Algorithms Synthesis of the Adaptive Logical Network on Basis of Universal Logical Elements

Volodymyr Opanasenko [1], Sergei Kryvyi [2]

1) V.M. Glushkov Institute of Cybernetics, The National Academy of Sciences of Ukraine,
Glushkov prospect 40, Kiev, Ukraine opanasenko@incyb.kiev.ua
2) Taras Shevchenko Kiev National University, Glushkov prospect 4D, Kiev, Ukraine, sl.krivoi@gmail.com

Abstract: *An approach to the synthesis of adaptive structures represented by multi-level logic, Boolean network described as an acyclic graph, whose vertices are universal logical elements, is proposed. As a result of the synthesis of such structures are determined the types of logic functions of the vertices of a given learning sample of binary vectors, which allows using of this structure for the problem of classification of input vectors. Unlike known methods for the synthesis of multilevel logic, two approaches to the synthesis of such schemes proposed in this paper. The first way based on the description of a Boolean network represented by polynomials, the coefficients of the polynomial in this case are given by Hadamard matrix. The second way based on the description of Boolean network by Zhegalkin's polynomials considered in the article.*

***Keywords*:** Adaptation, Boolean functions, universal logical element, polynomial.

1. INTRODUCTION

A wide range of classification tasks require adaptation (reconfiguration) structure to a given functioning algorithm [1–3]. The appearance of crystals FPGA types [4], what represent the functional field of universal logical elements (LE), what make it possible decide an issue of the hardware implementation of algorithms by configuring the structure of crystal to perform the required algorithm [4]. This article is a continuation of a works [2–6], which made in the field of synthesis of reconfigurable structures for various applications.

From the standpoint of topology [7] the adaptive logic network (ALN) is a matrix of LE, what are grouped into functional units (FU) and function blocks (FB), the location of what is fixed, while the change in their functioning occurs depending on the class of the tasks and their assignment.

LE will be called universal combinational automaton:

$$L = \langle n, F \rangle ,$$

where: n – the number of binary inputs or the dimensionality of the input variables LE;

$F = \{ f_\rho \}$, $\rho = [1 \div 2^{2^n}]$ – set of Boolean functions realized by LE.

The versatility of the LE allows adapting it for the implementation of an arbitrary Boolean function. In the case of $(n = 2)$ LE implements one of 16 logical functions representing the full (base) set of functions of two variables

$f_1 = a + b;$ $f_2 = a + \bar{b};$ $f_3 = \bar{a} + b;$ $f_4 = \bar{a} + \bar{b};$ $f_5 = a \& b;$
$f_6 = a \& \bar{b};$ $f_7 = \bar{a} \& b;$ $f_8 = \bar{a} \& \bar{b};$ $f_9 = a \oplus b;$ $f_{10} = a \oplus \bar{b};$
$f_{11} = a;$ $f_{12} = b;$ $f_{13} = \bar{a};$ $f_{14} = \bar{b};$ $f_{15} = 0;$ $f_{16} = 1.$

ALN structure can be described by the following system

$$A = \langle n, h, F, S, L, m, D, X, Y \rangle ,$$

where: n – the dimensionality of the input binary vectors (ALN dimension by input); h – dimension by output ($h = \overline{1 \div n}$), ALN dimension by output; $F = \{ F_{ij} \}$ – a set of logical functions of the system; S – the structure of communications between the LEs; $L = \{ L_{ij} \}$ – set of LEs (i – serial number of the element LE; j – number level processing); m – the number of levels of processing; $D = \{ d \}$ – the set n–volumetrically binary vectors (training set); X – a full set of input binary vectors; $Y = \{ Y_{ij} \}$ – function of the entire network, $Y_{ij} = f_{ij}(Y_{v,(j-1)}, Y_{w,(j-1)})$ – the value of the function f_{ij}, what is implemented by an element L_{ij}, $Y \in \{0,1\}$, whose structure is shown in Fig. 1 (v, w – index value of the inputs LE).

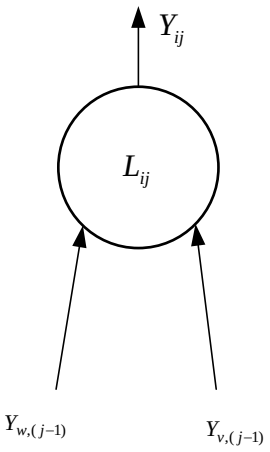

Fig. 1 – Structure of the universal LE

In what follows we consider the type of ALN "triangular matrix" (TM) – this is FB, which is composed from l functional units FU, for which ($h = 1$),

differentiated by the topological criterion. FU is a combination automaton without memory of having l –bit input, u –bit output and m – the number of rows of the matrix. Within the bounds of the one level the type of function set separately for each LE.

Basic set of logical functions is defined by the dimensionality of the universal LE and it is a full set of logic functions for a given number of input variables (binary vectors).

2. STATEMENT OF THE PROBLEM

The problem of constructing a logical network is reduced to the problem of determining the necessary set of functions as a superposition of elementary basic functions and the formation of nested function blocks.

The function block type TM [4] is designed to split the complete set-bit vectors into two subsets of vectors defined by the learning sample $D \subseteq X_1$ ($X_1 \bigcup X_2 = X;\ X_1 \bigcap X_2 = \varnothing$). TM implements a mapping $\Im : X \to Y$, such that $\Im : X_1 \to 1$, $\Im : X_2 \to 0$.

$$Y(g) = \begin{cases} 1, (\forall g \in D); \\ 0, (\forall g \notin D). \end{cases} \qquad (1)$$

Structurally TM is a m–level hierarchical matrix. Level means FU, each LE is configured for the implementation of an arbitrary Boolean function and realizes a mapping l–dimensional binary vectors ($l = \overline{2 \div n}$) into u–dimensional ($l \geq u$) vectors. The task of adjustment (adaptation) TM is formulated as follows. Suppose we have a full set of n–dimensional binary vectors $X = \{x_p\}$, where $p = 1 \div 2^n$ and given a set of n–dimensional binary vectors $D \subset X$, which is a learning sample for classification algorithm. For an arbitrary input set of n–dimensional binary vectors $G = \{g\}$, $(G \subset X)$ you must implement the following function.

In general, the task of adapting the structure of TM for the implementation of the function (1) reduce to the problem of constructing the universal logic element on the basis of arbitrary length LE fixed–width and is to determine the structure of communications S and types of logic functions f_{ij} for these LE, what in the aggregate implements the mapping $\Psi : X \to Y$.

In accordance with (1) will be considered TM with the respective structure of communications (Fig. 2).

TM has the following characteristics ($n = 3$): $m = (n-1)$; $j = 1 \div (n-1), i = 1 \div (n-j)$; $v = i, w = (i+1)$; $Card\{L_{ij}\} = (n^2 - n)/2$.

3. THE FIRST METHOD OF SOLVING THE DIRECT SYNTHESIS PROBLEM

To determine the set of logic functions $F = \{f_{ij}\}$ can be used approach [8] using polynomials for structural description of TM. Mathematical model (1) can be described by a Boolean network, which represent the multilevel combinational scheme, and the vertices of the network correspond to logic elements.

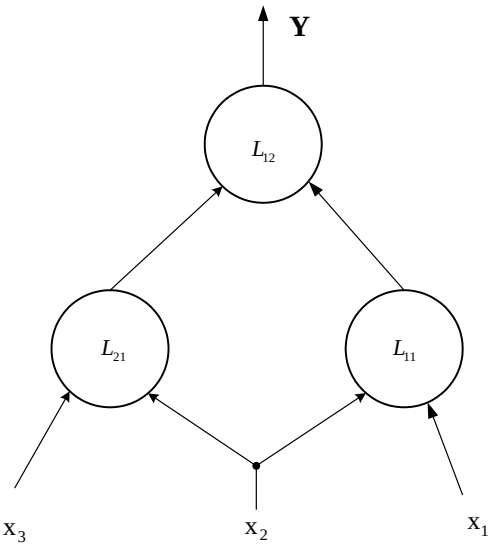

Fig. 2 – Structure of TM with honeycomb structure of intercommunications.

In the general case, the problem of synthesizing the structure TM reduced to determination the types of logic functions f_{ij} for all LE of network. For determination of the set of logic functions $F = \{f_{ij}\}$ will use polynomials to describe Boolean networks [6].

When encoding values of a Boolean function and its arguments to pass to another encoding, we will use the values of (1) and (-1). Thus, the set of variables $X = \{x_1, x_2, ..., x_n\}$ for a Boolean function f of the n variables will be submitted to the set $E = \{e_1, e_2, ..., e_n\}$ (where $e_i = (-1)^{x_i}$), and the set of values $Y = \{y_1, y_2, ..., y_{2^n-1}\}$ (where $y_j = \{0,1\}$) by set $V = \{v_0, v_1, ..., v_{2^n-1}\}$ where $v_j = (-1)^{y_j}$.

For any Boolean function f of the n variables taking values from the set $\{1,-1\}$, there is an equivalent polynomial $P_{f(n)}$ with coefficients from the set of real numbers $f(X) = P_{f(n)}(X)$ [6]. The coefficients of the polynomial for function f can be written by means Hadamard matrix:

$$A_{2^n} = \frac{1}{2^n} H_n V_n,$$

where: – $A_{2^n} = \{a_0, a_1, ..., a_{2^n-1}\}$ the set of coefficients of the polynomial; H_n – Hadamard matrix of dimension 2^n; V_n – the set of values of a Boolean function.

Hadamard matrix of n–th order H_n is a square matrix of dimension n, containing two types of elements $\{1,-1\}$. Hadamard matrix can be constructed for any value n:

$$H_0 = \|1\|;\ H_1 = \begin{Vmatrix} +1+1 \\ +1-1 \end{Vmatrix};\ H_n = \begin{Vmatrix} +H_{n-1} + H_{n-1} \\ +H_{n-1} - H_{n-1} \end{Vmatrix}.$$

A function of one variable is represented by a polynomial:

$$P_{f(1)} = a_0 + a_1 e_1 .$$

Functions of two and three variables are represented by polynomials:

$$\begin{aligned} P_{f(2)} &= a_0 + a_1 e_1 + a_2 e_2 + a_3 e_1 e_2 ; \\ P_{f(3)} &= a_0 + a_1 e_1 + a_2 e_2 + a_3 e_1 e_2 + a_4 e_3 + \\ &+ a_5 e_1 e_3 + a_6 e_2 e_3 + a_7 e_1 e_2 e_3 \end{aligned} \quad (2)$$

Accordingly, the polynomial of n variables:

$$P_{f(n)} = P_{f(n-1)} + a_{n+1} e_n P_{f(n-1)} .$$

4. EXAMPLE

As an example, consider the direct problem of the synthesis of parametric module TM with parameters: the dimension of binary vectors $(n=3)$, learning sample $D=\{(1,1,1);(1,-1,1);(-1,1,-1);(-1,-1,1)\}$. Thus, we have a three-input TM with elements $L_{1,1}, L_{1,2}, L_{2,1}$. It is necessary to define a set of logical functions $\{f_{1,1},\ f_{2,1},\ f_{1,2}\}$ by defining sets of coefficients of the polynomial (2). On the basis of the initial data (learning sample), we have the following values for v_j:

$$\begin{pmatrix} v_0 = -1; v_1 = 1; v_2 = -1; v_3 = 1; \\ v_4 = 1; v_5 = -1; v_6 = -1; v_7 = 1; \end{pmatrix}.$$

To determine the coefficients of the polynomial, we must solve a system of eight equations with twelve unknowns. In this case, we obtain four solut

ions. In accordance with the results obtained from the truth table logic functions ($f_1 \div f_{16}$) are determined by the functions.

Results of got options (four) for adjustment of the structure TM (types of logic functions for logic elements) shown in Fig. 2, given in Table 2.

Table. Types of logic functions for logic elements

$f_{1,1}$	$f_{2,1}$	$f_{1,2}$
$e_1 \oplus e_2$	$\overline{e_2} \,\&\, \overline{e_3}$	$y_{1,1} \oplus y_{1,2}$
$\overline{e_1} \oplus e_2$	$e_2 + e_3$	$y_{1,1} \oplus y_{1,2}$
$\overline{e_1}$	$\overline{e_2} \,\&\, e_3$	$y_{1,1} \oplus y_{1,2}$
e_1	$e_2 + \overline{e_3}$	$y_{1,1} \oplus y_{1,2}$

In the general case (for n variables), when using this method, it is necessary to solve a system of 2^n linear equations with $2(n^2-n)$ unknowns.

Note that the method for solving systems of equations for determining the polynomial (2) holds one of the modified algorithms described in [9, 10].

5. THE SECOND METHOD OF SOLVING THE DIRECT PROBLEM OF SYNTHESIS

Determine the polynomial $P_{f(3)}$ can be a different way by solving a system of linear Diophantine equations (SLDE) in the field of residues modulo 2 – F_2. In general, for solution of SLDE in the field F_p, where p – prime number, method was developed and on the his basis – algorithm. Details of the method and algorithm described in detail in [9, 10], therefore here are the basic properties of the algorithm, which is named TSS.

Theorem 1. TSS – algorithm for solving SLDE builds the general solution of this system in time proportional to the value of $m \times n$, where m – the number of equations, and n – the number of unknowns.

Theorem 2. The general solution of linear inhomogeneous Diophantine equations (SLNDE) has the form:

$$x = x^1 \oplus \sum_{i=1}^{k} x_i ,$$

where: x^1 – particular solutions inhomogeneous SLDU and x_i – a basic solution of the system of linear homogeneous Diophantine equation (SLHDE), which correspond to a given inhomogeneous SLDE.

Finding the polynomial $P_{f(3)}$ is that for a given learning sample for a polynomial built SLNDE, whose solutions give different versions of the synthesis of the structure nodes.

However, this method can be improved and extended to the more general case of decomposition given by way of connections.

We now describe the method.

Imagine the expression (2) for $n=3$ as a polynomial Zhegalkin for cellular structure links

$$\begin{aligned} P_{f(3)} &= \underbrace{a_0 + a_1 e_1 + a_2 e_2 + a_3 e_1 e_2 +}_{L_{11}} \\ &\underbrace{+a_4 + a_5 e_2 + a_6 e_3 + a_6 e_2 e_3}_{L_{21}} = L_{12} \end{aligned} \quad (3)$$

Learning sample for L_{12} has the same form as in the previous method:

$$D = \{(0,0,0),(0,1,0),(1,0,1),(1,1,0)\},$$

where: $x = (x_3, x_2, x_1) \in D$.

By means learning sample we build a system of linear inhomogeneous Diophantine equations (SLIDE), substituting elements from D to $P_{f(3)}$, in which $P_{f(3)}$ should be set to 1:

$$S_1 = \begin{cases} a_0 \oplus 0a_1 \oplus 0a_2 \oplus 0a_3 \oplus 0a_4 \oplus 0a_5 \oplus 0a_6 \oplus 0a_7 = 1, \\ a_0 \oplus 0a_1 \oplus a_2 \oplus 0a_3 \oplus 0a_4 \oplus a_5 \oplus 0a_6 \oplus 0a_7 = 1, \\ a_0 \oplus a_1 \oplus 0a_2 \oplus 0a_3 \oplus a_4 \oplus 0a_5 \oplus a_6 \oplus 0a_7 = 1, \\ a_0 \oplus 0a_1 \oplus a_2 \oplus 0a_3 \oplus a_4 \oplus a_5 \oplus a_6 \oplus a_7 = 1. \end{cases} \quad (4)$$

In matrix form, this SLIDE is written as

$Aa = b$, where $a = (a_0, a_1, ..., a_7)$, $b = (1,1,1,1)$ – column-vectors of dimension 4, and the matrix A is of the form:

$$\begin{pmatrix} 10000000 \\ 10100000 \\ 11001100 \\ 10101010 \end{pmatrix}.$$

A particular solution of S_1 is a vector $x^1 = (1,0,0,0,0,0,0,0)$, and the basic solution of the system of homogeneous linear Diophantine equations (SHLDE), which corresponds S_1, there vectors:

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

The general solution of system (4) takes the form

$$x = x^1 \oplus d_1 x_1 \oplus d_2 x_2 \oplus d_3 x_3 \oplus d_4 x_4,$$

where $d_i \in \{0,1\}, i = 1,2,3,4.$

By varying the values of the coefficients d_i, we find 16 possible implementations of computational structures (see Fig. 2). For example, for

1) $d_1 = d_2 = d_3 = 0, d_4 = 1$ we get the solution $(1,0,0,0,0,0,0,1)$, which corresponds to:

$$1 \oplus e_1 e_2 e_3 = \overline{e_1} \vee \overline{e_2} \vee \overline{e_3}.$$

2) $d_1 = d_2 = 0, d_3 = d_4 = 1$ we get the solution $(1,1,0,0,1,0,1,1)$, which corresponds to:

$$1 \oplus e_1 \oplus e_3 \oplus e_2 e_3 \oplus e_1 e_2 e_3 = \overline{e_1} \oplus e_3 (1 \oplus e_2 \oplus e_1 e_2) =$$
$$= \overline{e_1} \oplus e_3 (1 \oplus \overline{e_1} e_2) = \overline{e_1} \oplus e_3 (e_1 \vee \overline{e_2});$$

3) $d_1 = 0, d_2 = 1, d_3 = d_4 = 0$ we get the solution $(1,1,0,0,1,0,1,0)$, which corresponds to:

$$1 \oplus e_1 \oplus e_3 \oplus e_2 e_3 = \overline{e_1} \oplus \overline{e_2} e_3;$$

4) $d_1 = 1, d_2 = d_3 = d_4 = 0$ we get the solution $(1,1,0,0,0,1,0,0)$, which corresponds to:

$$1 \oplus e_1 \oplus e_1 e_3 = \overline{e_1} \oplus e_1 e_3.$$

Not all of the resulting functions will give on the elements not included in sample D, the value of 0. For example, in the case of functions from item 3) on the elements $(0,1,1)$ we get a value of 1.

If we require that all samples are not included into D, value of $P_{f(3)} = 0$, then we get next SLIDE:

$$S_2 = \begin{cases} S_1 \\ a_0 \oplus a_1 \oplus a_2 \oplus a_3 \oplus a_4 \oplus a_5 \oplus a_6 \oplus a_7 = 0, \\ a_0 \oplus a_1 \oplus 0a_2 \oplus 0a_3 \oplus 0a_4 \oplus 0a_5 \oplus 0a_6 \oplus 0a_7 = 0, \\ a_0 \oplus 0a_1 \oplus 0a_2 \oplus 0a_3 \oplus a_4 \oplus 0a_5 \oplus 0a_6 \oplus 0a_7 = 0, \\ a_0 \oplus a_1 \oplus a_2 \oplus a_3 \oplus 0a_4 \oplus 0a_5 \oplus 0a_6 \oplus 0a_7 = 0. \end{cases}$$

We will receive single solution:

$$x^1 = (1,1,0,0,1,0,1,0).$$

This solution give function: $\overline{e_1} \oplus \overline{e_2} e_3$.

The solutions from the table1 are an equivalence to $\overline{e_1} \oplus \overline{e_2} e_3$. Really:

$$a)\ e_1 \oplus e_2 \oplus \overline{e_2}\,\overline{e_3} = e_1 \oplus e_2 \oplus \overline{e_3} \oplus e_2 \overline{e_3} =$$
$$= e_1 \oplus e_2 \oplus 1 \oplus e_3 \oplus e_2 \oplus e_2 e_3 =$$
$$= 1 \oplus e_1 \oplus e_3 \oplus e_2 e_3 = \overline{e_1} \oplus \overline{e_2} e_3;$$

$$b)\ \overline{e_1} \oplus e_2 \oplus e_2 \vee e_3 = \overline{e_1} \oplus e_2 \oplus (e_2 \oplus e_3 \oplus e_2 e_3) =$$
$$= \overline{e_1} \oplus \overline{e_2} e_3;$$

$$c)\ e_1 \oplus (e_2 \vee e_3) = e_1 \oplus e_2 \oplus \overline{e_3} \oplus e_2 \oplus e_2 \overline{e_3} =$$
$$= e_1 \oplus e_2 \oplus 1 \oplus e_3 \oplus e_2 \oplus e_2 e_3 =$$
$$= 1 \oplus e_1 \oplus e_3 \oplus e_2 e_3 = \overline{e_1} \oplus \overline{e_2} e_3.$$

These combinations provide the same function at the nodes of structure, as can be seen directly (see Table).

Note that if the learning sample is changed, then the matrix of the system S_2 does not change and change only the free terms of the system. The solution process is the same, but the synthesized functions change. For example, if $D = \{(0,0,0),(0,1,0),(1,0,1),(0,1,1)\}$, then S_2 takes the form:

$Aa = (1,1,1,0,0,0,1,0)^t$, where A – the matrix S_2.

The single solution has the form:

$$x^1 = (1,1,0,1,1,0,0,1),$$

i.e., the general solution will has the form $x = x^1 \oplus d_1 x_1 \oplus d_2 x_2$.

Then, for example, when

1) $d_1 = d_2 = 1 \Rightarrow x = (0,1,1,1,1,1,1,0) \Rightarrow$

$$\Rightarrow (e_1 \oplus e_2 \oplus e_1 e_2) \oplus (1 + e_2 \oplus e_3) = \overline{e_1} e_2 \oplus \overline{e_2} =$$
$$= (e_1 \vee e_2) \oplus (\overline{e_2} \oplus e_3);$$

2) $d_1 = 0, d_2 = 1 \Rightarrow x = (1,1,1,1,0,1,1,0) \Rightarrow$

$$\Rightarrow (1 \oplus e_1 \oplus e_2 \oplus e_1 e_2) \oplus (e_2 \oplus e_3) = (1 \oplus (e_1 \cup e_2)) \oplus (e_2 \oplus e_3) =$$
$$= \overline{e_1}\,\overline{e_2} \oplus (e_2 \oplus e_3).$$

3) $d_1 = 1, d_2 = 0 \Rightarrow x = (0,1,0,1,1,0,1,0) \Rightarrow$

$$\Rightarrow (e_1 \oplus e_1 e_2) \oplus (1 + e_3) = \overline{e_1}\,\overline{e_2} \oplus \overline{e_3}.$$

4) $x = x^1 = (1,1,0,1,0,0,1,0) \Rightarrow (1 + e_1 + e_1 e_2) \oplus e_3 =$

$$= (1 + e_1 \overline{e_2}) \oplus e_3 = (\overline{e_1} \vee e_2) \oplus e_3.$$

If we change D, in this case will change only a column vector b. For example, if:

1) $D = \{(0,0,0),(0,1,0)\}$, in this case single solution:

$x^1 = (1,1,0,0,1,1,0,0)$

$f = \overline{e_1} \oplus e_3 \oplus e_1 e_3 = \overline{e_1} \oplus \overline{e_1} e_3 = \overline{e_1 e_3}$;

2) if $D = \{(0,0,0),(0,1,0),(0,0,1)\}$:

$x^1 = (1,0,1,1,0,0,1)$

$f = 1 \oplus e_3 + e_1 e_3 + e_1 e_2 e_3 = 1 \oplus e_3 \oplus e_3 e_1 \overline{e_2} =$;

$= 1 \oplus e_3 (1 \oplus e_1 \overline{e_2}) = 1 \oplus e_3 (\overline{e_1} \vee e_2) = \overline{e_3} \vee e_1 \overline{e_2}$

3) if $D = \{(0,0,0),(0,0,1)\}$:

$x^1 = (1,0,1,0,0,0,1)$

$f = 1 \oplus e_2 \oplus e_3 \oplus e_1 e_2 e_3 = x^1 = (1,0,1,1,0,0,1)$

$f = 1 \oplus e_3 + e_1 e_3 + e_1 e_2 e_3 = \overline{e_2} \oplus e_3 (1 \oplus e_1 e_2) =$;

$= \overline{e_2} \oplus e_3 (\overline{e_1} \vee \overline{e_2})$

etc.

6. CONCLUSION

An approach to the synthesis of adaptive structures represented by multi-level logic, Boolean network described as an acyclic graph, whose vertices are universal logical elements, is proposed. The synthesis of such structures is to determine the types of logic functions of the vertices of a given learning sample of binary vectors, which allows using of this structure for the problem of classification of input vectors. Unlike known methods for the synthesis of multilevel logic, in this paper are proposed two approaches to the synthesis of such schemes. The first is based on the description of a Boolean network polynomials, the coefficients of the polynomial in this case are given by Hadamard matrix.

Algorithms for their solution have got the different estimates of the complexity. If we have got for the first method the squared estimate of complexity, depending from n variables, then the second - polynomial the complexity evaluation.

6. REFERENCES

[1] Palagin A.V., Opanasenko V.N. Reconfigurable computing technology, *Journal Cybernetics and Systems Analysis*. 43 (5) (2007). p. 675–686

[2] Y.P. Kondratenko, L.P. Klymenko, I.V. Sidenko Comparative analysis of evaluation algorithms for decision-making in transport logistics, Advance Trends in Soft Computing. M. Jamshidi, V.Kreinovich, J.Kazprzyk (Eds.), *Proceedings of WCSC. Series: Studies in Fuzziness and Soft Computing*, Volume 312, (2014). p. 203–217.

[3] Palagin A., Opanasenko V., Kryvyi S. The structure of FPGA-based cyclic-code converters, *Journal Optical Memory & Neural Networks (Information Optics)* 22 (4) (2013). p. 207–216.

[4] Opanasenko V.N., Kryvyi S.L. Partitioning the full range of boolean functions based on the threshold and threshold relation. *Journal Cybernetics and Systems Analysis* 48 (3) (2012). p. 459–468.

[5] Palagin A.V., Opanasenko V.N., Kryvyi S.L. *FPGA-based reconfigurable structures: Synthesis of problem-oriented structures (in Russian).* Verlag: LAP Lambert Academic Publishing, 2014. p. 54.

[6] Kryvyi S.L. *Linear Diophantine constrains and their application (in Ukrainian)*. Bukrek. Chernivtsi–Kyiv, 2015. p. 224.

[7] Palagin A.V., Opanasenko V.N.: Design and application of the PLD-based reconfigurable devices. In: *Design of Digital Systems and Devices*, vol. 79. pp. 59–91. Springer, Verlag, Berlin, Heidelberg (2011).

[8] Bruck J., Blaum M. Neural networks, error–correcting codes, and polynomials over the binary n–cube: *Journal IEEE Transactions on information theory* 35 (5) (1989). p. 976–987.

[9] Kryvyi S.L. Algorithms for solving systems of linear Diophantine equations in integer domains. *Journal Cybernetics and Systems Analysis* 42 (2). (2006). – pp. 163–175.

[10] Kryvyi S.L. Algorithms for solving systems of linear Diophantine equations in residue fields. *Journal Cybernetics and Systems Analysis.* 43 (2). (2007). - pp. 171–178.