Twin Neurons for Efficient Real-World Data Distribution in Networks of Neural Cliques: Applications in Power Management in Electronic Circuits

Bartosz Boguslawski, *Student Member, IEEE*, Vincent Gripon, *Member, IEEE*, Fabrice Seguin, and Frédéric Heitzmann

***Abstract*— Associative memories are data structures that allow retrieval of previously stored messages given part of their content. They, thus, behave similarly to the human brain's memory that is capable, for instance, of retrieving the end of a song, given its beginning. Among different families of associative memories, sparse ones are known to provide the best efficiency (ratio of the number of bits stored to that of the bits used). Recently, a new family of sparse associative memories achieving almost optimal efficiency has been proposed. Their structure, relying on binary connections and neurons, induces a direct mapping between input messages and stored patterns. Nevertheless, it is well known that nonuniformity of the stored messages can lead to a dramatic decrease in performance. In this paper, we show the impact of nonuniformity on the performance of this recent model, and we exploit the structure of the model to improve its performance in practical applications, where data are not necessarily uniform. In order to approach the performance of networks with uniformly distributed messages presented in theoretical studies, twin neurons are introduced. To assess the adapted model, twin neurons are used with the real-world data to optimize power consumption of electronic circuits in practical test cases.**

***Index Terms*— Associative memory, clique, power management, real-world data, twin neurons.**

I. Introduction

A. Associative Memories Overview and Problem Statement

IN TRADITIONAL indexed memories, data are addressed using a known pointer. The principle of associative memories is different: data retrieval is accomplished presenting a part (possibly small) of it. Because of the partial input, the rest of the information is retrieved, and consequently no address is needed. As a toy example, retrieving the password of a user given its name in a database is typically a request of an associative memory. Associative memories are widely used in practical applications, for instance databases [1], intrusion detection [2], processing units' caches [3], or routers [4].

In order to assess the performance of associative memories, several parameters can be introduced. Probably the most important one [5] is termed memory efficiency and is defined as the best ratio of the total number of bits stored to the total number of bits used to store the memory itself, for a targeted performance. Note that this ratio is trivially one for indexed memories. Another important parameter is the computational complexity that depends on the operations needed to perform the retrieval. Again, this parameter usually makes no sense for the traditional memories in which computational complexity is often considered to be of $\mathcal{O}(1)$.

In practice, one can distinguish two main families of associative memories, namely, content-addressable memory (CAM) [6] and neuroinspired memories. A CAM compares the input search word against the stored data, and returns a list of one or more addresses where the matching data word is stored. CAMs combine memory efficiency with zero error rate and are often used in electronics, for example in network routers [6]. However, since it is a brute-force approach, the number of comparisons between the input search word and the stored data results in high complexity and energy consumption [7]. Moreover, CAMs assume that stored messages are couples, where only the second item can be erased in an input. Ternary CAM expands CAM functionality, allowing don't care bits matching both 0 and 1 values, thereby offering more flexibility. However, this comes at an additional cost, as the memory cells must encode three possible states instead of the two in the case of binary CAM. Consequently, the cost of parallel search within such a memory becomes even greater [8].

Neuroinspired associative memories combine lower complexity with higher flexibility, at the cost of reduced efficiency. In this category, Hopfield neural networks [9] is the most prominent model. In this model, stored messages are projected onto the connection weights of a complete graph. Nevertheless, when the size of this network is increased, the efficiency decreases. Sparse networks originally proposed in [10] use a small subset of connections to store each message, resulting in a much better efficiency [11]. Furthermore, the works

Manuscript received August 14, 2014; revised September 15, 2015; accepted September 17, 2015. Date of publication October 26, 2015; date of current version January 18, 2016. This work was supported in part by the European Research Council (ERC-AdG2011 290901 NEUCOD).

B. Boguslawski and F. Heitzmann are with Université Grenoble Alpes, Grenoble F-38000, France, and also with French Alternative Energies and Atomic Energy Commission (CEA)–Research Institute for Electronics and Information Technologies (LETI), Minatec Campus, Grenoble F-38054, France (e-mail: bartosz.boguslawski@cea.fr; frederic.heitzmann@cea.fr).

V. Gripon and F. Seguin are with the Electronics Department, TELECOM Bretagne, Brest 29238, France (e-mail: vincent.gripon@telecom-bretagne.eu; fabrice.seguin@telecom-bretagne.eu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2015.2480545

from [12] are also known to allow storing large number of messages.

Recently, Gripon and Berrou [13] proposed a new model that can actually be seen as a particular Willshaw network with cluster structure. This modification allows efficient retrieval algorithm without diminishing performance. This model is able to store a large number of messages and retrieve them, even when a significant part of the input is erased. The simulations of the network working as a data structure or an associative memory proved a huge gain in performance compared with Hopfield network [9] and Boltzmann machine [14] (when using comparable material) [13]. The fact that the network is able to retrieve messages with erasures on any position, or in the presence of noise, gives it an advantage over CAMs. These interesting properties originate in error correcting codes that underlie this network's principles [15].

The network as presented in [13] is analyzed only for uniform independent identically distributed values among all the messages. By the network construction, this means that the number of connections going out from each node is uniformly distributed across the whole network. It is well known that nonuniformity of messages to store can lead to a dramatic decrease in performance [16]. On the other hand, it is expected that real-world applications may contain highly correlated data. In order to approach the theoretical performances in real-world applications, the model needs to be improved.

B. Related Work

The impact of nonuniform data stored in the network introduced in [13] was analyzed in the work [17]. We exploit the structures of these networks to introduce several techniques in order to efficiently store the nonuniform data. First, the technique from [16] consisting in adding an additional layer to Willshaw network is adapted to the cluster structure of the network [13]. It is a simple method yet it requires a significant increase in the network size. The most efficient method introduced in [17] is based on Huffman coding. This approach exploits compression codes properties to minimize the impact of nonuniform data and gives very good results. However, it is an offline method where all the data have to be known in advance to compute the words of the code. It also requires additional operations related to coding and decoding.

C. Contributions

In this paper, a new technique that addresses the limitations of the formerly proposed methods is developed. The main contributions of this paper are the following.

1) The strategy based on Huffman coding proposed in [17] is modified to better suit real-world applications. It is shown to be much more efficient in a practical scenario compared with the best technique in [17].
2) A new online method is proposed in which coding/decoding phases are not necessary and there is no need to know the whole set of data in advance. This makes it much more flexible and suitable for real-world applications. Moreover, the new method offers better performances.
3) The new strategy is assessed in practical applications to manage power consumption of electronic circuits where real-world data from simulations and measurements is used. The results clearly indicate the interest of adapting the model proposed in [13].

This paper is organized as follows. Section II outlines the theory of the network. The problem of storing nonuniform data is explained in Section III. Section IV gives an overview of the techniques to store nonuniform data that were introduced in earlier work and proposes a new strategy improved for practical applications. Furthermore, it presents a theoretical analysis and an extensive comparison of the different techniques allowing storage of nonuniformly distributed data. In Section V, networks of neural cliques are applied in practical test cases to present the interest of adapting the formerly presented model to nonuniform data inherent in these applications.

II. Sparse Neural Networks With Large Learning Diversity

In this section, we introduce the family of sparse associative memories described in [13].

A. Message Definition

Throughout this paper, we consider that associative memories store messages that they are later capable of retrieving given a sufficiently large part of their content. In order to ease the readability of this document, and without loss of generality, we consider that a message consists of c submessages or segments. Each segment can be seen as a binary vector whose values range from 0 to $\ell-1$. An exemplary message, for $c=4$ and $\ell=4$, is $m=\{10\ 00\ 01\ 11\}$ (to be read 2 0 1 3).

B. Network Structure

In order to store messages, we use a network that consists of binary neurons and binary connections. Gripon and Berrou [13] use the term fanal (which means lantern or beacon) instead of neuron for two reasons: 1) at a given moment, in normal conditions, only one fanal within a group of them can be active and 2) for biological inspirations, fanals do not represent neurons but microcolumns [18]. Fig. 1 shows the general structure of the network and the notation. All the n fanals are organized in c disjoint groups called clusters. Fanals belonging to specific clusters are represented with different shapes. Each cluster groups $\ell = n/c$ fanals. A node in the network is identified by its index (i, j), where i corresponds to the cluster number and j to the number of the fanal inside the cluster. The connections are allowed only between fanals belonging to different clusters, i.e., the graph is multipartite. The connection between two fanals is denoted by a binary weight $w_{(i,j)(i',j')}$. Contrary to classical neural networks, the connections do not possess different weights, the connection exists or not. Hence, the weight (or adjacency) matrix of such a network consists of values $\{0, 1\}$ where 1 indicates the connection between two fanals, and 0 is the lack of the connection.

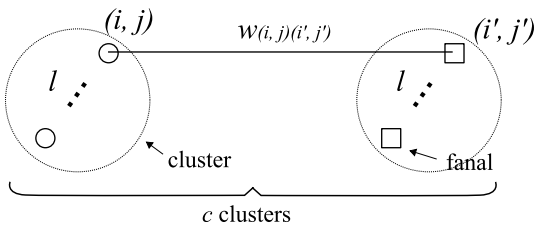

Fig. 1. Network general structure and notation. Different shapes (circles and squares) represent fanals belonging to different clusters.

C. Message Storing Procedure

To store a message m in the network, each of its c segments is associated with a distinct cluster, and more precisely with a unique fanal in its cluster (the one that index corresponds to the value of the segment). Then, this subset of fanals is fully interconnected forming a clique representing the message in the network. This term is also used in neurobiology to describe such groupings of neurons [19]. When a new message shares the same connection as an already stored message, this connection remains unchanged. Therefore, the result of the storage procedure is independent of the order in which messages are presented to the network. For more details about this storing procedure, refer to [13].

D. Message Retrieval Procedure

We call retrieval the process of retrieving a previously stored message when only part of its corresponding fanals is known. After the storing of all messages, the retrieval process is organized as follows. First, the known segments of the input message are used to stimulate appropriate fanals, i.e., the value on the given segment indicates which fanal should be chosen. These fanals are said to be active. After the initial stimulation, message passing phase comes next. The activated fanals send unitary signals to other clusters through all of their connections. Then, each of the fanals calculates the sum of the signals it received. Compared with [13], we take advantage of the improvement proposed in [20] that consists in summing only the signals from distinct clusters. This modification simplifies the hardware implementation as well [21]. Within each cluster, the fanal having the largest sum is chosen and its state becomes 1, i.e., it is made active. The other fanals inside the cluster represent the state equal to 0. The rule according to which the active fanal inside the cluster is elected is called winner takes all. The whole process may be iterative, allowing the fanals to exchange information with each other, such that ambiguous clusters (those containing more than one active fanal) will hopefully be correctly retrieved. When more than one iteration is needed (input with significant noise or erasures resulting in nonunique fanal with the largest sum) an extra value is added to the score of the last winners in the next iteration. More details on adjusting this memory effect are given in [13]. Because of this iterative retrieval procedure, the network converges step by step to the targeted previously stored message. In some cases though, it may happen that the output message is not correct, leading to nonzero error probability in the retrieval process.

As a result of the strong correlation brought by the connections of the clique embodying a message in the network, it is possible to retrieve the stored message based on partial or noisy information put into the network. In order to target the first applications [22], the network was implemented in hardware [23].

III. Nonuniformly Distributed Messages

A. Density and Error Probability Definition

As previously stated storing messages in the network corresponds to creating subgraphs of interconnected fanals. When the number of stored messages increases, these subgraphs share an increasing number of connections. Consequently, distinguishing between messages is more difficult. As a logical consequence, there is an upper bound for the number of distinct messages one can store then retrieve for a targeted maximum error probability. The network density d is defined as a ratio of the established connections to all the possible ones. Therefore, the density is a parameter of first importance to assess the network performance. A density close to 1 corresponds to an overloaded network. In this case, the network will not be able to retrieve stored messages correctly. For a network that stored M uniformly distributed messages expected density d is expressed by the following formula or its first-order approximation:

$$d = 1 - \left(1 - \frac{1}{\ell^2}\right)^M \approx \frac{M}{\ell^2} \quad \text{when } M \ll \ell^2. \tag{1}$$

The probability of correctly retrieving a message with c_e positions erased in a network constructed of c clusters is given by

$$P = (1 - d^{c-c_e})^{(\ell-1)c_e}. \tag{2}$$

This equation is valid for a single iteration. The probability of error increases with d, which is expressed by (1). Note that in the case of large density, iterations improve the ability of the network for retrieving messages correctly, what is confirmed by the evaluated simulations.

However, these equations only hold when input messages are drawn uniformly at random. Correlation between stored messages can lead to a dramatic decrease in the performance of such memories.

B. Nonuniform Distribution Example

Fig. 2 shows a case of a network made of four clusters. Fanals belonging to specific clusters are represented with different shapes. There are four fanals per cluster. The number of segments in the messages is four, each cluster corresponding to a given segment, and on each segment one of the four values is chosen. Cliques are formed by lines connecting fanals. Fig. 2 shows a situation where four messages are stored, each type of the line representing a different clique.

For a set of messages stored in the network, some fanals can have much more connections than the others. One can observe that in all of the clusters except cluster I, each node

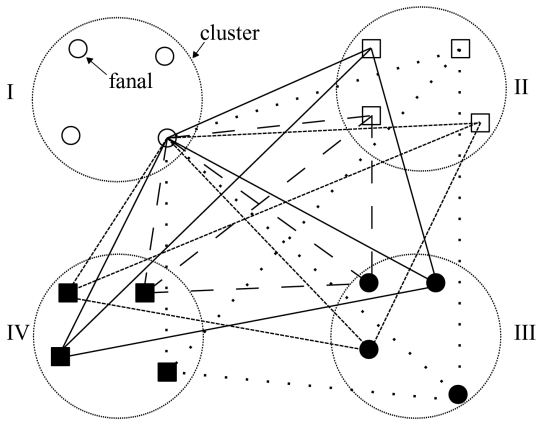

Fig. 2. Network with nonuniformly distributed messages. Different shapes (filled or empty circles or squares) represent fanals belonging to different clusters, and different types of lines represent connections belonging to distinct cliques. Clusters are numbered—they represent segments of messages.

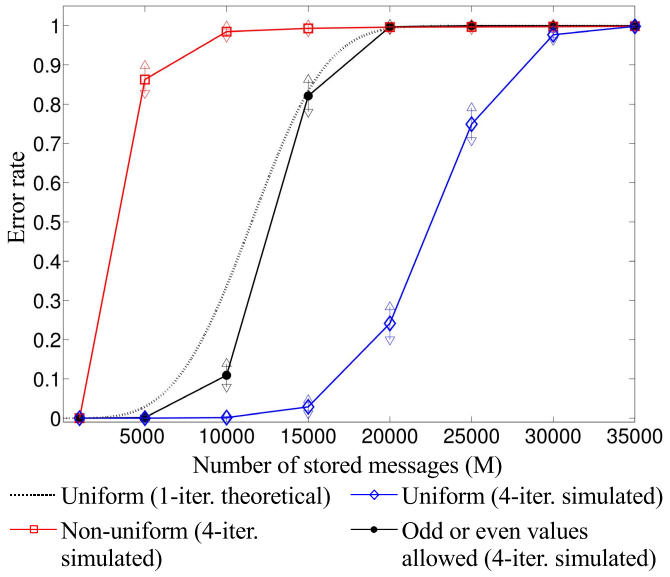

Fig. 3. Evolution of the error rate as a function of the number of stored messages for different types of data distributions. The network composed of eight clusters of size 256, and four randomly erased positions. For each point, 100 networks are evaluated doing 100 tests per network. The arrows indicate standard deviations of the error rates.

has the same number of connections. This means that on the segments II–IV of the messages each of the four possible values occurred. However, in the cluster I, only one of the fanals is always used, i.e., the value on the first segment is constant. This simple example depicts how the distribution of data stored in the network maps to the interconnection structure.

Fig. 3 shows the evolution of the error retrieval rate for a larger network with $c = 8$ and $\ell = 256$. In that specific case, half the clusters are not provided with information. This means that only four randomly chosen segments of a message are known, the remaining are erased. Hence, only four fanals are initially stimulated in four clusters. Fig. 3 shows also the theoretical curve for a single iteration and the network density. Note the interest of the iterative character of the decoding process. The simulation shows that the network of $n = 2048$ fanals can store up to 15 000 uniform messages of 64 bits each and retrieve them with a very high probability (error rate 0.029 for four iterations allowed) even when half the clusters are not provided with information. However, when the messages are generated from the truncated Gaussian distribution (mean $\mu = 135$ and standard deviation $\sigma = 25$), only 2000 messages can be stored (error rate 0.047). The truncated Gaussian distribution, contrary to the uniform one, implies that on a given segment of messages some values occur much more often than the others. Fig. 3 also shows a curve (indicated with a dot) for a data with a specific correlation between the values within each message. Within each message, either odd or even values are allowed. This means that if on the first segment there is an odd value, one knows that all the other values are odd (e.g., $m = \{1\ 3\ 5\ 7\ 9\ 11\ 13\ 15\}$ in decimal). For this data set, the network can store 8000 messages and retrieve them with the same error probability as in the uniform case. One can see that the correlation in the stored data clearly shifts the curve toward lowest number of stored messages. In this example, the same network filled with normally distributed data can store only 13% of the number of uniformly distributed messages.

As in most applications data cannot be considered uniform, it is of first importance to introduce techniques to counterbalance the effects of correlation on performance.

IV. Limiting Local Density for Storing Nonuniform Data

A. Using Compression Codes

In [17], compression codes (more precisely Huffman lossless compression coding [24]) were applied to store nonuniform data. The core idea relies on the fact that Huffman coding produces variable-length codewords—the values that occur most frequently are coded on a small number of bits, whereas less frequent values occupy more space. One dictionary is constructed for each segment of all the messages. Such procedure results in variable-length segments, the most often occurring values being the shortest. Therefore, the sizes of the frequent values that break the uniformity are minimized. The free space obtained within each segment is filled with random uniformly generated bits. Consequently, the most often appearing values are associated with the largest number of randomly chosen fanals, and the influence of local high-density areas is minimized. Decoding is possible because of the prefix-free property. This means that a set of bits (codeword) representing a value on a given segment is never a prefix of another codeword used for the same segment. For instance, a code consisting of $\{01, 11\}$ is a prefix-free code. Strictly speaking, in order to decode the retrieved encoded message, each segment is compared bit by bit with its dictionary. When a codeword is met, one knows that these bits are useful, the remaining being the random ones.

The baseline for the comparisons given in [17] is a strategy that consists in adding a number of random uniformly generated bits to each segment of a message. This simple method allows to spread a value across a group of fanals without the need to use a compression code and is added to the comparisons in this paper as well.

```
1: Input:
2: ℓ
3: ConnectionsLimit
4: Messages        \\ matrix; one message in each row, one segment in each column
5: SegVal          \\ value on the given segment
6: SegNum          \\ number of the given segment
7: AssocTab = [ ]  \\ empty cell matrix
8:
9: Output:
10: AssocTab       \\ cell matrix; number of row indicates the value in the message, number of column indicates the
11:                   number of segment. Each cell in the matrix holds the numbers of fanals associated to the given
12:                   value on the given segment
13:
14: procedure TWINFANALS(ConnectionsLimit)
15:     for each row in Messages do
16:         Store
17:         for each fanal used do
18:             if fan−out > ConnectionsLimit then        \\ fan − out - number of fanal's outgoing connections
19:                 if max(AssocTable(:, SegNum)) < ℓ then
20:                     AssocTab(SegVal, SegNum)(end + 1) = max(AssocTab(:, SegNum)) + 1
21:                 end if
22:             end if
23:         end for
24:     end for
25:     return AssocTab
26: end procedure
```

Fig. 4. Message storing procedure for twin fanals; pseudocode.

The results presented in [17] show that using compression codes is an effective solution for storing nonuniform data. Compression codes, however, require additional operations related to coding and decoding. Moreover, the whole set of messages has to be known to construct the words of the code.

B. Introducing Twin Neurons

In order to avoid the additional cost related to coding and decoding and provide an online and flexible approach where the whole set of messages does not have to be known, a new strategy to store nonuniform messages is developed here.

The principle of the method is to introduce twin neurons, i.e., changing the role of neurons in response to data distribution. If a fanal is overloaded (has high local density) another fanal will be designated to represent the same information. The pseudocode in Fig. 4 outlines the new algorithm which relies on modifying the message storing procedure. Before storing messages, the fanal's connection limit (ConnectionsLimit) is chosen. Furthermore, an association table (AssocTab) connecting each value on each segment with a specific fanal is necessary. Initially this table is empty. Alternatively, instead of the association table, fanals representing the same value can be interconnected. When a fanal is initially stimulated, it sends signals to all the other fanals in its cluster associated with the same value. Then, all the activated fanals start exchanging signals on the global (intercluster) level. Depending on the hardware/software implementation cost tradeoff or biological plausibility aspects any of the solutions can be chosen. In this paper, the association table is used.

After storing each message, the total number of outgoing connections of each fanal belonging to the newly created clique is controlled (line 18). If this number exceeds a formerly defined threshold, and there are unassociated fanals available in this cluster (line 19), a new fanal is associated with this value (line 20). This is similar to adding random bits that spread a given value over a number of fanals and results in limiting the local density. If there is no additional fanal left, the association table remains unchanged, and for each value, the last associated fanal is used. In order to retrieve a message based on partial input, all the fanals associated with the values on the known segments are stimulated.

As the result of this method, the network automatically adapts to the distribution of the stored data efficiently using the available material. Limiting local densities improves retrieval performance.

The method relying on Huffman coding is an algorithmic solution and, therefore, not very satisfactory in terms of biological plausibility. Storing messages according to the procedure described in this subsection exploits the fact that fanals that are close to each other receive inputs that are relatively close and become clones of each other called twins. Increasing the spatial diversity of the stored information allows reducing the density of the most stressed parts of the network.

C. Theoretical Analysis

For the presented curves, the improved retrieval procedure is used where only the signals from distinct clusters are included in the sum calculated by each fanal.

Based on the probability of correctly retrieving a message (2) one can obtain the probability of error after one iteration of the retrieval procedure as

$$P_e = 1 - P. \tag{3}$$

The error probability in the case of nonuniform data is analyzed with the following nonuniform distribution. On the first segment, the value one is chosen with probability p, any other value is chosen with the same probability that equals $1 - p/\ell - 1$. The rest of the $c-1$ segments are drawn uniformly. To simplify the discussion, this distribution is called first segment anomaly (FSA).

As a consequence, the density between any two clusters associated with the last $c-1$ segments is unchanged. Between the first cluster and any other one, the density is

$$d' = 1 - \left(1 - \frac{1-p}{\ell(\ell-1)}\right)^M \tag{4}$$

for any value on the first segment that is not one, and

$$d'_1 = 1 - \left(1 - \frac{p}{\ell}\right)^M \tag{5}$$

for the first segment equal one.

The error probability changes depending on whether the first segment equals one (to simplify the discussion, we call this case π) or is different from one ($\overline{\pi}$) and whether the first segment is erased (ϵ) or not erased ($\overline{\epsilon}$). In the following part of this subsection, all the theoretical error probabilities are expressed with appropriate equations.

In case $\pi\epsilon$, the error probability after one iteration of the retrieval procedure is

$$P_{e,\pi\epsilon} = 1 - (1 - d'^{c-c_e})^{\ell-1} \cdot (1 - d^{c-c_e})^{(c_e-1)(\ell-1)}. \tag{6}$$

In case $\pi\overline{\epsilon}$

$$P_{e,\pi\overline{\epsilon}} = 1 - \left(1 - d'_1 \cdot d^{c-c_e-1}\right)^{c_e(\ell-1)}. \tag{7}$$

In case $\overline{\pi}\epsilon$

$$P_{e,\overline{\pi}\epsilon} = (1-d'^{c-c_e})^{\ell-2} \cdot (1-d^{c-c_e})^{(c_e-1)(\ell-1)} \cdot (1-d'^{c-c_e}). \tag{8}$$

In case $\overline{\pi\epsilon}$

$$P_{e,\overline{\pi\epsilon}} = 1 - (1 - d' \cdot d^{c-c_e-1})^{c_e(\ell-1)}. \tag{9}$$

When using one of the techniques described in this paper (compression codes or twin fanals), the fanal corresponding to value one in the first cluster is duplicated when the density of its connections is large enough. Ideally, the density becomes constant and thus equals $d' < d$. The number of fanals $n_1$associated with value one in the first cluster is

$$n_1 = \frac{\log\left(1 - \frac{p}{\ell}\right)}{\log\left(1 - \frac{1-p}{\ell(\ell-1)}\right)}. \tag{10}$$

As a result, in case π, $P_{e,\pi\epsilon}$ is unchanged and $P_{e,\pi\overline{\epsilon}}$ becomes

$$P'_{e,\pi\overline{\epsilon}} = 1 - (1 - (1 - (1 - d')^{n_1}) \cdot d^{c-c_e-1})^{c_e(\ell-1)}. \tag{11}$$

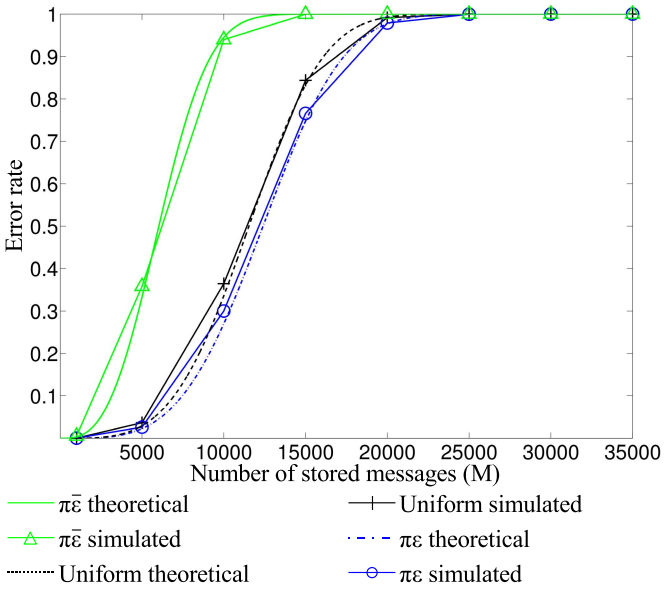

Fig. 5. Error rate for FSA and uniform distributions, with network from [13] used. The first segment equals one (π). Four positions are randomly erased—results after one iteration. For each point, 100 networks are evaluated, doing 100 tests per network.

In case $\overline{\pi}$, $P_{e,\overline{\pi\epsilon}}$ is unchanged and $P_{e,\overline{\pi}\epsilon}$ becomes

$$P'_{e,\overline{\pi}\epsilon} = 1 - (1 - d^{c-c_e})^{(c_e-1)(\ell-1)} \cdot (1 - d'^{c-c_e})^{n_1+\ell-2}. \tag{12}$$

In this section, the theoretical equations are verified with simulations. The parameters chosen for the analysis are $c = 8$, $c_e = 4$, $\ell = 256$, and $p = 0.5$. For the given parameters and distribution, $n_1 = 255$. Fig. 5 shows the cases $\pi\epsilon$ and $\pi\overline{\epsilon}$. Both (6) and (7) and the simulation results are plotted. One can see that the simulations correlate well with the theoretical analysis. In addition, the theoretical and simulated curves for uniformly distributed messages are given for comparison.

Fig. 6 shows the curve for (11) for compression codes or twin fanals. As expected in this case, the performance after one iteration is the same regardless of whether compression codes or twin fanals are used or not. The additional curve for four iterations shows the performance improvement.

Fig. 7 shows the cases $\overline{\pi}\epsilon$ and $\overline{\pi\epsilon}$. Equations (8) and (9) correlate well with the simulation results. Equation (12) and the corresponding simulations are shown in Fig. 8. One can notice the significant performance improvement.

D. Performance Comparison

The intrinsic characteristic of Huffman coding is the variable codeword's length. This parameter depends on the size and distribution of the data set. It is possible that some of the codewords exceed the available cluster size ℓ. In this case, in [17] no random bits are added to this segment and the remaining bits are pushed to the next segment. Moreover, after adding random bits, the bits from all the segments are shuffled in such a way that even and odd bits from each of the segments are stored in separate clusters. This means that in each cluster

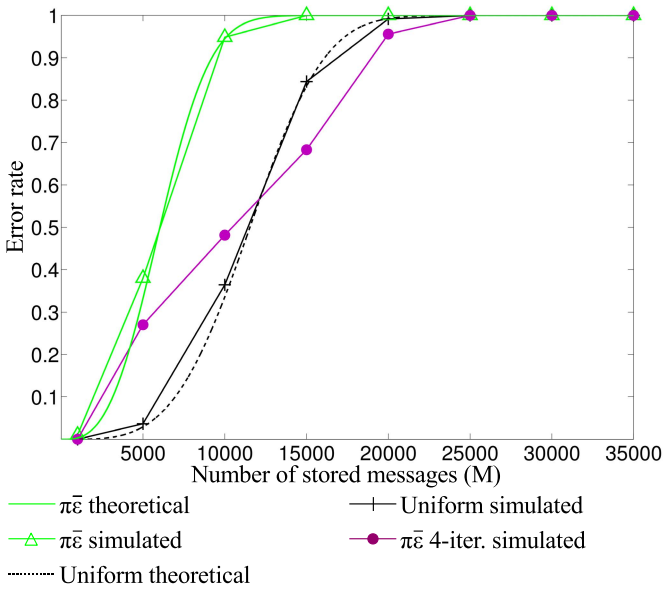

Fig. 6. Error rate for FSA and uniform distributions, with compression codes or twin fanals used for FSA. The first segment equals one (π). Four positions are randomly erased—results after one iteration unless otherwise stated. For each point, 100 networks are evaluated, doing 100 tests per network.

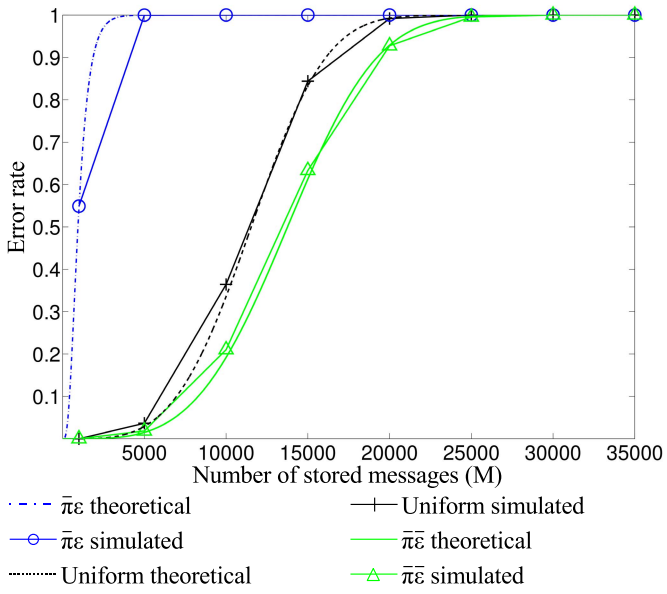

Fig. 7. Error rate for FSA and uniform distributions, with network from [13] used. The first segment is different from one ($\overline{\pi}$). Four positions are randomly erased—results after one iteration. For each point, 100 networks are evaluated doing 100 tests per network.

the chosen fanal depends not only on the value on the same segment in the initial message but also on the other segments. This implies that erasures are allowed only on the messages after the coding. As pointed out in [17], this characteristic makes this technique an effective solution for applications in which one can afford transforming data before storing it in the network. To simplify the discussion, in this paper, this technique is called Huffman shuffle.

As a consequence of the abovementioned property, each time the known segments of a message are used to stimulate

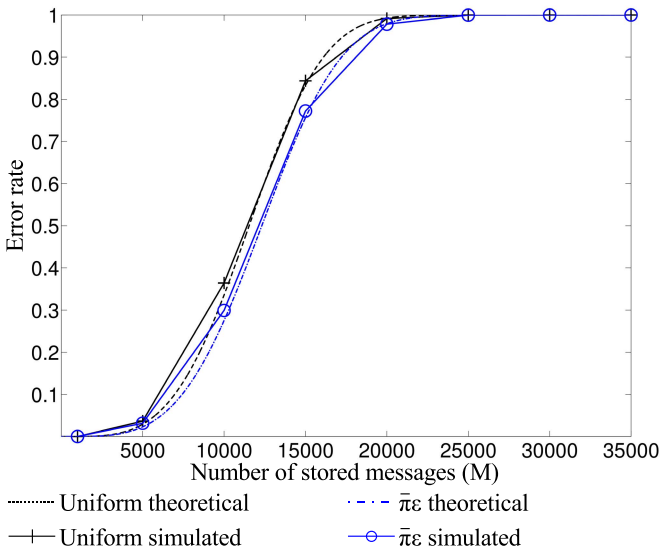

Fig. 8. Error rate for FSA and uniform distributions, with compression codes or twin fanals used for FSA. The first segment is different from one ($\overline{\pi}$). Four positions are randomly erased—results after one iteration. For each point, 100 networks are evaluated, doing 100 tests per network.

the network, only one fanal in each cluster is selected. In case of the strategy that relies on twin fanals, erasures are possible on the initial messages. This is important in terms of practical applications. Nevertheless, when retrieving messages, one does not know which fanal was used to create the clique representing the concerned message and all the fanals associated with the known segments are stimulated. If a unique fanal is stimulated in each cluster, the problem solved by the network is much easier, and one can expect better performances.

To compare the techniques, two cases can be considered: 1) messages are transformed before storing them in the network and 2) messages are not transformed before storing them in the network. In Case 1, the messages are modified such that they contain the number of fanal that was associated with a specific value in each message. Consequently, in all the strategies only a unique fanal is stimulated in each cluster. In Case 2, the association table is used to find the fanals to stimulate when a message is presented to the network. In this scenario, all the fanals associated with a value present in the message are stimulated. The first case, where erasures are allowed only on the transformed messages, can be applied to a very limited set of applications. Since the focus of this paper is on applications and real-world data, the techniques are compared in the second scenario. If Huffman shuffle is applied in the second scenario, after all the described processing steps (adding random bits, pushing bits to the next segment, shuffling bits) one fanal can be associated with many different values. To eliminate the impact of the interdependencies between different segments and to adapt Huffman shuffle to Case 2, a new strategy is added to the comparison. The bits that exceed the available space are now simply cut instead of being pushed to the next segment. Furthermore, there are no bit-shuffle operations. This technique is called Huffman simple. To provide a fair comparison, the nonuniform distribution is chosen so that the amount of codewords that

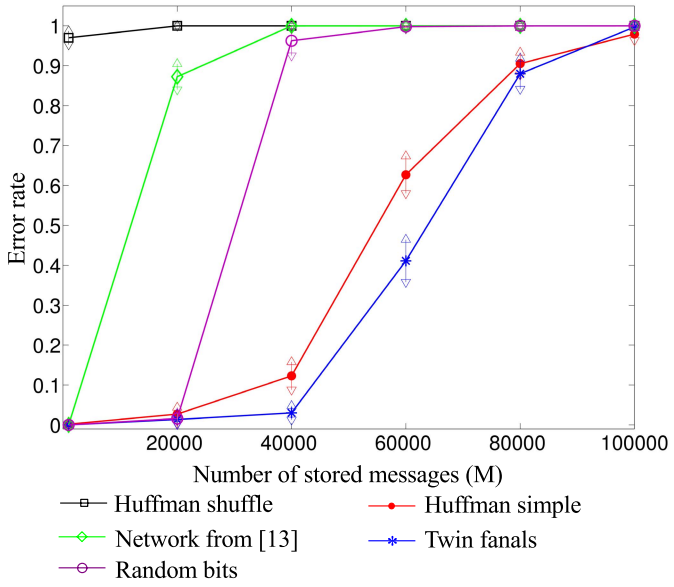

Fig. 9. Performance comparison for different proposed strategies. Four positions are randomly erased. For each point, 100 networks are evaluated doing 100 tests per network. Multiple fanals are stimulated in each cluster. Arrows: standard deviations of the error rates.

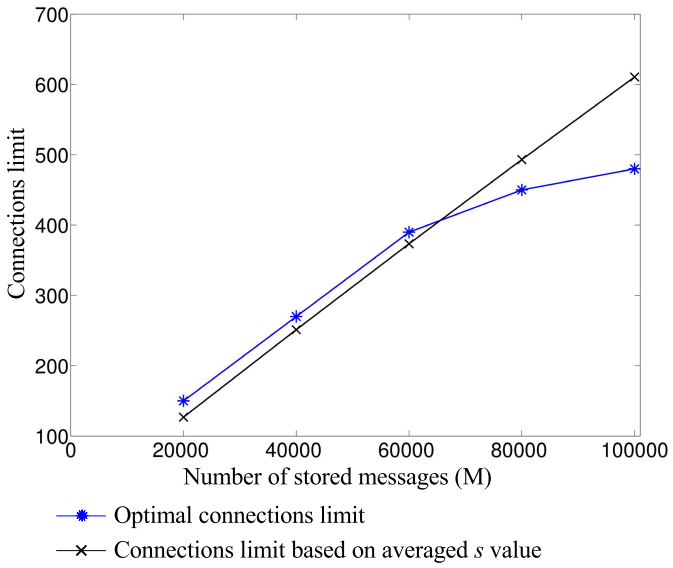

Fig. 10. Optimal and predicted connections limit values in function of the number of stored messages. For low numbers of stored messages, any connection limit can be chosen.

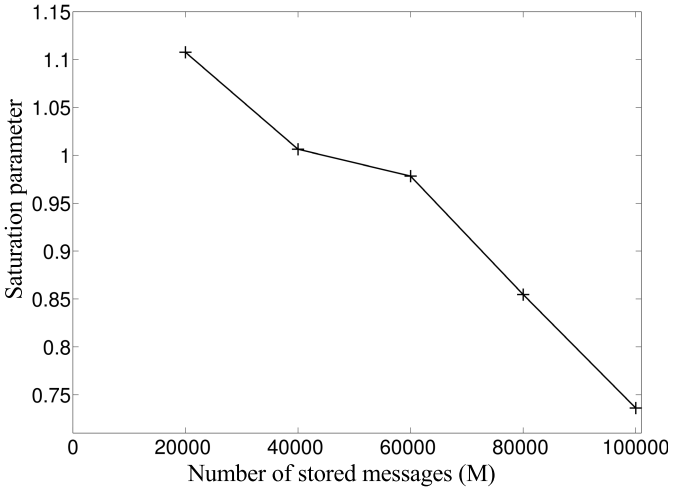

Fig. 11. Saturation parameter used to predict connections limit value in function of the number of stored messages. For low numbers of stored messages, any connections limit can be chosen.

exceed the space available in the cluster ($\ell = 1024$, 10 bits, the same size as in [17]) is negligible (<1%). This occurs for the truncated Gaussian distribution with $\mu = 135$, $\sigma = 65$.

Fig. 9 shows the results when multiple fanals can be stimulated in each cluster. First, note the performance of the network [13] in case of the nonuniform distribution ($\ell = 256$, the same as in Fig. 3). Then, several techniques are applied to the same network as in [17] ($\ell = 1024$). Random bits allow storing more messages; however, they are outperformed by the strategies proposed in this paper. Twin fanals present the best performance, Huffman simple being slightly less effective. As stated earlier, Huffman shuffle is not adapted to stimulating multiple fanals. In fact, it reaches retrieval error rate equal to 0.97 when only 1000 messages are stored.

Fig. 10 shows the connections limit values for each point of the curve for twin fanals. Each of these values was obtained by varying the connections limit and choosing the one that gave the lowest error rate. The connections limit parameter (ConnectionsLimit) can be connected to density d for a uniform distribution (1) and expressed as saturation s

$$s = \frac{\text{ConnectionsLimit}}{(c-1)\ell d} \tag{13}$$

$(c-1)\ell$ gives the maximal number of connections per fanal. By weighting this value with d, one obtains the expected number of fanal's connections. Dividing ConnectionsLimit by this number gives the saturation of the network.

Based on the density, which is a function of the number of stored messages M, the approximate optimal value of the connections limit can be predicted. Fig. 11 shows how the saturation depends on the number of stored messages based on the optimal connections limit values. Fig. 10 shows also the connections limit values obtained from an averaged s value as $\overline{s}(c-1)\ell d$. The proximity of the optimal and predicted connections limit values in the range where the network is not overloaded shows that the technique based on twin fanals is also easy to adjust to the stored data.

According to the obtained results, the strategy that relies on twin fanals presents the best performance. Moreover, it allows to avoid the coding/decoding overhead, which is important in practical applications, especially in the ones presented in the following section, where the system's time response and low complexity are crucial.

E. Influence of Distribution's Standard Deviation

As shown in the preceding sections, the distribution of the data stored in the network influences the retrieval performance. Here, the standard deviation of the nonuniform data distribution is varied to show what is the impact on the performance

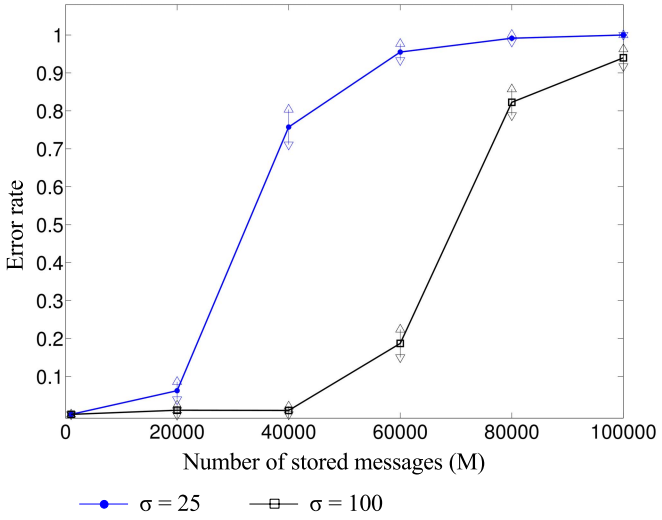

Fig. 12. Error rate when using twin fanals for different values of σ for nonuniform distribution. Four positions are randomly erased. For each point, 100 networks are evaluated, doing 100 tests per network. Arrows: standard deviations of the error rates.

of the network when using twin fanals. The experiments are done for truncated Gaussian distributions with different standard deviations. In addition to the distribution chosen in Section IV-D with $\sigma = 65$, a distribution with $\sigma = 25$ and $\sigma = 100$ is used. Fig. 12 shows how the retrieval process is impacted when the stored data is more nonuniform ($\sigma = 25$) and more uniform ($\sigma = 100$) than the previously chosen one. When comparing these curves with the results in Fig. 9, one can see how the retrieval performance depends on the considered distribution's parameter.

V. Dynamic Power Management Applications for Networks of Neural Cliques

In this section, networks of neural cliques are applied in dynamic power management applications. Because of these applications, the networks are assessed in a practical context using real-world data. It is shown that with such data, standard networks become inoperative, and one needs to definitely adapt to the model.

A. Dynamic Power Management in Multiprocessor System-on-Chip

Multiprocessor systems-on-chip (MPSoCs) have gained a lot of importance in recent years. Because of their distributed and scalable architecture, they offer high performance required in real-time applications with potential power savings allowing fulfilling energy restrictions under battery operation. An MPSoC is built of multiple processing elements (PEs) that can work in parallel (Fig. 13). Each PE or set of PEs form a voltage/frequency island (VFI), i.e., they work within the same power domain. The supply voltage V_{dd} and frequency f are set by dedicated switching circuits allowing dynamic voltage and frequency scaling. By decreasing the speed of the PEs with lower performance requirements, the energy consumption is

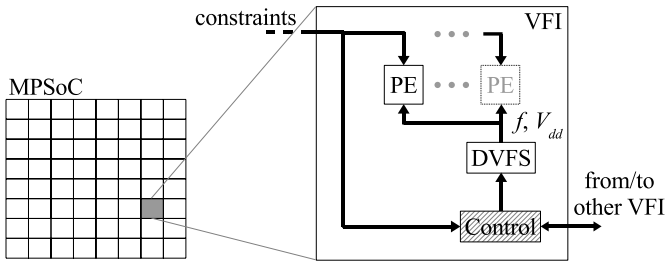

Fig. 13. MPSoC with power management capability.

reduced. A control unit decides on (V_{dd}, f), or power modes, based on workload, latency, and temperature, among others.

Among the proposed control units, one can mention approaches that rely on game theory [25], hybrid subgradient and consensus method [26], low-level hardware controllers [27], [28], PID automation controllers [29], and operating system control [30]. Networks of neural cliques were first proposed as a power controller in [22]. Compared with the state-of-the-art technique, they react 4500 times faster and consume 6800 times less energy. An extension of this paper [31] shows that because of the fast time response, the energy consumed by the MPSoC can be significantly reduced.

The impact of stored data distribution is not considered in the abovementioned works. In this section, the adapted networks of neural cliques are used as the power management control unit.

1) Energy Model and Optimization Formulation: The derivation of the energy model used for optimizing the energy in MPSoCs is described in detail in [31]. The whole energy needed by VFI_i working with the clock period T_i belonging to a finite set is given by

$$E_i(T_i) = \left[N_i + \gamma_i\left(\frac{L}{T_i} - N_i\right)\right]\frac{E_{\text{nom},i}T_{\text{nom},i}^2}{T_i^2} + LP_{\text{stat},i} \tag{14}$$

where N_i is the number of clock cycles needed to finish the computation, L is the time when the application is running, $E_{\text{nom},i}$ is the nominal energy consumed during nominal clock period $T_{\text{nom},i}$, and $P_{\text{stat},i}$ is the static power. γ_i is used to express the energy reduction per clock cycle in low-activity state and is equal to

$$\gamma_i = \frac{E_{L_{\text{dyn},i}}/(L - X_i)}{E_{H_{\text{dyn},i}}/X_i} \tag{15}$$

where $E_{H_{\text{dyn},i}}$ is the energy consumed in high-activity state, and $E_{L_{\text{dyn},i}}$ is the energy consumed in low-activity state.

Given the latency constraint L and the number n of VFIs, the optimization problem is

$$\begin{aligned} &\min \sum_{i=1}^{n} E_i(T_i) \\ &\text{s.t.} \sum_{i=1}^{n}(X_i = N_i T_i) \le L. \end{aligned} \tag{16}$$

TABLE I
LTE RECEIVER APPLICATION PARAMETERS [26]

	TRX OFDM	Channel Estim.	Coef. Interpol.	MIMO Decoding	RX Bit	Channel Decoding
N_i (clk cycles)						
Mode 1	39200	91296	1668	12210	6445	$16{\cdot}10^3$
Mode 2			3336	24420	12890	$32{\cdot}10^3$
Mode 3			3336	24420	7855	$32{\cdot}10^3$
Mode 4			16680	122100	39275	$160{\cdot}10^3$
Mode 5			16680	122100	34665	$264{\cdot}10^3$
$E_{nom,i}$ (pJ/clk cycle)	47.67	180.55	198.94	198	93.84	247.74

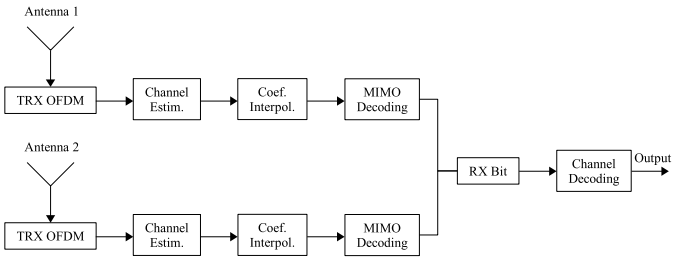

Fig. 14. LTE receiver application graph. The memory buffers between the processing blocks are not presented.

Thus, the objective of the optimization is to minimize the total energy consumption of the system, considering local clock periods as state variables, and preserving the time constraint. In our energy model for a given latency constraint L, global static power has a constant contribution. Therefore, it is discarded in the optimization process.

2) Test Case: In order to obtain data to store in the network, an applicative test case on MAGALI telecom chip is considered [32]. MAGALI is a semiheterogeneous MPSoC, i.e., there is a number of different PEs, but each is reproduced several times. The available resources communicate through a network-on-chip. Each PE can work with one of the 256 frequencies between 20 and 790 MHz [33]. The application mapped on the platform is the Long-Term Evolution (LTE) receiver. It consists of six tasks: 1) TRX OFDM; 2) channel estimation; 3) coefficient interpolation; 4) MIMO decoding; 5) RX bit; and 6) channel decoding (Fig. 14). The parameters of the application, necessary to calculate the energy (14), were characterized at $f_{\text{nom},i} = 400$ MHz and are summarized in Table I. The operating modes presented in the table differ in reachable data rates [32].

Since there are 256 frequencies available, each PE is associated with a cluster of 256 fanals. The frequencies that are applied to the PEs depend on the latency constraint L and the operating mode. There is, therefore, a cluster of five fanals for modes and a cluster for L. L ranges from 0.7 to 2 ms. Fifty one latency values are considered in this application and consequently one cluster of 51 fanals is created. The impact of the L cluster's size on the power management efficiency is analyzed in [31]. Each time the system needs to be reconfigured, the L cluster and mode cluster are stimulated and the set of frequencies is retrieved from the network.

The results of the evaluated experiments are summarized in Table II. First, to test the network, a random uniformly distributed set of frequencies is stored. The network is simulated for each of the possible L values. The simulations show that in this case each of the stored frequencies is retrieved correctly.

Then, the actual values of the frequencies are stored in the network and the same test procedure is applied. This time, the error rate (the amount of messages in which at least one frequency was not retrieved correctly) clearly increases. As a consequence of the incorrectly retrieved frequencies, there are 87 out of 255 L constraints that are violated. Note that there may be more incorrectly retrieved frequencies that not necessarily need to lead to L constraint violation. Nevertheless, they result in nonoptimal solutions which imply an excessive energy consumption.

In order to approach the optimal retrieval process of random uniform frequencies, twin fanals introduced in Section IV-B are applied. The structure of the network remains the same, the number of fanals does not change, and only the message storing procedure is modified. The connection limit is set to 6. In this case, the error rate is reduced, and there are eight constraint violations. The connection limit can be varied to find an optimal value. The minimal number of constraint violations was obtained for the connection limit equal to 10. The error rate and the number of constraint violations are slightly reduced.

B. Dynamic Variability Management

As the technology is scaled down and the supply voltage is reduced, the intradie variability becomes an increasing challenge. The fabrication process becomes less reliable and causes static process (P) variations. Dynamic parameters as voltage (V) and temperature (T) variations also have an increasing impact on the timing and power consumption. The optimal operating point set by a power controller can be shifted by dynamic voltage drops and temperature variation. A solution to avoid worst case design is to partition the system to a number of parts, allowing each to work at its best operating point. Based on the applicative constraints and thanks to the sensors that provide the information regarding current working conditions (V, T), each part adjusts again its operating point to reduce the power consumption or to preserve the functionality.

In order to provide the adjustment policy with current (V, T) approximations, specific sensors have to be embedded in the system. In this paper, a low-cost digital sensor multiprobe [34] is considered. Multiprobe is made of seven ring oscillators (ROs) that are designed to have different sensitivities to Process-Voltage-Temperature (PVT) variations. These sensors embedded in different parts of the circuit experience the same static and dynamic variations as the parts they are placed close to. Therefore, after a calibration phase, based on the frequencies of the ROs, current approximations ($\hat{V}$, $\hat{T}$) of (V, T) can be found. Frequency of an RO depends in a complex way on the PVT parameters and it is impossible to obtain ($\hat{V}$, $\hat{T}$) from a single frequency [35]. Since there are no models available, solving a set of equations in case of several ROs is mathematically infeasible.

TABLE II
MPSoC POWER MANAGEMENT RESULTS

Data type	Network type	Error rate	No. constraint violations
Uniform random	Standard	0	-
Real	Standard	0.71	87
Real	Twin fanals, conn. lim. = 6	0.07	8
Real	Twin fanals, conn. lim. = 10	0.03	3

TABLE III
VARIABILITY MANAGEMENT RESULTS

Data type	Network type	Error rate	MAE_V	MAE_T	σ_V	σ_T
Uniform random	Standard	0.2	10.7	10.64	54.9	29.12
Real	Standard	0.54	4.3	24.1	9	37.62
Real	Twin fanals, conn. lim. = 3	0.49	3.6	22.95	7.9	37.21
Real	Twin fanals, conn. lim. = 3, increased size	0.24	1	2.03	3.2	5.44

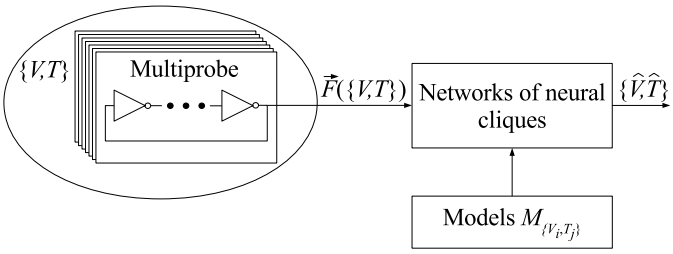

Fig. 15. Overview of the variability management system.

An estimation method to obtain $(\hat{V}, \hat{T})$ from the frequencies of multiprobe is proposed in [35]. The proposed method relies on statistical tests of goodness-of-fit. First, the data from the sensors are specially adapted to the further treatment. Then, the statistical tests are evaluated, a brute-force search in the database obtained during the calibration phase is performed and a weighted mean of the closest found entries is calculated. This method provides satisfactory estimation precision, yet implies certain complexity. Moreover, the estimation computed by a dedicated hardware block can be obtained in 25 μs which is not fast enough to follow fast voltage variations (order of magnitude of few microseconds). Here, applying networks of neural cliques to provide $(\hat{V}, \hat{T})$ pairs corresponding to frequencies of the ROs is explored (Fig. 15). The focus is on the impact of the data distribution. Nevertheless, published results [22], [23], [31] show that the time response of the hardware implementation of the networks is in the order of tens of nanoseconds.

In order to obtain data to store in the network, electrical simulations of the multiprobe sensor have been performed with an Eldo circuit simulator. The message set was obtained for V ranging from 0.7 to 1.3 V with a step of $\Delta V = 10$ mV and T ranging from -40 °C to 120 °C with a step of $\Delta T = 10$ °C. In this paper, only two ROs of the multiprobe are used to retrieve $(\hat{V}, \hat{T})$ couples. The simulations show that the impact of reducing the number of used frequencies is negligible on the accuracy of the retrieved $(\hat{V}, \hat{T})$. Among the two kept ROs, one is specifically designed to be more sensitive to temperature variations than the other. Due to the characteristics of the ROs, in the database containing 1037 models linking the frequencies with $(\hat{V}, \hat{T})$ points, a number of models share the same frequencies leading to different $(\hat{V}, \hat{T})$ couples. In such an ambiguous case, both $(\hat{V}, \hat{T})$ solutions are equitable and are output by the network. In a target system, the network can be followed by an additional processing step, which calculates a mean value of the obtained $(\hat{V}, \hat{T})$ responses. Since the focus of this paper is on exploring the impact of data distribution, these models are removed from the database. After this step, 1018 models are stored in the network.

The ROs can output 256 and 512 (temperature-sensitive RO) different frequency values. Therefore, they are associated with two clusters of 256 and 512 fanals, respectively. In addition, there is a cluster to store T values made of 17 fanals and a cluster of 61 fanals to store V values. Each time the values $(\hat{V}, \hat{T})$ are demanded by the variability management system, the frequency clusters are stimulated and $(\hat{V}, \hat{T})$ couple is obtained from the network.

The results are assessed in terms of the error rate [the amount of messages in which at least one of the $(\hat{V}, \hat{T})$ parameters is not retrieved correctly] and mean absolute errors

$$\mathrm{MAE}_V = \frac{1}{n}\sum_{i=1}^{n} |f_{V,i} - y_{V,i}| \tag{17}$$

$$\mathrm{MAE}_T = \frac{1}{n}\sum_{i=1}^{n} |f_{T,i} - y_{T,i}| \tag{18}$$

where n is the number of stored models, f is the response from the network. and y is the true value. In addition, the standard deviations associated with these errors are reported.

All the obtained results are summarized in Table III. First, a random uniformly distributed set of frequencies is stored. The network is simulated for all the stored models. When the actual values of the frequencies are stored in the network and the same test is applied, the error rate increases significantly. The mean absolute error and standard deviation increased in the case of temperature. In the case of voltage, the error and standard deviation decreased, which together with the error rate means that there are more errors but they are smaller.

In this section, twin fanals are applied. The network's dimensions stay the same. As a consequence, the error rate is slightly reduced and the error and standard deviation of V are lower than in the former experiments. The temperature

retrieval is slightly improved as well compared with real data stored in nonadapted network. In order to improve the retrieval significantly, the network's dimensions are changed. When T cluster is increased to 500 fanals, the results are further improved. For both the networks in which twin neurons were used, the connection limit was set to 3. This value proved to be the optimal one.

VI. Conclusion

Networks of neural cliques are associative memories able to store and successfully retrieve a large number of messages. Their performances have been theoretically studied exploiting uniformly distributed information. When facing real-world applications, the information that is stored in the networks is not necessarily uniformly distributed. Analyzing this model in case of nonuniform data and adapting it to practical applications is a subject of work [17]. The solution based on Huffman compression coding offers great performance enhancements and allows efficient storage of nonuniform messages. Nevertheless, in some applications its complexity can be a limiting factor. Furthermore, Huffman coding is an algorithmic solution and, therefore, not satisfactory in terms of biological plausibility.

In this paper, a method with a limited complexity and a comparable performance is proposed. Introducing twin fanals avoids complex coding and decoding phases and is biologically plausible. It allows erasures on initial messages without transforming the data before the storage. Furthermore, there is no constraint in terms of data distribution parameters as it is the case with Huffman coding. The results presented in this paper show that because of spreading frequent values among a group of fanals, twin fanals offer better performances than Huffman coding-based technique.

Two test cases in power management domain are exploited to obtain real-world data. When the networks presented in [13] are applied to these applications, their performance is largely degraded leading to nonfunctional system. In order to approach the theoretical performance of the model, twin fanals are used when real-world data are stored. The results show that this solution approaches the performances close to uniformly distributed data. Adapting the basic model is indispensable for using these networks in practical applications.

References

[1] C. S. Lin, D. C. P. Smith, and J. M. Smith, "The design of a rotating associative memory for relational database applications," *ACM Trans. Database Syst.*, vol. 1, no. 1, pp. 53–65, Mar. 1976.

[2] A. Papadogiannakis, M. Polychronakis, and E. P. Markatos, "Improving the accuracy of network intrusion detection systems under load using selective packet discarding," in *Proc. 3rd Eur. Workshop Syst. Secur.*, Paris, France, Apr. 2010, pp. 15–21.

[3] N. P. Jouppi, "Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers," in *Proc. 17th Annu. Int. Symp. Comput. Archit.*, Seattle, WA, USA, May 1990, pp. 364–373.

[4] N.-F. Huang, W.-E. Chen, J.-Y. Luo, and J.-M. Chen, "Design of multi-field IPv6 packet classifiers using ternary CAMs," in *Proc. IEEE Global Commun. Conf.*, San Antonio, TX, USA, Nov. 2001, pp. 1877–1881.

[5] V. Gripon and M. Rabbat, "Maximum likelihood associative memories," in *Proc. IEEE Inf. Theory Workshop*, Seville, Spain, Sep. 2013, pp. 1–5.

[6] K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (CAM) circuits and architectures: A tutorial and survey," *IEEE J. Solid-State Circuits*, vol. 41, no. 3, pp. 712–727, Mar. 2006.

[7] H. Jarollahi, V. Gripon, N. Onizawa, and W. J. Gross, "A low-power content-addressable memory based on clustered-sparse networks," in *Proc. 24th IEEE Int. Conf. Appl.-Specific Syst., Archit. Processors*, Washington, DC, USA, Jun. 2013, pp. 305–308.

[8] B. Agrawal and T. Sherwood, "Modeling TCAM power for next generation network devices," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw.*, Austin, TX, USA, Mar. 2006, pp. 120–129.

[9] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci. USA*, vol. 79, no. 8, pp. 2554–2558, 1982.

[10] D. J. Willshaw, O. P. Buneman, and H. C. Longuet-Higgins, "Non-holographic associative memory," *Nature*, vol. 222, no. 7, pp. 960–962, Jun. 1969.

[11] G. Palm, "Neural associative memories and sparse coding," *Neural Netw.*, vol. 37, no. 1, pp. 165–171, Jan. 2013.

[12] A. H. Salavati and A. Karbasi, "Multi-level error-resilient neural networks," in *Proc. IEEE Int. Symp. Inf. Theory*, Cambridge, MA, USA, Jul. 2012, pp. 1064–1068.

[13] V. Gripon and C. Berrou, "Sparse neural networks with large learning diversity," *IEEE Trans. Neural Netw.*, vol. 22, no. 7, pp. 1087–1096, Jul. 2011.

[14] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for Boltzmann machines," *Cognit. Sci.*, vol. 9, no. 1, pp. 147–169, Jan./Mar. 1985.

[15] V. Gripon and C. Berrou, "A simple and efficient way to store many messages using neural cliques," in *Proc. IEEE Symp. Comput. Intell., Cognit. Algorithms, Mind, Brain*, Paris, France, Apr. 2011, pp. 54–58.

[16] A. Knoblauch, G. Palm, and F. T. Sommer, "Memory capacities for synaptic and structural plasticity," *Neural Comput.*, vol. 22, no. 2, pp. 289–341, Feb. 2010.

[17] B. Boguslawski, V. Gripon, F. Seguin, and F. Heitzmann, "Huffman coding for storing non-uniformly distributed messages in networks of neural cliques," in *Proc. 28th Conf. Artif. Intell.*, Quebec City, QC, Canada, Jul. 2014, pp. 262–268.

[18] B. K. Aliabadi, C. Berrou, V. Gripon, and X. Jiang, "Storing sparse messages in networks of neural cliques," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 5, pp. 980–989, May 2014.

[19] L. Lin, R. Osan, and J. Z. Tsien, "Organizing principles of real-time memory encoding: Neural clique assemblies and universal neural codes," *Trends Neurosci.*, vol. 29, no. 1, pp. 48–57, Jan. 2006.

[20] V. Gripon and C. Berrou, "Nearly-optimal associative memories based on distributed constant weight codes," in *Proc. IEEE Inf. Theory Appl. Workshop*, San Diego, CA, USA, Feb. 2012, pp. 269–273.

[21] H. Jarollahi, N. Onizawa, V. Gripon, and W. J. Gross, "Reduced-complexity binary-weight-coded associative memories," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Vancouver, BC, Canada, May 2013, pp. 2523–2527.

[22] B. Larras, B. Boguslawski, C. Lahuec, M. Arzel, F. Seguin, and F. Heitzmann, "Analog encoded neural network for power management in MPSoC," in *Proc. 11th IEEE Int. New Circuits Syst. Conf.*, Paris, France, Jun. 2013, pp. 1–4.

[23] B. Larras, C. Lahuec, M. Arzel, and F. Seguin, "Analog implementation of encoded neural networks," in *Proc. IEEE Int. Symp. Circuits Syst.*, Beijing, China, May 2013, pp. 1612–1615.

[24] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proc. Inst. Radio Eng.*, vol. 40, no. 9, pp. 1098–1101, Sep. 1952.

[25] D. Puschini, F. Clermidy, P. Benoit, G. Sassatelli, and L. Torres, "Dynamic and distributed frequency assignment for energy and latency constrained MP-SoC," in *Proc. Conf. Design, Autom. Test Eur.*, Nice, France, Apr. 2009, pp. 1564–1567.

[26] I. Mansouri, F. Clermidy, P. Benoit, and L. Torres, "A run-time distributed cooperative approach to optimize power consumption in MPSoCs," in *Proc. IEEE Int. SOC Conf.*, Las Vegas, NV, USA, Sep. 2010, pp. 25–30.

[27] D. N. Truong *et al.*, "A 167-processor computational platform in 65 nm CMOS," *IEEE J. Solid-State Circuits*, vol. 44, no. 4, pp. 1130–1144, Apr. 2009.

[28] W. Kim, M. S. Gupta, G.-Y. Wei, and D. Brooks, "System level analysis of fast, per-core DVFS using on-chip switching regulators," in *Proc. IEEE 14th Int. Symp. High Perform. Comput. Archit.*, Salt Lake City, UT, USA, Feb. 2008, pp. 123–134.

[29] G. M. Almeida *et al.*, "PI and PID regulation approaches for performance-constrained adaptive multiprocessor system-on-chip," *IEEE Embedded Syst. Lett.*, vol. 3, no. 3, pp. 77–80, Sep. 2011.
[30] G. M. Almeida *et al.*, "Predictive dynamic frequency scaling for multi-processor systems-on-chip," in *Proc. IEEE Int. Symp. Circuits Syst.*, Rio de Janeiro, Brazil, May 2011, pp. 1500–1503.
[31] B. Larras, B. Boguslawski, C. Lahuec, M. Arzel, F. Seguin, and F. Heitzmann, "Analog encoded neural network for power management in MPSoC" *Analog Integr. Circuits Signal Process.*, vol. 81, no. 3, pp. 595–605, Dec. 2014.
[32] F. Clermidy, R. Lemaire, X. Popon, D. Kténas, and Y. Thonnart, "An open and reconfigurable platform for 4G telecommunication: Concepts and application," in *Proc. Euromicro Conf. Digit. Syst. Design, Archit., Methods Tools*, Patras, Greece, Aug. 2009, pp. 449–456.
[33] F. Clermidy *et al.*, "A 477 mW NoC-based digital baseband for MIMO 4G SDR," in *Proc. IEEE Int. Solid-State Circuits Conf.*, San Francisco, CA, USA, Feb. 2010, pp. 278–279.
[34] L. Vincent, E. Beigné, L. Alacoque, S. Lesecq, C. Bour, and P. Maurine, "A fully integrated 32 nm multiprobe for dynamic PVT measurements within complex digital SoC," in *Proc. 2nd Eur. Workshop CMOS Variability*, Grenoble, France, May 2011, pp. 1–4.
[35] L. Vincent, E. Beigné, S. Lesecq, J. Mottin, D. Coriat, and P. Maurine, "Dynamic variability monitoring using statistical tests for energy efficient adaptive architectures," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 6, pp. 1741–1754, Jun. 2014.

Bartosz Boguslawski (S'14) received the B.S. and M.S. degrees in electronics and telecommunications from the AGH University of Science and Technology, Kraków, Poland, in 2011 and 2012, respectively. He is currently pursuing the Ph.D. degree with CEA–LETI, Grenoble, France, and TELECOM Bretagne, Brest, France, directed by Prof. C. Berrou. His master's thesis was involved in the field of inverse problems for heat transfer in microelectronics with Ghent University, Ghent, Belgium.

He was an Exchange Student with Ghent University. His current research interests include power management, multiprocessor system-on-chip, and neural networks.

Mr. Boguslawski is a co-recipient of the Second Best Student Paper Award at the 11th IEEE NEWCAS Conference, Paris, France.

Vincent Gripon (M'10) received the Ph.D. degree from TELECOM Bretagne, Brest, France, in 2011, under the supervision of Prof. C. Berrou.

He is currently a Permanent Researcher and specializes in computer and information sciences. His work mainly focuses on connecting information theory and error correcting codes with neural networks. His main contribution is a new family of sparse associative memories based on distributed codes that provide almost optimal efficiency. He is also the creator and organizer of a programming contest named TaupIC, which targets French top undergraduate students.

Fabrice Seguin was born in Talence, France, in 1973. He received the Ph.D. degree from the University of Bordeaux 1, Talence, in 2001.

His Ph.D. research concerned the current mode design of high-speed current-conveyors and applications in RF circuits. In 2002, he joined the Electronic Engineering Department, TELECOM Bretagne, Brest, France, as a full-time Lecturer. He is currently involved in design issues of analogue channel decoders and related topics, energy harvesting, and neural networks.

Frédéric Heitzmann received the M.S. degree from Ecole Polytechnique, Palaiseau, France, and the M.S. degree from Telecom ParisTech, Paris, France, with a specialization in computer science and networks.

His current research interests include hardware–software co-design, and compilers for heterogeneous and homogenous MPSoC.