Error Correction and Detection System Based on Hopfield Networks

L. Ionescu, C. Anton, I. Tutănescu, A. Mazăre, G. Şerban
Faculty of Electronics, Communications & Computers
University of Pitesti, Romania

Abstract - **Error correction is a basic feature of a communication system. Therefore there are sought, and still seek, many solutions to this problem. In this paper we present a solution for the correction of errors that are made by means of associative memories based on Hopfield networks. The solution we found solves many problems regarding both random error correction and burst type occurring in the transmission of a code word to the communication channel.**

Hopfield networks, associative memories, error correction and detection.

I. INTRODUCTION

Associative memories are part of the content addressable memories. In other words, they do not require submission of an address to access a specific location (location content). By submitting a value (number, bits string, etc.), named input data, it searches location containing that value - that is why we say the content addressable memory [1]. Furthermore, others data with input data can exist to this location. These data can be returned by memory. So, we can say that the input data is associated with data from location which are returned as output data. Therefore, it is used the associative memory name.

Another situation could be the following: input data does not correspond exactly to the content of any location. In this situation it can return the contents of a memory location that contains a data close to the input data. What means "close"? It depends on how it was built memory: if the memory was designed to store bits strings, then “close” refers to the Hamming distance between strings (number of different bits). This latter approach is the one that interests us in this paper.

There are several types of applications which use this kind of memory. Especially helpful are the hardware implemented associative memories (as in ASIC, FPGA chips), where each location is a hardware component and the searching operation is done parallel and simultaneously in all locations. This results in a response time that is comparable to that of conventional memories.

Hardware associative memories are used in applications that can perform object recognition (for their classification) in an image, reconstruction and restoration of images, even in technology processes [2] or the identification of patterns of sounds. We decided to use these memories to detect and correct errors in a message.

The basic principle is simple: we define an alphabet - the message to be transmitted - and then for each letter (message) it is chosen a code word. The choice of code words is pursuing one rule - Hamming distance between them is not less than the value *d*. The error correction capacity is $d/2 - 1$ and the error detection capacity is $d/2$. Code words are stored in an associative memory, which is a part of the decoder. Encoder is a simple table that can associate letter (message) with code word. A block diagram of the system is shown in Fig. 1.

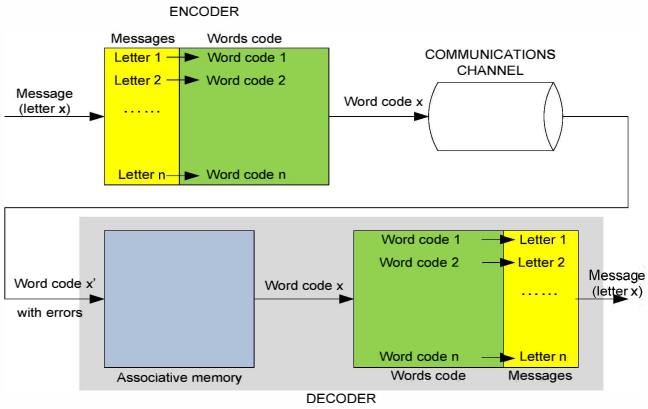

Figure 1. Block diagram of communication system based on associative memory.

At the reception site, the decoder (associative memory) may receive either a code word which corresponds identical to one found in memory (in which case we say that the reception was made without error), or a code word that does not match with one of the memory, but it is close to one stored in memory (the smallest Hamming distance). This adjacent word (in Hamming space) will be returned as output data in memory. Thus, we achieve automatic detection and correction of errors.

There are several schemes that can be implemented using associative memory. Most common scheme uses a “closer”: each location (transmission, reception) contains a registry where data is stored, a comparator circuit, which compares input with the contents and a circuit of minimum propagation [3]. This type of associative memory has been used to implement an encoder - decoder for BCH error correction with very good performances [4].

II. ASSOCIATIVE MEMORIES AND HOPFIELD NETWORKS

A. *Associative memories with Hopfield networks*

The Hopfield model is based on an interconnected network. Each node is a logic cell called neuron, and this is connected in hardware implemented networks to other neurons by weights circuits [5]. Although terms of the terminology are from neural networks, Hopfield network does not use training

algorithms. If we consider X as input of the network, which is the state of the network at a time then the state of the network at $t+1$ time is:

$$X_{t+1} = X_t W \quad (1)$$

where X_t and X_{t+1} are the array of neurons status at t and $t+1$ time and W is the weight array. In W array, the connection weight between a neuron and itself is 0. W is symmetrical to the main diagonal [6]. Thus, if we have x_i and x_j two neurons then $w_{ij} = w_{ji}$.

For n neurons network equation (1) becomes:

$$\begin{aligned} x_{t+1}^1 &= x_t^2 w_{12} + x_t^3 w_{13} + ... + x_t^n w_{1n} \\ x_{t+1}^2 &= x_t^1 w_{12} + x_t^3 w_{23} + ... + x_t^n w_{2n} ... \\ x_{t+1}^n &= x_t^1 w_{1n} + x_t^2 w_{2n} + ... + x_t^{n-1} w_{n-1n} \end{aligned} \quad (2)$$

where x_{t+1}^i, x_t^i represent the status of the neuron i at $t+1$ and t times and w_{ij} is the weight between i and j neuron.

For each neuron output is defined an activation function called threshold. Finally, output of each neuron is given by relation (3):

$$y_{t+1}^i = \begin{cases} 1 \ if \ \sum_{j \neq i} x_t^j w_{ij} > \theta_i \\ -1 \ otherwise \end{cases} \quad (3)$$

where y_{t+1}^i output of neuron i at $t+1$ time and θ_i threshold for i neuron.

The Hopfield network is a resonance, bidirectional associative memory (BAM), in sense that its status changes in order to achieve a stable state at an equilibrium point. Hopfield model is accepted by the physicists because it is isomorphic with Ising model of magnetism (to zero absolute temperature) [7]. Thus is introduced a function called energy function given by the relation:

$$E = -\frac{1}{2}\sum_{i,j=1}^{n} x_i x_j w_{ij} + \frac{1}{2}\sum_{i=1}^{n} y_i \theta_i \quad (4)$$

This function is important because it establishes the degree of convergence of the network. Thus, for any weights array W, there is a minimum amount of energy, in each iteration, that the system tends to. When this value is reached the system remains stable. The target is to find the W array which certain desired states are the stable states.

One of solutions for find W is to apply Hebb's rule. This rule says that if two neurons are connected then if they are in the same state of excitement, then the weight of connection between then is strong. By applying multiple patterns to be saved, we have weight ratio between two neurons given by relation:

$$w_{ij} = \sum_k x_i x_j \quad (5)$$

where w_{ij} represents the weight of the connection between x_i and x_j and k represents the patterns number that wish to store.

Because of the second term of the addition in relation (4), the storage capacity of Hopfield network is limited to:

$$m = 0.18 \cdot n \quad (6)$$

where m represent number of stored vectors and n number of neurons in the network.

In conclusion, for our Associative CAM (A-Content Addressable Memory) memory we use Hopfield network model. Stable states of the network represent stored data patterns. By applying a certain input, the data network will automatically converge to a stable state that will be associated with that data input.

B. Hopfield networks used to error correction and detection

A clearer analysis on Hopfield network storage method can be given using the Hamming distance between stored vectors (word codes). If we store two vectors with the Hamming distance d then each vector generates a basin of attraction for any other vector with Hamming distance $d/2 - 1$. So any vector with Hamming distance $d/2$ - 1 will converge to the attractor. Based on the foregoing we can issue the following sentence:

P1: *Hopfield network that has stored words (binary vectors) with minimum Hamming distance d between them can be used for correction of maximum d/2-1 errors and for detection of maximum d/2 errors.*

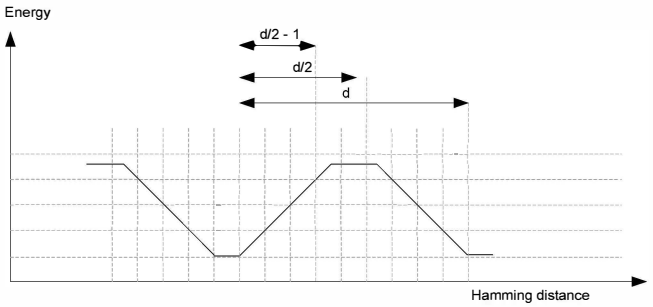

Figure 2. Network energy versus hamming distance diagram. Low level energy states correspond to stored words

Number of words you can store is limited by relation (6). For example, a binary Hopfield network with 16 neurons can store 16-bit words. This is because each neuron can be in state 1 (excited) or in state 0 (inhibited). The total number of combinations is 2^{16}. As we will demonstrate in the next section, we have 32 of these combinations of words having Hamming distance each other at least 7 (because we want to design a network that correcting 3 erroneous bits and detecting 4).

However the network can store only:

$m = 0.18 \cdot 16 = 2.88$ words.

To store 32 words we need:

$n = 32 / 0.18 = 177.77 \sim 178$ neurons.

Such a network would raise two major issues:

- the large size would lead to implementation in a very complex hardware circuit, which is manifested in high response times and large occupied area in silicon surface;
- it should apply an encoding schema in which the input word is converted into a 178-bit word code, unacceptably high in communications (in our example for 5 message bits we have 178 word code bits).

All these problems we solved in a personal manner which will be presented in the next section.

III. SYSTEM PRESENTATION

A. Bipolar coding schema

In the previous section we presented the network characteristics in terms of Hebb's learning rule. A neuron can have two states, observing biological model: excited and inhibited state. There are two ways you can encode these two states: unipolar and bipolar coding. In an unipolar encoding, excited means 1 and inhibited means 0. When it is used the bipolar encoding, excited is 1 and inhibited -1. This second method would seem to have better results than the first, so it will be used in our case.

However, the words that we receive are saved to a binary format (string consisting of 0s and 1s), which can be used in transmissions and can be stored in digital circuits. To facilitate conversion to a bipolar format we defined a special operator which is used to calculate weight, as it is displayed in Table I:

TABLE I. BIPOLAR CODING OPERATOR:

A	B	A ○ B
0	0	1
0	1	-1
1	0	-1
1	1	1

This operator can be implemented easily hardware. An interesting consequence of applying this coding schema, which was deduced experimentally, is that it takes place automatically the storage of the reverse (orthogonally) vector. For example, if you have a network with 8 neurons and store word *1111 0000,* then it automatically saves its reverse *0000 1111* so a word with Hamming distance given by the network size – 8. Based on this observation we issue a new sentence that is:

P2: *In a Hopfield network of n neurons, after Hebb's rule, for a bipolar encoding vector y, the orthogonal vector* y_{bar} *with Hamming distance n will be automatically stored.*

This vector is present "physically" in memory, even if its storing not is explicitly performed. For the above example, if we have an input vector *0000 1110* then it will convert to *0000 1111*.

B. Parallel Hopfield networks

The proposed objective in this article is the following: we have an alphabet consisting in *n* letters, each letter is encoded in binary $\log_2 n$ bits. We want to implement a system for error detection and correction, in conditions that we send these letters on a noisy communication channel. We wanted to achieve a 3 erroneous bits correction and 4 bits erroneous detection. The solution to this problem is the memorization of words code (which will be associated with messages), each having Hamming distance equal to 7. First, we determine the minimum bits size that code words must have in order to generate *n* code words with Hamming distance 7.

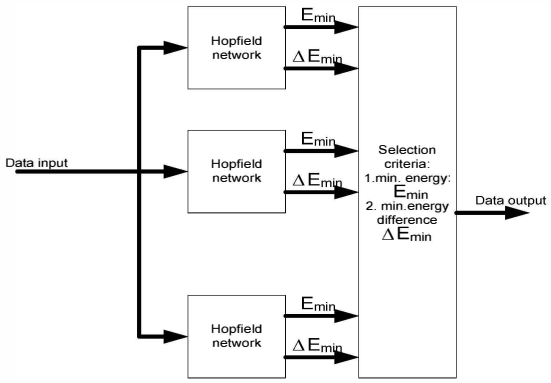

Figure 3. Associative memory using parallel Hopfield networks

For example, to encode a 32-letter alphabet will require a minimum of 16-bits word code. We had done this using of a simulator (developed by us), which will be described in the next section. Once we have obtained these codes, they were stored in an associative memory based on the Hopfield network, having a number of neurons equal with the size of code words (see Fig. 3).

IV. EXPERIMENTAL RESULTS

A block diagram of the system used for the experiment described above is shown in Fig. 4. The encoder is simple and consists only of a table with the associations of message and a word code. The association table contains the following data: message with LSB bit 0 (which will be associated with a word code) and letter having bit LSB 1 with its reverse. LSB bit status of the message decides whether or not to reverse the word code to be transmitted on the communication channel.

The decoder consists of two parts. The first module is an associative memory based on Hopfield networks operating in parallel. Here takes place the word code reception from communication channel, error detection and correction process based on the intrinsic property of associative memory. The second part of the decoder handles reception of words code (error free) and converts them back into messages.

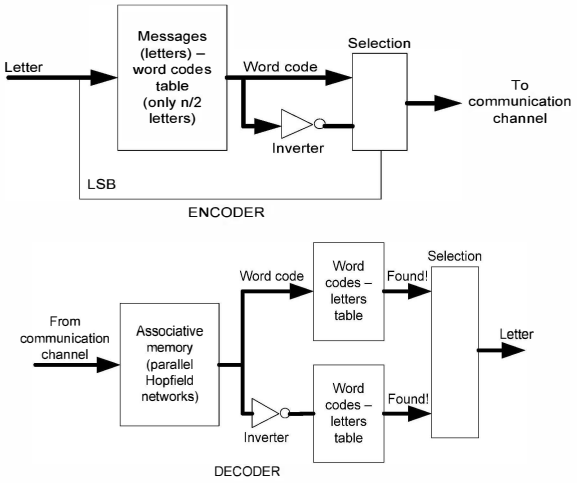

Figure 4. Encoder and decoder used in experiments.

In this paper, our final goal was to achieve a system for the detection of 4 erroneous bits and for 3-bit error correction. We simulated the system in the following steps:

1. We made a program (C) in order to generate *n*/2 words of code and their inverse with minimum Hamming distance 7 between them. Using C program we determined how many bits it takes to have the words.
2. With size of word code *d* we have built a Hopfield network having *d* neurons.
3. We determined how many networks are needed to store all *n* points, based on estimated formula for calculating the storage capacity. All these networks operate in parallel as previously described.
4. We computed the circuit complexity consisting of parallel Hopfield networks. Hopfield network includes two types of modules: neurons and weights between them. Neuron would be composed of a simplified model of adder which gathers weights and nonlinear threshold function. If in the software implementations the weight is just a number that defines the state of a connection between two neurons, in the hardware implementations a weight itself is a circuit. It consists of a number that increases or decreases in the stage of learning depending on the state where the connection is made between neurons. The problem is not the complexity of circuit weight but the number of circuits in a Hopfield network. Thus if we have *n* neurons we have $n(n-1)/2$ circuits corresponding weight interconnections between them. Under these conditions we have analyzed the complexity of a network in terms of the number of weight circuits as it would be an index of hardware resources used and circuit complexity.

Experimental results are shown in the Table II.

TABLE II. EXPERIMENTAL RESULTS:

Alphabet	Neurons/network	Cir. number	Weight used	Capacity
32	16	6	720	34.56
32	23	4	1012	33.12
64	19	10	1710	68.4

First, we wanted to implement a 32-letter alphabet. By running the application, we noticed that the minimum size in bytes of code words (together with their inverse) is 16-bit and have Hamming distance 7 between them. To the implementation of a 16-bit Hopfield network we have used a storage capacity of 2.88 words, and for a bipolar encoding scheme this capacity is doubled to 5.76. This means that for 32 words which want to memorize we need 5.55 Hopfield networks. So, six Hopfield network can provide a storage capacity of 34.56 words code, coverings what we need. The second line in the table indicates a situation in which we tried to use a smaller number of circuit, but with a greater number of neurons, each for the same type of coding.

Thus, a Hopfield network with 23 neurons can store approximately 4 x 2 code words. So we need 4 circuits working in parallel, reaching an actual storage capacity of 33.12 words of code. It is interesting that, in terms of complexity (number of weight circuits), this solution is less optimized than the first. This is because the number of weight circuits increases with increasing number of neurons.

Finally, we implemented an alphabet which has 64 letters (messages on 6 bits). Here, according to the simulation, it results that we need to generate 19-bit code words with Hamming distance at least 7. A Hopfield network with 19 neurons can store about 7 words of code, so we have a total of 10 parallel networks which have a total capacity of 68.4 words of code. In Table III is displayed the performance of communication. It can be seen that the last experiment generates a superior performance even than cyclic error-correcting codes.

TABLE III. COMPARATIVE RESULTS:

Method	Parameters (n, k) d	Vt
CRC 15,5	(15,5) 7	0.33
CRC 17,6	(17,6) 6	0.35
Hopfield networks 6x16	(16,5) 7	0.31
Hopfield networks 4x23	(23,5) 7	0.22
Hopfield networks 10x19	(19,6) 7	0.32

V. CONCLUSIONS

Hardware associative memories are used in applications that can perform object recognition (for their classification) in an image, reconstruction and restoration of images or the identification of patterns of sounds.

We decided to use the associative memories to detect and correct errors in a message.

In order to design and test our system for the detection of 4 erroneous bits and for 3-bit error correction, we followed steps: first, we realized a software for generating *n*/2 words of code and their inverse; second, we built a Hopfield network having *d* neurons; third, we calculated how many parallel networks are needed to store all code words; fourth, we computed the necessary circuit complexity; finally, we implemented a network with a 64 letters alphabet, tested it and analyzed the obtained results.

REFERENCES

[1] R. Rojas "*Neural networks. A systematic introduction*", Springer Verlag, Berlin, 1996

[2] D. Popescu, C. Amza, D. Lăptoiu, G. Amza, "*Competitive Hopfield neural network model for evaluating pedicle screw placement accuracy*", Strojniski Vestnik/Journal of Mechanical Engineering, Volume 58, Issue 9, 2012, Pages 509-516

[3] G. Stefan, R. Benea, "*Connex memories & rewriting systems*", Proceedings of the Mediterranean Electrotechnical Conference – MELECON, Volume 2, 1998, Pages 1299-1303

[4] L. Ionescu, C. Anton, I. Tutanescu, A. Mazare, G. Serban, "*Reed Muller decoder with associative memories*", International Conference on Internet Technology and Secured Transactions, ISBN 978-1-908320-00-1, Abu Dhabi, UAE, 2011, pp. 274-277

[5] L. Ionescu, A. Mazare, G. Serban, V. Barbu, A. Constantin, "FPGA implementation of an associative content addressable memory", International Conference on Applied Electronics, AE 2011, Pilsen, 2011, Article number6049113, Pages 181-184

[6] M. Löwe, F. Vermet, "*The storage capacity of the Hopfield model and moderate deviations*", Elsevier, Statistics and Probability Letters, Volume 75, Issue 4, 15 December 2005, Pages 237–248

[7] C.V. Morais, F.M. Zimmer, S.G. Magalhaes, "*Inverse freezing in the Hopfield fermionic Ising spin glass with a transverse magnetic field*", Physics Letters, Section A: General, Atomic and Solid State Physics, Volume 375, Issue 3, 17 January 2011, Pages 689-697.