DOI 10.1007/s10559-017-9965-z
Cybernetics and Systems Analysis, Vol. 53, No. 4, July, 2017

NEW TOOLS OF CYBERNETICS, INFORMATICS, COMPUTER ENGINEERING, AND SYSTEMS ANALYSIS

SYNTHESIS OF NEURAL-LIKE NETWORKS ON THE BASIS OF CONVERSION OF CYCLIC HAMMING CODES

V. N. Opanasenko[1] **and S. L. Kryvyi**[2]

UDC 51.681.3

Abstract. *This article considers the synthesis of a neural-like Hamming network with a view to implementing the problem of classification of an input set of binary vectors. The formation of a sequence sorted by the Hamming distance as the proximity measure is based on the conversion of cyclic Hamming codes. The correctness of the synthesis of such an implementation for an arbitrary Hamming distance and a binary input vector of arbitrary length is proved.*

Keywords: *Boolean function, neural-like network, Hamming distance, cyclic code.*

INTRODUCTION

A Hamming network [1–4] destined for solving problems of recognition of patterns (speech signals, images, etc.) by means of assigning a binary input vector to one set (or to several sets in the case of the same proximity measure) out of given sets of patterns is determined by the maximum proximity measure in the capacity of which the Hamming distance is used.

The correctness of a classification or a decision-making process depends on the spread in values of patterns, however, the pattern with the maximum proximity measure is not always most plausible for an input model. This drawback is eliminated during implementing a network at the second level of processing of which an ordered group containing patterns is formed, and their proximity measures do not exceed an allowable (from the viewpoint of the plausibility of classification) value. The implementation of such a network using a unified element base is very problematic in view of the large cardinality of a set of patterns in many recognition problems.

This article proposes to synthesize structures of a neural-like Hamming network that are oriented towards FPGA crystals [5, 6] and carrying out the formation of ordered groups of vectors–patterns whose proximity measures do not exceed admissible values [3]. The formation of such groups of vectors–patterns is based on cyclic Hamming codes [7, 8] whose set can be used as close (according to the Hamming distance) to a reference code.

SYNTHESIS OF A CONVERTER FOR A CYCLIC HAMMING CODE OF ARBITRARY LENGTH AND FOR AN ARBITRARY HAMMING DISTANCE

Let $X = \{0,1\}$ be the binary alphabet, and let $F_n(X)$ be the set of all words of length n over the alphabet X; the latter set is called the exhaustive set of words over this alphabet. It is obvious that $|F_n(X)| = 2^n$. Let $v = (x_1 x_2 \dots x_n) \in F_n(X)$, and let R be the operator of cyclic shift with the step 1, which is defined as follows:

$$Rv = (x_n x_1 x_2 \dots x_{n-1}) \quad \forall v = (x_1 x_2 \dots x_{n-1} x_n) \in F_n(X).$$

[1]V. M. Glushkov Institute of Cybernetics, National Academy of Sciences of Ukraine, Kyiv, Ukraine, *opanasenko@incyb.kiev.ua*. [2]Taras Shevchenko National University of Kyiv, Kyiv, Ukraine, *krivoi@i.com.ua*.
Translated from Kibernetika i Sistemnyi Analiz, No. 4, July–August, 2017, pp. 155–164. Original article submitted February 17, 2017.

We call a set V of words from $F_n(X)$ closed with respect to the shift operator R a cyclic Hamming code. The closedness of the set V means that $\forall v \in V\ (Rv \in V)$. A cyclic Hamming code can be considered as a subset of words close (in terms of the Hamming distance) to a given reference word over the alphabet X.

If a word composed of 1s is chosen as a reference word, then, in the case when the Hamming distance is equal to 1, all the words from the set V except for the reference one are generated by its arbitrary element by virtue of the closedness of this set with respect to the cyclic shift operator. In what follows, such a word is called a generating word.

We introduce the following notation:

$H_m(1)$ is a cyclic code of length n whose generating word includes an m-component ($m = \overline{1, n-1}$) group of cyclically adjacent 1s and a d-component ($d = (n-m)$) group of 0s;

$H(1)$ is a cyclic code of length n whose generating word includes an n-component group consisting of 1s;

$H_m(0)$ is a cyclic code of length n whose generating word includes an m-component ($m = \overline{1, n-1}$) group of cyclically adjacent 0s and a d-component ($d = (n-m)$) group of cyclically adjacent 1s;

$H(0)$ is a cyclic code of length n whose generating word includes an n-component group consisting of 0s.

PROBLEM STATEMENT

Let $F_n(X)$ be the exhaustive set of words of length n over the alphabet $X = \{0, 1\}$, and let $H_m(1)$ (accordingly, $H_m(0)$) be a cyclic code of length n. It is necessary to construct a logical structure that implements the mapping $\Im$ defined as follows:

$$\Im(H_m(1)) = 1,\ \Im(F_n(X) \setminus H_m(1)) = 0$$

$$(\text{accordingly},\ \Im(H_m(0)) = 1,\ \Im(F_n(X) \setminus H_m(0)) = 0).$$

The problem can be solved as a result of series connection [9–11] of cyclic structures of the types AND and NOR that have n inputs and n outputs and also an OR structure with n inputs and one output. In the general case, problems of synthesis of multilevel logical structures with one and many outputs for the classification of binary input vectors are considered in [12–15]. Structures for the posed problem have an r-level organization whose kth level ($k = \overline{1, r}$) consists of n logical elements AND implementing the conversion

$$\omega(a_i, a_{(i+s) \bmod (n)})\ \ \forall i = \overline{0, n-1},\ 1 \le s \le (n-1),$$

where ω is a logical function of two variables; a_i and $a_{(i+s)}$ are one-bit components of a binary input vector; $s \in [1, n-1]$ is a cyclic shift step. At one level, all elements are adjusted to perform the same logical AND or NOR function.

Proceeding from the truth table of the logical AND function and the structure of connections ($\forall k, s = 1$), the cardinality of a group of cyclic adjacent 1s decreases by one with increasing the number of level k. Thus, to convert a code word with a given value of m, it is necessary to have $m-1$ levels.

As multilevel logical structures (Fig. 1), the structures S_1 and S_2 are chosen ($n = 4$ and the step $i = 1$ in the operator R). Proceeding from the truth table of the logical operations AND and NOR and also the structure of connections ($\forall k, s = 1$), with increasing the number of the kth level, the number of cyclically adjacent 1s decreases by one for the operation AND, and, for the operation NOR, on the contrary, the number of 0s is increased by one if a word containing d cyclic adjacent 1s is applied to the input. The structures S_1 and S_2 were synthesized earlier in [7, 8, 11]. If a word containing a group of d cyclically adjacent 0s is applied to the input of the kth level of the structure S_1 ($1 \le k < r$) with such connections, then its output will consist of a word containing a group of $d+1$ cyclically adjacent 0s. As is shown in [7], it suffices to investigate a structure of the type S_1 since, after the first level, a structure of the type S_2 is transformed into a structure of the type S_1 with an input word $H_{n-d-1}(1)$.

An implementation of such structures (for $n = 4$ and $s = 1$) is presented in Fig. 1, and the conversion results are given in Table 1.

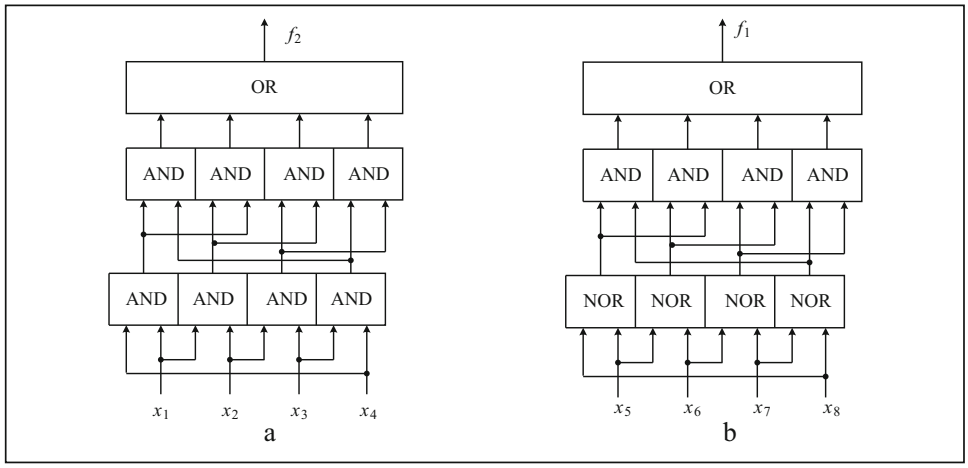

Fig. 1. Structure of a converter of cyclic codes on the basis of operations AND, NOR, and OR; (a) the structure of the type S_1 and (b) the structure of type S_2.

TABLE 1. Conversion of Cyclic Codes

Results of Conversion by the Structure S_1					**Results of Conversion by the Structure S_2**				
Input code				Output f_2	Input code				Output f_1
x_4	x_3	x_2	x_1		x_8	x_7	x_6	x_5	
1	1	1	1	1	0	0	0	0	1
0	1	1	1		0	0	0	1	
1	0	1	1		0	0	1	0	
1	1	0	1		0	1	0	0	
1	1	1	0		1	0	0	0	
0	0	0	0	0	0	0	1	1	0
0	0	0	1		0	1	0	1	
0	0	1	0		0	1	1	0	
0	0	1	1		0	1	1	1	
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	0	0	0		1	1	0	0	
1	0	0	1		1	1	0	1	
1	0	1	0		1	1	1	0	
1	1	0	0		1	1	1	1	

The substantiation of properties of the presented structures for cyclic Hamming codes is carried out in [7] where the following theorem is proved.

THEOREM 1. If a word with d cyclically adjacent 1s is applied to the input of the first level of the computational structure S_2 with the operation NOR, then we have a word with $d+1$ cyclically adjacent 0s at its output. If a word with d cyclically adjacent 0s is applied to the input of the kth level of the structure S_1 ($1 \le k < r$), then we have a word with $d+1$ cyclically adjacent 0s at the output of this level.

The structure S_1 identifies a half-byte (a 4-bit word consisting of 0s and 1s) that contains three or four 1s, and the structure S_2 identifies a half-byte containing one 1 or only 0s. The half-bytes consisting only of 0s $H(0)$ or only of 1s $H(1)$ are called singular points whose identification is described below.

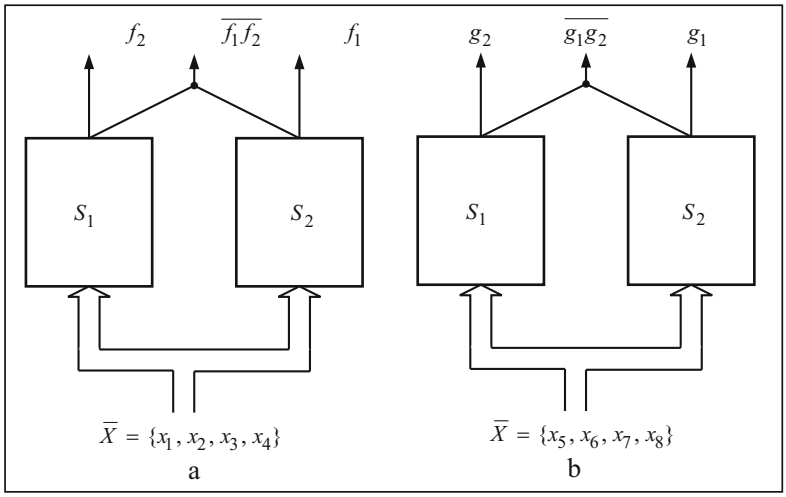

Fig. 2. Basic structure of converters of cyclic codes for the first half-byte (a) and for the second half-byte (b).

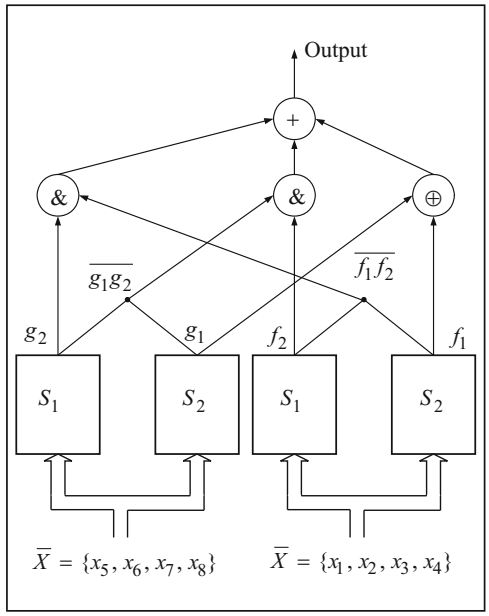

Fig. 3. Network structure for the identification of three 1s in the input byte.

The considered structures are structures of sequential type, and structures of parallel-sequential type can be similarly implemented on $r = \log_2 n$ levels. Each level of a parallel-serial structure has a regular structure of connections, i.e., the components with indices i and $(i+s)$ (the value of $i+s$ is taken modulo n) arrive at the input of any ith logical element of the kth level.

We now consider the general problem of classification of input half-bytes that can have an acyclic structure. To this end, we will consider a basic structure consisting of two 4-digit substructures of the same type that are constructed from the substructures S_2 and S_1 whose outputs are denoted by the symbols f_1, f_2 and g_1, g_2 respectively (Fig. 2).

A feature of this structure is substantiated by the following statement.

LEMMA 1. If singular points do not participate in classification, then l 1s in the input byte are identified with the help of the presented substructures, where $1 \le l \le 6$.

Proof. For the substructures, it follows from Theorem 1 and Table 1 that the outputs of both substructures are mutually exclusive, i.e., if 1 is at the output f_1 (f_2), then we have 0 at the outputs f_2 and $\overline{f_1 f_2}$ (f_1 and $\overline{f_1 f_2}$). This is also true for g_1 (g_2).

We have 1 at the outputs of the substructures f_2 and g_2 if, at the input of the substructures S_1 and S_2, the half-byte includes only one 1 or all 0s. The outputs of the substructures f_1 and g_1 assume 1 if, at the inputs of the substructures S_1 and S_2, the half-byte has exactly three 1s or has all 1s. The outputs of the substructures $\overline{f_1 f_2}$ and $(\overline{g_1 g_2})$ assume 1 if the input half-byte of the substructures S_1 and S_2 contains exactly two 1s.

This implies that the tuning of both substructures to the initialization of three 1s in one byte assumes the form $(f_2 \& \overline{g_1 g_2}) + (g_2 \& \overline{f_1 f_2}) + (f_1 \oplus g_1)$, where the symbol $\oplus$ denotes the XOR operation and the symbol + denotes the disjunction operation. The tuning to four 1s in the (input) byte (without regard for the singular point $H(1)$) assumes the form $(f_2 \& g_1) + (g_2 \& f_1) + (\overline{g_1 g_2} \& \overline{f_1 f_2})$. The tuning to five 1s in the byte (without regard for the singular point $H(1)$) assumes the form $(f_1 \& \overline{g_1 g_2}) + (g_1 \& \overline{f_1 f_2})$. The tuning to six 1s in the byte (without regard for the singular point $H(1)$) is of the form $f_1 \& g_1$. The statement is proved.

In Fig. 3, the network structure for the identification of three 1s in the byte is presented.

Remark 1. In the general case, if a sample consisting not of 2^k bits but, for example, of 11 bits is applied to the input of this structure, then it singles out two half-bytes, the remained three bits are supplemented by one bit to the half-byte, and, thereby, the problem is reduced to the previous case.

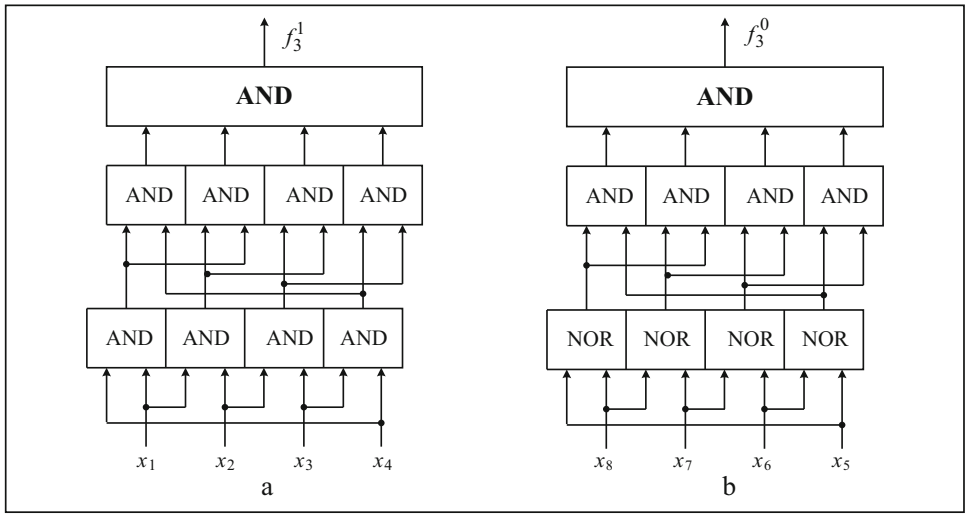

Fig. 4. Structure of the converter of cyclic codes on the basis of the AND and NOR operations for determining singular points; (a) the structure of type T_1 and (b) the structure of type T_2.

TABLE 2. Conversion of Cyclic Codes for Singular Points

Results of Conversion by the Structure T_1					**Results of Conversion by the Structure T_2**				
Input code				Output f_3^1	Input code				Output f_3^0
x_4	x_3	x_2	x_1		x_8	x_7	x_6	x_5	
1	1	1	1	1	0	0	0	0	1
0	1	1	1		0	0	0	1	
1	0	1	1		0	0	1	0	
1	1	0	1		0	1	0	0	
1	1	1	0		1	0	0	0	
0	0	0	0		0	0	1	1	
0	0	0	1		0	1	0	1	
0	0	1	0		0	1	1	0	
0	0	1	1	0	0	1	1	1	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	0	0	0		1	1	0	0	
1	0	0	1		1	1	0	1	
1	0	1	0		1	1	1	0	
1	1	0	0		1	1	1	1	

IDENTIFICATION OF SINGULAR POINTS

It follows from Lemma 1 that the tuning to seven and eight 1s in the input byte requires allowance for singular points, in particular, $H(1)$ points in input half-bytes. Since the structure S_2 (accordingly, S_1) also identifies four 1s (four 0s) together with three 1s (three 0s), it should be modified so that singular points can be singled out. This can be made as a result of a simple modification of the basic structure S_2 (accordingly, S_1) by replacing the output function OR with AND in it. The structure for converting codes is given in Fig. 4, namely, the structure T_1 for the singular point $H(1)$ and the structure T_2 for the singular point $H(0)$, and the results of conversion are given in Table 2.

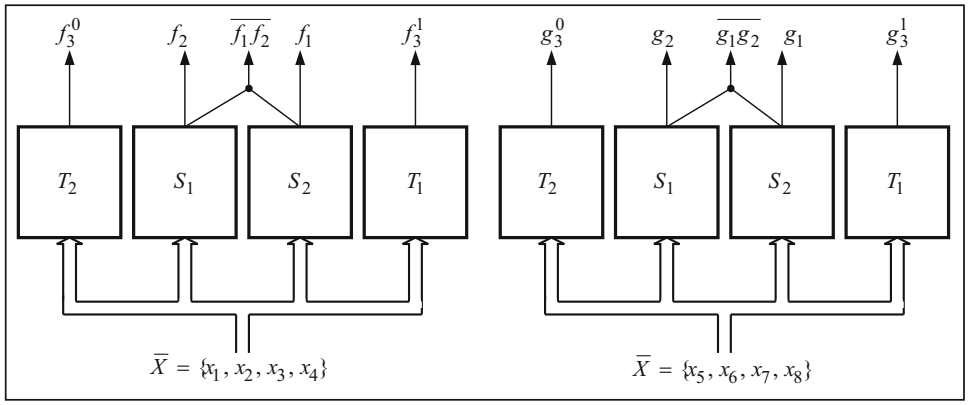

Fig. 5. General network structure with allowance made for singular points.

We denote the outputs of these substructures by f_3^0 (g_3^0) and f_3^1 (g_3^1) to identify four 0s and four 1s, respectively. Proceeding from the above tables, we have 1 at the output f_3^1 of the modified structure if the value of the input half-byte is the singular point $H(1)$. Similarly, for the second modified structure, we have 1 at the output f_3^0 if the value of the input half-byte is the singular point $H(0)$.

Thus, connecting the outputs f_3^1 and f_1 with the help of the AND operation, we obtain 1 at the output only when there are four 1s (the singular point $H(1)$) in the half-byte at the input. The semi-structure for the identification of the singular point $H(0)$ on the basis of the semi-structures f_3^0 and f_2 is similarly constructed. The construction described above is shown in Fig. 5.

The substantiation of the properties of the presented structure is given by the following theorem.

THEOREM 2. The general structure identifies an arbitrary content of the input byte by tuning its outputs to the number of 1s (0s) in the byte.

Proof. Lemma 1 and the modified substructures presented above imply that, to identify four 1s in the byte, the identification function is of the form

$$(\overline{g_3^1}\ \&\ f_2\ \&\ g_1\ \&\ \overline{f_3^0}) + (\overline{f_3^1}\ \&\ g_2\ \&\ f_1\ \&\ \overline{g_3^0}) + (\overline{g_1 g_2}\ \&\ \overline{f_1 f_2})$$

$$+ [(f_1\ \&\ f_3^1) \oplus (g_1\ \&\ g_3^1)].$$

In fact, 1s in the input byte can be distributed so that one 1 is in the first (in the second) half-byte of the first substructure and three 1s are in the second (first) half-byte of the second substructure.

The first two terms in the above-mentioned expression imply that

— if $g_3^1 = 0$, then the input half-byte of the second substructure contains no more than three 1s, and if $g_1 = 1$, then it contains exactly three 1s;

— if $f_3^1 = 0$, then the input half-byte of the first substructure contains no more than three 0s, and if $f_2 = 1$, then it contains only one 1.

Then the byte will contain exactly four 1s. The validity of the aforesaid for the second expression follows from the symmetry of arrangement, and it is obvious for the other two expressions. Five 1s in the input byte are identified with the help of tuning to the function

$$(f_2\ \&\ f_3^0\ \&\ g_1\ \&\ g_3^1) + (g_2\ \&\ g_3^0\ \&\ f_1\ \&\ f_3^1) + (\overline{f_1 f_2}\ \&\ g_1\ \&\ \overline{g_3^1}) + (\overline{g_1 g_2}\ \&\ f_1\ \&\ \overline{f_3^1}).$$

The other combinations of 1s in the byte can be similarly obtained.

Eight 1s in the byte are identified with the help of tuning to the function $(f_1 \,\&\, f_3^1) + (g_1 \,\&\, g_3^1)$.

The theorem is proved.

Taking into account Remark 1, note that Theorem 2 is true for an arbitrary Hamming distance and an arbitrary length of the input vector in bits.

STRUCTURAL ORGANIZATION OF HAMMING NETWORKS

Problem statement [5]. A set of vectors–patterns $B \subset U$ (U an exhaustive set of n-dimensional binary vectors $\{u_p\}$, $p = \overline{1, 2^n}$; $B = \{b_h\}$, $h = \overline{1, q}$, $q < 2^n$, $\bigcap\limits_h b_h = \varnothing$) is given.

For the given set of vectors–patterns B and an arbitrary set $U_1 \subseteq U$ ($U_1 = \{u_j\}$, $j = \overline{1, \alpha}$), determine sets $C_j = \{c_{jh}\}$ and form sets $Y_j = \{y_j^i : i = \overline{1, z_j}\}$; at the same time

$$y_j^i = \mu_{jh} \mapsto b_h, \tag{1}$$

where the symbol $\mapsto$ denotes concatenation; c_{jh} is a binary vector (nonpositional binary code) determining the Hamming distance between n-dimensional binary vectors u_j and b_h,

$$c_{jh} = \sum_{\tau=1}^{n} (u_{j\tau} \oplus b_{h\tau}); \tag{2}$$

the symbol $\oplus$ denotes the XOR operation; $u_{j\tau}$ and $b_{h\tau}$ are the τth binary components, $u_{j\tau}$, $b_{h\tau} \in \{0, 1\}$, of vectors u_j and b_h, respectively; μ_{jh} is the binary code of the Hamming distance (determined by the number of components composed of 1s in the nonpositional binary code c_{jh}).

According to formulas (1) and (2), a Hamming network includes the following basic functional units (Fig. 6): RAM for patterns; converter (2) based on the logical element XOR; a converter $\Lambda(c_{jh} \Rightarrow \mu_{jh})$, and RAM for results.

A set of vectors $C_j = \{c_{jh}\}$ arrives at the input of converter (2). A set $Y_j = \{y_{jh}^i : i = \overline{1, z_j}\}$ is formed at its output.

Consider now an example of functioning this neural-like Hamming network.

A proximity measure $\mu_{jh} \le 2$ and the set of patterns

$$B = \{b_1 = 10101011;\ b_2 = 11111001;\ b_3 = 00011101;\ b_4 = 00010111;\ b_5 = 10100100\}$$

are given. The binary vector $u = 10100111$ is applied to the input of the neural-like network. It is required to determine the vectors–patterns b_n in which the proximity measure of a binary input vector c_{jh} satisfies the constraint $\mu_{jh} \le 2$. Taking into account Theorems 1 and 2 and the synthesized structures (see Figs. 3 and 5), we obtain the results of conversion performed by the Hamming network (Table 3).

The constraint is satisfied by the patterns $b_1 = 10101011$ and $b_5 = 10100100$, and it is precisely these patterns that will be written into the corresponding memory area of RAM for results.

The maximum dimension of the Hamming distance μ_{jh} is $(\log_2 n + 1)$; thus, according to formula (1), the dimension (length) of a vector y_j^i is determined by the value of $(n + \log_2 n + 1)$. The vectors of the set y_j^i (1) are sequentially written into RAM for results. The size of RAM for results is determined by the total cardinality of the sets Y_j that is determined by the following expression:

$$\text{Card}\{Y\} = \sum_j \text{Card}\{Y_j\},$$

where

$$\text{Card}\{Y_j\} = 1 \quad (\mu_{jh} = 0);$$

$$\text{Card}\{Y_j\} = n \quad (\mu_{jh} = 1);$$

$$\vdots$$

$$\text{Card}\{Y_j\} = \frac{n!}{(n - \mu_{jh})!\,\mu_{jh}!} \quad (\mu_{jh} = n).$$

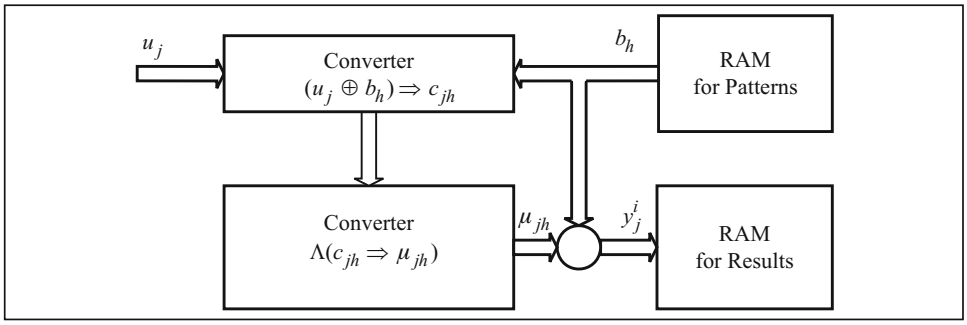

Fig. 6. Functional scheme of a Hamming network.

TABLE 3. Results of Conversion performed by the Neural-Like Hamming Network

Input Vector u_j	**Results of Conversion**		
	Vectors–patterns b_h	c_{1h}	μ_{1h}
$u_1 = 10100111$	$b_1 = 10101011$	$c_{11} = 00001100$	$\mu_{11} = 2$
	$b_2 = 11111001$	$c_{12} = 01011110$	$\mu_{12} = 5$
	$b_3 = 00011101$	$c_{13} = 10110010$	$\mu_{13} = 4$
	$b_4 = 00010100$	$c_{14} = 10111010$	$\mu_{14} = 5$
	$b_5 = 10100100$	$c_{15} = 00000011$	$\mu_{15} = 2$

According to the above expressions, for the subset of the input vectors that have the same Hamming distance μ_{jh}, a fixed memory space is allocated in RAM for results for the corresponding value of μ_{jh}, and this memory space strictly corresponds to the value of Card $\{Y_j\}$.

CONCLUSIONS

This article substantiates the correctness of the synthesis of a multilevel structure of combinational type with the help of the basic AND and NOR operations (a converter of cyclic Hamming codes) for recognizing a subset of binary vectors (of arbitrary length) whose proximity measure (the Hamming distance) belongs to some range.

Based on the considered structures, a neural-like Hamming network is implemented that recognizes a binary input vector by determining its proximity measure with given vectors–patterns and writes it (together with the Hamming distance value) into the corresponding memory area for results.

The considered structures can also be used to solve a wide class of pattern recognition problems in which the Hamming distance is used as the proximity measure.

REFERENCES

1. T. K. Vintsiuk, Analysis, Recognition, and Understanding of Speech Signals [in Russian], Naukova Dumka, Kyiv (1987).
2. J. Bruck and M. Blaum, “Neural networks, error-correcting codes, and polynomials over the binary *n*-cube,” IEEE Transactions on Information Theory, Vol. 35, No. 5, 976–987 (1989).
3. M. E. Robinson, H. Yoneda, and E. Sanchez-Sinencio, “A modular CMOS design of a Hamming network,” IEEE Transactions on Neural Networks, Vol. 3, No 3, 444–456 (1992).

4. V. D. Dmitrienko and A. Yu. Zakovorotnyi, “A neural network using the Hamming distance for image recognition on the border of several classes,” Herald of the NTU “KhPI,” No. 39 (1012), 57–67 (2013).
5. A. V. Palagin, V. N. Opanasenko, and L. G. Chigirik, “Synthesis of a Hamming network on the basis of programmable logic integrated circuits,” Engineering Simulation, Vol. 13, 651–666 (1996).
6. Y. P. Kondratenko and E. Gordienko, “Implementation of neural networks for adaptive control system on FPGA,” in: Proc. 23rd DAAAM Intern. Symp. on Intelligent Manufacturing and Automation, Vol 23 (1), B. Katalinic (ed.), DAAAM International, Vienna, Austria (2012), pp. 0389–0392.
7. A. V. Palagin, V. N. Opanasenko, and S. L. Kryvyi, FPGA-Based Reconfigurable Structures: Synthesis of Problem-Oriented Structures, Lambert Academic Publishing (2014).
8. A. Palagin, V. Opanasenko, and S. Kryvyi, “The structure of FPGA-based cyclic-code converters,” Optical Memory & Neural Networks (Information Optics), Vol. 22, No. 4, 207–216 (2013).
9. A. V. Palagin and V. N. Opanasenko, “Reconfigurable computing technology,” Cybernetics and Systems Analysis, Vol. 43, No. 5, 675–686 (2007).
10. A. V. Palagin and V. N. Opanasenko, “Design and application of the PLD-based reconfigurable devices,” in: M. Adamski, A. Barkalov, and M. Wegrzyn (eds.), Design of Digital Systems and Devices, Lecture Notes in Electrical Engineering, Vol. 79, Springer, Berlin-Heidelberg (2011), pp. 59–91.
11. A. V. Palagin and V. N. Opanasenko, and S. L. Kryvyi, “Method for synthesis of structures for transformations of a cyclic code on the basis of FPGA,” Electronic Modeling, Vol. 36, No. 2, 27–48 (2014).
12. R. K. Brayton, G. D. Hachtel, and A. L. Sangiovanni-Vincentelli, “Multi-level logic synthesis,” in: Proc. IEEE, Vol. 78, No. 2, 38–83 (1990).
13. V. N. Opanasenko and S. L. Kryvyi, “Partitioning the full range of Boolean functions based on the threshold and threshold relation,” Cybernetics and Systems Analysis, Vol. 48, No. 3, 459–468 (2012).
14. V. N. Opanasenko and S. L. Kryvyi, “Synthesis of adaptive logical networks on the basis of Zhegalkin polynomials,” Cybernetics and Systems Analysis. Vol. 51, No. 6, 969–977 (2015).
15. V. N. Opanasenko and S. L. Kryvyi, “Synthesis of multilevel structures with multiple outputs,” in: CEUR Workshop Proceeding of 10th International Conference of Programming (UkrPROG 2016), Vol. 1631, Code 122904, Kyiv, Ukraine (2016), pp. 32–37.