2024 IEEE 17th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC) | 979-8-3315-3047-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/MCSoC64144.2024.00048

Near-DRAM Accelerated Matrix Multiplications

Aman Sinha*, *Member, IEEE* and Bo-Cheng Lai, *Member, IEEE*
Institute of Electronics
National Yang Ming Chiao Tung University
Hsinchu, Taiwan
Email : amansinha.sw@gmail.com, bclai@nycu.edu.tw

***Abstract*—General matrix multiplication (GeMM) is a fundamental computing operation underpinning various applications in machine learning, data science, computer graphics, and scientific simulations. However, its performance on von Neumann computers, such as CPUs and GPUs, is bottlenecked due to challenges such as insufficient memory access bandwidth, low cache efficiency, hardware under-utilization, and massive power consumption. The performance of GPUs suffers drastically for complex sequential analysis often occurring along with GeMMs in various big data pipelines. Furthermore, GeMM-optimized Tensor cores available in modern Nvidia GPUs offer no extensibility to non-GeMM tasks. Various Near-Memory Computing (NMC) architectures have recently been explored to alleviate the data-intensive nature of analyses such as GeMM. This work evaluates the performance potential of GeMM using NMC through clusters of simple interconnected processing cores on a stacked DRAM platform. The organized design shows efficiency comparable to high-end Nvidia GPUs while consuming lower power and being highly extensible to various non-GeMM logically complex workloads.**

***Index Terms*—DNNs, GeMM, Matrix Multiplication, Near-Memory Computing, RISC-V, Stacked Memory**

I. Introduction

Matrix multiplications constitute the core computation [1] in various High-Performance Computing (HPC) and AI applications, primarily in linear algebra and Deep Neural Networks (DNNs). General Matrix Multiplication (GeMM) is defined as the operation D = α AB + β C, where the matrixes A, B, and C have dimensions of m $\times$ k, n $\times$ k and m $\times$ n, and α and β are scalar values. GeMM computation requires a transpose of the matrix B, which is often accomplished logically using various techniques instead of physically moving the data in memory [2].

Current state-of-the-arts for GeMMs. Widespread application of GeMMs has given rise to several architectural evolutions in the GPU design space, with Tensor Cores (TCs) [2] forming the latest development on massively parallel Nvidia GPUs. Besides the obvious intense computations, GeMM involves massive data accesses, often severely bottlenecked by cache-inefficient memory access [3]. High-bandwidth memory (HBM) has been the recent trend to alleviate the memory access challenges in GPUs. The general mechanism to enhance data reuse exploits the faster memory levels as a buffer for

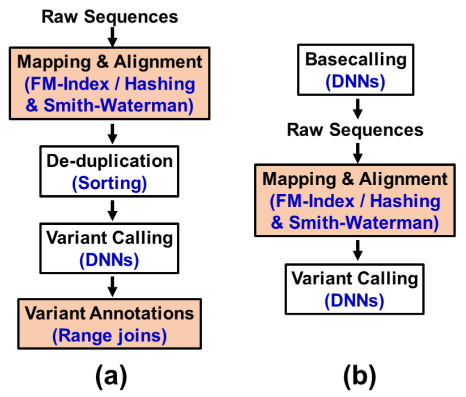


Fig. 1. Bioinformatics analysis pipelines involving sequential complex executions and data-parallel GeMMs. (a) Short read sequence analysis using DeepVariant [6] and (b) Oxford Nanopore long read analysis [5] pipelines. Red blocks represent non-GeMM complex data-intensive analysis.

shuffle operations in the overall memory hierarchy on GPUs, viz. global memory, L2 and L1 caches, shared memory, and registers. CUTLASS [4] provides access to high-performant GeMMs on the modern Nvidia Tensor cores by utilizing the plentiful core-local registers as the data buffering space for both inputs and outputs of GeMM. However, the deep memory hierarchy on GPUs still presents a tradeoff between throughput achieved through memory access latency hiding and the energy efficiency of the involved data movements. Furthermore, the architectural limitations of GPUs to executing complex sequential logic present another major challenge to its benefit-cost tradeoff for big data analysis that involves pipelines of both data-parallel and sequential computations. Fig. 1 depicts such a situation from Bioinformatics, where the analysis pipelines [5], [6] involve complex patterns of data-intensive parallel and sequential executions.

The limitations of GPUs in processing data-bound workloads have given rise to various Near-Memory Computing (NMC) techniques [7] that place execution units close to the memory device to overcome the limited access channels in von Neumann computers while offering inexpensive scalability [8]–[11]. However, to fully exploit the performance capability of NMC architectures, it becomes necessary to maximize the host-independent execution of NMC devices. This necessitates the adoption of such pipelined workloads (Fig. 1) explicitly

This work was supported by the Ministry of Science and Technology, Taiwan [MOST 111-2221-E-A49-092-MY3].
*Corresponding author

on the NMC device. The currently limited exploration of such designs is the major motivation behind this work. For instance, in Fig. 1a, while *Mapping & Alignment* have previously been accelerated on the NMC platform, the other steps are yet to exploit the same performance benefits [12], [13]. Note that NMC differs from Processing-In-Memory (PIM), which utilizes the memory circuitry for bit-parallel computations [14], which can affect the standard memory operations. This work focuses on NMC instead of PIM [15].

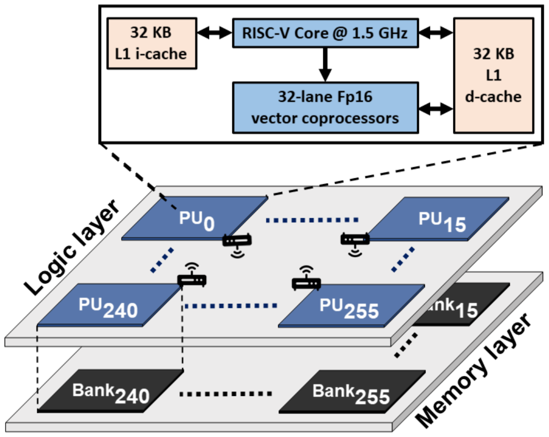


Fig. 2. Overall SEDRAM PU system organization, derived from [9]. All PUs connect over Network-on-Chip (NoC) in mesh topology on the logic layer, using single-cycle latency state-of-the-art routers [9].

Potential of Near-DRAM Computing over Stacked memory platform for GeMMs. While HBM has been the primary memory technology to reduce data bandwidth challenges, it has been shown to face challenges such as high interface energy [16], [17] and inflexible scalability [9]. Stacked Embedded DRAM (SEDRAM) solves these challenges by replacing the Through-Silica Vias (TSVs) with hybrid bondings [16], [17], thus achieving 6.67x lower interface power. Our previous work [9] has extensively explored the SEDRAM platform for the data-intensive task of string matching using the FM-Index data structure. This work extends and evaluates its performance potential for GeMM tasks.

TABLE I
PEAK PERFORMANCE, AREA AND POWER COMPARISON BETWEEN V100 GPU AND 5120 PU SEDRAM ORGANIZATION. SEDRAM PUS HAVE BEEN ESTIMATED AT 40NM TECHNOLOGY, WHILE V100 AT 12NM [18].

	SEDRAM PUs		
	1 PU	***5120 PUs***	***V100***
Peak Performance (TFLOPs)	0.048	245.76	125
Area (mm^2)	0.161	822.3	815
Power Consumption (W)	0.0479	245.45	300

Table I summarizes the peak theoretical performance of a SEDRAM-based Processing Unit (PU) with 32-lane vector coprocessors operating at 1500 MHz for half-precision floating point (FP16) computations. Compared to 640 Tensor cores on V100 GPU [18], SEDRAM organization with comparable 5120 PUs can achieve twice the throughput at nearly 20% lower power consumption. Moreover, the area efficiency is expected to improve at better technology nodes for SEDRAM PUs.

II. NEAR-DRAM ACCELERATED GEMMS

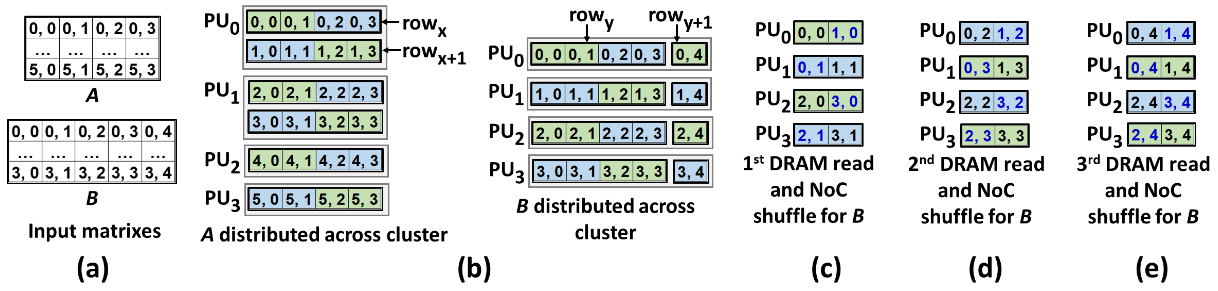
Fig. 2 shows the overall organization of SEDRAM PUs, as derived from our previous work [9], while the unnecessary components from the original design have been removed. Moreover, the vector coprocessors have been configured for half-precision floating point (FP16) operations instead of the original 64-bit design. However, the overall area and power characteristics would remain the same due to the equivalent number of vector lanes.

Cache-efficient Network-on-Chip (NoC)-based Matrix Transpose As discussed before in section I, state-of-the-art CUTLASS utilizes the plentiful registers on GPUs to perform blocked GeMMs [2], instead of physically transposing the matrix *B* required for maximizing Single Instruction Multiple Data (SIMD) computations. However, SEDRAM PUs lack such plentiful registers. Hence, it depends upon the optimized transpose of the matrix *B* to exploit the vector coprocessors' SIMD capabilities. Note that our design too performs blocked-GeMMs similar to CUTLASS, our explicit matrix transpose operation being the only algorithmic difference.

Fig. 3 illustrates the partial matrix transpose mechanism achieved through NoC transfers for each cache line filled through the DRAM bank read. The original input matrixes (*A* and *B* in Fig. 3a) are distributed across the PU cluster by the host CPUs (Fig. 3b). Each cache line has been assumed to have the capacity to hold two matrix cells, while each DRAM row has been assumed to store data for two such cache lines. The highlighted cells in Fig. 3c - 3e show the shuffling operations that generate the partial transpose of the matrix *B*. Furthermore, each round of shuffle operation is followed by partial GeMM execution through vector coprocessor computations for Fp16 products and reductions-sums. The overall design maximizes utilizing each DRAM row buffer activated for the matrix *B*. The NoC transfers are facilitated through single custom instruction on each PU, while the actual transfers overlap with the Fp16 computations on the vector coprocessors. It can be inferred that NoC performance forms a critical component for high throughput of the overall design.

III. EXPERIMENTAL EVALUATION

We performed experiments to evaluate the performance potential of stacked memory-based Near-DRAM Accelerated GeMMs and compare them to the highly efficient baseline, Nvidia CUTLASS [4] library, execution on a V100 GPU equipped with 640 Tensor cores [2], [18]. SEDRAM module composed of 256 SEDRAM PUs, each having 256 MB memory bank, was simulated, and the performance was scaled-up for 20 such SEDRAM modules in order to perform a fair comparison with the large die size of V100 GPU [18]. Each PU having RISC-V cores equipped with 32-lane FP16 vector coprocessors was simulated using the cycle-accurate Gem5



A
B
Input matrixes
(a)
PU0
PU1
PU2
PU3
rowx
rowx+1
A distributed across cluster
rowy
rowy+1
B distributed across cluster
(b)
1st DRAM read and NoC shuffle for B
(c)
2nd DRAM read and NoC shuffle for B
(d)
3rd DRAM read and NoC shuffle for B
(e)

Fig. 3. Illustration of NoC-centered GeMM design on four SEDRAM PU clusters. (a) Original input matrixes A and B with dimensions 6 × 4 and 4 × 5, respectively, (b) distribution of the input matrixes across the memory banks on the cluster. Assume that the cache line can accommodate two data items while each DRAM bank row can accommodate four such data items. (*i*, *j*) represent matrix cell value at row *i* and column *j*. (c) First, DRAM read, followed by data shuffles across PUs and vector coprocessor executions of products and reduction sums. (d) Second, and (e) third, DRAM reads, NoC-based shuffles, and product and reduction-sum operations to maximize cache line and DRAM row-buffer utilizations.

[19] simulator integrated with Ramulator [20], similar to our previous works [9], [10]. However, the operating frequency was upgraded to 1500 MHz for the MinorCPU configuration of Gem5. Vector coprocessor executions were simulated using custom instructions. In contrast, the NoC traffic pattern was separately simulated using Garnet [21] in mesh topology and integrated with PU execution results, with the assumption that NoC transfers overlap with PU executions. The overall PU cluster simulation methodology remains the same as the previous work [9].

TABLE II
PERFORMANCE COMPARISON WITH CUTLASS-BASED GEMM ON NVIDIA V100 GPU

Matrix dimensions	Runtime (ms) *V100*	*This work*	*Speedup*
64 X 64 X 64	1.405	0.512	2.74x
128 X 128 X 128	1.762	1.024	1.72x
256 X 256 X 256	2.483	2.048	1.21x
512 X 512 X 512	4.042	8.192	0.49x
1024 X 1024 X 1024	13.026	24.57	0.53x
2048 X 2048 X 2048	70.876	114.688	0.62x
4096 X 4096 X 4096	516.854	589.824	0.87x

As seen in Table II, the SEDRAM PU organization achieves up to 2.74x speedups for smaller GeMM workloads, while the performance degrades up to half for some of the larger workloads. Note that small and medium GeMM workloads are widely used in various applications such as Bioinformatics [22], thus indicating the efficacy of results in Table II. Moreover, the number of PUs will increase with improved technology nodes for SEDRAM PUs, further enhancing its performance. Finally, the high extensibility of the SEDRAM PU organization is another crucial tradeoff.

IV. CONCLUSION

General Matrix Multiplication (GeMMs) form a major component of various HPC workloads, and their execution on state-of-the-art GPUs faces challenges in terms of memory access efficiencies and extensibility to complete analysis pipelines. This work presents a preliminary evaluation of Near-Memory Computing over the recently developed Stacked Embedded DRAMs for GeMM tasks. The simple organization achieves performance characteristics comparable to high-end modern GPUs while offering much larger extensibility. Our future works will further optimize the design and evaluate the performance potential for end-to-end pipeline executions for The relevant bioinformatics workloads.

ACKNOWLEDGMENTS

This work was supported by the Ministry of Science and Technology, Taiwan [MOST 111-2221-E-A49-092-MY3].

REFERENCES

[1] E. Qin, A. Samajdar, H. Kwon, V. Nadella, S. Srinivasan, D. Das, B. Kaul, and T. Krishna, "Sigma: A sparse and irregular gemm accelerator with flexible interconnects for dnn training," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2020, pp. 58–70.

[2] S. Markidis, S. W. Der Chien, E. Laure, I. B. Peng, and J. S. Vetter, "Nvidia tensor core programmability, performance & precision," in *2018 IEEE international parallel and distributed processing symposium workshops (IPDPSW)*. IEEE, 2018, pp. 522–531.

[3] D. Chen, H. Jin, L. Zheng, Y. Huang, P. Yao, C. Gui, Q. Wang, H. Liu, H. He, X. Liao *et al.*, "A general offloading approach for near-dram processing-in-memory architectures," in *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2022, pp. 246–257.

[4] T. Faingnaert, T. Besard, and B. De Sutter, "Flexible performant gemm kernels on gpus," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 9, pp. 2230–2248, 2021.

[5] Y. Wang, Y. Zhao, A. Bollas, Y. Wang, and K. F. Au, "Nanopore sequencing technology, bioinformatics and applications," *Nature biotechnology*, vol. 39, no. 11, pp. 1348–1365, 2021.

[6] Y.-L. Lin, P.-C. Chang, C. Hsu, M.-Z. Hung, Y.-H. Chien, W.-L. Hwu, F. Lai, and N.-C. Lee, "Comparison of gatk and deepvariant by trio sequencing," *Scientific Reports*, vol. 12, no. 1, p. 1809, 2022.

[7] G. Singh, L. Chelini, S. Corda, A. J. Awan, S. Stuijk, R. Jordans, H. Corporaal, and A.-J. Boonstra, "A review of near-memory computing architectures: Opportunities and challenges," in *2018 21st Euromicro Conference on Digital System Design (DSD)*. IEEE, 2018, pp. 608–617.

[8] A. Sinha, H.-C. Yang, P.-Y. Liu, Y.-S. Kuo, Y. Fang, T.-S. Chang, K.-H. Li, and B.-C. Lai, "Dsim: Distributed sequence matching on near-dram accelerator for genome assembly," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 12, no. 2, pp. 486–499, 2022.

[9] A. Sinha, Y. Fang, and B.-C. Lai, "Regal: Reprogrammable engines for genome analysis on lpddr4x-based stacked dram," in *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2023, pp. 1–5.

[10] A. Sinha, P.-Y. Liu, Y. Fang, J.-Y. Mai, and B.-C. Lai, “Grona: A framework for gather-and-reduce on near-memory accelerators,” in *2023 IEEE 16th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*. IEEE, 2023, pp. 225–232.

[11] J. Gómez-Luna, I. El Hajj, I. Fernandez, C. Giannoula, G. F. Oliveira, and O. Mutlu, “Benchmarking memory-centric computing systems: Analysis of real processing-in-memory hardware,” in *2021 12th International Green and Sustainable Computing Conference (IGSC)*. IEEE, 2021, pp. 1–7.

[12] A. Sinha, B. C. Lai, and J. Y. Mai, “A bin-based indexing for scalable range join on genomic data,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2023.

[13] K. Wang, M. Li, and H. Hakonarson, “Annovar: functional annotation of genetic variants from high-throughput sequencing data,” *Nucleic acids research*, 2010.

[14] N. Verma, H. Jia, H. Valavi, Y. Tang, M. Ozatay, L.-Y. Chen, B. Zhang, and P. Deaville, “In-memory computing: Advances and prospects,” *IEEE Solid-State Circuits Magazine*, vol. 11, no. 3, pp. 43–55, 2019.

[15] C. Nie, C. Tang, J. Lin, H. Hu, C. Lv, T. Cao, W. Zhang, L. Jiang, X. Liang, W. Qian *et al.*, “Vspim: Sram processing-in-memory dnn acceleration via vector-scalar operations,” *IEEE Transactions on Computers*, 2023.

[16] B. Fujun, J. Xiping, W. Song, Y. Bing, T. Jie, Z. Fengguo, W. Chunjuan, W. Fan, L. Xiaodong, Y. Guoqing *et al.*, “A stacked embedded dram array for lpddr4/4x using hybrid bonding 3d integration with 34gb/s/1gb 0.88 pj/b logic-to-memory interface,” in *2020 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2020, pp. 6–6.

[17] D. Niu, S. Li, Y. Wang, W. Han, Z. Zhang, Y. Guan, T. Guan, F. Sun, F. Xue, L. Duan *et al.*, “184qps/w 64mb/mm 2 3d logic-to-dram hybrid bonding with process-near-memory engine for recommendation system,” in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65. IEEE, 2022, pp. 1–3.

[18] J. Peddie, “Compute accelerators and other gpus,” in *The History of the GPU-New Developments*. Springer, 2023, pp. 239–304.

[19] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti *et al.*, “The gem5 simulator,” *ACM SIGARCH computer architecture news*, vol. 39, no. 2, pp. 1–7, 2011.

[20] Y. Kim, W. Yang, and O. Mutlu, “Ramulator: A fast and extensible dram simulator,” *IEEE Computer architecture letters*, vol. 15, no. 1, pp. 45–49, 2015.

[21] N. Agarwal, T. Krishna, L.-S. Peh, and N. K. Jha, “Garnet: A detailed on-chip network model inside a full-system simulator,” in *2009 IEEE international symposium on performance analysis of systems and software*. IEEE, 2009, pp. 33–42.

[22] Z. Xu, Y. Mai, D. Liu, W. He, X. Lin, C. Xu, L. Zhang, X. Meng, J. Mafofo, W. A. Zaher *et al.*, “Fast-bonito: A faster deep learning based basecaller for nanopore sequencing,” *Artificial Intelligence in the Life Sciences*, vol. 1, p. 100011, 2021.