MINT: Mixed-precision RRAM-based IN-memory Training Architecture

Hongwu Jiang, Shanshi Huang, Xiaochen Peng and Shimeng Yu

School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA

Email: shimeng.yu@ece.gatech.edu

***Abstract*— On-chip training of large-scale deep neural networks (DNNs) is challenging. To solve the memory wall problem, compute-in-memory (CIM) is a promising approach that exploits the analog computation inside the memory array to speed up the vector-matrix multiplication (VMM). Challenges for on-chip CIM training include higher weight precision and higher analog-to-digital converter (ADC) resolution. In this work, we propose a mixed-precision RRAM-based CIM architecture that overcomes these challenges and supports on-chip training. In particular, we split the multi-bit weight into the most significant bits (MSBs) and the least significant bits (LSBs). The forward and backward propagations are performed with CIM transposable arrays for MSBs only, while the weight update is performed in regular memory arrays that store LSBs. Impact of ADC resolution on training accuracy is analyzed. We explore the training performance of a convolutional VGG-like network on the CIFAR-10 dataset using this *M*ixed-precision *IN*-memory *T*raining architecture, namely *MINT,* showing that it can achieve ~91% accuracy under hardware constraints and ~4.46TOPS/W energy efficiency. Compared with the baseline CIM architectures based on RRAM, it can achieve 1.35× higher energy efficiency and only 31.9% chip size (~98.86 mm^2 at 32 nm node).**

Keywords— RRAM, compute-in-memory, deep neural network, hardware accelerator

I. INTRODUCTION

As DNNs become deeper and more complicated, number of operations and parameters also increase significantly. For example, as a representative DNN, VGG-16 [1] has 138MB parameters using 8-bit precision for weights and activations. As a result, data movements between the computing units and memory units become the bottleneck. Especially, expensive off-chip DRAM access occurs frequently due to the limit on-chip SRAM buffer size for DNN workloads. There have been many research efforts on the design of application specific integrated circuit (ASIC) accelerators such as Eyeriss [2] and TPU [3], where the parameters are stored in global buffer and the computation is still performed at the digital multiply-accumulate (MAC) arrays. Compute-in-memory (CIM) is an efficient paradigm to address the memory wall problem in DNN hardware acceleration [4]. Convolution operation essentially contains vector-matrix multiplication (VMM), which takes up most of the computations in DNNs. The crossbar structure supports analog VMM operations by activating multiple rows and perform current summation along bit lines (BLs). Emerging non-volatile memory (eNVMs) such as phase change memory (PCM) [5] and resistive random-access memory (RRAM) [6] provide promising solutions to design CIM-based accelerator due to smaller cell size than SRAM at the same node. Though these eNVMs based CIM architectures are promising, grand challenges exist in designing a practical CIM accelerator that supports both training and inference. First, most of the CIM architectures proposed so far, such as PRIME [7] and ISAAC [8] could support the inference only. The data flow for CIM is largely unexplored. Second, the impact of ADC resolution on the training/inference accuracy is rarely explored. Third, the asymmetric and nonlinear conductance tuning introduces significant training accuracy loss [9], making it difficult to utilize the multilevel states of eNVMs for *in-situ* training.

In this paper, we propose a Mixed-precision RRAM-based IN-memory Training architecture, namely MINT, to overcome the aforementioned challenges and target at implementing efficient on-chip training of DNNs. Considering the practical non-ideal effects such as asymmetric conductance tuning, limited dynamic range, and variability/reliability for multilevel states, binary RRAM is used in this work. Multi-bit weight is thus implemented by multiple RRAM cells. The key idea is to split the multi-bit weight into two parts: the most significant bits (MSBs) and the least significant bits (LSBs). The forward and backward propagations are performed with a CIM transposable array for MSBs only, while the weight update is performed in a regular memory array that stores LSBs. We analyze the impact of ADC and RRAM's non-ideal effects on training performance, and evaluate hardware performance of our design.

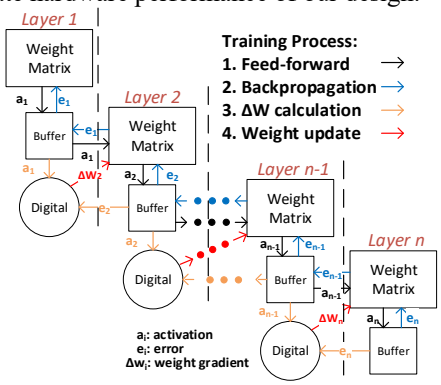

Fig. 1. Workflow of DNN training in CIM

II. WEIGHT MAPPING STRATEGY FOR TRAINING

Fig. 1 shows the simplified workflow of training process in CIM hardware, which could be divided into four steps, namely, 1) feed-forward (FF), 2) backpropagation (BP), 3) weight gradient (ΔW) calculation and 4) weight update. These four steps run in a loop through iterations. Here we consider the batch based training, thus steps 1) to 3) occur every input cycle within the batch but step 4) occurs only once at end of the batch. The FF process of the CIM array is shown in Fig. 2 (a). Here M is the

978-1-7281-3320-1/20/$31.00 ©2020 IEEE

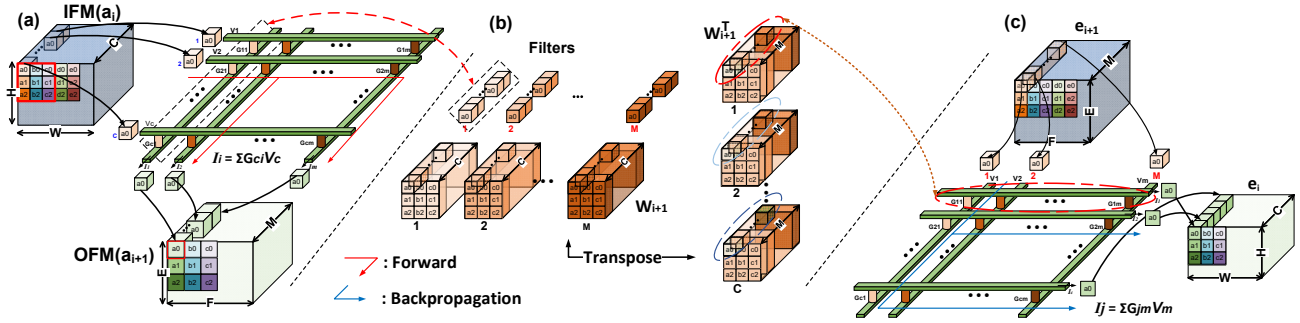

Fig. 2. Weight mapping scheme for convolution operations in CIM training

number of filters/output feature map (OFM) channels, C is number of input feature map (IFM)/filter channels, H/W is the IFM plane height/width, and E/F is the OFM plane height/width. The filters are unrolled to a weight matrix that is stored in the memory array. The IFM vector is applied to the row in parallel, and the OFM vector is obtained from the column in parallel. To maximize the input data reuse, the best weight mapping strategy is to use N×N processing elements (PEs) for mapping an N×N filter [10], and partial sums from all the PEs are accumulated outside PEs. Only one PE is shown in Fig. 2 (a) for the first channel group (a0 element), and 9 PEs are needed for 3×3 filter in one layer of DNNs. As shown in Fig. 2 (b), W_{i+1} is the weights of the first channel group (a0 element) in FF while W_{i+1}^T is the transposed weights for BP, and they are mapped to the same memory array as one PE. The backward pass for a convolution operation is also a convolution (but with spatially-flipped filters). For the FF, the products of input sliding window with the same filter across all the channels are summed up to generate one output, which means all the dot products in same column of the PE are summed up. However, for the BP, the products of input sliding window and the same channel across different filters are summed up, which means all the dot-products in same row are summed up. Essentially, we process the transposed version of the weight matrix in the BP. Fig. 2 (c) shows the details of error calculation for the first channel group (a0 element) of the filters. With such transpose array and weight mapping strategy, FF and BP can be performed within the same array.

III. HARDWARD IMPLEMENTATION

A. RRAM Subarray Design

In MINT architecture, RRAM subarray design is based on one-transistor-one-resistor (1T1R) pseudo-crossbar structure, which minimizes the sneak path and write disturbance. 1T1R array is preferred for embedded applications where the density is not the pursuit but the performance and the reliability are of the priority. Each cell only has binary on-state or off-state. According to the discrete training technique in WAGE framework [11], only MSBs of the weight are needed for FF and BP calculation (i.e., the first 2 bits out of an 8-bit weight). Then the weight update is performed with all the bits. To optimize such mixed-precision training in MINT, two types of subarrays are proposed: computing subarrays and storage subarrays. As shown in Fig. 3 (a), computing subarray is dedicated to the MSBs of the weight. It is used for transpose convolution operation supporting bi-directional read access. Each bit of inputs is fed into the array by cycles. In FF process, all the transistors will be transparent when all wordlines (WLs) are turned on in deep triode region. Thus, the input vector voltages are provided to the BLs, and the weighted sum currents are read out through sourcelines (SLs) in parallel. For BP process, bitlines (BLs) and SLs just need to exchange their roles by a Mode Mux, which means the partial sums are read out from parallel BLs. The analog values of current along BLs/SLs, which represent the VMM results, will be quantized by ADCs. Shift-adders are applied to accumulate partial sum results from different significant bits of the input. Computing arrays keep active during FF and BP operations. Storage subarray is shown in Fig. 3 (b), which is the same as conventional RRAM array for row-by-row read-out. Therefore, the area/energy expensive ADCs are eliminated in storage subarray. LSBs of weights are saved in storage subarray and only activated while updating new weights at end of a batch. The LSBs are read-out and added with the weight gradient, and then written back to storage subarray.

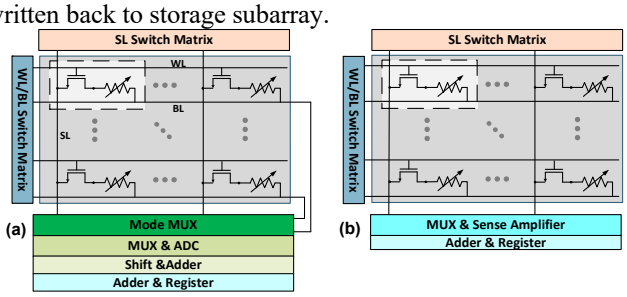

Fig. 3. RRAM subarray design: (a) computing subarray with expensive ADC (b) storage subarray with simple SA

B. Overall Architecture

MINT is implemented for an 8-bit VGG-8 network for CIFAR-10 dataset. Fig. 4 shows the top-level architecture for one convolution layer. To enable the bi-directional access, weight bits with different significance are stored on different tiles with shift-add to combine them together after obtaining their partial sums. In this example, we only use first two MSBs of the weight for convolution operation in FF and BP, corresponding to computing subarrays in Tile[1] and Tile[2]. Tile[3-8] consist of regular storage arrays for the other 6 LSBs of the weight. Tile[1-8] will be all used during weight update, though most of the weight update are incremental thus probably only storage subarrays are rewritten. Other peripheral modules such as pooling, activation, quantization, multiplier & adder tree and find max are included in the analysis. The typical weight matrix

size of one layer in DNNs could be several hundreds up to thousands, thus we partition the filters into subarrays and limit the subarray size to 128×128 considering the practical limits (e.g. IR drop and maximum current for the column). Hence, 16 subarrays are formed into one PE and 9 PEs (corresponding to 3×3 filters) are formed into one tile, which means most layers in this network can fit into 8 tiles for 8-bit weight (gradient) precision. Computing tile [1-2] contains multiple PEs, adder trees and L1 buffer. The red path shows the forward input feature maps direction while the green path denotes the transposed backward input error maps direction. In both FF and BP, subarray first completes shift-add calculation for multi-bit inputs cycle by cycle and then accumulate results for all the subarrays in one PE for one channel of the filter. Accumulated results from PEs in the same tile accumulate again through adder trees outside PE to process the entire filter, but only for single significance of the weight. Outside tiles, shift-add will be performed again for all the computing tiles to obtain the eventual desired output of OFM with appropriate bit significance. To minimize the on-chip buffer capacity, in FF/BP, activation/error outputs of each layer will be sent to off-chip DRAM for reuse in gradient calculation. Digital MAC computation for gradient calculation is performed by digital 8-bit multiplier and 10-stage adder tree. After the weight gradient calculation for each image, ΔW gradient will be stored off-chip DRAM. In the weight update at end of the batch, the saved gradients are fetched from DRAM, accumulated across batch, normalized by its maximum value and then go through stochastic quantization to obtain the final weight gradient ΔW.

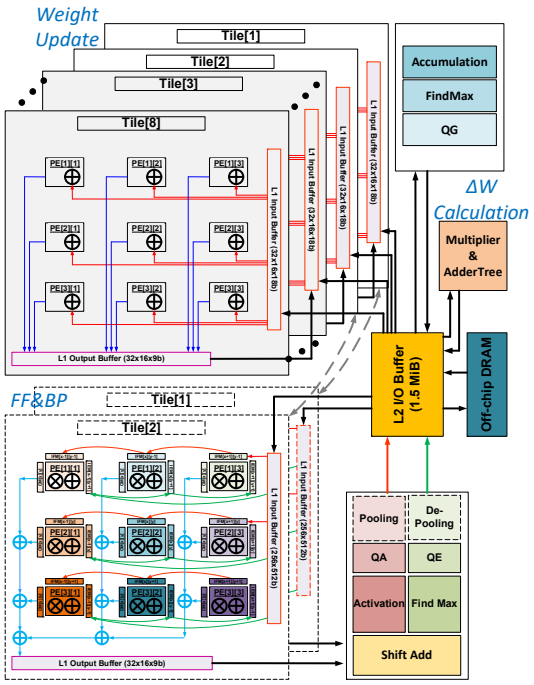

Figure 4. MINT overall architecture

IV. Evluations and Discussions

A. Impact of ADC range and resolution

ADCs in CIM impose huge overhead since their hardware cost in area and energy. The cost of some design (e.g. Flash ADC) may even grows exponentially as the ADC resolution increases. The impact of ADCs on inference performance has been discussed in prior works [12]. However, to our best knowledge, the ADC quantization impact on training performance is rarely explored so far. For XNOR-Net inference on CIFAR-10, 3-bit ADC is sufficient for 128×128 array [13]. We modified the convolution step in WAGE to include ADC quantization to partial sum from each subarray in the network. In our evaluations using PyTorch platform, we found that not only the resolution of ADC but also the full range of ADC will affect the training accuracy. According to distribution of the partial sum, the required range of ADC is less than the full precision for a certain subarray size. For example, in MINT architecture, we have 128×128 subarray, which means a 7-bit full precision for the column output in theory. However, in practice, the required full precision of partial sum is only 5-bit as most of the partial sum ranges from 0 to 32. Hence, in our case, 5-bit ADC means no accuracy loss and it could be the same accuracy ~91% as the software baseline. Then we further analyze the ADC resolution and range effects. The training performance under different ADC configurations is shown in Fig. 5. (a). We can see that generally CIM is very sensitive to the ADC quantization. First, we use 8-bit weight without MSB/LSB splitting. With the same ADC range (32) but 4-bit ADC (green line) will degrade accuracy from 5-bit ADC's 91% to 85%. ADC range (32) means ADC will only quantize sum value in (0, 32) range and any value larger than 32 will be treated as 32. Second, we use 2-bit MSB and 6-bit LSB splitting, we see that ADC range (32) and 5-bit ADC (black line) could still achieve the same 91% accuracy, however, reducing the ADC range to 16 and 4-bit resolution will not converge the training (red line). Cutting the range of ADC will have less effect than the reduced precision of ADC. In summary, 5-bit ADC is required to guarantee no accuracy loss in MINT design.

B. Non-idealities of RRAM

Cycling endurance may be a concern in write-dominant training. We count the number of weight updates during the training for our VGG-like network as shown in Fig. 5 (b). Each epoch includes 50K images, and the total epoch number is 85. However, weight update may only occur after one batch every 200 images. From the statistics, we can see 99% of RRAM cells flip less than 2000 times and the maximum write times is just 7,211. Today's binary RRAM technology could achieve more than 10^6 cycles [14], thus here endurance is not a concern.

C. Evaluation Setup

Our MINT architecture is built with a modified NeuroSim [15] framework. We evaluate the aforementioned VGG-8 network on CIFAR-10 dataset. As shown in Fig. 5 (a), we can split the 8-bit weight into 2-bit MSBs and 6-bit LSBs, while it could achieve a comparable accuracy. Therefore, for MINT design, we only use first 2-bit MSBs for FF/BP and they are stored in computing subarrays, and we use the rest 6-bit LSBs for weight update only and they are stored in storage subarrays. 5-bit ADCs are included in computing subarray according to the analysis of ADC. To improvement the throughput and energy efficiency, we use inter-layer parallelism scheme which means that each layer is a pipeline stage for both FF and BP

computation. Table. I shows the chip-level parameters including the hardware configuration, precision, area and energy for key circuit modules. The energy is given in energy per operation (or bit). We design two custom CIM-based architectures as the baselines: Baseline 1) without transpose subarray design and MSB/LSB splitting; Baseline 2) with transpose subarray but without MSB/LSB splitting; Other modules including digital logic for ΔW calculation and weight update are the same for MINT and baseline designs. We model the MINT and baseline designs in 32nm node, which is a practical node considering TSMC's 40nm RRAM [16] and Intel's 22nm RRAM [17] processes. All the architectures are built with a modified NeuroSim model by adding essential hardware components for supporting training and considerations of on-chip SRAM buffer and off-chip DRAM access.

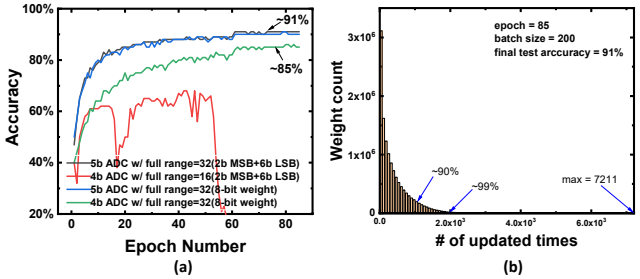

Fig. 5. (a). Training performance for different ADC quantization range and resolution. (b). Statistics of weight update frequency

D. Performance benchmarking

The area breakdown for FF, BP, ΔW calculation and weight update is shown in Fig. 6 (a). We see that area efficiency of MINT is much improved in FF/BP over that of baseline. This is because we optimize with the transpose design (reusing periphery) and MSB/LSB splitting (saving unnecessary ADCs). The total area is reduced to only 31.9% of the Baseline 1. For MINT architecture, the largest part of area overhead is actually the on-chip SRAM buffer, which taking 33.28% of total area. The throughput of MINT architecture is ~1,927 frame per second (FPS). Compared to the conventional GPU based training, the typical energy efficiency (e.g. NVidia Titan RTX) is ~0.06 TOPS/W. The TPU v3 (for training) energy efficiency is estimated to be approximately ~0.45TOPS/W [18]. MINT can achieve 4.46 TOPS/W. In Fig. 6 (b), we see that the energy saving of MINT in FF and BP process is 4× compared with Baseline 1&2 on average thanks to the mixed-precision training, as the ADCs are only used for the first 2-bit MSBs rather than the entire 8-bit weight. Digital blocks of ΔW calculation also contribute substantially to the total energy. For ΔW calculation, we use pure digital logic thus it consumes significant energy. In principle, ΔW calculation could also be implemented in CIM as it involves the convolution between errors and activations. For example, we could write the errors into another memory array, and load in the activations as the row input to the array, to generate the column output being ΔW. This is possible for SRAM based CIM [19], however, writing RRAM is expensive due to RRAM's large write energy/latency (than SRAM).

Table I: MINT Parameters

Main block	Spec.	Energy(pJ/op)	Area(mm^2)
Subarray Level			
RRAM array	Size: 128x128 Precision:1-bit	99.85	0.0003
MUX & Decoder		3.68	0.0013
ADC	Number: 32 Precision:5-bit	327.92	0.0019
Shift Add	Number: 32 Precision:12-bit	71.17	0.0009
WL/SL SwitchMatrix and other		15.74	0.0006
Subarray Total		518.36	0.0050
PE Level			
Subarray	Size: 4x4	8293.77	0.08
Adder Tree	Number: 64 Precision: 12-bit	54.77	0.05
L1 Buffer	Size: 128*128	0.05/bit	0.043
Output Buffer	Size: 32*54	0.01/bit	0.003
PE Total		8348.60	0.13
Tile Level			
PE	Size: 3x3	74643.93	1.17
Adder Tree	Number: 64 Precision: 16-bit	485.14	0.08
L2 Buffer	Size:256*512	0.10/bit	0.3
Output Buffer	Size: 16*32*9	0.014/bit	0.007
Tile Total		75129.10	1.60
Layer Level			
Shift Add	Number:64 Precision:17-bit	208.57	3.20
Chip Level			
ReLU+Find Max		45.40	0.006
Digital Gradient Calculation and Process		0.022	18.8
Global Buffer	Size:12*1024*1024	0.4/bit	32.90

DRAM access is also a significant part of the total energy since gradient calculation and weight update needs to access DRAM frequently. Considering our case of VGG-8, which has ~13MB weights, for batch size=200, there will be 200×13MB=2.6GB gradient calculated per batch. Due to the limited global buffer capacity, the gradients have to be sent to the DRAM after being calculated. Later they need to be loaded for accumulation, normalization and quantization.

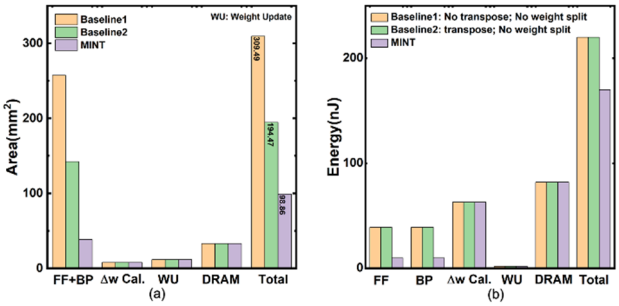

Fig. 6. (a). Breakdown of chip area. (b). Breakdown of energy consumption.

V. Conclusion

We propose a mixed-precision RRAM-based in-memory training architecture, namely MINT, which can maximize the hardware reuse with the transpose array design. Splitting MSB/LSB can also reduce hardware overhead (in particular ADCs). The evaluation results show that, MINT achieves the energy saving of 77× compared to GPU, 10× compared to TPU and 1.35× compared to the CIM architecture without optimizations. The area of MINT is only ~31.9% of the RRAM baseline designs.

Acknowledgment

This work is supported by NSF-CCF-1903951, NSF-CCF-1740225, and ASCENT, one of the SRC/DARPA JUMP Centers.

REFERENCES

[1] K. Simonyan, and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *International Conference on Learning Representations (ICLR),* 2014.

[2] Y.H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits, vol. 52, no. 1, pp. 127–138,* 2017.

[3] N.P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers and R. Boyle, "In-datacenter performance analysis of a tensor processing unit," *IEEE International Symposium on Computer Architecture (ISCA),* 2017, pp. 1-12.

[4] D. Ielmini and H.S.P. Wong, "In-memory computing with resistive switching devices," *Nature Electronics 1, no. 6 (2018): 333*.

[5] Giannopoulos, A. Sebastian, M.L. Gallo, V.P. Jonnalagadda, M. Sousa, M.N. Boon, and E. Eleftheriou, "8-bit precision in-memory multiplication with projected phase-change memory," *IEEE International Electron Devices Meeting (IEDM),* 2018, pp. 27-7.

[6] C.X. Xue, W.H. Chen, J.S. Liu, J.F. Li, W.Y. Lin, W.E. Lin, J.H. Wang, W.C. Wei, T.W. Chang, T.C. Chang and T.Y. Huang, "A 1Mb multibit ReRAM computing-in-memory macro with 14.6 ns parallel MAC computing time for CNN based AI edge processors," *IEEE International Solid-State Circuits Conference (ISSCC),* 2019, pp. 388-390.

[7] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang and Y. Xie, "PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," *IEEE International Symposium on Computer Architecture (ISCA),* 2016, pp. 27-39.

[8] Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J.P. Strachan, M. Hu, R.S Williams, and V. Srikumar, "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *IEEE International Symposium on Computer Architecture (ISCA),* 2016, pp. 14-26.

[9] G. W. Burr, P. Narayanan, R. M. Shelby, S. Sidler, I. Boybat, C. Di Nolfo and Y. Leblebici, "Large-scale neural networks implemented with non-volatile memory as the synaptic weight element: Comparative performance analysis (accuracy, speed, and power)," *IEEE International Electron Devices Meeting (IEDM),* 2015, pp. 4-4.

[10] X. Peng, R. Liu, and S. Yu, "Optimizing weight mapping and data flow for convolutional neural networks on RRAM based processing-in-memory architecture," *IEEE International Symposium on Circuits and Systems (ISCAS),* 2019, pp. 1-5.

[11] S. Wu, G. Li, F. Chen and L. Shi. "Training and inference with integers in deep neural networks," *International Conference on Learning Representations (ICLR),* 2018.

[12] X. Sun and S. Yu, "Impact of non-ideal characteristics of resistive synaptic devices on implementing convolutional neural networks," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2019.

[13] X. Sun, S. Yin, X. Peng, R. Liu, J. S. Seo and S. Yu, "XNOR-RRAM: A scalable and parallel resistive synaptic architecture for binary neural networks," *IEEE Design, Automation & Test in Europe Conference (DATE),* 2018, pp. 1423-1428.

[14] B. Govoreanu, G.S. Kar, Y.Y. Chen, V. Paraschiv, S. Kubicek, A. Fantini, I.P. Radu, L. Goux, S. Clima, R. Degraeve and N. Jossart, "10×10nm^2 Hf/HfO_X crossbar resistive RAM with excellent performance, reliability and low-energy operation," *IEEE International Electron Devices Meeting (IED*M), 2011, pp. 31-6.

[15] 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,* 37(12), 2018, pp.3067-3080.

[16] C.C. Chou, Z.J. Lin, P.L. Tseng, C.F. Li, C.Y. Chang, W.C. Chen, Y.D. Chih, and T.Y.J. Chang, "An N40 256K× 44 embedded RRAM macro with SL-precharge SA and low-voltage current limiter to improve read and write performance," *IEEE International Solid-State Circuits Conference (ISSCC),* 2018, pp. 478-480.

[17] P. Jain, U. Arslan, M. Sekhar, B.C. Lin, L. Wei, T. Sahu, J. Alzate-vinasco, A. Vangapaty, M. Meterelliyoz, N. Strutt and A.B. Chen, "A 3.6 Mb 10.1 Mb/mm 2 embedded non-volatile ReRAM macro in 22nm FinFET technology with adaptive forming/set/reset schemes yielding down to 0.5 V with sensing time of 5ns at 0.7 V," *IEEE International Solid-State Circuits Conference (ISSCC),* 2019, pp. 212-214.

[18] https://www.nextplatform.com/2018/05/10/tearing-apart-googles-tpu-3-0-ai-coprocessor/

[19] H. Jiang, X. Peng, S. Huang and S. Yu, "CIMAT: A transpose SRAM-based compute-in-memory architecture for deep neural network on-chip training," *ACM the International Symposium on Memory Systems (MEMSYS),* 2019.