eF^{2}lowSim: System-Level Simulator of eFlash-Based Compute-in-Memory Accelerators for Convolutional Neural Networks

Jooho Wang*†, Sunwoo Kim*†, Junsu Heo* and Chester Sungchung Park*
*Department of Electrical and Electronics Engineering, Konkuk University, †Memory Business, Samsung Electronics, Inc.
jooho.wang@samsung.com, sunwoo.k.kim@samsung.com, junsuheo@konkuk.ac.kr, chester@konkuk.ac.kr

***Abstract*—A new system-level simulator, eF^{2}lowSim, is proposed to estimate the bit-accurate and cycle-accurate performance of eFlash compute-in-memory (CIM) accelerators for convolutional neural networks. The eF^{2}lowSim can predict the inference accuracy by considering the impact of circuit nonideality such as program disturbance. Moreover, the eF^{2}lowSim can also evaluate the system-level performance of dataflow strategies that have a significant impact on hardware area and performance of eFlash CIM accelerators. The simulator helps to find the optimal dataflow strategy of an eFlash CIM accelerator for each convolutional layer. It is shown that the improvement of area efficiency amounts to 26.8%, 21.2% and 17.9% in the case of LeNet-5, VGG-9 and ResNet-18, respectively.**

***Index Terms*—Compute-in-memory (CIM), convolutional neural network (CNN), dataflow, embedded flash (eFlash), hardware accelerators, system-level simulators**

I. Introduction

Convolutional neural networks (CNNs) comprise multiple computation layers, and each layer performs a considerable number of multiply-accumulate (MAC) operations between the input data and trained weights. The performance and energy efficiency of data-intensive deep neural network chips can be limited by the available memory bandwidth and MAC engine throughput. Recently, the compute-in-memory (CIM) approach is gaining popularity where computation occurs where the data are stored using massively parallelized analog MAC engines [1]–[5]. For analog MAC engines employed in CIM accelerators, the input data are typically loaded onto multiple memory word lines, generating parallel cell currents that are summed up in a single cycle and converted into digital code [1]–[5]. A CIM accelerator is also suitable if the memory cell can support multilevel storage, which can improve the inference accuracy. In resistive random-access memory (ReRAM), weights are stored in each cell and MAC operations are performed in the analog domain [1]. However, ReRAM-based CIM accelerators are limited to small neural networks that infer with low precision, owing to difficulties in implementation compared to excellent computational efficiency. Interest in static RAM (SRAM)-based CIMs [2] is increasing because they can be fabricated reliably using a standard logic process. However, SRAM cells are larger than the denser cells of one-transistor (1T) or one-transistor-one-resistor (1T1R) memory technologies; hence, they cannot store the large number of trained weights required by CNNs. Recently, 3D NAND flash technology has been proposed, which can obtain a good inference accuracy and overcome density limitations of these CIM accelerators. It allows for the practical implementation of CIMs that can effectively compute CNNs comprising hundreds of millions of parameters [3]–[6].

Generally, the local memories of CIM accelerators are controlled using direct memory access controllers (DMACs) [7]–[10]. The CIM accelerator is a standalone IP connected to the shared memory through an on-chip bus (e.g., advanced extensible interface (AXI) crossbar [11]) [12]–[13]. Dataflow within the CIM accelerator refers to data movement between the local memory and the internal MAC operation of the accelerator [14]–[17]. In contrast, dataflow outside the accelerator refers to data movement between the external shared and internal memories of the accelerator [12]–[13]. Although existing CNN accelerator research [9]–[10], [12]–[13], [18] have adequately considered dataflows, CIM accelerator research has not sufficiently considered system-level. Therefore, determining an optimal design is challenging owing to the lack of system-level performance exploration for several dataflows [3]–[6].

In this study, a system-level simulator, eF^{2}lowSim, was developed to optimize the eFlash CIM accelerator by considering both its computation and external communication performances. The eF^{2}lowSim model based on SystemC transaction-level modeling (TLM) [19] is developed and plugged into and the simulation models for on-chip bus and memory provided by a commercial simulator, the Platform Architect from Synopsys [20]. The eF^{2}lowSim predicts both the inference accuracy and system-level performance. Additionally, the eF^{2}lowSim is used to evaluate various dataflow strategies for the CIM accelerator core and improve the area efficiency achieve.

II. Related Work

In recent years, the CIM approach has been widely studied; it has been proposed to reduce the power consumption and latency in accessing external memory, such as dynamic RAM. Several studies on CIM structures have focused on using nonvolatile memories, such as ReRAM and SRAM [1]–[5]. ReRAM-based CIM structures can perform complex

*Jooho Wang and Sunwoo Kim have equal contribution.

978-3-9819263-7-8/DATE23/© 2023 EDAA

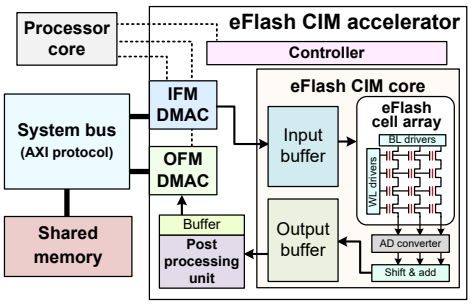

Fig. 1. Overview of the system including the eFlash-based CIM accelerator.

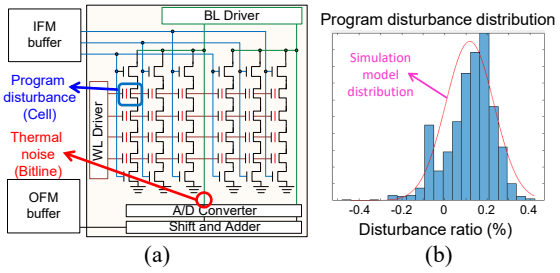

Fig. 2. Noise source modeling: (a) location of noise source in eFlash array and (b) distribution of measured program disturbance from eFlash core implementation.

operations, such as matrix–vector multiplication (MVM), with reduced power consumption by using an array programmed with values of various levels [1]. However, ReRAM has a disadvantage in that it is difficult to fabricate using the standard logic process; therefore, its application to large neural networks is challenging. Although SRAM-based CIM structures can be implemented via the standard logic process [2], their memory cell area per unit capacity is relatively large. In contrast, flash memory is advantageous in that it has a smaller area per unit capacity than most other nonvolatile memories, and employing the multilevel cell technique in flash memories maximizes their area efficiency [3], [4]. Moreover, embedded flash (eFlash) memory can be fabricated using the standard logic process, which makes it even more beneficial in terms of implementation simplicity and hardware area [21]. Owing to these advantages, several recent studies have focused on eFlash-based CIM structures [3]–[6].

However, existing CIM structure studies primarily focus only on performance analysis and the internal structure of CIM accelerators. Moreover, studies that consider blocks other than CIM accelerators, such as on-chip bus and shared memory, do not include performance analysis or dataflows outside the accelerators [7]–[10]. Additionally, any of the existing studies on eFlash-based CIMs [3]–[6] does not deal with the system-level structure, making it difficult to determine whether the performance of CIM accelerators is limited by the on-chip bus and shared memory.

In this study, the eF^{2}lowSim can help to evaluate the system-level performance considering the data communication. In addition, the eF^{2}lowSim predicts the impact of noise sources in eFlash CIM core to estimate the inference accuracy of the accelerator.

The remainder of this paper is organized as follows. Section III introduces the system considered in this study and details the configuration of the eF^{2}lowSim. Section IV describes some of the dataflow strategies adopted in this study and the design space, which comprises a combination of these dataflow strategies. Section V presents the performance and area simulation results of the CIM accelerator for various combinations of dataflow strategies. Finally, Section VI concludes the paper.

III. SYSTEM-LEVEL SIMULATOR

A. System Under Consideration

The system structure, including the eFlash CIM accelerator, used in this study is shown in Fig.1. The system assumed in this study adopts a structure generally implemented in existing studies for a CIM accelerator [7],[8] and a logic-based hardware accelerator [9], [12]–[13]. As illustrated in the figure, the shared memory and eFlash CIM accelerator are connected to the on-chip bus to load the input feature map (IFM) from the shared memory or store the output feature map (OFM) in the shared memory. The eFlash CIM accelerator, which communicates data to the system bus, includes a DMAC that complies with the bus protocol and drives data delivery, which is controlled by the processor core [12]–[13]. The processor core controls the DMAC to direct the movement of feature maps and give commands to the accelerator controller, which provides information to control the internal eFlash CIM core of the accelerator such that it fits well into the target neural network [12]–[13].

B. Modeling of eFlash CIM Core

The eFlash CIM core for CNNs performs a MAC operation with the IFM when it reads the programmed weight. In this case, the bit serial operation can support unlimited data bit widths by repeating the shift and add functions [3]. A cell current proportional to the weight stored in the flash cell is added to the bitline, which uses an analog-to-digital converter (ADC) to convert it into digital values.

An eFlash cell array has several sources of noise. Program disturbances cause errors in the charge amount injected into flash cells during flash programming. Additionally, thermal noise may cause errors in the analog signals containing cell and bitline currents, which can result in unwanted computational results [3]–[6]. The proposed simulator can model these two noise sources by using additive white Gaussian noise that appears in a normal distribution. In particular, the eFlash-specific program disturbance was obtained from the measurement results of the cell currents in the eFlash cell arrays of existing study [21].

Fig. 2 shows the location to which the noise sources of the proposed simulator are applied. Program disturbance may change the weight values stored in flash cells that persist before being newly programmed, and the proposed simulator models these weight value variations as shown in Fig. 2(b). The more flash cell currents that are aggregated on the bitline, the greater the impact of the modeled program disturbance. In the case of thermal noise, the noise source acts is directly in front of the ADC, since the additive Gaussian noise, is applied on the same basis for the entire analog circuit. Consequently, by considering the noise sources, the computational results

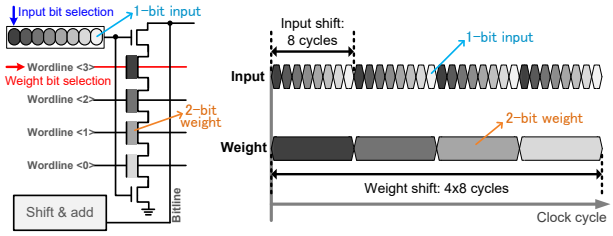

Fig. 3. Bit serial operation: Input and weight bit selection on an eFlash string (left) and the timing diagram of the bit selection sequence (right).

or inference accuracy, as well as the performance of the eFlash CIM core can be predicted. Fig. 3 illustrates the bit-serial operation for 8-bit input and 8-bit weight modeled in the proposed simulator. Since only one bit of input can be processed per cycle by enabling or disabling the cell string, the input bit selection must be shifted eight times for 8-bit input pixels. However, assuming that each flash cell is programmed with multiple bits of weight, it is not necessary to shift the weight bit selection eight times for an 8-bit weight. As shown in the right side of Fig. 3, it takes eight cycles to multiply an 8-bit input pixel by a 2-bit weight. In addition, this sequence is repeated four times, and multiplication of an 8-bit weight takes a total of 32 cycles.

C. Virtual Platform Simulator

System-level performance, which should be predicted to optimize the accelerator, is affected not only by the accelerator but also by the on-chip bus and shared memory. A SystemC/TLM-based virtual platform simulator, which supports accurate prediction and exploration of system-level performance, can be used to model the external communication bandwidth required by the accelerator [12]–[13]. Several bus and memory models are required to maximize the advantages of virtual platform simulations. Therefore, we developed an eF^{2}lowSim that includes bus and memory libraries using the Platform Architect (Synopsys) [20].

Fig. 4 shows the eFlash CIM accelerator structure modeled as a SystemC/TLM module in the eF^{2}lowSim. The IFM DMAC shown in the figure receives the IFM from the bus and enters it into the accelerator, which is then stored in an input buffer (IBUF) sequentially. After loading the IFM, the accelerator controller sends a control signal such that the eFlash CIM core reads the IFM from IBUF, and the internal eFlash CIM core performs the MAC operation. The eFlash CIM core model comprises hierarchical structures, which perform MVM to compute the convolutional layers. After the eFlash CIM core completes the operation for the loaded IFM and stores the computed OFM in output buffer 1 (OBUF1), the post processing unit (PPU) receives a control signal. Subsequently, the PPU performs nonlinear functions (e.g., rectified linear unit and pooling layer) and stores the processed OFMs in OBUF2. Finally, the OFM DMAC forwards the resulting data to the shared memory through the on-chip bus. The source code is available on the GitHub repository: https://github.com/SDL-KU/eF2lowSim.

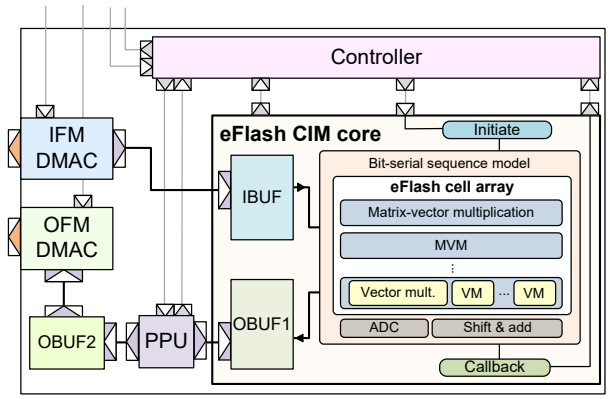

Fig. 4. Internal structure of CIM accelerator model in the eF^{2}lowSim.

IV. Dataflow Outside Accelerator

A. CNN Computation in eFlash CIM core

Fig. 5 (a) shows a visualization of convolutional layer and nonlinear layer operation. After the convolutional layer, activation functions, such as the pooling operations, are applied to introduce nonlinearity. Fig. 5 (b) shows the mapping scheme and computation of the convolutional layer in the eFlash cell array. As shown in figure, an IFM set (i.e., R×S×C) is input to an eFlash cell array to which all filter weights (i.e., R×S×C×M) are mapped. Thereafter, the MVM operations are performed inside the eFlash cell array to output M number of OFMs. In general, the size of OFM is E × F × M; therefore, MVM operations are repeated EF times to output all OFMs [3]–[6].

B. Dataflow Strategies

This subsection introduces five dataflow strategies between a CIM accelerator connected to the on-chip bus and external shared memory. As shown in Fig. 6 (a), dataflow strategy 1 loads all IFMs (i.e., H × W × C) from external shared memory into the IBUF of the eFlash CIM accelerator [3]–[5], [22]. After CIM accelerator loads all IFMs from external memory, IFM Set (i.e., R × S × C) is input to CIM core and the OFM of performing MVM operation EF times is stored in OBUF1. Thereafter, a nonlinear operation is performed on all OFMs at the PPU; they are then stored in OBUF2 and transmitted to the external shared memory of the accelerator through DMAC. In this dataflow strategy, the overheads of the DMAC and on-chip bus have a negligible effect on performance. However, as shown in Figure 6, this dataflow strategy has the disadvantage in that a sufficiently large buffer size is required to store large numbers of IFMs and OFMs.

As illustrated in Fig. 6 (b), unlike dataflow strategy 1, dataflow strategy 2 involves repeatedly loading IFM sets (i.e., R × S × C) from external shared memory of eFlash CIM accelerators and performing MVM computation [14], [23]. This dataflow strategy requires less IBUF capacity than the dataflow strategy 1. However, it may degrade the performance of the eFlash CIM accelerator system owing to the higher number of memory accesses to the on-chip bus through IFM DMAC. Unlike the aforementioned dataflow strategies, the dataflow strategy 3 illustrated in Fig. 6 (c) aims to perform the nonlinear

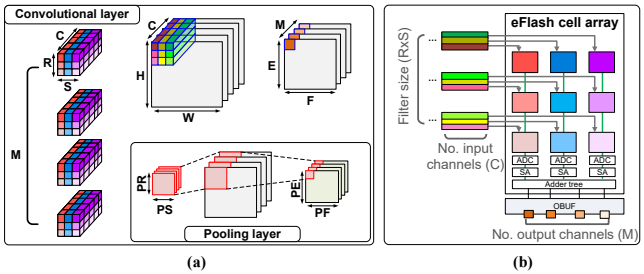

Fig. 5. Workload mapping for CNN: (a) Convolutional and pooling layer structure and (b) the corresponding mapping scheme.

process as fast as possible. In detail, dataflow strategy 3 is different from dataflow strategy 1 and 2, and when OFM capable of performing one nonlinear operation is stored in OBUF1, the results after the nonlinear operation are immediately transmitted to the external shared memory through DMAC. Thus, this dataflow strategy can be more effective in terms of memory size than the previous ones. However, this dataflow strategy tends to reduce overall performance because DMAC is frequently used compared to previous dataflow strategies.

As shown in Fig. 6 (d), the dataflow strategy 4 involves loading an IFM set extended to pooling size to the IBUF to perform convolutional and nonlinear operations effectively. Although this dataflow strategy requires an IBUF with a slightly larger memory capacity compared to the dataflow strategy 3, the memory capacities of OBUF1 and 2 required for storing OFMs are the same. However, this dataflow strategy can improve performance because the number of accesses to the external memory through DMAC of the eFlash CIM accelerator required to load the IFM is lower than that in the dataflow strategy 3.

As shown in Fig. 6 (e), dataflows strategy 5 introduces a method of reducing the number of memory accesses through IFM and OFM extended in column-wise compared to dataflow strategy 3. This dataflow strategy requires a larger local memory capacity for the IBUF and OBUFs than the dataflow strategy 3 and 4. However, the dataflow strategy 5 has fewer memory accesses than the data flow strategies 3 and 4. Thus, it can improve the performance of overall system by reducing on-chip bus overhead.

Table I shows the required buffer capacity when performing convolutional layer 1 and pooling layer 1 of LeNet-5 [15] for each dataflow strategy and the number of DMAC bursts when the burst length is set to 8. Within an external shared memory, the data layout assigns the pixels of a feature map to the memory locations in a row-major order. Recalling that it is more efficient to occupy with as consecutive reads or writes as possible from the viewpoint of DMA accesses, it is reasonable to determine data layout in row-major order [13]. As shown in the table, different buffer capacity and burst count are required for each dataflow strategy, and it is difficult to predict the system performance considering these. In the following section, we propose an optimal dataflow strategy combination in terms of area efficiency by applying five dataflow strategies to several neural networks through the eF^{2}lowSim.

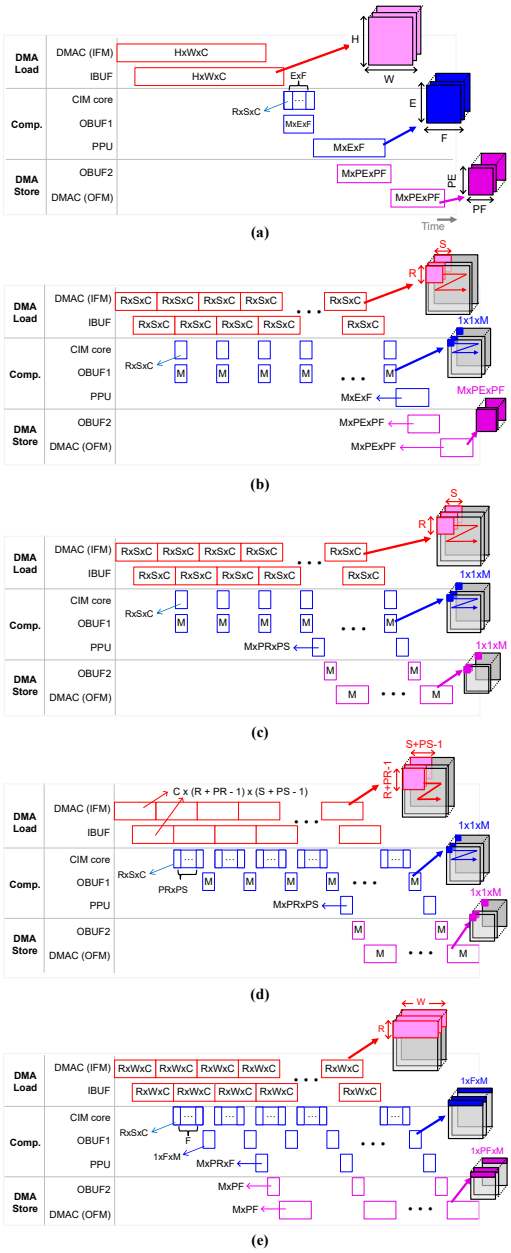

Fig. 6. Timing diagram of (a) dataflow strategy 1, (b) dataflow strategy 2, (c) dataflow strategy 3, (d) dataflow strategy 4 and (e) dataflow strategy 5.

V. Simulation Results

In this section, the proposed bit-accurate and cycle-accurate simulator, eF^{2}lowSim, is used to evaluate the inference accuracy and system-level performance of the eFlash CIM accelerator. Additionally, this section describes the area efficiencies of the proposed dataflow strategies when they are applied to each convolutional layer of several neural networks. LeNet-5 [15], VGG-9 [16], and ResNet-18 [17] were used in the simulations. For example, Table II shows the measured memory area according to the dataflow strategies in the first convolutional layer of LeNet-5 with CMOS standard 65-nm cell library. The eFlash CIM core fabricated with CMOS standard 65-nm cell library [3]–[5] operates at the clock frequency of equal or less

TABLE I
BUFFER CAPACITY AND THE NUMBER OF DMAC BURSTS FOR EACH DATAFLOW STRATEGY.

Buffer capacity [bits]					
Dataflow strategy	1	2	3	4	5
IBUF	6,272	200	200	288	1120
OBUF1	27,648	27,648	192	192	2,304
OBUF2	6,912	6,912	48	48	576
No. of DMAC bursts					
IFM DMAC	25	576	576	288	96
OFM DMAC	27	27	144	144	36

TABLE II
HARDWARE AREA ACCORDING TO DATAFLOW STRATEGY.

Hardware area (65-nm cell library) [um^2]					
Dataflow strategy	1	2	3	4	5
IBUF	95,822	3,056	3,056	4,397	17,110
OBUF1	422,403	422,403	2,932	2,932	35,201
OBUF2	105,603	105,603	733	733	8,800
eFlash CIM CORE [5]	2,350				
Total	626,178	533,412	9,071	10,412	63,461

than 20 MHz. However, hardware blocks connected to hardware accelerators are often assumed to operate at the clock frequency of 200 MHz with the same scale of cell library [18], [24]. Therefore, it is reasonable to assume that the eFlash CIM core operates ten times as slow as the other blocks in the system. The simulation results in this section showed that each network has different combinations of dataflow strategies that obtain the optimal area efficiency.

A. Inference Accuracy of eFlash-based CIM

As mentioned in the Section III.B, the proposed simulator reflects the circuit nonideality of the eFlash cell array by modeling several noise sources. Through this modeling, the accelerator model in the simulator can be used to experiment when unintended errors occur owing to noise inside the circuit. Although existing CIM studies have considered noise sources, they only assumed that their effects were fundamentally blocked [3]–[5]. However, this study also considers the cases where the simulator obtains incorrect results owing to noise inside the cell array.

Noise sources that have a significant impact on inference accuracy are program disturbances and quantization errors. In this study, we modeled the program disturbance bias, which varies with the program method [3]–[5]. Fig. 7 shows the experimental results on LeNet-5 when the bit widths of both IFM and weight are 4-bit and 8-bit. When 8-bit data were applied, inference accuracies of 96.8% and 76.7% were achieved when the disturbance bias was 1% and 5% of the maximum cell current, respectively. Additionally, when 4-bit data were applied, inference accuracies of 96.1% and 69.0% were achieved when the disturbance bias was 1% and 5% of the maximum cell current, respectively. It is shown that the simulator can reflect the impact of the noise and the simulation results imply that the eFlash CIM accelerator achieves sufficient inference accuracy with the program disturbance measured from the fabricated eFlash cell array [21].

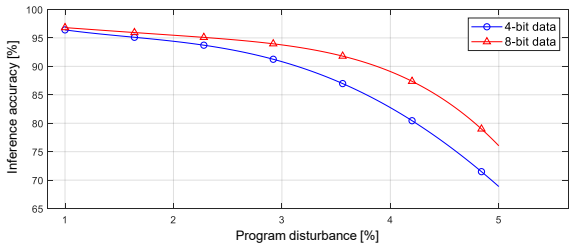

Fig. 7. Inference accuracy according to program disturbance.

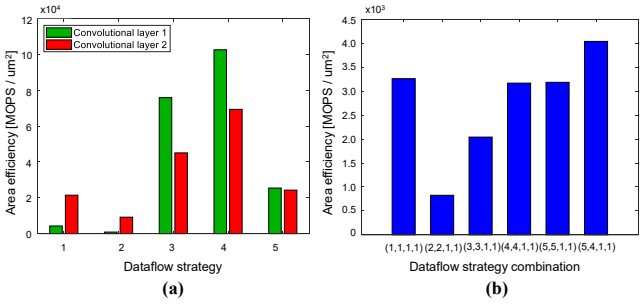

Fig. 8. (a) Hardware area efficiency of single convolutional layer according to dataflow strategy for LeNet-5 and (b) extended to entire network according to dataflow strategy combination.

B. System Performance Evaluation in LeNet-5

The optimal dataflow strategy must be determined by considering the impact of the buffer, on-chip bus, and external shared memory of the eFlash CIM accelerator and eFlash CIM core. Therefore, we developed an eF^{2}lowSim to predict system-level performance, which is affected by both the layer shape and the employed dataflow strategy. Furthermore, the IBUF capacity of the eFlash CIM accelerator varies with the dataflow strategy, which is considered for predicting the area efficiency because it changes the overall area of the eFlash CIM accelerator.

Fig. 8 (a) shows the area efficiency of each layer according to the dataflow strategy for LeNet-5 that received the MNIST dataset as input. It is found that dataflow strategy 4 shows the best area efficiency for both the first and second convolutional layers of LeNet-5. Fig. 8 (b) shows that the area efficiency for LeNet-5 can be explored by employing different strategies for each layer. Each element of x-axis in the figure denotes the combination of dataflow strategies for the first and second convolutional layers and the first and second fully-connected layers. The simulation results show that applying the dataflow strategy showing the best area efficiency for a single layer to the entire layer does not guarantee the best area efficiency. For example, among all combinations, the optimal strategy combination is (5, 4, 1, 1), which yields a 26.8% improvement in area efficiency compared to (4, 4, 1, 1). When both of convolutional layers using the same dataflow strategy, the second convolutional layer requires a larger buffer capacity than the first convolutional layer. Therefore, in the first convolutional layer in LeNet-5, it is possible to improve the area efficiency of the entire network by using dataflow strategy 5 with a larger area and better performance, rather than dataflow strategy 4 with the best area efficiency in a single convolutional layer.

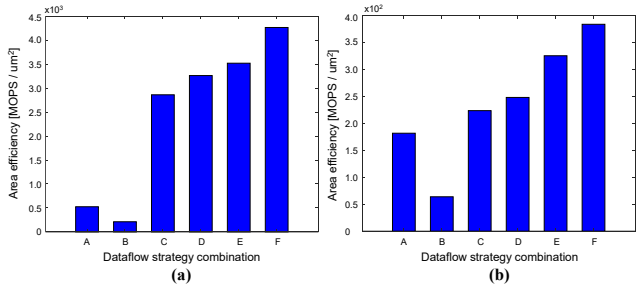

Fig. 9. Hardware area efficiency according to dataflow strategy combination for (a) VGG-9 and (b) ResNet-18.

C. Extension to Other Neural Networks

Fig. 9 shows the results of design space exploration for area efficiency according to the dataflow strategy combinations for VGG-9 and ResNet-18. In Fig. 9, Design A, Design B, Design C, Design D, and Design E refer to the collective application of dataflow strategies 1, 2, 3, 4, and 5 to all convolutional layers, respectively. Design F refers to a combination of dataflow strategies with optimal area efficiency found through design spaces exploration of the proposed simulator. Fig. 9 (a) and (b) show that the area efficiency of design F is improved by at least 21.2% and 17.9% compared to other designs for VGG-9 and ResNet-18, respectively. These optimal designs are chosen among the number of cases over 10 thousands of designs and the system-level simulator is crucial for this optimization process. Thus, the exploration using the simulator is more effective than empirical design selection for optimizing dataflow strategies for entire networks.

VI. Conclusion

In this study, a system-level simulator, eF^{2}lowSim, is developed to accurately estimate the performance of eFlash CIM accelerators considering the dataflow strategy and the circuit nonideality of eFlash cell array. The eF^{2}lowSim can model the noise source to measure inference accuracy according to the noise sources in the eFlash cell array. In addition, using the eF^{2}lowSim, it is possible to find the optimal design of an eFlash CIM accelerator by evaluating the system-level performance of different dataflow strategies for convolutional layers. The experimental results show that the improvement of area efficiency from the optimal design amounts to 26.8%, 21.2% and 17.9% in the case of LeNet-5, VGG-9 and ResNet-18, respectively. The future works include the extension of the eF^{2}lowSim for the multi-core eFlash CIM accelerators.

Acknowledgment

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No. 2021-0-01764). The authors would like to express their sincere thanks to Dr. S. Song of SEMIBRAIN Inc. for helpful suggestion and discussion. The authors would also like to express their sincere thanks to Dr. M. Kim and Prof. C. H. Kim of University of Minnesota for providing measurement data from the fabricated chips.

References

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

[2] X. Si et al., "A dual–split 6T SRAM–based computing–in–memory unitmacro with fully parallel product–sum operation for binarized DNN edge processors," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 66, no. 11, pp. 4172–4185, Nov. 2019.

[3] M. Kim et al., "A 3D NAND flash ready 8–bit convolutional neural network core demonstrated in a standard logic process," in IEDM Tech. Dig., pp. 38.3.1–38.3.4, Dec. 2019.

[4] M. Kim et al., "A 68 parallel row access neuromorphic core with 22K multi–level synapses based on logic–compatible embedded flash memory technology," in IEDM Tech. Dig., pp. 15.4.1–15.4.4, Dec. 2018.

[5] M. Kim et al., "An embedded nand flash–based compute–in–memory array demonstrated in a standard logic process," in IEEE Journal of Solid–State Circuits, pp. 625–638, Feb. 2022.

[6] H. –T. Lue et al., "Introduction of 3D AND–type flash memory and its applications to computing–in–memory (CIM)," 2021 International Symposium on VLSI Technology, Systems and Applications (VLSI–TSA), 2021, pp. 1–2.

[7] H. Jia et al., "A programmable heterogeneous microprocessor based on bit–scalable in–memory computing," in IEEE Journal of Solid–State Circuits, vol. 55, no. 9, pp. 2609–2621, Sept. 2020.

[8] A. Siemieniuk et al., "OCC: An automated end–to–end machine learning optimizing compiler for computing–in–memory," in IEEE Transactions on Computer–Aided Design of Integrated Circuits and Systems, vol. 41, no. 6, pp. 1674–1686, June 2022.

[9] C. Zhang et al., "Optimizing FPGA–based accelerator design for deep convolutional neural networks," In Proc. Intl. Symp. on Field–Programmable Gate Arrays (FPGA), pp. 161–170.

[10] T. Chen et al., "DianNao: A small–footprint high–throughput accelerator for ubiquitous machine–learning," in Proc. ACM Int. Conf. Archit. Support Program. Lang. Oper. Syst. (ASPLOS), 2014, pp. 269–284.

[11] ARM, AMBA AXI and ACE protocol specification, AXI3, AXI4, and AXI4–Lite, ACE and ACE–Lite, ARM Infocenter, 2011.

[12] S. Kim, S. Park, and C. S. Park, "System–level communication performance estimation for DMA–controlled accelerators," IEEE Access, vol. 9, 2021, pp. 141389–141402.

[13] J. Wang, S. Park, and C. S. Park, "Optimization of communication schemes for DMA–controlled accelerators," IEEE Access, 2021, vol. 9, pp. 139228–139247.

[14] D. T. Nguyen et al., "ShortcutFusion: From tensorflow to FPGA–based accelerator with a reuse–aware memory allocation for shortcut data," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 69, no. 6, pp. 2477–2489, June 2022.

[15] Y. LeCun et al., "Backpropagation applied to handwritten zip code recognition," in Neural Comput., vol. 1, no. 4, pp. 541–551, Dec. 1989.

[16] K. Simonyan and A. Zisserman, Very deep convolutional networks for large–scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

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

[19] OSCI, Open SystemC Initiative (OSCI), TLM–2.0 User Manual.

[20] Synopsys, Platform Architect, [Online]. Available: http://www.synopsys.com/Tools/SLD/VirtualPrototyping/ Pages/PlatformArch

[21] M. Kim, J. Song, and C. H. Kim, "Reliability characterization of logic–compatible NAND flash memory based synapses with 3–bit per cell weights and 1A current steps," IEEE Intl. Reliability Physics Symposium (IRPS), pp. 1–4, Apr. 2020.

[22] Z. Jiang et al., "Vesti: An in–memory computing processor for deep neural networks acceleration," Asilomar Conference on Signals, Systems, and Computers, 2019, pp. 1516–1521.

[23] H. E. Yantr, A. M. Eltawil, and K. N. Salama, "IMCA: An efficient in–memory convolution accelerator," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 29, no. 3, pp. 447–460, Mar. 2021.

[24] J. Yue et al., "A 3.77TOPS/W convolutional neural network processor with priority–driven kernel optimization," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 66, no. 2, pp. 277–281, Feb. 2019.