After the training stage and the formation of reference sets of robust noise and the spectral characteristics of the analysed noisy signals, as well as the spectral characteristics of the noise, a comparison is made between newly received object monitoring estimates from current noisy signals. At the same time, a decision is made as to whether there is a risk of changes in the technical state of the controlled object or not. In the first case, the object goes to the rank, requiring the attraction of mobile monitoring and diagnostic systems, through which the final analysis and decision-making is carried out. In the second case, the monitoring of the object continues.

References

1. Musaeva, N. Correlation matrices in problems of identification of seismic stability and technical condition of high-rise buildings and building structures [Text] / N. Musaeva, E. Aliyev, U. Sattarova, N. Rzayeva // 2012 IV International Conference «Problems of Cybernetics and Informatics» (PCI). – IEEE, 2012. – P. 157–173. doi:10.1109/icpci.2012.6486357
2. Kollakot, R. Diagnostika povrezhdenii [Text] / R. Kollakot. – Moscow: Mir, 1989. – 516 p.
3. Gaskin, V. V. Seismostoikost' zdanii i transportnyh sooruzhenii [Text]: Handbook / V. V. Gaskin, I. A. Ivanov. – Irkutsk: IrGUPS, 2005. – 76 p.
4. Sushchev, S. P. Monitoring ustoichivosti i ostatochnogo resursa vysotnyh zdanii i sooruzhenii s primeneniem mobil'nogo diagnosticheskogo kompleksa «Strela» [Text] / S. P. Sushchev // Unikal'nye i spetsial'nye tehnologii v stroitel'stve (UST-Build 2005). – Moscow: TsNTSMO, 2005. – P. 68–71.
5. Lei, Y. Structural damage detection with limited input and output measurement signals [Text] / Y. Lei, Y. Jiang, Z. Xu // Mechanical Systems and Signal Processing. – 2012. – Vol. 28. – P. 229–243. doi:10.1016/j.ymssp.2011.07.026
6. Moon, B. Statistical random response analysis and reliability design of structure system with non-linearity [Text] / B. Moon, C.-T. Lee, B.-S. Kang, B. S. Kim // Mechanical Systems and Signal Processing. – 2005. – Vol. 19, No. 5. – P. 1135–1151. doi:10.1016/j.ymssp.2004.05.003
7. Aliev, T. Digital Noise Monitoring of Defect Origin [Text] / T. Aliev. – Boston, MA: Springer US, 2007. – 224 p. doi:10.1007/978-0-387-71754-8
8. Aliev, T. Robust Technology with Analysis of Interference in Signal Processing [Text] / T. Aliev. – Boston, MA: Springer US, 2003. – 200 p. doi:10.1007/978-1-4615-0093-3
9. Aliev, T. A. System of robust noise monitoring of anomalous seismic processes [Text] / T. A. Aliev, A. M. Abbasov, Q. A. Guluyev, F. H. Pashaev, U. E. Sattarova // Soil Dynamics and Earthquake Engineering. – 2013. – Vol. 53. – P. 11–25. doi:10.1016/j.soildyn.2012.12.013
10. Aliev, T. A. Noise technologies and systems for monitoring the beginning of the latent period of accidents on fixed platforms [Text] / T. A. Aliev, T. A. Alizada, N. E. Rzayeva // Mechanical Systems and Signal Processing. – 2017. – Vol. 87. – P. 111–123. doi:10.1016/j.ymssp.2016.10.014

РАЗРАБОТКА РОБАСТНЫХ АЛГОРИТМОВ И СИСТЕМЫ КОНТРОЛЯ ТЕХНИЧЕСКОГО СОСТОЯНИЯ СТРОИТЕЛЬНЫХ ОБЪЕКТОВ

Предложена система контроля за изменениями в техническом состоянии строительных объектов. В основе системы лежат технологии робастного noise анализа зашумленных сигналов, которые улавливаются датчиками при сейсмических толчках, встряске при движении поезда метро, колебаниях от ветра и т. д., а также спектральный анализ помех. Система также позволяет выявлять неисправности в скрытом периоде зарождения.

Ключевые слова: зашумленный сигнал, корреляционная функция, спектральные характеристики, строительный объект, техническое состояние, система контроля.

Narmin Eldar Rzayeva, *Lecturer, Head of Research Division, Department of Information Technologies and Systems, Azerbaijan University of Architecture and Construction, Baku, Azerbaijan, e-mail: nikanel1@gmail.com, ORCID: http://orcid.org/0000-0003-0397-5412*

UDC 519.718
DOI: 10.15587/2312-8372.2017.118336

**Riznyk V.,
Solomko M.**

APPLICATION OF SUPER-STICKING ALGEBRAIC OPERATION OF VARIABLES FOR BOOLEAN FUNCTIONS MINIMIZATION BY COMBINATORIAL METHOD

Розглянуто нову процедуру алгебри логіки – супер-склеювання змінних, яка застосовується при наявності у структурі таблиці істинності повної бінарної комбінаторної системи з повторенням або неповної бінарної комбінаторної системи з повторенням. Ефективність алгебричної операції суперсклеювання змінних суттєво спрощує алгоритм мінімізації булевих функцій, що дозволяє здійснювати ручну мінімізацію функцій з числом змінних до 10.

Ключові слова: *булева функція, метод мінімізації, мінімізація логічної функції, блок-схема з повторенням, мінтерми, супер-склеювання змінних.*

1. Introduction

Boolean functions minimization is still popular in various areas of digital technologies, such as PLA design, built-in self-test (BIST), control system design and the like.

The problem of disjunctive normal form (DNF) minimization is one of the multiextremal logical-combinatorial problems and reduces to an optimal reduction in the number of logical elements of the gate circuit without loss of its functionality. It should be noted that in the general formulation this problem has not yet been solved, but it has been well studied in the class of disjunctive conjunctive normal forms (DCNF).

The disadvantages of the known methods for Boolean functions minimization are associated with a rapid growth in the amount of computation, which results in an increase in the number of computational operations, and, consequently, in the increase in the number of variables of the logical function. In particular, the Carnot map is usually difficult to recognize with an increase in the number of variables greater than four or five, so this method is inexpedient to use with a large number of variables. Despite the great perfection of the Quine-McClusky method compared to the Carnot maps, it also has limited practical application due to the exponential growth of calculation time with the increase in the number of variables. It can be shown that for a function of n variables, the upper limit of the number of basic implicants is $3^n \ln(n)$ [1]. For example, it is known that for $n=32$ the number of basic implicants can exceed 6.5×10^{15}.

The result of Boolean function minimization depends on the speed of the computing device, its reliability and power savings.

The peculiarities of the combinatorial method [2] are in the greater informativeness of the process of solving the problem in comparison with the algebraic method of the function minimization, due to tabular organization and the introduction of the image-transformation apparatus. The object of solving the problem of Boolean function minimization by a combinatorial method is a block-scheme with repetition, which properties, in turn, allow the rules of the algebra of logic to be supplemented with new rules for simplifying the logical function.

The algorithm for Boolean function minimization is one of the central practically important problems that arise when designing computing devices. Therefore, the study of new rules of the algebra of logic to simplify the algorithm for Boolean function minimization without losing its functionality with increasing the number of variables is relevant.

2. The object of research and its technological audit

The object of simplification of Boolean function minimization problem by a combinatorial method is a new procedure for the algebra of logic – super-sticking of variables, which is performed if there is a complete binary combinatorial system with repetition or an incomplete binary combinatorial system with repetition in the truth table structure.

The procedure for reducing the complete perfect disjunctive normal form (PDNF) of a logical function gives unity. Since the complete PDNF uniquely determines the complete binary combinatorial system with repetition, and vice versa, this gives grounds to delete all blocks of a complete binary combinatorial system with a repetition from a truth table which structure allows to apply rules for super-sticking of variables.

The procedure for simple sticking of variables is a special case of the procedure for super-sticking of variables. Variables that form a complete binary combinatorial system with repetition or an incomplete binary combinatorial system with repetition can occupy any bit of the minterm of the logical function.

The effectiveness of the algebraic operation of super-sticking of variables greatly simplifies the algorithm for Boolean function minimization and allows manual minimization of functions with a number of variables up to 10. The average complexity of the algorithm for logical function minimization by combinatorial method using the super-sticking procedure for variables is estimated from the growth dynamics of the number of image transformations of the combinatorial minimization method with increasing the bit capacity of the Boolean function. For $n<7$, the dynamics is characterized by the linear law $O(n)$, and with increasing number of variables up to 10 – by $O(n^2)$.

Disadvantages of the combinatorial method of manual minimization using the procedure of super-sticking of variables are associated with the rapid growth of algorithmic complexity with increasing number of variables of the logical function. Function minimization with a number of variables more than $12\div14$ requires updating the library of submatrices on which the super-sticking procedure is based.

3. The aim and objectives of research

The aim of research is simplification of the combinatorial method of Boolean function minimization using a new procedure for algebra of logic – super-sticking of variables and establishing the properties of such procedure.

To achieve this aim, it is necessary to solve the following tasks:

1. To establish the adequacy of the application of the algebraic procedure of super-sticking of variables for Boolean function minimization process.
2. To determine the properties of the operation of super-sticking of variables when using structures of a complete binary combinatorial system with repetition and an incomplete binary combinatorial system with repetition.
3. To verify the combinatorial method when applying the rule of super-sticking of variables and evaluate the complexity of the algorithm for Boolean function minimization by combinatorial method.
4. To conduct a comparative analysis of the performance and complexity of algorithms for Boolean functions minimization obtained using the super-sticking rule for variables, with other minimization methods.

4. Research of existing solutions of the problem

A classical object-oriented algorithm for Boolean function minimization using Carnot maps is described in [3], which presents language stereotypes and class diagrams, as well as a performance analysis of a unified Boolean function minimization model. In [4], cubic methods of Boolean functions minimization are considered as yet another variant of searching for a minimal function. The main aim of the paper [4] is taking advantage of the cubic method for minimization of the logical functions, in particular by achieving the minimum cost of the solution.

A fast and effective heuristic algorithm for Boolean functions minimization (ESOP) is considered in [5]. This algorithm is based on new transformations of the cube. Its authors prove that the quality of the corresponding coverage corresponds, and in some cases exceeds the current level of heuristic minimization. In [6], an extended QMC algorithm (*e* QMC) is presented, which improves the performance of Boolean function minimization by the Quine-McCluskey method. The paper demonstrates the increase in speed and performance of computer memory by simulating the function minimization process.

In the publication [7] the effective algorithm of synthesis and ESCT (Exclusive or Sum of Complex Terms) exact minimization of Boolean functions of not more than six variables is presented. This kind of logical expressions can be turned into a special honeycomb architecture, which is called the architecture of the reverse cascade wave. It is proved that such topology is reversible and can help in the development of quantum circuits. The proposed algorithm is the first one that gives a solution to the problem of finding minimum ESCT expressions for switching functions to six input variables. A comprehensive survey of methods for logical functions minimization is demonstrated in [8]. Methods are considered by their purpose, methodology, implementation and benefits. A comparison of the approaches taken to minimization of logical functions is presented.

A new technique for the two-step optimization of combinational logic is described in [9]. The technique can be applied to arbitrary combinational logic tasks, and often brings improvements even after optimization by standard methods. This optimization technique is used to improve software performance. In [10], a method is shown where the optimization process can include not only the search for an equivalent logical expression, but also the definition of specific conditions under which logical expressions can be further reduced. These types of elements in the logical design are considered as the «degree of freedom». In such cases, the user can optimize the given design based on the degree of freedom. Therefore, the search for alternative solutions is desirable, since it can eventually provide the optimal Boolean expression. In [11], multivalued logic is presented as a generalization of classical Boolean logic at higher levels of abstraction, where the variables often vary over a set of symbolic codes. Use of multi-valued logic can make the task of design more intuitive. The designer can first manipulate and optimize the meaningful logic, and then perform the appropriate encoding and output the problem to Boolean algebra. This allows to better study the design space, because binary coding is postponed and many-valued optimization does not affect the reliability of such solutions in the final stage. In the work attempts to build multivalued integrated circuits (ICs) are described, starting from 3 large designs can be traced back to 1970. In [12], the optimization of the scheme of a 2-bit comparator is presented by comparing different logical styles that are used to design the comparator circuit. The comparison between the different designs is calculated by modeling, which is performed for 90 nm technology in the Tanner EDA Tool. After simulating all projects, the final results are obtained with respect to power consumption, signal delay, and power. In particular, they compare PTL, NMOS, CMOS technology.

In contrast to the publications [3–12], in this paper, the object of simplifying the process of Boolean function minimization is a new procedure of the algebra of logic – super-sticking of variables, which occurs when there is a complete or incomplete binary combinatorial systems with repetition in the truth table structure. The procedure for reducing the complete perfect disjunctive normal form (PDNF) of a logical function gives unity. Since the complete PDNF uniquely determines the complete binary combinatorial system with repetition, and vice versa, this gives grounds to delete all blocks of a complete binary combinatorial system with a repetition from a truth table which structure allows to apply rules for super-sticking of variables.

The mathematical apparatus of the repetition block-design makes it possible to obtain more information about the orthogonality, contiguity, uniqueness of truth table blocks (combinatorial system). Equivalent transformations by graphic images in the form of two-dimensional matrices have a large information capacity in their properties, so they can effectively replace verbal procedures of algebraic transformations.

5. Methods of research

5.1. Binary combinatorial system with repetition. If a set A is given, then it is possible to consider a new set $M(A)$ – the set of all its subsets. The set of all subsets of A having k elements is denoted as $M_k(A)$.

Example 1. Let $A=\{a, b, c\}$, then:

$$M(A)=\{\{a\},\{b\},\{c\},\{a,b\},\{a,c\},\{b,c\},\{a,b,c\},\varnothing\};$$

$$M_2(A)=\{\{a,b\},\{a,c\},\{b,c\}\}.$$

Let's convinced that:

$$N(M(A))=8=2^3,\quad N(M_2(A))=3.$$

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

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

The following equality also holds:

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

Since C_n^k – the number of k-element subsets of a set of n elements, the sum on the left-hand side of expression (1) is the number of all subsets.

Example 2. By formula (1), calculate the number of all subsets of the set $A=\{a,b,c,d\}$.

$$N(M(A))=C_4^0+C_4^1+C_4^2+C_4^3+C_4^4=$$
$$=1+4+6+4+1=16=2^4.$$

Let's note that the set $A=\{a,b,c,d\}$, except for the recalculation of its elements, can also determine the numbers of the positions on which the element α is located. For example, a can mean the first position, b can mean the second position of the set $A=\{a,b,c,d\}$, etc. A subset of the set $A=\{a,b,c,d\}$ there will be subsets containing the element α at k positions, $k=0,...,n$, where n – the number of positions of the set A. In general, the element α can occupy several positions on the set A, thus the element α is repeated on the set A.

Let $\alpha=1$, then the positions on which the element α is absent are affected by zero.

Example 3. For set $A=\{a,b,c,d\}$, which determines the position numbers, let's take $\alpha=1$. Then the subsets of the set A will have the following form:

$$\begin{aligned}
&(0,0,0,0);\ (1,0,0,0);\\
&(0,0,0,1);\ (1,0,0,1);\\
&(0,0,1,0);\ (1,0,1,0);\\
&(0,0,1,1);\ (1,0,1,1);\\
&(0,1,0,0);\ (1,1,0,0);\\
&(0,1,0,1);\ (1,1,0,1);\\
&(0,1,1,0);\ (1,1,1,0);\\
&(0,1,1,1);\ (1,1,1,1).
\end{aligned} \tag{2}$$

The number of all k-element subsets of the set $A=\{a,b,c,d\}$, which determines the position numbers, is calculated by formula (1).

$$N\left(M_0\left(A\right)\right)=C_4^0=1,$$

$$N\left(M_1\left(A\right)\right)=C_4^1=4,$$

$$N\left(M_2\left(A\right)\right)=C_4^2=6,$$

$$N\left(M_3\left(A\right)\right)=C_4^3=4,$$

$$N\left(M_3\left(A\right)\right)=C_4^4=1.$$

$$\begin{aligned}
&N\left(M\left(A\right)\right)=N\left(M_0\left(A\right)\right)+N\left(M_1\left(A\right)\right)+\\
&+N\left(M_2\left(A\right)\right)+N\left(M_3\left(A\right)\right)+N\left(M_4\left(A\right)\right)=16.
\end{aligned}$$

The configuration (2) is a complete combinatorial system with the repetition of the element α, which denote as:

2-(n,b)-design,

where n – the block width of the system; b – the number of blocks of the complete system, 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-$(4,16)$-design is a complete binary combinatorial system with repetition, consisting of 4-bit blocks, the number of blocks is 16.

5.2. Algebraic operation of super-sticking of variables. Combinatorial properties of the block-scheme with repetition allow to supplement the rule of the algebra of the logic of sticking of variables [2], the rule of super-sticking of variables.

For a 4-bit logic function, the super-sticking rule for variables has the following form:

– the first rule:

$$\begin{vmatrix}
0 & 0 & 0 & x\\
0 & 0 & 1 & x\\
0 & 1 & 0 & x\\
0 & 1 & 1 & x\\
1 & 0 & 0 & x\\
1 & 0 & 1 & x\\
1 & 1 & 0 & x\\
1 & 1 & 1 & x
\end{vmatrix}=x; \tag{3}$$

– the second rule:

$$\begin{vmatrix}
0 & 0 & x & y\\
0 & 1 & x & y\\
1 & 0 & x & y\\
1 & 1 & x & y
\end{vmatrix}=xy; \tag{4}$$

– the third rule:

$$\begin{vmatrix}
0 & x & y & z\\
1 & x & y & z
\end{vmatrix}=xyz. \tag{5}$$

The first rule uses 2-$(3,8)$-design. The second rule uses 2-$(2,4)$-design. The third rule uses 2-$(1,2)$ design.

The procedure for reducing the total perfect disjunctive normal form (PDNF) of the logical function gives unity. For example, reducing the 3-bit full PDNF looks like this:

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

Since the complete PDNF uniquely determines the complete combinatorial system with the repetition of 2-(n,b)-design and vice versa, this gives grounds to remove all blocks of the complete combinatorial system from matrices that demonstrate super-sticking rules (3)–(5). Further, applying the law of idempotency to a variable x (xy; xyz) let's obtain the result of contraction by the rule of super-sticking of variables. Rule (5) manifests itself as a simple sticking of variables and is a particular case of rules (3) and (4).

The variables x, y, z, which form a complete combinatorial system with a repetition of 2-(n,b)-design, can occupy any discharge of the minterm of the logical function.

For a 5-bit logical function, the rules for super-sticking of variables are:

– the first rule:

$$\begin{vmatrix}
0 & 0 & 0 & 0 & x\\
0 & 0 & 0 & 1 & x\\
0 & 0 & 1 & 0 & x\\
0 & 0 & 1 & 1 & x\\
0 & 1 & 0 & 0 & x\\
0 & 1 & 0 & 1 & x\\
0 & 1 & 1 & 0 & x\\
0 & 1 & 1 & 1 & x\\
1 & 0 & 0 & 0 & x\\
1 & 0 & 0 & 1 & x\\
1 & 0 & 1 & 0 & x\\
1 & 0 & 1 & 1 & x\\
1 & 1 & 0 & 0 & x\\
1 & 1 & 0 & 1 & x\\
1 & 1 & 1 & 0 & x\\
1 & 1 & 1 & 1 & x
\end{vmatrix}=x; \tag{6}$$

– the second rule:

$$\begin{vmatrix} 0 & 0 & 0 & x & y \\ 0 & 0 & 1 & x & y \\ 0 & 1 & 0 & x & y \\ 0 & 1 & 1 & x & y \\ 1 & 0 & 0 & x & y \\ 1 & 0 & 1 & x & y \\ 1 & 1 & 0 & x & y \\ 1 & 1 & 1 & x & y \end{vmatrix} = xy; \qquad (7)$$

– the third rule:

$$\begin{vmatrix} 0 & 0 & x & y & z \\ 0 & 1 & x & y & z \\ 1 & 0 & x & y & z \\ 1 & 1 & x & y & z \end{vmatrix} = xyz; \qquad (8)$$

– the fourth rule:

$$\begin{vmatrix} 0 & x & y & z & t \\ 1 & x & y & z & t \end{vmatrix} = xyzt. \qquad (9)$$

The first rule (6) uses 2-(4, 16)-design. The second rule (7) uses 2-(3, 8)-design. The third rule (8) uses 2-(2, 4)-design. The fourth rule (9) uses 2-(1, 2)-design.

The variables *x, y, z, t,* which form a complete binary combinatorial system with a repetition of 2-(n, b)-design, can occupy any bit of the minterm of a 5-bit logical function.

Another variant of applying the rule for super-sticking of variables is shown by the combinatorial configuration where the combinatorial system 2-(3, 8)-design is used in the configuration variant, when there is one column with the same variables y, and the second column contains equally the variables x and $\bar{x}$:

$$\begin{vmatrix} 0 & 0 & 0 & \bar{x} & y \\ 0 & 0 & 1 & \bar{x} & y \\ 0 & 1 & 0 & \bar{x} & y \\ 0 & 1 & 1 & \bar{x} & y \\ 1 & 0 & 0 & x & y \\ 1 & 0 & 1 & x & y \\ 1 & 1 & 0 & x & y \\ 1 & 1 & 1 & x & y \end{vmatrix} = \begin{vmatrix} 0 & & & \bar{x} & y \\ 1 & & & x & y \end{vmatrix}. \qquad (10)$$

Similar to the rules for super-sticking of variables for functions with four or five variables, it is possible to represent super-sticking rules for functions of six or more variables.

In general, the configuration of the truth table of a given function, in addition to the submatrix of a complete combinatorial system with a repetition of 2-(n, b)-design, contains submatrices of an incomplete combinatorial system with a repetition of 2-($n, x/b$)-design. In this case, x – the number of blocks of an incomplete combinatorial system with repetition. The properties of an incomplete combinatorial system with a repetition of 2-(n, x/b)-design also allow the establishment of rules that ensure the effective Boolean functions minimization.

Let's single out a class of incomplete combinatorial systems with a repetition of 2-(n, $b/2$)-design, in which the number of blocks is half of the possible number of blocks of a complete combinatorial system with repetition. For 2-(n, $b/2$)-design for $n = 3$, the minimization rules are:

A. $$\begin{vmatrix} x & 0 & 0 & 1 \\ x & 0 & 1 & 1 \\ x & 1 & 0 & 1 \\ x & 1 & 1 & 1 \end{vmatrix} = \begin{vmatrix} x & & & 1 \end{vmatrix}.$$

B. $$\begin{vmatrix} x & 0 & 0 & 0 \\ x & 0 & 1 & 0 \\ x & 1 & 0 & 0 \\ x & 1 & 1 & 0 \end{vmatrix} = \begin{vmatrix} x & & & 0 \end{vmatrix}.$$

C. $$\begin{vmatrix} x & 1 & 0 & 0 \\ x & 1 & 0 & 1 \\ x & 1 & 1 & 0 \\ x & 1 & 1 & 1 \end{vmatrix} = \begin{vmatrix} x & 1 & & \end{vmatrix}.$$

D. $$\begin{vmatrix} x & 0 & 0 & 0 \\ x & 0 & 0 & 1 \\ x & 0 & 1 & 0 \\ x & 0 & 1 & 1 \end{vmatrix} = \begin{vmatrix} x & 0 & & \end{vmatrix}.$$

E. $$\begin{vmatrix} x & 0 & 1 & 0 \\ x & 0 & 1 & 1 \\ x & 1 & 0 & 0 \\ x & 1 & 0 & 1 \end{vmatrix} = \begin{vmatrix} x & 0 & 1 & \\ x & 0 & 1 & \\ x & 1 & 0 & \\ x & 1 & 0 & \end{vmatrix} = \begin{vmatrix} x & 0 & 1 & \\ x & 1 & 0 & \end{vmatrix}.$$

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

Rules like *A–F* exist for all possible 2-(n, $b/2$)-design with 2-(n, b)-design.

For other cases of a 4-bit logical function, using the 2-($n, x/b$)-design structure, the rules for minimizing a logical function can be, for example, the following:

1. $$\begin{vmatrix} x & 0 & 0 & 0 \\ x & 0 & 0 & 1 \\ x & 1 & 0 & 0 \\ x & 1 & 0 & 1 \\ x & 1 & 1 & 0 \\ x & 1 & 1 & 1 \end{vmatrix} = \begin{vmatrix} x & 0 & 0 & \\ x & 1 & 0 & \\ x & 1 & 1 & \end{vmatrix}. \qquad (11)$$

In the first matrix of the block-design (11), there is a complete combinatorial system with a repetition of 2-(3, 6/8)-design. Further minimization of the block-design (11) is possible in two ways, each of which gives the same result:

1) $$\begin{vmatrix} x & 0 & 0 & \\ x & 1 & 0 & \\ x & 1 & 1 & \end{vmatrix} = \begin{vmatrix} x & 0 & 0 & \\ x & 1 & & \end{vmatrix} = \begin{vmatrix} x & & 0 & \\ x & 1 & & \end{vmatrix},$$

2) $$\begin{vmatrix} x & 0 & 0 & \\ x & 1 & 0 & \\ x & 1 & 1 & \end{vmatrix} = \begin{vmatrix} x & & 0 & \\ x & 1 & 1 & \end{vmatrix} = \begin{vmatrix} x & & 0 & \\ x & 1 & & \end{vmatrix}.$$

In the two above variants, the first operation of sticking of variables is performed, the second operation of variable substitution is performed. As a result, for the block-design (11) let's obtain the following rule of function reduction:

$$\begin{vmatrix} x & 0 & 0 & 0 \\ x & 0 & 0 & 1 \\ x & 1 & 0 & 0 \\ x & 1 & 0 & 1 \\ x & 1 & 1 & 0 \\ x & 1 & 1 & 1 \end{vmatrix} = \begin{vmatrix} x & & 0 & \\ x & 1 & & \end{vmatrix}. \qquad (12)$$

$$2.\ \begin{vmatrix} x & 0 & 0 & 0 \\ x & 0 & 0 & 1 \\ x & 0 & 1 & 1 \\ x & 1 & 0 & 1 \\ x & 1 & 1 & 1 \end{vmatrix} = \begin{vmatrix} x & 0 & 0 & \\ x & 1 & 0 & 1 \\ x & & 1 & 1 \end{vmatrix} \qquad (13)$$

or

$$\begin{vmatrix} x & 0 & 0 & 0 \\ x & 0 & 0 & 1 \\ x & 0 & 1 & 1 \\ x & 1 & 0 & 1 \\ x & 1 & 1 & 1 \end{vmatrix} = \begin{vmatrix} x & 0 & 0 & \\ x & 0 & 1 & 1 \\ x & 1 & & 1 \end{vmatrix}. \qquad (14)$$

In the first matrix of the block-design (13), there is a complete combinatorial system with a repetition of 2-(3, 5/8)-design. Further minimization of the block-design (13) is possible in two ways, each of which gives the same result:

$$1)\ \begin{vmatrix} x & 0 & 0 & \\ x & 1 & 0 & 1 \\ x & & 1 & 1 \end{vmatrix} = \begin{vmatrix} x & 0 & 0 & \\ x & & 0 & 1 \\ x & & 1 & 1 \end{vmatrix} = \begin{vmatrix} x & 0 & 0 & \\ x & & & 1 \end{vmatrix},$$

$$2)\ \begin{vmatrix} x & 0 & 0 & \\ x & 1 & 0 & 1 \\ x & & 1 & 1 \end{vmatrix} = \begin{vmatrix} x & 0 & 0 & \\ x & 1 & & 1 \\ x & & 1 & 1 \end{vmatrix} = \begin{vmatrix} x & 0 & 0 & \\ x & 1 & & 1 \\ x & & 1 & 1 \\ x & & 0 & 1 \end{vmatrix} =$$

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

As a result, for the block-design (13) let's obtain the following rule for the function reduction:

$$\begin{vmatrix} x & 0 & 0 & 0 \\ x & 0 & 0 & 1 \\ x & 0 & 1 & 1 \\ x & 1 & 0 & 1 \\ x & 1 & 1 & 1 \end{vmatrix} = \begin{vmatrix} x & 0 & 0 & \\ x & & & 1 \end{vmatrix}. \qquad (15)$$

$$3.\ \begin{vmatrix} 0 & 0 & x & 0 \\ 0 & 0 & x & 1 \\ 0 & 1 & x & 0 \\ 1 & 0 & x & 0 \\ 1 & 0 & x & 1 \\ 1 & 1 & x & 0 \\ 1 & 1 & x & 1 \end{vmatrix} = \begin{vmatrix} 0 & 0 & x & \\ 0 & 1 & x & 0 \\ 1 & 0 & x & \\ 1 & 1 & x & \end{vmatrix} = \begin{vmatrix} 0 & 0 & x & \\ 0 & 1 & x & 0 \\ 1 & & x & \end{vmatrix}. \qquad (16)$$

In the first matrix of the block-design (16), there is a complete combinatorial system with a repetition of 2-(3, 7/8)-design. Further minimization of the block-design (16) is possible in two ways, each of which gives the same result.

$$1)\ \begin{vmatrix} 0 & 0 & x & \\ 0 & 1 & x & 0 \\ 1 & & x & \end{vmatrix} = \begin{vmatrix} 0 & 0 & x & \\ 0 & & x & 0 \\ 1 & & x & \end{vmatrix} = \begin{vmatrix} 0 & 0 & x & \\ & & x & 0 \\ 1 & & x & \end{vmatrix} = \begin{vmatrix} & 0 & x & \\ & & x & 0 \\ 1 & & x & \end{vmatrix},$$

$$2)\ \begin{vmatrix} 0 & 0 & x & \\ 0 & 1 & x & 0 \\ 1 & & x & \end{vmatrix} = \begin{vmatrix} 0 & 0 & x & \\ & 1 & x & 0 \\ 1 & & x & \end{vmatrix} = \begin{vmatrix} & 0 & x & \\ & 1 & x & 0 \\ 1 & & x & \end{vmatrix} = \begin{vmatrix} & 0 & x & \\ & & x & 0 \\ 1 & & x & \end{vmatrix}.$$

As a result, for the block-design (16) let's obtain the following rule of function reduction:

$$\begin{vmatrix} 0 & 0 & x & 0 \\ 0 & 0 & x & 1 \\ 0 & 1 & x & 0 \\ 1 & 0 & x & 0 \\ 1 & 0 & x & 1 \\ 1 & 1 & x & 0 \\ 1 & 1 & x & 1 \end{vmatrix} = \begin{vmatrix} & 0 & x & \\ & & x & 0 \\ 1 & & x & \end{vmatrix}. \qquad (17)$$

$$4.\ \begin{vmatrix} 0 & 0 & x & 0 \\ 0 & 0 & x & 1 \\ 1 & 0 & x & 1 \\ 1 & 1 & x & 1 \end{vmatrix} = \begin{vmatrix} 0 & 0 & x & \\ 1 & & x & 1 \end{vmatrix}. \qquad (18)$$

$$5.\ \begin{vmatrix} x & y & 0 & 1 \\ x & y & 1 & 0 \\ x & y & 1 & 1 \end{vmatrix} = \begin{vmatrix} x & y & 0 & 1 \\ x & y & 1 & \end{vmatrix} = \begin{vmatrix} x & y & & 1 \\ x & y & 1 & \end{vmatrix}. \qquad (19)$$

$$6.\ \begin{vmatrix} x & y & 0 & 1 \\ x & y & 1 & 0 \\ x & y & 0 & 0 \end{vmatrix} = \begin{vmatrix} x & y & 0 & 1 \\ x & y & & 0 \end{vmatrix} = \begin{vmatrix} x & y & 0 & \\ x & y & & 0 \end{vmatrix}. \qquad (20)$$

$$7.\ \begin{vmatrix} x & y & 1 & 1 \\ x & y & 1 & 0 \\ x & y & 0 & 0 \end{vmatrix} = \begin{vmatrix} x & y & 1 & 1 \\ x & y & & 0 \end{vmatrix} = \begin{vmatrix} x & y & 1 & \\ x & y & & 0 \end{vmatrix}. \qquad (21)$$

$$8.\ \begin{vmatrix} x & y & 1 & 1 \\ x & y & 0 & 1 \\ x & y & 0 & 0 \end{vmatrix} = \begin{vmatrix} x & y & 1 & 1 \\ x & y & 0 & \end{vmatrix} = \begin{vmatrix} x & y & & 1 \\ x & y & 0 & \end{vmatrix}. \qquad (22)$$

Rules for converting block-design (19)–(22) allow the establishment of new minimization rules, for example:

$$9.\ \begin{vmatrix} x & \bar{y} & 0 & 1 \\ x & \bar{y} & 1 & 0 \\ x & \bar{y} & 1 & 1 \\ x & y & 0 & 1 \\ x & y & 1 & 0 \\ x & y & 1 & 1 \end{vmatrix} = \begin{vmatrix} x & \bar{y} & & 1 \\ x & \bar{y} & 1 & \\ x & y & & 1 \\ x & y & 1 & \end{vmatrix} = \begin{vmatrix} x & & & 1 \\ x & & 1 & \end{vmatrix}. \qquad (23)$$

Rules *A*–*F*, (12), (15), (17)–(23) constitute a library of rules for the process of Boolean functions minimization as standard procedures, so applying a separate such rule to variables of a Boolean function reduces to the implementation of a single algebraic transformation.

5.3. 4-bit Boolean functions minimization. *Example* 4. To minimize the logic function $F(x_1, x_2, x_3, x_4)$ by the combinatorial method given by the following truth table (Table 1) [13].

Table 1

The truth table of a logical function $F(x_1, x_2, x_3, x_4)$

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

Let's make the perfect disjunctive normal form (PDNF) of the given function in blocks for which the function gets the value of one, that is, for the sets 0, 1, 2, 3, 6, 9, 10, 11, 13, 14, 15.

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

The first step is sticking, substituting and generalizing sticking of variables. With the many variants of minimization, obtained in the first stage, let's consider two options for minimizing the 4-bit function.

Option 1: minimization of the function using the rule of super-sticking of variables in the presence of a complete binary combinatorial system with a repetition of 2-(n, b)-design.

$$F=\left|\begin{array}{r|cccc} 0 & 0 & 0 & 0 & 0\\ 1 & 0 & 0 & 0 & 1\\ 2 & 0 & 0 & 1 & 0\\ 3 & 0 & 0 & 1 & 1\\ 6 & 0 & 1 & 1 & 0\\ 9 & 1 & 0 & 0 & 1\\ 10 & 1 & 0 & 1 & 0\\ 11 & 1 & 0 & 1 & 1\\ 13 & 1 & 1 & 0 & 1\\ 14 & 1 & 1 & 1 & 0\\ 15 & 1 & 1 & 1 & 1 \end{array}\right|=\begin{vmatrix} 0 & 0 & & \\ 0 & 1 & 1 & 0\\ 1 & & & 1\\ 1 & & 1 & \end{vmatrix}=\begin{vmatrix} 0 & 0 & & \\ 0 & & 1 & 0\\ 1 & & & 1\\ 1 & & 1 & \end{vmatrix}=$$
$$=\begin{vmatrix} 0 & 0 & & \\ & & 1 & 0\\ 1 & & & 1\\ 1 & & 1 & \end{vmatrix}=\begin{vmatrix} 0 & 0 & & \\ & & 1 & 0\\ 1 & & & 1 \end{vmatrix}. \qquad (24)$$

For blocks 0–3 of the first matrix of the block-design (24), a second super-sticking rule (4) is used, where there is a complete combinatorial system with repetition 2-(2, 4)-design; block 6 does not change; for blocks 9–15, the super-sticking rule (23) is used, where there is a complete binary combinatorial system with a repetition of 2-(3, 6/8)-design. The result of the super-sticking operation is recorded in the second matrix of the block-design (24).

Algebraic transformations of the second matrix (the result of the transformation is written in the third matrix):

– substitution of variables in the first and second blocks of the second matrix of the block-design (24):

$$\overline{x_1}\,\overline{x_2}+\overline{x_1}x_2x_3\overline{x_4}=\overline{x_1}\left(\overline{x_2}+x_2x_3\overline{x_4}\right)=$$
$$=\overline{x_1}\left(\overline{x_2}+x_3\overline{x_4}\right)=\overline{x_1}\left(\overline{x_2}+\overline{x_1}x_3\overline{x_4}\right),$$

$$\begin{matrix} 0 & 0 & & \\ 0 & 1 & 1 & 0 \end{matrix}\rightarrow\begin{matrix} 0 & 0 & & \\ 0 & & 1 & 0 \end{matrix}.$$

Algebraic transformations of the third matrix, the result of which is written in the fourth matrix:

– substitution of variables in the second and fourth blocks of the matrix of the block-design (24):

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

$$\begin{matrix} 0 & & 1 & 0\\ 1 & & 1 & \end{matrix}\rightarrow\begin{matrix} & & 1 & 0\\ 1 & & 1 & \end{matrix}.$$

Algebraic transformations of the fourth matrix, the result of which is written in the fifth matrix:

– generalized sticking of the variables of the 2nd, 3rd and 4th blocks of the 4th matrix 3 (24):

$$x_3\overline{x_4}+x_1x_4+x_1x_3=x_3\overline{x_4}+x_1x_4.$$

$$\begin{matrix} & & 1 & 0\\ 1 & & & 1\\ 1 & & 1 & \end{matrix}\rightarrow\begin{matrix} & & 1 & 0\\ 1 & & & 1 \end{matrix}.$$

As a result, let's obtain the minimum function:

$$F=\overline{x_1}\,\overline{x_2}+x_1x_4+x_3\overline{x_4}. \qquad (25)$$

Option 2: minimization of the function using the super-sticking rule for variables in the presence of an incomplete binary combinatorial system with a repetition of 2-(n, x/b)-design.

In blocks 1–5 of the first matrix of the block-design (24), let's select the left column with common zeros. The other three columns will form an incomplete combinatorial system with a repetition of 2-(3, 5/8)-design. To minimize blocks 1–5, let's use the super-sticking rule (13):

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

In blocks 6–11 of the first matrix of the block-design (24), let's select the left column with common units. The other three columns will form an incomplete combinatorial system with a repetition of 2-(3, 6/8)-design. For blocks 6–11 let's use rule (23):

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

Adding the results of minimizing 1–5 and 6–11 blocks into one matrix, let's obtain the third matrix of the block-design (24):

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

The minimization of the matrix F is analogous to the procedure for minimizing the first variant.

The second step is the verification of the obtained minimized function (25) using the original truth table (Table 1).

The minimized logic function (25) satisfies the original truth table (Table 1).

Table 2 presents the results of the $F(x_1,x_2,x_3,x_4)$ function minimization by means of an acyclic graph [13] and a combinatorial method.

Table 2

The result of $F(x_1,x_2,x_3,x_4)$ function minimization

Using an acyclic graph	Using combinatorial method
$F=\overline{x_1}\,\overline{x_2}+x_1x_4+x_1x_3\overline{x_4}+\overline{x_1}x_2x_3\overline{x_4}$	$F=\overline{x_1}\,\overline{x_2}+x_1x_4+x_3\overline{x_4}$

Considering Table 2, it is possible to see that the combinatorial method gives a function with a smaller number of input variables.

Example 5. To minimize by a combinatorial method a logical function [14]:

$$F(x_1,x_2,x_3,x_4)=(0,0,1,1,1,0,1,1,1,1,1,0,0,0,1).$$

Let's compile the truth table of a given 4-bit function from the blocks at which the function receives the value of 1, that is, for the sets: 2, 3, 4, 6, 7, 8, 9, 10, 11, 15, and minimize:

$$F=\left|\begin{array}{r|cccc} 2 & 0 & 0 & 1 & 0 \\ 3 & 0 & 0 & 1 & 1 \\ 4 & 0 & 1 & 0 & 0 \\ 6 & 0 & 1 & 1 & 0 \\ 7 & 0 & 1 & 1 & 1 \\ 8 & 1 & 0 & 0 & 0 \\ 9 & 1 & 0 & 0 & 1 \\ 10 & 1 & 0 & 1 & 0 \\ 11 & 1 & 0 & 1 & 1 \\ 15 & 1 & 1 & 1 & 1 \end{array}\right|=\left|\begin{array}{cccc} 0 & 0 & 1 & \\ 0 & 1 & & 0 \\ 1 & 0 & & \\ & 1 & 1 & 1 \end{array}\right|=\left|\begin{array}{cccc} & 0 & 1 & \\ 0 & 1 & & 0 \\ 1 & 0 & & \\ & 1 & 1 & 1 \end{array}\right|=\left|\begin{array}{cccc} & 0 & 1 & \\ 0 & 1 & & 0 \\ 1 & 0 & & \\ & & 1 & 1 \end{array}\right|.$$

To blocks 8–11 (highlighted in red) of the first matrix, a rule of super-sticking of variables is applied, where there is a combinatorial system of 2-(2, 4)-design. Simple sticking variables are highlighted in colors. Substitution (incomplete sticking) of the variables is carrying out in the last two matrices.

As a result, let's the minimum function:

$$F=x_1\overline{x_2}+\overline{x_1}x_2\overline{x_4}+\overline{x_2}x_3+x_3x_4.$$

Table 3 shows the results of $F(x_1,x_2,x_3,x_4)$ function minimization by means of parallel splitting of the conjuncterms [14] and the combinatorial method.

Table 3

The result of $F(x_1,x_2,x_3,x_4)$ function minimization

The method of parallel splitting of conjuncterms	Combinatorial method
$F=x_1\overline{x_2}+\overline{x_1}x_2\overline{x_4}+\overline{x_1}x_3+x_3x_4$	$F=x_1\overline{x_2}+\overline{x_1}x_2\overline{x_4}+\overline{x_2}x_3+x_3x_4$

Considering Table 3 it is possible see that both functions have the same parameters and undergo verification, although they differ in the composition of the variables in the third implicants. Example 5 demonstrates the less computational complexity of Boolean function minimization by combinatorial method.

Example 6. To minimize the logical function given in canonical form by combinatorial method [15]:

$$F(x_1,x_2,x_3,x_4)=(0,1,6,8,11,14,15).$$

$$F=\left|\begin{array}{r|cccc} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 6 & 0 & 1 & 1 & 0 \\ 8 & 1 & 0 & 0 & 0 \\ 11 & 1 & 0 & 1 & 1 \\ 14 & 1 & 1 & 1 & 0 \\ 15 & 1 & 1 & 1 & 1 \end{array}\right|=\left|\begin{array}{cccc} 0 & 0 & 0 & 1 \\ & 0 & 0 & 0 \\ & 1 & 1 & 0 \\ 1 & & 1 & 1 \end{array}\right|=\left|\begin{array}{cccc} 0 & 0 & 0 & \\ & 0 & 0 & 0 \\ & 1 & 1 & 0 \\ 1 & & 1 & 1 \end{array}\right|.$$

The results of function minimization with the help of parallel splitting of conjuncterms [15] and combinatorial method are presented in Table 4.

Table 4

The result of $F(x_1,x_2,x_3,x_4)$ function minimization

The method of parallel splitting of conjuncterms	Combinatorial method
$\{(000\sim),(\sim 000),(\sim 110),(1\sim 11)\}$	$\left\|\begin{array}{cccc} 0 & 0 & 0 & \\ & 0 & 0 & 0 \\ & 1 & 1 & 0 \\ 1 & & 1 & 1 \end{array}\right\|$

In Table 4 it can be seen that the results of minimization of the two compared methods are the same.

Coincidence and the minimization indicator $k_\theta/k_l=4/12$ is coincided, where k_θ – the number of simple implicants; k_l – the number of input variables.

However, the computational complexity of minimizing the Boolean function by a combinatorial method is less.

Example 7. To minimize the system of 4-bit Boolean functions f_1,f_2,f_3 [14] by combinatorial method:

$$\begin{cases} f_1=2,5,6,13,14, \\ f_2=5,7,13,14, \\ f_3=2,6,7,13,15. \end{cases}$$

Let's compile the truth table of a given system of 4-bit functions from blocks for which the function gets the value of one (Table 5).

ISSN 2226-3780

Table 5

The truth table of a system of Boolean functions f_1, f_2, f_3

	x_1	x_2	x_3	x_4	f_1	f_2	f_3
2	0	0	1	0	**1**	**0**	**1**
5	0	1	0	1	**1**	**1**	**0**
6	0	1	1	0	**1**	**0**	**1**
7	0	1	1	1	**0**	**1**	**1**
13	1	1	0	1	**1**	**1**	**1**
14	1	1	1	0	**1**	**1**	**0**
15	1	1	1	1	**0**	**0**	**1**

There are two approaches to minimizing the system of Boolean functions from n variables:

1) minimization is carried out separately for each function;

2) joint minimization of the system, when the method of the minimal system uses the general conjunctures of individual functions.

Eliminating redundant conjuncts in a separate function does not guarantee the elimination of redundancy in the system itself. On the other hand, the joint minimization of the system may not always be better. Therefore, for a number of systems of functions, it is necessary to apply both methods of minimization. The joint minimization of the system is more cumbersome than in the first method.

For joint minimization, let's combine all the different conjuncterms of individual functions into the function Y of system conjuncterms:

$$Y = 0010_{(1,3)} + 0101_{(1,2)} + 0110_{(1,3)} + 0111_{(2,3)} + \\ + 1101_{(1,2,3)} + 1110_{(1,2)} + 1111_{(3)}.$$

A system conjuncterm is called a minterm of a Boolean function with indices that show which functions it belongs to [14]. Among the systemic conjuncterms, the elements are identical-they have identical indices, and are identical – that they have different indices, but whose cross-section is not empty. For example, $(101)_{2,4}$ and $(111)_{2,4}$ form the identical element $(1\text{-}1)_{2,4}$, and $(101)_{2,4}$ and $(001)_4$ form the identity element $(-01)_4$ [14].

The function Y is represented by a truth table.

To jointly minimize the system, let's apply the following rules:

– the sticking of variables in the system conjuncterms of the function Y is carried out only for those conjuncterms that have at least one common index;

– the result of sticking the conjuncterms is assigned a set of indices, which is the intersection of the output sets of indices of the stick conjuncterms;

– if the conjuncterms do not have common indices, no sticking takes place;

– identical conjuncterms are stick together with other identical conjunctures;

– the same conjuncts are stick together with other non-identical conjunctures.

After the sticking operation, the conjuncts are identical in the following table for further minimization, except for the case when the indices of the sticking result coincide with the indices of one of the non-identical conjunctures. The absorption of one conjuncterm by another is carried out only if the sets of indices of the two conjuncterms coincide.

$$Y = \left|\begin{array}{cccccc} 0 & 0 & 1 & 0 & (1,3) \\ 0 & 1 & 1 & 0 & (1,3) \\ 0 & 1 & 0 & 1 & (1,2) \\ 0 & 1 & 1 & 1 & (2,3) \\ 1 & 1 & 0 & 1 & (1,2,3) \\ 1 & 1 & 1 & 0 & (1,2) \\ 1 & 1 & 1 & 1 & (3) \end{array}\right| = \left|\begin{array}{ccccc} 0 & & 1 & 0 & (1,3) \\ 0 & 1 & & 1 & (2) \\ & 1 & 0 & 1 & (1,2) \\ 0 & 1 & 0 & 1 & (1,2) \\ 1 & 1 & 1 & 0 & (1,2) \\ & 1 & 1 & 1 & (3) \\ 0 & 1 & 1 & 1 & (2,3) \\ 1 & 1 & 0 & 1 & (1,2,3) \\ 1 & 1 & & 1 & (3) \end{array}\right| = \left|\begin{array}{ccccc} 0 & & 1 & 0 & (1,3) \\ 0 & 1 & & 1 & (2) \\ & 1 & 0 & 1 & (1,2) \\ 1 & 1 & 1 & 0 & (1,2) \\ & 1 & 1 & 1 & (3) \\ 0 & 1 & 1 & 1 & (2,3) \\ 1 & 1 & 0 & 1 & (1,2,3) \\ 1 & 1 & & 1 & (3) \end{array}\right|.$$

Conversion of the first matrix:

– sticking of identical conjuncterms $0010_{(1,3)}$ and $0110_{(1,3)}$ (highlighted in red) the result of sticking – $0{\sim}10_{(1,3)}$ is transferred to the second matrix; after sticking the conjuncterm $0010_{(1,3)}$ and $0110_{(1,3)}$ in other operations sticking the first matrix is not involved;

– sticking of non-identical conjuncterms $0101_{(1,2)}$ and $0111_{(2,3)}$; the result of sticking – $01{\sim}1_{(2)}$ is transferred to the second matrix; since the conjuncterms $0101_{(1,2)}$ and $0111_{(2,3)}$ are not identical, they can participate in other operations of sticking the first matrix; conjuncterms $0101_{(1,2)}$ and $0111_{(2,3)}$ are transferred to the second matrix for further minimization;

– sticking of non-identical conjuncterms $0101_{(1,2)}$ and $1101_{(1,2,3)}$; the result of sticking – ${\sim}101_{(1,2)}$ is transferred to the second matrix; since the conjuncts are not identical, they can participate in other operations of sticking the first matrix; conjuncterms $0101_{(1,2)}$ and $1101_{(1,2,3)}$ are transferred to the second matrix for further minimization;

– the identical conjuncterm $1110_{(1,2)}$ isn't stick together with one conjuncterm, transferred to the second matrix for further minimization;

– sticking of non-identical conjuncterms $0111_{(2,3)}$ and $1111_{(3)}$; the result of sticking – ${\sim}111_{(3)}$ is transferred to the second matrix; since the result of the sticking – ${\sim}111_{(3)}$ and the conjuncterm $1111_{(3)}$ have the same indices, the conjuncterm $1111_{(3)}$ is not transferred to the second matrix; the indexes of the conjuncterm $1101_{(1,2,3)}$ do not coincide with the indexes of the sticking result, so this conjuncterm is transferred to the second matrix for further minimization;

– sticking of non-identical conjuncterms $1101_{(1,2,3)}$ and $1111_{(3)}$; the result of sticking – $11{\sim}1_{(3)}$ is transferred to the second matrix; since the result of the sticking – $11{\sim}1_{(3)}$ and the conjuncterm – $1111_{(3)}$ have the same indices, the conjuncterm – $1111_{(3)}$ is not transferred to the second matrix; the indexes of the conjuncterm $1101_{(1,2,3)}$ do not coincide with the indexes of the sticking result, so this conjuncterm is transferred to the second matrix for further minimization.

Transformation of the second matrix: absorption of identical conjuncterms ${\sim}101_{(1,2)}$ and $0101_{(1,2)}$ (highlighted in blue); result of absorption – ${\sim}101_{(1,2)}$ is transferred to the third matrix.

The third matrix represents the dead-end DNF of the function Y. Next, the problem of finding the minimal DNF of the function Y is solved on the basis of the cover table of B. Rytsar [14] (Fig. 1), in which it is necessary to remove all the extra simple implicants.

The elements of the minimum coverage, which are minimize the function Y by the method of joint minimization of the system, are selected by color in Table 1:

$$Y=\overline{x_1}x_3\overline{x_4}(1,3)+x_2\overline{x_3}x_4(1,2)+\overline{x_1}x_2x_3x_4(2,3)+$$
$$+x_1x_2x_4(3)+x_1x_2x_3\overline{x_4}(1,2). \qquad (26)$$

The result of minimizing the function of systemic conjuncterms (26) by combinatorial method coincides with the result of minimization by the method of parallel splitting of conjuncterms [14].

Since in order to minimize the function Y by the combinatorial method in Example 6, the sticking operation is used for identical conjuncterms and is not applied between identical and non-identical conjuncterms, this reduces the number of unnecessary simple implicants and the size of the cover table (Fig. 1).

	$01\sim1_{(2)}$		$01\sim1_{(2)}$	$\mathbf{11\sim1_{(3)}}$		$\mathbf{11\sim1_{(3)}}$
$\mathbf{0\sim10_{(1,3)}}$		$\mathbf{0\sim10_{(1,3)}}$				
	$\mathbf{\sim101_{(1,2)}}$		$\sim111_{(3)}$	$\mathbf{\sim101_{(1,2)}}$		$\sim111_{(3)}$
$0010_{(1,3)}$	$0101_{(1,2)}$	$0110_{(1,3)}$	$\mathbf{0111_{(2,3)}}$	$1101_{(1,2,3)}$	$\mathbf{1110_{(1,2)}}$	$1111_{(3)}$

Fig. 1. $F(x_1,x_2,x_3,x_4)$ function cover table

After the distribution of systemic conjuncterms of the function (26), let's obtain a minimized system of Boolean functions:

$$\begin{cases} f_1=\overline{x_1}x_3\overline{x_4}+x_2\overline{x_3}x_4+x_1x_2x_3\overline{x_4}, \\ f_2=x_2\overline{x_3}x_4+\overline{x_1}x_2x_3x_4+x_1x_2x_3\overline{x_4}, \\ f_3=\overline{x_1}x_3\overline{x_4}+\overline{x_1}x_2x_3x_4+x_1x_2x_3. \end{cases}$$

Example 8. To minimize a logical function by a combinatorial method [16]:

$$F(x_1,x_2,x_3,x_4)=(1,1,1,1,1,1,0,0,1,1,1,1,0,0,0,0).$$

Let's compile the truth table of a given 4-bit function from the blocks for which the function gets the value of 1, that is, for the sets: 0, 1, 2, 3, 4, 5, 8, 9, 10, 11, and minimize:

$$F=\left|\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 1 & 0 \\ 3 & 0 & 0 & 1 & 1 \\ 4 & 0 & 1 & 0 & 0 \\ 5 & 0 & 1 & 0 & 1 \\ 8 & 1 & 0 & 0 & 0 \\ 9 & 1 & 0 & 0 & 1 \\ 10 & 1 & 0 & 1 & 0 \\ 11 & 1 & 0 & 1 & 1 \end{array}\right|=\left|\begin{array}{cccc} & 0 & & \\ 0 & 1 & 0 & \end{array}\right|=\left|\begin{array}{cccc} & 0 & & \\ 0 & & 0 & \end{array}\right|.$$

To units 0–3 and 8–11 (highlighted in red) of the first matrix, a rule of super-sticking of variables is applied, where there is a combinatorial system of 2-(3, 8)-design. Simple sticking of variables is highlighted in black. A substitution (incomplete sticking) of the variables is carrying out in the last matrix.

As a result, let's obtain the minimum function:

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

The result of minimization by combinatorial method coincides with the result of minimization obtained by the self-reduction cycle method [16]. The method of self-reducing cycles to minimize a given function uses four lowering cycles and a cover table to remove unnecessary implicants. The combinatorial method minimizes the given function for three-dimensional transformations. Since the self-reducing cycle method uses a complete combinatorial system with a repetition of 2-(n, b)-design [16] to minimize the Boolean function, but does not use an incomplete combinatorial system with the repetition of 2-(n, x/b)-design, this method is attributed to a partial minimization by a combinatorial method.

5.4. 5-bit Boolean functions minimization. *Example* 9. To minimize the logic function $F(x_1,x_2,x_3,x_4,x_5)$ by the combinatorial method given by the following truth table (Table 6) [17, 18].

Table 6

The truth table of a logical function $F(x_1,x_2,x_3,x_4,x_5)$

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

Using Table 6, let's compose the PDNF of the given 5-bit function from the blocks for which the function receives the value of 1, that is, for the sets 1, 4, 5, 7, 9, 11, 12, 13, 16, 17, 18, 22, 28, 31:

$$F(x_1,x_2,x_3,x_4,x_5)=\overline{x_1}\,\overline{x_2}\,\overline{x_3}\,\overline{x_4}x_5+\overline{x_1}\,\overline{x_2}x_3\overline{x_4}\,\overline{x_5}+$$
$$+\overline{x_1}\,\overline{x_2}x_3\overline{x_4}x_5+\overline{x_1}\,\overline{x_2}x_3x_4x_5+\overline{x_1}x_2\overline{x_3}\,\overline{x_4}x_5+$$
$$+\overline{x_1}x_2\overline{x_3}x_4x_5+\overline{x_1}x_2x_3\overline{x_4}\,\overline{x_5}+\overline{x_1}x_2x_3\overline{x_4}x_5+$$
$$+x_1\overline{x_2}\,\overline{x_3}\,\overline{x_4}\,\overline{x_5}+x_1\overline{x_2}\,\overline{x_3}\,\overline{x_4}x_5+x_1\overline{x_2}\,\overline{x_3}x_4\overline{x_5}+$$
$$+x_1\overline{x_2}x_3\overline{x_4}\,\overline{x_5}+x_1x_2x_3\overline{x_4}\,\overline{x_5}+x_1x_2x_3x_4x_5. \qquad (27)$$

Recall that the value of «–» of the function F means an arbitrary state indicating that such set of input variables is not expected and the value of the function can be arbitrary – zero or one in the process of minimization.

Let's complete the definition of the function by substituting the value of the «–» function by one. After this substitution, the truth table (Table 6) takes on the following form (Table 7).

ISSN 2226-3780

Table 7

Table of truth of logic function $F(x_1,x_2,x_3,x_4,x_5)$ after changing the value of the «–» function to one

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

Using Table 7, it is necessary to write the PDNP of the 5-bit function from the blocks for which the function gets the value of 1, that is, for the sets 1, 2, 3, 4, 5, 7, 9, 11, 12, 13, 14, 15, 16, 17 , 18, 20, 22, 26, 28, 30, 31:

$$\begin{aligned}F(x_1,x_2,x_3,x_4,x_5)=&\overline{x_1}\,\overline{x_2}\,\overline{x_3}\,\overline{x_4}x_5+\overline{x_1}\,\overline{x_2}\,\overline{x_3}x_4\overline{x_5}+\\&+\overline{x_1}\,\overline{x_2}\,\overline{x_3}x_4x_5+\overline{x_1}\,\overline{x_2}x_3\overline{x_4}\,\overline{x_5}+\overline{x_1}\,\overline{x_2}x_3\overline{x_4}x_5+\\&+\overline{x_1}\,\overline{x_2}x_3x_4x_5+\overline{x_1}x_2\overline{x_3}\,\overline{x_4}x_5+\overline{x_1}x_2\overline{x_3}x_4x_5+\\&+\overline{x_1}x_2x_3\overline{x_4}\,\overline{x_5}+\overline{x_1}x_2x_3\overline{x_4}x_5+\overline{x_1}x_2x_3x_4\overline{x_5}+\\&+\overline{x_1}x_2x_3x_4x_5+x_1\overline{x_2}\,\overline{x_3}\,\overline{x_4}\,\overline{x_5}+x_1\overline{x_2}\,\overline{x_3}\,\overline{x_4}x_5+\\&+x_1\overline{x_2}\,\overline{x_3}x_4\overline{x_5}+x_1\overline{x_2}x_3\overline{x_4}\,\overline{x_5}+x_1\overline{x_2}x_3x_4\overline{x_5}+\\&+x_1x_2\overline{x_3}x_4\overline{x_5}+x_1x_2x_3\overline{x_4}\,\overline{x_5}+x_1x_2x_3x_4\overline{x_5}+\\&+x_1x_2x_3x_4x_5.\end{aligned} \tag{28}$$

Table 8

The truth table of a perfect disjunctive normal form $F(x_1,x_2,x_3,x_4,x_5)$ which blocks receive the value of a unit

No.	x_1	x_2	x_3	x_4	x_5	F	No.	x_1	x_2	x_3	x_4	x_5	F
1	0	0	0	0	1	**1**	12	0	1	1	1	1	**1**
2	0	0	0	1	0	**1**	13	1	0	0	0	0	**1**
3	0	0	0	1	1	**1**	14	1	0	0	0	1	**1**
4	0	0	1	0	0	**1**	15	1	0	0	1	0	**1**
5	0	0	1	0	1	**1**	16	1	0	1	0	0	**1**
6	0	0	1	1	1	**1**	17	1	0	1	1	0	**1**
7	0	1	0	0	1	**1**	18	1	1	0	1	0	**1**
8	0	1	0	1	1	**1**	19	1	1	1	0	0	**1**
9	0	1	1	0	0	**1**	20	1	1	1	1	0	**1**
10	0	1	1	0	1	**1**	21	1	1	1	1	1	**1**
11	0	1	1	1	0	**1**							

At the first stage, the constituants are stick together and the variables are substituted. Using a partial binary combinatorial system with a repetition of 2-$(n, x/b)$-design for a 5-bit logical function, blocks 1–12 (Table 8) allocate a left column with common zeros. The other four columns will form an incomplete combinatorial system with a repetition of 2-(4, 12/16)-design.

The process of minimizing blocks 1–12 (Table 8) is possible in two ways.

The first option: sticking, substituting and sticking of variables.

$$\left|\begin{array}{ccccc}\overline{x}&\overline{y}&0&0&1\\\overline{x}&\overline{y}&0&1&0\\\overline{x}&\overline{y}&0&1&1\\\overline{x}&\overline{y}&1&0&0\\\overline{x}&\overline{y}&1&0&1\\\overline{x}&\overline{y}&1&1&1\\\overline{x}&y&0&0&1\\\overline{x}&y&0&1&1\\\overline{x}&y&1&0&0\\\overline{x}&y&1&0&1\\\overline{x}&y&1&1&0\\\overline{x}&y&1&1&1\end{array}\right|=\left|\begin{array}{ccccc}\overline{x}&\overline{y}&0&1&0\\\overline{x}&&0&0&1\\\overline{x}&&0&1&1\\\overline{x}&&1&0&0\\\overline{x}&&1&0&1\\\overline{x}&y&1&1&0\\\overline{x}&&1&1&1\end{array}\right|=\left|\begin{array}{ccccc}\overline{x}&\overline{y}&0&1&0\\\overline{x}&&0&&1\\\overline{x}&&1&0&\\\overline{x}&y&1&1&0\\\overline{x}&&1&1&1\end{array}\right|=$$

$$=\left|\begin{array}{ccccc}\overline{x}&\overline{y}&0&1&\\\overline{x}&&0&&1\\\overline{x}&&1&0&\\\overline{x}&y&1&1&\\\overline{x}&&1&1&1\end{array}\right|=\left|\begin{array}{ccccc}\overline{x}&\overline{y}&0&1&\\\overline{x}&&0&&1\\\overline{x}&&1&0&\\\overline{x}&y&1&&\\\overline{x}&&1&&1\end{array}\right|=\left|\begin{array}{ccccc}\overline{x}&\overline{y}&0&1&\\\overline{x}&&1&0&\\\overline{x}&y&1&&\\\overline{x}&&&&1\end{array}\right|.$$

The second option: sticking, substituting, sticking, substituting, sticking and substituting of variables.

$$\left|\begin{array}{ccccc}\overline{x}&\overline{y}&0&0&1\\\overline{x}&\overline{y}&0&1&0\\\overline{x}&\overline{y}&0&1&1\\\overline{x}&\overline{y}&1&0&0\\\overline{x}&\overline{y}&1&0&1\\\overline{x}&\overline{y}&1&1&1\\\overline{x}&y&0&0&1\\\overline{x}&y&0&1&1\\\overline{x}&y&1&0&0\\\overline{x}&y&1&0&1\\\overline{x}&y&1&1&0\\\overline{x}&y&1&1&1\end{array}\right|=\left|\begin{array}{ccccc}\overline{x}&\overline{y}&0&0&1\\\overline{x}&\overline{y}&0&1&\\\overline{x}&\overline{y}&1&0&\\\overline{x}&\overline{y}&1&1&1\\\overline{x}&y&0&&1\\\overline{x}&y&1&0&\\\overline{x}&y&1&1&\end{array}\right|=\left|\begin{array}{ccccc}\overline{x}&\overline{y}&0&0&1\\\overline{x}&\overline{y}&0&1&\\\overline{x}&&1&0&\\\overline{x}&\overline{y}&1&1&1\\\overline{x}&y&0&&1\\\overline{x}&y&1&1&\end{array}\right|=$$

$$=\left|\begin{array}{ccccc}\overline{x}&\overline{y}&0&&1\\\overline{x}&\overline{y}&0&1&\\\overline{x}&&1&0&\\\overline{x}&&1&1&1\\\overline{x}&y&0&&1\\\overline{x}&y&1&1&\end{array}\right|=\left|\begin{array}{ccccc}\overline{x}&\overline{y}&0&1&\\\overline{x}&&1&0&\\\overline{x}&&1&1&1\\\overline{x}&&0&&1\\\overline{x}&y&1&1&\end{array}\right|=\left|\begin{array}{ccccc}\overline{x}&\overline{y}&0&1&\\\overline{x}&&1&0&\\\overline{x}&&1&&1\\\overline{x}&&0&&1\\\overline{x}&y&1&1&\end{array}\right|=$$

$$=\left|\begin{array}{ccccc}\overline{x}&\overline{y}&0&1&\\\overline{x}&&1&0&\\\overline{x}&&&&1\\\overline{x}&y&1&1&\end{array}\right|=\left|\begin{array}{ccccc}\overline{x}&\overline{y}&0&1&\\\overline{x}&&1&0&\\\overline{x}&&&&1\\\overline{x}&y&1&&\end{array}\right|.$$

In blocks 13–21 (Table 8), the left column with common units is selected. The other four columns will form an incomplete combinatorial system with a repetition of 2-(4, 9/16)-design.

The process of minimizing blocks 13–21 (Table 8) is possible in two ways.

The first option: sticking, sticking and substituting of variables.

$$\begin{vmatrix} x & \bar{y} & 0 & 0 & 0 \\ x & \bar{y} & 0 & 0 & 1 \\ x & \bar{y} & 0 & 1 & 0 \\ x & \bar{y} & 1 & 0 & 0 \\ x & \bar{y} & 1 & 1 & 0 \\ x & y & 0 & 1 & 0 \\ x & y & 1 & 0 & 0 \\ x & y & 1 & 1 & 0 \\ x & y & 1 & 1 & 1 \end{vmatrix} = \begin{vmatrix} x & \bar{y} & 0 & 0 & \\ x & & 0 & 1 & 0 \\ x & & 1 & 0 & 0 \\ x & & 1 & 1 & 0 \\ x & y & 1 & 1 & 1 \end{vmatrix} = \begin{vmatrix} x & \bar{y} & 0 & 0 & \\ x & & 1 & 0 & 0 \\ x & & & 1 & 0 \\ x & y & 1 & 1 & 1 \end{vmatrix} =$$

$$= \begin{vmatrix} x & \bar{y} & 0 & 0 & \\ x & & 1 & & 0 \\ x & & & 1 & 0 \\ x & y & 1 & 1 & \end{vmatrix}.$$

The second option: sticking, sticking and substituting of variables.

$$\begin{vmatrix} x & \bar{y} & 0 & 0 & 0 \\ x & \bar{y} & 0 & 0 & 1 \\ x & \bar{y} & 0 & 1 & 0 \\ x & \bar{y} & 1 & 0 & 0 \\ x & \bar{y} & 1 & 1 & 0 \\ x & y & 0 & 1 & 0 \\ x & y & 1 & 0 & 0 \\ x & y & 1 & 1 & 0 \\ x & y & 1 & 1 & 1 \end{vmatrix} = \begin{vmatrix} x & \bar{y} & 0 & 0 & \\ x & \bar{y} & 0 & 1 & 0 \\ x & \bar{y} & 1 & & 0 \\ x & y & 0 & 1 & 0 \\ x & y & 1 & & 0 \\ x & y & 1 & 1 & 1 \end{vmatrix} = \begin{vmatrix} x & \bar{y} & 0 & 0 & \\ x & & 0 & 1 & 0 \\ x & & 1 & & 0 \\ x & y & 1 & 1 & 1 \end{vmatrix} =$$

$$= \begin{vmatrix} x & \bar{y} & 0 & 0 & \\ x & & & 1 & 0 \\ x & & 1 & & 0 \\ x & y & 1 & 1 & \end{vmatrix}.$$

In the second step, the substitution and generalized sticking of the variables is performed.

To further minimize PDNF $F(x_1,x_2,x_3,x_4,x_5)$, let's combine the results of minimization of 1–12 and 13–21 columns of Table 8 into one matrix:

$$\begin{vmatrix} \bar{x} & \bar{y} & 0 & 1 & \\ \bar{x} & & 1 & 0 & \\ \bar{x} & & & & 1 \\ \bar{x} & y & 1 & & \\ x & \bar{y} & 0 & 0 & \\ x & & & 1 & 0 \\ x & & 1 & & 0 \\ x & y & 1 & 1 & \end{vmatrix} = \begin{vmatrix} \bar{x} & \bar{y} & 0 & 1 & \\ \bar{x} & & 1 & 0 & \\ \bar{x} & & & & 1 \\ \bar{x} & y & 1 & & \\ x & \bar{y} & 0 & 0 & \\ x & & & 1 & 0 \\ x & & 1 & & 0 \\ & y & 1 & 1 & \end{vmatrix} = \begin{vmatrix} \bar{x} & \bar{y} & 0 & 1 & \\ \bar{x} & & 1 & 0 & \\ \bar{x} & & & & 1 \\ x & \bar{y} & 0 & 0 & \\ x & & & 1 & 0 \\ x & & 1 & & 0 \\ & y & 1 & 1 & \end{vmatrix} = \begin{vmatrix} \bar{x} & \bar{y} & 0 & 1 & \\ \bar{x} & & 1 & 0 & \\ \bar{x} & & & & 1 \\ x & \bar{y} & 0 & 0 & \\ x & & & 1 & 0 \\ & & 1 & 0 & 0 \\ x & & 1 & & 0 \\ & y & 1 & 1 & \end{vmatrix} = \begin{vmatrix} \bar{x} & \bar{y} & 0 & 1 & \\ \bar{x} & & & & 1 \\ x & \bar{y} & 0 & 0 & \\ x & & & 1 & 0 \\ & & 1 & 0 & 0 \\ & y & 1 & 1 & \end{vmatrix}.$$

In the first combined matrix, a variable substitution is performed, in 2–4 joint matrices carrying out a generalized sticking of the variables.

Algebraic transformations of the second combined matrix, the result of which is recorded in the third joint matrix, define a generalized sticking of variables for 2, 4 and 8 blocks.

$$\bar{x}_1 x_3 \bar{x}_4 + \bar{x}_1 x_2 x_3 + x_2 x_3 x_4 =$$
$$= \bar{x}_1 x_3 \bar{x}_4 + x_2 x_3 x_4 + \bar{x}_1 x_2 x_3 = \bar{x}_1 x_3 \bar{x}_4 + x_2 x_3 x_4.$$

Algebraic transformations of the third combined matrix, the result of which is written in the fourth joint matrix, define the generalized sticking of the variables for 2 and 7 blocks:

$$\bar{x}_1 x_3 \bar{x}_4 + x_1 x_3 \bar{x}_5 = \bar{x}_1 x_3 \bar{x}_4 + x_1 x_3 \bar{x}_5 + x_3 x_3 \bar{x}_4 \bar{x}_5 =$$
$$= \bar{x}_1 x_3 \bar{x}_4 + x_1 x_3 \bar{x}_5 + x_3 \bar{x}_4 \bar{x}_5.$$

Algebraic transformations of the fourth joint matrix, the result of which is written in the fifth joint matrix, are defined as:

– generalized sticking of variables for 5, 6 and 7 blocks:

$$x_3 \bar{x}_4 \bar{x}_5 + x_1 x_4 \bar{x}_5 = x_3 \bar{x}_4 \bar{x}_5 + x_1 x_4 \bar{x}_5 + x_1 x_3 \bar{x}_5 \bar{x}_5 =$$
$$= x_3 \bar{x}_4 \bar{x}_5 + x_1 x_4 \bar{x}_5 + x_1 x_3 \bar{x}_5 = x_3 \bar{x}_4 \bar{x}_5 + x_1 x_4 \bar{x}_5;$$

– generalized sticking of variables for 2, 6 and 7 blocks:

$$\bar{x}_1 x_5 + x_3 \bar{x}_4 \bar{x}_5 + \bar{x}_1 x_3 \bar{x}_4 = \bar{x}_1 x_5 + x_3 \bar{x}_4 \bar{x}_5.$$

Attempts to further apply algebraic transformation operations do not give a result, which leads to a deadlock DNF of the function $F(x_1,x_2,x_3,x_4,x_5)$, which is presented in Table 8.

$$F(x_1,x_2,x_3,x_4,x_5) = \bar{x}_1 x_5 + x_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 +$$
$$+ x_1 x_4 \bar{x}_5 + \bar{x}_1 \bar{x}_2 \bar{x}_3 x_4 + x_2 x_3 x_4 + x_3 \bar{x}_4 \bar{x}_5.$$

The third step involves testing each simple implant in PDNF for redundancy to remove it and verifying the resulting function using a truth table (Table 8).

Further, the problem of finding the minimal DNF is solved on the basis of the coverage table (Table 9). In general, in order to obtain the minimum DNF it is necessary to remove all superfluous simple implicants from the dead-end DNF.

In the columns of Table 9 there are simple implicants of the reduced DNF function represented by the fifth joint matrix. The rows of Table 9 represent the constituents of the unit of the PDNF function, which is presented in Table 7.

A simple implicant absorbs a certain constituent unit when it has its own part. The corresponding cell of Table 9 at the intersection of the column (with the simple implicant under consideration) and the line (with constituent unit) is indicated by a circle ● of black color.

Table 9

$F(x_1,x_2,x_3,x_4,x_5)$ function cover eable

Constituants	$\bar{x}_1 x_5$	$x_1\bar{x}_2\bar{x}_3\bar{x}_4$	$x_1x_4\bar{x}_5$	$\bar{x}_1\bar{x}_2\bar{x}_3x_4$	$x_2x_3x_4$	$x_3\bar{x}_4\bar{x}_5$
00001	●	–	–	–	–	–
00010	–	–	–	●	–	–
00011	●	–	–	–	–	–
00100	–	–	–	–	–	●
00101	●	–	–	–	–	–
00111	●	–	–	–	–	–
01001	●	–	–	–	–	–
01011	●	–	–	–	–	–
01100	–	–	–	–	–	●
01101	●	–	–	–	–	–
01110	–	–	–	–	●	–
01111	●	–	–	–	●	–
10000	–	●	–	–	–	–
10001	–	●	–	–	–	–
10010	–	–	●	–	–	–
10100	–	–	–	–	–	●
10110	–	–	●	–	–	–
11010	–	–	●	–	–	–
11100	–	–	–	–	–	●
11110	–	–	●	–	●	–
11111	–	–	–	–	●	–

Considering Table 9 it can be seen that there are no excess implicants, and, consequently, Table 9 represents the minimum DNF of the function (28), presented in Table 7.

$$F(x_1,x_2,x_3,x_4,x_5)=\bar{x}_1x_5+x_1\bar{x}_2\bar{x}_3\bar{x}_4+ \\ +x_1x_4\bar{x}_5+\bar{x}_1\bar{x}_2\bar{x}_3x_4+x_2x_3x_4+x_3\bar{x}_4\bar{x}_5. \qquad (29)$$

The truth table (Table 7) helps to make minimization more convenient. It should be noted that the original logical function (28) is represented by a truth table (Table 6), in which there are sets of unpredictable variables. The value of the function F for such sets is affected by «–» and means its arbitrary state.

In this regard, the search for the minimum DNF function, represented by the original truth table (Table 6), is solved using the coverage table (Table 9), removing sets of unpredictable variables from its rows. After that, the table takes on the following form (Table 10).

Considering Table 10 it is possible to see that the implicant $\bar{x}_1\bar{x}_2\bar{x}_3x_4$ is an excess, which is removed from the expression of the function (29):

$$F(x_1,x_2,x_3,x_4,x_5)= \\ =\bar{x}_1x_5+x_1\bar{x}_2\bar{x}_3\bar{x}_4+x_1x_4\bar{x}_5+x_2x_3x_4+x_3\bar{x}_4\bar{x}_5. \qquad (30)$$

Expression (30) represents the dead-end and minimum DNF of the initial function (27), presented in Table 6.

Table 11 presents the results of minimizing the function by the method of «symmetric charts» [17, 18] and combinatorial method.

Table 10

Function $F(x_1,x_2,x_3,x_4,x_5)$ cover table with remote sets of unpredictable variables

Constituants	$\bar{x}_1 x_5$	$x_1\bar{x}_2\bar{x}_3\bar{x}_4$	$x_1x_4\bar{x}_5$	$\bar{x}_1\bar{x}_2\bar{x}_3x_4$	$x_2x_3x_4$	$x_3\bar{x}_4\bar{x}_5$
00001	●	–	–	–	–	–
00100	–	–	–	–	–	●
00101	●	–	–	–	–	–
00111	●	–	–	–	–	–
01001	●	–	–	–	–	–
01011	●	–	–	–	–	–
01100	–	–	–	–	–	●
01101	●	–	–	–	–	–
10000	–	●	–	–	–	–
10001	–	●	–	–	–	–
10010	–	–	●	–	–	–
10110	–	–	●	–	–	–
11100	–	–	–	–	–	●
11111	–	–	–	–	●	–

Table 11

The result of $F(x_1,x_2,x_3,x_4,x_5)$ functionminimization

«Symmetric maps» method
$F(x_1,x_2,x_3,x_4,x_5)=\bar{x}_1x_5+x_1\bar{x}_2\bar{x}_3\bar{x}_4+ +x_1\bar{x}_2\bar{x}_5+x_2x_3x_4+x_3\bar{x}_4\bar{x}_5.$
$F(x_1,x_2,x_3,x_4,x_5)=\bar{x}_1x_5+\bar{x}_2\bar{x}_3\bar{x}_4x_5+ +x_1\bar{x}_2\bar{x}_5+x_2x_3x_4+x_3\bar{x}_4\bar{x}_5.$
Combinatorial method
$F(x_1,x_2,x_3,x_4,x_5)=\bar{x}_1x_5+x_1\bar{x}_2\bar{x}_3\bar{x}_4+ +x_1x_4\bar{x}_5+x_2x_3x_4+x_3\bar{x}_4\bar{x}_5.$

The main difference between the minimal functions (Table 11) is the third implicant $x_1\bar{x}_2\bar{x}_5$. For the minimization function, the implicant method requires two inverters to support their functionality, whereas for a function minimized by a combinatorial method (implicant $x_1x_4\bar{x}_5$), one is needed. For the hardware implementation of the function (30), one inverter will need less in this case, if one chooses, for example, CMOS (complementary metal-oxide-semiconductor structure) technology.

The minimized logic function (30) satisfies a given truth table (Table 6).

5.5. 6-bit Boolean functions minimization. *Example* 10. To minimize a logical function by a combinatorial method [19]:

$$F(x_1,x_2,x_3,x_4,x_5,x_6)= \\ =(0,1,0,1,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1, \\ 1,0,0,1,1,0,0,1,1,0,0,1,1,1,1,1,1,1,1,0,0,1,1,0,0,1,1,1,1,1,1, \\ 1,1,1,1,1).$$

Let's compile the truth table of a given 6-bit function from blocks for which the function gets the value of 1, that is, for the sets: 1, 3, 10, 11, 12, 13, 26, 27, 28, 29, 32, 33, 36, 37, 40, 41, 42, 43, 44, 45, 46, 47, 50, 51, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63.

$$F=\left|\begin{array}{r|cccccc}
1 & 0&0&0&0&0&1\\
3 & 0&0&0&0&1&1\\
10 & 0&0&1&0&1&0\\
11 & 0&0&1&0&1&1\\
12 & 0&0&1&1&0&0\\
13 & 0&0&1&1&0&1\\
26 & 0&1&1&0&1&0\\
27 & 0&1&1&0&1&1\\
28 & 0&1&1&1&0&0\\
29 & 0&1&1&1&0&1\\
32 & 1&0&0&0&0&0\\
33 & 1&0&0&0&0&1\\
36 & 1&0&0&1&0&0\\
37 & 1&0&0&1&0&1\\
40 & 1&0&1&0&0&0\\
41 & 1&0&1&0&0&1\\
42 & 1&0&1&0&1&0\\
43 & 1&0&1&0&1&1\\
44 & 1&0&1&1&0&0\\
45 & 1&0&1&1&0&1\\
46 & 1&0&1&1&1&0\\
47 & 1&0&1&1&1&1\\
50 & 1&1&0&0&1&0\\
51 & 1&1&0&0&1&1\\
54 & 1&1&0&1&1&0\\
55 & 1&1&0&1&1&1\\
56 & 1&1&1&0&0&0\\
57 & 1&1&1&0&0&1\\
58 & 1&1&1&0&1&0\\
59 & 1&1&1&0&1&1\\
60 & 1&1&1&1&0&0\\
61 & 1&1&1&1&0&1\\
62 & 1&1&1&1&1&0\\
63 & 1&1&1&1&1&1
\end{array}\right|=
\begin{vmatrix}
0&0&0&0&&1\\
0&0&1&0&1&\\
0&0&1&1&0&\\
0&1&1&0&1&\\
0&1&1&1&0&\\
1&0&0&0&0&\\
1&0&0&1&0&\\
1&0&1&&&\\
1&1&0&0&1&\\
1&1&0&1&1&\\
1&1&1&&&
\end{vmatrix}=
\begin{vmatrix}
0&0&0&0&&1\\
0&&1&0&1&\\
0&&1&1&0&\\
1&0&0&&0&\\
1&&1&&&\\
1&1&0&&1&
\end{vmatrix}=
\begin{vmatrix}
0&0&0&0&&1\\
&&1&0&1&\\
&&1&1&0&\\
1&0&&&0&\\
1&&1&&&\\
1&1&&&1&
\end{vmatrix}. \quad (31)$$

To the blocks 40–47 and 56–63 of the first matrix of the block-design (31), a super-sticking rule for variables is applied, where the 2-(3, 8)-design systems are highlighted in red. For the remaining blocks of the first matrix, simple sticking of variables is carried out with the recording of the result of the image transformations into the second matrix. In the second matrix of the block-diagram block-design (31), simple sticking of variables is carried out, and in the third – the replacement of variables.

As a result, let's obtain the minimum function:

$$F=\overline{x}_1\overline{x}_2\overline{x}_3\overline{x}_4x_6+x_3\overline{x}_4x_5+x_3x_4\overline{x}_5+$$
$$+x_1\overline{x}_2\overline{x}_5+x_1x_3+x_1x_2x_5. \quad (32)$$

The result of minimization by combinatorial method (32) coincides with the result of minimization obtained by means of a three-dimensional Carnot map [19]. Example 10 demonstrates the less computational complexity of Boolean function minimization by combinatorial method.

5.6. 8-bit Boolean functions minimization. *Example* 11. To minimize the logical function [20] by the combinatorial method specified by the following truth table:

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

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

$$=\begin{vmatrix}
0&&0&1&0&1\\
1&&&0&0&\\
1&&&0&&0\\
1&&&1&1&1
\end{vmatrix}. \quad (33)$$

For every eight blocks of the first matrix of the block-design (33), the rule of super-sticking of variables is framed, since in each «eight» of blocks there is a complete combinatorial system with a repetition of 2-(3, 8)-design (highlighted in red). In the second matrix of the block-design (33), the super-sticking rule is applied (there are 2-(2, 4)-design, the matrix blocks are highlighted in red) and rule (10) (the matrix blocks are highlighted in blue). The sticking result is recorded in the third matrix. In the third matrix, there is a perfect incomplete sticking of variables with the record of the result to four matrices.

As a result, let's obtain the minimum function:

$$F=\overline{x_1}\,\overline{x_5}x_6\overline{x_7}x_8+x_1\overline{x_6}\,\overline{x_7}+x_1\overline{x_6}\,\overline{x_8}+x_1x_6x_7x_8. \qquad (34)$$

Table 12 shows the results of function minimization by the Carnot map [20] and the combinatorial method.

Table 12

The result of $F(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8)$ function minimization

Carnot map
$F(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8)=x_1\overline{x_6}\,\overline{x_7}+x_1\overline{x_6}x_7\overline{x_8}+x_1x_6x_7x_8+\overline{x_1}\,\overline{x_5}x_6\overline{x_7}x_8$
Combinatorial method
$F(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8)=x_1\overline{x_6}\,\overline{x_7}+x_1\overline{x_6}\,\overline{x_8}+x_1x_6x_7x_8+\overline{x_1}\,\overline{x_5}x_6\overline{x_7}x_8$

Considering Table 12 it is easy to see that the combinatorial method gives the second conjuncterm of a Boolean function with a smaller number of input variables.

6. Research results

The complexity of the Boolean function minimization algorithm is a quantitative characteristic that reflects the resources consumed by the algorithm during its execution. The main resources that evaluate the complexity of the algorithm are the calculus and the memory space necessary to implement this calculation using this algorithm. To evaluate the algorithmic complexity of the process of Boolean function minimization by combinatorial method, the shaped transformation operations that are performed when searching for the minimal function are used as consumed algorithmic resources. One operation of super-sticking, simple sticking, generalized sticking, absorption and substitution of variables is adopted as one transformation. The number of these transformations depends on the Boolean function capacity, the number of output conjuncts of the function, and the structure of the truth table. The possible number of such transformations, depending on the Boolean function capacity, is presented in Table 13.

Table 13

Spent shaped transformations of the combinatorial method

Capacity	The number of shaped transformations of the combinatorial method
4	4–5
5	7–18
6	8–32
7	10–58
8	15–117

Fig. 2 shows the dynamics of the growth of the number of shaped transformations by the combinatorial minimization method with increasing the capacity of the Boolean function.

From the available data, it is possible, in the first approximation, to consider the complexity of the algorithm by combinatorial method to be linearly dependent on the number of image transformations with the complexity estimate – $O(n)$ for $n<7$. With an increase in the number of variables from $n=6$ to 8, the growth dynamics of the number of transformations is characterized by the law $O(n^2)$, followed by the growth of $O(f(n))$ with the increase in Boolean function capacity according to the polynomial law.

Table 14 presents a comparison of the processes of Boolean functions minimization by the method of parallel splitting of conjuncterms [14] and combinatorial method.

Boolean function minimization by combinatorial method allows to do without automation of the process of minimizing a function with a number of variables up to 10. For a number of cases, manual minimization of a logical function is possible with a number of variables greater than 10.

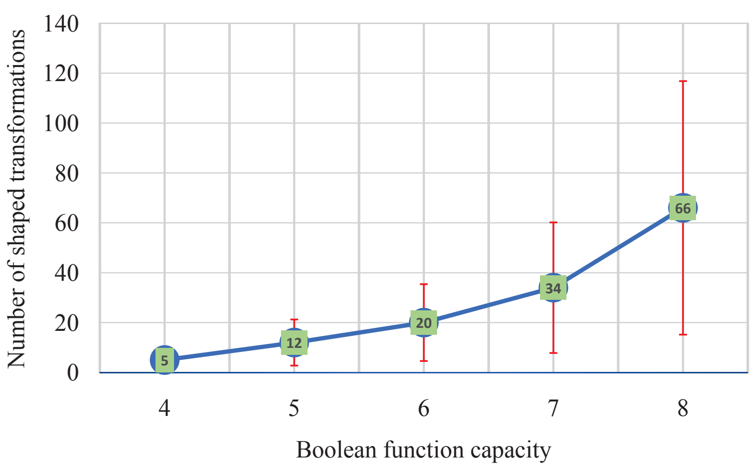

Fig. 2. Dynamics of growth of shaped transformations of the combinatorial minimization method, with increasing the Boolean function capacity

Table 14

Comparison table of two methods for Boolean functions minimization

The method of parallel splitting of conjuncterms	Combinatorial method
The first step in Boolean function minimization	
The method of parallel splitting of conjuncterms belongs to the class of heuristic methods for minimizing a logical function. The procedure of parallel splitting of conjuncterms is in overlapping on the double minterms m_1, m_2, ..., m_k function f of the masks of literals of the column matrix, resulting in the formation of a matrix of conjunctures from the submatrix of unbound conjuncts 1-, 2-, ..., (n–1)-, n-ranks. The number of identical disconnected conjuncts-copies is determined, which depends on the number k of given minterms. In the special matrix, the formed conjuncterms-copies are distinguished, and the selection of the covering elements is performed with priority for matrices of lower rank	Minimization of the logical function by combinatorial method is based on shaped transformations that increase the informative value of the minimization process in comparison with the verbal algebraic method of minimizing the function. At the first stage, let's identify blocks with variables that can be stick together. In the second step, let's search for sets of pairs of blocks with the possibility of minimizing them by sticking, absorbing, substituting, and generalizing the sticking of variables. The resulting sets of blocks are again minimized in a similar way, etc., until a dead-end DNF is obtained
The second step in Boolean function minimization	
The removal of excess implicants and the production of MDNF are carried out. To do this, the constituent's coverage table is built. The identification of the minimal function is possible under the algebraic Petrick's method	An attempt to minimize the dead-end DNF by the Blake-Poretsky method: all generalized sticking of the variables – $xy+\bar{x}z=xy+\bar{x}z+yz$ (1) is performed with the subsequent carrying out of all the absorption, or by carrying out less than two operations of generalized sticking of variables by the formula $xy+\bar{x}z+yz=xy+\bar{x}z$ for one operation (1). The specified procedure for converting the function is also possible at the first stage of minimization
Automating the process of Boolean function minimization	
Since the method uses the mathematical apparatus of matrices, submatrices, masks, and other calculations, the application of the method requires its automation	Applying the operation of super-sticking of variables, the combinatorial method can do without automating the process of Boolean functions minimization with a number of variables up to 10

7. SWOT analysis of research results

Strengths. The strength of the combinatorial method is the reduction in the complexity of the algorithm for Boolean function minimization, it makes it possible to dispense without automating the process of minimizing Boolean functions with a number of variables up to 10. This is more advantageous in comparison with analogues for the following factors:

– lower cost of development and implementation, since a significant proportion of functions are minimized by functions with a number of variables of no more than 16, and therefore, in general, the need for automation of the process of minimizing the function decreases;

– increase in manual minimization of 4–10 bit functions that facilitates control and study of the algorithm for minimizing the logic function.

Weaknesses. The weak side of the combinatorial method with manual minimization is associated with an increase in the complexity of computation with increasing number of variables of the logical function. Negative internal factors are inherent in the combinatorial method of manual minimization of a Boolean function and consist in increasing the time of obtaining the minimum function with increasing number of variables of a given function.

Opportunities. The opportunities of further studies of the combinatorial method can be the development of a protocol for optimal alternation of algebraic transformations in order to improve the accuracy of solving the problem of minimizing the function. Additional features that the implementation of the combinatorial method of Boolean function minimization can bring are the creation and support of the library of graphic images in order to optimize the algorithm for finding the minimal function by the selected criteria.

Threats. The protocol for Boolean function minimization of the combinatorial method is independent of the protocols of other minimization methods, so the threat of negative impact on the object of research of external factors is minimal. To a certain extent, the Quine-McCluskey method is an analog of the combinatorial method for Boolean function minimization. At the moment, the Quine-McCluskey method is the best because it already has an algorithm for automating the search for a minimal function.

8. Conclusions

1. The implementation of the algebraic operation of super-sticking of variables makes it possible to simplify the procedure for minimizing the Boolean function without loss of its functionality.

2. The algebraic operation of super-sticking of variables is performed if there is a complete binary combinatorial system with repetition or an incomplete binary combinatorial system with repetition in the truth table structure. The operation of super-sticking of variables is most effective when there is a complete binary combinatorial system with repetition. The efficiency of the operation of super-sticking of variables in the presence of an incomplete binary combinatorial system with repetition decreases not significantly.

3. It is established that the results of verification of the minimized function obtained using the super-sticking rule for variables that satisfy the output protocol for calculating a given function and, therefore, indicate an optimal reduction in the number of function variables without losing its functionality. The complexity of the algorithm for finding the minimal function by a combinatorial method is $O(n)$ and is linear for $n<7$. With an increase in the number of variables from $n=6$ to 8, the growth dynamics of the number of transformations is characterized by the law $O(n^2)$, followed by the growth of $O(f(n))$ with the increase in the degree of the Boolean function according to the polynomial law.

4. The efficiency of the combinatorial method is demonstrated by examples of minimizing functions borrowed from the work of other authors for the purpose of comparison:

– *Example* 4 [13], *Example* 5 [14], *Example* 6 [15], *Example* 8 [16] – minimization of 4-bit Boolean functions;

– *Example* 7 [14] – minimization of the system of 4-bit Boolean functions;

– *Example* 9 [17, 18] – 5-bit Boolean functions minimization;

– *Example* 10 [19] – 6-bit Boolean functions minimization;

– *Example* 11 [20] – 8-bit Boolean functions minimization.

Taking these examples into account, the combinatorial method of the function minimization gives grounds for the expediency of applying it in the processes of minimizing the logical function.

References

1. Quine–McCluskey algorithm [Electronic resource] // Wikipedia. – August 1, 2017 – Available at: \www/URL: https://en.wikipedia.org/wiki/Quine%E2%80%93McCluskey_algorithm
2. Riznyk, V. Minimization of Boolean functions by combinatorial method [Text] / V. Riznyk, M. Solomko // Technology Audit and Production Reserves. – 2017. – Vol. 4, No. 2 (36). – P. 49–64. doi:10.15587/2312-8372.2017.108532
3. Solairaju, A. Optimal Boolean Function Simplification through K-Map using Object-Oriented Algorithm [Text] / A. Solairaju, R. Periyasamy // International Journal of Computer Applications. – 2011. – Vol. 15, No. 7. – P. 28–32. doi:10.5120/1959-2621
4. Kumar, R. Cubical Representation and Minimization through Cubical Technique A Tabular Approach [Text] / R. Kumar, S. Rawat // International Journal of Applied Engineering Research. – 2016. – Vol. 11, No. 7. – P. 4822–4829. – Available at: \www/URL: https://www.ripublication.com/ijaer16/ijaerv11n7_27.pdf
5. Stergiou, S. A Fast and Efficient Heuristic ESOP Minimization Algorithm [Text] / S. Stergiou, K. Daskalakis, G. Papakonstantinou // Proceedins of the 14th ACM Great Lakes symposium on VLSI – GLSVLSI '04. – Boston, 2004. doi:10.1145/988952.988971
6. Dusa, A. Enhancing the Minimization of Boolean and Multivalue Output Functions WitheQMC [Text] / A. Dusa, A. Thiem // The Journal of Mathematical Sociology. – 2015. – Vol. 39, No. 2. – P. 92–108. doi:10.1080/0022250x.2014.897949
7. Voudouris, D. Exact ESCT minimization for functions of up to six input variables – PRELIMINARY VERSION [Electronic resource] / D. Voudouris, M. Sampson, G. Papakonstantinou. – 2005. – 17 p. – Available at: \www/URL: http://mule.cslab.ece.ntua.gr/xor/docs/xmin6.pdf
8. Valli, M. A State of Appraoches on Minimization of Boolean Functions [Text] / M. Valli, R. Periyasamy, J. Amudhavel // Journal of Advanced Research in Dynamical and Control Systems. – 2017. – No. 12. – P. 1322–1341. – Available at: \www/URL: http://www.jardcs.org/abstract.php?archiveid=1323#
9. Boyar, J. A New Combinational Logic Minimization Technique with Applications to Cryptology [Text] / J. Boyar, R. Peralta // Lecture Notes in Computer Science. – 2010. – P. 178–189. doi:10.1007/978-3-642-13193-6_16
10. Eungi, K. Derivations of Single Hypothetical Don't-Care Minterms Using the Quasi Quine-McCluskey Method [Text] / K. Eungi // Journal of the Korea Industrial Information Systems Research. – 2013. – Vol. 18, No. 1. – P. 25–35. doi:10.9723/jksiis.2013.18.1.025
11. Dubrova, E. Minimization of Multiple-Valued Functions in Post Algebra [Electronic resource] / E. Dubrova, Y. Jiang, R. Brayton. – 2001. – 5 p. Available at: \www/URL: https://pdfs.semanticscholar.org/e9fa/a422aad8b92c0448eb8d41425852717cb637.pdf
12. Anjuli, S. A. 2-Bit Magnitude Comparator Design Using Different Logic Styles [Text] / A. S. Anand // International Journal of Engineering Science Invention. – 2013. – Vol. 2, No. 1. – P. 13–24. – Available at: \www/URL: http://www.ijesi.org/papers/Vol(2)1%20(version%202)/C211324.pdf
13. Buniak, A. Elektronika ta mikroskhemotekhnika [Text] / A. Buniak. – Kyiv: Aston, 2001. – 382 p.
14. Rytsar, B. Ye. Minimization of logic functions system by konjuncterms parallel splittingmethod [Text] / B. Ye. Rytsar // Bulletin of the Lviv Polytechnic National University. Radio Electronics and Telecommunications. – 2013. – No. 766. – P. 18–27. – Available at: \www/URL: http://nbuv.gov.ua/UJRN/VNULPPT_2013_766_6
15. Rytsar, B. Ye. New minimization method of logical functions in polynomial set-theoretical format. 1. Generalized rules of conjuncterms simplification [Text] / B. Ye. Rytsar // Upravliaiushchie sistemy i mashiny. – 2015. – Vol. 2. – P. 39–57. – Available at: \www/URL: http://dspace.nbuv.gov.ua/handle/123456789/87194
16. Samofalov, K. G. Prikladnaia teoriia tsifrovyh avtomatov [Text] / K. G. Samofalov, A. M. Romlinkevich, V. N. Valuiskii, Yu. S. Kanevskii, M. M. Pinevich. – Kyiv: Vishcha shkola, 1987. – 375 p.
17. Plehanov, A. Simmetrichnye karty kak sredstvo minimizatsii bulevyh funktsii [Electronic resource] / A. Plehanov // Geektimes. – March 8, 2016. – Available at: \www/URL: https://geektimes.ru/post/272294/
18. Plehanov, A. Eshche raz o minimizatsii bulevyh funktsii [Electronic resource] / A. Plehanov // Habrahabr. – May 5, 2016. – Available at: \www/URL: https://habrahabr.ru/post/283030/
19. Triohmernaia karta Karno [Electronic resource] // Cyclowiki. – February 9, 2016. – Available at: \www/URL: http://cyclowiki.org/wiki/Трёхмерная_карта_Карно
20. Karta Karno [Electronic resource] // Wikipedia. – September 30, 2017. Available at: \www/URL: https://ru.wikipedia.org/wiki/Карта_Карно

ПРИМЕНЕНИЕ АЛГЕБРАИЧЕСКОЙ ОПЕРАЦИИ СУПЕР-СКЛЕИВАНИЯ ПЕРЕМЕННЫХ ДЛЯ МИНИМИЗАЦИИ БУЛЕВЫХ ФУНКЦИЙ КОМБИНАТОРНЫМ МЕТОДОМ

Рассмотрена новая процедура алгебры логики – супер-склеивание переменных, которая применяется при наличии в структуре таблицы истинности полной бинарной комбинаторной системы с повторением или неполной бинарной комбинаторной системы с повторением. Эффективность алгебраической операции супер-склеивания переменных существенно упрощает алгоритм минимизации булевых функций, что позволяет осуществлять ручную минимизацию функций с числом переменных до 10.

Ключевые слова: булева функция, метод минимизации, минимизация логической функции, блок-схема с повторением, минтермы, супер-склеивание переменных.

Riznyk Volodymyr, *Doctor of Technical Sciences, Professor, Department of Control Aided Systems, Lviv Polytechnic National University, Ukraine, e-mail rvv@polynet.lviv.ua, ORCID: http://orcid.org/0000-0002-3880-4595*

Solomko Mykhailo, *PhD, Associate Professor, Department of Computer Engineering, National University of Water and Environmental Engineering, Rivne, Ukraine, e-mail: doctrinas@ukr.net, ORCID: http://orcid.org/0000-0003-0168-5657*