DSC-ROM: A Fully Digital Sparsity-Compressed Compute-in-ROM Architecture for On-Chip Deployment of Large-Scale DNNs

Tianyi Yu, Zhonghao Chen, Yiming Chen, Shuang Wang, Yongpan Liu, Huazhong Yang, Xueqing Li
Department of Electronic Engineering, LFET/BNRist, Tsinghua University
Email: xueqingli@tsinghua.edu.cn

***Abstract*—Compute-in-Memory (CiM) is a promising technique for energy-efficient deep neural network (DNN) inference to mitigate the memory bottleneck. Unfortunately, conventional SRAM-based CiM has a low density and limited on-chip capacity, resulting in undesired weight reloading from off-chip DRAM. The emerging high-density ROM-based CiM architecture has recently revealed the opportunity of deploying large-scale DNNs on-chip, with optional assisting SRAM to ensure moderate flexibility. However, prior analog-domain ROM CiM still suffers from limited memory density improvement and low computing area efficiency due to stringent array structure and large A/D converter (ADC) overhead.**

This paper presents DSC-ROM, a fully digital sparsity-compressed compute-in-ROM architecture to address these challenges. DSC-ROM introduces a fully synthesizable macro-level design methodology that achieves a record-high memory density of 27.9 Mb/mm^2 in a 28nm CMOS technology. Experimental results show that the macro area efficiency of DSC-ROM improves by 5.6-6.6x compared with prior analog-based ROM CiM. Furthermore, a novel weight fine-tuning technique is proposed to ensure task transfer flexibility and reduce required assisting SRAM cells by 94.4%. Experimental results show that DSC-ROM designed for ResNet-18 pre-trained on ImageNet dataset achieves <0.5% accuracy loss in CIFAR-10 and FER2013, compared with the fully SRAM-based CiM.

***Index Terms*—compute-in-memory (CiM), read-only memory (ROM), multiply-and-accumulate (MAC), deep neural network (DNN), transfer learning.**

I. Introduction

Deep neural networks (DNNs) have found widespread application across various artificial intelligence domains, including computer vision and natural language processing [1]. However, as data volumes and computational demands continue to surge, the "memory wall" bottleneck inherent in the traditional von Neumann architecture has become increasingly evident [2]. To mitigate the frequent data transfer between memory and arithmetic units, compute-in-memory (CiM) has emerged as an efficient accelerator for multiply-and-accumulate (MAC) operations. Among the various existing techniques, SRAM-based CiM is notable for its high flexibility and mature fabrication support [3]–[6]. Nevertheless, SRAM cells occupy a substantial silicon area, leading to low density within SRAM-based CiM architectures. Consequently, the data in SRAM need to be reloaded from off-chip DRAM, leading to undesirable energy and latency costs [7]. To enhance the macro capacity of CiM, eDRAM-based CiM solutions have been developed [8], [9].

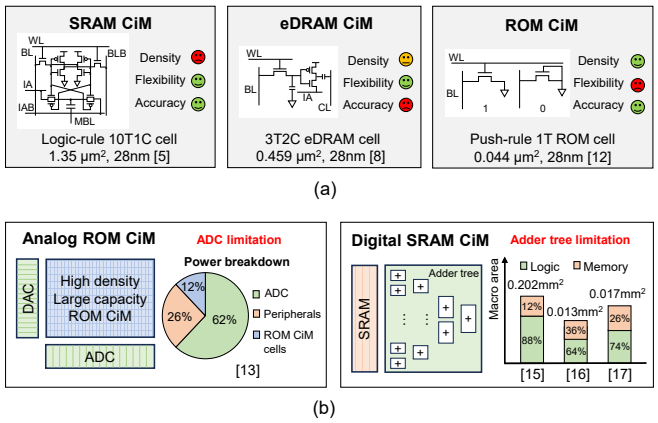


Fig. 1. Challenges in (a) diverse memory architecture and (b) different computing operations of CiM.

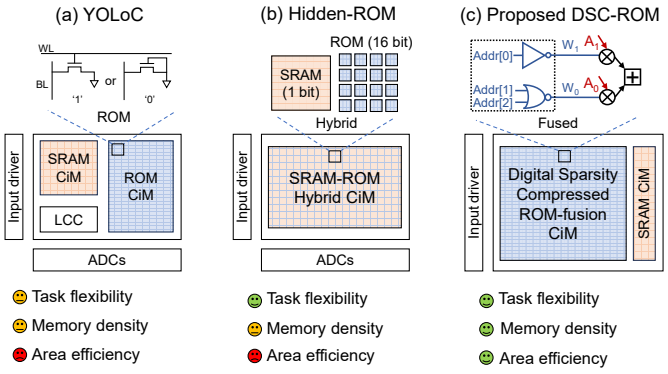


Fig. 2. Comparison of three ROM CiM architectures: (a) YOLoC [10], (b) Hidden-ROM [11], (c) Proposed DSC-ROM.

However, eDRAM requires frequent refreshing to counteract charge loss, which results in additional energy consumption.

Recently, high-density ROM-based CiM has been proposed, becoming a promising solution for deploying large-scale DNNs on chip [12]–[14]. However, analog-domain ROM CiM typically encounters limitations related to the A/D converter (ADC) [8]. In contrast, digital SRAM CiM enhances computing parallelism but suffers from low memory density due to the substantial area occupied by adder tree structures [15]–[17]. Additionally, ROM CiM faces challenges in flexibility due to its fixed weights. Efforts to address this issue include YOLoC

978-3-9826741-0-0/DATE25/© 2025 EDAA

[10] and Hidden-ROM [11]. YOLoC integrates an SRAM CiM branch, enabling transfer learning within the same task domain. Furthermore, Hidden-ROM proposes an SRAM-ROM hybrid CiM macro based on the hidden neural network (HNN) [18], which achieves flexibility across different task domains. However, the inclusion of SRAM limits density improvements, and the analog-domain ROM CiM suffers from low area efficiency.

In this work, we propose a fully digital sparsity-compressed compute-in-ROM (DSC-ROM) architecture to address the aforementioned challenges. The key contributions of this work include:

- Development of a synthesizable DSC-ROM macro design approach: This work presents a ROM-logic fusion macro that achieves a record-high memory density, overcoming the adder tree bottleneck within digital-domain CiM. Moreover, the entire macro is synthesizable using Electronics Design Automation (EDA) tools.
- Establishment of a ROM CiM architecture with enhanced flexibility and scalability: In comparison to YOLoC and Hidden-ROM, the proposed architecture delivers superior performance on complex datasets. Additionally, it reduces the required assisting SRAM CiM area by an impressive 94.4%.
- Comprehensive experiments: This work includes extensive hardware evaluations at both macro and system levels, alongside software assessments at the algorithmic level on a variety of datasets, validating the efficacy of our approach.

The rest of this paper is organized as follows. Section II introduces the background of the ROM CiM transfer learning algorithm. Section III presents the proposed DSC-ROM architecture and Section IV presents experimental evaluations of DSC-ROM. Section V concludes this work.

II. Background

ROM CiM flexibility. High-density ROM-based CiM becomes a compelling solution to enhance on-chip capacity. Although ROM CiM demonstrates high energy efficiency at the task level, it encounters challenges related to flexibility. As shown in Fig. 3(a), ROM CiM performs well on the pre-trained dataset but may experience a decline in accuracy when applied to diverse scenarios. Thus, strategies to improve flexibility are of considerable importance.

YOLoC architecture. Fig. 3(b) shows the architecture of YOLoC. The convolution operation is divided into two parallel blocks: fixed truck ROM CiM and trainable branch SRAM CiM. By training the branch weights, YOLoC can realize the same task domain flexibility, such as from ImageNet (item classification) to CIFAR-10 (item classification). While different task domains transfer learning, such as from ImageNet (item classification) to FER2013 (facial expression recognition), will lose >40% accuracy using YOLoC. This limitation arises because the weights in the fixed truck ROM CiM cannot be reconstructed.

Hidden-ROM architecture. To address the flexibility bottleneck of YOLoC, the Hidden-ROM architecture has been proposed. Based on a hidden neural network (HNN), Hidden-ROM enables the reconfiguration of ROM CiM through the control of a handful of SRAM cells. As shown in Fig. 3(c), the Hidden-ROM macro is divided into two components: a fixed ROM array stored random weights and trainable SRAM stored topology information. During HNN training, the topological structure is stored in SRAM, to select subnetworks from the randomly initialized network stored in ROM. Hidden-ROM improves ROM CiM flexibility to different task domains while offering model expansion capability. However, the inference accuracy of Hidden-ROM may degrade when applied to more complex tasks, as evidenced by a 12.5% accuracy loss of ResNet-18 on ImageNet dataset.

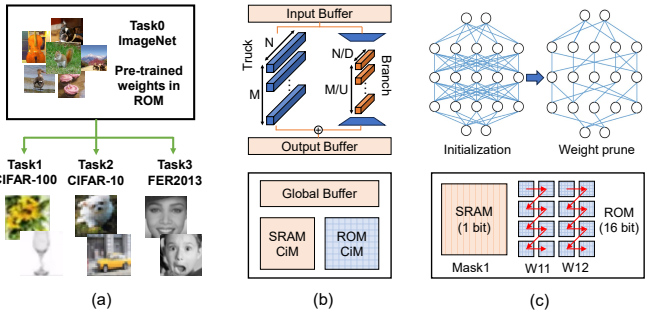


Fig. 3. Previous ROM CiM architectures to address (a) the challenge of flexibility: (b) Parallel SRAM CiM branch, (c) Topological network based on HNN.

III. Proposed DSC-ROM Architecture

A. Proposed Synthesizable DSC-ROM Macro

Goal. The memory density of analog ROM CiM macro mentioned above is limited by the ROM cell area, as there is a one-to-one mapping between the weights and the memory. Actually, the weights contain redundant information that can be effectively compressed. The goal of the DSC-ROM is to deliver a synthesizable macro-level design approach that achieves both high density and throughput through the application of weight compression.

Option-I: Digital Compute-in-ROM (D-ROM). As shown in Fig. 4(a), the D-ROM macro comprises a general ROM array and a general adder tree. Typically, the adder tree occupies over 60% of the macro area to ensure high parallelism, which substantially mitigates the density advantage of digital ROM CiM.

Option-II: Digital Compressed Compute-in-ROM (DC-ROM). Actually, as shown in Fig. 4(b), the memory and MAC logic can be simplified by the fixed weights stored in ROM. For example, when synthesizing the weights of a 64 output-channel (OCN) convolutional layer with the MAC logic unit, the optimization process can be divided into two parts. For the memory, ROM cells that store identical bits can be compressed and replaced by digital logic, and the ROM cells can be discarded as long as the readout remains unchanged. For the MAC unit, the multiplier and adder tree can be optimized by fixed '1' and '0', allowing for a reduction in the bit-width of

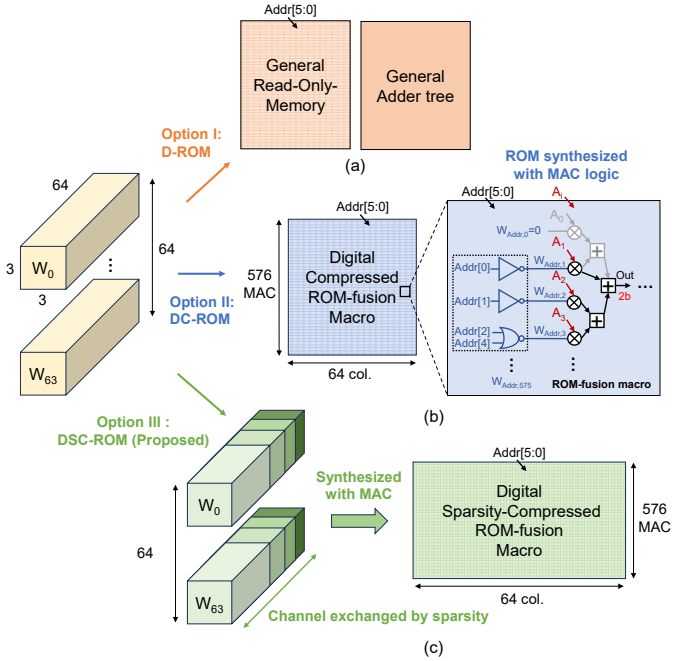


Fig. 4. Options for developing synthesizable digital ROM CiM macro: (a) Option I: digital ROM (D-ROM), (b) Option II: digital compressed ROM synthesized with MAC logic (DC-ROM), (c) Option III: proposed digital sparsity-compressed ROM-fusion macro (DSC-ROM).

the nodes. However, when storing a large number of OCN, the memory logic may become overly complex, which could hinder the simplification of the MAC unit.

Option-III: Proposed Digital Sparsity-Compressed Compute-in-ROM (DSC-ROM). To overcome the weakness of the prior options, Fig. 4(c) shows the proposed DSC-ROM macro. For each OCN, the 3x3 kernels are reorganized based on weight sparsity prior to synthesis with MAC logic. Given that the quantized DNN parameters exhibit significant sparsity, the compression optimization remains efficient even as the number of OCN increases.

B. ROM Inception (Rception)

Goal. After sparsity compression, the feature extraction capability of convolutional layers may decrease. The proposed Rception structure aims to restore flexibility across various tasks by integrating a small portion of SRAM CiM.

Rception. Inspired by the structures of MobileNet [19] and Inception [20], the point-wise convolution [21] emerges as an optimal choice for realizing the flexibility of DSC-ROM. As illustrated in Fig. 5(a), the point-wise convolution serves two primary functions: dimension transformation and convolutional layer reconstruction, with the latter being particularly relevant in this context. The Rception structure is shown in Fig. 5(b). All the 3x3 convolutional layers could be deployed in DSC-ROM macro, and the point-wise convolutional layers could be developed in SRAM CiM block. Practically, the number of SRAM cells could be 288x fewer than that of the ROM cells, ensuring both high memory density and flexible transfer learning capability.

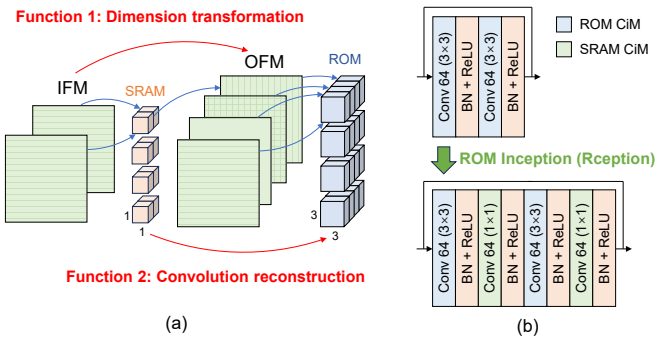


Fig. 5. Proposed Rception structure: (a) Functions of the point-wise convolutional layer, (b) ROM Inception operation.

C. Proposed DSC-ROM Computing Framework

DSC-ROM computing flow. Fig. 6 illustrates the computing architecture of DSC-ROM, while Fig. 7 details the computing flow during a convolution operation. As shown in Fig. 7(a), each MAC operation of a single OCN requires 9 cycles. The first cycle is dedicated to addressing the DSC-ROM macro. Subsequently, the 8-bit activation is transmitted to the DSC-ROM macro via a bit-serial approach. The MAC result is then sent to the Non-Linear Processing Unit (NLPU), where it is transformed into the activation of SRAM CiM macro. For a 64-OCN convolutional layer, each operation requires amounts to 576 cycles.

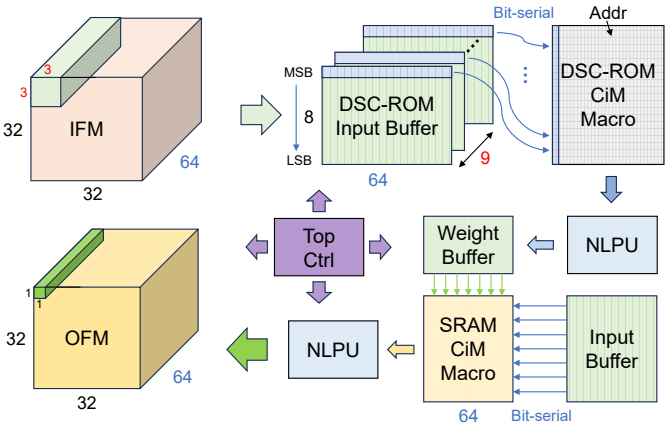


Fig. 6. Proposed computing architecture of DSC-ROM.

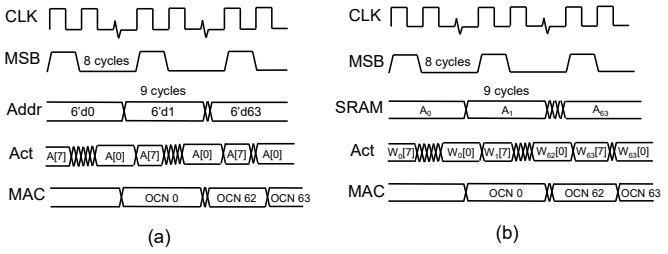


Fig. 7. Computing flow of (a) DSC-ROM CiM macro and (b) SRAM CiM macro in Fig. 6.

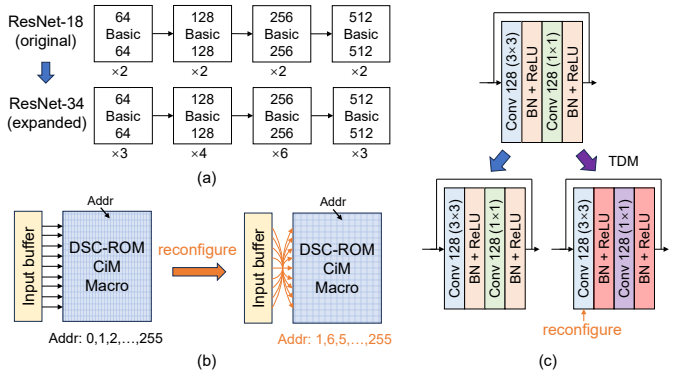


Fig. 8. The model expansion capability of the proposed DSC-ROM: (a) Model expansion from ResNet-18 to ResNet-34, (b) ROM CiM reconfiguration approach, (c) Implementation of model expansion by time division multiplex.

SRAM CiM computing flow. As shown in Fig. 7(b), the activation from NLPU is loaded to SRAM, and the weight of the point-wise layer is sent to SRAM macro by on-chip DRAM. The first cycle is dedicated to reloading SRAM cells, then the 8-bit weights are sent to the SRAM macro using a bit-serial method. MAC results per channel are obtained every 9 cycles. Consequently, each point-wise convolution operation costs 576 cycles, either.

D. Model expansion

Motivation. In theory, convolutional neural networks (CNNs) can be scaled up by incorporating additional layers to enhance their expressive capabilities. While the VGGNet architecture encountered the issue of vanishing gradients with increased layer depth, this challenge has been effectively addressed through the implementation of residual connections in ResNet architecture [22].

Approach. Fig. 8(a) shows the model expansion from ResNet-18 to ResNet-34. Given that the weights in DSC-ROM are reordered, it is possible to realize model expansion by macro reconfiguration, as shown in Fig. 8(b). The sequences of input and OCN address can be trained as parameters. The reconfiguration of DSC-ROM macro can be realized by software with 6% area overhead of extra SRAM cells. Fig. 8(c) presents a strategy for model expansion within the DSC-ROM architecture. Each macro from the original model can be reused by time-division multiplexing (TDM). During each TDM cycle, the weights stored in the DSC-ROM remain fixed, whereas the weights for point-wise convolutions and the associated reconfiguration information in the SRAM cells are updated from external DRAM or an on-chip buffer.

IV. Experimental Result

A. Experiment Setup

Hardware configuration. The macro-level evaluation is based on post-layout simulations in a 28nm CMOS process. Timing and energy analysis are conducted with Synopsys PrimeTime. The system-level energy evaluation includes off-chip data transport energy, estimated at 20 pJ/bit [23], in addition to the compute energy of the macro.

ResNet-18	Parameter (4 bit)	Weight Sparsity	Feature Map Size	Macro MAC Operation	Access Time
Layer1	0.56 Mb	0.706	(56,56,64)	576	1.8 ms
Layer2	1.97 Mb	0.713	(28,28,128)	576	1.8 ms
Layer3	7.88 Mb	0.732	(14,14,256)	576	1.8 ms
Layer4	31.5 Mb	0.763	(7,7,512)	576	1.8 ms

(a)

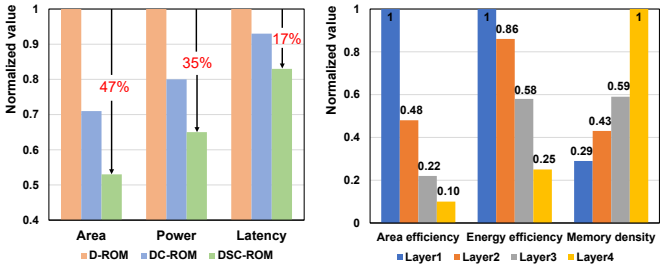


Fig. 9. Evaluations of the DSC-ROM macro: (a) Implementation of ResNet-18 convolutional layers, (b) PPA improvements compared to other synthesis approaches, (c) Area efficiency, energy efficiency, and memory density comparison of different layers.

Software configuration. Two original image classifier models VGG-16, ResNet-18, and two expanded models ResNet-34, and ResNet-50 are used in our benchmarks. PyTorch is used as the DNN framework for parameter training and transfer learning of the proposed DSC-ROM architecture. To enhance accuracy, we employ quantization-aware training (QAT). The models are pre-trained on ImageNet dataset.

B. DSC-ROM Macro Synthesis Evaluations

Mapping. The ResNet-18 comprises 4 feature extractor layers, each containing 4 convolutional layers. The entire 4-bit quantized network is mapped onto 16 DSC-ROM CiM macros. Fig. 9(a) shows the characteristics of the different layers. Each macro has consistent throughput and access time during a single inference operation. This uniformity facilitates the development of a macro-level pipeline design.

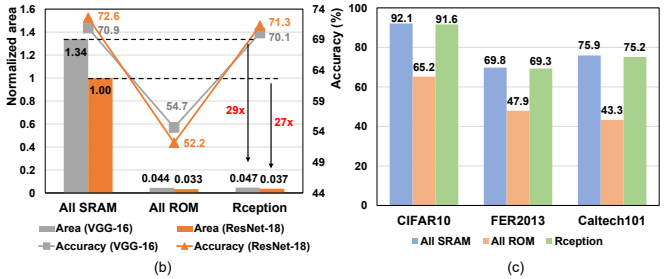
Three options comparison. Fig. 9(b) presents a comparison of power, performance, and area (PPA) for Layer1 of the three synthesis options proposed in Fig. 4. Without surprise, D-ROM exhibits the highest area and power consumption due to its general adder tree. Although DC-ROM provides improvements in PPA relative to D-ROM, its performance degrades when storing a larger number of OCN, particularly evident in Layer4. In contrast, by employing sparse optimization, DSC-ROM achieves effective weight compression and logic simplification, resulting in improved PPA metrics of 35%, 17%, and 47% over D-ROM, respectively.

Different layers comparison. Fig. 9(c) illustrates the area efficiency, energy efficiency, and memory density of the DSC-ROM macro across different layers. Layer1 has the highest area efficiency and energy efficiency, while Layer4 has the highest memory density. It is worth noticing that in ResNet-18, the earlier layers contribute significantly to the computation, whereas the later layers account for a substantial portion of the parameter. As a result, DSC-ROM serves as an efficient method for accelerating inference in ResNet.

Method	ROM	SRAM
All SRAM	-	Whole NN
All ROM	Feature extractor	Conv1, classifier
Rception	Feature extractor	Conv1, point-wise conv, classifier

Quantization	Activation	Weight
Conv1	FP	FP
Feature extractor	unsigned int8	signed int4
Point-wise conv	unsigned int8	signed int8
Classifier	unsigned int8	signed int8

(a)



(b) (c)

Fig. 10. Flexibility analysis of Rception: (a) Baseline and model quantization configurations, (b) Area overhead and accuracy comparison on CIFAR-100 transfer learning, (c) Accuracy comparison on diverse datasets.

C. Flexibility and Scalability Evaluations

Flexibility. Fig. 10 shows the evaluation results of the proposed Rception structure based on VGG-16 and ResNet-18. Fig. 10(a) shows the baseline and model quantization settings. The accuracy baseline is a pure SRAM-based CiM, while the density baseline is a pure ROM-based CiM. The inference accuracy on CIFAR-100 transfer learning and area overhead comparison are shown in Fig. 10(b). The Rception structure acquires the capability of transfer learning with only 10% extra area overhead compared with the pure ROM-based CiM, while improving memory density by 27x compared with the pure SRAM-based CiM. Fig. 10(c) compares accuracy on datasets of different task domains, revealing that Rception achieves flexibility with only <0.8% accuracy loss.

Scalability. Fig. 11 shows the results of model expansion from ResNet-18 to ResNet-34/50 on ImageNet dataset compared with YOLoC and Hidden-ROM. Notably, YOLoC does not support model expansion after fabrication because its fixed weights can not be reconstructed. Compared with HNN-based Hidden-ROM, Rception achieves higher accuracy even at lower quantization bit-width. In addition to ResNet, the Rception model can be expanded to other DNNs with isomorphic convolutional layers.

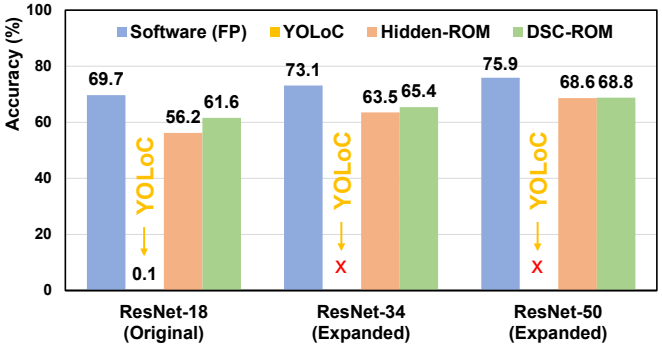


Fig. 11. Accuracy comparison of DSC-ROM model expansion with YOLoC and Hidden-ROM on ImageNet dataset.

TABLE I
SPECIFICATION SUMMARY

[1]Specifications	SRAM	YOLoC	Hidden-ROM	[2]DSC-ROM
Process	28nm CMOS			
CiM operation	Analog SRAM	Analog SRAM+ROM	Analog Hybrid	**Digital SRAM+ROM**
Capability (Mb)	0.064	1.26	1.26	**44.08**
Macro area (mm^2)	0.32	0.48	0.44	**1.58**
Precision (Input x weight)	8-bit x 8-bit			**8-bit x 4/8-bit**
[3]Throughput (GOPS)	30.2	57.3	61.7	**1251.6**
Macro density (Mb/mm^2)	0.20	2.62	2.86	**27.9** (9.8-10.6x)
[3]Macro energy efficiency (TOPS/W)	11.54	11.50	10.94	**11.63**
[3]Area efficiency ($GOPS/mm^2$)	94.25	119.4	140.3	**792.2** (5.6-6.6x)
#ROM/#SRAM ratio	0	16	16	**288** (18x)

[1]: Note that the specifications may improve as the CMOS process improves.

[2]: Evaluated by post-layout simulation at 0.9V, 16 macros average.

[3]: Normalized to 8-bit input x 8-bit weight.

D. Macro and System Evaluations

Macro-level. The proposed DSC-ROM macro has been simulated by Synopsys PrimeTime based on post-layout parasitic extraction. Table I shows the comparison with the prior ROM CiM architecture, YOLoC, and Hidden-ROM. This work achieves a record-high memory density of 27.9 Mb/mm^2, which is 9.8-10.6x higher than YOLoC and Hidden-ROM. Furthermore, as a fully digital CiM work, DSC-ROM improves 5.6-6.6x area efficiency compared with analog-domain ROM CiM.

System-level. The system evaluation of DSC-ROM is conducted based on the macro specifications. For the limited capacity SRAM-based works, only a partial model can be loaded on chip. It needs to reload all the weights of DNN from off-chip DRAM at each inference. Actually, the energy consumption of data transport is much higher than computation. As a result, large-capacity ROM-based CiM demonstrates superior energy efficiency at system-level.

Fig. 12 shows the task-level energy consumption of the DSC-ROM in comparison to conventional SRAM CiM. By storing the entire convolutional layer weights within ROM cells, the proposed DSC-ROM macro only requires reloading the weights to update the small portion of SRAM cells, which reduces more than 70% off-chip weights reloading. Therefore, this work achieves 3.2-4.8x task-level inference energy efficiency improvement. Moreover, the expanded models, ResNet-34 and ResNet-50, incur only a 3% and 14% increase in extra energy overhead, respectively.

Future works. DSC-ROM effectively combines the high capacity of analog ROM CiM with the high throughput of digital computation, offering enhanced flexibility and scalability. However, the heterogeneous DSC-ROM macros of different layers make it difficult to operate model transfer, such as from ResNet-18 to Xception. The computing framework is supposed to be

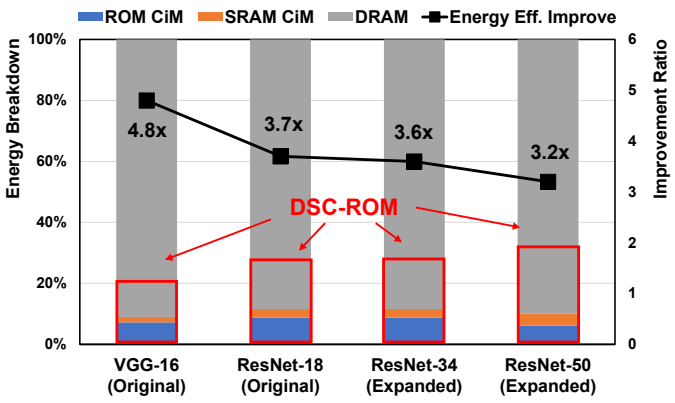


Fig. 12. Comparison of task-level inference energy consumption between conventional SRAM CiM and DSC-ROM CiM.

more reconfigurable. One promising direction is improving layer-wise reusability, and another direction is developing a dataflow pipelining design with more fine-grained computing blocks. Future work focused on thoroughly exploring the design space of ROM CiM, along with cross-layer co-optimization, holds significant potential for advancing this field.

V. Conclusions

This paper presents DSC-ROM, a fully digital sparsity-compressed compute-in-ROM architecture to deal with the bottlenecks of analog ROM CiM and digital SRAM CiM. The proposed synthesizable ROM-logic fusion macro achieves exceptional area efficiency and a record-high memory density. Furthermore, the proposed Rception demonstrates the flexibility of different task domains with negligible accuracy loss. It also facilitates high-accuracy model expansion through macro reconfiguration, making it suitable for more complex scenarios. Overall, DSC-ROM has provided a novel paradigm for deploying large-scale DNNs on-chip with both high performance and energy efficiency.

Acknowledgment

This work is supported in part by NSFC (Grant # U21B2030 and 92264204), in part by the National Key R&D Program of China (Grant # 2019YFA0706100).

References

[1] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, and S. S. Iyengar, "A survey on deep learning: Algorithms, techniques, and applications," *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 1–36, 2018.

[2] W. A. Wulf and S. A. McKee, "Hitting the memory wall: Implications of the obvious," *ACM SIGARCH computer architecture news*, vol. 23, no. 1, pp. 20–24, 1995.

[3] K. Yoshioka, "34.5 a 818-4094tops/w capacitor-reconfigured cim macro for unified acceleration of cnns and transformers," in *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 67. IEEE, 2024, pp. 574–576.

[4] M. Wu, W. Ren, P. Chen, T. Jia *et al.*, "S2d-cim: A 22nm 128kb systolic digital compute-in-memory macro with domino data path for flexible vector operation and 2-d weight update in edge ai applications," in *2024 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, 2024, pp. 1–2.

[5] B. Zhang, J. Saikia, J. Meng, J.-S. Seo, M. Seok *et al.*, "Macc-sram: A multistep accumulation capacitor-coupling in-memory computing sram macro for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, 2023.

[6] Y. Yuan, Y. Yang, X. Wang, J. Zhu *et al.*, "34.6 a 28nm 72.12 tflops/w hybrid-domain outer-product based floating-point sram computing-in-memory macro with logarithm bit-width residual adc," in *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 67. IEEE, 2024, pp. 576–578.

[7] H. Jiang, S. Huang, X. Peng, R. Liu, M.-F. Chang, S. Yu *et al.*, "A two-way sram array based accelerator for deep neural network on-chip training," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[8] S. Kim, Z. Li, S. Um, D. Han, H.-J. Yoo *et al.*, "Dynaplasia: An edram in-memory computing-based reconfigurable spatial accelerator with triple-mode cell," *IEEE Journal of Solid-State Circuits*, 2023.

[9] Y. Zhan, W.-H. Yu, K.-F. Un, R. P. Martins, and P.-I. Mak, "A 28-nm 18.7 tops/mm ^{2} 89.4-to-234.6 tops/w 8b single-finger edram compute-in-memory macro with bit-wise sparsity aware and kernel-wise weight update/refresh," *IEEE Journal of Solid-State Circuits*, pp. 1–11, 2024.

[10] Y. Chen, G. Yin, Y. Liu, H. Yang, K. Ma, X. Li *et al.*, "Yoloc: deploy large-scale neural network by rom-based computing-in-memory using residual branch on a chip," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 1093–1098.

[11] Y. Chen, G. Yin, M. Lee, W. Tang, Z. Yang, Y. Liu, H. Yang, and X. Li, "Hidden-rom: A compute-in-rom architecture to deploy large-scale neural networks on chip with flexible and scalable post-fabrication task transfer capability," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, 2022, pp. 1–9.

[12] G. Yin, Y. Chen, M. Lee, H. Yang *et al.*, "A 28nm 8928kb/mm^2-weight-density hybrid sram/rom compute-in-memory architecture reducing¿ 95% weight loading from dram," in *2024 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, 2024, pp. 1–2.

[13] G. Yin, Y. Chen, V. Narayanan, H. Yang *et al.*, "Cramming more weight data onto compute-in-memory macros for high task-level energy efficiency using custom rom with 3984-kb/mm^2 density in 65-nm cmos," *IEEE Journal of Solid-State Circuits*, 2023.

[14] R. Sehgal, R. Mehra, C. Ni, and J. P. Kulkarni, "Compute-mlrom: Compute-in-multi level read only memory for energy efficient edge ai inference engines," in *ESSCIRC 2023-IEEE 49th European Solid State Circuits Conference (ESSCIRC)*. IEEE, 2023, pp. 37–40.

[15] Y.-D. Chih, P.-H. Lee, H. Fujiwara, 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 International Solid-State Circuits Conference (ISSCC)*, vol. 64. IEEE, 2021, pp. 252–254.

[16] H. Fujiwara, H. Mori, W.-C. Zhao, T.-Y. J. Chang *et al.*, "A 5-nm 254-TOPS/W 221-TOPS/mm^2 Fully-Digital Computing-in-Memory Macro Supporting Wide-Range Dynamic-Voltage-Frequency Scaling and Simultaneous MAC and Write Operations," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65, 2022, pp. 1–3.

[17] H. Mori, W.-C. Zhao, C.-E. Lee, T.-Y. J. Chang *et al.*, "A 4nm 6163-TOPS/W/b 4790-TOPS/mm^2/b SRAM Based Digital-Computing-in-Memory Macro Supporting Bit-Width Flexibility and Simultaneous MAC and Weight Update," in *2023 IEEE International Solid-State Circuits Conference (ISSCC)*, 2023, pp. 132–134.

[18] V. Ramanujan, M. Wortsman, A. Kembhavi, A. Farhadi, and M. Rastegari, "What's hidden in a randomly weighted neural network?" in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 893–11 902.

[19] A. G. Howard, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[20] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[21] C. Szegedy, W. Liu, Y. Jia, V. Vanhoucke, A. Rabinovich *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[23] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC)*. IEEE, 2014, pp. 10–14.