J Real-Time Image Proc (2014) 9:479–489
DOI 10.1007/s11554-012-0254-9

SPECIAL ISSUE

Memristive fuzzy edge detector

Farnood Merrikh-Bayat · Saeed Bagheri Shouraki · Farshad Merrikh-Bayat

Received: 11 October 2011 / Accepted: 4 May 2012 / Published online: 24 May 2012

Abstract Fuzzy inference systems always suffer from the lack of efficient structures or platforms for their hardware implementation. In this paper, we tried to overcome this difficulty by proposing a new method for the implementation of the fuzzy rule-based inference systems. To achieve this goal, we have designed a multi-layer neuro-fuzzy computing system based on the memristor crossbar structure by introducing a new concept called the fuzzy minterm. Although many applications can be realized through the use of our proposed system, in this study we only show how the fuzzy XOR function can be constructed and how it can be used to extract edges from grayscale images. One main advantage of our memristive fuzzy edge detector (implemented in analog form) compared to other commonly used edge detectors is it can be implemented in parallel form, which makes it a powerful device for real-time applications.

Keywords Neuro-fuzzy computing system · Edge detection · Hardware implementation · Memristor · Memristive system · Crossbar structure

F. Merrikh-Bayat (✉) · S. Bagheri Shouraki
Department of Electrical Engineering,
Sharif University of Technology, Tehran, Iran
e-mail: farnoodmb@gmail.com; f_merrikhbayat@ee.sharif.edu

S. Bagheri Shouraki
e-mail: bagheri-s@sharif.edu

F. Merrikh-Bayat
Department of Electrical and Computer Engineering,
University of Zanjan, Zanjan, Iran
e-mail: f.bayat@znu.ac.ir

1 Introduction

Most of the signal processing tools currently used in practice, such as Digital Signal Processors (DSPs) and synchronous Field Programmable Gate Arrays (FPGAs), have two main similarities: first, they have been constructed based on the digital logic, and second, they generate signals only at distinct moments of time according to the system's clock pulse. As a well-known fact, the first similarity leads to the separation of memory and computing units [1], which can reduce the efficiency of the signal processing system. Moreover, it results in a limited numerical precision since the input signals are usually quantized (because of using the A/D converters) and also both the output and internal signals are rounded or truncated to a specific limited precision [2–4]. On the other hand, the second similarity puts a limitation on the speed of system, and moreover, it commonly leads to high area-consuming designs.

Although the integrated circuits industry has successfully overcome the above-mentioned difficulties in the past decades, it is obvious that this approach cannot last forever [5]. That is why many researchers try to develop alternative computer architectures, new computing schemes and advanced materials in addition to pushing the technology for producing smaller devices. One important development in this field happed on 1 May 2008 when a research group in Hewlett-Packard Labs reported the physical realization of the first *memristor* [6], the fourth fundamental passive circuit element, which was also predicted by Leon Chua in 1971 [7]. After that innovation, many researchers sought the applications of memristor in variety of scientific fields such as neuroscience, neural networks and artificial intelligence. It is now clear that this passive element can have many potential applications such as creation of analog

neural networks and emulation of human learning [8], building programmable analog circuits [9, 10], constructing hardware for soft computing tools [11], implementing digital circuits [12], and in the field of signal processing [13, 14]. One main reason for this great interest in memristive computing systems is that these systems have a high potential to overcome most of the aforementioned challenges confronting today's digital systems. For example, it is proved that these systems can be constructed much denser and faster than their counterparts through the use of nano-crossbar technology, and moreover, they consume a considerably less energy [15].

Recently, the authors [11] showed that it is possible to design a memristive soft computing system with learning capabilities by using fuzzy logic instead of the digital logic. The proposed system was implemented in analog form and therefore it was very fast and completely consistent with the analog nature of memristors. Moreover, our proposed neuro-fuzzy computing system had the advantage that in its hierarchical structure, memory units were assimilated within computational units, similar to the human brain.

In this paper, we propose another way to implement the fuzzy rule-based inference systems by introducing the concept of *fuzzy minterm*. Although the proposed analog multi-layer neuro-fuzzy system can be used in a wide variety of image processing applications, here we concentrate only on the detection of edges in grayscale images. For this purpose, first we develop the fuzzy XOR function by using a fuzzy rule-based inference system. Then, we propose a method for hardware implementation of the fuzzy XOR function based on memristor crossbars. Simulation results show that the proposed structure can effectively extract the edges of the given image even in a noisy environment. Moreover, since the proposed structure is in analog form it is very fast, and consequently, it provides us with a suitable device for real-time image processing applications. Furthermore, the proposed structure has the advantage of detecting all horizontal and vertical edges simultaneously. Finally, note that since our proposed computing system is constructed based on memristor crossbars, it can be reconfigured by reprogramming its memristors even when it is in use.

The rest of this paper is organized as follows. The notion of memristor crossbar is briefly reviewed in Sect. 2. Application of memristor crossbar for the implementation of an artificial neural network, which can approximate the binary XOR function by using the McCulloch–Pitts method, is discussed in Sect. 3. Section 4 introduces the notion of fuzzy XOR function and proposes a method for its construction based on the memristor crossbar structure. The reason for the importance of the fuzzy XOR function is that it can be used for detecting the edges in a given grayscale picture. The proposed structure for the hardware that can be used for edge detection purposes in practice is studied in Sect. 5. Simulation results are presented in Sect. 6. Section 7 is devoted to discuss on some of the advantages of the proposed structure, and finally, Sect. 8 concludes the paper.

2 Memristor crossbars

Any memristor or memristive device has this property that passing current from it in one direction increases its electrical resistance up to a certain value, while changing the direction of current causes a decrement in electrical resistance [6]. Interesting observation is that the memristor remembers its resistance when the current last went through. That is, turn off the current and the electrical resistance of the memristor freezes until the current is turned back on. Hence, the memristor can be considered as an analog variable resistor whose resistance can be properly adjusted by changing the direction and duration of the voltage applied to it. This property suggests the application of memristor as a simple storage device, where the analog data are stored as a memristance instead of the voltage or charge.

At this time, the main approach for designing memristive computing systems is to apply memristors in the crossbar structure as shown in Fig. 1. In this figure, all parallel wires are conductive and it is assumed that the value of each memristor is adjusted to the desired value by applying a suitable voltage to those wires that memristor is fabricated between them. Note that in practice the memristors appear exactly at the cross-points (i.e., the intersection point of two wires), where the conductive wires are separated by a thin film material [5]. The memristor crossbar shown in Fig. 1 provides us with a perfect structure to store the 2-dimensional analog weight matrix of a neural network [16, 17].

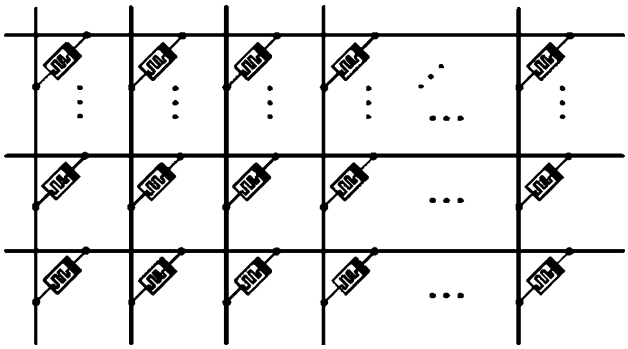


Fig. 1 A typical memristor crossbar

3 Realization of an artificial neural network for approximating the binary XOR function by means of memristor crossbar

As mentioned before, in this paper the edges of the given grayscale image are detected by applying the fuzzy XOR function to adjacent pixels of the picture under consideration. In order to develop a fuzzy XOR function, first in this section we review one of the simplest methods available for the implementation of the logical XOR function by artificial neural networks and then we represent our proposed method for the realization of this XOR function by means of a memristive crossbar structure. In the next section we will extend the results to the fuzzy XOR function.

In the following we base our design on the simple artificial neural network proposed by McCulloch and Pitts in 1943 for the realization of the logical XOR function [18, 19]. Figure 2a shows the artificial neural network under consideration which has two binary inputs (denoted as x_1 and x_2) and a binary output (denoted as y). In this figure, each neuron with binary activation receives a number of inputs (either from the original data or from the output of other neurons in the network), and each input comes via a connector that has a certain weight, which corresponds to the synaptic efficiency in a biological neuron. In each neuron, the weighted sum of inputs is formed and then a simple hard thresholding function is applied to this sum to produce the output of neuron. In the network of Fig. 2a, the threshold value of all neurons, except the neurons of input layer, is considered equal to 2 (as written on the top of each neuron in this figure). It means that when in these neurons the weighted sum of inputs is greater than or equal to 2 the output is set to logic 1 and otherwise it is set to logic 0. By examination of different logical inputs it can be easily verified that this network really acts as a logical XOR function.

In Fig. 2a the input–output relation of neurons can be expressed in a matrix product form as follows:

$$\begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix} = f(S.x) = f\left(\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right),$$
$$y = f\left(a^T.z\right) = f\left(\begin{pmatrix} 2 & 2 \end{pmatrix} \times \begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix}\right) \quad (1)$$

where $f(\cdot)$ is the activation function of neurons. It has already been shown by authors that systems with such input–output relations can be easily implemented by memristor crossbar structures [11]. In fact, the memristor crossbar structure can be used to calculate the matrix–vector product expressions by adjusting the memristance of memristors (which are located at cross-points) to suitable values. For example, the network of Fig. 2a can be

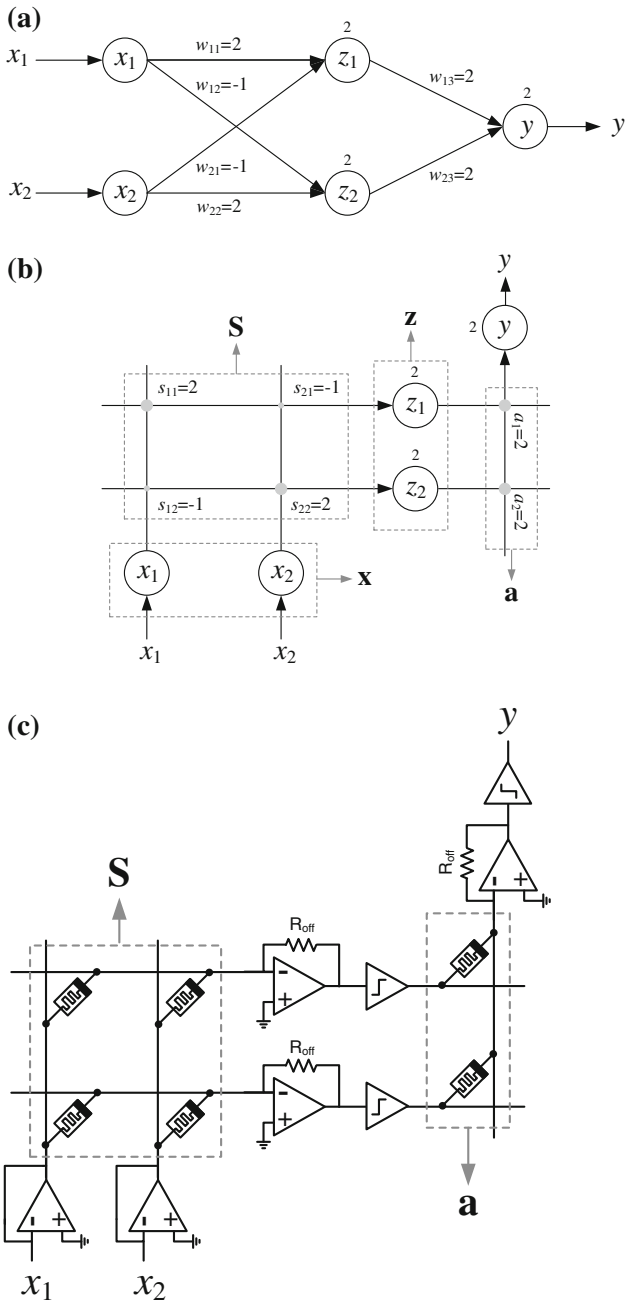


Fig. 2 **a** The artificial neural network proposed by McCulloch–Pitts for implementation of the logical XOR function. **b** Realization of the network shown in **a** by means of nano-crossbar structures. **c** Hardware implementation of the system shown in **b** by using memristor crossbars to model connection weights

implemented by using two memristor crossbars as shown in Fig. 2b. In this figure, value of the signal entered to neuron Z_1 (as well as the signal entered to the neuron Z_2) is equal to the sum of the product of the signals on vertical wires by the weights written on cross-points. Similarly, value of the signal entered to neuron y is equal to the sum of the product of the signals on horizontal wires by the weights written on

cross-points (note that in general the weights are not equal to the memristance value of memristors). Note that although the network shown in Fig. 2a can be easily implemented by using conventional digital gates, it has the advantage that by some modifications can achieve the learning capability. The memristive computing system shown in Fig. 2b (regardless of the hard thresholding function applied to neurons) can be implemented by using op-amp voltage summers as shown in Fig. 2c.

In the rest of this paper we can follow the discussions without the need to know how exactly the memristor crossbar performs the matrix product. See [11] for more details on the procedure of construction of synaptic weights through the use of memristor.

According to the above discussions, the input–output relation of the memristive system shown in Fig. 2b can be expressed as $y = x_1 \text{XOR } x_2$. Hence, the edges of the given black and white image can be detected by applying the value of two adjacent pixels to the inputs of this system. More precisely, in the case that both of the adjacent pixels are either black or white (corresponding to the case $x_1 = x_2 = 1$ or $x_1 = x_2 = 0$), we have $y = 0$, which means that no edge exists between two pixels. But in the case that a black pixel is located at the vicinity of a white pixel (corresponding to the case $x_1 = 1$ and $x_1 = 2$, or $x_1 = 0$ and $x_2 = 1$) we have $y = 1$, which shows the existence of an edge between two pixels. Unfortunately, this approach cannot be directly used to detect the edges of a grayscale picture since in that case the value of each pixel is a number between 0 and 255. In the next section we will modify the memristive system shown in Fig. 2b to arrive at a fuzzy XOR function with the ability of generating a meaningful result for the XOR of two pixels of a grayscale image.

4 Fuzzy XOR function and its memristor crossbar-based hardware implementation

In grayscale pictures each pixel of the image has a number between 0 and 255. Let x_1 and x_2 stand for the value of two adjacent pixels in the grayscale picture under consideration. A simple but reasonable approach to detect the edges in such a picture is to calculate the XOR of x_1 and x_2 in a fuzzy manner. For example, when x_1 is big (i.e., it is close to 255) and x_2 is small (i.e., it is close to 0) the confidence degree about the existence of an edge, denoted as y, should be big (e.g., close to 1 when the maximum value of confidence degree is equal to unity). Such an XOR fuzzy function can be constructed by designing a fuzzy rule-based inference system with the following fuzzy rules in the rule base:

IF x_1 is small AND x_2 is small THEN y is small

IF x_1 is small AND x_2 is big THEN y is big

IF x_1 is big AND x_2 is small THEN y is big

IF x_1 is big AND x_2 is big THEN y is small.

The aim of the following discussions in this section is to propose a method to construct a memristive computing system which can act as a fuzzy inference system with the above-mentioned rules.

In order to construct the fuzzy XOR function under consideration by means of memristor crossbars first we note that the input–output relation of the classical network shown in Fig. 2a can be expressed as:

$$y = x_1\bar{x}_2 + \bar{x}_1 x_2 \tag{2}$$

which is in the *sum of product* form. Note that since any Boolean function can be expressed in this form through logical minterms, the following discussions are quite general and can be used to construct the fuzzy logic equivalent expression of any Boolean function that is in the form of sum of products.

In dealing with grayscale pictures we interpret the logical expression given in (2) in a fuzzy manner. For this purpose first we define $\bar{x}_1$ and $\bar{x}_2$ as $\bar{x}_1 = 255 - x_1$ and $\bar{x}_2 = 255 - x_2$, respectively, and then we fuzzify each of the crisp numbers x_1, x_2, $\bar{x}_1$ and $\bar{x}_2$. Then, we substitute the logical AND and logical OR in (2) with fuzzy t-norm and fuzzy s-norm operators, respectively. So, the novelty of this work is mainly in the method proposed for implementation of fuzzy AND and fuzzy OR by means of memristor crossbar structure.

In brief, implementation of (2) in a fuzzy manner consists of three major parts: Fuzzification of the input variables to find an equivalent fuzzy expression for x_1 and x_2 (as well as $\bar{x}_1$ and $\bar{x}_2$), calculation of the *fuzzy minterms* $x_1\bar{x}_2$ and $\bar{x}_1 x_2$ by computing the t-norm of x_1 and $\bar{x}_2$ (or, x_2 and $\bar{x}_1$), and finally calculation of $x_1\bar{x}_2 + \bar{x}_1 x_2$ by computing the s-norm of the fuzzy expressions $x_1\bar{x}_2$ and $\bar{x}_1 x_2$. Figure 3 shows the proposed structure for the construction of (2) in fuzzy format when x_1 and x_2 are integer numbers between 0 and 255. Clearly, this structure can be thought of as a neuro-fuzzy memristive computing system. In the following we discuss about each of the above-mentioned three parts in more detail, and show that when in the system of Fig. 3 the value of two inputs is close to each other the output of system, y_1, is *small* and when they are very different this output is *big*.

4.1 The fuzzification unit

Inputs of the fuzzy edge detector system under consideration are crisp integer numbers in the range [0, 255]. Hence, the first step to interpret (2) in a fuzzy manner is to fuzzify these input numbers. The fuzzification unit in Fig. 3 shows the proposed structure for this purpose. Inputs of this system are four crisp integer numbers denoted as

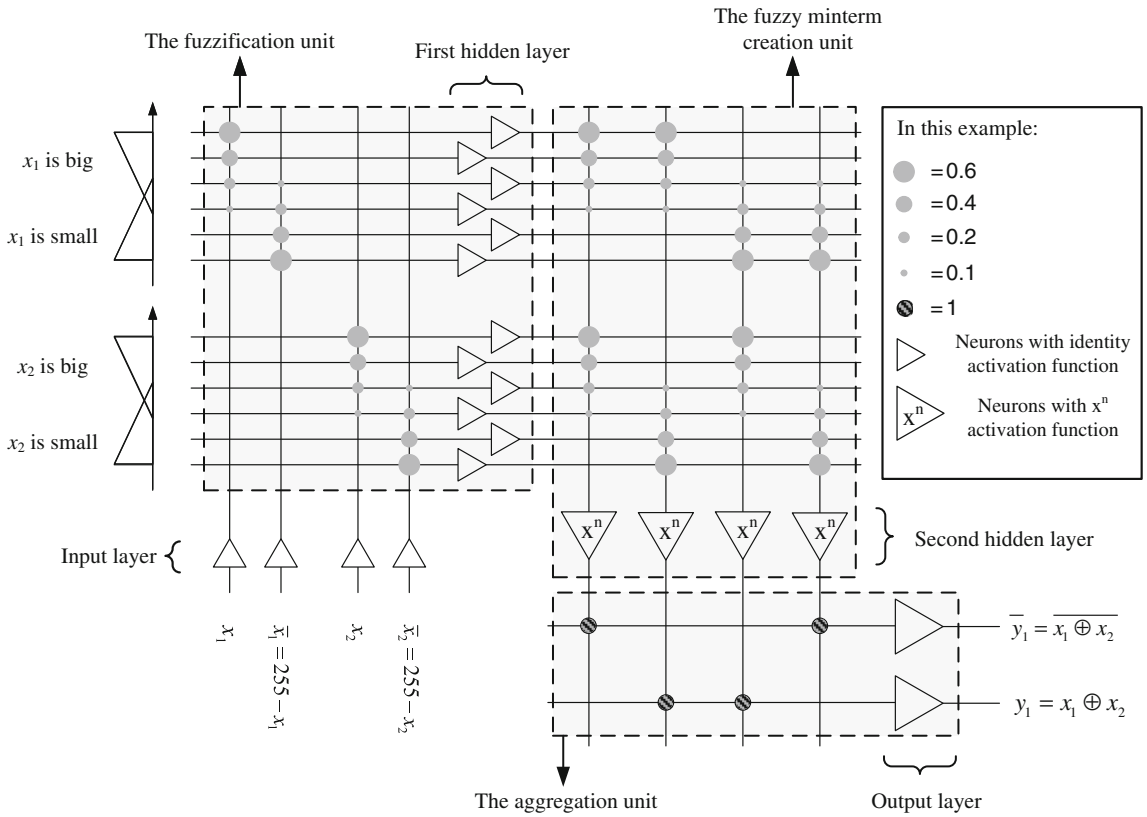


Fig. 3 Proposed structure for memristor crossbar-based hardware implementation of the fuzzy XOR function

x_1, $\bar{x}_1 = 255 - x_1$, x_2 and $\bar{x}_2 = 255 - x_2$. In this figure the first two inputs represent the concepts "brightness of the first input pixel" and "darkness of the first input pixel", respectively, and the values that we apply to them show the strength of the occurrence of these concepts. A similar discussion can be made on the second two inputs. These inputs are passed through voltage buffers to isolate memristors of the crossbar from the circuits that possibly exist at the previous level (note that since in the crossbar structure the weights are stored in the value of memristors located at cross-points, no considerable electrical current should pass through the memristors to avoid changing their value).

In the memristor crossbar structure used in the fuzzification unit, the first two vertical wires (from left) somehow act as a universe of discourse of variable x_1 while the next two vertical wires represent the universe of discourse of x_2. In this case, the configuration of the preprogrammed connection weights on the first (last) two vertical wires specifies the shape of the membership functions defined on the universe of discourse of input variable x_1 (x_2). In Fig. 3, the value of these connection weights (realized through the memristance of memristors) at each cross-point is denoted as a circle, such that larger the circle larger the corresponding weight. Clearly, by suitable choice of the memristance value of these memristors, any desired membership function can be generated. In the typical structure presented in Fig. 3, the shapes of membership functions used for the fuzzification of input variables are shown in the left side of the fuzzification unit. Note that by increasing the number of horizontal wires smoother membership functions can be generated.

Each triangle inside the fuzzification unit is actually a summer realized by a simple Op-Amp circuit (see [11] for more details). Hence, these triangles have the property that add those membership functions which are programmed on columns of the crossbar with different gains where these gains are equal to the inputs applied to columns of the crossbar. As a result, for example, on the upper (lower) six horizontal lines at the output of fuzzification unit we will have a membership function of the fuzzified input number x_1 (x_2). These lines also show our confidence degree about x_1 and $\bar{x}_1$, that is when in the next unit we need to deal with the concept "*considerably* x_1" we use the signal at the most upper horizontal line of the fuzzification unit, and when we need to deal with the concept "*considerably* $\bar{x}_1$" we use the signal at the sixth horizontal line (from the top) of the fuzzification unit. It concludes that, we use, for example, the signal on the second horizontal line of the fuzzification unit (from top) when we want to deal with, e.g., the concept "*almost* x_1" (or equivalently, "*slightly* $\bar{x}_1$"). Similarly, x_2 and $\bar{x}_2$ can be interpreted in a fuzzy manner by using the lower six horizontal lines of the fuzzification unit.

4.2 The fuzzy minterm creation unit

The aim of using the *fuzzy minterm creation unit* in Fig. 3 is to create and evaluate $x_1\bar{x}_2$ and $\bar{x}_1x_2$ when x_1 and x_2 are fuzzy expressions as mentioned above. For this purpose, this unit performs the dot product between the fuzzy inputs (which are actually the outputs of the first hidden layer) and the weight vectors formed on columns of the crossbar located inside the fuzzy minterm creation unit. Therefore, this unit somehow measures the similarities between the input numbers and preprogrammed weights on columns of the crossbar. To make it more clear, consider the first (the leftmost) column of the crossbar located inside this unit. The output of its corresponding neuron (connected to the bottom of this column) is maximized only when both of the input variables, x_1 and x_2, are equal to their maximum values and therefore their corresponding fuzzy numbers (created by the fuzzification unit) are similar to the pre-programmed patterns on the first column (note that in this structure both of the input variables and all of the weight vectors are always non-negative). It concludes the fact that this column of the crossbar implements the antecedent part of the following fuzzy rule:

IF x_1 is big AND x_2 is big THEN $x_1 \oplus x_2$ is small

and the output of the corresponding neuron for any given input data shows the result of its evaluation. In a similar way, it can be shown that the second column of the crossbar (from left) implements the antecedent part of the following fuzzy rule:

IF x_1 is big AND x_2 is small THEN $x_1 \oplus x_2$ is big

and the output of the corresponding neuron is equal to the result of this evaluation. Note that in general for any given input data all outputs of the fuzzy minterm creation unit and consequently their corresponding fuzzy rules are active to a certain degree.

The last point in relation to the fuzzy minterm creation unit is that the activation function of output neurons is considered as x^n $(n > 1)$ to magnify the difference between outputs of the neurons of this layer. It can be easily verified that such an activation function somehow emulates the role of the fuzzy AND operation between two parts of the antecedent of fuzzy rules. Note that this activation function is advantageous in a way that is free of parameters such as the threshold value, which should be determined very carefully.

4.3 The aggregation unit

The last step to construct the fuzzy XOR function under consideration is to calculate $x_1\bar{x}_2 + \bar{x}_1x_2$ based on the values obtained for $x_1\bar{x}_2$ and $\bar{x}_1x_2$ by the fuzzy minterm creation unit. This task is performed by the so-called *aggregation unit*, which simply adds the appropriate outputs of the second hidden layer together. In fact, in this unit outputs of those neurons of the second hidden layer, which correspond to fuzzy rules with the same consequent part, are summed. For example, the second and the third output of the fuzzy minterm creation unit in Fig. 3 (from left) correspond to the antecedent part of the fuzzy rules:

IF x_1 is big AND x_2 is small THEN $x_1 \oplus x_2$ is big

and

IF x_1 is small AND x_2 is big THEN $x_1 \oplus x_2$ is big,

respectively. Since these two rules have a same consequent part, at the output layer we have added their corresponding outputs. Clearly, in this manner the number of outputs of the aggregation unit will be equal to the number of fuzzy rules with different consequent part. To sum up, when in the system of Fig. 3 the values of x_1 and x_2 are significantly different, the value obtained for y_1 will be much bigger than the value obtained for $\bar{y}_1$, and consequently, it can be considered as the final output of the system. As another example, when both of x_1 and x_2 have high values, the value obtained for y_1 (although it may not be negligible by itself) will be much lower than the value obtained for $\bar{y}_1$.

Note that our proposed inference system has some differences with other typical inference engines. First of all, we have used the algebraic sum to aggregate the fuzzy rules. Second, since our primary goal was to construct a system with fuzzy input and fuzzy output, no defuzzification is intended at the system output. Consequently, as mentioned before, at the output of our proposed system instead of one simple crisp number there will be one output per each distinct consequence part.

In the next section, we will describe how this circuit can be used to extract the edges from any given grayscale image.

5 Application of the proposed fuzzy XOR function for edge detection in grayscale images

In this section we show that the proposed fuzzy XOR gate can be developed to arrive at a system with the ability of extracting the possible edges from grayscale images, but before that we briefly review the application of classical XOR gate for edge detection in binary (black and white) images.

Obviously, in black and white images the possible edges are located between two pixels with different intensities. More precisely, wherever a black pixel (with low intensity) is located next to a white pixel (with high intensity) an edge is located between them. Otherwise, no edge exists between two neighbor pixels when both of them have the same pixel value. It concludes the fact that in binary images the possible edges can be detected by applying the

XOR function to neighbor pixels: when these pixels have different intensities, the output of the logical XOR function is high (logic 1) which indicates the existence of an edge.

Now, consider the case in which we want to detect the edges in a grayscale image. Clearly, such a task can be performed by applying a kind of modified XOR function to neighbor pixels of the image. This modified XOR function should accept continuous values instead of binary numbers. More precisely, the output of this XOR function must be close to zero when the input pixels have almost similar intensities and its output must increase as the difference between the intensity of input pixels increases. From the mathematical point of view, the output of the XOR function under consideration must be a monotonically increasing function of the difference between the intensity of input pixels. Clearly, by applying such an XOR function to consecutive pixels of the given image, sharper edges will lead to higher values and vice versa. A possible approach to construct this XOR function is shown in Fig. 4. One main advantage of this circuit is that it can extract all horizontal or vertical edges of the given image simultaneously. This ability is due to the fact that one can apply several number of these fuzzy XOR gates at the same time.

The function of the system shown in Fig. 4 can be explained exactly based on the discussions presented in previous section. As it can be observed, at the output layer the system generates one output signal per each pair of consecutive pixels. It can be easily verified that the output signals y_1 and y_2 show the result of applying the fuzzy XOR function to the intensity of corresponding adjacent pixels. By adding more fuzzy XOR gates to this circuit in the same manner, we can construct a structure with the ability of extracting all vertical edges (in one row of the given image) simultaneously. By applying similar circuits to other rows of the image, all vertical edges in the entire image can be extracted. At the same time, by rotating the image by 90° and repeating the same procedure, horizontal edges can be extracted as well. Note that since the circuit shown in Fig. 4 (as well as the circuit shown in Fig. 3) is in analog form, it can potentially be used for detecting the edges in real-time applications. Note also that since the memristors can be made extremely small, the overall circuit can be implemented in the form of a small chip.

6 Simulation results

In this section, we will illustrate the efficiency and applicability of our proposed method for detecting the edges in grayscale pictures by performing several numerical simulations.

In all of the following simulations, the structure of Fig. 4 with the same preprogrammed membership functions is used where their numerical specifications are given in the inset of the figure. In addition, pixel values of output images are mapped into interval [0, 255] before presenting in this paper which can also be done in the proposed hardware by the modification of system's specifications and parameters. Moreover, to remove probable noises in input images, all inputs are smoothed with the Gaussian smoothing filter before being applied to the system. Finally, the proposed structure is simulated in HSPICE software (version Z-2007.03) by using the SPICE model given in [20] for memristors while its parameters are set as follows: $\mu_v = 10^{-14}$ m^2 s^{-1} V^{-1}, $D = 10^{-8}$ m, $R_{\text{on}} = 1$ KΩ and $R_{\text{off}} = 100$ KΩ.

The results of the first conducted simulation are presented in Fig. 5. In this simulation, the image shown in Fig. 5a is applied to the system of Fig. 4 as an input and extracted horizontal and vertical edges are presented in Fig. 5b and c, respectively. By merging (here adding) these two images, one single image as shown in Fig. 5d can be obtained as a final result of our proposed edge detection algorithm. Note that since outputs of the system of Fig. 4 are always non-negative, extracted horizontal and vertical edges can be summed directly. Figure 5d indicates that output images of this structure all have this property that their intensities at any point are directly proportional to the strength of the existing edges at that point in the original input images.

In order to have better comparison between the performances of the proposed method and other state-of-the-art edge detection algorithms, the result of the first two steps of the canny edge detection algorithm [21], i.e., smoothing and finding gradients, applied to the image of Fig. 5a is shown in Fig. 5e. By comparing Fig. 5d with Fig. 5e, it can be inferred that although the input image is smoothed, our structure has produced sharper edges. In addition, especially in those areas of Fig. 5a where the intensity of the image is almost uniform, unwanted detected edges are abundant and visible in Fig. 5e which is not the case in Fig. 5d. Finally, Fig. 5f shows the fuzzy inference surface (input–output relation) of the system of Fig. 3. In this figure, the overall behavior and shape of the fuzzy XOR function is clearly observable: in those areas where input variables have similar values (pixels have similar intensities), output of the system is near to zero. However, when the difference between inputs increases, the output of the system begins to approach its upper bound, i.e., 255. Although Fig. 5f is obtained by using the x^2 activation function for neurons of the second hidden layer of the circuit of Fig. 4, our experiments showed that using other similar activation functions like x^4 or x^7 has little impact on the shape of this fuzzy inference surface. It is also interesting to note that by the use of Mamdani's fuzzy inference method [22] or Takagi–Sugeno–Kang method of fuzzy

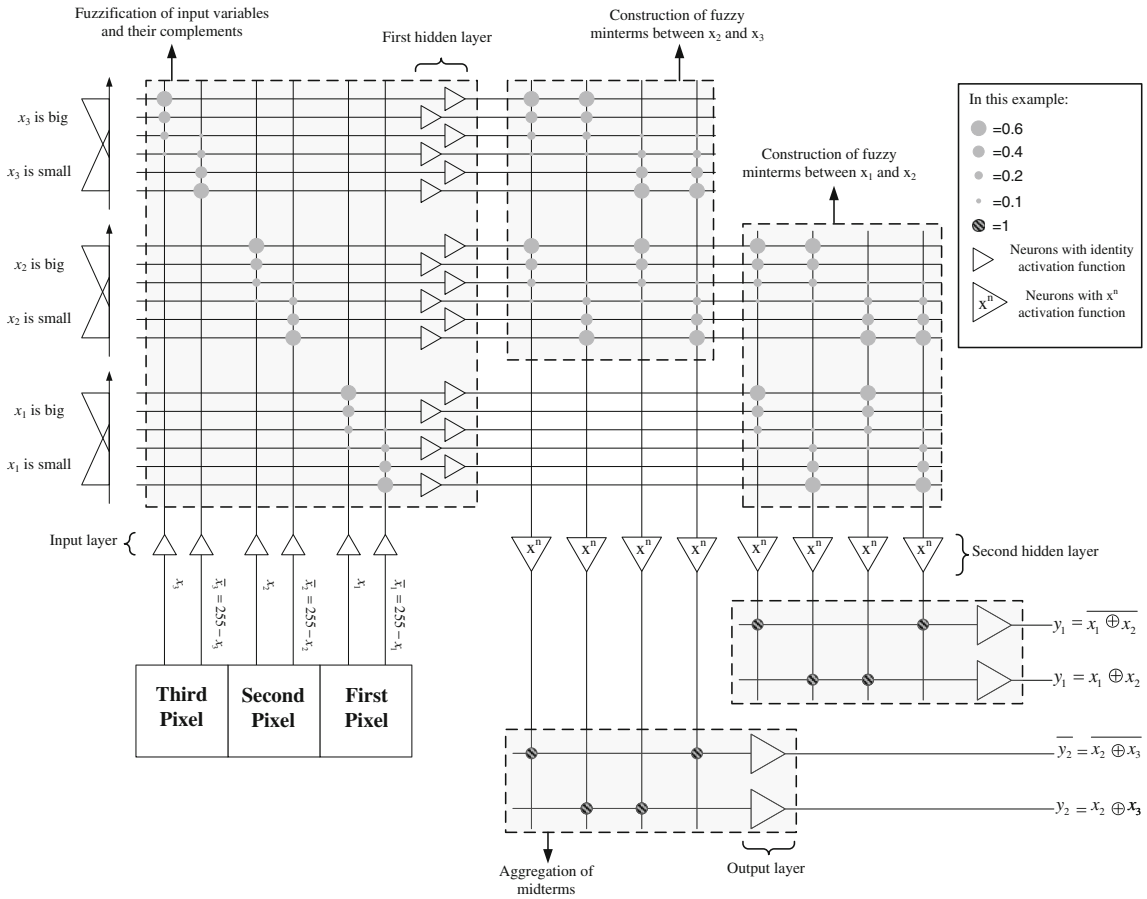


Fig. 4 The circuit that can extract the edges from three consecutive pixels of the given image

inference [23, 24] with the same number of rules (here 4 rules), it is almost impossible to create such a smooth surface for the fuzzy XOR function.

In the next simulation, we used two other images as an input and applied our proposed structure to extract edges from them. These input images are shown in Fig. 6a and g. The extracted edges from these figures by using our method and structure are presented in Fig. 6b and h while the result of applying the first two steps of the canny edge detection algorithm to them are demonstrated in Fig. 6c and i. To illustrate the stability and performance of our memristive neuro-fuzzy system in noisy environment, detected edges from noisy images of Fig. 6d and j which are obtained by adding Gaussian white noise of mean 0 and variance 0.03 to input images are shown in Fig. 6e and k, respectively. Again to have better comparison, the results of applying the smoothing and finding gradients steps of the canny edge detection algorithm to these noisy images are presented in Fig. 6f and l. From the result of this simulation, it can be inferred that although our proposed edge detection method only uses information of two neighbor pixels, it performs acceptably against noisy images.

7 Discussion

In this section we discuss on some important aspects of the proposed structure for edge detection in grayscale images. Note that since at this time the memristor crossbars are constructed only in few labs, we cannot experimentally verify the results presented in previous sections and the following discussions are based only on the information provided in the seminal papers [5] and [6].

One important fact in relation to memristor crossbars is that these structures can be made extremely small. More precisely, it is believed that the wires and switches of memristor crossbars will eventually get down to a width of around 4 nm. So, potentially, it is possible to construct the proposed memristive fuzzy edge detector in a very small area even in dealing with very high resolution images.

The second important observation is that memristors are very stable. That is, when a data is stored in a memristor it remains unchanged at least for several years, unless a sufficiently large electrical current passes through it. Thus it can be concluded that in the proposed structure the shape of the membership functions programmed in the crossbar

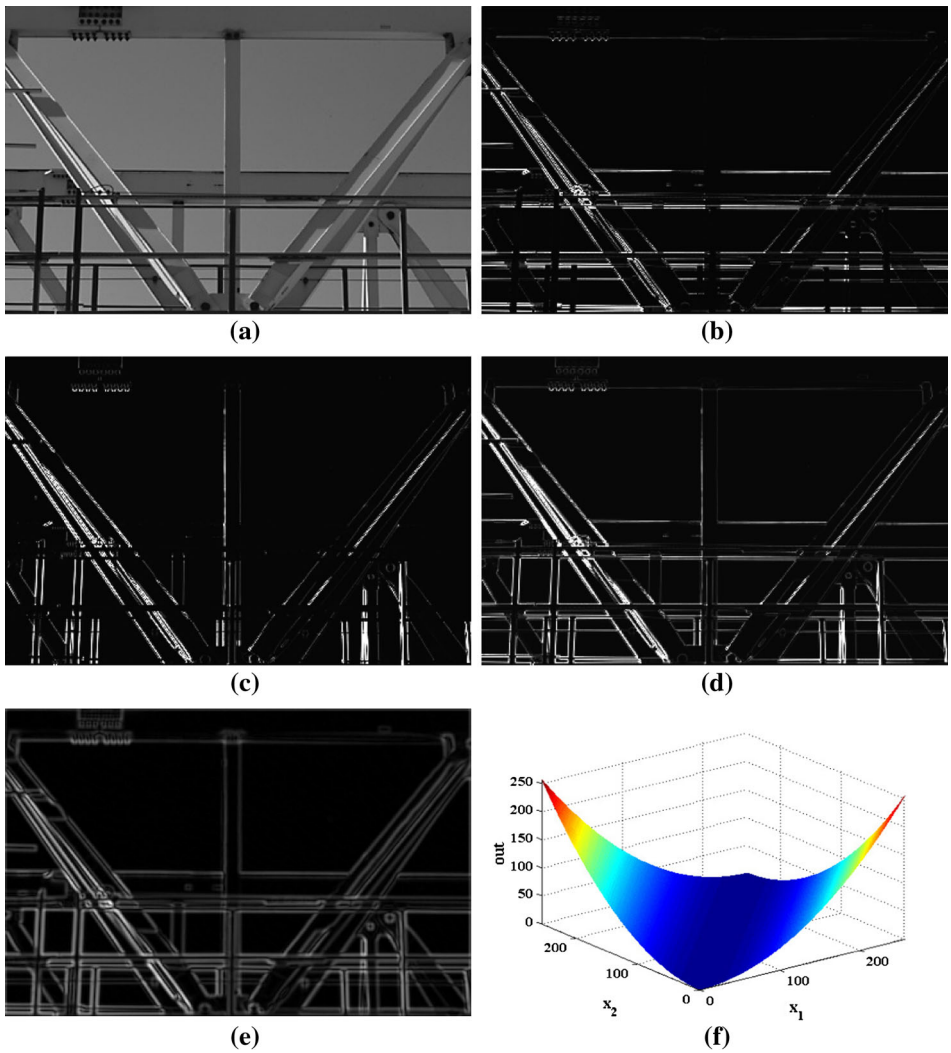


Fig. 5 Simulation results of the first conducted experiment. **a** Input image. **b** Extracted horizontal edges by using our method. **c** Extracted vertical edges by our proposed system. **d** Final output of our memristive edge detecting system. **e** Extracted edges by applying the first two steps of the canny edge detection algorithm. **f** Fuzzy inference surface of the system of Fig. 3

can last for a long period of time. Note that in the proposed structure an extremely small electrical current passes through memristors which cannot change their value.

Another point in relation to the proposed structure is that the pixels of the given image can be fed to it without any problem as they are currently fed to classical image processing systems: read the pixels from the memory or directly from the sensors and apply them to inputs of the proposed structure as voltage signals. The proposed structure, however, has the advantage that can directly deal with the analog data obtained from optical sensors. As a well-known fact, the output of optical sensors (such as CCDs) is actually in analog form, which is stored in digital form in the memory. This conversion from analog to digital is time-consuming and, of course, needs additional hardware, which increases the cost of implementation and also puts a limitation on the performance of the system (due to, e.g., data quantization and band-width limitations). The proposed system is, however, advantageous in the way that it can directly deal with the original analog data and consequently, no analog/digital data conversion is needed in this case. An interesting discussion on the possibility and advantages of construction of FPGAs by means of memristors can be found in [25]. Finally, it should be noted that since the proposed method can be used to implement any other mathematical proposition in the form of fuzzy IF–THEN rules, therefore it can be simply extended such that it considers other surrounding pixels to detect the edges with a higher accuracy. For this purpose it is only necessary to change fuzzy rules which can be considered as our future work.

8 Conclusion

In this paper, we proposed a new structure for memristor crossbar-based hardware implementation of an edge detection algorithm. The proposed structure, which is applicable to all grayscale images, works based on the calculation of the fuzzy XOR of two adjacent pixels of the

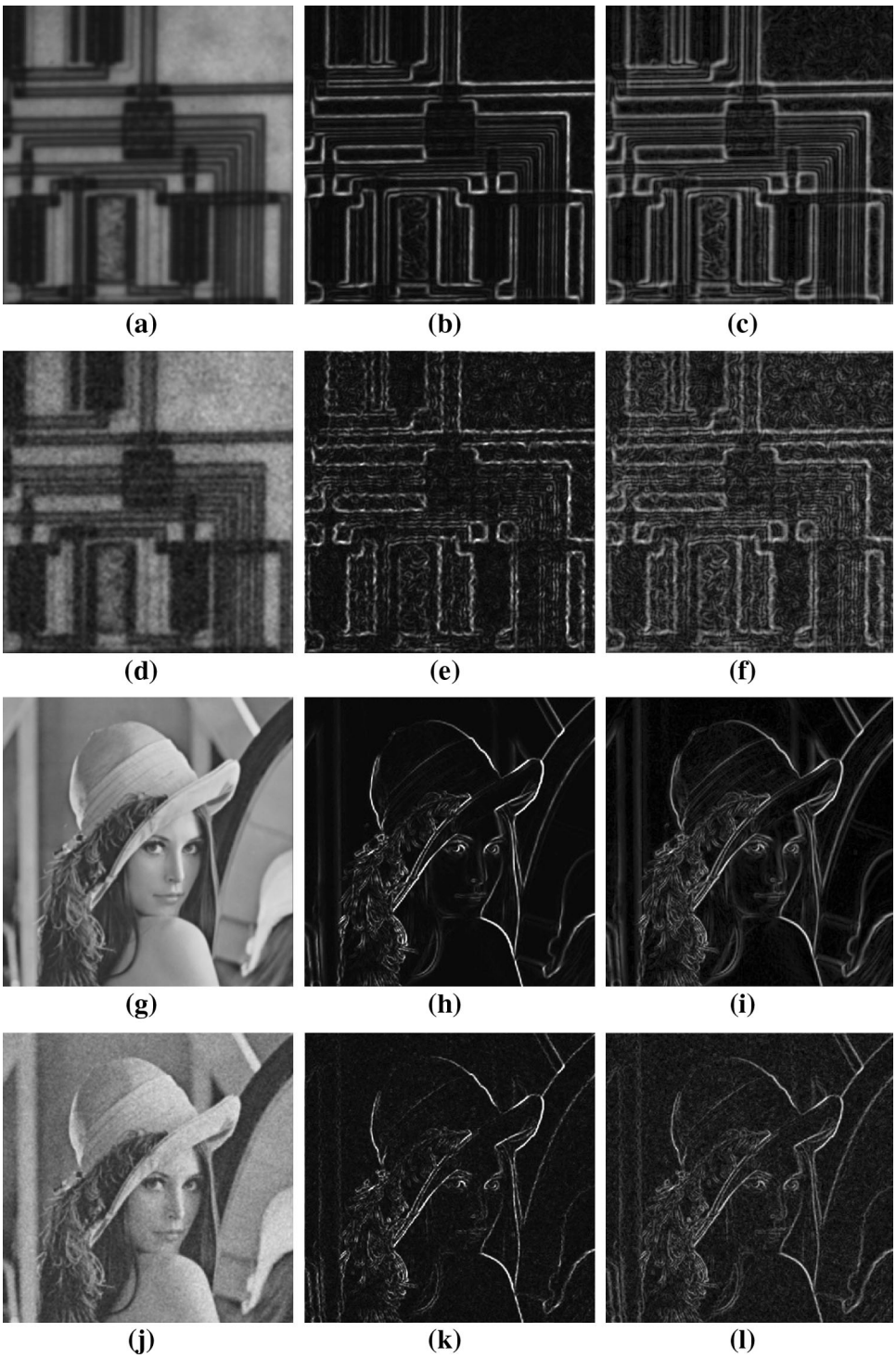


Fig. 6 Simulation results of the second conducted experiment. **a** and **g** Input images. **b** and **h** Extracted edges by using our proposed structure. **c** and **i** Extracted edges by applying the first two steps of the canny edge detection algorithm. **d** and **j** Input images degraded by Gaussian noise. **e** and **k** Extracted edges from noisy inputs by using our proposed structure. **f** and **l** Extracted edges from noisy inputs by applying the first two steps of the canny edge detection algorithm

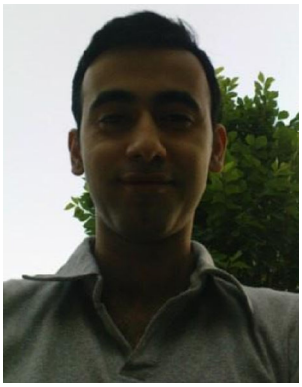
given grayscale image. One important advantage of the proposed structure is that it can directly work with the analog data obtained from optical sensors. Hence, no analog/digital data conversion is necessary in this case and no data quantization occurs. Simulation results show that the proposed structure can effectively extract both vertical and horizontal edges of the given image even in noisy environments. Finally, note that this paper presents the basic ideas; and clearly, additional research is needed to arrive at a practical system.

Acknowledgments The authors thank the anonymous reviewers for suggesting many useful comments to improve the quality of this work.

References

1. Versace, M., Chandler, B.: The brain of a new machine. IEEE Spectr. **47**(12), 30–37 (2010)
2. Cioffi, J.M.: Limited-precision effects in adaptive filtering. IEEE Trans. Circuits. Syst. **CAS-34**(7), 821–833 (1987)
3. Haykin, S.: Adaptive Filter Theory, second edn. Prentice-Hall, Englewood Cliffs (1991)
4. Treichler, J.R., Johnson, C.R., Larimore, M.G.: Theory and Design of Adaptive Filters. Wiley, New York (1987)
5. Williams, R.S.: How we found the missing memristor. IEEE Spectr. **45**(12), 28–35 (2008)
6. Strukov, D.B., Snider, G.S., Stewart, D.R., Williams, R.S.: The missing memristor found. Nature **453**, 80–83 (2008)
7. Chua, L.O.: Memristor—the missing circuit element. IEEE Trans. Circuit Theory **CT-18**(5), 507–519 (1971)
8. Pershin, Y.V., Fontaine, S.L., Ventra, M.D.: Memristive model of amoeba's learning. Phys. Rev. E **80**, 021926 (2009)
9. Pershin, Y.V., Ventra, M.D.: Practical approach to programmable analog circuits with memristors. IEEE Trans. Circuits Syst. I: Regular Paper **57**(8), 1857–1864 (2010)
10. Merrikh-bayat, F., Shouraki, S.B.: Memristorbased circuits for performing basic arithmetic operations. Procedia Comput. Sci. J. **3**, 128–132 (2011)
11. Merrikh-bayat, F., Shouraki, S.B.: Memristor crossbar-based hardware implementation of IDS method. IEEE Trans. Fuzzy Syst. **19**(6), 1083–1096 (2011)
12. Kuekes, P.: Material implication: digital logic with memristors. In: Memristor and Memristive Systems Symposium, 21 November 2008
13. Mouttet, B.L.: Proposal for memristors in signal processing. In: Nano-Net Conference, vol. 3, pp. 11–13, Sept 2008
14. Merrikh-Bayat, F., Shouraki, S.B.: Mixed analog–digital crossbar-based hardware implementation of sign–sign LMS adaptive filter. Analog Integr. Circuits Signal Process **3**(1), 41–48 (2011)
15. Snider, G., Amerson, R., Carter, D., Abdalla, H., Qureshi, M.S., Leveille, J., Versace, M., Ames, H., Patrick, S., Chandler, B., Gorchetchnikov, A., Mingolla, E.: From synapses to circuitry: using memristive memory to explore the electronic brain. IEEE Comput. **44**(2), 21–28 (2011)
16. Afifi, A., Ayatollahi, A., Raissi, F.: Implementation of biologically plausible spiking neural network models on the memristor crossbar-based CMOS/nano circuits. In: European Conference on Circuit Theory and Design (ECCTD 2009), pp. 563–566 (2009)
17. Snider, G.S.: Spike-timing-dependent learning in memristive nanodevices. In: IEEE International Symposium on Nanoscale Architectures (NANOARCH 2008), pp. 85–92, 12–13 June 2008
18. McCulloch, W., Pitts, W.: A logical calculus of ideas immanent in nervous activity. Bull. Math. Biophys. **5**, 115–133 (1943)
19. Fausett, L.: Neural Networks: Architectures, Algorithms and Applications. Prentice Hall, Englewood Cliffs (1994)
20. Biolek, D., Biolek, Z., Biolkova, V.: SPICE modeling of memristive, memcapacitative and meminductive systems. In: European Conference on Circuit Theory and Design (ECCTD2009), pp. 249–252, Antalya, 23–27 August 2009
21. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. **8**(6), 679–698 (1986)
22. Mamdani, E.H., Assilian, S.: An experiment in linguistic synthesis with a fuzzy logic controller. Int. J. Man Mach. Stud. **7**(1), 1–13 (1975)
23. Sugeno, M., Kang, G.T.: Structure identification of fuzzy model. Fuzzy Sets Syst. **28**, 15–33 (1988)
24. Takagi, T., Sugeno, M.: Fuzzy Identification of systems and its applications to modeling and control. IEEE Trans. Syst. Man Cybern. **15**, 116–132 (1985)
25. Snider, G., Williams, R.S.: Nano/CMOS architectures using a field-programmable nanowire interconnect. Nanotechnology **18**(3), 035204 (2007)

Author Biographies



Farnood Merrikh-Bayat received his B.Sc. degree in Electrical Engineering in 2005 from Khaje Nasir Toosi University of Technology, Iran, and M.Sc. degree in Digital Electronic from Department of Electrical Engineering, Sharif University of Technology, Iran, in 2007. He is now a Ph.D. candidate at the Sharif University of Technology. His research interests include memristor, memristive neuro-fuzzy systems, thinking chips, learning algorithms and brain emulation based on memristive crossbar structures.

Saeed Bagheri Shouraki received his B.Sc. in Electrical Engineering and M.Sc. in Digital Electronics from Sharif University of Technology, Tehran, Iran, in 1985 and 1987. He joined soon the Computer Engineering Department of Sharif University of Technology as a faculty member. He received his Ph.D. on fuzzy control systems from Tsushin Daigaku (University of Electro-Communications), Tokyo, Japan, in 2000. He continued his activities in Computer Engineering Department up to 2008. He is currently an associate professor in Electrical Engineering Department at Sharif University of Technology. His research interests include control, robotics, artificial life, and soft computing.

Farshad Merrikh-Bayat received his Ph.D. in electrical engineering (control) from the Sharif University of Technology, Tehran, Iran, in 2009. He is the author of 5 books, 10 journal and 11 conference papers. His research interests include fuzzy control systems, meta-heuristic optimization algorithms, nonlinear dynamics and control, and fractional-order PID controllers.