This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2021.3051856, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems



# Unary Coding and Variation-Aware Optimal Mapping Scheme for Reliable ReRAM-based Neuromorphic Computing

Yanan Sun, Chang Ma, Zhi Li, Yilong Zhao, Jiachen Jiang, Weikang Qian, Rui Yang, Zhezhi He, and Li Jiang

***Abstract*—Neural network (NN) computing contains a large number of Multiply-and-ACcumulate (MAC) operations. The performance of NN accelerator is limited with the traditional von Neumann architecture due to the tremendous off-chip memory accesses. Resistive Random-Access Memory (ReRAM)-based crossbars can naturally perform Matrix-Vector Multiplication (MVM) operations and is well suitable for NN accelerators. In the existing ReRAM-based NN accelerators, the synaptic weights represented by the conductances of ReRAMs are mainly based on the binary coding. However, the imperfect fabrication process combined with stochastic filament-based switching leads to resistance variations of ReRAMs, which can significantly alter the weights in binary synapses and degrade the NN accuracy. Moreover, the NN accuracy further deteriorates with Multi-Level Cells (MLCs) used for reducing hardware overhead.**

**In this paper, a novel unary coding of synaptic weights is proposed to overcome the resistance variations of MLCs and achieve reliable ReRAM-based neuromorphic computing. A variation-aware optimal mapping scheme is also proposed in compliance with the unary coding to guarantee high accuracy by leveraging a unique feature of unary coding—the existence of multiple ways to represent the same value. The optimal mapping obtains very small errors for weights with resistance variations of MLCs. Our simulation results show that under resistance variations, the proposed method achieves less than $0.08\%$ and $3.43\%$ accuracy loss on CIFAR10 and ImageNet, respectively, compared to the ideal accuracy. With each synaptic weight represented by four 2-bit MLCs, the proposed method improves the accuracy over traditional binary coding scheme by $83.39\%$ and $87.6\%$ for CIFAR10 and ImageNet, respectively.**

***Index Terms*—Neural network, ReRAM, crossbar, MLC, variation, unary coding, optimal mapping**

## I. Introduction

Neural Networks (NNs) require a large number of Matrix-Vector Multiplications and tremendous memory accesses, which make the traditional von Neumann architecture unsuitable. Processing-In-Memory (PIM) enables computing inside memory and reduces the data movement between the processor and memory, thereby attracting widespread attention. Resistive Random-Access Memory (ReRAM) is an attractive emerging Non-Volatile Memory (NVM), with desirable characteristics, such as zero-standy power, scalability to 5nm or below, compatibility with silicon fabrication process, and multiple resistance levels [1]–[3]. More importantly, ReRAM supports efficient implementation of PIM [2]. Recent works present several NN accelerators based on ReRAM crossbars, such as PRIME [4], ISAAC [5], and PipeLayer [6]. Moreover, several ReRAM-based chips have been fabricated to accelerate NN computing using Multi-Level Cells (MLCs) [7] [8]. They use conductances of ReRAM, voltages, and currents to represent synaptic weights, inputs, and outputs, respectively. They are faster and more energy-efficient than CMOS-based accelerators [9].

However, due to the immaturity of fabrication process combined with stochastic filament-based switching, ReRAM suffers from the resistance variations problem [10] [11], manifested as the deviation of actual resistance from its target value. The resistance deviation affects the encoded weights, which can significantly degrade the accuracy of NNs [12].

The existing solutions to overcome the device resistance variations are either at the device level or the algorithmic level. At the device level, several techniques repeatedly write the devices [3] [13] [14] to lower the device variations. Unfortunately, the frequent write operations shorten the lifetime of ReRAMs. At the algorithmic level, the weights are trained to enhance the tolerance of NNs to the resistance variations of ReRAMs [15]–[18]. The works [15]–[17] presented the off-device training methods. However, the methods in [15] and [16] can only address the variation problems in Single-Layer Perceptron (SLP), which has limited application. Although the method in [17] is applicable to Convolution Neural Networks (CNNs) like VGG and AlexNet, it suffers from significant accuracy loss in the presence of large variations. To achieve higher accuracy in CNN with acceptable hardware cost, a combination of off-device and on-device training methods is proposed in [18]. The on-device training however still needs to rewrite the ReRAMs to update the weights, which hurts the device endurance. A common feature of the existing methods is that they all utilize the traditional binary coding to represent the synaptic weights, where the Most Significant Bits (MSBs) can amplify the device variations, causing large errors in the encoded weights. The use of the MLC can even exacerbate the problem.

We notice that along with the recent development in devices and applications, some emerging computing paradigms are also being actively studied. One of them is stochastic computing (SC) [19]. Different from traditional binary radix computing, it operates on unary coding [20], which is a coding format where all the bits have the same significance. Such a

This work was supported in part by the National Key Research and Development Program of China under Grant No. 2020YFB2205502 and No. 2018YFB1403400, National Natural Science Foundation of China (NSFC) under Grant No. 61704104 and No. 61834006, Shanghai Sailing Program under award 19YF1424900, and Shanghai Science and Technology Committee under Grant No. 18ZR1421400.

Authors are all with the Shanghai Jiao Tong University. Yanan Sun, Chang Ma, Zhi Li, and Jiachen Jiang are with the Department of Micro-Nano Electronics; Yilong Zhao, Zhezhi He, and Li Jiang are with the Department of Computer Science and Engineering; Rui Yang and Weikang Qian are with the University of Michigan-Shanghai Jiao Tong University Joint Institute. Weikang Qian is also with the MoE Key Laboratory of Artificial Intelligence, Shanghai Jiao Tong University; Li Jiang is also with Shanghai Qi Zhi Institute, Shanghai, China. Weikang Qian and Li Jiang are the corresponding authors (E-mail: {qianwk, ljiang_cs}@sjtu.edu.cn).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2021.3051856, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems



coding has a strong tolerance to bit-flip errors.

In our previous work [21], we proposed a variation-aware priority mapping to tackle the variation problem based on the unary coded synaptic weights. The method maps the lower resistance states to those devices with smaller variation first, which reduces the error of the represented value. In this paper, we further exploit the error-tolerance feature of unary coding and propose a novel optimal mapping scheme to more effectively address the reliability problem of ReRAM-based NN accelerators. In summary, our contributions are as follows:

- We employ unary coding, implemented with MLC, for weight representation in ReRAM-based NN design for the first time. We apply it to tolerate the device resistance variations, leading to the accuracy recovery of the ReRAM-based NN accelerators.
- We propose a variation-aware optimal mapping to further improve the accuracy. This technique exploits a unique feature of unary coding—the existence of multiple ways to represent the same value. The errors of the weights are minimized by employing the mapping scheme.
- We provide a comprehensive analysis of the design trade-off between NN accuracy and hardware cost overhead by considering the number of ReRAMs and the number of levels in each MLC for representing a synaptic weight.
- We demonstrate that our methods can tolerate the resistance variations for typical NNs. Simulation results show that our proposed method can significantly improve the accuracy over the traditional binary coding scheme by $83.39\%$ and $87.6\%$ for CIFAR10 and ImageNet, respectively, even under large device variations with the $\sigma$ of 1.0 shown in Eq. (1).

The rest of the paper is organized as follows: Section II introduces the background on ReRAM-based NN accelerator and discusses the limitation of the related works. Section III describes the proposed methods, including unary coding based on MLC devices and the optimal mapping. Section IV presents the simulation results. Section V concludes the paper.

## II. Background and Related Works

### A. ReRAM-based NN Computing

ReRAM is a two-terminal variable resistor with an oxide layer sandwiched between two metal electrodes, such as the Ta/$HfO_2$/Pt structure [22]. The device resistance ranges from low resistance state (LRS) to high resistance state (HRS) and can be tuned by a specific voltage. If a proper positive voltage is applied across the device, the resistance will change from HRS to LRS, known as a SET operation. Conversely, applying a reverse voltage will cause the resistance to change from LRS to HRS, known as a RESET operation. An MLC ReRAM has one or more middle resistance states (MRSs) between LRS and HRS, which can be achieved by precisely controlling the programming voltage [23]. The MRSs of an MLC device can be used to represent different logic values. Although, the ReRAM devices can be theoretically programmed to any target conductance, it is not realistic to use this character in practice [24] [25]. For reducing the hardware storage and computing resources, the ReRAM devices are assumed to be programmed to a limited number of target levels in this paper.

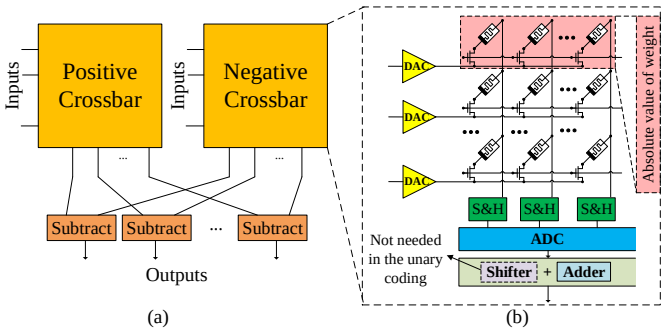


Fig. 1. ReRAM-based NN architecture. (a) Positive and negative weights are represented by two crossbars, and the output is the subtraction of two crossbars. (b) Internal structure of a crossbar, designed to implement MVM.

CNNs include convolution layers and fully connected layers. Both are MAC operations and can be converted into matrix-vector multiplications (MVMs). The 1T1R (one transistor and one ReRAM)-based crossbar can be employed as a matrix-vector multiplier [26] to accelerate NNs using the architecture shown in Fig. 1. Since weights in NNs are either positive or negative, two crossbars are used to represent the weight matrix, with one storing the positive weights and the other storing the negative weights [27] [28]. The absolute value of each weight is first converted into a binary number. Then, depending on the polarity of the weight, the bits of the binary number are used to configure multiple ReRAM devices in either the positive or the negative crossbar, while zero is written into the other crossbar. The output is the subtraction result of the two crossbars. In each crossbar, the input data are converted into a vector of voltage signals by a digital-to-analog converter (DAC) on each row. Usually, the DAC resolution is not very high [5]. Therefore, the input is divided into multiple bits and applied on the wordline cycle by cycle. The bitline current is converted into a voltage signal and fed into a sample-and-hold (S&H) unit for temporary storage. It is followed by an analog-to-digital converter (ADC). As the area of an ADC is relatively large, only one ADC is usually employed in each crossbar array with a time-multiplex fashion to perform conversion for each column. Each binary weight is represented by multiple ReRAMs with each having a different significance in binary coding. The output of the ADC is therefore required to be shifted. Besides, as the input voltage is divided into several cycles, the output of the ADC is further required to be shifted in each cycle and summed up to get the final result of the MVM.

### B. ReRAM Resistance Variation

Despite of the promising applications of ReRAM on NN accelerators, the ReRAM devices suffer from resistance variations where the actual resistance deviates from the target value. There are two components of the variation in memristive devices, that are device-to-device variation (DDV) and cycle-to-cycle variation (CCV) [24] [29] [30]. DDV (spatial variation) refers to the resistance variations among different

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2021.3051856, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems



cells mainly due to immature fabrication processes, such as oxide thickness variations and line edge roughness [30]. The DDV indicates the time-invariant deviations from the target resistance values for different cells which usually follows a log-normal distribution [24] [25]. Alternatively, CCV (temporal variation) means that different amount of resistance variations can occur in the same cell at different programming cycles due to stochastic filament-based switching [30]. The CCV imposes a time-varying additive resistance variation for each device regarding to different programming cycles, which typically follows a normal distribution [29].

In this work, we put emphasis on dealing with the DDV. The reasons are as follows. First, CCV is typically dominant for the on-device training where the weights are required to be written and updated frequently [31]. Alternatively, DDV is mainly considered for off-device training where weight matrix is only written into the crossbar once and followed by multiple read operations for inference process. In this study, the off-device training method is employed for reducing hardware cost and eliminating extra writing operations on RRAM-crossbars [18]. Second, there exist multiple write-and-verify methods [3] [13] [14] [32] to alleviate the resistance deviations induced by CCV. Unfortunately, there still lacks of effective ways to eliminate the DDV which is time-invariant and specific to the device with current imperfect fabrication processes. The DDV problem is of high priority and importance in current research stage.

In general, the actual resistance $R'$ of an ReRAM cell, under the effect of DDV, follows a log-normal distribution according to recent studies [3] [24]:

$$R' = R_0 \cdot e^{\theta} \qquad \theta \sim N(0, \sigma^2), \tag{1}$$

where $R_0$ is a target resistance value of ReRAM and $\theta$ follows a normal distribution with zero mean and a standard deviation of $\sigma$, ranging from 0 to 1.0 in the previous work [21] and this work. A large $\sigma$ indicates a large resistance variation. As mentioned in [15], we can detect the DDV by programming each device to a target state and sensing the deviation between the target state and the actual value. During the testing, the dedicated automated test equipment (ATE) [33] or built-in self-test (BIST) [34] [35] structure can be used to get the variation information of ReRAM devices.

### C. Related Works

A previous work [3] proposed to reduce the device resistance variations by repeatedly applying constant reset pulses to the ReRAM devices until the resistance level reaches the target range. Additionally, the Incremental Step Pulse and Verify Algorithm (ISPVA) was proposed in [13] [14]. It iteratively reads the actual resistance and writes it with higher programming voltage. However, these two methods have a common drawback: the frequent write operations for tuning the ReRAM resistance would inevitably shorten the device lifetime [36].

To overcome the resistance variations, some training methods were proposed to achieve robust NNs, including off-device training and on-device training. A variation-aware training method called *Vortex* was presented in [15]. It considers the variations during the training phase of the SLP and iteratively tunes a global parameter to obtain a relatively optimal weight matrix, which has a certain ability to tolerate device variations. The previous work [16] leveraged the self-healing capability of the NNs to decrease those weights that are mapped to the abnormal ReRAMs with large variations in the training phase for SLP. Unfortunately, these two approaches are not applicable to multi-layer perceptron (MLP) and CNN where the errors can be accumulated and amplified through multiple layers. Alternatively, the previous work [17] proposed a device variation-aware (DVA) training method which trains the DNN with random noise to get a robust model. DVA training cannot cope with large resistance variations as it does not take into account the variation information of each device. Moreover, this method exploits the redundancy of the NN, while the redundancy of the DNN is limited. The previous work [18] proposed a software and hardware co-design method to get higher accuracy in CNN, even under large device variations. The off-device training was employed to get a relatively high accuracy while the on-device training was used to further suppress the accuracy loss. However, the on-device training still needs to rewrite the ReRAMs, which is harmful for device endurance with shortened lifetime. Furthermore, the methods in [15], [16], [18] use only one device to represent the floating-point weight, which is unrealistic for MLC in current process technology. In most architectures, the MLC in ReRAM-based NN accelerator can be 2-bit or 3-bit, far below the precision of a floating-point data. Furthermore, the above methods all utilize the traditional binary coding to represent the synaptic weights. With the traditional binary coding, the MSBs are more significant than the other bits. Therefore, the weight deviations can be amplified, which induces significant loss of NN accuracy in the presence of large resistance variations. Furthermore, the MLC devices are commonly employed for reducing the hardware cost in NNs. Since the significance of an MLC bit is larger than that of a single-level cell (SLC) bit, the accuracy degradation of binary coding-based NNs is even worse under resistance variations.

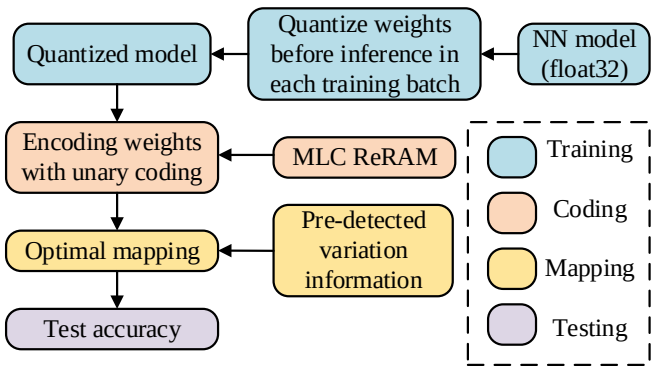


Fig. 2. The overall flow of the proposed unary coding with variation-aware optimal mapping method, from NN training to coding, mapping, and testing.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2021.3051856, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems



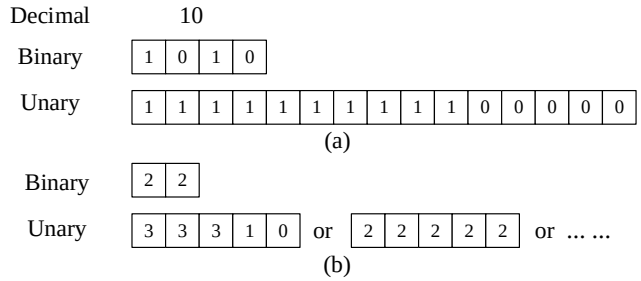


Fig. 3. An example for representing decimal value 10 in binary and unary coding: (a) For SLC; (b) For 2-bit MLC. The binary coding has only one form to represent the same value while the unary coding has multiple forms.

## III. Proposed Unary Coding and Variation-Aware Optimal Mapping Scheme

We propose a novel unary coding with variation-aware optimal mapping method to tolerate the resistance variations for MLC ReRAM-based NNs. The overall flow of the proposed method is shown in Fig. 2. First, we use a few iterations to retrain the NN model to obtain the quantized NN model. In the quantization process, we quantize the weights before inference in each training batch. Then, each synaptic weight is coded based on the proposed unary coding method (see Section III-A) with MLC devices (see Section III-B) for decreasing the weight deviation in the presence of resistance variations. The optimal mapping scheme (see Section III-C) is also applied to get the minimum weight error based on the pre-detected resistance variations information to further reduce the accuracy loss. Finally, we calculate the NN accuracy of the proposed method on the test dataset.

### A. Proposed Unary Coding Method

In the binary coding, each bit has different significance. If a binary number has $n$ bits, the significances from the least significant bit (LSB) to the MSB are $2^0, 2^1, \ldots, 2^{n-1}$, respectively. In contrast, the unary coding has all the bits of the same significance, equal to $2^0 = 1$, and the encoded value is expressed as the sum of all the bits. Thus, it essentially uses the number of 1s in the coding to represent a value, not the position of 1s. The straightforward unary coding maps 1s first, followed by 0s. Fig. 3(a) shows examples of the unary coding and binary coding with SLCs for the decimal value of 10. With 4 bits, the binary coding is 1010. For the unary coding, in order to represent the same range as the 4-bit binary coding, which is $[0, 15]$, 15 bits are needed. Thus, the unary coding for 10 contains ten 1s and five 0s.

The floating-point number, such as *float32*, is typically used in the training phase. If the NN model is deployed on a customized hardware, the overhead of using floating-point number is unacceptable. The weight is typically quantized to multiple-bit ReRAMs for reducing data storage and computation complexity in the binary coding [28], such as *int8*. Recent study [37] proved that NNs can be compressed in 2bit without accuracy degradation. However, the variations on the MSBs will magnify the weight deviation from the expectation severely in the traditional binary coding, as these bits have

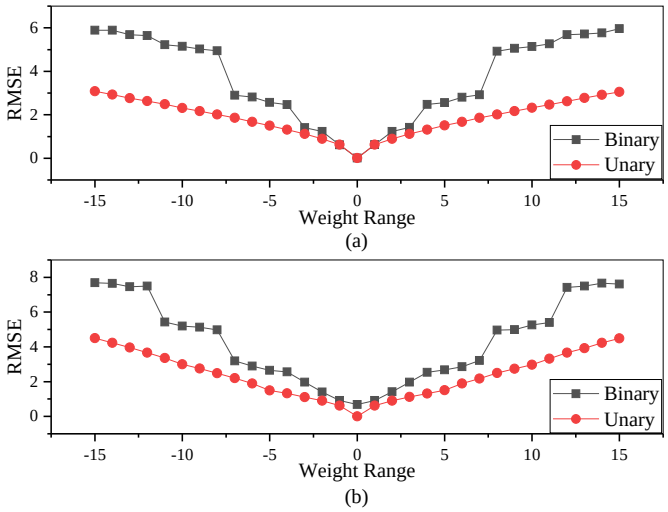


Fig. 4. The Root Mean Squared Error (RMSE) of the unary coding and the binary coding for all possible weights in the range $[-15, 15]$. (a) Fifteen SLCs for a synaptic weight. (b) Five 2-bit MLCs for a synaptic weight. The device resistance variation parameter $\sigma$ is set as 0.5.

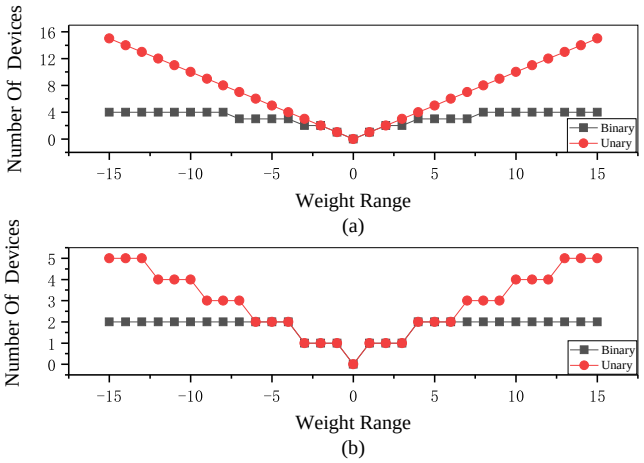


Fig. 5. The number of required devices of the unary coding and the binary coding for all possible weights in the range [-15,15]. (a) Based on SLCs. (b) Based on 2-bit MLCs.

larger significance. Alternatively, when the weight is unary coded, due to the equivalent significance of each bit, the weight deviation is reduced.

We perform simulation to demonstrate that the unary coding has lower deviation than the corresponding binary coding for representing the same value of weight. We use Root Mean Squared Error (RMSE) to measure the error of each possible weight with 50000 times Monte-Carlo simulation. Fig. 4(a) shows the RMSE of weight for the binary coding and the unary coding using SLCs. We assume $\sigma = 0.5$ in Eq. (1). The range of weights under consideration is $[-15, 15]$. For representing a synaptic weight in this range, fifteen SLC ReRAMs are required for the unary coding, while four SLC ReRAMs are needed for the binary coding. The number of required devices of the unary coding and the binary coding for all possible weight in the range $[-15, 15]$ is shown in Fig. 5(a) with SLCs. As shown in Fig. 4(a), the RMSE of the binary coding is larger

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2021.3051856, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems



TABLE I
THE FOUR RESISTANCE AND CONDUCTANCE VALUES IN A 2-BIT MLC.

| Logic state | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| Conductance | $\frac{1}{1000 \cdot LRS}$ | $\frac{334}{1000 \cdot LRS}$ | $\frac{667}{1000 \cdot LRS}$ | $\frac{1}{LRS}$ |
| Resistance | $1000 \cdot LRS$ | $3 \cdot LRS$ | $1.5 \cdot LRS$ | $LRS$ |
| Symbol | $HRS$ | $MRS1$ | $MRS2$ | $LRS$ |

than that of the unary coding in the whole range. Therefore, the unary coding causes less error for weight representation.

### B. Unary Coding with Multi-level Cell

An SLC has two resistance levels and can represent a 1-bit value. Using SLC to represent a synaptic weight requires a large number of ReRAMs. In contrast, an MLC has more than two resistance levels and can represent a value of more than one bit. For an MLC with $2^k$ resistance levels, it can represent a value of $k$ bits. We call it a $k$-bit MLC or a $2^k$-level MLC. Note that the number of levels of a device does not have to be an integer power of 2. An MLC can also be 3-level, 5-level, etc. Using MLC can shorten the encoding length. For example, to represent a value in the range $[0, 2^n - 1]$, $n$ and $2^n - 1$ SLCs are needed in the binary and unary coding, respectively. When $2^k$-level MLCs are used, only $\lceil \frac{n}{k} \rceil$ and $\left\lceil \frac{2^n-1}{2^k-1} \right\rceil$ of them are needed in the binary and unary coding, respectively. The number of required devices to represent the same weight range is shown in Fig. 5(b) with 2-bit MLCs. Comparing Figs. 5(a) and (b), by using MLCs, the gap between the unary coding and the binary coding shrinks. For example, to encode a decimal weight of 15, the unary coding with SLCs needs 11 more devices than that for binary coding as shown in Fig. 5(a). Alternatively, only 3 more devices are required for unary coding with 2-bit MLCs than that for binary coding as shown in Fig. 5(b).

Nowadays, the 2-bit MLC devices are widely used in NN architectures with current synthesis technology [5]. Each 2-bit MLC device can represent four different integral values 0, 1, 2, and 3. Fig. 3(b) shows two possible unary codings for the value 10 using 2-bit MLCs. Indeed, five 2-bit MLCs have the same representation range as fifteen SLCs for unary coding. In this case, with the help of MLC, only one third of ReRAMs are needed in comparison with SLC. The four different conductance levels of a 2-bit MLC are typically uniformly distributed. Current fabrication shows that the HRS-to-LRS ratio can reach 1000 [38], which is used in this study. The four resistance and conductance values in a 2-bit MLC are listed in Table I. The four different resistance levels LRS, MRS2, MRS1, and HRS represent four logic states 3, 2, 1, and 0, respectively.

From Sections III-A, each bit in the proposed unary coding with MLCs has equivalent significance. Therefore, all the bit positions are interchangeable without influencing the weight value. As shown in Fig. 3(b), the decimal number 10 can have many forms in the unary coding, such as "33310" and "22222". This makes the unary coding more flexible.

The RMSE of the unary coding and binary coding with 2-bit MLCs is compared in Fig. 4(b). To represent a synaptic weight range of $[-15, 15]$, five 2-bit MLCs are required for the unary coding, while two 2-bit MLCs are needed for the binary coding. Compared with the case with SLCs as shown in Fig. 4(a), the RMSEs of both unary coding and binary coding increase. Nevertheless, the RMSE of the unary coding with MLCs is still lower than that of the binary coding with MLCs for each synaptic weight, which shows the feasibility by using MLC-based unary coding to tackle the device variation problem.

Note that the unary coding has lower weight precision (i.e., smaller weight range) than the binary coding, which may degrade the NN accuracy. Therefore, we use a few training iterations to retrain the NN model to obtain the quantized NN model. In the quantization process, we quantize the weights before inference in each training batch. Then, the quantized NN model will have a smaller accuracy loss due to the reduction of the weight precision.

### C. Proposed Variation-aware Optimal Mapping Scheme

In this section, a variation-aware optimal mapping scheme is proposed to further lower the weight error and hence, the accuracy loss of NN, considering the resistance variations of MLC ReRAMs in crossbar arrays.

In this study, we use $N$ MLCs with $L$ levels to represent a weight. Assume that all the resistances follow the log-normal distribution in Eq. (1) with the same $\sigma$. The actual conductance of the $k$-th ($1 \le k \le N$) ReRAM $G'_k$ follows the distribution:

$$G'_k = G_k \cdot e^{-\theta_k}, \qquad \theta_k \sim N(0, \sigma^2),$$

where $G_k$ ($G_k \in \{0, 1, \ldots, L-1\}$) is the target conductance value of the $k$-th ($1 \le k \le N$) ReRAM and $\theta_k$ is a normally distributed random variable specific to the $k$-th ReRAM. The actual weight $w'$ can be represented as

$$w' = \sum_{k=1}^{N} G'_k = \sum_{k=1}^{N} G_k \cdot e^{-\theta_k}. \tag{2}$$

Under device variations, there exists an inevitable deviation between the actual weight and the target value. The goal of the proposed optimal mapping scheme is to find the optimal unary coding form to minimize the deviation, which can be formulated as the following optimization problem:

$$\min \left| \sum_{k=1}^{N} G_k \cdot e^{-\theta_k} - w \right|$$
$$\text{s.t. } G_k \in \{0, 1, \ldots, L-1\}, \text{ for } 1 \le k \le N.$$

The variables of the optimization problem are the target conductances $G_k$ ($1 \le k \le N$). The constraint is that $G_k$ must be an integer in the range $\{0, 1, \ldots, L-1\}$. Note that the widely-used MLC is 2-bit or 3-bit. Correspondingly, $L$ is no more than 8. The value of $N$ is also small, as a large $N$ means a large hardware cost. Therefore, $L^N$ is not large and we apply an exhaustive search over all $L^N$ possible $G_k$ combinations to find the optimal solution.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2021.3051856, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems



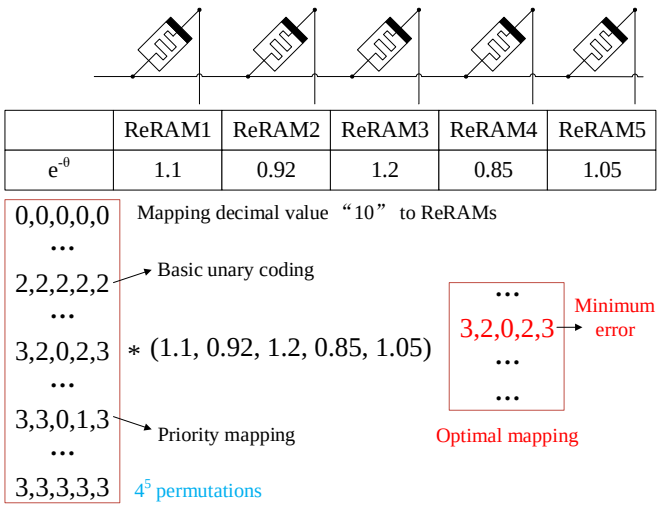


Fig. 6. An example of optimal mapping. Decimal value 10 is mapped to five 2-bit MLCs with the randomly assumed device conductance variance coefficients.

Fig. 6 gives an example of the proposed optimal mapping method. Assume that each weight is represented by five 4-level MLCs and that the target weight is 10. The table in Fig. 6 lists the variance coefficients $e^{-\theta_k}$ of the five devices. As noted in Section III-B, the unary coding has multiple forms for representing the same weight. To get the optimal form with the lowest weight error, we enumerate a total of $4^5$ possible $G_k$ combinations from "00000" to "33333". Then, we get the actual weight of each combination by applying Eq. (2). We find that the weight error is minimized when the unary coding form is "32023". The actual weight for "32023" is

$$w' = 1.1\times 3+0.92\times 2+1.2\times 0+0.85\times 2+1.05\times 3 = 9.99.$$

Compared to the target weight of 10, the weight error is $|9.99-10| = 0.01$. It is the smallest error among all possible $G_k$ combinations. Therefore, we adopt "32023" as the optimal unary coding form for mapping a weight of 10 in ReRAM devices.

To demonstrate the effectiveness of the proposed optimal mapping scheme, we further compare it with two other existing unary coding mapping methods. In the first method, all the components in the code are set as equal as possible, which makes the variance of the actual weight statistically lowest. For example, to represent the decimal weight of 10, it produces the unary coding "22222". The actual weight when mapping "22222" to the five MLCs in Fig. 6 is:

$$w' = 1.1\times 2+0.92\times 2+1.2\times 2+0.85\times 2+1.05\times 2 = 10.24.$$

Thus, it has a weight error of 0.24. Note that, we use this kind of unary coding format as the basic unary coding for comparison in the rest of paper.

The second method is the previously proposed priority mapping [21], which maps the lower resistance state to the devices with lower resistance variation. For example, it uses "33013" to represent the decimal weight of 10 based on the variance coefficients in Fig. 6. The actual weight is

$$w' = 1.1\times 3+0.92\times 3+1.2\times 0+0.85\times 1+1.05\times 3 = 10.06.$$

Thus, it has a weight error of 0.06, which is still larger than the proposed optimal mapping method. The optimal mapping can therefore obtain the lowest weight error as compared to the other two mapping methods.

Fig. 7 shows the weight errors of the optimal mapping ("Unary_opt"), priority mapping ("Unary_prio"), and basic unary coding ("Unary_basic") for all weights in the range $[-15, 15]$. Five 2-bit MLCs are used to represent a synaptic weight with unary coding. The RMSE is calculated with $\sigma$ set to 0.5. As shown in Fig. 7, the proposed optimal mapping provides the lowest RMSE among all the mapping methods. By considering the variation information of MLCs, the RMSE of "Unary_prio" is lower than that of "Unary_basic". Over the entire weight range, the average RMSE of the optimal mapping is reduced by 88.3% compared with the basic unary coding and by 81.2% compared with the priority mapping. As the optimal mapping scheme makes the weight more accurate, the accuracy loss of NN is expected to be reduced.

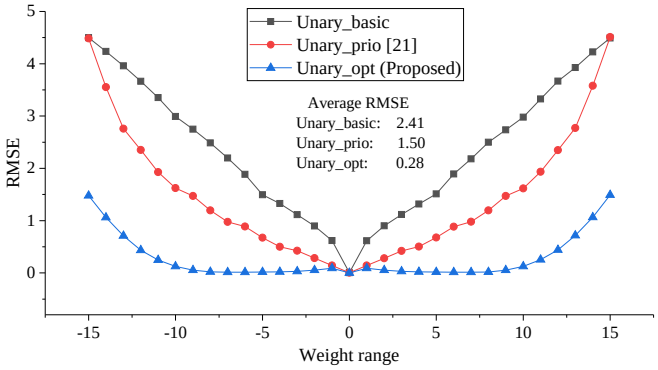


Fig. 7. The Root Mean Squared Error of basic mapping, priority mapping, and optimal mapping of unary coding for all the weights in the range $[-15, 15]$. We use five 2-bit MLC to represent a synaptic weight with unary coding. The device resistance variation parameter $\sigma$ is set as 0.5.

## IV. Simulation Results

### A. Simulation Setup

We evaluated our proposed method with two NNs (ResNet18 [39] and Vgg16 [40]) on two datasets (CIFAR10 [41] and ImageNet [42]). The CIFAR10 dataset consists of 10 classes of color images with size $32\times 32$, while the ImageNet dataset consists of 1000 classes of color images with size $224\times 224$. We implemented the NN models with Pytorch machine learning framework and ran them on a Nvidia 2080Ti GPU.

The simulation results are divided into two parts: accuracy and hardware performance. As shown in Fig. 8, we build an accuracy simulator based on Python and a hardware simulator based on C++. The accuracy simulator is faster and can get the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2021.3051856, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems



NN accuracy for a test dataset, while the hardware simulator is slower and can get the energy and area for calculating a single input picture.

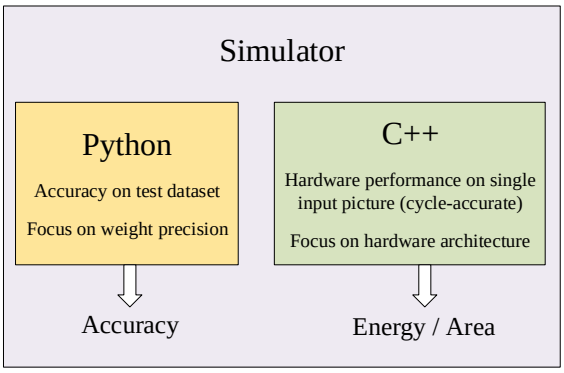


Fig. 8. The simulator used in this paper is implemented by Python and C++. The Python part is used to obtain the accuracy of NN with mapping scheme on the entire test dataset. The C++ part is used to obtain the energy required for the calculation of a single input image and the overall area of chip.

The accuracy simulator is based on Monte Carlo simulation. We used $\sigma$ shown in Eq. (1) to measure the degree of variation. We chose $\sigma$ ranging from 0 to 1.0, as the $\sigma$ in most fabrication processes does not exceed 1.0 [43]. The simulation flow of ReRAM variation in NN is shown in Fig. 9. We simulated the process of mapping the quantized weights to the corresponding ReRAM by setting the conductance of each cell to a certain level, and then injecting a log-normal distributed variation to the conductance. Afterwards, we updated the corresponding weight with respect to the new conductance. Finally, we conducted inference with the modified model on the test dataset to get the classification accuracy. The whole evaluation flow is based on cell's resistance, making the results more accurate.

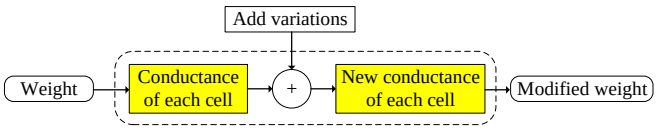


Fig. 9. The simulation flow of ReRAM variation in NN.

The hardware simulator is cycle-accurate. It can calculate the energy consumption and area of the architecture based on the data from ISAAC [5] as listed in Table II. The architecture for the binary coding is the same as that in ISAAC, which is shown in Fig. 10. Each chip is assumed to be composed of 168 tiles. Each tile has 12 in-situ multiply-accumulates (IMAs), which are the basic computation units in ISAAC. Eight $128 \times 128$ 1T1R crossbars and some peripheral circuits, such as ADC, DAC, S&H, shifter and adder, input register (IR), and output register (OR) are included in each IMA.

The architecture for the unary coding is almost the same as that for the binary coding except that the "shifter + adder" can be replaced by an adder. Since the energy and area of the "shifter + adder" only account for a small portion in the IMA, we still use the "shifter + adder" in the simulation of the architecture for the unary coding. Therefore, the unary-coding and binary-coding architectures have the same energy and area if they have the same $N$ and $L$.

TABLE II
THE PARAMETERS OF NEURAL NETWORK ACCELERATOR ARCHITECTURE [5].

| Component | Params | Spec | Power | Area ($mm^2$) |
|---|---|---|---|---|
| eDRAM Buffer | size<br>num_banks<br>bus_width | 64 KB<br>2<br>256 b | 20.7 mW | 0.083 |
| eDRAM -to-IMA bus | num_wire | 384 | 7 mW | 0.09 |
| Router | flit size<br>num_port | 32<br>8 | 42 mW | 0.151 (shared by 4 tiles) |
| Sigmoid | number | 2 | 0.52 mW | 0.0006 |
| Shift and Add | number | 1 | 0.05 mW | 0.00006 |
| MaxPool | number | 1 | 0.4 mW | 0.00024 |
| Output Register | size | 3 KB | 1.68 mW | 0.0032 |
| **Total** | | | **40.9 mW** | **0.215** $mm^2$ |
| IMA properties (12 IMAs per tile) | | | | |
| ADC* | resolution*<br>frequency<br>number | 8 bits<br>1.28 GSps<br>8 | 16 mW | 0.0096 |
| DAC | resolution<br>number | 1 bit<br>$8 \times 128$ | 4 mW | 0.00017 |
| Sample and Hold | number | $8 \times 128$ | 10 uW | 0.00004 |
| Memristor array | number<br>size<br>bits per cell | 8<br>$128 \times 128$<br>2 | 2.4 mW | 0.0002 |
| Shift and Add | number | 4 | 0.2 mW | 0.00024 |
| Input Register | size | 2 KB | 1.24 mW | 0.0021 |
| Output Register | size | 256 B | 0.23 mW | 0.00077 |
| **IMA Total** | **number** | **12** | **289 mW** | **0.157** $mm^2$ |
| **1 Tile Total** | | | **330 mW** | **0.372** $mm^2$ |
| **168 Tiles Total** | | | **55.4 W** | **62.5** $mm^2$ |
| Hyper Transport | links/freq<br>link bw | 4/ 1.6GHz<br>6.4 GB/s | 10.4 W | 22.88 |
| **One Chip Total** | | | **65.8 W** | **85.4** $mm^2$ |

*The resolution of ADC depends on the number of levels in ReRAM devices in this study.

## B. Accuracy

In this section, we compared the NN accuracy for various weight coding and mapping methods under resistance variations. We tested them using ResNet18 on ImageNet dataset. The weights are represented by four 2-bit MLCs in the simulation, which makes the weights encoded by the binary coding have a precision of 8 bits. It is known that the 8-bit quantized weights has the same NN accuracy as using the floating-point weights. For the unary coding, it can only represent weights in the range $[0, 12]$.

The comparison of various methods is shown in Fig. 11. "Ideal" refers to the ideal accuracy where the floating-point weights are used. "Binary" refers to traditional binary coding. The binary coding gets a large accuracy loss when variation exists. Even with $\sigma$ of 0.2, it has nearly 20% loss. The basic

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2021.3051856, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems



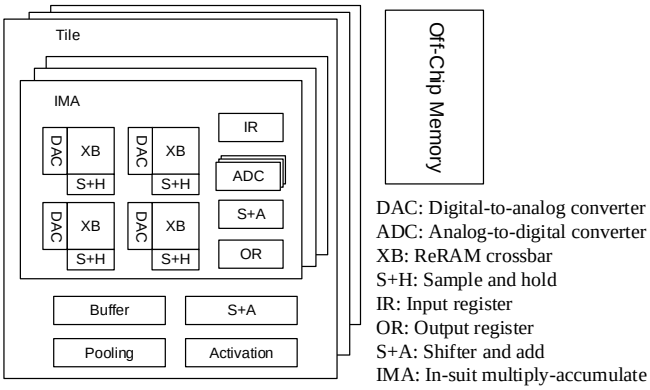


Fig. 10. The architecture of the ReRAM-based NN accelerator. The cycle-accurate simulator is built based on this architecture to get the energy and area.

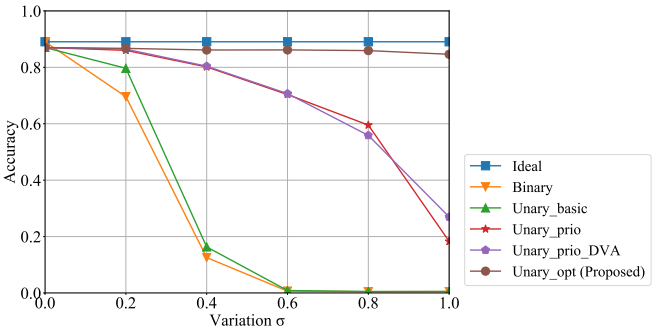


Fig. 11. The top5 accuracy of ResNet18 on ImageNet dataset by varying $\sigma$ from 0 to 1.0 among different methods. Both the number of ReRAMs for representing a weight and the MLC level are set to be 4 for these methods.

unary coding method, indicated by the curve "Unary_basic", is better for this situation, which is 10% higher than the binary coding. However, with the increase of variation $\sigma$, both the binary and unary codings have a large accuracy drop.

"Unary_prio" refers to the previously published priority mapping with unary coding [21]. "Unary_prio_DVA" further adds the DVA training to "Unary_prio". The "DVA" method [17] is conducted to train a robust NN model by adding random noise to weights at the training phase. The essence of NN training is to find a point in the weight space minimizing the loss function. Generally, the loss function change near the minimum point is steep, and the ability to resist weight deviation is poor. The "DVA" method adds a certain degree of random noise to the weights in the training phase in order to find a point in the weight space with a relatively small loss function (e.g., accuracy drop no more than 1%). Although the loss function at this point is not the smallest, the vicinity is flat, even if the weights fluctuate greatly. Thus, the NN accuracy will not be greatly affected even under resistance variations. This method essentially exploits the redundancy in the NN. However, it is less effective for big dataset such as ImageNet and deep NN such as ResNet18. As shown in Fig. 11, the "Unary_prio" is close to the "Unary_prio_DVA" curve, demonstrating the ineffectiveness of the DVA method. Thus, we only reproduce the "Unary_prio" method for comparison in the following part. Under a large variation of $\sigma = 1.0$, the priority mapping method has more than 60% loss compared with the ideal accuracy of 89.066%.

Our proposed optimal mapping with unary coding is shown in the "Unary_opt" curve. Compared to the "Unary_prio", our proposed method has only 4.4% loss when $\sigma = 1.0$, which shows great variation tolerance of our proposed method.

Note that, a large part of the accuracy loss in "Unary_opt" is due to the lower representation precision of the unary coding. Even after quantization training, the unary coding gives an accuracy of 87.014% at $\sigma = 0.0$, which is 2.052% lower than the ideal one. In this work, we only used the simplest quantitation training method. If a more effective quantization training method is used, the accuracy could be higher.

The accuracy of various methods for four different combinations of NN and dataset is listed in Table III. As shown in previous works, the larger the $\sigma$, the lower the accuracy of NN. Therefore, we chose an extreme value of $\sigma$ as 1.0. If the proposed method can still guarantee a small accuracy loss under this extreme case, the accuracy loss is also small for $\sigma < 1.0$. The accuracy for $\sigma$ as 1.0 is listed in Table III. The "Ideal" row gives the ideal accuracy with floating-point format weights and no variation injected. All the other methods use four 2-bit MLCs to represent a weight to make their hardware costs equal.

We calculate the accuracy loss of our proposed method by subtracting the accuracy of "Unary_opt" from that of "Ideal". We can see that the proposed "Unary_opt" method has small accuracy loss for all the four combinations of NN and dataset. Indeed, the proposed method gets less than $0.89\%$ top1 accuracy loss on CIFAR10 dataset, and less than $4.45\%$ top5 accuracy loss on ImageNet dataset. The row "Unary_opt - Binary" shows the accuracy improvement of the proposed "Unary_opt" method over the traditional binary coding. We can see that the proposed method has at least $62.58\%$ higher accuracy than the traditional binary coding. We also compare the accuracy difference between the previous "Unary_prio" and the proposed method, as shown in the row "Unary_opt - Unary_prio". We can see that we can get more accuracy improvement from "Unary_prio" to "Unary_opt" for Vgg16 on ImageNet dataset. The reason is that the "Unary_prio" method has lower accuracy on large dataset such as ImageNet. Furthermore, the "Unary_prio" has even lower accuracy for NN with more weights such as Vgg16.

### C. Energy and Area

In this section, we evaluate the hardware overheads of the proposed method and study the trade-off between hardware overhead and test accuracy of NNs. We mainly focus on two factors influencing the energy and area of the chip: ReRAM number ($N$) for representing a weight and MLC level ($L$).

In the cycle-accurate simulator, "shift + adder" is used to accumulate the outputs of the ADCs without shift operations. Note that, the input data to the DACs and the output data of the "shifter + adder" are all binary coded. Only the weights in the crossbars are in unary coding.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2021.3051856, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems



TABLE III
ACCURACY OF IDEAL, BINARY CODING, UNARY CODING, UNARY CODING+PRIORITY MAPPING, AND UNARY CODING+OPTIMAL MAPPING FOR FOUR COMBINATIONS OF NN AND DATASET. WE SET $N = 4$, $L = 4$, AND $\sigma = 1.0$.

| Network | ResNet18 | Vgg16 | ResNet18 | | Vgg16 | |
|---|---|---|---|---|---|---|
| Dataset | CIFAR10 | CIFAR10 | ImageNet | | ImageNet | |
| Accuracy / % | top1 | top1 | top1 | top5 | top1 | top5 |
| Ideal | 94.30 | 93.66 | 69.68 | 89.07 | 73.36 | 91.52 |
| Binary | 10.83 | 10.59 | 0.11 | 0.51 | 0.11 | 0.49 |
| Unary_basic | 12.56 | 10.23 | 0.11 | 0.54 | 0.10 | 0.56 |
| Unary_prio | 74.16 | 74.23 | 18.30 | 39.36 | 2.87 | 9.11 |
| Unary_opt | 94.22 | 92.77 | 62.69 | 84.62 | 67.57 | 88.09 |
| Unary_opt - Binary | 83.39 | 82.18 | 62.58 | 84.11 | 67.46 | 87.60 |
| Unary_opt - Unary_prio | 20.06 | 20.54 | 44.39 | 45.26 | 64.70 | 78.98 |
| Loss (Ideal - Unary_opt) | 0.08 | 0.89 | 6.99 | 4.45 | 5.79 | 3.43 |

We calculated the energy consumption of processing a single input image. The crossbar frequency is set to be 10 MHz, while the ADC frequency is set to be 1.28 GHz to ensure that in each crossbar cycle, ADC can complete the conversion of 128 columns.

The ADC resolution is determined by the MLC level. For 1-bit DAC, 128-row crossbar array, and $L$-level MLC, the ADC resolution is

$$r = \lceil \log_2 \left[ (L-1) \times 128 \times 1 \right] \rceil . \tag{3}$$

In this work, the successive approximation register (SAR) ADC is used. It contains a capacitive DAC. The power and area of the capacitive DAC increase exponentially with the ADC resolution, while those of the remaining part of the SAR ADC increase linearly [44]. We estimate the power and area of the ADC following this model using the data in [45].

Based on the above discussion, we can see that the MLC level $L$ affects the ADC power and area and hence, the chip energy and area. Besides, the ReRAM number $N$ for representing a weight determines the number of crossbar and hence, the chip area and energy. In summary, the energy and area are functions of $N$ and $L$. This can be expressed as

$$Energy \ / \ Area = f(N,\ L). \tag{4}$$

In this experiment, we varied $N$ from 1 to 5 and $L$ from 2 to 10. We tested the accuracy of the ResNet18 on CIFAR10 at $\sigma$ of 1.0.

Fig. 12 shows the relationship between the energy and the NN accuracy. Different colors correspond to different numbers of ReRAMs used for representing a weight, ranging from 1 to 5. Different dots on a curve represent different MLC levels, which vary from 2 to 10 with a step size of 2. The bottom left dot on each curve means using 2-level ReRAM to represent a weight, while the upper right dot means using 10-level MLCs. An ReRAM of 2 levels corresponds to an SLC, while ReRAMs of 4 and 8 levels correspond to 2-bit and 3-bit MLCs, respectively. In the unary coding, we do not need to set the ReRAM level as a power of 2 as in the binary coding. Therefore, ReRAMs of 6 and 10 levels are also possible, making the use of the device more flexible. The figure shows that, with the increase of the MLC level, the accuracy improves, but the energy increases. As the ReRAM number increases, the energy increases significantly. It can

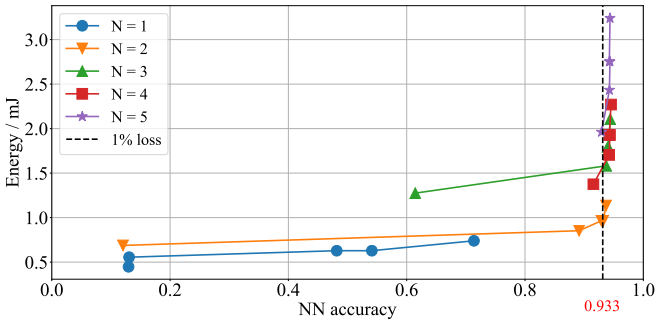


Fig. 12. The relationship between computing energy and NN accuracy with variation $\sigma$ set to 1.0. The NN and the dataset are ResNet18 and CIFAR10, respectively. For each curve, from the lower left corner to the upper right corner, $L$ is from 2 to 10 with a step size of 2.

be explained as follows. When $L$ increases, the increase in energy consumption mainly comes from the increase in ADC, as its resolution is higher. However, when $L$ doubles, the ADC resolution does not double, making the energy increase less than 100%. The increase of $N$ affects the number of tiles activated in the computing. When $N$ doubles, the numbers of crossbars and tiles used almost double, thus doubling the energy. Therefore, the energy is more sensitive to $N$, rather than $L$. This indicates that in order to achieve an acceptable accuracy while minimizing the energy, a high MLC level and a small ReRAM number are preferred.

As shown in the figure, there exists a trade-off between the energy and accuracy. We further identify a good trade-off point. In many situations, NN accuracy is of high priority. Therefore, we set an acceptable accuracy as 1% less than the ideal accuracy. The vertical dotted line in Fig. 12 corresponds to the acceptable accuracy. We then look for a point with the smallest $y$-coordinate near the vertical line. We find that the point corresponding to $L = 8$ and $N = 2$ has the lowest energy with only 1.1% accuracy loss. It is the best number and level configuration for such a combination of NN and dataset.

Similarly, we also show the relationship between the area and the NN accuracy in Fig. 13. The area is the total area of all the modules in a chip. Different colors correspond to different numbers of ReRAMs used for representing a weight, ranging from 1 to 5. Different dots on a curve correspond to different MLC levels, ranging from 2 to 10 with a step size

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2021.3051856, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems



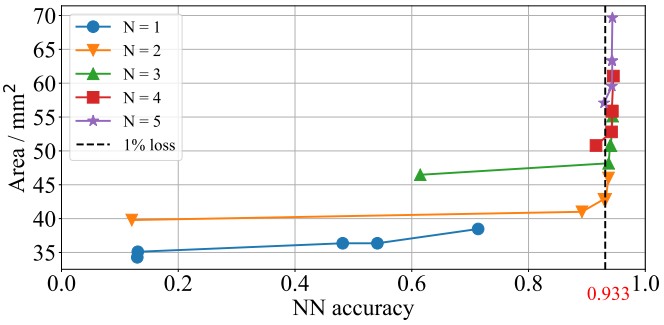


Fig. 13. The relationship between area and NN accuracy with variation $\sigma$ set to 1.0. The NN and the dataset are ResNet18 and CIFAR10, respectively. For each curve, from the lower left corner to the upper right corner, $L$ is from 2 to 10 with a step size of 2.

of 2. The bottom left dot in each curve means 2 levels, while the upper right dot means 10 levels. From this figure, we find that, with the increase of the ReRAM number, the area of the chip increases significantly, as more crossbars are needed to store the weights. The area also increases with the MLC level, and the larger the ReRAM number, the faster the area increases with MLC level. This is because with the MLC level increasing, the area of the ADC increases and hence, the area of a tile. When the ReRAM number is larger, more tiles are needed. Thus, the faster the chip area will increase. From this figure, we can conclude that in order to achieve an acceptable accuracy while minimizing the area, a high MLC level and a small ReRAM number are preferred.

Similarly, we also try to find the configuration of $N$ and $L$ that gives the smallest area while having an accuracy loss no more than 1% over the ideal case. Fig. 13 shows that the point has $L = 8$ and $N = 2$, and its accuracy loss is just 1.0%.

In order to show the impact of different configurations of $N$ and $L$ on each hardware module, we calculated the energy consumption ratio and area ratio of each module in the chip with the hardware simulator. Fig. 14(a) shows the energy breakdown of several major hardware modules, including DAC, ADC, Crossbar, Shift and Adder (S&A), and Embedded memory & Inter-tile link (Mem&Link). Their energy consumption accounts for a considerable proportion. The sum of the energy of all the other modules accounts for less than 0.2%, so these modules are not listed. Fig. 14(b) shows the area breakdown of DAC, ADC, Crossbar, Shift and Adder (S&A), and Embedded memory & Inter-tile link (Mem&Link) and the other modules. The figure presents three typical configurations of $N = 2$ and $L = 2$ (N2L2), $N = 2$ and $L = 8$ (N2L8), and $N = 8$ and $L = 2$ (N8L2).

As shown in Fig. 14(a), when $N$ remains unchanged, the proportion of energy consumed by ADC increases significantly with $L$. This is because increasing $L$ will increase the energy consumption of the ADC. When $L$ remains unchanged, the energy consumption of each part almost keeps constant with $N$ varying. This is because increasing $N$ will result in an increase in the number of times each module is activated, but the energy ratio of each module remains unchanged. For area, Fig. 14(b) shows that the increase of $L$ will cause the area ratio of the

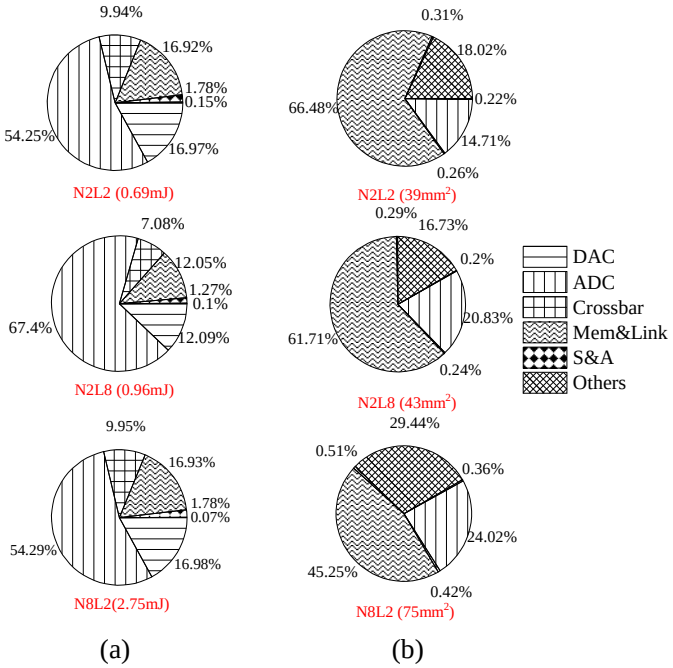


Fig. 14. (a) The energy breakdown and (b) area breakdown of DAC, ADC, Crossbar, Embedded memory&Inter-tile Link, S&A, and other modules. "N2L2", "N2L8", and "N8L2" means two 2-level MLCs, two 8-level MLCs, and eight 8-level MLCs for representing a synaptic weight, respectively. Network and dataset are ResNet18 and CIFAR10, respectively.

ADC increases. This is because with $L$ increasing, the ADC area increases, while the areas of the other modules are not affected. From the figure, we can also see that an increase in $N$ or $L$ will increase energy and area. By comparing the energy consumption and area of N8L2 with those of N2L8, we also find that the increase in energy consumption and area caused by increasing $N$ is more than those caused by increasing $L$. This aligns with the previous conclusion that a high MLC level $L$ and a small ReRAM number $N$ are preferred.

The above conclusions on energy and area are obtained for ResNet18 and CIFAR10. We actually conducted simulations for all the four combinations of NN and dataset, and the results are listed in Table IV. In this table, two different preferences are considered. The first is on lower hardware cost such as energy and area and the corresponding results are listed in the "HW" rows. The second is on lower accuracy loss and the corresponding results are listed in the "ACC" rows. The tuning variables are the ReRAM number $N$ for representing a weight and the MLC level $L$, which are shown in the "N&L" column. For example, "N2L8" means using two 8-level (3-bit) MLC to represent a synaptic weight. The number of tiles that is required for each case is also listed in Table IV. We also set an accuracy loss limit at $\sigma = 1.0$. For the simpler dataset CIFAR10, the constraint is more restrictive. The top1 accuracy loss is set to be less than 1%. For the more complex dataset ImageNet, it is hard to guarantee 1% accuracy loss. Therefore, we set a looser accuracy constraint aiming at top5 accuracy with less than 5% accuracy loss.

From this table, it can be seen that lower energy and area are achieved when the ReRAM number is small. Indeed, for

TABLE IV
THE TRADE-OFF BETWEEN ACCURACY LOSS AND ENERGY/AREA ON FOUR COMBINATIONS OF NN AND DATASET. "HW" MEANS THAT THE PREFERENCE IS ON LOWER ENERGY AND AREA AND "ACC" MEANS THAT THE PREFERENCE IS ON LOWER ACCURACY LOSS. IN "N&L" COLUMN, THE NUMBER AFTER THE LETTER "N" DENOTES THE RERAM NUMBER, WHILE THE NUMBER AFTER THE LETTER "L" DENOTES THE MLC LEVEL. THE ACCURACY LOSS IS OBTAINED AT $\sigma = 1.0$.

| NN&Dataset | Preference | N&L | Tiles | Loss | Energy/mJ | Area/mm$^2$ |
|---|---|---|---|---|---|---|
| ResNet&CIFAR | HW | N2L8 | 43 | 1.10% | 0.96 | 42.86 |
| | ACC | N4L4 | 71 | 0.08% | 1.7 | 52.81 |
| Vgg&CIFAR | HW | N2L8 | 92 | 1.00% | 0.57 | 65.63 |
| | ACC | N4L4 | 176 | 0.89% | 1.01 | 119.94 |
| ResNet&ImageNet | HW | N2L10 | 44 | 6.40% | 55.62 | 46.54 |
| | ACC | N4L4 | 73 | 4.45% | 83.47 | 53.65 |
| Vgg&ImageNet | HW | N2L10 | 358 | 6.32% | 26.61 | 261.14 |
| | ACC | N4L4 | 709 | 3.43% | 47.03 | 413.24 |

all the combinations of NN and dataset, lower energy and area are achieved when $N = 2$. This is because when fewer ReRAMs are used to represent a weight, a crossbar can store more weights and fewer IMAs and tiles are needed to deploy the whole NN. To maintain a high accuracy at the same time, we need a relatively large MLC level to make a wide enough range for weight, e.g., $L = 8$ for CIFAR10 and $L = 10$ for ImageNet.

The "ACC" rows all have $N = L = 4$. This configuration is a good trade-off between the energy/area and the accuracy loss. Although it does not leads to the lowest energy and area, it has a lower accuracy loss. However, it is definitely not the configuration with the minimum accuracy loss. In the unary coding, when the loss is minimum, the range of the weights needs to be as close as possible to the case of binary coding. This will cause $N$ and $L$ to be unacceptably large, which is obviously unnecessary. Therefore, if there are no strict restrictions on energy and area, "N4L4" is a better configuration with a relatively high accuracy.

The simulation data further verifies the previous conclusions and gives how to choose the ReRAM number and MLC level under the condition of energy or area limit. Minimal energy and area are achieved at "N2L8" for small dataset like CIFAR10 and at "N2L10" for big dataset like ImageNet. If we want a good tradoff between hardware cost and accuracy, "N4L4" is a good choice.

## V. CONCLUSION AND FUTURE WORKS

ReRAM-based crossbars can be used to accelerate NN inference. However, the immature fabrication process of the ReRAM devices leads to severe resistance variations, which degrades the accuracy of ReRAM-based NN accelerators. We propose a new unary coding method to represent synaptic weights for achieving reliable ReRAM crossbar-based NNs by leveraging its characteristics that the significance of different bits is the same and hence, it will not amplify the deviation of a particular device. A variation-aware optimal mapping scheme based on MLC ReRAMs is also proposed to work together with unary coding to further reduce the NN accuracy loss. Our simulation results show that the proposed method makes the ReRAM-based NN accelerators more tolerant to large device variations, even with large dataset like ImageNet. Besides, through the trade-off study between the energy/area and the NN accuracy, we demonstrate that a high MLC level and a small ReRAM number are preferred to achieve an acceptable accuracy while minimizing the energy and area.

## REFERENCES


[1] Y. Sun *et al*., "Energy-efficient nonvolatile SRAM design based on resistive switching multi-level cells," IEEE Transactions on Circuits and Systems II (TCAS-II), vol. 66, no. 5, pp. 753-757, May 2019.

[2] S. Kvatinsky *et al*., "MAGIC-memristor-aided logic," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 61, no. 11, pp. 895-899, November 2014.

[3] S. R. Lee *et al*., "Multi-level switching of triple-layered TaOx RRAM with excellent reliability for storage class memory," Symposium on VLSI Technology (VLSIT), pp. 71-72, June 2012.

[4] P. Chi *et al*., "PRIME: a novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), pp. 27-39, June 2016.

[5] A. Shafiee *et al*., "ISAAC: a convolutional neural network accelerator with in-situ analog arithmetic in crossbars," ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), pp. 14-26, June 2016.

[6] L. Song, X. Qian, H. Li and Y. Chen, "Pipelayer: a pipelined ReRAM-based accelerator for deep learning," IEEE International Symposium on High Performance Computer Architecture (HPCA), pp. 541-552, February 2017.

[7] C. Li, *et al*., "Analogue signal and image processing with large memristor crossbars," Nature Electronics, pp. 52-59, January 2018.

[8] P. Yao, *et al*., "Fully hardware-implemented memristor convolutional neural network," Nature, pp. 641-646, January 2020.

[9] Y. Chen *et al*., "Dadiannao: a machine-learning supercomputer," 47th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 609-622, December 2014.

[10] H. Li *et al*., "Variation-aware, reliability-emphasized design and optimization of RRAM using SPICE model," Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1425-1430, March 2015.

[11] Z. He, J. Lin, R. Ewetz, J. Yuan and D. Fan, "Noise injection adaption: end-to-end ReRAM crossbar non-ideal effect adaption for neural network mapping," 56th ACM/IEEE Design Automation Conference (DAC), pp. 1-6, August 2019.

[12] Y. Long, T. Na, S. Mukhopadhyay, "ReRAM-based processing-in-memory architecture for recurrent neural network acceleration," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 12, pp. 2781-2794, December 2018.

[13] F. Crupi *et al*., "Implications of the incremental pulse and verify algorithm on the forming and switching distributions in reram arrays," IEEE Transactions on Device and Materials Reliability, vol. 16, no. 3, pp. 413-418, September 2016.

[14] E. Perez, A. Grossi, C. Zambelli, P. Olivo, and C. Wenger, "Impact of the incremental programming algorithm on the filament conduction in hfo2-based rram arrays," IEEE Journal of the Electron Devices Society, vol. 5, no. 1, pp. 64-68, January 2017.

[15] B. Liu *et al*., "Vortex: variation-aware training for memristor X-bar," 52nd ACM/EDAC/IEEE Design Automation Conference, pp. 1-6, June 2015.

[16] L. Chen *et al*., "Accelerator-friendly neural-network training learning variations and defects in RRAM crossbar," Design, Automation & Test in Europe Conference & Exhibition (DATE) , pp. 19-24, March 2017.

[17] Y. Long, X. She and S. Mukhopadhyay, "Design of reliable DNN accelerator with un-reliable ReRAM," Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1769-1774, March 2019.

[18] Z. Song, Y. Sun, L. Chen, T. Li, N. Jing, X. Liang, and L. Jiang, "ITT-RNA: imperfection tolerable training for RRAM-crossbar based deep neural-network accelerator," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, April 2020.

[19] A. Alaghi, W. Qian and J. P. Hayes, "The promise and challenge of stochastic computing," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 37, no. 8, pp. 1515-1531, August 2018.

[20] S. Mohajer, Z. Wang and K. Bazargan, "Routing magic: performing computations using routing networks and voting logic on unary encoded data," International Symposium on Field-Programmable Gate Arrays, pp. 77-86, 2018.

[21] C. Ma, Y. Sun, W. Qian, Z. Meng, R. Yang and L. Jiang, "Go unary: a novel synapse coding and mapping scheme for reliable ReRAM-based neuromorphic computing," Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1432-1437, March 2020.
[22] H. Jiang *et al*., "Sub-10 nm ta channel responsible for superior performance of a hfo2 memristor," Scientific reports, vol. 6, June 2016.
[23] L. Gao, P. Chen and S. Yu, "Programming protocol optimization for analog weight tuning in resistive memories," IEEE Electron Device Letters, vol. 36, no. 11, pp. 1157-1159, November 2015.
[24] A. Fantini1 *et al*., "Intrinsic switching variability in HfO2 RRAM," 5th IEEE International Memory Workshop, pp. 30-33, May 2013.
[25] A. Mohanty, *et al*., "Random sparse adaptation for accurate inference with inaccurate multi-level RRAM arrays," IEEE International Electron Devices Meeting (IEDM), pp. 6.3.1-6.3.4, December 2017.
[26] M. Hu *et al*., "Dot-product engine for neuromorphic computing: programming 1t1m crossbar to accelerate matrix-vector multiplication," ACM 53rd annual design automation conference, pp. 1-6, August 2016.
[27] B. Li, Y. Wang, Y. Wang, Y. Chen and H. Yang, "Training itself: mixed-signal training acceleration for memristor-based neural network," 19th Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 361-366, January 2014.
[28] Z. Zhu *et al*., "A configurable multi-precision CNN computing framework based on single bit RRAM," 56th Annual Design Automation Conference, pp. 1-6, June 2019.
[29] J. Lee *et al*., "Exploring cycle-to-cycle and device-to-device variation tolerance in MLC storage-based neural network training," IEEE Transactions on Electron Devices, Vol. 66, No. 5, pp. 2172-2178, May 2019.
[30] R. Degraeve *et al*., "Causes and consequences of the stochastic aspect of filamentary RRAM," Microelectronic Engineering, vol. 147, pp. 171-175, November 2015.
[31] Y. Zhang, G. He, G. Wang and Y. Li, "Efficient and robust RRAM-based convolutional weight mapping with shifted and duplicated kernel," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, June 2020.
[32] Y. Luo, *et al*., "Array-level programming of 3-Bit per cell resistive memory and its application for deep neural network inference," IEEE Transactions on Electron Devices, vol. 67, no. 11, pp. 4621-4625, November 2020.
[33] A. Grossi *et al*., "An automated test equipment for characterization of emerging MRAM and RRAM arrays," IEEE Transactions on Emerging Topics in Computing, Vol. 6, No. 2, pp. 269-277, June 2018.
[34] H. Aziza *et al*., "A built-in self-test structure (BIST) for Resistive RAMs characterization: application to bipolar OxRRAM," Solid-State Electronics, pp. 73-78, 2015.
[35] X. Cui *et al*., "Design and test of the in-array build-in self-test scheme for the embedded RRAM array," IEEE Journal of the Electron Devices Society, Vol. 7, pp. 1007-1012, 2019.
[36] B. Chen *et al*., "Physical mechanisms of endurance degradation in TMO-RRAM," IEEE International Electron Devices Meeting (IEDM), pp. 12.3.1-12.3.4, December 2011.
[37] J. Choi, S. Venkataramani, V. Srinivasan, K. Gopalakrishnan, Z. Wang and P. Chuang, "Accurate and efficient 2-bit quantized neural networks," MLSys, 2019.
[38] R. Han *et al*., "Demonstration of logic operations in high-performance RRAM crossbar array fabricated by atomic layer deposition technique," Nanoscale Research Letters, January 2017.
[39] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770-778, December 2016.
[40] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition, " International Conference on Learning Representations (ICLR), November 2015.
[41] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Technical report, Citeseer, 2009.
[42] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Neural Information Processing Systems, 2012.
[43] A. Grossi *et al*., "Fundamental variability limits of filament-based RRAM," IEEE International Electron Devices Meeting (IEDM) , pp. 4.7.1-4.7.4, December 2016.
[44] M. Saberi, R. Lotfi, K. Mafinezhad, W. A. Serdijn, "Analysis of power consumption and linearity in capacitive digital-to-analog converters used in successive approximation ADCs," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 58, no. 8, pp. 1736-1748, Augest 2011.
[45] L. Kull *et al*., "A 3.1 mW 8b 1.2 GS/s single-channel asynchronous SAR ADC with alternate comparators for enhanced speed in 32 nm digital SOI CMOS," Journal of Solid-State Circuits, vol. 48, no. 12, pp. 3049-3058, December 2013.

**Yanan Sun** received the B.E. degree in Microelectronics from Shanghai Jiao Tong University, Shanghai, China in 2009, and the Ph.D. degree from the Hong Kong University of Science and Technology, Hong Kong in 2015. She is currently an Associate Professor with the Department of Micro-Nano Electronics at Shanghai Jiao Tong University, China. Her research area is low-power, high-performance, and variation-tolerant VLSI circuit and system design with emerging nanoscale device technologies. Dr. Sun received the best paper award nomination in 2020 IEEE Design, Automation, and Test in Europe Conference (DATE) and the best paper award (first place) in 2014 IEEE International Conference on Microelectronics (ICM). She currently serves on the editorial board of the Microelectronics Journal.

**Chang Ma** received the B.E. degree in the School of Physics and Technology from Wuhan University, Wuhan, in 2018. He is currently pursuing the M.S. degree in the School of Electronic, Information and Electrical Engineering, Shanghai Jiao Tong University. His research mainly focuses on non-volatile memory-based in-memory computing and high reliability ReRAM-based neuromorphic computing.

**Zhi Li** received the B.E. degree in the School of Electronic, Information and Electrical Engineering, Shanghai Jiao Tong University, in 2020. He is currently pursuing his M.E. degree in the School of Electronic, Information and Electrical Engineering, Shanghai Jiao Tong University. His research mainly focuses on image signal processing and computing in memory.

**Yilong Zhao** received the B.E. degree in the Department of Electronic Engineering from Shanghai Jiao Tong University in 2018. He is currently pursuing the M.Eng. degree in the Department of Computer Science and Engineering, Shanghai Jiao Tong University. His research mainly focuses on DNN accelerators and in-memory computing.

**Jiachen Jiang** received the B.E. degree in the Department of Electronics Science and Technology from Huazhong University of Science and Technology, Wuhan, in 2017, and the M.S. degree in the Department of Micro/Nano Electronics from Shanghai Jiaotong University, in 2020. His research area is 3D integration and high-yield memory circuit and system design with emerging nanotechnologies.

**Weikang Qian** is with the University of Michigan-Shanghai Jiao Tong University Joint Institute and MoE Key Laboratory of Artificial Intelligence, Shanghai Jiao Tong University, Shanghai, China. He received his Ph.D. degree in Electrical Engineering at the University of Minnesota and his B.Eng. degree in Automation at Tsinghua University. His main research interests include electronic design automation and digital design for emerging computing paradigms. His research works were nominated for the Best Paper Awards at International Conference on Computer-Aided Design (ICCAD), Design, Automation, and Test in Europe Conference (DATE), and International Workshop on Logic and Synthesis (IWLS).

**Rui Yang** received the B.E. degree from Tianjin University, Tianjin, China, in 2011, and the Ph.D. degree from Department of Electrical Engineering and Computer Science from Case Western Reserve University, Ohio, USA, in 2016. He then joined Department of Electrical Engineering at Stanford University as a postdoctoral scholar, from 2016 to 2018. He is currently a tenure-track Assistant Professor and Ph.D. advisor at the University of Michigan Shanghai Jiao Tong University Joint Institute in Shanghai Jiao Tong University. His research area includes nanoelectronic devices such as nanoelectromechanical systems and nonvolatile memories for energy-efficient sensing and computing.

**ZheZhi He** received the B.S. degree in information engineering from Southeast University, Nanjing, China, in 2012, and the M.E. degree in electrical and computer engineering from Oregon State University, Corvallis, OR, USA, in 2015. After that, he received the Ph.D. degree in electrical engineering from Arizona State University, Tempe, AZ, USA, in 2020.

He is currently an assistant professor with the department of computer science and engineering, Shanghai Jiao Tong University (SJTU), Shanghai, China. He has authored and coauthored over 50 peer-reviewed international journal articles/conference papers, such as DAC, ICCAD, DATE, TCAD, CVPR, ICCV, AAAI, MobiCom, etc. His primary research interests include neuromorphic computing, efficient and secure deep learning, in-memory computing, electronic design automation. He served as the technical program committee member of DAC, GLSVLSI. He is the receipt of the Best Paper Award of ISVLSI 2018 and ISVLSI 2017. He also served as the technical reviewer for over 30 international journals /conferences, such as MICRO, HPCA, DAC, ICCAD, DATE, TC, TNNLS, TCAS-I, NeurIPS, ECCV, CVPR, and etc.

**Li Jiang** is an associate professor from Dept. of CSE, Shanghai Jiao Tong University. He received the B.S. degree from the Dept. of CS&E, Shanghai Jiao Tong University in 2007, the MPhil and the Ph.D. degree from the Dept. of CS&E, the Chinese University of Hong Kong in 2010 and 2013 respectively. He has published more than 50 peer-reviewd papers in top-tier computer architecture and EDA conferences and journals, including a best paper nomination in ICCAD. He got best Ph.D. Dissertation award in ATS 2014, and was in the final list of TTTCs E. J. McCluskey Doctoral Thesis Award. He received ACM Shanghai Rising Star award, and CCF VLSI early career award. He is the 2020 Chinese Computer Federation (CCF) Distinguished Speaker. He serves as co-chair and TPC member in several international and national conferences, such as MICRO, DATE, ASP-DAC, ITC-Asia, ATS , CFTC, CTC and etc. He is an associate Editor of IET Computers Digital Techniques, VLSI the Integration Journal.