Microelectronics Journal 45 (2014) 1450–1462

Contents lists available at ScienceDirect

Microelectronics Journal

journal homepage: www.elsevier.com/locate/mejo

Spiking neuro-fuzzy clustering system and its memristor crossbar based implementation

Mohammad Bavandpour [a,*], Saeed Bagheri-Shouraki [a], Hamid Soleimani [b], Arash Ahmadi [b], Bernabé Linares-Barranco [c]

[a] Artificial Intelligence Lab (ACL), Department of Electrical Engineering, Sharif University of Technology, Tehran 11365-11155, Iran
[b] Department of Electrical Engineering, Razi University of Kermanshah, Kermanshah 67149-64346, Iran
[c] Instituto de Microelectrónica de Sevilla, IMSE-CNM (CSIC and Universidad de Sevilla), Av. Américo Vespucio s/n, 41092 Sevilla, Spain

ARTICLE INFO

Article history:
Received 27 December 2013
Received in revised form
22 August 2014
Accepted 2 September 2014
Available online 20 September 2014

Keywords:
Neuro-fuzzy clustering system
Spike encoding scheme
Memristor
Simon Lucas dataset
Fisher's Iris dataset
Variability analysis

ABSTRACT

This study proposes a spiking neuro-fuzzy clustering system based on a novel spike encoding scheme and a compatible learning algorithm. In this system, we utilize an analog to binary encoding scheme that properly maps the concept of "*distance*" in multi-dimensional analog spaces to the concept of "*dissimilarity*" of binary bits in the equivalent binary spaces. When this scheme is combined with a novel binary to spike encoding scheme and a proper learning algorithm is applied, a powerful clustering algorithm is produced. This algorithm creates flexible fuzzy clusters in its analog input space and modifies their shapes to different convex shapes during the learning process. This system has plausible biological support due to its spike-based learning mechanism, its Quasi Spike Time Dependent Plasticity learning policy and its brain-like fuzzy clustering performance. Moreover, this neuro-fuzzy system is fully implementable on the hybrid memristor-crossbar/CMOS platform. The resultant circuit was simulated on one clustering task carried out in the binary input space on the Simon Lucas handwritten dataset and another clustering task carried out in the analog input space on Fisher's Iris standard dataset. The results show that it attained a higher clustering rate in comparison with other algorithms such as the Self Organizing Map, *K*-mean and the Spiking Radial Basis Function. The circuit was also successfully simulated on an image segmentation task and some clustering tasks performed in noisy spaces with various cluster sizes. Furthermore, the circuit variability analysis shows that device and signal variations up to 20% had no significant impact on the circuit's clustering performance, so the system is sufficiently immune to different variations due to its fuzzy nature.

1. Introduction

The main goal of research in the field of Artificial Intelligence (AI) is to obtain an artificial system with the far-ranging capacities and high performance of the human brain. Research activities can be broken down into several subfields such as Artificial Neural Networks (ANN) and fuzzy logic focusing on the brain's different specifications and capabilities. The concept of fuzzy logic was introduced by Zadeh [1] to represent and manipulate data and information possessing with non-statistical uncertainties. From an arithmetical viewpoint, fuzzy logic is a multi-valued logic that makes it possible to define intermediate values between crisp values like yes/no, true/false, black/white, etc. In other words, linguistic terms are given indistinct boundaries by introducing gradual membership functions. In contrast to the classical set theory, in which an object or a case either is or is not a member of a given set, fuzzy logic makes it possible for an object or a case to belong to a set only to a certain degree.

Fuzzy logic, applied in combination with other theoretical and practical concepts, has resulted in robust systems such as fuzzy controllers [2–4], neuro-fuzzy networks [5,6], fuzzy decision makers [7], fuzzy arithmetic algorithms [8], etc. While fuzzy rules generally take the form of easily understood *if… then* statements, it is sometimes difficult to understand how neural network rules actually perform tasks. For this reason, fuzzy logic and neural networks are considered to be complementary areas of research [9].

One of the basic tasks which the human brain performs proficiently is environmental object clustering. In a clustering task, the network must categorize input samples into appropriate clusters. Clustering applications are problematic mostly because of the abstruse spreading of samples and the varying boundary

Abbreviations: SOM, Self Organizing Map; RBF, Radial Basis Function; IP, Integrated Potential; IIP, Integral of Integrated Potential; ZLL, Zero Logic Line; OLL, One Logic Line; PE, Pair Encoding; SE, Step Encoding; QSTDP, Quasi Spike Time Dependent Plasticity

* Corresponding author.
E-mail address: mbavandpour388@gmail.com (M. Bavandpour).

http://dx.doi.org/10.1016/j.mejo.2014.09.001
0026-2692/

shapes of the clusters. The complexity and limitations of the previously presented mechanisms are largely attributable to the lack of an effective way to define cluster boundaries. With higher-dimensionality inputs, this problem becomes even more serious.

In neural networks, synapses are the connections between neurons by which signals, known as spikes, are exchanged. Synaptic plasticity is the capability of synapses to tune their transmission strengths, known as weights, to learn a specific function. Some biological experiments [10] support a weight change policy called Spike Time Dependent Plasticity (STDP). Memristor, as two-terminal, nano-scale device possessing a combination of resistive and memory characteristics, is one of the most promising devices to be used as synapse for hardware realization of neural networks. Accordingly, the analog values of synaptic weights are stored in the memristors, and may vary through STDP. Although this so-called missing fourth circuit element was first theoretically predicted in 1971 [12], the Hewlett-Packard (HP) memristor [11] was the first acknowledged physical realization of this thin-film device using titanium dioxide. Potential applications of the memristor include resistive memories (RRAM) [13], programmable analog circuits, variable gain amplifiers and adaptive filters [14], digital logic [15], synapses in Spiking Neural Networks (SNN) [16–24], fuzzy systems [25] and bio-inspired neuron models [26].

The memristor's memory property makes it a suitable device for neuromorphic engineers and AI researchers for use in bio-inspired neural networks with specific applications [19–22]. Compared to the other options, such as capacitor-based memory cells, the memristor can retain its memristance forever with no input voltage or supplied current. Furthermore, its nano-scale size, low power consumption and fabrication process compatible with conventional CMOS [27] make it an excellent choice. At the architectural level, a crossbar-based structure appears to be the most promising nanotechnology architecture [28]. The major advantages of this architecture are its inherent defect-tolerance capability, its simplicity, its flexibility, its scalability and its ability to provide maximum density.

In this paper, we propose a novel neuro-fuzzy approach, where each neuron creates a flexible convex membership function for its specific cluster in the input space. Input samples are applied to the neurons using spikes with a certain encoding scheme. The "*winner neuron*", which displays the highest degree of membership, is trained through modifying its membership functions. Thanks to the adaptive membership functions used in this approach, complex clustering tasks can be performed. In the proposed structure, computing and memory units are integrated with each other in a similar way to how they are integrated in the brain. This structure is easily implementable on the memristor-crossbar/CMOS platform thanks to its spiking nature and its compatible network structure, spike encoding schemes and learning algorithm. In this scheme, we used a Quasi STDP (QSTDP) learning algorithm, which has been used before in robust approaches [21,22], and the compatible signal shapes proposed by Querlioz et al. [21].

One of the most significant spike-based computational systems is the Spiking Radial Basis Function (Spiking RBF) [29], a system that has achieved a high clustering performance in comparison with conventional neural networks, but is very hard to implement because of its various embedded delay units, its complex learning policy, and its complex data to spike encoder. A powerful neuro-fuzzy approach, fully implementable on memristor-crossbar/CMOS, was described in Ref. [25]. This approach applied an analog input current for computation, causing high sensitivity to variability and noise. Other troublesome issues are how to produce input currents proportional to the membership degrees and how to implement the learning mechanism to change the state of the memristors. One solid study in this direction proposed a memristive neural network [21]. This approach ignored the similarity between zero bits, and used a variable-threshold neuron with a homeostatic-type mechanism, which is difficult to implement. Moreover, it was not capable of working in multi-dimensional analog spaces. Another worthwhile study into spike-based computational systems proposed a classifier on the event-based digital video input [22]. This work proposed a dual Phase-Change Memory (2-PCM) synapse [22] for a memristor-based implementation of the classifier. The 2-PCM synapse can properly mimic synaptic behavior and apply the learning policy, but it has some drawbacks: firstly, it uses 2 memristors and 2 CMOS gates to implement one synapse, which occupies a relatively large area. Secondly, it cannot be implemented on a memristor-only crossbar structure. And thirdly, it requires an ongoing refresh process due to memristance changes in one direction. Another scheme for implementing a memristive synapse, called Bridge Synapse [17], suffered from the first two disadvantages mentioned above and also used external equipment for off-chip learning, thereby increasing the area consumption due to the computer-connecting auxiliary circuits and external computer. Taking into account all the defects of the other, similar works mentioned above, we have tried to develop a high performance, easily implementable, low cost approach that is robust to noise and variations and uses a stand-alone learning mechanism without the need for external equipment.

The paper is organized as follows: Section 2 discusses the encoding scheme in binary space and the learning mechanism, proposes the binary to spike Pair Encoding (PE) scheme and examines its benefits in comparison with a single spike encoding scheme. This section is the basis of the proposed neuro-fuzzy algorithm introduced in the next section. Section 3 illustrates the prerequisite conditions for the analog to binary encoding scheme, studies these conditions on three possible schemes, and proposes the analog to binary Step Encoding (SE) scheme as the most satisfactory choice. This section then provides a neuro-fuzzy interpretation of the final scheme. The memristor crossbar-based circuit of the system proposed in the previous sections is introduced in Section 4. In Section 5 we simulate the circuits step-by-step with standard case studies such as the Simon Lucas handwritten dataset (binary input space), Fisher's Iris dataset (analog input space), hierarchical and noisy data samples, and an image segmentation task, and compare the clustering performance of the network with other approaches. We also look at the circuit's immunity to device and signal variations.

2. Binary encoding scheme and learning mechanism

Among the significant factors that determine the performance, implementation difficulty and complexity of memristor-based spiking neural circuits are their spike coding schemes, their learning algorithms and their spike shapes. In our competitive network approach, the winner neuron is decided by the similarity between the input vector and the weights vector. At circuit level, the weighted sum of the inputs, called the integrated potential, represents the similarity concept. To consider both the similarity between zeros and the similarity between ones in a physical implementation, each bit should be represented by two input lines. This means that each weight includes two sub weights, and two memristors are therefore needed to implement each input weight. The learning algorithm should increase this similarity in a winner neuron by increasing the weight of the winner neuron's fired inputs and decreasing the weight of its non-fired inputs. This algorithm could be implemented by a special signal shape. As shown in Fig. 1, this scheme maps the degree of similarity to the membrane potential of the neurons. It means that the membrane

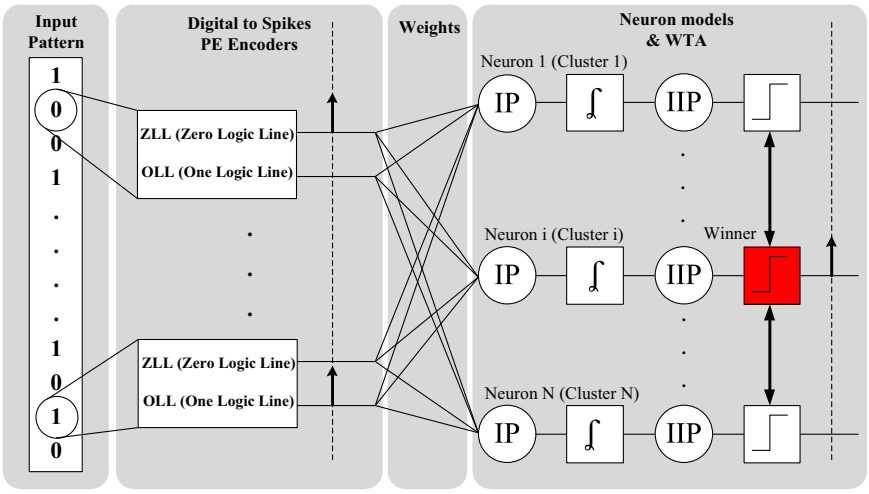

Fig. 1. Network structure of the clustering system using proposed PE encoding scheme. PE comprises two lines (OLL and ZLL) to represent one bit. The neurons receive Integrated Potential (*IP*), evaluate the Integral of Integrated Potential (*IIP*), compare it with the threshold value, and the winner neuron (here the *i*'th neuron) fires and stops other neurons from firing.

potential of the winner neuron reaches the predetermined threshold before the other neurons: thus, the winner neuron fires a spike and the other neurons remain in the resting potential.

In this scheme, the weights are positive and fall within the boundary values W_{min} and W_{max} $(0 < W_{min} < W < W_{max})$. The coding scheme uses two input lines, denominated OLL (One Logic Line) and ZLL (Zero Logic Line), to present one input bit to the circuit. As Fig. 1 shows, logical value "1" is represented by a spike on the OLL line when the ZLL line is subject to a rest potential. Conversely, logical value "0" is represented by a spike on the ZLL line when the OLL line is subject to a rest potential. This coding can be used in both supervised and unsupervised learning algorithms. Here, we concentrate on unsupervised learning, which plays a major role in the biological brain's learning mechanism. From another point of view, the learning algorithm can be interpreted considering two basic rules:

- **RULE1:** *IF* the input bit is "1" *THEN* neuron is winner.
- **RULE2:** *IF* the input bit is "0" *THEN* neuron is winner.

The OLL and ZLL line weights therefore represent the certainty of RULE1 and RULE2 respectively. Here we distinguish the uncertainty concept (decreasing the rule weight to the minimum value and increasing the opposite rule weight to the maximum value) from the lack of knowledge concept (keeping both rule weights at an initial middle value) by setting a constant threshold T_0 for all the neurons and resetting initial weights to W_0 $(W_{min} < W_0 < W_{max})$. In the learning process, the higher the certainty of RULE1, the lower the certainty of RULE2, and vice versa.

As shown in Fig. 1, the analog integrated potential (*IP*) values of the neurons are compared by applying an integral function over time to the integrated potential of the neurons and comparing the output values (*IIP*) with a constant threshold,

$$IIP_j(t) = \int_0^t IP_j(t)dt = \int_0^t \sum_i W_{ij} I_i(t)dt \quad (1)$$

where j is the neuron index and i is the input index. Thus, the winner neuron is the neuron with the greater integrated potential or with an integrated potential the integral of which exceeds the threshold voltage faster than the other neurons. Clearly, it is considerably easier to compare the integral of the analog values with a constant value over time than to compare a number of analog values at the same time. The winner neuron fires and the other neurons remain at the rest potential. One of the benefits of this scheme is that output spike encoding can easily be converted to input spike encoding and applied to the next layer in multilayer neural networks.

The STDP is the biological weight change policy which relates the time difference between a pre-synaptic and a post-synaptic spike to the synapse conductance changes or the weight changes. As shown in Fig. 2, this relationship involves two exponential curves. The punishment curve decreases the weight of a given input when the neuron fires before the input fires, whereas the reward curve increases its weight when the neuron fires after the input fires. In this paper, the learning algorithm is a Quasi STDP algorithm (QSTDP) [21], which uses two points of STDP curves to change the weights. In this algorithm, if the input fires and causes the output to fire, the connection weight increases and if the input does not fire when the output fires, the output firing is not relevant to the input and the connection weight decreases. This learning algorithm is compatible with the proposed circuit and is easy to implement. Weight change policy is given by

$$W_{ij}(k+1) = W_{ij}(k) + I_i \cdot \delta_P - (1 - I_i) \cdot \delta_N \quad (2)$$

where i is the index of input neurons in the input layer, j is the index of the winner and fired output neurons in the output layer, k is the iteration number, I_i is the input value of the i'th input neuron (0 or 1), δ_P is the positive weight change and δ_N is the negative weight change. δ_P and δ_N are set in accordance with preferred similarity and diversity in the same cluster patterns.

One of the advantages of this scheme is that it eliminates the initial weight value problem. This problem is a significant problem in neural networks which may influence network performance and convergence time or, in the worst case, prevent network convergence to the correct result [30].

3. Proposed analog to binary encoding scheme

In the previous section, a binary to spikes encoding scheme and a learning algorithm based on the similarity measurement between the input bits was introduced. In that scheme, we mapped the distance of the binary vectors to the dissimilarity of the binary bits. This process is more complicated in multi-dimensional analog input spaces. Applying analog numbers directly into the algorithm using one input wire is the first available solution, but it has some critical disadvantages. Firstly, it is difficult to implement data and learning processes on the

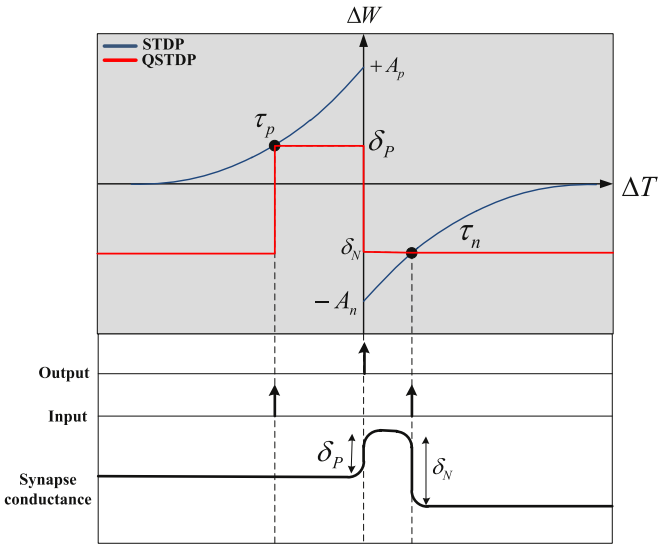

Fig. 2. STDP function as an exponential interpolation of biological samples, and the approximation curve (QSTDP). δ_P and δ_N are the reward and punishment values.

memristor-based circuit with analog numbers. It also causes high sensitivity to variability and noise. Secondly, the fuzzy approach of the clustering algorithm cannot be achieved and the algorithm turns into a conventional clustering algorithm. Thirdly, the resultant algorithm has low biological support considering the fact that spikes have the same amplitude and shape in SNNs. In this section, we seek a special scheme to encode the analog input numbers into the binary bits taking into account implementation capability, clustering performance, and biological plausibility. This analog to binary scheme can successfully be attached to the aforementioned PE-based binary clustering system to produce a clustering system in multi-dimensional analog spaces, if one essential condition is satisfied. It has to be able to map the "*distance*" concept in the analog space to the "*dissimilarity*" concept in the binary space correctly. This means that lower distance vectors must have more similarity in their binary representation. The implementation cost of the encoder must also be considered. For clarification, three different encoding schemes are examined here with regard to the above mentioned conditions and the proposed scheme is then introduced.

First, conventional analog to digital encoding (A/D), which encodes analog inputs to binary numbers. Consider, for example, three analog numbers, 15, 16 and 34, with the binary representation of 01111, 10000 and 11000 respectively. The pair formed by (15,16) have a lower distance in analog space compared with (16,34), but 15 and 16 have no similar bit in the binary representation, whereas the (16,34) pair are similar in 3 bits. So this encoding scheme, despite its comparatively low implementation cost, does not map the distance concept to the dissimilarity concept correctly, and it is not a suitable choice for our approach.

Second, interval encoding, which divides analog input range into a number of portions of equal lengths and allocates one bit to each portion. In this encoding scheme, if the analog input value falls within a given portion, the proportional bit is set to logic "1", otherwise it is reset to logic "0" (Similar to One-hot encoding). Consider, for example, three analog numbers falling within the first, second and last portions in a 5 bit encoder. These numbers are encoded into 00001, 00010 and 10000 respectively. Evidently, in this coding scheme each pair of presented numbers differ in two bits (the numbers have the same similarity) in the binary representation, while the distance between the numbers is not equal in the analog space. So this encoding scheme does not satisfy the requisite condition. Moreover, this encoder uses two comparators for each portion to check the upper and lower threshold conditions and a number of logical gates to produce the output bits. Its implementation cost is therefore relatively high.

Third, the Step Encoding (SE) scheme, which divides the analog input range into a number of portions of equal lengths by placing a predetermined number of indexes with equal distances on the input axis and allocating one bit to each index. If the analog input value is lower than the index, the proportional bit is set to logic "1" otherwise it is reset to logic "0". In other words, for each input number on the input axis, the upper hand indexes reset their bits to zero, and the lower hand indexes set their bits to one. The SE scheme equation can be derived as follows:

$$SE_i(x) = \begin{cases} 1 & x \geq (i \cdot P) \\ 0 & x < (i \cdot P) \end{cases}, \quad i = 1, 2, \ldots, (N = \lceil A/P \rceil) \tag{3}$$

where x is the analog input number, N is the number of indexes on the input axis, i is the index number which is an integer number from 1 to N, A is the maximum value of the analog input x and P is the length of the portions between two consecutive indexes. Consider, for example, a five bit encoder with the following parameters: $N=5$, $A=120$, $P=20$, $i=1$–5, as shown in Fig. 3. In this encoder, three analog values of $x=19$, $x=21$ and $x=95$ are encoded to 00000, 00001 and 01111 respectively. Evidently the pair formed by numbers (19,21), which have a lower distance in the analog space, have 4 similar bits in their binary representation and the pair formed by (21,95) which have a higher distance in analog space, have 2 similar bits in their binary representation. This encoding scheme can therefore correctly map the distance concept to the dissimilarity concept. Furthermore, this encoder uses just one comparator for each portion to check the index threshold condition, as shown in Fig. 3, so it has a lower implementation cost than the interval encoder. The SE scheme satisfies the baseline conditions as a suitable choice for our classifier. It should also be mentioned that the SE encoder encodes the analog input to binary bits and these output bits have to be encoded to spikes using two OLL and ZLL input lines, as explained in the previous section.

4. Fuzzy interpretation

One of the most important advantages of the SE scheme is its fuzzy nature resulting in a neuro-fuzzy clustering system if it combines with the aforementioned binary to spike PE encoding scheme and the introduced learning algorithm. Another aspect of the SE scheme is that neurons create fuzzy membership functions

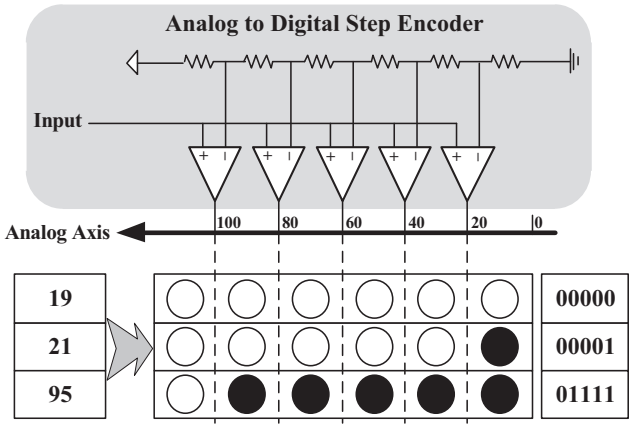

Fig. 3. 5-Bit step encoding scheme and its circuit level implementation.

of their clusters on each input axis and modify these membership functions to the optimized proper convex shapes during the learning process. When each input sample is applied to the network, each neuron evaluates the summation of the membership degrees obtained from its membership functions on the input axes. The winner neuron, which possesses the highest summation of membership degrees, then modifies its membership functions according to the learning algorithm. Here, a number of questions need to be addressed to clarify the relation between above-explained fuzzy interpretation and the introduced analog clustering algorithm. First, how can we obtain the membership functions? Second, how does the learning algorithm modify the membership functions? Third, why are the membership functions convex with flexible shapes?

To obtain the membership function of a given output neuron on one of its input axes, we need to evaluate the membership degree of the output neuron for all the indexes on the input axis. According to the SE analog to binary and PE binary to spikes encoding schemes, for a given input x in analog space, the upper hand ZLL input neurons and the lower hand OLL input neurons are fired and their weights contribute to the evaluation of the membership degree, while the other input neurons remain in the rest state and do not influence the membership degree. The membership degree equation can therefore be derived as follows:

$$MD_{jk}\left(X^{(k)}\right) = \sum_{i=1}^{i \le \left[X^{(k)}/P\right]} W_{ijk}^{OLL} + \sum_{i \ge \left[X^{(k)}/P\right]}^{i=N} W_{ijk}^{ZLL} \tag{4}$$

where i is the index number, j is the neuron number, k is the input axis number and $X^{(k)}$ is the k'th dimension of the X input vector. So W_{ijk}^{OLL} is the OLL input weight of the i'th index of the k'th input axis for the j'th output neuron. The total membership degree of the j'th neuron for a given input X is evaluated as follows:

$$TMD_j(X) = \sum_k MD_{jk}\left(X^{(k)}\right). \tag{5}$$

Fig. 4 shows an example of changing OLL and ZLL weights during the learning process for five input samples clustered by a given neuron, and the resultant normalized membership function. As can be seen in the figure, the initial membership functions have rectangular shapes, meaning that the clusters grant the same membership degree all over the input space. During the learning process, the clusters decrease their membership degree on the input space areas the lower the comparative density of the samples, and increase their membership degree on the input space areas the higher the comparative density of the samples. It is clear that if a given input sample creates more membership degree on the membership function, its encoded spikes will create more integrated potential on the neuron. Mathematical proof of the convexity of the membership functions is attached to the paper as Appendix A.

5. Memristor crossbar based circuit

To model the conductance increments and decrements of the memristive devices in our system simulations, we used the model introduced in [31]. This model can properly reproduce experimental memristive device measurements. An increase in conductance is modeled by the following equation:

$$\begin{cases} \delta G_p = \alpha_p e^{-\beta_p((G-G_{min})/(G_{max}-G_{min}))} \\ \delta G_m = \alpha_m e^{-\beta_m((G_{max}-G)/(G_{max}-G_{min}))} \end{cases} \tag{6}$$

where δG_p and δG_m are, respectively, the increment and decrement of the memristor's conductance. α_p, α_m, β_p and β_m are the model parameters, which are adjusted according to the memristor's behavior. These parameters, together with minimum and maximum conductance G_{min} and G_{max}, are subject to device variability in the real fabrication process.

One of the most important features of the memristor applied in the above model is the threshold condition. That is to say; small voltages across the nano-device, below its threshold (Vth), do not induce significant change in the memristance, while larger voltages induce much greater changes [32].

The proposed memristor crossbar-based neuro-fuzzy clustering circuit is shown in Fig. 5 and presynaptic and postsynaptic spike shapes are shown in Fig. 6. The circuit receives input analog values $X^{(k)}$ from the analog input layer, encodes them to the binary numbers using the SE encoders in the "analog to binary SE encoders unit", and then encodes the resultant binary values to presynaptic spikes using the PE encoders in the "binary to spikes PE encoders unit". Each horizontal wire in the crossbar is the joint of input memristors for one neuron and each pair of vertical wires is used to present one input bit to the network. Thus, each memristor is a variable synaptic weight with the value "$weight = R_f/R_{memristor}$". Here we used the Leaky Integrate and Fire (LIF) [33] neuron model to evaluate the IIP. The neuron included an input voltage summer and integrator, a threshold comparator, a spike generator, a resetting mechanism and a feedback for back propagating the output spike. When the input voltage exceeded V_{ref}, the output spike generator produced a spike with the shape of the postsynaptic spike shape shown in Fig. 6. This spike was back propagated to the joint terminal of the memristors through a virtual connection between the input pins of the operational amplifier (op-amp), and used for teaching input weights (memristors). The winner Take All (WTA) algorithm was applied to the neurons by propagating OR of the outputs to the reset pin of the neurons through a bi-stable circuit. The bi-stable CMOS circuit depicted in Fig. 5 kept the neurons in the reset state and stopped them from firing until a new pattern was inserted on the input wires, and a short positive pulse was received on the "*New*" wire. The state of the bi-stable circuit then changed from the reset state to the normal state, and the neurons were able to integrate the input voltages and evaluate the *IIP*.

In Fig. 6, if V_{PRE} and V_{POST}^{N} are smaller and V_{POST}^{P} is greater than the memristor threshold voltage, the learning policy can be applied correctly to the memristors:

- When a given input line and the output neuron fire simultaneously, the potential difference across the memristor will be like the signal shape shown in Fig. 6(c). In this event, the voltage exceeds the positive threshold and causes a decrease in memristance, which consequently results in a weight increase.
- When a given input line remains at the rest potential and the output neuron fires, the potential difference across the memristor will be similar to the signal shape shown in Fig. 6(d). In this event, the voltage exceeds the negative threshold and causes an increase in memristance which consequently results in a weight decrease.

Depending on the signal shapes and learning algorithm, the desired δ_N and δ_P can be set in Eq. (2) by changing the positive and negative pulse widths, respectively, of the output spike ($T_{POST}^{P}, T_{POST}^{N}$).

In this learning algorithm, all the weights are initialized to the same specific weight. Because of the competitive structure of the neurons, the exact value of the initial weight is less significant here than the equality of the weights before the learning process, and consequently, the important thing is not the exact value of the positive pulse width but the application of the same pulse widths to the memristors.

In a more futuristic design, the system could be laid out in CMOL architecture, with the memristor crossbar being fabricated on top of the CMOS neurons and driving circuits [34].

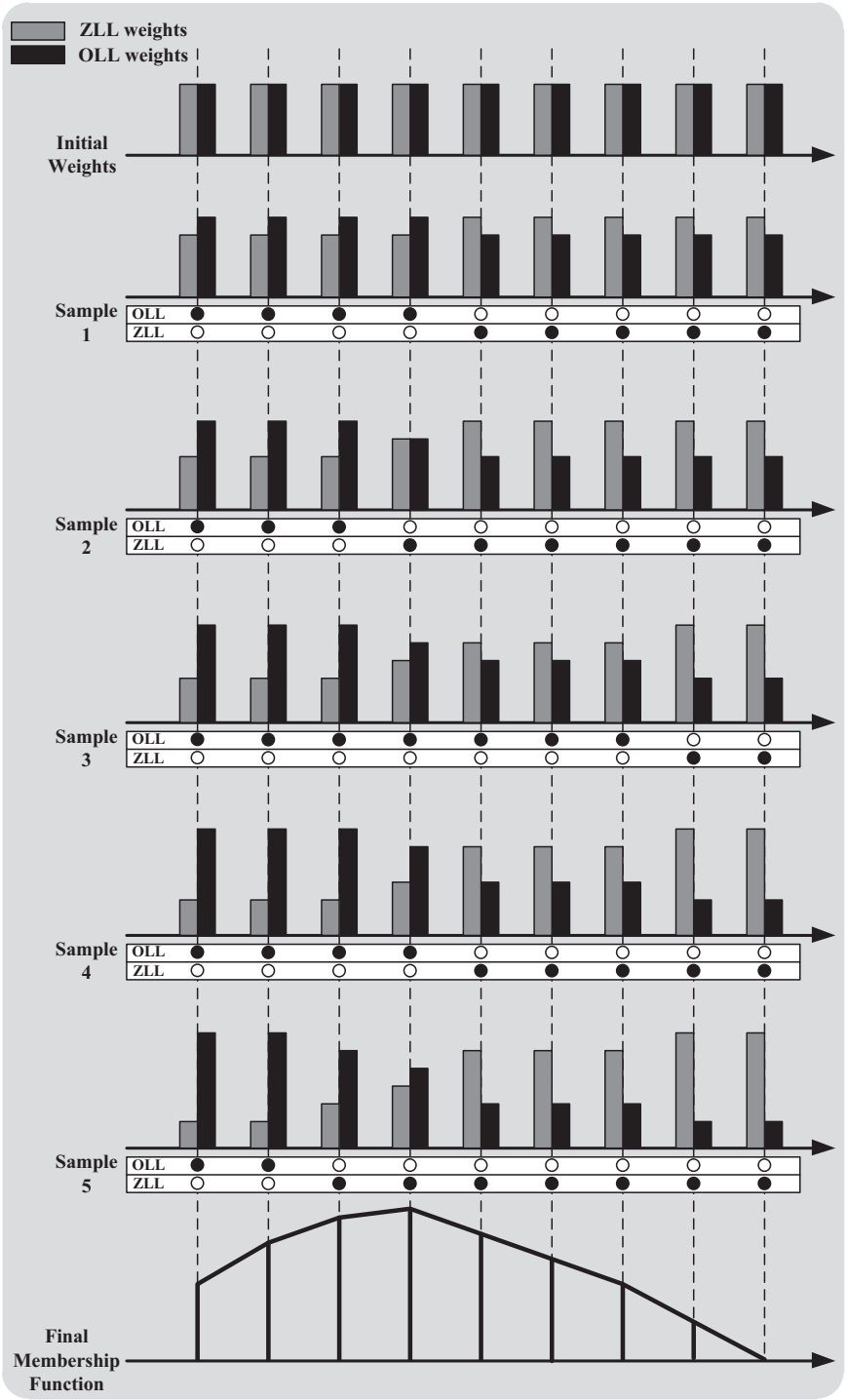

Fig. 4. An example of changing OLL and ZLL weights during the clustering process of five input samples, and the resultant fuzzy membership function of the cluster.

6. Results

In this section, to evaluate the performance of the proposed circuit in the clustering task, two widely studied realistic datasets are used: the Simon Lucas handwritten alphadigits dataset [35] with the binary input space and Fisher's 4-dimensional Iris dataset [36], together with some other applications, with the analog input space. The impact of different fabrication and signal variations on the final neuro-fuzzy clustering circuit is then examined and the circuit is shown to be suitably immune to different kinds of variation. Due to the huge computational requirements of the simulation tasks especially for the variability analysis, and their parallel capacity, simulations accomplished on the newly proposed GPU-based CUDA parallel platform [37].

6.1. Binary input space

The Simon Lucas dataset consists of handwritten binary 20×16 digits from “0” through “9” and capitals from “A” through “Z” by 39 writers in the binary input space. This database has been used

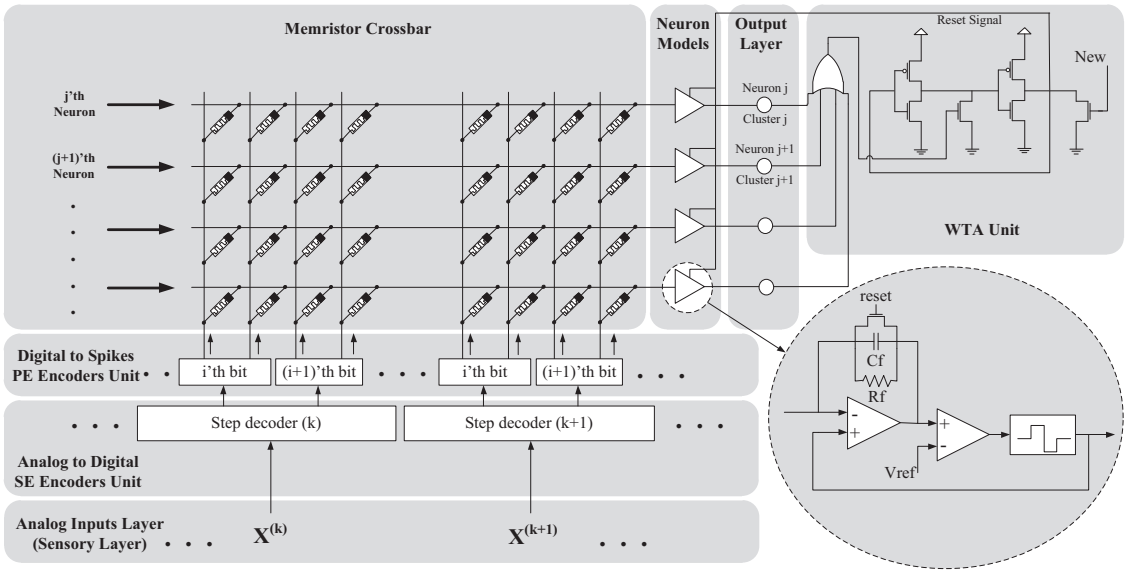

Fig. 5. proposed memristor crossbar-based neuro-fuzzy clustering circuit. This circuit contains SE and PE encoders, a memristor crossbar, neuron models and a WTA unit.

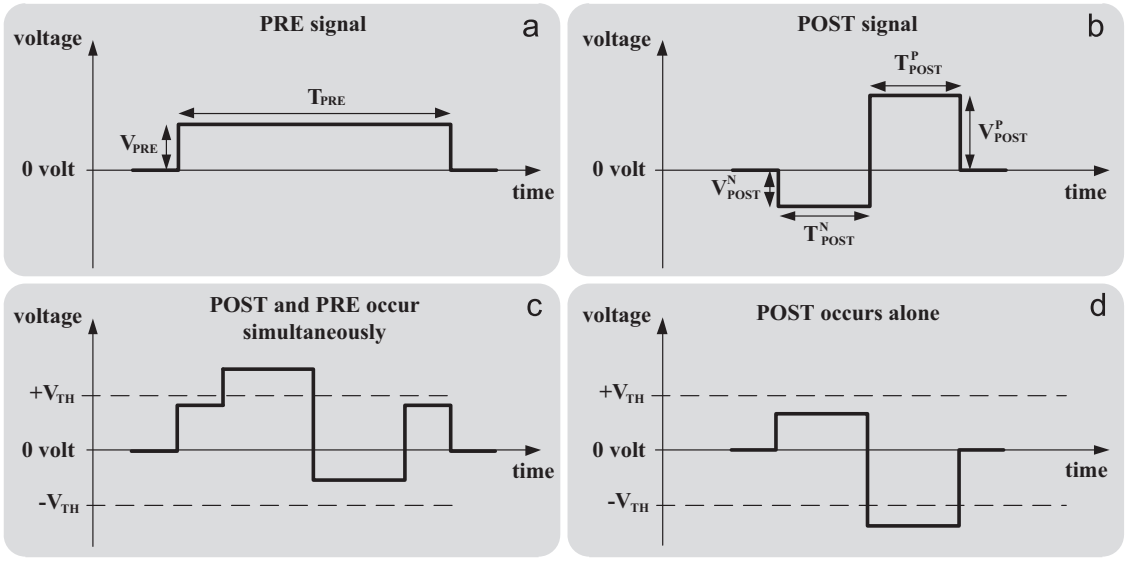

Fig. 6. (a) PRE signal shapes. (b) POST signal shapes. (c) Voltage applied to memristor when PRE and POST occur simultaneously. V_{TH} is the memristor threshold voltage. (d) Voltage applied to memristor when POST occurs alone.

as a test for many learning algorithms. During the learning process, we presented the full Simon Lucas training database (1404 characters) ten times to the circuit recursively. No kind of preprocessing on the digits was used, and the set was not augmented with distortions. The network was then tested on all 1404 characters. Input patterns were presented to the first layer of neurons, which we refer to as level 1 (each pair of input neurons were connected with one pixel of the image). In this clustering task we used networks with different second layer neurons (36–300) in which each neuron received all 320 input pixels (640 input neurons) in a fully connected, unsupervised neural network, and 36 output neurons (third layer) with a supervised learning algorithm to merge the second layer neurons clustering the same letter or digit with different fonts. In this study, each second layer neuron was labeled by one special letter or digit and the other patterns were wrong if recognized by this neuron. The recognition rate of the PE scheme was 57.88% with 36 s layer neurons, 66.66% with 100 s layer neurons, and 79.88% with 300 s layer neurons, as compared to the single spike scheme that achieved 67.50%, and a traditional artificial neural network with Self-Organizing Map (SOM) that attained 75.74% with 300 hidden neurons. Note that SOM requires a large computational effort to compute the input vector and weight vector correlation, which is not easy to implement. The recognition rate changed proportionally to the training epoch number and the second layer neuron number. These relations are depicted in the curves shown in Figs. 7 and 8.

6.2. Analog input space

After the PE clustering scheme had been tested in the binary space and produced an acceptable performance, it was incorporated into the SE scheme in order to develop our proposed neuro-fuzzy clustering scheme. To test the neuro-fuzzy approach in the analog space, we used Fisher's Iris dataset. The input features in this dataset are sepal length, sepal width, petal length, and petal width of three different kinds of iris. The data set consists of 50 samples from each of the three species. This dataset has been applied to some powerful networks, the clustering results of

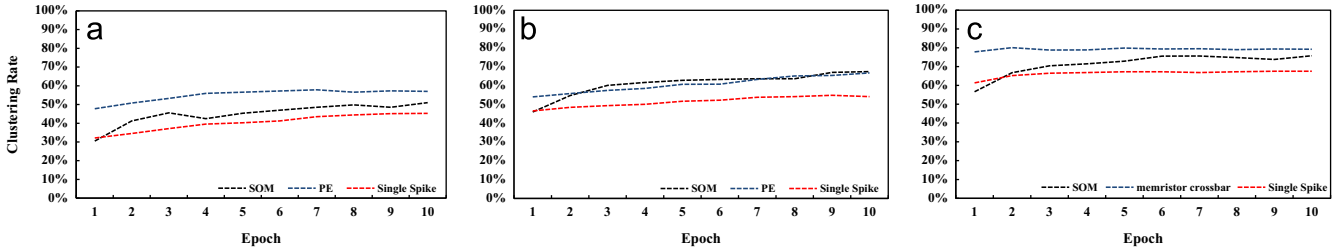

Fig. 7. Recognition rate of our PE scheme compared to the single spike scheme and SOM during learning phase with (a) 36 s layer neurons. (b) 100 s layer neurons. (c) 300 s layer neurons on Simon Lucas dataset. Each epoch contains 1404 iterations.

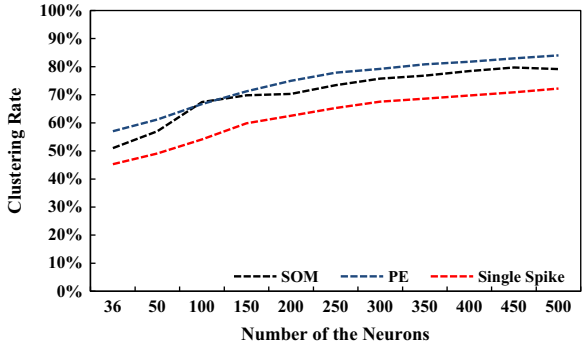

Fig. 8. Clustering rate of our PE scheme compared to the single spike scheme and SOM for different numbers of neurons on the Simon Lucas dataset.

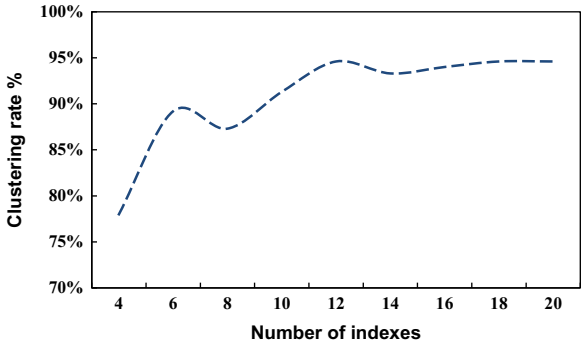

Fig. 9. Clustering performance of the proposed neuro-fuzzy clustering system on Fisher's Iris dataset for different numbers of indexes on each input axis.

Table 1
Clustering performance of different algorithms [29] on Fisher's Iris data-set.

Neural network	Accuracy (%)
Spiking RBF	92.6 ± 0.9
K-mean	88.6 ± 0.1
SOM	85.3 ± 0.1
Proposed (PE&SE) network	94.6 ± 0.7

which are reported in [29]. The clustering results of these networks and also the proposed PE&SE network are shown in Table 1. In this table, all the networks used 3 output neurons to cluster the Iris dataset. Our network used 20 indexes per input to encode analog data to spikes. This means that 40 input lines were used to represent each input feature, so in total $4 \times 40 = 160$ input lines were used in the circuit. A total of $3 \times 160 = 480$ memristors was therefore used in the circuit to implement three neurons. During the learning process, the dataset was presented to the circuit ten times recursively.

As can be seen in Table 1, the proposed network achieved a higher clustering performance in comparison with the other powerful networks. Note that the other networks require a comparatively huge computational effort to perform the clustering task, and are not easily implementable.

Fig. 9 shows the circuit's clustering performance for different numbers of indexes on each input axis on Fisher's Iris data-set. After setting the number of indexes on each input axis, we applied the dataset to the circuit ten times and evaluated the final clustering rate. This process was carried out for different numbers of indexes, as plotted in Fig. 9. The performance of the proposed scheme was clearly improved by increasing the number of indexes on each input axis until the effect of digitalizing error was almost eliminated and the performance became dependent exclusively on the learning algorithm. Thus, when the index number is big enough, the circuit's clustering performance does not depend on the index number and is not changed by increasing the index number. The local decrease in performance by increasing the number of indexes is predictable, because the index position on the input axis changes when the number of indexes is changed. This position change is negligible for greater number of indexes and does not influence circuit performance, but it is significant for smaller number of indexes and may decrease the clustering performance considerably. Fig. 10 shows the dataset sample distribution in the input space and the clusters created by the output neurons on the input space. To visually evaluate the clustering performance with the Iris dataset, the contour of cluster points could be compared with the real dataset samples. Fig. 11 shows the normalized membership functions created on the input axes by three output neurons after a learning phase with 20 indexes per input. As can be seen, the membership functions have been matched to the clusters shown in Fig. 10.

In Fig. 12, we applied our approach to a 2 dimensional hierarchical clustering task and investigated the clustering performance by changing the number of clusters from four to two. As can be seen, the system properly divided the hierarchical data samples into four clusters. By changing the number of clusters to two, two pairs of neighbor clusters were merged together and the data samples fell within two proper clusters.

In Fig. 13 we applied our approach to a 2 dimensional clustering task involving different cluster sizes and distances and including some noisy data samples, to investigate the effect of noise, cluster size, and clusters distances on its clustering performance. As can be seen, the system proved to be immune to the aforementioned effects and capable of properly dividing the data samples.

Finally, we tested our approach in the commonly used image segmentation application. Here, the input space was a 3 dimensional integer space and each sample vector, presenting one pixel of the picture, contained three integer numbers of red, green and blue color intensity, all falling within the [0–256] interval. Fig. 14 shows the target picture and its 3 segment version (3 clusters) using different clustering algorithms. No kind of preprocessing

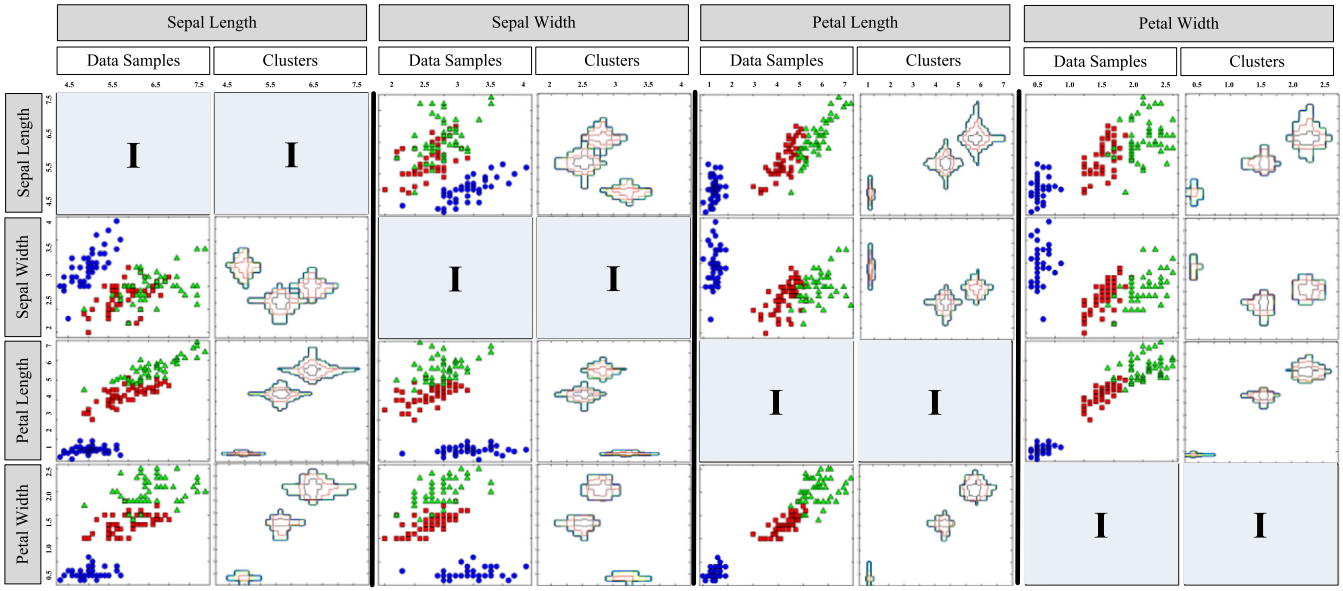

Fig. 10. Fisher's Iris data-set. Distribution of dataset samples in the input space (blue circles = setosa, red squares = versicolor and green triangles = virginica) and contour plot of the membership degree of clusters in the input space created by 3 output neurons in the proposed neuro-fuzzy network.

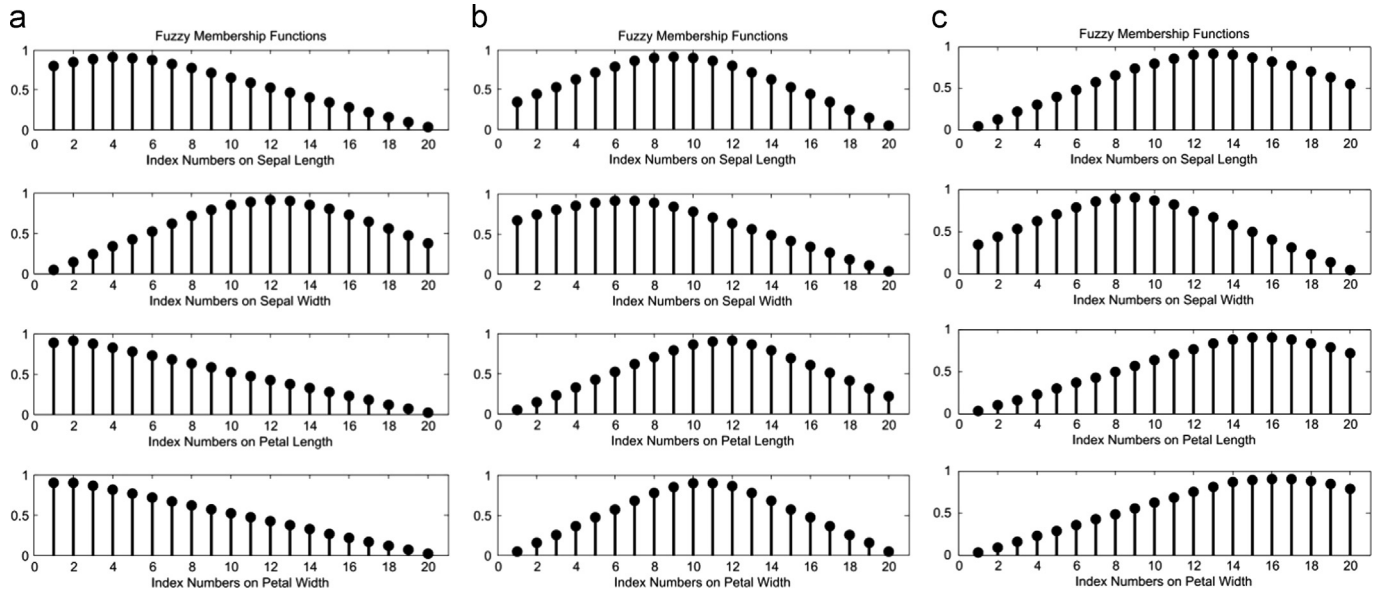

Fig. 11. Normalized membership functions created on four input feature axes by (a) first, (b) second and (c) third output neuron after a learning phase with 20 indexes per input.

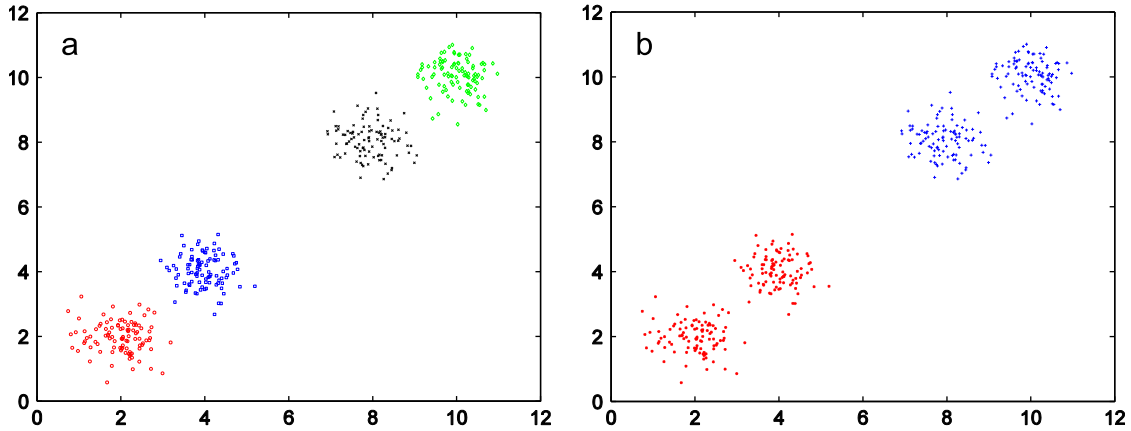

Fig. 12. Hierarchical data sample clustering task performed by PE&SE approach (a) with 4 output neurons (clusters) and (b) with 2 output neurons (clusters).

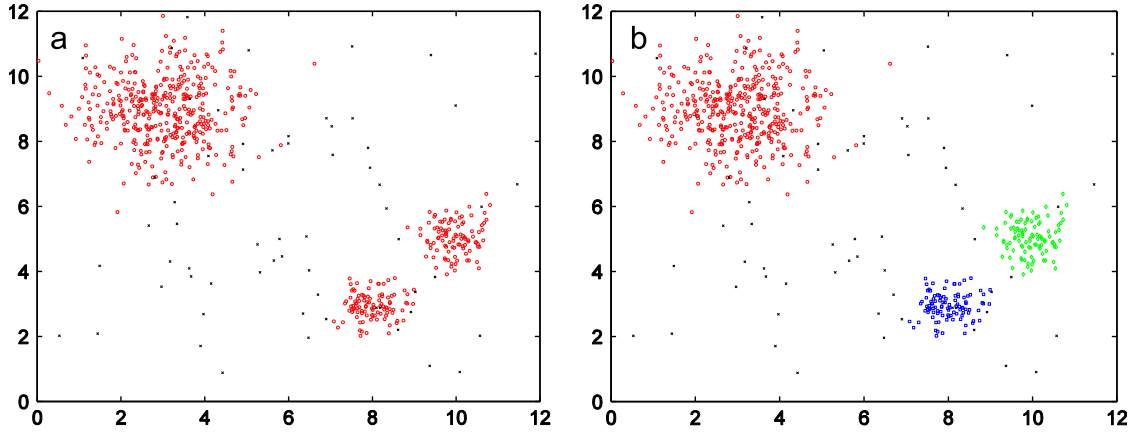

Fig. 13. (a) Noisy data samples with different cluster sizes (sample density) and distances in a 2 dimensional space. (b) Clustering task performed by PE&SE approach in this space.

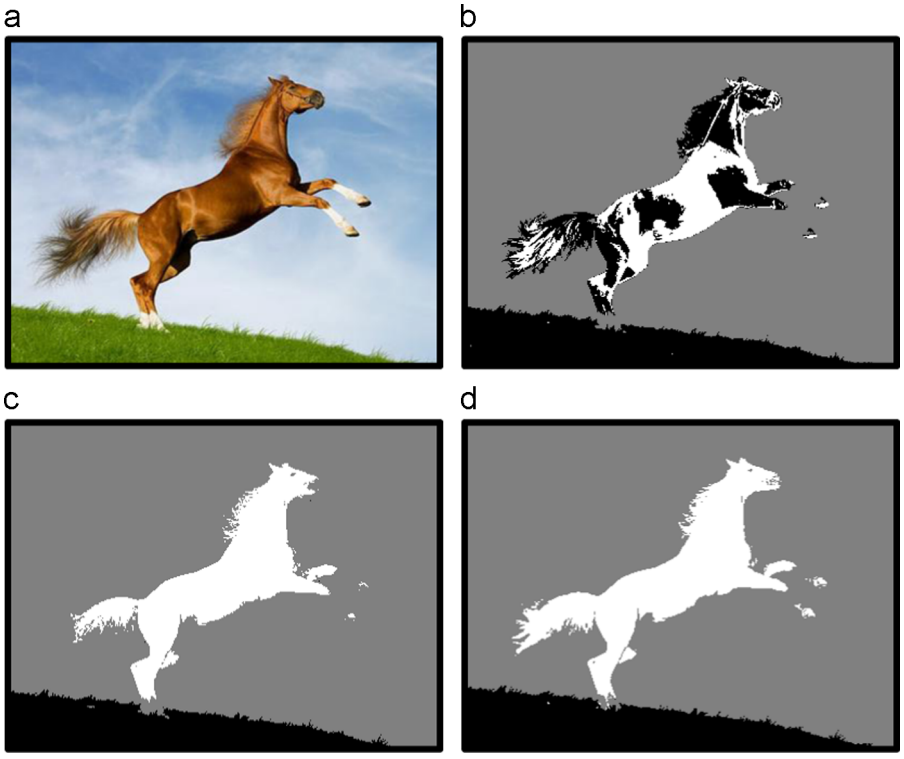

Fig. 14. Comparison of our approach against similar approaches for an unsupervised picture segmentation task. (a) Original target picture. (b) Segmented picture using the SOM algorithm. (c) Segmented picture using the *K*-mean algorithm. (d) Segmented picture using the neuro-fuzzy SE&PE system.

was carried out on the figure before applying it to the algorithms. Visual Comparison of the pictures shows that the neuro-fuzzy SE&PE system achieves a good segmentation performance compared to similar algorithms and can properly segment the picture into three parts: the horse, the cloudy sky, and the hill.

6.3. *Variability analysis*

The biggest issue in memristive circuit implementation is the variation of the devices. As mentioned earlier, a memristor is a nano-scale device, and device variation clearly becomes a major challenge when the area of implementation is decreased. The electrical characteristics of memristors are mainly determined by the properties of the materials used and the fabrication process. For example, process variations may cause the actual electrical behavior of memristors to deviate from the original design and result in the malfunctioning of the device. CMOS circuit variations can also lead to changes in input and output signal parameters which eventually influence learning parameters such as δ_P and δ_N in Eq. (2) and also influence how the winner neuron is determined. It is therefore very important to understand and characterize the impact of process variations on circuit performance. Here, we evaluate the effect of all the aforementioned variations on circuit performance by studying the normal distribution of the parameters around a mean value μ with different deviation values σ.

Fig. 15 shows how variations in different parameters can affect the clustering performance of the circuit. The full range of different results is shown, together with the distance between the worst case and best case curves. In Fig. 15(a), we examined the impact of variation in the reset signal, which in turn leads to variation in the initial conductance of memristors. As can be seen, initial weight variation has a slight, erratic influence on the

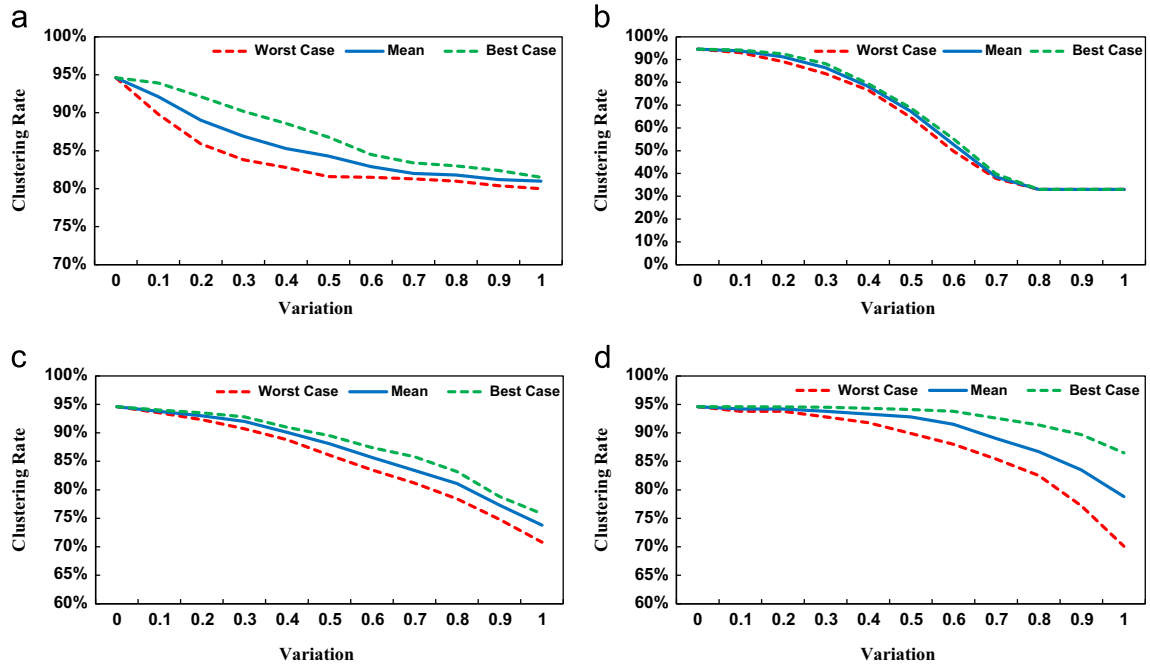

Fig. 15. Impact of (a) Initial conductance variation (σ/μ). (b) Full failure of memristor (%). (c) Fabrication process variation (σ/μ). (d) Learning signal shape variation (σ/μ) on the network's clustering performance.

clustering rate. It should be noted that variation in the initial weights increases the number of iterations required to achieve the best result.

During the fabrication process, in the worst case, some memristors may remain open circuit because of a failed implementation process. In Fig. 15(b), we randomly considered a portion of memristors as open circuit, and studied the circuit behavior. The results showed that the circuit was robust to this kind of variation, and its decrease in performance was negligible, with 30% fully failed memristors.

In Fig. 15(c), variation was applied to all the memristor's parameters to investigate the impact of general fabrication process variations such as cross-section area variation, thickness variation, etc. As can be seen, the circuit was robust to general fabrication variations.

Finally, Fig. 15(d) shows the impact of CMOS variation, which leads to learning signal shape variation in circuit performance. In this figure, we applied variations in the learning signal shape parameters introduced in Section 4. The results showed that the circuit remained suitably robust to learning signal shapes variation.

7. Conclusion

In this paper, we proposed a binary to spikes encoding scheme called PE and an analog to binary encoding scheme called SE which, when combined, result in a novel powerful spiking neuro-fuzzy clustering system based on the simplified STDP learning algorithm. We then introduced circuitry implementation for this scheme on the memristor crossbar associated with the CMOS neuromorphic circuit. By implementing the PE and SE encoding schemes in the network the QSTDP learning algorithm was transformed into a high performance neuro-fuzzy learning algorithm. The resultant learning algorithm creates a membership function of the cluster on each input feature axis and modifies it to different convex shapes during the learning phase. The scheme can create various convex cluster shapes in the analog input space due to its variable membership function shape, and this capability turns it into a powerful clustering algorithm. The PE encoding scheme applied distinguishes the concepts of "*lack of knowledge*" and "*uncertainty*", making the learning algorithm robust in comparison with the single spike scheme, and the SE encoding scheme applied introduces fuzziness into the network, making the learning algorithm more robust. Moreover, this fuzziness, combined with the competitive nature of the network makes the network immune to fabrication and signal variations. This scheme used spikes to represent input values. Using spikes enables the network to receive sensory layer data through an Address Event Representation (AER) bus [38], and also send its output through a low-cost, high bandwidth transmission channel. This design approach could provide the groundwork for future circuits capable of processing natural data in a compact, low power manner. Thanks to their unsupervised learning capabilities, such circuits will be able to adapt to various environments. Future work should focus on the experimental demonstration of these concepts, and on showing how they can be scaled to more complex multi-layer networks, which employ a combination of supervised and unsupervised learning, and to other kinds of sensory stimuli.

Acknowledgments

This work was partially funded by ERANET grant PRI-PIMCHI-2011-0768 (PNEUMA) awarded by the Spanish Ministerio de Economía y Competitividad (Ministry of Economy and Competitiveness).

Appendix A. Convexity of the membership functions

Statement: All the membership functions have convex shapes.

Proof: For the first step, we should evaluate the slope of the membership function in all the indexes. For the sake of simplification, we eliminate indexes j and k from Eq. (4) and replace $(X^{(k)})$ with the index number *idx*. Thus, the equation can be rewritten as

follows:

$$MD(idx) = \sum_{i=1}^{i=idx} W_i^{OLL} + \sum_{i=idx+1}^{i=N} W_i^{ZLL} \quad (A.1)$$

The slope equation for a given index *idx* on the input axis is

$$Slope(idx) = MD(idx) - MD(idx-1). \quad (A.2)$$

Replacing $MD(idx)$ and $MD(idx-1)$ from Eq. (A.1) in Eq. (A.2) gives

$$Slope(idx) = \left(\sum_{i=1}^{i=idx} W_i^{OLL} + \sum_{i=idx+1}^{i=N} W_i^{ZLL}\right) - \left(\sum_{i=1}^{i=idx-1} W_i^{OLL} + \sum_{i=idx}^{i=N} W_i^{ZLL}\right)$$

$$\Rightarrow Slope(idx) = W_{idx}^{OLL} - W_{idx}^{ZLL}. \quad (A.3)$$

According to the derived equation, the slope of the membership function in each index point is equal to the ZLL weight of that index subtracted from its OLL weight.

For the second step, we prove that the OLL weight curve is a decreasing curve and the ZLL curve is an increasing curve by increasing the index number. As we know, for a given input *x*, the lower hand OLL input neurons fire and the upper hand OLL input neurons remain in the rest state. Therefore, according to the learning algorithm, in each learning step the lower hand OLL weights increase and the upper hand OLL weights decrease. Consider a given input sample x and two consecutive indexes *idx* and $idx+1$. We now have three different conditions during one step of the learning process

$$1: \text{ If } x < idx.P \Rightarrow \begin{cases} W_{idx}^{OLL}(n+1) = W_{idx}^{OLL}(n) - \delta_N \\ W_{idx+1}^{OLL}(n+1) = W_{idx+1}^{OLL}(n) - \delta_N \end{cases}. \quad (A.4)$$

$$2: \text{ If } idx.P \le x < (idx+1).P$$

$$\Rightarrow \begin{cases} W_{idx}^{OLL}(n+1) = W_{idx}^{OLL}(n) + \delta_P \\ W_{idx+1}^{OLL}(n+1) = W_{idx+1}^{OLL}(n) - \delta_N \end{cases}. \quad (A.5)$$

$$3: \text{ If } x \ge (idx+1).P \Rightarrow \begin{cases} W_{idx}^{OLL}(n+1) = W_{idx}^{OLL}(n) + \delta_P \\ W_{idx+1}^{OLL}(n+1) = W_{idx+1}^{OLL}(n) + \delta_P \end{cases}. \quad (A.6)$$

As seen in Eqs. (A.4)–(A.6), considering the same initial value of the weights, in all the conditions $W_{idx}^{OLL} \ge W_{idx+1}^{OLL}$, thus proving that the OLL weight curve is a decreasing curve.

Following the same reasoning as for the ZLL weight curve, for a given input *x*, the lower hand ZLL input neurons remain in the rest state and the upper hand ZLL input neurons fire. According to the learning algorithm, therefore, in each learning step the lower hand ZLL weights decrease and the upper hand ZLL weights increase. Consider a given input sample *x* and two consecutive indexes *idx* and $idx+1$. We now have three different conditions during one step of learning process

$$1: \text{ If } x < idx.P \Rightarrow \begin{cases} W_{idx}^{ZLL}(n+1) = W_{idx}^{ZLL}(n) + \delta_P \\ W_{idx+1}^{ZLL}(n+1) = W_{idx+1}^{ZLL}(n) + \delta_P \end{cases}. \quad (A.7)$$

$$2: \text{ If } idx.P \le x < (idx+1).P$$

$$\Rightarrow \begin{cases} W_{idx}^{ZLL}(n+1) = W_{idx}^{ZLL}(n) - \delta_N \\ W_{idx+1}^{ZLL}(n+1) = W_{idx+1}^{ZLL}(n) + \delta_P \end{cases}. \quad (A.8)$$

$$3: \text{ If } x \ge (idx+1).P \Rightarrow \begin{cases} W_{idx}^{ZLL}(n+1) = W_{idx}^{ZLL}(n) - \delta_N \\ W_{idx+1}^{ZLL}(n+1) = W_{idx+1}^{ZLL}(n) - \delta_N \end{cases}. \quad (A.9)$$

As seen in Eqs. (A.7)–(A.9), considering the same initial value of the weights, in all the conditions $W_{idx}^{ZLL} \le W_{idx+1}^{ZLL}$, thus proving that the OLL weight curve is an increasing curve.

The OLL weight curve is therefore a decreasing curve and the ZLL curve is an increasing curve, and these two curves pass over each other in one index or in several consecutive indexes. On the upper hand of the cross point (or line) W^{ZLL} is bigger than W^{OLL} and this difference increases by getting distance from the cross point (or line) index (or indexes). Thus, according to Eq. (A.3), the membership function slope is negative and becomes more negative by getting distance from the cross point (or line) index (or indexes). Likewise, on the lower hand of the cross point (or line) W^{OLL} is bigger than W^{ZLL} and this difference increases by getting distance from the cross point (or line) index (or indexes). Thus, the slope of the membership function is positive and becomes more positive by getting distance from the cross point (or line) index (or indexes). On the cross point (or line) $W^{ZLL} = W^{OLL}$ and the slope of the membership function is zero and the membership function is flat. These specifications of the membership functions fulfill the required conditions sufficiently to prove this statement that all the membership functions have convex shapes.

References

[1] L. Zadeh, Fuzzy sets, Inf. Control 8 (1965) 338–353.
[2] C.-C. Lee, Fuzzy logic in control systems: fuzzy logic controller. I, IEEE Trans. Syst. Man Cybern. 20 (2) (1990) 419–435.
[3] B. Kosko, Neural Networks and Fuzzy Systems – A Dynamical Systems Approach to Machine Intelligence, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1992.
[4] W. Pedrycz, Fuzzy Control and Fuzzy Systems, Wiley, Toronto, 1993.
[5] C.-T. Lin, C.-C. Lee, Neural Fuzzy Systems. A Neuro-Fuzzy Synergism to Intelligent Systems, Prentice Hall, New York, 1996.
[6] D. Nauck, F. Klawonn, R. Kruse, Foundations of Neuro-Fuzzy Systems, Wiley, Chichester, 1997.
[7] R.E. Bellman, L.A. Zadeh, Decision-making in a fuzzy environment, Manag. Sci. 17 (4) (1970) 141–164.
[8] M. Hanss, Applied Fuzzy Arithmetic – An Introduction With Engineering Applications, Springer, Springer-Verlag Berlin Heidelberg (2005) 2 (chapter 1).
[9] L.A. Zadeh, Fuzzy logic, neural networks, and soft computing, Commun. ACM 37 (3) (1994) 77–84.
[10] G. Bi, M. Poo, Synaptic modification in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type, J. Neurosci. 18 (1998) 10464–10472.
[11] D.B. Strukov, G.S. Snider, D.R. Stewart, R.S. Williams, The missing memristor found, Nature 453 (2008) 80–83.
[12] L.O. Chua, Memristor – the missing circuit element, IEEE Trans. Circuit Theory 18 (1971) 507–519.
[13] Y. Ho, G.M. Huang, P. Li, Dynamical properties and design analysis for nonvolatile memristor memories, IEEE Trans. Circuits Syst. 58 (4) (2011) 724–736.
[14] F. Merrikh-Bayat, S. Bagheri-Shouraki, Mixed analog–digital crossbar-based hardware implementation of sign–sign LMS adaptive filter, J. Analog Integr. Circuits Signal Process. 66 (1) (2011) 41–48.
[15] T. Raja, S. Mourad, "Digital logic implementation in memristor-based Crossbars", ICCCAS 2009, in: Proceedings of the International Conference on Communications, Circuits and Systems, 2009, pp. 939–943.
[16] B. Linares-Barranco, T. Serrano-Gotarredona, Memristance can explain spike-time-dependent-plasticity in neural synapses, Nat. Preced. (2009) 1–4.
[17] S.P. Adhikari, Y. Changju, K. Hyongsuk, L.O. Chua, Memristor bridge synapse-based neural network and its learning, IEEE Trans. Neural Netw. Learn. Syst. 23 (9) (2012) 1426–1435.
[18] G. Howard, E. Gale, L. Bull, B.d.e. Lacy Costello, A. Adamatzky, Evolution of plastic learning in spiking networks via memristive connections, IEEE Trans. Evolut. Comput. 16 (5) (2012) 711–729.
[19] A. Afifi, A. Ayatollahi, F. Raissi, Implementation of biologically plausible spiking neural network models on the memristor crossbar-based CMOS/nano circuits, in: Proceedings of the European Conference on Circuit Theory and Design, ECCTD 2009, pp. 563–566, 2009.
[20] T. Serrano-Gotarredona, T. Masquelier, T. Prodromakis, G. Indiveri, B. Linares-Barranco, STDP and STDP variations with memristors for spiking neuromorphic learning systems, Front. Neurosci. 7 (2) (2013), http://dx.doi.org/10.3389/fnins.2013.00002.
[21] D. Querlioz, O. Bichler, P. Dollfus, C. Gamrat, Immunity to device variations in a spiking neural network with memristive nanodevices, IEEE Trans. Nanotechnol. 12 (3) (2013) (288, 295).
[22] O. Bichler, M. Suri, D. Querlioz, D. Vuillaume, B. DeSalvo, C. Gamrat, Visual pattern extraction using energy-efficient 2-PCM synapse neuromorphic architecture, IEEE Trans. Electron Devices 59 (8) (2012) 2206–2214.
[23] S. Yu, Y. Wu, R. Jeyasingh, D. Kuzum, H.S. Wong, An electronic synapse device based on metal oxide resistive switching memory for neuromorphic computation, IEEE Trans. Electron Devices 58 (8) (2011) 2729–2737.

[24] K. Seo, I. Kim, S. Jung, M. Jo, S. Park, J. Park, H. Hwang, Analog memory and spike-timing-dependent plasticity characteristics of a nanoscale titanium oxide bilayer resistive switching device, Nanotechnology 22 (25) (2011) 254023.
[25] F. Merrikh-Bayat, S. Bagheri-Shouraki, Neuro- Fuzzy Computing System with the Capacity of Implementation on Memristor-Crossbar and Optimization-Free Hardware Training, IEEE Trans. Fuzzy Syst. 99 (2013) 1, http://dx.doi.org/10.1109/TFUZZ.2013.2290140.
[26] M. Bavandpour, H. Soleimani, S. Bagheri-Shouraki, A. Ahmadi, D. Abbott, L.O. Chua, Cellular memristive dynamical systems (CMDS), Int. J. Bifurc. Chaos 24 (05) (2014), 1430016.1–1430016.22, http://dx.doi.org/10.1142/S021812741430016X.
[27] D. Strukov, K. Likharev, A reconfigurable architecture for hybridCMOS/nano-device circuits, in: Proceedings of the 2006 ACMISIGDA 14th International Symposium on Field Programmable Gate Arrays, pp. 131–140, 2006.
[28] O. Kavehei, S. Al-Sarawi, K.R. Cho, K. Eshraghian, D. Abbott, An analytical approach for memristive nanoarchitectures, IEEE Trans. Nanotechnol. 11 (2) (2012) 374–385.
[29] S.M. Bohte, H. La Poutré, J.N. Kok., Unsupervised clustering with spiking neurons by sparse temporal coding and multilayer RBF networks, IEEE Trans. Neural Netw. 13 (2) (2002) 426–435.
[30] J.Y. Yam, T.W. Chow, A weight initialization method for improving training speed in feedforward neural network, Neurocomputing 30 (1) (2000) 219–232.
[31] D. Querlioz, P. Dollfus, O. Bichler, C. Gamrat, Learning with memristive devices: how should we model their behavior, in: Proceedings of the IEEE/ACM International Symposium on Nanoscale Architecture, pp. 150, 2011.
[32] H. Jo, T. Chang, I. Ebong, B.B. Bhadviya, P. Mazumder, W. Lu, Nanoscale memristor device as synapse in neuromorphic systems, Nano Lett. 10 (2010) 1297–1301.
[33] C. Koch, S. Idan, Methods in Neuronal Modeling, MITpress, Massachusetts, 1988.
[34] H. Lee, K.K. Likharev, Defect-tolerant nanoelectronic pattern classifiers, Int. J. Circuit Theory Appl. – Nanoelectron. 35 (3) (2007) 239–264.
[35] Simon Lucas dataset. ⟨http://www.cs.nyu.edu/⟩ roweis/data/binaryalphadigs.mat (accessed 10.09.12).
[36] R.A. Fisher, The use of multiple measurements in taxo-nomic problems, Ann. Eugen. 7 (2) (1936) 179–188.
[37] M. Bavandpour, S.B. Shouraki, H. Soleimani, A. Ahmadi, A.A. Makhlooghpour, Simulation of memristor crossbar structure on GPU platform, in: Proceedings of the 2012 20th Iranian Conference on Electrical Engineering (ICEE), 2012, pp. 178, 183.
[38] E. Culurciello, R. Etienne-Cummings, K.A. Boahen, A biomorphic digital image sensor, IEEE J. Solid-State Circuits 38 (2) (2003) 281–294.