Memristor Bridge Synapse-Based Neural Network and Its Learning

Shyam Prasad Adhikari, Changju Yang, Hyongsuk Kim, *Member, IEEE*, and Leon O. Chua, *Fellow, IEEE*

***Abstract*— Analog hardware architecture of a memristor bridge synapse-based multilayer neural network and its learning scheme is proposed. The use of memristor bridge synapse in the proposed architecture solves one of the major problems, regarding nonvolatile weight storage in analog neural network implementations. To compensate for the spatial nonuniformity and nonideal response of the memristor bridge synapse, a modified chip-in-the-loop learning scheme suitable for the proposed neural network architecture is also proposed. In the proposed method, the initial learning is conducted in software, and the behavior of the software-trained network is learned by the hardware network by learning each of the single-layered neurons of the network independently. The forward calculation of the single-layered neuron learning is implemented on circuit hardware, and followed by a weight updating phase assisted by a host computer. Unlike conventional chip-in-the-loop learning, the need for the readout of synaptic weights for calculating weight updates in each epoch is eliminated by virtue of the memristor bridge synapse and the proposed learning scheme. The hardware architecture along with the successful implementation of proposed learning on a three-bit parity network, and on a car detection network is also presented.**

***Index Terms*— Chip-in-the-loop, memristor, memristor bridge synapse, neural network.**

I. Introduction

ARTIFICIAL neural networks are among the most effective learning methods currently known for learning, to interpret many complex real-world data. Out of many neural network learning rules, the back-propagation algorithm has been used successfully in many practical problems, such as speech recognition [1], handwritten character recognition [2], face recognition [3], robot control [4], [5], etc. An essential step in applying neural networks to real-life problems is their implementation in hardware. A large variety of hardware implementations of neural networks using analog and digital electronics, optics and hybrid techniques are found in the literature [6]–[9]. A survey of different hardware implementations can be found in [10]–[13].

Manuscript received September 21, 2011; revised April 30, 2012 and June 7, 2012; accepted June 10, 2012. Date of publication July 5, 2012; date of current version August 1, 2012. This work was supported in part by the National Research Foundation of Korea (NRF) funded by the Korean Government (MEST) under Grant 2010-0006871 and the U.S. Air Force Grant FA9550-10-1-0290.

S. P. Adhikari, C. Yang, and H. Kim (corresponding author) are with the Division of Electronics Engineering, Chonbuk National University, Jeonju 561-756, Korea (e-mail: hskim@jbnu.ac.kr).

L. O. Chua is with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720-1770 USA.

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

Digital Object Identifier 10.1109/TNNLS.2012.2204770

The success of neural network hardware design depends largely on the tradeoffs between accuracy, chip area, and processing speed. A high degree of accuracy can be achieved with digital implementations but this comes at a cost of relatively larger chip area, reduced speed, and increased power consumption. Unlike their digital counterparts, analog implementations are usually more efficient in terms of chip area and processing speed but with limited accuracy that arises due to spatial nonuniformity of analog components and their nonideal responses [14], [15]. In addition, another major bottleneck in analog neural network hardware is the implementation of nonvolatile weight storage [12]. In analog hardware implementations, the weights are usually stored in resistors [16], capacitors [17], and floating gate transistors [7]. The resistors are static and cannot be changed once fabricated, thus can be used only for nonlearning hardware. Capacitors have short synaptic weights retention time due to charge leakage, and require dynamic weight updating at frequent intervals. Floating gate transistors has been used successfully as synapses in conjunction with analog multipliers, but it suffers from high nonlinearity in synaptic weightings.

Another difficulty in implementing multilayer neural networks in hardware is the implementation of learning algorithms. The back-propagation method passes the error signals recursively backward from the output layer to estimate the weight change required in the hidden layers. This is a relatively complex operation to be implemented in electronic circuits, and the complications are amplified by imperfections and mismatch in the circuit components. Techniques, involving weight perturbation [18], [19] and random weight change [20] are shown to be more suitable for hardware implementation but these approaches only estimate the gradient rather than actually computing them.

However, it is considerably easy to implement back-propagation learning in software with desired accuracy. So, the circuit implementation of neural network with chip-in-the-loop [21], [22] is more practical. But, the hardware implementation of multilayer neural networks in chip-in-the-loop still requires the readout of synaptic weights of the succeeding layer, to calculate the weight-update values for the preceding layers.

These are some of the difficulties that have limited progress in the hardware implementations of multilayer neural networks. But, the recent physical realization of "memristor," a nonvolatile variable resistor, has opened up new possibilities in the neural network hardware domain. The existence of "memristor" as the fourth basic circuit element was predicted from theory by Chua [23] in 1971, and its physical realization was achieved recently by researchers at Hewlett Packard

Laboratories [24]. It is now widely known that one of the promising applications of memristors is for implementing artificial neural networks as they act as nonvolatile analog memories, are programmable, and scalable to nano dimensions [24]– [29]. Recently, an artificial synapse consisting of multiple identical memristors in a bridge-like fashion capable of performing signed synaptic weights was proposed in [30] and [31].

In this paper, a neural network hardware architecture using the memristor bridge synapse [30] is proposed to solve the problem of nonvolatile weight storage. This paper also proposes a modified chip-in-the-loop learning method utilizing the inherent advantage of the memristor bridge synapse. The proposed hardware learning method incorporates the hardware nonidealities of the memristor bridge synapse, eliminates the need for readout of synaptic weights in each epoch, and reduces the communication overhead across the network/host interface. Thus, the proposed scheme overcomes the difficulties associated with implementing chip-in-the-loop multilayer neural networks.

Rest of this paper is organized as follows. Memristor bridge synapse and its nonidealities are introduced in Section II. The proposed hardware architecture is presented in Section III, followed by its learning scheme in Section IV. Section V contains the simulation results, followed by concluding remarks in Section VI.

II. Memristor-Based Synapse

A. Memristor Bridge Synapse

Memristor is a two-terminal passive circuit element that maintains a functional relationship between charge and magnetic flux. Memristor acts as a variable resistor whose value can be varied by varying the current passing through it. Since it remembers the amount of current that has passed through it in the past, it can be used as a nonvolatile memory. The resistance R at time t of a memristor, also called memristance M, is defined as

$$R(t) = M(t) = \frac{v(t)}{i(t)} = \left.\frac{d\varphi}{dq}\right|_{(q_Q,\varphi_Q)} \tag{1}$$

where $\varphi(t)$ and $q(t)$ are the flux and charge, respectively. The memresistance can be interpreted as the slope at the operating point $q = q_Q$ at time t on the memristor $\phi - q$ curve. The memristance of a memristor can be controlled by applying a voltage or current signal across it. Immense interest was generated in this area by the recent physical realization of a nanoscale TiO_2 memristor by Stanley Williams Group from Hewlett Packard [24]. In the TiO_2 memristor, a thin un-doped titanium-dioxide layer and an oxygen deficient doped TiO_{2-x} layer are sandwiched between two platinum electrodes. The resistance (memristance) of the memristor depends on the width of the doped area, which can be altered by applying an appropriate current or voltage. The relation between the voltage $v(t)$ and current $i(t)$ is given by

$$v(t) = \left(R_{\rm ON}\frac{w(t)}{D} + R_{\rm OFF}\left(1 - \frac{w(t)}{D}\right)\right) i(t) \tag{2}$$

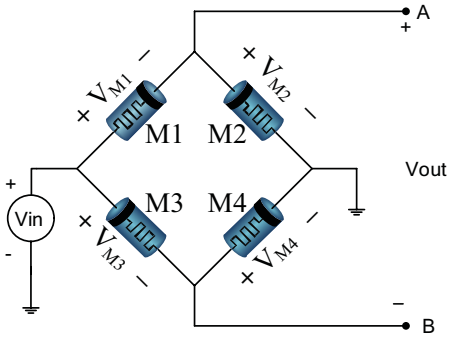


Fig. 1. Memristor bridge circuit. The synaptic weight is programmable by varying the input voltage. The weighting of the input signal is also performed in this circuit.

where $R_{\rm ON}$ is the low resistance value, $R_{\rm OFF}$ is the high resistance value, D is thickness of the sandwiched TiO_2 layer, and $w(t)$ is the thickness of doped area at time t. Recently, a memristor bridge synaptic circuit, shown in Fig. 1, consisting of four identical memristors capable of performing zero, positive, and negative synaptic weightings was proposed in [30]. When a positive or a negative pulse $V_{\rm in}$ is applied at the input, the memristance of each memristor is altered depending on its polarity. By using the voltage divider formula, the output voltage between nodes A and B is given by

$$V_{\rm out} = V_A - V_B = \left(\frac{M_2}{M_1 + M_2} - \frac{M_4}{M_3 + M_4}\right) V_{\rm in}. \tag{3}$$

Equation (3) can be rewritten as a relationship between a synaptic weight ψ and a synaptic input signal $V_{\rm in}$ as follows:

$$V_{\rm out} = \psi \times V_{\rm in} \tag{4}$$

where

$$\psi = \left(\frac{M_2}{M_1 + M_2} - \frac{M_4}{M_3 + M_4}\right).$$

By virtue of (4), this bridge circuit is ideal for implementing nonvolatile weight storage in artificial neural networks. In addition, the bridge circuit acts as a multiplier circuit, which can be used to replace the conventional nonlinear and power-hungry analog multipliers. The synaptic weight programming can be executed by applying strong programming pulses to set the desired synaptic weight. The conditions for the bridge to function as weights in different regimes are listed as follows:

$$\begin{aligned} \text{positive synaptic weight;}\ \text{if}\ \frac{M_2}{M_1} &> \frac{M_4}{M_3} \\ \text{negative synaptic weight;}\ \text{if}\ \frac{M_2}{M_1} &< \frac{M_4}{M_3} \\ \text{zero synaptic weight;}\ \text{if}\ \frac{M_2}{M_1} &= \frac{M_4}{M_3}. \end{aligned} \tag{5}$$

B. Memristor Bridge Nonidealities

In the TiO_2 memristor, the state equation for $w(t)$ is defined as a function of the current $i(t)$; namely

$$\frac{dw(t)}{dt} = \mu_V \frac{R_{\rm ON}}{D} i(t) \tag{6}$$

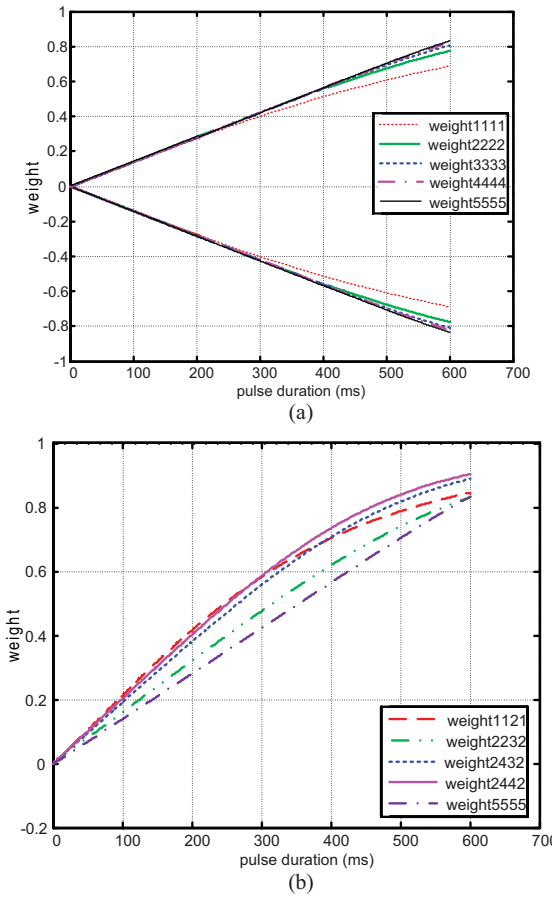


Fig. 2. (a) Positive and negative weight programming after resetting the weight to zero for different values of p with identical memristors in the bridge. (b) Positive weight programming after resetting the weight to zero for different values of p with dissimilar memristors in the bridge. The numbers appended at the end of the legend indicate the value of p for the four memristors in the bridge. The parameters used for simulations are $R_{\rm ON} = 116\ \Omega$, $R_{\rm OFF} = 16\ \text{K}\Omega$, $D = 10$ nm, $\mu_v = 10^{-14}\ \text{m}^2\text{V}^{-1}\text{S}^{-1}$.

where μ_V is the dopant mobility. This model is based on the linear-drift model, since the velocity of the width of the doped region is proportional to the current. This model is an oversimplified model of the TiO_2 memristor. Nonlinear phenomenon often appears at the boundaries of nanoscale devices as strong electric field is produced even when a small voltage is applied. Due to the strong electric field, the drift velocity of the ion boundary is nonlinear. One model of the memristor which accounts for the nonlinear drift is the window model [32] given by

$$\frac{dw(t)}{dt} = \mu_V \frac{R_{\rm ON}}{D} i(t)\, F_p(w) \tag{7}$$

where p is an integer and $F_p(w)$ is defined as

$$F_p(w) = 1 - \left(2\frac{w}{D} - 1\right)^{2p}. \tag{8}$$

The nonlinearity of the window function is inversely proportional to the parameter p. The spatial nonuniformity and mismatch during memristor fabrication can make it difficult to model memristors with a particular value of p. As a result, it becomes difficult to predict the change in memristance of the memristor to a particular applied voltage or current.

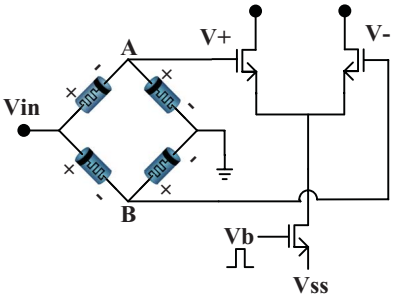


Fig. 3. Memristor bridge synaptic circuit. The memristor bridge on the left performs the weighting operation, while the differential amplifier on the right performs the voltage-to-current conversion.

This nonlinear behavior of the memristor has direct influence on the linearity of the programmed weight in the memristor bridge synapse. The slope of the curve between the weight and the applied charge is linear and equal (for a certain range of weights) if, and only if, all the memristors in the bridge are identical and have similar nonlinear drifts, i.e., p, as shown in Fig. 2(a). However, if the memristors in the bridge synapse have different values for p, then the slope of the weight curve is not linear, as shown in Fig. 2(b), as different value of p leads to different rate of change of memristance in individual memristors for each fixed applied charge.

So, equal weight values cannot be programmed with a fixed applied charge, if the memristors in the bridge have different nonlinear dopant drift charactersitics. This nonlinearity in the programmed weight of the synapse is difficult to model and include in the learning algorithm when training a neural network, as it is difficult to ascertain the nonlinear dopant drift characteristic of each and every memristor in the bridge synapse.

III. Memristor Bridge Synapse-Based Neural Network

The memristor bridge synapse [30] has been shown to be a promising device for implementing synaptic weights in artificial neural networks as it acts as a programmable nonvolatile memory. In addition, the bridge circuit also acts as a multiplier circuit, which can be used to replace the conventional analog multipliers. In this section, we present a neural network hardware architecture that utilizes the dual advantage of the memristor bridge circuit.

In neural networks, weighted input signals are summed in each unit. An easier way to execute the sum operation is with the current mode, where current is summed by the direct connection of output lines via Kirchhoff current law. Aiming for this, a differential amplifier with three transistors is combined with the memristor bridge as in Fig. 3 to convert the weighted voltage to its corresponding current. Fig. 4 shows a typical neural network where each neuron is comprised of multiple synapses and one activation unit. The whole structure of the neural network is simply the repeated connections of such neurons. The schematic of a memristor bridge synapse-based neuron and its equivalent circuit for the neuron in Fig. 4 are shown in Fig. 5(a) and (b), respectively. In Fig. 5(a),

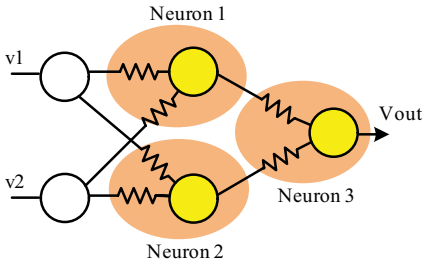


Fig. 4. Typical multilayer neural network where each neuron is comprised of multiple synapses.

voltage inputs are weighted by memristor bridge synapses. Then, they are converted to currents by differential amplifiers.

Let the input voltage be V_{in} and the kth weight be W^K. Then, the weighted voltage V_W is

$$V_W = W^K V_{\text{in}}^K \tag{9}$$

where W^K is produced by the memristor bridge synapse as defined in (4) and its range is $[-1.0, +1.0]$. This voltage is converted to a corresponding current with the transconductance parameter g_m. The currents at the positive and the negative output terminals of the differential amplifier associated with the kth synapse are

$$\left.\begin{aligned} i_k^+ &= -\tfrac{1}{2} g_m W^K V_{\text{in}}^K \\ i_k^- &= \tfrac{1}{2} g_m W^K V_{\text{in}}^K \end{aligned}\right\} \tag{10}$$

where i_k^+ and i_k^- are the currents at the positive and negative terminals. In the proposed circuit, all positive terminals of the input synapses are connected together, as are the negative terminals, and the sum of each signed current is computed separately. The sum of each signed current is

$$\left.\begin{aligned} i_{\text{SUM}}^+ &= -\tfrac{1}{2} \sum_k g_m W^K V_{\text{in}}^K \\ i_{\text{SUM}}^- &= \tfrac{1}{2} \sum_k g_m W^K V_{\text{in}}^K \end{aligned}\right\} \tag{11}$$

where i_{SUM}^+ and i_{SUM}^- are sum of currents at the positive and the negative terminals, respectively. The output current of the active load circuit is the difference between these two current components. It follows that

$$i_{\text{OUT}} = \sum_k g_m W^K V_{\text{in}}^K. \tag{12}$$

The output of each neuron is connected to another memristor bridge circuit. Since the memristor bridge synapse is comprised of two serially-connected memristors with opposite polarities, they operate complementally; the total memristance of two serial memristors is constant. Let such constant resistance be R_L. Then, the output voltage of the neuron is

$$V_{\text{OUT}} = R_L \sum_k g_m W^K V_{\text{in}}^K. \tag{13}$$

The voltage at the output is not linearly proportional to the current, and is soon saturated when the output voltages exceed $V_{DD} - 2V\text{th}$ or $-V_{SS} + 2V\text{th}$, where $V\text{th}$ is the threshold voltage of the two transistors of the active load, or synapses. Thus, the range of V_{OUT} is restricted as follows:

$$-V_{SS} + 2V\text{th} \le V_{\text{OUT}} \le V_{DD} - 2V\text{th}. \tag{14}$$

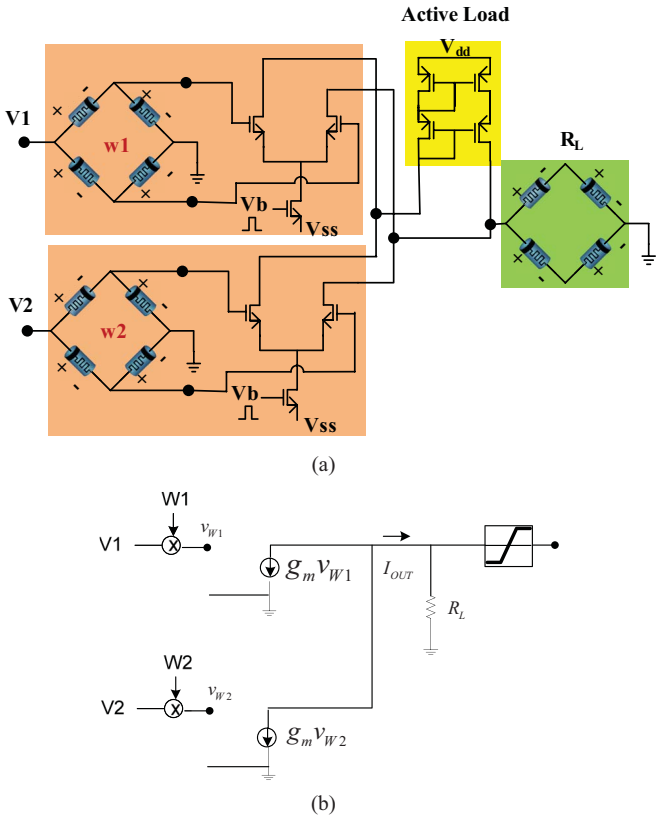


Fig. 5. Neuron at the first layer. Inputs are in voltage form. (a) Schematic of a neuron. (b) Equivalent circuit.

Let the minimum voltage $-V_{SS} + 2V\text{th}$ be V_{min} and the maximum voltage $V_{DD} - 2V\text{th}$ be V_{max}. Then, the circuit performs the activation function automatically as in Fig. 6(a). The HSPICE simulation of the activation unit was implemented with the circuit shown in Fig. 6(b). The differential input was varied from -1.5V to $+1.5$V, and the value of the load resistance was $R_L = 8$ KΩ. The output of the activation function to the differential input is shown in Fig. 6(c)

$$V_{\text{out}} = \begin{cases} R_L I_{\text{OUT}}, & \text{if } \frac{-V_{\text{SS}}+2V\text{th}}{R_{\text{OUT}}} \le I_{\text{out}} \le \frac{V_{\text{DD}}-2V\text{th}}{R_{\text{OUT}}} \\ V_{\text{max}}, & \text{if } \frac{V_{DD}-2V\text{th}}{R_{\text{OUT}}} \le I_{\text{out}} \\ V_{\text{min}}, & \text{if } I_{\text{out}} \le \frac{-V_{SS}+2V\text{th}}{R_{\text{OUT}}}. \end{cases} \tag{15}$$

The neuron at the noninput layer is a little different from that of the first layer. Instead of voltage inputs, its inputs are the currents. However, after passing through the memristor bridge, the input voltage of the differential amplifier is

$$V_W = W^K i_{\text{in}}^K R_L \tag{16}$$

where i_{in}^K is the current output of the preceding neuron. Since the active load is cascade type, and its resistance is much bigger (>160 KΩ) than 8 KΩ, the effect of the active load can be neglected. Once the signals are converted to voltages, all processings are the same as those of the input layer. The schematic of a memristor bridge synapse-based neural network corresponding to the neural network in Fig. 7(a) is shown in Fig. 7(b). The terminals "**t**" implemented in front of each memristor bridge serve a double purpose: for

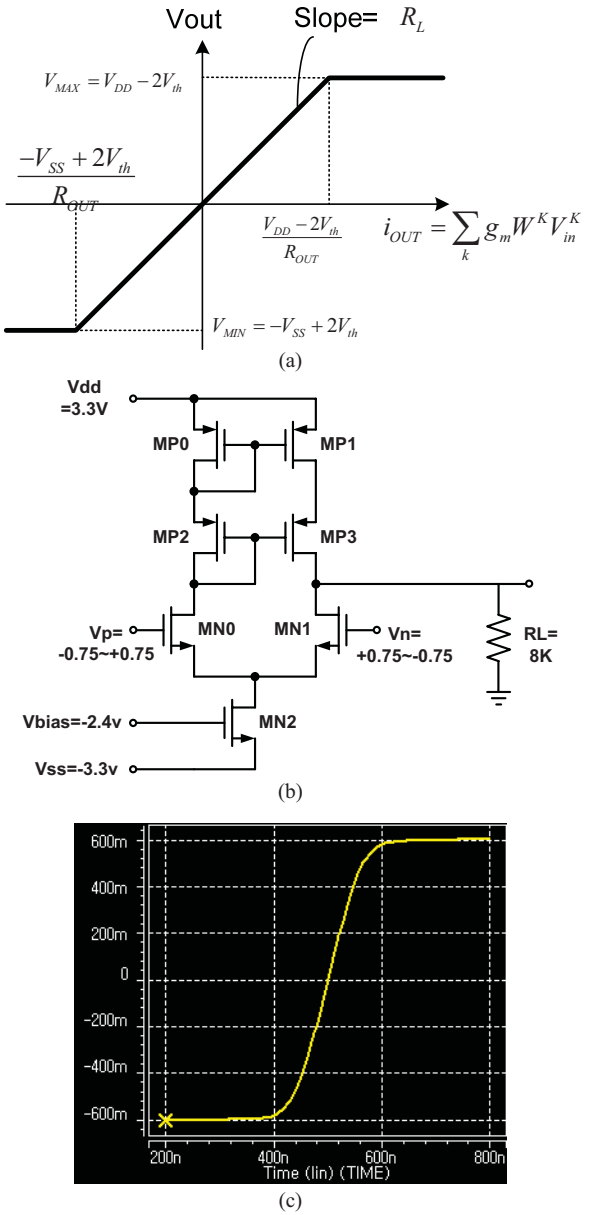


Fig. 6. (a) Activation function of neuron, (b) circuit for HSPICE simulation of the activation function, and (c) output of the activation function.

reading out the output of each neuron and for presenting the programming signal to each memristor bridge synapse. One thing which is worth to be mentioned for the proposed memristor bridge synapse-based neural network is that the operation at all transistor circuits as well as all memristor circuits, are based on pulses. Since the circuit operates only during the pulse width period, power consumption is reduced greatly. The circuit dissipates power only when the voltage pulses indicated in V_b at the bottom of each synapse circuit are in the high states.

IV. Memristor Synapse-Based Neural Network Learning

The hardware implementation of the popular back-propagation learning in multilayer neural network is a

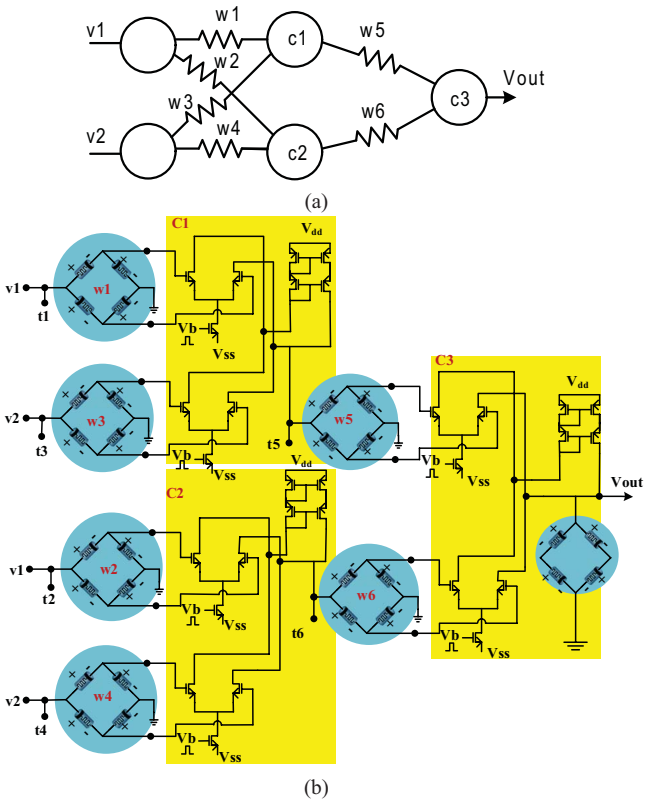


Fig. 7. Part of neural network architecture. (a) Sample network. (b) Portion of the equivalent hardware circuit.

difficult task, which is further aggravated by the inevitable variations between the on-chip components in analog circuits. These nonuniformities are troublesome when the training of the network is conducted off the chip without taking the nonideal implementation or component variations into account. As discussed in Section II, the nonlinearity in synaptic weighting occurs due to the variation of parameter p in individual memristors of the bridge synapse. Since an accurate model of the memristor bridge synapse is difficult to model during training, the chip-in-the-loop scheme is used to incorporate the memristor bridge nonidealities in the learning process without explicitly modeling it.

A. Chip-in-the-Loop Learning

Conventional chip-in-the-loop learning, shown in Fig. 8, is conducted as follows.

1) Target network is learned on a host computer.
2) Weight matrix is downloaded to the circuit.
3) The multilayer neural network circuit is retrained (tuned) by placing the circuit in the learning feedback loop; the forward network computation is performed in the circuit, weights, and outputs of the network are readout, and the weight update calculation is performed by the host computer [21].

The initial learning is done on the computer, and the weight matrix w^{initial} is downloaded to the circuit. The weights are then retrained by placing the circuit in the learning feedback loop. The forward computation is performed in the circuit,

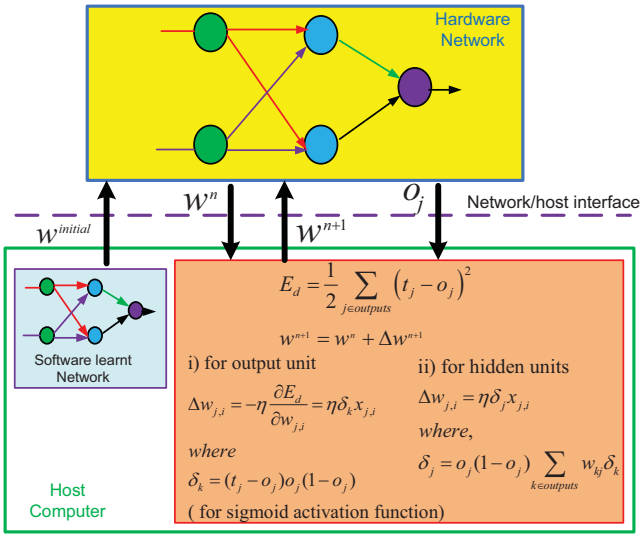


Fig. 8. Chip-in-the-loop learning scheme for multilayer neural network.

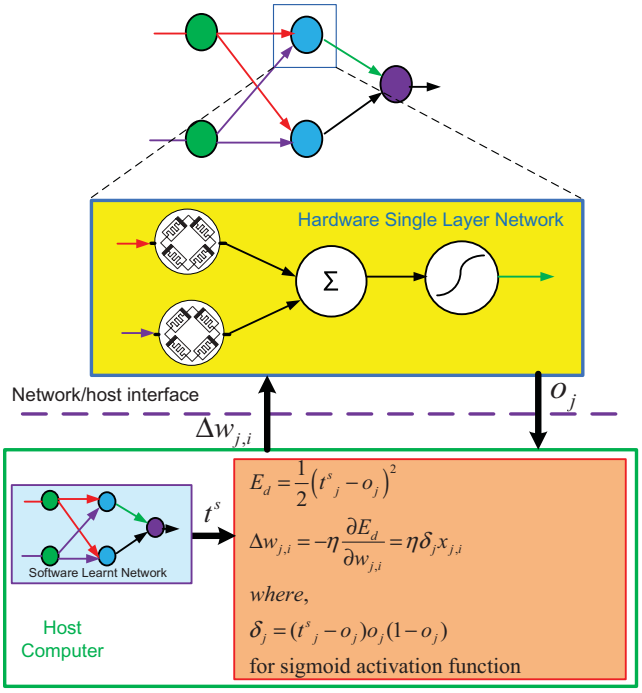


Fig. 9. Modified chip-in-the-loop learning. A complicated multilayer learning problem is learned after decomposing it into multiple but simple single layer learning problems.

and the output o_j is fed back to the computer. To calculate the updated weights w^{n+1} in the n+1th iteration, the weight values w^n of the synapse in the nth iteration have to be readout and fed to the computer. In this scheme, the synaptic weights of the succeeding layer should also be readout in order to calculate the weight update value for the synapse in the preceding layer of the network for back-propagation.

The main drawback of this learning is the communication overhead in continually reading and writing data across the network/ host interface. The reading of synapse in every epoch constitutes around half of the communication overhead, and writing to the synapse constitutes the other half. In addition, the readout of synaptic weights requires complicated circuitry; each synaptic weight requires read-out port and an auxiliary circuit, which is expensive in terms of chip area.

B. Modified Chip-in-the-Loop Learning

We propose a modified chip-in-the-loop learning rule that overcomes the difficulties of the conventional chip-in-the-loop learning, and is better suited for the memristor bridge synapse-based neural network. Unlike the conventional chip-in-the-loop learning, a complicated multilayer learning problem is learned after decomposing it into multiple but simple single layer learning in the proposed scheme. The proposed learning, shown in Fig. 9, is conducted as follows.

1) Target network is learned on a computer.
2) The outputs of all nodes in the network corresponding to all training data are stored in computer memory.
3) Unlike the conventional chip-in-the-loop learning where whole of the multilayer network is retrained at once, each constituent single layer network is retrained separately in chip-in-the-loop fashion. The forward network computation is done in the circuit, and the weight update matrix (Δw) is calculated by the host computer.

In the modified chip-in-the-loop learning, back-propagation of error from the output layers to hidden layers is avoided as each node is retrained using single layer learning. The proposed learning scheme also utilizes the inherent advantage of the memristor bridge synapse, whereby signal corresponding to Δw can be applied directly to the hardware synapses to obtain the updated synaptic weights. Unlike the conventional chip-in-the-loop, it is not necessary to readout the weight values w^n of the synapse in the nth iteration in order to calculate the updated weights w^{n+1} in the n+1th iteration. Hence, the proposed learning scheme reduces the communication overhead between the network/host interface by half, and also eliminates any additional auxiliary circuits for readout of the weights, which leads to simpler circuit design and better chip area utilization.

V. Simulation Results

The proposed learning scheme was simulated on the classical nonlinear problem of three-bit parity detection and a real-world application on car detection. At first, the multilayer neural network was learned by the host computer, and the output value of each node corresponding to each training data was stored in computer memory. The proposed hardware learning was then implemented, and each constituent single-layered neuron was learned independently. The forward pass on single-layered neuron learning was implemented in a circuit model simulated in MATLAB, and the weight update values were calculated based on the error of the simulated circuit model on the training data and the stored output values for that neuron.

The activation function in the circuit model was realized with a current-mode differential pair, as shown in Fig. 6(b), and the nonvolatile weight storage in the hardware network was realized by a memristor bridge synapse. Circuit modeling of the memristor bridge synapse was performed using the TiO_2 memristor model. The parameters used for the simulations

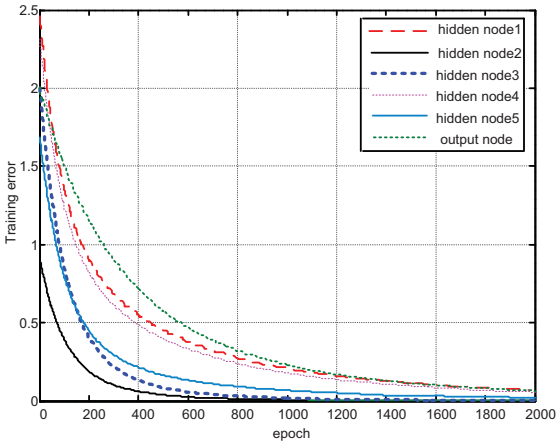


Fig. 10. Training error versus epoch curve for each node of hardware network.

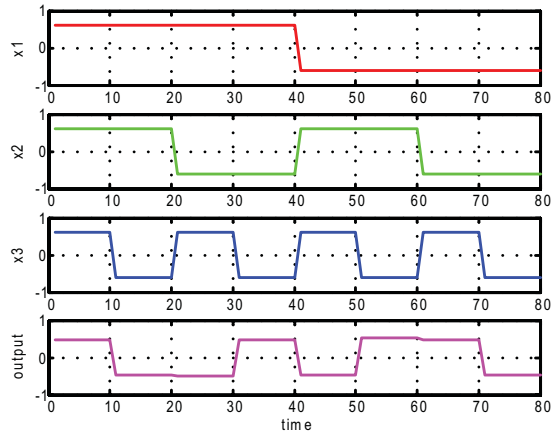


Fig. 11. Inputs and the corresponding output values at the output node of the hardware network trained using the proposed method.

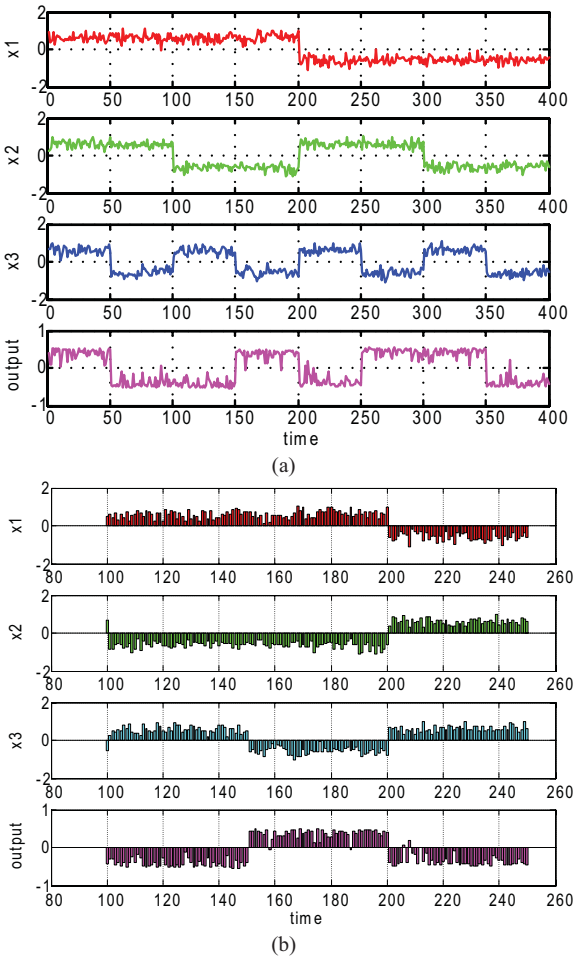


Fig. 12. (a) Input data with SNR = 10 dB and output of the hardware network trained using the proposed learning method. (b) Magnified portion of the results for data points 100–250.

were; $R_{\mathrm{ON}} = 116\ \Omega$, $R_{\mathrm{OFF}} = 16\ \mathrm{K}\Omega$, $D = 10$ nm, $\mu_v = 10^{-14}\ \mathrm{m}^2\mathrm{V}^{-1}\mathrm{S}^{-1}$ and $p = 6$. The synaptic weights were initialized with random values by applying random pulses to the memristor synapse. The range of weights supported by memristor bridge synapse was in the range of $[-1, +1]$. During hardware learning, the new weight values were programmed in the memristor bridge synapse by applying pulses, corresponding to Δw, in addition to the present state of the synapse. For this particular model, a pulse of amplitude 1 volt and width [0, 0.645] sec was required to change the weight of the synapse from [0, 0.9].

The three-bit parity problem was learned on the network of 3 inputs $\times$ 5 hidden $\times$ 1 output. The initial learning of the network was done in software and the output of all nodes was stored in computer memory. Assuming that the moderate input/output voltage range in actual hardware network is $[-0.6, +0.6]$ volts, the input/output range of $[+1, -1]$ in software network is translated to input/output range of $[+0.6, -0.6]$ in the simulated hardware network. The hardware network was then trained using the proposed learning scheme. The error versus epoch curve of hardware training for each neuron is shown in Fig. 10. The inputs and their corresponding outputs at the output node of the hardware trained network are shown in Fig. 11.

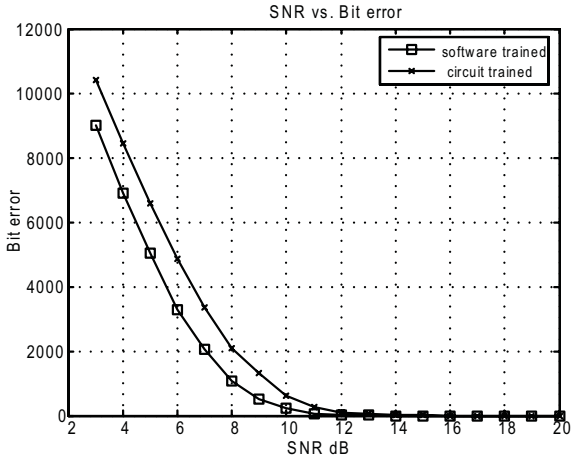


Fig. 13. Bit-error versus SNR curve for the network trained using the proposed method (circuit-trained) and the initial software-trained network (software-trained).

The performance of the hardware network was compared with that of the initial software-trained network. The robustness of the network was compared by applying 40 000 input data with additive Gaussian white noise, resulting in

Fig. 14. Sample of training images each of size 24 × 18 pixels. (a) Positive training images of rear of car. (b) Negative training images.

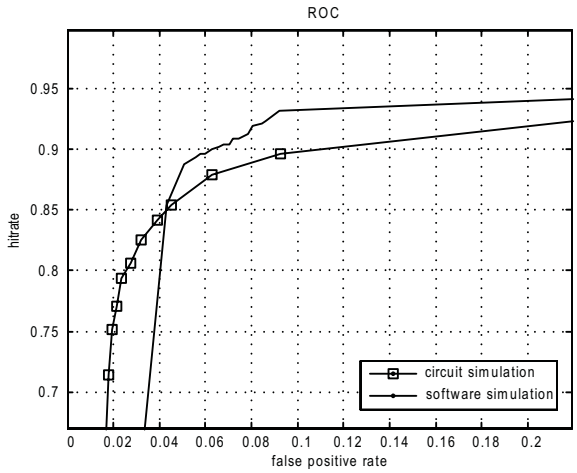


Fig. 15. ROCs curve of the initial software-trained network (software simulaion) and the proposed hardware-trained network (circuit simulation).

different signal-to-noise ratio (SNR). Fig. 12 shows the result of the hardware network trained using the proposed method for a SNR of 10 dB. Fig. 13 shows the error (described in terms of bit error) versus the SNR curve for the hardware-trained and the initial software-trained network. It can be observed that the performance of the hardware network using the proposed method is similar to the initial software-trained network.

The car-detection problem was simulated on a network of 432 inputs × 10 hidden × 1 output. The network was trained on grayscale images of size 24 × 18 pixels. Each image was preprocessed for correcting the extreme lighting conditions and histogram equalized to improve contrast [3]. The training set conatined 1500 positive (car) images and 1500 negative (noncar) images each of size 24 × 18 pixels, as shown in Fig. 14. The inputs to the network were the intensity values of the 24 × 18 = 432 pixels of the training images. The range [0, 255] of intensity value of the pixels in software, was translated to the range of [0, 0.6] in hardware implementation. The input/output range of [+1, −1] in software network was translated to an input/output range of [+0.6, −0.6] in the simulated hardware network. As before, the network was initially trained in software and later using the proposed modified chip-in-the-loop learning scheme. The network was tested on a test set containing 480 car and 3000 noncar images, each of size 24 × 18 pixels. The receiver operating characteristic (ROC) [33] curves for the network trained in hardware using the proposed learning method (circuit-trained) and the initial software-trained network are shown in Fig. 15. The ROC was drawn by varying the detection threshold at output from [+0.6, −0.6] with a step of 0.1. It can be seen that the performance of the hardware network trained using the proposed method is comparable to the initial software-trained network.

VI. Conclusion

A multilayer neural network based on memristor bridge synapse and its learning scheme was proposed in this paper. The memristor bridge synapse acts as a nonvolatile memory due to the nonvolatile nature of the memristors, and is ideal for implementing synaptic weights in neural networks. The memristor bridge synapse is able to perform the synaptic weightings in the range of [−1, +1], and the weight programming and weight processing (synaptic multiplication) are implemented with pulse signals by utilizing the same input terminal discriminated only by different time slots. The memristor bridge synapse was used in the proposed architecture to solve the problem regarding nonvolatile weight storage in analog neural network. The proposed architecture could be implemented in a smaller chip area due to the small size of the memristors and with reduced power consumption due to its pulse-based operation.

A modified chip-in-the-loop scheme suitable for memristor synapse-based neural network was also proposed for easy learning, and to compensate for the spatial nonuniformity and nonideal response of the memristor bridge synapse. A complicated multilayer learning problem was learned after decomposing into multiple but simple single layer learning problems. Unlike the conventional chip-in-the-loop scheme, the proposed learning scheme, along with the advantage of the memristor bridge synapse, was able to eliminate the need to readout synaptic weights in each epoch for calculating weight updates, and reduced the communication overhead across the network/host interface. This also facilitates simpler circuit design and better chip area utilization as auxiliary circuits for readout are not necessary.

Network for solving the classical problem of three-bit parity and a real-world problem on car detection was learned. The simulation results showed that the proposed hardware network trained using the proposed learning scheme was able to learn the behavior of its software-trained counterpart with comparable performance.

In contrast to the existing hardware implementation techniques, the proposed memristor synapse-based multilayer

neural network has benefits of simpler architecture, reduced chip area, linearity, and reduced power consumption. The simulation results showed that memristor synapse-based neural network can be applied successfully in real-world applications.

REFERENCES

[1] A. Waibal, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 37, no. 3, pp. 328–339, Mar. 1989.

[2] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Back-propagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989.

[3] H. Rowley, S. Baluja, and T. Kanade, "Neural network based face detection," *IEEE Trans. Pattern Recognit. Mach. Intell.*, vol. 2, no. 1, pp. 23–38, Jan. 1998.

[4] R. Fierro and F. L. Lewis, "Control of nonholonomic mobile robot using neural networks," *IEEE Trans. Neural Netw.*, vol. 9, no. 4, pp. 589–600, Jul. 1998.

[5] F. L. Lewis, A. Yegildirek, and K. Liu, "Multilayer neural net robot controller with guaranteed tracking performance," *IEEE Trans. Neural Netw.*, vol. 7, no. 2, pp. 388–399, Mar. 1996.

[6] J. B. Lont and W. Guggenbuhl, "Analog CMOS implementation of a multilayer perceptron with nonlinear synapses," *IEEE Trans. Neural Netw.*, vol. 2, no. 3, pp. 457–465, May 1992.

[7] M. Holler, S. Tam, H. Castro, and R. Benson, "An electrically trainable artificial neutral network (ETANN) with 102 040 floating gate synapses," in *Proc. Neural Netw. Int. Joint Conf.*, 1989, pp. 191–196.

[8] A. J. Montalvo, R. S. Gyuresik, and J. J. Paulos, "Toward a general purpose analog VLSI neural network with on-chip learning," *IEEE Trans. Neural Netw.*, vol. 8, no. 2, pp. 413–423, Mar. 1997.

[9] T. Shima, T. Kimura, Y. Kamatani, T. Itakura, Y. Fujita, and T. Iida, "Neuro chips with on-chip back-propagation and/or hebbian learning," *IEEE J. Solid State Circuits*, vol. 27, no. 12, pp. 1868–1876, Dec. 1992.

[10] J. Misra and I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress," *Neurocomputing*, vol. 74, nos. 1–3, pp. 239–255, 2010.

[11] F. M. Diasa, A. Antunesa, and A. M. Motab, "Artificial neural networks: A review of commercial hardware," *Eng. Appl. Artif. Intell.*, vol. 17, no. 8, pp. 945–952, 2004.

[12] L. M. Reyneri, "Implementation issues of neuro-fuzzy hardware: Going toward HW/SW co-design," *IEEE Trans. Neural Netw.*, vol. 14, no. 1, pp. 176–194, Jan. 2003.

[13] D. Hammerstrom, "A survey of bio-inspired and other alternative architectures," in *Nanotechnology: Information Technology-II*, vol. 4, R. Waser, Ed. New York: Wiley-VCH Verlag, 2008, pp. 251–285.

[14] P. D. Moerland and E. Fiesler, *Neural Network Adaptations to Hardware Implementations*, E. Fiesler and R. Beale Eds. London, U.K.: Oxford Univ. Press, 1997, pp. 1–13.

[15] S. Draghici, "Neural networks in analog hardware–design and implementation issues," *Int. J. Neural Syst.*, vol. 10, no. 3, pp. 19–42, 2000.

[16] H. P. Graf, L. D. Jackel, R. E. Howard, B. Straughn, J. S. Denker, W. Hubbard, D. M. Tennant, and D. Schwartz, "VLSI implementation of a neural network memory with several hundreds of neurons," in *Proc. AIP Conf. Neural Netw.*, 1987, pp. 182–187.

[17] T. Morishita, Y. Tamura, T. Otsuki, and G. Kano, "A BiCMOS analog neural network with dynamically updated weights," *IEICE Trans. Electron.*, vol. 75, no. 3, pp. 297–302, 1992.

[18] M. Jabri and B. Flower, "Weight perturbation: An optimal architecture and learning technique for analog VLSI feedforward and recurrent multilayer networks," *IEEE Trans. Neural Netw.*, vol. 3, no. 1, pp. 154–157, Jan. 1992.

[19] Y. Maeda, H. Hirano, and Y. Kanata, "A learning rule of neural networks via simultaneous perturbation and its hardware implementation," *Neural Netw.*, vol. 8, no. 2, pp. 251–259, 1995.

[20] K. Hirotsu and M. A. Brooke, "An analog neural network chip with random weight change learning algorithm," in *Proc. Int. Joint Conf. Neural Netw.*, 1993, pp. 3031–3034.

[21] S. M. Tam, B. Gupta, H. A. Castro, and M. Holler, "Learning on analog VLSI network chip," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, Nov. 1990, pp. 701–703.

[22] B. Erkmen, N. Kahraman, R. A. Vural, and T. Yildirim, "Conic section function neural network circuitry for offline signature recognition," *IEEE Trans. Neural Netw.*, vol. 21, no. 4, pp. 667–672, Apr. 2010.

[23] L. O. Chua, "Memristor-the missing circuit element," *IEEE Trans. Circuit Theory*, vol. 18, no. 5, pp. 507–519, Sep. 1971.

[24] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, 2008.

[25] G. Snider, "Self-organized computation with unreliable memristive nanodevices," *Nanotechnology*, vol. 18, no. 36, pp. 1–13, 2007.

[26] Y. Ho, G. M. Huang, and P. Li, "Dynamical properties and design analysis for nonvolatile memristor memories," *IEEE Trans. Circuits Syst. I*, vol. 58, no. 4, pp. 724–736, Apr. 2011.

[27] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano Lett.*, vol. 10, no. 4, pp. 1297–1301, 2010.

[28] A. Afifi, A. Ayatollahi, and F. Raissi, "Implementation of biologically plausible spiking neural network models on the memristor crossbar-based CMOS/nano circuits," in *Proc. Eur. Conf. Circuit Theory Design*, 2009, pp. 563–566.

[29] K. D. Cantley, A. Subramaniam, H. J. Stiegler, R. A. Chapman, and E. M. Vogel, "Neural learning circuits utilizing nano-crystalline silicon transistors and memristors," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 4, pp. 565–573, Apr. 2012.

[30] H. Kim, M. P. Sah, C. Yang, T. Roska, and L. O. Chua, "Memristor bridge synapse," *Proc. IEEE*, vol. 100, no. 6, pp. 2061–2070, Jun. 2012.

[31] H. Kim, M. P. Sah, C. Yang, T. Roska, and L. O. Chua, "Neural synaptic weighting with a pulse-based memristor circuit," *IEEE Trans. Circuit Syst. I*, vol. 59, no. 1, pp. 148–158, Jan. 2012.

[32] Y. Joglekar and S. Wolf, "The elusive memristor: Properties of basic electrical circuits," *Eur. J. Phys.*, vol. 30, no. 4, pp. 661–675, 2009.

[33] T. Fawcett, *ROC Graphs: Notes and Practical Considerations for Researchers*. Boston, MA: Kluwer, 2004.

Shyam Prasad Adhikari was born in Pokhara, Nepal. He received the M.Tech. degree in electronics and communication engineering from the Malaviya National Institute of Technology, Jaipur, India, in 2008. He is currently pursuing the Ph.D. degree in electronics engineering with Chonbuk National University, Jeonju, Korea.

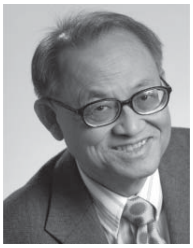
His current research interests include machine vision, pattern recognition, and memristors and their application to neural networks.

Changju Yang received the B.S. and M.S. degrees in electronics from Chonbuk National University, Jeonju, Korea, in 2008 and 2010, respectively, where he is currently pursuing the Ph.D. degree in electronics and information engineering.

His current research interests include circuit design, analog Viterbi decoders, and analysis of memristors and memristive systems.

Hyongsuk Kim (M'09) received the Ph.D. degree in electrical engineering from the University of Missouri, Columbia, in 1992.

He is currently a Professor with the Division of Electronics Engineering, Chonbuk National University, Jeonju, Korea. His current research interests include memristors and their application to cellular neural and nonlinear networks.



Leon O. Chua (F'74) received the M.S. degree from the Massachusetts Institute of Technology, Cambridge, and the Ph.D. degree from the University of Illinois at Champaign-Urbana, Urbana, in 1961 and 1964, respectively.

He was an Assistant Professor of electrical engineering with Purdue University, Lafayette, IN, in 1964, and was promoted to Associate Professor in 1967. He joined the University of California, Berkeley, in 1970, and has been a Professor of electrical engineering and computer sciences. His current research interests include memristors, chaos, cellular automata, and cellular neural networks.

Dr. Chua was the first recipient of the Gustav Kirchhoff Award in 2005 and the highest IEEE Technical Field Award for outstanding contributions to the fundamentals of any aspect of electronic circuits and systems that has a long-term significance or impact. He was also awarded the prestigious IEEE Neural Networks Pioneer Award in 2000 for his contributions in neural networks, the Guggenheim Fellow Award in 2010, and a Leverhulme Trust Visiting Professorship in 2011.