A Pipelined Multi-Bank GEMM Accelerator for Memory Bottleneck Mitigation

Anonymous
Anonymous Organization
anonymous@anonymous.org

Abstract

General Matrix Multiplication (GEMM) is a fundamental operation in AI and high-performance computing (HPC). However, conventional CPUs and GPUs suffer from severe memory bottlenecks, which significantly limit utilization of compute resources. This paper proposes a GEMM accelerator architecture incorporating a Multi-Bank Memory Crossbar (MBMC). The proposed architecture connects four GEMM units with on-chip SRAM banks through a crossbar to enable parallel memory access, while a Cortex-M0 controls data transfers using DMA-driven event control to maximize overlap between computation and communication. Compared with a conventional single-bus design, the proposed accelerator reduces data load cycles by an average of 1.49× and improves throughput by 1.49×. These results demonstrate the potential of MBMC to fundamentally mitigate memory bottlenecks and unlock scalable performance for AI and HPC GEMM accelerators.

Keywords

General Matrix Multiplication (GEMM); Accelerator Architecture; Memory Bottleneck; Multi-Bank Crossbar; Tiling; Output-Stationary (OS) Dataflow

1 Introduction

General Matrix Multiplication (GEMM) is a core operation in artificial intelligence (AI) and high-performance computing (HPC). However, conventional CPUs and GPUs, bound by the processor–memory separation inherent to the Von Neumann architecture, are limited by memory bandwidth. This so-called Von Neumann bottleneck is particularly detrimental to GEMM workloads, which require repeated access to large datasets. Although prior research has explored various techniques to alleviate this issue, most efforts have focused primarily on improving the speed of Multiply-Accumulate (MAC) operations [2, 3]. For instance, systolic array architectures employ pipelining to maximize throughput [4, 5]. However, no matter how fast the compute units become, if the memory subsystem cannot provide data in time, the compute units inevitably remain idle. Thus, computation-centric optimizations alone are insufficient to overcome the fundamental memory access bottleneck.

To address this challenge, this paper proposes a GEMM accelerator architecture based on a Multi-Bank Memory Crossbar (MBMC) [1, 6]. Unlike conventional designs that only increase compute parallelism, the proposed architecture alleviates memory loading bottlenecks, thereby reducing idle compute time and enhancing overall system throughput. The main contribution of this paper is the design of a Processing Element (PE) module based on a MBMC, combined with a lightweight Cortex-M0 controller that dynamically orchestrates GEMM units and the memory subsystem enhance system-level utilization.

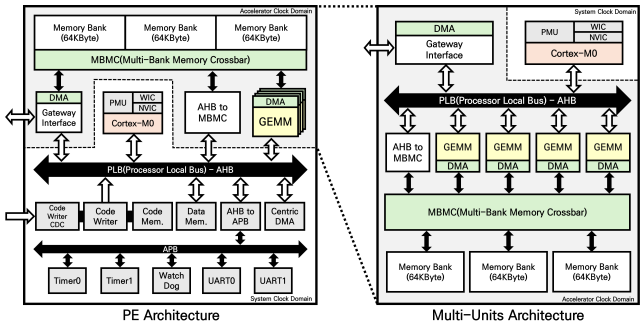


Figure 1: The System Architecture of the Processing Element

2 System Architecture

2.1 Processing Element Overview

As shown in Figure 1, each PE integrates four GEMM units, a Arm Cortex-M0, three on-chip SRAM banks, and an MBMC interconnect. Unlike conventional static control, the Cortex-M0 dynamically configures tiling parameters and data transfer schedules based on workload size and memory constraints. It issues optimized DMA commands, handles completion interrupts, and adjusts schedules to maximize overlap between computation and communication. This event-driven approach reduces unnecessary CPU intervention, improves energy efficiency, and enables intelligent parallelism across the system.

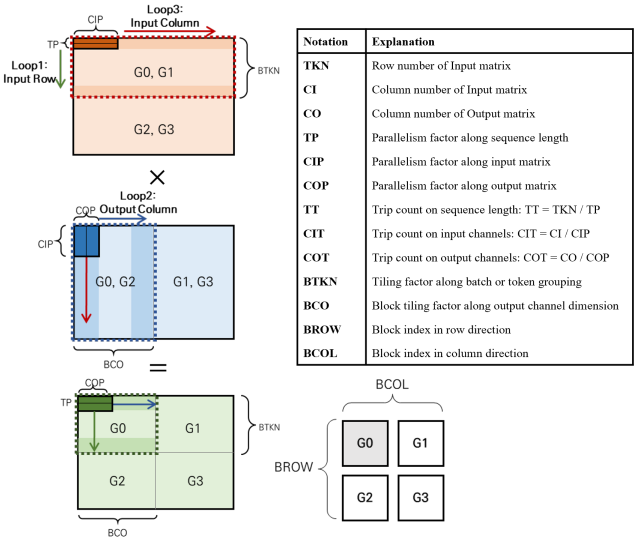
2.2 General Matrix Multiplication Unit

The GEMM unit implements a 4-stage pipelined MAC unit. Each MAC engine consists of six 8-bit multipliers and seven 32-bit accumulators, with eight such engines operating in parallel to produce a 2×4 output tile per cycle. To scale to AI/HPC workloads, four GEMM units are instantiated per PE. Data are loaded from on-chip SRAM banks into local buffers, which cache frequently reused operands and sustain the compute pipeline.

Matrix multiplication follows three nested loops over tokens (TKN), input channels (CI), and output channels (CO). All loops are tiled to maximize data locality. An Output-Stationary (OS) dataflow is adopted, storing partial sums locally to reduce intermediate memory traffic (Figure 2). Tiling ensures that data resides longer in local buffers, reducing off-chip access latency.

2.3 Multi-Bank Memory Crossbar

The central innovation is the MBMC interconnect module. In conventional single-bus architectures, four GEMM units must sequentially load operands, resulting in load latency proportional to the



Notation	Explanation
TKN	Row number of Input matrix
CI	Column number of Input matrix
CO	Column number of Output matrix
TP	Parallelism factor along sequence length
CIP	Parallelism factor along input matrix
COP	Parallelism factor along output matrix
TT	Trip count on sequence length: TT = TKN / TP
CIT	Trip count on input channels: CIT = CI / CIP
COT	Trip count on output channels: COT = CO / COP
BTKN	Tiling factor along batch or token grouping
BCO	Block tiling factor along output channel dimension
BROW	Block index in row direction
BCOL	Block index in column direction

Figure 2: The notations of tiled matrix multiplication

number of units. In contrast, MBMC allows each GEMM to concurrently access distinct SRAM banks, effectively parallelizing memory loads. This reduces load latency by up to 4× in theory, enabling earlier compute initiation and higher throughput. Arbitration among concurrent requests is handled by round-robin logic, ensuring fairness and preventing starvation.

3 Experiment

The proposed design was implemented in Verilog RTL and evaluated using Synopsys VCS at 200 MHz with identical SRAM timing models across all comparisons. Each architecture consists of four GEMM units and three 64-KB on-chip SRAM banks. Benchmarks included various matrix sizes (TKN×CI, CI×CO), with measurements of total cycles, data load cycles, and compute cycles. Tiling parameters were drawn from 149 candidate sets representative of real AI workloads and memory constraints, narrowed to 40 unique cases after deduplication by operation count (CI×CO×TKN). From these, throughput and utilization were derived.

Across workloads, MBMC significantly reduced GEMM idle cycles compared to the single-bus baseline. As matrix sizes grew, the single-bus design exhibited sharp increases in idle time due to sequential load conflicts, while MBMC sustained high utilization by enabling parallel bank accesses. For example, in workload (TKN=64, CI=64, CO=64), idle cycles dropped from 5936 (single bus) to 2968 (MBMC). On average, MBMC reduced idle cycles by 1.49×. Figure 3 summarizes idle cycle ratios across workloads, highlighting ability of MBMC to alleviate memory bottlenecks.

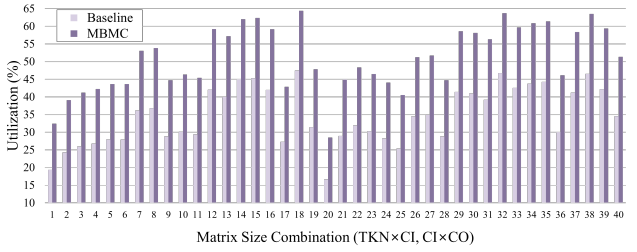


Figure 3: Variation in GEMM idle-cycle ratios with increasing matrix dimensions

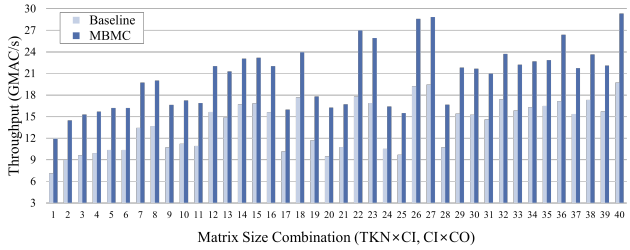


Figure 4: Comparative throughput analysis across diverse matrix-dimension combinations

The performance results confirm that MBMC consistently outperformed the single-bus baseline. By reducing load latency, MBMC shortened execution time and advanced compute start times. For instance, in workload (TKN=64, CI=64, CO=64), throughput improved from 7.12 GMAC/s (single bus) to 11.94 GMAC/s (MBMC), yielding a 1.68% gain. Across 40 cases, MBMC achieved an average throughput improvement of over 1.49×. Figure 4 presents throughput comparisons across workloads, demonstrating the consistent performance benefits of MBMC.

4 Conclusion

This paper presented a GEMM accelerator architecture incorporating a MBMC to address the memory bottleneck inherent in GEMM workloads. By enabling parallel access to multiple SRAM banks and coupling this with event-driven scheduling by a Cortex-M0, the proposed design minimized idle cycles and maximized compute utilization. Simulation results confirmed significant reductions in idle time and throughput improvements compared to a conventional single-bus baseline. This study demonstrates a structural solution to fundamental bottleneck of matrix multiplication operation and provides a foundation for future extensions, including scaling to multiple PEs and compiler-driven optimizations for large-scale AI and HPC deployments.

References

[1] Melvin E. Conway. 1963. A multiprocessor system design. In *Proceedings of the November 12-14, 1963, Fall Joint Computer Conference* (Las Vegas, Nevada) *(AFIPS '63 (Fall))*. Association for Computing Machinery, New York, NY, USA, 139–146. doi:10.1145/1463822.1463838

[2] Cong Guo, Chiyue Wei, Jiaming Tang, Bowen Duan, Song Han, Hai Li, and Yiran Chen. 2025. Transitive Array: An Efficient GEMM Accelerator with Result Reuse. In *Proceedings of the 52nd Annual International Symposium on Computer Architecture (ISCA '25)*. Association for Computing Machinery, New York, NY, USA, 990–1004. doi:10.1145/3695053.3731043

[3] Eric Qin, Ananda Samajdar, Hyoukjun Kwon, Vineet Nadella, Sudarshan Srinivasan, Dipankar Das, Bharat Kaul, and Tushar Krishna. 2020. SIGMA: A Sparse and Irregular GEMM Accelerator with Flexible Interconnects for DNN Training. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 58–70. doi:10.1109/HPCA47549.2020.00015

[4] Md Mizanur Rahaman Nayan, Ritik Raj, Gouse Basha Shaik, Tushar Krishna, and Azad J Naeemi. 2025. Axon: A Novel Systolic Array Architecture for Improved Run Time and Energy Efficient GeMM and Conv Operation with On-Chip im2col. In *2025 Design, Automation Test in Europe Conference (DATE)*. 1–7. doi:10.23919/DATE64628.2025.10992860

[5] Mohammadreza Soltaniyeh, Richard P. Martin, and Santosh Nagarakatte. 2022. An Accelerator for Sparse Convolutional Neural Networks Leveraging Systolic General Matrix-matrix Multiplication. *ACM Trans. Archit. Code Optim.* 19, 3, Article 42 (May 2022), 26 pages. doi:10.1145/3532863

[6] Shouyi Yin, Xianqing Yao, Tianyi Lu, Dajiang Liu, Jiangyuan Gu, Leibo Liu, and Shaojun Wei. 2017. Conflict-Free Loop Mapping for Coarse-Grained Reconfigurable Architecture with Multi-Bank Memory. *IEEE Transactions on Parallel and Distributed Systems* 28, 9 (2017), 2471–2485. doi:10.1109/TPDS.2017.2682241