Learning in Memristive Neural Network Architectures Using Analog Backpropagation Circuits

Learning in Memristive Neural Network Architectures using Analog Backpropagation Circuits

Olga Krestinskaya, *Graduate Student Member, IEEE,* Khaled Nabil Salama, *Senior Member, IEEE* and Alex Pappachen James, *Senior Member, IEEE*

Abstract—The on-chip implementation of learning algorithms would speed-up the training of neural networks in crossbar arrays. The circuit level design and implementation of backpropagation algorithm using gradient descent operation for neural network architectures is an open problem. In this paper, we proposed the analog backpropagation learning circuits for various memristive learning architectures, such as Deep Neural Network (DNN), Binary Neural Network (BNN), Multiple Neural Network (MNN), Hierarchical Temporal Memory (HTM) and Long-Short Term Memory (LSTM). The circuit design and verification is done using TSMC 180nm CMOS process models, and TiO$_2$ based memristor models. The application level validations of the system are done using XOR problem, MNIST character and Yale face image databases.

Index Terms—Analog circuits, Backpropagation, Learning, Crossbar, Memristor, Hierarchical Temporal Memory, Long-Short Term Memory, Deep Neural Network, Binary Neural Network, Multiple Neural Network

I. INTRODUCTION

THE developments in Internet of Things (IoT) applications led to the demand to develop the near-sensor edge computation architectures [1]. The edge computing provides motivation to develop near-sensor data analysis that support non-Von Neumann computing architectures such as neuromorphic computing architectures [2], [3]. In such architectures, implementing the on-chip neural network learning remain as an important task that determines its overall effectiveness and use.

There are several works proposing the implementation of memristive neural network with backpropagation algorithm in digital and mixed-signal domain domain [4], [5], [6], [7], [8], [9], [10]. However, the analog learning circuits based on conventional backpropagation learning algorithm [11], [12], [8], [13], [14] in memristive crossbars have not been fully implemented. The implementation of such learning algorithm opens up an opportunity to create an analog hardware-based learning architecture. This would transfer the learning algorithms from the separate software and FPGA-based units to on-chip analog learning circuits, which can simplify and speed up the learning process.

O. Krestinskaya is a graduate student and research assistant with the Bioinspired Microelectronics Systems Group, Nazarbayev University

K. N. Salama is a Professor in King Abdullah University of Science and Technology in Kingdom of Saudi Arabia and principal investigator of Sensors Lab.

A.P. James is the Chair of Electrical and Computer engineering with the School of Engineering, Nazarbayev University, e-mail: apj@ieee.org.

Extending our previous work on analog circuits for implementing backpropagation learning algorithm [15], we present a system level integration of the analog learning circuits with that of traditional neuro-memristive crossbar array. We illustrate how this learning circuit can be used in different biologically inspired learning architectures, such as three layer Artificial Neural Networks (ANN), Deep Neural Networks (DNN) [5], [8], Binary Neural Network (BNN) [16], Multiple Neural Network (MNN) [17], Hierarchical Temporal Memory (HTM) [18] and Long-Short Term Memory (LSTM) [19].

The algebraic and integro-differential operations of backprogation learning algorithm, which are difficult to accurately implement on a digital system, are available inherently on the analog computing system. Further, modern edge-AI computing solutions warrant intelligent data processing at sensor levels, and analog system can reduce the demands for having high speed data converters and interface circuits. The proposed analog backpropagation learning circuit enables a natural on-chip analog neural network architecture implementation which is beneficial in terms of processing speed, reducing overall power and lesser complexity, comparing to digital counterparts.

The main contributions of this paper include the following.

- We introduce the complete design of analog backpropagation learning circuit proposed in [15] with control switches sign control circuit and weight update unit.
- We illustrate how the proposed backpropagation learning circuit can be integrated into different neuromorphic architectures, like DNN, BNN, MNN, LSTM and HTM.
- We show the implementation of additional activation functions that are useful for various neuromorphic architectures.
- We verified the proposed architecture for XOR problem, handwritten digits recognition and face recognition, and illustrate the effect of non-ideal behavior of memristors on the performance of the system.

This paper is organized into following sections: Section II introduces the relevant background of the learning architectures and backpropagation algorithm. Section III describes the proposed architecture of the backpropagation and the circuit level design. Section IV illustrates how the proposed backpropagation circuits can be integrated into different learning architectures. Section V contains the circuit and system level simulation results. Section VI discusses advantages and limitations of the proposed circuits and introduces the aspects of the

design that should be investigated in future, and Section VII concludes the paper. There is also a supplementary Material that includes the expanded background information, explicit explanation of the proposed circuit, the device parameters of the main backpropagation circuit proposed in Section III and simulation results for the learning circuit performance.

II. BACKGROUND

A. Learning algorithms and biologically inspired learning architectures

Three main brain inspired learning architectures that we consider in this work are neural networks [5], [16], [17], [20], HTM [18] and LSTM [19].

1) Neural Networks: There is a variety of the architectures and learning algorithms for the neural networks. In this work, we integrate analog learning circuits to three different types of artificial neural network [21]: DNN [5], [8], BNN [16] and MNN [17]. DNNs consist of many hidden layers and can have various combination of activation functions between the layers. Deep learning neural networks are useful for classification [22], regression [23], clustering [24] and prediction tasks [25]. A neural network which uses any combination of binary weight or hard threshold activation functions is typically known as BNN. There have been several successful implementations of BNN algorithms in software [26], [27], [28], [29] and an attempt to implement it on hardware [30], [31], [32]. The analog hardware implementation of BNN system with learning remains as an open problem [33], [32]. MNN is a systematic arrangement of the artificial neural networks that can process information from different data sources to perform data fusion and classification. Multiple neural network implies that the data from different data sources, such as various sensors, is applied to separate neural networks, and the output from each network is fetched into the decision network. This approach allows simplifying the complex computation processes, especially when there is a large number of data sources [17]. The analog hardware implementation of MNN is another new idea proposed in this paper.

2) Hierarchical Temporal Memory: HTM is a neuromorphic machine learning algorithm that mimics the information processing mechanism of neocortex in the human brain. HTM architecture is hierarchical and modular, and it enables sparse processing of information. HTM is divided into two parts: (1) Spatial Pooler (SP) and (2) Temporal Memory (TM) [34], [35]. The main purpose of the HTM SP is to encode the input data and produce its sparse distributed representation that finds application in various data classification problems. The HTM TM is primarily known for contextual analysis, sequence learning [36] and prediction tasks [37], [38]. The HTM SP consists of four main phases: (1) initialization, (2) overlap, (3) inhibition, and (4) learning. There are several hardware implementations proposed for the HTM SP, such as conventional HTM SP [39] and modified HTM SP [40]. Both architectures are based on memristive devices located in the initialization and overlap stages. The hardware implementation of the learning stage for HTM SP has not been proposed yet. According to [18], the backpropagation algorithm can be one of the approaches to updating weight in the HTM SP.

3) Long-Short Term Memory: LSTM is a cognitive architecture that is based on the sequential learning and temporal memory processing mechanisms in the human brain. LSTM processing relies on state change and time dependency of processed events. The LSTM algorithm is a modification of recurrent neural network that takes into account history of processed data and controls the information flow through gates [41], [42]. LSTM is used in a wide range of applications in the contextual data processing based on prediction making and natural language processing. Hardware implementation of LSTM is a new topic studied in [43], [44].

B. Backpropagation with gradient descent

In this paper, an analog implementation of gradient descent backpropagation algorithm [45], [46] is proposed for different neural network configurations. The algorithm consists of four steps: forward propagation, backpropagation to the output layer, backpropagation to the hidden layer, and weight updating process. In this section, we present the main equations of backpropagation algorithm with gradient descent for a three-layer neural network with sigmoid activation function to relate it to the proposed hardware implementation.

In the forward propagation step, the dot product of input matrix X and weighted connections between input layer and hidden layer w_{12} is calculated and passed through the sigmoid activation function: $Y_h = \sigma(X \cdot w_{12})$, where Y_h is an output of the hidden layer [47], [48]. The forward propagation step is repeated in all the neural network layers. The output of the three-layer network Y_o is calculated as $Y_o = \sigma(Y_h \cdot w_{23})$, where w_{23} is the matrix representing the weighted connections between the hidden and output layers.

The backpropagation algorithm uses the cost function defined in Eq.1 for the calculation of derivative of error with respect to the change in weight. In Eq.1, E is an error, N is a number of neurons in the layer, y_{target} is an ideal output and y_{real} is the obtained output after the forward propagation [49].

$$E = \frac{1}{2} \sum_{i=1}^{N} (y_{target} - y_{real})^2 \qquad (1)$$

Equation 2 shows the calculation of the derivative of output layer error E_o, where δ denotes the rate of change of the error with respect to the weight w_{23} [47], [48]. The error for the output layer e is calculated as a difference between the expected neural network output Y and real output of the network Y_o: $e = Y - Y_o$. The derivative of the sigmoid function is the following: $\frac{\partial Y_o}{\partial w_{23}} = Y_o(1 - Y_o)$.

$$\frac{\partial E_o}{\partial w_{23}} = Y_h \cdot \delta_2 = Y_h \cdot (e \odot \frac{\partial Y_o}{\partial w_{23}}) \qquad (2)$$

The derivative of error for the hidden layer is shown in Eq. 3, where X' is an inverted input matrix and e_h is the error of the hidden layer. The error e_h is calculated propagating back δ_2 as following: $e_h = \delta_2 \cdot w'_{23}$. And the derivative of the hidden layer output E_h is the same as in the output layer: $\frac{\partial Y_h}{\partial w_{12}} = Y_h(1 - Y_h)$ [50].

$$\frac{\partial E_h}{\partial w_{12}} = X' \cdot \delta_1 = X' \cdot (e_h \odot \frac{\partial Y_h}{\partial w_{12}}) \qquad (3)$$

In the final stage, the weight update calculation is performed using Eq. 4 and Eq. 5, where η is the learning rate responsible for the speed of convergence. The optimized learning rate depends on the type and number of inputs and number of the neurons in the hidden layers [48], [50].

$$\Delta w_{23} = \frac{\partial E_o}{\partial w_{23}} \times \eta \qquad (4)$$

$$\Delta w_{12} = \frac{\partial E_h}{\partial w_{12}} \times \eta \qquad (5)$$

The weight matrices are updated considering the calculated change in weight: $w_{23\_new} = w_{23} + \Delta w_{23}$ and $w_{12\_new} = w_{21} + \Delta w_{12}$.

III. BACKPROPAGATION WITH MEMRISTIVE CIRCUITS

A. Overall architecture

This subsection illustrates the overall implementation of the backpropagation algorithm on hardware, while the details of the implementation of the activation functions and particular blocks are shown in Section III-B.

The proposed hardware implementation of the learning algorithm is illustrated in Fig. 1. Depending on the application requirements and limitations of the memristive devices, the inputs to the system can be either binary or non-binary. For example, for the HTM applications, the inputs are non-binary [40], whereas, for the binary neural network, inputs can be binary [33]. The outputs of the neural network can also be binary or non-binary, depending on the activation function. Memristive crossbar arrays emulate the set of synapses between the neurons in the neural network layers. The synapses can also be binary or non-binary depending on the applications and practical limitations of programming states of memristor device. While an ideal non-volatile memristor device can store and be programmed to any particular value between R_{ON} and R_{OFF}, the real memristor devices can have problems with switching to the intermediate resistive values. It is easier and simpler to switch the memristor to either R_{ON} and R_{OFF} state. The implementation of the analog weights is also possible using 16-level $Ge_2Sb_2Te_5$ (GST) memristors [51]. However, the memristor technology is not mature like CMOS, and even if the memristor can be precisely programmed and work accurately under the controlled environment in the lab, the behavior of the memristor in the multi-level large-scale simulation still needed to be verified. Therefore, the binary synapses are the easiest to be implemented.

The example shown in Fig. 1 demonstrates the basic three-layer ANN with the proposed backpropagation architecture, control circuit and weight update circuits. The neural network has three input neurons, two output neurons and five neurons in a hidden layer. The operation of the crossbar and switching between forward propagation, backpropagation and weight update operations is controlled by the switching transistors M_{in}, M_u and M_r, which in turn are controlled by the
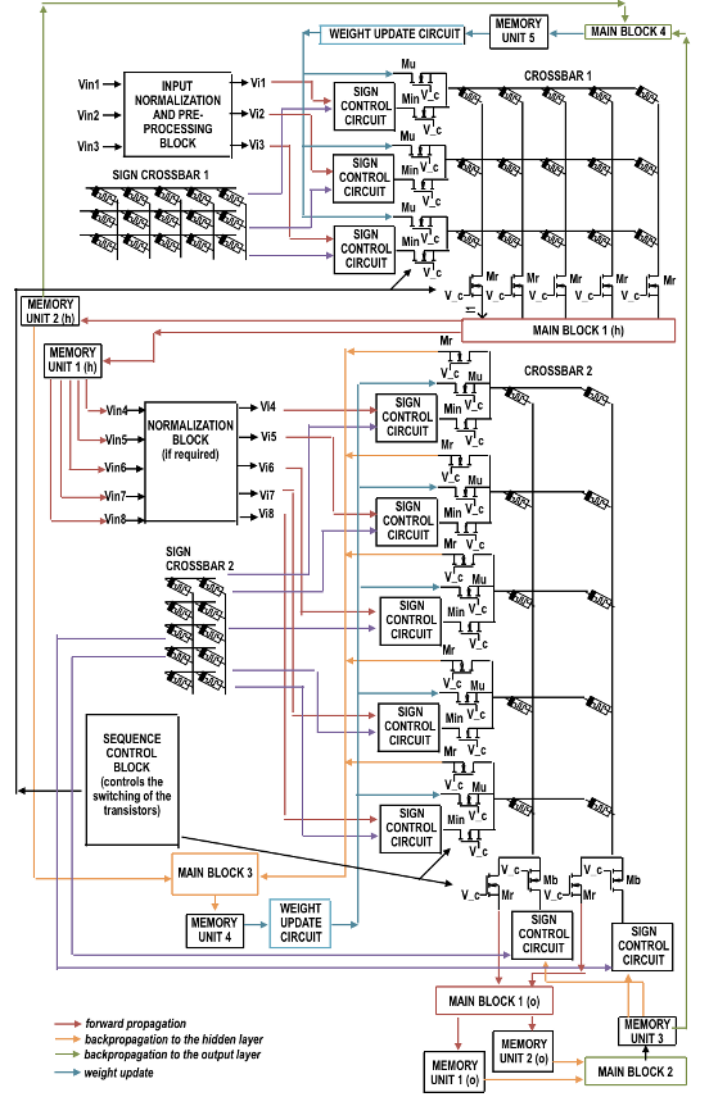

Fig. 1. Overall architecture of the proposed analog backpropagation learning circuits for memristive crossbar neural networks. In the forward propagation process, MAIN BLOCK 1 (MB1) is involved. The backpropagation through the output layer is performed by MAIN BLOCK 2 (MB2) and MAIN BLOCK 4 (MB4). The backpropagation through the hidden layer is performed by MAIN BLOCK 3 (MB3). The weight update process of the output layer and the hidden layer is performed by MB4 and MB3, respectively. The blocks with the notation (o) correspond to output layer and the block with notation (h) correspond to hidden layer.

sequence control block. The CROSSBAR 1 corresponds to the set of synaptic connections between the input layer and the hidden layer, and the CROSSBAR 2 represents the synapses connecting the hidden layer to the output layer. The three input signals are shown as V_{in1}, V_{in2} and V_{in3}, and the corresponding normalized input signals are shown as V_{i1}, V_{i2} and V_{i3}, respectively. The range of output signals from the normalization circuit depends on the application, limitations of memristors and linearity of the switch transistors. The inputs are fetched to the rows of the CROSSBAR 1 i.e. to the input switching transistors M_{in}. Each memristor in a single column of the crossbar corresponds to the connections of all inputs to a particular single output. The crossbar performs dot product multiplication of inputs and the weights of a single column.

The output of the multiplication for feed-forward propagation is read from the NMOS read transistor M_r connected to a crossbar column. In this work, we investigate the approach, when the output of the crossbar is represented by a current flowing through the read transistor. However, the configuration can be changed to the voltage output from the crossbar column if required. The voltage-based approach is more complicated than current-based approach because the amplifier or OpAmp-based buffer is required to read the voltages without the loading effect from the following interfacing circuits. The current-based approach requires only the use of a current mirror ensuring the reduction of the on-chip area, and also compatible with simple current driven sigmoid implementation.

The parameters of input transistors M_{in}, read transistors W_r and corresponding control signals V_c should be selected carefully, considering the range of input signal. When the signals are propagated back through in the CROSSBAR 2, the propagated error can be both positive and negative. Therefore, if it is important to make sure that the size of the transistors and V_c are set to eliminate the current when transistor in OFF state and conduct the current in linear region when transistor is ON. These parameters should be adjusted depending on the technology.

The outputs from the crossbar columns are read sequentially one at a time to avoid the interference with the currents from the other columns. The read-transistors M_r at the end of each column are used to switch ON and OFF the columns and maintain the order of the reading sequence for the forward propagation process. As the negative resistance is not practical to implement in the memristive device, the negative weights are implemented using either input controller, which changes the input sign according to the sign of the synapse weight, or by the sign control circuit and additional crossbar that stores the sign of the weights. Fig. 1 illustrates the approach with sign control circuit and additional SIGN CROSSBAR 1 and SIGN CROSSBAR 2.

MAIN BLOCK 1 (MB1) in Fig. 1 performs the forward propagation and is responsible for the calculation of the activation function. MB1 (h) and MB1 (o) correspond to hidden layer and output layer, respectively. Depending on the application, the activation function can include the sigmoid, derivative of the sigmoid, tangent, derivative of the tangent, approximate sigmoid and approximate tangent functions. The approximate functions represented here refer to "hard" logical threshold sigmoid and tangent as shown in [50]. To implement the conventional backpropagation algorithm with gradient descent, MB1 performs the calculation of the sigmoid and the sigmoid derivative functions. The outputs of MB1 are stored in MEMORY UNIT (MU) 1(h) and used as the inputs to CROSSBAR 2. The outputs of MU1(h) can be normalized by normalization circuit. The output currents from the second crossbar are fetched to the second MB1 and the final output of the feed-forward propagation are obtained from MB1(o) and stored in MU1(o) for further application in backpropagation. The final outputs depend on the activation function. In addition, MB1 produces the outputs of the activation function, stored in MU1(h) and MU1(o), and derivative of the activation function, stored in MU2(h) and MU2(o) that are useful for

backpropagation process.

After the forward propagation process, the backpropagation process is implemented. The sequence control block switches off the column read transistors M_r of both crossbars and switches on the row read transistors V_r of CROSSBAR 2. The column transistors M_{in} of CROSSBAR 2 are switched ON to propagate the inputs MU3, which stores the outputs of MAIN BLOCK 2 (MB2) corresponding to the propagation through the output layer. It ensures that the propagation is performed in the opposite direction. If the neural network contains more than three layers, all crossbars except the first crossbar corresponding to the synaptic weights between the input and the first hidden layer are reconnected to perform backpropagation operation. The backpropagation process through the output layer is implemented using MB2 and MAIN BLOCK 4 (MB4), and the backpropagation through the hidden layer corresponds to the MAIN BLOCK 3 (MB3). The possible architecture of analog memory unit is illustrated in [40], [52], [53].

The final stage in the backpropagation algorithm is the weight update stage, where the values of the memristors are updated based on the specific rules. As the crossbar values are read and processed sequentially, MEMORY UNIT 4 and 5 store the update value before the update process starts. The weight update process is implemented by applying the voltage pulse of a particular duration and amplitude across each memristor. The update pulse depends on the required change of the weights, calculated gradient of error, and the memristor type and technology. The amplitude of the update pulse depends on the outputs of MB2 and MB4 and calculated by weight update circuit. While the duration of the pulses are controlled by the sequence control circuit. The update process of the memristor is controlled by transistors M_{in} and M_r. To update the memristor, corresponding row transistors M_{in} and column transistor M_r are switched ON. As the main architecture for the synapses that we consider in this work is 1M, each memristor is updated either one at a time. However, this process is slow, and in particular cases memristor weight can be updated in several cycles as illustrated in [54]. If the two state memristors are used in the crossbar (R_{ON} and R_{OFF}, which is useful for binary neural network), the update process can be performed in two cycles: (1) update of all R_{ON} memristors, which should be switched to R_{OFF} and (2) update of all R_{OFF} memristors, which should be switched to R_{ON}. This method is useful, when the other states between R_{ON} and R_{OFF} are not important for processing. Such method increases the speed of update process, however it is useful for neural network with binary synapses.

B. Circuit-level implementation of main blocks in backpropagation algorithm

This section briefly introduces the proposed architecture for backpropagation circuits shown in [15], while the detailed explanation of the circuit and all circuit parameters are provided in Supplementary Material (Section III). In this work, the analog circuits for the proposed backpropagation implementation are designed for 180nm CMOS process. The circuit level implementation of all backpropagation blocks is

illustrated in Fig. 2, while the components used in the circuits are shown in Fig. 3. MB1 performs the forward propagation for the conventional backpropagation architecture. MB2 performs the backpropagation process through the output layer, which is finished by MB4. MB3 performs the backpropagation through the hidden layer. The circuit level implementations of components from the main backpropagation blocks are shown in Fig. 3. Fig. 3(a) illustrates the implementation of current buffer. Fig. 3(b) shown the implementation of adjusted sigmoid function proposed in [55]. Fig. 3(c) shows the OpAmp circuit. Fig. 3 (d) illustrated the multiplication circuit based on the current difference in transistors M_{31} and M_{32}. Fig. 3 demonstrated the implementation of analog switch circuit.

C. Sign control circuit

As the neural network weights can be both positive and negative, and the negative weight cannot be practically implemented by the memristor, the implementation of the additional weight control circuit is required. For each negative weight, the sign of the input voltage is changed. There are two possible ways to implement the sign. One of the solutions is to store the sign for each sequence in the external storage unit and apply it to the circuit with the weight normalization circuit. The other solution is to store the sign of each weight in the additional memristive crossbar elements. A memristive weight sign control circuit shown in Fig. 4 is proposed.

The sign of each memristor in the crossbar is stored in the memristive crossbar or separate memristors as R_{ON} or R_{OFF}. The analog sign read circuit follows the memristor storing the sign of the weight. There are two possible solutions. The first is to implement a single analog sign read circuit and switch it between the memristors in the crossbar, which requires additional on-chip area and storage. And the second and more effective solution is to implement the number of sign read circuits equivalent to the number of rows in a crossbar, which allows reading the sign of all the memristors in a single column. This allows achieving the trade-off between the required area, power and processing time. In Fig. 4, the sign of the memristor representing the weight is stored in the memristor R_{sign}. When the sign is read, $V_c = 1.25V$ is applied. If R_{sign} is set to R_{ON}, the output of the analog switch V_{sign} is positive and vice versa. The weight sign read circuit acts as a switch. If $R_{sign} = R_{ON}$, the voltage V_c is above the switching threshold, it selects, and outputs the positive voltage V_{in}, which is the input voltage to the crossbar. If $R_{sign} = R_{OFF}$, the voltage drops and V_c is below the switching threshold, and the switch outputs the voltage $-V_{in}$. The parameters of the transistors are the following: $M_{61} = 0.18\mu/0.72\mu$, $M_{62} = 0.18\mu/10.36\mu$ and $M_{63} = 0.18\mu/0.36\mu$. The transistors in the circuit have an underdrive voltage $V_{DD} = 1V$.

Comparing to the existing implementations of negative voltages in the crossbar array [56], [57], [58], the method for sign control reduces the complexity of the implementation and ensures the stability of the output. For example, the crossbar in [57] can perform dot product multiplication for both positive and negative signals. However, the system is complex because of the amplifiers that perform subtraction

of the voltages. To ensure the amplification is not affected by the following circuits, such amplifier should include the capacitor, which increases the on-chip area of the circuit. Also, this way of implementing negative signals require the accuracy preprocessing stage and additional adjustment of the input signals. A similar method of implementing negative sign is shown in [58]. It involves a set of summing amplifiers with resistors, which also consume a large amount of area.

D. Weight update circuit

The implementation of memristive weight update circuit is illustrated in Fig. 5. The weight update circuit determined the pulse amplitude required to program the memristor in a crossbar array, depending on the calculated weight update value by MB2 and MB4. The circuit is adaptable for different learning the due to the application of memristive devices R_{40} and R_{41} in the amplifiers. All the resistors in weight update circuit are $1k\Omega$ and the memristors are programmed considering the required learning rate. As the negative (to switch from R_{OFF} to R_{ON}) and positive (to switch from R_{ON} to R_{OFF}) programming amplitudes are not of the same amplitude for memristive devices, the analog switch selects the amplitude of the signal based on sign of the input voltage from MB1 and MB4. The implementation of analog switch is shown in Fig. 3 (e) The shifted input signal is applied to analog switch control V_c that determines, which input to the switch should be selected V_{SW1} or V_{SW2}. The input to V_{SW1} corresponds to the positive input voltage, while V_{SW2} corresponds to the negative input voltage.

E. Modular approach

As large scale crossbars usually suffer from leakage currents, the most widely used architecture for the crossbar synapses is 1 transistor 1 memristor $1T1M$ [20], [59], [60]. Different variants of transistors and selector devices are used in the literature for the crossbar architecture for improving the crossbar performance. Architecture based in 1T1M synapses allows to remove the leakages which cause the reduction of output current in read transistors. However, this architecture of the synapses has significantly larger on-chip area and power consumption, comparing to single memristor (1M) crossbar architectures. In this paper, we avoid the application of $1T1M$ synapses and investigate the application of $1M$ synapses to maintain small on-chip area and low power consumption, and use the modular approach to reduce the leakage current problem and make the programming of the memristive arrays less complicated. As illustrated in [61], modular approach allows to reduce the leakage currents in the crossbar. In this approach, the large crossbar is divided into smaller crossbars as shown in Fig. 6, and the current from all modular crossbars is summed up to process through the activation function in MB1. As illustrated in simulation results, this approach allow to achieve similar performance accuracy, as single crossbar approach.

In addition, if the network is scaled, the sequential processing can introduce the limitation to the system in the form of reduced processing speed. In this case, the parallel processing
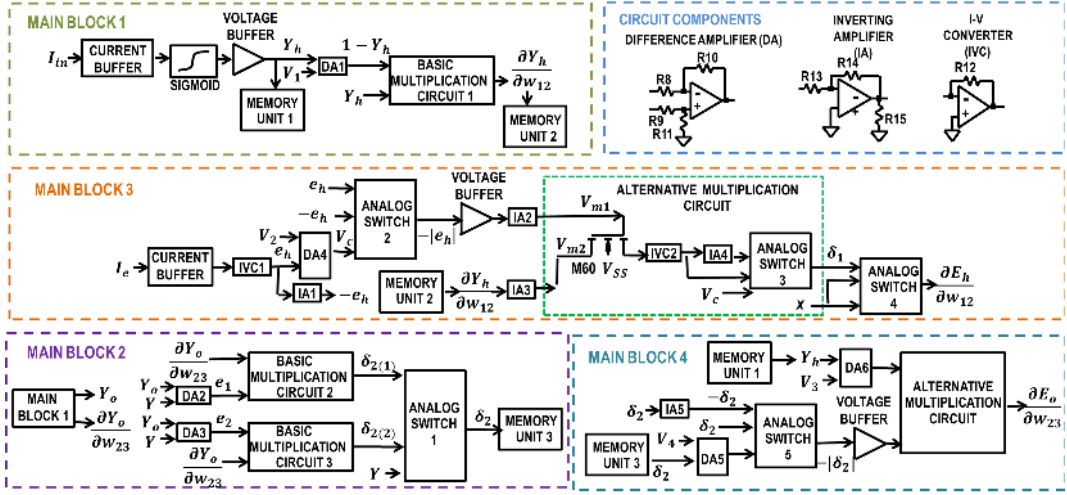
Fig. 2. The circuit level architecture of the proposed backpropagation implementation. The separate implementation of the MB1, MB2, MB3 and MB4 is illustrated. In addition, the involved circuit components, such as DA, IA and IVC, are shown.
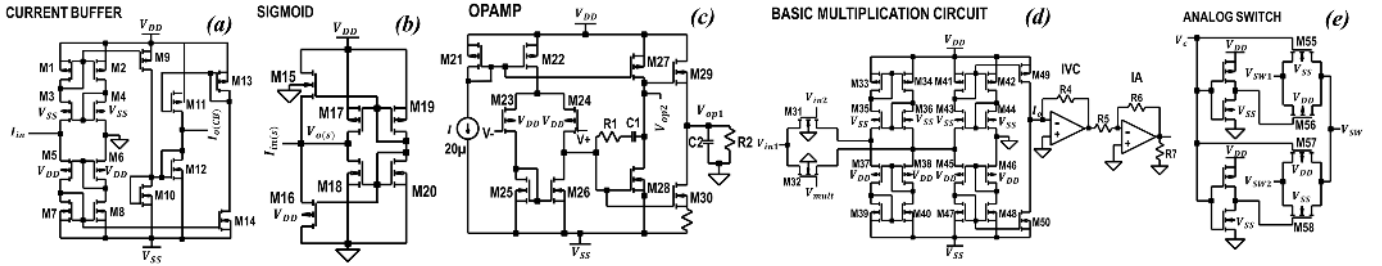

Fig. 3. Circuit components: (a) The current buffer circuit which is connected to the read transistor M_r in Fig. 1. The circuit is used in MB1 and MB3 to eliminate the loading effect of the activation function to the performance of the crossbar. (b) Sigmoid activation function used in MB1 inspired from [55]. (c) Two stage OpAmp design used for all OpAmp based components in the proposed analog memristive learning circuit. (d) Multiplication circuit based on the Hilbert multiplier principle. The circuit is used in MB1 and MB2. (e) Analog switch design used in MB2, MB3 and MB4.
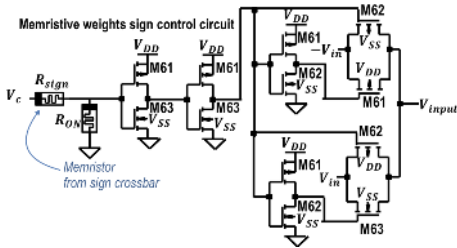

Fig. 4. Memristive weight sign control circuit that can be integrated to the crossbar to control the weight of the synapse or applied as an external circuit with a separate memristors to store the sign of the weight.
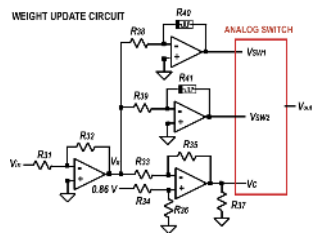

Fig. 5. Memristive weight update circuit that converts the weight update value from MB3 and MB4 to the pulse of particular amplitude. The used analog switch is shown in Fig. 3 (e).

can be introduced, which involves the concurrent computation and simultaneous execution of the output computations. The modular approach can be useful as well, however, each modular crossbar should have corresponding processing blocks for backpropagation algorithm (MB1, MB2, MB3 and MB4). This introduces additional complexity for the system and increases on-chip area and power but reduces the processing time. Modular approach may also allow to remove the analog storage units. As the size of the crossbar will be reduced, a time delayed signal produced by a signal delay circuit can be used instead of analog storage unit.

IV. LEARNING ARCHITECTURES

The proposed analog memristive backpropagation learning circuits can be used for various applications and learning architectures, such as neural networks, HTM and LSTM hardware implementations. To apply the proposed backpropagation circuits for various architectures, the implementation of additional functional blocks and activation functions is required. In this section, we illustrate the implementations of tangent, current and voltage driven approximate sigmoid and tangent, and linear activation functions and thresholding circuit to normalize the output of the neural network.

To implement the tangent function, the sigmoid function can be adjusted. The use of the sigmoid circuit (Fig. 3(b)) allows
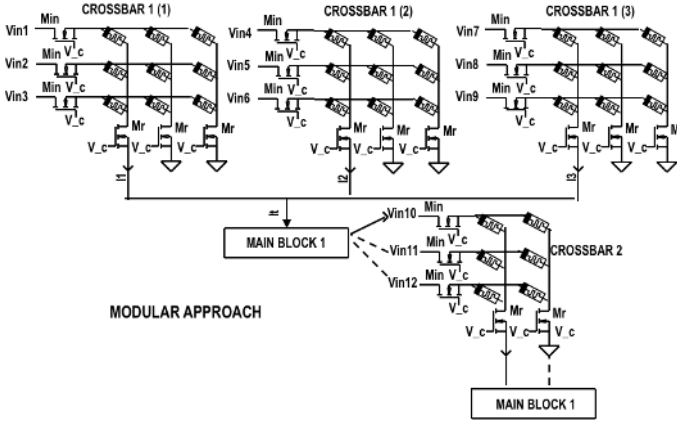
Fig. 6. Modular approach to reduce the leakage current and complexities in programming of 1M array.

building a single circuit for both of the functions and switch between the sigmoid and tangent implementations when it is required. The implementation of the tangent function is shown in Fig. 7(a). The sigmoid and buffer part remain the same as in the sigmoid implementation and the voltage shift circuit based on the difference amplifier is added. The difference amplifier is based on the same OpAmp shown in Fig. 3(c) with $R_{16} = 10k\Omega$, $R_{17} = 1k\Omega$, $R_{18} = 2.5k\Omega$, $R_{19} = 15k\Omega$ and $R_{20} = 1k\Omega$. The circuit shifts the voltage level of the sigmoid and allows to implement a tangent function with the same circuit.

The implementation of an approximate sigmoid and tangent functions can be done with a simple thresholding circuit shown in Fig.7(b) and Fig.7(c). There are options: current-control and voltage-control approximate functions. The current-control circuit is shown in Fig. 7(b). The input current I_{in} is applied to the current-to-voltage converter based on the OpAmp with $R_{22} = 20k\Omega$ and inverted by the inverter with $M64 = 0.18\mu/0.36\mu$ and $M65 = 0.18\mu/1.72\mu$. The W/L ratio of $M64$ and $M65$ can be adjusted depending on the required transition part between the high and low value of approximate sigmoid and tangent functions. The voltages V_{DD1} and V_{SS1} are different for sigmoid and tangent implementations. For the approximate sigmoid $V_{DD1} = 1V$ and $V_{SS1} = 0V$, which means that the transistors have an under-drive voltage level for TSMS 180nm CMOS technology. In the approximate tangent implementation, $V_{DD1} = 1V$ and $V_{SS1} = -1V$. The simple thresholding circuits can implement the voltage-controlled sigmoid and tangent with two inverters shown in Fig. 7(c). The W/L transistor ratios of $W_{66}-W_{69}$ and voltage levels of V_{DD1} and V_{SS1} can be adjusted to obtain a required amplitude, range and transition region for the sigmoid and tangent.

The implementation of linear activation functions are shown in Fig. 7(d) and Fig. 7(e). Both units are driven by voltage and the OpAmp circuits are shown in Fig. 3(c). We propose the implementation of linear activation function based on analog switch shown in Fig. 7(d). The analog switch selects the $0V$ output for negative input signal, and V_{in} for positive input. The switch is controlled by the shifted inverted input signal which is fed to the switch control V_c. All values of resistances

are set to $1k\Omega$. The possibility to implement linear activation function is to use diode (Fig. 7(e)). The diode based linear activation function has smaller on-chip area and lower power consumption, however the output range is smaller comparing to linear activation function with switch. To implement a current controlled linear activation function, IVC can be used in both circuits. In linear activation unit based on the analog switch, OpAmp $R_{22} - R_{23}$ can be replaced by IVC. In diode based linear activation function. additional IVC component before OpAmp is required.

As we verified with the simulation results, if the ideal outputs is required to be binary, additional thresholding circuit is required in the output layer to normalize the outputs and achieve high accuracy. This was demonstrated using XOR problem in simulation results. The thresholding circuit that has been used for simulations is shown in Fig. 7(f), where the parameters of the transistors are $M66 = M68 = 0.36\mu/0.18\mu$ and $M67 = M69 = 0.72\mu/0.18\mu$. The thresholding function is connected to the output layer after the training process and allows to increase the accuracy significantly during the testing stage.

A. Neural Networks

There is a number of neural network architectures, where the proposed learning circuit can be used. The complete analog learning and training circuits for most of the architectures and networks covered in Section IV has not been implemented in analog hardware. There are different architectures and types of the neural networks that can use the proposed learning circuit without making a significant modification to the proposed design, such as DNN, BNN, and MNN.

1) Deep Neural Network: The proposed configuration for DNN with memristive analog learning circuits is shown in Fig. 8. The DNN configuration contains N+1 layers, and N crossbars correspond to the synapses between the layers. In the forward propagation process, MB1 is used. MB1 can be modified to implement various activation functions. MB2 and MB4 perform the backpropagation process through the output layer of DNN, and MB3 does the backpropagation through hidden layers. In the update process, MB4 and MB3 are applied. The blocks MB2, MB3 and MB4 in each layer can be modified depending on the activation function applied in forward propagation in MB1 for each layer.

2) Binary Neural Network: BNN can be implemented with the proposed circuit using two-stage memristors in the memristive crossbar representing the weights. The implementation of the BNN is shown in Fig. 9. In BNN, the forward propagation process and backpropagation process are the same as in a three-layer neural network shown in Section III. However, due to the limitations of the binary weights, the direct update of the weights after the error calculation will not provide high accuracy results. We suggest to store the value of the change in error in the external storage and training units in time and, after a certain period of the training, update the weights. This method can improve the accuracy results for the classification problems using BNNs.
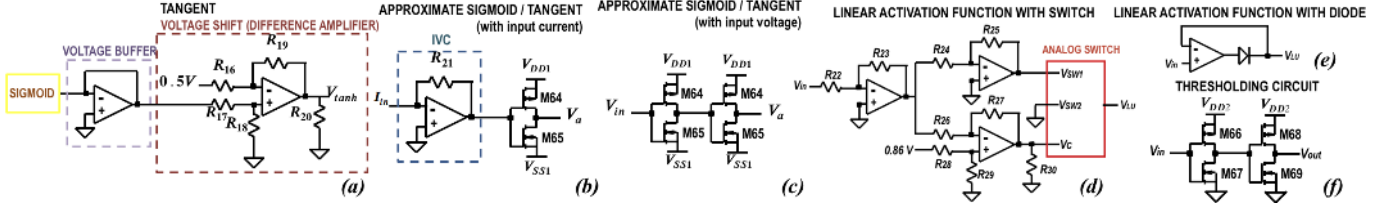
Fig. 7. Implementation of various activation functions for DNN: (a) Tangent function based on the sigmoid circuit from Fig. 3 (b). This architecture allows to implement a single circuit for both sigmoid and tangent functions in a multilayer neural network or another learning architecture and switch between these two functions. (b) Approximate sigmoid and approximate tangent functions driven by input current. To implement the sigmoid and tangent, the voltage levels V_{DD1} and V_{SS1} are varied. (c) Approximate sigmoid and approximate tangent functions driven by input voltage. (d) Linear activation function based on analog switch. (e) Linear activation function based on diode. (f) Additional thresholding circuit to normalize the neural network output for binary outputs.
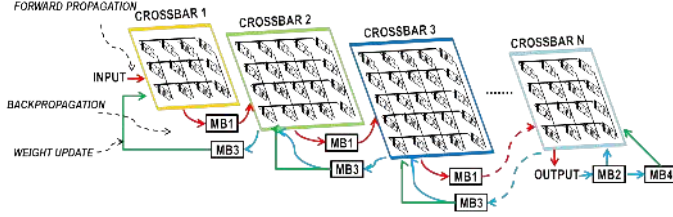

Fig. 8. Deep neural network implementation with the backpropagation learning. Red arrows correspond to forward propagation process. black arrows refer to backpropagation process. Green arrows show the weight update process.
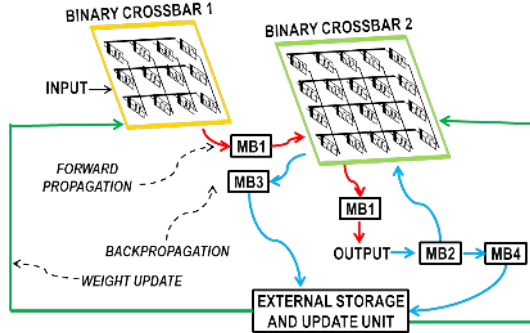

Fig. 9. Three layer BNN with backpropagation learning. The crossbars contain memristors that can be programmed only for R_{ON} and R_{OFF} stages. To improve the accuracy and the performance of the network, the change in error is stored in the external storage and update unit. The crossbar weights are updated based on several iterations in time.
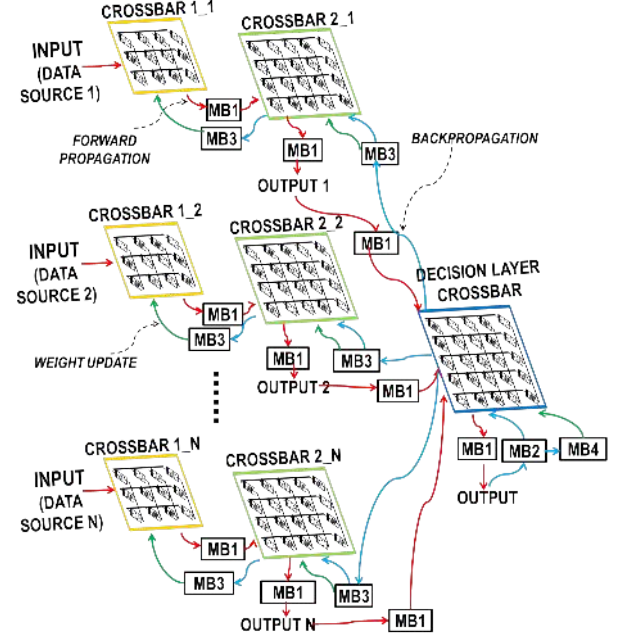

Fig. 10. Multiple neural network with backpropagation learning. The inputs from different data sources are fetched into different crossbars and the outputs from the crossbars are used as the inputs to the decision layer containing the memristive synapses.

3) Multiple Neural Network: The proposed memristive analog learning circuits can be used for the MNN approach. This is useful when several sources of input data are used, and the decision on the output depends on the fusion of the results from each data source. The output from the different data sources are normalized, and outputs of each data source are fetched to separate crossbars, and the outputs of all the crossbars are fed into the decision layer. The decision layer crossbar is the same as the other crossbars in the system. As MNN consists of several networks and the decision layer can be treated as a separate network, all the networks can be trained either separately [17] or as a single network. The architecture of MNN with backpropagation learning that is trained as a single network is illustrated in Fig. 10. We recommend using this approach, when the neural network

inputs are taken from different data sources, such as various sensors in the system. One of the examples of the use of such system is gender recognition, where voice signals and face images can be used as inputs to MNN. The activation functions of the layer are different for separate crossbars and the decision layer and depend on the data that is used for the processing. The number of the required backpropagation blocks is equaled to the number of the crossbars that a network contains.

B. Hierarchical Temporal Memory

The other learning architecture, where the proposed circuit can be used is HTM. There are several hardware implementations proposed for the HTM SP and HTM TM [62], [39], [40]. However, the learning stage of the HTM SP has not been implemented on hardware yet. This stage can include either update process based on Hebb's rule or the backpropagation update of the HTM SP weights [18].
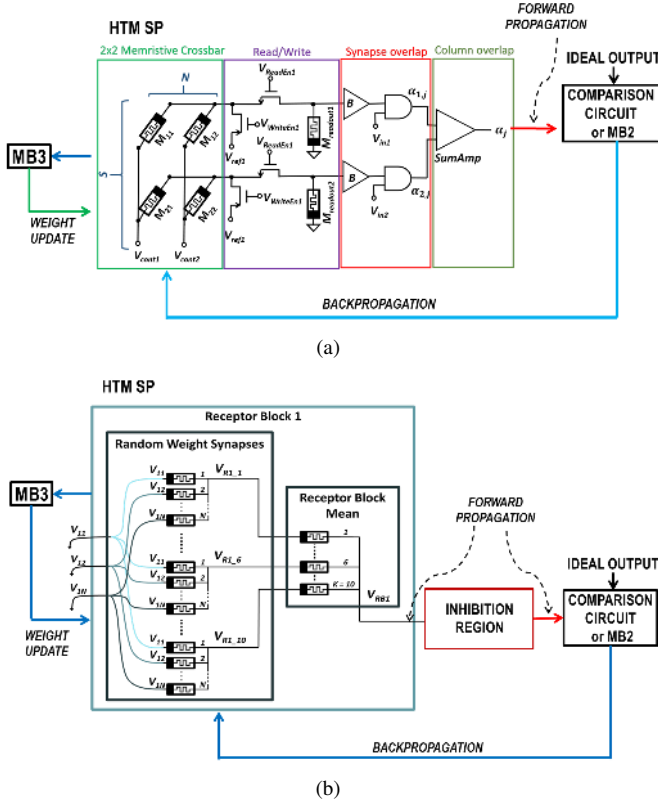
(a)

(b)

Fig. 11. Analog hardware implementation of the (a) conventional HTM SP algorithm and (b) modified HTM SP algorithm with backpropagation learning stage.

Fig. 12. Analog memristive hardware implementation of the LSTM algorithm.

There are two main analog architectures for the HTM SP: conventional HTM SP [39] and modified HTM SP [40]. The application of the proposed learning circuits for both architectures is shown in Fig. 11. Fig. 11(a) illustrates the application of the proposed learning architecture for the conventional HTM circuits. After the forward propagation through the HTM SP, the HTM SP output is compared to the ideal HTM output. In the conventional HTM SP circuit, the calculated error from the comparison circuit or MB2 is fetched back to the memristive crossbar to calculate the error in the weights, and the weights are updated. Fig. 11(b) shows the application of the proposed circuits for the modified HTM SP architecture. The conventional HTM SP architecture consists of the receptor and inhibition blocks. The weights of the synapses are located in the receptor block. After the comparison of the HTM SP output to the ideal output, the error is propagated back through the receptor block and MB3, and the memristive weights are updated.

C. Long Short Term Memory

LSTM architecture can also be implemented using the proposed memristive analog backpropagation circuits. The full implementation of LSTM with analog circuits has not been proposed yet. However, the analog implementation of the separate LSTM components has been shown in [43], [44]. Fig. 12 illustrates the full system level LSTM architecture consisting of the output gate, input gate, write gate and forget
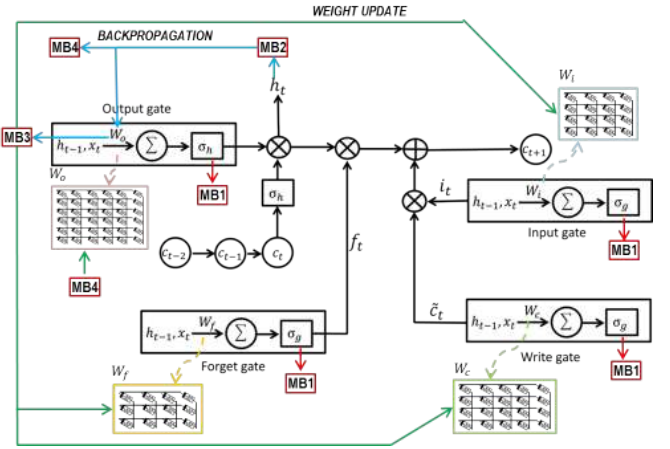
gate. The weights of LSTM gates W_i, W_o, W_f and W_c can be stored in the memristive crossbars. The activation functions in the LSTM architecture can be replaced with different variations of MB1. While, the weight update process of the crossbar W_o is performed by MB4 as the update of the output layer, while MB3 performs the update of W_i, W_f and W_c.

V. SIMULATION RESULTS

The circuit simulations were performed in SPICE, and the verification of the ideal backpropagation algorithm is done in MATLAB.

A. Circuit performance

The memristor model used in the crossbar simulations is Biolek's modified S-model [63] for HP TiO$_2$ memristor with the threshold voltage V_{th} of 1V [64]. This memristor model is developed for large-scale simulations to simplify the computation and processing [63]. The memristor characteristics and switching time for $R_{ON} = 3k\Omega$ and $R_{OFF} = 62k\Omega$ are shown in Fig. 13. Fig. 13 (b) and Fig. 13 (c) illustrate memristor updated process applying pulse of 1s with different amplitudes. The switching time is large, and speed of learning process with the memristive elements is slow. However, the learning and training process is a one-time process in the neural network. After the training during the testing stage, the reading time is small, and the data processing is fast.

Simulation results illustrating the performance of the circuits in terms of amplitude are shown in Supplementary Material (Section IV). The simulation results for additional activation functions are shown in Fig. 14. Fig. 14 (a) represents the simulation of the proposed tangent function. Fig. 14 (b) and 14 (c) illustrate the simulation of current driven approximate sigmoid and tangent, respectively. Fig. 14 (d) and 14 (e) show the simulation of linear activation functions with diode and switch, respectively. Fig. 14 shows that the activation function with switch is more linear. The timing diagram for the memristive weight sign control circuit is shown in Fig. 15. Fig. 15 (a) represents the positive input voltage. Fig. 15 (b) illustrates the ideal output and real memristive
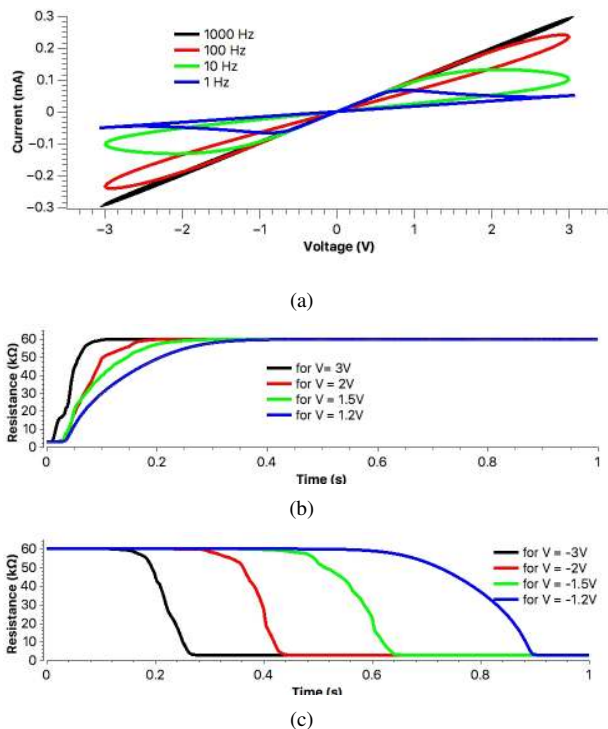
Fig. 13. Memristor characteristics: (a) hysteresis for different frequencies for $R_{initial} = 10k\Omega$, (b) switching time from $R_{ON} = 3k\Omega$ to $R_{OFF} = 62k\Omega$ for different applied voltage amplitudes, and (c) switching time from $R_{OFF} = 62k\Omega$ to $R_{ON} = 3k\Omega$ for different applied voltage amplitudes.

weight sign control circuit output for R_{ON}, when the weight is positive. Fig. 15 (c) illustrates the ideal output and real output of the proposed weight sign control circuit for R_{OFF}, when the weight is negative. The output of weight update circuit is shown in Fig. 16. The memristors is the circuit are programmed for high negative update amplitude and low positive update amplitude, as shown in Fig. 16 (c).

Table I represents the calculation of the on-chip area and maximum power dissipation for separate components for the analog learning circuit implementation and additional components and activation functions. Also, the example of the area and power dissipation for a small crossbar is shown.

Table II shows the on-chip area and maximum power dissipation for separate components for the main blocks of the proposed analog backpropagation learning circuit implementation.

B. System level simulations

The system level simulations have been performed for XOR problem and handwritten digits recognition for ANN and face recognition for DNN. For setup in XOR problem, there are 2 neurons in the input layers, 4 neurons in the hidden layer and 1 neuron in the output layer. During the training, input was selected randomly out of four possible inputs. The simulation results for XOR problem for different learning rates for ideal simulations and backpropagation circuit are shown in Fig. 17 for number of iterations $n = 50,000$. In the real circuit with non-ideal behavior, the error after $50,000$ iterations is 15%

Circuit component	Power consumption	On-chip area
Crossbar (4 input neurons and 10 output neurons)	$5\mu W$	$1.36\mu m^2$
Crossbar with control switches	$1200\mu W$	$115.3\mu m^2$
Weight sign control circuit	$195.1\mu W$	$16.64\mu m^2$
Sigmoid	$11.4\mu W$	$184.00\mu m^2$
Current buffer	$149.0\mu W$	$280.00\mu m^2$
OpAmp (maximum)	$39.8mW$	$2801.76\mu m^2$
Analog switch	$162.3\mu W$	$1.55\mu m^2$
Approximate current driven sigmoid/tangent	$52.9mW$	$2118.00\mu m^2$
Approximate voltage driven sigmoid/tangent	$41.2pW$	$0.40\mu m^2$
Linear activation units with diode	$963.7\mu W$	$244\mu m^2$
Linear activation unit with switch	$23.214mW$	$951.06\mu m^2$
Weight update circuit	$14.34mW$	$1269.63\mu m^2$

TABLE II
AREA AND POWER CALCULATIONS FOR THE MAIN BLOCKS OF THE
PROPOSED DESIGN AND TOTAL AREA AND POWER FOR THREE LAYER
NETWORK.

Configuration	Area (μm^2)	Maximum Power(mW)
MB1 (hidden layer)	4885.86	3.70
MB2 + MB1 (output layer)	8264.88	10.64
MB3	15238.69	61.78
MB4	9734.33	39.53
Total	**38123.76**	**115.65**

higher than in ideal simulations. We verified that it is caused by the non-ideal behavior of analog multiplication circuits, which will be improved further in the future work. The accuracy results for XOR simulation for the cases with and without thresholding circuits are shown in Table III.

The variability analysis for random offsets in memristor programming value is shown in Fig. 18 for learning rates $\eta = 0.15$, $\eta = 0.3$ and $\eta = 0.5$. The offset in the weight update value is represented as following: $w = w + (\Delta w + \Delta w \cdot x)$, where x is a random variation of the weight of particular percentage. This variation can be caused by non-ideal behavior of processing, update circuit and control circuits for update pulse duration. The architecture was tested for the variation of 50%, 100%, 200% and 300 %. The simulation results on Fig. 18 show that the variation in the update value does not have a significant effect on the performance of the architecture. Also, the offsets in update value affect the system with smaller learning rate ($\eta = 0.15$) more than the system with larger learning rate ($\eta = 0.5$). However, the value of error converges to small error in all the cases. Therefore, even the significant error in the memristor update value, does not effect the performance of the learning process. The final accuracy for 100,000 iterations is illustrated in Table III.

The performance analysis for the case of random mismatch in final memristor values after update is shown Fig. 19. The performance accuracy after 100,000 iterations for all cases is
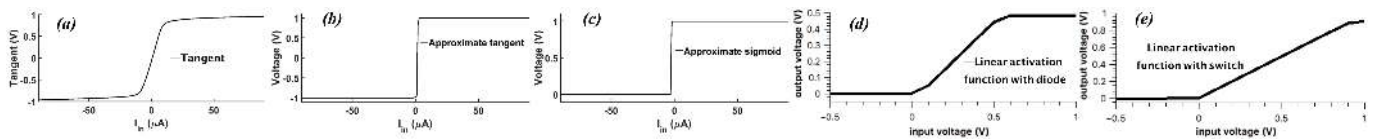
Fig. 14. Simulation of additional activation functions versus current: (a) tangent, (b) approximate current driven sigmoid and (c) approximate current driven tangent, (d) linear activation with diode, (e) linear activation with switch.
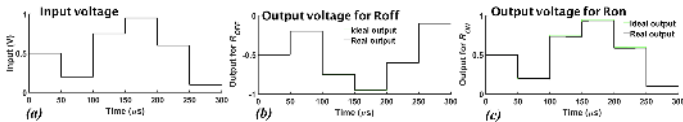

Fig. 15. Timing diagram for memristive weight sign control circuit implementation: (a) input to the circuit, (b) ideal output and real output for R_{ON} (when the weight is positive) and (c) ideal and real outputs for R_{OFF} (when the weight is negative).
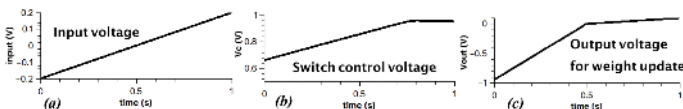

Fig. 16. Output of weight update circuit: (a) input voltage, (b) switch control voltage, (c) output of weight update circuit programmed to high amplitude voltage for negative update voltages (update from R_{OFF} to R_{ON}) and low amplitude voltage for positive update voltages (R_{ON} to R_{OFF}).
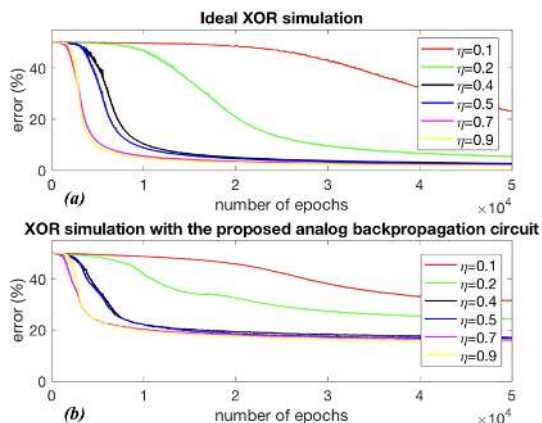

Fig. 17. Error rate versus number of iterations for (a) simulations ideal algorithm and (b) simulation of proposed backpropagation circuit.
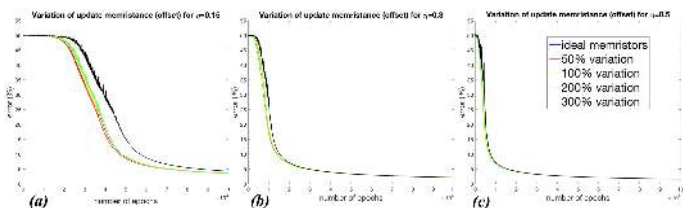

Fig. 18. Effect of the offset of the memristor update value on the performance of the architecture for (a) $\eta = 0.15$, (b) $\eta = 0.3$ and (c) $\eta = 0.5$.

shown in Table III. The mismatch is defined as following: $w = (w+\Delta w)+(w+\Delta w) \cdot x$, where x is the percentage of variation, shown in Fig. 19 as 1%, 2%, 4% and 5%. This mismatch has more significant effect on the performance of the architecture. For the small learning rate (Fig. 19 (a)), the mismatch in the

memristor values does not allow system to converge. For the larger learning rates, the system converge slower that in ideal case for 1-2% of mismatch and does not converge for larger mismatches. However, such case is the effect of the non-linear behavior and instability of memristive device, which should be investigated further at the device level.
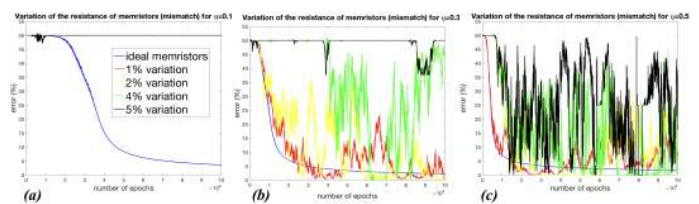

Fig. 19. Mismatch in the final weight of memristor of 1%, 2%, 4% and 5% for (a) $\eta = 0.15$ (b) $\eta = 0.3$ (c) $\eta = 0.5$.

To verify the performance of the proposed approaches for real pattern recognition problems, we tested ANN for handwritten digits recognition and DNN for face recognition for 2 approaches: single crossbar (shown in Fig. 1) and modular crossbar (shown in Fig. 6) using $Ge_2Sb_2Te_5$ (GST) memristors with 16 resistive levels [65], [51]. In ANN simulation, MNIST database [66] with $70,000$ images of the size of 28×28 was used, where 86% of images was selected for testing and 14% for testing. The setup for ANN consisted of 28×28 input layer neurons, 42 hidden layer neurons and 10 output neurons (corresponding to 10 classes of digits). In the modular approach, 16 crossbars with 49 input neurons, 8 crossbars with 98 input neurons and 4 crossbars with 196 input neurons were tested. For DNN verification, we performed face recognition task using Yale database for face recognition with 165 images of 15 people [67]. The images were rescaled by the size of 32×32, and 45% of the dataset was used for training and 55% for testing. The DNN configuration consisted of 6 layers of 1024, 800, 500, 100, 30 and 15 neurons. The simulation results are shown in Table IV. As the accuracy for all modular configurations is approximately the same, the modular crossbar approach is presented by a single value in the table. The simulation results show that the performance accuracy for both real ANN and DNN is reduced slightly, comparing to ideal case. As the obtained accuracy is the same and the research work [61] shows that the leakage currents are reduced in modular approach, the crossbar with 1M synapses can be divided into modular crossbars to avoid 1T1M synapses and reduce the on-chip area of the crossbar.

VI. DISCUSSION

The proposed analog hardware implementation of the backpropagation algorithm can be used to implement the online

TABLE III
ACCURACY FOR XOR SIMULATIONS (2 INPUT NEURONS, 4 HIDDEN
LAYER NEURONS AND 1 OUTPUT NEURON).)

Configuration for XOR simulation		ANN accuracy (without thresholding circuit)	ANN accuracy (with thresholding circuit, $\theta = 0.5$)
Ideal memristors $\eta = 0.15$, $n = 50,000$		84.8%	100%
Ideal memristors $\eta = 0.15$, $n = 100,000$		96.26%	100%
Ideal memristors $\eta = 0.3$, $n = 100,000$		97.76%	100%
Ideal memristors $\eta = 0.5$, $n = 100,000$		98.31%	100%
Offset in memristors programming value $\eta = 0.15$, $n = 100,000$	50%	96.10%	100%
	100%	96.10%	100%
	200%	96.07%	100%
	300%	96.10%	100%
Offset in memristors programming value $\eta = 0.3$, $n = 100,000$	50%	96.48%	100%
	100%	96.48%	100%
	200%	96.41%	100%
	300%	95.72%	100%
Offset in memristors programming value $\eta = 0.5$, $n = 100,000$	50%	98.56%	100%
	100%	98.58%	100%
	200%	98.56%	100%
	300%	98.33%	100%
Random mismatches in memristor value $\eta = 0.15$, $n = 100,000$	1%	50.02%	50%
	2%	50.82%	50%
	4%	50%	50%
	5%	50%	50%
Random mismatches in memristor value $\eta = 0.3$, $n = 100,000$	1%	99.7%	100%
	2%	99.89%	100%
	4%	56.73%	62.5%
	5%	50%	50%
Random mismatches in memristor value $\eta = 0.5$, $n = 100,000$	1%	91.77%	100%
	2%	80.29%	100%
	4%	99.22%	100%
	5%	63.23%	87.5%

TABLE IV
ANN ACCURACY FOR HANDWRITTEN DIGITS RECOGNITION APPLICATION
AND DNN ACCURACY FOR FACE RECOGNITION APPLICATION.

Configuration	ANN accuracy (MNIST, handwritten digits)	DNN accuracy (Yale, face recognition)
Ideal simulations	93%	78.9%
Single crossbar	92%	73.3%
Modular crossbar	92%	75.5%

training of different learning architectures, which can be used for near-sensor processing. The analog memristive learning architecture allows removing the additional software based or digital offline training and learning process. This can increase the processing speed and reduce the processing time, comparing to digital analogies, where the number of components to achieve high sampling rates in analog-to-digital converters (ADC) and digital-to-analog converters (DAC) are large. The possible errors in the training caused by leakage currents and parasitic effects can be mitigated by the increase of the number of iterations in the learning stage.

If the sneak path problems occur during the training state and the memristor update value is not accurate, this can be fixed in the following learning stages, but more update iterations are required for error to converge and reach high

accuracy in this case. As demonstrated in the XOR simulation, the problem of the non-ideal variation of the update value of the memristor due to non-ideal performance of the circuit and other device instabilities can be eliminated by increasing number of training iterations.

The limitations of the proposed architecture include the scalability of the memristive crossbar arrays and limitations of the current memristive devices. The problems of the parasitics, leakage current and sneak paths in the memristive crossbar have to be investigated further. The other drawback is a limitation of the memristive devices in terms of the number of resistance levels that can be achieved for particular memristive devices. The future work will include the implementation of the crossbar with a physically realizable memristor [68], evaluation of the performance of the architecture with different memristive devices, evaluation of the abilities of different memristive devices and adjustment of circuit parameters for particular devices.

In addition, the limitations of the memristive devices, electromagnetic effects, frequency effects and their effect on the accuracy and the performance of the proposed learning architecture have to be studied. Also, the endurance of the memristive devices should be studied, especially for the case of several iterations in the learning process.

The testing of the complete systems for large scale problems has to be performed and the limitations, such as loading effects and parasitics, have to be identified from the physical design constraints perspective. The effect of the additional components of the overall system performance and processing speed has to be determined under such conditions that become technology specific issues. The future work will include the full circuit implementation of the proposed HTM, LSTM and MNN architectures and verification of their performance for large scale problems.

VII. CONCLUSION

In this paper, we presented the circuit design of an analog CMOS-memristive backpropagation learning circuit and its integration to different neural network architectures. The circuit architectures are presented for a three-layer neural network, DNN, BNN, MNN, conventional and modified HTM SP and LSTM. We presented the analog circuit implementation of interfacing circuits and activation functions that can be used to implement various learning architectures. The implementation of backpropagation with analog circuits offers simplicity of building differential operations combined with a dot-product operator as memristor crossbar that is useful of building neural networks. Using databases of MNIST (character recognition) and Yale (face recognition) an application level validation of the proposed learning circuits for ANN and DNN architectures is successfully demonstrated. The presented design of crossbar does not take into account physical design issues of memristive devices, while sneak path problem of crossbar arrays is accounted in the simulations by including conductance variability of real memristor devices and wire resistors in the crossbar. However, the signal integrity issues is a topic to investigate further, when the memristor technology is mature

and is suitable for a fabricating reliable large-scale arrays. The area and power of the proposed circuit design need to be further optimized a fully parallel implementation for real-time applications.

References

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[2] O. Vermesan, M. Eisenhauer, H. Sunmaeker, P. Guillemin, M. Serrano, E. Z. Tragos, J. Valino, A. van der Wees, A. Gluhak, and R. Bahr, "Internet of things cognitive transformation technology research trends and applications," *Cognitive Hyperconnected Digital Transformation; Vermesan, O., Bacquet, J., Eds*, pp. 17–95, 2017.

[3] B. Hoang and S.-K. Hawkins, "How will rebooting computing help iot?" in *Intelligence in Next Generation Networks (ICIN), 2015 18th International Conference on*. IEEE, 2015, pp. 121–127.

[4] P. Narayanan, A. Fumarola, L. L. Sanches, K. Hosokawa, S. C. Lewis, R. M. Shelby, and G. W. Burr, "Toward on-chip acceleration of the backpropagation algorithm using nonvolatile memory," *IBM Journal of Research and Development*, vol. 61, no. 4/5, pp. 11:1–11:11, July 2017.

[5] M. Cheng, L. Xia, Z. Zhu, Y. Cai, Y. Xie, Y. Wang, and H. Yang, "Time: A training-in-memory architecture for memristor-based deep neural networks," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2017, pp. 1–6.

[6] G. W. Burr, R. M. Shelby, A. Sebastian, S. Kim, S. Kim, S. Sidler, K. Virwani, M. Ishii, P. Narayanan, A. Fumarola *et al.*, "Neuromorphic computing using non-volatile memory," *Advances in Physics: X*, vol. 2, no. 1, pp. 89–124, 2017.

[7] D. Soudry, D. D. Castro, A. Gal, A. Kolodny, and S. Kvatinsky, "Memristor-based multilayer neural networks with online gradient descent training," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 10, pp. 2408–2421, Oct 2015.

[8] R. Hasan, T. M. Taha, and C. Yakopcic, "On-chip training of memristor based deep neural networks," in *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017, pp. 3527–3534.

[9] S. Kim, T. Gokmen, H.-M. Lee, and W. E. Haensch, "Analog cmos-based resistive processing unit for deep neural network training," *arXiv preprint arXiv:1706.06620*, 2017.

[10] H.-y. Tsai, S. Ambrogio, P. Narayanan, R. M. Shelby, and G. W. Burr, "Recent progress in analog memory-based accelerators for deep learning," *Journal of Physics D: Applied Physics*, 2018.

[11] Y. Zhang, X. Wang, and E. G. Friedman, "Memristor-based circuit design for multilayer neural networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2017.

[12] D. Negrov, I. Karandashev, V. Shakirov, Y. Matveyev, W. Dunin-Barkowski, and A. Zenkevich, "An approximate backpropagation learning rule for memristor based neural networks using synaptic plasticity," *Neurocomputing*, vol. 237, pp. 193–199, 2017.

[13] Y. Zhang, X. Wang, and E. G. Friedman, "Memristor-based circuit design for multilayer neural networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 2, pp. 677–686, Feb 2018.

[14] Y. Zhang, Y. Li, X. Wang, and E. G. Friedman, "Synaptic characteristics of ag/aginsbte/ta-based memristor for pattern recognition applications," *IEEE Transactions on Electron Devices*, vol. 64, no. 4, pp. 1806–1811, April 2017.

[15] O. Krestinskaya, K. N. Salama, and A. P. James, "Analog backpropagation learning circuits for memristive crossbar neural networks," in *Circuits and Systems (ISCAS), 2018 IEEE International Symposium on*. IEEE, 2018.

[16] X. Sun, X. Peng, P. Y. Chen, R. Liu, J. s. Seo, and S. Yu, "Fully parallel rram synaptic array for implementing binary neural network with (x002b;1, x2212;1) weights and (x002b;1, 0) neurons," in *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan 2018, pp. 574–579.

[17] D. W. Patterson, "Artificial neural networks, theory and applications, prentice hall, singapore, 1996."

[18] N. Inc., "Hierarchical temporal memory including htm cortical learning algorithms," Tech. Rep., 2006.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov 1997.

[20] O. Krestinskaya, A. P. James, and L. O. Chua, "Neuro-memristive circuits for edge computing: A review," *arXiv preprint arXiv:1807.00962*, 2018.

[21] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[22] N. Ahad, J. Qadir, and N. Ahsan, "Neural networks in wireless networks: Techniques, applications and guidelines," *Journal of Network and Computer Applications*, vol. 68, pp. 1 – 27, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804516300492

[23] Y. Xu, J. Du, L. R. Dai, and C. H. Lee, "A regression approach to speech enhancement based on deep neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 1, pp. 7–19, Jan 2015.

[24] P. Zhou, H. Jiang, L. R. Dai, Y. Hu, and Q. F. Liu, "State-clustering based multiple deep neural networks modeling approach for speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 4, pp. 631–642, April 2015.

[25] L. Jiang, R. Hu, X. Wang, W. Tu, and M. Zhang, "Nonlinear prediction with deep recurrent neural networks for non-blind audio bandwidth extension," *China Communications*, vol. 15, no. 1, pp. 72–85, Jan 2018.

[26] N. Funabiki and S. Nishikawa, "A binary hopfield neural-network approach for satellite broadcast scheduling problems," *IEEE Transactions on Neural Networks*, vol. 8, no. 2, pp. 441–445, Mar 1997.

[27] D. L. Gray and A. N. Michel, "A training algorithm for binary feedforward neural networks," *IEEE Transactions on Neural Networks*, vol. 3, no. 2, pp. 176–194, Mar 1992.

[28] C. Curto, A. Degeratu, and V. Itskov, "Encoding binary neural codes in networks of threshold-linear neurons," *Neural Computation*, vol. 25, no. 11, pp. 2858–2903, Nov 2013.

[29] M. M. Gupta, L. Jin, and N. Homma, *Binary Neural Networks*. Wiley-IEEE Press, 2003, pp. 0–. [Online]. Available: https://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5260357

[30] J. Secco, M. Poggio, and F. Corinto, "Supervised neural networks with memristor binary synapses," *International Journal of Circuit Theory and Applications*, vol. 46, no. 1, pp. 221–233, 2018.

[31] Y. Zhou, S. Redkar, and X. Huang, "Deep learning binary neural network on an fpga," in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, Aug 2017, pp. 281–284.

[32] O. Krestinskaya and A. James, "Binary weighted memristive analog deep neural network for near-sensor edge processing," in *The 18th International Conference on Nanotechnology (IEEE NANO)*. IEEE, 2018.

[33] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in neural information processing systems*, 2016, pp. 4107–4115.

[34] J. Hawkins and S. Blakeslee, "On intelligence. 2004," *New York St. Martins Griffin*, pp. 156–8.

[35] O. Krestinskaya, I. Dolzhikova, and A. P. James, "Hierarchical temporal memory using memristor networks: A survey," *arXiv preprint arXiv:1805.02921*, 2018.

[36] Y. Cui, C. Surpur, S. Ahmad, and J. Hawkins, "A comparative study of htm and other neural network models for online sequence learning with streaming data," in *2016 International Joint Conference on Neural Networks (IJCNN)*, July 2016, pp. 1530–1538.

[37] J. Hawkins, S. Ahmad, S. Purdy, and A. Lavin, "Biological and machine intelligence (bami)," 2016, initial online release 0.4. [Online]. Available: http://numenta.com/biological-and-machine-intelligence/

[38] D. George and J. Hawkins, "A hierarchical bayesian model of invariant pattern recognition in the visual cortex," in *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, vol. 3. IEEE, 2005, pp. 1812–1817.

[39] A. P. James, I. Fedorova, T. Ibrayev, and D. Kudithipudi, "Htm spatial pooler with memristor crossbar circuits for sparse biometric recognition," *IEEE Transactions on Biomedical Circuits and Systems*, vol. PP, no. 99, pp. 1–12, 2017.

[40] O. Krestinskaya, T. Ibrayev, and A. P. James, "Hierarchical temporal memory features with memristor logic circuits for pattern recognition," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. PP, no. 99, pp. 1–1, 2017.

[41] M. Sundermeyer, H. Ney, and R. Schlter, "From feedforward to recurrent lstm neural networks for language modeling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 517–529, March 2015.

[42] K. Greff, R. K. Srivastava, J. Koutnk, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, Oct 2017.

[43] K. Smagulova and A. P. James, "A memristor-based long short term memory circuit," *Proceedings of Oxford Circuits and Systems Conference*, 2017.

[44] K. Smagulova, K. Adam, O. Krestinskaya, and A. P. James, "Design of cmos-memristor circuits for lstm architecture," in *IEEE International Conferences on Electron Devices and Solid-State Circuits*, 2018.

[45] Y. Chauvin and D. E. Rumelhart, *Backpropagation: theory, architectures, and applications*. Psychology Press, 1995.

[46] K. Mehrotra, C. K. Mohan, and S. Ranka, *Elements of artificial neural networks*. MIT press, 1997.

[47] A. Tisan and J. Chin, "An end-user platform for fpga-based design and rapid prototyping of feedforward artificial neural networks with on-chip backpropagation learning," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 3, pp. 1124–1133, June 2016.

[48] H. M. Vo, "Implementing the on-chip backpropagation learning algorithm on fpga architecture," in *2017 International Conference on System Science and Engineering (ICSSE)*, July 2017, pp. 538–541.

[49] J. Leonard and M. Kramer, "Improvement of the backpropagation algorithm for training neural networks," *Computers & Chemical Engineering*, vol. 14, no. 3, pp. 337–341, 1990.

[50] R. M. Golden, *Mathematical methods for neural network analysis and design*. MIT Press, 1996.

[51] S. Xiao, X. Xie, S. Wen, Z. Zeng, T. Huang, and J. Jiang, "Gst-memristor-based online learning neural networks," *Neurocomputing*, 2017.

[52] A. Irmanova and A. P. James, "Neuron inspired data encoding memristive multi-level memory cell," *Analog Integrated Circuits and Signal Processing*, vol. 95, no. 3, pp. 429–434, 2018.

[53] ——, "Multi-level memristive memory with resistive networks," in *Postgraduate Research in Microelectronics and Electronics (PrimeAsia), 2017 IEEE Asia Pacific Conference on*. IEEE, 2017, pp. 69–72.

[54] N. Dastanova, S. Duisenbay, O. Krestinskaya, and A. P. James, "Bit-plane extracted moving-object detection using memristive crossbar-cam arrays for edge computing image devices," *IEEE Access*, vol. 6, pp. 18 954–18 966, 2018.

[55] G. Khodabandehloo, M. Mirhassani, and M. Ahmadi, "Analog implementation of a novel resistive-type sigmoidal neuron," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 4, pp. 750–754, April 2012.

[56] M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. J. Yang, and R. S. Williams, "Dot-product engine for neuromorphic computing: programming 1t1m crossbar to accelerate matrix-vector multiplication," in *Proceedings of the 53rd annual design automation conference*. ACM, 2016, p. 19.

[57] S. N. Truong and K.-S. Min, "New memristor-based crossbar array architecture with 50-% area reduction and 48-% power saving for matrix-vector multiplication of analog neuromorphic computing," *Journal of semiconductor technology and science*, vol. 14, no. 3, pp. 356–363, 2014.

[58] M. Hu, H. Li, Q. Wu, and G. S. Rose, "Hardware realization of bsb recall function using memristor crossbar arrays," in *Proceedings of the 49th Annual Design Automation Conference*. ACM, 2012, pp. 498–503.

[59] P. Yao, H. Wu, B. Gao, S. B. Eryilmaz, X. Huang, W. Zhang, Q. Zhang, N. Deng, L. Shi, H.-S. P. Wong *et al.*, "Face classification using electronic synapses," *Nature communications*, vol. 8, p. 15199, 2017.

[60] Z. Wang, W. Zhao, W. Kang, Y. Zhang, J.-O. Klein, and C. Chappert, "Ferroelectric tunnel memristor-based neuromorphic network with 1t1r crossbar architecture," in *Neural Networks (IJCNN), 2014 International Joint Conference on*. IEEE, 2014, pp. 29–34.

[61] D. Mikhailenko, C. Liyanagedera, A. P. James, and K. Roy, "M2ca: Modular memristive crossbar arrays," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2018, pp. 1–5.

[62] T. Ibrayev, U. Myrzakhan, O. Krestinskaya, A. Irmanova, and A. P. James, "On-chip face recognition system design with memristive hierarchical temporal memory," *arXiv preprint arXiv:1709.08184*, 2017.

[63] D. Biolek, Z. Kolka, V. Biolkova, and Z. Biolek, "Memristor models for spice simulation of extremely large memristive networks," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2016, pp. 389–392.

[64] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *nature*, vol. 453, no. 7191, p. 80, 2008.

[65] D. Kuzum, R. G. Jeyasingh, B. Lee, and H.-S. P. Wong, "Nanoelectronic programmable synapses based on phase change materials for brain-inspired computing," *Nano letters*, vol. 12, no. 5, pp. 2179–2186, 2011.

[66] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[67] A. Georghiades, P. Belhumeur, and D. Kriegman, "Yale face database," *Center for computational Vision and Control at Yale University, http://cvc. yale. edu/projects/yalefaces/yalefa*, vol. 2, p. 6, 1997.

[68] I. Messaris, A. Serb, S. Stathopoulos, A. Khiat, S. Nikolaidis, and T. Prodromakis, "A data-driven verilog-a reram model," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.

Olga Krestinskaya is working towards her graduate degree thesis in the area of neuromorphic memristive system from Electrical and Computer Engineering department at Nazarbayev University. She completed her bachelor of Engineering degree with honors in Electrical Engineering, with a focus on bio-inspired memory arrays in May 2016. Currently, she focuses on memristive circuits for hierarchical temporal memory, deep learning neural networks, and pattern recognition algorithms. She is a Graduate Student Member of IEEE.

Khaled N. Salama (SM' 10) received the B.S. degree (Hons.) from the Department of Electronics and Communications, Cairo University, Cairo, Egypt, in 1997, and the M.S. and Ph.D. degrees from the Department of Electrical Engineering, Stanford University, Stanford, CA, USA, in 2000 and 2005, respectively.He was an Assistant Professor with the Rensselaer Polytechnic Institute, Troy, NY, USA, from 2005 to 2009. In 2009, he joined the King Abdullah University of Science and Technology, Saudi Arabia, where he is currently a Professor and was also the founding Program Chair until 2011. His work on CMOS sensors for molecular detection has been funded by the National Institutes of Health and the Defense Advanced Research Projects Agency, received the Stanford-Berkeley Innovators Challenge Award in biological sciences and was acquired by Lumina Inc. He has authored 225 papers and 14 patents on low-power mixed signal circuits for intelligent fully integrated sensors and nonlinear electronics, in particular memristor devices.

Alex Pappachen James (SM'13) works on brain-inspired circuits, memristor circuits, algorithms and systems, and has a PhD from Griffith School of Engineering, Griffith University. He is currently the Vice Dean of research and graduate studies and also the Chair of Electrical and Computer Engineering department at Nazarbayev University. He is also the chair of IEEE Kazakhstan subsection. He has a sustained experience of managing industry and academic projects in board design, VLSI and pattern recognition algorithms, and semiconductor industry. He was editorial board member of Information fusion and is currently Associate Editor of Human-centric Computing and Information Sciences, IEEE Access, IEEE Transactions on Circuits and Systems 1, and served as guest associate editor to IEEE Transactions on Emerging Topics in Computational Intelligence. He is a Senior Member of IEEE and Senior Fellow of HEA. More see http://biomicrosystems.info/alex/