Design Framework for SRAM-Based Computing-In-Memory Edge CNN Accelerators

Yimin Wang, Zhuo Zou, and Lirong Zheng
Fudan University, Shanghai, China
Email: {ymwang19.zhuo.lrzheng}@fudan.edu.cn

2021 IEEE International Symposium on Circuits and Systems (ISCAS) | 978-1-7281-9201-7/20/$31.00 ©2021 IEEE | DOI: 10.1109/ISCAS51556.2021.9401746

***Abstract*—This paper presents an architectural framework and an evaluation model for Static Random Access Memory (SRAM)-based Computing-in-Memory (CIM) edge Convolutional Neural Network (CNN) accelerators. To provide a baseline for system-level design perspectives, an architectural framework for SRAM-CIM design concerning the key design points in state-of-the-art works is proposed. Furthermore, a configurable evaluation model featuring top-down design flow based on the proposed framework is established to investigate design space explorations. Case studies validated the framework and evaluation model using LeNet-5, AlexNet and VGG-16 to achieve energy-aware optimizations. The optimized memory scale for LeNet-5 is "16 PEs and 120 tiles" with the minimal estimated inference energy of 0.0018μJ, while for AlexNet and VGG-16, "16 PEs and 120 tiles" is better achieving minimal energy consumption of 0.1733mJ and 0.6825mJ respectively. Estimation results highlight tradeoffs among data representation parameters and memory partitioning parameters. This work provides specific SRAM-CIM design guidelines from a system-level perspective.**

Keywords—computing-in-memory, SRAM, edge computing, CNN

I. Introduction

With the rapid development of Artificial Intelligence (AI), Convolutional Neural Networks (CNN) plays an important role in data processing, boosting "the Information Era" into "the Intelligent Era" [1]. Nevertheless, CNN processing is accompanied by enormous multiply-and-accumulate (MAC) operations. Extensive computation complexity is challenging for hardware: tradeoffs among power, latency, and area are critical particularly in resource-constrained edge devices.

Meanwhile, novel Computing-in-Memory (CIM) architecture for edge accelerator has aroused immense research interest to overcome the drawbacks of traditional computing architecture where processing units and memory units are separate. CIM architecture is characterized as massive parallelism and locality of data processing, breaking the obstacle of "memory wall" by performing computations directly within memory macros. This novel architecture suppresses the energy and time overhead during intermediate data transfer, enabling power reduction for CNN processing.

CIM accelerators implemented in prior works using SRAM [2]–[9] as basic processing cell have shown great potential for CNN inference. However, there are still several concerns to be noted. First, current implemented CIM accelerator are non-standardized: most efforts are focusing on fully-customized design of devices and circuit modules for CIM, while no proportional attentions are paid to system-level explorations that regard CIM-based circuit macro as an on-chip module. Secondly, dedicated lightweight neural networks are commonly utilized in current SRAM-CIM works, while how to map and schedule commonly-used complicated CNNs onto the chip is also required to be considered coordinately in hardware design phase. Thirdly, remaining open source Computer Aided Design (CAD) tools for CIM design are mainly bottom-up simulators established according to eNVM (emerging non-volatile memory) [10]–[12] characteristics, which cannot completely cover the SRAM-CIM design configurations. Therefore, an analysis framework that is configurable for state-of-the-art SRAM-CIM design should be proposed to facilitate top-down design guidance and explore design baselines towards co-design of CNNs and SRAM-CIM chips.

To mitigate these gaps, we propose an architectural framework for SRAM-CIM edge CNN accelerators in this paper. An evaluation model based on the proposed architecture is established to explore optimizations and investigate the effects of design parameters, providing design guidelines for future design. Case studies using LeNet-5, AlexNet and VGG-16 are shown to demonstrate the feasibility of the architectural framework and evaluation model.

II. SRAM-CIM Preliminaries

SRAM cells used in remaining CIM works vary from 6T to 12T. Except for basic 6 transistors in SRAM cell, the additional transistors help enhance signal stability and enable signed operations. The equivalent model of SRAM cell can be simplified to 2 transistors as shown in Fig. 1(a). In (a), the transistor connected to GND is controlled by the stored data, which represents the 1-bit weight of CNN kernels. When the stored weight is "1", current I_{DS} will be driven from BL to GND. While the other transistor is controlled by WL, and when WL is activated, there is a voltage change $\Delta V = \frac{1}{C_{BL}} \times \int_0^{T_{WL}} I(t)\mathrm{d}t$ on the BL. Only if the stored weight is "1" and the WL is activated, the ΔV can be reflected on BL as ΔV_{BL}. In this way, SRAM cells implemented the multiplication of two unsigned 1-bit numbers.

When stacked in array like (b), the column composes a fundamental PE for vector multiplication, while the element precision in one of the two vectors is limited to 1-bit. As shown in (c), the voltage on BL represents the sum of 1-bit multiplication results conducted on R cells, as (1) shows.

$$\Delta V_{BL} = \left(W_1 \times \frac{1}{C_{BL}} \times \int_0^{T_{WL}} I_1(t)\mathrm{d}t \right) + \left(W_2 \times \frac{1}{C_{BL}} \times \int_0^{T_{WL}} I_2(t)\mathrm{d}t \right) + \ldots + \left(W_R \times \frac{1}{C_{BL}} \times \int_0^{T_{WL}} I_R(t)\mathrm{d}t \right) \quad (1)$$

This change on voltage will be sampled and quantized as a digital output by peripheral circuit. And in order to perform high precision multiplication, for example, N_w bit elements have to be stored in N_w cells within a row but in N_w separate columns. Thus, N_w columns composes a PE for multiplication of two multibit-element vectors, which will be discussed in III(A).

978-1-7281-9201-7/21/$31.00 ©2021 IEEE

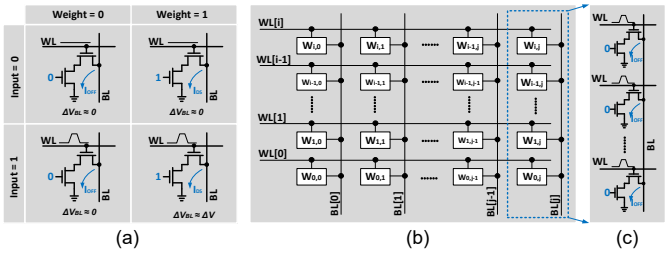

Fig. 1. (a) SRAM cells, (c) SRAM in column and (b) SRAM array for CIM operation.

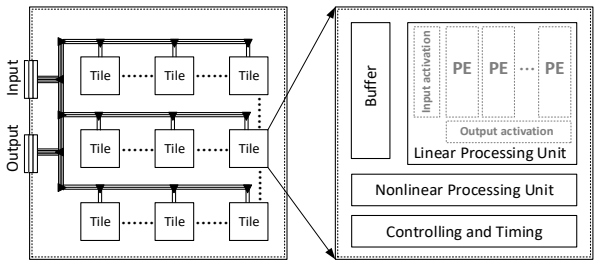

Fig. 2. Hierarchical architecture of SRAM-CIM accelerator.

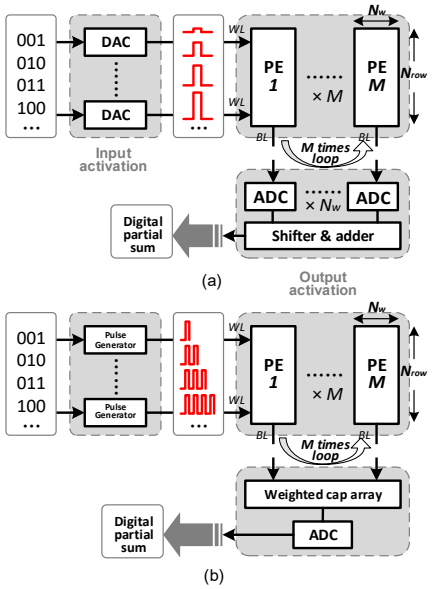

Fig. 3. Block diagram of linear processing unit: (a) VIVO architecture, (b) TICO architecture.

III. Architectural Framework for SRAM-CIM

A. SRAM-CIM chip architecture

To set up an analysis model for SRAM-CIM accelerators, a baseline architectural framework is required. Fig. 2 shows the hierarchical architecture of the SRAM-CIM accelerator in 2 levels: chip-level and tile-level. In (a), buses transfer input feature map, weights, and partial sum between tile buffers and I/O interfaces. Within a tile, linear processing unit processes operations in convolutional layers and full-connected layers. Nonlinear operations are processed by pooling units, sigmoid units, and others contained in the nonlinear processing unit. The controlling and timing units are responsible for the tile's configurations and communications.

In the linear processing unit, processing involves three phases: input activation, multiplication in PE, and output activation. We use N_{in} and N_w to represent the precision of input data and weight data. N_w bit weight data is stored in N_w cells within a row but distributed in N_w columns, where each column generates a partial result which are to be weighted and summed to gain the full precision output by peripheral circuits in output activation module. Output activation circuits are multiplexed by the PEs to save area overhead. We propose two kinds of architectures for linear processing unit: 1) voltage domain input and voltage domain output (VIVO) activation; and 2) time domain input and charge domain output (TICO) activation, as illustrated in Fig. 3.

1) VIVO architecture: As shown in (a), the input signal is activated by the amplitude-modulated voltage which is proportional to the input data. This input voltage is put onto the WL of N_w bit-cells simultaneously, and after generating the results of multiplication on BLs, N_w ADCs sample the voltage respectively on N_w BLs in parallel. Then, the quantified digital results are put into the shifter and adder so as to generate the final result. The feasibility of VIVO based on 6T SRAM has been validated in [2], and works on dual-split 6T [3] and 12T [4] are also presented. One concern for VIVO architecture is limited signal margin in voltage domain when supply voltage is fixed, which may cause accuracy penalty. Another concern is high area overhead since each BL is equipped with an ADC.

2) TICO architecture: In (b), the input signal is activated by pulse sequences, whose number is proportional to the input data. N_w columns are also activated by the same input signal in parallel. When pulses are transferred onto WLs, each pulse will generate a fixed voltage change on BLs which will be further shifted into the charge domain by capacitor arrays, in this way, input pulses are able to control the charge on the capacitor linearly. The cap arrays are stacked by caps varying from $2C_0$ to $2^{N_w}C_0$ (C_0 is the minimal capacitor representing the least significant bit), after each capacitor has been charged, the full precision output is read out by only one ADC. Saving the consumption of (N_w—1) ADCs, cap array here plays a core role in full precision result generation, the controlling scheme of cap array has been discussed in [5]–[8]. Compared with VIVO architecture, TICO requires less area, however, pulse-based input activation will bring challenges for time delay.

Besides, what should be considered is the method for signed multi-bit weight operation. The operations of positive weights and negative weights are processed separately. In recent designs, signed weights are stored in 2's complement pattern [5], [9]. And during processing steps, judging by the sign bit, multiplication results will be put onto different signal rails.

B. CNN scheduling and mapping

Fig. 4 depicts the dataflow of on-chip CNN inference. We define one time of reloading weights and inference processing of the entire chip as one "reload cycle", and one time of partial sum to be generated by output activation circuit as one "readout cycle". In each reload cycle, the PEs access the buffer at the beginning for reading in weights and storing the weights into bit-cells, following by multiple readout cycles due to the sliding of kernels. L1 in Fig. 4 stands for a lightweight layer that can be processed within a reload cycle. The output of L1 is processed by a nonlinear processing unit

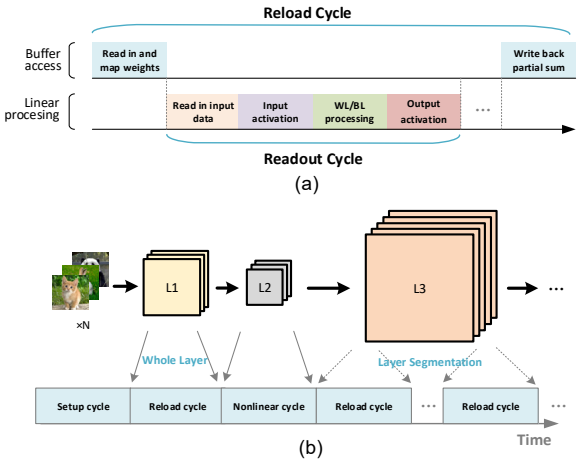

Fig. 4. Diagram of reload cycle and readout cycle; (b) scheduling strategies for different layers.

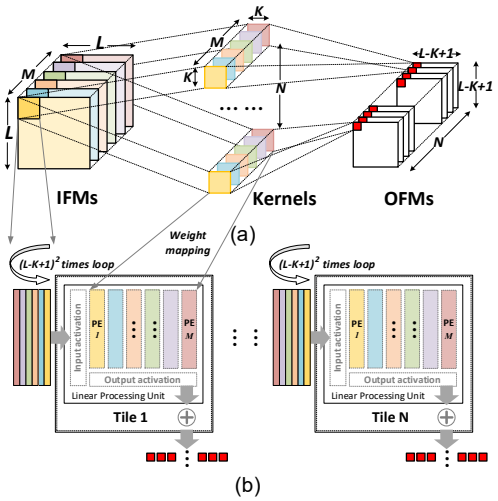

Fig. 5. (a) A convolutional layer of DNN; (b) mapping the layer to SRAM array.

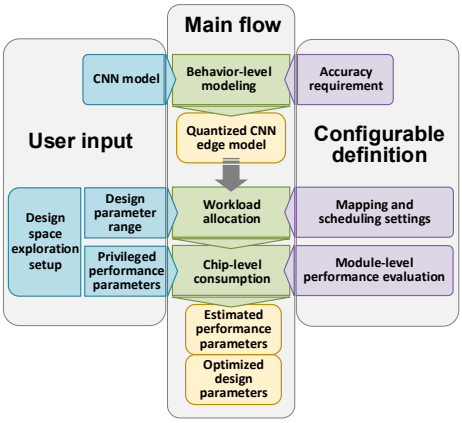

Fig. 6. Overview of the evaluation model.

subsequently in a nonlinear processing cycle. Whereas L3 stands for a heavyweight layer which has to be split into multiple reload cycles and the partial sum output should be further combined by controlling circuits to generate the complete output feature map.

Considering the convolutional layers, as shown in Fig. 5, the kernels are reshaped into 1-D vectors and stored into bit-cells along columns. Since the 1-D kernel size K^2 is affordable for the row number, for example, 25 in LeNet-5, and 9 in VGG Net, which makes sure that multiplications of inputs and kernels can be processed in one column without segmentation. The input channel number M indicates the number of utilized PEs, and the output channel number N indicates the number of utilized tiles. If the number of required PEs or tiles exceeds the total number, the exceeding part will be split and processed in another reload cycle, therefore processing latency extends proportionally. For the fully-connected layers, since the size of FC layers are usually larger than the memory size in a PE, segmentation is inevitable. Processing of fully-connected layers commonly dominates the total overhead of CNN processing on CIM chips.

IV. Evaluation Model

To explore the design space of SRAM-CIM, we developed an evaluation model. The main functions of this evaluation model contain: support CNN with different shapes and depth; support coarse estimation of accelerator performance; support design parameter optimizations under privileged metrics, and support customized configurations by user to explore design space.

Fig. 6 shows the workflow of the proposed evaluation model. There are three main steps in the evaluation process. The first thing to be considered is the quantization of the pre-trained CNN. The behavior-level modeling based on the architectural framework in Section III is used to estimate the CNN classification accuracy under various quantization rate. The output of this step is a quantized CNN model meet the required accuracy. Then, according to the shape of the quantized model and the mapping-scheduling settings, workload is allocated to specific tiles and communications among them is defined within the design parameter range. The user-defined design parameters are shown in Table I. The design parameters are categorized into two types: data representation parameters and memory partitioning parameters. In the third step, chip-level consumption characterized by energy, latency and area are estimated based on the workload and module-level performance parameters which can be configured to explore user's customized designs. Finally, optimized design parameters are generated to qualify the user's privileged performance parameters. The data representation parameters are optimized in behavior-level modeling, and the memory partitioning parameters are optimized in the latter two steps.

TABLE I. Parameters Involved in Evaluation Model

	Param.	Description
Data representation parameter	P_{in}	Input precision
	P_w	Weight precision
	P_{CONV}	ADC/DAC precision
Memory partitioning parameter	N_{tile}	Tile number
	N_{PE}	PE number per tile; multiplexing rate of output activation circuit
	N_{row}	Row number per PE; row parallelism

V. Results and Discussions

The feasibility of the proposed architectural framework and the evaluation model is benchmarked with three widely used and publicly available CNNs as LeNet-5, AlexNet and VGG-16. The module-level performance evaluation parameters and basic configurations used in case studies are shown in Table II.

TABLE II. CONFIGURATION PARAMETERS IN CASE STUDIES

Parameter	Config.	Parameter	Config.
Supply voltage	1.8 V	Frequency	100 MHz
Input precision	8 bit	ADC precision	8 bit
Bit-cell area	0.1648 μm²	Per-ADC area	172 μm²
Write power	11.6 pJ [12]	Per-ADC power	80 μW
Read power	40 pJ [12]		

TABLE III. ENERGY-AWARE OPTIMIZATION RESULTS

NN type	Category	N_{row}	N_{PE}	N_{tile}	Energy	Energy saving
LeNet-5	*VIVO-O*[a]	25	16	120	0.0018 μJ	81.63%
	VIVO-B[b]	25	64	8	0.0098 μJ	
	TICO-O	25	16	120	0.0057 μJ	81.37%
	TICO-B	25	64	8	0.0306 μJ	
AlexNet	*VIVO-O*	32	4	128	0.1733 mJ	2.15%
	VIVO-B	32	64	8	0.1771 mJ	
	TICO-O	32	4	128	0.6194 mJ	2.55%
	TICO-B	32	64	8	0.6356 mJ	
VGG-16	*VIVO-O*	64	64	8	0.6825 mJ	/
	VIVO-B	64	64	8	0.6825 mJ	
	TICO-O	64	64	8	2.99 mJ	/
	TICO-B	64	64	8	2.99 mJ	

[a] (O) stands for optimized parameters; [b] (B) stands for baseline parameters.

A. Optimizations for tiled SRAM-CIM macros

To compare the performance of SRAM-CIM accelerators with different memory scale, we choose "64@PE per tile, 8@tile, and 8@weight precision" as the baseline configuration which are common parameters in state-of-the-art SRAM-CIM chip design. Table III shows the optimized design parameters and estimated energy consumption when energy is the privileged optimization metric. The results shows that "64 PEs and 8 tiles" fits well for AlexNet and VGG-16, but "16 PEs and 120 tiles" performs better for LeNet-5 with more than 80% energy saving.

B. Tradeoffs among design parameters

1) The influence of data representation parameters: We utilize signal-to-quantization noise ratio (SQNR) to quantify their effect, whose equation is as follows:

$$\mathrm{SQNR} = \frac{P_{\mathrm{signal}}}{P_{\mathrm{noise}}} = \frac{\sum_{n=1}^{N} {y_n}^2}{\sum_{n=1}^{N} (y_n - \hat{y}_n)^2} \qquad (2)$$

where y_n denotes the theoretical inner product of the input data and the stored weight, $\hat{y}_n$ denotes the actual inner product quantized by readout circuits (ADC). Fig. 7 shows SQNR varying with different input precision and row parallelism. It reveals that, with fixed readout ADC precision, increasing of input bit number and row parallelism both compromise the output signal quality.

2) The influence of memory partitioning parameters: The influence of row parallelism has been investigated formerly, while the other two dimensions are dominant in latency and area overhead as we found in the simulation results.

The relationship between latency and area varying with the tile number with fixed PE number per tile is shown in Fig. 8: the area is in inverse proportion to latency, and as the curve shows, 80% of latency can be reduced when normalized area is altered from 0.02 to 0.1, still in a small area level. And as shown in Fig. 9, the area-latency product doesn't vary with the tile number. This reveals that latency is mainly determined by the tile number (memory scale in Table. I).

As illustrated in Fig. 9, the area over the cell number is reduced evidently with increasing PE number per tile. This is because PE number per tile indicates the multiplexing rate of

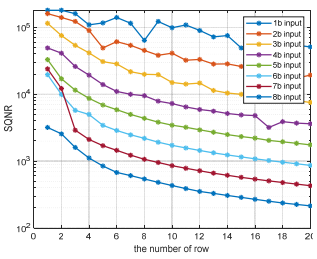

Fig. 7. SQNR under different input precision and row parallelism.

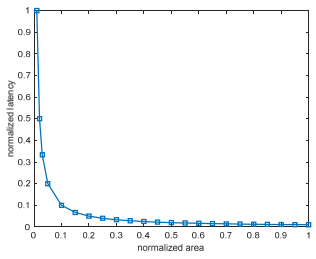

Fig. 8. The relationship between latency and area varying with the tile number with fixed PE number per tile.

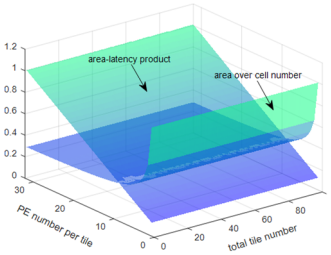

Fig. 9. The normalized latency-area product and normalized area over cell number under different PE number and tile number.

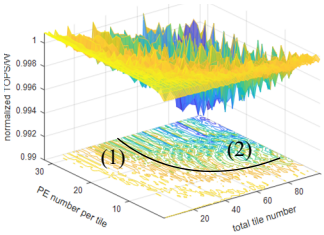

Fig. 10. The tendency of energy-efficiency under different PE number and tile number.

peripheral readout circuits. This trend makes it possible to reduce the area intensity in area-constrained applications. However, the increment of PE number per tile has no effect on the total latency under a fixed tile number. Latency is stuck because the output activation circuits are multiplexed serially among PEs.

Fig. 10 illustrates the tendency of energy efficiency under various memory sizes under the workload of VGG-16. In Fig. 9 (b), the yellow lines in the region (1) are close to the axes, which means that "less intensive tile scale" or "less intensive PE scale" is applicable to energy-efficiency applications. Whereas when the tile number and the PE number are both large in the region (2), the total power is more probable to be higher. This attributes to the low utilization rate of the memory cell and more intensive standby power when the number of tile and PE per tile are both extended to a higher level.

VI. CONCLUSION

In this paper, an architectural framework and an evaluation model for SRAM-CIM edge CNN accelerator is proposed for design space exploration. This work provides specific SRAM-CIM design guidelines from a system-level perspective, and main contributions are: 1) a holistic architectural framework for SRAM-CIM accelerator concerning the key design points in state-of-the-art works is proposed; 2) top-down design flow and configurable aided tools for SRAM-CIM design are instantiated and validated by energy-aware optimization case study; 3) tradeoffs among data representation parameters and memory partitioning parameters are analyzed quantitatively under different resource constraints.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China under Grant 61876039, 62076066 and 62011530132, in part by the Shanghai Platform for Neuromorphic and AI Chip under Grant 17DZ2260900, Shanghai Municipal Science and Technology Major Project (No.2018SHZDZX01) and ZJ Lab.

REFERENCES

[1] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the gap to human-level performance in face verification," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 1701–1708, 2014.

[2] X. Si *et al.*, "A 28nm 64Kb 6T SRAM Computing-in-Memory Macro with 8b MAC Operation for AI Edge Chips," *Dig. Tech. Pap. - IEEE Int. Solid-State Circuits Conf.*, vol. 2020-Febru, pp. 246–248, 2020.

[3] X. Si *et al.*, "A Dual-Split 6T SRAM-Based Computing-in-Memory Unit-Macro With Fully Parallel Product-Sum Operation for Binarized DNN Edge Processors," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 66, no. 11, pp. 4172–4185, 2019.

[4] Z. Jiang, S. Yin, M. Seok, and J. S. Seo, "XNOR-SRAM: In-memory computing SRAM macro for binary/ternary deep neural networks," *Dig. Tech. Pap. - Symp. VLSI Technol.*, vol. 2018-June, pp. 173–174, 2018.

[5] X. Si *et al.*, "A Twin-8T SRAM Computation-in-Memory AI Edge Processors," *IEEE J. Solid-State Circuits*, vol. PP, pp. 1–14, 2019.

[6] Q. Dong *et al.*, "A 351TOPS/W and 372.4GOPS Compute-in-Memory SRAM Macro in 7nm FinFET CMOS for Machine-Learning Applications," *2020 IEEE Int. Solid- State Circuits Conf.* -, pp. 488–489, 2020.

[7] A. Biswas and A. P. Chandrakasan, "CONV-SRAM: An energy-efficient SRAM with in-memory dot-product computation for low-power convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 217–230, 2019.

[8] Z. Jiang, S. Yin, J.-S. Seo, and M. Seok, "C3SRAM: In-Memory-Computing SRAM Macro Based on Capacitive-Coupling Computing," *IEEE Solid-State Circuits Lett.*, vol. 2, no. 9, pp. 131–134, 2019.

[9] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A 64-Tile 2.4-Mb In-Memory-Computing CNN Accelerator Employing Charge-Domain Compute," *IEEE J. Solid-State Circuits*, vol. 54, no. 6, pp. 1789–1799, 2019.

[10] L. Xia *et al.*, "MNSIM: Simulation Platform for Memristor-Based Neuromorphic Computing System," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 37, no. 5, pp. 1009–1022, 2018.

[11] W. Zhang *et al.*, "Design guidelines of RRAM based neural-processing-unit: A joint device-circuit-algorithm analysis," *Proc. - Des. Autom. Conf.*, 2019.

[12] X. Peng, S. Huang, Y. Luo, X. Sun, and S. Yu, "DNN+NeuroSim: An End-to-End Benchmarking Framework for Compute-in-Memory Accelerators with Versatile Device Technologies," *Tech. Dig. - Int. Electron Devices Meet. IEDM*, vol. 2019-Decem, pp. 771–774, 2019.