrow x_3)\downarrow(\overline{x_2}\downarrow\overline{x_1})}.
\end{aligned}$$

PNFPW-1 takes the following form:

$$F_{\text{DNFPW-1}}=\overline{(x_4\downarrow(x_2\downarrow(\overline{x_3}\downarrow x_1)))\downarrow(\overline{x_4}\downarrow x_3)\downarrow(\overline{x_2}\downarrow\overline{x_1})}. \qquad (39)$$

DCNF simplification of function (37) takes the following form:

$F_{\text{MCNF}}=$

No.	x_4	x_3	x_2	x_1
0	0	0	0	0
1	0	0	0	1
5	0	1	0	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0

$=$

x_4	x_3	x_2	x_1
1	1	1	1
1	1	1	0
1	0	1	0
0	0	1	1
0	0	1	0
0	0	0	1

$=$

x_4	x_3	x_2	x_1
1	1	1	
	0	1	0
0	0		1

$=$

$$=(x_4+x_3+x_2)(\overline{x_3}+x_2+\overline{x_1})(\overline{x_4}+\overline{x_3}+x_1). \qquad (40)$$

Realization price of MKNF (40) $k_\theta / k_l / k_{in} = 3 / 9 / 4$.
For MCNF (40), PNFPW-2 is recorded:

$$F_{\text{DNFPW–2}} =$$
$$= (x_4 + x_3 + x_2)(\overline{x_3} + x_2 + \overline{x_1})(\overline{x_4} + \overline{x_3} + x_1) =$$
$$= (x_4 + x_3 + x_2)(\overline{x_3} + (x_2 + \overline{x_1})(\overline{x_4} + x_1)) =$$
$$= (\overline{x_4 \downarrow x_3 \downarrow x_2})(\overline{x_3} + (\overline{(x_2 \downarrow \overline{x_1})(\overline{x_4} \downarrow x_1)})) =$$
$$= (\overline{x_4 \downarrow x_3 \downarrow x_2})(\overline{x_3} + ((x_2 \downarrow \overline{x_1}) \downarrow (\overline{x_4} \downarrow x_1))) =$$
$$= (\overline{x_4 \downarrow x_3 \downarrow x_2})(\overline{\overline{x_3} \downarrow ((x_2 \downarrow \overline{x_1}) \downarrow (\overline{x_4} \downarrow x_1))}) =$$
$$= (x_4 \downarrow x_3 \downarrow x_2) \downarrow (\overline{x_3} \downarrow ((x_2 \downarrow \overline{x_1}) \downarrow (\overline{x_4} \downarrow x_1))).$$

PNFPW-2 takes the following form:

$$F_{\text{DNFPW-2}} =$$
$$= (x_4 \downarrow x_3 \downarrow x_2) \downarrow (\overline{x_3} \downarrow ((x_2 \downarrow \overline{x_1}) \downarrow (\overline{x_4} \downarrow x_1))). \quad (41)$$

Due to the smaller number of inversions in PNFPW-2 (41), the latter has a simpler logical structure (Fig. 7, *b*) compared to PNFPW-1 (39) (Fig. 7, *a*).

Verification of PNFPW-1 and PNFPW-2 is given in Table 10.

Table 10 demonstrates that $F_{\text{DNFPW-1}}$ and $F_{\text{DNFPW-2}}$ pass the verification.

Based on the results of simplifying the two normal forms – PNFPW-1 and PNFPW-2, we choose PNFPW-2 as the minimum (41).

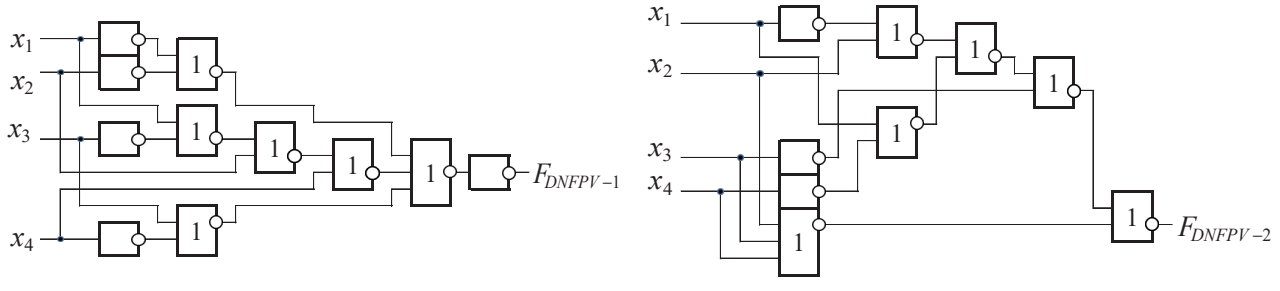

Fig. 7. Implementation of the minimum logical function $f(x_1, x_2, x_3, x_4)$ (37) in the Peirce-Webb basis by a combinational scheme: *a* – PNFPW-1; *b* – PNFPW-2

Table 10

Verification of functions $F_{\text{DNFPW-1}} = \overline{(x_4 \downarrow (x_2 \downarrow (\overline{x_3} \downarrow x_1))) \downarrow (\overline{x_4} \downarrow x_3) \downarrow (\overline{x_2} \downarrow \overline{x_1})}$,
$F_{\text{DNFPW-2}} = (x_4 \downarrow x_3 \downarrow x_2) \downarrow (\overline{x_3} \downarrow ((x_2 \downarrow \overline{x_1}) \downarrow (\overline{x_4} \downarrow x_1)))$

No. of entry	x_1	x_2	x_3	x_4	$F(x_1, x_2, x_3, x_4)$	$F_{\text{DNFPW-1}}$	$F_{\text{DNFPW-2}}$
2	0	0	1	0	1	1	1
3	0	0	1	1	1	1	1
4	0	1	0	0	1	1	1
6	0	1	1	0	1	1	1
7	0	1	1	1	1	1	1
8	1	0	0	0	1	1	1
9	1	0	0	1	1	1	1
10	1	0	1	0	1	1	1
11	1	0	1	1	1	1	1
15	1	1	1	1	1	1	1
No. of entry	x_1	x_2	x_3	x_4	$F(x_1, x_2, x_3, x_4)$	$F_{\text{DNFPW-1}}$	$F_{\text{DNFPW-2}}$
0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0
5	0	1	0	1	0	0	0
12	1	1	0	0	0	0	0
13	1	1	0	1	0	0	0
14	1	1	1	0	0	0	0

5. 5. Comparison with the matrix algorithm, the methods for algebraic bi-decomposition and bitwise partitioning of the set of output conjuncterms

Boolean functions can be expressed in algebraic form through the half-tensor product of matrices [21]. As a result, this gives a matrix algorithm for simplifying logical functions [22].

Example 9. Use a non-standard system to simplify the function $f(w, x, y, z)$ given in the canonical form [22]:

$$f(w,x,y,z)=w\bar{x}y\bar{z}+w\bar{x}yz+w\bar{x}\bar{y}\bar{z}+ \\ +\bar{w}xyz+\bar{w}x\bar{y}z+\bar{w}\bar{x}yz+\bar{w}\bar{x}\bar{y}z. \quad (42)$$

Solution:

For convenience, the variables in expression (42) have been replaced:

$$w\to x_1; x\to x_2; y\to x_3; z\to x_4.$$

Function with new variables:

$$f=x_1\bar{x}_2x_3\bar{x}_4+x_1\bar{x}_2x_3x_4+x_1\bar{x}_2\bar{x}_3\bar{x}_4+ \\ +\bar{x}_1x_2x_3x_4+\bar{x}_1x_2\bar{x}_3x_4+\bar{x}_1\bar{x}_2x_3x_4+\bar{x}_1\bar{x}_2\bar{x}_3x_4. \quad (43)$$

Simplification of the function $f(x_1, x_2, x_3, x_4)$ (43) in the disjunctive normal form (DNF):

$f_{\text{MDNF}}(x_1,x_2,x_3,x_4)=$

No.	x_1	x_2	x_3	x_4
1	0	0	0	1
3	0	0	1	1
5	0	1	0	1
7	0	1	1	1
10	1	0	1	0
11	1	0	1	1
14	1	1	1	0

$=$

x_1	x_2	x_3	x_4
0			1
1	0	1	
1		1	0

$=$

$$=\bar{x}_1x_4+x_1\bar{x}_2x_3+x_1x_3\bar{x}_4. \quad (44)$$

The result of simplification (44) of the function $f(x_1, x_2, x_3, x_4)$ coincides with [22], but the non-standard system is significantly simpler in terms of procedure.

In the general case, the search for the minimum function is carried out on the complete truth table and on various logical bases.

Simplification of the function $f(x_1, x_2, x_3, x_4)$ (43) in the conjunctive normal form (CNF):

$f_{\text{MCNF}}(x_1,x_2,x_3,x_4)=$

No.	x_1	x_2	x_3	x_4
0	0	0	0	0
2	0	0	1	0
4	0	1	0	0
6	0	1	1	0
8	1	0	0	0
9	1	0	0	1
12	1	1	0	0
13	1	1	0	1
15	1	1	1	1

$=$

x_1	x_2	x_3	x_4
1	1	1	1
1	1	0	1
1	0	1	1
1	0	0	1
0	1	1	1
0	1	1	0
0	0	1	1
0	0	1	0
0	0	0	0

$=$

x_1	x_2	x_3	x_4
1			1
0		1	
0	0		0

$$=(x_1+x_4)(\bar{x}_1+x_3)(\bar{x}_1+\bar{x}_2+\bar{x}_4). \quad (45)$$

MDNF (44) and MCNF (45) demonstrate that MCNF (45) contains one less literal.

For MCNF (45), PNFPW-2 is recorded:

$$f_{\text{DNFPW-2}}(x_1,x_2,x_3,x_4)= \\ =(x_1+x_4)(\bar{x}_1+x_3)(\bar{x}_1+\bar{x}_2+\bar{x}_4)= \\ =(x_1+x_4)(\bar{x}_1+(x_3)(\bar{x}_2+\bar{x}_4))= \\ =(\overline{x_1\downarrow x_4})(\bar{x}_1+(x_3)(\overline{\bar{x}_2\downarrow\bar{x}_4}))= \\ =(\overline{x_1\downarrow x_4})(\bar{x}_1+(\bar{x}_3\downarrow(\bar{x}_2\downarrow\bar{x}_4)))= \\ =(\overline{x_1\downarrow x_4})(\overline{\bar{x}_1\downarrow(\bar{x}_3\downarrow(\bar{x}_2\downarrow\bar{x}_4))})= \\ =(x_1\downarrow x_4)\downarrow(\bar{x}_1\downarrow(\bar{x}_3\downarrow(\bar{x}_2\downarrow\bar{x}_4))).$$

Therefore, PNFPW-2 contains two literals less than MDNF (44).

There are well-known examples of the use of bi-decomposition methods to reduce the delay of signals by combinational circuits [23, 24] and during the synthesis of FPGA-based circuits [25].

Example 10. Using a non-standard system, simplify the partially defined Boolean function $f(x_1, x_2, x_3, x_4, x_5)$ given by matrices M^1, M^0 [26] (through numbering is used for the matrix rows):

$M^1=$

x_1	x_2	x_3	x_4	x_5	
–	–	1	1	–	1
0	–	1	–	–	2
–	1	–	1	–	3
0	0	–	–	–	4
–	1	0	–	0	5
1	1	–	–	1	6
1	–	0	0	1	7

;

$M^0=$

x_1	x_2	x_3	x_4	x_5	
0	1	0	0	1	8
1	–	1	0	0	9
1	0	0	1	–	10
1	0	1	0	–	11

. (46)

Solution.

The ternary vector – –11– in M^1 is considered as an interval of the Boolean space $I(\alpha, \beta)$ of five Boolean variables, which is formed by the following terms:

$$I(\alpha,\beta)=I(00110,11111)=\begin{matrix}00110\\00111\\01110\\01111\\10110\\10111\\11110\\11111\end{matrix}.$$

The set of internal components of the interval $I(\alpha, \beta)$ form a combinatorial system of 2-(n, b)-design. For a given example, 2-(3, 8)-design.

The terms of the given partially defined function (46) are obtained through the interval $I(\alpha, \beta)$ constructed in the Boolean space for all ternary vectors that are placed in the matrices M^1, M^0 (46). It is important to note that during the generation of terms, the logical law of idempotency must be applied, since part of the terms of the function will be repeated.

The truth table of the partially defined function $f(x_1, x_2, x_3, x_4, x_5)$ (46) takes the following form (Table 11).

Table 11

Truth table of the partially defined function $f(x_1, x_2, x_3, x_4, x_5)$ (46)

No. of entry	x_1	x_2	x_3	x_4	x_5	F
0	0	0	0	0	0	1
1	0	0	0	0	1	1
2	0	0	0	1	0	1
3	0	0	0	1	1	1
4	0	0	1	0	0	1
5	0	0	1	0	1	1
6	0	0	1	1	0	1
7	0	0	1	1	1	1
8	0	1	0	0	0	1
9	0	1	0	0	1	0
10	0	1	0	1	0	1
11	0	1	0	1	1	1
12	0	1	1	0	0	1
13	0	1	1	0	1	1
14	0	1	1	1	0	1
15	0	1	1	1	1	1
16	1	0	0	0	0	–
17	1	0	0	0	1	1
18	1	0	0	1	0	0
19	1	0	0	1	1	0
20	1	0	1	0	0	0
21	1	0	1	0	1	0
22	1	0	1	1	0	1
23	1	0	1	1	1	1
24	1	1	0	0	0	1
25	1	1	0	0	1	1
26	1	1	0	1	0	1
27	1	1	0	1	1	1
28	1	1	1	0	0	0
29	1	1	1	0	1	1
30	1	1	1	1	0	1
31	1	1	1	1	1	1

Simplification of the function $f(x_1, x_2, x_3, x_4, x_5)$ (46) in the disjunctive normal form (DNF):

$$f_{\text{MDNF}}(x_1,x_2,x_3,x_4,x_5) =$$

No.	x_1	x_2	x_3	x_4	x_5	F
0	0	0	0	0	0	1
1	0	0	0	0	1	1
2	0	0	0	1	0	1
3	0	0	0	1	1	1
4	0	0	1	0	0	1
5	0	0	1	0	1	1
6	0	0	1	1	0	1
7	0	0	1	1	1	1
8	0	1	0	0	0	1
10	0	1	0	1	0	1
11	0	1	0	1	1	1
12	0	1	1	0	0	1
13	0	1	1	0	1	1
14	0	1	1	1	0	1
15	0	1	1	1	1	1
17	1	0	0	0	1	1
22	1	0	1	1	0	1
23	1	0	1	1	1	1
24	1	1	0	0	0	1
25	1	1	0	0	1	1
26	1	1	0	1	0	1
27	1	1	0	1	1	1
29	1	1	1	0	1	1
30	1	1	1	1	0	1
31	1	1	1	1	1	1
16	1	0	0	0	0	–

=

x_1	x_2	x_3	x_4	x_5
0	0			
		0	0	0
	1		1	
0		1		
	0	0	0	
		1	1	
1	1			1

$$= \overline{x_1}\,\overline{x_2} + \overline{x_1}x_3 + x_2x_4 + x_3x_4 +$$
$$+ x_1x_2x_5 + \overline{x_2}\,\overline{x_3}\,\overline{x_4} + \overline{x_3}\,\overline{x_4}\,\overline{x_5} =$$
$$= \overline{x_1}\left(\overline{x_2} + x_3\right) + x_4\left(x_2 + x_3\right) +$$
$$+ x_1x_2x_5 + \overline{x_3}\,\overline{x_4}\left(\overline{x_2} + \overline{x_5}\right). \qquad (47)$$

In the first matrix of expression (47), the following actions are performed: to blocks 0, 1, 2, 3, 4, 5, 6, 7; 10, 11, 14, 13, 26, 27, 30, 31; 4, 5, 6, 7, 12, 13, 14, 15; 6, 7, 14, 15, 22, 23, 30, 31, which form intervals of the Boolean space containing the complete combinatorial system of 2-(3, 8)-design, and to blocks 0, 8, 24, 16; 0, 1, 17, 16; 25, 27, 29, 31, which form the interval of the Boolean space, containing the complete combinatorial system of 2-(2, 4)-design, the operation of super-gluing the variables is applied.

The zero block of variables "00000" is common to three systems – $\Sigma m(0, 1, 2, 3, 4, 5, 6, 7)$, $\Sigma m(0, 1, 17, 16)$, $\Sigma m(0, 8, 24, 16)$ and three operations of super-gluing the variables, the location of which is determined by the combinatorial systems of 2-(3, 8)-design and 2-(2, 4)-design [16, 17].

The first block of variables "00001" is common to two systems – $\Sigma m(0, 1, 2, 3, 4, 5, 6, 7)$, $\Sigma m(0, 1, 17, 16)$ and two operations of super-gluing the variables, location which is determined by the combinatorial systems of 2-(3, 8)-design and 2-(2, 4)-design.

The sixth and seventh blocks of variables "00110", "00111" are common to the two systems – $\Sigma m(0, 1, 2, 3, 4, 5, 6, 7)$, $\Sigma m(6, 7, 14, 15, 22, 23, 30, 31)$ and two operations of super-gluing the variables, the location of which is determined by combinatorial systems of 2-(3, 8)-design.

The eighth block of variables "01000" is common to two systems – $\Sigma m(0, 8, 24, 16)$, $\Sigma m(25, 27, 29, 31)$ and two operations of super-gluing the variables, the location of which is determined by combinatorial systems of 2-(2, 4)-design.

The fourteenth and fifteenth blocks of variables "01110", "01111" are common to the two systems – $\Sigma m(4, 5, 6, 7, 12, 13, 14, 15)$, $\Sigma m(10, 11, 14, 13, 26, 27, 30, 31)$ and two operations of super-gluing the variables, the location of which is determined by combinatorial systems of 2-(3, 8)-design.

As a result, the minimal function of the basic basis is obtained in one step:

$$f_{\text{MDNF}}(x_1,x_2,x_3,x_4,x_5)=\overline{x}_1\left(\overline{x}_2+x_3\right)+ \\ +x_4\left(x_2+x_3\right)+x_1x_2x_5+\overline{x}_3\overline{x}_4\left(\overline{x}_2+\overline{x}_5\right). \quad (48)$$

For MDNF (48), PNFPW-1 is recorded:

$$\overline{x}_1\left(\overline{x}_2+x_3\right)+x_4\left(x_2+x_3\right)+x_1x_2x_5+\overline{x}_3\overline{x}_4\left(\overline{x}_2+\overline{x}_5\right)= \\ =\overline{x}_1\left(\overline{x}_2\downarrow x_3\right)+x_4\left(x_2\downarrow x_3\right)+\left(\overline{x}_1\downarrow\overline{x}_2\downarrow\overline{x}_5\right)+\overline{x}_3\overline{x}_4\left(\overline{x}_2\downarrow\overline{x}_5\right)= \\ =\overline{\overline{x}_1}\downarrow\left(\overline{\overline{x}_2\downarrow x_3}\right)+\overline{x}_4\downarrow\left(\overline{x_2\downarrow x_3}\right)+\left(\overline{x}_1\downarrow\overline{x}_2\downarrow\overline{x}_5\right)+ \\ +\left(\overline{\overline{x}_3}\downarrow\overline{\overline{x}_4}\right)\left(\overline{x}_2\downarrow\overline{x}_5\right)=\left(x_1\downarrow\left(\overline{x}_2\downarrow x_3\right)\right)+\left(\overline{x}_4\downarrow\left(x_2\downarrow x_3\right)\right)+ \\ +\left(\overline{x}_1\downarrow\overline{x}_2\downarrow\overline{x}_5\right)+\left(\left(x_3\downarrow x_4\right)\left(\overline{x}_2\downarrow\overline{x}_5\right)\right)=\left(x_1\downarrow\left(\overline{x}_2\downarrow x_3\right)\right)+ \\ +\left(\overline{x}_4\downarrow\left(x_2\downarrow x_3\right)\right)+\left(\overline{x}_1\downarrow\overline{x}_2\downarrow\overline{x}_5\right)+\left(\overline{x_3\downarrow x_4}\right)\downarrow\left(\overline{\overline{x}_2\downarrow\overline{x}_5}\right)= \\ =\left(x_1\downarrow\left(\overline{x}_2\downarrow x_3\right)\right)+\left(\overline{x}_4\downarrow\left(x_2\downarrow x_3\right)\right)+\left(\overline{x}_1\downarrow\overline{x}_2\downarrow\overline{x}_5\right)+ \\ +\left(\left(\overline{x_3\downarrow x_4}\right)\downarrow\left(\overline{x}_2\downarrow\overline{x}_5\right)\right)=\overline{\left(x_1\downarrow\left(\overline{x}_2\downarrow x_3\right)\right)\downarrow\left(\overline{x}_4\downarrow\left(x_2\downarrow x_3\right)\right)}\downarrow \\ \overline{\downarrow\left(\overline{x}_1\downarrow\overline{x}_2\downarrow\overline{x}_5\right)\downarrow\left(\left(\overline{x_3\downarrow x_4}\right)\downarrow\left(\overline{x}_2\downarrow\overline{x}_5\right)\right)}.$$

PNFPW-1 takes the following form:

$$F_{\text{DNFPW-1}}=\overline{\left(x_1\downarrow\left(\overline{x}_2\downarrow x_3\right)\right)\downarrow\left(\overline{x}_4\downarrow\left(x_2\downarrow x_3\right)\right)}\downarrow \\ \overline{\downarrow\left(\overline{x}_1\downarrow\overline{x}_2\downarrow\overline{x}_5\right)\downarrow\left(\left(\overline{x_3\downarrow x_4}\right)\downarrow\left(\overline{x}_2\downarrow\overline{x}_5\right)\right)}. \quad (49)$$

PNFPW-1 (49) provides the corresponding values of one and zero for the matrices M^1, M^0. It is important to note that the "logical NOR" operation $\overline{x}_2\downarrow\overline{x}_5$ occurs twice in PNFPW-1 (49). This provides an additional resource for simplifying the logic circuit implementing PNFPW-1 (49) (Fig. 8).

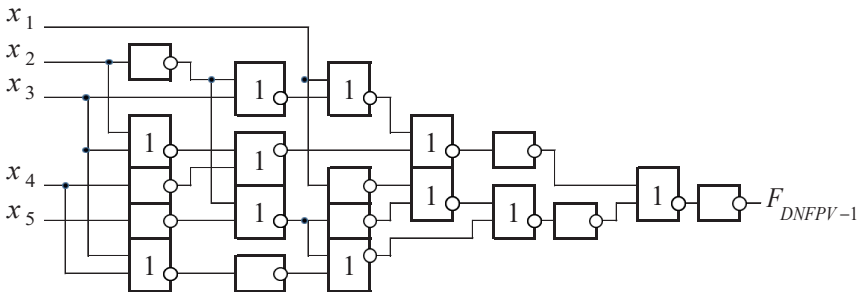

Fig. 8. The logic circuit that implements PNFPW-1 (49): the complexity of the circuit is 11 2-input logic elements "Logical NOR", 9 inverters, a total of 20 logic elements, the depth of the circuit is 8 logic elements

Simplification of the function $f(x_1, x_2, x_3, x_4, x_5)$ (46) in the conjunctive normal form (CNF):

$f_{\text{MCNF}}(x_1,x_2,x_3,x_4,x_5)=$

No.	x_1	x_2	x_3	x_4	x_5	F
9	0	1	0	0	1	0
18	1	0	0	1	0	0
19	1	0	0	1	1	0
20	1	0	1	0	0	0
21	1	0	1	0	1	0
28	1	1	1	0	0	0
16	1	0	0	0	0	–

=

	x_1	x_2	x_3	x_4	x_5	F
9	1	0	1	1	0	0
18	0	1	1	0	1	0
19	0	1	1	0	0	0
20	0	1	0	1	1	0
21	0	1	0	1	0	0
28	0	0	0	1	1	0
16	0	1	1	1	1	–

=

x_1	x_2	x_3	x_4	x_5
1	0	1	1	0
0	1	1	0	
0	1	0	1	
0		0	1	1

$$=\left(x_1+\overline{x}_2+x_3+x_4+\overline{x}_5\right)\left(\overline{x}_1+x_2+x_3+\overline{x}_4\right)\times \\ \times\left(\overline{x}_1+x_2+\overline{x}_3+x_4\right)\left(\overline{x}_1+\overline{x}_3+x_4+x_5\right). \quad (50)$$

According to Nelson's method, in the first matrix of expression (50), the variables are inverted [19]. In the second matrix of expression (50), the following actions are performed: to blocks 18, 19; 20, 21; 20, 28, which form the intervals of the Boolean space containing combinatorial systems of 2-(1, 2)-design, the operation of simple gluing of variables is applied.

It should be noted that the undefined set of variables 16 in the second matrix of expression (50) is not written to the third matrix of expression (50) because the set 16 does not participate in the further simplification of the partially defined function $f(x_1, x_2, x_3, x_4, x_5)$ (46) [16]. This ultimately reduces the complexity of simplifying the function $f(x_1, x_2, x_3, x_4, x_5)$ (46) in the conjunctive normal form (CNF).

The twentieth block of variables "01011" is common to two systems – $\Sigma m(20, 21)$, $\Sigma m(20, 28)$ and two operations of simple

gluing of variables, the location of which is determined by combinatorial 2-(1, 2)-design systems [16, 17].

As a result, in one step, the minimum function in CNF is obtained:

$$f_{\text{MCNF}}(x_1,x_2,x_3,x_4,x_5)=\left(x_1+\overline{x_2}+x_3+x_4+\overline{x_5}\right)\times$$
$$\times\left(\overline{x_1}+x_2+x_3+\overline{x_4}\right)\left(\overline{x_1}+x_2+\overline{x_3}+x_4\right)\left(\overline{x_1}+\overline{x_3}+x_4+x_5\right). \quad (51)$$

PNFPW-2 is recorded for MCNF (51):

$$\left(x_1+\overline{x_2}+x_3+x_4+\overline{x_5}\right)\left(\overline{x_1}+x_2+x_3+\overline{x_4}\right)\times$$
$$\times\left(\overline{x_1}+x_2+\overline{x_3}+x_4\right)\left(\overline{x_1}+\overline{x_3}+x_4+x_5\right)=$$
$$=\left(\left(\overline{x_1}+x_2\right)+\left(x_3+\overline{x_4}\right)\left(\overline{x_3}+x_4\right)\right)\times$$
$$\times\left(x_4+\left(x_1+\overline{x_2}+x_3+\overline{x_5}\right)\left(\overline{x_1}+\overline{x_3}+x_5\right)\right)=$$
$$=\left(\left(\overline{\overline{x_1}\downarrow x_2}\right)+\left(\overline{x_3\downarrow\overline{x_4}}\right)\left(\overline{\overline{x_3}\downarrow x_4}\right)\right)\times$$
$$\times\left(x_4+\left(\overline{x_1\downarrow\overline{x_2}\downarrow x_3\downarrow\overline{x_5}}\right)\left(\overline{\overline{x_1}\downarrow\overline{x_3}\downarrow x_5}\right)\right)=$$
$$=\left(\left(\overline{\overline{\overline{x_1}\downarrow x_2}}\right)+\left(\left(x_3\downarrow\overline{x_4}\right)\downarrow\left(\overline{x_3}\downarrow x_4\right)\right)\right)\times$$
$$\times\left(x_4+\left(\left(x_1\downarrow\overline{x_2}\downarrow x_3\downarrow\overline{x_5}\right)\downarrow\left(\overline{x_1}\downarrow\overline{x_3}\downarrow x_5\right)\right)\right)=$$
$$=\overline{\left(\left(\overline{\overline{\overline{x_1}\downarrow x_2}}\right)\downarrow\left(\left(x_3\downarrow\overline{x_4}\right)\downarrow\left(\overline{x_3}\downarrow x_4\right)\right)\right)}\times$$
$$\times\left(\overline{x_4\downarrow\left(\left(x_1\downarrow\overline{x_2}\downarrow x_3\downarrow\overline{x_5}\right)\downarrow\left(\overline{x_1}\downarrow\overline{x_3}\downarrow x_5\right)\right)}\right)=$$
$$=\left(\left(\overline{\overline{\overline{x_1}\downarrow x_2}}\right)\downarrow\left(\left(x_3\downarrow\overline{x_4}\right)\downarrow\left(\overline{x_3}\downarrow x_4\right)\right)\right)\downarrow$$
$$\downarrow\left(x_4\downarrow\left(\left(\overline{x_1\downarrow\overline{x_2}}\downarrow\overline{x_3\downarrow\overline{x_5}}\right)\downarrow\left(\overline{\overline{x_1}\downarrow\overline{x_3}}\downarrow x_5\right)\right)\right).$$

PNFPW-2 takes the following form:

$$F_{\text{DNFPW-2}}=\left(\left(\overline{\overline{x_1}\downarrow x_2}\right)\downarrow\left(\left(x_3\downarrow\overline{x_4}\right)\downarrow\left(\overline{x_3}\downarrow x_4\right)\right)\right)\downarrow$$
$$\downarrow\left(x_4\downarrow\left(\left(\overline{x_1\downarrow\overline{x_2}}\downarrow\overline{x_3\downarrow\overline{x_5}}\right)\downarrow\left(\overline{\overline{x_1}\downarrow\overline{x_3}}\downarrow x_5\right)\right)\right). \quad (52)$$

PNFPW-2 (52) provides the corresponding values of one and zero for matrices M^1, M^0. The logic circuit implementing PNFPW-2 (52) is shown in Fig. 9.

Since the first matrix of expression (47) is a singular function [27], this makes it possible to simplify the function $f(x_1, x_2, x_3, x_4, x_5)$ (46) in the polynomial basis with the transition of the minimal polynomial form to the main basis.

$f_{\text{MPNF}}(x_1,x_2,x_3,x_4,x_5)=$

No.	x_1	x_2	x_3	x_4	x_5
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	0
3	0	0	0	1	1
4	0	0	1	0	0
5	0	0	1	0	1
6	0	0	1	1	0
7	0	0	1	1	1
8	0	1	0	0	0
10	0	1	0	1	0
11	0	1	0	1	1
12	0	1	1	0	0
13	0	1	1	0	1
14	0	1	1	1	0
15	0	1	1	1	1
17	1	0	0	0	1
22	1	0	1	1	0
23	1	0	1	1	1
24	1	1	0	0	0
25	1	1	0	0	1
26	1	1	0	1	0
27	1	1	0	1	1
29	1	1	1	0	1
30	1	1	1	1	0
31	1	1	1	1	1
16	1	0	0	0	0

$=$

0 0 0 0 0
0 0 0 0 1
0 0 0 1 0
0 0 0 1 1
0 0 1 0 0
0 0 1 0 1
0 0 1 1 0
0 0 1 1 1
0 1 0 0 0
0 1 0 0 1
0 1 0 1 0
0 1 0 1 1
0 1 1 0 0
0 1 1 0 1
0 1 1 1 0
0 1 1 1 1
0 1 0 0 1
1 0 0 0 1
1 0 1 1 0
1 0 1 1 1
1 1 0 0 0
1 1 0 0 1
1 1 0 1 0
1 1 0 1 1
1 1 1 0 0
1 1 1 0 1
1 1 1 1 0
1 1 1 1 1
1 1 0 0 0
1 0 0 0 0

$=$

x_1	x_2	x_3	x_4	x_5
0				
0	1	0	0	1
1	0	0	0	1
1	0	1	1	0
1	0	1	1	1
1	1			
1	1	1	0	0
1	0	0	0	0

$=$

x_1	x_2	x_3	x_4	x_5
0				
0	1	0	0	1
1	0	0	0	
1	0	1	1	
1	1			
1	1	1	0	0

$=$

x_1	x_2	x_3	x_4	x_5
0				
0	1	0	0	1
1	0		0	
1	0	1		
1	1			
1	1	1	0	0

$=$

x_1	x_2	x_3	x_4	x_5
0				
0	1	0	0	1
1	0		0	
1	0	0		
1				
1	1	1	0	0

$=$

x_1	x_2	x_3	x_4	x_5
0	1	0	0	1
1	0		0	
1	0	0		
1	1	1	0	0

$=1\oplus$

x_1	x_2	x_3	x_4	x_5
0	1	0	0	1
1	0	0	0	
1	0	1	0	
1	0	0		
1	1	1	0	0

$=1\oplus$

x_1	x_2	x_3	x_4	x_5
0	1	0	0	1
1	0	0	1	
1	0	1	0	
1	1	1	0	0

. (53)

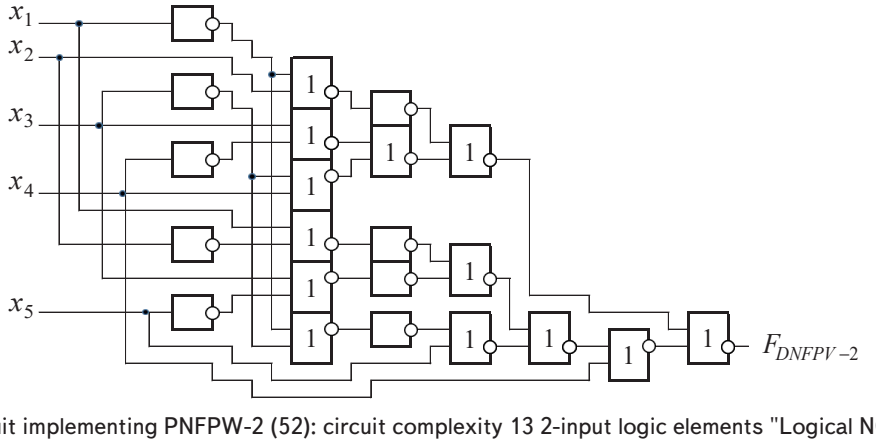

Fig. 9. Logic circuit implementing PNFPW-2 (52): circuit complexity 13 2-input logic elements "Logical NOR", 9 inverters, total 22 logic elements, circuit depth 7 logic elements

The last matrix of expression (53) is a singular function. This makes it possible to obtain the minimum function in the main basis and write PNFPW-1 for it:

$$1\oplus\begin{array}{|ccccc|}\hline x_1 & x_2 & x_3 & x_4 & x_5\\ \hline 0 & 1 & 0 & 0 & 1\\ 1 & 0 & 0 & 1 & \\ 1 & 0 & 1 & 0 & \\ 1 & 1 & 1 & 0 & 0\\ \hline\end{array}=$$

$$=\overline{x_2\overline{x_4}\left(\overline{x_1}\,\overline{x_3}x_5+x_1x_3\overline{x_5}\right)+x_1\overline{x_2}\left(\overline{x_3}x_4+x_3\overline{x_4}\right)}=$$

$$=\overline{x_2\overline{x_4}\left(\overline{\overline{x_1}\,\overline{x_3}x_5\downarrow x_1x_3\overline{x_5}}\right)+x_1\overline{x_2}\left(\overline{\overline{x_3}x_4\downarrow x_3\overline{x_4}}\right)}=$$

$$=\overline{\overline{x_2\overline{x_4}}\downarrow\left(\overline{\overline{x_1}\,\overline{x_3}x_5\downarrow x_1x_3\overline{x_5}}\right)+\overline{x_1\overline{x_2}}\downarrow\left(\overline{\overline{x_3}x_4\downarrow x_3\overline{x_4}}\right)}=$$

$$=\overline{\overline{x_2\overline{x_4}}\downarrow\left(\overline{x_1}\,\overline{x_3}x_5\downarrow x_1x_3\overline{x_5}\right)+\overline{x_1\overline{x_2}}\downarrow\left(\overline{x_3}x_4\downarrow x_3\overline{x_4}\right)}=$$

$$=\overline{\left(\overline{x_2}+x_4\right)\downarrow\left(\overline{x_1}\,\overline{x_3}x_5\downarrow x_1x_3\overline{x_5}\right)+\left(\left(\overline{x_1}+x_2\right)\downarrow\left(\overline{x_3}x_4\downarrow x_3\overline{x_4}\right)\right)}=$$

$$=\overline{\left(\left(\overline{\overline{x_2}\downarrow x_4}\right)\downarrow\left(\left(x_1\downarrow x_3\downarrow\overline{x_5}\right)\downarrow\left(\overline{x_1}\downarrow\overline{x_3}\downarrow x_5\right)\right)\right)+}$$

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

$$=\overline{\left(\left(\overline{\overline{x_2}\downarrow x_4}\right)\downarrow\left(\left(x_1\downarrow x_3\downarrow\overline{x_5}\right)\downarrow\left(\overline{x_1}\downarrow\overline{x_3}\downarrow x_5\right)\right)\right)\downarrow}$$

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

$$=\left(\left(\overline{\overline{x_2}\downarrow x_4}\right)\downarrow\left(\left(x_1\downarrow x_3\downarrow\overline{x_5}\right)\downarrow\left(\overline{x_1}\downarrow\overline{x_3}\downarrow x_5\right)\right)\right)\downarrow$$

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

$$=\left(\left(\overline{\overline{x_2}\downarrow x_4}\right)\downarrow\left(\left(x_1\downarrow\overline{x_3\downarrow\overline{x_5}}\right)\downarrow\left(\overline{x_1}\downarrow\overline{\overline{x_3}\downarrow x_5}\right)\right)\right)\downarrow$$

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

PNFPW-1 takes the following form:

$$F_{\text{DNFPW-1}}=\left(\left(\overline{\overline{x_2}\downarrow x_4}\right)\downarrow\left(\left(x_1\downarrow\overline{x_3\downarrow\overline{x_5}}\right)\downarrow\left(\overline{x_1}\downarrow\overline{\overline{x_3}\downarrow x_5}\right)\right)\right)\downarrow$$
$$\downarrow\left(\left(\overline{\overline{x_1}\downarrow x_2}\right)\downarrow\left(\left(x_3\downarrow\overline{x_4}\right)\downarrow\left(\overline{x_3}\downarrow x_4\right)\right)\right). \quad (54)$$

PNFPW-1 (54) provides the corresponding values of one and zero for matrices M^1, M^0. The logic circuit implementing PNFPW-1 (54) is shown in Fig. 10.

PNFPW-1 (49), (54) and PNFPW-2 (52) represent the class of dead-end forms of the function $f(x_1, x_2, x_3, x_4, x_5)$ (46). Depending on the technical conditions for designing a logic circuit, one of the considered dead-end forms may become minimal. The parameters of the logic circuits that implement the specified Peirce-Webb functions are given in Table 12.

Table 12

Parameters of logic circuits that implement Peirce-Webb functions

Logic form	Number of logical NOR elements	Number of inverters	Total logic elements	Circuit depth	Fig.
PNFPW-1	11	9	20	8	7
PNFPW-2	13	9	22	7	8
PNFPW-1	13	9	22	7	9

Fig. 11 shows a logic scheme designed by the method of algebraic bi-decomposition of the Boolean function [26].

Considering the schemes in Fig. 8–10, we can see that these schemes show a smaller number of logic elements, which they consist of, compared to the scheme in Fig. 11. The depth of the scheme in Fig. 9–11 is the same, it is 7 logic elements.

The peculiarity of the method of bitwise partitioning of the set of output conjuncterms is the absence of tautology when simplifying Boolean functions, the conjuncterms of the lower rank are established without performing intermediate operations of gluing the variables [28]. This gives the optimal complexity of the function simplification procedure.

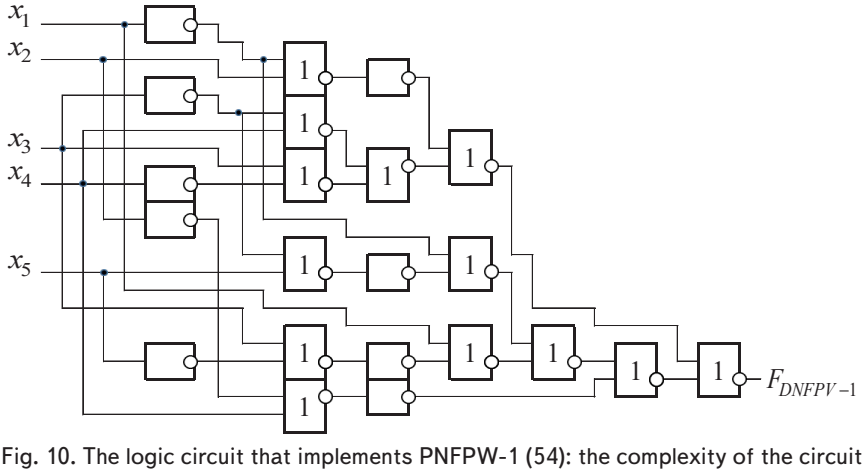

Fig. 10. The logic circuit that implements PNFPW-1 (54): the complexity of the circuit is 13 2-input logic elements "Logical NOR", 9 inverters, a total of 22 logic elements, the depth of the circuit is 7 logic elements

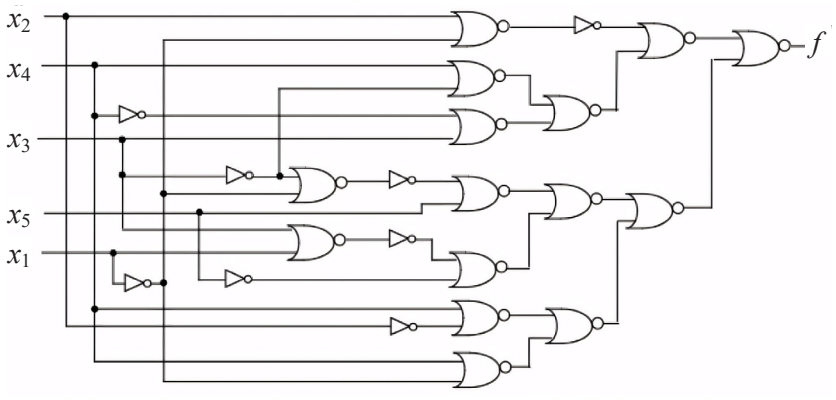

Fig. 11. A logic circuit designed by the method of algebraic decomposition of a Boolean function: the complexity of the circuit is 15 2-input logic elements "Logical NOR" and 8 inverters, a total of 23 logic elements, the depth of the circuit is 7 logic elements

Example 11. Use a non-standard system to simplify the function $f(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ given in the canonical form [28]:

$$f(x_1,x_2,x_3,x_4,x_5,x_6,x_7)= \\ =\sum\begin{pmatrix}0,1,5,6,7,8,10,14,22,23,38,39,48,49,\\ 53,54,55,58,62,86,87,102,103,122\end{pmatrix}. \quad (55)$$

Solution.

Simplification of the function $f(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ (55) in the disjunctive normal form (DNF):

$$f_{\text{MDNF}}(x_1,x_2,x_3,x_4,x_5,x_6,x_7)=$$

No.	x_1	x_2	x_3	x_4	x_5	x_6	x_7	F
0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	1	1
5	0	0	0	0	1	0	1	1
6	0	0	0	0	1	1	0	1
7	0	0	0	0	1	1	1	1
8	0	0	0	1	0	0	0	1
10	0	0	0	1	0	1	0	1
14	0	0	0	1	1	1	0	1
22	0	0	1	0	1	1	0	1
23	0	0	1	0	1	1	1	1
38	0	1	0	0	1	1	0	1
39	0	1	0	0	1	1	1	1
48	0	1	1	0	0	0	0	1
49	0	1	1	0	0	0	1	1
53	0	1	1	0	1	0	1	1
54	0	1	1	0	1	1	0	1
55	0	1	1	0	1	1	1	1
58	0	1	1	1	0	1	0	1
62	0	1	1	1	1	1	0	1
86	1	0	1	0	1	1	0	1
87	1	0	1	0	1	1	1	1
102	1	1	0	0	1	1	0	1
103	1	1	0	0	1	1	1	1
122	1	1	1	1	0	1	0	1

$=$

x_1	x_2	x_3	x_4	x_5	x_6	x_7
0	0	0		0	0	0
0	0	0	0		0	1
0			0	1	1	
0	0	0	1		1	0
0	1	1	0	0	0	
0	1	1	0		0	1
0	1	1	1		1	0
	0	1	0	1	1	
	1	0	0	1	1	
	1	1	1	0	1	0

. (56)

In the first matrix of expression (56), the following operations are performed: to blocks 6, 7, 22, 23, 38, 39, 54, 55, which form intervals of the Boolean space containing the complete combinatorial system of 2-(3, 8)-design, and to blocks 22, 23, 86, 87; 38, 39, 102, 103, which form intervals of the Boolean space containing complete combinatorial systems of 2-(2, 4)-design, the operation of super-gluing the variables is applied.

The 22nd "0010110" and 23rd "0010111" variable blocks are common to the two systems – $\Sigma m(6, 7, 22, 23, 38, 39, 54, 55)$, $\Sigma m(22, 23, 86, 87)$ and two operations of super-gluing the variables, the location of which is determined by the combinatorial systems of 2-(3, 8)-design and 2-(2, 4)-design.

The 38th "0100110" and 39th "0100111" variable blocks are common to two systems – $\Sigma m(6, 7, 22, 23, 38, 39, 54, 55)$, $\Sigma m(38, 39, 102, 103)$ and two operations of super-gluing the variables, the location of which is determined by the combinatorial systems of 2-(3, 8)-design and 2-(2, 4)-design.

In the first matrix of expression (54), to blocks 0, 8; 1, 5; 10, 14; 48, 49; 49, 53; 58, 62; 62, 122, which form the intervals of the Boolean space containing the complete combinatorial system of 2-(1, 2)-design, the operation of simple gluing of variables is applied.

The 49th "0110001" block of variables is common to two systems – $\Sigma m(48, 49)$, $\Sigma m(49,53)$ and one operation of simple gluing of variables, the location of which is determined by combinatorial systems of 2-(1, 2)-design.

The 62nd "0111110" block of variables is common to two systems – $\Sigma m(58, 62)$, $\Sigma m(62, 122)$ and one simple variable gluing operation, the location of which is determined by combinatorial 2-(1, 2)-design systems.

As a result, the minimal function in disjunctive normal form is obtained in one step:

$$f_{\text{MDNF}}(x_1,x_2,x_3,x_4,x_5,x_6,x_7)= \\ =\overline{x}_1x_2x_3\overline{x}_5x_6x_7+\overline{x}_1x_2\overline{x}_3x_4x_6x_7+\overline{x}_1\overline{x}_4x_5x_6+ \\ +\overline{x}_1x_2\overline{x}_3x_4\overline{x}_6\overline{x}_7+\overline{x}_1\overline{x}_2x_3\overline{x}_4\overline{x}_5\overline{x}_6+ \\ +\overline{x}_1x_2\overline{x}_3\overline{x}_4x_6\overline{x}_7+\overline{x}_1x_2\overline{x}_3x_4\overline{x}_6\overline{x}_7+\overline{x}_2\overline{x}_3\overline{x}_4x_5x_6+ \\ +\overline{x}_2\overline{x}_3x_4\overline{x}_5x_6+x_2\overline{x}_3x_4\overline{x}_5\overline{x}_6\overline{x}_7. \quad (57)$$

Decimal equivalent of MDNF (57):

$$f_{\text{MDNF}}(x_1,x_2,x_3,x_4,x_5,x_6,x_7)= \\ =\begin{Bmatrix}(0,8),(1,5),(6,7,22,23,38,39,54,55),\\ (10,14),(22,\ 23,86,87),(38,39,102,103),\\ (48,49),(49,53),(58,62),(62,122)\end{Bmatrix}= \\ =\begin{Bmatrix}0,1,5,6,7,8,10,14,22,\ 23,38,39,48,49,\\ 53,54,55,58,62,86,87,102,103,122\end{Bmatrix},$$

coincides with the decimal counterpart, represented in abbreviated form [28].

Simplification of the function $f(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ (55) in the conjunctive normal form (CNF) (Fig. 12).

No.	x_1	x_2	x_3	x_4	x_5	x_6	x_7	f
2	0	0	0	0	0	1	0	0
3	0	0	0	0	0	1	1	0
4	0	0	0	0	1	0	0	0
9	0	0	0	1	0	0	1	0
11	0	0	0	1	0	1	1	0
12	0	0	0	1	1	0	0	0
13	0	0	0	1	1	0	1	0
15	0	0	0	1	1	1	1	0
16	0	0	1	0	0	0	0	0
17	0	0	1	0	0	0	1	0
18	0	0	1	0	0	1	0	0
19	0	0	1	0	0	1	1	0
20	0	0	1	0	1	0	0	0
21	0	0	1	0	1	0	1	0
24	0	0	1	1	0	0	0	0
25	0	0	1	1	0	0	1	0
26	0	0	1	1	0	1	0	0
27	0	0	1	1	0	1	1	0
28	0	0	1	1	1	0	0	0
29	0	0	1	1	1	0	1	0
30	0	0	1	1	1	1	0	0
31	0	0	1	1	1	1	1	0
32	0	1	0	0	0	0	0	0
33	0	1	0	0	0	0	1	0
34	0	1	0	0	0	1	0	0
35	0	1	0	0	0	1	1	0
36	0	1	0	0	1	0	0	0
37	0	1	0	0	1	0	1	0
40	0	1	0	1	0	0	0	0
41	0	1	0	1	0	0	1	0
42	0	1	0	1	0	1	0	0
43	0	1	0	1	0	1	1	0
44	0	1	0	1	1	0	0	0
45	0	1	0	1	1	0	1	0
46	0	1	0	1	1	1	0	0
47	0	1	0	1	1	1	1	0
50	0	1	1	0	0	1	0	0
51	0	1	1	0	0	1	1	0
52	0	1	1	0	1	0	0	0
56	0	1	1	1	0	0	0	0
57	0	1	1	1	0	0	1	0
59	0	1	1	1	0	1	1	0
60	0	1	1	1	1	0	0	0
61	0	1	1	1	1	0	1	0
63	0	1	1	1	1	1	1	0
64	1	0	0	0	0	0	0	0
65	1	0	0	0	0	0	1	0
66	1	0	0	0	0	1	0	0
67	1	0	0	0	0	1	1	0
68	1	0	0	0	1	0	0	0
69	1	0	0	0	1	0	1	0
70	1	0	0	0	1	1	0	0

No.	x_1	x_2	x_3	x_4	x_5	x_6	x_7	f
71	1	0	0	0	1	1	1	0
72	1	0	0	1	0	0	0	0
73	1	0	0	1	0	0	1	0
74	1	0	0	1	0	1	0	0
75	1	0	0	1	0	1	1	0
76	1	0	0	1	1	0	0	0
77	1	0	0	1	1	0	1	0
78	1	0	0	1	1	1	0	0
79	1	0	0	1	1	1	1	0
80	1	0	1	0	0	0	0	0
81	1	0	1	0	0	0	1	0
82	1	0	1	0	0	1	0	0
83	1	0	1	0	0	1	1	0
84	1	0	1	0	1	0	0	0
85	1	0	1	0	1	0	1	0
88	1	0	1	1	0	0	0	0
89	1	0	1	1	0	0	1	0
90	1	0	1	1	0	1	0	0
91	1	0	1	1	0	1	1	0
92	1	0	1	1	1	0	0	0
93	1	0	1	1	1	0	1	0
94	1	0	1	1	1	1	0	0
95	1	0	1	1	1	1	1	0
96	1	1	0	0	0	0	0	0
97	1	1	0	0	0	0	1	0
98	1	1	0	0	0	1	0	0
99	1	1	0	0	0	1	1	0
100	1	1	0	0	1	0	0	0
101	1	1	0	0	1	0	1	0
104	1	1	0	1	0	0	0	0
105	1	1	0	1	0	0	1	0
106	1	1	0	1	0	1	0	0
107	1	1	0	1	0	1	1	0
108	1	1	0	1	1	0	0	0
109	1	1	0	1	1	0	1	0
110	1	1	0	1	1	1	0	0
111	1	1	0	1	1	1	1	0
112	1	1	1	0	0	0	0	0
113	1	1	1	0	0	0	1	0
114	1	1	1	0	0	1	0	0
115	1	1	1	0	0	1	1	0
116	1	1	1	0	1	0	0	0
117	1	1	1	0	1	0	1	0
118	1	1	1	0	1	1	0	0
119	1	1	1	0	1	1	1	0
120	1	1	1	1	0	0	0	0
121	1	1	1	1	0	0	1	0
123	1	1	1	1	0	1	1	0
124	1	1	1	1	1	0	0	0
125	1	1	1	1	1	0	1	0
126	1	1	1	1	1	1	0	0
127	1	1	1	1	1	1	1	0

Fig. 12. Conjunctive normal form (CNF) of function (55)

The last matrix of the CNF simplification of the function (Fig. 12) takes the following form:

$$f_{\text{MCNF}}(x_1,x_2,x_3,x_4,x_5,x_6,x_7)=$$

x_1	x_2	x_3	x_4	x_5	x_6	x_7
			1	1	0	
				0	1	1
			0			0
	1	0		1		
	1	0			1	
	1	0	0			
	0	1		1		
	0	1			1	
	0	1	0			
	0		0		1	
0	1	1				
0			1	1		
0	0				1	
0	0	0		0		

$$=\left(\overline{x_1}+\overline{x_2}+\overline{x_3}+\overline{x_5}\right)\times$$
$$\times\left(\overline{x_1}+\overline{x_2}+x_6\right)\times$$
$$\times\left(\overline{x_1}+x_4+x_5\right)\times$$
$$\times\left(\overline{x_1}+x_2+x_3\right)\times$$
$$\times\left(\overline{x_2}+\overline{x_4}+x_6\right)\times$$
$$\times\left(\overline{x_2}+x_3+\overline{x_4}\right)\times$$
$$\times\left(\overline{x_2}+x_3+x_6\right)$$
$$\times\left(\overline{x_2}+x_3+x_5\right)\times$$
$$\times\left(x_2+\overline{x_3}+\overline{x_4}\right)\times$$
$$\times\left(x_2+\overline{x_3}+x_6\right)\times$$
$$\times\left(x_2+\overline{x_3}+x_5\right)\times$$
$$\times\left(\overline{x_4}+\overline{x_7}\right)\times$$
$$\times\left(\overline{x_5}+x_6+x_7\right)\times$$
$$\times\left(x_4+x_5+\overline{x_6}\right). \qquad (58)$$

MCNF (58) contains 14 fewer literals compared to MDNF (57).

For MCNF (58), PNFPW-2 is recorded as follows:

$$F_{\text{DNFPW-2}}=$$
$$=\left(\overline{x_1}+\overline{x_2}+\overline{x_3}+\overline{x_5}\right)\left(\overline{x_1}+\overline{x_2}+x_6\right)\left(\overline{x_1}+x_4+x_5\right)\left(\overline{x_1}+x_2+x_3\right)\left(\overline{x_2}+\overline{x_4}+x_6\right)\times$$
$$\times\left(\overline{x_2}+x_3+\overline{x_4}\right)\left(\overline{x_2}+x_3+x_6\right)\left(\overline{x_2}+x_3+x_5\right)\left(x_2+\overline{x_3}+\overline{x_4}\right)\left(x_2+\overline{x_3}+x_6\right)\times$$
$$\times\left(x_2+\overline{x_3}+x_5\right)\left(\overline{x_4}+\overline{x_7}\right)\left(\overline{x_5}+x_6+x_7\right)\left(x_4+x_5+\overline{x_6}\right)=$$
$$=\left(\overline{x_1}+\left(\left(\overline{x_2}+\overline{x_3}+\overline{x_5}\right)\left(\overline{x_2}+x_6\right)\left(x_4+x_5\right)\left(x_2+x_3\right)\right)\right)\times$$
$$\times\left(\overline{x_2}+\left(\overline{x_4}+x_6\right)\left(x_3+\overline{x_4}\right)\left(x_3+x_6\right)\left(x_3+x_5\right)\right)\left(x_2+\left(\overline{x_3}+\overline{x_4}\right)\left(\overline{x_3}+x_6\right)\left(\overline{x_3}+x_5\right)\right)\left(\overline{x_4}+\overline{x_7}\right)\times$$
$$\times\left(\overline{x_5}+x_6+x_7\right)\left(x_4+x_5+\overline{x_6}\right)=$$
$$=\left(\overline{x_1}+\left(\left(\overline{x_2}+\left(\overline{x_3}+\overline{x_5}\right)\left(x_6\right)\right)\left(\left(x_4+x_5\right)\left(x_2+x_3\right)\right)\right)\right)\times$$
$$\times\left(\overline{x_2}+\left(\overline{x_4}+x_6\right)\left(x_3+\left(\overline{x_4}x_6x_5\right)\right)\right)\times\left(x_2+\left(\overline{x_3}+\left(\overline{x_4}x_6x_5\right)\right)\right)\times$$
$$\times\left(\overline{x_4}+\overline{x_7}\right)\left(\overline{x_5}+x_6+x_7\right)\left(x_4+x_5+\overline{x_6}\right)=$$
$$=\left(\overline{x_1}+\left(\overline{x_2}+\left(\left(\overline{\overline{x_3}\downarrow\overline{x_5}}\right)x_6\right)\right)\left(\left(\overline{x_4\downarrow x_5}\right)\left(\overline{x_2\downarrow x_3}\right)\right)\right)\times$$
$$\times\left(\overline{x_2}+\left(\overline{\overline{x_4}\downarrow x_6}\right)\left(x_3+\left(x_4\downarrow\overline{x_6}\downarrow\overline{x_5}\right)\right)\right)\times\left(x_2+\left(\overline{x_3}+\left(x_4\downarrow\overline{x_6}\downarrow\overline{x_5}\right)\right)\right)\times$$
$$\times\left(\overline{\overline{x_4}\downarrow\overline{x_7}}\right)\left(\overline{\overline{x_5}\downarrow x_6\downarrow x_7}\right)\left(\overline{\left(\overline{x_4\downarrow x_5}\right)\downarrow\overline{x_6}}\right)=$$
$$=\left(\overline{x_1}+\left(\overline{x_2}+\left(\left(\overline{x_3}\downarrow\overline{x_5}\right)\downarrow\overline{x_6}\right)\right)\left(\left(x_4\downarrow x_5\right)\downarrow\left(x_2\downarrow x_3\right)\right)\right)\times$$
$$\times\left(\overline{\overline{x_2}\downarrow\left(\overline{\overline{x_4}\downarrow x_6}\right)}\left(\overline{x_3\downarrow\left(x_4\downarrow\overline{x_6}\downarrow\overline{x_5}\right)}\right)\right)\times\left(x_2+\left(\overline{\overline{x_3}\downarrow\left(x_4\downarrow\overline{x_6}\downarrow\overline{x_5}\right)}\right)\right)\times$$
$$\times\left(\overline{x_4}\downarrow\overline{x_7}\right)\downarrow\left(\overline{x_5}\downarrow x_6\downarrow x_7\right)\downarrow\left(\left(\overline{x_4\downarrow x_5}\right)\downarrow\overline{x_6}\right)=$$
$$=\left(\overline{x_1}+\left(\overline{\overline{x_2}\downarrow\left(\left(\overline{x_3}\downarrow\overline{x_5}\right)\downarrow\overline{x_6}\right)}\right)\left(\left(x_4\downarrow x_5\right)\downarrow\left(x_2\downarrow x_3\right)\right)\right)\times$$
$$\times\left(\left(\overline{x_2}\downarrow\left(\overline{\overline{x_4}\downarrow x_6}\right)\right)\downarrow\left(x_3\downarrow\left(x_4\downarrow\overline{x_6}\downarrow\overline{x_5}\right)\right)\right)\times$$
$$\times\left(\overline{x_2\downarrow\left(\overline{\overline{x_3}\downarrow\left(x_4\downarrow\overline{x_6}\downarrow\overline{x_5}\right)}\right)}\right)\times\left(\overline{x_4}\downarrow\overline{x_7}\right)\downarrow\left(\overline{x_5}\downarrow x_6\downarrow x_7\right)\downarrow\left(\left(\overline{x_4\downarrow x_5}\right)\downarrow\overline{x_6}\right)=$$
$$=\overline{\overline{x_1}\downarrow\left(\left(\overline{x_2}\downarrow\left(\left(\overline{x_3}\downarrow\overline{x_5}\right)\downarrow\overline{x_6}\right)\right)\downarrow\overline{\left(\left(x_4\downarrow x_5\right)\downarrow\left(x_2\downarrow x_3\right)\right)}\right)}\times$$
$$\times\overline{x_2}\downarrow\left(\left(\overline{x_4}\downarrow\left(\overline{x_3}\downarrow\overline{x_6}\right)\right)\downarrow\left(x_3\downarrow\left(\overline{x_5}\downarrow\overline{x_6}\right)\right)\right)\times\overline{\left(\overline{x_2\downarrow\overline{x_3}}\right)\downarrow\left(x_4\downarrow\overline{x_5}\downarrow\overline{x_6}\right)}\times$$
$$\times\left(\overline{x_4}\downarrow\overline{x_7}\right)\downarrow\left(\overline{x_5}\downarrow x_6\downarrow x_7\right)\downarrow\left(x_4\downarrow x_5\downarrow\overline{x_6}\right)=$$
$$=\left(\overline{x_1}\downarrow\left(\left(\overline{x_2}\downarrow\left(\left(\overline{x_3}\downarrow\overline{x_5}\right)\downarrow\overline{x_6}\right)\right)\downarrow\left(\overline{\left(x_4\downarrow x_5\right)\downarrow\left(x_2\downarrow x_3\right)}\right)\right)\right)\downarrow$$
$$\downarrow\left(\overline{x_2}\downarrow\left(\overline{x_4}\downarrow\left(\overline{x_3}\downarrow\overline{x_6}\right)\right)\downarrow\left(x_3\downarrow\left(\overline{x_5}\downarrow\overline{x_6}\right)\right)\right)\downarrow\left(\left(\overline{x_2\downarrow\overline{x_3}}\right)\downarrow\left(x_4\downarrow\overline{x_5}\downarrow\overline{x_6}\right)\right)\downarrow$$
$$\downarrow\left(\left(\overline{x_4}\downarrow\overline{x_7}\right)\downarrow\left(\overline{x_5}\downarrow x_6\downarrow x_7\right)\downarrow\left(x_4\downarrow x_5\downarrow\overline{x_6}\right)\right).$$

$$F_{\text{DNFPW-2}} = \left(\overline{x_1}\downarrow\left(\begin{array}{l}\left(\overline{x_2}\downarrow\left(\left(\overline{x_3}\downarrow\overline{x_5}\right)\downarrow\overline{x_6}\right)\right)\downarrow\\ \downarrow\left(\overline{\left(x_4\downarrow x_5\right)\downarrow\left(x_2\downarrow x_3\right)}\right)\end{array}\right)\right)\downarrow$$
$$\downarrow\left(\overline{x_2}\downarrow\left(\left(\overline{x_4}\downarrow\left(\overline{x_3}\downarrow\overline{x_6}\right)\right)\downarrow\left(x_3\downarrow\left(\overline{x_5}\downarrow\overline{x_6}\right)\right)\right)\right)\downarrow$$
$$\downarrow\left(\left(\overline{x_2\downarrow\overline{x_3}}\right)\downarrow\left(x_4\downarrow\overline{x_5}\downarrow\overline{x_6}\right)\right)\downarrow$$
$$\downarrow\left(\overline{\left(\overline{x_4}\downarrow\overline{x_7}\right)\downarrow\left(\overline{x_5}\downarrow x_6\downarrow x_7\right)\downarrow\left(x_4\downarrow x_5\downarrow\overline{x_6}\right)}\right). \quad (59)$$

PNFPW-2 (59) (Fig. 13) contains 27 fewer literals compared to MDNF (57).

The logic circuit in Fig. 13 consists of 17 2-input Peirce-arrow logic elements, 4 3-input Peirce-arrow logic elements, 1 4-input Peirce-arrow logic element, and 10 inverters.

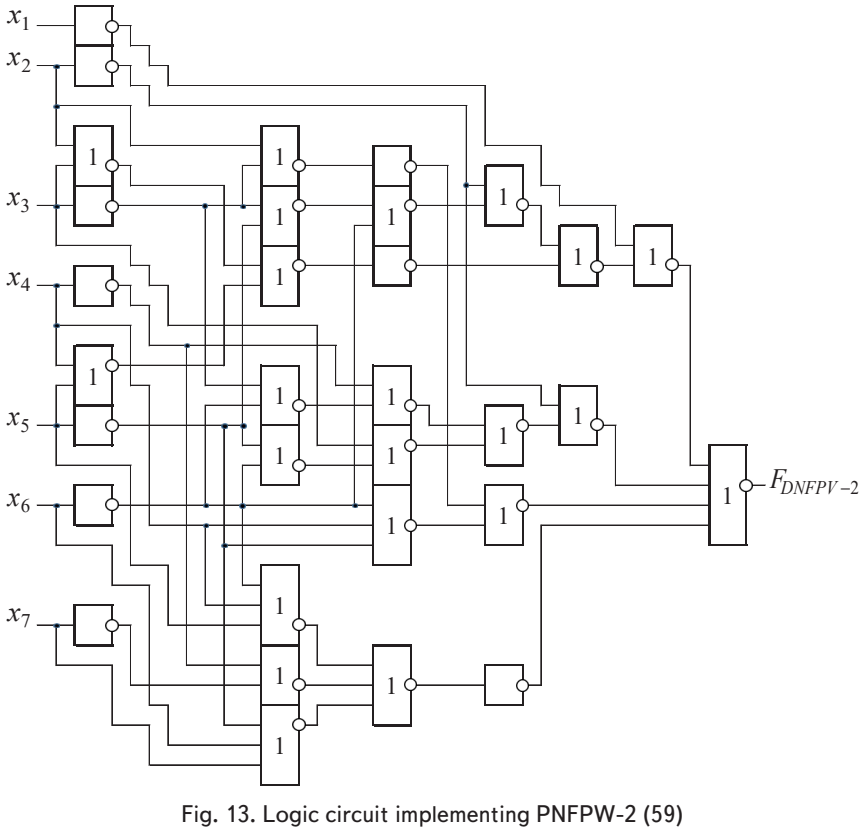

Fig. 13. Logic circuit implementing PNFPW-2 (59)

6. Discussion of results from the simplification of Peirce-Webb functions by a non-standard system

The beginning of the simplification of Boolean functions by a non-standard system in the Peirce-Webb basis is the search for intervals of the Boolean space containing the combinatorial systems of 2-(n, b)-design, 2-$(n, x/b)$-design, in particular, in the case when different intervals of the Boolean space partially coincide. The non-standard is heuristic. Heuristics are inventive, the answer is not to "calculate" but to find, this is the true search – the desire to find. Detection through the search for the necessary intervals of the Boolean space unambiguously implies the locations of equivalent transformations and provides the "trigger causality" of the consequence, in the form of the very solution to the problem of the systematized procedure for simplifying the Peirce-Webb functions by the visual-matrix form of the analytical method [16].

The mathematical apparatus of the non-standard system for simplifying Boolean functions in the Peirce-Webb basis is the method of figurative transformations, which is considered in works [29–32] and others.

A new component in the technology of simplifying Boolean functions with a non-standard system represents the algebra of equivalent transformations in the class of perfect normal forms of Peirce-Webb functions.

The following results were obtained for each task, in order to achieve the overall goal of the research:

1. Boolean functions in the Peirce-Webb basis – PNFPW-1 and PNFPW-2 – are represented by matrices with the same combinatorial properties. The difference between the specified matrices lies in the hermeneutics of logical operations. The PNFPW-1 matrix gives the PNFPW-1 terms and the logical NOR operation for them, and the PNFPW-2 matrix gives the PNFPW-2 terms and the logical NOR operation for them.

It is important to see that the combinatorial properties of binary structures PNFPW-1, PNFPW-2 functions in the Peirce-Webb basis and PDNF, DCNF functions of the main basis do not depend on the selected logical basis. This makes it possible to carry out equivalent transformations in the PNFPW-1, PNFPW-2 matrices according to the rules of the algebra of the main basis.

The hermeneutics of logical operations are demonstrated on matrices (1) to (3) and others.

2. The rules for simplifying the Peirce-Webb functions in 3 exceptional situations have been established. The graphic equivalent of the specified rules in the form of logic elements is presented in Fig. 3–5. The rules for simplifying the Peirce-Webb functions for the first and second exceptional situations are discussed in examples 1–5. The third exceptional situation is demonstrated by the rules of equivalent transformation of Peirce-Webb functions – (14), (20), (28).

3. The algebra of equivalent transformations for the simplification of Boolean functions in the Peirce-Webb basis has been extended (Table 13).

Logical operations 4–8, 12, 13 (Table 10) are new operations of equivalent transformation of Boolean functions in the Peirce-Webb basis. This extends the capabilities of the analytical method when simplifying the Peirce-Webb functions.

4. In the general case, the search for the minimal function is carried out on the complete truth table and on various logical bases. A complete truth table holds the sets of variables for which the function returns either "1" or "0" at the output. The minimum function in the Peirce-Webb basis should be chosen based on the results of minimization of two normal forms – PNFPW-1 and PNFPW-2, which is demonstrated by example 8.

5. A comparative analysis of the results of the simplification of Boolean functions in the Peirce-Webb basis by a non-standard system and examples of the simplification of functions by the Veitch diagram, matrix and graph algorithms,

methods for algebraic bi-decomposition and bitwise division of the set of initial conjunctives was carried out (Table 14).

The non-standard system shows the same result of simplifying functions in the main basis and a better result in the Peirce-Webb basis. For all the considered examples, the simplification procedure with a non-standard system is simpler. This makes it possible, to a certain extent, to simplify functions without the use of automated calculations.

The interpretation of the result is determined by the properties of the binary combinatorial structures of PNFPW-1, PNFPW-2 functions in the Peirce-Webb basis and PDNF, DCNF functions of the main basis. The specified properties do not depend on the selected logical basis. This makes it possible to carry out equivalent transformations on the binary matrices PNFPW-1, PNFPW-2 according to the rules of the algebra of the main basis. The result of the transformation of the terms of the binary matrix in the end is some combinatorial system, metadata that can explain other data, for example, determine the minimum function for another logical basis. Equivalent transformations involving combinatorial systems of 2-(n, b)-design, 2-$(n, x/b)$-design, which by their properties have a greater information capacity, could effectively replace verbal procedures of algebraic transformations.

In contrast to the transition to the Peirce-Webb basis at the level of logic elements, by replacing AND and OR gates with OR-NOT or AND-NOT gates [1–4, 6, 7], or replacing AND and OR logic elements with OR elements NOT or AND-NOT by applying recurrent dependences [5], the feature of the simplification of Boolean functions in the Peirce-Webb basis by a non-standard system consists in changing the form of simplification of Boolean functions in the Peirce-Webb basis within the following representations:

– formation of new rules for equivalent transformation of Boolean functions in the Peirce-Webb basis;

– providing a procedure for simplifying the Peirce-Webb functions in one step;

– performing the minimization of Peirce-Webb functions on the complete truth table;

– application of the distributive law of the 2nd kind when simplifying the Peirce-Webb functions;

– using the Reed-Müller basis to simplify the Peirce-Webb functions.

The non-standard system is based on combinatorial properties of binary structures of PNFPW-1, PNFPW-2 functions in the Peirce-Webb basis and PDNF, DCNF functions of the main basis. These properties do not depend on the selected logical basis, which makes it possible to carry out equivalent transformations on the binary matrices PNFPW-1, PNFPW-2 according to the rules of the algebra of the main basis. The result of the transformation of the terms of the binary matrix in the end is some combinatorial system, metadata that can explain other data, for example, determine the minimum function for another logical basis.

Implication of the Peirce-Webb function simplification algorithm by the 2-(n, b)-design, 2-$(n, x/b)$-design systems through their search on the binary structure of the truth table is also possible when different intervals of the Boolean space partially coincide. It is important to note that the standard (verbal) procedure for simplifying Boolean functions does not provide for the intersection of minterms (maxterms).

Table 13

Equivalent transformation operations of Boolean functions for the OR-NOT monobasis

No. of entry	Logic operation ID	Reference in the text	Representation form
1	Gluing of 2-digit terms of variables	(13), (14)	PNFPW-1
2	Gluing of 3-digit terms of variables	(15), (16)	PNFPW-1
3	Gluing of 3-digit terms of variables	(18)	PNFPW-2
4	Supergluing of 3-digit terms of variables	(19), (20)	PNFPW-1
5	Supergluing of 4-digit terms of variables	(21), (22)	PNFPW-1
6	Incomplete supergluing of 2-digit terms of variables	(23), (24)	PNFPW-1
7	Incomplete supergluing of 3-digit terms of variables	(25)	PNFPW-1
8	Generalized gluing of variables	(26)	PNFPW-1
9	Operation of absorbing the variables	(27), (28)	PNFPW-1
10	Operation of absorbing the variables	(29)	PNFPW-1
11	Operation of absorbing the variables	(30)	PNFPW-1
12	Operation of semi-gluing the variables	(31), (32)	PNFPW-2
13	Rule without a name	(33), (34)	PNFPW-2

Table 14

Comparative table of examples of simplification of Boolean functions borrowed from the works by other authors and a non-standard system

Example No.	Number of input variables	Simplification method ID	Simplification result	Non-standard system
7	4	Veitch diagram [18]	MCNF 8 literals	MCNF 8 literals. PNFPW-2 7 literals
8	4	Graph method [19]	MDNF 9 literals	MDNF 9 literals. PNFPW-2 8 literals
9	4	Matrix method [21]	MDNF 8 literals	MDNF 8 literals. PNFPW-2 6 literals
10	5	Bi-decomposition method [25]	OR-NOT basis 23 gates, circuit depth 7	OR-NOT basis 22 gates, circuit depth 7; 20 gates; circuit depth 8
11	7	The method of bitwise partitioning of the set of output conjuncterms [28]	MDNF 56 literals	MDNF 56 literals. PNFPW-2 29 literals

The application of the obtained result makes it possible to improve and expand the capabilities of the design technology of electronic components and devices for their use in digital technologies, which are based on the Peirce-Webb functions.

The visual representation of 2-dimensional binary matrices allows, to a certain extent, for a manual way to simplify the Peirce-Webb functions using a mathematical editor, for example, Math Type 7.4.0 (USA): examples 1–10.

The non-standard system for simplifying the Peirce-Webb functions opposes the transition to the Peirce-Webb basis at the level of logic elements, by replacing AND and OR gates with OR-NOT or AND-NOT gates and brings the problem of simplifying Boolean functions in the Peirce-Webb basis to the level of a well-researched problem in the class of disjunctive-conjunctive normal forms (DCNF) of representation of Boolean functions. The limitation of the application of the non-standard system is the cases when the switching function is represented in a mixed basis. In this case, the function must be represented by one logical basis.

The weakness of the considered system is its limited practical application for simplifying the Peirce-Webb functions with the subsequent design and manufacture of the corresponding computing components. The negative internal factors of a non-standard system are associated with additional time costs for establishing the protocols for simplifying the Peirce-Webb functions, followed by the creation of a library of protocols that illustrate equivalent image transformations.

The lack of installation of the simplification method in the automated design system can be noted as a shortcoming of this study. This shortcoming can be eliminated in the future by writing a software tool that implements the information technology of performing the functions of designing logical circuits according to the considered method.

Equivalent figurative transformations are a universal toolset for simplifying Boolean functions. In this regard, the prospect of further research may be, for example, the use of a non-standard system to simplify the logical functions of the majority basis.

7. Conclusions

1. The perfect normal form of an n-bit Peirce-Webb function can be represented by binary sets or a matrix, which in this case will give the terms of the Peirce-Webb function and the logical NOR operation for them. Boolean functions of PNFPW-1 and PNFPW-2 representations are set by matrices with the same combinatorial properties. The difference between the specified matrices is the hermeneutics of logical operations. The PNFPW-1 matrix gives PNFPW-1 terms and the logical NOR operation for them, and the PNFPW-2 matrix gives PNFPW-2 terms and the logical NOR operation for them.

The combinatorial properties of the binary structures PNFPW-1, PNFPW-2 of functions in the Peirce-Webb basis and PDNF, PCNF of the functions of the main basis do not depend on the selected logical basis (Boolean basis or Peirce-Webb basis). This makes it possible to carry out equivalent transformations in the PNFPW-1, PNFPW-2 matrices according to the rules of the algebra of the main basis.

It is advisable to use the hermeneutics of logical operations when simplifying the Peirce-Webb functions.

2. When simplifying the Peirce-Webb functions, exceptional situations should be taken into account and equivalent transformation rules should be created for them. The rules of equivalent transformation of Peirce-Webb functions in 3 exceptional situations have been established. The graphic equivalent of the specified rules in the form of connections of logic elements is presented. The application of the rules of equivalent transformation of Peirce-Webb functions in exceptional situations is required not only when deriving the result of simplification from a binary matrix but also when entering the algebraic form of a function into a binary matrix.

3. To properly simplify the Peirce-Webb functions with a non-standard system, we have developed rules for their equivalent transformation, in which logical operations 4–8, 12, 13 are new equivalent transformation operations of Boolean functions in the Peirce-Webb basis. This increases the analytical method's ability to obtain the optimal function in the Peirce-Webb basis.

4. It was found that the best result in the minimization of Peirce-Webb functions can be achieved both in PNFPW-1 and in PNFPW-2. It follows that the minimization of the given function should be carried out in two perfect normal forms – PNFPW-1 and PNFPW-2, using a complete truth table. And the optimal function should be chosen based on the results of minimization of two normal forms – PNFPW-1 and PNFPW-2.

5. A comparative analysis of the results of the simplification of Boolean functions in the Peirce-Webb basis by a non-standard system and examples of the simplification of functions by the Veitch diagram, matrix and graph algorithms, methods for algebraic bi-decomposition and bitwise partitioning of the set of initial conjuncterms was carried out. The non-standard system shows the same result of simplifying the functions in the main basis, but a better result in the Peirce-Webb basis. From the comparative analysis, it follows that the non-standard system, in contrast to the replacement of AND and OR logic elements with OR-NOT or AND-NOT elements, makes it possible to obtain a simpler Boolean function during the transition from the basic basis to the Peirce-Webb basis. For all the considered examples, the simplification procedure with a non-standard system is simpler. This makes it possible, to a certain extent, to simplify functions without the use of automated calculations.

Conflicts of interest

The authors declare that they have no conflicts of interest in relation to the current study, including financial, personal, authorship, or any other, that could affect the study, as well as the results reported in this paper.

Funding

The study was conducted without financial support.

Data availability

All data are available, either in numerical or graphical form, in the main text of the manuscript.

Use of artificial intelligence

The authors confirm that they did not use artificial intelligence technologies when creating the current work.

References

1. Pucknell, D. A. (1990). Fundamentals of Digital Logic Design, With VLSI Circuit applications. Prentice Hall, 472.
2. Mano, M., Kime, C. (2004). Logic and Computer Design Fundamentals. Prentice Hall.
3. Baranov, S. (2008). Logic and System Design of Digital Systems. Tallinn: TUT Press.
4. Micheli, G. (1994). Synthesis and Optimization of Digital Circuits. McGraw-Hill.
5. Zakrevskij, A., Pottosin, Yu., Cheremisinova, L. (2009). Optimization in Boolean Space. Tallinn: TUT Press.
6. Baranov, S., Karatkevich, A. (2018). On Transformation of a Logical Circuit to a Circuit with NAND and NOR Gates Only. INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 64 (3), 373–378. Available at: https://journals.pan.pl/dlibra/publication/123535/edition/107750/content
7. Maxfield, M. (2018). Implementing and Converting Logic Circuits Using Only NAND or NOR Gates. Available at: https://www.eeweb.com/implementing-logic-functions-using-only-nand-or-nor-gates/
8. Poomiga, M., Ananthi, M., Sinega, M., Aashika, S M., Nambi Rajan, M. (2021). Optimization of simple combinational universal logic gates. International Research Journal of Modernization in Engineering Technology and Science, 03 (08), 1000–1006. Available at: https://www.irjmets.com/uploadedfiles/paper/volume_3/issue_8_august_2021/15911/final/fin_irjmets1630308031.pdf
9. Olenev, A., Potekhina, E., Khabarov, A., Zvereva, L. (2024). Information and logical transformations in Schaeffer and Pierce Bases in Maple. ITM Web of Conferences, 59, 02006. https://doi.org/10.1051/itmconf/20245902006
10. Menshikh, V. V., Nikitenko, V. A. (2022). Minimization of representations of the logical function in schaeffer and pierce bases. Bulletin of the South Ural State University Series "Mathematics. Mechanics. Physics", 14 (4), 20–27. https://doi.org/10.14529/mmph220403
11. Barland, I. (2012). An algorithm to implement a boolean function using only NAND's or only NOR's. Rice University. Available at: https://archive.org/details/cnx-org-col10347/page/n1/mode/2up
12. Rajaei, A., Houshmand, M., Rouhani, M. (2011). Optimization of Combinational Logic Circuits Using NAND Gates and Genetic Programming. Soft Computing in Industrial Applications, 405–414. https://doi.org/10.1007/978-3-642-20505-7_36
13. Dychka, I. A., Tarasenko, V. P., Onai, M. V. (2019). Osnovy prykladnoi teorii tsyfrovykh avtomativ. Kyiv, 505.
14. Kalyadin, N. I. (2006). Praktikum po diskretnoy matematike (Chast' IV. Minimizatsiya FAL). Izhevsk: Izd-vo IzhGTU, 48.
15. 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. https://doi.org/10.15587/2312-8372.2017.118336
16. Solomko, M., Antoniuk, M., Voitovych, I., Ulianovska, Y., Pavlova, N., Biletskyi, V. (2023). Implementing the method of figurative transformations to minimize partially defined Boolean functions. Eastern-European Journal of Enterprise Technologies, 1 (4 (121)), 6–25. https://doi.org/10.15587/1729-4061.2023.273293
17. Solomko, M. (2024). Development of a non-standard system for simplifying boolean functions. Eastern-European Journal of Enterprise Technologies, 3 (4 (129)), 6–34. https://doi.org/10.15587/1729-4061.2024.305826
18. Kondratenko, N. R. (2010). Kompiuternyi praktykum z matematychnoi lohiky. Vinnytsia: VNTU, 117.
19. 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. https://doi.org/10.15587/2312-8372.2018.146312
20. Ritsar, B. E. (1997). Metod minimizatsii bulevyh funktsiy. Problemy upravleniya i informatiki, 3, 100–113.
21. Cheng, D. (2005). Semi-tensor Product of Matrices and its Applications to Dynamic Systems. New Directions and Applications in Control Theory, 61–79. https://doi.org/10.1007/10984413_5
22. Feng, J., Zhao, R., Cui, Y. (2022). Simplification of logical functions with application to circuits. Electronic Research Archive, 30 (9), 3320–3336. https://doi.org/10.3934/era.2022168
23. Cortadella, J. (2003). Timing-driven logic bi-decomposition. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 22 (6), 675–685. https://doi.org/10.1109/tcad.2003.811447
24. Mishchenko, A., Steinbach, B., Perkowski, M. (2001). An algorithm for bi-decomposition of logic functions. Proceedings of the 38th Conference on Design Automation-DAC'01, 103–108. https://doi.org/10.1145/378239.378353
25. Chang S.-C., Marek-Sadowdka, M., Hwang, T. (1996). Technology mapping for TLU FPGAs based on decomposition of binary decision diagrams. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 15 (10), 1226–1236. https://doi.org/10.1109/43.541442
26. Pottosin, Yu. V. (2022). Synthesis of combinational circuits by means of bi-decomposition of Boolean functions. Informatics, 19 (1), 7–18. https://doi.org/10.37661/1816-0301-2022-19-1-7-18
27. Solomko, M., Batyshkina, I., Khomiuk, N., Ivashchuk, Y., Shevtsova, N. (2021). Developing the minimization of a polynomial normal form of boolean functions by the method of figurative transformations. Eastern-European Journal of Enterprise Technologies, 2 (4 (110)), 22–37. https://doi.org/10.15587/1729-4061.2021.229786
28. Minziuk, V. (2023). Method of minimizing boolean functions for designing digital combinational circuits. Information and Communication Technologies, Electronic Engineering, 3 (1), 146–153. https://doi.org/10.23939/ictee2023.01.146

29. Solomko, M., Khomiuk, N., Ivashchuk, Y., Nazaruk, V., Reinska, V., Zubyk, L., Popova, A. (2020). Implementation of the method of image transformations for minimizing the Sheffer functions. Eastern-European Journal of Enterprise Technologies, 5 (4 (107)), 19–34. https://doi.org/10.15587/1729-4061.2020.214899
30. Solomko, M. (2021). Developing an algorithm to minimize boolean functions for the visual-matrix form of the analytical method. Eastern-European Journal of Enterprise Technologies, 1 (4 (109)), 6–21. https://doi.org/10.15587/1729-4061.2021.225325
31. 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. https://doi.org/10.15587/1729-4061.2020.206308
32. Solomko, M., Tadeyev, P., Zubyk, L., Babych, S., Mala, Y., Voitovych, O. (2021). Implementation of the method of figurative transformations to minimizing symmetric Boolean functions. Eastern-European Journal of Enterprise Technologies, 4 (4 (112)), 23–39. https://doi.org/10.15587/1729-4061.2021.239149