PACS 681.3.14/21:519.713

[0000-0003-3642-2849] **S. V. Burmistrov,** *Ph.D. (Engineering)*,
e-mail: sergij.burmistrov@ukr.net
[0000-0002-2093-1270] **V. I. Khotunov,** *Ph.D. (Pedagogy)*, *Associate Professor,*
[0000-0001-6314-5838] **M. V. Zakharova,** *Ph.D. (Engineering)*, *Associate Professor,*
[0000-0001-9864-3386] **S. L. Mykhaylyuta,** *Ph.D. (Engineering)*, *Associate Professor,*
[0000-0002-0248-0461] **M. V. Liuta,** *Head of the Software Engineering Department*
Cherkasy State Business-College
Vyacheslav Chornovil st., 243, Cherkasy, 18028, Ukraine

INDEX METHOD OF MINIMIZATION OF BOOLEAN FUNCTIONS

*The paper presents a new method of minimization that implements the Boolean function in classical minimal form of representation by means of a directed selection of possible ways of minimization according to the criteria of a necessary and sufficient condition – the index method. This method is a continuation of evolutionary development of methods of minimization by reducing the value of the basis coefficient **K**: the method of minimization by parts, the method of parallel decomposition by reducing **K**, the matrix method of parallel decomposition. The evolution of methods by reducing the value of the basis coefficient **K** is a thorough study of the structure and structural organization of a set of Boolean functions, a detailed analysis of the strengths and weaknesses of already existing previous variants of the methods, the identification of critical places that significantly slow down the minimization process, and the search for alternative ways of accelerating the minimization process. The index method is developed based on the use of a new way of recording individual Boolean functions in the form of indexes of significant rows of the truth table. Thanks to this form of recording, it has been possible both to realize the strengths of the previous methods and significantly improve the weak stages of the previous methods, which in general gives a big gain in the time of minimization. The advantage of the method is two-stage minimization of the process, which makes it possible not to use the directed sorting criterion directly. When forming a complete list of elements, the elements of the final answer are immediately obtained without specifying intermediate results. Structural elements of the method – a complete set of possible elements of the final answer for Boolean functions, containing one number of arguments for the value of the basic coefficient K=1...n, are formed even before the beginning of the execution of the method. and are used as a table value. When implementing the method, only units without zeros are processed in the columns of the truth table, which reduces the number of processing objects. The method is implemented by two-level column processing – checking necessary and sufficient conditions. The machine implementation of the method uses parallelization of the minimization process. All this significantly reduces the minimization time – the main value that distinguishes this method from others. The developed method of minimization is one of the constituent parts of the creation of the software code, which is the basis of the development of a fractal computer. The main feature of a fractal computer is the presence of fractal (non-smooth) functions in its software code, which will radically expand its capabilities in certain areas of computing. To date, none of modern computers uses these functions in the program code.*

Keywords: *method of minimization of Boolean functions, matrix method of parallel decomposition, resulting lines of Boolean function, parallelization of minimization process, basic coefficient **K**, fractal computer.*

Introduction. An urgent problem in the practical development of various types of digital systems remains the need to synthesize combinational circuits described by Boolean functions (***BF***) containing a large number of arguments.

A classical combinational circuit (***CS***) is a logic circuit, the values of output signals of which at an arbitrary point in time are completely determined by the values of signals at its inputs [1, 2].

The optimal configuration of the ***CS*** is determined by a set of criteria for the complexity of ***CS*** implementation [3, 4]. They determine the complex characteristics of ***CS***. Optimization of one of the set of criteria, as a rule, worsens the value of other criteria. There are two fundamental criteria that underlie the structure of ***CS*** and to

DOI: 10.24025/2306-4412.2.2023. 273763

the meaning of which should be paid special attention. This is the operation time of ***CS*** (***T_CS***) and the number of conditional transistors that make up ***CS*** (***KUT_CS***).

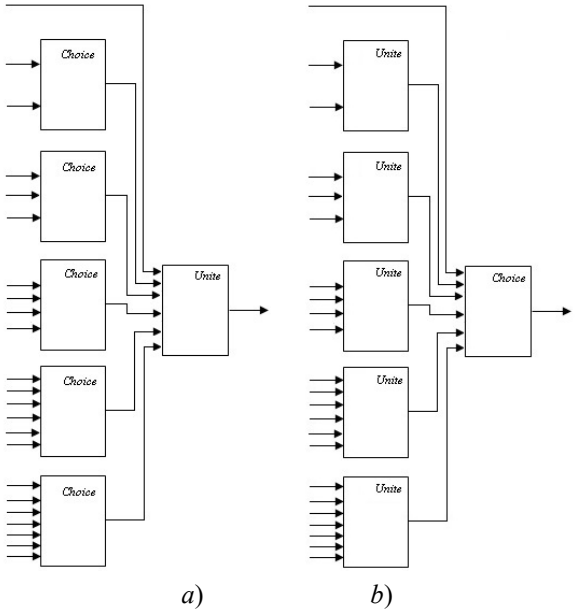
T_CS affects the performance of a digital device – the smaller the value of ***T_CS***, the greater its performance is.

KUT_CS affects the dimensions of digital device and energy intensity – electrical energy consumed by the device. The smaller the value of ***KUT_CS***, the smaller the final circuit of the device is in terms of geometric dimensions, the less electrical energy it consumes.

Historically, ***CS*** synthesis methods are usually divided into two large classes [1]: two-level and multi-level ones. With two-level synthesis, the signal passes from the input to the output of combinational circuit through two levels of logic elements. With multi-level synthesis, a complex structure is created, the nodes of which are analogues of two-level schemes.

One of the main stages of the synthesis of combination schemes, which affects the value of ***T_CS*** and ***KUT_CS*** coefficients, is the stage of BF minimization, which logically describes the operation of ***CS***. The consequence of solving ***BF*** minimization problem [2, 3] is the synthesis of two-level or multi-level ***CS***.

The result of ***BF*** minimization is the implementation of a two-level ***CS*** in one of the implementation options (Figure 1) [5].



a) *b*)

Figure 1. Options for implementing a two-level CS:
a) with a uniting (*Unite_LE*) logic element;
b) with a choice (*Choice_LE*) logic element

In ***CS***, the selection logic element (***Choice_LE***) can be one of the types of logic elements, which, according to its logic function, selects one of the possible variants of combinations of input signals.

The uniting logic element (***Unite_LE***) in ***CS*** can be one of the types of logic elements, which by its logic function selects several possible variants of combinations of input signals.

The result of ***BF*** minimization in case *a*) is the ***BF*** in classical form of representation (***CFP***) or polynomial form of Reed-Müller representation (***RMFP***). ***CS***s built on the basis of *BF* specified in ***KFP*** or ***RMFP*** have the best values of ***T_CS*** criterion and differ in maximum speed.

A successful result of ***BF*** minimization, which leads to the creation of a multi-level ***CS*** (Figure 2), is the orthogonal form of ***BF*** representation (***ORFP***) [3]. Multi-level ***KS***, as a rule, have better indicators of ***KUT_CS*** criterion [6, 7].

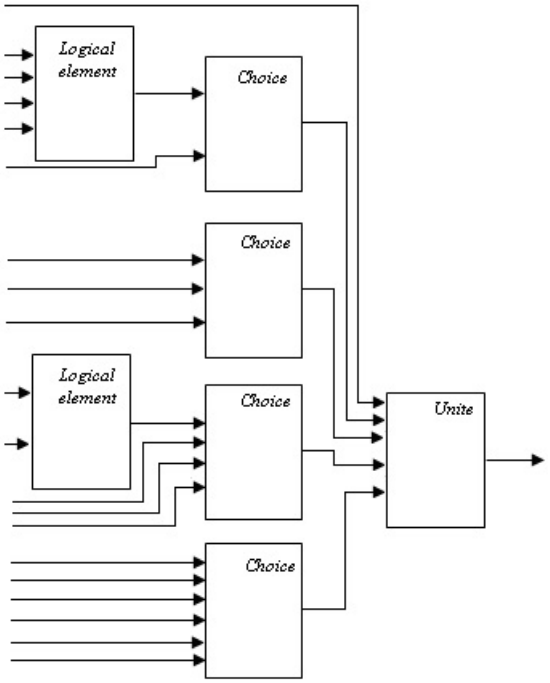


Figure 2. Implementation of a multi-level CS

As a result, when choosing the construction of either two-level or multi-level ***CS***, a contradiction arises when constructing a ***CS*** – either to build a scheme with greater work productivity, or to build an energy-saving scheme with minimal geometric dimensions [8, 9].

DOI: 10.24025/2306-4412.2.2023. 273763

An actual problem in the development of ***BF*** minimization methods containing a large number of arguments is the construction of methods that lead to a complete answer in less time.

The purpose of the research is to find a ***BF*** minimization method with a large number of arguments, given in classical form of representation, which is practically implemented in the minimum possible time.

CFP is the most implemented form of representation in computer electronics. The advantage of the minimal form of ***BF*** in ***CFP*** compared to other forms of representation of ***BF*** is that ***CS*** has the best indicators of the value of ***T_CS*** criterion. At the same time, the value of ***KUT_CS*** criterion is much better than in other forms of ***BF*** representation.

There is a large number of methods for obtaining the minimal form of ***BF*** in ***CFP***, a simple list of which is far from a trivial task [13, 14, 15]. All these methods, as a rule, provide a complete solution to the problem. However, all of them have one significant drawback – the speed of obtaining a complete result, the minimal form of ***BF*** in ***CFP***, in them falls in an avalanche even with a slight increase in the number of arguments containing ***BF***, even when using powerful hardware and software tools designed for obtaining the final answer.

In order to understand the search for the direction of the optimal minimization method, the following questions and answers to the questions that will arise during their investigation should be considered:

– what is ***BF***, what is its structure from the point of view of its minimization;

– the problem of arranging and classifying ***BF***, determining the criteria for combining ***BF*** into a single set;

– interdependence and subordination between ***BF*** containing the same and different number of arguments;

– arrangement of studies of the properties of ***BF*** sets according to their key properties;

– implementation of ***BF*** properties and properties of ***BF*** sets to build an optimal minimization method according to the given criteria of the complexity of ***CS*** implementation.

The implementation of the set tasks is based on a detailed analysis of already implemented methods, identification of problematic stages of implementation and methods of accelerating these stages.

BF is a type of logical function defined by a truth table (***TI BF***) (Table 1). Therefore, the main object of research in the article is ***TI BF*** as the main element on the basis of which the ***BF*** minimization method is built.

Table 1. **Definition of a Boolean function using a truth table**

Line number	x_{n-1}	...	x_1	x_0	F
0	0	…	0	0	f_0
1	0	…	0	1	f_1
…	…	…	…	…	…
2^{n-2}	1	…	1	0	f_{m-2}
2^{n-1}	1	…	1	1	f_{m-1}

To order the elements of the set ***BF***, it is customary to consider the column of the result ***F*** in ***TI BF*** as the number of the Boolean function $\boldsymbol{F}=\{f_{m-1}f_{m-2}\ldots f_1f_0\}$ (see Table 1), where the digits f_i can take the value 0 or 1. Depending on the number of arguments n of the Boolean function, ***BF*** number consists of 2^n digits. Accordingly, the set BF containing the same number of arguments contains 2^{2^n} BF (Table 2).

Table 2. **An avalanche of the number of Boolean functions when the number of their arguments increases**

№ in order	Quantity arguments	The quantity of digits in Boolean function number	The quantity of Boolean functions containing the same number of arguments
1	1	2	4
2	2	4	16
3	3	8	256
4	4	16	65 536
5	5	32	4 294 967 296

A fundamental property of ***BF***, which combines Boolean functions containing a different number of arguments, is the Shannon decomposition. Using the Shannon decomposition, an arbitrary ***BF*** containing n arguments can be represented by a ***BF*** containing n-1 arguments in n ways [10].

In general case of the ***CFP***, the Shannon decomposition has the form:

DOI: 10.24025/2306-4412.2.2023. 273763

$$y = f(x_{n-1},...,x_i,...,x_0) =$$
$$= x_i \cdot \Phi_1(x_{n-1},...,x_{i+1},x_{i-1},...,x_0) \vee \overline{x_i} \cdot \Phi_0(x_{n-1},...,x_{i+1},x_{i-1},...,x_0).$$

An arbitrary ***BF*** with n arguments is written in the form of a logical disjunctive series containing two terms, each of which contains a ***BF*** with n-1 arguments [10].

If the decomposition is performed several times, it will lead to the formation of a logical disjunctive series (1), where each member of the series consists of two parts, which, by analogy with geometry, is called the basic part Q_i, containing k arguments ($0 \le k \le n$-1) and informational part Φ_i. Information part Φ_i is a ***BF*** containing n-k arguments.

$$y = f(x_{n-1},...,x_i,...,x_0) = \bigvee_{i=1}^{2^k-1} Q_i \wedge \Phi i \qquad (1)$$

In order to show the degree of decomposition of the members of ***BF*** series, let's enter the value of the base coefficient ***K*** – the degree of decomposition of the member of logical disjunctive series ***BF***. It indicates the number of arguments contained in the base part in this member.

It is this property that unites BF into a complete set of Boolean functions and is decisive when studying the properties of ***BF***.

If we consider the interdependence between Boolean functions in the complete set ***BF*** through the Shannon decomposition, it can be shown that ***BF*** has a pronounced fractal structure.

An arbitrary Boolean function containing n arguments is the result of the mutual composition of n pairs containing n-1 arguments.

In other words, an arbitrary ***BF*** stores information about all parent functions starting from the primary parent ***BF***s containing 1 argument – "logical 0" (00), "logical 1" (11) and BF non-constants (01, 10).

As a result of the decomposition of an arbitrary ***BF*** in informational part of Φ_i, the mother ***BF***s that are in the genetic code of the specified ***BF*** are obtained. It is maternal ***BF*** that affects the degree of ***BF*** minimization. If, at a certain stage of the decomposition, the parent ***BF***, the informational part of Φ_i, is a "logical 0", then the value of the specified member of logical disjunctive series is zero. If, at a certain stage of the decomposition, the parent ***BF*** is a "logical 1", then the value of the indicated member of logical disjunctive series is equal to the value of the basic part Q_i.

Only those Boolean functions that contain the parent ***BF*** "logical 0" or "logical 1" in their genetic code are minimized in ***CFP***.

Presenting main material. For a long time, minimization methods built on the basis of ***BF*** decomposition in ***CFP*** were irrelevant, since they did not compete with classical ***BF*** minimization methods. For Boolean functions containing a large number of arguments, the situation is radically different – the time to obtain the minimization result by classical method increases exponentially.

The fractal character of ***BF*** structure indicates that the solution of a typical problem of processing fractal structures, including ***BF*** minimization, is solved by a directed search with the application of certain path selection criteria using ***BF*** decomposition. Solving this problem is a search for the shortest way to perform the decomposition of the members of a disjunctive logical series until informational parts Q_i in all members of the series are equal to either the value "logical 0" or "logical 1".

As practical experience shows, solving this problem by directly performing the decomposition in the shortest way, the problem is generally unsolvable. The reason is that there is no clear criterion for finding the shortest decomposition path.

Another way of using the ***BF*** decomposition to create an efficient minimization method can be suggested. When decomposing BF, three groups of logical terms can be formed:

– logical terms in which informational part Fi is not equal to "logical 1" and not equal to "logical 0";

– logical terms in which informational part of Φ is equal to "logical 0";

– logical terms in which informational part of Fi is equal to "logical 1".

Only the third group of logical terms is of practical interest, and not all, but only those in which the value of the coefficient ***K*** is the maximum possible.

Due to the lack of a clear criterion for the optimal direction of the decomposition, it is advisable to implement the minimization method by replacing the direct decomposition of BF with the solution of the two-stage problem. At the first stage, all terms of logical series of the third group are searched for the values of the basic coefficient ***K***=1,2,...,n-1 (namely, the list of the corresponding basic parts of Q_i).

DOI: 10.24025/2306-4412.2.2023. 273763

At the second stage, a final minimized logic series is constructed from the specified Q_i according to the full coverage criterion.

As a result, either 1 or several solutions are obtained by the decomposition method – the minimal forms of the specified ***BF*** in ***CFP***.

The development of optimal minimization decomposition methods is implemented in two directions:

1. Search for terms of the third group of logical series by increasing the value of the basic coefficient ***K.***

2. Search for terms of the logical series of the third group by reducing the value of the basic coefficient ***K.***

Each direction has its strengths and weaknesses. The authors of the article consider the direction of searching for terms of logical series of the third group to be more promising by reducing the value of the basic coefficient ***K*** from the point of view of the time of obtaining the final result. On the long way to obtaining a result, the authors have proposed several minimization methods that realize the possibilities of this direction (in chronological order of creation), namely:

1. The method of minimization by parts [10, 11, 12].

2. Parallel decomposition by reducing the value of the base coefficient ***K*** [13, 14].

3. Matrix method of parallel decomposition [15].

Each of the subsequent obtained methods is an evolutionary development of the previous ones. They were developed through a detailed analysis of the strengths and weaknesses of previous methods, identification of critical points that significantly slow down the minimization process, and search for alternative ways to speed up the minimization process.

The index method of minimization of Boolean functions is an evolutionary continuation of the previous methods. It is developed based on the analysis of the matrix method (Figure 3), takes into account the concept and the strengths of the previous methods and significantly accelerates the minimization process.

The directed sorting of the elements of the ready answer, which is the basis of this method, involves the use of sorting criteria that indicate the direction of the shortest movement to the result during sorting. A detailed analysis of selection criteria makes it possible to divide them into two groups:

– criteria of the necessary condition of the search;

– criteria of a sufficient condition of the search.

The criteria of a sufficient condition of search are decisive. The minimization method can be built only on the criteria of a sufficient condition of search. They accurately indicate whether the basis part Q_i of disjunctive logic series is an element of the final answer with the maximum value of the basis coefficient ***K***. The implementation of the minimization method in ***CFP*** makes it possible to obtain a guaranteed full final answer. The disadvantage of this approach is that the verification of criteria for the sufficient condition of sorting is quite lengthy in time, which in turn slows down the entire minimization process.

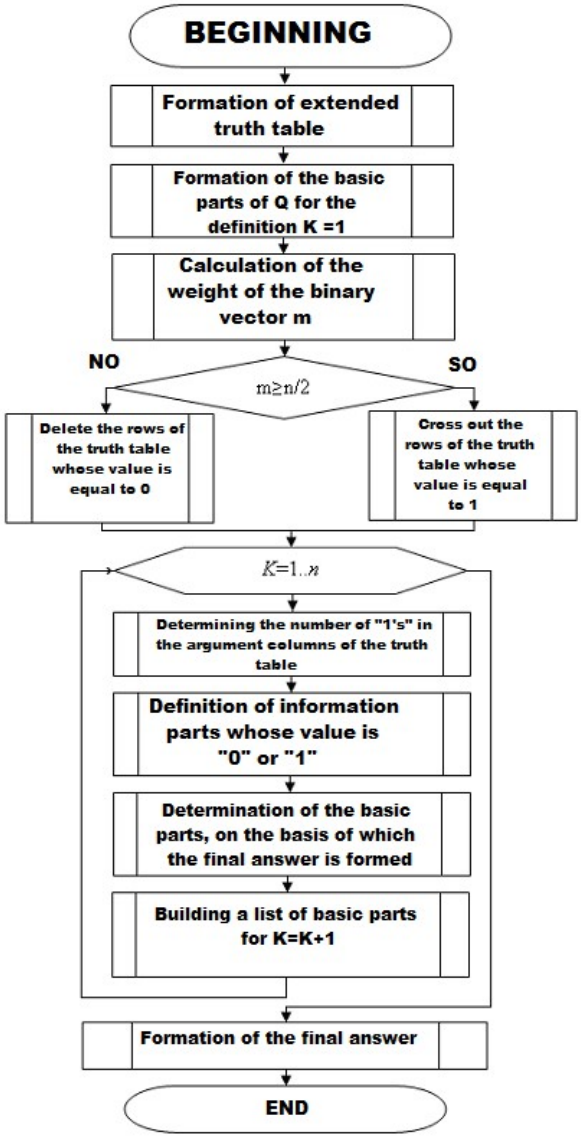


Figure 3. Algorithm for implementation of the matrix method of parallel decomposition

DOI: 10.24025/2306-4412.2.2023. 273763

The necessary search condition criteria are not determinative when searching for elements of the final answer, but they specify exactly which basis part of Q_i is not an element of the final answer. As a rule, these criteria are not obvious when used for Boolean functions with a small number of arguments. Their role and number grow precisely with the growth of arguments in ***BF***. The advantage of using the necessary condition criterion is that the check is several orders of magnitude faster than the sufficient condition check.

It is the search and application of the criteria for non-necessary condition of directed search that makes it possible to significantly speed up the minimization method.

The index method of minimization of Boolean functions is developed on the basis of and taking into account the features of the matrix method of parallel decomposition [15].

When analyzing the implementation of the matrix method (see Figure 3), it has been found that the slowest stage in the execution time of the specified method is the stage "Determination of the number of units in the columns of TI arguments". As a rule, it is necessary to sort 2^{n-1} digits to calculate the sum of the criterion of the sufficient sorting condition. As the number of arguments n increases, this number grows exponentially. What is important, as a rule, when counting, most of numbers are zeros, numbers that do not affect the result, but significantly reduce the counting time.

The index minimization method is designed to exclude these zeros from the counting process and, accordingly, to speed up the search for elements that meet the criterion.

It is advisable to explain the essence of the method using an example, describing in detail the ***BF*** minimization algorithm. Let *BF* be given, having the number $1\ 467\ 447\ 187_{10}$. The binary digital code ***BF*** consists of 5 arguments (n=5) and is

$0101\ 0111\ 0111\ 0111\ 0111\ 0111\ 0111\ 0101_2$.

Stage 1. Formation of a complete list of elements of the final answer.

1. Write all lines of ***TI BF*** $f^{(5)}_{1467447157}$ (Table 3), the value of which in the column of the result is equal to 1. These are the resulting lines of ***TI BF*** for the value of the basic coefficient ***K***=1.

Write lines in the table of result lines in the form of sets of 0 and 1 line arguments – digital binary codes of TI BF lines {00000, 00100, 00100, 00101, 00110, 01000, 01001, 01010, 01100, 01101, 01110, 100001, 10000 10010, 10100, 10101, 10110, 11000, 11001, 11010, 11100, 11110} and the corresponding row indices {00,02,04,05, 06,08,09,10, 12,13,14,16, 17,18,20,21, 22,24,25,26, 28,30}. As a result, the index digital code of the Boolean function is

3028262524222120181716141312100908060504020 0

Table 3. **Truth table of the Boolean function** $f^{(5)}_{1467447157}$

Strings of arguments					y	The resulting string (K=1)	Row index
x_5	x_4	x_3	x_2	x_1			
0	0	0	0	0	1	00000	00
0	0	0	0	1	0		
0	0	0	1	0	1	00010	02
0	0	0	1	1	0		
0	0	1	0	0	1	00100	04
0	0	1	0	1	1	00101	05
0	0	1	1	0	1	00110	06
0	0	1	1	1	0		
0	1	0	0	0	1	01000	08
0	1	0	0	1	1	01001	09
0	1	0	1	0	1	01010	10
0	1	0	1	1	0		
0	1	1	0	0	1	01100	12
0	1	1	0	1	1	01101	13
0	1	1	1	0	1	01110	14
0	1	1	1	1	0		
1	0	0	0	0	1	10000	16
1	0	0	0	1	1	10001	17
1	0	0	1	0	1	10010	18
1	0	0	1	1	0		
1	0	1	0	0	1	10100	20
1	0	1	0	1	1	10101	21
1	0	1	1	0	1	10110	22
1	0	1	1	1	0		
1	1	0	0	0	1	11000	24
1	1	0	0	1	1	11001	25
1	1	0	1	0	1	11010	26
1	1	0	1	1	0		
1	1	1	0	0	1	11100	28
1	1	1	0	1	0		
1	1	1	1	0	1	11110	30
1	1	**1**	1	1	0		

DOI: 10.24025/2306-4412.2.2023. 273763

2. Form ***TI*** arguments of ***BF*** $f(5)$. For the convenience and compactness of the image on paper, turn ***TI*** by 90^0 (Table 4, *a*, *b*, *c*) so that the columns of ***TI*** become rows. The index digital code of argument lines is obtained by writing out the indices of the units of the binary code. The list of arguments of the truth table of the Boolean function can be written in different forms:

1) In the form of the row number of the argument of the truth table of the Boolean function (see Table 4, *a*).

Table 4, *a*. **A list of truth table arguments of a Boolean function in the form of an argument line number**

№ in order	№ of arguments	Argument line number
1	x_1	1010 1010 1010 1010 1010 1010 1010 1010
2	x_2	1100 1100 1100 1100 1100 1100 1100 1100
3	x_3	1111 0000 1111 0000 1111 0000 1111 0000
4	x_4	1111 1111 0000 0000 1111 1111 0000 0000
5	x_5	1111 1111 1111 1111 0000 0000 0000 0000

2) In the form of an index number code of argument lines (see Table 4, *b*).

Table 4, *b*. **A list of truth table arguments of a Boolean function in the form of an index numeric code of argument strings**

№ in order	№ of arguments	Index numeric code of argument strings
1	x_1	31292725232119171513110907050301
2	x_2	31302726232219181514111007060302
3	x_3	31302928232221201514131207060504
4	x_4	31302928272625241514131211100908
5	x_5	31302928272625242322212019181716

The index code of argument lines is obtained by writing out the indices of ***TI*** argument column units from bottom to top.

3) In the form of a digital binary code of argument lines (see Table 4, *c*).

Table 4, *c*. **A list of truth table arguments of a Boolean function in the form of a digital binary code**

№ in order	№ of arguments	Digital binary code
1	x_1	****1
2	x_2	***1*
3	x_3	**1**
4	x_4	*1***
5	x_5	1****

3. On the basis of the truth table of arguments of the Boolean function, which contains 5 arguments, build a table of possible elements of the final answer – the basic parts Q_i of the terms of disjunctive logical series, in which the information part Φ_i is equal to "logical 1" for the value base coefficient ***K***=1, by adding additional rows with inverse arguments to the truth table of arguments of the Boolean function (Table 5).

Table 5. **The table of possible elements of the final answer of the truth table of the Boolean function at *K*=1**

№ in order	Kind of literal	A complete list of possible elements of the final truth answer of the Boolean function at K=1	
		Index digital code	Binary digital code
0	x_1	31292725232119171513110907050301	****1
1	$\overline{x_1}$	30282624222018161412100806040200	****0
2	x_2	31302726232219181514111007060302	***1*
3	$\overline{x_2}$	29282524212017161312090805040100	***0*
4	x_3	31302928232221201514131207060504	**1**
5	$\overline{x_3}$	27262524191817161110090803020100	**0**
6	x_4	31302928272625241514131211100908	*1***
7	$\overline{x_4}$	23222120191817160706050403020100	*0***
8	x_5	31302928272625242322212019181716	1****
9	$\overline{x_5}$	15141312111009080706050403020100	0****

4. Check the necessary condition: whether the elements of the full list of possible elements of the final truth answer of the Boolean function at ***K***=1 (see Table 5) belong to the set of elements of the resulting lines (see Table 3).

DOI: 10.24025/2306-4412.2.2023. 273763

When checking, the sign "*" replaces numbers "0" and "1". If the condition is not fulfilled, the possible element of the final truth answer does not belong to the set. It must be crossed out (Table 6).

In this case, the necessary condition is fulfilled for all possible elements of the final truth answer of the Boolean function at ***K***=1.

5. Check the sufficient condition: whether possible elements are really the elements of the final truth answer of the Boolean function at ***K***=1. Sufficient condition criterion: all index values of the index code of a possible element of the final answer must have the index code of the Boolean function (see Table 5).

Table 6. **Checking of the necessary condition of belonging to possible elements of the final truth answer of the Boolean function at *K*=1**

Possible elements of the final answer at *K*=1		The matching element of the result string (*K*=1)	Result
x_1	****1	00101	+
$\overline{x_1}$	****0	00000	+
x_2	***1*	00110	+
$\overline{x_2}$	***0*	00000	+
x_3	**1**	00100	+
$\overline{x_3}$	**0**	00000	+
x_4	*1***	01000	+
$\overline{x_4}$	*0***	00000	+
x_5	1****	10000	+
$\overline{x_5}$	0****	00000	+

Result: with ***K***=1, one element of the final answer is obtained: $\overline{x_1}$.

6. Analogous actions: formation and editing of the next full list of possible elements of the final answer, checking the necessary and sufficient conditions, must be performed at ***K***=2...5.

7. Form the list of indices of the result lines with ***K***=2. For this, it is necessary to cross out the list of elements of the final truth answer of the Boolean function at ***K***=1 from the list of resulting lines at K=1 (Table 7). In this case, the lines described by the mask (****0) should be crossed out from the table. As a result, 16 lines are crossed out. The result of the check is in Table 6: indices of the resulting rows at ***K***=2 constitute (05, 09, 13, 17, 21, 25).

Table 7. **List of result lines at *K*=2**

№ in order	The index of the resulting row	Result line (*K*=1)	Result
1	00	00000	-
2	02	00010	-
3	04	00100	-
4	05	00101	+
5	06	00110	-
6	08	01000	-
7	09	01001	+
8	10	01010	-
9	12	01100	-
10	13	01101	+
11	14	01110	-
12	16	10000	-
13	17	10001	+
14	18	10010	-
15	20	10100	-
16	21	10101	+
17	22	10110	-
18	24	11000	-
19	25	11001	+
20	26	11010	-
21	28	11100	-
22	30	11110	-

8. Check the necessary condition: whether the elements of the full list of possible elements of the final truth answer of the Boolean function at ***K***=2 belong to the set of elements of the resulting lines. When checking, the sign "*" replaces numbers "0" and "1". If the condition is not fulfilled, the possible element of the final truth answer does not belong to the set. It should be crossed out (Table 8).

DOI: 10.24025/2306-4412.2.2023. 273763

Table 8. **The table of verification of the necessary condition of possible elements of the final answer of the truth table of the Boolean function at *K*=2**

№ in order	Conjunctive set of arguments (*K*=2)	A complete list of possible elements of the final truth answer of the Boolean function at *K*=2		Result
		Index digital code	Binary digital code	
1	$\bar{x}_2\bar{x}_1$	2824201612080400	***00	-
2	$x_2\bar{x}_1$	3026221814100602	***10	-
3	$\bar{x}_3\bar{x}_1$	2624181610080200	**0*0	-
4	$x_3\bar{x}_1$	3028222014120604	**1*0	-
5	$\bar{x}_4\bar{x}_1$	2220181606040200	*0**0	-
6	$x_4\bar{x}_1$	3028262414121008	*1**0	-
7	$\bar{x}_5\bar{x}_1$	1412100806040200	0***0	-
8	$x_5\bar{x}_1$	3028262422201816	1***0	-
9	$\bar{x}_2x_1$	2925211713090501	***01	+
10	x_2x_1	3127231915110703	***11	-
11	$\bar{x}_3x_1$	2725191711090301	**0*1	+
12	x_3x_1	3129232115130705	**1*1	+
13	$\bar{x}_4x_1$	2321191707050301	*0**1	+
14	x_4x_1	3129272515131109	*1**1	+
15	$\bar{x}_5x_1$	1513110907050301	0***1	+
16	x_5x_1	3129272523211917	1***1	+
17	$\bar{x}_3\bar{x}_2$	2524171609080100	**00*	+
18	$x_3\bar{x}_2$	2928212013120504	**10*	+
19	$\bar{x}_4\bar{x}_2$	2120171605040100	*0*0*	+
20	$x_4\bar{x}_2$	2928252413120908	*1*0*	+
21	$\bar{x}_5\bar{x}_2$	1312090805040100	0**0*	+
22	$x_5\bar{x}_2$	2928252421201716	1**0*	+
23	$\bar{x}_3x_2$	2726191811100302	**01*	-
24	x_3x_2	3130232215140706	**11*	-
25	$\bar{x}_4x_2$	2322191807060302	*0*1*	-
26	x_4x_2	3130272615141110	*1*1*	-
27	$\bar{x}_5x_2$	1514111007060302	0**1*	-
28	x_5x_2	3130272623221918	1**1*	-
29	$\bar{x}_4\bar{x}_3$	1918171603020100	*00**	-
30	$x_4\bar{x}_3$	2726252411100908	*10**	+
31	$\bar{x}_5\bar{x}_3$	1110090803020100	0*0**	+
32	$x_5\bar{x}_3$	2726252419181716	1*0**	+
33	$\bar{x}_4x_3$	2322212007060504	*01**	+
34	x_4x_3	3130292815141312	*11**	+
35	$\bar{x}_5x_3$	1514131207060504	0*1**	+
36	x_5x_3	3130292823222120	1*1**	+
37	$\bar{x}_5\bar{x}_4$	0706050403020100	00***	+
38	$x_5\bar{x}_4$	2322212019181716	10***	+
39	$\bar{x}_5x_4$	1514131211100908	01***	+
40	x_5x_4	3130292827262524	11***	+

9. Check the sufficient condition: whether possible elements for which the necessary condition is fulfilled are really the elements of the final truth answer of the Boolean function at ***K***=2. Sufficient condition criterion: all index values of the index code of a possible element of the final answer must have the index code of the Boolean function (see Table 7).

Result: at ***K***=2, no possible element is an element of the final answer.

10. Form the list of result lines with ***K***=3. Since at ***K***=2 there are no elements of the final truth answer of the Boolean function, the list of resulting lines remains unchanged (Table 9).

Table 9. **List of resulting lines with *K*=3**

№ in order	The index of the resulting row	The resulting string (*K*=3)
1	05	00101
2	09	01001
3	13	01101
4	17	10001
5	21	10101
6	25	11001

11. Check the necessary condition: whether the elements of the full list of possible elements of the final truth answer of the Boolean function at ***K***=3 belong to the set of elements of the resulting lines (Table 10). For the sake of compactness of the record, the table lists only possible elements for which the necessary condition is fulfilled (out of 80 lines, 36 lines are shown).

12. Check the sufficient condition: whether possible elements for which the necessary condition is fulfilled are really the elements of the final truth answer of the Boolean function for ***K***=3 (see Table 10).

13. Formation of the list of result lines with ***K***=4. As a result of crossing out the final elements in the table of result lines, an empty table has been obtained. Therefore, there is no need to check the necessary and sufficient condition for ***K***=4 and ***K***=5.

DOI: 10.24025/2306-4412.2.2023. 273763

Table 10. **Table of verification of the necessary condition of possible elements of the final answer of the truth table of the Boolean function at *K*=3**

№ in order	Conjunctive set of arguments (*K*=3)	A complete list of possible elements of the final truth answer of the Boolean function at *K*=3	
		Index digital code	Binary digital code
1	$\bar{x}_3\bar{x}_2x_1$	25170901	**001
2	$x_3\bar{x}_2x_1$	29211305	**101
3	$\bar{x}_4\bar{x}_2x_1$	21170501	*0*01
4	$x_4\bar{x}_2x_1$	29251309	*1*01
5	$\bar{x}_4\bar{x}_3x_1$	19170301	*00*1
6	$\bar{x}_4x_3x_1$	23210705	*01*1
7	$x_4\bar{x}_3x_1$	27251109	*10*1
8	$x_4x_3x_1$	31291513	*11*1
9	$\bar{x}_4x_3\bar{x}_2$	21200504	*010*
10	$x_4\bar{x}_3\bar{x}_2$	25240908	*100*
11	$x_4x_3\bar{x}_2$	29281312	*110*
12	$\bar{x}_5\bar{x}_2x_1$	13090501	0**01
13	$x_5\bar{x}_2x_1$	29252117	1**01
14	$\bar{x}_5\bar{x}_3x_1$	11090301	0*0*1
15	$\bar{x}_5x_3x_1$	15130705	0*1*1
16	$x_5\bar{x}_3\bar{x}_1$	26241816	1*0*0
17	$x_5\bar{x}_3x_1$	27251917	1*0*1
18	$x_5x_3x_1$	31292321	1*1*1
19	$\bar{x}_5\bar{x}_3\bar{x}_2$	09080100	0*00*
20	$\bar{x}_5x_3\bar{x}_2$	13120504	0*10*
21	$x_5\bar{x}_3\bar{x}_2$	25241716	1*00*
22	$x_5x_3\bar{x}_2$	29282120	1*10*
23	$\bar{x}_5\bar{x}_4x_1$	07050301	00**1
24	$\bar{x}_5x_4x_1$	15131109	01**1
25	$x_5\bar{x}_4x_1$	23211917	10**1
26	$x_5x_4x_1$	31292725	11**1
27	$\bar{x}_5\bar{x}_4\bar{x}_2$	05040100	00*0*
28	$\bar{x}_5x_4\bar{x}_2$	13120908	01*0*
29	$x_5\bar{x}_4\bar{x}_2$	21201716	10*0*
30	$x_5x_4\bar{x}_2$	29282524	11*0*
31	$\bar{x}_5\bar{x}_4x_3$	07060504	001**
32	$\bar{x}_5x_4\bar{x}_3$	11100908	010**
33	$\bar{x}_5x_4x_3$	15141312	011**
34	$x_5\bar{x}_4\bar{x}_3$	19181716	100**
35	$x_5\bar{x}_4x_3$	23221918	101**
36	$x_5x_4\bar{x}_3$	27262524	110**

Result: with ***K***=3, 6 elements are elements of the final answer: $\bar{x}_4x_3\bar{x}_2$, $x_4\bar{x}_3\bar{x}_2$, $\bar{x}_5x_3\bar{x}_2$, $x_5\bar{x}_3\bar{x}_2$, $\bar{x}_5x_4\bar{x}_2$, $x_5\bar{x}_4\bar{x}_2$.

Stage 2. Formation of the final answer.

The essence of the stage is the construction of the minimal form of ***BF*** in the form of a disjunctive logical series, the terms of which are those basic parts of Q_i with ***K***=1…*n*, in which the informational part of Φ_i is equal to "1".

A disjunctive logical series is considered complete – a final answer if the list of indices of all its terms overlaps the indices of the Boolean function. Therefore, not all elements of the final answer can belong to the same disjunctive logical series. a Boolean function has multiple responses, if ***BF*** indices overlap in multiple ways.

1. To form the minimum form of ***BF***, you need to build a table (Table 11), the columns of which are the indices of ***TI BF*** lines, and the rows are the elements of the final answer [7]. If the element of the final answer contains a certain column index, "1" is placed in the corresponding cell. For each column, you need to calculate the sum of "1". Based on the line of sums, construct the answer.

Table 11.1. **Table for formation of the final answer (beginning)**

Elements of the final answer	Index digital code	Indexes of the resulting rows							
		30	28	26	25	24	22	21	20
$\bar{x}_1$	30 28 26 24 22 20 18 16 14 12 10 08 06 04 02 00	1	1	1		1	1		1
$\bar{x}_4x_3\bar{x}_2$	21 20 05 04							1	1
$x_4\bar{x}_3\bar{x}_2$	25 24 09 08				1	1			
$\bar{x}_5x_3\bar{x}_2$	13 12 05 04								
$x_5\bar{x}_3\bar{x}_2$	25 24 17 16				1	1			
$\bar{x}_5x_4\bar{x}_2$	13 12 09 08								
$x_5\bar{x}_4\bar{x}_2$	21 20 17 16							1	1
A number of sums		1	1	1	2	3	1	2	3

To form the minimum form of ***BF***, you need to build a table (Table 11.1-11.3 is divided into 3 parts to accommodate all the columns in the columns of the article), the columns of which are the indexes of ***TI BF*** lines, and the rows are the elements of the final answer. If the element of the final answer contains a certain column index, "1" is put in the corresponding cell. For each column, you need to calculate the sum "1". Based on the line of sums, construct the answer.

DOI: 10.24025/2306-4412.2.2023. 273763

Table 11.2. **Table for formation of the final answer (continuation)**

Elements of the final answer	Index digital code	Indexes of the resulting rows							
		20	18	17	16	14	13	12	10
$\overline{x}_1$	30 28 26 24 22 20 18 16 14 12 10 08 06 04 02 00	1	1		1	1		1	1
$\overline{x}_4x_3\overline{x}_2$	21 20 05 04	1							
$x_4\overline{x}_3\overline{x}_2$	25 24 09 08								
$\overline{x}_5x_3\overline{x}_2$	13 12 05 04						1	1	
$x_5\overline{x}_3\overline{x}_2$	25 24 17 16			1	1				
$\overline{x}_5x_4\overline{x}_2$	13 12 09 08						1	1	
$x_5\overline{x}_4\overline{x}_2$	21 20 17 16	1		1	1				
A number of sums		3	1	2	3	1	2	3	1

Table 11.3. **Table for formation of the final answer (end)**

Elements of the final answer	Index digital code	Indexes of the resulting rows						
		09	08	06	05	04	02	00
$\overline{x}_1$	30 28 26 24 22 20 18 16 14 12 10 08 06 04 02 00		1	1		1	1	1
$\overline{x}_4x_3\overline{x}_2$	21 20 05 04				1	1		
$x_4\overline{x}_3\overline{x}_2$	25 24 09 08	1	1					
$\overline{x}_5x_3\overline{x}_2$	13 12 05 04	1	1					
$x_5\overline{x}_3\overline{x}_2$	25 24 17 16							
$\overline{x}_5x_4\overline{x}_2$	13 12 09 08	1	1					
$x_5\overline{x}_4\overline{x}_2$	21 20 17 16							
A number of sums		3	4	1	1	2	1	1

Research results. The index minimization method is designed as a machine-based minimization method for the use in cluster computing systems to minimize systems of fully defined ***BF*** containing a large number of arguments. With the increase in the number of arguments in ***BF***, the number of possible elements of the final answer of ***BF*** for the values of the basic coefficient ***K***=1…*n* increases exponentially. Therefore, to expand the capabilities of the method and speed up the minimization process, parallelization of the minimization process is used for the slowest time steps. In this case, the minimization algorithm has the following form (Figure 4).

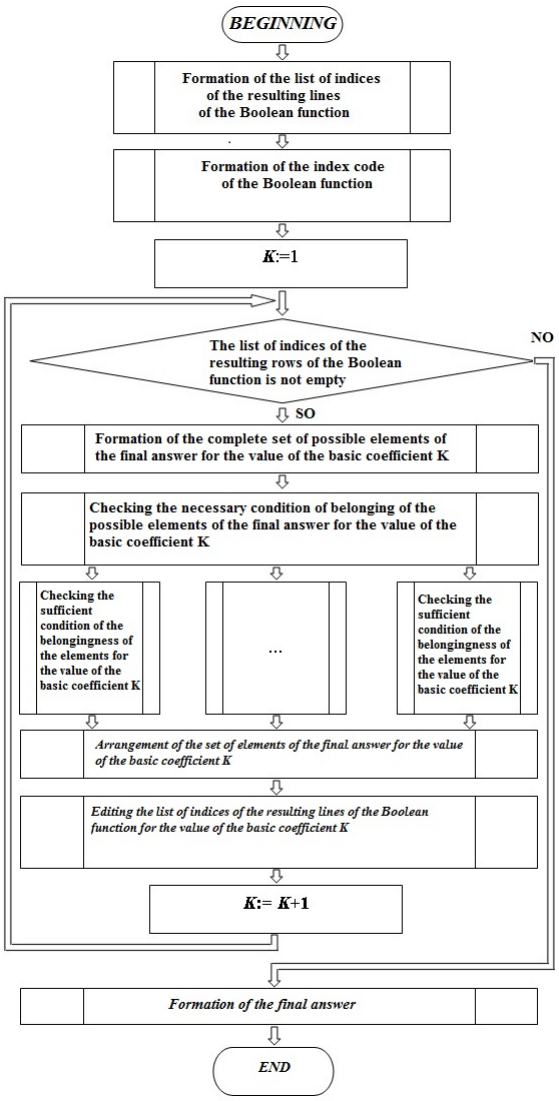


Figure 4. Algorithm of parallelization of the index method of minimization of Boolean functions

Discussion of the results. The developed minimization method is one of the constituent parts of creating the software code for the fractal computer. The main feature of a fractal computer is the presence of fractal (non-smooth) functions in its software code, which will radically expand its capabilities in specific areas of calculations. To this day, none of modern computers use these functions in the program code.

Conclusions. The article proposes a new method – the index method of minimization of Boolean functions containing a large number of arguments. This method is an implementation of evolutionary development of ***BF*** minimization methods [8, 9, 10] by reducing the value of the basic coefficient ***K*** with the aim of accelerating the minimization process over time.

DOI: 10.24025/2306-4412.2.2023. 273763

The scientific novelty consists in the development of a method based on a new form of representation of the Boolean function in the form of a digital index code, which allows to speed up the minimization process in time.

The practical value is represented by a software product that, based on the algorithm of de-parallelization of the minimization process using the index method, makes it possible to minimize Boolean functions on cluster computer systems.

The advantages of this method are:

1. During the formation of a complete list of elements, the elements of the final answer are received immediately without defining intermediate results, which significantly speeds up the search.

2. The complete set of possible elements of the final answer for all Boolean functions containing one number of arguments for the value of the basic coefficient $\boldsymbol{K}$=1…n is one-one. Therefore, the formation of the set is performed even before the start of the algorithm execution and is used during minimization as a reference value. This significantly reduces the minimization time.

3. The main object of the method implementation is the columns of the truth table of the Boolean function. The use of a list of indexes of the resulting rows reduces the number of rows in the truth table to be processed from 2^n to the number of units in the resulting column. As the value of $\boldsymbol{K}$ increases, the list of indexes decreases, accordingly, the algorithm speeds up

4. The implementation of two-level processing of the columns of the truth table of the Boolean function – the level of checking the necessary condition and the level of checking the sufficient condition significantly reduce the time of searching for a complete list of elements of the final answer.

5. Processing of the columns of the truth table of the Boolean function when checking the sufficient condition by the method of indices for obtaining the result makes it possible to speed up the processing of a part of the column. Only column elements whose value is equal to "1" are checked, not the entire column, and stop the check early according to the condition of the absence of certain indexes.

6. The formation of the final answer by the index method is the easiest to implement in comparison with the versions of the implementation in previous methods.

7. Parallelization of the minimization process for the most time-consuming phase of the sufficient condition check at the hardware level significantly speeds up the minimization process.

8. The search for new methods of minimization is an integral part of creating elements of the software code of a fractal computer, namely, the development of tools for the implementation of fractal (non-smooth) functions in the software code, which should radically expand the possibilities of using a fractal computer.

References

[1] E. McCluskey, "Minimization of Boolean functions", *The Bell System Technical Journal*, vol. 35, pp. 1417-1444, Nov. 1956.

[2] W. Quine, "The problem of simplifying truth functions", *American Mathematical Monthly*, vol. 59, pp. 521-531, 1952.

[3] C. E. Shannon, "A mathematical theory of communication", *Bell System Technical Journal,* vol. 27, pp. 379-423, 623-656, 1948.

[4] Yu. A. Kochkaryev, N. M. Panteleeva, N. L. Kazarinova, and S. A. Shakun, *Classical and alternative minimal forms of logical functions*: Reference catalog. Cherkasy: Cherkasy Institute of Management, 1999 [in Ukrainian].

[5] O. P. Pinko, "Minimization of CNF of partially monotone Boolean functions", *Dopovidi Natsionalnoi akademii nauk Ukrainy*, no. 3, pp. 18-21, 2019 [in Ukrainian].

[6] V. Riznyk, and M. Solomko, "Research of 5-bit boolean functions minimization protocols by combinatorian method", *Technology Audit and Production Reserves*, vol. 4, iss. 2 (42), pp. 41-52, 2018.

[7] I. P. Chukhrov, "On the minimization of Boolean functions for additive complexity measures", *Journal of Applied and Industrial Mathematics*, vol. 13, pp. 418-435, 2019.

[8] V. Riznyk, and M. Solomko, "Minimization of Boolean functions by combinatorial method", *Technology Audit and Production Reserves*, vol. 4, no. 2 (36), pp. 49-64, 2017.

[9] I. P. Chukhrov, "On the complexity of minimizing quasicyclic Boolean functions", *Diskretn. Anal. Issled.*, Open 25 (3), pp. 126-151, 2018 [*J. Appl. Indust. Math.*, vol. 12, pp. 426-441, 2018].

[10] Yu. A. Kochkarev, S. V. Burmistrov, and S. F. Aksyonov, "Minimization of Boolean functions by parts", *Radioelektronni ta*

DOI: 10.24025/2306-4412.2.2023. 273763

kompiuterni systemy, no. 4, pp. 110-116, 2012 [in Ukrainian].

[11] Yu. A. Kochkarev, S. V. Burmistrov, and S. F. Aksyonov, "Minimization of partially defined Boolean functions in the orthogonal form of representation", *Prykladna radioelektronika*, vol. 12, no. 3, pp. 413-420, 2013 [in Ukrainian].

[12] Yu. A. Kochkarev, V. N. Rudnytskyi, and S. V. Burmistrov, *Minimization of systems of fully defined Boolean functions in the orthogonal form of representation. Heuristic algorithms and distributed computing in applied problems*: Coll. monograph, Prof. Melnikov (Ed.), iss. 2. Vinnitsa, Kharkiv, 2013, pp. 87-100 [in Ukrainian].

[13] S. V. Burmistrov, "Parallel decomposition as the main method of minimizing Boolean functions in the orthogonal form of representation", *Visnyk Cherkaskoho derzhavnoho tekhnolohichnoho universytetu,* no. 2, pp. 67-73, 2014 [in Ukrainian].

[14] S. V. Burmistrov, and E. N. Panasco, "Parallel decomposition by reducing the value of the basic coefficient K as an alternative method of minimizing Boolean functions", *Visnyk Pryazovskoho derzhavnoho tekhnichnoho universytetu. Seriia: Tekhnichni nauky,* no. 1, pp. 189-195, 2015 [in Ukrainian].

[15] S. V. Burmistrov, and O. B. Piven, "The matrix method of parallel decomposition as a generalized method of minimization in the orthogonal form of representation", *Nauka i tekhnika Povitrianykh Syl Zbroinykh Syl Ukrainy*, no. 4 (21), pp. 151-157, 2015 [in Ukrainian].

Список використаних джерел

[1] E. McCluskey, "Minimization of Boolean functions", *The Bell System Technical Journal*, vol. 35, pp. 1417-1444, Nov. 1956.

[2] W. Quine, "The problem of simplifying truth functions", *American Mathematical Monthly*, vol. 59, pp. 521-531, 1952.

[3] C. E. Shannon, "A mathematical theory of communication", *Bell System Technical Journal,* vol. 27, pp. 379-423, 623-656, 1948.

[4] Ю. А. Кочкарьов, Н. М. Пантелєєва, Н. Л. Казарінова, та С. А. Шакун, *Класичні та альтернативні мінімальні форми логічних функцій*: каталог-довідник. Черкаси: Черкас. ін-т управління, 1999.

[5] О. П. Пинько, "Мінімізація КНФ частково-монотонних булевих функцій", *Доповіді Національної академії наук України*, № 3, с. 18-21, 2019.

[6] V. Riznyk, and M. Solomko, "Research of 5-bit boolean functions minimization protocols by combinatorian method", *Technology Audit and Production Reserves*, vol. 4, iss. 2 (42), pp. 41-52, 2018.

[7] I. P. Chukhrov, "On the minimization of Boolean functions for additive complexity measures", *Journal of Applied and Industrial Mathematics*, vol. 13, pp. 418-435, 2019.

[8] V. Riznyk, and M. Solomko, "Minimization of Boolean functions by combinatorial method", *Technology Audit and Production Reserves*, vol. 4, no. 2 (36), pp. 49-64, 2017.

[9] I. P. Chukhrov, "On the complexity of minimizing quasicyclic Boolean functions", *Diskretn. Anal. Issled.*, Open 25 (3), pp. 126-151, 2018 [*J. Appl. Indust. Math.*, vol. 12, pp. 426-441, 2018].

[10] Ю. А. Кочкарьов, С. В. Бурмістров, та С. Ф. Аксьонов, "Мінімізація булевих функцій частинами", *Радіоелектронні та комп'ютерні системи*, № 4, с. 110-116, 2012.

[11] Ю. А. Кочкарьов, С. В. Бурмістров, та С. Ф. Аксьонов, "Мінімізація частково-визначених булевих функцій в ортогональній формі представлення", *Прикладна радіоелектроніка*, т. 12, № 3, с. 413-420, 2013.

[12] Ю. А. Кочкарьов, В. Н. Рудницький, та С. В. Бурмістров, *Мінімізація систем повністю визначених булевих функцій в ортогональній формі представлення. Евристичні алгоритми та розподілені обчислення у прикладних задачах*: кол. монографія за ред. проф. Мельникова, вип. 2. Вінниця, Харків, 2013, с. 87-100.

[13] С. В. Бурмістров, "Паралельна декомпозиція як основний метод мінімізації булевих функцій в ортогональній формі представлення", *Вісник Черкаського державного технологічного університету*, № 2, с. 67-73, 2014.

[14] С. В. Бурмістров, та Е. Н. Панаско, "Паралельна декомпозиція шляхом зменшення значення базисного коефіцієнта К як альтернативний метод мінімізації булевих функцій", *Вісник Приазовського державного технічного університету. Серія: Технічні науки*, № 1, с. 189-195, 2015.

DOI: 10.24025/2306-4412.2.2023. 273763

[15] С. В. Бурмістров, та О. Б. Півень, "Матричний метод паралельної декомпозиції як узагальнений метод мінімізації в ортогональній формі представлення", *Наука і техніка Повітряних Сил Збройних Сил України*: наук.-техн. журн., № 4 (21), с. 151-157, 2015.

[0000-0003-3642-2849] **С. В. Бурмістров,** *канд. техн. наук,*
e-mail: sergij.burmistrov@ukr.net
[0000-0002-2093-1270] **В. І. Хотунов,** *канд. пед. наук, доцент,*
[0000-0001-6314-5838] **М. В. Захарова,** *канд. техн. наук, доцент,*
[0000-0001-9864-3386] **С. Л. Михайлюта,** *канд. техн. наук, доцент,*
[0000-0002-0248-0461] **М. В. Люта,** *завідувач відділення інженерії програмного забезпечення*
Черкаський державний бізнес-коледж
вул. В. Чорновола, 243, м. Черкаси, 18028, Україна

ІНДЕКСНИЙ МЕТОД МІНІМІЗАЦІЇ БУЛЕВИХ ФУНКЦІЙ

У роботі представлено новий метод мінімізації, що реалізує булеву функцію у класичній мінімальній формі представлення шляхом направленого перебору можливих шляхів мінімізації за критеріями необхідної і достатньої умови – індексний метод. Цей метод є продовженням еволюційного розвитку методів мінімізації шляхом зменшення значення базисного коефіцієнта К: методу мінімізації по частинах, методу паралельної декомпозиції шляхом зменшення К, матричного методу паралельної декомпозиції. Еволюція методів шляхом зменшення значення базисного коефіцієнта К йде шляхом досконалого вивчення будови та структурної організації множини булевих функцій, детального аналізу сильних і слабких сторін уже існуючих попередніх варіантів методів, виявлення критичних місць, що суттєво сповільнюють процес мінімізації, та пошуку альтернативних шляхів прискорення процесу мінімізації. Індексний метод розроблено на основі використання нового способу запису окремих булевих функцій у вигляді індексів значущих рядків таблиці істинності. Завдяки такій формі запису вдалося як реалізувати сильні сторони, шо використовували попередні методи, так і значно поліпшити слабкі етапи попередніх методів, що в цілому дає великий виграш у часі мінімізації. Перевагою методу є двоетапна мінімізація процесу, що дає можливість безпосередньо не використовувати критерій спрямованого сортування. При формуванні повного списку елементів одразу отримують елементи остаточної відповіді без зазначення проміжних результатів. Структурні елементи методу – повний набір можливих елементів кінцевої відповіді для булевих функцій, що містить одну кількість аргументів для значення базового коефіцієнта К=1...n, – формуються ще до початку виконання методу і використовуються як табличне значення. При реалізації методу в стовпцях таблиці істинності обробляються тільки одиниці без нулів, що зменшує кількість об'єктів обробки. Метод реалізується дворівневою обробкою стовпців – перевіркою необхідних і достатніх умов. Машинна реалізація методу використовує розпаралелювання процесу мінімізації. Все це істотно скорочує час мінімізації – основну цінність, що відрізняє цей метод від інших. Розроблений метод мінімізації є однією зі складових частин створення програмного коду, що є основою розробки фрактального комп'ютера. Головною особливістю фрактального комп'ютера є наявність у його програмному коді фрактальних (негладких) функцій, що дозволить радикально розширити його можливості в окремих областях обчислень. На сьогоднішній день жоден із сучасних комп'ютерів не використовує ці функції в програмному коді.

Ключові слова: *метод мінімізації булевих функцій, матричний метод паралельної декомпозиції, результативні рядки булевої функції, розпаралелювання процесу мінімізації, базисний коефіцієнт* ***К****, фрактальний комп'ютер.*

Стаття надійшла 12.02.2023
Прийнято 15.03.2023

DOI: 10.24025/2306-4412.2.2023. 273763