Pipeline Design of Nonvolatile-based Computing in Memory for Convolutional Neural Networks Inference Accelerators

Lixia Han, Peng Huang*, Zheng Zhou, Yiyang Chen, Haozhang Yang, Xiaoyan Liu, Jinfeng Kang
School of Integrated Circuits, Peking University, Beijing 100871, China
Beijing Advanced Innovation Center for Integrated Circuits, Beijing 100871, China
Email: *phwang@pku.edu.cn

***Abstract* — Nonvolatile-based computing-in-memory inference chips show great potential to accelerate convolutional neural networks. The intrinsic weight stationary characteristic makes pipeline design a crucial solution to further enhance throughput. In this work, we propose a balanced pipeline design and establish performance/area evaluation models for the optimal pipeline solution. The evaluation results indicate that our pipeline design achieves 30× computational efficiency improvement.**

1. Introduction

Convolutional neural networks (CNNs) have gained widespread application in a variety of visual tasks. The implementation of computation-intensive CNNs necessitates processors with higher computational efficiency and lower power consumption. A critical impediment to enhancing energy efficiency and throughput in current processing architectures is the data movement between processing and memory units, commonly referred to as the von Neumann bottleneck. Nonvolatile-based computing-in-memory (nvCIM) inference accelerators have emerged as efficient solutions, leveraging the capability of in-situ high parallel computation in nonvolatile memory. Extensive research on nvCIM macros[1, 2], architectures [3, 4], and system demonstrations [5, 6] have confirmed the potential of nvCIM chips to achieve 10~1000× throughput improvement than von Neumann processors.

The pre-programmed weights in nvCIM chips enforce a global weight stationary approach during CNN inference, thereby limiting dynamic resource allocation for accelerating the specific CNN layers. To enhance throughput, introducing pipeline design into nvCIM CNN inference accelerators is imperative. Pipeline designs in [4, 7] face challenges of unbalanced latency across layers and significant buffer demands for activations. Therefore, a balanced pipeline design and performance/area evaluation models are desired. The following pipeline tradeoffs should be considered: firstly, while deeper pipeline yields throughput benefits, it necessitates additional buffers for storing intermediate data. Secondly, pipeline design suffers from unbalanced challenge due to the disparities in computational load across different pipeline stages. To speed up bottleneck stage and compress the hardware area of non-bottleneck stage are essential.

2. Balanced Pipeline Design

In this work, we propose a balanced pipeline design for nvCIM-based CNN inference accelerators. Taking a block in DenseNet as an example, Fig. 1(a) shows the performance and hardware area of baseline (sequential execution). The total latency is the sum of the delays of all CNN layers.

Building a balanced pipeline initially involves arranging the pipeline depth. Throughout the entire CNN execution process, we evenly divide the total latency into N stages and add new pipeline stage after the nearest CNN layer (named the beginning layer of new stage as $S(layer)$). The latency and area of N-stage pipeline are shown in Eq. (1).

$$Latency = \max\left(\sum_{layer=1}^{l_1} T(layer), \cdots, \sum_{layer=l_M}^{l_N} T(layer)\right) \tag{1}$$

$$Area = \sum_{layer=1}^{L} W(layer)\cdot K_1(layer) + \sum_{layer=1}^{L} A(layer)\cdot K_2(layer)$$

The total latency is the maximum delay $\sum T(layer)$ of all pipeline stages. The area of pipeline is the sum of the weight $W(layer)$ area and the sum of the stored activation $A(layer)$ area. The weight is stored in nonvolatile memory and the activation is stored in buffer. $K_1(layer)$ and $K_2(layer)$ respectively represents the quantities of programmed weights and stored activations in the pipeline design. As illustrated in Fig. 1(b), pipeline new stage begins after the *1st* layer and the *3rd* layer. Considering the interlayer dependence induced by the CNN topology. the $K_2(l_1)$ is three (activations A~C) and $K_2(l_2)$ is two (activations A~B) to ensure sufficient data for pipeline execution.

For the bottleneck layers of pipeline, the weights duplication strategy is adopted to speed up these layers (named the layer with duplicated weights as $D(layer)$). In Fig. 1(c), for the *1st* layer as the bottleneck layer, the weights of the *1st* layer are duplicated ($K_1(l_1)$ is two) to reduce the total latency to half.

For the non-bottleneck layers of pipeline, the activation recomputing strategy is adopted to alleviate the hardware area (named the layer with recomputing activations as $R(layer)$. In Fig. 1(d), we recompute the activation of the *3rd* layer ($K_2(l_2)$ is reduced to one) rather than directly store the activation. Although the delay of the *3rd* layer is double, the total latency is not increased.

After modeling of the pipeline strategies, proper solving algorithms are utilized to achieve the optimal pipeline.

979-8-3503-4859-0/DATE24/© 2024 EDAA

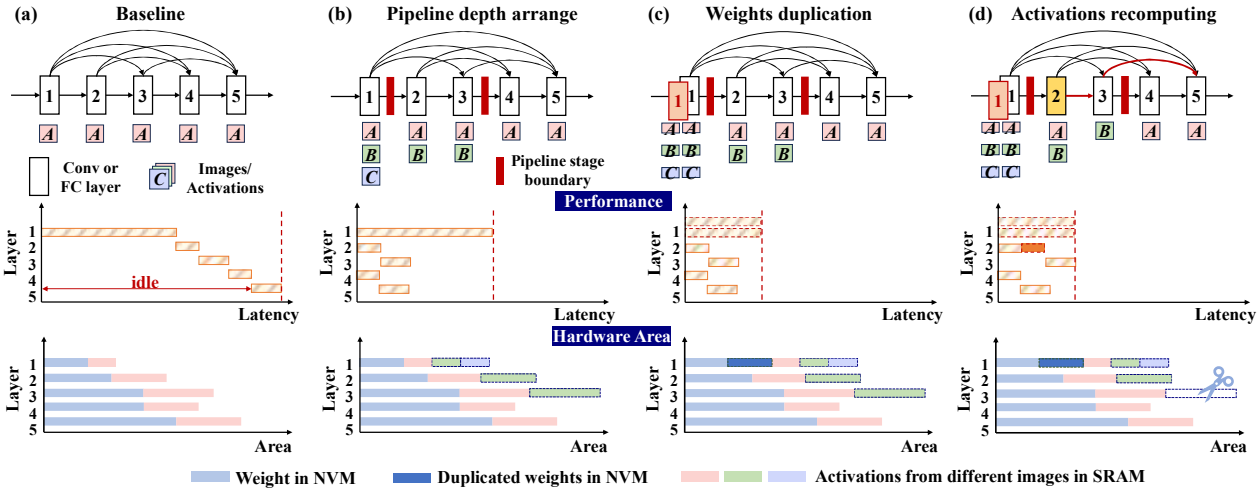

Fig. 1. Overview of the proposed pipeline design for nvCIM-based CNN inference accelerators. (a) baseline; (b) pipeline depth arrange; (c) weights duplication strategy; (d) activations recomputing strategy.

3. Evaluation Results

We establish criterion for pipeline optimization to maximize the computational efficiency *CE*, as shown in Eq. (2). Optimizing the nvCIM-based CNN pipeline involves identifying the pipeline stage beginning layer *S*(*layer*), the weight duplication layer *D*(*layer*), and the activation recomputing layer *R*(*layer*) to minimize 1/*CE* value.

$$1/CE = Latency \cdot Area\big(S(layer), D(layer), R(layer)\big)$$
$$\big(S(layer), D(layer), R(layer)\big) = arg\ min\big(1/CE\big) \quad (2)$$

Taking DenseNet121 as an example, we demonstrate the effectiveness of the proposed pipeline strategies. In this study case, the nvCIM chip employs the ISAAC architecture [4], which is consists of tiles. Each tile is comprised 64 RRAM-based macros [6]. The macros within each tile share a 16KB SRAM buffer for storage of CNN activations.

Fig. 2 illustrates the normalized 1/*CE* with the pipeline depth. If no pipeline design is adopted in the nvCIM chip, the normalized 1/*CE* is 30. With the pipeline depth arrangement, a minimum 1/*CE* value of 4.7 is attained at 7-stage pipeline design. Following the implementation of weight duplication strategy, a significantly reduced 1/*CE* value of 1.4 is achieved at an increased pipeline depth of 35. The employment of an activation recomputing strategy allows for further reduction of the 1/*CE* value to 1 without altering the pipeline depth.

4. Convolutions

In this work, we present a balanced pipeline design for nvCIM-based CNN inference accelerators. We explore the pipeline depth and propose strategies of weight duplication and activation recomputing to achieve optimal performance at the lowest hardware area.

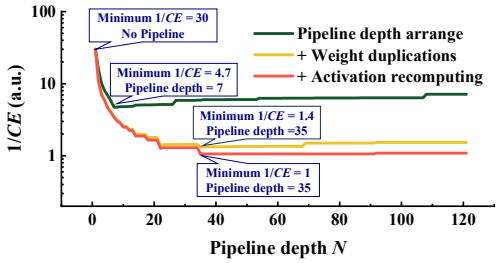

Fig. 2. The 1/*CE* (computational efficiency) evolution with the pipeline depth in nvCIM-based CNN inference accelerators.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 62034006 and Grant 92064001, and in part by the National Key Research Plan of China under Grant 2023YFB4402400.

References

[1] W.-S. Khwa *et al.*, "A 40-nm, 2M-Cell, 8b-Precision, Hybrid SLC-MLC PCM Computing-in-Memory Macro with 20.5-65.0 TOPS/W for Tiny-Al Edge Devices," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, 2022, vol. 65: IEEE, pp. 1-3.

[2] M. Le Gallo *et al.*, "A 64-core mixed-signal in-memory compute chip based on phase-change memory for deep neural network inference," *Nat. Electron.,* vol. 6, no. 9, pp. 680-693, 2023.

[3] X. Qiao *et al.*, "AtomLayer: A universal ReRAM-based CNN accelerator with atomic layer computation," in *Proceedings of the 55th Annual Design Automation Conference*, 2018, pp. 1-6.

[4] A. Shafiee *et al.*, "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *ACM SIGARCH Computer Architecture News,* vol. 44, no. 3, pp. 14-26, 2016.

[5] S. Ambrogio *et al.*, "An analog-AI chip for energy-efficient speech recognition and transcription," *Nature,* vol. 620, no. 7975, pp. 768-775, 2023.

[6] W. Wan *et al.*, "A compute-in-memory chip based on resistive random-access memory," *Nature,* vol. 608, no. 7923, pp. 504-512, 2022.

[7] X. Peng *et al.*, "Optimizing weight mapping and data flow for convolutional neural networks on processing-in-memory architectures," *IEEE Trans. Circuits Syst. I Regul. Pap.,* vol. 67, no. 4, pp. 1333-1343, 2019.