The object of this study is models of low-power digital logic circuits. The problem being solved is the effectiveness of the technique for simplifying Boolean functions to obtain optimal structures of logic circuits. A new theorem of a non-standard system of simplification of Boolean functions has been formulated, according to which in order to obtain a minimal function it will suffice to perform all non-redundant operations of simple and/or super-gluing of variables, which ultimately provides a minimal function in the main basis without using an implicant table. Thus, the problem of simplifying Boolean functions to the simplest normal equivalent is solved in one step. The interpretation of the result is that the properties of 2-(n, b)-design combinatorial systems make it possible to reproduce the definition of logical operations of super-gluing variables, to represent logical operations in a different way, and vice versa. This, in turn, ensures the establishment of the locations of equivalent transformations on the binary structure of the truth table and the implication of a systematic procedure for simplifying Boolean functions by an analytical method. Special feature of the results is that unambiguous identification of the locations of equivalent transformations is possible even when different intervals of the Boolean space containing the 2-(n, b)-design systems have common modules.

It has been experimentally confirmed that the non-standard system improves the efficiency of simplifying Boolean functions, including partially defined ones, by 200–300 % compared to analogs.

In terms of application, a non-standard system for simplifying Boolean functions will 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: simplification of Boolean functions, non-standard system, intervals of Boolean space, location of equivalent transformations

UDC 519.718

DOI: 10.15587/1729-4061.2024.305826

DEVELOPMENT OF A NON-STANDARD SYSTEM FOR SIMPLIFYING BOOLEAN FUNCTIONS

Mykhailo Solomko
PhD, Associate Professor
Department of Computer Engineering
National University of Water and Environmental Engineering
Soborna str., 11,
Rivne, Ukraine, 33028
E-mail: doctrinas@ukr.net

Received date 01.04.2024
Accepted date 06.06.2024
Published date 28.06.2024

How to Cite: *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*

1. Introduction

The digital industry implies a management system (road map), which represents the unity of theoretical, production, and economic processes and connections in the movement of production funds. This process is continuous and purposeful, so the management system must be controlled and managed. For example, the technology of manufacturing microcircuits becomes more expensive with each new reduction in the size of semiconductor elements. Even when Moore's law ceased to be relevant due to the fact that the reduction of silicon transistors approached physical limits, it turned out that improving the KMON technology remained more profitable than looking for something fundamentally new.

GlobalFoundries is rebuilding its advanced FinFET roadmap to serve the new wave of customers that will adopt the technology in the coming years. The company will shift development resources to make its 14/12 nm FinFET platform more relevant to these customers by providing a range of innovative IT solutions and features. To support this transition, GF is suspending its 7 nm FinFET program indefinitely and restructuring research teams to support its expanded portfolio initiatives.

The race to physically reduce the size of transistors for digital devices has stopped but the performance of chips is increasing. This, in particular, is due to the use of 3-dimensional integration, photonic crystals, including the results of designing based on logical functions. Measures to simplify logic functions for the design of digital circuits will always be used since low power supply, cheapness, and speed of calculation are important, and are universal and relevant parameters, regardless of the dynamics of the industry, when design standards are reduced, or production programs are introduced that are implemented outside the boundaries of Moore's law.

The logical operations of the analytical method for the equivalent transformation of Boolean functions are neither systematic nor suitable for computer implementation. In addition to the above, this technique takes more time to simplify.

Possible measures to overcome the problem of implementing the simplification of Boolean functions by an analytical method are the use of hermeneutics [1, 2], metadata [3], established locations of equivalent transformations [4], implication of the algorithm of simplification of Boolean functions. As a result, verbal procedures of algebraic transformations are replaced by equivalent figurative transformations.

The figurative form of information, in particular, in the form of combinatorial objects, passes possibly accidentally from the tendency of causality into the necessary-valid tendency of the sequence of a non-standard procedure, which in the end should provide more chances to determine the algorithm for simplifying Boolean functions when they are represented in a visual matrix form. The combinatorial objects in this case will be 2-dimensional complete 2-(n, b)-design, and/or incomplete 2-$(n, x/b)$-design systems with repetition and are actually combinatorial images. The model of non-standard

simplification of Boolean functions, which needs to be represented, admits the peculiarity that there is some analogy of the algorithm, which transforms the «messy» complexity of the simplification procedure by the analytical method into a complex order of figurative transformations.

Given this, a topical aspect of theoretical scientific research on the non-standard system of simplification of Boolean functions is studies aimed, in particular, at:

– obtaining a new result when simplifying Boolean functions;

– measurement and evaluation of the results of the implementation of a non-standard simplification system in the real synthesis of digital components with their further application in digital technologies;

– cost optimization of the technology of minimization of Boolean functions;

– ensuring the reliability of the obtained minimal forms of Boolean functions using a non-standard system of simplifying functions.

2. Literature review and problem statement

A new method for simplifying Boolean functions based on a directed selection of possible minimization paths according to the criteria of a necessary and sufficient condition is represented in [5]. It is noted that the proposed method is an improvement of the methods of simplification by parts, parallel decomposition, and other methods. The considered method was developed on the basis of a new way of recording individual Boolean functions in the form of indices of significant rows of the truth table. The advantage of the method is the two-stage simplification of the specified functions, which makes it possible not to use the directed sorting criterion directly. When implementing the method, only single variable values are processed in the columns of the truth table, which reduces the amount of data processing. The machine implementation of the method uses the parallelization of the function simplification procedure. To confirm the theoretical statements of the reported method [5], it is advisable to represent the results of the simplification of the Boolean function in a demonstration example.

A new algorithm for finding simple implicants, which uses the ideas of Quine and McCluskey in a more thorough implementation, is considered in [6]. The algorithm has a complexity of $O(3^n n)$ bit operations on any function of n variables, which is further reduced by implementing a bit-fragment style that makes maximum use of processor registers. A new implementation based on the bit slicing technique is proposed, which is particularly applicable in the implementations and designs of symmetric key cryptographic primitives such as block ciphers and is called DenseQMC. The idea is that processor instructions operate on entire registers rather than individual bits. The most common size of the general register is $\omega=64$ bits, although there are even vector expansions handling more bits at once. In essence, you can perform a bitwise operation on a register, such as OR, AND, NOT ω bits in parallel using a single instruction. All such bit streams are always aligned by the least significant bit of the register and complemented to the size of the register. When these bit streams are stored in memory, they are called modules. Unused bits in modules are not used (that is, wasted). Therefore, it is desirable to choose a block size equal to a small multiple of the register size, close to the maximum capacity. Best practice is 256-bit modules. For this block size, only 5 % of memory is wasted. This is a small multiple of the 64-bit register size provided by modern high-performance CPUs and also corresponds to the bit size of the AVX2 vector extension. These bitwise operations require, however, a regular data structure.

The new approach expands the possible number of n input variables of the function to $n=23$ when performing the task on a laptop and to $n=27$ when working on a server equipped with 1 TiB of RAM. The efficient solution of the first step of the Quine-McCluskey algorithm opens up a prospect for fast heuristic approximate methods for the second step of the algorithm.

Work [6] gives a description of a common error in implementations of the Quine-McCluskey algorithm, which leads to a quadratic slowdown of calculations. A corrected implementation of the Quine-McCluskey algorithm in the form of a classical approach – SparseQMC – is also presented and freely available.

The general problem of calculating the smallest simple representation of a non-clausal propositional formula, which is called formula simplification, is considered in [7]. In addition to the above, the paper proposes a new, fully SAT-based approach to the problem of formula simplification. In this regard, the original problem posed by the Quine-McCluskey procedure can be considered as a special case of the problem discussed in this paper. Experimental results demonstrate that the SAT-based approach to formula simplification is a viable alternative to existing implementations of the Quine-McCluskey procedure.

A typical Quine-McCluskey implementation starts by computing all prime implicants of the CNF formula or DNF, and then implements a cover with the minimum number of prime implicants equivalent to the original function. A more general scenario is when the original formula is non-clausal. Obviously, it is still possible to generate all the implicants of the formula, then generate all the prime implicants, and then do the minimal cover. However, in practice, there may be a number of implications far greater than the number of simple implicants. Here I can add that the apparatus for simplifying the non-clausal formula is sufficiently developed, in addition to the implementation of optimal coverage. It should also be noted that work [7] does not provide an acceptable demonstration example. This makes it impossible to quickly compare and evaluate the method from [7] with other simplification methods.

The analysis of three possible (practical) approaches to solving the problem of obtaining minimal propositional formulas in conjunctive normal format (CNF) or in disjunctive normal format (DNF) for their use in hardware design is considered in paper [8]. First, the use of brute force (Quine-McCluskey algorithm). To do this, all possible formulas are generated in increasing size and checked whether they are equivalent to the original formula. Second, generation of Zeitin coding for all formulas and verification of equivalence with the original using SAT solvers. Third, encoding the problem as a Quantitative Boolean Formula (QBF) using a QBF solver. The results show that the QBF approach significantly outperforms the other two. Therefore, QBF solvers represent a state-of-the-art solution for Boolean minimization.

The comparison criterion represents the average and median time required for the three algorithms relative to the size of the original formula and the resulting minimized formula. Paper [8] also notes that although in most cases (given the median) the three algorithms minimize the formula in less than

one second, for function sizes less than 20 (variables), only a few cases force the SAT basis to require an average about 1 hour of time when the function size is close to 20 (variables).

The practice of hardware design of digital devices, however, involves a management system that represents the unity of production and economic processes and connections in the movement of production funds. This process is continuous and purposeful, so the management system must be controlled and managed. The algorithm for simplifying Boolean functions belongs to the project and production management system. Let some algorithm simplifies the function faster by 1 hour. For strategic management, this gives a temporary advantage because the competitive pressing after 1 hour will catch up with the production and make it uncontrollable, and therefore the management system will not be continuous and focused. A fast algorithm becomes a fakir for 1 hour. In contrast to the execution time of the algorithm, the criterion of optimal simplification of the function directly affects the area of the chip, power consumption, and heat generation of the digital device. In this regard, the quality criterion of simplification provides an innovative process, and therefore forms a stable unity of production and economic processes for hardware design.

Work [8] does not provide an acceptable demonstration example, which does not make it possible to quickly compare and evaluate the accuracy of the simplification of logical formulas by the presented method with other methods.

A new heuristic technique for simplifying Boolean functions is reported in [9]. In general, the exact Boolean simplification problem is NP-complex, even – complete. Therefore, with an increase in the bitness of the function, exact approaches quickly become infeasible. Therefore, simplification methods in this area are often a compromise between the quality of the result and the execution time. With the help of algebraic coding, the minimization problem is transferred to the algebraic domain, in which the algorithms for calculations are applicable on the Graebner basis. An equivalent, more compact version of the Boolean formula is reconstructed in the Graebner basis. It is noted in [9] that the considered approach is the first one that uses the Graebner basis for simplifying Boolean functions. Given an input formula in DNF, the method encodes it as a system of equations. The resulting Graebner basis can thus again be interpreted as an empirically smaller equivalent Boolean formula. The method is especially suitable for formulas that contain many XOR operations. It is also noted in [9] that currently the considered algorithm can process functions of up to 20 input bits, while exact synthesis can only process up to 8 input bits. Empirically, the algorithm produces smaller formulas compared to two-level minimization by the ESPRESSO algorithm.

The method for simplifying the terms of a Boolean function in the context of a formal, axiomatically defined theory is considered in [10]. The key idea of the method is to represent large, even infinite sets of terms using special data structures, which makes it possible to apply axioms to sets as a whole, rather than to individual terms. In this case, the terms can be simplified (i.e., minimized) in linear time. The method is demonstrated for Boolean terms with a small number of variables. Also, in [10], a «targeted» algorithm is proposed, which calculates only small parts of the underlying theory in order to simplify a specific term. Simplifying a term means choosing the simplest term in the equivalence class. It is shown that the algorithm is able to simplify Boolean terms with a much larger number of variables, but optimality can no longer be certified. Considering the presented method, it is obvious that the results of simplification [10] will be similar to the results of applying the heuristic technique for simplifying Boolean functions.

Obtaining all simple implicants of the Boolean function by expanding and simplifying the format of the product of the sum of literals is demonstrated in [11]. This gives a new method for generating all prime implicants. A simple algorithm for obtaining one minimal set of simple implicants from all possible simple implicants without using minterms is presented. Examples are given to illustrate the method. The new approach has two important advantages:

1) it can use any form of representation of the underlying basis function (no minterms or maxterms are required, which are usually too many);

2) it makes it possible to break a large function into smaller functions and manipulate them.

After all simple implicants are generated, the next task is to obtain one minimal set of them so as to cover all the minterms of the function. The presented method is noticeably different from many known methods for simplifying Boolean functions, in particular, not all rules of Boolean algebra are used here. The specified feature of the method can reduce the entropy of the recommendations regarding the rational search for minimal dead-end forms of Boolean functions by the analytical method.

Paper [12] considers the use of parallel computing for the implementation of the Quine-McCluskey algorithm. The Quine-McCluskey algorithm has naturally parallel parts that can be implemented on a SIMT architecture, and thus on a GPGPU, to find simple implicants. The parallel part of the algorithm concerns the process of combining strings. The problem calculated by this algorithm is NP-complex, and the execution time of the algorithm increases exponentially with the increase in the number of variables. The goal is to prove that the parallel implementation of the Quine-McCluskey algorithm on graphics processors (GPUs) gives a significant acceleration of the computational process. Algorithm parallelization is implemented using Compute Unified Device Architecture (CUDA), which is a parallel computing platform and programming model created by NVIDIA and implemented by the graphics processing unit (GPU).

The synthesis of partial specifications of logical circuits (LSFPS) is presented in [13]. LSFPS is the problem of finding a hardware implementation of partially defined Boolean functions. Logical Synthesis from Partial Specifications (LSFPS) introduces the additional concept of «don't know» to terms that are not in specifications. The exact solution to LSFPS is the optimal size logic scheme of the corresponding problem in which the undefined sets of variables are invalid. In practice, specification information may not be sufficient to determine exact functionality, so the goal becomes maximizing the accuracy of the scheme over the available subset of uncertain sets. Therefore, to the traditional goal of minimizing the size of the scheme, the goal of maximizing the accuracy of the scheme when evaluated by a subset of uncertain sets is added. The problem is relevant because effective solutions can lead to hardware-friendly machine learning models that do not rely on black-box approaches. LSFPS directly addresses the problem of automatically generating optimal topologies for binary neural networks. In addition, the combination of an exact solution with modern methods for logical synthesis unlocks unprecedented opportunities for optimization. Previous works have proven the effectiveness of approximate logic synthesis (ALS) for designing circuits with

sufficient accuracy. However, these methods sacrifice specification accuracy, which excludes them from being legitimate candidates for LSFPS. Paper [13] proposes the restoration of accuracy, which involves the procedure of comparing an approximate version of the scheme with a new one that satisfies the exact functionality of the specifications. Experimental testing showed a reduction in the number of gates by 17.38 % and a reduction in the depth of the logic circuit by 12.02 %. The use of the procedure for restoring the accuracy of the scheme based on the decomposition gives an accuracy of 95.73 %, which exceeds the current level of ALS at which the accuracy is 92.76 %.

Given the extensive research on logic synthesis for high-performance systems, its potential role in the development of hardware-aware machine learning techniques needs to be explored. It is worth noting that some machine learning tasks allow formulation as a fundamental problem of logical synthesis.

The problem of finding an approximate Boolean scheme from a set of examples is considered in [14]. Many computer programs are inherently error-tolerant. This makes it possible to reduce the precision of calculations to achieve greater efficiency for chip area, computational performance, and/or power consumption. In recent years, a number of automated methods for approximate calculations have been proposed; however, most of these methods require full knowledge of the exact or «golden» description of the circuit. At the same time, there is considerable interest in the synthesis of calculations based on examples, a form of supervised learning. Paper [14] presents the relationship between supervised learning of Boolean schemes and existing work on the synthesis of incompletely defined Boolean functions. When viewed through the lens of machine learning, the latter work has been shown to provide good training accuracy but low test accuracy. The paper compares with previous work from the 1990s that uses mutual information to guide the search process, aiming for good generalization. By combining this early work with the current approach to learning logic functions, a scalable and efficient machine learning approach for Boolean circuits can be achieved in terms of area/delay/test-error trade-off. The results of the study indicate that the proposed method has the potential to generate Boolean schemes with high accuracy from large training sets of examples with a large amount of primary input data. However, this method is limited to only 1-bit output. New research is needed that will take into account word-level searches that will help efficiently search logic values of a circuit with many bit outputs.

The method for minimizing Boolean functions, which is based on nonlinear mixed integer programming, is reported in [15]. Experimental results show that the method gives the same or better results compared to other methods available in the literature. However, other methods do not guarantee obtaining a minimal solution. The main advantages of the proposed method for minimization are that the presented method guarantees obtaining a minimum function and can also be used to minimize incompletely defined Boolean functions.

To confirm the theoretical statements of the reported method [15], it is advisable to present a demonstration example of the simplification of a partially defined Boolean function by at least 4 variables.

In [15], it is also stated that all experimental examples were run on the NEOS server with free access, which implements deterministic algorithms. However, the NEOS free access service limits the maximum calculation time to 8 hours, which is not enough to complete some of the examples. Therefore, for such examples, the final solution is not found, but instead the best solution is found.

The methods for simplifying Boolean functions, which are considered in the literature [5–15], mainly demonstrate the use of an additional mathematical apparatus for solving the Boolean problem, such as the DenseQMC cut bit technique; a fully SAT-based approach to the problem of simplifying a non-clausal propositional formula; checking equivalence with the original using QBF solvers; transformation of the problem from the domain of Boolean algebra to the classical algebraic domain, in which algorithms for calculations are applicable on the Graebner basis; the technique of simplifying the terms of a Boolean function in the context of a formal, axiomatically defined theory; parallel calculations for the implementation of the Quine-McCluskey algorithm implemented by a graphics processor (GPU). A mandatory technological point for the implementation of the specified algorithms and methods is the automation of calculations.

But additional mathematical apparatus complicates the method. In turn, the increase in the complexity of the simplification of Boolean functions is accompanied by positive and negative factors that can affect the innovative capacity of the simplification technology in the future.

The problem is the «trigger causality» of the possible consequence of the complication of the method, which is important for any IT production in any situation, in particular:

1) gradually increasing the potential of the innovation process and transforming it into an innovation;

2) an additional mathematical apparatus complicates the method to the point of self-denial, when only a mathematical technique remains, unable to solve the Boolean problem in practice.

In other words, is there a life cycle for a complex innovation after methods for simplifying functions based, in particular, on the use of:

– transfer of the minimization problem to the algebraic domain, in which algorithms for calculations are applicable on the Graebner basis [9];

– simplification of the terms of the Boolean function in the context of a formal, axiomatically defined theory [10];

– machine learning techniques [13];

– techniques for finding approximate Boolean schemes, large training sets of examples with a large number of primary input data, methods for restoring the accuracy of a logical scheme [14];

– nonlinear mixed integer programming [15], and others.

The non-standard Boolean simplification system is based on binary systems with repetition, 2-(n, b)-design, 2-$(n, x/b)$-design, which are both combinatorial objects that can be defined, set, and logical operations, which can be carried out. Such properties of the 2-(n, b)-design, 2-$(n, x/b)$-design systems ensure the unambiguous identification of the locations of equivalent transformations and the implication of a systematic procedure for simplifying Boolean functions. This, in turn, makes it possible to reduce the complexity of simplification without loss of functionality, compared to the algorithms and methods for simplifying Boolean functions considered in works [5–15]. The non-standard simplification system uses the visual-matrix form of the analytical method [16] and does not exclude the manual method for simplifying Boolean functions.

Thus, algorithms and methods, software tools created for them [5–15] and a non-standard system for simplifying Boolean functions have excellent approaches (principles). And so they see different prospects regarding the possibility of algorithmic simplification of Boolean functions.

And this is a reason to believe that the software-technological base, which is represented by algorithms and methods with additional mathematical apparatus for solving the Boolean problem [5–15], is insufficient for conducting theoretical research on the optimal simplification of Boolean functions. This predetermines the need to carry out research using a non-standard system for simplifying Boolean functions.

In terms of application, a non-standard system for simplifying Boolean functions could ensure the development of the innovation process and the transfer of innovations to material production: from decision-making, conducting fundamental research, expanding the capabilities of digital component design technology based on Boolean functions in the main $\{\vee, \wedge, \neg\}$ and polynomial $\{\wedge, \oplus, 1\}$ bases, construction of prototypes, their testing, to the organization of serial or mass production of novelties and their implementation.

3. The aim and objectives of the study

The aim of my work is to extend the non-standard system for the simplification of disjunctive normal forms (DNF), conjunctive normal forms (CNF) and polynomial normal forms (PNF) of partially and fully defined Boolean functions. This will make it possible to simplify and increase the productivity of simplifying Boolean functions in the basic, polynomial, and other bases, using their algebraic apparatus.

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

– to demonstrate the property of combinatorial systems with repeated 2-(*n, b*)-design to the ability to reproduce (represent) the definition of logical super-gluing operations of variables in the form of changing their own definition in the process of interaction with them;

– to establish the implication between combinatorial objects, 2-(*n, b*)-design, 2-(*n, x/b*)-design, of the truth table and equivalent transformations with a non-standard system of simplification of Boolean functions;

– to state a theorem of a non-standard system for simplification of Boolean functions;

– to define the thesaurus of a non-standard system of simplifying Boolean functions;

– to conduct a comparative analysis of the results of the simplification of Boolean functions by a non-standard system and examples of the simplification of functions by the evolutionary method, genetic, swarm algorithms, and the method for directed sorting in order to compare the cost of implementing the minimum function.

4. The study materials and methods

The object of my research is models of optimal logic circuits with a small chip area, power supply, and low heat generation.

It should be expected that the regular and constant application of the visual-matrix form of the analytical method for the simplification of Boolean functions will make it possible to provide a one-step simplification procedure and formulate the corresponding theorem.

The term «Binary combinatorial systems with repetition, 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».

Boolean. Some given set A implies the Boolean $M(A)$, which is the set of all subsets of the set A. $M_k(A)$ shall denote the Boolean, which is the set of all subsets of A containing k elements. Let $A=\{a, b, c, d, e\}$, then:

$$M(A)=\left\{\begin{array}{l}\{a\},\{b\},\{c\},\{d\},\{e\},\{a,b\},\{a,c\},\{a,d\},\\ \{a,e\},\{b,c\},\{b,d\},\{b,e\},\{c,d\},\{c,e\},\\ \{d,e\},\{a,b,c\},\{a,b,d\},\{a,b,e\},\{a,c,d\},\\ \{a,c,e\},\{a,d,e\},\{b,c,d\},\{b,c,e\},\{b,d,e\},\\ \{c,d,e\},\{a,b,c,d\},\{a,b,c,e\},\{a,c,d,e\},\\ \{a,b,d,e\},\{b,c,d,e\}\{a,b,c,d,e\},\varnothing.\end{array}\right\}.$$

$$M_2(A)=\left\{\begin{array}{l}\{a,b\},\{a,c\},\{a,d\},\{a,e\},\{b,c\},\\ \{b,d\},\{b,e\},\{c,d\},\{c,e\},\{d,e\}\end{array}\right\}.$$

$$M_3(A)=\left\{\begin{array}{l}\{a,b,c\},\{a,b,d\},\{a,b,e\},\{a,c,d\},\{a,c,e\},\\ \{a,d,e\},\{b,c,d\},\{b,c,e\},\{b,d,e\},\{c,d,e\}\end{array}\right\}.$$

The number of all k-element subsets of a set of n elements is:

$$N\left(M_k(A)\right)=C_n^k=\frac{n!}{k!(n-k)!}.$$

There is also equality:

$$\sum_{k=0}^{n}C_n^k=2^n. \quad (1)$$

Since C_n^k is the number of k-element subsets of a set of n elements, the sum in the left part of expression (1) determines the number of all subsets, the set $A=\{a, b, c, d, e\}$. Thus, the number of all subsets of set A will be:

$$N\left(M(A)\right)=C_5^0+C_5^1+C_5^2+C_5^3+C_5^4+C_5^5=$$
$$=1+5+10+10+5+1=32=2^5.$$

Boolean configuration. The set $A=\{a, b, c, d, e\}$, in addition to enumerating its elements, can also determine the numbers of the positions where the element α is located. For example, a can define the first position, b the second position on the set $A=\{a, b, c, d, e\}$, etc. Subsets of the set $A=\{a, b, c, d, e\}$, in this case, are the subsets containing the element α in k positions, $k=0, ..., n$, where n is the number of positions in the set A. In the general case, the element α can occupy several positions in the set A, which means that the element α is repeated in the set A.

The positions in which the element α is present are denoted by one ($\alpha=1$). Positions in which the element α is missing are marked with zero ($\alpha=0$). Then the Boolean configuration of the set $A=\{a, b, c, d, e\}$ will take the form:

$$\begin{array}{llll}(0_1,0_2,0_3,0_4,0_5); & (0_1,1_2,0_3,0_4,0_5); & (1_1,0_2,0_3,0_4,0_5); & (1_1,1_2,0_3,0_4,0_5);\\ (0_1,0_2,0_3,0_4,1_5); & (0_1,1_2,0_3,0_4,1_5); & (1_1,0_2,0_3,0_4,1_5); & (1_1,1_2,0_3,0_4,1_5);\\ (0_1,0_2,0_3,1_4,0_5); & (0_1,1_2,0_3,1_4,0_5); & (1_1,0_2,0_3,1_4,0_5); & (1_1,1_2,0_3,1_4,0_5);\\ (0_1,0_2,0_3,1_4,1_5); & (0_1,1_2,0_3,1_4,1_5); & (1_1,0_2,0_3,1_4,1_5); & (1_1,1_2,0_3,1_4,1_5);\\ (0_1,0_2,1_3,0_4,0_5); & (0_1,1_2,1_3,0_4,0_5); & (1_1,0_2,1_3,0_4,0_5); & (1_1,1_2,1_3,0_4,0_5);\\ (0_1,0_2,1_3,0_4,1_5); & (0_1,1_2,1_3,0_4,1_5); & (1_1,0_2,1_3,0_4,1_5); & (1_1,1_2,1_3,0_4,1_5);\\ (0_1,0_2,1_3,1_4,0_5); & (0_1,1_2,1_3,1_4,0_5); & (1_1,0_2,1_3,1_4,0_5); & (1_1,1_2,1_3,1_4,0_5);\\ (0_1,0_2,1_3,1_4,1_5); & (0_1,1_2,1_3,1_4,1_5); & (1_1,0_2,1_3,1_4,1_5); & (1_1,1_2,1_3,1_4,1_5).\end{array} \quad (2)$$

From an examination of the configuration of Boolean (2), it follows that it forms a complete combinatorial system with the repetition of the element α, which I denote by:

2-(n, b)-design,

where n is the bit rate of the system block; b is the number of modules of the complete system, which is determined by the formula $b=2^n$, the number 2 in front of the brackets means the binary structure of the configuration (2). For example, 2-(5, 32)-design is a complete binary combinatorial system with repetition consisting of 5-bit modules, the number of modules is 32.

In the general case, the configuration of the truth table of a given Boolean function, in addition to the submatrix of the complete combinatorial system with repetition, 2-(n, b)-design, also contains the submatrices of the incomplete combinatorial system with repetition:

2-(n, x/b)-design.

In this case, x is the number of modules of an incomplete combinatorial system with repetition. For example, 2-(3,7/8)-design is an incomplete binary combinatorial system with repetition consisting of 3-bit modules, the number of modules is 7. Properties of an incomplete combinatorial system with repetition 2-(n, x/b)-design also makes it possible to set rules that, in the general case, ensure effective minimization of Boolean functions.

Visual matrix form. On the Boolean configuration (2), I shall replace the variables:

– replace 1_n with x_n;

– 0_n is replaced by $\overline{x}_n$.

Here, n is an index that determines the number of bits of the literal «x_n» or «$\overline{x}_n$» in a logical function minterm. After replacing the variables with Boolean configurations (2), I get the visual-matrix form of the 5-bit Boolean function:

$$
\begin{array}{cccc}
(\overline{x_1},\overline{x_2},\overline{x_3},\overline{x_4},\overline{x_5}); & (\overline{x_1},x_2,\overline{x_3},\overline{x_4},\overline{x_5}); & (x_1,\overline{x_2},\overline{x_3},\overline{x_4},\overline{x_5}); & (x_1,x_2,\overline{x_3},\overline{x_4},\overline{x_5});\\
(\overline{x_1},\overline{x_2},\overline{x_3},\overline{x_4},x_5); & (\overline{x_1},x_2,\overline{x_3},\overline{x_4},x_5); & (x_1,\overline{x_2},\overline{x_3},\overline{x_4},x_5); & (x_1,x_2,\overline{x_3},\overline{x_4},x_5);\\
(\overline{x_1},\overline{x_2},\overline{x_3},x_4,\overline{x_5}); & (\overline{x_1},x_2,\overline{x_3},x_4,\overline{x_5}); & (x_1,\overline{x_2},\overline{x_3},x_4,\overline{x_5}); & (x_1,x_2,\overline{x_3},x_4,\overline{x_5});\\
(\overline{x_1},\overline{x_2},\overline{x_3},x_4,x_5); & (\overline{x_1},x_2,\overline{x_3},x_4,x_5); & (x_1,\overline{x_2},\overline{x_3},x_4,x_5); & (x_1,x_2,\overline{x_3},x_4,x_5);\\
(\overline{x_1},\overline{x_2},x_3,\overline{x_4},\overline{x_5}); & (\overline{x_1},x_2,x_3,\overline{x_4},\overline{x_5}); & (x_1,\overline{x_2},x_3,\overline{x_4},\overline{x_5}); & (x_1,x_2,x_3,\overline{x_4},\overline{x_5});\\
(\overline{x_1},\overline{x_2},x_3,\overline{x_4},x_5); & (\overline{x_1},x_2,x_3,\overline{x_4},x_5); & (x_1,\overline{x_2},x_3,\overline{x_4},x_5); & (x_1,x_2,x_3,\overline{x_4},x_5);\\
(\overline{x_1},\overline{x_2},x_3,x_4,\overline{x_5}); & (\overline{x_1},x_2,x_3,x_4,\overline{x_5}); & (x_1,\overline{x_2},x_3,x_4,\overline{x_5}); & (x_1,x_2,x_3,x_4,\overline{x_5});\\
(\overline{x_1},\overline{x_2},x_3,x_4,x_5); & (\overline{x_1},x_2,x_3,x_4,x_5); & (x_1,\overline{x_2},x_3,x_4,x_5); & (x_1,x_2,x_3,x_4,x_5).
\end{array}
\qquad (4)
$$

It should be noted that the visual-matrix form (4) represents the complete perfect disjunctive normal form (PDNF) of a 5-bit Boolean function. Therefore, the PDNF of the Boolean function (4) uniquely defines a complete combinatorial system with the repetition, 2-(n, b)-design (2), and vice versa [17].

It is known that the reduction of a complete PDNF of logical function always gives unity. For example, the reduction of a 3-bit full PDNF of Boolean function takes the form:

$$
\begin{aligned}
&\overline{x_1}\,\overline{x_2}\,\overline{x_3}+\overline{x_1}\,\overline{x_2}x_3+\overline{x_1}x_2\overline{x_3}+\overline{x_1}x_2x_3+\\
&+x_1\overline{x_2}\,\overline{x_3}+x_1\overline{x_2}x_3+x_1x_2\overline{x_3}+x_1x_2x_3=\\
&=\overline{x_1}\,\overline{x_2}\left(\overline{x_3}+x_3\right)+\overline{x_1}x_2\left(\overline{x_3}+x_3\right)+\\
&+x_1\overline{x_2}\left(\overline{x_3}+x_3\right)+x_1x_2\left(\overline{x_3}+x_3\right)=\\
&=\overline{x_1}\,\overline{x_2}+\overline{x_1}x_2+x_1\overline{x_2}+x_1x_2=\\
&=\overline{x_1}\left(\overline{x_2}+x_2\right)+x_1\left(\overline{x_2}+x_2\right)=\overline{x_1}+x_1=1.
\end{aligned}
\qquad (5)
$$

Considering (5), the operation of super-gluing variables in the analytical representation can take the following form, for example:

$$
\begin{aligned}
&\overline{x_1}\,\overline{x_2}\,\overline{x_3}x_4+\overline{x_1}\,\overline{x_2}x_3x_4+\overline{x_1}x_2\overline{x_3}x_4+\overline{x_1}x_2x_3x_4+\\
&+x_1\overline{x_2}\,\overline{x_3}x_4+x_1\overline{x_2}x_3x_4+x_1x_2\overline{x_3}x_4+x_1x_2x_3x_4=\\
&=x_4\left(\begin{array}{l}\overline{x_1}\,\overline{x_2}\,\overline{x_3}+\overline{x_1}\,\overline{x_2}x_3+\overline{x_1}x_2\overline{x_3}+\overline{x_1}x_2x_3+\\+x_1\overline{x_2}\,\overline{x_3}+x_1\overline{x_2}x_3+x_1x_2\overline{x_3}+x_1x_2x_3\end{array}\right)=\\
&=x_4\left(\begin{array}{l}\overline{x_1}\,\overline{x_2}\left(\overline{x_3}+x_3\right)+\overline{x_1}x_2\left(\overline{x_3}+x_3\right)+\\+x_1\overline{x_2}\left(\overline{x_3}+x_3\right)+x_1x_2\left(\overline{x_3}+x_3\right)\end{array}\right)=\\
&=x_4\left(\overline{x_1}\,\overline{x_2}+\overline{x_1}x_2+x_1\overline{x_2}+x_1x_2\right)=\\
&=x_4\left(\overline{x_1}\left(\overline{x_2}+x_2\right)+x_1\left(\overline{x_2}+x_2\right)\right)=x_4\left(\overline{x_1}+x_1\right)=x_4.
\end{aligned}
$$

An interval of the Boolean space. The concept of a ternary vector and an interval of a Boolean space is presented in [18]:

$$
\overline{x_1}\,\overline{x_2}x_3x_4+\overline{x_1}x_2x_3x_4=\overline{x_1}\,x_3x_4,
$$

$$
\begin{bmatrix}0011\\3\end{bmatrix}\quad\begin{bmatrix}0111\\7\end{bmatrix}\quad\begin{bmatrix}0-11\\3,7\end{bmatrix}.
$$

$$
x_1\overline{x_2}\,\overline{x_3}\;+\;x_1x_2\overline{x_3}\;=\;x_1\,\overline{x_3}.
$$

$$
\begin{bmatrix}100-\\8,9\end{bmatrix}\quad\begin{bmatrix}110-\\12,13\end{bmatrix}\quad\begin{bmatrix}1-0-\\8,9,12,13\end{bmatrix}.
$$

«Binary and decimal numbers in parentheses below the sum indicate the rows of the truth table for which the corresponding terms take the value of one. A binary expression in which a dash occurs represents two binary numbers formed by replacing the dash with a «0» and then a «1». Similarly for a binary expression, in which two dashes represent the corresponding four binary numbers, formed by replacing two dashes by entries «0» and «0», «0» and «1», etc.» [18].

The decimal equivalent of the analytic notation of the Boolean function is also given in [18]:

$$
T=\sum\begin{bmatrix}(0,1,2,3),(7,15,23,31),\\(29,31),(22,23),(14,15)\end{bmatrix}.
$$

$$
\begin{aligned}
&T=\overline{x_5}\,\overline{x_4}\,\overline{x_3}+x_3x_2x_1+x_5x_4x_3x_1+\\
&+x_5\overline{x_4}x_3x_2+\overline{x_5}x_4x_3x_2.
\end{aligned}
$$

An ordered set of Boolean variables (x_1, x_2, ..., x_n) form an n-component Boolean vector. The set consisting of 2^n of these vectors is called a Boolean space, which I denote by M. Each vector from the set M is an argument of the Boolean function $f(x_1, x_2, ..., x_n)$, which returns the value 0 or 1 for each set of variables (x_1, x_2, ..., x_n).

The easiest way to specify a Boolean function is to use a truth table, which lists all possible sets of argument values and their corresponding function values.

The representation of the function $f(x_1, x_2, ..., x_n)$ can be compressed if I limit myself to the enumeration of the elements of its characteristic set, that is, those elements of the space M on which the function $f(x_1, x_2, ..., x_n)$ returns the value 1. This set is denoted by M_f^1.

The compactness of the representation of the characteristic set M_f^1 can be improved if you use ternary vectors, the components of which can take as their values, in addition to the symbols «0» and «1», also the symbol «–».

I shall interpret the ternary vector as a set of all Boolean vectors obtained from it by all possible substitutions of the values «0» and «1» instead of the value «–». It is worth noting that carrying out all possible substitutions of the values «0» and «1» instead of the value «–» is the process of synthesis of a complete combinatorial system, 2-(n, b)-design, with its parameters. If the ternary vector has k values «–», then it generates 2^k Boolean vectors that form the interval $I(\alpha, \beta)$ in the Boolean space, the minimum element (α) of which is determined by substituting zeros instead of «–», the maximum (β) – substitution of units [19].

For example, the ternary vector 11–0– in this interpretation is considered as an interval of the space $I(\alpha, \beta)$ of five Boolean variables formed by the following terms:

$$I(\alpha,\beta)=I(11000,11101)=\begin{matrix}11000\\11001\\11100\\11101\end{matrix}.$$

The components along which the boundaries (and, therefore, all the vectors of the interval) coincide are called the external components of the interval, the rest are internal. The external components form an implicant of rank (r), and the number of internal components gives the bit rate of the interval (n) [20]. The set of internal components of the interval $I(\alpha, \beta)$ form a combinatorial system, 2-(n, b)-design, for this example, 2-(2, 4)-design.

Numerical conjunct term. The correspondence between the Boolean vectors $\alpha=x_1x_2...x_n$ and the numbers $N\in\{0,1,...,2^n-1\}$ is established by the following relation:

$$N=\sum_{i=1}^{n}x_i\times 2^{n-i}, \quad (6)$$

where n is the bit rate of the Boolean vector, the components of the vector $x_1x_2...x_n$ represent the binary numbers 0 and 1.

Let the Boolean vector $\alpha=1010$ be given. Substituting its components into formula (6), I get the number $N=1\times2^3+0\times2^2+1\times2^1+0\times2^0=8+2=10$.

The given number is $N=14$. Expanding it into the sum of powers of two: $14=8+4+2=1\times2^3+1\times2^2+1\times2^1+0\times2^0$, I get the Boolean vector $\alpha=1110$.

If I arrange the Boolean vectors $\alpha=x_1x_2...x_n$ in lexicographic order, then the integers N_i will correspond to the numbers of the sets of variables $(x_1,x_2,...,x_n)$ in the truth table of the Boolean function.

A conjunct term (minterm, constituent of a unit) is a term that unites all Boolean variables in a direct or inverse code with a conjunction sign. The minterm is denoted as follows:

$$F_i=\begin{cases}1, \text{ if the number of setting } (x_1,x_2,...,x_n)\\ \text{is equal to } N,\\ 0, \text{ if the number of setting } (x_1,x_2,...,x_n)\\ \text{is not equal to } N.\end{cases}$$

Example, $F_1=\overline{x}_1\overline{x}_2\overline{x}_3x_4$, $F_2=\overline{x}_1x_2\overline{x}_3$.

The rank of the term r is determined by the number of logical variables included in this term. For example, for minterm $F_1=\overline{x}_1\overline{x}_2\overline{x}_3x_4$, the rank is $r=4$, for minterm $F_2=\overline{x}_1x_2\overline{x}_3$, the rank is $r=3$.

In [21, 22] it is proposed to consider the numerical conjunct term θ^r of rank r (i.e., a set of glued numerical minterms) of the Boolean function $f(x_1, x_2, ..., x_n)$ as a set-theoretic element of a preformed set of conjunct terms of all ranks r (r=0, 1, 2, ..., n), called a conjunct term field, namely: of rank n (subfield of mingerms θ^n), ranks (n–1), (n–2), ..., 2, 1 (subfields of implicants θ^{n-1}, θ^{n-2}, ..., θ^2, θ^1) and rank 0 (constant function $1-\theta^0$).

In [23], a method for constructing the conjunct term field of the Boolean function $f(x_1, x_2, ..., x_n)$ is proposed, the essence of which is the formation of subfields $P_n^{n-1}, P_n^{n-2}, ..., P_n^1$ using the matrices L_n^r of the literal masks $l\in\{0, 1\}$ and the simple implementation of the recurrent procedure for their formation.

As a result, for a certain mask of literals of rank r, it is possible to obtain 22^{n-r} pseudo-ternary numbers. Then, if instead of the symbol «–» I substitute all its possible values 0 and 1, then for a certain conjunct term of rank r it is possible to obtain 2^r numerical (decimal) minterms.

The transition from the analytical notation of the conjunct term to its decimal equivalent, as an ordered set of numerical minterms, is possible by filling the symbols «–» in the pseudo-ternary code with the values 0 and l, starting from the smallest, that is, the symbols «–» are replaced by the system of 2-(n, b)-design illustrating an example:

$$\overline{x}_2x_3x_5\Rightarrow(-01-1)\Rightarrow\begin{vmatrix}0&0&1&0&1\\0&0&1&1&1\\1&0&1&0&1\\1&0&1&1&1\end{vmatrix}\Rightarrow(5,7,21,23).$$

The technique for constructing a conjunct term field can be successfully used to find a set of simple conjunct terms when solving the problem of minimizing not only complete but also partially defined Boolean functions, as well as their systems specified by DNF [23].

5. Results of a non-standard system for simplifying Boolean functions

5. 1. Determining logical operations for super-gluing of variables by 2-(n, b)-design combinatorial systems

The relationship between the PDNF of the logical function and the complete combinatorial system with repetition, 2-(n, b)-design (2), and vice versa, is provided by the visual matrix form (4). Therefore, the latter can be considered as an abstract class, a superclass. An abstract class can only be an ancestor class. Then instances of this class will be objects with attributes:

– perfect disjunctive normal form (PDNF) of the Boolean function;

– a complete combinatorial system with repeated 2-(n, b)-design. (7)

To reproduce (represent) the determination of the logical operation for super-gluing of variables by combinatorial systems with repetition, 2-(n, b)-design, a mechanism for providing polymorphism will also be needed. The literal meaning of this term translated from Greek means «the presence of many forms». In other words, polymorphism is the property of some object to take different forms depending on the situation. In object-oriented programming, this term is used in relation to a method.

Properties of polymorphism make it possible to define a class, which is a general description of some group of similar objects of actual reality. But very often, describing such a class, it is impractical or even impossible to specify the implementation of some methods common to the entire group. In this case, the methods are described as abstract, and their implementation is not indicated.

In UML – the Unified Modeling Language – a class is represented as a rectangle divided into three parts. The first part contains the name of the class, the second – attributes, the third – methods (Fig. 1).

Visual-matrix form
- perfect disjunctive normal form (PDNF); - complete combinatorial system with repeated 2-(*n, b*)-design
- logical operation of super-gluing of variables () - logical operation of super-gluing of variables ()

Fig. 1. Mapping a class to a UML diagram

Since the complete combinatorial system with repeated 2-(*n, b*)-design is a variant of the matrix representation of the Boolean function, the logical operation of super-gluing variables for the 2-(*n, b*)-design system will have the correct meaning.

The determination of the logical operation of super-gluing the variables by the 2-(*n, b*)-design combinatorial system is based on the presence of the same methods (logical operations of super-gluing the variables) in different objects from the list of instances of the abstract class (7).

Given that full PDNF of the function:

– uniquely defines a complete combinatorial system with repetition, 2-(*n, b*)-design (2), and vice versa [17];

– reduction of the full PDNF function always gives a unit (5),

this then gives reason to represent the logical operation of super-gluing the variables by a complete combinatorial system with repetition, 2-(*n, b*)-design, according to the following verbal concept:

$$\begin{aligned}&2\text{-}(n,b)\text{-design} \equiv \overline{x_1}\,\overline{x_2}...\overline{x_n} + \overline{x_1}\,\overline{x_2}...x_n + ... + x_1x_2x_3 = \\ &= \overline{x_1}\,\overline{x_2}...\left(\overline{x_n} + x_n\right) + \overline{x_1}x_2...\left(\overline{x_n} + x_n\right) + \\ &+ ... + x_1x_2...\left(\overline{x_n} + x_n\right) = \overline{x_1}...\overline{x_{n-1}} + ... + x_1x_{n-1} = \\ &= \overline{x_1}...\left(\overline{x_{n-1}} + x_{n-1}\right) + ... = 1. \end{aligned} \quad (8)$$

Thus, the 2-(*n, b*)-design system represents a logical operation of super-gluing the variables ($n > 1$, $b > 3$).

For $n = 1$, $b = 2$, the combinatorial system, 2-(*n, b*)-design, provides a logical operation of simple gluing of variables:

$$2\text{-}(1,2)\text{-design} \equiv \overline{x_1} + x_1 = 1.$$

Then the visual-matrix form (4) for the operation of super-gluing the variables may take the following form, for example:

– the first rule of super-gluing for a 4-bit Boolean function:

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

$= x_4.$ (9)

The literals repeated in the variable sets of the PDNF of function (9) are simple implicants for the reduced form of the Boolean function. Therefore, x_4 is a simple implicant in the shortened form of the Boolean function (9).

The general rule of gluing on the intervals of the Boolean space containing combinatorial 2-(*n, b*)-design systems can be formulated as follows: the sets of variables subject to gluing are the number that is a power of 2. The new elementary product obtained (prime implicant) is defined as the product of variables that do not change their value on all sets that are glued together. The number *m* of variables remaining in the elementary product is determined by the formula:

$$m = n - \log_2 M,$$

where *n* is the number of function variables; *M* is the number of sets to be glued. Therefore, for function (9), the number *m* is $m = n - \log_2 M = 4 - \log_2 8 = 1$.

The ternary vector of the visual matrix form (9) representing the interval of the Boolean space *I* (0001, 1111) takes the form:

$$(---1).$$

The decimal equivalent of the analytical record of the first rule (9) takes the following form:

$$x_4 \Rightarrow (---1) \Rightarrow$$

$$\Rightarrow \begin{vmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{vmatrix} \Rightarrow (1,3,5,7,9,11,13,15);$$

– the second rule of super-gluing for a 4-bit Boolean function:

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

$= x_3x_4.$ (10)

The number *m* for function (10) is $m = n - \log_2 M = 4 - \log_2 4 = 2$.

The ternary vector of the visual matrix form (10) representing the interval of the Boolean space *I* (0011, 1111) takes the form:

$$(--11).$$

The decimal equivalent of the analytical record of the second rule (10) takes the following form:

$$x_3x_4 \Rightarrow (--11) \Rightarrow$$

$$\Rightarrow \begin{vmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{vmatrix} \Rightarrow (3,7,11,15);$$

– the third rule of super-gluing for a 4-bit Boolean function:

$$\begin{array}{|cccc|}\hline x_1 & x_2 & x_3 & x_4 \\ \hline 0 & x_2 & x_3 & x_4 \\ 1 & x_2 & x_3 & x_4 \\ \hline\end{array} = x_2x_3x_4. \quad (11)$$

The number m for function (11) is $m = n - \log_2 M = 4 - \log_2 2 = 3$.

The ternary vector of the visual matrix form (11) representing the interval of the Boolean space I (0111, 1111) takes the form:

(– 111).

The decimal equivalent of the analytic notation of the third rule (11) is as follows:

$$x_2x_3x_4 \Rightarrow (-111) \Rightarrow \left|\begin{array}{cccc} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{array}\right| \Rightarrow (7,15).$$

The first rule of super-gluing the variables uses the 2-(3, 8)-design system. The second rule uses a 2-(2, 4)-design system. The third rule uses a 2-(1, 2)-design system. Rule (11) manifests itself as a simple gluing of variables and is a partial case of rules (9) and (10).

The variables forming a complete combinatorial system with repetition, 2-(n, b)-design, can occupy any bits of the logical function.

The decimal equivalents of analytical entries (9) to (11) directly represent the necessary intervals of the Boolean space for carrying out equivalent figurative transformations when simplifying Boolean functions.

The following examples demonstrate verbal and figurative ways of simplifying Boolean functions, which represent standard and non-standard implementations of their simplification algorithms.

Example 1. It is required to simplify the perfect disjunctive normal form (PDNF) of a Boolean function using the analytical method [24]:

$$f(x,y,z) = \bar{x}y\bar{z} + \bar{x}yz + x\bar{y}z + xy\bar{z} + xyz. \quad (12)$$

Solution.

When moving out of parentheses in the first pair of minterms of function (12) $\bar{x}y$, in the last pair xy, I obtained:

$$f(x,y,z) = \bar{x}y(\bar{z} + z) + x\bar{y}z + xy(\bar{z} + z).$$

It is obvious that $\bar{z} + z = 1$. Then function (12) takes the form:

$$f(x,y,z) = \bar{x}y + x\bar{y}z + xy. \quad (13)$$

When rearranging the term in (13) and taking out the variable y out of parentheses, the following is obtained:

$$f(x,y,z) = \bar{x}y + x\bar{y}z + xy = y(\bar{x} + x) + x\bar{y}z = y + x\bar{y}z. \quad (14)$$

The law of distributivity is applied to expression (14):

$$f(x,y,z) = y + x\bar{y}z = (y + x)(y + \bar{y})(y + z). \quad (15)$$

A simplification of expression (15) has been carried out:

$$f(x,y,z) = y + x\bar{y}z = (y + x)(y + z). \quad (16)$$

Once again, the law of distributivity is applied to expression (16):

$$f(x,y,z) = (y + x)(y + z) = y + xz. \quad (17)$$

Further simplification of the logical expression (17) is no longer possible. Thus, the minimum function is obtained:

$$f(x,y,z) = y + xz. \quad (18)$$

The simplification of the function $f(x, y, z)$ (12) using the visual-matrix form of the analytical method takes the following form:

$$f(x,y,z) = \begin{array}{|c|ccc|}\hline \text{No.} & x & y & z \\ \hline 2 & 0 & 1 & 0 \\ 3 & 0 & 1 & 1 \\ 5 & 1 & 0 & 1 \\ 6 & 1 & 1 & 0 \\ 7 & 1 & 1 & 1 \\ \hline\end{array} = \begin{array}{|ccc|}\hline x & y & z \\ \hline & 1 & \\ 1 & & 1 \\ \hline\end{array} = y + xz. \quad (19)$$

In the first matrix of expression (19), the following actions are performed:

– to modules 2, 3, 6, 7, which contain the complete combinatorial system, 2-(2, 4)-design, and the common literal «y», the super-gluing operation of variables is applied [17];

– to modules 5, 7, which contain the complete combinatorial system, 2-(1, 2)-design, and common literals «y», «z», a simple variable gluing operation is applied.

The seventh block of variables «111» is common to two systems – $\Sigma m(2,3,6,7)$, $\Sigma m(5,7)$ and two operations of super- and simple gluing of variables, the localization of which is determined by combinatorial systems, 2-(2, 4)-design, and 2-(1, 2)-design [4]. As a result, the minimal function (19) is obtained, which coincides with the minimal function (18). The result is the same, but the non-standard implementation of the simplification algorithm is simpler.

Example 2. It is required to simplify the switching circuit in Fig. 2 (x – on, $\bar{x}$ – off) [24].

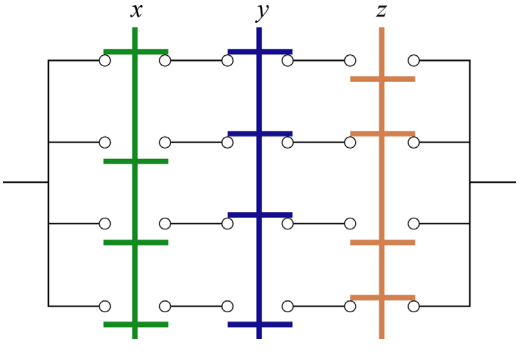

Fig. 2. Switching circuit

Solution.

The switching circuit in Fig. 1 corresponds to a logical function:

$$f(x,y,z) = xy\bar{z} + \bar{x}yz + \bar{x}y\bar{z} + \bar{x}\bar{y}z. \quad (20)$$

In the second and third minterms of the function (20), the common factor $\bar{x}y$, is taken out in parentheses, and I obtained:

$$f(x,y,z) = xy\bar{z} + \bar{x}y(z + \bar{z}) + \bar{x}\bar{y}z.$$

It is obvious that $z+\bar{z}=1$. Then function (20) takes the form:

$$f(x,y,z)=xy\bar{z}+\bar{x}y+\overline{xy}z. \quad (21)$$

Expression (21) is transformed as follows:

$$f(x,y,z)=xy\bar{z}+\bar{x}y+\overline{xy}z=xy\bar{z}+\bar{x}(y+\bar{y}z).$$

When applying the law of elementary absorption, the following is obtained:

$$f(x,y,z)=xy\bar{z}+\bar{x}(y+\bar{y}z)=xy\bar{z}+\bar{x}y+\bar{x}z. \quad (22)$$

The elementary absorption operation was also performed on the first two terms of expression (22):

$$f(x,y,z)=xy\bar{z}+\bar{x}y+\bar{x}z=y(x\bar{z}+\bar{x})+\bar{x}z=$$
$$=y(\bar{z}+\bar{x})+\bar{x}z=y\bar{z}+\bar{x}y+\bar{x}z.$$

Once in the law of generalized gluing of variables, $\bar{x}$ is taken instead of x, then the following will be obtained:

$$f(x,y,z)=\bar{x}z+y\bar{z}+\bar{x}y=\bar{x}z+y\bar{z}.$$

Answer: the minimal function is:

$$f=\bar{x}z+y\bar{z}. \quad (23)$$

Accordingly, a simplified switch diagram will look like this (Fig. 3).

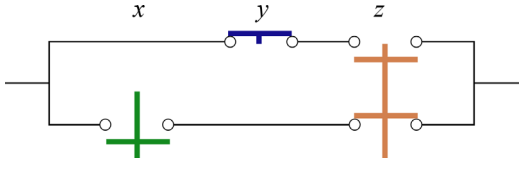

Fig. 3. Simplified switch diagram

The simplification of the function $f(x, y, z)$ (20) using the visual-matrix form of the analytical method takes the following form:

$$f(x,y,z)=\begin{array}{|c|ccc|}\hline \text{No.} & x & y & z\\ \hline 1 & 0 & 0 & 1\\ 2 & 0 & 1 & 0\\ 3 & 0 & 1 & 1\\ 6 & 1 & 1 & 0\\ \hline\end{array}=\begin{array}{|ccc|}\hline x & y & z\\ \hline 0 & & 1\\ & 1 & 0\\ \hline\end{array}=\bar{x}z+y\bar{z}. \quad (24)$$

In the first matrix of expression (24), the following actions are performed:

– the location of logical operations is determined by the place of 2-(1, 2)-design combinatorial systems in the truth table of the given function;

– to modules 1, 3, and 2, 6, each of which contains a complete combinatorial system, 2-(1, 2)-design, and common literals «z» and «y», «$\bar{x}$» respectively, a simple variable gluing operation is applied.

As a result, the minimal function (24) is obtained, which coincides with the minimal function (23).

Contemplating examples 1 and 2, it is important to note that the sequence of the standard (verbal) procedure of minimization of Boolean functions is necessarily deterministic in its own determination of the steps of representation of the analytical method. In turn, a non-standard system implies a simplification algorithm through the detection of 2-(n, b)-design systems in the truth table, and therefore defines the appropriate, necessary and sufficient logical operations for the equivalent transformation of Boolean functions.

5. 2. Identifying the locations of equivalent transformations with a non-standard system for simplifying Boolean functions

Analytical method has no established recommendations for the rational search for minimal dead-end forms of Boolean functions. The success of minimization depends on the level of knowledge of axioms, the laws of algebra of logic, and the skills of their application. For functions with the number of variables four or more, this method involves multi-pass combinations of logical transformations that must be performed during simplification. The process of simplification is time-consuming and does not provide any guarantees of obtaining the required minimum, since neither the axioms nor the laws of the algebra of logic give a direct indication of the achievement of the minimal dead-end form. The existence of a minimal form can be determined only in simple cases, when the number of variants of the search procedure does not exceed two to three, and the depth of the completed search is limited to approximately the same number of steps. These estimates largely depend on the engineering experience of the performer.

A non-standard minimization system has innovative reserves for reducing the complexity of simplifying logical functions. Let's consider it in detail.

Assertion 1. The 2-(n, b)-design, 2-(n, x/b)-design systems have two implementations:

– an object, the position of which can be determined, established;

– method, logical operation that can be performed, carried out.

Assertion 2. Each combinatorial system, 2-(n, b)-design, 2-(n, x/b)-design, implies its logical operation:

– 2-(n, b)-design – super- and/or simple gluing of variables;

– 2-(n, x/b)-design – incomplete super-gluing of variables [17], in particular in the case when different intervals of the Boolean space containing the systems, 2-(n, b)-design, 2-(n, x/b)-design, partially coincide, have an intersection.

Assertion 3. Each combinatorial system, 2-(n, b)-design, 2-(n, x/b)-design, implies an algorithm for simplifying logical functions according to the principle of their probable, free possibility.

This «reduction a priori» to the experience of things is in itself an event of synergistic «self» organization – the formation of a binary matrix structure in hyperparametric tendencies. Self-organization is not a cause since it is not random, but regular; it is not a consequence since there is no consequence if there is no cause.

The formal basis for this reduction of the visual-matrix form is the algebraic simplification of the corresponding complete and/or incomplete PDNF of Boolean functions, such as (5).

Assertion 4. The beginning (principle) of a non-standard system for simplifying Boolean functions is the search for intervals in the truth table containing combinatorial systems, 2-(n, b)-design, 2-(n, x/b)-design, and not multi-pass logical transformations, which must be performed when simplifying the function by an analytical method.

The self-organization of the intervals of the Boolean space, containing the 2-(n, b)-design, 2-(n, x/b)-design systems, in the truth table gives the possibility of free selection of

simplification variants, in the «optimal variant form». Having turned into its own opposite, as required by the dialectic of the formation of the «new modern» disclosure of «old» (labyrinth) problems, from a possible-temporal probability forecast into a possible – necessary one, it provides an unambiguous identification of the locations of equivalent transformations. These herparic manifestations of synergistic self-organization of the binary matrix structure, accumulating, give a qualitative jump! Mutually unambiguous correspondence between the PDNF of the Boolean function and the combinatorial system, 2-(n, b)-design, gives a hyperparameter in the form of a fixed location, a place of equivalent transformations. In turn, this implies the algorithm of the non-standard system of simplifying Boolean functions. The problem of the algorithm for simplifying Boolean functions by analytical metol solves itself!

Only the fundamental non-standard nature in principle solves the problem of the laboriousness of the procedure for simplifying analytical functions when they are represented in a visual-matrix form.

Unambiguous identification of the locations of equivalent transformations with a non-standard simplification system is also possible when different intervals of the Boolean space containing the 2-(n, b)-design systems have common modules, intersections. The use of intersection preempts a possible subsequent semi-gluing operation of variables.

The intersection of modules of 2-(n, b)-design systems has its own model in the form of an algebraic analog, which is demonstrated by an example of simplifying a Boolean function in the main basis.

Let the PDNF function of three variables be given [25]:

$$f(x_3,x_2,x_1)=\overline{x}_3\overline{x}_2\overline{x}_1+\overline{x}_3x_2\overline{x}_1+\overline{x}_3x_2x_1+x_3x_2\overline{x}_1.$$

Here, to simplify the function $f(x_3,x_2,x_1)$, the minterm $\overline{x}_3x_2\overline{x}_1$ must be written three times:

$$\begin{aligned}f(x_3,x_2,x_1)&=\left(\overline{x}_3\overline{x}_2\overline{x}_1+\overline{x}_3x_2\overline{x}_1\right)+\\&+\left(\overline{x}_3x_2x_1+\overline{x}_3x_2\overline{x}_1\right)+\left(x_3x_2\overline{x}_1+\overline{x}_3x_2\overline{x}_1\right)=\\&=\overline{x}_3\overline{x}_1\left(\overline{x}_2+x_2\right)+\overline{x}_3x_2\left(x_1+\overline{x}_1\right)+x_2\overline{x}_1\left(x_3+\overline{x}_3\right)=\\&=\overline{x}_3\overline{x}_1+\overline{x}_3x_2+x_2\overline{x}_1.\end{aligned}$$

Contemplating this elementary example, it is clear that the analytical method is quite complex due to the laboriousness of searching for neighboring minterms [25].

The technique for simplifying the logical function $f(x_3, x_2, x_1)$, using the visual-matrix form of the analytical method, by focusing, makes it possible to simultaneously represent the system of relations between the individual variables of the problem. As a result, verbal procedures of algebraic transformations are replaced by equivalent figurative transformations. This significantly reduces the complexity of searching for neighboring minterms:

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

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

$=$

x_3	x_2	x_1
0		0
0	1	
	1	0

$=\overline{x}_3\overline{x}_1+\overline{x}_3x_2+x_2\overline{x}_1.$

The detection of the locations of equivalent transformations using 2-(n, b)-design combinatorial systems is demonstrated by the following example.

Example 3. Use a non-standard system to simplify the function $f(x_1, x_2, x_3, x_4)$ in the Reed-Muller basis given in the canonical form [26]:

$$f(x_1,x_2,x_3,x_4)=\sum(0,3,5,6,7,8,9,10,12,15). \quad (25)$$

Solution.

The beginning of the simplification of function (25) is carried out in the basic logical basis:

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

No.	x_1	x_2	x_3	x_4	f
0	0	0	0	0	1
3	0	0	1	1	1
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	1
12	1	1	0	0	1
15	1	1	1	1	1

$=$

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

. (26)

In the first matrix of expression (26), the following actions are performed:

– to modules 0, 8; 3.7; 5.7; 6.7; 8,9; 8,10; 8,12; 7,15, each of which form a complete combinatorial system of 2-(1, 2)-design [17] and contain corresponding common literals, the operation of simple gluing of variables is applied.

The seventh block of variables «0111» is common to four systems – $\Sigma m(3,7)$, $\Sigma m(5,7)$, $\Sigma m(6,7)$, $\Sigma m(7,15)$ and four operations of simple gluing of variables, the location of which is determined by the corresponding intervals of the Boolean space containing combinatorial 2-(1, 2)-design systems.

The eighth block of variables «1000» is common to four systems – $\Sigma m(0,8)$, $\Sigma m(8,9)$, $\Sigma m(8,10)$, $\Sigma m(8,12)$ and four operations of simple gluing of variables, the location of which is determined by the corresponding intervals of the Boolean space containing combinatorial 2-(1, 2)-design systems.

The minimal function in disjunctive normal form (DNF) takes the form:

$$\begin{aligned}f_{\text{MDNF}}(x_1,x_2,x_3x_4)&=\overline{x}_2\overline{x}_3\overline{x}_4+\overline{x}_1x_3x_4+\overline{x}_1x_2x_4+\\&+\overline{x}_1x_2x_3+x_1\overline{x}_2\overline{x}_3+x_1\overline{x}_2\overline{x}_4+x_1\overline{x}_3\overline{x}_4+x_2x_3x_4.\end{aligned} \quad (27)$$

To obtain the minimum function in the polynomial normal form (PNF), three extensions are possible using:

– singular function [2];

– Zhegalkin polynomial [2];

– Reed-Muller polynomial [2].

Consider the first two extensions. The procedure for obtaining a singular function takes the form [2]:

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

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

$=$

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

. (28)

Simplification of the singular function (28) in the Reed-Muller basis:

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

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

$=$

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

. (29)

In the first matrix of expression (29), the following transformations are carried out:

$$\overline{x_1}\,\overline{x_2}\,\overline{x_3}\,\overline{x_4}\oplus\overline{x_1}\,\overline{x_2}x_3x_4=\overline{x_1}\,\overline{x_2}\left(\overline{x_3}\,\overline{x_4}\oplus x_3x_4\right)=$$
$$=\overline{x_1}\,\overline{x_2}\left(\overline{x_3\oplus x_4}\right)=\overline{x_1}\,\overline{x_2}\left(x_3\oplus\overline{x_4}\right)=\overline{x_1}\,\overline{x_2}x_3\oplus\overline{x_1}\,\overline{x_2}\,\overline{x_4};$$

$$\overline{x_1}x_2\overline{x_3}x_4\oplus\overline{x_1}x_2x_3\overline{x_4}=\overline{x_1}x_2\left(\overline{x_3}x_4\oplus x_3\overline{x_4}\right)=$$
$$=\overline{x_1}x_2\left(x_3\oplus x_4\right)=\overline{x_1}x_2x_3\oplus\overline{x_1}x_2\boldsymbol{x_4};$$

$$x_1\overline{x_2}\,\overline{x_3}x_4\oplus x_1\overline{x_2}x_3\overline{x_4}=x_1\overline{x_2}\left(\overline{x_3}x_4\oplus x_3\overline{x_4}\right)=$$
$$=x_1\overline{x_2}\left(x_3\oplus x_4\right)=x_1\overline{x_2}x_3\oplus x_1\overline{x_2}\boldsymbol{x_4}.$$

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

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

$=$

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

$=$

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

. (30)

In the second matrix of expression (30), the following transformations are carried out:

$$\overline{x_1}x_2x_4\oplus x_1\overline{x_2}x_4=x_4\left(\overline{x_1}x_2\oplus x_1\overline{x_2}\right)=$$
$$=x_4\left(x_1\oplus x_2\right)=x_1x_4\oplus x_2x_4.$$

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

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

$=$

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

$=$

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

. (31)

In the first matrix of expression (31), a logical operation of polynomial absorption of variables [2] is carried out:

$$x_2x_4\oplus x_2x_3x_4=x_2\overline{x_3}x_4.$$

In the second matrix of expression (31), two logical operations of polynomial semi-gluing of variables [2] are carried out:

$$\overline{x_2}x_3\oplus\overline{x_1}x_2x_3=x_3\left(\overline{x_2}\oplus\overline{x_1}x_2\right)=x_3\left(\overline{x_1}+\overline{x_2}\right)=$$
$$=x_3\overline{x_1x_2}=x_3\left(1\oplus x_1x_2\right)=x_3\oplus x_1\boldsymbol{x_2}x_3.$$

$$x_1x_4\oplus x_1\overline{x_3}\,\overline{x_4}=x_1\left(x_4\oplus\overline{x_3}\,\overline{x_4}\right)=x_1\left(\overline{x_3}+x_4\right)=$$
$$=x_1\overline{x_3\overline{x_4}}=x_1\left(1\oplus x_3\overline{x_4}\right)=x_1\oplus x_1\boldsymbol{x_3}\overline{x_4}.$$

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

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

$=$

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

$=$

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

$=$

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

$=$

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

.

The dead-end function in PNF is as follows:

$$f_{\text{The dead end PNF}}(x_1,x_2,x_3x_4)=$$
$$=x_1\oplus x_2\oplus x_3\oplus\overline{x_4}\oplus x_1\overline{x_2}\,\overline{x_3}\,\overline{x_4}\oplus\overline{x_1}x_2x_3\boldsymbol{x_4}, \quad (32)$$

it contains 12 literals, which is two literals less compared to [26]:

$$\left(\begin{array}{l}f=x_1\oplus x_2\oplus\overline{x_2}x_3x_4\oplus\\ \oplus\overline{x_1}\,\overline{x_3}\,\overline{x_4}\oplus x_1x_2\overline{x_3}\oplus x_1x_2x_4\end{array}\right).$$

For TPNF (32), it is possible to continue the simplification using the following transformations [27]:

$$x_1\oplus x_2\oplus x_3\oplus\overline{x_4}\oplus x_1\overline{x_2}\,\overline{x_3}\,\overline{x_4}\oplus\overline{x_1}x_2x_3x_4=$$
$$=x_1\oplus x_2\oplus x_3\oplus\overline{x_4}\oplus\left(x_1\oplus x_2\right)\left(x_1\oplus x_3\right)\left(x_1\oplus x_4\right). \quad (33)$$

As a result, a 3-level logical expression was also obtained, but one containing 10 literals. Further simplification of expression (33) gives the following result:

$$x_1 \oplus x_2 \oplus x_3 \oplus \overline{x_4} \oplus (x_1 \oplus x_2)(x_1 \oplus x_3)(x_1 \oplus x_4) = \\ = x_2 \oplus x_3 \oplus \left((x_1 \oplus x_2)(x_1 \oplus x_3) + \overline{(x_1 \oplus x_4)}\right). \quad (34)$$

The minimal function (34) represents a 4-level logical expression containing 8 literals.

For function (26), the Zhegalkin polynomial takes the form:

$$P(x_1, x_2, x_3, x_4) = 1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_1x_2 \oplus x_1x_3 \oplus \\ \oplus x_1x_4 \oplus x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4.$$

Simplifying $P(x_1, x_2, x_3, x_4)$ gives the minimal Boolean function in the mixed basis:

$$f_{\text{MPNF}}(x_1, x_2, x_3, x_4) = \\ = 1 \oplus x_1x_2 \oplus x_1x_3 \oplus x_1x_4 \oplus x_1x_2x_3 \oplus x_1x_2x_4 \oplus \\ \oplus x_1x_3x_4 \oplus x_2 \oplus x_2x_3x_4 \oplus x_3 \oplus x_4 =$$

$$= 1 \oplus \begin{vmatrix} 1 & 1 & & \\ 1 & & 1 & \\ 1 & & & 1 \\ 1 & 1 & 1 & \\ 1 & 1 & & 1 \\ 1 & & 1 & 1 \\ & 1 & & \\ & 1 & 1 & 1 \\ & & 1 & \\ & & & 1 \end{vmatrix} = 1 \oplus \begin{vmatrix} 1 & 1 & & \\ 1 & 0 & 1 & \\ 1 & 0 & & 1 \\ 1 & & 1 & 1 \\ & 1 & & \\ & 1 & 1 & 1 \\ & & 1 & \\ & & & 1 \end{vmatrix} =$$

$$= 1 \oplus x_1x_2 \oplus x_1\overline{x_2}(x_3 \oplus x_4) \oplus (x_1 \oplus x_2)x_3x_4 \oplus \\ \oplus x_2 \oplus x_3 \oplus x_4 = 1 \oplus \overline{x_1}x_2 \oplus \\ \oplus \overline{x_1}\,\overline{x_2}(x_3 \oplus x_4) \oplus (x_1 \oplus x_2)x_3x_4 = \\ = 1 \oplus \overline{x_1}x_2 \oplus x_1\overline{x_2} \oplus x_1\overline{x_2} \oplus \overline{x_1}\,\overline{x_2}(x_3 \oplus x_4) \oplus \\ \oplus (x_1 \oplus x_2)x_3x_4 = 1 \oplus (x_1 \oplus x_2) \oplus x_1\overline{x_2} \oplus \\ \oplus \overline{x_1}\,\overline{x_2}(x_3 \oplus x_4) \oplus (x_1 \oplus x_2)x_3x_4 = \\ = 1 \oplus x_1\overline{x_2} \oplus \overline{x_1}\,\overline{x_2}(x_3 \oplus x_4) \oplus (x_1 \oplus x_2)\overline{x_3x_4} = \\ = 1 \oplus \left(x_1\overline{x_2} + (x_3 \oplus x_4)\right) \oplus (x_1 \oplus x_2)\overline{x_3x_4} = \\ = \left(x_1\overline{x_2} + (x_3 \oplus x_4)\right) \oplus \left(\overline{(x_1 \oplus x_2)} + x_3x_4\right). \quad (35)$$

The minimum function (35), compared to (34), also consists of 8 literals, but has two logical XOR operations less, which can give technological advantages in the manufacture of a digital circuit.

5. 3. Theorem of the non-standard system of simplification of Boolean functions

Simplification of Boolean functions is carried out by equivalent transformations, in particular, with the help of the following logical operations:

– simple gluing of variables:

$$xy + \overline{x}y = y;$$

$$(x + y)(\overline{x} + y) = y. \quad (36)$$

– super-gluing of variables:

$$xyz + \overline{x}yz + x\overline{y}z + \overline{x}\,\overline{y}z = (x + \overline{x})yz + (x + \overline{x})\overline{y}z = \\ = yz + \overline{y}z = (y + \overline{y})z = z;$$

$$(x + y + z)(\overline{x} + y + z)(x + \overline{y} + z)(\overline{x} + \overline{y} + z) = \\ = (y + z)(\overline{y} + z) = z. \quad (37)$$

Theorem. (without proof) Once all operations of simple and/or super-gluing variables (36), (37) are carried out in the perfect disjunctive normal form of the Boolean function, the minimum DNF of the given function will be obtained as a result.

Note. Operations of gluing variables (36), (37) may be redundant. This means the presence in the binary structure of the truth table of other minimal DNFs of the given function. Therefore, to obtain a minimal function, all non-redundant operations of simple and/or super-gluing of variables must be performed. This will ensure that the minimum function is obtained in the main basis without using the implicant table.

A reduced DNF is determined by means of a disjunctive combination of conjunct terms under the condition that all sets of variables of the truth table of a Boolean function containing 2-(n, b)-design systems completely cover that part of the set of sets on which the Boolean function returns one. At the same time, the intersection of the sets of variables holding the 2-(n, b)-design systems is allowed.

Example 4. Using a non-standard system, obtain the minimum Boolean function of four variables $f(x_1, x_2, x_3, x_4)$ given by the perfect disjunctive normal form [28]:

$$f(x_1, x_2, x_3x_4) = \\ = \overline{x_1}\,\overline{x_2}\,\overline{x_3}x_4 + \overline{x_1}\,\overline{x_2}x_3\overline{x_4} + \overline{x_1}x_2\overline{x_3}\,\overline{x_4} + \\ + \overline{x_1}x_2\overline{x_3}x_4 + \overline{x_1}x_2x_3\overline{x_4} + \\ + x_1\overline{x_2}\,\overline{x_3}\,\overline{x_4} + x_1\overline{x_2}\,\overline{x_3}x_4 + x_1x_2\overline{x_3}\,\overline{x_4} + \\ + x_1x_2\overline{x_3}x_4 + x_1x_2x_3\overline{x_4} + x_1x_2x_3x_4. \quad (38)$$

Solution:

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

No.	x_1	x_2	x_3	x_4
1	0	0	0	1
2	0	0	1	0
4	0	1	0	0
5	0	1	0	1
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
14	1	1	1	0
15	1	1	1	1

$=$

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

$$= x_1x_2 + x_1\overline{x_3} + x_2\overline{x_4} + \overline{x_1}x_3\overline{x_4} + \overline{x_3}x_4. \quad (39)$$

In the first matrix (39), which represents the PDNF of $f(x_1, x_2, x_3, x_4)$, the following actions are performed:

– to modules 1, 5, 9, 13; 4, 6, 12, 14; 8, 9, 12, 13; 12, 13, 14, 15, each of which forms a corresponding interval of the Boolean space containing the complete 2-(2, 4)-design combinatorial system, the super-gluing operation of variables is applied [17].

– to modules 2, 6, which form the corresponding interval of the Boolean space containing the combinatorial system 2-(1, 2)-design and contain the common literal «x_2», a simple variable gluing operation is applied.

The results of non-redundant logical operations of gluing variables are written to the second matrix of expression (39).

The decimal equivalent of the analytical notation of the minimal function (39) takes the form:

$$f(x_1,x_2,x_3,x_4)= \\ =\begin{Bmatrix}(1,5,9,13),(4,6,12,14), \\ (8,9,12,13),(12,13,14,15),(2,6)\end{Bmatrix}. \quad (40)$$

The structure of the decimal counterpart (40) accommodates the necessary intervals of the Boolean space containing the 2-(n, b)-design, 2-(n, x/b)-design systems. This implies all operations of simple and/or super-gluing of variables (36), (37), which according to the theorem gives a minimal function.

The sixth block of variables «0110» in the first matrix of expression (39) is common to two systems – $\Sigma m(4,6,12,14)$, $\Sigma m(2,6)$ and one operation of simple gluing of variables, the localization of which is determined by the corresponding intervals of the Boolean space, containing combinatorial 2-(2, 4)-design and 2-(1, 2)-design systems.

The twelfth block of variables «1100» in the first matrix of expression (39) is common to three systems – $\Sigma m(4,6,12,14)$, $\Sigma m(8,9,12,13)$, $\Sigma m(12,13,14,15)$ and one operation of super-gluing variables, the localization of which is determined by the corresponding intervals of the Boolean space containing combinatorial systems of 2–(2, 4)-design.

The thirteenth block of variables «1101» in the first matrix of expression (39) is common to three systems – $\Sigma m(1,5,9,13)$, $\Sigma m(8,9,12,13)$, $\Sigma m(12,13,14,15)$ and one operation of super-gluing variables, the localization of which is determined by the corresponding intervals of the Boolean space containing combinatorial systems of 2-(2, 4)-design.

The fourteenth block of variables «1110» in the first matrix of expression (39) is common to two systems – $\Sigma m(4,6,12,14)$, $\Sigma m(12,13,14,15)$ and one super-gluing operation of variables, the localization of which is determined by the corresponding intervals of the Boolean space containing combinatorial 2-(2, 4)-design systems.

As a result, in one step, without using the implicant table, the minimal function (39) is obtained, which is two literals smaller compared to the reduced function [28] $\left(f^2 = \overline{x_1}x_3\overline{x_4} + \right.$ $\left. + \overline{x_3}x_4 + x_2\overline{x_3} + x_2\overline{x_4} + x_1x_2 + x_1\overline{x_3}\right)$.

Example 5. Using a non-standard system, obtain the minimum Boolean function of four variables $f(x_1, x_2, x_3, x_4)$ given in PDNF [29, 30]:

$$f(x_1,x_2,x_3,x_4)=\sum(0,1,2,5,7,10,14,15). \quad (41)$$

Solution.

I simplify the function $f(x_1, x_2, x_3, x_4)$ (41) 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
3	0	0	1	1
4	0	1	0	0
6	0	1	1	0
8	1	0	0	0
9	1	0	0	1
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1

$=$

No.	x_1	x_2	x_3	x_4
3	1	1	0	0
4	1	0	1	1
6	1	0	0	1
8	0	1	1	1
9	0	1	1	0
11	0	1	0	0
12	0	0	1	1
13	0	0	1	0

$=$

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

$=$

$$=(\overline{x_1}+x_3)(x_1+\overline{x_2}+x_4)(x_2+\overline{x_3}+\overline{x_4}). \quad (42)$$

The decimal equivalent of MCNF (42) is:

$$f_{\text{MCNF}}(x_1, x_2, x_3, x_4)= \\ =\{(3,11),(4,6),(8,9,12,13)\}=\{3,4,6,8,9,11,12,13\}.$$

In the first matrix of expression (42) according to the Nelson method, I invert the values of the variables [1]. In the second matrix of expression (42), which represents the DCNF of $f(x_1, x_2, x_3, x_4)$ (41), the following actions are performed:

– to modules 8, 9, 12, 13, which form the interval of the Boolean space containing the complete combinatorial system 2-(2, 4)-design, the super-gluing operation of variables is applied [17];

– to modules 3, 11; 4, 6, which form the corresponding intervals of the Boolean space, containing combinatorial systems 2-(1, 2)-design, the operation of simple gluing of variables is applied.

The results of logical operations of gluing the variables are written to the third matrix of expression (42). As a result, the minimal function (42) in the conjunctive normal form, which is four literals smaller than the reduced function in the disjunctive normal form [29, 30], was obtained in one step:

$$\left(Y_{\text{MDNF}}=\begin{cases}1.\{(0.1),(2.10),(5,7),(14,15)\} \\ 2.\{(0,2),(1,5),(7,15),(10,14)\}\end{cases}\right).$$

Example 6. Using a non-standard system, obtain the minimum Boolean function of four variables $f(x_1, x_2, x_3, x_4)$, specified in PNF [29]:

$$f(x_1,x_2,x_3,x_4)=\sum(0,1,2,5,7,10,14,15).$$

Solution:

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

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

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

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

$$=\left(\overline{x_1}+x_3\right)\oplus\overline{x_1}x_2\overline{x_4}\oplus\overline{x_2}x_3x_4=$$

$$=\overline{x_1}\overline{x_3}\oplus\overline{x_1}x_2\overline{x_4}\oplus\overline{x_2}x_3x_4=$$

$$=1\oplus x_1\overline{x_3}\oplus\overline{x_1}x_2\overline{x_4}\oplus\overline{x_2}x_3x_4=$$

$$=x_1\overline{x_3}\oplus\overline{\overline{x_1}x_2\overline{x_4}}\oplus\overline{x_2}x_3x_4=$$

$$=x_1\overline{x_3}\oplus\left(x_1+\overline{x_2}+x_4\right)\oplus\overline{x_2}x_3x_4. \qquad (43)$$

A minimal function (43) in a mixed basis containing two fewer inversions is obtained compared to [29]:

$$\left(Y^{\mathrm{DNF}}=\left\{\begin{array}{l}(0000),(0001),\\(0010),(0101),\\(0111),(1010),\\(1110),(1111)\end{array}\right\}^{\mathrm{DNF}}\underset{\Rightarrow}{\overset{\mathrm{PNF}}{}}\begin{vmatrix}-&-&-&-\\1&-&0&-\\-&0&1&1\\0&1&-&0\end{vmatrix}\right).$$

Simplification of symmetric Boolean functions is provided by the polynomial basis algebra [27]. However, after the first equivalent transformation of a symmetric function in a polynomial basis, it becomes possible to simplify it in the main basis as well.

Example 7. Using a non-standard system, obtain the minimum Boolean function of four variables $f(x_1, x_2, x_3, x_4)$, specified in PDNF [29]:

$$f_{\mathrm{PDNF}}(x_1,x_2,x_3,x_4)=\{(0000),(0011),(1111)\}. \qquad (44)$$

Solution.

Function (44) is symmetric [27]. For the sake of simplicity (44), I apply the procedure of inserting identical conjunct terms, followed by the operation of super-gluing the variables:

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

$$=\begin{vmatrix}0&0&0&0\\0&0&1&1\\1&1&1&1\end{vmatrix}=\begin{vmatrix}0&0&0&0\\0&0&0&1\\0&0&0&1\\0&0&1&0\\0&0&1&0\\0&0&1&1\\0&1&0&0\\0&1&0&0\\0&1&0&1\\0&1&0&1\\0&1&1&0\\0&1&1&0\\0&1&1&1\\0&1&1&1\\1&0&0&0\\1&0&0&0\\1&0&0&1\\1&0&0&1\\1&0&1&0\\1&0&1&0\\1&0&1&1\\1&0&1&1\\1&1&0&0\\1&1&0&0\\1&1&0&1\\1&1&0&1\\1&1&1&0\\1&1&1&0\\1&1&1&1\end{vmatrix}=1\oplus\begin{vmatrix}0&0&0&1\\0&0&1&0\\0&1&0&0\\0&1&0&1\\0&1&1&0\\0&1&1&1\\1&0&0&0\\1&0&0&1\\1&0&1&0\\1&0&1&1\\1&1&0&0\\1&1&0&1\\1&1&1&0\end{vmatrix}=$$

$$=1\oplus\begin{vmatrix}0&0&0&1\\0&0&1&0\\0&1&&\\1&0&&\\1&1&0&0\\1&1&0&1\\1&1&1&0\end{vmatrix}=1\oplus\begin{vmatrix}0&0&0&1\\0&0&1&0\\0&1&&\\1&0&&\\1&1&&0\\1&1&0&1\end{vmatrix}=$$

$$=1\oplus\begin{vmatrix}0&&0&1\\0&&1&0\\0&1&&\\1&0&&\\1&&&0\\1&&0&1\end{vmatrix}=1\oplus\begin{vmatrix}&&0&1\\0&&1&0\\0&1&&\\1&0&&\\1&&&0\end{vmatrix}=1\oplus\begin{vmatrix}&&0&1\\&&1&0\\0&1&&\\1&0&&\\1&&&0\end{vmatrix}=$$

$$=\overline{x_1\overline{x_4}+(x_1\oplus x_2)+(x_3\oplus x_4)}. \qquad (45)$$

The result of the first equivalent transformation in the polynomial basis (the procedure of inserting the same conjunct terms followed by the operation of super-gluing the variables) is written to the third matrix of expression (45). The function represented by the third matrix of expression (45) remains singular [2]. Therefore, for further simpli-

fication in the third matrix, I switch to the main basis and obtain the minimal function (45) in the mixed basis, which has four literals less, compared to [29]:

$$Y^{\text{DNF}}=\begin{Bmatrix}(0000),\\(0011),\\(1111)\end{Bmatrix}^{\text{DNF}}\Rightarrow^{\text{PNF}}\left(\begin{array}{l}1.\begin{Bmatrix}(000-),\\(00-1),\\(1111)\end{Bmatrix}^{\text{PNF}}\\2.\begin{Bmatrix}(00-0),\\(001-),\\(1111)\end{Bmatrix}^{\text{PNF}}\\3.\begin{Bmatrix}(0000),\\(0-11),\\(-111)\end{Bmatrix}^{\text{PNF}}\\4.\begin{Bmatrix}(0000),\\(-011),\\(1-11)\end{Bmatrix}^{\text{PNF}}\end{array}\right).$$

Therefore, a minimal Boolean function of four variables $f(x_1, x_2, x_3, x_4)$ was obtained using a non-standard system, which has four fewer literals compared to [29].

5. 4. Thesaurus of non-standard system of simplification of Boolean functions

Binary systems with repetition, 2-(n, b)-design, 2-$(n, x/b)$-design, which are part of the Boolean space in the form of combinatorial structures of truth tables of Boolean functions, represent logical operations of simple and/or super-gluing variables. In this regard, establishing the placement of the 2-(n, b)-design, 2-$(n, x/b)$-design systems provides unambiguous identification of the locations of equivalent transformations, and hence the implication of the algorithm for simplifying Boolean functions. Accordingly, the non-standard system for simplifying Boolean functions has its own systematized composition of information (knowledge) and settings, in the form of a thesaurus (Tables 1, 5), which provides orientation in this system.

Table 1

Thesauri of the basic concepts of the Quine-McCluskey method and the non-standard system of simplification of Boolean functions

No. of entry	Thesaurus for simplifying Boolean functions using the Quine-McCluskey method	Thesaurus of a non-standard system for simplifying Boolean functions
1	DDNF	DDNF
2	Laws of logic algebra	Laws of logic algebra
3	Constituents and implants of logical functions	Intervals of the Boolean space and their possible intersection containing the 2-(n, b)-design, 2-$(n, x/b)$-design systems
4	Operations of incomplete gluing and absorption of variables	Location of equivalent transformations
5	Abbreviated form of the function	The minimal form of the function

The division into groups of conjunct terms, according to the Quine-McCluskey method, provides a logical operation of only simple gluing of variables. This increases the amount of computation and makes it difficult to simplify the function.

Example 8. Using a non-standard system, obtain the minimum Boolean function of four variables $f(x_4, x_3, x_2, x_1)$, given in the canonical form [18]:

$$f(x_4,x_3,x_2,x_1)=\sum(0,1,2,5,6,7,9,10,11,13,14,15). \quad (46)$$

Solution.

Option 1.

I choose the minimum number of intervals containing combinatorial 2-(n, b)-design systems to cover the PDNF of function (46) (Fig. 4).

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

Fig. 4. Disjunctive normal form of function $f(x_4, x_3, x_2, x_1)$

Table 2 gives the minimal sets of intervals containing 2-(n, b)-design systems for covering the PDNF of function (46) and the corresponding simple implicants.

Table 2

Minimal sets of intervals containing 2-(n, b)-design systems for covering the PDNF of function (46) and the corresponding simple implicants

No. of entry	Minimal sets of intervals containing 2-(n, b)-design systems	Simple implicants of a minimal function
Variant 1		
1	$\Sigma m(0,1)$	$\neg x_4\,\neg x_3\,\neg x_2$
2	$\Sigma m(2,6,10,14)$	$x_2\,\neg x_1$
3	$\Sigma m(5,7,13,15)$	$x_3\,x_1$
4	$\Sigma m(9,11,13,15)$	$x_4\,x_1$
Variant 2		
1	$\Sigma m(0,2)$	$\neg x_4\,\neg x_3\,\neg x_1$
2	$\Sigma m(1,5,9,13)$	$\neg x_2\,x_1$
3	$\Sigma m(6,7,14,15)$	$x_3\,x_2$
4	$\Sigma m(10,11,14,15)$	$x_4\,x_2$

Contemplating Table 2 demonstrates that there are two identical minimum functions:

$$f_{\text{MDNF1}}=\overline{x_4}\,\overline{x_3}\,\overline{x_2}+x_2\overline{x_1}+x_3x_1+x_4x_1;$$

$$f_{\text{MDNF2}}=\overline{x_4}\,\overline{x_3}\,\overline{x_1}+\overline{x_2}x_1+x_3x_2+x_4x_2. \quad (47)$$

The result of simplification (47) coincides with [18], in which the simplification of function (46) was carried out in two steps by the Quine-McCluskey method.

Option 2.

I choose not the minimal set of intervals that hold the 2-(n, b)-design systems but one that covers the constituents of the PDNF unit (Fig. 4) of function (46), for example, as in Table 3.

Table 3

A non-minimal set of intervals containing 2-(n, b)-design systems for covering the PDNF of function (46) and the corresponding simple implicants

No. of entry	Combinatorial 2-(n, b)-design systems	Simple implicants of systems
A	$\Sigma m(0,1)$	$\neg x_4\,\neg x_3\,\neg x_2$
B	$\Sigma m(1,5,9,13)$	$\neg x_2\,x_1$
C	$\Sigma m(2,6,10,14)$	$x_2\,\neg x_1$
D	$\Sigma m(5,7,13,15)$	$x_3\,x_1$
E	$\Sigma m(9,11,13,15)$	$x_4\,x_1$
F	$\Sigma m(10,11,14,15)$	$x_4\,x_2$

Table 4

Implicant table of DNF of functions $f(x_4,x_3,x_2,x_1)$ (46)

No.	x_4	x_3	x_2	x_1	f	000 –	– –01	– –10	–1–1	1– –1	1–1–	$f_{\min}$
0	0	0	0	0	1	•	–	–	–	–	–	1
1	0	0	0	1	1	•	•	–	–	–	–	1
2	0	0	1	0	1	–	–	•	–	–	–	1
5	0	1	0	1	1	–	•	–	•	–	–	1
6	0	1	1	0	1	–	–	•	–	–	–	1
7	0	1	1	1	1	–	–	–	•	–	–	1
9	1	0	0	1	1	–	•	–	–	•	–	1
10	1	0	1	0	1	–	–	•	–		•	1
11	1	0	1	1	1	–	–	–	–	•	•	1
13	1	1	0	1	1	–	•	–	•	•	–	1
14	1	1	1	0	1	–	–	•	–		•	1
15	1	1	1	1	1	–	–	–	•	•	•	1

Reviewing Table 4 reveals that simple implicants 000 –, – –10, –1–1, 1– –1 cover (highlighted in green) all constituents of the PDNF unit of function (46). Thus, the MDNF of function (46) takes the form:

$$f_{\text{MDNF}} = \overline{x_4}\,\overline{x_3}\,\overline{x_2} + x_2\overline{x_1} + x_3x_1 + x_4x_1. \qquad (48)$$

The results of the simplification of (47), (48) of example 8 coincide with [18], but the algorithm of their simplification is significantly simpler.

Example 9. Using a non-standard system, obtain the minimum Boolean function of four variables $f(x_1, x_2, x_3, x_4)$, given by the algebraic form [31]:

$$\begin{aligned} &f(x_1,x_2,x_3,x_4) = \\ &= \overline{x_1}\,\overline{x_2}x_3x_4 + \overline{x_1}x_2\overline{x_3}x_4 + \overline{x_1}x_2x_3x_4 + x_1\overline{x_2}\,\overline{x_3}\,\overline{x_4} + \\ &+ x_1\overline{x_2}x_3\overline{x_4} + x_1\overline{x_2}x_3x_4 + x_1x_2\overline{x_3}\,\overline{x_4} + x_1x_2x_3\overline{x_4}. \end{aligned} \qquad (49)$$

Solution.

Simplification of function $f(x_1, x_2, x_3, x_4)$ (49) in DNF:

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

No.	x_1	x_2	x_3	x_4
3	0	0	1	1
5	0	1	0	1
7	0	1	1	1
8	1	0	0	0
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
14	1	1	1	0

$=$

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

$=$

$$= x_1\overline{x_4} + \overline{x_1}x_2x_4 + \overline{x_2}x_3x_4 =$$
$$= \{(3,11),(5,7),(8,10,12,14)\} =$$
$$= \{3,5,7,8,10,11,12,14\}. \qquad (50)$$

In one step, the minimal function in DNF (50) is obtained, which is six literals smaller than the reduced function in [31], in which the Quine-McCluskey method is used for simplification.

Simplification of the function $f(x_1, x_2, x_3, x_4)$ in PNF: since function (49) is singular [2], let's move on to the Reed-Muller basis:

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

No.	x_1	x_2	x_3	x_4
3	0	0	1	1
5	0	1	0	1
7	0	1	1	1
8	1	0	0	0
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
14	1	1	1	0

$=$

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

$=$

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

$=$

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

$=$

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

$=$

$$= x_1 \oplus x_4\left(1 \oplus x_1\overline{x_2} \oplus \overline{x_2}\,\overline{x_3}\right) =$$
$$= x_1 \oplus x_4\left(1 \oplus \overline{x_2}\left(x_1 \oplus \overline{x_3}\right)\right) =$$
$$= x_1 \oplus x_4\left(\overline{\overline{x_2}\left(x_1 \oplus \overline{x_3}\right)}\right) =$$
$$= x_1 \oplus x_4\left(x_2 + \left(\overline{x_1 \oplus \overline{x_3}}\right)\right) =$$
$$= x_1 \oplus x_4\left(x_2 + (x_1 \oplus x_3)\right). \qquad (51)$$

A minimal function in a mixed basis (51) is obtained, which is nine literals less than the reduced function in work [31].

Table 5

Thesauri of the main concepts of the Carnot map and the non-standard system of simplifying Boolean functions

No. of entry	Thesaurus of simplification of Boolean functions by the Carnot map	Thesaurus of a non-standard system for simplifying Boolean functions
1	DDNF, DCNF	DDNF, DCNF
2	Laws of logic algebra	Laws of logic algebra
3	Correct contours and their possible intersection	Intervals of the Boolean space and their possible intersection containing the 2-(n, b)-design, 2-$(n, x/b)$-design systems
4	Rules for grouping minterms (maxterms) into correct contours	Location of equivalent transformations
5	Definition rules for minimal function	The minimal form of the function

Disadvantages of the method for Carnot maps, Veitch diagrams:

– the actual additional construction of Carnot maps, Veitch diagrams with their structure according to the given truth table of the Boolean function is necessary;

– it is necessary to unambiguously match the indices of the bit of the binary code, using the Gray code, when placing the terms of the given function in the cells of the Carnot maps, Veitch diagrams;

– it is advisable to use Carnot maps and Veitch diagrams when the number of variables does not exceed 5;

– the selection of the correct contours is carried out intuitively and there is no algorithm that would provide the best solution;

– after choosing the correct contours on the Carnot map, the Veitch diagram, the control rules for determining the minimum function must be followed.

Example 10. Use a non-standard system to simplify the DNF of a partially defined function $f(x_1, x_2, x_3, x_4)$ given by the truth table (Fig. 5) [32].

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

Fig. 5. Truth table of function $f(x_1, x_2, x_3, x_4)$

Solution:

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

No.	x_1	x_2	x_3	x_4	f
1	0	0	0	1	1
3	0	0	1	1	1
9	1	0	0	1	1
11	1	0	1	1	1
12	1	1	0	0	1
14	1	1	1	0	1
15	1	1	1	1	1
2	0	0	1	0	–
4	0	1	0	0	–
13	1	1	0	1	–

$=$

x_1	x_2	x_3	x_4	f
	0		1	1
1	1			1
0	0	1	1	–
1	0	1	1	–

$=$

$$= x_1x_2 + \overline{x_2}x_4. \qquad (52)$$

The first matrix of expression (52) represents the truth table of the partially defined DNF function $f(x_1, x_2, x_3, x_4)$ (Fig. 5). Modules 1, 3, 9, 11 and 12, 14, 15, 13 of the first matrix of expression (52) were treated with the super-gluing operation of variables [17]. The result of logical operations of the first matrix is written to the second matrix of expression (52).

The minimum DNF takes the form:

$$f_{\text{MDNF}}(x_1,x_2,x_3,x_4) = x_1x_2 + \overline{x_2}x_4.$$

It is important to note that the undefined sets of variables 2, 4 were not used during the simplification of function $f(x_1, x_2, x_3, x_4)$ (Fig. 5) by the non-standard system. This ultimately reduced the overall complexity of simplifying the function (Fig. 5).

The result of simplifying the function $f(x_1, x_2, x_3, x_4)$ (Fig. 5) using the Carnot map in work [32] is $f_{\text{MDNF}}(x_1,x_2,x_3,x_4) = x_1x_2 + x_3x_4$. However, testing with code No. 1 – 0001, the specified result of simplification does not give unity, as required by the truth table of the function in Fig. 5.

Example 11. Simplify the Boolean function $f(x_1, x_2, x_3, x_4)$, represented by the algebraic form [32] with a non-standard system:

$$f(x_1,x_2,x_3x_4) = x_1x_2\overline{x_3}\,\overline{x_4} + x_1x_2x_3\overline{x_4} + x_1\overline{x_2}x_3\overline{x_4} + x_1\overline{x_2}\,\overline{x_3}\,\overline{x_4} + x_1\overline{x_2}x_3x_4 + x_1\overline{x_2}\,\overline{x_3}x_4 + \overline{x_1}x_2x_3x_4 + \overline{x_1}\,\overline{x_2}\,\overline{x_3}x_4. \qquad (53)$$

Solution.

Simplification of function $f(x_1, x_2, x_3, x_4)$ (53) is to be carried out in 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
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
13	1	1	0	1
15	1	1	1	1

$=$

No.	x_1	x_2	x_3	x_4
0	1	1	1	1
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
13	0	0	1	0
15	0	0	0	0

$=$

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

$=$

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

$$= \{(0,2,4,6),(2,3),(4,5),(13,15)\} = \{0,2,3,4,5,6,13,15\}. \qquad (54)$$

In the first matrix of expression (54) according to the Nelson method, I invert the values of variables [1].

In the second matrix of expression (54), which represents the DCNF of $f(x_1, x_2, x_3, x_4)$, the following actions are performed:

– to the modules 0, 2, 4, 6, which form the interval of the Boolean space containing the complete combinatorial system, 2-(2, 4)-design, the super-gluing operation of variables is applied [17];

– to modules 2, 3; 4, 5; 13, 15, which form the corresponding intervals of the Boolean space, containing combinatorial systems, 2-(1, 2)-design, the operation of simple gluing of variables is applied.

The results of logical operations of gluing variables are written to the third matrix of expression (54). As a result, in one step, the minimum function (54) in CNF was obtained, which has one less inversion, compared to the simplification of the function $f(x_1, x_2, x_3, x_4)$ (52) in CNF using the Veitch diagram [32] $(f_{\text{MCNF}}(x_1,x_2,x_3,x_4)=(x_1+x_4)(x_1+x_2+\overline{x_3})\times$ $\times(\overline{x_2}+x_3+\overline{x_4})(\overline{x_1}+\overline{x_2}+\overline{x_4}))$.

5. 5. Comparison with the evolutionary method, genetic, swarm algorithms, and the directed selection method

Example 12. Simplify the Boolean function $f(x_1, x_2, x_3, x_4)$ by a non-standard system given in the canonical form [33]:

$$f(x_1,x_2,x_3,x_4)=\sum(7,10,11,13,14,15). \quad (55)$$

Solution.

I simplify the function $f(x_1, x_2, x_3, x_4)$ (55) 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
7	0	1	1	1
10	1	0	1	0
11	1	0	1	1
13	1	1	0	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	1		1

$=$

$$=x_1x_3+x_1x_2x_4+x_2x_3x_4=$$
$$=x_1x_3+x_2x_4(x_1+x_3). \quad (56)$$

The decimal equivalent of MDNF (56) is:

$$f_{\text{MDNF}}(x_1, x_2, x_3, x_4)=$$
$$=\{(10,11,14,15),(7,15),(13,15)\}=$$
$$=\{7,10,11,13,14,15\}.$$

In the first matrix of expression (56), the following actions are performed:

– to modules 10, 11, 14, 15, which form an interval of the Boolean space containing the complete combinatorial system, 2-(2, 4)-design, the operation of super-gluing of variables is applied [17];

– to modules 7, 15; 13, 15, which form an interval of the Boolean space containing the complete combinatorial system, 2-(1, 2)-design, the operation of simple gluing of variables is applied.

The fifteenth block of variables «1111» is common to three systems – $\Sigma m(10,11,14,15)$, $\Sigma m(7,15)$, $\Sigma m(13,15)$ and three operations of super- and simple gluing of variables, the location of which is determined by the combinatorial systems, 2-(2, 4)-design and 2-(1, 2)-design [4]. As a result, a minimal function is obtained in one step:

$$f_{\text{MDNF}}(x_1,x_2,x_3x_4)=x_1x_3+x_2x_4(x_1+x_3). \quad (57)$$

The results of simplification of function (55) by a non-standard system and an evolutionary method are given in Table. 6. Convergence with the optimal result was achieved at the 175th generation of the evolutionary method [33].

Table 6

The result of minimizing the function $f(x_1, x_2, x_3, x_4)$ (55)

Non-standard system	Evolutionary method
$f_{\text{MDNF}}(x_1,x_2,x_3x_4)=x_1x_3+x_2x_4(x_1+x_3)$	$F_{\min}=(bd+c)(bdc+a)$

Contemplating Table 6 reveals that the result of simplifying the function $f(x_1, x_2, x_3, x_4)$ (55) by a non-standard system is a minimal function containing six literals. This is one literal less compared to [33]. Decreasing the number of literals gives a smaller depth of the logical scheme that implements the minimal function (57) (Fig. 6).

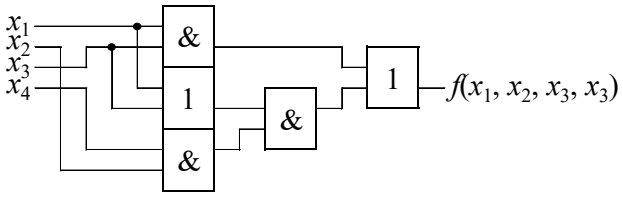

Fig. 6. A logic circuit that implements a minimum function (57) (circuit complexity of 5 logical elements, circuit depth of 3 typical logic elements)

Fig. 7 shows the optimal variant of the logic scheme that implements the DNF of the minimal function $f(x_1, x_2, x_3, x_4)=$ $=x_1x_3+x_1x_2x_4+x_2x_3x_4$ (56).

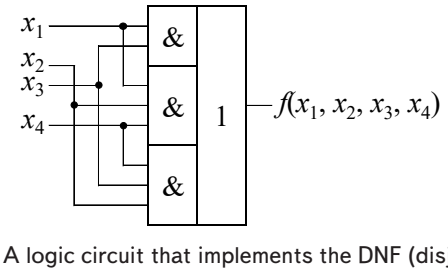

Fig. 7. A logic circuit that implements the DNF (disjunctive normal form) of the minimal function (56) (complexity of the circuit is 4 logic elements, depth of the circuit is 2 typical logic elements)

Fig. 8 shows a logical scheme designed on the basis of the evolutionary method [33].

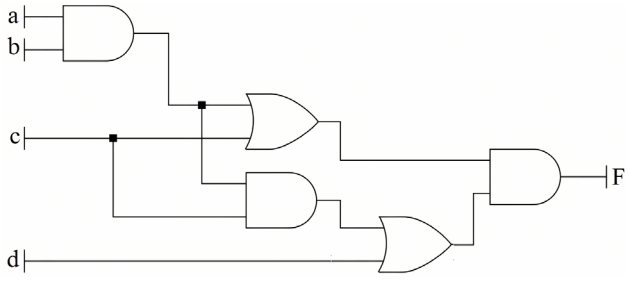

Fig. 8. A logic scheme designed on the basis of the evolutionary method (the complexity of the scheme is 5 logical elements; the depth of the scheme is 4 typical logical elements)

It should be noted that according to the diagram in Fig. 8, the corresponding logical function will be of the form:

$$F_{\min} = (bd + c)(bdc + a). \tag{58}$$

Continuing the simplification of the function (56), one can achieve the result (57):

$$F_{\min} = (bd + c)(bdc + a) = bdc + abd + bdc + ac =$$
$$= abd + bdc + ac = ac + bd(a + c).$$

Thus, the transformation of the given logical function (55) over 175 generations of the evolutionary method turned out to be insufficient to obtain the optimal result of simplifying the function.

Example 13. Simplify the Boolean function $F(A, B, C, D)$ with a non-standard system given by the truth table (Fig. 9) [34]:

No.	A	B	C	D	F
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	1

Fig. 9. Truth table of function $F(A, B, C, D)$

Solution. I shall simplify the function $F(A, B, C, D)$ (Fig. 9) in PNF. Consider four dead-end forms of the given function $F(A, B, C, D)$ (Fig. 9). Depending on the technical conditions for designing a logic circuit that implements simplified functions, one of the considered dead-end forms may become minimal.

The first dead-end form (DF).

Simplification of the function in PNF:

$$f_{The\ first\ dead-end\ form}(A,B,C,D) =$$

No.	A	B	C	D
0	0	0	0	0
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
13	1	1	0	1
15	1	1	1	1

$=$

A	B	C	D
	0	0	0
0	1		
1	0	0	
1	0		0
1	1		1

$=$

A	B	C	D
	0	0	0
	1		
1	0	0	
1	0		0
1	1		0

$=$

A	B	C	D
	0	0	0
	1		
1	0	0	
1			0

$=$

$$= A\overline{D} \oplus B \oplus A\overline{B}\overline{C} \oplus \overline{B}\overline{C}\overline{D} =$$
$$= A\overline{D} \oplus B \oplus \overline{B}\overline{C}(A \oplus \overline{D}) =$$
$$= A\overline{D} \oplus \overline{A}D \oplus \overline{A}D \oplus B \oplus \overline{B}\overline{C}(A \oplus D) =$$
$$= A \oplus D \oplus \overline{A}D \oplus B \oplus \overline{B}\overline{C}\overline{(A \oplus D)} =$$
$$= (A \oplus D + \overline{B}\overline{C}) \oplus \overline{A}D \oplus B =$$
$$= (A \oplus D + \overline{B + C}) \oplus \overline{A}D \oplus B. \tag{59}$$

The results of simplifying the function $F(A, B, C, D)$ (Fig. 9) by a non-standard system and a genetic algorithm [34] are given in Table 7. Convergence with the optimal result was achieved in 400 generations of the genetic algorithm, the total number of population members is 170 [34]. It is worth noting that the problem of optimal stopping of the genetic algorithm during the simplification of Boolean functions has not been solved.

Contemplating Table 7, it can be seen that the dead-end function (59) has one XOR less, which can give technological advantages when it is implemented by a logical scheme (Fig. 10).

Table 7

Result of simplifying the function $F(A, B, C, D)$ (Fig. 9)

Non-standard system	Genetic algorithm
$F(A,B,C,D) = (A \oplus D + \overline{B + C}) \oplus \overline{A}D \oplus B$	$F = \overline{(A \oplus D) + C} + ((B \oplus D) \oplus (A + D))$
7 gates	7 gates
1 AND, 2 OR, 2 XORs, 2 NOT	1 AND, 2 OR, 3 XORs, 1 NOT
Number of connections – 13	Number of connections – 13

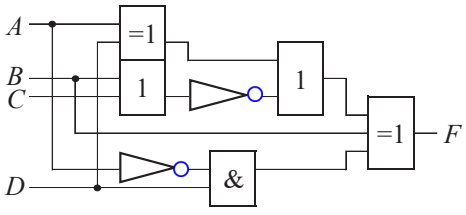

Fig. 10. A logic scheme designed by a non-standard system to implement the dead-end form of the function *F*(*A*, *B*, *C*, *D*) (59)

Fig. 11 shows a logical scheme designed on the basis of a genetic algorithm to implement a minimal function [34].

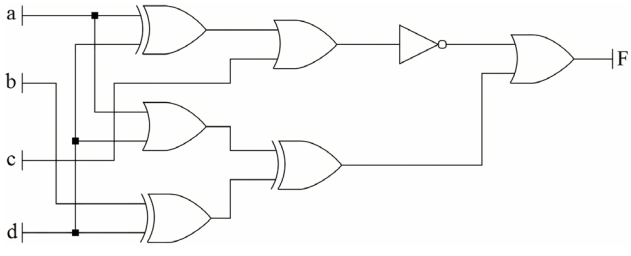

Fig. 11. A logical circuit designed on the basis of a genetic algorithm

The second dead end form.

Simplifying the function in Fig. 9 in PNF:

$$f_{\text{The second dead end form}}(A,B,C,D)=$$

No.	A	B	C	D
0	0	0	0	0
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
13	1	1	0	1
15	1	1	1	1

$=$

A	B	C	D
0	0	0	0
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	0	1	1
1	1	0	1
1	1	1	1

$=$

A	B	C	D
0	0	0	0
0	1		
1	0		
1			1
1	0	1	1
1	1	0	1
1	1	1	1

$=$

A	B	C	D
0	0	0	0
0	1		
1	0		
1			1
1	0	0	1

$=$

A	B	C	D
0	0	0	0
0	1		
1	0		
1			1
	0	0	0

$=$

A	B	C	D
	0	0	
0	0	0	
0	1		
1	0		
1			1
	0	0	0

$=$

A	B	C	D
	0	0	
0	0	1	
0			
1	0		
1			1
	0	0	0

$=$

A	B	C	D
	0	0	
0	0	0	
0			
	0		
1			1
	0	0	0

$$=\overline{B}\overline{C}\left(\overline{A\oplus D}\right)\oplus A\oplus B\oplus AD=$$

$$=\overline{\overline{B}\overline{C}+\left(\overline{A\oplus D}\right)}\oplus A\oplus B\oplus AD=$$

$$=\overline{B+C+(A\oplus D)}\oplus B\oplus A\overline{D}. \qquad (60)$$

The results of simplifying the function *F*(*A*, *B*, *C*, *D*) (Fig. 9) by a non-standard system and a genetic algorithm [34] are given in Table 8.

Table 8 demonstrates that the dead-end form of function (59) has one XOR and one less logical element. A smaller number of logical elements requires a smaller number of connections, which can also provide technological advantages when implementing function (59) in a logic scheme (Fig. 12).

Table 8

The result of simplifying the function *F*(*A*, *B*, *C*, *D*) (Fig. 9)

Non-standard system	Genetic algorithm
$F(A,B,C,D)=$ $\overline{B+C+(A\oplus D)}\oplus B\oplus A\overline{D}$	$F=\overline{(A\oplus D)+C}+$ $+((B\oplus D)\oplus(A+D))$
6 gates	7 gates
1 AND, 1 OR, 2 XORs, 2 NOT	1 AND, 2 OR, 3 XORs, 1 NOT
Number of connections – 12	Number of connections – 13

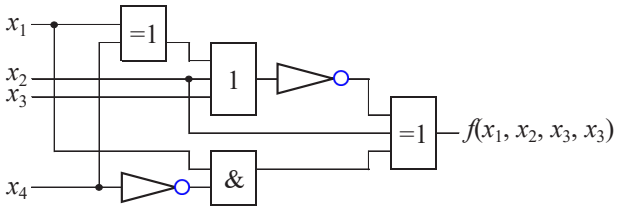

Fig. 12. A logic scheme designed by a non-standard system to implement the dead-end form of the function *F*(*A*, *B*, *C*, *D*) (59)

The third dead-end form.

The simplification of function (Fig. 9) for the third dead-end form begins with the fifth matrix of expression (60):

$$f_{\text{The third dead-end form}}(A,B,C,D)=$$

$=$

A	B	C	D
0	0	0	0
0	1		
1	0		
1			1
1	0	0	1

$=$

A	B	C	D
0	0	0	1
0	0	0	1
0	0	0	0
0	1		
1	0		
1			1
1	0	0	1
1	0	0	0
1	0	0	0

$=$

A	B	C	D
	0	0	
0	0	0	1
0	1		
1	0		
1			1
1	0	0	0

$=$

A	B	C	D
	0	0	
0	0	0	
0	1		
1	0		
1			1
	0	0	0

$=$

A	B	C	D
	0	0	
0	0	1	
0			
1	0		
1			1
	0	0	0

$=$

A	B	C	D
	0	0	
0	0	0	
0			
	0		
1			1
	0	0	0

$=$

A	B	C	D
1	0	0	
0			
	0		
1			1
	0	0	0

$$=\overline{B}\oplus A\overline{B}\overline{C}\oplus\overline{B}\overline{C}\overline{D}\oplus\left(\overline{A}+D\right)=$$

$$=\overline{B}\left(1\oplus A\overline{C}\oplus\overline{C}\overline{D}\right)\oplus\left(\overline{A}+D\right)=$$

$$=\overline{B}\left(1\oplus\overline{C}\left(A\oplus\overline{D}\right)\right)\oplus\left(\overline{A}+D\right)=$$

$$=\overline{B}\left(\overline{\overline{C}\left(A\oplus\overline{D}\right)}\right)\oplus\left(\overline{A}+D\right)=$$

$$=\overline{B}\left(C+(A\oplus D)\right)\oplus\left(\overline{A}+D\right). \qquad (61)$$

The results of simplifying the function $F(A, B, C, D)$ (Fig. 9) by a non-standard system and a genetic algorithm [35] are given in Table 9.

Table 9

Result of simplifying the function $F(A, B, C, D)$ (Fig. 9)

Non-standard system	Genetic algorithm
$F(A,B,C,D)=$ $\overline{B}(C+(A\oplus D))\oplus(\overline{A}+D)$	$F=\overline{(A\oplus D)+C}+$ $+((B\oplus D)\oplus(A+D))$
7 gates	7 gates
1 AND, 2 OR, 2 XORs, 2 NOT	1 AND, 2 OR, 3 XORs, 1 NOT
Number of connections – 12	Number of connections – 13

Looking at Table 9, it can be seen that the dead-end form of function (61) has one literal and one XOR less. A smaller number of literals requires a smaller number of connections in the scheme, which can give technological advantages in the implementation of function (61) (Fig. 13). It is important to note that the diagram in Fig. 13 has a longer signal delay compared to the scheme in Fig. 11. However, the choice of logic scheme depends on the design tasks and technical conditions.

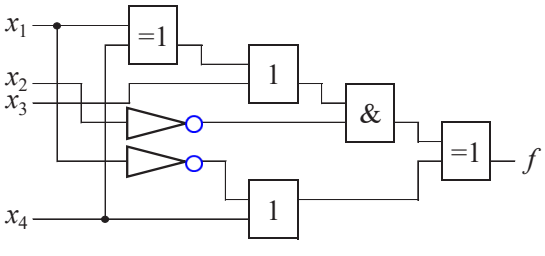

Fig. 13. A logic scheme designed by a non-standard system to implement the dead-end form of function $F(A, B, C, D)$ (61)

The fourth dead-end form.
Simplifying the function in Fig. 9 in DNF.

$f_{\text{The fourth dead–end form}}(A,B,C,D)=$

No.	A	B	C	D
0	0	0	0	0
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
13	1	1	0	1
15	1	1	1	1

$=$

A	B	C	D
	0	0	0
0	1		
1		0	1
1	0		0
	1		1

$=$

A	B	C	D
	0	0	0
0	1		
1	0	0	
1	0		0
	1		1

$=$

A	B	C	D
	0	0	0
0	1		0
1	0	0	
1	0		0
	1		1

$=$

A	B	C	D
	0	0	0
0	1	0	0
0	1	1	0
1	0	0	
1	0		0
	1		1

$=$

A	B	C	D
	0	0	0
0	1	0	
0	1	1	0
1	0	0	
1	0		0
	1		1

$=$

A	B	C	D
	0	0	0
0	1	0	
0	1		0
1	0	0	
1	0		0
	1		1

$=$

$$=\overline{x}_1x_2\overline{x}_3+\overline{x}_1x_2\overline{x}_4+x_1\overline{x}_2\overline{x}_3+x_1\overline{x}_2\overline{x}_4+\overline{x}_2\overline{x}_3\overline{x}_4+x_2x_4=$$
$$=\overline{x}_1x_2(\overline{x}_3+\overline{x}_4)+x_1\overline{x}_2(\overline{x}_3+\overline{x}_4)+\overline{x}_2\overline{x}_3\overline{x}_4+x_2x_4=$$
$$=(\overline{x}_3+\overline{x}_4)(\overline{x}_1x_2+x_1\overline{x}_2)+\overline{x}_2\overline{x}_3\overline{x}_4+x_2x_4=$$
$$=(\overline{x}_3+\overline{x}_4)(x_1\oplus x_2)+\overline{x}_2\overline{x}_3\overline{x}_4+x_2x_4=\overline{x_3x_4}(x_1\oplus x_2)+\overline{x_2+x_3+x_4}+x_2x_4. \quad (62)$$

The implementation of the obtained dead-end form of function (62) requires fewer transistors, compared to the scheme in Fig. 11, and, therefore, is technologically simpler (Fig. 14).

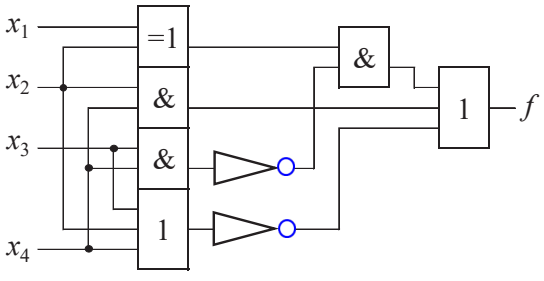

Fig. 14. A logic circuit designed by a non-standard system to implement the dead-end form of function $F(A, B, C, D)$ (62)

Work [34] considered the result of simplifying the function in Fig. 9, such that $F(A,B,C,D)=\overline{A}B+A(\overline{B}\overline{D}+\overline{C}D)$. However, testing with code No. 0 – 0000, the specified simplification option does not give unity, as required by the truth table of the function in Fig. 9.

Example 14. Simplify the Boolean function $F(E, D, C, B, A)$ by a non-standard system given by the truth table (Fig. 15) [35]:

No.	E	D	C	B	A	F
0	0	0	0	0	0	0
1	0	0	0	0	1	0
2	0	0	0	1	0	0
3	0	0	0	1	1	1
4	0	0	1	0	0	0
5	0	0	1	0	1	1
6	0	0	1	1	0	1
7	0	0	1	1	1	0
8	0	1	0	0	0	0
9	0	1	0	0	1	1
10	0	1	0	1	0	1
11	0	1	0	1	1	0
12	0	1	1	0	0	1
13	0	1	1	0	1	0
14	0	1	1	1	0	0
15	0	1	1	1	1	0
16	1	0	0	0	0	0
17	1	0	0	0	1	1
18	1	0	0	1	0	1
19	1	0	0	1	1	0
20	1	0	1	0	0	1
21	1	0	1	0	1	0
22	1	0	1	1	0	0
23	1	0	1	1	1	0
24	1	1	0	0	0	1
25	1	1	0	0	1	0
26	1	1	0	1	0	0
27	1	1	0	1	1	0
28	1	1	1	0	0	0
29	1	1	1	0	1	0
30	1	1	1	1	0	0
31	1	1	1	1	1	0

Fig. 15. Truth table of function $F(E, D, C, B, A)$

Solution. The function (Fig. 15) is singular [2]. To simplify it, we choose the Reed-Muller basis:

$F_{The\ dead-endPNF}(E,D,C,B,A)=$

=

No.	E	D	C	B	A	f
3	0	0	0	1	1	1
5	0	0	1	0	1	1
6	0	0	1	1	0	1
9	0	1	0	0	1	1
10	0	1	0	1	0	1
12	0	1	1	0	0	1
17	1	0	0	0	1	1
18	1	0	0	1	0	1
20	1	0	1	0	0	1
24	1	1	0	0	0	1

=

E	D	C	B	A
0	0	0	1	1
0	0	1		1
0	0	1	1	
0	1	0		1
0	1	0	1	
	1	1	0	0
1	0	0		1
1	0	0	1	
1		1	0	0
1	1	0	0	0

=

E	D	C	B	A
0	0	0	1	1
0		1		1
0		1	1	
0	1			1
0	1		1	
	1	1	0	0
1	0	0		1
1	0	0	1	
1		1	0	0
1	1	0	0	0

=

E	D	C	B	A
0	0	0	1	0
0		1		1
0		1	1	
0	1			1
0	1		1	
	1		0	0
1	0	0		1
	0	0	1	
1		1	0	0
0	1	0	0	0

=

E	D	C	B	A
0		0	1	1
0		1		1
0			1	
0	1			
0	1		1	
	1		0	0
1	0	0		1
	0	0	1	
1		1	0	0
0	1	1		0

=

E	D	C	B	A
0		0	1	1
0		1		1
0			1	
0	1		0	
	1		0	0
1	0	0		1
	0	0	1	
1		1	0	0
0	1	1		0

=

E	D	C	B	A
0		0	0	1
0				1
0			1	
0	1		0	
	1		0	0
1	0	0		1
	0	0	1	
1		1	0	0
0	1	1		0

=

E	D	C	B	A
0		0	0	1
0				1
0				
0	0		0	
	1		0	0
1	0	0		1
	0	0	1	
1		1	0	0
0	1	1		0

=

E	D	C	B	A
0		0	0	1
0				0
0	0		0	
	1		0	0
1	0	0		1
	0	0	1	
1		1	0	0
0	1	1		0

.

The dead-end PNF of the function (Fig. 15) is as follows:

$$F_{The\ dead-endPNF}(E,D,C,B,A)=\overline{E}\,\overline{B}\left(\overline{C}A\oplus\overline{D}\right)\oplus$$
$$\oplus\overline{E}\,\overline{A}\,\overline{D}\overline{C}\oplus\overline{B}\,\overline{A}(EC\oplus D)\oplus\overline{D}\overline{C}(EA\oplus B). \qquad (63)$$

The Zhegalkin polynomial for TPNF (63) takes the following form:

$$P(E,D,C,B,A)=ED\oplus EC\oplus EB\oplus EA\oplus$$
$$\oplus EDC\oplus EDB\oplus EDA\oplus ECB\oplus ECA\oplus EBA\oplus$$
$$\oplus DC\oplus DB\oplus DA\oplus DCB\oplus DCA\oplus$$
$$\oplus DBA\oplus CB\oplus CA\oplus CBA\oplus BA. \qquad (64)$$

With the help of the visual-matrix form, the simplification of the polynomial (64) is carried out and the minimum function in the mixed basis is obtained.

$F_{\min}(E,D,C,B,A)=$

=

E	D	C	B	A
1	1			
1		1		
1			1	
1				1
1	1	1		
1	1		1	
1	1			1
1		1	1	
1		1		1
1			1	1
	1	1		
	1		1	
	1			1
	1	1	1	
	1	1		1
	1		1	1
		1	1	
		1		1
		1	1	1
			1	1

=

E	D	C	B	A
1	1			
1	0	1		
1	0		1	
1	0			1
1		1	1	
1		1		1
1			1	1
	1	1		
	1		1	
	1			1
	0	1	1	
	0	1		1
	0		1	1
		1	1	1

=

$$=ED\oplus(C\oplus B\oplus A)\left(E\overline{D}\oplus D\right)\oplus$$
$$\oplus\left(E\oplus\overline{D}\right)(CB\oplus CA\oplus BA)\oplus CBA=$$
$$=ED\oplus(C\oplus B\oplus A)(E+D)\oplus$$
$$\oplus\left(E\oplus\overline{D}\right)\left(C(B\oplus A)\oplus BA\right)\oplus CBA=$$
$$=ED\oplus\overline{E}\overline{D}\oplus\overline{E}\overline{D}\oplus(C\oplus B\oplus A)(E+D)\oplus$$
$$\oplus\left(E\oplus\overline{D}\right)\left(C(B\oplus A)\oplus BA\right)\oplus CBA=$$
$$=E\oplus\overline{D}\oplus\overline{E+D}\oplus(C\oplus B\oplus A)(E+D)\oplus$$
$$\oplus\left(E\oplus\overline{D}\right)\left(C(B\oplus A)\oplus BA\right)\oplus CBA=$$
$$=\left((C\oplus B\oplus A)+\left(\overline{E+D}\right)\right)\oplus$$
$$\oplus\left(E\oplus\overline{D}\right)\overline{\left(C(B\oplus A)\oplus BA\right)}\oplus CBA=$$
$$=\left((C\oplus B\oplus A)+\left(\overline{E+D}\right)\right)\oplus$$
$$\oplus\overline{\left((E\oplus D))+C(B\oplus A)\oplus BA\right)}\oplus CBA=$$
$$=\left((C\oplus B\oplus A)+\left(\overline{E+D}\right)\right)\oplus$$
$$\oplus\left((E\oplus D)+C(B\oplus A)\oplus BA\right)\oplus\overline{CBA}=$$
$$=\overline{\left(\overline{(C\oplus B\oplus A)}(E+D)\right)}\oplus\oplus$$
$$\left((E\oplus D)+C(B\oplus A)\oplus BA\right)\oplus\overline{CBA}=$$
$$=\left(\overline{C\oplus B\oplus A}\right)(E+D)\oplus$$
$$\oplus\left((E\oplus D)+C(B\oplus A)\oplus BA\right)\oplus CBA.$$

The minimum function looks like this:

$$F_{\min}(E,D,C,B,A)=\left(\overline{C\oplus B\oplus A}\right)(E+D)\oplus$$
$$\oplus\left((E\oplus D)+C(B\oplus A)\oplus BA\right)\oplus CBA,$$

and contains 15 literals, which is two literals less than [35]:

$$S=\left(\left((B\oplus D)+(A\oplus B)\right)\oplus CE\right)\oplus\left(\left((B\oplus D)+(A\oplus B)\right)\oplus CE\right)\times$$
$$\times\left(\left(E\oplus(D\oplus C)\right)\oplus(A\oplus B)\right),$$

where the simplification of the function (Fig. 15) is carried out using the particle swarm optimization algorithm.

Example 15. Simplify the partially defined function $f(x_1, x_2, x_3, x_4, x_5, x_6)$ with a non-standard system given by the truth table (Fig. 16) [36].

No.	x_1	x_2	x_3	x_4	x_5	x_6	f
0	0	0	0	0	1	0	0
1	0	0	0	0	1	1	0
2	0	0	0	1	0	1	0
3	0	0	0	1	1	0	0
4	0	0	0	0	0	1	1
5	0	0	1	0	0	1	1
6	0	0	1	0	1	0	1
7	0	0	1	1	0	1	1
8	0	0	1	1	1	0	1

Fig. 16. Truth table of partially defined function $f(x_1, x_2, x_3, x_4, x_5, x_6)$

Solution.

The truth table of the PDNF of the partially defined function $f(x_1, x_2, x_3, x_4, x_5, x_6)$ (Fig. 16) takes the form (Fig. 17).

In the PDNF of the partially defined function $f(x_1, x_2, x_3, x_4, x_5, x_6)$ in Fig. 17, the following actions were carried out: to modules 9, 10, 13, 14, 8, 11, 12, 15, 24, 25, 26, 27, 28, 29, 30, 31, 40, 41, 42, 43, 44, 45 , 46, 47, 56, 57, 58, 59, 60, 61, 62, 63; 1, 9, 0, 8, 16, 17, 24, 25, 32, 33, 40, 41, 48, 49, 56, 57, the corresponding groups of which form intervals of the Boolean space containing complete combinatorial systems, 2-(5 , 32)-design and 2-(4, 16)-design, the operation of super-gluing variables was applied [17].

The result of logical operations of super-gluing the variables is recorded in the following matrix in Fig. 18.

It is important to note that the undefined sets of variables 4, 7, 18, 19, 20, 21, 22, 23, 34, 35, 36, 37, 38, 39, 50, 51, 52, 53, 54, 55 of the matrix in Fig. 17 are not recorded to the matrix in Fig. 18 since they do not participate in the simplification of the partially defined function $f(x_1, x_2, x_3, x_4, x_5, x_6)$ (Fig. 16). This ultimately reduces the complexity of simplifying the function in Fig. 16.

In one step, the minimum DNF of the function $f(x_1, x_2, x_3, x_4, x_5, x_6)$ is obtained (Fig. 16), which takes the form:

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

it coincides with [36], in which the simplification of the function is carried out by the method of directed sorting.

No.	x_1	x_2	x_3	x_4	x_5	x_6	f	No.	x_1	x_2	x_3	x_4	x_5	x_6	f
1	0	0	0	0	0	1	1	34	1	0	0	0	1	0	–
9	0	0	1	0	0	1	1	35	1	0	0	0	1	1	–
10	0	0	1	0	1	0	1	36	1	0	0	1	0	0	–
13	0	0	1	1	0	1	1	37	1	0	0	1	0	1	–
14	0	0	1	1	1	0	1	38	1	0	0	1	1	0	–
0	0	0	0	0	0	0	–	39	1	0	0	1	1	1	–
4	0	0	0	1	0	0	–	40	1	0	1	0	0	0	–
7	0	0	0	1	1	1	–	41	1	0	1	0	0	1	–
8	0	0	1	0	0	0	–	42	1	0	1	0	1	0	–
11	0	0	1	0	1	1	–	43	1	0	1	0	1	1	–
12	0	0	1	1	0	0	–	44	1	0	1	1	0	0	–
15	0	0	1	1	1	1	–	45	1	0	1	1	0	1	–
16	0	1	0	0	0	0	–	46	1	0	1	1	1	0	–
17	0	1	0	0	0	1	–	47	1	0	1	1	1	1	–
18	0	1	0	0	1	0	–	48	1	1	0	0	0	0	–
19	0	1	0	0	1	1	–	49	1	1	0	0	0	1	–
20	0	1	0	1	0	0	–	50	1	1	0	0	1	0	–
21	0	1	0	1	0	1	–	51	1	1	0	0	1	1	–
22	0	1	0	1	1	0	–	52	1	1	0	1	0	0	–
23	0	1	0	1	1	1	–	53	1	1	0	1	0	1	–
24	0	1	1	0	0	0	–	54	1	1	0	1	1	0	–
25	0	1	1	0	0	1	–	55	1	1	0	1	1	1	–
26	0	1	1	0	1	0	–	56	1	1	1	0	0	0	–
27	0	1	1	0	1	1	–	57	1	1	1	0	0	1	–
28	0	1	1	1	0	0	–	58	1	1	1	0	1	0	–
29	0	1	1	1	0	1	–	59	1	1	1	0	1	1	–
30	0	1	1	1	1	0	–	60	1	1	1	1	0	0	–
31	0	1	1	1	1	1	–	61	1	1	1	1	0	1	–
32	1	0	0	0	0	0	–	62	1	1	1	1	1	0	–
33	1	0	0	0	0	1	–	63	1	1	1	1	1	1	–

Fig. 17. The truth table of DDNF of the partially defined function $f(x_1, x_2, x_3, x_4, x_5, x_6)$ (Fig. 16)

x_1	x_2	x_3	x_4	x_5	x_6	f
–	–	–	0	0	–	1
–	–	1	–	–	–	1

Fig. 18. Completing the simplification of function $f(x_1, x_2, x_3, x_4, x_5, x_6)$ (Fig. 16)

Table 10 gives the results of simplification of Boolean functions borrowed from the works of other authors and a non-standard system.

Table 10

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

Example No.	Number of input variables	Simplification method ID	The result of simplification	Non-standard system
3	4	Method of uncoupling of conjunct terms [26]	14 literals	12 literals
4	4	Method by Quine-McCluskey [28]	13 literals abbreviated function	11 literals the minimum function
5	4	Method of uncoupling of conjunct terms [29, 30]	12 DNF literals	8 CNF literals
6	4	Method of uncoupling of conjunct terms [29]	5 inversions	3 inversions
7	4	Method of uncoupling of conjunct terms [29]	10 literals	6 literals
8	4	Method by Quine-McCluskey	Minimization results are the same	
9	4	Method by Quine-McCluskey	14 literal abbreviated function	MDNF of 8 literals MPNF of 5 literals
10	4	Carnot map [32]	Does not pass verification	4 literals
11	4	Veitch diagram [32]	6 inversions	5 inversions
12	4	Evolutionary method [33]	7 literals	6 literals
13	4	Genetic algorithm [34]	7 literals	6 literals
14	5	Swarm algorithm [35]	17 literals	15 literals
15	6	Method of directed search [36]	Minimization results are the same	

Table 10 gives a representative sample of examples in the simplification of Boolean functions by various methods. The non-standard function simplification system shows better or the same result.

6. Discussion of results of simplifying the Boolean functions by a non-standard system

The beginning of the simplification of Boolean functions by a non-standard system is the search for intervals of the Boolean space containing the combinatorial systems, 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 of 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 Boolean functions by the visual-matrix form of the analytical method.

The mathematical apparatus of a non-standard system for simplifying Boolean functions is the method of figurative transformations, which is considered in works [37–40], and others.

The technology for simplifying Boolean functions by a non-standard system is given in Table 11.

New components of the technology of simplification of Boolean functions by a non-standard system are given in Table 12.

The following results were obtained for each task:

1. A property of combinatorial systems with 2-(n, b)-design repetition is demonstrated, which consists in the ability of 2-(n, b)-design systems to reproduce (represent) the definition of logical super-gluing operations of variables in the form of a change in self-determination in the process of interaction with them. Thus, to simplify Boolean functions with a non-standard system, the logical operation of super-gluing variables can be represented by a complete combinatorial system, 2-(n, b)-design ($n>1, b>3$), and vice versa. For $n=1$, $b=2$, the combinatorial system, 2-(n, b)-design, represents a logical operation of simple gluing of variables.

To reproduce (represent) the definition of the logical operation of super-gluing variables by the combinatorial system with repeated 2-(n, b)-design, the mechanism of providing polymorphism (Fig. 3) and the verbal concept (8) are used. Examples of representation of logical operations of super-gluing of variables by combinatorial systems with repeated 2-(n, b)-design demonstrate the rules of super- and simple gluing of variables (9) to (11).

Table 11

Technology of simplification of Boolean functions by a non-standard system

1	Binary combinatorial systems with repetition, 2-(n, b)-design, 2-($n, x/b$)-design
2	Verbal and figurative presentation of information
3	The logical operation of super-gluing variables
4	Logical operation of incomplete super-gluing of variables
5	Hermeneutics of logical operations on binary equivalents of logical functions
6	Protocols of figurative transformations
7	The sign of the minimal logical function,
8	Minimization of Boolean functions on a complete truth table
9	The algorithm of the analytical method and its automation
10	Extension of the analytical method to other logical bases
11	Algebra of equivalent transformations in the class of perfect normal forms of functions of the Scheffer algebra
12	Algebra of equivalent transformations in the class of perfect implicative normal forms
13	Algorithms for simplifying Boolean functions using logical operations of absorption and super-gluing of variables
14	Stack of logical operations
15	Algorithms for simplifying the PNF of Boolean functions using the insertion of identical conjunctions with the following operation of super-gluing variables
16	Singular function
17	Algebra of equivalent transformations in the class of polynomial normal forms of Boolean functions
18	Mixed basis
19	Combining a sequence of logical operations of super- and simple gluing of variables with the possible use of an implicant table to identify redundant simple implicants
20	Dead-end DNFs can be simplified by carrying out all operations of generalized gluing of variables, followed by the use of an implicant table: – to detect extra simple implicants, – to select simple implicants with a minimum number of inversions
21	Identifying the locations of equivalent transformations using the 2-(n, b)-design, 2-($n, x/b$)-design combinatorial systems

Table 12

New components of the technology of simplification of Boolean functions by a non-standard system

1	Decimal format of equivalent figurative transformations
2	Basically, every 2-(n, b)-design, 2-($n, x/b$)-design system implies a logical operation of super- and/or incomplete gluing of variables, in particular in the case when different intervals of the Boolean space, which accommodate 2-(n, b)-design, 2-($n, x/b$)-design systems partially coincide
3	Theorem of the non-standard system of simplification of Boolean functions
4	Thesaurus of a non-standard system for simplifying Boolean functions

2. The beginning (principle) of a non-standard system for simplifying Boolean functions is the search for intervals in the truth table containing combinatorial systems, 2-(*n, b*)-design, 2-(*n, x/b*)-design, and not multi-pass logical transformations, which must be performed when simplifying a function by an analytical method (Assertions 1–4). The mutually unambiguous correspondence between the PDNF of the Boolean function and the combinatorial system, 2-(*n, b*)-design (2), gives a hyperparameter in the form of a set location of equivalent transformations. The findings of the 2-(*n, b*)-design, 2-(*n, x/b*)-design system directly and unambiguously point to logical operations for equivalent transformations. In turn, this implies an optimal algorithm for a non-standard system for simplifying Boolean functions. The formal basis of this reduction of the visual-matrix form is the algebraic simplification of the corresponding complete and/or incomplete PDNF of Boolean functions, such as (5).

Example 3 demonstrates the identification of the locations of equivalent transformations using 2-(*n, b*)-design combinatorial systems.

3. The theorem of the non-standard system of simplification of Boolean functions is formulated (theorem). According to the theorem, all non-redundant simple and/or super-gluing operations of variables must be performed to obtain a minimal function. As a result, this will provide a minimal function in the main basis without using the implicant table (examples 4, 5, 8, etc.). Thus, the problem of simplifying Boolean functions to the simplest normal equivalent is solved in one step, unlike the methods by McCall [41], Quine [42], McCluskey [18], in which the Boolean problem is solved in two steps.

4. The non-standard system for simplifying Boolean functions has its own systematized composition of information (knowledge) and settings, in the form of a formed thesaurus (Table 1 and Table 5), which provides orientation in this system.

5. A comparative analysis of the results of the simplification of Boolean functions by a non-standard system and examples of the simplification of functions by the evolutionary method, genetic, swarm algorithms, and the method for directed selection was carried out (Examples 12–15). In the vast majority of cases, the cost of implementing the minimum function obtained using a non-standard system is lower. For all the considered examples, the simplification procedure with a non-standard system is simpler. This makes it possible to simplify functions without the use of automated calculations.

The interpretation of the result is that the object of solving the problem of simplifying Boolean functions is combinatorial systems with repeated 2-(*n, b*)-design, 2-(*n, x/b*)-design, which is actually the truth table of the given functions. This makes it possible to focus the principle of simplification within the limits of the truth table and do without auxiliary objects, such as the Carnot map, Veitch diagrams, acyclic graph, etc. Equivalent transformations with combinatorial images, which by their properties have a greater information capacity, can effectively replace verbal procedures of algebraic transformations. In this regard, the non-standard system of simplification of Boolean functions allows the peculiarity that there is some analogy of the algorithm, which transforms the «messy» complexity of the simplification procedure by the analytical method into a complex order of figurative transformations.

Table 13 gives a comparison of methods for simplifying Boolean functions in the main basis.

Looking at Table 13, it can be seen that the Quine-McCluskey method uses the division of terms into groups with the same number of ones (zeros). In turn, a non-standard system uses combinatorial systems with repeated 2-(*n, b*)-design, 2-(*n, x/b*)-design. This simplifies and speeds up the search for the minimum function. The interpretation of 2-(*n, b*)-design systems on the Carnot map is carried out by correct contours. However, with an increase in the number of bits of the Boolean functions, the correct contours lose visibility. In this regard, the Carnot map becomes a difficult mathematical device for simplifying a function with more than five to six variables.

The peculiarity of the simplification of logical functions by a non-standard system, in contrast to the division of terms into groups with an equal number of ones (zeros), is the implication of the algorithm for simplifying functions by combinatorial systems with repeated 2-(*n, b*)-design, 2-(*n, x/b*)-design by searching for them on the binary structure of the truth table, in particular in the case when different intervals of the Boolean space partially coincide. As a result, verbal procedures of algebraic transformations, without established recommendations for the rational search for minimal dead-end forms of Boolean functions, are replaced by systematized equivalent figurative transformations.

Table 13

Comparative table of methods for simplifying Boolean functions in the main basis

Simplification method	The principle and complexity of simplification
Quine-McCluskey (method of simple implicants) [25]	Division of terms into groups with an equal number of units (zeros). This makes it possible to exclude comparisons that do not allow the operation of gluing variables in advance. The Quine-McCluskey method becomes complicated with a large number of variables. Then the time to search for the optimal function increases by 2^{2n}, where n is the bit size of the Boolean function. The limitation of the application area of the Quine-McCluskey method occurs when the time of operation of the method increases exponentially with the increase of input data. For a Boolean function of n variables, the upper limit of the number of basic implicants is $3^n/n$. If $n=32$, there may be more than 6.5×10^{15}
Carnot map (Veitch diagram)	For the Carnot map (Veitch diagram), the intervals of the Boolean space manifest themselves in the form of regular contours. The convenience and clarity of such a representation of a logical function is due to the fact that the logical terms, to which the operations of pairwise incomplete gluing and elementary absorption can be applied, are grouped on the Carnot map in the form of visually obvious rectangular arrays (regular contours) containing the same values in their cells (zeros and ones). However, the interpretation of the intervals of the Boolean space by regular contours is not of fundamental importance for the consequence – the Carnot map remains a redundant matrix. In addition to the above, the Carnot map becomes a difficult mathematical device for simplifying a function with more than five to six variables
Non-standard system	The simplification of Boolean functions by a non-standard system is based on the search for intervals of the Boolean space containing the combinatorial systems, 2-(*n, b*)-design, 2-(*n, x/b*)-design. Since these systems are logical operations at the same time, this ensures the effectiveness of the implication of the algorithm for simplifying the given functions. When increasing the bit rate of Boolean functions, the properties of the 2-(*n, b*)-design, 2-(*n, x/b*)-design systems do not change

In contrast to the correct contours of the Carnot map, the properties of the 2-(n, b)-design, 2-(n, x/b)-design systems do not change with the increase in the number of bits of the Boolean functions, and clarity is not lost. This makes it possible to search for them efficiently, and therefore effectively imply the algorithm for simplifying logical functions for a larger number of input variables.

The established location of equivalent transformations by the 2-(n, b)-design, 2-(n, x/b)-design systems implies a systematic procedure for simplifying Boolean functions. As a result, the problem of implementing the simplification of logical functions with a non-standard system is self-solved. The implication of the algorithm for simplifying Boolean functions is provided by the interpretation of 2-(n, b)-design systems by logical operations of simple and/or super-gluing variables and vice versa, for carrying out equivalent transformations of logical expressions. Thus, the principle of minimizing functions by a non-standard system is established, which reduces the complexity and improves the efficiency of the procedure for simplifying Boolean functions, compared, in particular, with the transfer of the problem of simplification to the algebraic domain, in which algorithms for calculations are applicable on the Graebner basis [9]; by simplifying the terms of Boolean functions in the context of a formal, axiomatically defined theory [13]; machine learning techniques [10]; techniques for finding approximate Boolean schemes, large training sets of examples with a large number of primary input data, methods for restoring the accuracy of a logical scheme [14]; nonlinear mixed integer programming [15], and others.

Only the fundamental non-standard nature in principle solves the problem of the laboriousness of the procedure for simplifying logical functions when they are represented in a visual-matrix form.

The application of the result makes it possible to improve and expand the technology of designing electronic components and devices for their use in digital technologies, which are based on basic, polynomial, and other bases.

The visibility of 2-dimensional binary matrices allows for a manual way of simplifying Boolean functions using a mathematical editor, for example MathType 7.4.0 (USA): Examples 1, 2, 3, 4, 7, 9, 10, 12, 15 (minimization of DNF), 3, 6, 9, 13, 14 (minimization of PNF), 5, 11 (minimization of CNF), or using MS Word tables: example 8 (minimization of DNF).

The use of a non-standard system for simplifying functions in the basic and polynomial bases brings, to a certain extent, the problem of simplifying Boolean functions to the level of a well-researched problem in the class of disjunctive-conjunct term normal forms (DCNF) of Boolean functions. A limitation of the application of the method for figurative transformations is the cases when the switching function is represented in a mixed basis. In this case, the function must be represented by one logical base.

The weakness of the considered method is in its small practical application for the simplification of Boolean 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 protocols for simplifying logical functions in Boolean and Reed-Muller logic bases, followed by the creation of a library of protocols that illustrate the corresponding image transformations.

A prospect for further research on the simplification of logical functions may be the use of Boolean formulas of a special type, called conjunct term normal form (2-CNF) or Krohm formulas. The problem is known as the computational problem of assigning values to variables, each of which has two possible values 0 or 1, in order to satisfy a pairwise constraint system – 2-satisfiability, (2-SAT), 3-satisfiability, (3-SAT), so that the result of this function is equal to unity.

7. Conclusions

1. A property of combinatorial systems with 2-(n, b)-design repetition has been demonstrated, which consists in the ability of 2-(n, b)-design systems to reproduce (represent) the definition of logical super-gluing operations of variables in the form of a change in self-determination in the process interaction with them. Thus, to simplify Boolean functions with a non-standard system, the logical operation of super-gluing variables can be represented by a complete combinatorial system, 2-(n, b)-design ($n>1$, $b>3$), and vice versa. For $n=1$, $b=2$, the combinatorial system, 2-(n, b)-design, represents a logical operation of simple gluing of variables.

2. Detection of the 2-(n, b)-design, 2-(n, x/b)-design systems in the truth table directly and unambiguously establishes the locations of equivalent transformations for Boolean functions. The interpretation of the result is that the 2-(n, b)-design, 2-(n, x/b)-design systems represent logical operations. Therefore, the detection of combinatorial systems in the truth table directly and unambiguously indicates logical operations for equivalent transformations of Boolean expressions. This, in turn, implies an optimal algorithm for a non-standard system of simplifying logical functions.

3. A theorem of the non-standard system for the simplification of Boolean functions has been formulated. According to the theorem of the non-standard system of simplification of Boolean functions, all non-redundant simple and/or super-gluing operations of variables must be performed to obtain the minimum function. The result of these actions will be the minimal function in the main basis without using the implicant table.

4. The non-standard system for simplifying Boolean functions has its own systematized composition of information (knowledge) and settings, in the form of a thesaurus, which provides orientation in this system. Through the comparison of thesauri, correspondence between the concepts of a non-standard system, the Quine-McCluskey method and the Carnot map was established.

5. A comparative analysis of examples of simplification by the evolutionary method, genetic and swarm algorithms demonstrated that heuristic methods and algorithms provide a minimum function with a higher cost of implementation, compared to simplification by a non-standard system. In addition to the above, obtaining a minimum function by a non-standard system is a simpler procedure.

A comparative analysis of the example of simplifying a 6-bit partially defined Boolean function in the main basis by the method for directed sorting demonstrated that increasing the bitness of Boolean functions does not change the properties of 2-(n, b)-design systems. This makes it

possible to search for them effectively, and therefore effectively imply the algorithm for simplifying functions for a larger number of input variables. In one step, a non-standard system, without using an implicant table, obtained the minimum DNF of the function. The results of the simplification of the two methods coincide.

Conflict of interest

The author declares that he has no conflicts of interest in relation to the current study, including financial, personal, authorship, or any other, that could affect the study and the results reported in this paper.

Funding

The study was conducted without financial support.

Data availability

All data are available in the main text of the manuscript.

Use of artificial intelligence

The author confirms that he did not use artificial intelligence technologies when creating the current work.

References

1. 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
2. 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
3. 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
4. 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
5. Burmistrov, S. V., Khotunov, V. I., Zakharova, M. V., Mykhaylyuta, S. L., Liuta, M. V. (2023). Index method of minimization of Boolean functions. Bulletin of Cherkasy State Technological University, 2, 24–37. https://doi.org/10.24025/2306-4412.2.2023.273763
6. Udovenko, A. (2023). DenseQMC: an efficient bit-slice implementation of the Quine-McCluskey algorithm. arXiv. https://doi.org/10.48550/arXiv.2302.10083
7. Ignatiev, A., Previti, A., Marques-Silva, J. (2015). SAT-Based Formula Simplification. Theory and Applications of Satisfiability Testing -- SAT 2015, 287–298. https://doi.org/10.1007/978-3-319-24318-4_21
8. Calò, E., Levy, J. (2023). General Boolean Formula Minimization with QBF Solvers. Artificial Intelligence Research and Development. https://doi.org/10.3233/faia230705
9. Faroß, N., Schwarz, S. (2023). Gröbner Bases for Boolean Function Minimization. 8th International Workshop on Satisfiability Checking and Symbolic Computation. Available at: https://ceur-ws.org/Vol-3455/short4.pdf
10. Le Charlier, B., Atindehou, M. M. (2016). A Method to Simplify Expressions: Intuition and Preliminary Experimental Results. EPiC Series in Computing. https://doi.org/10.29007/jv63
11. Prasad, V. C. (2018). Novel method to simplify Boolean functions. Automatyka/Automatics, 22 (2), 29. https://doi.org/10.7494/automat.2018.22.2.29
12. Siládi, V., Filo, T. (2013). Quine-Mccluskey algorithm on GPGPU. AWERProcedia Information Technology & Computer Science, 04, 814–820. https://doi.org/10.13140/2.1.2113.1522
13. Costamagna, A., De Micheli, G. (2023). Accuracy recovery: A decomposition procedure for the synthesis of partially-specified Boolean functions. Integration, 89, 248–260. https://doi.org/10.1016/j.vlsi.2022.12.008
14. Boroumand, S., Bouganis, C.-S., Constantinides, G. A. (2021). Learning Boolean Circuits from Examples for Approximate Logic Synthesis. Proceedings of the 26th Asia and South Pacific Design Automation Conference. https://doi.org/10.1145/3394885.3431559
15. Dimopoulos, A. C., Pavlatos, C., Papakonstantinou, G. (2022). Multi-output, multi-level, multi-gate design using non-linear programming. International Journal of Circuit Theory and Applications, 50 (8), 2960–2968. https://doi.org/10.1002/cta.3300
16. 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
17. 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
18. McCluskey, E. J. (1956). Minimization of Boolean Functions. Bell System Technical Journal, 35 (6), 1417–1444. https://doi.org/10.1002/j.1538-7305.1956.tb03835.x

19. Zakrevskiy, A. D. (1981). Logicheskiy sintez kaskadnyh shem. Moscow: Nauka, 416.
20. 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
21. Rytsar, B. E. (1997). Metod minimizatsii bulevyh funktsiy. Problemy upravleniya i informatiki, 2, 100–113.
22. Rytsar, B. (1997). Minimization method of Boolean functions. SPIE Proceedings. https://doi.org/10.1117/12.284818
23. Rytsar, B. Ye. (2005). Sposib pobudovy koniunktermovoho polia bulovoi funktsiyi. Visnyk Natsionalnoho universytetu «Lvivska politekhnika», 534, 21–24. Available at: http://surl.li/siygp
24. Minimizatsiya bulevyh funktsiy v klasse DNF. Available at: http://vuz.exponenta.ru/PDF/book/logic.pdf
25. Kapuro, P. A. (2014). Tsifrovye funktsional'nye ustroystva v telekommunikatsiyah. Ch. 1: Bazovye tsifrovye funktsional'nye ustroystva. Minsk: BGUIR, 64. Available at: http://surl.li/siygv
26. Rytsar, B. Ye. (2015). The Minimization Method of Boolean Functionns in Polynomial Set-theoretical Format. Conference: Proc. 24th Inter. Workshop, CS@P'2015. Vol. 2. Rzeszow, 130–146. Available at: https://www.researchgate.net/publication/298158364_The_Minimization_Method_of_Boolean_Functionns_in_Polynomial_Set-theoretical_Format
27. 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
28. Zakrevskiy, A. D., Pottosin, Yu. V., Cheremisinova, L. D. (2007). Logicheskie osnovy proektirovaniya diskretnyh ustroystv. Moscow: FIZMATLIT, 592.
29. Rytsar, B. Ye., Belovolov, A. O. (2021). A New Method of the Logical Functions Minimization in the Polynomial Set-Theoretical Format. «Handshaking» Procedure. Control Systems and Computers, 1 (291), 03–14. https://doi.org/10.15407/csc.2021.01.003
30. Rytsar, B. Ye. (2004). Teoretyko-mnozhynni optymizatsiyni metody lohikovoho syntezu kombinatsiynykh merezh. Lviv, 33. Available at: http://www.irbis-nbuv.gov.ua/publ/REF-0000263283
31. Metod Kvayna – Mak-Klaski nahozhdeniya sokrashchennoy DNF dvoichnoy funktsii. Available at: https://ematica.xyz/metodichki-i-knigi-po-matematike/kurs-lektcii-po-matematicheskoi-logike-i-teorii-algoritmov-aliev/6-1-metod-kvaina-mak-klaski-nakhozhdeniia-sokrashchennoi-dnf-dvoichnoi-funktcii
32. Kupriyanova, D. V., Luk'yanova, I. V., Lutsik, Yu. A. (2021). Arifmeticheskie i logicheskie osnovy vychislitel'noy tehniki. Minsk: BGUIR, 72.
33. Chong, K. H., Aris, I. B., Sinan, M. A., Hamiruce, B. M. (2007). Digital Circuit Structure Design via Evolutionary Algorithm Method. Journal of Applied Sciences, 7 (3), 380–385. https://doi.org/10.3923/jas.2007.380.385
34. Coello Coello, C. A., Aguirre, A. H. (2002). Design of combinational logic circuits through an evolutionary multiobjective optimization approach. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 16 (1), 39–53. https://doi.org/10.1017/s0890060401020054
35. Coello Coello, C. A., Hernandez Luna, E., Hernandez Aguirre, A. (2004). A comparative study of encodings to design combinational logic circuits using particle swarm optimization. Proceedings. 2004 NASA/DoD Conference on Evolvable Hardware, 2004. https://doi.org/10.1109/eh.2004.1310811
36. Sysoienko, A. A., Sysoienko, S. V. (2023). Method for minimization of boolean functions with a large number of variables based on directed enumeration. Bulletin of Cherkasy State Technological University, 1, 42–51. https://doi.org/10.24025/2306-4412.1.2023.274914
37. Solomko, M., Batyshkina, I., Voitovych, I., Zubyk, L., Babych, S., Muzychuk, K. (2020). Devising a method of figurative transformations for minimizing boolean functions in the implicative basis. Eastern-European Journal of Enterprise Technologies, 6 (4 (108)), 32–47. https://doi.org/10.15587/1729-4061.2020.220094
38. Riznyk, V. V., Solomko, M. T. (2017). Combinatorial method of minimizing boolean functions. Visnyk Natsionalnoho universytetu «Lvivska politekhnika». Serie: Kompiuterni systemy ta merezhi, 881, 135–151. https://doi.org/10.23939/csn2017.881.135
39. Riznyk, V., Solomko, M. (2017). Minimization of Boolean functions by combinatorial method. Technology Audit and Production Reserves, 4 (2(36)), 49–64. https://doi.org/10.15587/2312-8372.2017.108532
40. Riznyk, V., Solomko, M. (2018). Research of 5-bit boolean functions minimization protocols by combinatorial method. Technology Audit and Production Reserves, 4 (2 (42)), 41–52. https://doi.org/10.15587/2312-8372.2018.140351
41. Markham Brown, F. (2010). McColl and Minimization. History and Philosophy of Logic, 31 (4), 337–348. https://doi.org/10.1080/01445340.2010.517387
42. Quine, W. V. (1952). The Problem of Simplifying Truth Functions. The American Mathematical Monthly, 59 (8), 521–531. https://doi.org/10.1080/00029890.1952.11988183