OpenC^2: An Open-Source End-to-End Hardware Compiler Development Framework for Digital Compute-in-Memory Macro

Tianchu Dong, Shaoxuan Li, Yihang Zuo, Hongwu Jiang, Yuzhe Ma, and Shanshi Huang†
Microeletronics Thrust, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China
{tdong351, slieu, yzuo099}@connect.hkust-gz.edu.cn, {hongwujiang, yuzhema, †shanshihuang}@hkust-gz.edu.cn

***Abstract*—Digital Compute-in-Memory (DCIM), which inserts logic circuits into SRAM arrays, presents a significant advancement in CIM architecture. DCIM has shown great potential in applications, and the diversity of applications requires rapid hardware iteration. However, the hardware design flow from user specifications to layout is extremely tedious and time-consuming for manual design. Commercial EDA tools are limited by restrictive licenses and the inability to specifically optimize the datapath, which calls for an open-source end-to-end hardware compiler for DCIM. This paper proposes OpenC^2, the first open-source end-to-end development framework for DCIM macro compilation. OpenC^2 provides a template-based generation platform for DCIM macros across various technologies, sizes, and configurations. It can automatically generate a datapath-optimized, compact DCIM macro layout based on a hierarchical physical design methodology. Our experiment results show that OpenC^2's compact design on FreePDK45 delivers over 30% area reduction and over 40% improvement in area efficiency compared to AutoDCIM on TSMC40.**

I. Introduction

The success of deep learning (DL) depends on both algorithm advances and hardware development. The conventional Von-Neumann architecture faces the "memory wall" problem, leading to the popularity of CIM techniques. DCIM [1], a type of CIM architecture inserting logic circuits into SRAM arrays, has shown impressive hardware performance improvement in accelerating vector-to-matrix multiplications (VMMs) in DL algorithms. It offers enhanced accuracy and robustness compared to traditional analog CIM approaches. However, current DCIM designs are highly dependent on time-consuming manual efforts. The first end-to-end compiler for DCIM, AutoDCIM [2], has limitations such as the privacy and the dependence on commercial EDA tools. Therefore, this work proposes OpenC^2 to provide an open-source customizable compiler, and serve as a baseline for developing DCIM macro compilers.

The main contributions of this work are as follows:

- OpenC^2 is the first open-source end-to-end framework that can generate the front-end netlist (in VERILOG and SPICE format) and the back-end layout (in DEF and GDSII format) of the DCIM macro from top-level specifications. It provides reference circuits and physical implementations in 45nm FreePDK and uses only open-source tools, which can be ported to commercial PDKs or EDA tools.
- It offers a fine-grained floorplan template and hierarchical implementation for the physical design of the DCIM macro, balancing runtime and optimism, and providing optimization opportunities for different modules.
- Experimental results show that OpenC^2's compact physical design methodology on FreePDK45 can achieve a 30% reduction in area and an improvement in area efficiency of over 40% compared to AutoDCIM on TSMC40.

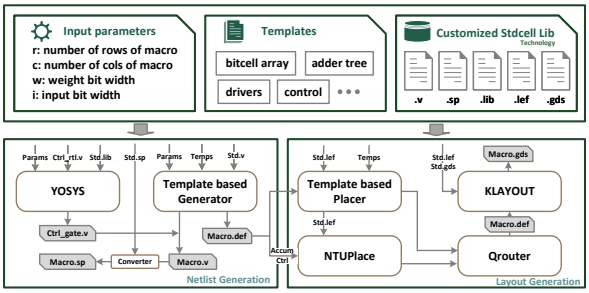

Fig. 1 Overall Flow of OpenC^2

II. OpenC^2 Framework

A. Overview

Fig. 1 depicts an overview of the OpenC^2 framework. The input of the framework consists of user specifications, technology files, and a standard cell library with customized cells (DCIM bitcell, SRAM r/w circuit, and adder for sign extension [1]). It automatically generates design files for front-end netlist (in VERILOG and SPICE) and back-end layout (in DEF and GDSII).

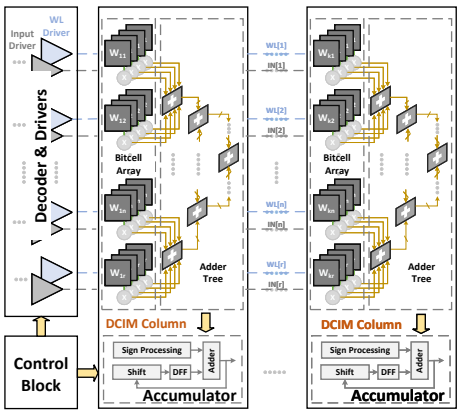

Fig. 2 DCIM Floorplan of OpenC^2

978-3-9826741-0-0/DATE25/© 2025 EDAA

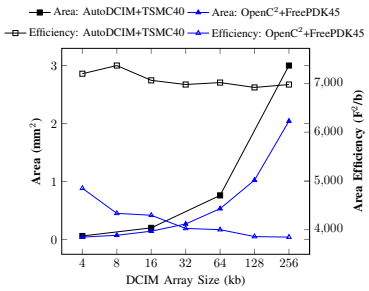

Fig. 3 Area Comparison with AutoDCIM

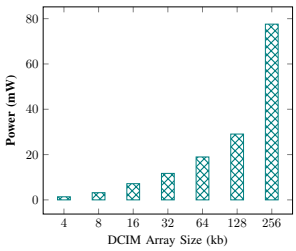

Fig. 4 Power Analysis

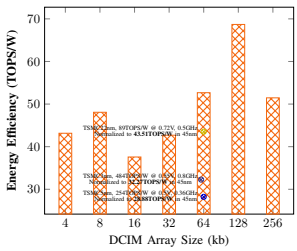

Fig. 5 Energy Efficiency Analysis

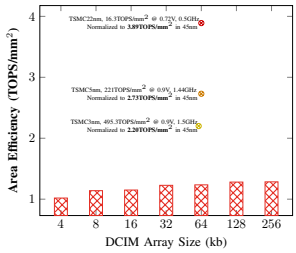

Fig. 6 Area Efficiency Analysis

B. Netlist Generation

Under the guidance of a clear datapath, the structures of sub-modules in DCIM macro are quite regular, except for the control block. Therefore, The netlists in VERILOG and SPICE formats of these modules could be efficiently generated with a template-based flow. Due to the complex timing relationships inside the control block, OpenC^2 only provides an RTL template. The gate-level netlist is then generated from logic synthesis, for which OpenC^2 embraces an open-source tool named YOSYS [3].

C. Hierarchical Physical Design

1) Floorplan

As illustrated in Fig. 2, OpenC^2 employs a highly efficient datapath-based floorplanning strategy.

2) Hierarchical Physical Design Approach

In the context of DCIM macro which features a distinct datapath and numerous identical modules, the strategy of dividing them into sub-modules for individual placement and routing, followed by their reassembly, can significantly enhance the efficiency of the physical design process.

For the initial phase of physical design, OpenC^2 creates the layout for the 7 sub-modules based on the standard cell library. The accumulator and control block are placed using the NTUPlace algorithm [4], whereas the placement of other sub-modules is optimized based on the datapath. The routing of these sub-modules is facilitated by the open-source tool Qrouter [5]. During the physical design process, OpenC^2 primarily utilizes easily editable files in DEF and LEF format. With the assistance of the open-source layout editing tool, KLAYOUT [6], it becomes feasible to merge the GDSII files of standard cells to produce files in GDSII format corresponding to the DEF files.

III. Evaluation

A. Layout of DCIM Macro Generated by OpenC^2

Fig. 7 illustrates the OpenC^2-generated layout designs of DCIM macro in 64x64 (4kb) for 4-bit by 4-bit computations.

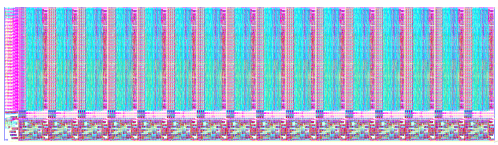

Fig. 7 64x64 DCIM Marco Layout Generated by OpenC^2

B. Area Analysis

We compare the area of OpenC^2 on FreePDK45 with the best result of the design using AutoDCIM on TSMC40. Furthermore, we assess the area efficiency, defined by area/(technology feature size)^2/bits (the lower, the better), which can fairly measure the area required to store each bit to be calculated at different tech nodes. The experimental results are illustrated in Fig. 3.

Across all design sizes, OpenC^2 delivered more than a 30% reduction in area and an enhancement of over 40% in area efficiency in comparison to AutoDCIM. It is worth noting that AutoDCIM adopts a sharing mechanism for adder trees, which comes at the expense of reduced computational parallelism.

C. Power and Efficiency Analysis

The other common evaluation metrics of DCIM Macro include power, energy efficiency, and area efficiency. We also compare these metrics of OpenC^2-generated designs (@1V, 0.1GHz) with some state-of-the-art (SOTA) DCIM macros published recently, ensuring that the comparison is normalized to the same process node and INT4, as illustrated in Fig. 4-6.

IV. Conclusion

In this paper, we propose OpenC^2, the first open-source end-to-end hardware compiler development framework for digital CIM. The framework can generate a datapath-optimized, compact DCIM macro layout tailored to user specifications, leveraging a hierarchical physical design methodology. For public releases, it provides reference circuits and physical implementations in 45nm FreePDK and uses only open-source tools during the flow. It also facilitates adaptation to a diverse array of technologies, sizes, and configurations. The generated design space will enable architecture-level research and various applications of DCIM, building a good ecosystem for DCIM-based design.

References

[1] Y.-D. Chih, P.-H. Lee, H. Fujiwara, Y.-C. Shih, C.-F. Lee, R. Naous, Y.-L. Chen, C.-P. Lo, C.-H. Lu, H. Mori *et al.*, "16.4 an 89tops/w and 16.3 tops/mm 2 all-digital sram-based full-precision compute-in memory macro in 22nm for machine-learning edge applications," in *2021 IEEE ISSCC*, vol. 64, 2021.

[2] J. Chen, F. Tu, K. Shao, F. Tian, X. Huo, C.-Y. Tsui, and K.-T. Cheng, "Autodcim: An automated digital cim compiler," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 2023.

[3] C. Wolf, "Yosys-a free verilog synthesis suite," 2013. [Online]. Available: https://api.semanticscholar.org/CorpusID:202611483

[4] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang, "Ntuplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints," *IEEE TCAD*, 2008.

[5] RTimothyEdwards, "Qrouter," 2023. [Online]. Available: https://github.com/RTimothyEdwards/qrouter

[6] klayoutmatthias, "Klayout," 2024. [Online]. Available: https://github.com/KLayout