A ReRAM-Based Computing-in-Memory Convolutional-Macro With Customized 2T2R Bit-Cell for AIoT Chip IP Applications

Fei Tan, Yiming Wang, Yiming Yang, Liran Li, Tian Wang, Feng Zhang, Xinghua Wang, Jianfeng Gao, and Yongpan Liu, *Senior Member, IEEE*

***Abstract*—To reduce the energy-consuming and time latency incurred by Von Neumann architecture, this brief developed a complete computing-in-memory (CIM) convolutional macro based on ReRAM array for the convolutional layers of a LeNet-like convolutional neural network (CNN). We binarized the input layer and the first convolutional layer to get higher accuracy. The proposed ReRAM-CIM convolutional macro is suitable as an IP core for any binarized neural networks' convolutional layers. This brief customized a bit-cell consisting of 2T2R ReRAM cells, regarded 9 × 8 bit-cells as one unit to achieve high hardware compute accuracy, great read/compute speed, and low power consuming. The ReRAM-CIM convolutional macro achieved 50 ns product-sum computing time for one complete convolutional operation in a convolutional layer in the customized CNN, with an accuracy of 96.96% on MNIST database and a peak energy efficiency of 58.82 TOPS/W.**

***Index Terms*—ReRAM, computing-in-memory, AIoT application, CNN, convolutional layer, edge computing, artificial intelligence.**

I. Introduction

ARTIFICIAL neural networks (ANN) have significantly fueled the development of the Internet of Thing (IoT) applications and edge computing. However, their high demands for computing and memory resource contradicts limited resources on edge devices. This contradiction poses various challenges for artificial intelligence (AI) edge devices. The tremendous amount of data movement between memory and computing-units is power and time consuming as well, which raises more problems to hardware based on Von Neumann architecture.

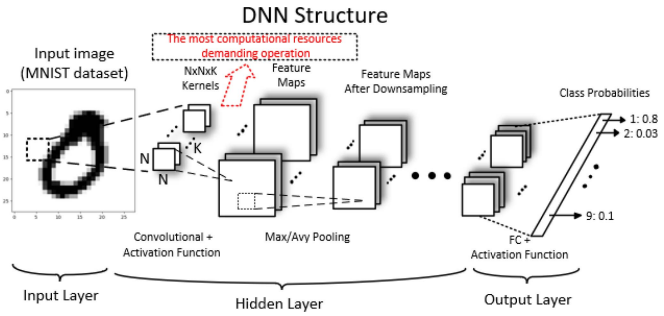


Fig. 1. The typical deep neural network structure.

Take deep neural networks (DNNs) for example, as shown in Fig. 1, a common deep neural network (DNN) contains an input layer, a hidden layer, and an output layer. The hidden layer consists of convolutional layers (CNN), pooling layers, activation layers, and fully-connected (FC) layers. Reference [1] shows that 90% of the total operations are performed in the convolutional layers (CNN), which mostly contains product-sum (PS) operation. All these neural network layers are computationally intensive and require the movement and storage of enormous volumes of data. [2]–[4] show that compressing the size of the neural network proves effective. Nevertheless, they suffer from huge power waste and large latency due to data movement between memory and computing-units.

Computing-in-memory (CIM) methods have been proposed to improve computing efficiency owing to their excellent capability of parallel computing in memory. ReRAM devices are considered a right candidate for CIM structure. As shown in Fig. 2, CIM devices do not need to transfer the vast amount of intermediate data between memory and processors. Besides, the ReRAM with crossbar structure is naturally suitable for matrix-vector/matrix-kernel multiplication (MVM/MKM), which enables ReRAM devices to realize high parallel computation with low power consumption.

In this brief, we implemented a complete customized convolutional neural network (CNN) on the MNIST database with convolutional layers computing in our ReRAM-CIM convolutional macro. To achieve high parallelism and energy efficiency, we customized a bit-cell consisting of 2T2R ReRAM

Manuscript received May 29, 2020; revised July 25, 2020; accepted July 27, 2020. Date of current version September 3, 2020. This work was supported in part by the National Key Research Plan of China under Grant 2018YFB0407500, in part by the National Major Science and Technology Special Project under Grant 2017ZX01028101-303, and in part by the National Natural Science Foundation of China under Grant 61720106013. This brief was recommended by Associate Editor A. Cabrini. *(Corresponding authors: Feng Zhang; Xinghua Wang.)*

Fei Tan, Yiming Wang, Yiming Yang, Liran Li, Tian Wang, and Xinghua Wang are with the School of Information and Electronics, Beijing Institute of Technology, Beijing 100081, China (e-mail: 89811@bit.edu.cn).

Feng Zhang and Jianfeng Gao are with the Key Laboratory of Microelectronics Devices and Integrated Technology, Institute of Microelectronics Chinese Academy of Sciences, Beijing 100029, China (e-mail: zhangfeng_ime@ime.ac.cn).

Yongpan Liu is with the Department of Electronic Engineering, Tsinghua University, Beijing 100084, China.

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSII.2020.3013336

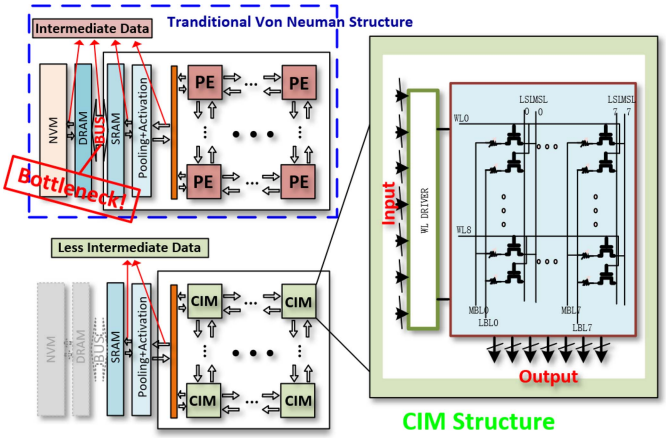


Fig. 2. Conceptual of ReRAM-CIM for AI Edge processors.

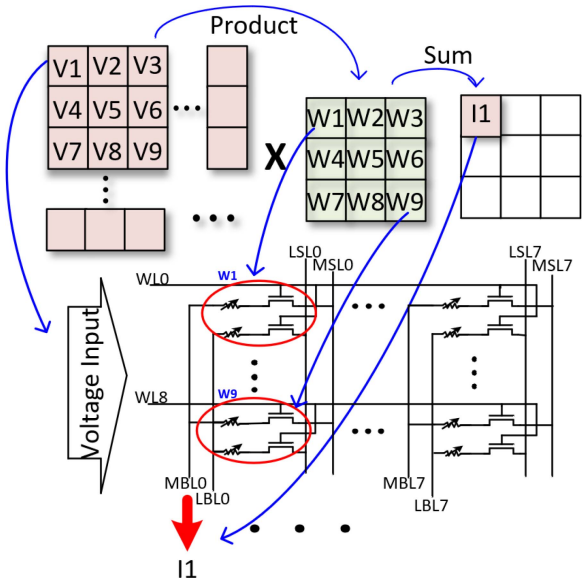


Fig. 3. The proposed 2T2R-ReRAM array for vector-matrix multiplication:the matrix V1 to V9 is applied as a vector to the inputs WL_0 to WL_8 and is processed in 2 steps with serially entering 2 bits.

cells and regarded 9×8 bit-cells as one unit to process the convolution operations in hidden layers.

The remainder of this brief is organized as follows: Section II briefly describes Neural networks for IoT application and ReRAM Computing-in-memory (CIM) Technology. Section III describes the proposed ReRAM computing-in-memory convolutional macro (RRAM-CIMCM). Section IV outlines performance results, and Section V presents a conclusion of this brief.

II. Background: Neural Networks for IoT Application and ReRAM Computing-in-Memory Technology

A. Background of Neural Networks for IoT Application

Because of the great generalization ability of deep neural network (DNN), more and more researches displayed DNN on edge devices for various kinds of tasks. Nonetheless, most kinds of DNNs are unfriendly with hardware owing to the floating-point operation, a large amount of data movement, and large word size. To address the practical need, [5] proposed XNORNN neural network (XNORNN), which dramatically reduces hardware requirements by constraining weights and activations to "+1" or "−1". Reference [6] restricted the inputs and weights to ± 1 or $0/1$. Pioneering works about computing-in-memory (CIM) widely adopted some of these neural networks [7]–[10].

In this brief, an inspired LeNet-5-like network was displayed on the proposed computing-in-memory (CIM) system on the MNIST database and achieved 96.96% accuracy. Our neural network (NN) contains two convolutional layers, two pooling layers, and two fully-connected layers. We binarized the input layer and the first convolutional layer by rounding those float-point numbers to the nearest 2bit fixed-point number with ties.

B. Background of ReRAM Computing-in-Memory Technology

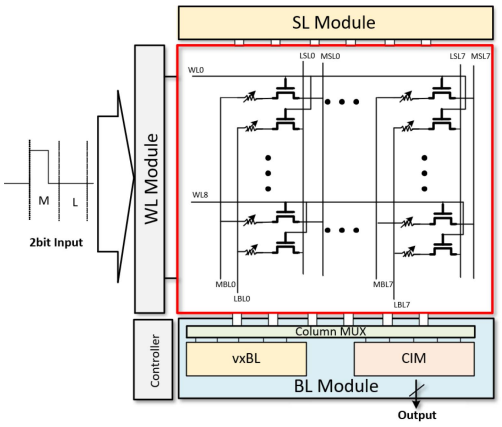
DNNs, displayed on Von-Neumann structures always require a large amount of data movement between memory and processor. The movement of the vast intermediate data leads to massive energy consumption and time latency. The technology of computing-in-memory (CIM) is one way to overcome this problem. Reference [11] shows that emerging non-volatile memory devices have attracted significant attention for CIM technology. Resistive switching memory (ReRAM) is one of the most promising kinds, which is two-terminal circuits element with resistance/conductance that can be tuned by an external stimulus. ReRAMs could be used not only as ultra-low standby power builders but active participants in data processing as well. Shown as Fig. 3, ReRAMs are usually assembled in crossbar formation, which is able to implement vector-matrix multiplication intrinsically.

High energy efficiency and low time latency can be achieved by using ReRAM array as the key processing unit. As shown in Fig. 4(a), this brief not only used ReRAM array as product-sum (PS) processing unit but for high parallelism, customized a bit-cell consisting of 2T2R cells, and regarded nine bit-cells as one unit to process the convolution operations in hidden layers. Details would be described in the next section.

III. Proposed ReRAM Computing-in-Memory Convolutional Macro for CONV Operation

A. Architecture of ReRAM-CIM Convolutional Macro

Fig. 4 shows the architecture of the proposed ReRAM-CIM convolutional macro and the truth table of the multiply operation. The whole device consists of customized 2T2R bit-cells and peripheral circuits for operations in two modes: memory and computing-in-memory (CIM) mode. Memory mode is to store the trained weight while CIM mode is for the convolutional computing operations in DNNs (CIM operation). The peripheral circuits are divided into three blocks: the word line (WL) signal generation module, the bit line (BL) signal generating module, and the source line (SL) signal generating module corresponding to the BL module. Controlled



(a)

WL Input(2bit)	Unit Resistance			
	HH(00)	HL(01)	LH(10)	LL(11)
00	0000/0	0000/0	0000/0	0000/0
01	0000/0	0001/1	0010/2	0011/3
10	0000/0	0010/2	0100/4	0110/6
11	0000/0	0011/3	0110/6	1001/9

(b)

Fig. 4. (a). The architecture of the proposed ReRAM-CIM convolutional macro. (b). The truth table of the multiply operation.

by inputting clock signals, enable signals, and voltage signals, these three modules work together toward the goal of integration of storage and calculation.

In memory mode, WL module would activate one row for write operation, while BL & SL would activate one column. Thus, like the traditional memory write operation, only one customized bit-cell can be accessed every clock cycle. In each clock cycle, BL & SL modules would assign different programming voltage on ReRAM cell's two terminals to change its resistance. Meanwhile, the WL module, controlled by a counter, would auto-select WL_0 to WL_8 one by one every clock cycle. Thus, all the ReRAM cells on this column would be set to high resistance (HR) or low resistance (LR) after 9 clock cycles. In such a situation, the trained weights are stored in the customized 2T2R cells through a write operation in ReRAM mode.

In CIM mode, the proposed ReRAM-CIM convolutional Macro complete the convolutional computing operation, which is based on the product-sum operation, with kernels of 3×3 and data of 2 bit. The 2bit input multiplies by the trained weight stored in ReRAM ($W_j = HR = 0$ or $W_j = LR = 1$) by controlling the state of the transistor within each ReRAM cell. If the input voltage on the corresponding WL node (V_{WL}) is high, which means the input is "1", that is, the transistor is on. Thus, a current can be get:

$$V \times Y = I. \tag{1}$$

For one multiplication operation, all the possible cases are shown in the truth table in Fig. 4(b). Different calculation results lead to different currents. Then, on the BL terminal, the current is collected and digitalized. The sum would be

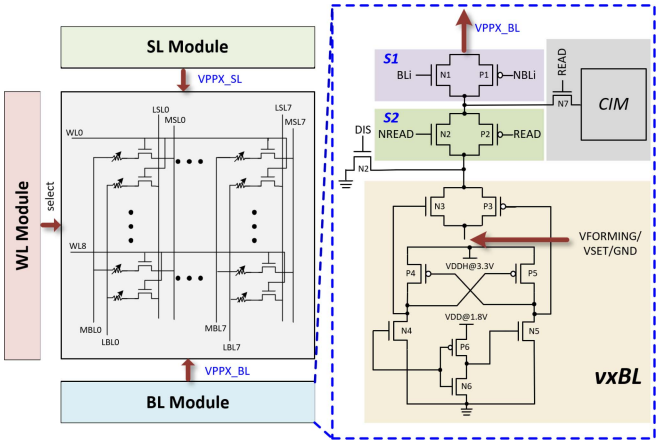


Fig. 5. The circuits for write operation.

calculated and digitalized within the BL module. The whole process can be represented as the equation of the form:

$$I_i = \sum W_{ij} \times V_{WL_j}, j = 0, 1, \ldots, 8 \tag{2}$$

which is exactly the same with the convolutional operation.

B. Operation and Circuits of ReRAM-CIM Convolutional Macro

1) Write Operation in ReRAM Array: As shown in Fig. 5, according to the input address, the decoder in the BL and SL modules will control the closing and opening of $S1$, while the WL module will auto-select WL_0 to WL_8 one by one. Thus, in every clock cycle, one specific ReRAM cell is selected under the control of WL, BL and SL modules. The enable signal READ controls the conversion between the memory mode and the CIM mode in the BL and SL modules. The READ signal, which passes the voltage conversion module, could generate an opposite voltage signal NREAD to control $S2$.

In the memory mode, the NREAD signal is high and the READ signal is low. Thus, $S2$ in Fig. 5, which is for writing, is on, and the switch for CIM operation is off. As shown in Fig. 5, the bit line driving voltage generating module vxBL generates a voltage VPPX_BL, which's value is under the control of the enable signals SET, RESET, and FORMING. The VPPX_BL signal will pass through $S1$, $S2$ and finally reach one end of the selected ReRAM cell. At the same time, the source line driving voltage generating module vxSL is generating VPPX_SL as well. Under control of the same clock, vxBL and vxSL will apply VPPX_BL and VPPX_SL to the two ends of the selected ReRAM cell to change its resistance.

2) The CONV Operation in ReRAM-CIM Convolutional Macro: during the convolution operation, the SL module is connected to ground. As for the BL module, the enable signal READ is high. $S2$ in Fig. 5 is closed, and the N7 is on. In this way, the currents generated by multiplying the input signal W_VREAD (V_{in}) with the resistance value of the ReRAM cell will come from the CIM module, pass through N7, then through the $S1$ controlled by the address line signal, and finally go through the ReRAM array to the SL module to ground. Fig. 6 shows the detailed waveform of ReRAM-CIM

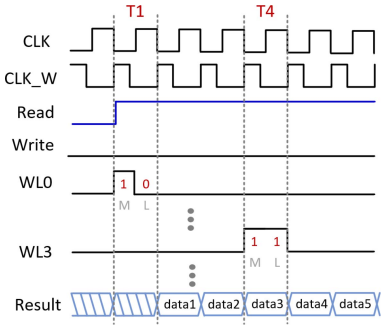


Fig. 6. The detailed waveform of ReRAM-CIM in CONV operation.

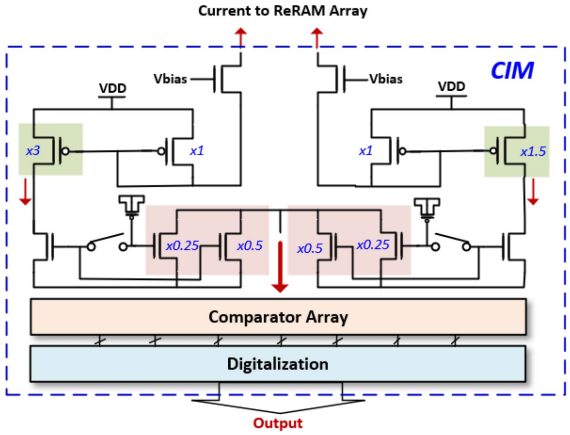


Fig. 7. The weighted circuits module: the switch on and off depends on the MSB or LSB of the WL and the LSB contribution is stored in the capacitance.

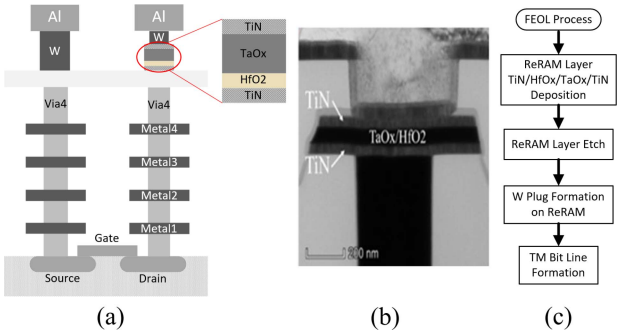


Fig. 8. (a). The schematic of HFOx embedded ReRAM cells. (b). The structure of TiN top electrode//TiOx/HfO2//TiN bottom electrode. (c). Process flow based on 0.18um CMOS.

Technology	0.18um CMOS Logic Process
RRAM	Logic Process
Cell Size(2T2R)	16.3488 μm^2
RRAM Mode	Memory and CIM
CIM Mode	ConV Operation for CNN
Read Delay @VDD = 1.8V	CIM Mode:50ns (2b-input/2b-weight/7bit output)
Energy Efficiency @VDD = 1.8V	CIM Mode: 58.82 TOPS/W (2b-input/2b-weight/7bit output)

(a) (b)

Fig. 9. (a). Die photo. (b). Summary of the fabricated test-chip.

IV. Experiment Results

This brief employed HfOx embedded ReRAM cells, as shown in Fig. 8, which is provided with a wide operating temperature ranges. The proposed ReRAM-CIM convolutional macro is fabricated in a 0.18 μm CMOS process. Fig. 9 presents a die photograph and a chip summary. The effective area excluding the pads is $297 \times 233 \mu m^2$. The chip is wire-bonded to a PCB with a chip-on-board (COB) setup and is tested with the stc89c51 and ax7035 as shown in Fig. 10(a).

The demo system used the proposed ReRAM-CIM convolutional macro for the CONV operation in the first CNN layer and used the FPGA host, which contains an embedded processor, to do all the calculations beside CONV operation.

Firstly, we initialized the whole ReRAM array by enabling the FORMING signal and applying 3.3V and 0V to the ReRAM cells' two ends through the BL and SL modules, respectively. Then, we set the chip in memory mode and stored the trained weights in the ReRAM array. We used pulse width modulation method and FPGA for inputting the signal. We read out the resistance value after putting the writing pulse, and if it is not right, we would increase the writing pulse width and put it to the chip again. We would repeat these steps until the correct resistance was written to RRAM. The set operation is to apply voltages of 2.5V and 0V to the ReRAM cells' two ends through the BL and SL modules, while the reset operation is to apply 0V and 2.5V voltages through BL and SL modules. In our test, the write accuracy rate is 93%.

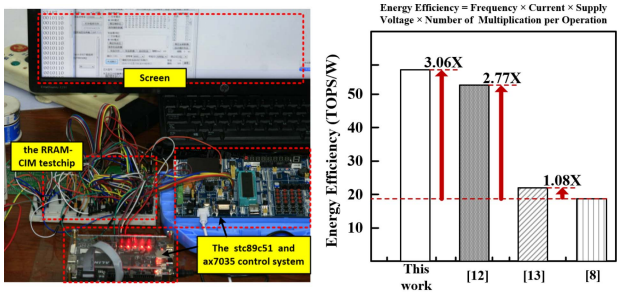
for CONV operation. In the proposed ReRAM-CIM convolutional macro, when V_{in} is "1", the transistor, which connects to the ReRAM cell, is turned on. Thus, the supply voltage (*VDD* in Fig. 7) is applied on the ReRAM cell. If the admittance value of the ReRAM cell is low, the weight it represents is 0 ($W_j = 0$). According to formula (1), the multiplication result is 0; otherwise, if the admittance value of the ReRAM cell is high, the weight it represents is 1 ($W_j = 1$), and the multiplication result would be the input value.

One bit-cell represents 2 bits. The two ReRAM cells in one bit-cell correspond to different weights. In the circuits shown in Fig. 7, the logic value of the high-order ReRAM cell will be enlarged by 3×, while the logic value of the low-order ReRAM cell will only be enlarged by 1.5×. Similarly, the continuous 2bit input data from the WL module would be processed. AS shown in Fig. 7, the high-order input data will be reduced to 1/2 of its original value, while the low-order input data will be reduced to 1/4 of its original value to achieve low power consumption.

Because of the customized 2T2R bit-cell, the multiply operation of high-order ReRAM cell and low-order ReRAM cell will go together, which significantly reduces the computing time and increases the energy efficiency of this brief. The result produced in each cycle will pass through the comparator array and voltage conversion module HVSW for waveform processing and digitization to avoid the misjudgment of computing results caused by the low and high resistance window of ReRAM.

TABLE I
COMPARISON WITH REFERENCE WORKS

	This work	[12]	[13]	[8]
Technology	180nm	55nm	180nm	65nm
Input bit # (bit)	2	1	1	1
Weight bit # (bit)	2	3	8	Ternary
Accuracy(MNIST)	96.96%	98.2%	90.8%	98%
Sensing Scheme	Voltage comparator	ML − CSA	Current comparator	ML − CSA
Output Precision(bit)	7b output	1b sign+3b DOUT	1	3b output
Energy Efficiency (TOPS/W)	58.82	53.17	20.7	19.2



(a) (b)

Fig. 10. (a). The demo system. (b). Magnification of this brief.

After the write operation is completed, the binary graphic data was serialized into the WL module and view the output on the BL side. The input on the WL side is 2 bits, and the output on the BL side is 7 bits. That is, a 3×3 convolution kernel performs a convolution on the feature map and sums it up.

The ReRAM-CIM convolutional macro achieved 50 ns product-sum computing time for one complete convolutional operation in a convolutional layer in the customized CNN and achieved an inference accuracy of 96.96% on the MNIST database.

Table I shows a performance summary and comparison with previous works. This brief demonstrated a better or comparable peak energy efficiency of 58.82 TOPS/W. As shown in Fig. 10, the proposed ReRAM-CIM convolutional macro improves the peak energy efficiency by 3.06× compared with [8], 2.84×cmopared with [13] and 1.11×cmopared with [12].

V. CONCLUSION

This brief developed a complete computing-in-memory (CIM) convolutional macro based on ReRAM array for the convolutional layers of a LeNet-5-like CNN to achieve lower power consumption and less time latency. Unlike previous works adapting 1T1R structure, to get high accuracy and high energy efficiency, this brief customized a bit-cell consisting of 2T2R cells to represent 2bit data. The ReRAM-CIM convolutional macro achieved 50 ns product-sum computing time for one complete convolutional operation in a convolutional layer in the customized CNN and achieved 96.96% accuracy on the MNIST database. The measured peak energy efficiency reached 58.82 TOPS/W.

REFERENCES

[1] G. Desoli *et al.*, "14.1 A 2.9TOPS/W deep convolutional neural network SoC in FD-SOI 28nm for intelligent embedded systems," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, San Francisco, CA, USA, Feb. 2017, pp. 238–239.

[2] N. Ma, X. Zhang, H. Zheng, and J. Sun, "ShuffleNet V2: Practical guidelines for efficient CNN architecture design," 2018. [Online]. Available: https://arxiv.org/abs/1807.11164

[3] Y. Yang *et al.*, "Synetgy: Algorithm-hardware co-design for ConvNet accelerators on embedded FPGAs," 2018. [Online]. Available: https://arxiv.org/abs/1811.08634

[4] C. Ding, S. Wang, N. Liu, K. Xu, Y. Wang, and Y. Liang, "REQ-YOLO: A resource-aware, efficient quantization framework for object detection on FPGAs," in *Proc. ACM/SIGDA Int. Symp. Field Program. Gate Arrays*, 2019, pp. 33–42.

[5] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Computer Vision—ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham, Switzerland: Springer, 2016, pp. 525–542.

[6] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Red Hook, NY, USA: Curran Assoc., Inc., 2016, pp. 4107–4115.

[7] X. Sun, R. Liu, X. Peng, and S. Yu, "Computing-in-memory with SRAM and RRAM for binary neural networks," in *Proc. 14th IEEE Int. Conf. Solid-State Integr. Circuit Technol. (ICSICT)*, Qingdao, China, Oct. 2018, pp. 1–4.

[8] W. Chen *et al.*, "A 65nm 1Mb nonvolatile computing-in-memory ReRAM macro with sub-16ns multiply-and-accumulate for binary DNN AI edge processors," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, San Francisco, CA, USA, Feb. 2018, pp. 494–496.

[9] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binary neural networks: Training deep neural networks with weights and activations constrained to +1 or −1," 2016. [Online]. Available: https://arxiv.org/abs/1602.02830

[10] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," 2016. [Online]. Available: https://arxiv.org/abs/1603.05279

[11] C. Li *et al.*, "In-memory computing with memristor arrays," in *Proc. IEEE Int. Memory Workshop (IMW)*, May 2018, pp. 1–4.

[12] C. Xue *et al.*, "Embedded 1-Mb ReRAM-based computing-in- memory macro with multibit input and weight for CNN-based AI edge processors," *IEEE J. Solid-State Circuits*, vol. 55, no. 1, pp. 203–215, Jan. 2020.

[13] R. Mochida *et al.*, "A 4M synapses integrated analog ReRAM based 66.5 TOPS/W neural-network processor with cell current controlled writing and flexible network architecture," in *Proc. IEEE Symp. VLSI Technol.*, Honolulu, HI, USA, Jun. 2018, pp. 175–176.