# OCTAN: An On-Chip Training Algorithm for Memristive Neuromorphic Circuits

Mohammad Ansari, Arash Fayyazi, Mehdi Kamal, Ali Afzali-Kusha, and Massoud Pedram

***Abstract*— In this paper, we propose a hardware friendly On-Chip Training Algorithm for the memristive Neuromorphic circuits (OCTAN). Although the proposed algorithm has a simple hardware like that of the random weight change (RWC) algorithm, it is much more efficient in terms of convergence speed and accuracy. In this algorithm, weights of the circuit are updated individually by a small value and the effect of individual weight update is assessed. If the weight change causes an increase in the error of the network, the weight update is reversed by applying the same change in the reverse direction twice. The usefulness of the proposed algorithm is verified by training some neuromorphic circuits for different applications. Compared to RWC and stochastic least-mean-squares (SLMS) training algorithms, our proposed algorithm needs, on average, 329× fewer epochs to find the minimum error point. Moreover, the accuracy of the networks trained by OCTAN is, on average, about 46% higher than those of RWC and SLMS algorithms. Additionally, a hardware for OCTAN is presented. This hardware provides a speedup of 172× (61×) compared to that of the RWC (SLMS) algorithm. Finally, the impact of PVT (process, voltage, and temperature) variations is studied on the proposed training hardware indicating an average training error increase of less than 3.27% in the presence of variations.**

***Index Terms*— On-chip training, online learning, neuromorphic computing, memristor.**

## I. Introduction

PERFORMANCE and energy-efficiency are among the most critical design parameters of any processor to be included in portable devices such as laptops, smart phones, and tablets [1], [2]. Recent advances in machine learning have opened up new venues to improve the efficiency of processors used for pattern recognition, object classification, and signal processing [3]. Approaches used in these processors are based on artificial neural networks (ANN). The networks may be used to realize processors which are inspired by the brain operation and, hence, referred to as neuromorphic processors.

Manuscript received March 19, 2019; revised June 11, 2019 and July 18, 2019; accepted August 2, 2019. This article was recommended by Associate Editor A. James. *(Corresponding author: Ali Afzali-Kusha.)*

M. Ansari and M. Kamal are with the School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran 1439957131, Iran (e-mail: mo.ansari@ut.ac.ir; mehdikamal@ut.ac.ir).

A. Fayyazi and M. Pedram are with the Ming Hsieh Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, CA 90089 USA (e-mail: fayyazi@usc.edu; pedram@usc.edu).

A. Afzali-Kusha is with the School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran 1439957131, Iran, and also with the School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran (e-mail: afzali@ut.ac.ir).

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

Digital Object Identifier 10.1109/TCSI.2019.2934560

Among neuromorphic processors, ANNs show much better energy efficiency in large scale designs as compared to the spiking neural networks (which use neuroscientific learning methods such as spike timing dependent plasticity) [4]. There are two types of hardware implementations for the ANNs: analog and digital. The analog artificial neural networks are often faster and more power efficient compared to the digital implementations [4].

In ANNs, synapses are important parts determining the impacts (weights) of the inputs on the output. Recently, a circuit element called memristor has been introduced for the analog implementation of synapses. These devices are resistors which conductance values are controlled by current or voltage previously applied to them [5], [6]. The non-volatile nature, low power consumption, dense layout, and plasticity in their resistance [6] have opened the opportunity for the hardware realization of ANNs using a crossbar array of memristors. For example, the proposed memristor device in [7] has a size of 50nm$\times$50nm, on/off ratio of $\sim 10^6$, endurance of $>10^8$, switching speed of $<$10ns, and data retention of $\sim$ 7 years [7].

One of the primary challenges in realizing large scale memristive neuromorphic processors is the convergence speed of the network during a training phase (*i.e.*, finding the weights of the network such that its outputs approach their desired values). The convergence speed is in turn affected by the choice of the learning algorithm. Additionally, the inherent precision limitation of analog synapses exacerbates the problem. The output accuracy and robustness requirements further increase the complexity of memristive neuromorphic processor designs [8], [9]. This makes identifying and using an effective and accurate training algorithm crucial in the hardware design of these processors.

The backpropagation (BP) algorithm is considered as the most powerful learning algorithm for neural networks [10], [11]. This algorithm, however, is not suitable for analog implementation as it requires circuitry for calculating the derivative of the activation function, multiplication, and extra blocks for the error back-propagation. An example of a hardware implementation of the standard BP scheme has been discussed in [12]. Another gradient-based training algorithm which is used widely is stochastic gradient descent (SGD). SGD simply does away with the expectation in the weight update and computes the gradient of the parameters using only a *single* or a few training examples. In [13], a multi-layer memristive network was trained using on-chip SGD to classify handwritten digits in MNIST dataset. These works assume that the control signals required for the training are provided from an external unit. No circuit implementation is provided for this external unit where MATLAB is actually used for training the network. In addition, a hardware architecture for training memristive crossbar-based neural networks is proposed in [14]. It utilizes dedicated digital and analog blocks (such as multiplier, reconfigurable sense amplifier, preprocessors for calculating the derivations) for implementing the BP algorithm in the hardware. These blocks impose a large hardware overhead in the realization of the network.

Stochastic least-mean-squares (SLMS) algorithm has also been suggested in [15] and [16]. In this algorithm, the need for an analog multiplier is eliminated. This is performed by transforming analog variables to stochastic bit streams obtained from the Bernoulli distributions whose variable values had the property of digital values. Thus, in the implementation, the weight values are updated by a constant magnitude. One of the drawbacks of the SLMS algorithm is that its accuracy is inversely proportional to the number of hidden layers. This originates from the fact that weights between the input and hidden layers as well as between hidden layers themselves are fixed random values where the algorithm has no control over them. This makes the SLMS algorithm effective for applications where the network may be trained without requiring hidden layers. This leads to a simple hardware implementation for the network where only comparators and small amount of digital logic gates are required.

Other research groups have utilized an approximation of the gradient for proposing several simpler training methods for hardware implementation of the network (see, *e.g.*, [17]–[19]). A major representative of the work in this category is Random Weight Change (RWC) algorithm [19]–[21]. In [22] and [23], a circuit-based RWC learning algorithm architecture is proposed. In RWC algorithm, the weight of each synapse is changed randomly by a fixed amount at each iteration requiring only simple random number generator circuits. This makes its hardware considerably simpler than that of the BP algorithm. The algorithm, however, suffers from a very slow convergence owing to the use of random numbers as the gradient steps.

In the present work, we propose On-Chip Training Algorithm for memristive Neuromorphic circuits (OCTAN) with focus on online mode learning. In this algorithm, the weights of the circuit (memristor states) are updated individually by a small value and the effect of individual weight update is assessed immediately after the write operation. If the weight change causes an increase in the error of the network, the weight update is reversed by applying the same change in the reverse direction twice. The capability of the algorithm to observe the circuit non-idealities (such as variations in memristors and CMOS devices and sneak-path problem [24]), it is suitable for applications which require a high tolerance to these imperfections.

Contributions of this work are listed next.

1) Proposing a hardware friendly training algorithm to improve the convergence speed compared to the state-of-the-art stochastic and gradient approximation training algorithms.
2) Reducing the sensitivity to process variations by introducing an on-chip error monitoring hardware verifying the error of the circuit immediately after each weight update.
3) Providing a hardware of the proposed training algorithm with a rather small area overhead.
4) Assessing the effect of the process variation and other non-ideal conditions on the training accuracy and convergence speed.

The rest of the paper is organized as follows. Section II describes the details of the proposed training algorithm as well as its hardware implementation. The efficacy and accuracy of the proposed training algorithm and the characteristics of its hardware are studied in Section III. Finally, the paper is concluded in Section IV.

## II. Proposed Training Algorithm and Its Hardware Implementation

### A. The Memristive Circuit

Before proposing the training algorithm and its hardware, we present the configuration of the memristive neuromorphic circuit used throughout this paper. It is noteworthy that the application of OCTAN is not limited to this type of memristive circuits and OCTAN can be applied to other types of memristive or even non-memristive neuromorphic circuits (as well as the circuits based on Deep Neural Network (DNN)). Here, the memristive crossbar-based circuit with CMOS inverters discussed in [24] is considered. The circuit, which is shown in Fig. 1, has a better performance, power consumption, energy efficiency, and area in comparison to other memristive neuromorphic circuits such as op-amp based ones (see, *e.g.*, [25]). For this circuit, the dot-product operation is performed in the memristive crossbar while the inverters carry out the nonlinear operation of the neuron considered as the activation function (note that the outputs of the inverter-based neural network are analog). The circuit, which has differential inputs, makes use of two memristors per weight for implementing both negative and positive weights. The voltage of the inverter input could be expressed as

$$V_{net_j} = \frac{\sum_{i=1}^{N}\left(V_{i_p}\sigma_{ji_p} + V_{i_n}\sigma_{ji_n}\right)}{\sum_{i=1}^{N}\left(\sigma_{ji_p} + \sigma_{ji_n}\right)} \tag{1}$$

where $v_{i_p}(v_{i_n})$ is the voltage of the $i^{th}$ non-inverted (inverted) input, $\sigma_{ji_p}$ $(\sigma_{ji_n})$ is the conductance of the memristor located in the non-inverted (inverted) row $i$ and column $j$, $N$ is the number of inputs of this layer, and $V_{net_j}$ is the voltage of the node $net_j$ (the input of the inverter of column $j$). This voltage may be considered as the weighted sum of the inputs. The inverter voltage transfer characteristics (VTC) has the form of a scaled sigmoid function, which acts as the neuron activation function. To provide differential inputs for the next layer, two inverters are used at the output of each layer (except the last layer). It should be noted that for the classifier ANNs, each output of the memristive ANN circuit (e.g. $O_1$, $O_2$, and $O_3$ in Fig. 1) corresponds to one of the output classes and has a digital value of either logical "0" or logical "1". In the case of

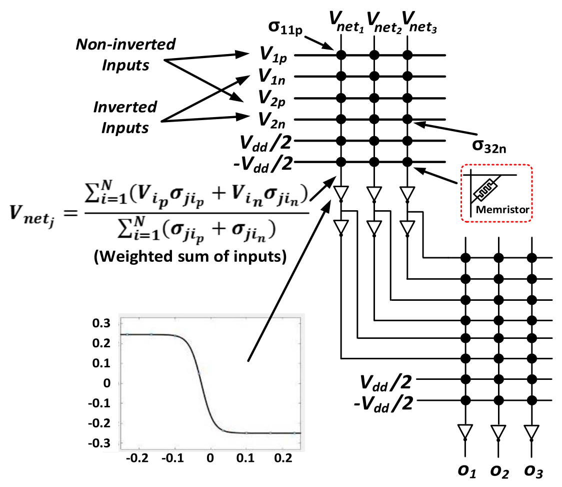


Fig. 1. A sample two-layer inverter-based memristive ANN with two inputs, three hidden neurons, and three outputs ($2 \rightarrow 3 \rightarrow 3$ network implemented using $6 \times 3$ and $8 \times 3$ memristive crossbars).

```
1:  Initialize all memristors in the network to a random high resistance state
    // the pattern index is k, the training set size is K, and the network has M
       outputs
2:  For each memristor $m$: $\delta_m = +\delta$
3:  Do
4:    For every pattern in the training set
5:      Present the k^th pattern to the network
6:      // Propagated the input forward through the network:
7:      Evaluate the error signal $Err_k^{old} = \sum_{i=1}^{M} |t_{ik} - o_{ik}|$
8:      If ($Err_k^{old} \leq Err_{Desired}$): break
9:      For every memristor $m$ in the network:
10:         Add the memristor conductance by $\delta_m$
11:         // Propagated the input forward through the network:
12:         Reevaluate the error signal $Err_k^{new} = \sum_{i=1}^{M} |t_{ik} - o_{ik}|$
13:         If ( $Err_k^{new} > Err_k^{old}$)
14:           Change the conductance of the memristor twice by $\delta_m^{new} = -\delta_m$
15:           $\delta_m = \delta_m^{new}$
16:         Else
17:           $Err_k^{old} = Err_k^{new}$
18:         End If
19:         If ($Err_k^{new} \leq Err_{tolerance}$): break
20:     End For
21:   End For
22: Till (maximum number of epochs > $Max_{epochs}$) or
23:      (($\sum_{k=1}^{K} Err_k$)< $Err_{target}$)
```

Fig. 2. Pseudo code of the propsoed training algorithm.

function approximation applications, however, the outputs of ANNs are analog and an analog-to-digital converter is needed to digitalize their outputs [26].

## *B. Proposed Training Algorithm*

The proposed training algorithm is based on altering the memristor conductance by applying write pulses to the memristors, and subsequently, measuring the output error for each training sample. In OCTAN, the focus has been on simplifying the algorithm to reduce the hardware required for implementing the training algorithm whose pseudo code is shown in Fig. 2. Also, Table I compares the characteristics of OCTAN algorithm with those of BP, RWC, and SLMS algorithms. Among these algorithms, BP can find the exact local minimum and has the highest hardware complexity since it needs building blocks for derivation and multiplication. On the other hand, SLMS, which only updates the weights of the output layer, has the lowest hardware complexity. Both of OCTAN and RWC, update all of the weights of the ANN, however, do not need hardware for multiplication and derivation, therefore, they have relatively medium hardware complexity. In OCTAN, first, to reduce the loading effect of the memristors [24] and power consumption of the circuit, all of the memristors are reset to a random high resistance (small conductance) state by applying a long negative pulse to the memristors (~10ns). This implies that the initial direction of weight updating is increasing the conductance values which is performed by applying a short positive pulse to the target memristor.

TABLE I
CHARACTERISTICS OF OCTAN, BP, RWC, AND SLMS TRAINING ALGORITHMS

| | Hardware Complexity | Weight Update Rule | | |
|---|---|---|---|---|
| | | Direction | Magnitude | Coverage |
| BP | High | Deterministic | Deterministic | All Weights |
| OCTAN | Medium | Deterministic | Constant | All Weights |
| RWC | Medium | Random | Constant | All Weights |
| SLMS | Low | Random | Constant | Last Layer Weights |

Before changing weights of the circuit, the error of the circuit is measured. The error for the $k^{th}$ training sample is defined by:

$$Err_k = \sum_{i=1}^{M} |t_{ik} - o_{ik}| \tag{2}$$

where $M$ is the number of outputs, $t_{ik}$ is the $i^{th}$ element of the target set for the applied input set and $o_{ik}$ is the $i^{th}$ output of the circuit for this input set. Since we do not use the derivative of the error in the training algorithm, there is no need to calculate the mean square error.

In the next step, if the calculated error is not satisfying the accuracy requirements, for each memristor, a positive (negative) pulse is applied to increase (decrease) its conductance based on the weight updating rule. The current error of the circuit is then recalculated and compared with the previous error of the circuit to make the proper decision. By iterating the procedure for a specific dataset, the circuit is trained and the overall error of the circuit is lowered. The iterative procedure is continued until the error becomes equal or less that the target value or the number of iterations exceeds the limit.

## *C. Proof of the Efficacy of the OCTAN Approach*

To train the ANN, the cost function (or the total energy of network, E) must be minimized using the gradient with respect to each weight. This can be evaluated using the gradient approximation method of forward finite difference defined as [27]

$$\frac{\partial E}{\partial w_{ij}^{l}} = \frac{\Delta E}{\Delta_{pert} w_{ij}^{l}} + O(\Delta_{pert} w_{ij}^{l}) \tag{3}$$

where $w_{ij}^{l}$ is the weight between the $j^{th}$ neuron of previous layer $(l-1)$ and the $i^{th}$ neuron of the current layer $(l)$,

$\Delta_{pert} w_{ij}^{l}$ is the perturbation, and $O\left(\Delta_{pert} w_{ij}^{l}\right)$ is the representation of the higher order terms of the approximation. By expanding $\Delta E$ in (2), and assuming that $O(\Delta_{pert} w_{ij}^{l})$ is small, the gradient is written as,

$$\frac{\partial E}{\partial w_{ij}^{l}} = \frac{E\left(w_{ij}^{l} + \Delta_{pert} w_{ij}^{l}\right) - E(w_{ij}^{l})}{\Delta_{pert} w_{ij}^{l}} \tag{4}$$

Now, the weight updating rule could be defined by

$$w_{ijnew}^{l} = w_{ijold}^{l} - \eta \frac{\partial E}{\partial w_{ij}^{l}} \tag{5}$$

$$\eta \frac{\partial E}{\partial w_{ij}^{l}} = \frac{\eta}{\Delta_{pert} w_{ij}^{l}} \left(E\left(w_{ij}^{l} + \Delta_{pert} w_{ij}^{l}\right) - E\left(w_{ij}^{l}\right)\right) \tag{6}$$

where $w_{ijnew}^{l}$ is the weight after updating, $w_{ijold}^{l}$ is the weight before updating, and $\eta$ is the learning rate. The term $\eta \frac{\partial E}{\partial w_{ij}^{l}}$ is considered as a weight updating value, which is represented by $\Delta w_{ij}^{l}$. Note that $\eta$ and $\Delta_{pert} w_{ij}^{l}$ are constants.

In our memristive array, weights (synapses) are the conductances of the memristors, and the amount of the perturbation is assumed to be $\delta$. In (5), when $E\left(w_{ij}^{l} + \Delta_{pert} w_{ij}^{l}\right)$ is less (more) than $E\left(w_{ij}^{l}\right)$, $\Delta w_{ij}^{l}$ is positive (negative). By keeping this direction in mind and for simplicity in the hardware implementation, we propose to use the following equations for determining weight updating value based on (5):

$$\Delta w_{ij}^{l} = \begin{cases} \Delta w_{ijprev}^{l}, & Err\left(w_{ij}^{l} + \Delta w_{ijprev}^{l}\right) < Err(w_{ij}^{l}) \\ -\Delta w_{ijprev}^{l}, & Err\left(w_{ij}^{l} + \Delta w_{ijprev}^{l}\right) > Err(w_{ij}^{l}) \end{cases} \tag{7}$$

where $\Delta w_{ijprev}^{l}$ is $\Delta w_{ij}^{l}$ in the previous iteration. Obviously, in the initial iteration, (6) is not employed while the $\Delta w_{ij}^{l}$ for each memristor could be considered as either $+\delta$ or $-\delta$. Also, the maximum/minimum limits of conductance values of the memristors should not be violated. In this case, the $w_{ij}^{l}$ is not changed.

Considering the fact that each output of the ANN (*e.g.*, $o_{ik}$) is a function of the ANN weights, one can express the energy of network as

$$E = f\left(w_1, w_2, .. w_j, .., w_h\right) = \sum\nolimits_{k=1}^{K} \sum_{i=1}^{M} (t_{ik} - o_{ik})^2 \tag{8}$$

where $h$ is the number of weights, $w_j$ is the $j^{th}$ weight, $M$ is the number of outputs, $K$ is the number of training samples, $t_{ik}$ is the target value for the $i^{th}$ output in the $k^{th}$ training sample, and $E$ is the total cost function. We can approximate $E$ around an operating point $\vartheta(w_{1\theta}, w_{2\theta}, \ldots, w_{h\theta})$ by

$$E = f\left(w_{1\theta}, w_{2\theta}, \ldots, w_{h\theta}\right) + \frac{\partial f}{\partial w_1}(w_1 - w_{1\theta}) + \ldots + \frac{\partial f}{\partial w_h}(w_h - w_{h\theta}) \tag{9}$$

Since the energy of the operating point $\vartheta$ is $E_\theta = f\left(w_{1\theta}, w_{2\theta}, \ldots, w_{h\theta}\right)$, one may rewrite (8) as

$$E = E_\theta + \sum\nolimits_{r=1}^{h} \frac{\partial f}{\partial w_r}(w_r - w_{r\theta}) \tag{10}$$

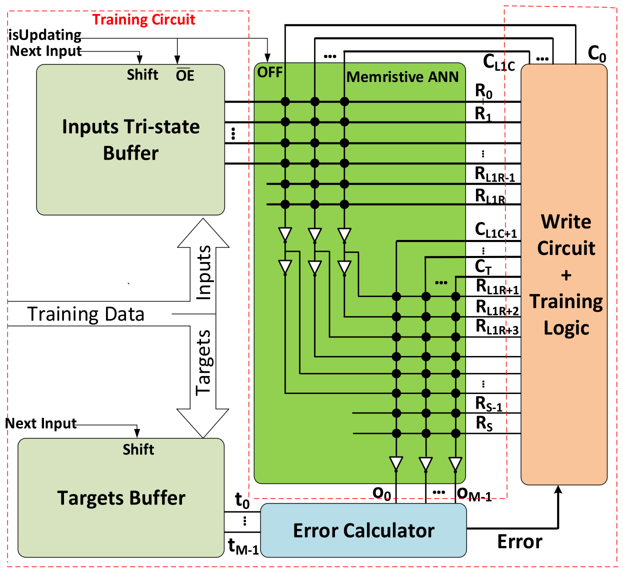


Fig. 3. Overall view of the hardware of OCTAN.

Therefore, the energy change with respect to the operating point may be expressed as

$$\Delta E = E - E_\theta = \sum\nolimits_{r=1}^{h} \frac{\partial f}{\partial w_r}(w_r - w_{r\theta}) = \sum\nolimits_{r=1}^{h} a_r \Delta w_r = a^T \Delta w \tag{11}$$

where $a$ is a matrix whose elements are $a_r = \frac{\partial f}{\partial w_r}$, $r = 1, \ldots, h$. Therefore, if $\Delta w_r$ is randomly set to $\pm\delta$, the probability of energy reduction $(P(\Delta E \leq 0))$ is equal to $1/2$. Since for every set of $\Delta w = (\Delta w_{1\theta}, \Delta w_{2\theta}, \ldots, \Delta w_{h\theta})$ yielding a negative $\Delta E$, there is another set of $\Delta w^c = (-\Delta w_{1\theta}, -\Delta w_{2\theta}, \ldots, -\Delta w_{h\theta})$ that gives rise to a positive $\Delta E$. Hence, the mean change of the operating point $\vartheta$ should be zero. As will be explained next, invoking the proposed algorithm, the change of the operating point $\vartheta$ moves toward lowering the energy function.

In the proposed algorithm, in cases where $\Delta E \leq 0$, the previous update $\Delta w_r$ is used again in the next iteration. Since $\Delta w_r$ (i.e., $\delta$) is small enough, the rate of energy decrease is almost constant near the operating point $\vartheta$. In these cases, the total energy of the next iteration should decrease again implying that P$(\Delta E \leq 0)$ (i.e., the probability of energy reduction) is 1. When $\Delta E > 0$, first, the previously applied $\delta$ is eliminated by applying a perturbation of $\delta$. Also, since $\Delta w_r$ of the next iteration is set as opposite of $\Delta w_r$ of the current iteration, the $\Delta E$ will be negative. After these two updates, we have P$(\Delta E \leq 0) = 1$ and P$(\Delta E > 0) = 0$. These probabilities in RWC algorithm are P$(\Delta E \leq 0) = 3/4$ and P$(\Delta E > 0) = 1/4$, and hence, the operating point may go up and down on the energy curve. This fluctuation makes RWC slower than OCTAN. More clearly, in the proposed algorithm, the error is decreased in every operating point except for the operating point that the approach has been reached a local minimum point.

### D. Hardware of OCTAN

Hardware of OCTAN is shown in Fig. 3 which contains the overall view of the system including the memristive

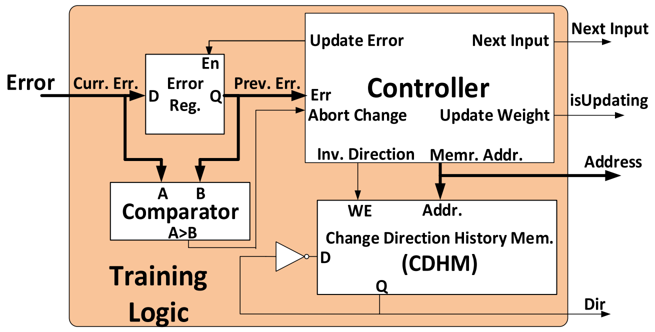


Fig. 4. Training logic of OCTAN including the controller, change direction history memory (CDHM), and error register.

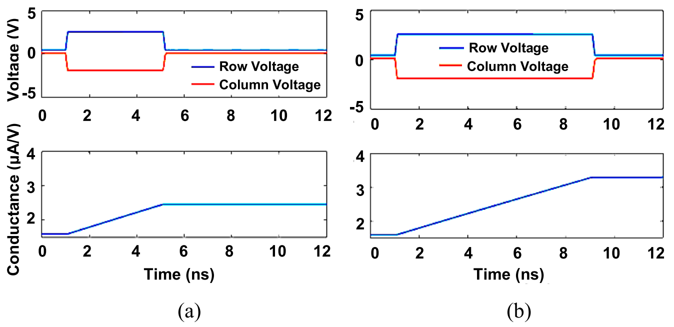


Fig. 5. The effect of the write pulse duration on the conductance change (a) pulse duration = 4ns, (b) pulse duration = 8ns. The memristor model of [28] for the memristor device of [7] with parameters of $A_1 = 1.6 \times 10^{-4}$, $A_2 = 1.6 \times 10^{-4}$, $b = 0.05$, $V_{tp} = 4$, $V_{tn} = 4$, $Ap = 816,000$, $An = 816,000$ is used for HSPICE simulation.

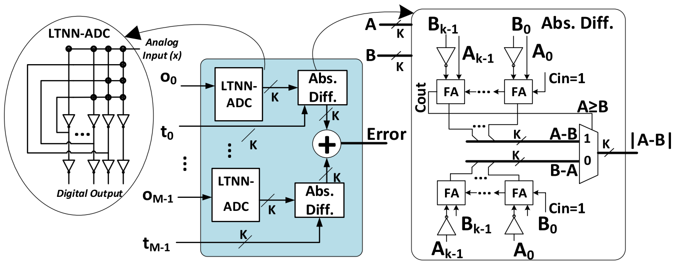


Fig. 6. Error calculator unit.

crossbar-based ANN and the training circuit. The training circuit consists of an error calculator unit, training logic, a write circuit, and two optional buffers. The training samples (inputs and their corresponding expected outputs) are loaded in the two buffers. Both buffers have a *"Next Input"* signal which is asserted by the controller unit of the training logic (see Fig. 4) when the training for the current sample is finished and a new training sample may be applied.

The "Input Tri-state Buffer" also has an output enable (*"OE"*) signal used for making the outputs of the buffer in the floating state. To prevent a conflict between the write circuit voltages and the buffer output voltages, the *"OE"* signal of the buffer is connected to *"isUpdating"*output of the controller. The *"isUpdating"* signal is asserted by the controller when the weights are being updated.

The value of perturbation $\delta$ depends on the width and the amplitude of the pulse that is applied across the target memristor. The amplitude of the applied voltage is constant in our proposed hardware. Therefore, a pulse width modulation (PWM) unit is included in the controller to control the value of $\delta$ by applying a pulse with a proper width on the *"isUpdating"* output signal of the controller. As an example, effects of two different write pulses with pulse widths of 4ns and 8ns on the conductance of the memristor are depicted in Fig. 5. This results were obtained by HSPICE simulation using memristor model of [28] for the memristor device presented in [7]. The figure reveals that the value of $\delta$ (the change in the conductance) is approximately doubled by doubling the duration of the pulse.

The error of the ANN is calculated in the Error Calculator unit (cf. Fig. 6), which includes low triangular neural network analog to digital converters (LTNN-ADC) [26] and absolute difference (Abs. Diff.) units to compare the outputs of the circuit with the expected outputs (targets). When the number of the ANN outputs is $M$, and $K$-bit ADCs are used, a $(log_2 KM + 1)$-bit adder is needed to aggregate the partial errors. The calculated error is passed to the training logic for a proper decision.

The training logic includes a register to store the error of the previous step, a comparator which compares the previous and current step error, a controller that controls the sequence of the training, and a Change Direction History Memory (CDHM) to store the last direction of the weight change. In our design, D-Flip-flops are used to implement CDHM. However, SRAM cells could also be used to improve the efficiency. The number of CDHM bits are equal to the number of memristors of the ANN. At the beginning of the training, all bits of CDHM are initialized to "1", which signifies that $+\delta$ is considered as the initial change value for memristors (line 2 of the pseudo code). After each weight update, the current step error is compared with the previous step error in the comparator. If the error has been increased, the output of the comparator becomes "1" and the *"Abort Change"* signal of the controller is activated to indicate that the current change direction is wrong and should be reversed (lines 13 and 14 of the pseudo code). Therefore, the controller flips the corresponding bit of the CDHM by applying the "*Invert Direction"* signal (which is indeed the write enable (*"WE"*) signal of the CDHM) for reversing the change direction of the current memristor and repeats the weight update operation in the reverse direction.

The task of the weight updating is performed by the Write Circuit unit (Fig. 7(a)). This unit consists of a pulse generator and two tri-state decoders for selecting rows and columns of the memristive crossbar. The pulse generator circuit is depicted in Fig. 7(b). Depending on the controller unit decision to increase or decrease the conductance (based on the "*Dir"*signal), this circuit puts a write voltage of either *Vddw* or –*Vddw*across its output nodes, which are *"VRow"*and "*VCol"*. The write voltage should be larger than the write threshold voltage of the memristors so that it can change the conductance of the memristor. In this work, the write threshold voltage of memristors was considered to be 4V and we used ±3.3V as the write voltage. This voltage is applied to the corresponding

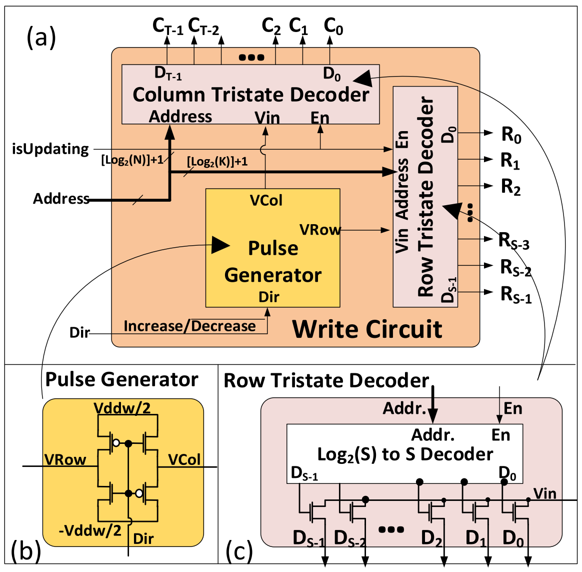


Fig. 7. The structure of the (a) write circuit, (b) pulse generator, and (c) tri-state decoder.

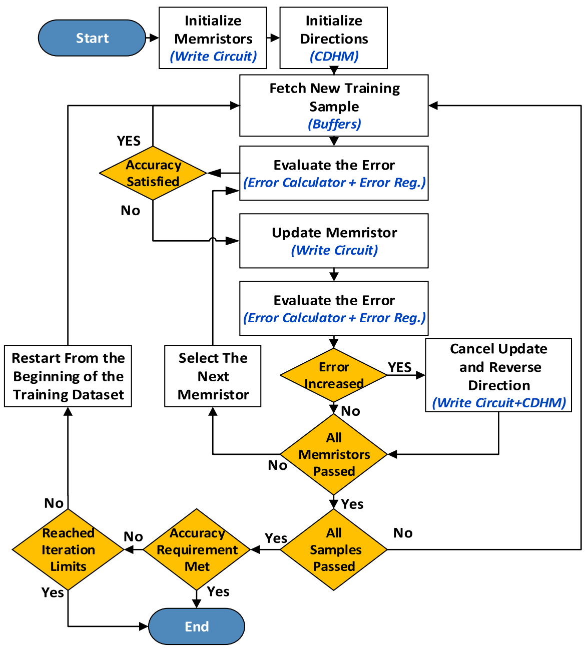

Start
Initialize Memristors (Write Circuit)
Initialize Directions (CDHM)
Fetch New Training Sample (Buffers)
Evaluate the Error (Error Calculator + Error Reg.)
Accuracy Satisfied
YES
No
Update Memristor (Write Circuit)
Evaluate the Error (Error Calculator + Error Reg.)
Error Increased
YES
No
Cancel Update and Reverse Direction (Write Circuit+CDHM)
Restart From the Beginning of the Training Dataset
Select The Next Memristor
All Memristors Passed
All Samples Passed
Accuracy Requirement Met
Reached Iteration Limits
End

Fig. 8. Flowchart of OCTAN representing the involving building block in each training stage.

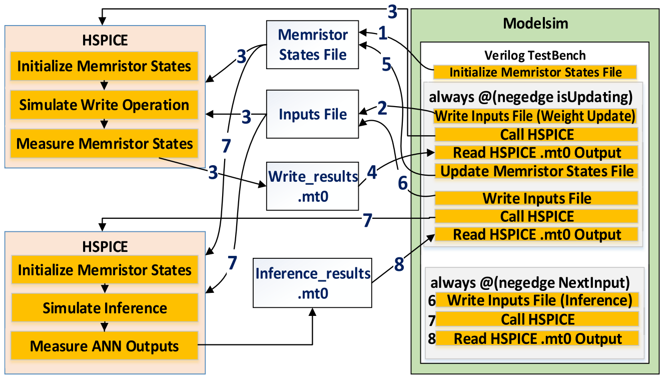


Fig. 9. Hybrid gate-level/circuit-level simulation framework for simulating the circuit behavior.

memristor through the row and column tri-state decoders. The circuit of a $\log_2(S)$ to $S$ tri-state decoder is shown in Fig. 7(c). In this circuit, each output of a digital decoder is applied to the gate of a pass transistor. The source terminals of these transistors are connected to the *Vin* pin of the tri-state decoder which is provided by the pulse generator unit and approximately has a voltage of either $Vddw/2$ or $Vddw/2$. Also, the drain of each transistor is connected to a row/column of the memristive crossbar. The size ((W/L) ratio) of the access transistors and NMOS transistors of the pulse generator is 12, the size of the PMOS transistors of the pulse generator is 15, and $L = 100nm$ was considered for all of these transistors. To control the amount of the shift in the conductance ($\delta$), the controller unit applies a single pulse with a desired duration to the input "*En*" of the tri-state decoder ("*isUpdating*" output signal of the controller). When *"En"* is "0", all of the outputs of the digital decoder are "0". This makes the write voltage disconnected from the target memristor. When *En* is "1", only one of the outputs of the encoder is activated while other outputs are deactivated. This makes other rows and columns in the floating state protecting other memristors from experiencing unwanted writes. During the memristor write operation, the controller activates the *"isUpdating"* signal. Thus, inputs of the memristive crossbar-based ANN float, and the ANN is powered off to prevent voltage conflict between the write circuit applied voltages and the memristive ANN internal voltages. Although the voltage drop on pass transistors could be compensated by increasing the duration of the pulse, one may replace pass transistors of the tri-state decoder with transmission gates to reduce the write voltage drop. After applying the write pulses to the all memristors, the controller unit fetches a new training sample from the buffer by applying the *"Next Input"* signal or terminates the training if the termination conditions (line 22 and 23 of Fig. 2) are satisfied.

The flowchart of Fig. 8 illustrates the utilization of the building blocks of the proposed hardware of OCTAN in each stage of the training process. It should be noted that the controller unit which has the role of managing the whole procedure and checking the conditions, is not depicted in this flowchart.

## III. Results and Discussion

### A. Hardware Characteristics

In this section, we assess the functionality and specifications of the hardware of OCTAN using post-synthesis (gate-level) simulations in Modelsim and accurate circuit-level simulations in HSPICE tool. The main test bench was developed in Verilog. In this test bench, to accurately simulate the write operation and inference of the memristive ANN and measure corresponding parameters (e.g. power), HSPICE tool is called from inside Modelsim. The details of this simulation setup are shown in Fig. 9 where:

1. At the beginning of the simulation, the test bench writes the "Memristor States file" which is included in the spice

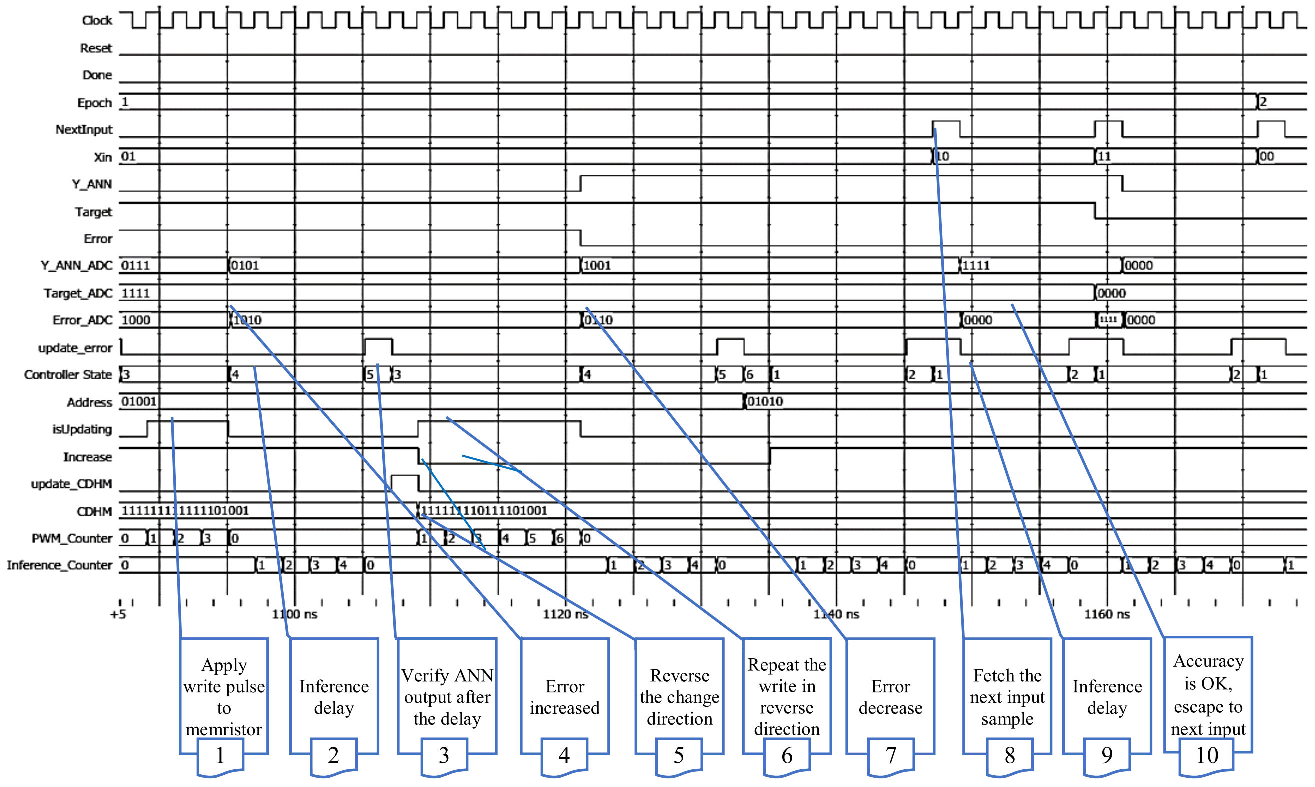


Fig. 10. Post-synthesis simulation output waveform from 1090ns to 1170ns for a 2 layer, 2 input ANN with 2 hidden neurons (a 2→ 2→1 network).

netlist and contains.param statements which define the initial condition of the memristors states in the HSPICE simulation.

2. On the negative edge of *"isUpdating"*, the test bench writes "Inputs File" based on the outputs of the digital Row/Column decoders. Each digital output of the decoders ($D_0$ to $D_{s-1}$ in Fig. 7(c)) is written to this file as a trapezoidal voltage pulse source. The duration of these pulses is measured using a counter that counts the clock cycles when "*isUpdating*" is HIGH. This file also contains the access transistors of the Row/Column tri-state decoders and the pulse generator circuit.
3. The test bench calls HSPICE to simulate the memristors write operation and measure the final states of the memristors after the write operation. After the simulation is finished, HSPICE generates an output file (e.g. "Write_results.mt0") which contains the measurements results.
4. The test bench reads the HSPICE output file.
5. The test bench updates "Memristor States file" to save the new states for the next simulations.
6. After the write is done (or a new input vector is fetched), the inference (forward propagation) should be simulated by calling HSPICE again. Therefore, in this step, the test bench rewrites the "Inputs File" so that it includes zero voltage sources for digital outputs of the decoders ($D_0$ to $D_{s-1}$ in Fig. 7(c)) which disables the write access transistors. It also generates voltage sources based on the current input vector and applies them to the primary inputs of the memristive ANN. In addition, the LTNN_ADC circuit is also included in this file to simulate all analog parts of the circuit at once.
7. HSPICE is called to simulate the inference, measure the memristive ANN outputs, and generate the corresponding output file (e.g. "Inference_results.mt0").
8. The test bench reads the HSPICE output file and applies outputs to the digital hardware for the next steps of the simulation.

The HSPICE simulations were performed using TSMC 90nm technology model for transistors and the memristor model proposed in [28] for the memristors devices of [7] with parameters of $A_1 = 1.6 \times 10^{-4}$, $A_2 = 1.6 \times 10^{-4}$, $b = 0.05$, $V_{tp} = 4$, $V_{tn} = 4$, $A_p = 816000$, $A_n = 816000$. The memristor model used in this work has been based on three main physical mechanisms which are electron tunneling, nonlinear drift, and a voltage threshold [28]. In the considered memristor technology, the minimum (maximum) resistance of the memristors was about 125K$\Omega$ (8.3M$\Omega$). Also, gate-level synthesis and design parameters of the digital parts of OCTAN hardware were extracted using Synopsys Design Compiler tool with TSMC 90nm standard cell library files. Without loss of generality, we have applied the described hardware simulation setup for simulating OCTAN on a two-layer ANN with 18 memristors and 2 hidden neurons to train a 2-input parity function. Fig. 10 shows the post-synthesis simulation waveform of this experiment from 1090ns to 1170ns. In this experiment, the clock cycle was 2ns and the circuit was completely trained in 1187ns. The area, power consumption,

TABLE II
AREA, POWER, AND DELAY FOR DIFFERENT COMPONENTS OF OCTAN HARDWARE IN TYPICAL 90NM TECHNOLOGY WITH $V_{dd} = 1V$ AT 25 °C

| Component | Area (μm$^2$) | Power (mW) | Delay (ns) |
|---|---|---|---|
| Controller | 1847 | 0.42 | 0.20 |
| Error Calculator | 122 | 0.02 | 4.52 |
| Row/Col Decoders | 135 | 0.02 | 0.66 |
| Write Circuit | 1344 | 9.52 | 1.15 |
| Sum | 3448 | 9.98 | 6.53 |

and delay of the synthesized components are shown in Table II. In the power analysis, the IR loss of the memristors was considered while the IR loss of the crossbar interconnects (which is in series with the memristors) was neglected. The reason is that the latter resistance is much smaller ($\sim 10^{-5}\times$) than that of the memristors [8]. Also, we have used the power reports of the employed synthesis tool (i.e., Synopsys Design Compiler) which reports the average power consumption of the digital components. This average power consumption is calculated based on the switching activities of the nodes in the design which are obtained by transient simulations (where we have used Modelsim tool to extract the transitions as VCD (value change dump) file format). The main portion of the power was due to the Write Circuit because of the high operating voltage (±3.3V). This component, however, was only active in the weight update stage of the training. After the training phase is finished, the whole training circuit could be turned off to save power. For instance, using the measurement methods of [24], when the training circuit was turned off, the power consumption of the memristive crossbar-based ANN circuit was 3.5$\mu$W, 6.6$\mu$W, and 275$\mu$W for BCW, IRIS, and MHEALTH benchmarks, respectively. Also, the delay of the ANNs was 2.6ns, 7.3ns, and 2.3ns for BCW, IRIS, and MHEALTH benchmarks, respectively. The low power consumption of the ANN was due to using inverter as neurons, memristors as synapses, and operating circuit at 0.5V supply voltage. Also, the power consumptions and the areas of the proposed design are compared with those of RWC and SLMS implementations in Table IV. The results reveal that the area of the proposed hardware is about 7.5% (16%) larger than that of RWC (SLMS) implementation while its power consumption is about 0.03% (0.14%) lower than that of the RWC (SLMS) implementation.

## B. Training Accuracy and Convergence Speed

In this section, we experimentally assess the efficacy of OCTAN on different datasets including IRIS [29], Breast Cancer Wisconsin (BCW) [30], Mobile Health (MHEALTH) [31], MNIST [13], and E. coli [32]. Table III shows the details of each benchmark application, their dataset specifications, and the configurations for the corresponding neural networks. In the neural network configurations, the order of the numbers is as the numbers of the inputs, hidden layer neurons, and outputs of the neural network which is a fully connected multilayer perceptron (MLP). Network configurations are mainly chosen based on [24] and [13]. Also, 80% of the dataset was

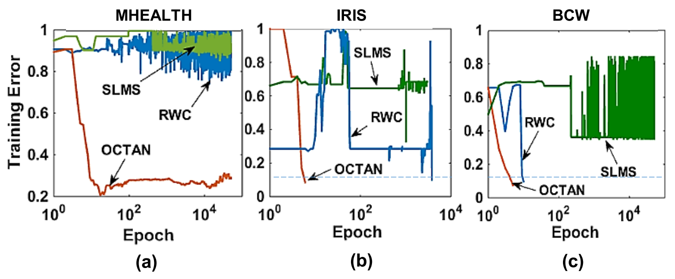


Fig. 11. Training curves for (a) MHEALTH, (b) IRIS, and (c) BCW benchmarks trained with RWC, SLMS, and OCTAN algorithms.

used for training while 20% was utilized for testing the final accuracy of the network. The training of the neuromorphic circuit discussed in Subsection II.A was performed with OCTAN, RWC, and SLMS algorithms in the online mode with a target accuracy of 90% (classification error of 10%), epoch limit of 50,000, and $\delta$ of 0.01 (which was proportional to the minimum conductance of the employed memristor). Note that in our settings, one epoch is when an entire dataset is passed forward and backward through the neural network only once. Also, the reported results are the best achieved results for each algorithm which are obtained by running each algorithm on each benchmark 10 times.

The hardware simulation setup discussed in the previous subsection is very time consuming for these datasets. Thus, we used the approach of [24] for extracting an accurate mathematical model of the ANN circuit and employed in it the training process in MATLAB. In this modeling scheme, after the HSPICE simulation of a single neuron, a hyperbolic tangent function is fitted to the neuron characteristics. The whole circuit is then modeled based on this fitted function and the KCL equations of the memristive netlist [24]. The training curves of these algorithms for MHEALTH, IRIS, and BCW benchmarks are shown in Fig. 11. Table III shows the minimum training error of each benchmark (the minimum value of the error obtained using the training algorithms) and the number of epochs processed to find the minimum error in each algorithm. The results revealed that the training accuracy of OCTAN was, on average, 46% better than those of the two other algorithms. Also, the training epochs that OCTAN needed to find the minimum error point was, on average, 329X smaller than those of the RWC and SLMS algorithms. Also, the test rates of the trained networks were 80.8%, 97.9%, 78.9%, 91.0%, and 96.7% for IRIS, BCW, MHEALTH, MNIST, and ECOLI benchmarks, respectively.

To evaluate the hardware training time of OCTAN and compare it with those of RWC and SLMS algorithms, here, we derive a model to estimate the needed time for training in the OCTAN algorithm. The proposed training algorithm has two phases for each memristor. In the first phase, the error of the circuit is calculated through the Error Calculator unit and compared with the previous step error in the controller. The delay of this phase is $(D_{fw} + D_{ec})$, where $D_{fw}$ is the forward propagation delay of the ANN and $D_{ec}$ is the delay of the Error Calculator unit. It should be noted that the training flow is controlled by the controller with the clock period of

TABLE III
BENCHMARKS SPECIFICATIONS AND TRAINING RESULTS

| Benchmark | Application | Dataset | Neural Network Configuration | Training Error | | | Epoch of the Optimal Err. | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | RWC | SLMS | OCTAN | RWC | SLMS | OCTAN |
| **IRIS** | Iris flower recognition | Features of 150 iris plants | 4→3→3 | 10% | 32.3% | 8.8% | 3778 | 1014 | 5 |
| **BCW** | Breast cancer prediction | Features computed from attributes of 699 cell nucleuses from the breast mass | 10→1→2 | 9.8% | 34.6% | 7.5% | 10 | 458 | 5 |
| **MHEALTH** | Human behavior analysis | 161279 time-series data samples of human activity | 23→100→12 | 73.8% | 84.9% | 20.5% | 13513 | 25786 | 17 |
| **MNIST** | Handwritten digits recognition | 1000 of 8×8 images of handwritten digits | 64→100→10 | 77.5% | 83.3% | 19.2% | 26055 | 39657 | 52 |
| **ECOLI** | Prediction of localization site of protein | Attributes of 336 Escherichia coli bacteria | 7→20→80→8 | 36.7% | 98.2% | 18.2% | 16656 | 18072 | 143 |
| **Average** | - | - | - | 41.6% | 66.7% | 14.8% | 12002 | 16997 | 44 |

TABLE IV
COMPARISON OF AREA AND POWER CONSUMPTION OF OCTAN WITH THOSE OF RWC AND SLMS

| Hardware | Power Consumption | | Area | |
|---|---|---|---|---|
| | Value (μW) | OCTAN Improvement (%) | Value (μm$^2$) | OCTAN Overhead (%) |
| OCTAN | 9978.6 | - | 3448 | - |
| RWC | 9981.6 | 0.03 | 3189 | 7.51 |
| SLMS | 9992.8 | 0.14 | 2889 | 16.21 |

TABLE V
THE COMPONENT DELAYS OF THE HARDWARE IN THE 90NM TECHNOLOGY

| Delay Component | Symbol | Value (ns) | Equivalent Clock Cycles |
|---|---|---|---|
| Controller clock period | $D_{ctrl}$ | 0.18 | 1 |
| Forward propagation delay of the memristive ANN | $D_{fw}$ | 4.06 | 23 |
| Error calculator delay | $D_{ec}$ | 4.52 | 26 |
| Write duration | $D_{wr}$ | 10 | 56 |

$D_{ctrl}$. Therefore, other delays should be expressed in terms of clock cycles. The delay of the second phase which is weight updating, corresponds to the time for updating the memristors given by $D_{wr}$. Here, $D_{wr}$ includes the delay of the decoders and pulse generator and the duration of the write pulse. Additionally, there is a probability, denoted by $P_{ac}$, that the change should be aborted and a reverse write pulse should be applied to the same memristor. Therefore, the time needed to train a memristive circuit with $N_m$ memristors using OCTAN ($T_{OCTAN}$) could be expressed by

$$T_{OCTAN} = N_{e_{OCTAN}} N_m N_s \left(1+P_{ac}\right)\left(D_{fw}+D_{ec}+D_{wr}\right) \quad (12)$$

where $N_e$ is the number of epochs, $N_m$ is the total number of memristors in the network, and $N_s$ is the number of training dataset samples.

Similarly, the training times of RWC [23] and the SLMS algorithms were estimated by, respectively, as

$$T_{RWC} = N_{e_{RWC}} N_s \left(D_{fw} + D_{ec} + N_m D_{wr}\right) \quad (13)$$
$$T_{SLMS} = N_{e_{SLMS}} N_s \left(D_{fw} + D_{ec} + N_{mOL} D_{wr}\right) \quad (14)$$

where $N_{mOL}$ is the number of the memristors of the output layer. Note that we assumed that the delay of the random number generator (which is used in RWC and SLMS algorithms) and comparator was included in $D_{ctrl}$. To evaluate the above expressions, delay components derived from hardware simulation are reported in Table V (the average values from the benchmarks are given). Although, the delay component of the controller may slightly vary for different algorithm implementations, we considered the same value for all algorithms. Also, to obtain $P_{ac}$, the number of aborted changes during the training were logged which showed an average $P_{ac}$ of 0.32 for the assessed benchmarks.

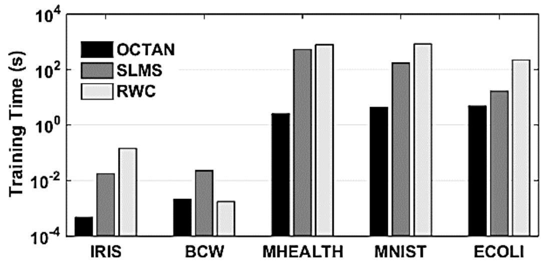


Fig. 12. Training time of different benchmarks using RWC, SLMS, and OCTAN algorithms.

Based on above calculations, training times of OCTAN, RWC, and SLMS algorithms are compared in Fig. 12. The results indicate an average speed-up of 172× (61×), for OCTAN compared to RWC (SLMS). Even if it is assumed that the hardware of the RWC and SLMS algorithms can update the memristors in parallel, the average speed-up of OCTAN compared to that of RWC (SLMS) will be 36× (8×). In addition, a comparison between the training time of the RWC and the BP algorithm performed in [23] showed a faster training for RWC hardware. Therefore, it could be concluded that the training time of OCTAN is even less than that of the BP algorithm.

### *C. Impacts of Process Variation and Other Non-Ideal Characteristics*

One of the main challenges in the designs based on nanoscale devices, especially memristors, is the uncertainty of the device parameters. The process variation in both memristor devices and MOS transistors may reduce the accuracy of the trained neuromorphic circuits [9]. In memristive crossbar-based ANN, the process variation of the memristor devices shifts the weights of the network and the process variation of the MOS transistors changes the characteristics of the activation function.

The impact of the process variation on the accuracy of the network is much more in the case of the activation function when compared to that of the memristors [9]. In addition to the process variation, other variation types such as the environmental variations and interconnect IR drops [8] may lead to the efficacy reduction of the memristive crossbar-based ANN. All of these conditions may cause deviations in the training procedure (e.g., changing in the amount of conductance change ($\delta$) and altering the VTC of the inverters) which may affect the convergence speed and accuracy of the training. In the following paragraphs, the impacts of the activation function variation, variation in the amount of conductance change ($\delta$), and instability of the memristor in its intermediate states on the OCTAN efficacy will be discussed. For the sake of brevity, only the results for the IRIS and BCW datasets will be presented.

*1) CMOS Inverters Variation:* The effect of the process variation on the activation function has been studied by applying a Gaussian distribution for the threshold voltage, length, and width of the transistors of the activation function inverters. The nominal value of each parameter was considered as the mean ($\mu$) value of its Gaussian distribution. Also, the variability ($\sigma/\mu$) of 20%, 10%, and 20% were considered for the distributions of the threshold voltage, width, and length of the transistors, respectively. Fig. 13(a) and (b) show the histograms of the number of epochs needed to train the circuits with variation in the activation function for IRIS and BCW benchmarks, respectively. For these simulations, 1,000 Monte-Carlo samples were used.

*2) Variation of the Amount of Conductance Change ($\delta$):* The nonlinear behavior of memristor, impact of process variation on the memristor, IR drop (and other voltage variations) [8], variation of the write circuit, and other environmental conditions such as temperature may also cause variations in the value of the amount of conductance change ($\delta$). Thus, we studied the effect of the $\delta$ variation on the quality of the training. In this assessment, we have considered two types of variations which impact the value of $\delta$ during the training process. The first type of variations which are due to the process variations were assumed to be fixed during the training process. The second type of variations which are due to environmental variations like temperature variation were assumed to vary during the training process. More explicitly, the value of $\delta$ was modeled by

$$\delta_{var} = \delta_{nom} + N_1(\sigma_{PV}, 0) + N_2(\sigma_{EV}, 0) \tag{15}$$

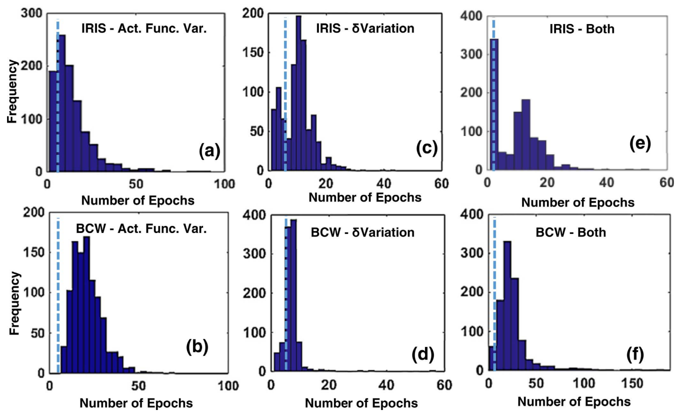
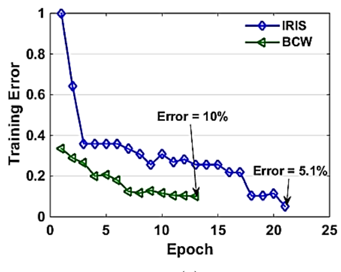


Fig. 13. Distributions of the number of epochs needed to train the circuit under the variations in the activation function for (a) IRIS and (b) BCW, value of $\delta$ for (c) IRIS and (d) BCW, and both activation function and $\delta$ value for (e) IRIS and (f) BCW (dashed lines indicate the nominal values).

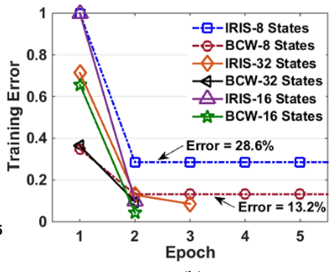
where $\delta_{var}$ is the value of $\delta$ with variations, $\delta_{nom}$ is the nominal value of $\delta$, and $N_1(\sigma, \mu)(N_2(\sigma, \mu))$ is a random number which is constant (variable) in the time domain. We used normal distribution for the random numbers. Based on the reported values in [33], $\sigma_{PV}$(process variation) is about 6%. In addition, to study the effect of the temperature and voltage variations and measure $\sigma_{EV}$ (environmental variation), we performed 1,000 Monte-Carlo simulations with random temperatures ($\mu = 25°\text{C}$, $\sigma = 100\%$) and random write voltages with variation range of $\pm 0.1 V_{DDW}$ (i.e. $\pm 0.33$V) on the whole write circuit (including pulse generator, access transistors, and memristive crossbar of size 256 with random initial states). The measured conductance change after 2ns for these 1,000 simulations had a variation of $\sigma_{EV} = 16.7\%$. By assuming these values for $\sigma_{PV}$and $\sigma_{EV}$, we performed Monte-Carlo simulations on the training procedure. The results of this experiment are reported in Fig. 13(c) and (d).

*3) Variation of Both CMOS Inverters and the Amount of Conductance Change:* The effect of both activation function and $\delta$ variations is depicted in Fig. 13(e) and (f) for the benchmarks. Also, the mean and variance of the training error of the benchmarks in the presence of these three kinds of variation are given in Table VI. Additionally, the difference between the typical training error ($E_{typ}$) and average of the training error in the presence of variation ($\mu_E$) is reported. The results show that the average training error in the presence of the PVT variations is increased by only 3.27% compared to that of the typical condition. Also, it is noteworthy that the variation in the $\delta$ value may even improve the training accuracy.

*4) Memristor Unstable Intermediate States and Limited Number of Resistive States:* In some memristor devices, there is a probability that the memristor lose its conductive filaments [34]. To simulate this instability during the training process, after each epoch, we randomly selected 5% of the memristors and reset them to a random high resistance state. The results of this simulation which are shown in Fig. 14 (a) reveal that this phenomenon affects the training conver-



(a)



(b)

Fig. 14. (a) Training curves considering unstable intermediate states, (b) limited number of resistive states.

TABLE VI
AVERAGE AND STANDARD DEVIATION OF THE TRAINING ERROR IN THE PRESENCE OF DIFFERENT KINDS OF VARIATION

| Variation | Benchmark | $\mu_E$ (%) | $\sigma_E$ (%) | $\mu_E - E_{typ}$ (%) |
|---|---|---|---|---|
| Activation Function | IRIS | 14.60 | 7.68 | 5.80 |
| | BCW | 12.25 | 3.61 | 4.75 |
| δ Value | IRIS | 8.14 | 1.73 | -0.66 |
| | BCW | 8.19 | 1.41 | 0.69 |
| Both δ and Activation Function | IRIS | 14.43 | 9.31 | 5.63 |
| | BCW | 10.89 | 5.57 | 3.39 |
| Average | - | 11.42 | 4.89 | 3.27 |

gence speed. It, however, does not have a considerable effect on the training accuracy. Also, to study the effect of limited number of resistive states, we have quantized the resistance of the memristors with 32, 16, and 8 states and increased the amount of weight change value accordingly. The results which are shown in Fig. 14 (b) reveal that by limiting the states to 32 and 16, the ANN could be trained without loss in accuracy. However, if the number of memristor states is decreased to 8, the best achievable error of the ANN increases to 13.2% (28.6%) for BCW (IRIS) benchmark. It should be noted that this is not due to the shortcoming of the training algorithm but the limitation of the memristor device.

## IV. CONCLUSION

In this work, we presented an on-chip training algorithm for memristive neuromorphic circuits (OCTAN). The proposed algorithm had the ease of hardware implementation of random weight change (RWC) algorithm as well as much higher efficiency in the convergence speed. The results of training with the proposed algorithm on five different benchmarks revealed that the training accuracy of the proposed algorithm was, on average, 46% better than those of the RWC and SLMS algorithms. Additionally, compared to RWC and SLMS algorithms, OCTAN required, on average, 329× less number of epochs to find the minimum error. Also, the hardware of OCTAN improved the time spent on training by 172× (61×) compared to that of the RWC (SLMS) algorithm. Finally, we performed Monte-Carlo simulations to investigate the robustness of the proposed algorithm in the presence of the process variation and other non-ideal effects such as IR drops

The results revealed that the PVT variations decreased the training accuracy by only about 3.27% on average.

Future works may include altering OCTAN hardware for batch mode training, finding optimal order for the weight update procedure, optimizing the CDHM memory, write circuit, and other components of OCTAN hardware for lower power/higher performance, and adding new training options such as momentum to OCTAN.

## REFERENCES

[1] M. Ansari, H. Afzali-Kusha, B. Ebrahimi, Z. Navabi, A. Afzali-Kusha, and M. Pedram, "A near-threshold 7T SRAM cell with high write and read margins and low write time for sub-20 nm FinFET technologies," *Integration*, vol. 50, pp. 91–106, Jun. 2015.

[2] M. Ansari, M. Imani, H. Aghababa, and B. Forouzandeh, "Estimation of joint probability density function of delay and leakage power with variable skewness," in *Proc. Int. Conf. Electron., Comput. Comput. (ICECCO)*, Ankara, Turkey, Nov. 2013, pp. 251–254.

[3] C. Mead, "Neuromorphic electronic systems," *Proc. IEEE*, vol. 78, no. 10, pp. 1629–1636, Oct. 1990.

[4] Z. Du *et al.*, "Neuromorphic accelerators: A comparison between neuroscience and machine-learning approaches," in *Proc. 48th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, New York, NY, USA, Dec. 2015, pp. 494–507.

[5] L. O. Chua, "Everything you wish to know about memristors but are afraid to ask," *Radioengineering*, vol. 24, no. 2, p. 319, 2015.

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

[7] W. Lu, K.-H. Kim, T. Chang, and S. Gaba, "Two-terminal resistive switches (memristors) for memory and logic applications," in *Proc. 16th Asia South Pacific Design Automat. Conf. (ASP-DAC)*, Yokohama, Japan, Jan. 2011, pp. 217–223.

[8] B. Liu, H. Li, Y. Chen, X. Li, Q. Wu, and T. Huang, "Vortex: Variation-aware training for memristor X-bar," in *Proc. 52nd ACM/EDAC/IEEE Design Automat. Conf. (DAC)*, San Francisco, CA, USA, Jun. 2015, pp. 1–6.

[9] A. BanaGozar, M. A. Maleki, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Robust neuromorphic computing in the presence of process variation," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Lausanne, Switzerland, Mar. 2017, pp. 440–445.

[10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986.

[11] E. Rosenthal, S. Greshnikov, D. Soudry, and S. Kvatinsky, "A fully analog memristor-based neural network with online gradient training," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Montreal, QC, Canada, May 2016, pp. 1394–1397.

[12] D. Soudry, D. Di Castro, A. Gal, A. Kolodny, and S. Kvatinsky, "Memristor-based multilayer neural networks with online gradient descent training," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 10, pp. 2408–2421, Oct. 2015.

[13] C. Li *et al.*, "Efficient and self-adaptive *in-situ* learning in multilayer memristor neural networks," *Nature Commun.*, vol. 9, no. 2385, Jun. 2018, Art. no. 2385.

[14] M. Cheng *et al.*, "TIME: A training-in-memory architecture for memristor-based deep neural networks," in *Proc. 54th ACM/EDAC/IEEE Design Automat. Conf. (DAC)*, Jun. 2017, pp. 26–31.

[15] C. Merkel and D. Kudithipudi, "A stochastic learning algorithm for neuromemristive systems," in *Proc. 27th IEEE Int. Syst.-Chip Conf. (SOCC)*, Las Vegas, NV, USA, Sep. 2014, pp. 359–364.

[16] T. Gokmen, M. Onen, and W. Haensch, "Training deep convolutional neural networks with resistive cross-point devices," *Front. Neurosci.*, vol. 11, p. 538, Oct. 2017.

[17] G. Cauwenberghs, "A fast stochastic error-descent algorithm for supervised learning and optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 1993, pp. 244–251.

[18] 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.

[19] K. Hirotsu and M. A. Brooke, "An analog neural network chip with random weight change learning algorithm," in *Proc. Int. Conf. Neural Netw. (IJCNN)*, Nagoya, Japan, vol. 3, Oct. 1993, pp. 3031–3034.

[20] B. Burton, F. Kamran, R. G. Harley, T. G. Habetler, M. Brooke, and R. Poddar, "Identification and control of induction motor stator currents using fast on-line random training of a neural network," *IEEE Trans. Ind. Appl.*, vol. 33, no. 3, pp. 697–704, May/Jun. 1997.

[21] J. Liu, M. A. Brooke, and K. Hirotsu, "A CMOS feedforward neural-network chip with on-chip parallel learning for oscillation cancellation," *IEEE Trans. Neural Netw.*, vol. 13, no. 5, pp. 1178–1186, Sep. 2002.

[22] S. P. Adhikari, C. Yang, K. Slot, M. Strzelecki, and H. Kim, "Hybrid no-propagation learning for multilayer neural networks," *Neurocomputing*, vol. 321, pp. 28–35, Dec. 2018.

[23] S. P. Adhikari, H. Kim, R. K. Budhathoki, C. Yang, and L. O. Chua, "A circuit-based learning architecture for multilayer neural networks with memristor bridge synapses," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 62, no. 1, pp. 215–223, Jan. 2015.

[24] M. Ansari *et al.*, "PHAX: Physical characteristics aware *ex-situ* training framework for inverter-based memristive neuromorphic circuits," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 37, no. 8, pp. 1602–1613, Aug. 2017.

[25] R. Hasan, C. Yakopcic, and T. M. Taha, "*Ex-situ* training of dense memristor crossbar for neuromorphic applications," in *Proc. IEEE/ACM Int. Symp. Nanosc. Archit. (NANOARCH)*, Boston, MA, USA, Jul. 2015, pp. 75–81.

[26] A. Fayyazi, M. Ansari, M. Kamal, A. Afzali-Kusha, and M. Pedram, "An ultra low-power memristive neuromorphic circuit for Internet of Things smart sensors," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1011–1022, Apr. 2018.

[27] 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.

[28] C. Yakopcic, T. M. Taha, G. Subramanyam, and R. E. Pino, "Generalized memristive device SPICE model and its application in circuit design," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 8, pp. 1201–1214, Aug. 2013.

[29] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugen.*, vol. 7, no. 2, pp. 179–188, 1936.

[30] O. L. Mangasarian, W. N. Street, and W. H. Wolberg, "Breast cancer diagnosis and prognosis via linear programming," *Oper. Res.*, vol. 43, no. 4, pp. 570–577, Aug. 1995.

[31] O. Banos *et al.*, "mHealthDroid: A novel framework for agile development of mobile health applications," in *Proc. Int. Workshop Ambient Assist. Living*, 2014, pp. 91–98.

[32] K. Nakai and M. Kanehisa, "A knowledge base for predicting protein localization sites in eukaryotic cells," *Genomics*, vol. 14, no. 4, pp. 897–911, Dec. 1992.

[33] M. Zanganeh, "Designing energy-efficient sub-threshold logic circuits using equalization and non-volatile memory circuits using memristors," Ph.D. dissertation, Boston Univ., Boston, MA, USA, 2015.

[34] D. B. Strukov and H. Kohlstedt, "Resistive switching phenomena in thin films: Materials, devices, and applications," *MRS Bull.*, vol. 37, no. 2, pp. 108–114, Feb. 2012.

**Mohammad Ansari** received the B.Sc., M.Sc., and Ph.D. degrees in electrical and electronics engineering from the University of Tehran, Tehran, Iran, in 2011, 2013, and 2018, respectively. He is currently a Researcher with the Low-Power High-Performance Nanosystems Laboratory, University of Tehran. His research interests include neuromorphic computing, low power VLSI designs, machine learning, the Internet of Things, and memristive circuit design.

**Arash Fayyazi** received the B.Sc. degree in electrical and electronics engineering from the Ferdowsi University of Mashhad, Mashhad, Iran, in 2014, and the M.Sc. degree from the University of Tehran in 2017. He is currently pursuing the Ph.D. degree with the Ming Hsieh Department of Electrical and Computer Engineering, University of Southern California. His research interests include learning in neuromorphic hardware, low-power designs, superconducting electronics, and machine-learning-based VLSI design.

**Mehdi Kamal** received the B.Sc. degree from the Iran University of Science and Technology, Tehran, Iran, in 2005, the M.Sc. degree from the Sharif University of Technology, Tehran, in 2007, and the Ph.D. degree from the University of Tehran, Tehran, in 2013, all in computer engineering. He is currently an Assistant Professor with the School of Electrical and Computer Engineering, University of Tehran. His current research interests include reliability in nanoscale design, approximate computing, neuromorphic computing, design for manufacturability, embedded systems design, and low-power design.

**Ali Afzali-Kusha** received the B.Sc. degree from the Sharif University of Technology, Tehran, Iran, in 1988, the M.Sc. degree from the University of Pittsburgh, Pittsburgh, PA, USA, in 1991, and the Ph.D. degree from the University of Michigan, Ann Arbor, MI, USA, in 1994, all in electrical engineering. He was a Post-Doctoral Fellow with the University of Michigan from 1994 to 1995. He has been with the University of Tehran, since 1995, where he is currently a Professor with the School of Electrical and Computer Engineering and the Director of the Low-Power High-Performance Nanosystems Laboratory. He was a Research Fellow with the University of Toronto, Toronto, ON, Canada, and the University of Waterloo, Waterloo, ON, Canada, in 1998 and 1999, respectively. His current research interests include low-power high-performance design methodologies from the physical design level to the system level, new memory, as well as digital design and implementation paradigms.

**Massoud Pedram** received the Ph.D. degree in electrical engineering and computer sciences from the University of California at Berkeley, Berkeley, CA, USA, in 1991.

He is currently the Stephen and Etta Varra Professor with the Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA, USA. He holds ten U.S. patents and has authored four books, 12 book chapters, and over 190 archival and 430 conference papers. His current research interests include low-power electronics, energy-efficient processing, and cloud computing to photovoltaic cell power generation, energy storage, and power conversion, and RT level optimization of VLSI circuits to synthesis and physical design of quantum circuits.

Prof. Pedram and his students have received six conference and two IEEE transactions best paper awards for the research. He was a recipient of the 1996 Presidential Early Career Award for Scientists and Engineers and an ACM Distinguished Scientist. He has served on the Technical Program Committee of a number of premiere conferences in his field. He was the Founding Technical Program Co-Chair of the 1996 International Symposium on Low-Power Electronics and Design and the Technical Program Chair of the 2002 International Symposium on Physical Design. He currently serves as the Editor-in- Chief of the *ACM Transactions on Design Automation of Electronic Systems*.