IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 8, 2020, accepted December 11, 2020, date of publication December 14, 2020,
date of current version December 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3044652

IR-QNN Framework: An IR Drop-Aware Offline Training of Quantized Crossbar Arrays

MOHAMMED E. FOUDA[1], (Member, IEEE), SUGIL LEE[2], JONGEUN LEE[2], (Member, IEEE), GUN HWAN KIM[3], FADI KURDAHI[1], (Fellow, IEEE), AND AHMED M. ELTAWI[1,4], (Senior Member, IEEE)

[1]Center for Embedded and Cyber-Physical Systems, University of California at Irvine, Irvine, CA 92697, USA
[2]School of ECE, UNIST, Ulsan 44776, South Korea
[3]Division of Advanced Materials, Korea Research Institute of Chemical Technology (KRICT), Daejeon 34114, South Korea
[4]Computer, Electrical and Mathematical Science and Engineering Division, King Abdullah University of Science and Technology (KAUST), Thuwal 23955, Saudi Arabia

Corresponding author: Mohammed E. Fouda (foudam@uci.edu)

ABSTRACT Resistive Crossbar Arrays present an elegant implementation solution for Deep Neural Networks acceleration. The Matrix-Vector Multiplication, which is the corner-stone of DNNs, is carried out in $O(1)$ compared to $O(N^2)$ steps for digital realizations of $O(log_2(N))$ steps for in-memory associative processors. However, the IR drop problem, caused by the inevitable interconnect wire resistance in RCAs remains a daunting challenge. In this article, we propose a fast and efficient training and validation framework to incorporate the wire resistance in Quantized DNNs, without the need for computationally extensive SPICE simulations during the training process. A fabricated four-bit $Au/Al_2O_3/HfO_2/TiN$ device is modelled and used within the framework with two-mapping schemes to realize the quantized weights. Efficient system-level IR-drop estimation methods are used to accelerate training. SPICE validation results show the effectiveness of the proposed method to capture the IR drop problem achieving the baseline accuracy with a 2% and 4% drop in the worst-case scenario for MNIST dataset on multilayer perceptron network and CIFAR 10 dataset on modified VGG and AlexNet networks, respectively. Other nonidealities, such as stuck-at fault defects, variability, and aging, are studied. Finally, the design considerations of the neuronal and the driver circuits are discussed.

INDEX TERMS RRAM, memristor, deep neural networks, quantized neural networks, IR drop, nonidealities, variability, offset.

I. INTRODUCTION

Artificial Intelligence hardware acceleration has attracted significant interest [1], [2] especially accelerating deep neural networks (DNNs) with in-memory processing, alleviating the memory-wall bottleneck problem in the Von-Neumann computing architecture. In-memory computing paradigm offers a powerful tool for accelerating artificial intelligence and machine learning algorithms where the matrix-vector multiplication (MVM) computation can be performed in $O(1)$ with resistive crossbar structures and in $O(log_2(N))$ with an in-memory associative processor [3], [4], unlike other digital implementations that require $O(N^2)$ steps.

Recent advances in non-volatile memory devices, such as Resistive Random Access Memory (RRAM) (memristor), Spin-transfer torque magnetic random-access memory (STTRAM), and Phase-change memory (PCM) that form the crossbar structure, promise an efficient implementation of MVM computation. The data are stored locally inside the crossbar array eliminating the memory-access need. The ability to efficiently perform MVM computations, especially with RRAM crossbar arrays (RCAs), is very appealing for DNNs since almost all DNN computations can be reduced to MVM operations [5], [6]. Recently, memristor-based crossbar arrays have been used to implement and accelerate many machine learning and deep learning algorithms including pattern recognition [7]–[11], sentimental analysis [12], neural processing [13], reinforcement learning [14] and neuromorphic systems [15].

Recently many RRAM devices have been introduced, experimentally showing the ability to be programmed to multiple distinct states, from 2 states (1-bit) up to 64 states (6-bit)

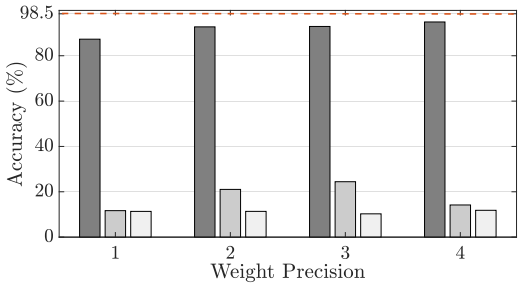
The associate editor coordinating the review of this manuscript and approving it for publication was Shiping Wen.

in fabricated devices [16]. These devices have been applied to in-memory computing involving weight parameters for quantized neural networks. Practically, there are multiple technical challenges to realizing RRAM-based DNN hardware. One important issue is how to overcome RRAM arrays' intrinsic inaccuracy and provide an accurate result at the application level, without the need for online (i.e., *in-situ*) training.RCAs' computation result is susceptible to the device's nonidealities such as device variability, read/write disturbance, stuck-at fault defects, etc. in addition to the inherent noise of analog computation [17], [18]. In particular, the IR drop[1] problem is caused due to the interconnect wire resistance, which affects the MVM computation quite significantly for large arrays and/or advanced technologies [19]–[23]. The interconnect wire resistivity is exponentially increasing with decreasing the technology node due to the increase in electron scattering at grain boundaries, surfaces, and interfaces according to the International Technology Roadmap for Semiconductors (ITRS) roadmap [24]. The intuitive way to mitigate the wire resistance is by improving the metal technology (i.e., less resistivity) or by increasing the interconnect dimensions, which might prevent 3D integration and causes a reduction in density per unit area. For instance, In [25], the authors had to replace nanowire by nano-wall to be able to fabricate a 2nm device to reduce the wire resistance from $10^5\Omega/\mu m$ to $10^2\Omega/\mu m$ which is not desirable for high dense RCA, especially for 3D integration technologies.

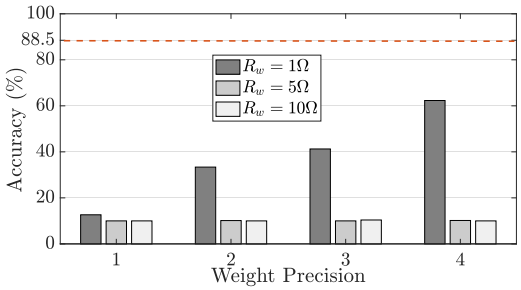
The prior works show that the IR drop problem is a leading cause for degraded performance in MVM computations [26], [27]. For instance, our study based on SPICE-simulation-integrated QNN inference finds that the recognition accuracy for MNIST and CIFAR10 datasets suffers a huge drop as depicted in Fig. 1, which is not acceptable. The higher the wire resistance, the higher accuracy drop.Since SPICE simulation is computationally expensive, it cannot be used for *training* of QNN. Hardware solutions, such as using 1T1R structure to activate one column at a time, increase the time complexity of MVM to O(N) and require extra hardware to store data [28]. Another solution, such as using 1S1R, helps to mitigate the IR drop problem while achieving the same parallelism of passive structure. However, it highly disturbs the MVM computation due to the exponential nonlinearity of the selector [29].

Prior works typically include numerical or SPICE IR-drop simulations or hardware experiments during training, which highly impacts the training procedure. In [27], the authors proposed an additive noise injection technique, referred to as NIA, to compensate for the effect of the IR drop where the crossbar output current shift with or without IR-drop are collected for all of the testing data. The impact of IR-drop is approximated as a Gaussian noise source at each crossbar output end, with crossbar-wise mean and standard deviation

[1]The term *IR drop* originates from the fact that the amount of voltage drop is proportional to the product of the current (commonly denoted by *I*) and the wire resistance (indicated by *R*).



(a) MNIST on MLP Network



(b) CIFAR10 on VGG9 Network

FIGURE 1. Recognition accuracy under IR drop problem for different wire resistance R_W and different weight precision. The dashed line represented the baseline accuracy.

extracted from the collected statistics. Finally, the network is retrained with the approximated additive IR drop noise applied at each output of the crossbar array. This technique requires a substantial analog memory (or ADC and memory) to save the analog output currents of each crossbar array, or at least it would involve an iterative SPICE simulation of the entire network, which is not practical for large networks and is not even preferred for small networks. Another method is introduced in [30], where an iterative post-processing technique is proposed for finding the best weight matrix under the IR drop that is very close to the trained weight, which required at least seven iterations for 0.005 MSE. Each iteration requires one IR drop simulation for each weight matrix, which is not practical for large networks. Moreover, If the RCA hardware is available, a software-hardware co-design approach is used where the inference is wholly carried out on the RCA hardware that contains all the nonidealities. The backward path is performed on the software level with ideal parameters [7]. This approach is optimal at the expense of a prolonged training cycle, making it unpractical for DNNs that have billions of parameters, and the quality of model transferability is not guaranteed. In addition, in [31]. we proposed in two neural networks models to model IR drop problem in RCAs. These models were designed and optimized for binary neural networks and require an IR drop dataset to train for each IR drop scenario.

Our work aims to avoid any kind of numerical/SPICE/Hardware experiments for the IR drop problem during the training and also to avoid any kind of retraining, which would require extra hardware, leading to an increase in the power and area. Our technique directly maps the weights to the

hardware without retraining or extra hardware. We use the SPICE-equivalent simulator for validation only, and the proposed techniques are independent of the weight values or network structure. These techniques can also involve hardware for a more accurate estimation of the IR drop problem. The contributions of this work are summarized in the following points:

- We first model a fabricated four-bit $Au/Al_2O_3/HfO_2/TiN$ device for accurately mapping the weights to the device's conductances where we introduce two possible mappings for quantized neural network realization.
- We then explain the IR drop problem and evaluate the performance degradation quantitatively.
- An IR-QNN training and validation framework tailored for RRAM crossbar array hardware is introduced, considering the IR drop problem.
- We also introduce efficient software-level methods to incorporate the effect of the IR drop problem without the need for running extensive and time-consuming SPICE simulations.
- We experimentally show that the proposed methods capture the IR drop problem showing a 2% and 4% drop in the worst-case scenario for MNIST dataset on multilayer perceptron network and CIFAR 10 dataset on modified VGG and AlexNet networks, respectively.
- We evaluate the effect of other nonidealities, such as stuck-at fault defects, variability, and retention on the performance.
- Finally, we discuss the design requirements of the neuronal and the driver circuits to guide the designers to have robust and efficient circuits.

This article is organized as follows: Section I discusses the hardware accelerator's hardware realization, where we introduce the multi-bit RRAM device model and the weight mapping. The IR-drop problem is explained in detail in Section III. Section IV discusses the proposed framework for the training and inference and the proposed software methods to estimate the IR drop problem without involving SPICE simulations. Section V discusses the training results and studies the effects of other nonidealities such as device variability, stuck-at fault defects, and aging. Finally, peripheral circuit requirements are discussed in Section VI.

II. IN-MEMORY MVM IMPLEMENTATION

In this section, we discuss the In-Memory MVM using RRAM-crossbar arrays (RCAs), where the weight matrix is stored in the RRAMs. We first model the multi-bit device that is used to realize the weights. Then, we discuss the quantized weight mapping onto the multi-bit device.

A. MULTI-BIT RRAM DEVICE UNDER STUDY

The authors in [16] demonstrated the fabrication of $Au/Al_2O_3/HfO_2/TiN$ RRAM device with a junction area of $10 \times 10\mu m^2$ patterned via photolithography and followed by a wet-etching process. The thicknesses of the top electrode (Au) and the bottom electrode (TiN) were 150 and 100 nm, respectively, and the switching material thicknesses are 2 and 6 nm for Al_2O_3 and HfO_2, respectively. This device was optimized to have self-compliance and gradual set-switching behavior and is capable of generating up to 16 states with good reliability.

To precisely program the RRAM device, an incremental step pulse programming technique with error correction is used. Starting from a low conductance state, incremental step pulses are applied to the device using a pulse generator until the device reaches the required state. Figures 2a and 2b show the gradual incremental-step programming and the current-voltage hysteresis of the programmed device under different programming conditions. Figures 3a and 3b show the histogram and cumulative distribution function of the measured conductances, respectively. It is worth mentioning that the conductance's variation is due to cycle to cycle read variability. The measured samples are curve-modeled into a Gaussian distribution, and we have found that the mean value of the device's conductance can be modeled as $G_i = 14+6\times i \quad (\mu S)$ where G_i is the i^{th} highest conductance state for $i \in [1, 15]$ while the low conductance state is $46.7nS$.

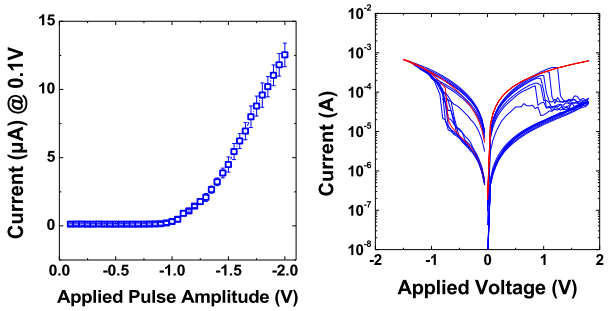


FIGURE 2. $Au/Al_2O_2/HfO_2/TiN$-based RRAM device adopted from [16] (a) device behavior under incremental step pulse programming and (b) current-voltage characteristics.

In order to incorporate the variability in the hardware simulations, there are two ways to integrate the device model in hardware simulation: (i) random sampling of the Gaussian model of each state, and (ii) random sampling from the measured data. In this work, we choose the latter, random sampling from the measured data, since the Gaussian distribution may not accurately describe the randomness of the device's states and device to device variations.

B. MVM USING RCAs

RRAM crossbar arrays can perform the MVM operation, which is equivalent to n^2 multiply and accumulate (MAC) operations, with $O(1)$ time complexity compared to $O(n^2)$. The weight matrix is programmed/stored in the RRAM array cells as conductance values, and the input is applied as a voltage at the rows of the array. By grounding the columns of the array, the output current per column is proportional to the inner product between the input voltage vector and the

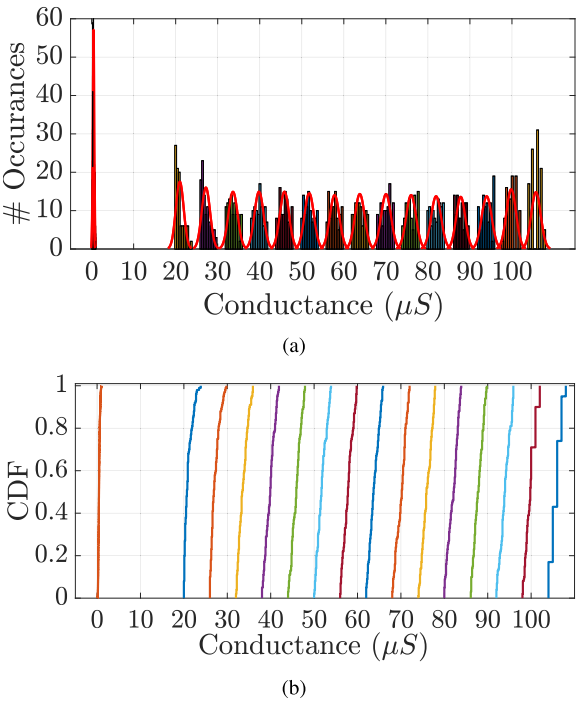


FIGURE 3. Histogram and cumulative distribution function (CDF) of 100 measured samples per state.

conductance vector of the column, which can be written as

$$I_j = \sum_{i=1}^{n+1} G_{ij} V_i \tag{1}$$

where I_j is the current of the j^{th} column (i.e. *post-synaptic current*), G_{ij} the synaptic weight in conductance, V_i the i^{th} input voltage (i.e. pre-synaptic voltage) and $V_{n+1} = b$ which is the bias.

The conductance of RRAM can only realize a positive value; however, both negative and positive weight realizations are needed in any neural network. In order to create negative weights, two weight realization techniques have been introduced: (i) using two RRAM cells per weight [32] as shown in Fig. 4, which is referred to as *balanced realization*, and (ii) using one RRAM as weight, in addition to one shared reference RRAM with the conductance of $G_r = (G_{\max} + G_{\min})/2 \approx G_{\max}/2$, which is referred to as *unbalanced realization* [26], [33] where $G_{\max}$ and $G_{\min}$ are the minimum and maximum achievable conductances, respectively. In this work, we consider the first realization, which has double the dynamic range (conductance range $\approx (-G_{\max}, G_{\max})$), making it less susceptible to noise and variability at the expense of doubling the area and power. We would like to highlight that the realization, shown in Fig. 4, including analog partial sum and sum & compare circuits, is one way for realizing the MVM partitioning in the analog domain. Other works, such as ISAAC [6], use ADCs/ DACs to preforms the partial sum and sum & compare in the digital domain. We would like to emphasize that our framework is agnostic of the peripheral operations' realization since it focuses on IR drop problem. Besides, the IR drop has less impact on the overlap between the states, as will be discussed in Section III. The IR drop also causes each device conductance to be scaled differently, which could cause more dependency on the stored data. The differential output current can be written as

$$I_j = \sum_{i=1}^{n+1} (G_{ij}^{+} - G_{ij}^{-}) V_i = \sum_{i=1}^{n+1} G_{ij} V_i \tag{2}$$

where G_{ij} is the differential conductance and can be written as matrix-vector multiplication as follows

$$\mathbf{I} = (\mathbf{G}^{+} - \mathbf{G}^{-})\mathbf{V} = \mathbf{G}\mathbf{V} \tag{3}$$

where $\mathbf{V}$ is the input voltage vector (e.g., input image) and the bias value. The current vector, $\mathbf{I}$, is sensed and shaped by the activation function, which is mathematically described as

$$\mathbf{O} = f(\mathbf{G}\mathbf{V}) \tag{4}$$

where $f(\cdot)$ is the activation function.

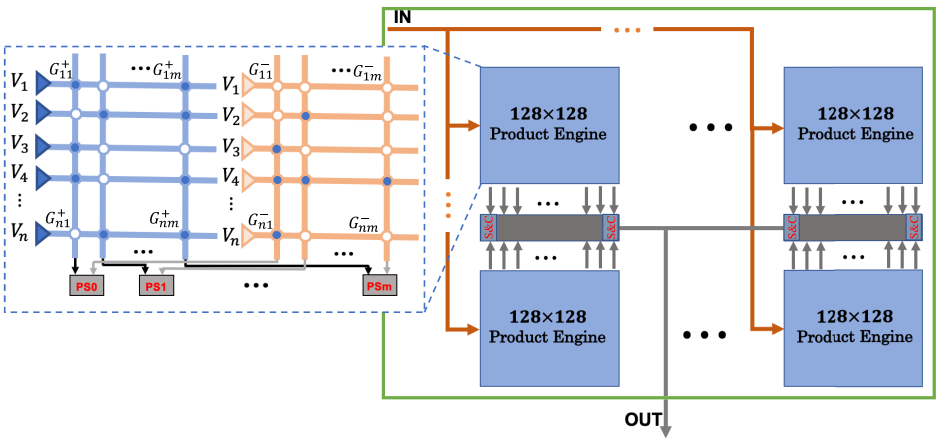


FIGURE 4. Matrix Vector Multiplication partitioned into different product engines and using separate RCAs per engine. The Sensed current of each column is converted to voltage using partial sum (PS) circuit. The PS voltages are linearly summed and compared using sum and compare circuit (S&C) to generate the binary output.

In practice, a DNN layer can be too large to be realized in hardware using a single crossbar array. Thus, these large layers are partitioned into smaller crossbar arrays connected to perform MVM as a single layer [6], [26]. In this work, we partition each layer into differential 128 × 128 arrays, which is the same size as recently fabricated arrays [34]. Larger array sizes lead to worse IR drops, causing higher degradation in performance, as discussed in [26].

C. WEIGHT MAPPING

Each weight is translated into a pair of conductance values, which can be mathematically formulated as

$$G = G^+ - G^- = \frac{W}{W_{\max}} \Delta G, \tag{5}$$

where $W_{\max}$ is the maximum value of the weight. If it is required to realize $W_{\max}$, G^+ and G^- are set to $G_{\max}$ and $G_{\min}$, respectively, and $\Delta G = G_{\max} - G_{\min}$. The difference between the two conductance values is constant and proportional to the required weight value, and each conductance is constrained to be between $G_{\min}$ and $G_{\max}$.

Using the aforementioned 4-bit device, it is possible to realize up to 5-bit weight when two devices per weight are used. Table 1 and Fig. 5 show the weight mapping from quantized weight to device's conductance. Mapping-I is designed to occupy the entire dynamic range of the devices, which results in less overlapping for smaller precision. On the other hand, Mapping-II is designed to use the closest possible states to the zero state (i.e., high conductance state) with equal spacing. This results in less power consumption with higher overlap, as shown later in the results section. There is a linear relation between the device conductance and the weight if chosen properly except for the 5-bit case because of the high gap between the low conductance state and the first high conductance state. Thus, in this work, we consider up to 4-bit quantized neural network.

TABLE 1. Weight-conductance mapping for quantized states.

#bits	Weight (W)	Mapping-I	Mapping-II	Range
1 bit	0 & ± 1	0 & $\pm G_{15}$	0 & $\pm G_1$	-
2 bit	0 & $\pm \frac{i}{2}$	0 & $\pm G_{8i-1}$	0 & $\pm G_{2i-1}$	$1 \le i \le 2$
3 bit	0 & $\pm \frac{i}{4}$	0 & $\pm G_{4i-1}$	0 & $\pm G_{4i-1}$	$1 \le i \le 4$
4 bit	0 & $\pm \frac{i}{8}$	0 & $\pm G_{2i-1}$	0 & $\pm G_{2i-1}$	$1 \le i \le 8$
5 bit	0 & $\pm \frac{i}{16}$	0 & $\pm G_{i-1}$	0 & $\pm G_{i-1}$	$1 \le i \le 16$

III. INEVITABLE WIRE RESISTANCE PROBLEM IN CROSSBAR STRUCTURES

The wire resistance is inevitable in nanostructure crossbar arrays. It is expected that the wire resistance would reach around 90Ω for 5 *nm* feature size [35]. The wire resistance creates undesired IR voltage drops that accumulate across the columns in the array leading to unwanted paths between the input and output nodes. Thus, the columns are not grounded anymore, resulting in a highly distorted MVM result.

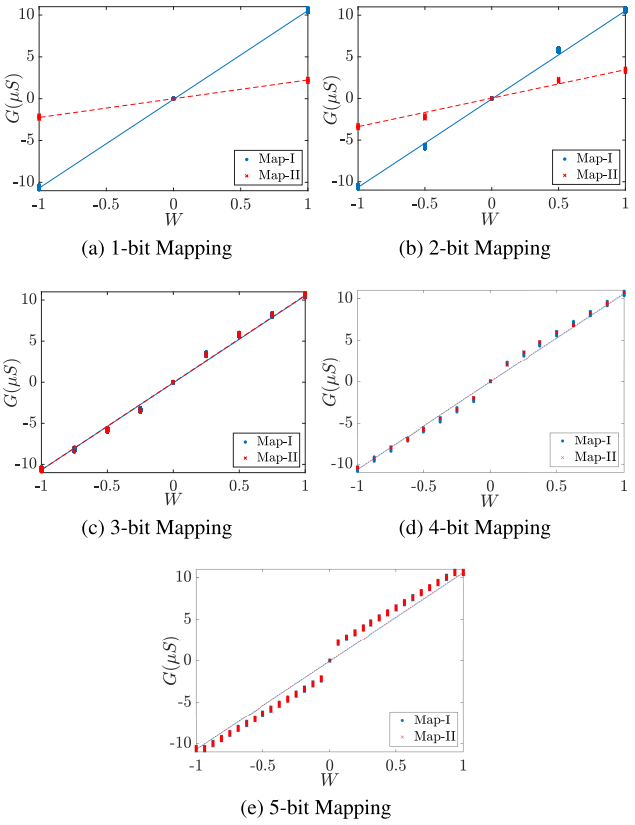


FIGURE 5. Possible weight mappings for the used RRAM device.

These voltage drops are a function of the stored data and the wire resistance.

Due to the lack of the experimental data for 128 × 128 crossbar arrays, we incorporated the device model discussed in the previous section into a SPICE-like simulator [35]. This simulator accurately simulates the interconnect and devices parasitics three orders of magnitude faster than SPICE with no loss in accuracy. Then, this simulator is incorporated in our framework for generating the training masks and for validating the performance of the retrained networks with the proposed method, which will be discussed in Section IV. It is also worth mentioning that any device model can be incorporated in our framework and the device impact on the neural network performance can be evaluated accordingly.

Using the analysis in [35], it can be shown that the output current with wire resistance effect can be modeled as (see Appendix A for the proof)

$$\mathbf{I} = g(\mathbf{G}, \mathbf{u}) = \mathbf{G}_e \mathbf{u} \tag{6}$$

where $g(\cdot)$ is the IR drop nonideality function, $\mathbf{G}$ is the *programmed conductance matrix* . $\mathbf{u}$ is the input vector and $\mathbf{G}_e$ the *effective weight matrix* which has the same size as the RCA, which is $n \times m$. In the balanced realization case, the output current is $\mathbf{I} = (\mathbf{G}_e^+ - \mathbf{G}_e^-)\mathbf{u}$ where the IR drop behaviour is the same for both RCAs. Thus, the difference can be seen as constant. On the other hand, in case of the unbalanced realization, the output current is $\mathbf{I} = g([\mathbf{G}^+, \mathbf{G}^-])\mathbf{u})$ which is

a more complex relation. Thus, the IR drop has severe impact on the output current in the unbalanced realization case.

Fig. 6a shows the normalized measured effective weights for differential crossbar array with 1Ω wire resistance for two crossbar arrays, 256 × 256 array and 128 × 128 array, populated with random data. The measured weights decrease exponentially across the diagonal. Increasing the crossbar array size increases the IR voltage drop across the array, creating more sneak paths. Fig. 6b also shows the histogram of random data with 1-bit ternary quantization (i.e $-1, 0, 1$). The histogram of the 256×256 array has a smaller mean value and a larger standard variation compared to the 128 × 128 array. In addition, Fig. 6c shows the effect of the wire resistance on the histogram of the measured data for 1Ω, 5Ω, and 10Ω wire resistance for the 128 × 128 array. The higher the wire resistance, the more severe the IR drop. Figures 6 (d) and (e) show histograms of the quantized states for 3-bit and 4-bit

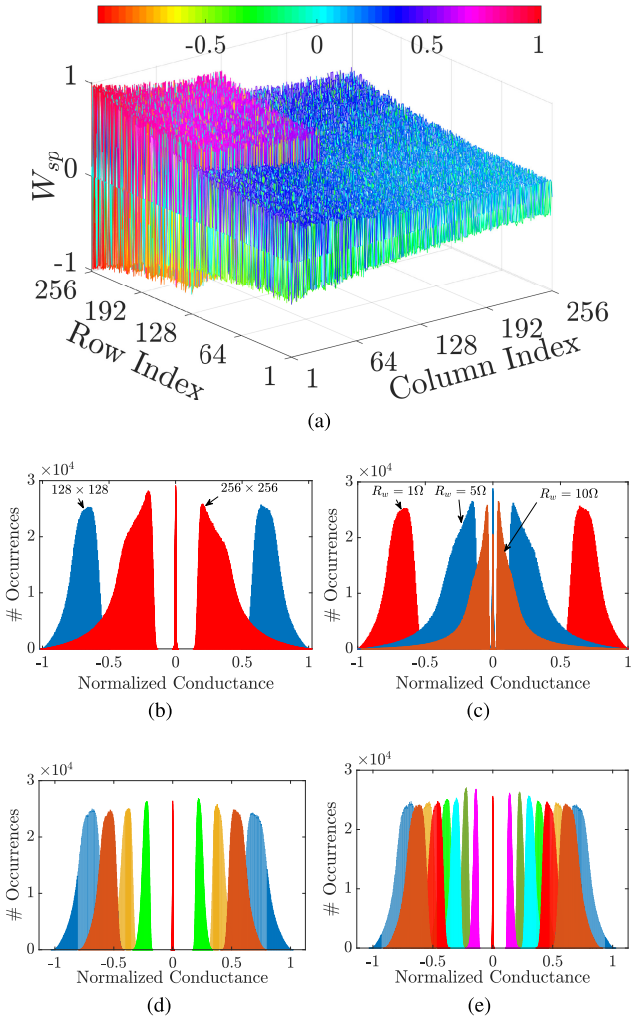


FIGURE 6. **(a) Measured weight per cell for 128 × 128 and 256 × 256 crossbar arrays at $R_W = 1\Omega$, (b) Histogram of the measured conductance normalized to G_{max} of the crossbar arrays shown in (a), (c) (5) Histogram of the measured conductance for 1-bit case for different wire resistance, (d) and (e) Histograms of the normalized conductance for 3-bit and 4-bit cases at $R_W = 1\Omega$.**

weights for the 128 × 128 case. Ideally, each state should be a narrow pulse and non-overlapped, but because of the IR drop problem, it becomes wider and overlapped.

Partitioning each layer into the small arrays is necessary for three main reasons;

- IR drop problem: partitioning helps to reduce the effect of the problem, compromising the main benefit of RRAMs, which is the density.
- Driver nonideality: each crossbar is driven by a driver circuit or buffer. The loading of the driver circuit is the input resistance of the driven row creating a voltage divider with the output resistance of the driver circuit. For example, the worst-case occurs when all the devices within the same row have a low resistance state (LRS), and the output resistance of the driver is R_o. Thus, the voltage delivered to the crossbar input is $V_d = V_{in}\frac{LRS}{LRS+NR_o}$. So, it is necessary to reduce the number of devices per row, N, to mitigate the driver's effect. Otherwise, it has to be taken into consideration during the training. We do not consider it in this work for two reasons 1) with a well-designed peripheral circuit, its effect can be eliminated or mitigated. And 2) the IR drop problem is the main cause of the performance degradation [26]. However, we study its effect on the performance to find the required output resistance value of the driver circuit for the designers in section V.
- Fabrication problem: It is less complex to fabricate small crossbar arrays with high reliability.

In addition, it is recommended to use two separate crossbar arrays for positive and negative conductances to have symmetric IR drop behavior. The corresponding conductances are scaled by the same value, unlike using a single crossbar array for both positive and negative conductances where each conductance will be scaled with different values.

To have accurate inference results, it is needed to run the inference with SPICE simulation with all circuits included. A SPICE simulator is adopted where the weight matrices are partitioned into small crossbar arrays as discussed in [26], [36] and simulated using a transient simulation for different input samples. Fig. 7 shows the simulation time of matrix-vector multiplication using SPICE for a 256 × 256 array partitioned into smaller arrays and for different input samples. The SPICE simulation time, without the peripheral circuits such as neuronal sensing circuit and drivers, increases exponentially with increasing the crossbar array size and linearly with increasing the input samples. On the other hand, the same Fig.shows the numerical SPICE-equivalent simulator adopted from [35]. The numeric simulator runs 140× faster than SPICE for one input sample and 1000× faster than SPICE for ten successive input samples. It is worth highlighting that the numerical results of the MVM are the same as the SPICE results.

Although the numerical model runs orders of magnitude faster than SPICE simulations, it is better not to be included in the DNN framework. It would take considerable training time even for small networks such as the MNIST dataset. Thus, it is

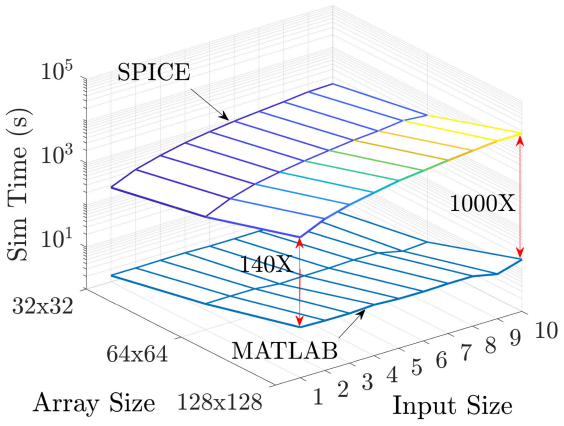


FIGURE 7. Simulation time comparison between SPICE and numerical simulator, adopted from [35], for performing MVM of 256 × 256 array partitioned into 32 × 32, 64 × 64 and 128 × 128 and for different number of input samples.

better to have solutions that can be used to describe fabricated hardware. In this work, we use the numerical or SPICE-like simulator, without loss of generality, as a reference due to the lack of the hardware.

IV. PROPOSED IR-QNN TRAINING AND INFERENCE

A. QNN TRAINING FRAMEWORK

Due to differential weight realization, it is possible to realize 2^n+1 states where $n \in \{1, 2, 3, 4\}$ is the device's precision in bits. We use binarized activation function $\{-1, 1\}$ for simple and fast communication between the crossbar layers and eliminate area- and power-expensive blocks such as ADCs and DACs.

IR-QNN framework is an extended version of BinaryNet [37] to support more weight states and a bipolar activation function. During training, real-valued weights are quantized through stochastic rounding to the equally distributed point sets within $[-1, 1]$. Activations are binarized into $\{-1, 1\}$ and physically implemented with ± 0.1 V. Multilayer perceptron (MLP) and convolutional neural network(CNN) models are used for MNIST and CIFAR10 dataset, respectively (see Table 4 and 5). Convolution filters are 4D tensors but reshaped to 2D matrices with the number of output channels as the leading dimension so that convolution can be performed by matrix multiplication.

The modifications to the BinaryNet framework, to include higher quantized states and the IR drop estimation technique in the forward and backward computation, are shown in Algorithm 1. In the forward computation section, the modifications are as follows: (1) partitioning the quantized weights of each layer, W_k^b, into small weight matrices P_k^b, (2) application of the IR drop estimation method (*IRestimate* function) to each partitioned array to create W_e to be used in (3) the forward inference and (4) backward computation instead of the quantized weights W^q. Similar modifications can be added to other QNN frameworks to capture the effect of the sneak path problem.

Algorithm 1 Proposed IR-QNN Training Algorithm

Require: a minibatch of inputs and targets (a_0, a^*), previous weights W, device precision n_b, previous BatchNorm parameters θ, weights initialization coefficients from γ, and previous learning rate η

Ensure: updated weights W^{t+1}, updated BatchNorm parameters θ^{t+1} and updated learning rate η^{t+1}.

{1. Computing the parameters gradients:}
{1.1.Forward propagation:}
for $k = 1$ to L **do**
  $W_k^q \leftarrow \text{Quantize}(W_k, nb)$
  $P_k^q \leftarrow \text{Partition}(W_k^q)$
  $W_{e_k} \leftarrow \text{IRestimate}\ (P_k^q)$
  $s_k \leftarrow a_{k-1}^b W_{e_k}$
  $a_k \leftarrow \text{BatchNorm}\ (s_k, \theta_k)$
  if $k < L$ **then**
    $a_k^b \leftarrow \text{Binarize}\ (a_k)$
  end if
end for
{1.2. Backward propagation:}
{Compute $g_{aL} = \frac{\partial C}{\partial a_L}$.}
for $k = L$ to 1 **do**
  if $k < L$ **then**
    $g_{a_k} \leftarrow g_{a_k}^b \circ 1_{|ak|\leq 1}$
  end if
  $(g_{s_k}, g_{\theta_k}) \leftarrow \text{BackBatchNorm}\ (g_{a_k}, s_k, \theta_k)$
  $g_{a_{k-1}^b} \leftarrow g_{s_k} W_{e_k}$
  $g_{W_k^q} \leftarrow g_{s_k}^T a_{k-1}^b$
end for
{2. Accumulating the parameters gradients:}
for $k = 1$ to L **do**
  $\theta_k^{t+1} \leftarrow \text{Update}(\theta_k, \eta, g_{\theta_k})$
  $W_k^{t+1} \leftarrow \text{Clip}(\text{Update}(W_k, \gamma_k \eta, g_{W_k^q}), -1, 1)$
  $\eta^{t+1} \leftarrow \lambda \eta$
end for

B. SYSTEM LEVEL ESTIMATION OF IR DROP PROBLEM

In this section, we introduce different methods to estimate the IR drop problem without the need for SPICE or numerical simulations.

1) TRAINING WITH MULTIPLICATIVE NOISE

It is clear in Fig. 6 that each state is spread with a certain statistical distribution due to the IR drops. One way to overcome this problem and to enable quick estimation of realistic weights, is to create a statistical model for each state and include it into the DNN framework which can be done as follows:

$$W_e = \sum_{i=1}^{Q} W_i^q \odot n_i \tag{7}$$

where W_e is the wire-resistance-effect-compensated weight matrix, W_i^q is the quantized weight matrix having i^{th} state (such that $\sum_i W_i^q = W^q$ equals the quantized weight matrix),

Q the number of states, $\odot$ element-wise multiplication and n_i is multiplicative noise of i^{th} state.

Each state is statistically modeled to different statistical distributions. The log-normal distribution is found to the best distribution (e.g., the highest likelihood) to describe the variability per state. The positive and negative states have similar histograms. Table 2 shows the curve-fitted model parameters for different wire resistance simulating different IR drop scenarios.

TABLE 2. Fitted Lognormal distributions of multiplicative noise for each state.

		$R_w = 1\Omega$		$R_w = 5\Omega$		$R_w = 10\Omega$	
	State	μ	σ	μ	σ	μ	σ
1 bit	1	-0.362	0.121	-1.337	0.460	-2.118	0.750
2 bit	0.5	-0.941	0.112	-1.859	0.427	-2.605	0.707
	1	-0.343	0.113	-1.273	0.437	-2.031	0.717
3 bit	0.25	-1.466	0.109	-2.341	0.409	-3.066	0.680
	0.5	-0.926	0.107	-1.811	0.412	-2.540	0.682
	0.75	-0.580	0.107	-1.468	0.412	-2.204	0.687
	1	-0.328	0.108	-1.228	0.419	-1.964	0.693
4 bit	0.125	-1.924	0.109	-2.783	0.397	-3.500	0.665
	0.25	-1.453	0.105	-2.317	0.397	-3.034	0.663
	0.375	-1.144	0.104	-2.010	0.400	-2.732	0.668
	0.5	-0.914	0.103	-1.784	0.401	-2.506	0.668
	0.625	-0.723	0.104	-1.597	0.404	-2.321	0.671
	0.75	-0.567	0.104	-1.445	0.405	-2.173	0.673
	0.875	-0.432	0.104	-1.312	0.406	-2.041	0.674
	1	-0.316	0.104	-1.198	0.408	-1.931	0.678

Although the method would have the same effect on the summed current per column, this method is not very effective since it treats all the locations in the array equally, which does not describe the real behavior of the IR drop problem, as shown in Fig. 6a.

2) TRAINING WITH MASKS

Another solution is to generate average mask that can account for the cell location in the array. This mask is element-wise multiplied by the quantized weight matrix similar to [26], as follows:

$$W_e = W^q \odot M \tag{8}$$

where M is the average mask matrix.

This mask method helps to predict more realistic behavior of a crossbar array and can be easily calculated with fabricated crossbar array. The mask matrix is generated from either SPICE simulations or equivalent numerical methods [35] and is normalized to the ideal desired current. Masks are generated by averaging the results of many (e.g., 1000) SPICE simulations using random input weight matrices [26], [38].

Due to the averaging, the generated mask is static and fixed. However, in practice, W_e has some stochastic behavior around this average mask. Thus, an additive white Gaussian noise can be added to the mask to exhibits more practical behavior, which we refer to as a stochastic mask.

The third solution is to generate a mask for each state and element-wise multiplied by its corresponding state matrix.

TABLE 3. Mean square error between estimated effective weight matrix and accurate weight matrix.

		1-bit	2-bit	3-bit	4-bit
Mapping-I	Single-Mask	1.76e-4	2.55e-2	2.3e-2	2.2e-2
	Multiple-Mask		1.25e-4	1.1e-4	9.5e-5
Mapping-II	Single-Mask	3.2e-5	1.04e-2	2.3e-2	2.2e-2
	Multiple-Mask		4.12e-5	1.1e-4	9.5e-5

This solution can be mathematically formulated as follows:

$$W_e = \sum_{i=1}^{Q} W_i^q \odot M_i \tag{9}$$

where M_i is the corresponding mask matrix of i^{th} state.

Fig. 8 shows mask examples for training a 2-bit neural network having 5 states per weight. Fig. 8a shows the $M_{\pm 0.5}$ and $M_{\pm 1}$ masks for 1Ω wire resistance as an example. Furthermore,, Fig. 8b shows the effect of the wire resistance on the $M_{\pm 1}$ mask. It is clear the high degradation with higher wire resistance values. It is worth mentioning that applying masks during training has a negligible effect on the training time.

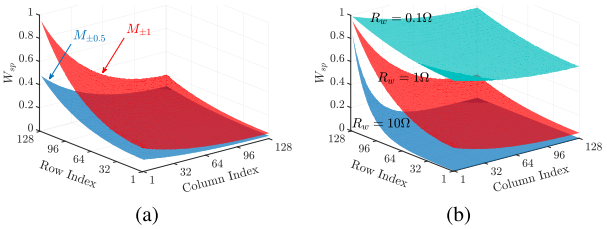


FIGURE 8. The generated masks for 2-bit neural network training.

In order to test the efficacy of estimating the IR drop effect, we calculated the mean square error (MSE) between the estimated effective weight matrix and the measured effective weight matrix for the single mask and multiple mask sets for 128 × 128 RCA with 10Ω. Clearly, the multiple mask set shows more than 100× better MSE compared to the single mask set, which is also is verified with DNN experiments later.

C. BATCH NORMALIZATION DURING INFERENCE

One of the important practices in training deep neural networks is adding a batch normalization layers to speed up the training, improve the performance and overcome vanishing gradient problem in DNNs, without the need for small learning rates which slow down the convergence [39]. In other words, the batch normalization is data whitening (i.e., removing the mean and variance of the data), which can be mathematically defined as follows for j^{th} neuron

$$\mathbf{BN}(\mathbf{y}_j) = \frac{\mathbf{y}_j - \mu_j}{\sigma_j}\gamma_j + \beta_j \tag{10}$$

where μ_j and σ_j are the mean and the standard deviation values of the input vector $\mathbf{y}_j$, and γ_j and β_j are trainable parameters.

During the inference, these batch normalization layers can be removed by merging them with the preceding layers. As aforementioned, we use the sign activation function, thus the output activation o can be computed as

$$o_j = sign\left(\mathbf{BN}\left(\sum_i w_{ij}V_i + b_j\right)\right)$$
$$= sign\left(\frac{\gamma_j}{\sigma_j}\left(\sum_i w_{ij}V_i + b_j - \mu_j + \frac{\sigma_j}{\gamma_j}\beta_j\right)\right) \quad (11)$$

Using $sign(AB) = sign(sign(A)B)$ property yields

$$o_j = sign\left(sign\left(\frac{\gamma_j}{\sigma_j}\right)\left(\sum_i w_{ij}V_i + b_j - \mu_j + \frac{\sigma_j}{\gamma_j}\beta_j\right)\right) \quad (12)$$

Since σ always has positive value, batch normalized layer merged with the proceeding layer can be written as

$$o_j = sign\left(\sum_i \tilde{w}_{ij}V_i + \tilde{b}_j\right) \quad (13)$$

where $\tilde{w}_{ij} = sign(\gamma_j)w_{ij}$ and $\tilde{b}_j = sign(\gamma_j)(b_j - \mu_j + \frac{\sigma_j}{\gamma_j}\beta_j)$. The matrix form of (13) can be written as

$$\mathbf{O} = sign\left(\tilde{\mathbf{W}}\mathbf{V} + \tilde{\mathbf{b}}\right) \quad (14)$$

where $\tilde{\mathbf{W}} = \mathbf{W}\mathbf{D}_{sign(\gamma)}$ and $\tilde{\mathbf{b}} = \left(\boldsymbol{b} - \boldsymbol{\mu}\,\boldsymbol{\mathcal{C}}\,\boldsymbol{D}_{\boldsymbol{\sigma}}\boldsymbol{D}_{\frac{1}{\boldsymbol{\gamma}}}\boldsymbol{\beta}\right)\mathbf{D}_{sign(\gamma)}$, where $\mathbf{D}_v$ is a diagonal matrix whose diagonal is v. It is worth highlighting that the weight parameters are kept quantized even after merging batch normalization.

D. EXPERIMENTAL SETUP

To evaluate the effectiveness of our proposed technique we use the MNIST and CIFAR10 datasets [40], [41]. For each dataset, we use the same network architecture as given in BinaryNet [37], with these two changes: (1) instead of binary weights, we use up to 4-bit (or 17-state) quantized weights, (2) the model sizes are reduced. For MNIST the number of hidden neurons is reduced to one-fourth, and for CIFAR10, the number of channels is reduced to half, roughly reducing the number of model parameters to about 1/16 and 1/4, respectively. Furthermore, we also present results for modified AlexNet [42] for CIFAR10 dataset, with 3x channel sizes to justify our work on larger networks. Details of the networks can be found in Table 4, 5, and 6.

The experiments are performed in three main steps. The first step is the baseline training, which uses floating-point weights/activations to obtain the best test accuracy, where test accuracy is the ratio of the correctly recognized samples for unseen data. We use the default training parameters for 100 epochs of MNIST training and 500 epochs of CIFAR10. Learning rates halve every 20 or 50 epochs for MNIST and CIFAR10, respectively, initially from 2^{-6}. The baseline accuracies are 98.4% for MNIST dataset on multilayer perceptron network, 88.5% for CIFAR10 dataset on modified VGG9 network and 81% for CIFAR10 dataset on modified AlexNet network, which is similar to the best accuracies for the datasets reported in the literature. The accuracy reported is test accuracy, that is, the inference accuracy for unseen data.

The second step is fine-tuning, which is running additional training iterations using the weight from the first step as the initial weight. While the weight in the first step gives a very high accuracy on GPU, it is unlikely to give good results if used for RRAM crossbar arrays due to distorted MVM computation caused by IR drop. The fine-tuning retrains the networks to mitigate the discrepancy by using the mask methods for different wire resistances and quantization levels. During fine-tuning, we use learning rates starting from 2^{-9}, training additional 50/200 epochs for MNIST/CIFAR10 models. The other parameters remain the same as the baseline training. At the beginning of fine-tuning, the accuracy plummets due to the introduction of the mask but eventually recovers through fine-tuning. Note that the accuracy at the end of fine-tuning is not indicative of the real performance of RRAM crossbar arrays since it is not trained with SPICE simulations, for which we need a separate validation step.

TABLE 4. MLP network configuration for MNIST dataset.

#	type	#output	#input	#RCAs ×2
1	fully connected	512	784	28
2		512	512	16
3		512	512	16
4		10	512	4

TABLE 5. Modified VGGNet configuration.

#	Type	# Output Channels	# Input Channels	Filter Size	#RCAs ×2 Serial	#RCAs ×2 Parallel
1	convolution	64	3	3x3	1	512
2		64	64	3x3	5	2560
–	max pooling	–	–	2x2	-	-
3	convolution	128	64	3x3	5	1280
4		128	128	3x3	9	2304
–	max pooling	–	–	2x2	-	-
5	convolution	256	128	3x3	18	1152
6		256	256	3x3	36	2304
–	max pooling	–	–	2x2	-	-
7	fully connected	1024	4096	–	256	256
8		1024	1024	–	64	64
9		10	1024	–	8	8

TABLE 6. Modified AlexNet configuration.

#	Type	# Output Channels	# Input Channels	Filter Size	#RCAs ×2 Serial	#RCAs ×2 Parallel
1	convolution	192	3	3x3	2	384
–	max pooling	–	–	2x2	-	-
2	convolution	576	192	3x3	70	4060
–	max pooling	–	–	2x2	-	-
3	convolution	1152	576	3x3	369	5904
4		768	1152	3x3	486	7776
5		256	768	3x3	108	1728
–	max pooling	–	–	2x2	-	-
6	fully connected	1024	4096	–	256	256
7		1024	1024	–	64	64
8		10	1024	–	8	8

TABLE 7. Validation accuracy without retraining of the MNIST and CIFAR10 datasets.

	1-bit				2-bit				3-bit		4-bit	
	MLP		Modified VGG		MLP		Modified VGG		MNIST	Modified VGG	MNIST	Modified VGG
Map-	I	II	I	II	I	II	I	II	I & II	I & II	I & II	I & II
1Ω	87.31	11.35	12.65	10.00	92.74	24.87	33.38	10.00	92.92	41.26	94.90	62.32
5Ω	11.68	11.35	10.00	10.00	21.05	11.96	10.14	10.00	24.44	10.00	14.18	10.18
10Ω	11.35	11.35	10.00	10.00	11.36	11.45	10.00	10.00	10.27	10.39	11.83	10.00

TABLE 8. MNIST dataset validation results using Mapping-I after retraining with M. Noise: Multiplicative Noise, Single mask set, Sto. Mask: Stochastic Mask and Multiple Mask set.

	1-bit				2-bit				3-bit				4-bit			
	M. Noise	Single Mask	Sto. Mask	Multiple Mask	M. Noise	Single Mask	Sto. Mask	Multiple Mask	M. Noise	Single Mask	Sto. Mask	Multiple Mask	M. Noise	Single Mask	Sto. Mask	Multiple Mask
1Ω	98.16	98.34	98.24	98.44	96.75	98.03	98.02	98.32	89.92	97.39	97.4	98.36	85.94	96.98	97.16	98.28
5Ω	95.55	97.36	97.54	97.70	95.07	97.59	97.72	97.48	92.91	97.43	97.48	97.3	89.15	97.09	96.93	97.31
10Ω	91.03	96.96	97.03	97.04	93.3	97.37	97.37	96.93	91.53	97.23	97.13	96.83	83.95	96.84	96.82	96.13

TABLE 9. MNIST dataset validation results using Mapping-II after retraining with stochastic mask and multiple mask sets.

	1-bit		2-bit		3-bit		4-bit	
	Stochastic Mask	Multiple Mask	Stochastic Mask	Multiple Mask	Stochastic Mask	Multiple Mask	Stochastic Mask	Multiple Mask
1Ω	98.35	98.33	97.98	98.48	97.4	98.36	97.03	98.41
5Ω	98.27	98.27	98.08	98.11	97.48	97.3	97.5	97.36
10Ω	98.06	98.01	97.9	97.92	97.13	96.83	97.22	96.69

The third step is validation. The output of the second step is quantized weights after merging the batch normalization layers to be programmed to RRAM crossbar arrays. The trained weights are mapped to the RCAs, as discussed in section II. The unused portions of the RCAs are populated with random data so that the distribution of IR drops do not change [26]. The required number of RCAs to implement each network is shown in Table 4, 5, and 6. For CNN, there are two ways to implement conventional layers either by flattening the convolutional layers into one large matrix-vector multiplication, which results in the lowest latency at the expense of the area and power or by iterating over the same kernel multiple times, which results in high latency but saves area. In our framework, we use generalized matrix multiplication (GEMM) with im2col mapping for accelerating the convolutions layers where the filters and input patches are laid out into a 2-D matrices, which are multiplied to compute the same dot product of the convolutional operation [43].

Our *validation setup* takes the quantized weights, and runs SPICE-based RRAM crossbar simulation, to get the effective output currents with the device model that has been presented in Section II-A. The effective output currents are fed back to our QNN inference framework to obtain network-level inference results. Note that neither training nor mask is used during validation. The test accuracy obtained from validation is what we can expect to see if the quantized weights are perfectly programmed to RRAM crossbar arrays, barring stochastic and other unmodeled noise/faults during RRAM read. Some of those nonidealities are considered in our additional experiments (see Section V).

V. RESULTS AND DISCUSSION

In this work, we consider three test scenarios, with 1Ω, 5Ω, and 10Ω wire resistances to consider different technology nodes. For instance, for a 50 nm feature size, it is expected to have around 5Ω wire resistance [24], [35]. The results shown in Table 7 illustrates that without considering the IR drop problem in training, the accuracy drops to around $10 \sim 12\%$ from the baseline test accuracies regardless of the number of bits.

After the retraining using the proposed techniques, the networks were able to reach close to the baseline accuracies. Tables 8-11 show the validation accuracy for MNIST and CIFAR10 datasets for different wire resistance scenarios and different mappings. The higher the wire resistance, the higher the drop in performance. Clearly, mapping-II shows a better performance for 1-bit and 2-bit cases since the used resistance values are much higher than the one used for mapping-I, and the severity of the IR drop problem is determined by the ratio between the device's LRS and the wire resistance value. The smaller the ratio, the more severe the IR drop problem. In general, training with multiple mask set achieves the best performance among the proposed solutions. On the other hand, increasing the number of bits does not improve the performance monotonically. The accuracy drops with increasing the number of bits to more than 2 bits, which is attributed to the overlap between states, as illustrated in Fig. 6. The drop in CNN test accuracy is much higher than MLP test accuracy due to its high sensitivity to weight variations. Clearly, from these results, the proposed training method provides the best performance with 2-bit devices.

TABLE 10. CIFAR10 dataset on modified VGGNet validation results after retraining.

	1-bit				2-bit				3-bit		4-bit	
	Map-I		Map-II		Map-I		Map-II		Map-I & II		Map-I & II	
R_w	Stochastic Mask	Multiple Mask	Stochastic Mask	Multiple Mask	Stochastic Mask	Multiple Mask	Stochastic Mask	Multiple Mask	Stochastic Mask	Multiple Mask	Stochastic Mask	Multiple Mask
1Ω	86.53	87.05	86.79	87.07	76.99	87.27	76.95	87.24	63.39	87.31	59.98	87.04
5Ω	83.95	83.62	86.32	86.71	75.84	85.2	78.93	86.85	65.14	85.96	58.79	85.21
10Ω	81.75	82.35	86.39	86.27	72.32	83.54	76.14	85.95	61.32	84.39	55.7	83.82

TABLE 11. CIFAR10 dataset on the modified AlexNet validation results with mapping-I after retraining.

	1-bit		2-bit		3-bit		4-bit	
R_w	Stochastic Mask	Multiple Mask	Stochastic Mask	Multiple Mask	Stochastic Mask	Multiple Mask	Stochastic Mask	Multiple Mask
1Ω	79.03	78.93	68.98	80.24	55.12	79.88	52	80.5
5Ω	76.62	76.84	67.62	78.65	59.07	79.2	50.83	80
10Ω	75.06	74.98	63.59	77.27	54.78	76.9	35.2	78.61

In Fig. 9, we compare our method against noise-injection Adaption (NIA) method [27] which was proposed for QNNs. The comparison is performed on VGGNet on CIFAR10 dataset with mapping-I. NIA method performance highly drops with increasing the wire resistance and with increasing the weight precision as well. Clearly, our method outperforms NIA method in all scenarios thanks to capturing the spatial behavior of the IR drop problem.

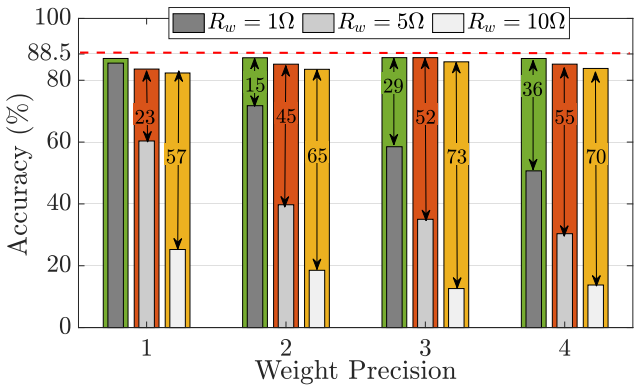
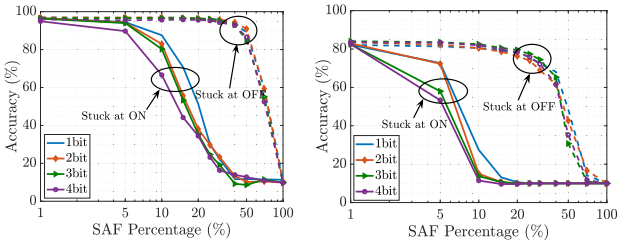


FIGURE 9. Comparison of the proposed mask technique against NIA method [27] on modified VGGNet on CIFAR10 dataset. The grey-colored inner bars represent NIA method and the colored outer bars represent the proposed method.

A. STUCK-AT FAULT EFFECT

Stuck-At Fault (SAF) defects cause another inevitable problem that affects the accuracy results of the MVM, which is the main operation in DNNs. SAF defects vary based on the fabrication technology and RRAM switching materials. In some recent works, the percentage of SAF fabricated crossbar arrays is about 10% for 1024×1024 for an in-house test array [44], [45] and is about 0.2% for 128×64 array (with only 15 devices stuck off) [46]. With the knowledge of the exact locations of the SAF devices, the network can be trained to isolate the SAF devices or at least mitigate their effect. However, this is not practical for DNNs where the trained weights are not designed for specific hardware. It should run without any knowledge of the location of SAF defects. Thus, in this work, we explore the effect of different SAF percentages on the recognition accuracy without retraining the network, assuming that the SAF devices are randomly distributed in each crossbar array.

Fig. 10 shows the recognition accuracy with changing the SAF percentage for a 10Ω wire resistance scenario. The performance has no significant drop up to 50% and 20% stuck-at open (OFF) for MNIST and CIFAR10 datasets, respectively. On the other hand, performance is more sensitive to stuck-at close (ON) case, where the stuck at close can cause weight-sign flip, a scenario that significantly affects performance and does not happen for the stuck-at open. For instance, for 1-bit, the possible weight values are $\{-1, 0, 1\}$ which are mapped to $G^+ = \{HRS, HRS, LRS\}$ and $G^- = \{LRS, HRS, HRS\}$, respectively. The stuck at OFF would occur to one of the 2 LRSs, which would result in mapping all original weights to zero weight value in the worst case. On the other hand, the stuck at ON can occur to one of the four HRSs, which causes that mapping $\{1, -1\}$ to "0" and "0" to be mapped to either "-1" or "1". Thus, the performance is more sensitive to the stuck-at ON. However, The effect of stuck at On can be mitigated by implementing "0" by two LRSs, which would hurt the stuck-at OFF performance. Thus,



(a) MLP Network (b) Modified VGG Network

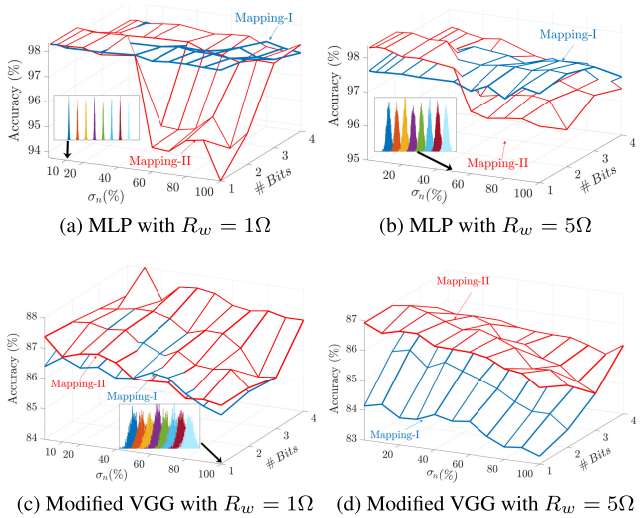
FIGURE 10. Effect of changing the SAF percentage on the recognition accuracy using mapping-I. Solid and dashed lines show stuck-at ON (LRS) results and stuck-at OFF (HRS) results, respectively.

In our implementation, we choose two HRSs to implement "0" since stuck-at OFF is the most common in these devices [44]–[46]. The results in Fig. 10 shows that 1-bit and 2-bit networks are more robust against Stuck at ON compared to 3-bits and 4-bits cases. In addition, the stuck-at OFF results show some accuracy drop regardless of weight precision. The higher weight precision has a slight accuracy improvement after knee point. In conclusion, there is no need to retrain the network with the full knowledge of the SAF devices' locations, especially that the reported SAF percentages are less than 20% [44], [45].

B. EFFECT OF DEVICE VARIABILITY

In order to achieve low variation in each conductance's state, i.e., precise programming, write error-correcting techniques such as write-verify technique are usually used. However, such techniques require multiple write and read cycles, increasing the programming time of the entire crossbar array. In addition, a write-disturb problem occurs where writing some cells might disturb the written data in other cells [47]. With multiple writes to the same cell, a higher rate of the disturbed cell can occur. Thus, it is better to write once and take the variability into consideration during the training and validation. In order to consider these device variabilities, we have added Gaussian noise to each conductance's state where we vary the normalized standard deviation, σ_n, normalized to the difference between the states ($\Delta g = 6\mu S$) from zero to 100%.

Fig. 11 shows the effect of changing the normalized standard deviation on the MLP and modified VGG networks using MNIST and CIFAR10 datasets, respectively. In this experiment, we have used the trained model with the multiple mask set and sampled from the measured data shown in Fig. 3a without performing any retraining to include new noise. Clearly, the performance slightly drops with increasing σ_n. In case of the MLP network, the performance drops around 1.5% in the worst case for mapping-I. Approximately, the performance is the same regardless of the number of bits for mapping-I. On the other hand, mapping-II is more sensitive to the variations. The performance drops around 4% for 1-bit and 2-bit cases with 1Ω wire resistance after $\sigma_n = 0.5$. This drop is caused by the narrow spacing between the states in mapping-II. Training with higher wire resistance values reduces the drop in the performance to 1.5%. In the case of a modified VGG network, the accuracy is slightly dropped by 2.1% at worst case compared to the zero noise case. Thus, network sensitivity to the programming variability is small since the IR drop problem dominates the programming noise model.



(a) MLP with $R_w = 1\Omega$ (b) MLP with $R_w = 5\Omega$

(c) Modified VGG with $R_w = 1\Omega$ (d) Modified VGG with $R_w = 5\Omega$

FIGURE 11. Effect of changing the variability of each conductance's state on the recognition accuracy for different R_w and for an MLP network (a, b) and a modified VGG network (c, d). The subplots show the added noise to the network for each state at different σ_n percentages.

C. EFFECT OF LIMITED RETENTION

The main factor that would affect the performance of the DNNs over time (i.e., aging) is the RRAMs' retention. Recent works show different retention values based on device structure and materials. These works also show that the device drifts over the time towards a very low conductance state, G_f, which is less than the formed low conductance state, G_{min} [48]–[50]. In addition, the drift speed is a function of temperature. The higher the temperature, the higher the drift that occurs [48]. Due to the lack of aging modeled, we adopt the following retention model to be incorporated in the validation simulations to study the performance degradation with time. We emphasize that aging was taken into consideration in the training process to simulate practical scenarios. The conductance change versus time can be modeled as follows

$$G(t) = G_i - (G_i - G_f)\left(\frac{e^{v_d t_n} - 1}{v_d - 1}\right) \tag{15}$$

where G_i is the initial conductance state, v_d is the drift coefficient, and t_n normalized retention time which is normalized to the retention value of the device. We chose this normalized model to simulate different RRAMs' behaviors with different drift coefficients.

Fig. 12 shows the effect of aging on the validation accuracy of the MNIST dataset for two scenarios $v_d = 10$ and 0.1 with 25% variability in the normalized retention time to simulate different device conditions. Clearly, the network was able to achieve the baseline accuracy for more than the 50% of the retention time. Then, we start seeing performance degradation regardless of the number of bits. The accuracy degrades faster for a smaller drift coefficient.

D. POWER AND AREA RESULTS

The power dissipation during the inference consists of the power dissipation of RCAs and the power dissipation in the peripheral circuits. However, due to the resistive nature of RCAs, the power is mainly consumed inside the RCAs. Fig. 13 shows the power dissipation of RCAs for processing one input image at 0.1 V. Clearly, mapping-II consumes

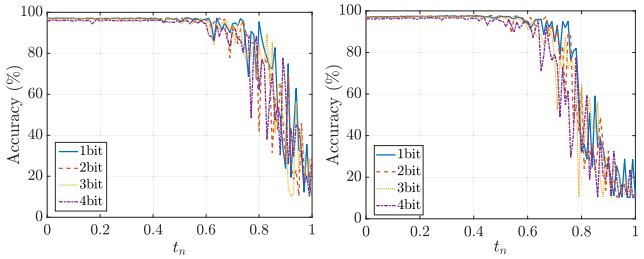


FIGURE 12. MNIST Recognition Accuracy against normalized retention time for (a) $v_d = 10$, and (b) $v_d = 0.1$.

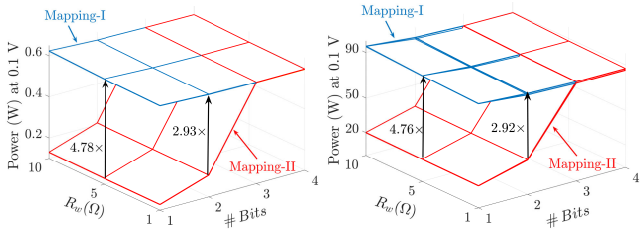


FIGURE 13. Static power dissipation in RCAs per image at 0.1V read voltage.

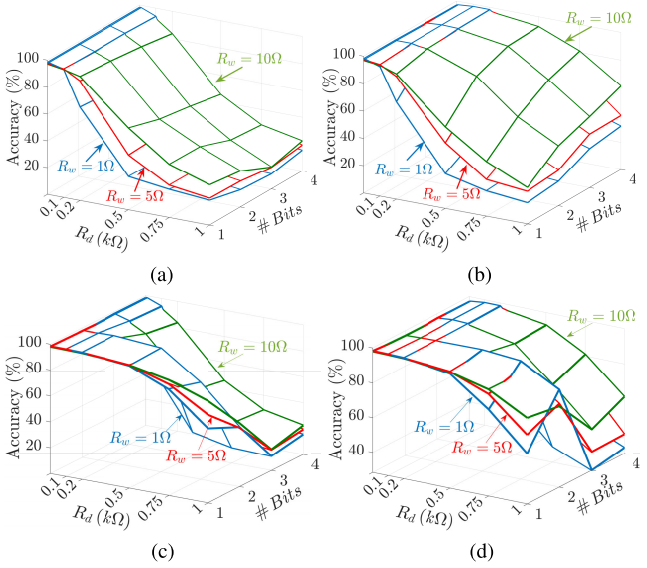


FIGURE 14. Effect of the driver resistance on the performance of MNIST recognition; (a) and (b) for Mapping-I and (c) and (d) for mapping-II with multiple mask training for (a) and (c) and with stochastic mask training for (b) and (d).

around 20% and 35% of the power consumed in mapping-I for 1-bit and 2-bit cases, respectively. It is worth to highlight that the power consumption of using the model trained with the average mask is the same as the one trained with multiple mask set. To estimate the performance metrics, we used the hardware setup shown in Fig. 4, discussed in [36] in which direct communication between the RCAs without network on chip for routing purposes (i.e., fully dedicated hardware) which would give the highest performance. Adding a network on chip offers a highly flexible design; however, it costs power, area, and latency. Thus, with hardware setup, the total power consumption per image is estimated to be around 0.9 W, 132 W, and 225 W for the MLP, Modified VGG, and Modified AlexNet networks, respectively, where RCAs consumes 65.8%, 69.65%, and 91.5% of the total power while the rest is consumed in the sensing circuits and memory cells needed to store intermediate stages while pipelining. On the other hand, the total area is estimated to be around 0.0185 mm^2, 2.81 mm^2, 2.6mm^2 for the MLP, Modified VGG, and modified AlexNet networks, respectively, distributed as $\{17.7\%, 45.5\%, 36.8\%\}$, $\{29.7\%, 38.9\%, 31.4\%\}$ and $\{64\%, 19.9\%, 161\%\}$ for RCAs, peripheral circuits and storage cells, respectively, using 25nm technology node, representing the recent fabricated stable device with more than 4bit precision [51]. The energy per image is estimated to be 27.8 nJ, 3.72 mJ and 0.8 mJ for MLP, Modified VGG, and Modified AlexNet networks, respectively, at 100MHz operating frequency. According to aforementioned power and area results, this hardware is able to achieve 204 $TOp/s/W$, 239$TOp/s/W$,143$TOp/s/W$ with 1.23 KW/mm^2, 1.175 KW/mm^2 and 2.16 KW/mm^2 power density

VI. DRIVER AND NEURONAL CIRCUITS REQUIREMENTS

As previously discussed in section IV, we trained the networks to have binary activation function, $\{-1, 1\}$, for efficient communication and buffering between the fully connected layers. Three nonidealities need to be considered while designing the periphery circuits:

- Driver output resistance: Each crossbar array is driven by a driver or buffer circuit. The output resistance of the driver circuit creates a voltage divider with the parallel RRAMs within the same driven row.
- Neuronal input resistance: After the current is summed within the crossbar array, a current sensing circuit is needed to sense the summed current from the positive array and compare it with the summed current from the negative array, and give a positive or negative output voltage. The input resistance of the sensing circuit creates extra loading to the crossbar array.
- Neuronal circuit variability: Due to PVT variations of the circuit, the comparison between current sensed from positive and negative RCAs is biased to one of them with a random value.

These nonidealities disturb the MVM computation, which affects the DNN performance. With well-designed circuits, there is no need to consider them during training. Ideally, the driver circuit should have zero output resistance, and the neuronal circuit should have zero input resistance and zero current offset.

In this section, we study the effect of these nonidealities on the MNIST network performance to find the maximum values that the network can tolerate without affecting the performance. It is worth to highlight that there is no retraining with peripheral circuits nonidealities. Including them in training

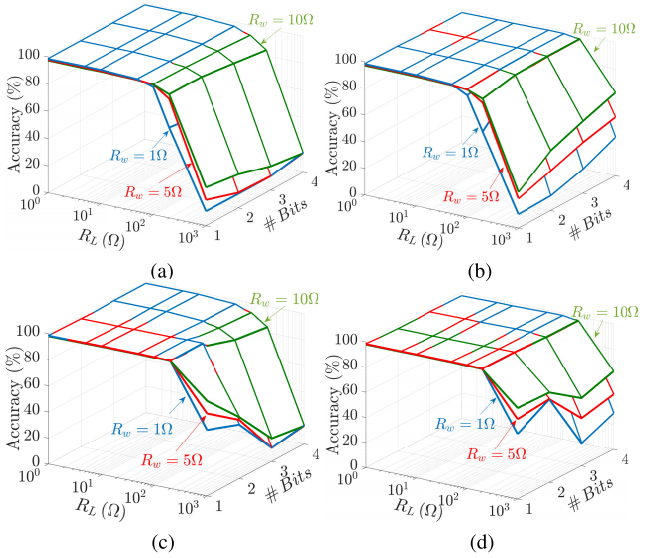


FIGURE 15. Effect of the load resistance (the input resistance of the sensing circuit) on the performance of MNIST recognition; (a) and (b) for Mapping-I and (c) and (d) for mapping-II with multiple mask training for (a) and (c) and with stochastic mask training for (b) and (d).

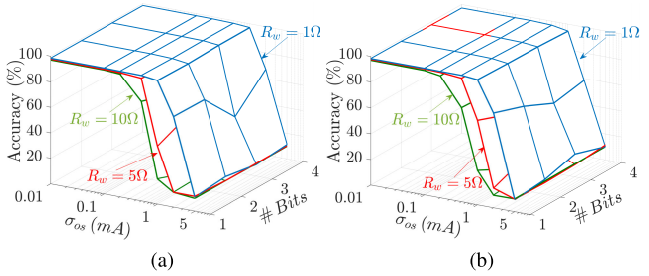


FIGURE 16. Effect of the neuronal offset current deviation on the performance of MNIST recognition for Mapping-I; (a)multiple mask training and (b) stochastic mask training.

will relax the design requirements. Although, we see it is more beneficial to consider the worst case.

Fig. 14 shows performance degradation due to the output resistance of the driver circuit with changing the number of bits and wire resistance. The results show that the trained network with multiple mask set can tolerate higher driver resistance, especially with a higher number of bits. In addition, mapping-II exhibits better behavior compared to mapping-I. Similarly, the effect of the load resistance is studied and shown in Fig. 15 for different wire resistance, number of bits, and different training methods. The performance degradation due to increasing the load resistance is the same for any number of bits. Training with a higher wire resistance value exhibits better performance against load and driver resistances. The effect of changing the standard deviation of the offset current on recognition accuracy is depicted in Fig. 16. Per these results, the network can tolerate up to 0.1 mA, 0.5 mA, and 1 mA standard deviation of the offset current for 1Ω, 5Ω, and 10Ω wire resistance, respectively, regardless of the number of bits per device.

VII. CONCLUSION AND FUTURE WORK

The paper proposed a QNN framework with a software-level technique to incorporate the IR drop problem in training the deep quantized neural networks. The efficiency of the proposed method is proven with three neural networks and is compared with prior work showing a significant improvement. We also studied the impact of other nonidealities, such as device variability, stuck-at faults, and aging. Our results show that the 2-bit device exhibits the best performance and training with multiple mask set. Our experimental results recommend using a driver with output resistance to be less than 200Ω, and input resistance of the sensing circuit should be less than 100Ω. In addition, the input-referred current offset deviation should be less than 0.1mA.

The main limitation of the proposed method is that the masks are generated assuming random data stored in RCAs which generate a static mask which might show lower performance than expected for nonrandom data patterns. Adding random noise to the average mask improves the performance in some cases, mainly for MLP networks and for Convnets with 1-bit weight precision case only, as shown in our experiments. In addition, the proposed method shows less performance with increasing the wire resistance for instance less the accuracy dropped around 5%p at 10Ω wire resistance.
In this work, a software evaluation of the framework's performance is performed, taking into consideration the hardware limits. A full circuit validation with a full implementation of the peripheral circuits, such as partial sum, sample & compare, and max-pooling circuits, is also needed to validate the performance. Besides, other datasets should be tested in our framework. These two points are left for future work.

APPENDIX A
STEADY-STATE MODEL OF MVM USING CROSSBAR ARRAY

Figure 17 show the crossbar array with wire and capacitive parasitics. According to [35], the nodal voltages can be obtained by solving the following 1^{st} order system of differential equations:

$$\mathbf{MV} + \mathbf{N}\dot{\mathbf{V}} = \mathbf{Gu} \tag{16}$$

where $\mathbf{V}$ and $\mathbf{u}$ are the nodal voltage and the excitation vectors, respectively and $\mathbf{M}$, $\mathbf{N}$ and $\mathbf{G}$ are the coefficient matrices containing the RRAM's conductances, wire and capacitive parasitics values. The construction of these matrices can be found in detail in [35]. Since our concern is the steady-state behaviour, the capacitive parasitics can be ignored. Thus, the steady-state nodal voltage vector can be written as

$$\mathbf{V}_{ss} = \mathbf{M}^{-1}\mathbf{Gu} \tag{17}$$

The output current is needed to have accurate MVM as discussed. Thus, the steady-state output current equation can be defined as

$$\mathbf{I}_{ss} = \Psi\mathbf{V}_{ss} \tag{18}$$

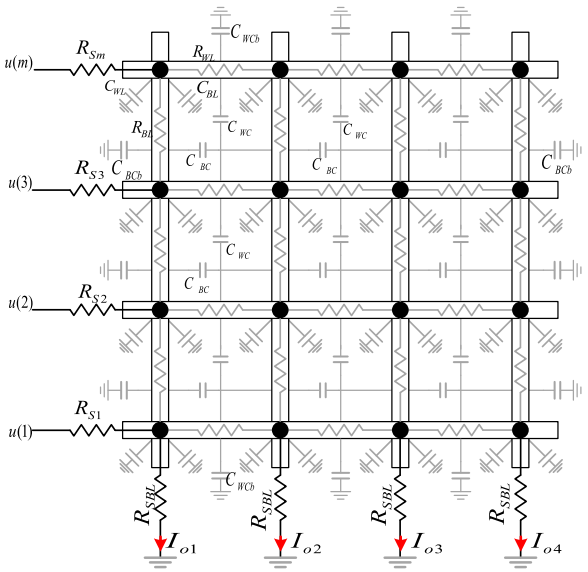


FIGURE 17. Circuit model of the crossbar array.

where $\mathbf{I_o}$ is the output current vector and Ψ is the selection matrix which is given as

$$\Psi = \frac{1}{R_{sBL}} \left[\mathbf{0}_{n\times mn} \; \mathbf{I}_{n\times n} \; \mathbf{0}_{n\times(m-1)n} \right]. \tag{19}$$

where R_{sBL} is the parasitic load resistance of the crossbar array and m and n are the array dimensions. Consequently, the output current can be written as

$$\mathbf{I}_{ss} = \Psi \mathbf{M}^{-1} \mathbf{G} \mathbf{u} \tag{20}$$

This equation can be written as $\mathbf{I}_{ss} = \mathbf{G_e u}$ where $G_e = \Psi \mathbf{M}^{-1} \mathbf{G}$ Similar analysis can be adapted for nonlinear switching devices.

REFERENCES

[1] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.
[2] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.
[3] H. E. Yantir, A. M. Eltawil, and F. J. Kurdahi, "A hybrid approximate computing approach for associative in-memory processors," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 4, pp. 758–769, Dec. 2018.
[4] H. E. Yantir, A. M. Eltawil, and F. J. Kurdahi, "A two-dimensional associative processor," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 9, pp. 1659–1670, Sep. 2018.
[5] Q. Xia and J. J. Yang, "Memristive crossbar arrays for brain-inspired computing," *Nature Mater.*, vol. 18, no. 4, pp. 309–323, Apr. 2019.
[6] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "ISAAC: A convolutional neural network accelerator with *in-situ* analog arithmetic in crossbars," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 14–26, 2016.
[7] C. Li, D. Belkin, Y. Li, P. Yan, M. Hu, N. Ge, H. Jiang, E. Montgomery, P. Lin, Z. Wang, W. Song, J. P. Strachan, M. Barnell, Q. Wu, R. S. Williams, J. J. Yang, and Q. Xia, "Efficient and self-adaptive *in-situ* learning in multilayer memristor neural networks," *Nature Commun.*, vol. 9, no. 1, p. 2385, Dec. 2018.
[8] P. Yao, H. Wu, B. Gao, J. Tang, Q. Zhang, W. Zhang, J. J. Yang, and H. Qian, "Fully hardware-implemented memristor convolutional neural network," *Nature*, vol. 577, no. 7792, pp. 641–646, Jan. 2020.
[9] F. Cai, J. M. Correll, S. H. Lee, Y. Lim, V. Bothra, Z. Zhang, M. P. Flynn, and W. D. Lu, "A fully integrated reprogrammable memristor–CMOS system for efficient multiply–accumulate operations," *Nature Electron.*, vol. 2, no. 7, pp. 290–299, Jul. 2019.
[10] S. Wen, H. Wei, Z. Yan, Z. Guo, Y. Yang, T. Huang, and Y. Chen, "Memristor-based design of sparse compact convolutional neural network," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 3, pp. 1431–1440, Jul. 2020.
[11] S. Wen, J. Chen, Y. Wu, Z. Yan, Y. Cao, Y. Yang, and T. Huang, "CKFO: Convolution kernel first operated algorithm with applications in memristor-based convolutional neural network," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, early access, Sep. 4, 2020, doi: 10.1109/TCAD.2020.3019993.
[12] S. Wen, H. Wei, Y. Yang, Z. Guo, Z. Zeng, T. Huang, and Y. Chen, "Memristive LSTM network for sentiment analysis," *IEEE Trans. Syst., Man, Cybern. Syst.*, early access, Apr. 16, 2019, doi: 10.1109/TSMC.2019.2906098.
[13] Z. Liu, J. Tang, B. Gao, P. Yao, X. Li, D. Liu, Y. Zhou, H. Qian, B. Hong, and H. Wu, "Neural signal analysis with memristor arrays towards high-efficiency brain–machine interfaces," *Nature Commun.*, vol. 11, no. 1, pp. 1–9, Dec. 2020.
[14] Z. Wang, C. Li, W. Song, M. Rao, D. Belkin, Y. Li, P. Yan, H. Jiang, P. Lin, M. Hu, J. P. Strachan, N. Ge, M. Barnell, Q. Wu, A. G. Barto, Q. Qiu, R. S. Williams, Q. Xia, and J. J. Yang, "Reinforcement learning with analogue memristor arrays," *Nature Electron.*, vol. 2, no. 3, pp. 115–124, Mar. 2019.
[15] M. Payvand, M. E. Fouda, F. Kurdahi, A. M. Eltawil, and E. O. Neftci, "On-chip error-triggered learning of multi-layer memristive spiking neural networks," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 10, no. 4, pp. 522–535, Dec. 2020.
[16] G. H. Kim, H. Ju, M. K. Yang, D. K. Lee, J. W. Choi, J. H. Jang, S. G. Lee, I. S. Cha, B. K. Park, J. H. Han, T.-M. Chung, K. M. Kim, C. S. Hwang, and Y. K. Lee, "Four-bits-per-cell operation in an HfO_2-based resistive switching device," *Small*, vol. 13, no. 40, Oct. 2017, Art. no. 1701781.
[17] S. Ambrogio, S. Balatti, A. Cubeta, A. Calderoni, N. Ramaswamy, and D. Ielmini, "Understanding switching variability and random telegraph noise in resistive RAM," in *Proc. IEEE Int. Electron Devices Meeting*, Dec. 2013, pp. 5–31.
[18] Q. Zhang, H. Wu, P. Yao, W. Zhang, B. Gao, N. Deng, and H. Qian, "Sign backpropagation: An on-chip learning algorithm for analog RRAM neuromorphic computing systems," *Neural Netw.*, vol. 108, pp. 217–223, Dec. 2018.
[19] Y. Jeong, M. A. Zidan, and W. D. Lu, "Parasitic effect analysis in memristor-array-based neuromorphic systems," *IEEE Trans. Nanotechnol.*, vol. 17, no. 1, pp. 184–193, Jan. 2018.
[20] I. Chakraborty, D. Roy, and K. Roy, "Technology aware training in memristive neuromorphic systems for nonideal synaptic crossbars," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 5, pp. 335–344, Oct. 2018.
[21] M. E. Fouda, F. Kurdahi, A. Eltawil, and E. Neftci, "Spiking neural networks for inference and learning: A memristor-based design perspective," 2019, *arXiv:1909.01771*. [Online]. Available: http://arxiv.org/abs/1909.01771
[22] S. Yu, "Neuro-inspired computing with emerging nonvolatile memorys," *Proc. IEEE*, vol. 106, no. 2, pp. 260–285, Feb. 2018.
[23] S. Jain, A. Ankit, I. Chakraborty, T. Gokmen, M. Rasch, W. Haensch, K. Roy, and A. Raghunathan, "Neural network accelerator design with resistive crossbars: Opportunities and challenges," *IBM J. Res. Develop.*, vol. 63, no. 6, pp. 10:1–10:13, Nov. 2019.
[24] L. Wilson, "International technology roadmap for semiconductors (ITRS)," Semicond. Ind. Assoc., San Jose, CA, USA, Tech. Rep., 2013.
[25] S. Pi, C. Li, H. Jiang, W. Xia, H. Xin, J. J. Yang, and Q. Xia, "Memristor crossbar arrays with 6-nm half-pitch and 2-nm critical dimension," *Nature Nanotechnol.*, vol. 14, no. 1, pp. 35–39, Jan. 2019.
[26] M. E. Fouda, S. Lee, J. Lee, A. Eltawil, and F. Kurdahi, "Mask technique for fast and efficient training of binary resistive crossbar arrays," *IEEE Trans. Nanotechnol.*, vol. 18, pp. 704–716, 2019.
[27] Z. He, J. Lin, R. Ewetz, J.-S. Yuan, and D. Fan, "Noise injection adaption: End-to-End ReRAM crossbar non-ideal effect adaption for neural network mapping," in *Proc. 56th Annu. Design Autom. Conf.*, Jun. 2019, p. 57.
[28] Y. Zhang, M. Cui, L. Shen, and Z. Zeng, "Memristive quantized neural networks: A novel approach to accelerate deep learning on-chip," *IEEE Trans. Cybern.*, early access, May 3, 2019, doi: 10.1109/TCYB.2019.2912205.
[29] H. Kim, T. Kim, J. Kim, and J.-J. Kim, "Deep neural network optimized to resistive memory with nonlinear current-voltage characteristics," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 14, no. 2, p. 15, 2018.

[30] B. Liu, H. Li, Y. Chen, X. Li, T. Huang, Q. Wu, and M. Barnell, "Reduction and IR-drop compensations techniques for reliable neuromorphic computing systems," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2014, pp. 63–70.

[31] S. Lee, G. Jung, M. E. Fouda, J. Lee, A. Eltawil, and F. Kurdahi, "Learning to predict IR drop with effective training for ReRAM-based neural network hardware," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, Jul. 2020, pp. 1–6.

[32] M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, G. C. Adam, K. K. Likharev, and D. B. Strukov, "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, vol. 521, no. 7550, p. 61, 2015.

[33] C.-C. Chang, P.-C. Chen, T. Chou, I.-T. Wang, B. Hudec, C.-C. Chang, C.-M. Tsai, T.-S. Chang, and T.-H. Hou, "Mitigating asymmetric nonlinear weight update effects in hardware neural network based on analog resistive synapse," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 1, pp. 116–124, Mar. 2018.

[34] M. Hu, C. E. Graves, C. Li, Y. Li, N. Ge, E. Montgomery, N. Davila, H. Jiang, R. S. Williams, J. J. Yang, Q. Xia, and J. P. Strachan, "Memristor-based analog computation and neural network classification with a dot product engine," *Adv. Mater.*, vol. 30, no. 9, Mar. 2018, Art. no. 1705914.

[35] M. E. Fouda, A. M. Eltawil, and F. Kurdahi, "Modeling and analysis of passive switching crossbar arrays," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 1, pp. 270–282, Jan. 2018.

[36] J. Kim, C. Lee, J. Kim, Y. Kim, C. S. Hwang, and K. Choi, "VCAM: Variation compensation through activation matching for analog binarized neural networks," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Jul. 2019, pp. 1–6.

[37] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst. (NIPS)*. Red Hook, NY, USA: Curran Associates, 2016, pp. 4114–4122.

[38] M. E. Fouda, J. Lee, A. M. Eltawil, and F. Kurdahi, "Overcoming crossbar nonidealities in binary neural networks through learning," in *Proc. 14th IEEE/ACM Int. Symp. Nanosc. Archit.*, Jul. 2018, pp. 1–3.

[39] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*. [Online]. Available: http://arxiv.org/abs/1502.03167

[40] Y. LeCun, C. Cortes, and C. Burges. (2010). MNIST Handwritten Digit Database. AT&T Labs. vol. 2, p. 18. [Online]. Available: http://yann.lecun.com/exdb/mnist

[41] A. Krizhevsky, V. Nair, and G. Hinton. (2014). *The Cifar-10 Dataset*. vol. 55. [Online]. Available: http://www.cs.toronto.edu/kriz/cifar.html

[42] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, vol. 1. Red Hook, NY, USA: Curran Associates, 2012, pp. 1097–1105.

[43] A. Vasudevan, A. Anderson, and D. Gregg, "Parallel multi channel convolution using general matrix multiplication," in *Proc. IEEE 28th Int. Conf. Appl.-Specific Syst., Archit. Processors (ASAP)*, Jul. 2017, pp. 19–24.

[44] C.-Y. Chen, H.-C. Shih, C.-W. Wu, C.-H. Lin, P.-F. Chiu, S.-S. Sheu, and F. T. Chen, "RRAM defect modeling and failure analysis based on march test and a novel squeeze-search scheme," *IEEE Trans. Comput.*, vol. 64, no. 1, pp. 180–190, Jan. 2015.

[45] L. Xia, W. Huangfu, T. Tang, X. Yin, K. Chakrabarty, Y. Xie, Y. Wang, and H. Yang, "Stuck-at fault tolerance in RRAM computing systems," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 1, pp. 102–115, Mar. 2018.

[46] C. Li, M. Hu, Y. Li, H. Jiang, N. Ge, E. Montgomery, J. Zhang, W. Song, N. Dávila, C. E. Graves, and Z. Li, "Analogue signal and image processing with large memristor crossbars," *Nature Electron.*, vol. 1, no. 1, p. 52, Jan. 2018.

[47] M. E. Fouda, A. M. Eltawil, and F. Kurdahi, "Minimal disturbed bits in writing resistive crossbar memories," in *Proc. 14th IEEE/ACM Int. Symp. Nanosc. Archit.*, Jul. 2018, pp. 1–3.

[48] B. Traore, P. Blaise, E. Vianello, H. Grampeix, S. Jeannot, L. Perniola, B. De Salvo, and Y. Nishi, "On the origin of low-resistance state retention failure in HfO_2-based RRAM and impact of Doping/Alloying," *IEEE Trans. Electron Devices*, vol. 62, no. 12, pp. 4029–4036, Dec. 2015.

[49] Y.-F. Lai, F. Chen, Z.-C. Zeng, P. Lin, S.-Y. Cheng, and J.-L. Yu, "Thermal stability and data retention of resistive random access memory with HfOx/ZnO double layers," *Chin. Phys. B*, vol. 26, no. 8, Aug. 2017, Art. no. 087305.

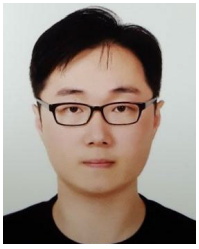
[50] L. Cai, W. Chen, Y. Zhao, X. Liu, J. Kang, X. Zhang, and P. Huang, "Insight into effects of oxygen reservoir layer and operation schemes on data retention of HfO_2-based RRAM," *IEEE Trans. Electron Devices*, vol. 66, no. 9, pp. 3822–3827, Sep. 2019.

[51] X. Sheng, C. E. Graves, S. Kumar, X. Li, B. Buchanan, L. Zheng, S. Lam, C. Li, and J. P. Strachan, "Low-conductance and multilevel CMOS-integrated nanoscale oxide memristors," *Adv. Electron. Mater.*, vol. 5, no. 9, Sep. 2019, Art. no. 1800876.

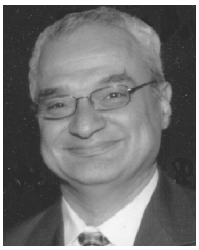
MOHAMMED E. FOUDA (Member, IEEE) received the B.Sc. degree (Hons.) in electronics and communications engineering and the M.Sc. degree in engineering mathematics from the Faculty of Engineering, Cairo University, Cairo, Egypt, in 2011 and 2014, respectively, and the Ph.D. degree from the University of California at Irvine, USA, in 2020. He is currently working as an Assistant Researcher with the University of California at Irvine. He has published more than 100 peer-reviewed journal and conference papers, one Springer book and three book chapters. His H-index is 20 with more than 1300 citations. His research interests include analog AI hardware, neuromorphic circuits and systems, brain-inspired computing, memristive circuit theory, fractional circuits and systems, and analog circuits. He was a recipient of the Best Paper Award in ICM 2013 and the Broadcom Foundation Fellowship from 2016 to 2017.He serves as a peer-reviewer for many prestigious journals and conferences. He also serves as a Review Editor for *Frontier of Electronics* and *International Journal of Circuit Theory and Applications* and the technical program committee member in many conferences.

SUGIL LEE received the B.S. degree in mathematical science from the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2017, and the M.S. degree in computer science and engineering from the Ulsan National Institute of Science and Technology (UNIST), Ulsan, South Korea, in 2019, where he is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering. His current research interest includes quantized deep neural networks and their energy-efficient implementation methodologies.

JONGEUN LEE (Member, IEEE) received the B.S. and M.S. degrees in electrical engineering and the Ph.D. degree in electrical engineering and computer science from Seoul National University, Seoul, South Korea, in 1997, 1999, and 2004, respectively. He is currently an Associate Professor with the Ulsan National Institute of Science and Technology (UNIST), Ulsan, South Korea. Since 2009, he has been on the faculty of the School of Electrical and Computer Engineering, UNIST. His research interests include neural network processors, reconfigurable architectures, and compilers.



GUN HWAN KIM received the Ph.D. degree in materials science and engineering from the Seoul National University, in 2012. He was a Senior Engineer with the ReRAM Team, Samsung Electronics, from 2012 to 2015. He is currently a Senior Researcher with the Korea Research Institute of Chemical Technology (KRICT). His current research interests include fabrication and characterization of ReRAM for non-volatile memory and neuromorphic applications.



FADI KURDAHI (Fellow, IEEE) received the B.E. degree in electrical engineering from the American University of Beirut, in 1981, and the Ph.D. degree from the University of Southern California, in 1987. Since then, he has been a Faculty Member with the Department of Electrical Engineering and Computer Science, University of California at Irvine, where he conducts research in the areas of computer-aided design, high-level synthesis, and design methodology of large scale systems. He serves as the Director of the Center for Embedded and Cyber-Physical Systems, comprised of world-class researchers in the general area of embedded and cyber-physical systems. He is a Fellow of the AAAS. He was the Program Chair or the General Chair on program committees of several workshops, symposia, and conferences in the area of CAD, VLSI, and system design. He received the Best Paper Award of the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION, in 2002, the Best Paper Award in 2006 at ISQED, and four other Distinguished Paper Awards at DAC, EuroDAC, ASP-DAC, and ISQED. He also received the Distinguished Alumnus Award from his Alma Mater, the American University of Beirut, in 2008. He served on numerous editorial boards.

AHMED M. ELTAWIL (Senior Member, IEEE) received the Ph.D. degree from the University of California, Los Angeles, in 2003, and the M.Sc. and B.Sc. degrees (Hons.) from Cairo University, Giza, Egypt, in 1999 and 1997, respectively. Since September 2019, he has been a Professor with the Computer, Electrical and Mathematical Science and Engineering Division (CEMSE), King Abdullah University of Science and Technology (KAUST), Thuwal, South Korea, where he is the Director of the Communications and Computing Systems Laboratory (CCSL). Since 2005, he has been with the Department of Electrical Engineering and Computer Science, University of California at Irvine, where he is the Founder and Director of the Wireless Systems and Circuits Laboratory. His research interests include mobile computing and communication platforms. Dr. Eltawil has been on the technical program committees and steering committees for numerous workshops, symposia, and conferences in the areas of low power computing and wireless communication system design. He received several awards, as well as distinguished grants, including the NSF CAREER grant supporting his research in low power systems.