DEVELOPING THE MINIMIZATION OF A POLYNOMIAL NORMAL FORM OF BOOLEAN FUNCTIONS BY THE METHOD OF FIGURATIVE TRANSFORMATIONS

M. Solomko  
PhD, Associate Professor  
Department of Computer Engineering*
E-mail: doctrinas@ukr.net

Iu. Batyshkina  
PhD, Associate Professor  
Department of Informational and Communal Technologies and Computes Science Teaching Methods**
E-mail: yuliia.batyshkina@rshu.edu.ua

N. Khomiuk  
PhD  
Department of International Economic Relations and Project Management  
Lesya Ukrainka Volyn National University  
Volî str., 13, Lutsk, Ukraine, 43000  
E-mail: nataljabilous@gmail.com

Ya. Ivashchuk  
PhD  
Department of Higher Mathematics*
E-mail: zrsd@i.ua

N. Shevtsova  
PhD  
Department of Informatics and Applied Mathematics**
E-mail: natalia.shevtsova@rshu.edu.ua

* National University of Water and Environmental Engineering  
Soborna str., 11, Rivne, Ukraine, 33028
**Rivne State University of Humanities  
S. Bandery str., 12, Rivne, Ukraine, 33028

sions (RM-polynomials) [1]. The taxonomy of RM-polynomials, the relationship between different classes, and the complexity of their implementation are considered in [1–6].

For comparison, we shall illustrate possible polynomials by following the examples:

- \( x_1 x_2 \oplus x_3 x_4 \) – PPRM-polynomial (Positive Polarity Reed-Muller expression), that is, a polynomial of the \( n \)-th power by Zhegalkin, all variables of which have a direct polarity;
- \( x_1 x_2 \oplus x_3 x_4 \) – NPRM-polynomial (Negative Polarity Reed-Muller expression) whose all variables have inverted polarity.

The development of microelectronic technology has ensured the creation of elements that form multiple disjunctions with exception (EXOR elements). This, in turn, ensured the synthesis of similar two-level AND/EXOR circuits that contain the same elements in the second cascade. The structure of these schemes is described by formulas similar to a disjunctive normal form (DNF), in which disjunction operators with an exception are used instead of disjunction operators. Such formulas are called ESOP – the exclusive sum of products. The advantages of these formulas are justified by the fact that the number of logic elements in their respective schemes is usually less. For example, after minimizing the DNF of arbitrary Boolean functions, the four variables contain an average of 4.13 conjuncters, and ESOP contains only 3.66 [4,7]. When considering Boolean functions typical of schemes that implement arithmetic operations, the win is even greater. In addition, AND/EXOR circuits are more easily diagnosed [7,8].

The representation of Boolean functions by Zhegalkin polynomials, the optimization problem does not arise since the solution is unambiguous. The optimization problem appears for non-fully defined Boolean functions. If the value of the function remains undefined in \( k \) sets, \( 2^k \) different function determination and, accordingly, \( 2^k \) different Zhegalkin polynomials forming a given function are possible. The choice among them of the simplest polynomial is a complex combinatorial problem. The task becomes even more difficult when implementing functions in the form of ESOP (which contain literals with different inversions) or when it is necessary to implement a system of Boolean functions.

As stated in [2], effective algorithms for minimizing ESOP of the Boolean functions \( f(x_1, x_2, ..., x_n) \) do not exist. Such a conclusion, however, could be some start, which would eventually move to its own opposite.

The advantages of these algorithms are justified by the fact that the number of logic elements in their respective schemes is usually less. For example, after minimizing the DNF of arbitrary Boolean functions, the four variables contain an average of 4.13 conjuncters, and ESOP contains only 3.66 [4,7]. When considering Boolean functions typical of schemes that implement arithmetic operations, the win is even greater. In addition, AND/EXOR circuits are more easily diagnosed [7,8].

In the case of representing Boolean functions by Zhegalkin polynomials, the optimization problem does not arise since the solution is unambiguous. The optimization problem appears for non-fully defined Boolean functions. If the value of the function remains undefined in \( k \) sets, \( 2^k \) different function determination and, accordingly, \( 2^k \) different Zhegalkin polynomials forming a given function are possible. The choice among them of the simplest polynomial is a complex combinatorial problem. The task becomes even more difficult when implementing functions in the form of ESOP (which contain literals with different inversions) or when it is necessary to implement a system of Boolean functions.

As stated in [2], effective algorithms for minimizing ESOP of the Boolean functions \( f(x_1, x_2, ..., x_n) \) do not exist. Such a conclusion, however, could be some start, which would eventually move to its own opposite.

The evolution of the visual-matrix form of the analytical method of simplification of logic functions is the result of continuous optimization, in particular [9]. In this regard, theoretical studies on minimizing ESOP of Boolean functions remain relevant, in particular, to improve the following factors:

- the visual and matrix methods of minimizing logic functions in the ESOP class;
- the cost of a technique to minimize ESOP of logic functions.

The representation of Boolean functions by reverse circuits, on the elements of Toffoli, is considered in paper [15]. Interest in this issue is associated with the implementation of «cold» calculations. This means that when performing such calculations, there is no heat dissipation. In general, reversible schemes implement reversible functions. Therefore, the Toffoli-Fredkin method is used to represent a Boolean function by a reversible function. The paper describes an algorithm for finding the minimum representation of a Boolean function in the class of reversing chains built on Toffoli elements. The algorithm uses ESOP of Boolean functions, as well as the class of polarized Zhegalkin polynomials or the Reed-Muller forms. The results of the computational algorithm that minimizes Boolean functions in the class of reversible circuits are presented.

Minimizing logic has been attracting considerable attention lately, as it is important for many applications to have the most compact images possible. Paper [16] proposes a fast

2. Literature review and problem statement

Generalized rules for simplifying logic expressions in the format of the theory of polynomial sets are considered in paper [10]. The rules reported there are based on the proposed theorems for different initial conditions for the transformation of paired conjuncters, the Hamming distance between which can be arbitrary. These rules can be useful for minimizing arbitrary logic functions with \( n \) variables in the theoretical format of polynomials. The advantages of the proposed simplification rules are illustrated by examples.

A method to search for the exact ESOP expression for a not fully specified arbitrary logic function up to six input variables is proposed in work [11]. To this end, the weights of all 5-variable functions are entered in the table, which is used in the proposed approach and speeds up the computation time. It is believed that this is the first paper concerning the exact solutions of minimization for not fully defined Boolean functions.

Work [12] reports the results of a study into minimizing AND-XOR expressions for functions with a large number of literals. The object of the study is the use of a simple greedy algorithm for finding a minimum function, based on a set of local transformations, to the expressions of the Boolean functions of the Reed-Muller basis with positive polarity. It is noted that experiments with large functions demonstrate good results. The number of literals was reduced by an average of 23 %. It is assumed that much better results could be achieved once a more complex non-greedy algorithm for finding minimal functions is applied.

Minimization of multilevel representation of Boolean functional systems based on Shannon extension with finding equal coefficients (exact within inversion), and using Zhegalkin polynomials for these purposes, is proposed in [13]. Zhegalkin polynomials are easy to compare and easy to obtain the inversion of a function, and that significantly reduces the calculation time. The application of the program that implements the proposed algorithms makes it possible to receive smaller areas of VLSI circuits compared to chains that are synthesized using minimized DNF and Shannon expansion schemes where coefficient inversion is not taken into consideration.

Paper [14] reports a method for determining the upper limit of complexity in the implementation of arbitrary Boolean functions, which can be implemented by Zhegalkin polynomials. A computational method for improving these boundaries is proposed.
minimizing algorithm (FMA) for the Reed-Muller fixed polarity expressions (FPRM). The basic FMA idea is to find the minimum FPRM function with the fewest conjuncterms. This uses the proposed binary differential search evolution (BDE) algorithm. The authors described experimental results involving 24 MCNC control circuits, which show that FMA surpasses the genetically based effectiveness of finding the minimum Reed-Muller expressions. It is assumed that the use of the differential evolution algorithm to minimize FPRM was first considered in paper [16]. FMA can be expanded so that it is possible to obtain a minimum mixed polarity of the expression on the polynomial basis.

The above literary sources [10–16] mostly report algorithms and methods for minimizing Boolean functions in the Reed-Muller basis by using theoretical objects of adjacent theory. Specifically, Hamming distance tables, greedy (non-greedy) algorithms, Shannon extensions, genetic algorithm, differential evolution algorithm to minimize FPRM, etc. Not all methods provide an accurate solution to minimization. An obligatory technological point for these algorithms and methods are automated calculations. In the complex search for the optimal function, compensation may be an approximate synthesis — the tendency of logic synthesis, when some results of the logical specification change within the permissible optimality of the digital circuit to be designed.

A method of figurative transformations based on binary combinatorial systems with repeated 2-(n, b)-design, 2-(a, x/b)-design belongs to the classical analytical method by qualification. They do not allow for the approximate result of minimization and do not exclude the manual technique for minimizing Boolean functions, including in the Reed-Muller basis.

Thus, the algorithms and methods that employ theoretical objects of adjacent theory, software tools for them, which cover the general procedure for minimizing the logic functions of the polynomial basis [10–16] and a method of figurative transformations, follow different approaches (principles of minimization). Therefore, they consider different prospects regarding the possibility of algorithmic minimization of logic functions of the polynomial basis.

The prospect for a figurative transformation method, which is a descendant of the analytical method, regarding the proper minimization of logic functions in the Reed-Muller basis is to create the necessary algebra in terms of the rules for the equivalent transformation of polynomial functions. As well as identify the reserves of the analytical method, such as the sequence in the procedure of inserting the same conjuncterms of polynomial functions, followed by the operation of super-gluing the variables. Thus, the classical analytical method still has the prospect of increasing its hardware capabilities to minimize functions on the Reed-Muller basis. And this is the reason to believe that the software and technological base, represented by the algorithms and methods with theoretical objects of adjacent theories [10–16], is insufficient for theoretical research into the optimal minimization of Boolean functions on the Reed-Muller basis.

This determines the need to investigate equivalent figurative transformations in order to minimize logic functions on the Reed-Muller basis. In particular, the peculiarities of relatively complex algorithms of simplification of functions with the procedure of inserting the same conjuncterms of polynomial functions followed by the operation of super-gluing the variables, a stack of logical operations for the first binary matrix of a polynomial function [9], ways to simplify arbitrary functions in the Reed-Muller basis.

In practical terms, a figurative transformation method would provide an expansion of the capabilities of technology for designing digital components based on the basis $\Sigma_1 = \{\land, \lor, \neg\}$.

### 3. The aim and objectives of the study

The purpose of this work is to extend the method of figurative transformations to minimize Boolean functions in the class of Exclusive-OR Sum-Of-Product forms (ESOP) and perfect Exclusive-OR Sum-Of-Product forms (PESOP). That could simplify the performance of minimizing functions in the Reed-Muller basis by refining Reed-Muller’s algebra in terms of algebraic rules for the equivalent transformation of ESOP functions.

To accomplish the aim, the following tasks have been set:
- to establish the hermeneutics of logic operations for the class of equivalent binary matrices of functions of the Reed-Muller basis;
- to establish patterns in the use of algorithms for the equivalent transformation of Boolean functions in the Reed-Muller basis, consisting of the procedure of inserting two identical ESOP conjuncterms with the following operation of super-gluing the variables;
- to devise a method for the orthogonalization of logic functions using figurative transformations in order to establish singular functions;
- to refine Reed-Muller’s algebra in terms of the necessary algebraic rules for the equivalent transformation of ESOP functions;
- to analyze the results from simplifying functions in the Reed-Muller basis by a figurative transformation method and the examples of minimization of functions in the polynomial basis in order to compare the cost of the minimum function implementation and the number of procedural steps.

### 4. The Reed-Muller basis

Along with the well-known Boolean basis $\{\lor, \land, \neg\}$ and non-redundant bases $\{\lor, \neg\}$ and $\{\land, \neg\}$, an important role in the theory of logic functions and in its practical application belongs to the Reed-Muller basis $\{\land, \lor, \neg\}$, which includes the operation «sum of modulo 2» (⊕). The completeness of this basis is proven by the following ratios, which demonstrate that it is reduced to a well-known full basis $\{\neg, \lor\}$:

$$\overline{a} = 1 \oplus a; \quad a \lor b = a \oplus b \oplus ab.$$  

Similar to disjunction and conjunction operations, the sum by module two has the properties of commutativity and associativity, and is also generalized in the case of a large number of variables.

Multiple amount for module two of:

$$\Theta(a, b, s, d) = a \oplus b \oplus c \oplus d = \bmod_2 \sum (a, b, s, d),$$

arbitrary elementary conjunctions is a polynomial. A separate case of the polynomial is the Zhegalkin polynomial, consisting of non-inverted variables.

Any Boolean function can be represented by a formula in the Reed-Muller basis (Zhegalkin) $\Sigma_1 = \{\land, \lor, \neg\}$. 

5. Reed-Muller Algebra

An algebra over a set of logic functions with two binary operations $\&$ and $\oplus$ is the Reed-Muller (Zhegalkin) algebra. The following interrelations are satisfied in the Reed-Muller algebra:

$$x \oplus y = x\bar{y} + \bar{x}y = (x + y)(\bar{x} + \bar{y});$$

$$x_1 \bar{x}_2 \oplus \bar{x}_1 x_2 = x_1 \bar{x}_2 \oplus x_2 \bar{x}_1.$$  

Identity (2) has an illustration of the representation:

$$10 \quad 01 \quad 1 \quad 1$$

or

$$10 \quad 01 \quad 0 \quad 0$$

$$xy \oplus \bar{x} \bar{y} = x \oplus y = x \oplus \bar{y} = x \oplus \bar{y} = \bar{x} \oplus y.$$  

Identity (3) has an illustration of the representation:

$$11 \quad 00 \quad 1 \quad 0$$

or

$$11 \quad 00 \quad 1 \quad 0$$

$$xy \oplus x \oplus x = x \oplus \bar{y} = x \oplus \bar{y} = x \oplus \bar{y} = \bar{x} \oplus y.$$  

For the function of addition modulo two, there are displaceable and connecting laws, as well as the distribution law with respect to the conjunction.

$$x \oplus y = y \oplus x;$$

$$x \oplus (y \oplus z) = (x \oplus y) \oplus z;$$

$$x(y \oplus z) = (xy) \oplus (xz).$$

The following valid interrelations are obvious [17]:

$$[x \oplus x = 0;]$$

$$[x \oplus 0 = x;]$$

$$[x \oplus 1 = \bar{x};]$$

$$[x \oplus \bar{x} = 1.]$$

In addition, the following formulas hold:

$$x + y = x\bar{y} \oplus \bar{x}y = (x \oplus 1)(y \oplus 1) \oplus 1 = x \oplus y \oplus x = x \oplus \bar{x};$$

$$x_1 \cdot x_2 = (x_1 \oplus x_2) \oplus (x_1 \oplus x_2).$$

Logic identities for two variables are given in Table 1.

<table>
<thead>
<tr>
<th>Logical equality</th>
<th>$x \cdot y$</th>
<th>$\bar{x} \bar{y} + y \oplus x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical inequality</td>
<td>$x \oplus y$</td>
<td>$x \bar{y} + \bar{x}y = x \oplus y$</td>
</tr>
<tr>
<td>Disjunction</td>
<td>$x + y$</td>
<td>$x + y = x \oplus y \oplus x$</td>
</tr>
<tr>
<td>Schaefer’s stroke</td>
<td>$xy$</td>
<td>$x \bar{y} = \bar{x} + y = \bar{x} \oplus y$</td>
</tr>
<tr>
<td>Implication</td>
<td>$y \rightarrow x$</td>
<td>$y \rightarrow x = \bar{x} + \bar{y} = \bar{x} \oplus \bar{y}$</td>
</tr>
<tr>
<td>Implication</td>
<td>$y \rightarrow x$</td>
<td>$y \rightarrow x = \bar{x} + \bar{y} = \bar{x} \oplus \bar{y}$</td>
</tr>
<tr>
<td>Conjunction</td>
<td>$xy$</td>
<td>$xy = x \bar{y} \oplus \bar{x}$</td>
</tr>
<tr>
<td>Conjunction</td>
<td>$xy$</td>
<td>$xy = x \bar{y} \oplus \bar{x}$</td>
</tr>
<tr>
<td>Conjunction</td>
<td>$xy$</td>
<td>$xy = x \bar{y} \oplus \bar{x}$</td>
</tr>
<tr>
<td>Pierce’s arrow</td>
<td>$x \downarrow y$</td>
<td>$x \downarrow y = \bar{x} \oplus \bar{y} = \bar{x} \oplus \bar{y}$</td>
</tr>
</tbody>
</table>

The Reed-Muller algebra provides the creation of classical rules of equivalent transformation to simplify logic expressions in the ESOP class by analytical method. The difference between a Reed-Muller polynomial and a Zhegalkin polynomial is that the Zhegalkin polynomial represents 2-level logic, and the Reed-Muller polynomial represents 3-level logic. However, a Reed-Muller polynomial generally contains fewer literals.

In some cases, the transformation of the Reed-Muller polynomial to the mixed basis produces 2-level logic. Such as:

$$1 1 2 3 1 2 3 1 2 3 \oplus \oplus = \oplus = \oplus +() = +() = +().$$

The 3-level logic of the Reed-Muller polynomial is transformed into 2-level mixed basis logic.

6. Interpretation of the truth table of a logic function by the binary combinatorial systems with repetition

For some set $A$, a new set $M(A)$ can be considered – a set of all its subsets – a Boolean. $M_k(A)$ denotes the set of all subsets $A$ that have $k$ elements.

Assume $A = \{a, b, c\}$, then:

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

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

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

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

The following equality holds:

$$\sum_{k=0}^{n} C_n^k = 2^n.$$
Since \( C_k^n \) is the number of \( k \)-element subsets of the set of \( n \) elements, the sum in the left-hand side of expression (7) is the number of all subsets.

The set \( A = \{a, b, c\} \), in addition to recalculating its elements, can also determine the number of positions on which the element \( a \) is located. For example, \( a \) can mean the first position, \( b \) can mean the second position of the set \( A \), etc. Subsets of the set \( A = \{a, b, c\} \), in this case, are the subsets containing the element \( a \) at \( k \) positions, \( k = 0, ..., n \), where \( n \) is the number of positions of the set \( A \). In a general case, the element \( a \) can occupy several positions on the set \( A \), so the element \( a \) is repeated on the set \( A \).

Let \( \alpha = 1 \), then the positions where the element \( a \) is missing are denoted by zero. For the set \( A = \{a, b, c\} \), which defines position numbers, we accept \( \alpha = 1 \). Then the subsets of set \( A \) take the following form:

\[
(0,0,0); \quad (1,0,0); \\
(0,0,1); \quad (1,0,1); \\
(0,1,0); \quad (1,1,0); \\
(0,1,1); \quad (1,1,1).
\]

(8)

The number of all \( k \)-element subsets of the set \( A = \{a, b, c\} \), which determines the position numbers, is calculated from formula (7).

Configuration (8) is a complete combinatorial system with a repeated element \( a \), which is denoted:

\( 2-(n, b) \)-design,

where \( n \) is the bit size of the system block; \( b \) is the number of blocks of the complete system, determined from formula \( b = 2^n \). The number 2 before the brackets denotes the binary structure of configuration (8). For example, \( 2-(4, 16) \)-design is the complete binary combinatorial system with a repetition consisting of 4-bit blocks, the number of blocks is 16.

It is easy to see that configuration (8), which makes up the complete combinatorial system with repetition, can be interpreted as a truth table of the logic function \( f(a, b, c) \), with a full set of minterms or maxterms (Table 2).

Another interpretation variant is demonstrated by a truth table that contains a combinatorial system with repeated \( 2-(2, 4) \)-design in the table configuration variant when there is one column with the same variable values (Table 3).

In this case, the combinatorial system with repeated \( 2-(2, 4) \)-design (combinatorial representation) would interpret the login operation, in a given case – super-gluing the variables.

The procedure of reducing the complete perfect disjunctive normal form (PESOP) of a logic function yields unity. For example, reducing a 2-variable full PESOP takes the following form:

\[
x_1 \overline{x}_2 + x_2 \overline{x}_3 + x_1 \overline{x}_3 + \overline{x}_1 x_2 x_3 = \\
= \overline{x}_1 (x_2 + x_3) + x_1 (\overline{x}_2 + x_3) = \overline{x}_1 + x_1 = 1.
\]

Since the complete PESOP uniquely determines the complete combinatorial system with repeated \( 2-(n, b) \)-design and vice versa, this gives reason to remove all blocks of the full combinatorial system with a repetition from the truth table of the assigned function.

The visual representation of the logic operation of super-gluing the variables involving \( 2-(2, 4) \)-design takes the following form:

\[
\begin{array}{c|ccc}
0 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 \\
0 & 1 & 1 & 1 \\
1 & 1 & 1 & 1
\end{array}
\]

The algebraic notation of the logic operation of super-gluing the variables is as follows:

\[
\overline{x}_1 x_2 x_3 + x_1 \overline{x}_2 x_3 + x_1 x_2 \overline{x}_3 + x_1 x_2 x_3 = \\
= (\overline{x}_1 (x_2 + x_3) + x_1 (\overline{x}_2 + x_3)) x_1 = (\overline{x}_1 + x_1) x_1 = x_1.
\]

Similarly, other operations of equivalent transformation of logic expressions are interpreted.

In a general case, the truth table of the logic function, in addition to the configuration of the complete combinatorial system with repeated \( 2-(n, b) \)-design, can also contain the configuration of an 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 an incomplete combinatorial system with repeated \( 2-(n, x/b) \)-design also make it possible to set rules that ensure the effective minimization of Boolean functions.

Configuration (8) can also be interpreted as a truth table of the logic function ESOP \( f(a, b, c) \), with a full set of conjuncterms of the function in the Reed-Muller basis and the operation of adding modulo two for them.

7. Results of minimizing Boolean functions in the Reed-Muller basis by a figurative transformation method

Equivalent figurative transformations when minimizing functions in the Reed-Muller basis yield the following result:

- determining the hermeneutics of logic operations for the class of equivalent binary matrices of functions in the Reed-Muller basis;
- a protocol with relatively complex algorithms for simplifying logic expressions, which consists of the procedure for inserting two identical conjuncterms of the functions of a polynomial basis with the following operation of super-gluing the variables. This protocol increases the efficiency of the procedure, which makes it possible, in particular, to simplify logic functions in the Reed-Muller basis with a relatively large number of input variables manually;
– ensuring the method of orthogonalization of specified logic functions for the establishment of singular functions;
– creating algebra in terms of the rules for equivalent transformation of Boolean functions in the Reed-Muller basis.

7.1. Hermeneutics of logic operations for the class of equivalent binary matrices of functions in the Reed-Muller basis

Represent the logic function \( f(x_1, x_2, \ldots, x_n) \) in a perfect Exclusive-OR Sum-Of-Product form (PESOP):

\[
 f(x_1, x_2, \ldots, x_n) = \oplus x_1^{a_1} x_2^{a_2} \ldots x_n^{a_n},
\]

where the symbol \( \oplus \) means that the sum by modulo two is taken only on sets of variables \( <a_1, a_2, \ldots, a_n> \), on which \( f(x_1, x_2, \ldots, x_n) = 1 \).

To represent PESOP (9) by a binary equivalent or a matrix, variables with inversion \( x_n \) must be replaced with \( 1_a \), variables without inversion \( x_n \) – with \( 1_n \), where \( n \) is a numeric index that determines the bit size of the variable character \( {1}_a \) or \( {1}_n \) in the conjuncterms of the Reed-Muller basis function. Then PESOP (9) can be represented by the following binary sets (tuples).

\[
 F = (0,0,\ldots,0)(0,1,\ldots,1),
\]

or the following matrix:

\[
 F = \begin{bmatrix}
 0 & 0 & \cdots & 0
 \\
 0 & 0 & \cdots & 1
 \\
 \vdots & \vdots & \ddots & \vdots
 \\
 0 & 1 & \cdots & 1
\end{bmatrix}, \quad (11)
\]

We shall call matrix (11) an instance of the binary matrix class of the Reed-Muller basis functions.

The hermeneutics of logic operations for matrix (11) expresses the conjuncterms of the Reed-Muller basis function:

\[
 F = \begin{bmatrix}
 1 & 0 & 0 & 0
 \\
 1 & 0 & 0 & 1
 \\
 1 & 1 & 0 & 0
 \\
 1 & 1 & 1 & 1
\end{bmatrix}, \quad (12)
\]

The implementation of logic operations on binary and algebraic function structures is to be highlighted in color if you know what we mean. That would provide for a better didactic of the method under consideration.

For the method of figurative transformations, the algorithm of function simplification involving the procedure of inserting two identical conjuncterms of polynomial basis functions with the following operation of super-gluing the variables may have the following variants, for example:

\[
 x_1 x_2 x_3 \oplus x_1 x_2 x_3 \oplus x_1 x_2 x_3 \oplus x_1 x_2 x_3 =
\]

\[
 x_1 x_2 x_3 \oplus x_1 x_2 x_3 \oplus x_1 x_2 x_3 \oplus x_1 x_2 x_3 \oplus x_1 x_2 x_3 =
\]

\[
 x_1 x_2 x_3 \oplus x_1 x_2 x_3 \oplus x_1 x_2 x_3 \oplus x_1 x_2 x_3 \oplus x_1 x_2 x_3 =
\]

The results from the matrix (12), (14), (16), and algebraic (13), (15), (17) techniques of minimizing a logic expression coincide.

Comparatively complex algorithms for simplifying logic expressions involving the procedure of inserting two identical conjuncterms of polynomial basis functions with the following operation of super-gluing the variables (12), (14), (16) expand the variants of their application. This ensures an increase in the efficiency of the procedure for minimizing Boolean functions in ESOP by using a method of figurative transformations.

7.3. Singular functions

To solve optimization problems of logic synthesis, it is necessary to have PESOP functions and RM-polynomials with a minimum number of conjuncterms of the assigned function \( f(x_1, x_2, \ldots, x_n) \). At the same time, if it is possible to choose an RM polynomial (except for PPRM- and NPRM-polynomials), then in the case of the same number of conjuncterms, preference is given to the RM-polynomial with the minimum total number of literals. And when the number of the latter is the same, the minimum RM polynomial is the one with fewer inverted literals. Thus, the cost of implementing the RM-polynomial of the assigned function \( f(x_1, x_2, \ldots, x_n) \) can be estimated by the numerical ratio \( k_i / k_n \) where
The Reed-Muller algebra allows the operation to glue variables (17) that is set by PESOP.

Example 1: It is required to orthogonalize function

\[ F(x_1, x_2, x_3, x_4) \]  \( (18) \)

The procedure of orthogonalization using figurative transformations is as follows:

\[ F(x_1, x_2, x_3, x_4) = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix} \]

The conjuncterms of function (19) are pairwise orthogonal. In algebraic notation, function (19) takes the following form, for example [18]:

\[ x_1 \oplus x_2 = x_1 \lor x_2. \]  \( (20) \)

Functions (19), (20) are singular since the equivalent transformations for them can be carried out by choosing the algebra of one of the bases – Bull or Reed-Muller \{∧, ∨, ¬\}.

Orthogonalizing PESOP function (18) by the algebraic procedure is as follows:

\[ f = x_1 x_3 + x_2 x_4 x_1 + x_1 x_2 x_3 + x_1 x_2 x_4 = x_1 x_3 + x_2 x_4 x_1 + x_1 x_2 x_3 + x_1 x_2 x_4 = x_1 x_3 + x_2 x_4 x_1 + x_1 x_2 x_3 + x_1 x_2 x_4 = x_1 x_3 + x_2 x_4 x_1 + x_1 x_2 x_3 + x_1 x_2 x_4. \]  \( (21) \)

The result of the orthogonalization of assigned function (18) by figurative transformations (19) and algebraic technique (21) is the same.

7. 4. Equivalent transformations of logic functions on the Reed-Muller basis

Since \( x \oplus \bar{x} = 1 \), the Reed-Muller algebra allows the operation to glue variables:

\[ x_1 x_2 x_3 \oplus x_1 \bar{x}_2 x_3 = x_1 x_3. \]  \( (22) \)

Proof:

\[ x_1 x_2 x_3 \oplus x_1 \bar{x}_2 x_3 = x_1 x_3 (x_2 \oplus x_2) = x_1 x_3. \]

The rule of gluing the variables for ESOP (22) has an illustration of the combinatorial representation (23):

\[ \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \end{bmatrix} \]

The logic operation of semi-gluing the variables for ESOP (24) is as follows:

\[ x_1 \oplus x_2 = x_1 \lor x_2. \]  \( (24) \)

Proving rule (24) is based on formula (4). And, since the left-hand side of (24) is singular, in order to implement the simplification procedure by means of the operation of semi-gluing the variables, it is necessary to switch from ESOP to PESOP of the logic function.

The operation of semi-gluing the variables (24) has an illustration of the representation (25):

\[ \begin{bmatrix} 1 & 1 \end{bmatrix} \]

The hermeneutics of the logic operation of semi-gluing the variables (25) implies a transition from the ESOP function to the PESOP function.

Other variants of semi-gluing the variables:

\[ \bar{x}_1 \oplus x_1 x_2 = \bar{x}_1 \lor x_2. \]  \( (26) \)

\[ x_1 \oplus \bar{x}_2 x_2 = x_2 \lor \bar{x}_2. \]  \( (27) \)

The operation of semi-gluing the variables for ESOP can take the following form, for example [18]:

\[ x_1 x_2 \oplus x_1 \bar{x}_2 = x_1 \oplus x_1 \bar{x}_2 x_2. \]  \( (28) \)

Proof:

\[ x_1 x_2 \oplus x_1 \bar{x}_2 = x_1 (x_1 \oplus x_1 \bar{x}_2) = x_1 (x_1 \lor x_1) = x_1 x_2 x_2 x_2 x_2 x_2 x_2 x_2. \]  \( (29) \)

The operation of semi-gluing the variables (28) has an illustration of the representation (30) [18].

\[ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix} = \bar{x}_1 \oplus x_1 \bar{x}_2 x_2. \]  \( (30) \)

Rule (28) in paper [18] is proved by using the Hamming distance.

The rule of semi-gluing the variables for PESOP logical functions.

For 4-variable conjuncterms of PESOP logic functions, the rule of super-gluing the variables can take the following form, for example:

\[ x_1 \oplus x_1 \bar{x}_2 x_2 x_2 x_2 x_2 x_2 x_2 = x_1 x_2 x_2 x_2 x_2 x_2 x_2 x_2 \]

\[ = x_1 x_2 x_2 x_2 x_2 x_2 x_2 x_2. \]  \( (31) \)

The rule of super-gluing the variables is as follows:

\[ x_1 \oplus x_1 \bar{x}_2 x_2 x_2 x_2 x_2 x_2 x_2 = x_1 x_2 x_2 x_2 x_2 x_2 x_2 x_2. \]  \( (32) \)

\[ = x_1 x_2 x_2 x_2 x_2 x_2 x_2 x_2. \]  \( (33) \)

\[ = x_1 x_2 x_2 x_2 x_2 x_2 x_2 x_2. \]  \( (34) \)

\[ = x_1 x_2 x_2 x_2 x_2 x_2 x_2 x_2. \]  \( (35) \)
The equivalent transformations for the rule of semi-gluing the variables (31) have an illustration of representation (32):

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 \\
\end{bmatrix} = 10 = x_1 \overline{x}_2. \tag{32}
\]

The super-gluing rule for ESOP (32) is based on the use of a complete combinatorial system with repeated 2-(2, 4)-design [19].

The super-gluing variable rule for ESOP, which uses a complete combinatorial system with repeated 2-(3, 8)-design [19] may take the following form, for example:

\[
F = \begin{bmatrix}
0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 \\
0 & 1 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\end{bmatrix} \cdot 1 \cdot 1 = x_1 x_2. \tag{33}
\]

The rule of incomplete super-gluing the variables for ESOP logic functions.

Combinatorial properties of an incomplete combinatorial system with repeated 2-\((n, x/b)\)-design [19] ensure a rule of incomplete super-gluing the variables in the Reed-Muller basis.

For a 2-variable ESOP function’s conjuncterms, the rule of incomplete super-gluing the variables may take the following form, for example:

\[
f(x_1, x_2) = x_1 x_2 \oplus x_1 x_3 \oplus x_2 x_3 = \\
x_1 (x_2 \oplus x_3) \oplus x_1 x_2 = \\
x_1 x_2 + x_2 x_3. \tag{34}
\]

Equivalent transformations for the rule of incomplete super-gluing the variables (34) have an illustration of representation (35):

\[
\begin{bmatrix}
0 & 1 & 1 \\
1 & 1 & 0 \\
\end{bmatrix} = 1 \cdot 1 = x_1 + x_2. \tag{35}
\]

In the second matrix of (35), the operation of semi-gluing the variables for ESOP of logic functions (24) was applied, the result of which is represented in the third matrix (35).

Rule (35) uses an incomplete combinatorial system with repeated 2-(2, 3/4)-design [19].

Generalized gluing the variables in the Reed-Muller basis can be carried out using the following transformations:

1. \(x_1 x_2 \oplus x_3 x_4 \oplus x_3 \overline{x}_2 = x_1 x_3 x_2 \oplus x_3 x_2 x_4\).

Proof:

\[
x_1 x_2 \oplus x_3 x_4 \oplus x_3 \overline{x}_2 = \\
x_1 x_2 + x_3 x_4 + x_3 \overline{x}_2 = \\
= x_1 x_2 + x_3 \overline{x}_2 x_4 + x_3 \overline{x}_2 x_3 = x_1 x_2 + x_2 x_3. \tag{36}
\]

Equivalent transformations for the rule of generalized gluing the variables (36) have an illustration of representation (37):

\[
\begin{bmatrix}
1 & 1 & 0 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 0 & 1 \\
\end{bmatrix} = 1 \cdot 10 = x_1 \overline{x}_2. \tag{37}
\]

The second and third matrices (37) involve an operation of absorbing the variables for ESOP (40) to (45).

2. \(x_1 x_2 \oplus x_3 x_4 \oplus x_3 \overline{x}_2 = x_1 (x_2 \oplus x_3) \oplus x_2 \overline{x}_3\).

The rule of generalized gluing the variables (38) has an illustration of representation (39):

\[
\begin{bmatrix}
1 & 1 & 0 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 0 & 1 \\
\end{bmatrix} = 1 \cdot 10 = x_1 \overline{x}_2. \tag{39}
\]

Both generalized variable gluing rules (36) and (38) have a 3-level logic. Applying rule (38) reduces the original expression by one literal.

The logic operation of variable absorption for ESOP is as follows:

\[
x_1 \oplus x_3 \overline{x}_2 = x_1 x_2. \tag{40}
\]

The validity of logic expression (40) is confirmed by a truth table (Table 4).

<table>
<thead>
<tr>
<th>(x_1)</th>
<th>(x_2)</th>
<th>(\overline{x}_2)</th>
<th>(x_1 \oplus x_2)</th>
<th>(x_1 \oplus \overline{x}_2)</th>
<th>(x_1 \overline{x}_2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Other variants of the operation of absorbing the variables for ESOP:

\[
x_1 \oplus x_2 x_3 = x_1 \overline{x}_2; \tag{41}
\]

\[
x_1 x_2 x_3 \oplus x_2 \overline{x}_3 = x_1 \overline{x}_2 \overline{x}_3; \tag{42}
\]

\[
x_1 x_2 \oplus x_3 = x_1 \overline{x}_2; \tag{43}
\]

\[
x_1 x_2 \oplus x_3 \overline{x}_2 = x_1 \overline{x}_2 \overline{x}_3; \tag{44}
\]

\[
x_1 \oplus x_2 x_3 = x_1 \overline{x}_2 x_3 = x_1 (x_2 + \overline{x}_3). \tag{45}
\]

The created classical rules for the equivalent transformation of logic functions in the Reed-Muller basis ensure their effective simplification by a figurative transformation method.
7.5. Examples of minimizing Boolean functions in the Reed-Muller basis by a figurative transformation method

Proving the advantage of logic expressions in a polynomial format comes down to a smaller number of logic elements in their respective schemes, compared to the expressions of the disjunctive (conjunctive) form. It is also promising to use a mixed basis.

**Example 2:** It is required to simplify the Boolean function \( f(x_1, x_2, x_3, x_4) \) in the polynomial form (ESOP), assigned by the Karnaugh map (Fig. 1) \[20\].

![Karnaugh Map](image)

**Fig. 1. Minimizing Boolean functions using a Karnaugh map.**

**Solution.**

Since the conjuncters of the initial function are pairwise orthogonal (a singular function), in order to simplify the assigned function, we select the Reed-Muller algebra. The minimization of \( f(x_1, x_2, x_3, x_4) \) (Fig. 1) in a polynomial format is carried out using the following figurative transformations:

\[
f = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1
\end{bmatrix} = \begin{bmatrix}
0 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 1 & 1 & 0
\end{bmatrix} = \begin{bmatrix}
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Function \( f(x_1, x_2, x_3, x_4) \) MPNF (Fig. 1):

\[
f_{\text{MPNF}} = 1 \oplus \overline{x_1} \overline{x_2} x_3 \overline{x_4}.
\]

The cost of (46) implementation is \( k_1/k_2 = 3/6/3 \).

To minimize the function in Fig. 1, an algorithm to simplify the function involving the procedure of inserting two identical ESOP conjunctors with the following operation of super-gluing the variables (chapter 7.2) was applied.

The simplified function \( f(x_1, x_2, x_3, x_4) \) (Fig. 1) in the mixed basis:

\[
f_{\text{MPMB}} = \left(x_1 + x_3 + \overline{x_4}\right) \oplus x_2 x_3 x_4.
\]

The cost of (47) implementation is \( k_2/k_2 = 2/6/2 \).

Both functions (46) and (47) represent a 3-level logic. On a mixed basis, the minimum function (47) has better implementation indicators.

Table 5 gives the results from minimizing the function \( f(x_1, x_2, x_3, x_4) \) (Fig. 1) in ESOP using a Karnaugh map \[20\] and the method of figurative transformations.

<table>
<thead>
<tr>
<th>Karnaugh map</th>
<th>Method of figurative transformations</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_{\text{MB}}(a, b, c, d) = 1 \oplus ab \oplus \overline{cd} \oplus \overline{ac} \oplus \overline{bc} \oplus \overline{ad} )</td>
<td>( f_{\text{MPMB}} = 1 \oplus \overline{x_1} \overline{x_2} x_3 \overline{x_4} )</td>
</tr>
<tr>
<td>( P_{\text{MB}}(a, b, c, d) = 1 \oplus \overline{a} \oplus \overline{bc} \oplus \overline{ab} \oplus \overline{cd} )</td>
<td></td>
</tr>
</tbody>
</table>

When contemplating Table 5, we see that the result of simplifying the function \( f(x_1, x_2, x_3, x_4) \) (Fig. 1) by a figurative transformation method is a minimal function containing six literals. This is four literals less compared to \[20\].

The verification of the resulting MPNF (46) is given in Table 6.

<table>
<thead>
<tr>
<th>No.</th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( x_4 )</th>
<th>( f(\text{Fig. 1}) )</th>
<th>( f_{\text{MPNF}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1 \oplus \overline{b} \oplus \overline{a} \oplus \overline{d}</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1 \oplus \overline{b} \oplus \overline{a} \oplus \overline{d}</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1 \oplus \overline{b} \oplus \overline{a} \oplus \overline{d}</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1 \oplus \overline{b} \oplus \overline{a} \oplus \overline{d}</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1 \oplus \overline{b} \oplus \overline{a} \oplus \overline{d}</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1 \oplus \overline{b} \oplus \overline{a} \oplus \overline{d}</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1 \oplus \overline{b} \oplus \overline{a} \oplus \overline{d}</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1 \oplus \overline{b} \oplus \overline{a} \oplus \overline{d}</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1 \oplus \overline{b} \oplus \overline{a} \oplus \overline{d}</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1 \oplus \overline{b} \oplus \overline{a} \oplus \overline{d}</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1 \oplus \overline{b} \oplus \overline{a} \oplus \overline{d}</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1 \oplus \overline{b} \oplus \overline{a} \oplus \overline{d}</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1 \oplus \overline{b} \oplus \overline{a} \oplus \overline{d}</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1 \oplus \overline{b} \oplus \overline{a} \oplus \overline{d}</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1 \oplus \overline{b} \oplus \overline{a} \oplus \overline{d}</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1 \oplus \overline{b} \oplus \overline{a} \oplus \overline{d}</td>
<td>1</td>
</tr>
</tbody>
</table>

Verification of MPNF (46): \( 1 \oplus \overline{x_1} \overline{x_2} x_3 \overline{x_4} \oplus x_2 x_3 x_4 \).
Table 6 demonstrates that MPNF (46) $1 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5$ satisfies the assigned logic function $f(x_1, x_2, x_3, x_4)$ (Fig. 1).

**Example 3**: It is required, by using the method of figurative transformations, to simplify the Boolean function $f(x_1, x_2, x_3)$ in ESOP which is set in the canonical form [18]:

$$f = (0, 6, 14, 15).$$

**Solution**:

$$f = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 6 & 0 & 1 & 1 \\ 14 & 1 & 1 & 1 \\ 15 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} = x_1 x_2 x_3 \oplus x_1 x_4 \oplus x_2 x_4 \oplus x_3 x_4.$$

MPNF of function (48):

$$f_{\text{MPNF}} = x_1 x_2 x_3 \oplus x_1 x_4 \oplus x_2 x_4 \oplus x_3 x_4.$$ (49)

When simplifying function (48), identity (3) is taken into consideration.

The results of minimizing (49) function (48) by the method of figurative transformations and by the method of splitting ESOP conjunctions [18] coincide. The indicator of the implementation of function

$$k_{\theta} = \frac{k_l}{k_{\theta}} = \frac{39}{5},$$

where $k_l$ is the number of simple implicants, $k_{\theta}$ is the number of input variables, $k_{\theta}$ is the number of inverted variables. However, the computational complexity of the procedure for minimizing the Boolean function with figurative transformations is less.

Function (48) can be simplified using the Zhegalkin polynomial.

Zhegalkin polynomials for the constituents of function (48) are given in Table 7.

**Example 4**: It is required, by using a figurative transformation method, to simplify the Boolean function $f(x_1, x_2, x_3)$ in ESOP which is set in the canonical form [18]:

$$f = (0, 3, 5, 6, 7, 8, 9, 10, 12, 15).$$ (51)

The simplification of the Zhegalkin polynomial is carried out with the help of figurative transformations.

$$Y = 1 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_8 \oplus x_9 \oplus x_{10} \oplus x_{11} \oplus x_{12} \oplus x_{13} \oplus x_{14} \oplus x_{15} \oplus x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12} x_{13} x_{14} x_{15}.$$ (50)

Minimal functions (49) and (50) match.
Solution:

\[
\begin{array}{c|cccc}
0 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 1 & 1 \\
5 & 0 & 1 & 0 & 1 \\
6 & 0 & 1 & 1 & 0 \\
7 & 0 & 1 & 1 & 1 \\
8 & 1 & 0 & 0 & 0 \\
9 & 1 & 0 & 0 & 1 \\
10 & 1 & 0 & 1 & 0 \\
12 & 1 & 1 & 0 & 0 \\
15 & 1 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{c|cccc}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

MPNF of function \( f(x_1, x_2, x_3, x_4) \) (51):

\[
f_{\text{MPNF}} = x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_8 \oplus x_9 \oplus x_{10}. \quad (52)
\]

Table 8 gives the results from minimizing the function \( f(x_1, x_2, x_3, x_4) \) (51) in ESOP by splitting the ESOP conjunctive terms [18] and by a figurative transformation method.

\[\begin{array}{|c|c|}
\hline
\text{Method of splitting conjunctive terms} & \text{Figurative transformation method} \\
\hline
f = x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_8 \oplus x_9 \oplus x_{10} & f = x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_8 \oplus x_9 \oplus x_{10} \\
\hline
\end{array}\]

When contemplating Table 8, we see that the result of minimizing the function \( f(x_1, x_2, x_3, x_4) \) (51) by a figurative transformation method is a minimal function (52), containing 12 literals. This is 2 literals less compared to [18] and 3 literals less compared to [22].

The verification of the resulting MPNF (52) is given in Table 9. Table 9 shows that MPNF (52) \( x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus \) \( x_5 \oplus x_6 \oplus x_7 \oplus x_8 \oplus x_9 \oplus x_{10} \) satisfies the assigned logic function \( f(x_1, x_2, x_3, x_4) \) (51).

\textbf{Example 5:} It is required, by using a figurative transformation method, to simplify the Boolean function \( f(x_1, x_2, x_3, x_4) \) in ESOP, which is set in the canonical form (53) [23]:

\[
f = (0, 2, 4, 7, 9, 10, 12, 13). \quad (53)
\]

\[
\begin{array}{c|c|c|c}
\hline
\text{No.} & x_1 & x_2 & x_3 & x_4 & f(x_1, x_2, x_3, x_4) (51) & x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_8 \oplus x_9 \oplus x_{10} & f_{\text{ESOP}} \\
\hline
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
4 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
5 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\
6 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
7 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
8 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
9 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
10 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\
11 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
12 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
13 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\
14 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\
15 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
\end{array}
\]
Solution:

\[
\begin{array}{ccc|ccc}
0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 1 & 0 & 0 \\
4 & 0 & 1 & 0 & 0 & 0 \\
7 & 0 & 1 & 1 & 1 & 0 \\
9 & 1 & 0 & 0 & 1 & 1 \\
10 & 1 & 0 & 1 & 0 & 1 \\
12 & 1 & 1 & 0 & 0 & 0 \\
13 & 1 & 1 & 0 & 1 & 0 \\
\end{array}
\]

\[
\begin{array}{ccc|ccc}
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 \\
\end{array}
\]

\[
\begin{array}{ccc|ccc}
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 0 & 1 \\
\end{array}
\]

MPNF of function \( f(x_1, x_2, x_3, x_4) \) (53):

\[
\overline{f_{MPNF}} = x_1 x_2 x_4 \oplus x_1 x_3 \oplus x_4 \overline{x_2} \oplus \overline{x_1}.
\]  

The cost of (55) implementation is

\[
k_m / k_i / k_e = 4 / 9 / 3,
\]  

that matches [23], however, the function represented by the eleventh matrix in (54):

\[
\begin{array}{ccc|ccc}
0 &  &  &  &  &  \\
1 &  &  &  &  &  \\
1 &  &  &  &  &  \\
0 & 1 & 0 &  &  & \\
0 & 1 & 0 &  &  & \\
0 & 1 & 0 &  &  & \\
0 & 1 & 0 &  &  & \\
0 & 1 & 0 &  &  & \\
\end{array}
\]

can be represented in a mixed basis:

\[
\overline{f_{MPMB}} = \overline{x_1} \oplus (x_1 + x_2) \oplus (x_1 + x_2) x_3,
\]  

which represents a 3-level logic as well, but at the following cost of implementation:

\[
k_m / k_i / k_e = 3 / 7 / 1,
\]  

which is better than the implementation cost (56) of minimum function (55).

The verification of the resulting MPNF of assigned function (53) in a mixed basis (57) is given in Table 10.

**Table 10**

<table>
<thead>
<tr>
<th>No.</th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( x_4 )</th>
<th>( f(x_1, x_2, x_3, x_4) )</th>
<th>( \overline{f(x_1, x_2, x_3, x_4)} )</th>
<th>( \overline{f_{MPNF}} )</th>
<th>( \overline{f_{MPMB}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 10 shows that MPNF in a mixed basis (57) \( \overline{x_1} \oplus (x_1 + x_2) \oplus (x_1 + x_2) x_3 \) satisfies the assigned logic function \( f(x_1, x_2, x_3, x_4) \) (53).
8. Discussion of results of minimizing Boolean functions in the Reed-Muller basis by a figurative transformation method

The mathematical apparatus of Boolean function minimization by a figurative transformation method is considered in works [24–26], and others. The technique of the method of figurative transformations is given in Table 11.

Table 11

<table>
<thead>
<tr>
<th>No. of entry</th>
<th>Logic operation title</th>
<th>Reference to the text</th>
<th>Representation form</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Gluing the variables</td>
<td>(22), (23)</td>
<td>ESOP</td>
</tr>
<tr>
<td>2</td>
<td>Semi-gluing the variables</td>
<td>(24), (25), (26), (27), (29)</td>
<td>ESOP</td>
</tr>
<tr>
<td>3</td>
<td>Super-gluing the variables</td>
<td>(31), (32), (33)</td>
<td>ESOP</td>
</tr>
<tr>
<td>4</td>
<td>Incomplete super-gluing the variables</td>
<td>(34), (35)</td>
<td>ESOP</td>
</tr>
<tr>
<td>5</td>
<td>Generalized gluing the variables</td>
<td>(36), (37), (38), (39)</td>
<td>ESOP</td>
</tr>
<tr>
<td>6</td>
<td>Variable absorption</td>
<td>(40), (41), (42), (43), (44), (45)</td>
<td>ESOP</td>
</tr>
</tbody>
</table>

When simplifying Boolean functions on the Reed-Muller basis, it is advisable to use a mixed basis.

Example 6: It is required, by using a figurative transformation method, to simplify the Boolean function \( f(x_1, x_2, x_3, x_4) \) in ESOP, which is set in the canonical form (58) [23]:

\[ f = (1, 2, 4, 6, 7, 8, 9, 10, 15). \]

and is represented by a truth table (Table 14).

Table 14

<table>
<thead>
<tr>
<th>No. of entry</th>
<th>Logic operation title</th>
<th>Reference to the text</th>
<th>Representation form</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Gluing the variables</td>
<td>(22), (23)</td>
<td>ESOP</td>
</tr>
<tr>
<td>2</td>
<td>Semi-gluing the variables</td>
<td>(24), (25), (26), (27), (29)</td>
<td>ESOP</td>
</tr>
<tr>
<td>3</td>
<td>Super-gluing the variables</td>
<td>(31), (32), (33)</td>
<td>ESOP</td>
</tr>
<tr>
<td>4</td>
<td>Incomplete super-gluing the variables</td>
<td>(34), (35)</td>
<td>ESOP</td>
</tr>
<tr>
<td>5</td>
<td>Generalized gluing the variables</td>
<td>(36), (37), (38), (39)</td>
<td>ESOP</td>
</tr>
<tr>
<td>6</td>
<td>Variable absorption</td>
<td>(40), (41), (42), (43), (44), (45)</td>
<td>ESOP</td>
</tr>
</tbody>
</table>

Solution:
Function (58) is singular. Simplification of function (58) is to be performed for PESOP and ESOP.

Minimization in PESOP.

Define the stack of logic operations of the first matrix of the function \( f(x_1, x_2, x_3, x_4) \) as follows. We combine the sets of variables that contain unity in the far-right position of the set into a separate matrix. Such a matrix is presented in (59) first. In another separate matrix, we combine the sets of variables of function (58) which contain zeros in the extreme right position of the set. Such a matrix is given in (60) first. Simplification of function (58) in each matrix is performed separately.
Mathematics and cybernetics – applied aspects

Combine the results of simplification of (59) and (60) in a total matrix:

\[ f_{\text{MDNF}} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}. \]  

Further simplifications of function (58) are no longer possible. MDNF of function \( f(x_1, x_2, x_3, x_4) \) (58):

\[ f_{\text{MDNF}} = x_1 \bar{x}_3 + x_2 x_4 + x_2 \bar{x}_1 + x_2 x_1 \bar{x}_4 + x_1 x_2 \bar{x}_4. \]  

The cost of (61) implementation is:

\[ k_n / k_i / k_o = 5 / 15 / 8. \]  

The minimal function \( f(x_1, x_2, x_3) \) (61) in a mixed basis:

\[ f_{\text{MPNF}} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \]  

The selection of the stack of logic operations [9] for ESOP functions is demonstrated by the following example.

Example 8: It is required to choose the optimal stack of logic operations to simplify the Boolean function \( f(x_1, x_2, x_3, x_4) \) in ESOP set in the canonical form (69) [18]:

\[ f = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}. \]

Solution.

Function (69) is singular. To simplify (69), we choose Reed-Muller’s algebra.

The procedure for splitting conjuncterms [18] and the corresponding stack of logic operations in the first matrix of function (69) produce the following result:

\[ f_{\text{MPNF}} = (1-1) \oplus (10-0) \oplus (000-0) \oplus (0110). \]
Next, the underlined pair of terms in expression (70) is treated with the actual procedure of splitting conjuncterms [18]:

\[
\begin{pmatrix}
000 -
\end{pmatrix} * \begin{pmatrix}
00--
\end{pmatrix} = \begin{pmatrix}
0 -1-
\end{pmatrix} \oplus \begin{pmatrix}
0111
\end{pmatrix}.
\]

Upon deriving the following expression:

\[
f_{MPNF} = (1-1-) \oplus (10-0) \oplus \begin{pmatrix}
00--
\end{pmatrix} \oplus \begin{pmatrix}
0 -1-
\end{pmatrix} \oplus \begin{pmatrix}
0111
\end{pmatrix}.
\]

The algorithm of ESOP function simplification, which consists of the procedure for inserting two identical conjuncterms with the following operation of super-gluing the variables (chapter 7.2), and the corresponding stack of logic operations in the first matrix of function (69), produce the following result:

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 \\
6 & 0 & 1 & 1 \\
8 & 1 & 0 & 0 \\
11 & 1 & 0 & 1 \\
14 & 1 & 1 & 0 \\
15 & 1 & 1 & 1
\end{array}
\]

\[
f_{MPNF} = (1-1-) \oplus (10-0) \oplus (11-1) \oplus (10-1) \oplus (0111).
\]

Next, the minimal PESOP function is derived [18]:

\[
f_{MPNF} = (-1-) \oplus (-0-) \oplus (10-1) \oplus (0111). \quad (71)
\]

The weak side of the method considered here is the small practical application of the method of figurative transformations to minimize Boolean functions in a polynomial format, followed by the design and manufacture of the corresponding computational components. The negative internal factors of the method are associated with additional time costs for establishing protocols for simplifying logic functions in the Reed-Muller basis, followed by the creation of a library of protocols that have an illustration of the corresponding figurative transformations.

The prospect of further research could be the search for new rules for the transformation of symmetrical logic functions and their minimization.

9. Conclusions

1. The perfect normal form of polynomial basis functions can be represented by equivalent binary sets (10) or an equivalent binary matrix (11), which, in this case, would give the conjuncterms of polynomial functions and the operation of adding modulo two for them. Such hermeneutics should be effectively used in simplifying logic functions and when deriving the result of logic operations in the binary matrix class of functions in the Reed-Muller basis.

2. Relatively complex algorithms for simplifying logic expressions involving the procedure of inserting two identical conjuncterms of polynomial functions with the following operation of super-gluing the variables (12), (14), (16) expand variants for their application, which increases the efficiency of the procedure for minimizing Boolean functions in ESOP by a figurative transformation method.

3. The apparatus of the method of figurative transformations effectively ensures the orthogonalization of logic functions and detects the singular function.

4. In order to properly simplify functions in the Reed-Muller basis by a figurative transformation method, Reed-Muller’s algebra was refined in terms of the classical rules of equivalent transformation of ESOP and PESOP of Boolean functions of polynomial basis. The creation of polynomial algebra in terms of these rules largely solves the problem of minimizing functions on the Reed-Muller basis.

5. The effectiveness of the figurative transformation method to minimize Boolean functions in the Reed-Muller basis is demonstrated by the following examples:

- example 2 [20] – minimization of the 4-bit Boolean function;
- examples 3, 4 [18], examples 5, 6 [23], example 7 [18, 27] – minimization of 4-bit Boolean functions.

Based on the results of our comparison, it was established that the effectiveness of the method of figurative transformations to minimize Boolean functions in the Reed-Muller basis gives grounds for its application in the procedures for minimizing logic functions since the method of figurative transformations is capable of the following:

- to ensure the operational selection of the logic operations stack in the first binary matrix, which ultimately gives an optimal scenario for minimizing logic functions in the Reed-Muller basis;
- to improve the efficiency of the procedure for minimizing logic functions in the Reed-Muller basis by implementing relatively complex algorithms for simplifying logic expressions, which consist of the procedure for inserting two identical conjuncterms of ESOP functions followed by the operation of super-gluing the variables.
References