UDC 004.272

V. OPANASENKO[1], S. KRYVYI[2]

[1] *V. M. Glushkov Institute of Cybernetics of the National Academy of Sciences, Ukraine*
[2] *Taras Shevchenko Kiev National University, Ukraine*

METHOD SYNTHESIS OF THE CONFIGURABLE LOGICAL BLOCKS ON BASIS OF UNIVERSAL LOGICAL ELEMENTS

An approach to the synthesis of adaptive structures represented by multi-level logic, Boolean network described as an acyclic graph with universal logical elements, is proposed. As a result of the synthesis of such structures are determined the type of logic function of the Boolean network 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, method of the synthesis of such schemes proposed in this paper. It based on the description of a Boolean network by Zhegalkin's polynomials, starting from subnet (n=3).

Key words: *adaptation, Boolean functions, binary vector, FPGA, polynomial, universal logical element.*

Introduction

A wide range of classification tasks requires adaptation (reconfiguration) structure to a given functioning algorithm [1-3]. The appearance of crystals FPGA types [4], which represent the functional field of universal logical elements (LE), makes it possible decide an issue of the hardware implementation of algorithms by configuring the structure of crystal to perform the required algorithm [4].

From the standpoint of topology [5] the adaptive logic network (ALN) is a matrix of the LE, which is grouped into functional units (FU) and function blocks (FB), the location of which is fixed, while the change in their functioning occurs depending on the class of the tasks and their assignment.

Let us call LE a universal combinational automaton:

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

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

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

Thus, as a universal LE will use a multiplexer (Fig. 1).

An address inputs (z_0, z_1) of multiplexer will be as inputs for binary variables and inputs of data ($x_0 \div x_3$) - as control inputs, specify the type of logical functions, for the realization of which the multiplexer is configured.

Table 1 shows example of a truth table of the multiplexer functioning as a universal logic element.

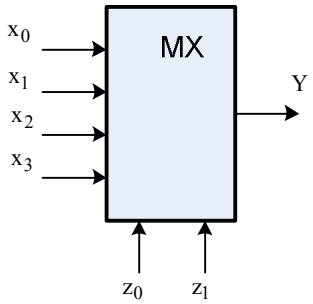

Fig. 1. Multiplexer structure

Table 1

The truth table of multiplexer

Inputs			Output	Logic function
x_i	z_0	z_1	Y	
$x_0 = 0$	0	0	0	AND
$x_1 = 0$	0	1	0	
$x_2 = 0$	1	0	0	
$x_3 = 1$	1	1	1	
$x_0 = 0$	0	0	0	OR
$x_1 = 1$	0	1	1	
$x_2 = 1$	1	0	1	
$x_3 = 1$	1	1	1	

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 (a, b):

$f_1 = a + b; f_2 = a + \bar{b}; f_3 = \bar{a} + b; f_4 = \bar{a} + \bar{b}; f_5 = a \,\&\, b;$

$f_6 = a \,\&\, \bar{b}; f_7 = \bar{a} \,\&\, b; f_8 = \bar{a} \,\&\, \bar{b}; f_9 = a \oplus b; f_{10} = a \oplus \bar{b};$

$f_{11} = a; f_{12} = b; f_{13} = \bar{a}; f_{14} = \bar{b}; f_{15} = 0; f_{16} = 1.$

The truth table for the full set of logical functions of two variables is shown in Table 2. From the values of the column of the table can will be determined the type of logic function.

Table 2

The truth table of logical functions

Input data		Types of logic functions							
a	b	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
0	0	0	1	1	1	0	0	0	1
0	1	1	0	1	1	0	0	1	0
1	0	1	1	1	1	0	1	0	0
1	1	1	1	0	0	1	0	0	0
a	b	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
0	0	0	1	0	0	1	1	0	1
0	1	1	0	0	1	1	0	0	1
1	0	1	0	1	0	0	1	0	1
1	1	0	1	1	1	0	0	0	1

Statement of the Problem

Let the system for arbitrary logic function f_z ($\forall z = 1 \div 2^{2^n}$) of the full feature set is specified:

$$f_z = (x_1^1, x_2^1, ..., x_n^1) = y_1;$$
$$f_z = (x_1^2, x_2^2, ..., x_n^2) = y_2;$$
$$...$$
$$f_z = (x_1^k, x_2^k, ..., x_n^k) = y_k;$$
$$...$$
$$f_z = (x_1^{2^n}, x_2^{2^n}, ..., x_n^{2^n}) = y_{2^n}.$$

That is, for any logical function $f_z(x)$ ($\forall z = 1 \div 2^{2^n}$) the variables we obtain a system that determines the values of these functions y_k ($\forall k = 1 \div 2^n$).This system is a training set D (for which $y_k = 1$).

It is necessary, on the basis of the set values of components x_1^k of the vectors of the training set D to define the type of logic function $f_z(x)$ and, on the basis of the type of functions from Table 1 to determine the value of the column corresponding to this function. Column values are fed to the data inputs of the multiplexer (who now are control and define the type of logic function for LE (multiplexer).

The method of solving the direct synthesis problem

In [1-3] considered the adaptation issues of logical networks with one output based on the triangular matrices that realize a partition of binary input vectors $e = (e_n, ..., e_2, e_1) \in E$ as a binary vector into two subsets based on a predetermined learning sample $D \subseteq E$.

It is assumed that the output value of the logical network is determined as follows:

$$Y = \begin{cases} 1, (\forall e \in D); \\ 0, (\forall e \in \bar{D}), \end{cases} \quad (1)$$

where $\bar{D} = E \setminus D$ – the complement of the set D in the set E.

In this paper is considered the method of synthesis of the logical network with one output by means of the input learning sample.

Structurally TM is a multilevel hierarchical matrix. The task of adjustment (adaptation) TM formulated as follows. Suppose we have a full set of n–dimensional binary vectors $e = (e_n, ..., e_2, e_1) \in E$ and given a set of n–dimensional binary vectors $D \subseteq E$, which is a learning sample for classification algorithm.

In general, the task of adapting the structure of TM (1) reduces to problem of synthesis of logic function for to implementation the mapping $\Re : D \to Y$.

In accordance with (1) will be considered TM [4] with the respective structure (for $n = 3$) of links (Fig. 2).

For determination of the logic function TM will use polynomials to describe Boolean networks [6]. A function of one variable is represented by a polynomial:

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

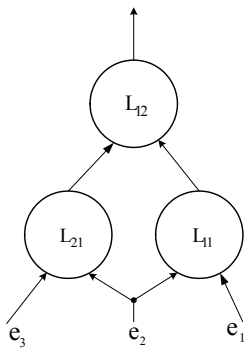

Fig. 2. Structure of TM with honeycomb structure of intercommunications

Accordingly, the polynomial of n variables:

$$P_{f(n)} = P_{f(n-1)} \oplus a_{n+1} e_n P_{f(n-1)}. \quad (2)$$

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 [7,8], therefore here are the basic properties of the algorithm, which is named TSS.

Theorem 1. The general solution of linear inhomogeneous Diophantine equations (SLIDE) has the form:

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

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

Theorem 2. 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.

Finding the polynomial $P_{f(3)}$ is that for a given learning sample for a polynomial built SLIDE, 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 will describe the method.

Example. The configurable logic block (as LE) in the structure of modern FPGA has 6 binary inputs. Therefore, we consider the problem of synthesis of logical function for this unit. Given the learning sample ($n = 6$) for L_{12} and has next form:

$D_6 = \{(000000),(000010),(000101),(000110),$ (001111), (010001), (011000), (011010), (011100), (011101), (011110), (010111), (010011), (010100), (011011), (011001), (100001), (101000), (101010), (101100), (101101),(101110), (100111), (100011), (100100), (101011), (101001), (110000), (110010), (110101), (110110), (111111)}.

By means learning sample we build a system of linear inhomogeneous Diophantine equations (SLIDE), substituting elements from D to $P_{f(6)}$ (corresponding system is not possibility to present because of great his dimension). As a result of solution of SLIDE we get the function for LE:

$$f = \overline{e_1 e_4} \oplus \overline{e_2} e_3 \oplus e_3 e_4 (e_1 \vee \overline{e_2}) \oplus e_5 \oplus e_6 .$$

For compensation we consider imagine the expression (2) for $n = 3$ as a Zhegalkin's polynomial for honeycomb structure links:

$$P_{f(3)} = a_0 \oplus a_1 e_1 \oplus a_2 e_2 \oplus a_3 e_1 e_2 \oplus a_4 e_3 \oplus \oplus a_5 e_1 e_3 \oplus a_6 e_2 e_3 \oplus a_7 e_1 e_2 e_3 . \quad (3)$$

Function synthesized by means a wave method, based on the initial learning sample:

$$D_3 = \{(000), (010), (101), (110), (111)\}.$$

By means learning sample we build a system of linear inhomogeneous Diophantine equations (SLIDE), substituting elements from D_3 to $P_{f(3)}$.

$$S = \begin{cases} 1a_0 \oplus 0a_1 \oplus 0a_2 \oplus 0a_3 \oplus 0a_4 \oplus 0a_5 \oplus 0a_6 \oplus 0a_7 = 1, \\ 1a_0 \oplus 0a_1 \oplus 1a_2 \oplus 0a_3 \oplus 0a_4 \oplus 0a_5 \oplus 0a_6 \oplus 0a_7 = 1, \\ 1a_0 \oplus 1a_1 \oplus 0a_2 \oplus 0a_3 \oplus 1a_4 \oplus 1a_5 \oplus 0a_6 \oplus 0a_7 = 1, \\ 1a_0 \oplus 0a_1 \oplus 1a_2 \oplus 0a_3 \oplus 1a_4 \oplus 0a_5 \oplus 1a_6 \oplus 0a_7 = 1, \\ 1a_0 \oplus 1a_1 \oplus 1a_2 \oplus 1a_3 \oplus 1a_4 \oplus 1a_5 \oplus 1`a_6 \oplus 1a_7 = 1, \\ \quad . \ . \ . \qquad\qquad . \ . \ . = 0, \\ \quad . \ . \ . \qquad\qquad . \ . \ . = 0, \\ \quad . \ . \ . \qquad\qquad . \ . \ . = 0. \end{cases}$$

Solving this system by the TSS-method [6, 7], we find the single solution $x^1 = (1,1,0,0,1,0,1,1)$, which corresponds to the Zhegalkin's polynomial:

$$P_{f(3)} = \overline{e_1} \oplus e_3 (e_1 \vee \overline{e_2}) .$$

If the learning sample D is changed, for example, $D = \{(0,0,0),(0,1,0),(0,0,1)\}$, the matrix of the system does not change, but it is change only the free terms:

$$S = \begin{cases} 1a_0 \oplus 0a_1 \oplus 0a_2 \oplus 0a_3 \oplus 0a_4 \oplus 0a_5 \oplus 0a_6 \oplus 0a_7 = 1, \\ 1a_0 \oplus 0a_1 \oplus 1a_2 \oplus 0a_3 \oplus 0a_4 \oplus 0a_5 \oplus 0a_6 \oplus 0a_7 = 1, \\ 1a_0 \oplus 1a_1 \oplus 0a_2 \oplus 0a_3 \oplus 0a_4 \oplus 0a_5 \oplus 0a_6 \oplus 0a_7 = 1, \\ 1a_0 \oplus 0a_1 \oplus 1a_2 \oplus 0a_3 \oplus 1a_4 \oplus 0a_5 \oplus 1a_6 \oplus 0a_7 = 0, \\ 1a_0 \oplus 1a_1 \oplus 1a_2 \oplus 1a_3 \oplus 1a_4 \oplus 1a_5 \oplus 1`a_6 \oplus 1a_7 = 0, \\ \quad . \ . \ . \qquad\qquad . \ . \ . = 0, \\ \quad . \ . \ . \qquad\qquad . \ . \ . = 0, \\ \quad . \ . \ . \qquad\qquad . \ . \ . = 0. \end{cases}$$

This system has a single solution $x^1 = (1,0,0,1,1,0,0,1)$, which corresponds to the Zhegalkin's polynomial:

$$P_{f(3)} = \overline{e_3} (\overline{e_1} \vee \overline{e_2}) .$$

If the learning sample D and $\overline{D}$ change places, i.e. learning sample becomes sample $\overline{D}$, then the Zhegalkin's polynomial takes the form:

$$P^1_{f(3)} = 1 \oplus P_{f(3)} .$$

The examined decisions is a determining factor in the sense, that when you add a new input variable, system allows no calculations in order to determine the new functions in the nodes.

Indeed, if we consider a network with four inputs with the same sample for the three variables, the polynomial has the form

$$P_{f(4)} = \overline{e_1} \oplus \overline{e_2} e_3 \oplus e_4 \ .$$

This makes it possible to solve the general problem of the above-described method of the synthesis of logical network, which was named by the wave method [9].

Conclusion

The synthesis of adaptive structures, represented by universal logical elements is proposed. The synthesis of such structures is to determine the types of logic functions of a given learning sample of binary vectors. A method for the synthesis of logic functions for configurable logic blocks of the crystals FPGA for a given set of binary vectors (learning set), based on the description of Boolean network by means Zhegalkin's polynomials. Determine the polynomial $P_{f(i)}$ on the basis polynomial $P_{f(r)}$ $(\forall r < i)$ by means the wave method can be a different way by solving a system of linear Diophantine equations in the field of residues modulo 2. Since the synthesis by using the wave method requires only the synthesis of the network for $n = 3$, the complexity of the algorithm synthesis of the whole network is a polynomial of the number of input variables.

References (GOST 7.1:2006)

1. *Palagin, A. V. Reconfigurable computing technology [Text] / A .V. Palagin, V. N. Opanasenko // Cybernetics and Systems Analysis. – 2007. – Vol. 43, № 5. – P. 675–686.*

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

3. Palagin, A. The structure of FPGA-based cyclic-code converters [Text] / A. Palagin, V. Opanasenko, S. Kryvyi // Optical Memory & Neural Networks (Information Optics). – Springer, Heidelberg, 2013. – Vol. 22, № 4. – P. 207–216.

4. Opanasenko, V. N. Partitioning the full range of boolean functions based on the threshold and threshold relation [Text] / V. N. Opanasenko, S. L. Kryvyi // Cybernetics and Systems Analysis. – 2012. – Vol. 48, № 3. – P. 459–468.

5. Palagin, A. V. Design and application of the PLD-based reconfigurable devices [Text] / A. V. Palagin, V. N. Opanasenko // Design of Digital Systems and Devices. – Springer, Heidelberg, 2011. – Vol. 79. – P. 59–91.

6. Bruck, J. Neural networks, error–correcting codes, and polynomials over the binary n–cube [Text] / J. Bruck, M. Blaum // IEEE Transactions on information theory. – IEEE Press, New York, 1989. – Vol. 35, № 5. – P. 976–987.

7. Kryvyi, S. L. Algorithms for solving systems of linear Diophantine equations in integer domains [Text] / S. L. Kryvyi // Cybernetics and Systems Analysis. – 2006. – Vol. 42, № 2. – P. 163–175.

8. Kryvyi, S. L. Algorithms for solving systems of linear Diophantine equations in residue fields [Text] / S. L. Kryvyi // Cybernetics and Systems Analysis. – 2007. – Vol. 43, № 2. – P. 171–178.

9. Opanasenko, V. N. Synthesis of Adaptive Logical Networks on the Basis of Zhegalkin Polynomials [Text] / V. N. Opanasenko, S. L. Kryvyi // Cybernetics and Systems Analysis. – 2015. – Vol. 51, № 6. – P. 969–977.

References (BSI)

1. Palagin, A. V., Opanasenko, V. N. Reconfigurable computing technology. *Cybernetics and Systems Analysis*, 2007, vol. 43, no. 5, pp. 675–686.

2. Kondratenko, Y. P., Klymenko, L. P., Sidenko, I. V., Comparative analysis of evaluation algorithms for decision-making in transport logistics. *Advance Trends in Soft Computing. Proceedings of WCSC. Series: Studies in Fuzziness and Soft Computing*, 2014, vol. 312, pp. 203–217.

3. Palagin, A., Opanasenko, V. N., Kryvyi, S. L. The structure of FPGA-based cyclic-code converters. *Optical Memory & Neural Networks (Information Optics),* 2013, Springer, Heidelberg Publ., vol. 22, no. 4, pp. 207–216.

4. Opanasenko, V. N., Kryvyi, S. L. Partitioning the full range of boolean functions based on the threshold and threshold relation. *Cybernetics and Systems Analysis*, 2012, vol. 48, no. 3, pp. 459–468.

5. Palagin, A. V., Opanasenko, V. N. Design and application of the PLD-based reconfigurable devices. *Design of Digital Systems and Devices,* 2011, Springer, Heidelberg Publ., vol. 79, pp. 59–91.

6. Bruck, J., Blaum, M. Neural networks, error–correcting codes, and polynomials over the binary n–cube. *IEEE Transactions on information theory*, 1989, IEEE Press, New York Publ., vol. 35, no. 5, pp. 976–987.

7. Kryvyi, S. L. Algorithms for solving systems of linear Diophantine equations in integer domains. *Cybernetics and Systems Analysis*, 2006, vol. 42, no. 2, pp. 163–175.

8. Kryvyi, S. L. Algorithms for solving systems of linear Diophantine equations in residue fields. *Cyber-*

netics and Systems Analysis, 2007, vol. 43, no. 2, pp. 171–178.

9. Opanasenko, V. N., Kryvyi, S. L. Synthesis of Adaptive Logical Networks on the Basis of Zhegalkin Polynomials, *Cybernetics and Systems Analysis*, 2015, vol. 51, No. 6, pp. 969–977.

Поступила в редакцию 4.04.2016, рассмотрена на редколлегии 14.04.2016

МЕТОД СИНТЕЗА КОНФИГУРИРУЕМЫХ ЛОГИЧЕСКИХ БЛОКОВ НА ОСНОВЕ УНИВЕРСАЛЬНЫХ ЛОГИЧЕСКИХ ЭЛЕМЕНТОВ

В. Н. Опанасенко, С. Л. Крывый

Предлагается подход к синтезу адаптивных структур, представленных многоуровневой булевой сетью, описываемой ациклическим графом с универсальными логическими элементами. В результате синтеза таких структур определяется тип логической функции булевой сети, заданной обучающей выборкой двоичных векторов, что позволяет использовать эту структуру для задачи классификации входных векторов. В отличие от известных методов синтеза многоуровневой логики, в данной статье предложен метод синтеза таких структур, основанный на описании булевой сети полиномами Жегалкина, начиная с подсети (n=3).

Ключевые слова: адаптация, булева функция, двоичный вектор, FPGA, полином, универсальный логический элемент.

МЕТОД СИНТЕЗУ КОНФІГУРОВНИХ ЛОГІЧНИХ БЛОКІВ НА ОСНОВІ УНІВЕРСАЛЬНИХ ЛОГІЧНИХ ЕЛЕМЕНТІВ

В. М. Опанасенко, С. Л. Кривий

Запропоновано підхід до синтезу адаптивних структур, які представлено багаторівневою бульовою мережею, що описана ациклічним графом з універсальними логічними елементами. В результаті синтезу таких структур визначається тип логічної функції бульової мережі, яка задана навчальною виборкою двійкових векторів, що дозволяє використовувати цю структуру для задачі класифікації вхідних векторів. На відміну від відомих методів синтезу багаторівневої логіці, у даний статті запропоновано метод синтезу таких структур, заснований на опису бульової мережі поліномами Жегалкіна, починаючі з підмережі (n=3).

Ключові слова: адаптація, бульова функція, двійковий вектор, FPGA, поліном, універсальний логічний елемент.

Опанасенко Владимир Николаевич – д-р техн. наук, проф., ведущий научный сотрудник отдела микропроцессорной техники, Институт кибернетики им. В. М. Глушкова НАН Украины, Киев, Украина, e-mail: opanasenko@incyb.kiev.ua.

Крывый Сергей Лукьянович – д-р физ.-мат. наук, проф., проф. факультета кибернетики, Киевский национальный университет им. Тараса Шевченко, Киев, Украина, e-mail: sl.krivoi@gmail.com.

Opanasenko Volodymyr – Doctor of Technical Science, Professor, Leading Researcher of Dept. of Microprocessor Devices, V.M. Glushkov Institute of Cybernetics, National Academy of Sciences, Kiev, Ukraine, e-mail: opanasenko@incyb.kiev.ua.

Kryvyi Sergii – Doctor of Physical and Mathematical Sciences, Professor, Professor of Dept. of Cybernetics, Taras Shevchenko Kiev National University, Kiev, Ukraine, e-mail: sl.krivoi@gmail.com.