Проведеними дослідженнями встановлена можливість збільшення продуктивності алгоритму мінімізації булевих функцій методом оптимального комбінування послідовності логічних операцій з використанням різних способів склеювання змінних – простого та супер-склеювання.

Встановлена відповідність інтервалів $I(\alpha, \beta)$ у булевому просторі B^n, які задаються парою булевих векторів α і β, таких, що $\alpha \leqslant \beta$ з повною комбінаторною системою з повторенням 2-(n, b)-блок-схем (англ. 2-(n, b)-designs). Внутрішні компоненти інтервалу $I(\alpha, \beta)$ відповідають повній системі 2-(n, b)-design, а зовнішні визначаються розрахунком кількості нулів або одиниць у стовпчиках таблиці істинності заданої логічної функції. Це дозволяє використовувати теорію інтервалів $I(\alpha, \beta)$ у математичному апараті комбінаторних систем 2-(n, b)-design для проведення мінімізації булевих функцій методом рівносильних образних перетворень, зокрема здійснювати автоматизований пошук систем 2-(n, b)-design у структурі таблиці істинності.

Експериментальними дослідженнями підтверджено, що комбінаторна система 2-(n, b)-design і послідовне чергування логічних операцій супер-склеювання змінних (якщо така операція можлива) та простого склеювання змінних у першій таблиці істинності підвищує ефективність процесу та достовірність результату мінімізації булевих функцій. При цьому спрощується алгоритмізація пошуку системи 2-(n, b)-design у структурі таблиці істинності заданої логічної функції, що правитиме інструментарієм для подальшої автоматизації пошуку системи 2-(n, b)-design. У порівнянні з аналогами це дає змогу підвищити продуктивність процесу мінімізації булевих функцій на 100–200 % шляхом використання оптимального чергування операцій супер-склеювання та простого склеювання змінних методом рівносильних образних перетворень.

Є підстави стверджувати про можливість збільшення продуктивності процесу мінімізації булевих функцій, шляхом оптимального комбінування послідовності логічних операцій супер-склеювання змінних та простого склеювання змінних, методом рівносильних образних перетворень

Ключові слова: мінімізація булевих функцій, оптимальне комбінування послідовності образних перетворень, карта Махоні

UDC 519.718
DOI: 10.15587/1729-4061.2020.206308

THE ALGORITHM FOR MINIMIZING BOOLEAN FUNCTIONS USING A METHOD OF THE OPTIMAL COMBINATION OF THE SEQUENCE OF FIGURATIVE TRANSFORMATIONS

V. Riznyk
Doctor of Technical Sciences, Professor
Department of Automated Control Systems
Lviv Polytechnic National University
S. Bandery str., 12, Lviv, Ukraine, 79013
M. Solomko
PhD, Associate Professor*
E-mail: doctrinas@ukr.net
P. Tadeyev
PhD, Doctor of Pedagogical Sciences, Professor
Department of Higher Mathematics**
V. Nazaruk
PhD*
L. Zubyk
PhD, Associate Professor
Department of Software Systems and Technologies
Taras Shevchenko National University of Kyiv
Volodymyrska str., 60, Kyiv, Ukraine, 01033
V. Voloshyn
PhD
Department of Computer Technology
and Economic Cybernetics**
*Department of Computer Engineering**
**National University of Water
and Environmental Engineering
Soborna str., 11, Rivne, Ukraine, 33028

Received date 20.05.2020
Accepted date 24.06.2020
Published date 30.06.2020

1. Introduction

The algebra of logic, like any computation apparatus, is a totality of axioms, identities, laws, rules, which enable the conversion of logical expressions. However, here, as a rule, there are no guidelines on how to use this apparatus for the synthesis of optimal logic schemes. The optimal solution can be provided only by building in a certain sequence of these transformations (algorithms). Methods of such construction are described in works [1–5]. In turn, a prerequisite for the creation of automated methods for reducing Boolean functions is to develop simplified algorithms for the optimum synthesis of minimal logical functions.

As the process of minimizing logical functions occupies an important position within the design technology of digital components, it is still a relevant task to ensure the adequate conformity of the developed product to the specified requirements for cost, simplification, thereby warranting the optimum result from minimizing different representations of logical functions.

A method of figurative transformations has the following scope of application: minimizing the Boolean functions in the DNF and CNF representation; the minimization of incompletely defined Boolean functions; minimizing based on a full truth table; determining an attribute of the minimum logical function; the minimization of Boolean function systems [6–9]. A promising area to study the application of a method of figurative transformations is the minimization of Boolean functions in the monobases of Schaeffer, Webb (Pierce); the minimization of the randomly given Boolean functions (Blake-Poretsky algorithm).

The evolution of methods to simplify the logical functions and their automation is the result of relentless optimization, therefore, the studies are relevant that are aimed, in particular, at the improvement of factors such as:

– algorithms of minimization (and its automation) of logical functions;

– the reliability of an optimal result;

– the cost of the logical function minimization process.

2. Literature review and problem statement

Classical methods of Boolean function minimization, the Karnaugh map and a Quine McCluskey algorithm, are given in [10], which notes that the use of a tabular Karnaugh method to minimize Boolean functions requires a lot of time, so minimizing by a manual method is limited to six variables. If a logical function has a larger number of variables, it is quite practical to use a method developed by Quine and McCluskey, which can be implemented as software. The development of software and hardware components of computer systems makes it possible to simplify the algorithm of minimization if the software has an acceptable execution speed. Paper [10] reports an algorithm and the corresponding software for minimizing the logical functions down to 20 variables, whose number is limited only by the memory of the computer system. The software was developed in the Visual Basic language. The algorithm is based on sequential clustering of terms, starting with grouping terms with one change into two terms of the same rank. As a result of this grouping, new terms are generated, with the number of variables reduced by unity. The clustering algorithm ends when variables can no longer be grouped. The algorithm described is similar to the Quine McCluskey algorithm but is simpler because it has fewer procedures.

One of the most powerful procedures to simplify Boolean expressions is the Quine McCluskey (QM) method, considered in [11]. Compared to other approaches, this method is more often applied in practice, making it possible to process a greater number of variables. The QM method is easier to implement by software, which makes it an effective apparatus to minimize Boolean functions. Study [11] describes a QM-simulator, written in the C language. The considered minimizing algorithm theoretically can process any number of variable Boolean functions.

A new heuristic algorithm for the maximum minimization of Boolean functions with a normal form of SOP is proposed in [12]. Implementing this algorithm employs graphic data; certain conditions are given to achieve the maximum level of Boolean function minimization.

A classic object-oriented algorithm to minimize Boolean functions by means of Karnaugh maps is described in [13], where language stereotypes and class diagrams are given, as well as an analysis of the productivity of the unified model of Boolean function minimization.

A new technique of the two-step optimization process of the combinational logic is described in [14]. This technique can be applied to arbitrary combinational logical tasks and often produces an improved outcome even after optimization based on standard methods. This optimization technique is used to improve software performance.

A quick granular method to minimize Boolean functions is proposed in [15]. The paper states that, first, a Boolean function changes for the sum of products. Second, the resulting truth table is obtained while statistical information in different knowledge domains is computed as heuristic information for minimizing functions. The algorithm of minimization is implemented in the MATLAB programming environment. Experimental studies confirm its high efficiency.

A new approach to minimizing Boolean expressions is suggested in work [16]. The reported minimization technique is general but the emphasis is on the «Exclusive» or «Sum of the Terms» (ESOP) functions. This method is used for solving the classic Boolean algebra problems. The resulting problem becomes a nonlinear integer program, for solving which an original branch and bound procedure with several relaxations was developed. The proposed method is convenient to minimize the incompletely defined logical functions, which is considered a complicated problem in the Boolean area. The paper reports numerous demonstrative examples of application and outlines the effectiveness of the considered approach to Boolean function minimization, presenting possible areas to continue research in future, related to solving the complex problems of ESOP. The software package, which can be used to minimize the logical functions, is given in article [17]. The package is a practical tool for teaching digital design and other related courses. The input data to this software is the number of variables at switching and a switching function to minimize. The user can choose any of the three methods for implementing the process of minimization: algebraic manipulations using theorems, Karnaugh maps, and a method by Quine McCluskey. The software begins to minimize gradually until the optimum analytical expression is reached. The user can visualize the stages of the procedure used in a particular minimization technique. This software package provides several options for minimizing logical functions and then selects the best approach among them, which employs the minimum number of logical elements in the digital component schema.

The above literary sources [10–17] mostly consider completed algorithms of Boolean function minimization and the software written for them, specifically object-oriented ones, providing the automated synthesis of minimum Boolean functions.

A feature of minimizing Boolean functions by the method of figurative transformations is greater informativeness of the solution to a problem in comparison with the algebraic way of function minimization, which is a verbal procedure due to the presence in the structure of truth tables of the complete 2-(n, b)-design or incomplete 2-(n, x/b)-design binary combinatorial systems with repetition and essentially combinatorial images. Since such objects take the form of combinatorial images, they provide more information on orthogonality, adjacency, unambiguity of blocks of combina-

torial system in comparison with algebraic transformations that opens new possibilities of application of combinatorial images for equivalent transformation of logical functions. That makes it possible to improve mental performance as an intellectual component when minimizing Boolean functions, which promotes the detection of reserves to improve the process of minimization and enables to improve the result of the figurative transformation, to increase the control function, ensuring the optimum solution is guaranteed without the need, to some extent, to use the automation of the minimization process of logical functions.

Thus, the algorithmic programs covering the overall procedure for minimizing logical functions [10–17], and a method of figurative transformations imply different approaches (principles of minimization), and thus promise various prospects on the possibility of the algorithmic minimization of logical functions.

In this regard, there are reasons to believe that the software and hardware base, which is represented by the complete algorithmizing programs [10–17], is insufficient for theoretical research into the optimum minimization of Boolean functions. This necessitates undertaking a study involving the equivalent figurative transformations of logical functions. In particular, employing the protocol of the optimum combination of figurative transformations to ensure optimal solution by the criterion of all revealed combinatorial images in a truth table, which can participate in the process of Boolean function minimization. In the applied aspect, the specified approach could expand the capabilities of the digital component design technology.

3. The aim and objectives of the study

The aim of this study is to establish the optimum alternating protocols of equivalent transformations for the initial combinatorial system, which is essentially the truth table of the assigned logic function. This would make it possible to define a principle (Latin: *principium* – beginning) to minimize the logical functions by figurative transformations, specifically in the DNF and CNF representation, and to extend the established principle on the algorithm for automating the process of logical function minimization based on a method of figurative transformations.

To accomplish the aim, the following tasks have been set:

– to determine patterns in the process of logical function minimization when using combinatorial structures of a complete binary system with repeated 2-(n, b)-design and an incomplete binary system with repeated 2-(n, x/b)-design;

– to construct an algorithm to automate the process of logical function minimization based on the method of figurative transformations within the initial combinatorial system, which is essentially the truth table of the assigned logical function;

– to demonstrate examples of Boolean functions minimization borrowed from works by other authors to compare the efficiency of the selected alternating protocols of equivalent figurative transformations at the minimization of logical functions;

The criterion for the optimum Boolean function minimization using a method of figurative transformations is described in detail in work [6]. The essence of the criterion is the need to minimize the functions on the full truth table, with the subsequent choice of the minimization result in the DNF or CNF representation. The same criterion could be extended for other bases of the possible representation of a function – monobases, Zhegalkin's basis, Reed-Muller basis, etc. Various bases by which the function can be represented form an optimization area for the assigned logical function.

4. Binary combinatorial system with repetition

If some set A is assigned, it is possible to consider a new set $M(A)$ – the set of all its subsets, *Boolean*. $M_k(A)$ is used to denote the set of all subsets A that have k elements.

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

$$M(A)=\begin{Bmatrix}\{a\},\{b\},\{c\},\{d\},\{a,b\},\{a,c\},\{a,d\},\\ \{b,c\},\{b,d\},\{c,d\},\{a,b,c\},\{a,b,d\},\\ \{a,c,d\},\{b,c,d\},\{a,b,c,d\},\varnothing\end{Bmatrix};$$

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

Check that:

$$N\left(M(A)\right)=16=2^4,\ \ N\left(M_2(A)\right)=6.$$

The number of all k-element subsets of the set of n elements equals:

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

Another equality holds:

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

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

Example 2. It is required, from formula (1), to calculate the number of all subsets of the set $A=\{a, b, c, d, e\}$.

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

Note that the set $A=\{a, b, c, d\}$, in addition to the recalculation of its elements, can also specify the numbers of the positions at which the element α is located. For example, a can denote the first position, b can denote the second position of the set $A=\{a, b, c, d\}$, etc. The subsets of the set $A=\{a, b, c, d\}$, in this case, are those subsets that contain the element α at positions k, $k=0, ..., n$, where n is the number of positions of the set A. In a general case, the element α may take several positions on the set A, thus the element α is repeated on the set A.

Let $\alpha=1$, then the positions at which the element α is absent are denoted by a zero.

Example 3. Assume $\alpha=1$ for the set $A=\{a, b, c, d, e\}$, which defines the position numbers. Then the subsets of the set A will take the form:

$$\begin{array}{cccc}(0,0,0,0,0); & (0,1,0,0,0); & (1,0,0,0,0); & (1,1,0,0,0);\\ (0,0,0,0,1); & (0,1,0,0,1); & (1,0,0,0,1); & (1,1,0,0,1);\\ (0,0,0,1,0); & (0,1,0,1,0); & (1,0,0,1,0); & (1,1,0,1,0);\\ (0,0,0,1,1); & (0,1,0,1,1); & (1,0,0,1,1); & (1,1,0,1,1);\\ (0,0,1,0,0); & (0,1,1,0,0); & (1,0,1,0,0); & (1,1,1,0,0);\\ (0,0,1,0,1); & (0,1,1,0,1); & (1,0,1,0,1); & (1,1,1,0,1);\\ (0,0,1,1,0); & (0,1,1,1,0); & (1,0,1,1,0); & (1,1,1,1,0);\\ (0,0,1,1,1); & (0,1,1,1,1); & (1,0,1,1,1); & (1,1,1,1,1);\end{array} \qquad (2)$$

The number of all k-element subsets of the set $A=\{a, b, c, d, e\}$, which defines the positions' numbers, is determined from formula (1).

$$N\left(M_0(A)\right)=C_5^0=1,$$

$$N\left(M_1(A)\right)=C_5^1=5,$$

$$N\left(M_2(A)\right)=C_5^2=10,$$

$$N\left(M_3(A)\right)=C_5^3=10,$$

$$N\left(M_4(A)\right)=C_5^4=5.$$

$$N\left(M_5(A)\right)=C_5^5=1.$$

$$N\left(M(A)\right)=N\left(M_0(A)\right)+N\left(M_1(A)\right)+N\left(M_2(A)\right)+$$
$$+N\left(M_3(A)\right)+N\left(M_4(A)\right)+N\left(M_5(A)\right)=32.$$

Configuration (2) is a complete combinatorial system with a repetition of the α element, which we denote:

2-(n, b)-design,

where n is the bit size of the system's block; b is the number of blocks in the complete system, determined from formula $b=2^n$, the number 2 before brackets denotes the binary structure of configuration (2). For example, 2-(5, 32)-design is a complete binary combinatorial system with repetition consisting of 5-bit blocks, the number of blocks is 32.

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

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

In this case, x is the number of blocks of an incomplete combinatorial system with repetition. The properties of the incomplete combinatorial system with repeated 2-(n, x/b)-design can also establish the rules, which, in a general case, ensure the effective minimization of Boolean functions.

5. Protocol of the equivalent conversion of DNF into CNF of the logical function

The protocol for minimizing the logical functions, which includes terms with the same variables in the corresponding term's bits, may take, for example, the following form:

$$F_{DNF}=\begin{vmatrix} 0 & & 0 & \\ 0 & & & 1 \\ & 1 & 0 & \\ & 1 & & 1 \end{vmatrix}=\overline{x_1}\,\overline{x_3}+\overline{x_1}x_4+x_2\overline{x_3}+x_2x_4=$$
$$=\overline{x_1}\left(\overline{x_3}+x_4\right)+x_2\left(\overline{x_3}+x_4\right)=$$
$$=\left(\overline{x_3}+x_4\right)\left(\overline{x_1}+x_2\right)=$$
$$=F_{CNF}=\begin{vmatrix} 0 & 1 & & \\ & & 0 & 1 \end{vmatrix}. \qquad (3)$$

Protocol (3), in addition to simplifying the logical expression, transforms the DNF representation into the CNF representation of the logical function. Given the matrix notation for F_{DNF} and F_{CNF} in (3), we see that the DNF and CNF of the logical function are given by matrices with identical combinatory structures. The difference between these matrices is determined by the hermeneutics of logical operations. The matrix reflecting the CNF of the logical function yields the maxterms of the function and a conjunction operation for them. The matrix reflecting the DNF of the logical function produces the minterms of the function and a disjunction operation for them [14]. The equivalence of the specified transformation is confirmed by verifying the protocol (3) (Tables 1, 2).

Table 1

Truth table of the logical function $F_{DNF}=\overline{x_1}\,\overline{x_3}+\overline{x_1}x_4+x_2\overline{x_3}+x_2x_4$ before transformation

x_1	x_2	x_3	x_4	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_3}$	$\overline{x_4}$	$\overline{x_1}\,\overline{x_3}$	$\overline{x_1}x_4$	$x_2\overline{x_3}$	x_2x_4	$\overline{x_1}\,\overline{x_3}+\overline{x_1}x_4+x_2\overline{x_3}+x_2x_4$
0	0	0	0	1	1	1	1	1	0	0	0	1
0	0	0	1	1	1	1	0	1	1	0	0	1
0	0	1	0	1	1	0	1	0	0	0	0	0
0	0	1	1	1	1	0	0	0	1	0	0	1
0	1	0	0	1	0	1	1	1	0	1	0	1
0	1	0	1	1	0	1	0	1	1	1	1	1
0	1	1	0	1	0	0	1	0	0	0	0	0
0	1	1	1	1	0	0	0	0	1	0	1	1
1	0	0	0	0	1	1	1	0	0	0	0	0
1	0	0	1	0	1	1	0	0	0	0	0	0
1	0	1	0	0	1	0	1	0	0	0	0	0
1	0	1	1	0	1	0	0	0	0	0	0	0
1	1	0	0	0	0	1	1	0	0	1	0	1
1	1	0	1	0	0	1	0	0	0	1	1	1
1	1	1	0	0	0	0	1	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	1	1

Table 2

Truth table of the logical function $F_{CNF}=\left(\overline{x_3}+x_4\right)\left(\overline{x_1}+x_2\right)$ after transformation

x_1	x_2	x_3	x_4	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_3}$	$\overline{x_4}$	$\left(\overline{x_3}+x_4\right)$	$\left(\overline{x_1}+x_2\right)$	$\left(\overline{x_3}+x_4\right)\left(\overline{x_1}+x_2\right)$
0	0	0	0	1	1	1	1	1	1	1
0	0	0	1	1	1	1	0	1	1	1
0	0	1	0	1	1	0	1	0	1	0
0	0	1	1	1	1	0	0	1	1	1
0	1	0	0	1	0	1	1	1	1	1
0	1	0	1	1	0	1	0	1	1	1
0	1	1	0	1	0	0	1	0	1	0
0	1	1	1	1	0	0	0	1	1	1
1	0	0	0	0	1	1	1	1	0	0
1	0	0	1	0	1	1	0	1	0	0
1	0	1	0	0	1	0	1	0	0	0
1	0	1	1	0	1	0	0	1	0	0
1	1	0	0	0	0	1	1	1	1	1
1	1	0	1	0	0	1	0	1	1	1
1	1	1	0	0	0	0	1	0	1	0
1	1	1	1	0	0	0	0	1	1	1

The values of the functions in the extreme right-hand columns in Tables 1, 2 are the same, meaning the equivalence of the algebraic transformation based on protocol (3).

6. Features of using the combinatorial structures 2-(*n*, *b*)-design and 2-(*n*, *x*/*b*)-design to minimize Boolean functions

Logical function minimization using figurative transformations is performed as follows. In the first step, one finds the blocks (constituents) of the truth table with variables that can be glued together (cover them). The next step is to search for the sets of pairs of blocks (implicants) with the ability to minimize them by the algebraic operations of semi-gluing, gluing, generalized gluing, absorption of variables for these pairs. The obtained sets of blocks are again minimized in a similar way, and so on, until deriving the deadlock DNF (DDNF). The sets of DDNF lso contain the minimum functions (MDNF). The last step is to verify the resulting minimum function using the assigned table, specifically the optimum minimization criterion [6].

The algebraic transformations required for the process of minimizing Boolean functions are replaced by equivalent transformations using submatrices (combinatory images) of the truth table, which is essentially a proper combinatorial system. Because combinatorial images provide more information on orthogonality, adjacency, single-nobility of combinatorial system blocks, compared to algebraic transformations, which are a verbal procedure, their use in searching for the objects for equivalent transformations, in the process of minimizing the logical function, is effective [7–9].

In a general case, the combinatorial structure of the truth table of the assigned logical function can contain combinatorial images (sub-matrices) with the structure of a complete combinatorial system with repeated 2-(*n*, *b*)-design and an incomplete combinatorial system with repeated 2-(*n*, *x*/*b*)-design. Properties of these combinatorial structures make it possible to establish rules, which, in a general case, ensure the effective minimization of Boolean functions.

To choose the optimum alternation of protocols to minimize by figurative transformations, it is necessary to determine the initial logical operation of the algebraic transformation of Boolean functions. In this regard, it is necessary to establish the peculiarities of the process of minimizing the logical functions when using combinatorial structures of a complete binary system with repeated 2-(*n*, *b*)-design and an incomplete binary system with repeated 2-(*n*, *x*/*b*)-design.

Example 4. It is required to minimize the logical function $F\left(x_1,x_2,x_3,x_4\right)$ by figurative transformations, which is assigned by the following truth table:

$F=\Sigma(6, 8, 9, 10, 11, 12, 13, 14)$

Note: The values in Σ are the minterms for rows when the function $F\left(x_1,x_2,x_3,x_4\right)$ returns «1» at the output.

To minimize the assigned function $F\left(x_1,x_2,x_3,x_4\right)$, we use a combinatorial structure of an incomplete binary system with repeated 2-(*n*, *x*/*b*)-design (Fig. 1).

6	0	1	1	0
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0

Fig. 1. Combinatorial system (truth table) of the function $F\left(x_1,x_2,x_3,x_4\right)$ with a structure of the incomplete binary system with repeated 2-(3, 7/8)-design (in red color)

Perform the minimization of the function $F\left(x_1,x_2,x_3,x_4\right)$ by figurative transformations using 2-(3, 7/8)-design.

$$F=\left|\begin{array}{c|cccc} 6 & 0 & 1 & 1 & 0 \\ 8 & 1 & 0 & 0 & 0 \\ 9 & 1 & 0 & 0 & 1 \\ 10 & 1 & 0 & 1 & 0 \\ 11 & 1 & 0 & 1 & 1 \\ 12 & 1 & 1 & 0 & 0 \\ 13 & 1 & 1 & 0 & 1 \\ 14 & 1 & 1 & 1 & 0 \end{array}\right|=$$

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

The minimized function:

$$F=x_2x_3\overline{x_4}+x_1\overline{x_2}+x_1\overline{x_3}+x_1\overline{x_4}. \quad (4)$$

The minimization protocol for 2-(3, 7/8)-design:

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

or

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

Example 5. It is required to minimize the logical function $F(x_1,x_2,x_3,x_4)$ from example 4 by figurative transformations, using a combinatorial structure of the complete binary system with repeated 2-(n, b)-design (Fig. 2).

6	0	1	1	0
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0

Fig. 2. Combinatorial system (truth table) of the function $F(x_1,x_2,x_3,x_4)$ with a structure of the complete binary system with repeated 2-(2, 4)-design (in red)

Minimizing the function $F(x_1,x_2,x_3,x_4)$ by figurative transformations using 2-(2, 4)-design.

$$F=\left|\begin{array}{c|cccc} 6 & 0 & 1 & 1 & 0 \\ 8 & 1 & 0 & 0 & 0 \\ 9 & 1 & 0 & 0 & 1 \\ 10 & 1 & 0 & 1 & 0 \\ 11 & 1 & 0 & 1 & 1 \\ 12 & 1 & 1 & 0 & 0 \\ 13 & 1 & 1 & 0 & 1 \\ 14 & 1 & 1 & 1 & 0 \end{array}\right|=\left|\begin{array}{ccc} & 1 & 1 & 0 \\ 1 & 0 & & \\ 1 & 1 & 0 & \end{array}\right|=\left|\begin{array}{ccc} & 1 & 1 & 0 \\ 1 & 0 & & \\ 1 & & 0 & \end{array}\right|.$$

The minimized function:

$$F=x_2x_3\overline{x_4}+x_1\overline{x_2}+x_1\overline{x_3}. \quad (5)$$

The operation of super-gluing the variables in the first matrix is performed for blocks 8–11, which are highlighted in red. Simple gluing of variables is carried out for blocks 12, 13, which are highlighted in blue, and 6, 14, which are highlighted in black. Comparing the minimum functions (4) and (5), we see that the minimum function (5) is simpler by one term.

It should be noted that the minimization of function (4) can continue using the implicant table (Table 3).

Table 3

Implicant table for function $F=x_2x_3\overline{x_4}+x_1\overline{x_2}+x_1\overline{x_3}+x_1\overline{x_4}$

No.	implicant	–110	10– –	1–0 –	1– – 0
6	0110	+			
8	1000		+	+	+
9	1001		+	+	
10	1010		+		+
11	1011		+		
12	1100			+	+
13	1101			+	
14	1110	+			+

Contemplating Table 3, we see that the simple implicant 1– –0 is redundant, so it can be removed from function (4). After the removal of the simple implicant 1– –0, we will obtain a minimum function (5).

Note also that the additional term $x_1\overline{x_4}$ in (4) eliminates the potential danger of a signals race. This term is redundant in terms of the static logic of the system but such redundant or conciliation terms are often necessary to ensure the non-problematic dynamic characteristics of logical circuits.

Example 6. It is required to minimize the logical function $F(x_1,x_2,x_3,x_4,x_5)$, which is assigned by the truth table $\Sigma(1,2,3,4,5,7,9,11,12,13,14,15,16,17,18,20,22,26,28,30,31)$ by figurative transformations [9].

$$F=\left|\begin{array}{c|ccccc}1&0&0&0&0&1\\2&0&0&0&1&0\\3&0&0&0&1&1\\4&0&0&1&0&0\\5&0&0&1&0&1\\7&0&0&1&1&1\\9&0&1&0&0&1\\11&0&1&0&1&1\\12&0&1&1&0&0\\13&0&1&1&0&1\\14&0&1&1&1&0\\15&0&1&1&1&1\\16&1&0&0&0&0\\17&1&0&0&0&1\\18&1&0&0&1&0\\20&1&0&1&0&0\\22&1&0&1&1&0\\26&1&1&0&1&0\\28&1&1&1&0&0\\30&1&1&1&1&0\\31&1&1&1&1&1\end{array}\right|=\left|\begin{array}{ccccc}0&0&0&0&1\\0&0&0&1&0\\0&0&0&1&1\\0&0&1&0&1\\0&0&1&1&1\\0&1&0&0&1\\0&1&0&1&1\\0&1&1&0&1\\0&1&1&1&0\\0&1&1&1&1\\1&0&0&0&0\\1&0&0&0&1\\1&0&0&1&0\\1&0&1&1&0\\1&1&0&1&0\\1&1&1&1&0\\1&1&1&1&1\\\sim&\sim&1&0&0\end{array}\right|=$$

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

The efficiency of minimizing the function $F(x_1,x_2,x_3,x_4,x_5)$ is based on the primary application of the logical operation of super-gluing the variables in the first matrix, which was carried out for blocks 4, 12, 20, 28 (highlighted in red). A complete combinatorial system with repeated (2, 4)-design was used here. The minimization of blocks in the second matrix, highlighted in blue, was carried out by means of protocol (6) [9].

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

$$=\left|\begin{array}{ccccc}y&\bar{x}&\sim&\sim&1\\y&\bar{x}&0&1&\sim\\y&x&\sim&\sim&1\\y&x&1&1&\sim\end{array}\right|=\left|\begin{array}{ccccc}y&\sim&\sim&\sim&1\\y&\bar{x}&0&1&\sim\\y&x&1&1&\sim\end{array}\right|. \qquad (6)$$

The variables' gluing (covering) protocol (6) is used on a configuration that has one column with the same variables y, and the second column contains the same number of variables x and $\bar{x}$, with the redundant combinatorial system 2-(3, 6/8)-design [9].

The blocks in the second matrix, in green, are minimized by protocol (7) [9].

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

$$=\left|\begin{array}{ccccc}y&\bar{x}&0&0&\sim\\y&\bar{x}&\sim&1&0\\y&x&\sim&1&0\\y&x&1&1&\sim\end{array}\right|=\left|\begin{array}{ccccc}y&\bar{x}&0&0&\sim\\y&\sim&\sim&1&0\\y&x&1&1&\sim\end{array}\right|. \qquad (7)$$

The variables' gluing (covering) protocol (7) is used on a configuration that has one column with the same variables y, and the second column contains the same number of variables x and $\bar{x}$, with the redundant combinatorial system 2-(3, 6/8)-design [9].

The minimized function:

$$F=\bar{x}_1x_5+\bar{x}_1\bar{x}_2\bar{x}_3x_4+x_2x_3x_4+$$
$$+x_1\bar{x}_2\bar{x}_3\bar{x}_4+x_1x_4\bar{x}_5+x_3\bar{x}_4\bar{x}_5.$$

Thus, comparing the peculiarities of the process of minimizing the logical functions using combinatorial structures of the complete binary system with repeated 2-(n, b)-design and the incomplete binary system with repeated 2-$(n, x/b)$-design, it can be concluded that the combinatorial system 2-(n, b)-design and the consistent combination of logical operations of super-gluing the variables (if such an operation is possible) and of simple gluing the variables in the first matrix (truth table) ensures a high efficiency of the process and the reliability of the results of minimizing Boolean functions.

7. Results of minimizing Boolean functions by the method of optimum combination of equivalent figurative transformations

The protocol to minimize Boolean functions by the method of optimum combination of equivalent figurative transformations has the following advantages:

– it extends the possibilities of applying the vector intervals of the Boolean space $\mathfrak{B}^n$;

– it defines a partial algorithm of recognizing the combinatorial systems 2-(n, b)-design and finding their boundaries;

– it enables the automated search for combinatorial systems 2-(n, b)-design in the structure of the truth table of the assigned logical function.

7. 1. Using the vector intervals of Boolean space $\mathfrak{B}^n$ while minimizing Boolean functions

Definition 1. The interval $I(\alpha, \beta)$ in the Boolean space $\mathfrak{B}^n$, which is assigned by a pair of Boolean vectors α and β, such that $\alpha \preccurlyeq \beta$ denotes the set of all Boolean vectors γ of length n, which satisfy the condition $\alpha \preccurlyeq \gamma \preccurlyeq \beta$, that is,

$I(\alpha, \beta)=\{\gamma\in B^n: \alpha\leqslant\gamma\leqslant\beta\}$. The Boolean vectors α and β are called the boundaries of an interval, the vector α is the smallest element of the interval, and β is the largest [18].

Example 7. $I(000,101)=\{000,001,100,101\}$, boundary $\alpha=000$ is the smallest element, boundary $\beta=101$ is the largest element.

It follows from definition 1 that either the α and β boundaries coincide in the i-th component of the Boolean vector $(a_i=b_i)$, then all vectors of the γ interval $I(\alpha, \beta)$ accept in the i-th component the same values. Or, the boundaries $(a_i<b_i)$, do not match, then such components accept in the vectors γ all possible values.

Definition 2. The components for which the boundaries (and, therefore, all vectors on the interval) coincide are termed the external components of the interval, the rest are internal. The number of the external components is termed the rank of the interval (r), and the number of internal – its bit size (s).

Example 8. In the preceding example 8, the second component is external, the first and third – internal, rank $r=1$, bit size $s=2$.

For clarity, we shall record the vectors of the interval under each other and leave the braces.

Example 9.

$$I(000,101)=\begin{matrix}000\\001\\100\\101\end{matrix}.$$

Consider extreme cases:

– $I(\alpha, \alpha)=\{\alpha\}$, the interval boundaries are the same, so it consists of a single Boolean vector, rank $r=n$, bit size $s=0$.

– $I(00...0,11...1)$ – the entire Boolean space B^n is the interval, rank $r=0$, bit size $s=n$.

Statement. The number of the Boolean vectors in the interval (interval's power) of bit size s equals 2^s.

Example 10. The number of the Boolean vectors in the interval $I(000, 111)$ equals $2^3=8$, the number of the Boolean vectors in the interval $I(000, 001, 100, 101)$ equals $2^2=4$, the number of the Boolean vectors in the interval $I(101)$ is $2^0=1$.

Comparing the interval $I(\alpha, \beta)$ of the Boolean space B^n, for example, $I(000, 101)$ from example 9, with the combinatorial structure of the truth table, it is easy to see that the internal components of the interval $I(000, 101)$ correspond to the complete combinatorial system with repeated 2-(2, 4)-design. The external components of the interval are determined by calculating the number of zeros or unities in the columns of the truth table of the logical function (paragraph 7. 3, step 5 of the considered algorithm).

In this way, the interval $I(\alpha, \beta)$ of the Boolean space B^n represents a class of the combinatorial structures of truth tables of logical functions and, therefore, may be an object to simplify its structure.

It is known that the reduction of the complete perfect disjunctive normal form (PDNF), which is a combinatorial system 2-(n, b)-design, produces unity [8]. Under the algebraic technique to minimize the interval $I(\alpha, \beta)$, the external components need to be taken out of brackets. The internal components will remain in parentheses. If the internal components of the interval are represented by a combinatorial system 2-(n, b)-design, the external interval components will become the result of minimizing. In this case, it is necessary to take into consideration the logical identity operation.

Example 11. It is required to minimize the interval from example 9 by an algebraic method.

$$I(000,101)=\begin{vmatrix} x_1 & x_2 & x_3\\ 0 & 0 & 0\\ 0 & 0 & 1\\ 1 & 0 & 0\\ 1 & 0 & 1\end{vmatrix}. \quad (8)$$

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

The minimized interval:

$$I=\overline{x_2}. \quad (9)$$

The result of minimization (9) corresponds to the external components (x_2) of interval (8).

Example 12. It is required to minimize the interval from example 9 with the help of figurative transformations.

$$I(000,101)=\begin{vmatrix}0&0&0\\0&0&1\\1&0&0\\1&0&1\end{vmatrix}=|\ \sim\ \ 0\ \ \sim\ |=\overline{x_2}. \quad (10)$$

The minimized interval:

$$I=\overline{x_2}. \quad (11)$$

The result of minimization (11) corresponds to the external components of interval (10).

The efficiency of the minimization of the interval structure in example 12 is based on the primary use of the logic operation of super-gluing the variables within the combinatorial structure's boundaries.

A similar result of minimization can be achieved with the help of the 2-(n, b)-design system search algorithm in the truth table's structure, discussed in chapter 7. 3. The specified algorithm makes it possible to perform the automated search for the intervals or the combinatorial systems 2-(n, b)-design in the structure of the truth table and is a tool for automating the process of minimizing the logical functions by a method of equivalent figurative transformations.

7. 2. A partial algorithm for the recognition of intervals (or combinatorial systems 2-(n, b)-design) and to search for its boundaries

Start: A set **A** of the Boolean vectors of length n is assigned (or a truth table of the Boolean function $F(x_1, x_2, ..., x_n)$ is assigned).

Step 1. If the power of the **A** set is not an integer power of two, that is, $|\mathbf{A}|\neq 2^c$, where c is the integer, then the ser **A** is not an interval, proceed to the completion of the algorithm.

Step 2. Determine the number s of the nonmatching components in the vectors of the set **A**, that is, the number of the components that claim be internal. If $s\neq c$, then **A** is not an interval, proceed to the completion of the algorithm; otherwise, **A** is the interval, s is its bit size, $r=n-S$ – its rank.

Step 3. Find the limits α and β of the interval. The minimum weight vector (of the entire set of vectors **A**) is the smallest element (α) in the interval, and the maximum weight vector is the largest element (β).

Complete. [18]

Example 13. $\mathbf{A}=\{010,011,001\}$: the set **A** does not form an interval because its power is 3, and thus it is not an integer power of two.

Example 14. $\mathbf{A}=\{0010,0011,0001,1000\}$: the set **A** does not form an interval – the power is an integer power of two but the power of degree $c=2$ does not coincide with the number of components $s=3$ that claim to be internal (these are the first, third, and fourth components).

Example 15. $\mathbf{A}=\{010,011,001,000\}$: the set **A** forms an interval as its power is an integer power of two ($c=2$) and this power coincides with the number of components ($s=2$) that claim to be internal (these are the second and third components). Interval boundaries: $\alpha=000$, $\beta=011$.

The algorithm of the recognition of intervals (or combinatorial systems-(n, b)-design) recognizes the required objects provided $|\mathbf{A}|=2^c$, where c is an integer.

However, in most cases, the combinatorial system 2-(n, b)-design must be searched on the condition $|\mathbf{A}|\neq 2^c$, where c is the integer and $b\geq|A|$, where b is the number of binary blocks in the truth table of the Boolean function $F(x_1,x_2,...x_n)$. Therefore, the considered interval recognition algorithm (or combinatorial systems 2-(n, b)-design) is a partial algorithm and can be applied at the final stage of searching for the combinatorial systems 2-(n, b)-design.

7. 3. Automated search for the 2-(n, b)-design combinatorial systems in the structure of a truth table of the logical function

The 2-(n, b)-design system can take a compact arrangement in the structure of the truth table of the assigned logical function or non-compact, for instance, in the first matrix of example 6. The truth table can contain several 2-(n, b)-design combinatorial systems. In a general case, and especially when increasing the bit size of a logic function, the unambiguous detection of the 2-(n, b)-design system or the 2-(n, b)-design systems in the truth table's structure would enable the automated search for the examined combinatorial systems.

For the automated search for the 2-(n, b)-design system in the structure of the truth table of any Boolean function, it is necessary to perform a sequence of actions, which can be represented by the following algorithm:

Start: the truth table of the Boolean function $F(x_1,x_2,...x_n)$ is assigned.

Step 1. The structure of the truth table should be appropriately represented in the perfect disjunctive normal form (PDNF).

Step 2. Arrange the truth table's blocks in lexicographical order.

Step 3. Check the structure of the truth table for the presence of identical blocks. If the same blocks are present, represent them in a single block.

Step 4. Check whether the assigned structure of the truth table is the complete combinatorial system with repetition. If the assigned structure of the truth table is a complete combinatorial system with repetition, complete the search for the 2-(n, b)-design system.

Step 5. Calculate the number of unities and zeros separately in each column of the truth table.

Step 6. Based in the results of the calculations in p. 5, ensure a taxonomic search for the 2-(n, b)-design combinatorial systems. For example, if the number of zeros or unities (k) corresponds to the condition $k\geq 8$ a 4-bit logical function is considered, the search for the Combinatorial system 2 (3, 8)-design is possible. If the number of zeros or units (k) meets condition $4\leq k<8$ and a 4-bit logical function is considered, it is possible to find the 2-(2, 4)-design combinatorial system, etc.

Step 7. Search for intervals, starting with the maximum, or the 2-(n, b)-design combinatorial systems.

Complete.

Fig. 3 shows a block diagram of the 2-(n, b)-design system search algorithm 2 in the structure of the truth table of the assigned logical function.

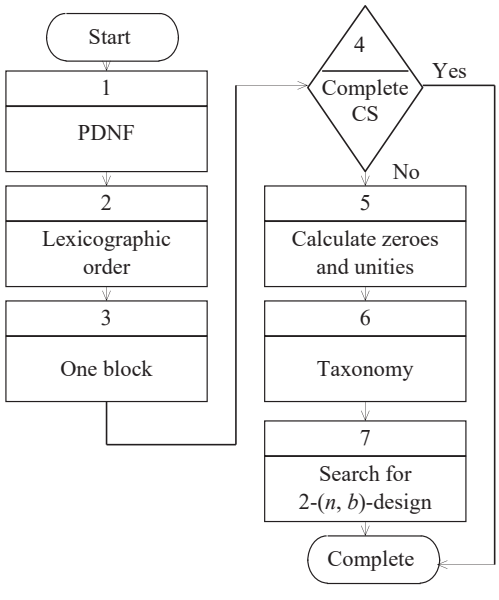

Fig. 3. Block diagram to search for the 2-(n, b)-design system: 1 – representation of the structure of the truth table in the PDNF; 2 – arrange the blocks of the truth table in lexicographical order; 3 – representation of several blocks of the truth table in one block; 4 – check whether the assigned structure of the truth table is a complete combinatorial system with repetition; 5 – calculate separately the number of unities and zeros in each column of the truth table; 6 – provide the taxonomy for the search of the 2-(n, b)-design combinatorial systems; 7 – search for the 2-(n, b)-design combinatorial systems

Fig. 4 shows the results of the automated search for the 2-(n, b)-design in the structure of the truth table of the assigned logical function.

0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
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
13	1	1	0	1

a

0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
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
13	1	1	0	1

b

Fig. 4. Automated search for the 2-(2, 4)-design system: *a* – truth table to search for the 2-(2, 4)-design system; *b* – truth table after searching for the 2-(2, 4)-design system

When increasing the bit size of the Boolean function, the software search for the 2-(n, b)-design combinatorial system in the structure of the truth table becomes substantially more productive and more reliable.

A set of argument sets (2-(n, b)-design configuration), in other words, a set of the vertices of the n-dimensional single cube, assigns the area for determining the algebra logic function. The single n-dimensional cube is two (n–1)-dimensional single cubes, in which all their corresponding vertices are connected by segments of the single length.

The two 0-dimensional single cubes (two points) at a distance equal to unity forms a 1-dimensional single cube. Two 1-dimensional single cubes, whose corresponding vertices are located at a distance equal to unity, forms a 2-dimensional single cube, two 2-dimensional single cubes form a 3-dimensional single cube. Similarly, a 4-dimensional cube is built. The corresponding vertices of the two 3-dimensional cubes are also connected by segments of the single length.

The elements of the cube representing, for example, an arbitrary 3-variable logical function can be assigned the conjunctions of different ranks: to vertices – the conjunctions of rank 3, to edges – the conjunctions of rank 2, to facets – the conjunctions of rank 1.

The vertices, edges, and facets are the geometric equivalents of conjunctions. The sum of the dimensionality of the geometric equivalent and rank assigned to this geometric equivalent of the conjunction is constant and equals the number of the function's arguments (in our case, 3). Each geometric equivalent of a smaller dimensionality is covered by all geometric equivalents of larger dimensionality, that is, the greater rank conjunctions are covered by the conjunction of a smaller rank. For example, the conjunctions x_1,x_2,x_3, $x_1,x_2,\overline{x_3}$ are covered by the conjunction x_1, x_2.

Geometric equivalents of some rank are termed intervals [19]. In a 3-dimensional cube, the intervals of rank 3 are the vertices, the intervals of rank 2 – edges, the intervals rank 1 – facets.

For example, the conjunction x_1 corresponds to a set of vertices with the coordinates (1,0,0), (1,0,1), (1,1,0), (1,1,1). The corresponding interval of rank 1 coincides with the cube's facet covering these four vertices.

The vertices that match the set of arguments on which the function returns «1» form the set $\mathbf{T}_1$. Representing some DNF for a logical function is equivalent to representing some covering the set $\mathbf{T}_1$ by intervals, which are defined by the conjunctions included in DNF.

That is how one defines the correspondence between representing a function in the DNF form and covering the set $\mathbf{T}_1$ by intervals of some rank for a given function.

If the ranks of all intervals that form the covering of the logical function are denoted through r_1 , r_2 ,... r_n, the total rank of DNF:

$$R=\sum_{i=1}^{n} r_i,$$

numerically coincides with the number of variables included in DNF and would provide the optimum simplification of the logical function by covering the set $\mathbf{T}_1$, at which R is minimal [19].

The submatrix of the truth table containing 2-(n, b)-design is an interval for the specified covering of Boolean functions. Finding all the required intervals increases the effectiveness of a method of figurative transformations. Fig. 4, b demonstrates the two found intervals to accommodate the 2-(2, 4)-design combinatorial systems (highlighted in red and blue). One more interval accommodates the 2-(1, 2)-design system (highlighted in green).

A variant to search for the 2-(n, b)-design system in the truth table's structure is a so-called method of self-reducing cycles [20]. It makes it possible to receive the reduced DNF of an arbitrary Boolean function based on special fragments – the self-reducing cycles (Table 4) of its truth table.

Table 4

Thesaurus of minimization methods

No. of entry	Thesaurus of minimization using figurative transformations	Thesaurus of minimization using Boolean space vectors	Thesaurus of minimization using self-reducing cycles
1	Submatrix of truth table containing 2-(n, b)-design	Maximum interval	Self-reducing cycle

Each fragment is distinguished in a way that only one conjunction k_i can be represented. The shortened DNF is determined using the disjunctive conjunction, provided that all highlighted fragments of the truth table of the Boolean function fully cover that part of the sets on which the Boolean function returns «1» (Fig. 5).

Variables	Function
$x_1, x_2, ...x_n$	f
Fragment 1	1
Fragment 2	1
…	
Fragment i	1

Fig. 5. Fragments of the truth table containing 2-(n, b)-design: fragment 1 corresponds to conjunction k_1, fragment 2 corresponds to conjunction k_2, fragment i corresponds to conjunction k_i

If the truth table of any Boolean function has a submatrix, with 2-(n, b)-design (Table 5), then its reduced DNF on this sub-matrix is given by expression:

$$f = x_1x_2...x_{n-3},$$

since the operation of super gluing the variable can be applied to the variables x_{n-2}, x_{n-1}, x_n.

Table 5

Submatrix that accommodates 2-(n, b)-design

x_1	x_2	...	x_{n-3}	x_{n-2}	x_{n-1}	x_n	f
...	...	...	...	...	...	...	...
α_1	α_2	...	α_{n-3}	0	0	0	1
α_1	α_2	...	α_{n-3}	0	0	1	1
α_1	α_2	...	α_{n-3}	0	1	0	1
α_1	α_2	...	α_{n-3}	0	1	1	1
α_1	α_2	...	α_{n-3}	1	0	0	1
α_1	α_2	...	α_{n-3}	1	0	1	1
α_1	α_2	...	α_{n-3}	1	1	0	1
α_1	α_2	...	α_{n-3}	1	1	1	1
...	...	...	...	...	...	...	...

Note that when there is a submatrix with 2-(n, b)-design and it contains 2^m binary sets of length n, where m is the number of glue variables, such a sub-matrix is represented by the conjunction of rank $r=n-m$. And the assigned Boolean function, on the basis of the submatrix with 2-(n, b)-design, is transformed into a reduced DNF.

The sub-matrix in the form (Table 5) in [20] is termed a self-reducing cycle, and the number of glued together variables in the submatrix – the rank of the self-reducing cycle.

Example 16. It is required to find the self-reducing cycles of ranks 3, 2, 1 for the Boolean function $F(x_1,x_2,x_3,x_4)$, which is assigned by the truth table (Table 6) [20].

Table 6

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

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

Self-reducing cycle 1. Cycle rank is 3. Glued variables: x_1, x_2, x_3. Reduced DNF: $f=x_2$. The submatrix with 2-(3, 8)-design is given in Table 7.

Table 7

Submatrix with 2-(3, 8)-design

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

Self-reducing cycle 2. Cycle rank is 2. Glued variables: x_2, x_4. Reduced DNF: $f=x_1x_3$. The submatrix with 2-(2, 4)-design is given in Table 8.

Self-reducing cycle 3. Cycle rank is 2. Glued variables: x_1, x_4. Reduced DNF: $f=x_2x_3$. The submatrix with 2-(2, 4)-design is given in Table 9.

Self-reducing cycle 4. Cycle rank is 1. Glued variables: x_4. Reduced DNF: $f=x_1x_2x_3$. The submatrix with 2-(1, 2)-design is given in Table10.

Table 8

Self-reducing cycle 2

x_1	x_2	x_3	x_4	f
0	0	**0**	0	1
0	0	**0**	1	1
0	1	**0**	0	1
0	1	**0**	1	1

Table 9

Self-reducing cycle 3

x_1	x_2	x_3	x_4	f
0	**0**	**1**	0	1
0	**0**	**1**	1	1
1	**0**	**1**	0	1
1	**0**	**1**	1	1

Table 10

Self-reducing cycle 4

x_1	x_2	x_3	x_4	f
0	**1**	**0**	0	1
0	**1**	**0**	1	1

Thus, for the automated search for the resulting reduced DNF of any Boolean function $F(x_1,x_2,...x_n)$, it is necessary to perform a sequence of actions that can be represented by the following algorithm [20]:

Start: the assigned truth table of the Boolean function $F(x_1,x_2,...x_n)$.

Step 1. Set $i=1$. Proceed to step 2.

Step 2. Find all the self-reducing cycles of rank $r=n-i$. If the cycles of all the found ranks cover all the unities of the Boolean function in its truth table, then proceed to step 5, otherwise – to step 2.

Step 3. Disregard all self-reducing cycles of rank $r<(n-i)$, which are fully included into one or more self-reducing cycles of rank $r\geq(n-i)$. Set $i=i+1$. Proceed to step 4.

Step 4. If $i>(n-1)$, proceed to step 5, otherwise – to step 2.

Step 5. Synthesize the reduced DNF of the Boolean function $F(x_1,x_2,..x_n)$ by means of disjunctive conjunction, which are found from the self-reducing cycles.

Complete.

The minimization of function $F(x_1,x_2,x_3,x_4)$ (Table 6) by the method of figurative transformations is reduced to the following procedure [8]:

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

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

In the submatrix of blocks 0–3 and 8–11 (highlighted in red), which accommodates the 2-(3, 8)-design combinatorial system, the operation of super-gluing the variables is applied. The simple gluing of variables is highlighted with black color. In the last matrix, we carried out the incomplete gluing of variables. The result is the following minimum function:

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

The result of minimizing by the method of figurative transformations coincides with the result of minimization obtained by the method of self-reducing cycles [20]. The method of self-reducing cycles uses four self-reducing cycles, which yields four conjunctions. It is usually necessary to have an implicant table (covering table) to detect the redundant conjunction. The method of figurative transformations minimizes the function $F(x_1,x_2,x_3,x_4)$ (Table 6) in three transformations, so it can be attributed to the procedure of minimization with less complexity.

8. Comparative analysis of the method of optimum alternation of figurative transformations with other methods for minimizing functions

The application of the optimal solution according to the criterion of all identified combinatorial images in the truth table that can take part in the process of minimization of Boolean functions and the protocol of optimal alternation of figurative transformations (logical operations) in minimizing Boolean functions is demonstrated by the examples of minimizing the logical functions borrowed from papers by other authors for comparison.

8. 1. Comparison with Mahoney maps

Example 17. It is required to find the minimum DNF and CNF of logical functions by using a Mahoney map, obtained from the assigned truth table (Fig. 6) [21].

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

Fig. 6. Truth table of the logical function for example 17

Fig. 7 is the resulting Mahoney map, derived from the truth table in Fig. 6. The contours of unities are denoted by the solid ellipses, and the contours of zeros – dotted ellipses.

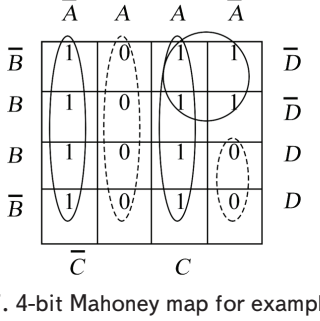

Fig. 7. 4-bit Mahoney map for example 17

Mahony maps in many ways are much more efficient compared to Karnaugh maps as they easily expand to the required amount of input data, which significantly expands the overall scope of the Mahoney maps application [21].

Using the contours of unities in Fig. 6, the minimum DNF of the assigned logical function (Fig. 6) takes the form:

$$OUT = C\cdot A + \overline{D}\cdot C + \overline{C}\cdot\overline{A}. \qquad (12)$$

The minimization of function $F(x_1,x_2,x_3,x_4)$ (Fig. 6) by a method of figurative transformations is reduced to the following procedure:

Variant 1.

$$F = \left|\begin{array}{c|cccc} \text{No.} & x_4 & x_3 & x_2 & x_1 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 \\ 4 & 0 & 1 & 0 & 0 \\ 5 & 0 & 1 & 0 & 1 \\ 6 & 0 & 1 & 1 & 0 \\ 7 & 0 & 1 & 1 & 1 \\ 8 & 1 & 0 & 0 & 0 \\ 10 & 1 & 0 & 1 & 0 \\ 13 & 1 & 1 & 0 & 1 \\ 15 & 1 & 1 & 1 & 1 \end{array}\right| = \left|\begin{array}{ccc} & 0 & 0 \\ 0 & 1 & \\ 1 & 1 & 1 \end{array}\right| = \left|\begin{array}{ccc} & 0 & 0 \\ 0 & 1 & \\ & 1 & 1 \end{array}\right|.$$

Blocks 4–7 (highlighted in red) and blocks 0, 2, 8, 10 (highlighted in blue) are minimized based on the protocol for super-gluing the variables. The other blocks are minimized based on the protocols for simple gluing and semi-gluing the variables [7]. The minimized DNF of the function:

$$F = \overline{x_1}\,\overline{x_3} + x_3\overline{x_4} + x_1x_3. \qquad (13)$$

The result of minimization (13) coincides with the result of minimization (12), conducted by using a Mahoney map [21], however, the synthesis of the minimum DNF of the logical function by a method of figurative transformations is a simpler procedure.

Variant 2.

$$F = \left|\begin{array}{c|cccc} \text{No.} & x_4 & x_3 & x_2 & x_1 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 \\ 4 & 0 & 1 & 0 & 0 \\ 5 & 0 & 1 & 0 & 1 \\ 6 & 0 & 1 & 1 & 0 \\ 7 & 0 & 1 & 1 & 1 \\ 8 & 1 & 0 & 0 & 0 \\ 10 & 1 & 0 & 1 & 0 \\ 13 & 1 & 1 & 0 & 1 \\ 15 & 1 & 1 & 1 & 1 \end{array}\right| = \left|\begin{array}{ccc} 0 & & 0 \\ & 1 & 1 \\ 1 & 0 & 0 \end{array}\right| =$$

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

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

Blocks 0, 2, 4, 6 (highlighted in red) and 5, 7, 13, 15 blocks (highlighted in blue) are minimized based on the protocol for super-gluing the variables. The other blocks are minimized based on the protocols for simple gluing and semi-gluing the variables [7].

The minimized DNF of the function:

$$F = \overline{x_1}\,\overline{x_3} + x_3\overline{x_4} + x_1x_3. \quad (14)$$

The result of minimization (14) coincides with the result (12).

Using the contours of zeros in Fig. 6, the minimum CNF of the logical function (Fig. 6) takes the form:

$$\overline{OUT} = \overline{C}\cdot A + D\cdot C\cdot \overline{A}.$$

By applying de Morgan's law, we obtain:

$$\overline{\overline{OUT}} = \overline{\left(\overline{C}\cdot A\right) + \left(D\cdot C\cdot \overline{A}\right)},$$

$$OUT = \left(\overline{\overline{C}\cdot A}\right)\left(\overline{D\cdot C\cdot \overline{A}}\right),$$

and, ultimately, the minimal CNF:

$$OUT = \left(C + \overline{A}\right)\left(\overline{D} + \overline{C} + A\right). \quad (15)$$

The minimization of the CNF of the function $F(x_1, x_2, x_3, x_4)$ (Fig. 6) by a method of figurative transformations is reduced to the following procedure [6]:

$$F = \left|\begin{array}{cccccc} \text{No.} & x_4 & x_3 & x_2 & x_1 \\ 1 & 0 & 0 & 0 & 1 \\ 3 & 0 & 0 & 1 & 1 \\ 9 & 1 & 0 & 0 & 1 \\ 11 & 1 & 0 & 1 & 1 \\ 12 & 1 & 1 & 0 & 0 \\ 14 & 1 & 1 & 1 & 0 \end{array}\right| = \left|\begin{array}{cccc} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{array}\right| = \left|\begin{array}{ccc} & 1 & 0 \\ 0 & 0 & 1 \end{array}\right|.$$

Blocks 1, 3, 9, 11 (highlighted in red) are minimized based on the protocol for super-gluing the variables. The other blocks are minimized based on the protocols for simple gluing the variables [7].

The minimized CNF of the function:

$$F = \left(\overline{x_1} + x_3\right)\left(x_1 + \overline{x_3} + \overline{x_4}\right). \quad (16)$$

The result of minimization (16) coincides with the result of minimization (15), conducted by using a Mahoney map [21], however, the synthesis of the minimum CNF of the logical function by a method of figurative transformations is a simpler procedure.

8. 2. Comparison with the method of non-directional graph

The method of non-directional graph is based on graph theory, which include digital transformations. Minimization using the specified method is carried out by the following algorithm [22]:

1. Define the minterms (the sets of variables for a Boolean function), at which the function returns the logical unity.

2. Determine by the graph levels (indexes) based on the number of unities in the set of variables.

3. Synthesize a fragment of the graph determining the arcs connecting the vertices of the graph.

In order to synthesize the graph, it is necessary to define its main characteristics. Total vertices:

$$N_{vertice} = 2n,$$

where n is the number of variables. The number of graph levels:

$$N_{level} = n + 1.$$

The number of vertices in the graph levels is determined from a formula for defining a number per one connection:

$$N_{vertice\ in\ level} = C_n^i,$$

where i is the level number, i=0, 1, 2, ..., n.

After the graph is synthesized, it is necessary to connect the vertices, which have the difference in only one position. The main stage in the minimization of a logical function based on the considered method is the identification of closed circuits. If the four vertices, which are interconnected, represent a closed geometric figure, then the result of minimizing is two variables. If it is possible to combine only two vertices, then three variables are derived. If a vertex cannot be paired with any other, then its lettering would be fully included into the summary notation. The result (the minterms obtained) is recorded through a disjunction.

One advantage of solving by the non-directional graph method is the possibility to minimize the logical functions with the help of the pre-created graph-shaped structure. For 4 variables, it takes the form similar to Fig. 8.

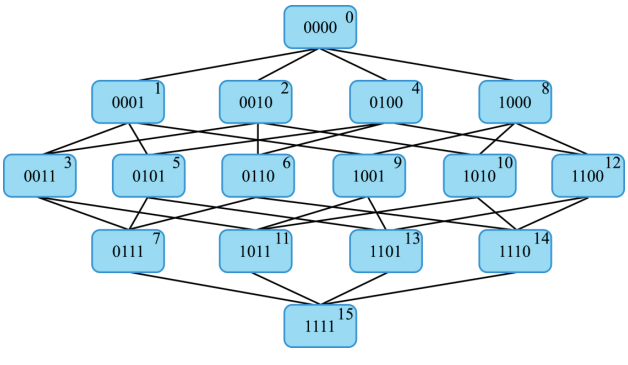

Fig. 8. Full graph for 4-variable logical function

Thus, the solution is not always required to use a full graph, it is possible to synthesize its fragment (Fig. 9).

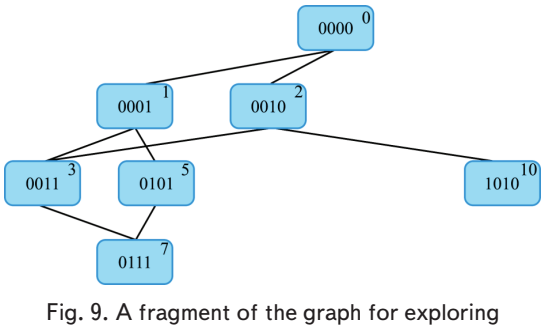

Fig. 9. A fragment of the graph for exploring a logical function

Example 18. Let a function be set in the following form:

$$f=\bar{a}\bar{b}\bar{c}\bar{d}+\bar{a}\bar{b}\bar{c}d+\bar{a}\bar{b}c\bar{d}+\bar{a}\bar{b}cd+\bar{a}b\bar{c}\bar{d}+\bar{a}bcd+\bar{a}bc\bar{d}+ab\bar{c}\bar{d}. \quad (17)$$

Logical function (17) depends on 4 variables; a full graph for its is shown in Fig. 8. Synthesize the corresponding fragment of the assigned graph. Because logical function (17) returns the logical unity in the recorded minterms, we leave in the fragment the corresponding vertices with the numbers 0000, 0001, 0010, 0011, 0101, 0111, 1010 (for simplification, record their numbers in the decimal number system: 0, 1, 2, 3, 5, 7, 10 (Fig. 9)).

Given Fig. 9, we see that on the obtained graph one can select two quadrilaterals (0-1-3-2 and 1-3-7-5) and the segment 2-10.

Contemplating the quadrilateral 0-1-3-2, we see that the shared part of all vertices is the two first zeros «00_ _». We shall also select the shared part of the vertices of a second quadrilateral 1-3-7-5: «0_ _1»; the shared part of the segment 2-10 vertices: «_010». By recording the resulting expression in a letter form, we obtain the result of the minimization of the assigned Boolean function:

$$f=\bar{a}\bar{b}+\bar{a}b+\bar{b}c\bar{d}. \quad (18)$$

Check the derived minimization result (18) by algebraic method by applying the laws and identities from the algebra of logic.

$$\begin{aligned}f&=\bar{a}\bar{b}\bar{c}\bar{d}+\bar{a}\bar{b}\bar{c}d+\bar{a}\bar{b}c\bar{d}+\bar{a}\bar{b}cd+\bar{a}b\bar{c}\bar{d}+\\&+\bar{a}bcd+a\bar{b}c\bar{d}=\bar{a}\bar{b}\bar{c}\left(\bar{d}+d\right)+\bar{b}c\bar{d}\left(\bar{a}+a\right)+\\&+\bar{a}bd\left(\bar{c}+c\right)+\bar{a}\bar{b}c\left(\bar{d}+d\right)+\bar{a}\bar{b}d\left(\bar{c}+c\right)=\\&=\bar{a}\bar{b}\bar{c}+\bar{b}c\bar{d}+\bar{a}bd+\bar{a}\bar{b}c+\bar{a}\bar{b}d=\bar{a}\bar{b}\left(\bar{c}+c\right)+\\&+\bar{b}c\bar{d}+\bar{a}d\left(\bar{b}+b\right)=\bar{a}\bar{b}+\bar{b}c\bar{d}+\bar{a}d.\end{aligned}$$

The result of the algebraic validation method confirms the fairness of the minimization result (18) of the logical function (17), obtained by the method of the non-directional graph.

Minimizing the function (17) by a method of figurative transformations is reduced to the following procedure:

Variant 1.

$$F=\left|\begin{array}{c|cccc}0&0&0&0&0\\1&0&0&0&1\\2&0&0&1&0\\3&0&0&1&1\\5&0&1&0&1\\7&0&1&1&1\\10&1&0&1&0\end{array}\right|=$$

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

Blocks 0–3 (highlighted in red) are minimized based on the protocol for super-gluing the variables. The other blocks are minimized based on the protocols for simple gluing and semi-gluing the variables [7].

The minimized function:

$$F=\overline{x_1}\,\overline{x_2}+\overline{x_1}x_4+\overline{x_2}x_3\overline{x_4}. \quad (19)$$

The minimization result (19) coincides with the result of minimization (18), conducted by the non-directional graph method [22], however, the synthesis of the minimal logic function (19) by a method of figurative transformations is a simpler procedure.

Variant 2.

$$F=\left|\begin{array}{c|cccc}0&0&0&0&0\\1&0&0&0&1\\2&0&0&1&0\\3&0&0&1&1\\5&0&1&0&1\\7&0&1&1&1\\10&1&0&1&0\end{array}\right|=$$

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

Blocks 1, 3, 5, 7 (highlighted in red) are minimized based on the protocol for super-gluing the variables. The other blocks are minimized based on the protocol for simple gluing and semi-gluing the variables [7].

The minimized function:

$$F=\overline{x_1}\,\overline{x_2}+\overline{x_1}x_4+\overline{x_2}x_3\overline{x_4}. \quad (20)$$

The result of minimization (20) coincides with the result of minimization (18), conducted by a non-directional graph method [22].

8. 3. Comparison of Boolean function minimization using a cubic technique

Paper [22] considered the cubic methods of minimizing logical functions at minimal cost.

Example 19. It is required to minimize logical function:

$$F(a,b,c,d)=\Sigma(0,4,8,10,11,12,13,15) \quad (21)$$

using a cubic method (Fig. 10) [23]. Note: values in Σ are the sets of variables when the function $F(a,b,c,d)$ returns «1» at the output.

Larger cubes can be formed only from those minterms that differ only in one variable. This makes it possible to reduce the number of pairwise comparisons if one splits the minterms into groups where the cubes in each group have the same number of unities. Thus, it will be necessary to compare each cube in a given group only with all cubes from the directly preceding group (Table 11).

For example, the minterms called the 0-cubes can be merged with the minterms called the 1-cubes. If the 0-cubes are included in 1-cubes, then this fact is accounted for in a certain way. The implicant tables [23] are used to determine the minimum coverage.

Minimizing by a cubic method for a given example results in the following Boolean function:

$$F=a\bar{b}c+abd+\bar{c}\bar{d}. \quad (22)$$

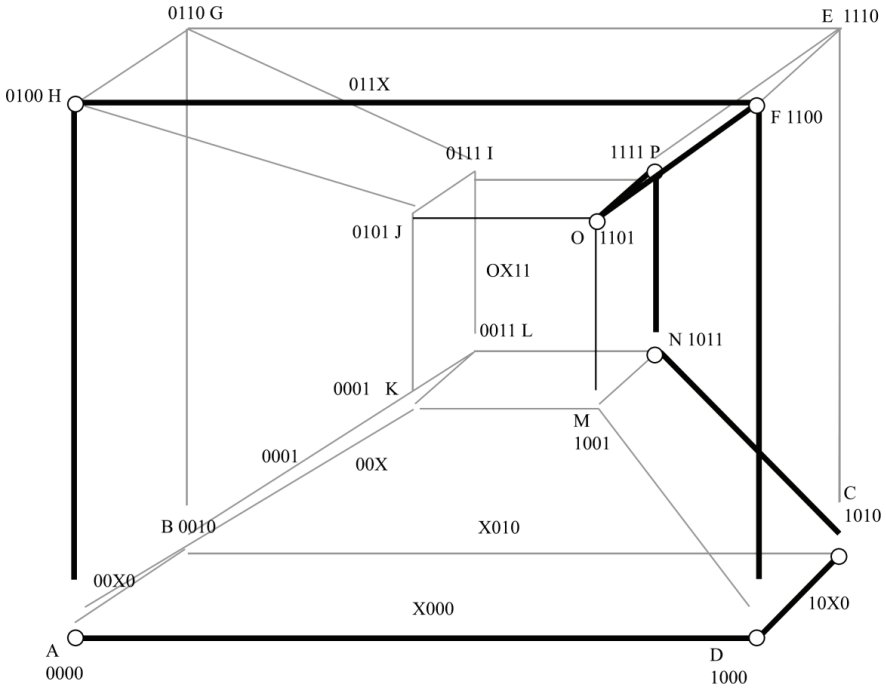

Fig. 10. 4-dimensional cube to explore the logical function $F(a,b,c,d)$

Table 11

Groups of cubes for function $F(a,b,c,d)=\Sigma(0,4,8,10,11,12,13,15)$

No.	minterms	cubes
0	0000	0-cubes
4	0100	1-cubes
8	1000	
10	1010	2-cubes
12	1100	
11	1011	3-cubes
13	1101	
15	1111	4-cubes

Minimizing the function $F(x_1,x_2,x_3,x_4)$ (21) by a method of figurative transformations is reduced to the following procedure:

$$F=\left|\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 \\ 8 & 1 & 0 & 0 & 0 \\ 10 & 1 & 0 & 1 & 0 \\ 11 & 1 & 0 & 1 & 1 \\ 12 & 1 & 1 & 0 & 0 \\ 13 & 1 & 1 & 0 & 1 \\ 15 & 1 & 1 & 1 & 1 \end{array}\right| = \left|\begin{array}{cccc} & & 0 & 0 \\ 1 & 0 & 1 & \\ 1 & 1 & & 1 \end{array}\right|.$$

Blocks 0, 4, 8, 12 (highlighted in red) are minimized based on the protocol for super-gluing the variables. The other blocks are minimized based on the protocols for simple gluing the variables [7].

The minimized function:

$$F=\overline{x}_3\overline{x}_4+x_1\overline{x}_2x_3+x_1x_2x_4. \qquad (23)$$

The result of minimization (23) coincides with the result of minimization (22), performed by a cubic method [23], however, the synthesis of the minimum logical function (22) by the method of figurative transformations is a simpler procedure.

The chosen sequence of logical operations for equivalent figurative transformations (the protocols of equivalent transformations) in the initial combinatorial system (truth table) in examples 17–19 is given in Table 12.

Table 12

Sequence of logical operations for equivalent figurative transformations in the initial combinatorial system (truth table)

Example No.	Which minimization method is compared to	Sequence of logical operations for equivalent transformations
17	Mahoney maps	Super-gluing of variables, simple gluing of variables
18	Non-directional graph method	Super-gluing of variables, simple gluing of variables
19	Cube method	Super-gluing of variables, simple gluing of variables

Contemplating Table 12, we see that minimizing the logical functions by the method of figurative transformations uses the same sequence of logical operations in the initial truth table – the super-gluing of variables with the subsequent application of the simple gluing of variables. In each comparative example, the minimization results are the same, but the synthesis of the minimum logical functions by the method of figurative transformations is a simpler procedure.

Thus, the alternation of logical operations of the super-gluing of variables and the simple gluing of variables is an optimal sequence of the application of protocols for equivalent figurative transformations in order to minimize logical functions. Establishing the optimum algorithms for simplification creates a prerequisite for constructing the automated methods for minimizing Boolean functions.

9. Discussion of results of alternating the logical operations of the super-gluing of variables and the simple gluing of variables to minimize Boolean functions

Methods for minimizing the logical functions, for example, the Quine method, the Quine-McCluskey method, an analytical method, Karnaugh maps, a Mahoney map method, Veitch's diagram method, hypercube method, Harvard method, an unpaired graph method, a combining indices method, and others, require the movement of the minimization principle to auxiliary objects, such as implicant tables, algebraic expressions, Karnaugh maps, Mahoney maps, Veitch's diagrams, graphs, etc.

The apparatus of equivalent figurative transformations is based on the properties of the binary block-schemes with repetition, which are essentially the truth tables of the assigned Boolean functions. This makes it possible to concentrate the principle of minimization within a truth table and, thus, disregard auxiliary objects. The information capacity of the method of figurative transformations makes it possible easy enough to perform manual minimization of 4–, ..., 10-variable Boolean functions.

The Boolean function minimization using a method of figurative transformations manually requires certain abilities to identify in the structure of a truth table the 2-(n, b)-design and 2-(n, x/b)-design combinatorial systems. These very systems are used to carry out equivalent transformations by using the laws and axioma of the algebra of logic.

Increasing the efficiency to identify the 2-(n, b)-design and 2-(n, x/b)-design combinatorial systems, especially at an increase in the number of variable Boolean functions, is possible by applying the constructed algorithm (p. 7.3) followed by the subsequent automation of the search for the 2-(n, b)-design and/or 2-(n, x/b)-design systems in a first truth table.

The effectiveness of the consecutive application of logical operations of the super-gluing of variables and the simple gluing of variables to minimize Boolean functions is demonstrated by examples 17–19. Other examples of the Boolean function minimization are given in papers [6–9].

The following is the minimization of a 4-bit Boolean function F using a method of figurative transformations [7].

$$F=\begin{Bmatrix}0011,\ 0100,\ 0101,\ 0111,\\ 1001,\ 1101,\ 1110,\ 1111\end{Bmatrix}. \qquad (24)$$

Variant 1.

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

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

Variant 2.

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

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

The algebraic transforms, starting at the third matrix in the second variant:

3 matrix

$$\overline{x}_1x_3x_4+\overline{x}_1x_2\overline{x}_3+x_2x_4+x_1\overline{x}_3x_4+x_1x_2x_3,$$
$$\overline{x}_1x_3x_4+\overline{x}_1x_2\overline{x}_3=\overline{x}_1x_3x_4+\overline{x}_1x_2\overline{x}_3+\overline{x}_1x_2x_4,$$
$$x_1\overline{x}_3x_4+x_1x_2x_3=x_1\overline{x}_3x_4+x_1x_2x_3+x_1x_2x_4,$$

4 matrix

$$\overline{x}_1x_3x_4+\overline{x}_1x_2\overline{x}_3+\overline{x}_1x_2x_4+$$
$$+x_2x_4+x_1\overline{x}_3x_4+x_1x_2x_3+x_1x_2x_4,$$
$$\overline{x}_1x_2x_4+x_1x_2x_4+x_2x_4=\overline{x}_1x_2x_4+x_1x_2x_4,$$

5 matrix

$$\overline{x}_1x_3x_4+\overline{x}_1x_2\overline{x}_3+\overline{x}_1x_2x_4+x_1\overline{x}_3x_4+x_1x_2x_3+x_1x_2x_4,$$
$$\overline{x}_1x_3x_4+\overline{x}_1x_2\overline{x}_3+\overline{x}_1x_2x_4=\overline{x}_1x_3x_4+\overline{x}_1x_2\overline{x}_3,$$
$$x_1\overline{x}_3x_4+x_1x_2x_3+x_1x_2x_4=x_1\overline{x}_3x_4+x_1x_2x_3,$$

6 matrix

$$\overline{x}_1x_3x_4+\overline{x}_1x_2\overline{x}_3+x_1\overline{x}_3x_4+x_1x_2x_3.$$

Variant 3.

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

Three variants for minimizing the assigned function (24) yield the same result (25):

$$F=\overline{x}_1x_2\overline{x}_3+\overline{x}_1x_3x_4+x_1\overline{x}_3x_4+x_1x_3x_4. \qquad (25)$$

In the first and second variants, the minimization was carried out by the sequence of the logical operations of the

super-gluing of variables and the simple gluing of variables. The third variant of minimization was carried out by means of a single logical operation – the simple gluing of variables.

The third variant to minimize function (24) is a simpler procedure compared to the first and second variants. Thus, the sequence of the logical operations of super-gluing the variables and the simple gluing of variables to minimize logical functions is not always optimal. However, logical functions, similar to (24), are not common. It should be noted that over the entire time of the development of the method of figurative transformations [6–9], function (24) is the only instance of this kind. Therefore, the sequence of the logical operations of super-gluing the variables and the simple gluing of variables remains a strategic procedure to minimize Boolean functions.

The weak side of the described method is related to the limited practice of using the equivalent figurative transformations for the process of minimizing Boolean functions with the subsequent production of corresponding computing components. The negative internal factors, inherent in the process of the Boolean function minimization by the specified method, are the need for additional time costs for establishing the protocols for the minimization of Boolean functions with the subsequent creation of a rule library for the algebra of logic that could illustrate the relevant figurative transformations.

The clarity of figurative transformations makes it possible to execute the manual method of the Boolean function minimization (using a mathematical editor, for example MathType v. 6.9) approximately in the range of up to ten input variables. In a general case, the identification of the combinatorial systems-images 2-(n, b)-design and 2-(n, x/b)-design in the structure of a truth table manually requires certain abilities, especially at an increase in the number of variables of Boolean functions. Consequently, the prospect of further research into the Boolean function minimization by the method of figurative transformations may be related to software to search for the 2-(n, b)-design and/or 2-(n, x/b)-design systems in the combinatorial structure of a truth table for the assigned logical function.

8. Conclusions

1. The comparison of patterns in the process of minimizing the logical functions by using the 2-(n, b)-design and 2-(n, x/b)-design combinatorial structures allows us to conclude that the 2-(n, b)-design system and the consistent alternation of the logical operations of super-gluing the variables (if such an operation is possible) and the simple gluing of variables, in the first matrix (a truth table), ensures the optimal solution by the criterion of all revealed combinatorial images in the truth table, which can participate in the minimization of Boolean functions. The reliability of the minimization result is ensured by the procedure of minimizing the assigned logical function on the complete truth table, followed by the subsequent selection of a minimization result in the DNF or CNF representation.

2. The algorithm for automating the process of minimizing the logical functions based on the method of figurative transformations within the initial combinatorial system is similar to the procedure of searching for the intervals $I(\alpha, \beta)$ in the Boolean space B^n, which are assigned by a pair of Boolean vectors α and β, such that $\alpha \preccurlyeq \beta$. The internal components in the interval $I(\alpha, \beta)$ correspond to a complete combinatorial system with repeated 2-(n, b)-design. The interval external components are determined by calculating the number of zeros or unities in the columns in the truth table of a logical function.

The optimal solution for the minimization of Boolean functions is based on the primary application of the operation of super-gluing the variables within the truth table. For the possible application of the operation of super-gluing the variables, one uses the algorithm to search for the 2-(n, b)-design system in the structure of a truth table for the assigned function (p. 7. 3). The algorithm allows for the automated search for the intervals or combinatorial systems 2-(n, b)-design in the structure of the truth table and is a tool to automate the process of minimizing the logical functions by the method of figurative transformations.

3. The effectiveness of the combined alternation of the protocols for minimizing the Boolean functions is demonstrated by examples borrowed from works by other authors, for comparison: example 17 [21] – minimizing a 4-bit Boolean function, example 18 [22] – minimizing a 4-bit Boolean function, example 19 [23] – minimizing a 4-bit Boolean function. Given the above examples, the effectiveness of the consistent application of the logical operations of super-gluing the variables and the simple gluing of variables to minimize Boolean functions allows us to argue about the feasibility of the application of the specified sequence of logical operations in the procedures of minimizing logical functions as the specified sequence of logical operations can:

– maintain the automated search for the 2-(n, b)-design combinatorial systems in the structure of a truth table for the assigned logical function, followed by the subsequent minimization of logical functions by the method of figurative transformations;

– improve the productivity of the process of logical function minimization.

The algorithm to minimize Boolean functions by a method of the optimum combination of the sequences of figurative transformations creates a prerequisite for the simplified automation of calculations in the method of figurative transformations.

References

1. Curtis, H. A. (1962). A new approach to the design of switching circuits. N.J.: Princeton, Toronto, 635.
2. Mayorov, S. A. (Ed.) (1972). Proektirovanie tsifrovyh vychislitel'nyh mashin. Moscow: Vysshaya shkola, 344.
3. Pospelov, D. A. (1974). Logicheskie metody analiza i sinteza shem. Moscow: Energiya, 368.
4. Zakrevskiy, A. D. (1981). Logicheskiy sintez kaskadnyh shem. Moscow: Nauka, 416.
5. Rytsar, B. E. (1997). Metod minimizatsii bulevyh funktsiy. Problemy upravleniya i informatiki, 2, 100–113.
6. Riznyk, V., Solomko, M. (2018). Minimization of conjunctive normal forms of boolean functions by combinatorial method. Technology Audit and Production Reserves, 5 (2 (43)), 42–55. doi: https://doi.org/10.15587/2312-8372.2018.146312

7. Riznyk, V., Solomko, M. (2017). Minimization of Boolean functions by combinatorial method. Technology Audit and Production Reserves, 4 (2 (36)), 49–64. doi: https://doi.org/10.15587/2312-8372.2017.108532
8. Riznyk, V., Solomko, M. (2017). Application of super-sticking algebraic operation of variables for Boolean functions minimization by combinatorial method. Technology Audit and Production Reserves, 6 (2 (38)), 60–76. doi: https://doi.org/10.15587/2312-8372.2017.118336
9. Riznyk, V., Solomko, M. (2018). Research of 5-bit boolean functions minimization protocols by combinatorial method. Technology Audit and Production Reserves, 4 (2 (42)), 41–52. doi: https://doi.org/10.15587/2312-8372.2018.140351
10. Dan, R. (2010). Software for The Minimization of The Combinational Logic Functions. The Romanian Review Precision Mechanics, Optics & Mchatronics, 37, 95–99. Available at: https://pdfs.semanticscholar.org/b881/59ffd963e4cb44d513eba58230e-56f1e5605.pdf
11. Huang, J. (2014). Programing implementation of the Quine-McCluskey method for minimization of Boolean expression. arXiv. Available at: https://arxiv.org/ftp/arxiv/papers/1410/1410.1059.pdf
12. Nosrati, M., Karimi, R. (2011). An Algorithm for Minimizing of Boolean Functions Based on Graph DS. World Applied Programming, 1 (3), 209–214.
13. Solairaju, A., Periyasamy, R. (2011). Optimal Boolean Function Simplification through K-Map using Object-Oriented Algorithm. International Journal of Computer Applications, 15 (7), 28–32. doi: https://doi.org/10.5120/1959-2621
14. Boyar, J., Peralta, R. (2010). A New Combinational Logic Minimization Technique with Applications to Cryptology. Lecture Notes in Computer Science, 178–189. doi: https://doi.org/10.1007/978-3-642-13193-6_16
15. Chen, Z., Ma, H., Zhang, Y. (2014). A Rapid Granular Method for Minimization of Boolean Functions. Lecture Notes in Computer Science, 577–585. doi: https://doi.org/10.1007/978-3-319-11740-9_53
16. Papakonstantinou, K. G., Papakonstantinou, G. (2018). A Nonlinear Integer Programming Approach for the Minimization of Boolean Expressions. Journal of Circuits, Systems and Computers, 27 (10), 1850163. doi: https://doi.org/10.1142/s0218126618501633
17. Kabalan, K. Y., El-Hajj, A., Fakhreddine, S., Smari, W. S. (1995). Computer tool for minimizing logic functions. Computer Applications in Engineering Education, 3 (1), 55–64. doi: https://doi.org/10.1002/cae.6180030108
18. Bulevy konstanty i vektory. Available at: https://studfile.net/preview/4243601/
19. Nazarova, I. A. (2012). Dyskretnyi analiz. Donetsk, 277. Available at: http://ea.donntu.edu.ua/bitstream/123456789/27328/1/%D0%9D%D0%9F_%D0%94%D0%90_%D0%A3%D0%9A%D0%A0%20%28%D0%9F%D0%BE%D0%BB%D0%BD%D0%B8%D0%B9%29.pdf
20. Samofalov, K. G., Romlinkevich, A. M., Valuyskiy, V. N., Kanevskiy, Yu. S, Pinevich, M. M. (1987). Prikladnaya teoriya tsifrovyh avtomatov. Kyiv, 375. Available at: http://stu.scask.ru/book_pta.php?id=62
21. Bonal, D. (2013). Karnaugh and Mahoney – Map Methods for Minimizing Boolean Expressions. Available at: http://davidbonal.com/karnaugh-and-mahoney-map-methods-for-minimizing-boolean-expressions/
22. Filippov, V. M., Manohina, T. V., Evdokimov, A. A., Zayats, D. S. (2016). Minimizatsiya funktsiy algebry logiki metodom nenapravlennogo grafa. Mezhdunarodniy zhurnal prikladnyh i fundamental'nyh issledovaniy, 8 (4), 509–511. Available at: https://applied-research.ru/ru/article/view?id=10112
23. Kumar, R., Rawat, S. (2016). Cubical Representation and Minimization through Cubical Technique A Tabular Approach. International Journal of Applied Engineering Research, 11 (7), 4822–4829.