ASTES

Advances in Science, Technology and Engineering Systems Journal Vol. 3, No. 1, 168-190 (2018)

www.astesj.com

Special Issue on Recent Advances in Engineering Systems

ASTESJ

ISSN: 2415-6698

Linear algebra as an alternative approach to the synthesis of digital devices of automation and control systems

Nikolay Chernov[1], Nikolay Prokopenko[*,2,3], Vladislav Yugai[1], Nikolay Butyrlagin[2]

[1]*Systems of automation control, Southern Federal University, SFedU, Taganrog, 347928, Russia*

[2]*Information systems and radioengineering, Don State Technical University, DSTU, Rostov-on-Don, 344000, Russia*

[3]*Institute for Design Problems in Microelectronics of Russian Academy of Sciences, IPPM RAS, Zelenograd, 124681, Russia*

A R T I C L E I N F O

Article history:
Received: 18 January, 2018
Accepted: 18 January, 2018
Online: 30 January, 2018

Keywords :
multivalued component base
multivalued logic
linear logic synthesis

A B S T R A C T

The article considers linear algebra as an alternative mathematical tool of logic synthesis of digital structures to Boolean algebra and synthesis methods of digital electronic component base (ECB) on its ground. The methods of solving the applied problems of logic synthesis are shown, including the expansion of an arbitrary logic function by means of monotonic functions. The proposed mathematical apparatus actually provides the creation of digital structures on the principles of analog circuitry. It can find application in the design of multivalued digital ECB, specialized system-on-chip and analog-digital sensors with current output. The examples of synthesis of the combinational and sequential two-valued and multivalued digital devices are given. In conclusion, the advantages of linear algebra in comparison with Boolean algebra are formulated.

1. Introduction

This article is a continuation of the studies presented at the conference SIBCON-2017 [1].

Boolean algebra is known [2] as a leading mathematical tool for logic synthesis of two-valued digital structures. Almost all existing methods of logic synthesis are formed on its ground. The success of Boolean algebra is caused, among other things, by the fact that the Boolean representation of the realized logic function turned out to be rather technological: the circuit implementation of logic elements was relatively simple. TTL, C-MOS and ESL and other technologies occurred to be the most preferable for this purpose.

Despite this, the history of the development of digital microelectronics knows the attempts to replace both the approach to logic synthesis (the use of spectral representations [3 - 6] using the arithmetic polynomials [7 - 9]) and technological realization (I^2L, I^3L, ...).

The most significant contribution to the alternative theory of logic synthesis has been made by the threshold interpretation of Boolean algebra, called the threshold logic.

*Corresponding Author: Nikolay Prokopenko, Email: prokopenko@sssu.ru

Threshold logic related to one of the directions of synthesis of the digital structures [10], is constantly evolving. We know a significant number of publications devoted to this problem [11 – 14]. For two-valued functions, the threshold synthesis problems have already been solved beginning from the well-known papers of M. Dertouzos [10]. One of the advanced approaches of synthesis for the *k*-valued threshold functions is considered in this article [11].

The threshold logic was initially implemented in a two-valued version, but many publications on the multivalued threshold logic have recently appeared [14 - 17]. Why are they dedicated to the multivalued logic? The fact is that the multivalued logic is currently considered as a means of improving the quality characteristics of LSI (the ratio of “number of elements / number of links”, “total area / real-estate” of the crystal, etc.), which doesn’t require a drastic change in LSI technology. The authors of this article also follow this approach.

To develop this ideology we proposed an alternative approach to the logic synthesis of digital devices - the replacement of the mathematical tool of Boolean algebra by another body of mathematics - linear algebra. Such a replacement entailed fundamental changes in various aspects of the implementation of digital structures:

- the potential realization of Boolean logic was replaced by the current linear implementation;

- the key operating mode of the elements was replaced by the linear one;

- Boolean values of the variables took a quantitative form instead of the qualitative (logic) one;

- the value of logic was determined not by the scheme, but by the significance of signals;

- the output signal was presented by the difference of signals of two circuits operating in parallel, which improves the performance of digital structures.

Thus, the use of linear algebra as a mathematical tool of the logic synthesis ensured the development of alternative methods for the logic synthesis of current digital circuits and their practical implementations.

The authors published a number of papers [18-32] devoted to the synthesis of logic (nonthreshold) two-valued and multivalued digital structures. This paper considers the use of linear algebra as a mathematical tool for the logic synthesis of two-valued and multivalued, logic and threshold digital structures.

The fuzzy concept of the "threshold synthesis" can be interpreted in two ways:

- as a normal logic synthesis of digital structures with circuit implementation on threshold logic elements (any logic function can be implemented in this way);

- as a logic synthesis of threshold logic functions (an arbitrary logic function can be implemented by a network of threshold logic elements).

The purpose of this article is to propose logical and threshold current hardware components for constructing digital structures within the two specified areas. Within the framework of this goal, the authors' solutions for the following tasks are given below:

- since linear algebra is used as a mathematical tool for the logic synthesis of current logic structures, the article presents the main definitions and aspects of the practical use of linear algebra;

- as there is a close relationship between the threshold and monotonic functions, the article gives a definition of the monotonic function and explains the ways of representing arbitrary functions by monotonic linear functions;

- some versions of transformation of two-valued and multivalued monotonic functions into a threshold form in linear algebra are analyzed;

- versions of circuit implementation of linear threshold logic elements are considered.

2. Linear Algebra

A. Definition of linear algebra. Let $\boldsymbol{P} \to \langle P; +, \cdot, 0.1\rangle$ – field, $\langle A; +, \cdot, \theta\rangle$ – algebra with two binary operations and one nullary operation. The system $\boldsymbol{\Lambda} \to \langle A; +, \cdot, \theta, \mathbf{P}\rangle$ is called *linear algebra*, if the following conditions are met:

– the system $\boldsymbol{\Lambda} \to \langle A; +, \cdot, \theta, \boldsymbol{P}\rangle$ – linear (vector) space over the field $\boldsymbol{P}$;

– distributivities of operations + and ·

$\forall(a, b, c \in A)\ (a + b)c = ac + bc \wedge c(a + b) = ca + cb$;

– associativities of vector multiplication by elements of the field $\boldsymbol{P}$

$\forall(a, b \in A) \wedge \forall(k \in P)\ k(ab) = (ka)b = a(kb)$.

B. Extension of linear algebra.

Let $\boldsymbol{A} \to \langle A; +, \cdot, \theta\rangle$ – vector space of linear algebra $\boldsymbol{\Lambda}$, $\boldsymbol{P} \to \langle P; \Omega = \{\omega_k | k \in P\}, 0.1\rangle$ – field of linear algebra $\boldsymbol{\Lambda}$, which contains the operations ω_k, which in general case do not necessarily coincide with the operations of linear space $\boldsymbol{A}$. Then the system $\boldsymbol{\Lambda}' \to \langle\{A; +, \cdot, \theta\}, \{P; \Omega, 0, 1\}\rangle$ is called *the extension of linear algebra* $\boldsymbol{\Lambda}$.

When interpreting this algebraic system in a certain way, we can obtain algebras with different properties. For example, interpreting A as a set of terms of Boolean functions $f(x_1,\ldots, x_n)$, the operations "+" and "·" - as $max(x_1,\ldots, x_n)$, $min(x_1,\ldots, x_n)$, we obtain algebra $\boldsymbol{A} \to \langle A;\ max,\ min,\ \theta;\ \boldsymbol{P}\rangle$. Similarly, leaving the semantics of operations in the original form (i.e., defining them as ordinary arithmetic operations), we can consider the reduced system as linear algebra on the set A of vectors in a linear space. The reduced algebraic system is considered below in this form.

C. Creation of bases. To form the bases from logic variables, it is possible to construct different constructions of linearly independent vectors with given properties. The choice of operations for creation of the bases is made independently on the operations of linear space and can be determined by various (mathematical, circuit, technological and other) requirements. In the applied plan, this allows obtaining the ideologically unified (based on operations of linear space) circuit solutions of functional elements (from different implementations based on field operations).

A basis is a system of m linearly independent vectors $\{F\} = \{\varphi_0, \varphi_1,\ldots, \varphi_{m-1}\}$, which enables to describe any vector $a \in \boldsymbol{A}$ in the linear form

$$a = \textstyle\sum_{i=0}^{m-1} a_i\varphi_i\ , \qquad (1)$$

with respect to these vectors.

Each vector of the basis $\{F\} = \{\varphi_0, \varphi_1,\ldots, \varphi_{m-1}\}$ is uniquely determined by the set of coordinates $\varphi_i = \{\varphi_{i0}, \varphi_{i1},\ldots, \varphi_{i,m-1}\}$, with the help of which we can make a square matrix of order m:

$$F = \begin{bmatrix} \varphi_{10} & \varphi_{11} & \cdots & \varphi_{1,m-1} \\ \varphi_{20} & \varphi_{21} & \cdots & \varphi_{2,m-1} \\ \cdots & \cdots & \cdots & \cdots \\ \varphi_{m-1,0} & \varphi_{m-1,1} & \cdots & \varphi_{m-1,m-1} \end{bmatrix},$$

corresponding to the given basis {F}.

Two bases $\{F\} = \{\varphi_1, \varphi_2, \ldots, \varphi_{m-1}\}$ and $\{Y\} = \{\gamma_1, \gamma_2, \ldots, \gamma_{m-1}\}$, the matrices F and Y of which are inverse to each other, are reciprocal (dual). Besides,

$$F \cdot Y = E,$$

where E – diagonal matrix of order m with $\varepsilon_{ij} = 1$, when $i = j$ and $\varepsilon_{ij} = 0$, if $i \neq j$, which is an orthonormal basis $\{E\}$.

Since the resolution of an arbitrary vector a over the basis $\{F\}$ has the form

$$a = a_0\varphi_0 + a_1\varphi_1 + \cdots + a_{m-1}\varphi_{m-1},$$

then, multiplying both parts of this resolution by γ_i, $i = [0, m-1]$, we obtain:

$$a_i = a\cdot\gamma_i = a\cdot(Y\cdot\varepsilon_i) = a\cdot(F^{-1}\cdot\varepsilon_i).$$

Then the procedure for representing (i.e. obtaining the values of the expansion coefficients) of the arbitrary vector a in the given basis {F} is reduced to performing the following operations:

– construction of the basic matrix F;

– construction of the inverse basic matrix F^{-1};

– multiplication of the row-vector a by the column-vector of the matrix F^{-1} and computation of the expansion coefficients of the vector a over the basis $\{F\}$;

– writing of the expression for the vector a in the linear form (1) with respect to the basis $\{F\}$.

Example 1. Get a representation of the conjunction operation of two arguments $x_1\&x_2$ of the value 2 in the basis

$$F = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}, \quad (F)^{-1} = \frac{1}{2}\begin{bmatrix} 2 & -1 & -1 & -1 \\ 0 & 1 & -1 & 1 \\ 0 & -1 & 1 & 1 \\ 0 & 1 & 1 & -1 \end{bmatrix}.$$

Solution.

a) we represent the sequence of values of the two-valued logic function by the row-vector

$$x_1\&x_2 = [0,0,0,1];$$

b) we multiply the resulting row-vector by the columns of the inverse basic matrix $(F)^{-1}$ and obtain the row-vectors of the expansion coefficients of the represented logic function with respect to the given basis and the expression of the conjunction operation

$$x_1\&x_2 = [0,0,0,1]\cdot\frac{1}{2}\begin{bmatrix} 2 & -1 & -1 & -1 \\ 0 & 1 & -1 & 1 \\ 0 & -1 & 1 & 1 \\ 0 & 1 & 1 & -1 \end{bmatrix} =$$

$$= [0,0,0,1]\cdot\frac{1}{2}[0,1,1,-1] = \frac{x_1 + x_2 - |x_1 - x_2|}{2}.$$

It is noteworthy that the last expression has been known since 1953 [3], but the author didn't describe the method of its obtaining (more precisely, it was described later).

The authors of this article propose three approaches to the creation of the basis vectors of linear bases, depending on the operations used for this purpose:

- based on Boolean operations $\vee$, $\&$, $\oplus$, $\sim$, etc .;

- based on the truncated difference operation $\dot{-}$;

- based on the comparison operation $\geq$.

When using Boolean operations, the upper and lower "cuts" [14, 18] of variables of different orders are used as operands for creating basis vectors:

– variables of the first order $x^{(i)} = x\&i$, $x_{(i)} = x \vee i$;

– variables of the second order

$$x^{(ij)} = x\&i - x\&j, x_{(ij)} = x \vee i - x \vee j$$

and so on.

When using the *truncated difference* operation

$$x_1 \dot{-} x_2 = \begin{cases} x_1 - x_2 & \text{when } x_1 \geq x_2; \\ 0 & \text{when } x_1 < x_2, \end{cases}$$

all logical operations are replaced by some combinations of this operation on variables involving operations of linear space. For example, the upper and lower "cuts" of the first and second orders are replaced by the expressions

$$x^{(i)} = i \dot{-} (i \dot{-} x) = x \dot{-} (x \dot{-} i);$$

$$x_{(i)} = i + (x \dot{-} i) = x + (i \dot{-} x);$$

$$x^{(ij)} = (x \dot{-} j) \dot{-} (x \dot{-} i);$$

$$x_{(ij)} = (i - j) \dot{-} (x - j).$$

Similarly, Boolean operations on two variables are replaced by the following expressions:

$$x_1\&x_2 = x_1^{(x_2)} = x_2 \dot{-} (x_2 \dot{-} x_1) = x_1 \dot{-} (x_1 \dot{-} x_2);$$

$$x_1 \vee x_2 = x_{1(x_2)} = x_2 + (x_1 \dot{-} x_2) = x_1 + (x_2 \dot{-} x_1);$$

$$x_1 \oplus x_2 = x_1 + x_2 - k\{1 \dot{-} [k \dot{-} (x_1 + x_2)]\};$$

$$x_1 \ominus x_2 = x_1 - x_2 + k\{1 \dot{-} [1 \dot{-} (x_2 \dot{-} x_1)]\};$$

and others.

When using the *comparison* operation, the above expressions are reduced to the following form

$$x^{(i)} = \sum_{j=0}^{i-1} (x > j);$$

$$x^{(i)} - x^{(j)} = \sum_{t=j}^{i-j} (x > t);$$

$$x_{(i)} = x + \sum_{j=0}^{i} (j > x) = i + \sum_{j=i+1}^{k-2} (x > j);$$

$$x_{(i)} - x_{(j)} = (i-j) - \sum_{t=j}^{i-1}(x > t).$$

It should be noted that the most interesting result of the studies is the fact of constructing logical structures based on *truncated difference* and *comparison* operations other than Boolean ones. Naturally, there are certain dependencies between the *truncated difference* and *comparison* operations, some of which are given below:

-"*truncated difference – comparison*":

$$x \dot{-} i = \sum_{j=i+1}^{k-1}(x > j);$$

$$i \dot{-} x = \sum_{j=1}^{i}(j > x);$$

$$j \dot{-} (x \dot{-} i) = \sum_{p=1}^{j}[(i+p) > x];$$

$$j \dot{-} (i \dot{-} x) = (j \dot{-} i) + \sum_{p=i-1}^{i \dot{-} j}(x > p);$$

- "*comparison - truncated difference*":

$$x \le i = 1 \dot{-} \{1 \dot{-} [(i+1) \dot{-} x]\};$$

$$x < i = 1 \dot{-} [1 \dot{-} (i \dot{-} x)];$$

$$x \ge i = 1 \dot{-} [1 \dot{-} (x \dot{-} i)];$$

$$x > i = 1 \dot{-} [(i+1) \dot{-} x];$$

$$x_1 > x_2 = 1 \dot{-} [1 \dot{-} (x_1 \dot{-} x_2)];$$

$$x_1 \ge x_2 = 1 \dot{-} \{1 \dot{-} [(x_1 \dot{-} x_2) + 1]\};$$

$$x_1 < x_2 = 1 \dot{-} \{1 \dot{-} [(x_2 \dot{-} x_1) + 1]\}.$$

3. Monotonic Logic Functions

Regarding the close connection between threshold and monotonic functions, we present some known results obtained in linear algebra in a simpler and more obvious way.

A. Construction of sequences of nondecreasing componentes. Suppose we have an arbitrary vector $a = (a_0, a_1, , \dots, a_{m-1}) \in Z^m$. We renumber the components of the vector with k-ary n-bit numbers. The vector a∈ Z^m is called monotonically increasing (decreasing), if for a bitwise comparison of k-ary number codes of the components a_i and a_j we have:

$$\forall (i,j \in Z^m) i \ge j \Rightarrow a_i \ge a_j \wedge i \le j \Rightarrow a_i \le a_j. \quad (2)$$

The bitwise comparison of k-ary number codes of the components enables to single out the sequences of nondecreasing (nonincreasing) components.

The necessity to create these sequences is that for the monotonicity of the vector, condition (2) must be met in each sequence. Hence follows the first factor of simplicity of the analysis results in linear algebra: to verify the vector by monotonicity it is sufficient to establish its monotonicity within each sequence. This reduces the amount of computation and the overall complexity of the verification process.

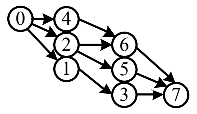
The sequences of nondecreasing components are constructed as follows. The set G of components (constituents 1) of logic functions of the chosen number of n variables is divided into n groups $G = \{g_0, \dots, g_{n-1}\}$. Each group g_{it} includes the sequences of nondecreasing components with starting numbers t, $i \le t \le n-1$, determined by the following relation

$$k^i - k^{i-1} \le j \le k^{i+1} - k^i - 1,$$

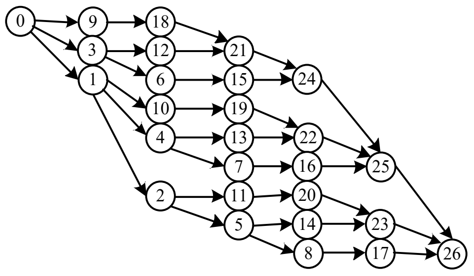
in this case we take $(k^i - k^{i-1})\big|_{i=0} = 0$. The starting numbers are component numbers that should be compared with the values of the other elements in the sequence. Each sequence contains components with numbers the decimal equivalents of which are determined as $t + k^t, t + k^{t+1}, \dots, t + k^{n-1}$. As a result, the entire sequence generally takes on the form

$$g_{it} = \{g_t, g_t + k^t, g_t + k^{t+1}, \dots, g_t + k^{n-1}\}.$$

Thus, it is possible to construct the sequences of nondecreasing components for the given values of k and n. The sequence graphs of nondecreasing components for $k = 2$, $k = 3$ and $n = 3$ are shown in Figure 1. The digits in the circles denote the decimal numbers of the vector components, and the sequence itself includes a certain vertex and the nearest right vertexes connected with it by the lines.



a



b

Figure 1: Sequence graphs of the nondecreasing components for $k = 2$, $k = 3$ and $n = 3$.

Example 2. Construct sequences of nondecreasing components for $k = n = 3$. Since $n = 3$, there are 3 groups of sequences with starting numbers 0, 1 and 2 for the initial data under consideration.

For group "0", the value of t_0 varies in the range

$$3^0 - 3^{-1} \le t_0 \le 3^1 - 3^0 - 1,$$

or $\quad 0 \le t_0 \le 1.$

Therefore, this group includes two sequences with the initial elements $t_{00} = 0$ and $t_{01} = 1$. The first sequence g_{00} of the group "0" contains the elements with numbers t_{00}, $t_{00} + 3^0$, $t_{00} + 3^1$, $t_{00} + 3^2$, i.e. 0, 1, 3, 9. The second sequence g_{01} of the group "0" comprises the elements with numbers t_{01}, $t_{01} + 3^0$, $t_{01} + 3^1$, $t_{01} + 3^2$, i.e. the elements with numbers 1, 2, 4, 10.

Similarly, for group 1, the value of t_1 is

$$3^1 - 3^0 \leq t_1 \leq 3^2 - 3^1 - 1,$$

or

$$2 \leq t_1 \leq 5.$$

Consequently, this group includes sequences with the initial elements $g_{1t} = 2 \ldots 5$, and each sequence of the group comprises the elements with numbers t_{1j}, $t_{1j} + 3^1$, $t_{1j} + 3^2$.

Finally, for group 2, the value of t_2 is

$$3^2 - 3^1 \leq t_2 \leq 3^3 - 3^2 - 1,$$

or

$$6 \leq t_2 \leq 17,$$

i.e. this group contains sequences with the initial elements $g_{2t} = 6 \ldots 17$, and each sequence includes the elements with numbers t_{2j}, $t_{2j} + 3^2$. Thus, we obtain the sequence structure of nondecreasing components in the form shown in Figure. 1*b*.

The monotonicity property of vectors from Z^m enables to obtain various representations of an arbitrary vector by means of monotonic vectors. Let's consider such representations in the following three variants:

– the difference of two monotonic vectors that have a value greater than the significance of the represented vector;

– the algebraic sum of monotonic vectors of the same value as the original vector;

– the algebraic sum of monotonic vectors of the value, the smaller value of the represented vector.

B. The problem of representing the logic function by the difference of two monotonic vectors of greater significance is solved by the following algorithm [18]:

Algorithm 1.

```
1. i=0.   $b_0 = a_0$,   $c_0 = b_0 - a_0$;
2. $1 \leq i \leq m-1$.
   $b_i = \begin{cases} b_{i-1} \text{ when } a_i < a_{i-1} \\ b_{i-1} + a_i - a_{i-1} \text{ when } a_i \geq a_{i-1} \end{cases}$,  $c_i = b_i - a_i$;
3. $i = m$, the end.
```

Here i and i–1 are neighboring indices of the elements in the sequence of nondecreasing components, $m = k^n$. The validity of the algorithm results from the following elementary arguments.

1. Let $b_0 = a_0$, then $c_0 = b_0 - a_0 = 0$.

2. Let the pair of components a_δ and a_γ satisfies the condition of monotonicity, i.e. $a_\delta \leq a_\gamma$. In this case

$$a_\delta - a_\gamma \geq 0,$$

and, consequently, from the identity

$$b_\gamma = b_\delta + a_\gamma - a_\delta,$$

it follows that

$$b_\gamma \geq b_\delta,$$

i.e. b_γ also meets the monotonicity condition.

3. If the pair of components a_γ and a_δ doesn't fulfill the condition of monotonicity, i.e. $a_\delta \geq a_{\gamma\gamma}$, then, taking $b_\gamma = b_\delta$ we remain the monotonicity condition for the components of the vector b again.

Example 3. Construct a representation of the vector $a = \{0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0\}$ as a difference of two monotonic vectors for $k = 2$, $n = 4$, $m = k^n = 16$.

The sequence of nondecreasing components for this case has the following form:

Table 1

g	Sequences	Values of components
0	0, 1, 2, 4, 8	0, 1, 1, 1, 1
1	1, 3, 5, 9,	1, 0, 0, 0
2	2, 6, 10	1, 0, 1
	3, 7, 11	0, 1, 1
3	4, 12	1, 1
	5, 13	0, 1
	6, 14	0, 1
	7, 15	1, 0

We construct the components of the vectors b and c:

$$\begin{array}{ll} b_0 = 0 & c_0 = 0 \\ b_1 = b_1 + a_1 - a_0 = 1 & c_1 = 0 \\ b_2 = b_1 + a_2 - a_1 = 1 & c_2 = 0 \\ b_3 = b_1 = 1 & c_3 = 1 \\ b_4 = b_2 + a_4 - a_2 = 1 & c_4 = 0 \\ b_5 = b_1 = 1 & c_5 = 0 \\ b_6 = b_2 = 1 & c_6 = 1 \\ b_7 = b_3 + a_7 - a_3 = 2 & c_7 = 1 \\ b_8 = b_4 + a_8 - a_4 = 1 & c_8 = 0 \\ b_9 = b_5 + a_9 - a_5 = 1 & c_9 = 1 \end{array}$$

$$b_{10} = b_6 + a_{10} - a_6 = 2 \qquad c_{10} = 1$$

$$b_{11} = b_7 + a_{11} - a_7 = 2 \qquad c_{11} = 1$$

$$b_{12} = b_8 + a_{12} - a_8 = 1 \qquad c_{12} = 0$$

$$b_{13} = b_9 + a_{13} - a_9 = 2 \qquad c_{13} = 1$$

$$b_{14} = b_6 + a_{14} - a_6 = 1 \qquad c_{14} = 1$$

$$b_{15} = b_7 = 2 \qquad c_{15} = 2$$

The described algorithm demonstrates one more fact, which confirms the simplicity of analysis in linear algebra.

As it can be seen from the example, when using the described algorithm in the general case, the value of the original vector does not coincide with the significance of the resolution vectors.

C. The expansion of the arbitrary logic function of n arguments into the algebraic sum of monotonic logic functions of the same value of the following form

$$f(\tilde{x}^{(n)}) = (-1)^i \sum_{i=0}^{p} \varphi_i(\tilde{x}^{(n)}),$$

where $p \leq kn$, $\varphi_i(\tilde{x}^{(n)})$ – monotonic expansion functions with the property $\varphi_1 \supset \varphi_2 \supset \ldots$, are made in accordance with the following algorithm [18].

Algorithm 2.

1. We choose a minimal summation of the positive summands of the arithmetic-logical representation of the logic function $f(\tilde{x}^{(n)})$ covering (in the logical sense) all other positive summands, and combine it by the operation $\vee$. Thus, we form the first expansion function $\varphi_1(\tilde{x}^{(n)})$. To remain the equality, all possible logical products of summands from $\varphi_1(\tilde{x}^{(n)})$ with the signs defined as $(-1)^j$, where $1 < j < s$, s is a number of summands in $\varphi_1(\tilde{x}^{(n)})$, are added to $\varphi_1(\tilde{x}^{(n)})$. Then we reduce similar terms and represent the initial logic function $f(\tilde{x}^{(n)})$ in the following form $f(\tilde{x}^{(n)}) = \varphi_1(\tilde{x}^{(n)}) - f_1(\tilde{x}^{(n)})$, where $f_1(\tilde{x}^{(n)})$ is a remainder of the initial function after reduction of similar terms.

2. We repeat clause 1 for $f_1(\tilde{x}^{(n)})$. As a result, the initial logic function is represented in the following form

$$f(\tilde{x}^{(n)}) = \varphi_1(\tilde{x}^{(n)}) - \varphi_2(\tilde{x}^{(n)}) + f_2(\tilde{x}^{(n)}).$$

3. We repeat clause 2 until the remainder of the initial logic function becomes zero.

Since the number of arguments of the logic function is n, and the violation of monotonicity is possible in each of the k values of each argument, then, the maximum number of the expansion functions doesn't exceed $2n(k-1)$.

The process convergence follows from the fact that each successive resolution vector eliminates some violation of monotonicity in the original vector and doesn't introduce new monotonicity violations, since it is monotonic itself.

The problem of representing the arbitrary logic function by the algebraic sum of monotonic functions of the same value can be solved analytically. As is known [13], the minimal disjunctive normal form (DNF) of the monotonic logic function doesn't contain inversions over variables. Consequently, the given logic function must be reduced to the representation in the form of the algebraic sum of such functions. To do this, we must perform the following actions:

– conversion of the Boolean expression of the logic function into the linear one using the following identities

$$x_1 \vee x_2 = x_1 + x_2 - x_1 x_2;$$

$$\bar{x} = 1 - x;$$

– inverse transformation to the given (that doesn't contain inversions over variables) form.

Example 4. Obtain the two-valued logic function mapping of three arguments

$$f(\tilde{x}^{(3)}) = (0,1,1,0,1,0,0,1),$$

into the algebraic sum of monotonic logic functions.

Solution. The logic function is represented by a vector of values. Its linear representation has the following form

$$f(\tilde{x}^{(3)}) = x_1 + x_2 + x_3 - 2x_1x_2 - 2x_1x_3 - 2x_2x_3 + 4x_1x_2x_3.$$

Now we obtain the mapping of this function into the algebraic sum of monotonic functions. The first expansion function is formed from the first three summands, covering in aggregate all the remaining positive terms of sum:

$$f(\tilde{x}^{(3)}) = (x_1 \vee x_2 \vee x_3 + x_1x_2 + x_1x_3 + x_2x_3 - x_1x_2x_3) -$$
$$-2x_1x_2 - 2x_1x_3 - 2x_2x_3 - 4x_1x_2x_3 =$$
$$= \varphi_1(\tilde{x}^3) - x_1x_2 - x_1x_3 - x_2x_3 + 3x_1x_2x_3.$$

The subsequent expansion functions are obtained in the same way:

$$f(\tilde{x}^{(3)}) = \varphi_1(\tilde{x}^{(3)}) - (x_1x_2 \vee x_1x_3 \vee x_2x_3 + x_1x_2x_3 +$$
$$+x_1x_2x_3 + x_1x_2x_3 + x_1x_2x_3 - x_1x_2x_3) + 3x_1x_2x_3 =$$
$$= \varphi_1(\tilde{x}^{(3)}) - \varphi_2(\tilde{x}^{(3)}) + \varphi_3(\tilde{x}^{(3)}).$$

To obtain the threshold representation of the initial function, it suffices to transform the monotonic functions of the resulting expansion into the threshold form and to perform their algebraic addition.

Each conjunctive term of any logical expansion function is transformed into the threshold form in accordance with the following identical equation

$$x_i x_j \ldots x_s = P[(x_i + x_j + \cdots + x_s) > t - 1],$$

where t - a number of arguments in the term, P - the predicate symbol. Then all the terms are reduced to the common value of the right-hand side with the introduction of the corresponding coefficients for the variables, then they are added together and the threshold value of the received sum is determined.

Let's explain the transformation of the monotonic logic function into the threshold form with the function from the previous example.

With the transformation of $\varphi_1(\tilde{x}^{(3)})$ everything is simple:

$$\varphi_1(\tilde{x}^{(3)}) = x_1 \vee x_2 \vee x_3 = P_1[(x_1 + x_2 + x_3) > 0];$$

The transformation of $\varphi_2(\tilde{x}^{(3)})$ looks somewhat more complicated:

$$\varphi_2(\tilde{x}^{(3)}) = x_1x_2 \vee x_1x_3 \vee x_2x_3 =$$

$$= P_2[(x_1 + x_2) > 1 + (x_1 + x_3) > 1 + (x_2 + x_3) > 1] =$$

$$P_2[(2x_1 + 2x_2 + 2x_3) > 2] = P_2[(x_1 + x_2 + x_3) > 1];$$

The transformation of $\varphi_3(\tilde{x}^{(3)})$ is produced just as easily as $\varphi_1(\tilde{x}^{(3)})$:

$$\varphi_3(\tilde{x}^{(3)}) = x_1x_2x_3 = P_3[(x_1 + x_2 + x_3) > 2].$$

Thus, to realize the considered logic function, three threshold logic elements are required. The simplicity of the transformations in linear algebra becomes obvious.

D. Representation of the arbitrary vector by the vectors of lower significance. This problem has many solutions. The variant considered below assumes the solution of this problem in two stages:

- obtaining a representation of the arbitrary vector from Z^m by the vectors of lower significance;

- transformation of the obtained representation into the representation by means of monotonic vectors.

Let us first consider the solution of the first stage of the problem. The representation of the arbitrary vector by the lower-valued vectors can be obtained by weighting (each resolution vector is included into the final representation with some weight coefficient) or unitary (each resolution vector is included into the final representation with a unitary weight coefficient) coding.

1. It is known from the theory of numbers [33] that any number a can be uniquely represented in the following form:

$$\sum_1^s a_{s-i}k^{s-i}, \qquad (3)$$

where $s = [\log_k a]$ – the closest integer to $\log_k a$, a_{s-i} – the values of the expansion coefficients of the number a in the k-valued number system. Due to the uniqueness, this relation determines the isomorphism between any number a and the described representation of this number.

The quantity a_{s-i} can be determined from the following relation:

$$a_{s-i} = \frac{a\,(mod\ k^{s-i+1}) - a\,(mod\ k^{s-i})}{k^{s-i}}.$$

This operation is linear

$$a + b = \sum_{i=1}^{s} a_{s-i}\,k^{s-i} + \sum_{i=1}^{s} b_{s-i}\,k^{s-i} =$$

$$= \sum_{i=1}^{s}(a_{s-i} + b_{s-i})\,k^{s-i};$$

$$\lambda a = \sum_{i=1}^{s} \lambda a_{s-i}\,k^{s-i} = \lambda \sum_{i=1}^{s} a_{s-i}\,k^{s-i},$$

thus, it is applicable to the vectors of the linear space. If now we associate the vector a to the number a in the linear space Z^m, and the weighted sums of the vectors a_{s-i} of the spaces K_1^m and K_2^m to the weighted sums $\sum_{i=1}^{s_1} a_{s_1-i}\,k^{s_1-i}$ and $\sum_{i=1}^{s_2} a_{s_2-i}\,k^{s_2-i}$ correspondingly, then the expression (3) is equivalent to the representation of the vector $a \in Z^m$ by the weighted sum of the vectors from K_1^m or K_2^m respectively. Since in both spaces the vector a is uniquely represented, both representations are isomorphic. Let's call such a representation by coding, and the corresponding linear operator - by *the encoding operator.*

Let the vector $a \in Z^m$ in the canonical basis be represented as

$$a = a_1e_1 + a_2e_2 + \cdots + a_me_m.$$

Using expression (3), we expand each coordinate of the vector a with respect to the bases *k1* and *k2*. As a result, we obtain its representation in the following form

$$a_i = \sum_{i=1}^{s_1} a_{i,s_1-i}\,k_1^{s_1-i},$$

or

$$a_j = \sum_{j=1}^{s_2} a_{j,s_2-j}\,k_2^{s_2-j},$$

where $a_{i,s-i}$ – the expansion components of the number a_i (i – e vector components a_i of the vector representation a).

Then the vector representation a in each of the spaces can be described in the following form

$$a = \left(a_{1,s-1}k^{s-1} + a_{1,s_1-2}k^{s-2} + \cdots + a_{1,0}k^0\right) +$$

$$+\left(a_{2,s-1}k^{s-1} + a_{2,s_1-2}k^{s-2} + \cdots + a_{2,0}k^0\right) + \cdots$$

$$+ \left(a_{m,s-1}k^{s-1} + a_{m,s_1-2}k^{s-2} + +a_{m,0}k^0\right),$$

or

$$a = (a_{1,s-1} + a_{2,s-1} + \cdots + a_{m,s-1})k^{s-1} +$$

$$+(a_{1,s-2} + a_{2,s-2} + \cdots + a_{m,s-2})k^{s-2} + \cdots$$

$$+(a_{1,0} + a_{2,0} + \cdots + a_{m,0})k^0.$$

Thus, the arbitrary vector a can be written in the following form

$$a = \sum_{j=1}^{s}\left(\sum_{i=1}^{m} a_{i,s-j}\right)k^{s-j},$$

i.e. a is represented by the weighted sum s of the vectors from K_1^m or K_2^m

Example 5. Let $k_1 = 10$, $n = 1$. The problem is to represent the vector $a = \{0, 1, 3, 9, 7, 6, 9, 4, 3, 8\}$ by its k_2-weighted and k_2-unitary sums of the vectors when $k_2 = 3$.

Solution. Since $\lceil log_{k_2} k_1 \rceil = \lceil log_3 10 \rceil = 2$, then the vector a can be represented by the weighted sum of three vectors a_2, a_1 and a_0. The coordinates of the highest weight vector a_2 are determined as

$$a_2 = \frac{a\,mod 3^3 - a\,mod 3^2}{3^2} =$$

$$= \frac{(0139769348) - (0130760438)}{9} = (0001001000).$$

The components of the subsequent vectors a_1 and a_0 are determined similarly:

$$a_1 = \frac{a\,mod 3^2 - a\,mod 3^1}{3^1} =$$

$$= \frac{(0130760348) - (0100100102)}{3} = (0010220112);$$

$$a_0 = \frac{a\,mod 3^1 - a\,mod 3^0}{3^0} =$$

$$= \frac{(0100100102) - (0000000000)}{1} = (0100100102).$$

Thus,

$$a = \textstyle\sum_0^3 a_i 3^i = (0001001000)3^2 + (0010220112)3^1 + (0100100102)3^0.$$

2. We now represent the arbitrary number a as follows:

$$a = \textstyle\sum_{i=0}^{s-1} (a - \sum_{i=0}^{s-1} a_i)\,\&(k_2 - 1), \quad (4)$$

where s - the nearest larger integer

$$s = \left\lceil \frac{k_1 - 1}{k_2 - 1} \right\rceil.$$

For the reasons stated above, this representation is also an isomorphism.

If we associate each coordinate of the vector a with its representation in form (4), we obtain:

$$a = \left[a_1\&(k_2-1) + \left(a_1 - a_{11}\&(k_2-1)\right) + \cdots + a_1 - \sum_{i+1}^{s-1} a_{1i}\right]\&(k_2-1) +$$

$$+\left[a_2\&(k_2-1) + \left(a_2 - a_{21}\&(k_2-1)\right) + \cdots + a_2 - \sum_{i+1}^{s-1} a_{2i}\right]\&(k_2-1) + \cdots$$

$$+\left[a_m\&(k_2-1) + \left(a_m - a_{m1}\&(k_2-1)\right) + \cdots + a_m - \sum_{i+1}^{s-1} a_{mi}\right]\&(k_2-1).$$

Thus,

$$a = \sum_0^{s-1} a_i,$$

where $a_i = (a - \sum_{i+1}^{s-1} a_i)\&(k_2 - 1)$, i.e. the arbitrary vector $a \in Z^m$ is represented by the k_2-unitary sum of vectors from Z^m.

Example 6. Present the original vector from the previous example using unitary coding.

Solution. Since under the same initial conditions $s = \left\lceil \frac{10-1}{3-1} \right\rceil = 4$, the vector a can be represented by the unitary sum of five vectors a_4, a_3, a_2, a_1 and a_0, where

$a_4 = a\&(3-1) = (0139769438)\&(3-1) = (0122222222)$;

$a_3 = (a - a_4)\&(3-1) = (0012222212)$;

$a_2 = (a - a_4 - a_3)\&(3-1) = (0002222002)$;

$a_1 = (a - a_4 - a_3 - a_2)\&(3-1) = [0002102002]$;

$$a_0 = (a - a_4 - a_3 - a_2 - a_1)\&(3-1) =$$

$$= [0001001000].$$

Thus,

$$a = \sum_0^3 a_i = (0122222222) + (0012222212) +$$

$$+(0002222002) + (0002102002) +$$

$$+0001001000.$$

The solution of the second stage of the problem, i.e. the transformation of the obtained representation into the representation by means of monotonic lower-valued vectors, can be obtained on the base of algorithm 2.

Thus, the body of the linear space theory determines easy-to-use means of representing arbitrary vectors of any given value by monotonic vectors of greater, similar or lesser significance, and also converting monotonic functions to the threshold form.

4. Synthesis of Logic and Threshold Elements

In linear algebra, the logic function is represented by a weighted algebraic sum of terms. As a whole the variables, terms and function take values from the set of numbers 0, 1, ..., k - 1. The algebraic sum is realized by operations of the linear space, and individual terms - depending on the choice made - by other operations chosen for technical, technological or operational reasons. In this article, *"Truncated Difference"* and *"Comparison"* are used as such operations. In the future, the developers of linear LSIs can be motivated to choose other operations. Thus, the process and the possibilities of logic synthesis will be further demonstrated on the bases based on the *truncated difference* and *comparison* operations.

To solve the problems of logic synthesis in linear algebra, first, it is necessary to solve the problem of the formation of bases.

In Boolean algebra, 17 previously defined functionally complete systems of logic functions of two arguments are known [34]. Selecting any of them, you can get its linear analog and perform a logic synthesis of the circuit that implements the given logic function according to the established algorithm. Therefore, it is possible to construct bases as a mapping of Boolean functionally complete systems to a linear space. For *truncated difference* and *comparison* operations, the bases can be formed directly.

To form the bases, we need:

- a set of operations;

- a set of generating functions.

The following operation are considered as the basic ones:

- logical operations in their traditional understanding;

- operation *"truncated difference";*

- operation *"comparison".*

Constants 1, logical variables x_1, x_2, ..., x_n and their totals subjected to the effect of basic operations are used as generators for the formation of bases.

The following methods of logic synthesis of digital structures are considered:

- direct synthesis in a given basis (logic synthesis in the linear space);

- synthesis with preliminary transformation of the value of the given function (for example, a multivalued two-digit implementation);

- synthesis with preliminary expansion of the given function into the algebraic sum of monotonic functions (threshold synthesis).

The first method is the most obvious and consists in the representation of the realized logic function in the chosen basis.

The second method presupposes a preliminary representation of the realized function in the form of an algebraic sum of logic functions larger, smaller or equal to the original value. The methods of this transformation have been described in the previous section.

Finally, the third method consists of the preliminary representation of the realized function by the algebraic sum of monotonic logic functions. In this case, each monotonic function is realized by a single threshold element.

Note that the synthesis of digital structures with increasing of their complexity, as usual, requires the involvement of methods of system engineering.

A. Basic operations of logic synthesis. The set of operations most often used to represent logic functions will be considered as basic operations of logic synthesis. In Boolean algebra, such operations can be considered as operations of the basic functional system AND (&), OR (∨), ($\overline{\ }$), and also the operations "modulo 2 sum" (⊕), "logic equivalence" (~) "dual stroke" (↓), "Sheffer stroke" (|), "prohibition with respect to x_1 (x_2)" ($x_1\bar{x}_2$), "implication from x_1 to x_2)" ($x_1 \vee \bar{x}_2$), majority operation x_1 # x_2 # x_3.

By *basic operations of linear algebra* we mean arithmetic operations (operations of the linear space), as well as operations used to form bases. We classify the current realizations of logical operations: the difference module, *the truncated difference*, the arithmetic sum and the difference, the multiplication by a constant coefficient, the operations of changing the sign to them. Perhaps there are other arithmetic operations suitable for the use in linear logic synthesis, but they are not considered in this article.

Since our task is logic synthesis in linear algebra, we start with the logic synthesis of basic operations of linear algebra.

Algebraic sum. The operation is realized by the assembly connection of conductors, over which the currents flow. The conditional graphic representation of the operation of the algebraic sum is shown in Figure 2.

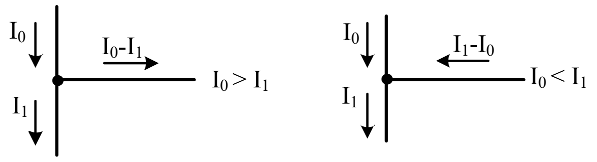


Figure 2: Conditional graphical representation of the algebraic addition of currents.

The arithmetic sum differs from the algebraic one in that in the latter case the signals of only one sign (i.e. only inflowing or only flowing out) are fed to the input of the element. From the output, the sum of the signals is removed. The positive direction of the input currents is the current direction to the connection point Σ_1, i.e. the positive current flowing into the connection point. The negative direction of the input currents is the direction of the current from the connection point of the conductors Σ_1, i.e. negative - the current flowing from the connection point. For the output (resulting) current, the positive direction is the direction from the connection point Σ_1, and the negative direction - to the connection point Σ_1,

The number of positive and negative inputs is determined by the function being implemented. Since the logical variables take values from the positive semi-axis of the numerical axis, when subtracting, it is necessary to meet the condition that the total sum of the negative summands (current quanta) doesn't exceed the total sum of the positive terms of the sum. Otherwise, instead of the algebraic sum, the *truncated difference* operation is performed, in which the subtraction from zero is a logical component of this operation and which carries the meaning of the *comparison* operation with zero (Figure 3)

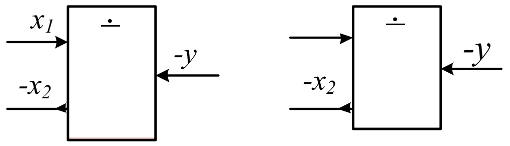


Figure 3: Conditional graphic representation o*f the truncated difference* operation.

The operation of multiplying (dividing) by a constant coefficient consists in multiplying the input signal by several outputs and then combining the outputs of the multiplied signal (when multiplying) or outputting some part of the input signal (when dividing), as shown in Figure 4.

The comparison operation is used to determine the relationship between the compared variables. It can be performed in two forms: in the relative form, i.e. in the form of determining the difference in the values of the variables, or in the absolute form, i.e. in the form of the "more-less" comparison itself. The conditional graphic symbols of the comparison elements in these forms are shown in Figure 5.

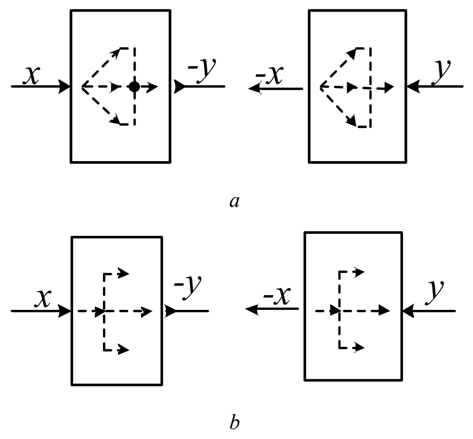


Figure 4: Conditional graphic representation of the operation of multiplication *(a)* and division *(b).*

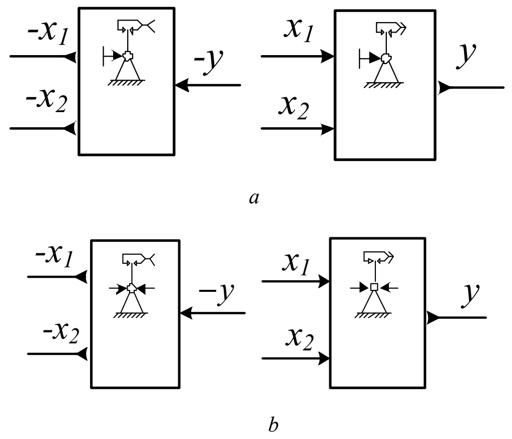


Figure 5: Conditional graphic representation of the *comparison* operation: relative *(a)* and absolute *(b).*

The operation of *changing the sign* of terms allows changing the sign of the term, thereby transforming it from the summand to the subtrahend and vice versa. It is performed by the current inverter, the conditional graphic representation of which is shown in Figure 6.

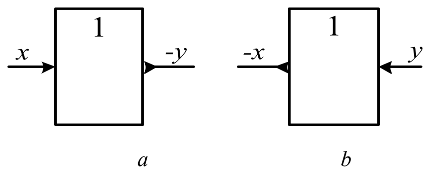


Figure 6: Conditional graphic representation of the operation of changing the sign.

The operation of changing the sign can functionally be combined with the operations of multiplication and division by the constant coefficient.

B. Formation of bases. The bases being formed are bases of the linear space, and according to the form they are sets of variables and their various combinations (terms) combined by the selected operations. We can approach to the formation of bases from different sides. For the two-valued synthesis, as it was shown above, it is possible, for example, to construct bases equivalent to 17 functionally complete systems (logical-arithmetic), and then create their analogs in linear algebra.

To form the multivalued bases, we can use the cuts of multivalued variables (10 combinations of such cuts are proposed in [18]), and then construct their analogs in linear algebra. Such a solution is also applicable for the two-valued bases. The problem of forming the bases is solved uniformly for the cuts of any type, so we demonstrate its solution on the basis of the upper cuts of the first level, which are analogs of the functionally complete system AND, NOT (OR, NOT).

The bases of this type for $k = 2$, $n = 2$ and $n = 3$, and for $k = 3$, $n = 2$ are given below. For each basis, the basic and inverse matrices are presented. Two-digit operations & and ∨ and their multivalued analogs $min(x_1x_2)$ and $max(x_1x_2)$ are used as generating operations. Therefore, the bases of this type are called logical-arithmetic.

The two-valued logical-arithmetic bases of two variables have the form:

- conjunctive (AND, NOT)

$$\&(\mathrm{B1},\mathrm{B1})^{(2)}=\begin{bmatrix}1\\x_1^{(1)}\\x_2^{(1)}\\x_1^{(1)}x_2^{(1)}\end{bmatrix}=\begin{bmatrix}1\\x_1\\x_2\\x_1\&x_2\end{bmatrix}=\begin{bmatrix}1&1&1&1\\0&1&0&1\\0&0&1&1\\0&0&0&1\end{bmatrix};$$

$$\left(\&(\mathrm{B1},\mathrm{B1})^{(2)}\right)^{-1}=\begin{bmatrix}1&-1&-1&1\\0&1&0&-1\\0&0&1&-1\\0&1&1&1\end{bmatrix};$$

- disjunctive (OR, NOT)

$$\vee(\mathrm{B1},\mathrm{B1})^{(2)}=\begin{bmatrix}1\\x_1^{(1)}\\x_2^{(1)}\\x_1^{(1)}\vee x_2^{(1)}\end{bmatrix}=\begin{bmatrix}1&1&1&1\\0&1&0&1\\0&0&1&1\\0&1&1&1\end{bmatrix};$$

$$(\vee(\mathrm{B1},\mathrm{B1})^{2})^{-1}=\begin{bmatrix}1&0&0&-1\\0&0&-1&1\\0&-1&0&1\\0&1&1&-1\end{bmatrix}.$$

The two-valued logical-arithmetic bases of three variables have the similar form:

$$\&(\mathrm{B1},\mathrm{B1},B1)^{(2)}=\begin{bmatrix}1\\x_1\\x_2\\x_3\\x_1x_2\\x_1x_3\\x_2x_3\\x_1x_2x_3\end{bmatrix}=\begin{bmatrix}1&1&1&1&1&1&1&1\\0&1&0&1&0&1&0&1\\0&0&1&1&0&0&1&1\\0&0&0&0&1&1&1&1\\0&0&0&1&0&0&0&1\\0&0&0&0&0&1&0&1\\0&0&0&0&0&0&1&1\\0&0&0&0&0&0&0&1\end{bmatrix};$$

$$\left(\&(\mathrm{B1},\mathrm{B1},B1)^{(2)}\right)^{-1}=\begin{bmatrix}1&-1&-1&-1&1&1&1&-1\\0&1&0&0&-1&-1&0&1\\0&0&1&0&-1&-1&-1&1\\0&0&0&0&1&0&0&-1\\0&0&0&1&0&-1&-1&1\\0&0&0&0&0&1&0&-1\\0&0&0&0&0&0&1&-1\\0&0&0&0&0&0&0&1\end{bmatrix};$$

$$\vee\,(\mathrm{B1},\mathrm{B1},B1)^{(2)}=\begin{bmatrix}1\\x_1\\x_2\\x_3\\x_1\vee x_2\\x_1\vee x_3\\x_2\vee x_3\\x_1\vee x_2\vee x_3\end{bmatrix}=\begin{bmatrix}1&1&1&1&1&1&1&1\\0&1&0&1&0&1&0&1\\0&0&1&1&0&0&1&1\\0&0&0&0&1&1&1&1\\0&1&1&1&0&1&1&1\\0&1&0&1&1&1&1&1\\0&0&1&1&1&1&1&1\\0&1&1&1&1&1&1&1\end{bmatrix};$$

$$\left(\vee\,(\mathrm{B1},\mathrm{B1},B1)^{(2)}\right)^{-1}=\begin{bmatrix}1&0&0&0&0&0&0&-1\\0&0&0&0&0&0&-1&1\\0&0&0&0&0&-1&0&1\\0&0&0&-1&0&1&1&-1\\0&0&0&0&-1&0&0&1\\0&0&-1&0&1&0&1&-1\\0&-1&0&0&1&1&0&-1\\0&1&1&1&-1&-1&-1&1\end{bmatrix}.$$

The three-valued logical-arithmetic analogs of these bases look like this:

- conjunctive

$$min(B1,B1)^{(3)}=\begin{bmatrix}1\\x_1^{(1)}\\x_1^{(2)}\\x_2^{(1)}\\min\left(x_1^{(1)},x_2^{(1)}\right)\\min\left(x_1^{(2)},2x_2^{(1)}\right)\\x_2^{(2)}\\min\left(2x_1^{(1)},x_2^{(2)}\right)\\min\left(x_1^{(2)},x_1^{(2)}\right)\end{bmatrix}=\begin{bmatrix}1&1&1&1&1&1&1&1&1\\0&1&1&0&1&1&0&1&1\\0&1&2&0&1&2&0&1&2\\0&0&0&1&1&1&1&1&1\\0&0&0&0&1&1&0&1&1\\0&0&0&0&1&2&0&1&2\\0&0&0&1&1&1&2&2&2\\0&0&0&0&1&1&0&2&2\\0&0&0&0&1&1&0&1&2\end{bmatrix};$$

$$\left(min(B1,B1)^{(3)}\right)^{-1}=\begin{bmatrix}1&-1&0&-1&1&0&0&0&0\\0&0&0&2&-2&0&-1&1&0\\0&0&0&-1&1&0&1&-1&0\\0&2&-1&0&-2&1&0&0&0\\0&0&0&0&2&-1&0&-1&1\\0&0&0&0&0&0&0&1&-1\\0&-1&1&0&1&-1&0&0&0\\0&0&0&0&0&1&0&0&-1\\0&0&0&0&-1&0&0&0&1\end{bmatrix};$$

- disjunctive

$$max(B1,B1)^{(3)}=\begin{bmatrix}1\\x_2^{(1)}\\x_2^{(2)}\\x_1^{(1)}\\max\left(x_2^{(1)},x_1^{(1)}\right)\\max\left(x_2^{(2)},2x_1^{(1)}\right)\\x_1^{(2)}\\max\left(2x_2^{(1)},x_1^{(2)}\right)\\max\left(x_2^{(2)},x_1^{(2)}\right)\end{bmatrix}=\begin{bmatrix}1&1&1&1&1&1&1&1&1\\0&0&0&1&1&1&1&1&1\\0&0&0&1&1&1&2&2&2\\0&1&1&0&1&1&0&1&1\\0&2&2&1&2&2&2&2&2\\0&0&0&0&1&1&0&2&2\\0&1&2&0&1&2&0&1&2\\0&1&2&2&2&2&2&2&2\\0&1&2&1&1&2&2&2&2\end{bmatrix};$$

$$\left(max(B1,B1)^{(3)}\right)^{-1}=\begin{bmatrix}1&0&0&0&-1&0&0&0&0\\0&0&0&0&2&0&0&-1&0\\0&-1&0&0&-1&0&0&1&0\\0&0&0&0&2&-1&0&0&0\\0&0&0&0&-2&1&0&1&-1\\0&2&-1&0&0&0&0&-1&1\\0&0&0&-1&-1&1&0&0&0\\0&0&0&2&0&-1&-1&0&1\\0&-1&1&-1&1&0&1&0&-1\end{bmatrix}.$$

To obtain linear analogs of the above logical-arithmetic bases, we can use the relations that establish the connection between logical operations, on the one hand, and with the operations used to form the bases of linear space, on the other hand. For the bases of the considered type these relations are as follows:

- logical operations - *truncated difference* operation

$$x^{(i)}=i\dot{-}(i\dot{-}x)=x\dot{-}(x\dot{-}i);$$

$$x^{(i)}-x^{(j)}=x^{(ij)}=(x\dot{-}j)\dot{-}(x\dot{-}i);$$

$$min(x_1,x_2)=x_1\dot{-}(x_1\dot{-}x_2)=x_2\dot{-}(x_2\dot{-}x_1);$$

$$max(x_1,x_2)=x_1+(x_2\dot{-}x_1)=x_2+(x_1\dot{-}x_2).$$

- logical operations - *comparison* operation

$$x^{(i)}=\sum_{j=0}^{i-1}(x>j);$$

$$x^{(i)}-x^{(j)}=x^{(ij)}=\textstyle\sum_{t=1}^{i-j}P_t(x>t);$$

$$min(x_1,x_2)=P\{[(x_1>0)+(x_2>0)]>1\}+$$
$$+P\{[(x_1>1)+(x_2>1)]>1\};$$

$$max(x_1,x_2)=P\{[(x_1>0)+(x_2>0)]>0\}+$$
$$+P\{[(x_1>1)+(x_2>1)]>0\}.$$

Using the above relations, we can easily obtain linear analogs of the above bases:

- conjunctive *k* = 2, *n* = 2 and *n* = 3

$$\&(\mathrm{B1},\mathrm{B1})^{(2)}=\begin{bmatrix}1\\x_1^{(1)}\\x_2^{(1)}\\x_1^{(1)}x_2^{(1)}\end{bmatrix}=\begin{bmatrix}1\\x_1\\x_2\\(x_1+x_2)\dot{-}1\end{bmatrix}=\begin{bmatrix}1&1&1&1\\0&1&0&1\\0&0&1&1\\0&0&0&1\end{bmatrix};$$

$$\left(\&(\mathrm{B1},\mathrm{B1})^{(2)}\right)^{-1}=\begin{bmatrix}1&-1&-1&1\\0&1&0&-1\\0&0&1&-1\\0&1&1&1\end{bmatrix};$$

$$\&(\mathrm{B1},\mathrm{B1},B1)^{(2)}=\begin{bmatrix}1\\x_1\\x_2\\x_3\\x_1x_2\\x_1x_3\\x_2x_3\\x_1x_2x_3\end{bmatrix}=\begin{bmatrix}1\\x_1\\x_2\\x_3\\(x_1+x_2)\dot{-}1\\(x_1+x_3)\dot{-}1\\(x_2+x_3)\dot{-}1\\(x_1+x_2+x_3)\dot{-}2\end{bmatrix}=$$

$$=\begin{bmatrix}1&1&1&1&1&1&1&1\\0&1&0&1&0&1&0&1\\0&0&1&1&0&0&1&1\\0&0&0&0&1&1&1&1\\0&0&0&1&0&0&0&1\\0&0&0&0&0&1&0&1\\0&0&0&0&0&0&1&1\\0&0&0&0&0&0&0&1\end{bmatrix};$$

$$\left(\&(\mathrm{B1},\mathrm{B1},B1)^{(2)}\right)^{-1} = \begin{bmatrix} 1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 \\ 0 & 1 & 0 & 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 & -1 & -1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & -1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix};$$

- disjunctive $k = 2$, $n = 2$ and 3

$$\vee\,(\mathrm{B1},\mathrm{B1})^2 = \begin{bmatrix} 1 \\ x_1^{(1)} \\ x_2^{(1)} \\ x_1^{(1)} \vee x_2^{(1)} \end{bmatrix} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1 + (x_2 \dot{-} x_1) \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix},$$

$$(\vee\,(\mathrm{B1},\mathrm{B1})^2)^{-1} = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 0 & -1 & 1 \\ 0 & -1 & 0 & 1 \\ 0 & 1 & 1 & -1 \end{bmatrix},$$

$$\vee\,(\mathrm{B1},\mathrm{B1},B1)^{(2)} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \\ x_1 \vee x_2 \\ x_1 \vee x_3 \\ x_2 \vee x_3 \\ x_1 \vee x_2 \vee x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \\ 1 \dot{-} [1 \dot{-} (x_1 + x_2)] \\ 1 \dot{-} [1 \dot{-} (x_1 + x_3)] \\ 1 \dot{-} [1 \dot{-} (x_2 + x_3)] \\ 1 \dot{-} [1 \dot{-} (x_1 + x_2 + x_3)] \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix},$$

$$\left(\vee\,(\mathrm{B1},\mathrm{B1},B1)^{(2)}\right)^{-1} =$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 & 1 & 1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 1 & 0 & 1 & -1 \\ 0 & -1 & 0 & 0 & 1 & 1 & 0 & -1 \\ 0 & 1 & 1 & 1 & -1 & -1 & -1 & 1 \end{bmatrix}.$$

Similarly, we obtain linear analogs of multivalued bases:

- conjunctive, $k = 3$, $n = 2$

$$min(B1,B1)^{(3)} = \begin{bmatrix} 1 \\ 1 \dot{-} (1 \dot{-} x_1) \\ x_1 \\ 1 \dot{-} (1 \dot{-} x_2) \\ [1 - (1 \dot{-} x_1)] - (1 \dot{-} x_2) \\ x_1 \dot{-} 2(1 \dot{-} x_2) \\ x_2 \\ 2[1 - (1 \dot{-} x_1)] - (2 \dot{-} x_2) \\ x_1 \dot{-} (x_1 \dot{-} x_2) \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 2 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 2 & 2 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 2 \end{bmatrix};$$

$$\left(min(B1,B1)^{(3)}\right)^{-1} =$$

$$= \begin{bmatrix} 1 & -1 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & -1 & 0 & -2 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & -2 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 2 & -1 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 1 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \end{bmatrix};$$

- disjunctive

$$max(B1,B1)^{(3)} = \begin{bmatrix} 1 \\ 1 \dot{-} (1 \dot{-} x_1) \\ x_1 \\ 1 \dot{-} (1 \dot{-} x_2) \\ 1 \dot{-} (x_1 + x_2) \\ x_2 + \{2[1 \dot{-} (1 \dot{-} x_1)] \dot{-} x_2\} \\ x_2 \\ x_1 + \{2[1 \dot{-} (1 \dot{-} x_2)] \dot{-} x_1\} \\ x_1 + (x_2 \dot{-} x_1) \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 0 & 2 & 2 & 1 & 2 & 2 & 2 & 2 & 2 \\ 0 & 1 & 2 & 1 & 1 & 2 & 2 & 2 & 2 \end{bmatrix};$$

$$\left(max(B1,B1)^{(3)}\right)^{-1} =$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -1 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & -2 & 2 & 0 & 1 & -2 \\ 0 & 0 & 0 & 2 & 0 & -1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & -1 & 1 & 0 & 1 & -1 \\ 0 & 2 & -1 & 0 & 0 & -1 & 0 & -1 & 2 \\ 0 & -1 & 1 & -1 & 1 & 0 & 0 & 0 & -1 \end{bmatrix}.$$

C. Logic synthesis of basic functions. The resulted conditional graphic designations of operations of the linear space can be used for graphic representation of basic Boolean logic operations in linear algebra and construct functional schemes of logic elements on their base. For this it is sufficient:

- to choose the basis from the listed above;

- to use the previously described technique of logic synthesis in linear algebra.

We choose a conjunctive version of the chosen basis as an object of logic synthesis in which we perform a logic synthesis of the two-valued and three-valued functions of circular shift

$$(x_1\&x_2)\oplus 1=[1\ 1\ 1\ 0],$$

and

$$min(x_1,x_2)\oplus 1=[1\ 1\ 1\ 1\ 2\ 2\ 1\ 2\ 0],$$

and

$$min(x_1,x_2)\ominus 1=[2\ 2\ 2\ 2\ 0\ 0\ 2\ 0\ 1].$$

In accordance with the above, the first action is to obtain the expansion vector of the value vector of the function in terms of the basis. Multiplying the vectors written above by the columns of the inverse matrix of the bases $\&(\mathrm{B1},\mathrm{B1})^{(2)}$ and $min(B1,B1)^{(3)}$ results in the following:

$$w[(x_1\&x_2)\oplus 1]=[0\ 1\ 1-2];$$

$$w[min(x_1,x_2)\oplus 1]\Rightarrow[1\ 0\ 0\ 0\ 3\ 0\ 0\ 0-2];$$

$$w[min(x_1,x_2)\ominus 1]\Rightarrow[2\ 0\ 0\ 0-3\ 0\ 0\ 01].$$

Weighing the basis vectors with respect to the obtained coefficients, we obtain expressions of the logic functions in the given basis

$$(x_1\&x_2)\oplus 1=1-(x_1+x_1-2x_1\&x_1);$$

$$min(x_1,x_2)\oplus 1=1+3min\left(x_1^{(1)},x_2^{(1)}\right)-2min\left(x_1^{(2)},x_2^{(2)}\right);$$

$$min(x_1,x_2)\ominus 1=2-3min\left(x_1^{(1)},x_2^{(1)}\right)+min\left(x_1^{(2)},x_2^{(2)}\right).$$

The functional schemes corresponding to these expressions are shown in Figure 7 *a, b, c.*

To obtain the equivalent expressions of the functions in linear analogs of the logical-arithmetic basis under consideration, one can proceed in two ways:

- replace the used basis vectors with their linear analogs in the expressions obtained;

- use the above relations between logic and linear operations and convert the expressions written above into the linear form.

In all three cases, we obtain the following results:

$$(x_1\&x_2)\oplus 1=1-\{x_1+x_1-2[(x_1+x_1)\dot{-}1]\};$$

$$min(x_1,x_2)\oplus 1=1+3\{[1\dot{-}(1\dot{-}x_1)]\dot{-}(1\dot{-}x_2)\}-$$
$$-2\{[(x_1\dot{-}1)+(x_2\dot{-}1)]\dot{-}1\};$$

$$min(x_1,x_2)\ominus 1=2-3\{[1\dot{-}(1\dot{-}x_1)]\dot{-}(1\dot{-}x_2)\}+$$
$$+\{[(x_1\dot{-}1)+(x_2\dot{-}1)]\dot{-}1\}.$$

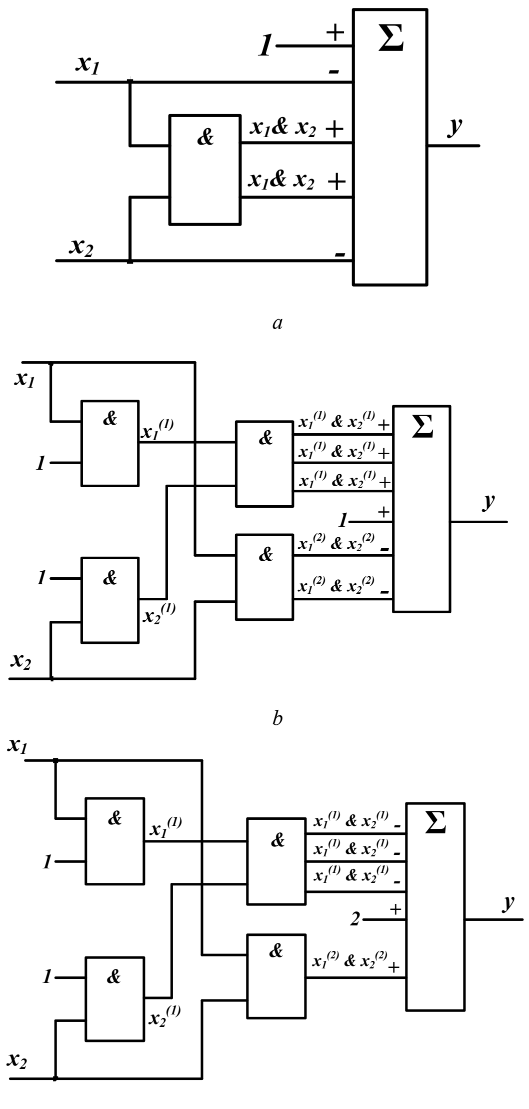
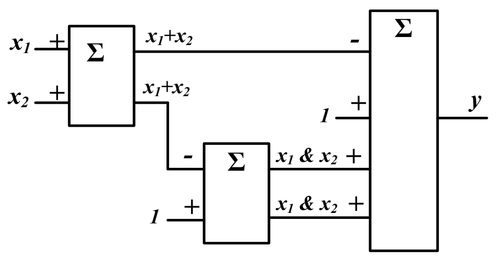


Figure 7: Functional schemes of the logical-arithmetic realization of the circular shift operation: a - two-valued, b - three-valued with a shift to the right, c - three-valued with a shift to the left.

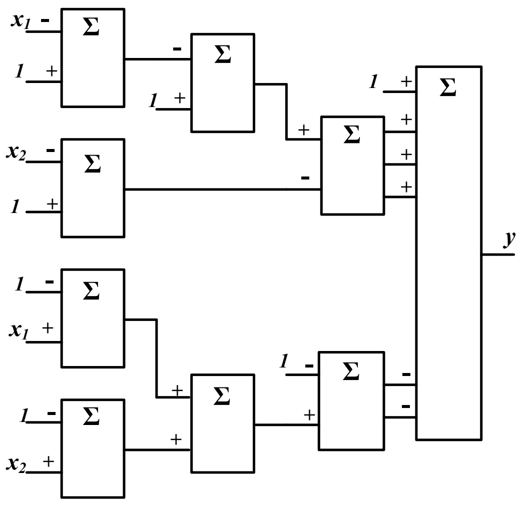
The functional schemes corresponding to these expressions are shown in Figure 8, *a, b, c.*

The difference between the linear representations obtained from the arithmetic-logical representation is the possibility of physical realization. The authors of the article have obtained more than 25 patents for the circuit implementation of two-valued and three-valued logic circuits.

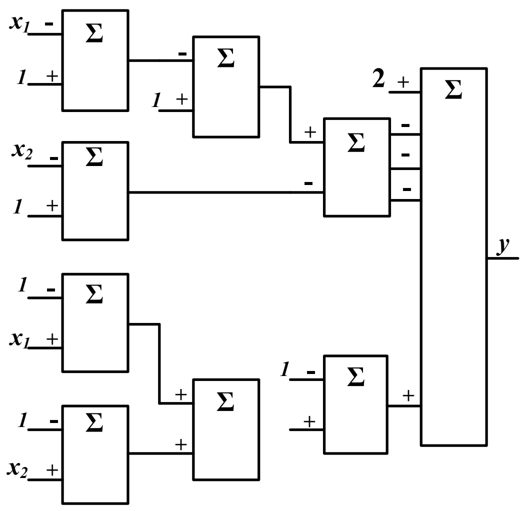
D. Expansion of the arbitrary function into the algebraic sum of monotonic functions of the same value. To obtain the representation of the functions under consideration by the algebraic sum of monotonic functions, we expand them into the algebraic sum of monotonic functions within each value. For the two-valued function, the sequence graph of nondecreasing components has the form shown in Figure 9.



a



b



c

Figure 8: Functional schemes of the linear realization of the circular shift operation based on the *truncated difference*: *a* - two-valued, *b* - three-valued with a shift to the right, *c* - three-valued with a shift to the left.

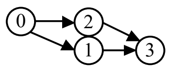


Figure 9: Sequence graph of nondecreasing components of the two-valued functions of two variables.

The structure of the expansion function of the equivalence $\bar{x}_1\bar{x}_2 \vee x_1x_2 = (x_1\&x_2) \oplus 1$ into the algebraic sum of monotonic functions is given in Figure 10.

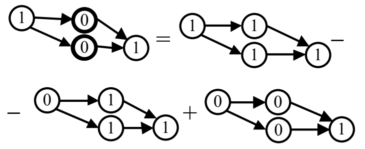


Figure 10: The structure of the expansion of the two-valued function $\bar{x}_1\bar{x}_2 \vee x_1x_2$ into the algebraic sum of monotonic functions.

Here and below, the double circle marks the terms the values of which violate the monotonicity of the function.

It follows from the structure of the expansion that the representation of the function under consideration by the algebraic sum of monotonic functions has the following form

$$f(\tilde{x}^{(2)}) = (x_1\&x_2) \oplus 1 = 1 - x_1 \vee x_2 + x_1\&x_2.$$

Carrying out the similar transformations for the three-valued functions, the sequence graph of nondecreasing components of which is shown in Figure 11.

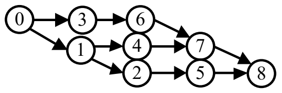


Figure 11: Sequence graph of nondecreasing components of the three-valued functions of two variables.

we obtain the following representations of the functions under consideration by the algebraic sum of monotonic functions:

- for $min(x_1, x_2)\oplus 1$.

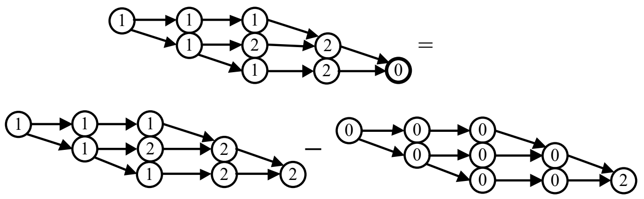


Figure 12: The structure of the expansion of the three-valued function $min(x_1, x_2)\oplus 1$ into the algebraic sum of monotonic functions.

$$min(x_1, x_2)\oplus 1 = 1 - min(x_1^{(1)}, x_2^{(1)}) -$$
$$-2min(x_1^{(2)}, x_2^{(2)}) + min(x_1^{(1)}, x_2^{(1)});$$

– for $min(x_1, x_2) \ominus 1$.

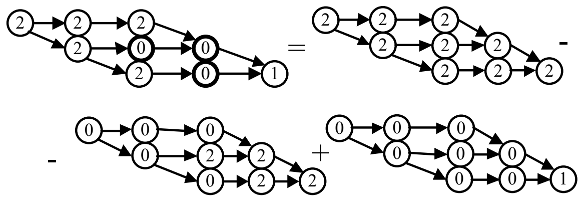


Figure 13: The structure of the expansion of the three-valued function $min(x_1, x_2) \ominus 1$ into the algebraic sum of monotonic functions.

$$min(x_1, x_2) \ominus 1 = 2 - 2min(x_1^{(1)}, x_2^{(1)}) +$$
$$+min(x_1^{(2)}, x_2^{(2)}).$$

To obtain the equivalent representations of the expansion functions in linear analogs of the basis under consideration, we can proceed in the same way as with the functions themselves.

Expansion of the arbitrary function into the algebraic sum of smaller value.

We consider this operation with the help of the above transformation of the three-valued function $min(x_1, x_2)\oplus 1$, by unitary and weighted coding of the values of variables and functions.

In these cases, the encoding of variables and functions looks like this:

- unitary coding

x	x_2	x_1
0	0	0
1	0	1
2	1	1

Hence, it follows that for the unitary coding

$x = x_1 + x_2$;

– weighted coding

x	x_2	x_1
0	0	0
1	0	1
2	1	0

It follows that at the weighted coding

$$x = x_1 + 2x_2.$$

Using the basic and its inverse matrix, we obtain the expressions for the two-valued expansion functions from the multivalued arguments:

- for the unitary coding

$$f_1 = 1 + min\left(x_1^{(1)}, x_2^{(1)}\right) - min\left(x_1^{(2)}, x_2^{(2)}\right);$$

$$f_2 = 2min\left(x_1^{(1)}, x_2^{(1)}\right) - min\left(x_1^{(2)}, x_2^{(2)}\right);$$

$$f = f_1 + f_2;$$

- for the weighted coding

$$f_1 = 1 - min\left(x_1^{(1)}, x_2^{(1)}\right);$$

$$f_2 = 2min\left(x_1^{(1)}, x_2^{(1)}\right) - min\left(x_1^{(2)}, x_2^{(2)}\right);$$

$$f = f_1 + 2f_2.$$

The representation of the three-valued function $min(x_1, x_2)\oplus 1$ by the two-valued ones for both versions of coding is given in Table 2.

Table 2 – The expansion of the three-valued function $min(x_1, x_2)\oplus 1$ into the sum of the two-valued ones

x_2	x_1	Unitary		Weighted	
		f_2	f_1	f_2	f_1
0	0	0	1	0	1
0	1	0	1	0	1
0	2	0	1	0	1
1	0	0	1	0	1
1	1	1	1	1	0
1	2	1	1	1	0
2	0	0	1	0	1
2	1	1	1	1	0
2	2	0	0	0	0

Replacing the three-valued variables with the two-valued ones in accordance with the rules given above, we obtain the two-valued expansion functions in the following form:

- for the unitary coding

$$f_1 = 1 - x_{11}x_{12}x_{21}x_{22};$$

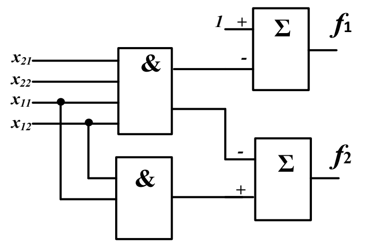
$$f_2 = x_{11}x_{21} - x_{11}x_{12}x_{21}x_{22};$$

- for the weighted coding

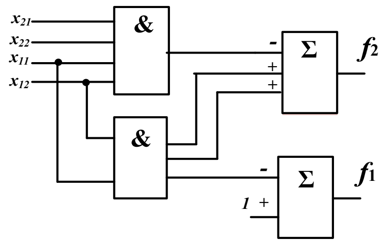
$$f_1 = 1 - x_{11}x_{21};$$

$$f_2 = 2x_{11}x_{21} - x_{11}x_{12}x_{21}x_{22}.$$

The functional schemes of the linear realization of the circular shift operation by the expansion into the sum of the two-valued functions are shown in Figure 14.



a



b

Figure 14: Functional schemes of linear realization of the circular shift operation by the expansion into the sum of the two-valued functions: *a* - on the basis of the unitary coding, *b* - on the basis of the weighted coding.

For the threshold representation and the subsequent threshold realization, it suffices to transform the equations obtained above into the threshold form. After completing this transformation, we get:

- for the unitary coding

$$f_1 = 1 > [(x_{11}+x_{12}+x_{21}+x_{22}) > 3];$$

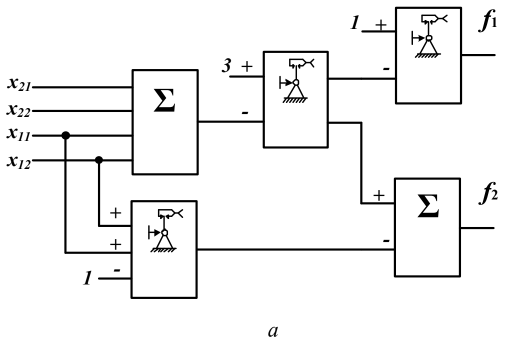
$$f_2 = [(x_{11}+x_{21}) > 1] - [(x_{11}+x_{12}+x_{21}+x_{22}) > 3];$$

- for the weighted coding

$$f_1 = 1 - [(x_{11}+x_{21}) > 1];$$

$$f_2 = 2[(x_{11}+x_{21}) > 1] - [(x_{11}+x_{12}+x_{21}+x_{22}) > 3].$$

The functional schemes shown in Figure 15 correspond to these equations.



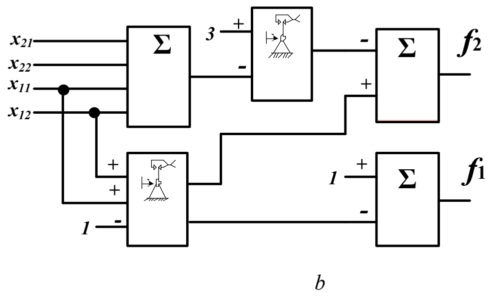


Figure 15: Functional schemes of linear realization of the circular shift operation by the expansion into the sum of the threshold two-valued functions: *a* - on the basis of the unitary coding, *b* - on the basis of the weighted coding.

The considered complex of transformations of logic functions in linear algebra proves useful in the design of digital structures for various applications.

5. Circuitry of the Linear Logic and Threshold Elements

A. Basic functional nodes of digital logic elements. The implementation of the mathematically predetermined set of linear operations over the current signals requires the corresponding set of hardware. The circuit implementation of digital signal transformation functions based on the mathematical tool of linear algebra can be reduced to performing a relatively simple set of operations over the current signals. These operations include:

- conversion of the standard logic signals into the binary current signals matched with a reference current quantum I_0;

- formation of the multivalued (non-binary) algebraic sums of current signals;

- performing the *comparing* operations of the received sums with the given levels of the reference currents.

These operations are typical for analog microcircuitry, therefore the main nodes of various functional elements can be constructed on the basis of the widely used analog structures. In addition, TTL circuitry and CMOS circuitry of these operations completely coincide.

The reference signal driver. It is designed to generate voltages that provide operational modes of elements of digital circuits synthesized in linear algebra. The schematic configuration of the driver is shown in Figure 16.

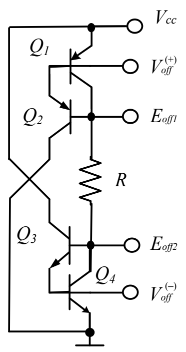


Figure 16: Reference signal driver.

The symmetric structure of the reference signal source is necessary for realizing the operations of converting "inflowing" and "flowing out" currents when creating the algebraic sums of variables in the mathematical tool of linear algebra. The only current-stabilizing two-terminal network, in particular case the resistor *R*, determines the levels of all reference signals. $V_{off}^{(+)}$ and $V_{off}^{(-)}$ are reference for setting the operating modes of current mirrors, and E_{off1} and E_{off2} - for setting the offset in the comparators of the currents. It can also be replaced by some semiconductor structure, and then the circuit becomes completely homogeneous and highly technological. It is also possible to make the two-terminal network *R* as an external element, which will allow changing the power consumption and the associated characteristics of the LSI during the debugging process.

The reference current source (RCS, Figure 17). The operating mode of the RCS is set by the reference voltages $V_{off}^{(+)}$ for the current sources and $V_{off}^{(-)}$ for the current sinks. The problem of constructing the RCS for digital circuits synthesized in linear algebra is completely similar to their problems in analog circuitry.

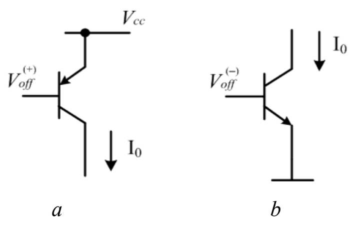


Figure 17: RCS of: a – the current current, b – the current sink.

The current follower (logic interpretation - direction converter). It is intended for the coordination of current directions at their algebraic summation. The schematic configurations of some versions of current followers in TTL circuitry are shown in Figure 18.

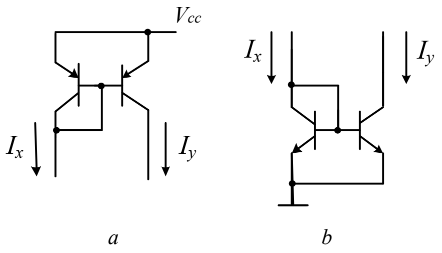


Figure 18: Current direction converter of: *a* – the sink current, *b* - the source current.

Circuitry of CMOS converters of the current direction is similar.

Algebraic current adder. It is an wired pack of the outputs of current mirrors with current directions determined by the mathematical representation of the realized logic function. To agree the operating modes of subsequent elements, it is provided with a buffer stage. The schematic configuration of the algebraic adder is shown in Figure 19.

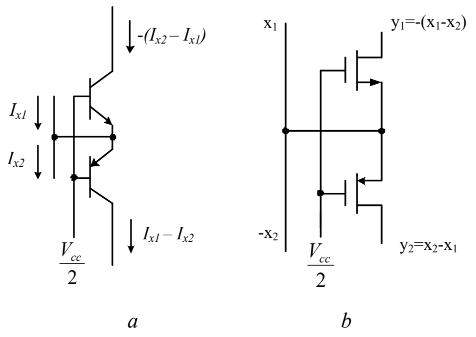


Figure 19: Algebraic current adder:

a - TTL circuit, *b* - CMOS circuit.

Current comparators. They are designed to determine the excess value of one of the input currents over the other. The schematic configurations of current comparators are given in Figures 20 and 21.

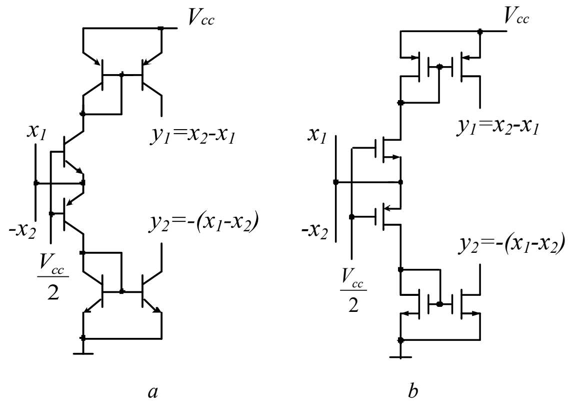


Figure 20: Comparator based on the *truncated difference*:

a - bipolar circuit, *b* - CMOS circuit.

In the circuit in Figure 21, the excess of one of the input currents (I_{x1}) over the other (I_{x2}) is determined by subtracting the second current from the first. One of the compared currents must be a source current and the second one – a current sink. Such a comparator is applicable for any logic value.

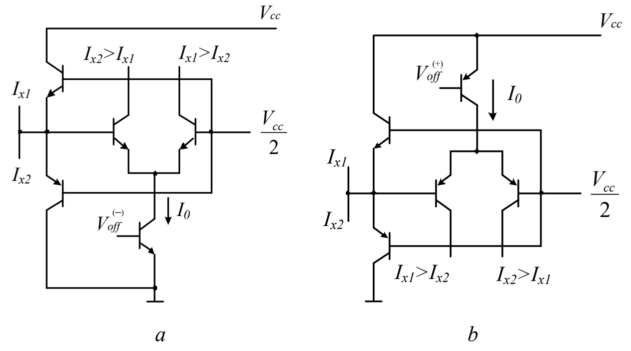


Figure 21: Current comparator based on the comparison: *a* - for current sinks, *b* - for source currents.

In the circuit of Figure 21, *a* at $I_{x1} > I_{x2}$ the left-hand transistor of the differential stage (DS) is closed and the current of the current mirror "leaves" through the right transistor of the DS. At $I_{x1} < I_{x2}$, the right transistor of the DS is closed and the current of the current mirror "goes" through the left transistor. The output current of the DS I_0 is the current sink.

In the circuit in Figure 21, *b* everything appears in a similar way, but with other current directions. Such a scheme is applicable for implementation of the two-valued digital structures.

In the previous section, the difference between the two-valued and multivalued implementations of the *comparison* operation is shown. In the latter case, it is possible to determine not only the fact, but the magnitude of the excess of one input signal over another one. In Figure 22 there is a circuit of the three-valued comparator performing such a modified *comparison* operation.

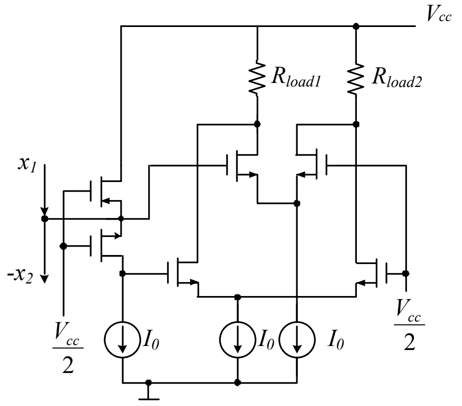


Figure 22: Modified multivalued comparator based on the comparison.

Such a scheme can be constructed for any value, increasing accordingly the number of parallelly operating DSs available to work with different values of currents.

Logic elements AND, OR, NOT. The representations of the operations of the two-valued basic functionally complete system in linear algebra using the *truncated difference* have the following form

$$x_1 \& x_2 = (x_1 + x_2) \dot{-} 1;$$

$$x_1 \vee x_2 = 1 \dot{-} [1 \dot{-} (x_1 + x_2)];$$

$$\bar{x} = 1 - x.$$

The schematic configuration of these elements are shown in Figures 23, 24 and 25, correspondingly.

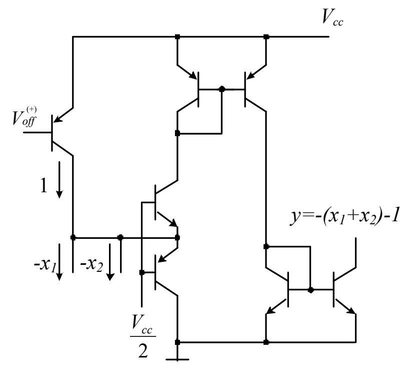


Figure 23: Schematic configuration of the element AND.

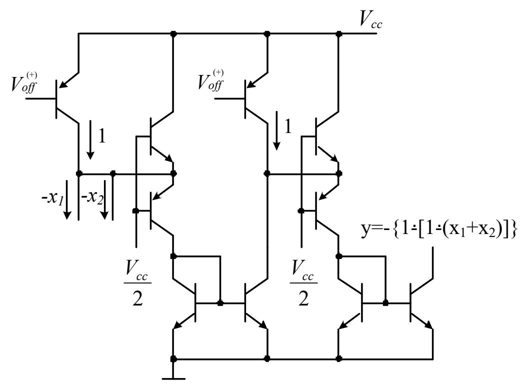


Figure 24: Schematic configuration of the element OR.

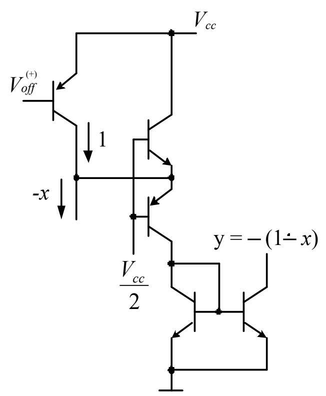


Figure 25: Schematic configuration of the element NOT.

Comparing the schemes shown in Figures 24 and 25, it is easy to see that removing the circuit of in Figure 25 from the scheme of Figure 24, we obtain the OR-NOT element.

Logic element "Inhibition". The representation of the two-valued operation in linear algebra using the *truncated difference* has the following form

$$x_1 \& \bar{x}_2 = x_1 \dot{-} x_2.$$

For the multivalued version, the two-valued inversion operation (in accordance with the accepted generalization ideology) should be replaced with the direct $min(x_1, x_2 \oplus 1)$, or inverse min $min(x_1, x_2 \ominus 1)$ cycle operation, or left unchanged; i.e. in the following form $min(x_1, 1 \dot{-} x_2)$. Other ideologies of generalization are also possible.

When using the *comparison* operation, the above expressions for operations of the functionally complete system take the following form

$$x_1 \& x_2 = (x_1 + x_2) > 1;$$

$$x_1 \vee x_2 = (x_1 + x_2) > 0;$$

$$\bar{x} = 1 > x.$$

The implementation of these functions on the basis of the *comparison* operation can be performed with the help of the universal logic element (ULE), the schematic configuration of which is shown in Figure 26.

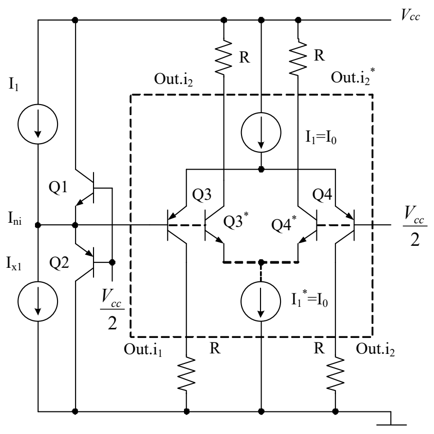


Figure 26: Basic scheme of the universal logic element.

At the inequality $I_1 < I_{x1}$, the source difference current is generated at the node In.i of the ULE. It will "go" to the emitter of the transistor Q1, increasing the voltage at the first input (In.1) of the voltage comparator up to the value $V_{OFF} + U_{be1}$, where $U_{be1} \approx 0.7V$ - voltage of the open emitter junction of the transistor Q1. In this case, the input transistors Q3 and Q4 of the voltage comparator (VC) switch to the inverse states - the collector current of the transistor Q3 becomes zero, and the transistor Q4 starts to transmit the current of the reference current source $I_2 = I_0$ to the second current output (Out.i$_2$) of the ULE.

Thus, depending on the difference in the numerical values of the currents I_1 and I_{x1}, the output currents of the ULE take one of two values: either it is the current of the reference current source $I_1 = I_0$ or "zero" (no current). Since the current I_1 (I_1^*) is equal to the current quantum I_0, then in one of the current outputs of the voltage comparator a standard current signal I_0 of one of the logic levels is generated, and in its second output - an inverse logic level signal. Depending on the numerical values of I_1 ($I_1 = 0{,}5I_0$, $I_1 = I_0$, $I_1 = 1{,}5I_0$) and the methods of forming the input current signals of the ULE (Figure 26), various logic functions can be performed, for example

$$-\bar{x} = 1 - x \text{ at } y_1 = \begin{cases} I_0 \; at \; I_1 > I_x \\ 0 \; at \; I_1 \le I_x \end{cases};$$

$$-x_1 \& x_2 = (x_1 + x_2) > 1 \text{ when } y_1 = \begin{cases} I_0 \; at \; 1{,}5I_1 > I_x \\ 0 \; at \; 1{,}5I_1 \le I_x \end{cases};$$

$$-x_1 \vee x_2 = (x_1 + x_2) > 0 \text{ if } y_1 = \begin{cases} I_0 \; at \; 0{,}5I_1 > I_x \\ 0 \; at \; 0{,}5I_1 \le I_x \end{cases}.$$

In the last two expressions $I_x \equiv x_1 + x_2$.

The element considered above can be used as a logic element, or as a threshold one; i.e. the generalization of the ordinary and threshold logics in the linear representation is very close. In the first case, unitary variables are fed to the inputs of the element. Besides, the number of inputs must correspond to the number of the variables. In the second case, the weighted sum of the variables should be fed to the "positive" input, and the constant equal to the calculated threshold value should be fed to the "negative" input. It is noteworthy that the described element can serve as an element of the homogeneous matrix, which can be used for matrix synthesis of the current digital structures. In addition, it is possible to construct universal current logic modules on its basis.

Similarly, the Ban operation can be expressed in terms of the *truncated difference* as

$$min(x_1, \{x_2 + 1 \dot{-} k[1 \dot{-} (x_2 \dot{-} 1)]\});$$

$$min(x_1, \{x_2 - 1 + k[1 \dot{-} x_2]\}),$$

and through the *comparison* - as

$$min(x_1, [x_2 + 1 \dot{-} k(x_2 > 2)]);$$

$$min(x_1, \{x_2 - 1 + k[1 > x_2]\}),$$

Cut formers. In fact, the cut former is an input signal limiter at the given level. As a former, the schemes that implement the operations $min(x_1, x_2)$ (upper cuts) and $max(x_1, x_2)$ (lower cuts) can be used.

Buffer output stage. The combination of the algebraic adder and the current follower (converter) can be applied as a buffer output stage. With the help of the latter, the given number of the output circuits can be arranged to provide the required output branching factor.

B. The synthesis of logical schemes. Let's consider it through the example of the two-valued and multivalued circular shift elements discussed above. They can be represented as a *single operation* in the form $min(x_1, x_2 \oplus 1)$ or $m(x_1, x_2 \ominus 1)$, or as a *compound operation*, i.e. as the sequential combination of operations working on one another $min(x_1, x_2)$ or $max(x_1, x_2)$ and operations $\oplus 1$ or $\ominus 1$, that is, in the form $min(x_1, x_2) \oplus 1$ or $min(x_1, x_2) \ominus 1$. We confine ourselves to the synthesis of the circuits using the operation $min(x_1, x_2)$.

In accordance with [17-32], the synthesis of the functional scheme corresponding to any logic function consists in multiplying the row-vector of the values of the function by the inverse basic matrix and obtaining the expansion vector of the function from the selected basis, and then recording the representation of the function as a weighted sum of the basis vectors. We perform the synthesis of the selected schemes using the basis presented below.

In the two-valued case, the element AND-NOT has an arithmetic-logical representation, described by the expression

$$\overline{x_1 \& x_2} = 1 - x_1 \& x_2,$$

which can be represented by the *truncated difference* in the following form

$$\overline{x_1 \& x_2} = 1 - [(x_1 + x_2) \dot{-} 1],$$

and by *comparison* - in the following form

$$\overline{x_1 \& x_2} = 1 > [(x_1 + x_2) > 1].$$

The schematic configurations of the elements can be constructed directly according to these expressions. Figure 27 shows an element scheme based on the *truncated difference*.

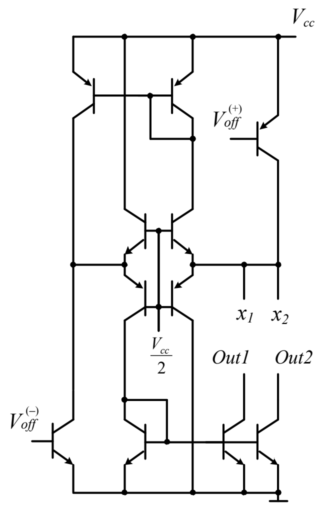


Figure 27: Schematic configuration of the element of AND-NOT based on the *truncated difference*.

We proceed similarly for the three-valued schemes in the first case (by the single operation). Using the basis

$$min(B1, B1)^{(3)} = \begin{bmatrix} 1 \\ 1 \dot{-} (1 \dot{-} x_1) \\ x_1 \\ 1 \dot{-} (1 \dot{-} x_2) \\ [1 - (1 \dot{-} x_1)] - (1 \dot{-} x_2) \\ x_1 \dot{-} 2(1 \dot{-} x_2) \\ x_2 \\ 2[1 - (1 \dot{-} x_1)] - (2 \dot{-} x_2) \\ x_1 \dot{-} (x_1 \dot{-} x_2) \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 2 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 2 & 2 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 2 \end{bmatrix}$$

with the inverse matrix

$$\left(min(B1, B1)^{(3)}\right)^{-1} =$$

$$= \begin{bmatrix} 1 & -1 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & -1 & 0 & -2 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & -2 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 2 & -1 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 1 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \end{bmatrix},$$

we obtain the following results for the *single operation*:

- resolution vectors

$$w(min(x_1, x_2) \oplus 1) = [1\ 0\ 0\ 0\ 3\ 0\ 0\ 0 - 2];$$

$$w(min(x_1, x_2) \ominus 1) = [2\ 0\ 0\ 0 - 3\ 0\ 0\ 0\ 1];$$

- linear expressions of the functions

$$min(x_1, x_2) \oplus 1 = 1 + 3[1 - (1 \dot{-} x_1) - (1 \dot{-} x_2)] -$$

$$-2[x_1 \dot{-} (x_1 \dot{-} x_2)] .$$

$$min(x_1, x_2) \ominus 1 = 2 - 3[1 - (1 \dot{-} x_1) - (1 \dot{-} x_2)] +$$

$$+[x_1 \dot{-} (x_1 \dot{-} x_2)].$$

The schematic configuration of the element of the right circular shift, synthesized directly from the above expression, is shown in Figure 28.

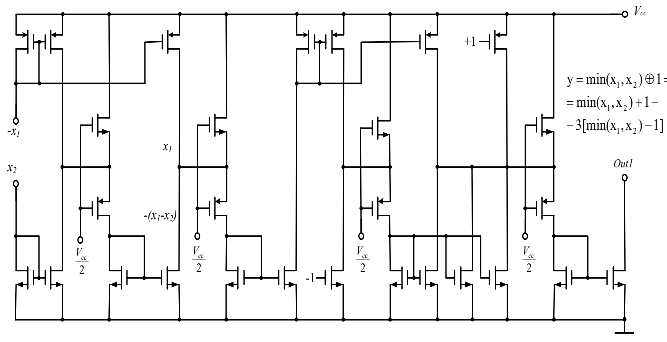


Figure 28: CMOS element of the right circular shift.

The schematic configuration of the element of the left circular shift, synthesized directly from the expression given above, is given in Figure 29.

Similarly, for the *compound operation*:

- the resolution vector $min(x_1, x_2)$

$$w\left(min(x_1, x_2)\right) = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1],$$

- linear expression of the function $min(x_1, x_2)$

$$min(x_1, x_2) = x_1 \dot{-} (x_1 \dot{-} x_2). \tag{5}$$

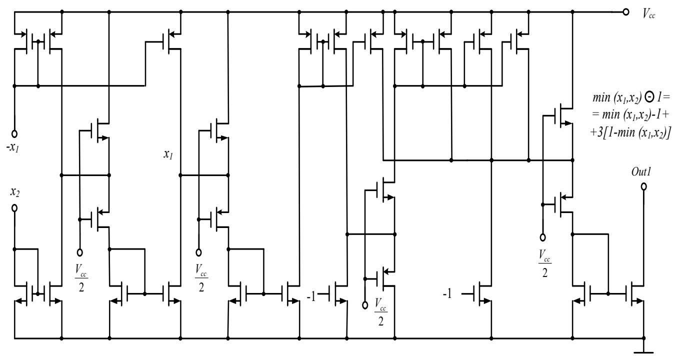


Figure 29: CMOS element of the left circular shift.

Similarly, for the *compound operation*:

- the resolution vector $min(x_1, x_2)$

$$w\left(min(x_1, x_2)\right) = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1],$$

- linear expression of the function $min(x_1, x_2)$

$$min(x_1, x_2) = x_1 \dot{-} (x_1 \dot{-} x_2). \tag{5}$$

- resolution vectors of the operations $x \oplus 1$ or $x \ominus 1$:

$$w(x \oplus 1) = [1\ 3 - 2];$$

$$w(x \ominus 1) = [2 - 3\ 1].$$

- linear expressions of the functions:

$$x \oplus 1 = 1 + [1 \dot{-} (1 \dot{-} x)] - 2x;$$

$$x \ominus 1 = 2 - 3[1 \dot{-} (1 \dot{-} x)] + x.$$

The schematic configuration of the element $min(x_1, x_2)$, synthesized directly from expression (5) discussed above, is shown in Figure 30.

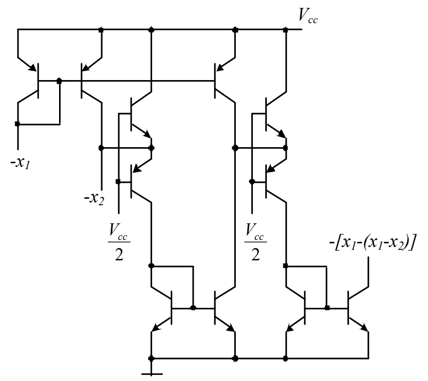


Figure 30: Schematic configuration of the element $min(x_1, x_2)$.

The schemes of the elements of the left and right cycles are shown in Figures 31 and 32.

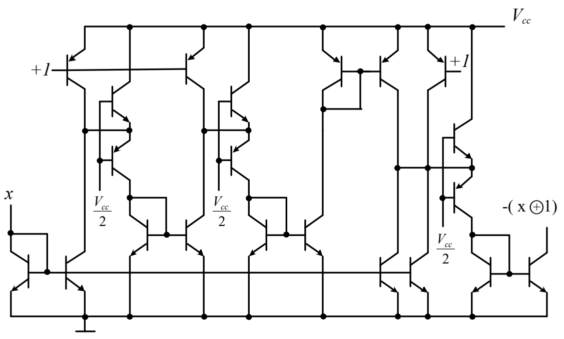


Figure 31: Element of the right cycle.

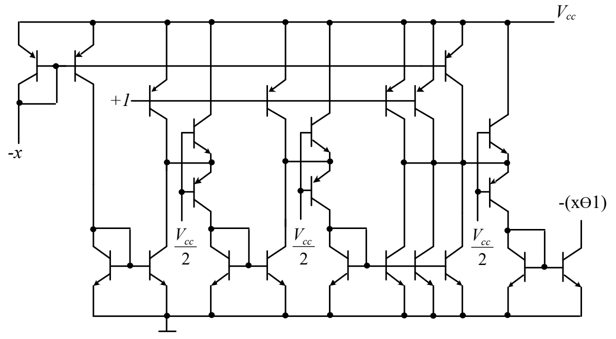


Figure 32: The element of the left cycle.

To obtain the final expression in the procedure for synthesizing a particular logic element, it is sufficient to substitute $min(x_1, x_2)$, for x in the last expression, and for the circuit implementation, - use the above implementation of the function $min(x_1, x_2)$, then connect the circuit $(x \oplus 1)$ or $(x \ominus 1)$ to its output.

C. To demonstrate the possibilities of the *sequential circuit synthesis*, we use the results of the logic element synthesis obtained above.

Boolean approach to the logic synthesis of triggers consists in supplying the memory element with a control circuit that provides a specified law for the operation of a specific type of the trigger. In the two-valued case, the memory element is a scheme of two 2AND-NOT elements covered by the positive feedback (Figure 33):

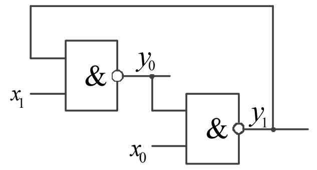


Figure 33: Two-valued trigger (memory element).

The linear synthesis of the two-valued triggers in linear algebra does not differ fundamentally from Boolean synthesis [17]. The schematic configuration of the trigger can be constructed in the same way as it is done in Boolean logic (for example – Figure 34):

When moving to higher values while remaining the general idea of synthesis, it is necessary to use the elements and operations that are a generalization of the two-valued operations and logic elements. The operations *min* (x_1,x_2) and *max* (x_1,x_2) are generalization of the operations & and ∨ is. As for the inversion operation, to generalize it for a multi-valued case it is convenient to represent it in the form of $\bar{x} = x \oplus 1 = x \ominus 1$ and generalize it by a circular shift of Post (left or right). The general functional configuration of the memory element based on three-valued elements of the direct circular shift is shown in Figure 35.

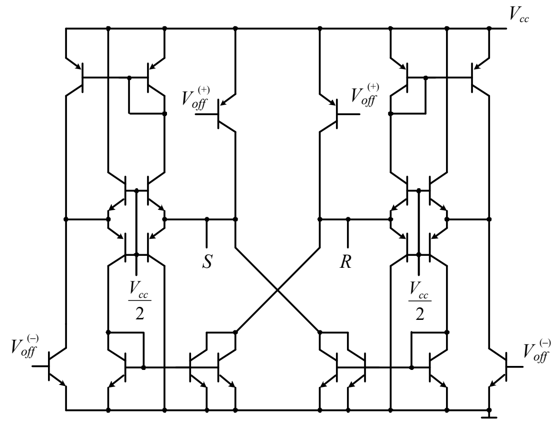


Figure 34: Two-valued RS-flip-flop in the studied basis based on the *truncated difference.*

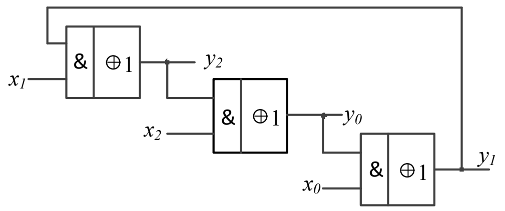


Figure 35: Three-valued trigger on the elements of the direct circular shift.

The general functional configuration of the memory element based on the three-valued elements of the inverse circular shift is given in Figure 36.

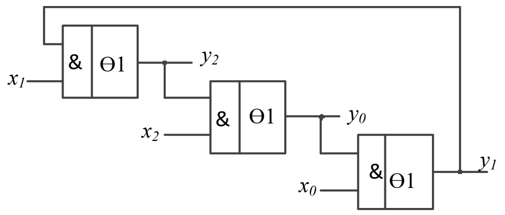


Figure 36: Three-valued trigger on the elements of the inverse circular shift.

The functional schemes of the memory elements of higher significance look similar.

Thus, to construct any multivalued memory element, it is necessary to synthesize the elements of direct and inverse circular shifts. To create a trigger of the given type (D-, RS-, JK-, etc.), it is required to equip the memory element with the corresponding control circuit.

The control circuits of the memory elements are constructed on the base of the verbal description of the operation of a specific type of the trigger. The result of the analysis of this description is the detection of logic control functions for each input of the memory element.

For example, the RS-flip-flop operates according to the following algorithm:

- the values of the input signals $S = R = k - 1$ correspond to the storage mode;

- the signal S increases the trigger state index relatively to the current state towards the state k-1, and the signal R reduces the state index towards the state “0”;

- the value of the state index change is equal to the value of the input signal: the signal equal to 1 can increase (or decrease) the current state index by 1, the signal equal to 2 - by 2, etc .; besides, the signal S increases the trigger state index to the state k-1, and the signal R reduces the state index to “0”; the state change on the cycle is impossible (we leave this for the universal triggers!);

- to change the trigger state index relatively to the current state i, the signal S can take values from 1 to k-1-i (larger values are equivalent to the value of k-1-i). Similarly, the signal R can take the values from 1 to i -1 (larger values are equivalent to the value i -1);

- all combinations are inhibited combinations of values of the input signals, except 0 – k-1, 1 – k-1, k-2 – k-1, k-1 – k-1, k-1 – 0, k-1 – 1, …, k-1– k-2.

Now it is possible to synthesize the logic function of the control circuit of the RS-flip-flop. Its output signal is a multi-valued signal x that sets the next state of the memory element, and the input signals are multivalued signals S and R, as well as the current trigger state index.

The truth table of the control functions of the three-valued RS-flip-flop, compiled on the basis of this description, has the following form:

S	R	Q_{t+1}	x_0	x_1	x_2
0	2	Q_2	0	1	1
1	2	Q_1	1	0	1
2	2	Q_t	1	1	1
2	1	Q_1	1	0	1
2	0	Q_0	1	1	0

From the last table it follows that the logic functions for controlling the state of the memory element are described as follows:

$$x_0 = P(S > 0) = 1 \dot{-} (1 \dot{-} S);$$

$$x_1 = 1 > [1 > (1 > S)] > [1 > (1 > R)] =$$

$$= 1 \dot{-} [1 \dot{-} |1 \dot{-} S|] \dot{-} [1 \dot{-} |1 \dot{-} R|];$$

$$x_2 = P(R > 0) = 1 \dot{-} (1 \dot{-} R).$$

The combination of the memory element circuit and the control circuit results in the implementation of the trigger of the given type.

Similarly, arguing, we can obtain the logical control functions of the three-valued D-flip-flop:

$$x_0 = (1 \dot{-} C)\&(1 \dot{-} D) = 1 \dot{-} [(1 \dot{-} D) \dot{-} C] =$$

$$= 1 \dot{-} [(1 \dot{-} C) \dot{-} D];$$

$$x_1 = (1 \dot{-} C)\&[D \dot{-} 2(D \dot{-} 1)] = 1 \dot{-} \{D \dot{-} [2(D \dot{-} 1) \dot{-} C]\} =$$

$$1 \dot{-} [1 \dot{-} (|D \dot{-} 1| + C)];$$

$$x_2 = (1 \dot{-} C)\&(D \dot{-} 1) = 1 \dot{-} [(D \dot{-} 1) \dot{-} C] =$$

$$= D \dot{-} [(1 \dot{-} C) \dot{-} 1].$$

In the same way, we can obtain the description of the control system of any other multivalued trigger, the number of types of which is certainly greater than the two-valued one.

Combining the control circuit and the memory element in series, it is possible to obtain the schematic configuration of the trigger of any type and any value.

6. Conclusion

1. The mathematical tool of linear algebra can be newly applied in problems of logic synthesis and circuit implementation of the current digital structures.

2. Linear algebra:

- enables to create not only a two-valued, but also a really properly functioning multivalued element base for digital signal processing devices;

- conduces the design of the digital element base with the improved technological, technical and operational characteristics (in comparison with the potential logic based on Boolean algebra);

- can serve as a basis for creating LSI on the basis of matrix fields of homogeneous elements (as in modern Altera design systems, etc.);

- improves the reliability of current digital LSIs synthesized on its basis, under extreme operating conditions (temperature, radiation, in-phase interferences, etc.).

3. Two-valued and three-valued triggers based on cyclic shift elements are considered. It is shown that in the transition to greater significance, while preserving the general idea of synthesis, it is necessary to use operations that are generalizations of two-valued operations and the corresponding logical elements.

4. The presents a basic set of current logic elements for the devices of automation, which allows solving the problems of transformation of the current signals in a different and more efficient way.

5. In the schemes of the developed class is provided a differential representation of the output signal that minimizes the effect of temperature and radiation on their basic parameters.

Conflict of Interest

The authors declare that there is no conflict of interests regarding of publication of this paper.

Acknowledgment

The research is carried out at the expense of the Grant of the Russian Science Foundation (project № 16-19-00122).

References

[1] N. N. Prokopenko, N. I. Chernov, V. Ya. Yugai, N. V. Butyrlagin, "The Element Base of the Multivalued Threshold Logic for the Automation and Control Digital Devices," on International Siberian Conference on Control and Communications, SIBCON-2017, Astana, Kazakhstan, 29-30 June, 2017.

[2] A. S. Karpenko, "Multi-valued logics (monograph)," in series "Logics and computer", Moscow, issue 4, 1997, 223 p. (in Russian).

[3] L. A. Zalmanson, "Fourier transform, Walsh, Haar and their application in management, communications and other fields," Moscow, Nauka, Fizmatlit publ., 1989, 496 p. (in Russian).

[4] V. S. Vykhovanets, V. D. Malugin, "Spectral methods in logical management," Proceedings of the 2nd international scientific-technical conference "Modern methods of digital processing of signals in spelthorne, monitoring, diagnosis and control (OS 1998)", Minsk, 1998, pp. 56-59. (in Russian).

[5] M. Thornton, "Mixed representation of the multi-Boolean function spectra and critical charts," Automatics and telemechanics, 2004, vol.6, pp. 188-200. (in Russian).

[6] G. I. Ivchenko, V. A. Mironov, "Some questions of spectral analysis of random Boolean functions with constraints," Discrete mathematics, 2013, vol. 1, pp. 90-110. (in Russian).

[7] V.N. Kondratyev, A.A. Shalyto, "Implementation of systems of Boolean functions using arithmetical polynomials," Automatics and telemechanics, 1993, vol. 2, pp. 114-122. (in Russian).

[8] V. D. Malyugin, "The parallel logic computation by the arithmetic of polynomials," Moscow, Nauka, Fizmatlit publ., 1997, 192 p. (in Russian).

[9] A. V. Sokolov, O. N. Zhdanov, A. A. Ayvazyan, "Methods for the synthesis of the algebraic normal forms of functions of multivalued logic," System analysis and applied Informatics, No. 1, 2016, p. 69-76. (in Russian).

[10] M. Dertouzos, "Threshold logic," Moscow: Mir publ., 1967. (in Russian).

[11] V. G. Nikonov, "Threshold representations of Boolean functions," Review in applied and industrial mathematics, Series discrete mathematics, 1994, vol. 1, No. 3, pp. 402-457. (in Russian).

[12] E. A. Butakov, "Methods of synthesis of relay devices," Moscow: Energy publ., 1970, 328 p. (in Russian).

[13] E. N. Vavilov, etc. The Synthesis of threshold circuits for the elements – Moscow: Soviet Radio publ., 1970, 368 p. (in Russian).

[14] S. Muroga, "Threshold logic and its applications," New York: Wiley, 1971.

[15] W. H. Piers, "Failure Tolerant computer dezign," New York and London: Academic Press, 1965.

[16] V. G. Nikonov, N. Nikonov, "Features of threshold concepts k-valued functions," Tr. on discr. mod., 2008, volume 11, issue 1, pp. 60-85. (in Russian).

[17] J. Hastad, "On the size of weights for threshold gates." SIAM J. Discr. Math. 1994.

[18] N. I. Chernov, "Foundations of Logic Synthesis of Real Numbers Field Digital Structures", Taganrog: TRTU, 2000, p.146 (in Russian).

[19] N. I. Chernov, "Boolean Linear Space as an Algebraic Structure for Logic Synthesis of Digital Units," The News of TRTU, 2003, No.1, pp. 215-220. (in Russian).

[20] N. I. Chernov, "Structural Synthesis of Digital Units within Boolean Linear Spaces," The News of TRTU, 2003, No.2, pp. 73-76. (in Russian).

[21] N. I. Chernov, "Efficiency of Application of the Body of Linear Spaces within Logic Synthesis," Conferences on Artificial Intelligence Systems (IEEE AIS' 05)' and Intelligent CAD (CAD-2005, vol.1, pp. 420–424. (in Russian).

[22] N. I. Chernov, "Logic Design of Digital Structures on Controlled Current Generator," The News of TREU, 2005, No.11, pp. 77- 83. (in Russian).

[23] N. I. Chernov, "The effectiveness of the apparatus of linear spaces in the logic synthesis of digital structures," Proceedings of the international scientific and technical conferences "Intelligent systems (IEEE AIS'05)" and "Intelligent CAD (CAD-2005",vol.1, pp. 420 –424. (in Russian).

[24] N. I. Chernov and V. Ya. Yugai, "Nonclassical Synthesis of Digital Structures by Tools of Analogous Circuits Engineering," Problems of Today's Analogous Circuits Engineering: The Collected Articles of IX International Scientific-Practical Seminar edited by N.N. Prokopenko, Shakhty, Rostov-on-Don Region: FSBEU HPE "SRSUES" Publishers, 2012, pp. 138 – 143. (in Russian).

[25] N. N. Prokopenko, N. I. Chernov, V. Ya. Yugai, "Base Concept of Linear Synthesis multi-Valued Digital Structures within Linear Spaces," Proceedings of The IS&IT13 Congress, The Scientific Edition in four volumes, Moscow: PhisMathLit publ., 2013, v. 1, pp. 284−289. (in Russian).

[26] N. N. Prokopenko, N. V. Butyrlagin, N. I. Chernov, V. Ya. Yugay, "Synthesis of binary triggers in the apparatus of linear algebra," Izvestiya SFedU. Technical Sciences. No. 2. 2015, pp. 115-125. (in Russian).

[27] N. N. Prokopenko, N. I. Chernov, V. Ya. Yugai, N. V. Butyrlagin, "Linear synthesis of k-valued digital components of the base current logical signals: the principle of generalization", Problems of development of perspective micro and nanoelectronic systems - 2016. Proceedings / under the General editorship of academician RAS A.L. Stempkovskogo. Moscow: IPPM RAS, 2016. (in Russian).

[28] N. I. Chernov, V. Y. Yugai, N. N. Prokopenko, N. V. Butyrlagin, "Basic concept of linear synthesis of multivalued digital structures in linear spaces," Proceedings of IEEE East-West Design and Test Symposium, EWDTS 2014, art. no. 7027045. DOI: 10.1109/EWDTS.2014.7027045

[29] N. N. Prokopenko, N. V. Butyrlagin, N. I. Chernov, V. Ya. Yugai, "The linear concept of logic synthesis of digital IP-modules of control and communication systems," 2015 International Siberian Conference on Control and Communications, SIBCON 2015 - Proceedings, art. no. 7147182. DOI: 10.1109/SIBCON.2015.7147182.

[30] N. N. Prokopenko, N. V. Butyrlagin, N. I. Chernov, V. Ya. Yugai, "Basic linear elements of k-Valued digital structures," 2016 International Conference on Signals and Electronic Systems, ICSES 2016 - Proceedings, pp. 7-12. DOI: 10.1109/ICSES.2016.7847763.

[31] N. N. Prokopenko, N. I. Chernov, V. Yugai, N. V. Butyrlagin, "The element base of the multivalued threshold logic for the automation and control digital devices," Proceedings of 2017 IEEE International Siberian Conference on Control and Communications, SIBCON 2017, art. no. 7998508. DOI: 10.1109/SIBCON.2017.7998508.

[32] N. N. Prokopenko, N. I. Chernov, V. Yugai, N. V. Butyrlagin, "The multifunctional current logic element for digital computing devices, operating on the principles of linear (not boolean) algebra," Proceedings of 2016 IEEE East-West Design and Test Symposium, EWDTS 2016, art. no. 7807723. DOI: 10.1109/EWDTS.2016.7807723.

[33] V. I. Nechaev, "Numeric system," Moscow: Education publ.,1975, 199 p. (in Russian).

[34] V. A. Gorbatov, "Foundations of discrete mathematics," Moscow: Higher school publ., 1986, 311 p. (in Russian).