Design Guidelines of RRAM based Neural-Processing-Unit: A Joint Device-Circuit-Algorithm Analysis

Wenqiang Zhang[1], Xiaochen Peng[2], Huaqiang Wu[1]*, Bin Gao[1]*,
Hu He[1], Youhui Zhang[3], Shimeng Yu[2]*, and He Qian[1]
[1]Institute of Microelectronics, Tsinghua University, Beijing, China;
[2]Georgia Institute of Technology, Atlanta, GA, USA;
[3]Department of Computer Science and Technology, Tsinghua University, Beijing, China;
*Email: wuhq@tsinghua.edu.cn; gaob1@tsinghua.edu.cn; shimeng.yu@ece.gatech.edu

ABSTRACT

RRAM based neural-processing-unit (NPU) is emerging for processing general purpose machine intelligence algorithms with ultra-high energy efficiency, while the imperfections of the analog devices and cross-point arrays make the practical application more complicated. In order to improve accuracy and robustness of the NPU, device-circuit-algorithm codesign with consideration of underlying device and array characteristics should outperform the optimization of individual device or algorithm. In this work, we provide a joint device-circuit-algorithm analysis and propose the corresponding design guidelines. Key innovations include: 1) An end-to-end simulator for RRAM NPU is developed with an integrated framework from device to algorithm. 2) The complete design of circuit and architecture for RRAM NPU is provided to make the analysis much close to the real prototype. 3) A large-scale neural network as well as other general-purpose networks are processed for the study of device-circuit interaction. 4) Accuracy loss from non-idealities of RRAM, such as I-V nonlinearity, noises of analog resistance levels, voltage-drop for interconnect, ADC/DAC precision, are evaluated for the NPU design.

1 Introduction

Deep neural network (DNN) has produced extremely promising results in recent years and has been shown to outperform many other machine learning techniques in image classification and generation, video tracking, game playing, natural language processing [1-3]. However, with the increased fabrication cost and fundamental physical limits, device scaling alone can no longer close the desired performance gap between the state-of-the-art CMOS-based chip and the demand for future edge inference of DNN [4,5]. Besides, the fundamental limitation of modern computers lies in physical separation of computation and memory, which constrains the data exchange between the processing unit and the memory unit, known as "memory wall" [6, 7]. Consequently, in order to further improve energy efficiency, computing-in-memory architecture with emerged non-volatile memory has gained widespread attentions [8, 9].

DAC '19, June 2–6, 2019, Las Vegas, NV, USA

ACM ISBN 978-1-4503-6725-7/19/06…$15.00
https://doi.org/10.1145/3316781.3317797

Recently, Resistive Random-Access Memory (RRAM) has demonstrated great progresses on accelerating some neural networks with specific topology efficiently [10-12]. However, to reconfigure the hardware accelerator for versatile network topologies, it is highly attractive to develop a RRAM based neural-processing-unit (NPU) for general purpose deep neural network algorithms. For the NPU design, the desired characteristics of the RRAM devices is slightly different from the previous analysis for a specific neural network design, owing to the requirements of analog input and reconfigurable RRAM arrays [13, 14]. Meanwhile, simple analysis on a fully-connected neural network with small dataset such as MNIST, cannot well reveal the influence of some non-ideal effects. To our best knowledge, up to now, there is still no prior work on analyzing the device-circuit interaction of scalable and reconfigurable RRAM NPU towards a large-scale system. Since general purpose NPU is much more complex than a specific neural network accelerator, it is highly required to develop a methodology to evaluate RRAM NPU with the measured characteristics from the devices, and provide design guidelines.

In this work, a comprehensive analysis on RRAM NPU is proposed. The contributions of this paper include:

- An end-to-end (from device to array, circuit, and to algorithm) simulator is presented.
- The simulator includes the developed optimizations on the algorithms for RRAM NPU, and the complete peripheral circuits of NPU.
- To guide the RRAM macro design, the optimization of RRAM array is studied and discussed.
- Based on the measured results from the fabricated analog RRAM array, we clarify the impact of some of the non-ideal effects on the RRAM NPU, and provide further optimization strategies.

The rest of this paper is organized as follows: Section 2 introduces the related background and the motivation of our work; Section 3 proposes the RRAM-based NPU architecture, especially the full-stack design; Section 4 introduces the simulator design; Section 5 and 6 present two case studies of middle- and large- scale convolutional neural networks on MNIST, CIFAR10 and ImageNet datasets to analyze non-ideal impact on classification accuracy, area and latency; and Section 7 concludes the whole paper.

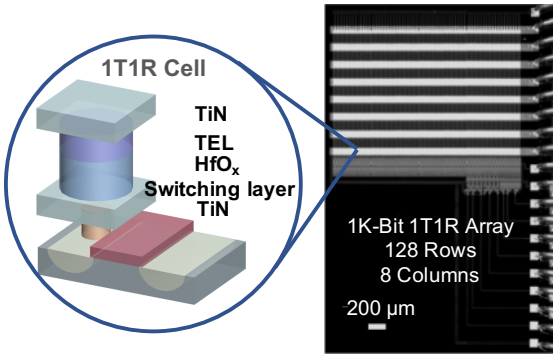

Figure 1: The schematic of 1T1R RRAM cell with TiN/TEL /HfOx /TiN structure and 128×8 array.

2 Preliminaries and Motivation

2.1 RRAM device and array:

The fabricated array with 1024 1T1R cells is shown in Fig. 1. Each 1T1R cell consists of one transistor to determine the selection and one two-terminal resistive element with variable resistance states. To realize better analog behavior, RRAM elements employ TiN/TEL/HfOx/TiN stack. Unlike the conventional HfOx RRAM, the increased local temperature can help to form multiple conductive filaments due to the lower thermal conductivity of thermal enhanced layer (TEL). The I-V nonlinearity and noises of analog resistance levels are measured from single RRAM cell and 1kb RRAM array respectively, which are modeled in the simulator (discussed in Section 6).

2.1 Deep Neural Network with RRAM array

DNN is structured as the cascaded convolutional layers and full-connected layers that run sequentially. For instance, the output results of previous layer are sent to the input of next layer. There are two compute-intensive operations in DNN: convolution (CONV) and vector-matrix multiplication (VMM). Fig. 2 demonstrates that the RRAM arrays are employed to complete the above two types of operations. For CONV operation, multiple 3D kernels (C, K, K, N) are used to extract features from the input feature maps (FMs). The output FMs are generated by sliding the 3D kernels on the input FMs. To use RRAM arrays perform CONV operations, the input FMs (C, H, W) and the kernels (C, K, K, N) are mapped to input voltage (H×W, C×K×K) and conductance (C×K×K, N) of array, respectively. Similarly, for VMM operation, the input vector (H) and the matrix (H, W) are mapped to input voltage (H) and conductance (H, W). When the size of RRAM array is smaller than that of kernels or weights, we split the kernels and weights into multiple small weight blocks vertically and transversely according to the size of array. Then, these weight blocks are mapped to the conductance of multiple RRAM arrays. When the voltages are applied to the input terminal of the array, the results of each array can be expressed as output currents:

$$\boldsymbol{I_n = \sum_{m=1}^{M} G_{mn} V_m, (n = 1, \ldots, N).}$$

Then, in order to accumulate the computing results of one layer and send results to next layer with digital circuits, the output currents are quantized by ADCs.

2.3 Motivation and Challenges

Due to the underlying non-ideal characteristics of RRAM devices and arrays, such as variability and voltage drop, codesign between

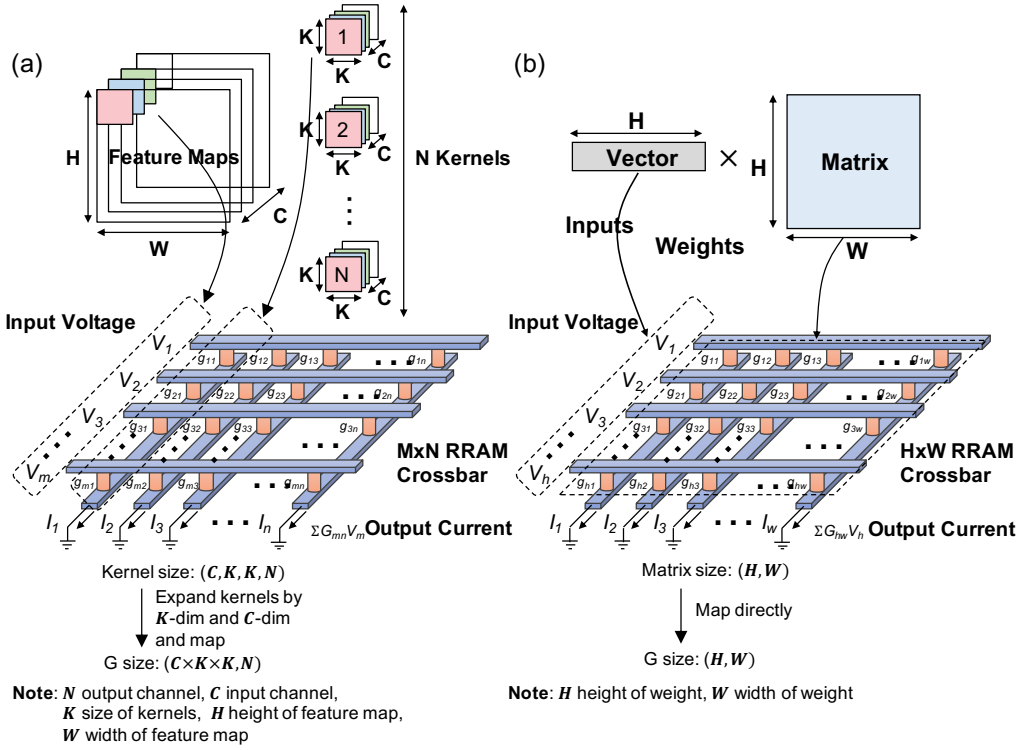

Figure 2: The mapping schedules of NPU. (a) CONV operation: the four-dimensional kernel is expanded by second K–dim and C-dim as two-dimensional matrix. The matrix is mapped to the conductance of RRAM crossbar. (b) VMM operation: the weight matrix is directly mapped to the conductance of RRAM crossbar.

the RRAM based NPU and algorithms is need to compensate or reduce these effects. Recent works have presented the architectural simulator platforms to support system-level design of deep neural network accelerators, but they have limited considerations of the device-level nonidealities [15, 16]. Other circuit-level non-volatile memory macro model simulators are lack of architectural design and optimization of algorithm [17, 18]. Then, to facilitate the exploration of design space of RRAM based NPU, a simulator for device-circuit-algorithm codesign is required. Besides, when the NPU scale increases, several challenges limit the performance of the design.

1) Peripheral Circuit Overhead: The estimation of the overhead of peripheral circuits, such as DACs, ADCs, driver circuits, is crucial to design a RRAM based NPU that is better than CMOS NPU in terms of the area efficiency and latency.

2) Non-idealities of Devices: The non-ideal effects of RRAM device and array, such as I-V nonlinearity, noises of analog resistance levels, and voltage-drop along interconnections, have impact on In. As illustrated in Fig. 3, when the DAC is 2-bit and ADC is 8-bit, 14-bit outputs are generated by Shift & Adder. Due to the device-level non-idealities, the low significance bits of ADC output are filled with variations. Thus, the 14-bit results may be inaccurate owing to the shifting of low inaccurate bits, and it may have uncertain impact on the performance of NPU.

Thus, we developed an end-to-end simulator from the training framework to the backend simulator that supports device-circuit-algorithm codesign. Further, as two case studies, the peripheral circuit overhead and non-idealities of devices are evaluated with the developed simulator.

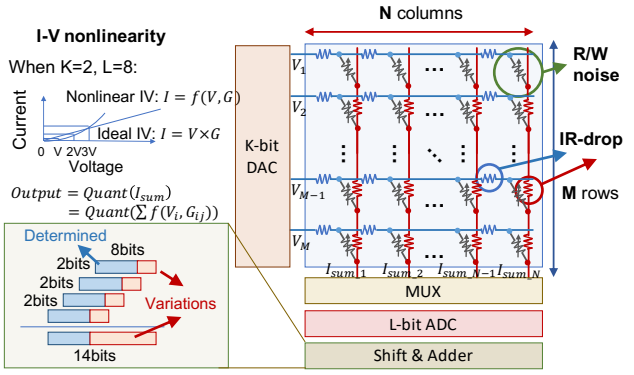

Figure 3: Schematic of non-idealities of RRAM discussed in this work. When the precision of DAC is less than that of input, the input is divided into several parts and the ADC outputs of these parts are shifted and added.

3 RRAM based NPU Design

3.1 The Hierarchical Architecture of NPU

To develop a simulator for RRAM based NPU, a baseline architecture is required. As shown in Fig. 4a, the proposed high-level architecture of a general-purpose scalable RRAM NPU includes a number of tiles, on-chip interconnection fabric and Input/Output (I/O) modules. The tile fetches input data from I/O modules or other tiles and performs calculation between stored kernels/weights and input data. The on-chip interconnection fabric is used to send or receive intermediate data between two arbitrary tiles (e.g. send the output FMs to the next layer). The I/O modules can receive input images from off-chip and send the results to off-chip.

Besides, we also developed the architecture of integrated software stack for the RRAM NPU. The top of stack is deep learning frameworks, such as PyTorch or Theano. The framework integrates some features of RRAM device like variation of conductance during the training process of deep neural network (DNN). The trained model will be transformed to memory-centric intermediate representation (IR) by frontend compiler, which is optimized for RRAM arrays of different sizes. The RRAM based NPU is a spatial hardware, which means the weights stay in the chip persistently during the inference phase and the capacity of on-chip RRAM devices should be larger than that of DNN model. Otherwise, a DNN model should be mapped to several chips. Then, the memory-centric IR is sent to NPU and the NPU starts to map the kernels/weights to RRAM arrays.

3.2 Circuit Modules

Each Tile in the RRAM based NPU is the same to reduce the design difficulties and the effects of technology variations. A Tile consists of several RRAM crossbars with corresponding periphery circuits (*XBs*), a special function unit (*SFU*) with buffers (Fig. 4b) and a Tile-level controller.

a) XB: XB is the basic computing element of NPU. In the configuration phase, the write and driver circuits are used to change the conductance of RRAM devices to the target. Compared to the memory-specified RRAM, the computing RRAM has a stricter requirement of device conductance. Then, a verify scheme should be supported by the *Controller*. In the computing phase, the input data of XB from *Buffers* is converted to analog voltage with DACs. When the *Controller* sends the start signal, the analog voltages are applied to one edge of RRAM array and the output current at another edge is quantized by ADCs. In order to reduce the area and

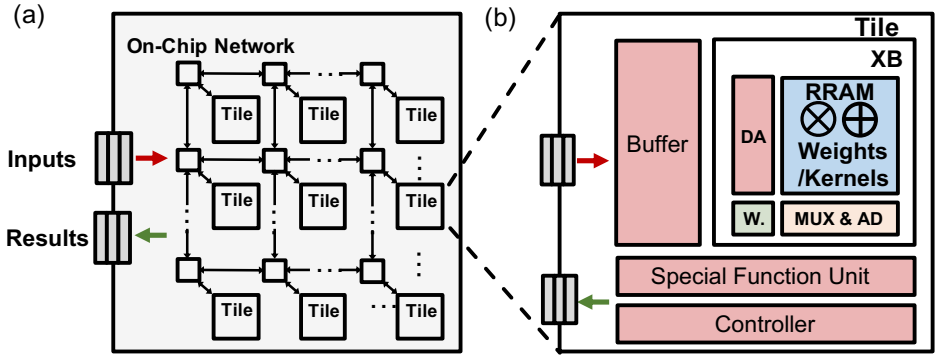

Figure 4: The hierarchical architecture of RRAM-based NPU. (a) the top interconnect architecture of NPU and its input and output. (b) the microarchitecture of Tile and the modules in XB.

power dissipation of ADCs, the quantization processes of output currents are temporally multiplexed.

b) Special Function Unit (SFU): The *SFU* is a vector unit, which has two functions: (1) when the size of a convolution layer or full-connected layer is larger than that of RRAM array, the kernels or weights are split and mapped to multiple *XBs*. Then, the *SFU* sums the results of *XB*s; (2) the *SFU* performs activation function such as Sigmoid, Tanh or Rectified Linear Unit (ReLU).

c) Buffers: For CONV operations, to reuse the input FMs, a structure of line buffer is employed for intermediate data buffering and fetching in the design. For instance, when the height of kernel is K, K-1 lines of data in FMs are buffered and wait for the next pixels in following line. Once the data for CONV operation is enough, the XB start computing. For VMM operation, the regular buffers are used.

d) Controller: The Controller has two phases: (1) configuration phase. When the structure or weights/kernels of DNN model have been change, on-chip RRAM devices and configuration registers, such as the routing path, are initialized. (2) computing phase. The data-driven controller determinates the tile-level pipeline by monitoring the buffer. When the buffers have sufficient data, the controller triggers the *XB* or *SFU* to perform computing. After the results is generated, the *Controller* sends the output data to the next stage according the dataflow topology.

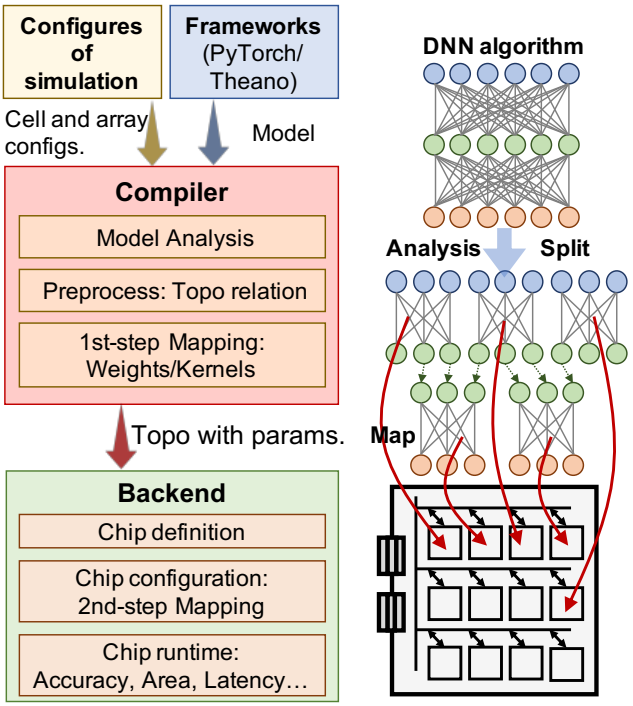

Figure 5: Left: The architecture of simulator from training framework to RRAM NPU-backend simulation, Right: Schematic workflow of simulator from DNN algorithm to the NPU chip.

4 End-to-end simulator for RRAM NPU

To explore the design space of RRAM NPU, we developed an end-to-end simulator. The features of the simulator include: support end-to-end training and evaluation of classification accuracy; support different scale neural network from MLP to VGG [19]; support traced evaluation of RRAM NPU, including area, latency and power. The simulator consists of training framework, memory-centric compiler (Fig. 5), and NPU-backend evaluation tools.

a) Training framework: The training framework is used to optimized the kernels and weights in the DNN model which is defined in the input configurations. Rather than hiding the non-ideal effects of RRAM from top application, the training framework integrates some device variations in the training process to improve the on-chip accuracy and robustness. When the training phase of neural network algorithm is finished, the DNN model and optimized kernels/weights are packed and sent to the compiler and mapper.

b) Memory-centric compiler and mapper: The input of compiler includes the above parameters of trained DNN model and the input configurations of RRAM cell and array. Then, the compiler analyzes the model automatically and splits the model into the crossbar-level weights to fit the *XB*-level constraints, such as the array size and computing unit cell configurations. The unused RRAM devices are configured to the high resistance to reduce the computing power and improve the computing accuracy. The mapper decides the placement and routing paths of Tiles. The dataflow topology of Tiles and corresponding conductance targets are sent to the backend simulator.

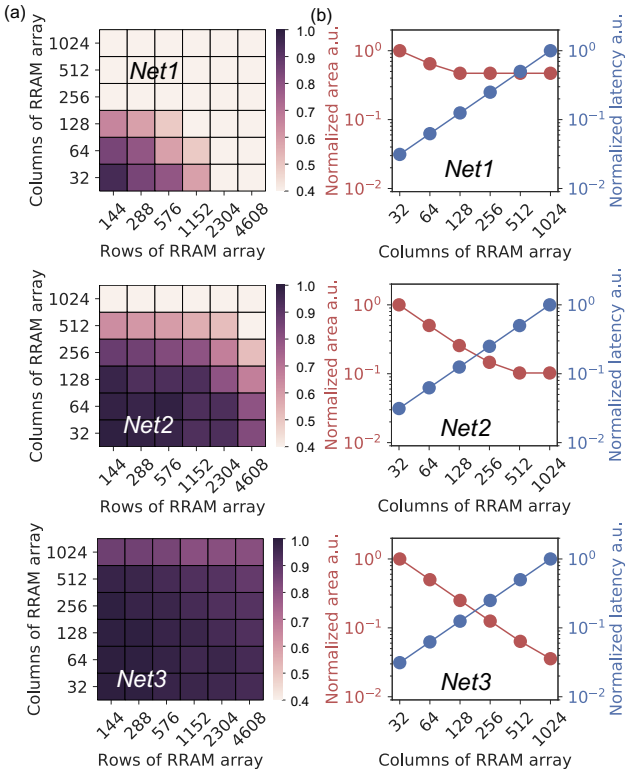

Figure 6: (a) Utilization ratio (used cells/all cells) in NPU when mapping models to arrays of different sizes. (b) Normalized area and latency of NPU when the row amount of RRAM array is 1152.

c) Backend simulator: The backend simulator is used to evaluate the area, latency, power and leakage energy of the RRAM based NPU and simulate on-chip computing accuracy in the

inference phase. The estimation tool includes the complete design of NPU architecture and circuits, which is divided as two parts: accuracy evaluation tool and *XB*-level circuit evaluation tool. During the accuracy evaluation, the measured non-idealities of RRAM device in the experiments are added into the calculation of output results of *XB*. To accelerate the inference phase with non-idealities of the simulation, a compact model of RRAM is employed. In the *XB*-level circuit evaluation, the *XB* circuit design includes the following modules: a crossbar, driver circuits, ADCs, DACs, multiplexes and shift-adders. The backend simulator has a trace mode to evaluate the breakdown of area/power/latency/leakage of different circuit modules.

5 Case Study: Optimization of RRAM Macro

The RRAM array size influences the performance of NPU a lot. Here, we evaluate different RRAM macro configurations with the simulator to determinate which size of RRAM array is best for NPU. The dataset and DNN architectures are listed in Table I (Net1: MNIST; Net2: CIFAR-10; Net3: IMAGENET). The configurations of simulation are listed in Table II. To reduce the device variations and represent the negative weights, we employed the differential 2T2R computing unit configuration. As the scale of neural network increases and the size of RRAM array decreases, the utilization ratio of RRAM cell increases (Fig. 6a). However, when the size of RRAM array decreases, the area of chip increases rapidly, but the latency decreases (Fig. 6b). Thus, to make a deal among utilization ratio, area efficiency and latency, the array size should not be larger than 1152×256.

6 Case Study: Non-idealities of Device and Array

When the NPU performs inference tasks, the non-idealities of devices and array have significant impact on the accuracy. Here, based on the measured results, we evaluate the influences of I-V nonlinearity, read & write noises and voltage-drop effects on the accuracy. Since Net2 and Net3 are quite similar, the non-ideal impact on accuracy are only evaluated on Net1 and Net2.

Table I Summary of dataset and DNN architecture

Dataset	Input Size	DNN Architectures
MNIST	28x28x1	**Net1:** 16-16-'M'-32-32-'M'-128-10
CIFAR10	32x32x3	**Net2:** 64-64-'M'-128-128-'M'-256-256-'M'-512-512-'M'-128-10
IMAGENET	224x224x3	**Net3:** VGG-16

Red: Output Channel of Conv layer Green: Output size of FC layer 'M': Maxpooling layer

Table II Summary of parameters in simulation

Parameter	Configuration	Parameter	Configuration
RRAM	2T2R ~ 1 weight	AD precision	4 ~ 10 bit
Range of res.	50K~500K ohm	XB #/Tile	1
Input precision	8 bit	S&H #/XB	8
DA precision	2 bit	AD #/XB	4

6.1 Nonlinearity of I-V

Compared with the weight update nonlinearity, I-V nonlinearity is usually ignored in prior works. However, as mentioned in Section 2.3, it significantly affects the computing accuracy of NPU when using analog DAC input, since the stored weights may change with

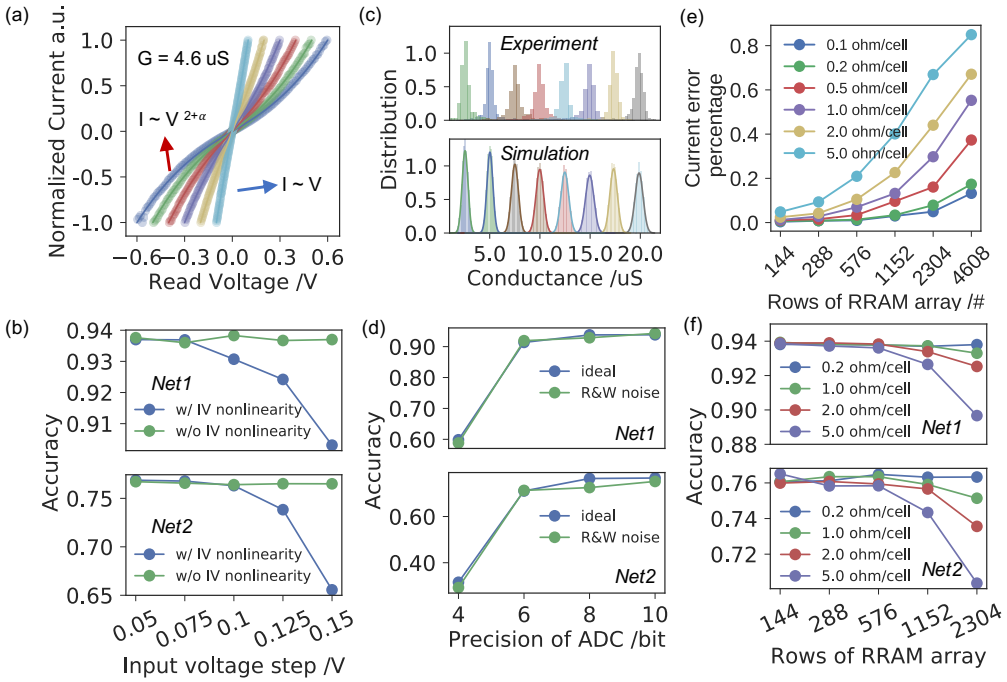

Figure 7: (a) Measured data and the fitting curves of I-V when the ranges of read voltage are varied from -0.6 V to 0.6 V. (c) The distributions of conductance with read and write noises at each level. Top: the measured data. Bottom: simulated results of compact model. (e) Simulated results of output current error when the column of RRAM array is 128. (b) (d) (f) Impact of I-V nonlinearity, combined noise and voltage-drop on the classification accuracy. Top: Net1. Bottom: Net2.

different voltage inputs. Fig. 7a shows the I-V curves when the conductance of device is 4.6 uS (measured read voltage @ 0.15V). When the voltage is small, the current is approximately linear proportional to voltage. As the voltage grows larger and the conductance becomes smaller, the I-V nonlinearity increases. The relation between current and voltage can be fitted as: $I=aV+bV^{(2+\alpha)}$. As shown in Fig. 7b, for both of Net1 and Net2, as the input voltage step becomes greater than 0.1V, the accuracy loss increases.

6.2 Read and write noises of resistance

Based on the measured results, a compact model with the combined noises (read and write noises) is shown in Fig. 7c. The read and write noise of eight analog levels only reduces the accuracy by about 1% (Fig. 7d), indicating that noise does not impact so much on NPU. This can attribute to that the read and write noises of RRAM have small overlap between the distributions in the adjacent conductance states. However, the precision of ADC should be larger than 6-bit to avoid a significant accuracy drop.

6.3 Voltage-drop effect in arrays with different sizes

Based on the parasitic parameter extraction with layout in post-simulation, the resistance of interconnect in 130nm 1kb array is around 0.2 ohm/cell. When the RRAM devices scale down into a more advance technology node, the influence of interconnect resistance will be more serious. As the row amount of array and the interconnect resistance increase, the output currents of array show nonlinear distortion, especially when the array has 4608 rows and 5.0 ohm/cell (Fig. 7e). The interconnect resistance should be smaller than 1 ohm/cell to obtain a small accuracy loss (Fig. 7f). Further, to mitigate the voltage-drop effect and pull the accuracy up, a compensation scheme by retrained the dominant neurons can be adopted in the RRAM macro design [20]. In summary, as design guidelines for RRAM NPU, a summary of recommended parameters are listed in Table III.

Table III Summary of optimized parameters in RRAM NPU

RRAM based NPU Design Guidelines	
Parameter	**Recommend**
Range of Res.	50K~500K ohm
Input voltage	<= 0.1 V/step
Interconnect	<= 1 ohm/cell
I/O precision	>= 6 bit
DAC precision	2bit
ADC precision	>= 6 bit
Rows of Array	576 or 1152
Cols of Array	128 or 256

7 Conclusion

In this paper, we proposed a comprehensive device-circuit-algorithm joint analysis on RRAM NPU. Key achievements: 1) Design guidelines of RRAM NPU are provided based on the measured characteristics of analog RRAM array and device-circuit joint analysis. 2) General purpose neural networks, including VGG-16 are demonstrated on the developed end-to-end RRAM NPU simulator. 3) I-V nonlinearity, ADC/DAC precision, large interconnect resistance is found to have noticeable impact on computing accuracy loss, while current noises and low interconnect resistance are found to have much less impact.

ACKNOWLEDGMENTS

This work is supported in part by the MOST of China (2016YFA0201801), Beijing Innovation Center for Future Chips (ICFC), Beijing Municipal Science and Technology Project (D161100001716002, Z181100003218001), HUAWEI Project (YBN2018025563) and NSFC (61674087, 61674089, 61674092, 61076115)

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, pp. 436–444, 2015.
[2] V. Mnih et al., “Human-level control through deep reinforcement learning,” Nature, vol. 518, no. 7540, pp. 529–533, 2015.
[3] D. Silver et al., “Mastering the game of Go without human knowledge,” Nature, vol. 550, no. 7676, pp. 354–359, 2017.
[4] S. Salahuddin, K. Ni, and S. Datta, “The era of hyper-scaling in electronics,” Nature Electronics, vol. 1, no. 8, pp. 442–450, 2018.
[5] M. A. Zidan, J. P. Strachan, and W. D. Lu, “The future of electronics based on memristive systems,” Nature Electronics, vol. 1, no. 1, pp. 22–29, 2018.
[6] W. Wulf and S. A. McKee, “Hitting the Memory Wall: Implications of the Obvious,” University of Virginia, 1994.
[7] M. Horowitz, “Computing’s energy problem (and what we can do about it),” in International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014, pp. 10–14.
[8] D. Ielmini and H.-S. P. Wong, “In-memory computing with resistive switching devices,” Nature Electronics, vol. 1, no. 6, pp. 333–343, 2018.
[9] W. Haensch, T. Gokmen, and R. Puri, “The Next Generation of Deep Learning Hardware: Analog Computing,” Proceedings of the IEEE, pp. 1–15, 2018.
[10] S. Ambrogio et al., “Equivalent-accuracy accelerated neural-network training using analogue memory,” Nature, vol. 558, no. 7708, pp. 60–67, 2018.
[11] C. Li et al., “Efficient and self-adaptive in-situ learning in multilayer memristor neural networks,” Nature Communications, vol. 9, no. 1, 2018.
[12] P. Yao et al., “Face classification using electronic synapses,” Nature Communications, vol. 8, p. 15199, 2017.
[13] H. Wu et al., “Device and circuit optimization of RRAM for neuromorphic computing,” in International Electron Devices Meeting (IEDM), 2017, pp. 11.5.1-11.5.4.
[14] Y. Ji, Y. Zhang, W. Chen, and Y. Xie, “Bridge the Gap Between Neural Networks and Neuromorphic Hardware with a Neural Network Compiler,” in International Conference on Architectural Support for Programming Languages and Operating Systems, 2018, pp. 448–460.
[15] P. Chi et al., “PRIME: A Novel Processing-in-Memory Architecture for Neural Network Computation in ReRAM-Based Main Memory,” in International Symposium on Computer Architecture (ISCA), 2016, pp. 27–39.
[16] A. Shafiee et al., “ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars,” in 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), Seoul, South Korea, 2016, pp. 14–26.
[17] L. Xia et al., “MNSIM: Simulation platform for memristor-based neuromorphic computing system,” in Design, Automation Test in Europe Conference Exhibition (DATE), 2016, pp. 469–474.
[18] P.-Y. Chen, X. Peng, and S. Yu, “NeuroSim: A Circuit-Level Macro Model for Benchmarking Neuro-Inspired Architectures in Online Learning,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, pp. 1–1, 2018.
[19] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” arXiv:1409.1556, 2014.
[20] Y. Jeong, M. A. Zidan, and W. D. Lu, “Parasitic Effect Analysis in Memristor-Array-Based Neuromorphic Systems,” IEEE Transactions on Nanotechnology, vol. 17, no. 1, pp. 184–193, 2018.