Optimizing Matrix Operations for Machine Learning:

Analysis of AMD's MI100 GPU

Author: Arjun Sarkar

Date: April 7th, 2025

Abstract

The need for purpose-built, specialized hardware that can efficiently execute large-scale matrix operations is a result of the rapid development in artificial intelligence (AI) and machine learning (ML). Deep learning models advance along with AI and machine learning technologies, especially in the areas of tensor calculations. In particular, general matrix multiplication (GEMM), fused multiply-accumulate (FMA) operations, and other metric computations have to be done using highly parallelized, energy economical, and high throughput processes. The GPU, or even the traditional GPU, was invented as a device to handle graphically intensive workloads. It often does not capture all the possible value when trying to optimize for ML operations. The AMD Radeon Instinct MI100 with CDNA architecture gets around Instinct GPUs issues in execution units with its matrix cores which are dedicated hardware blocks built to optimize tensor operations. These matrix cores accept F16 and BF16 precision and concentrate with the SIMD units of the GPU to perform deep learning model training and inference focused FMA needed during training and inference execution FMA. This paper focuses on the MI100's matrix cores architecture with an emphasis on the instruction set, the data path, the execution model, and consider how AMD supports computation. Moreover, this discussion will outline the crucial functional units relating to the architectural trade-offs, implementation methodologies essential to scalable machine learning computation on general-purpose accelerators and how matrix cores support the high-throughput computation required for machine learning workloads.

Introduction

The evolution of artificial intelligence and machine learning technologies has increased the need for efficient large scale hardware parallel execution of matrix operations. Almost all of the deep learning workloads, spanning from computer vision to natural language processing, involve matrix multiplications and tensor contractions as key operations. These tasks are quite demanding in terms of computation, and their execution times depend greatly on the hardware architecture **[2]**. Due to such requirements, modern processors are now equipped with dedicated functional units that speed up matrix operations, especially those encountered in ML workloads. In machine learning, matrix computations are usually located in the layers of deep neural networks where convolutional and fully connected layers as well as the recurrent layers. These layers perform General Matrix Multiplication and fused multiply accumulate operations, which can create serious performance problems if they are not optimized on hardware. With the evolution of transformer-based architectures, increasing a model's size and complexity requires the capability to execute large scale matrix manipulations in parallel, with efficient throughput processing **[2],[5]**. The MI100 GPU has been designed specifically for high-performance computing and machine learning and follows AMD's CDNA architecture. The MI100, unlike traditional GPUs, does not perform raster and display operations, but executes computational workloads that require deep learning MI100s, CAD specific GPU modules, and focus on computer-aided design standards. These comprise functional blocks that implement parallel processors for deep learning. In addition, they employ advanced precision mixed-precision arithmetic optimized for tensor computations (critical in training and inferring neural networks), also known as AMD's built-in matrix cores **[3,4]**. AMD traditionally integrates matrix compute architecture directly with SIMD-based (single instruction multiple data) execution model. The MI100 matrix cores are constructed to utilize both FP16 and BF16 precisions with high throughput FMA instruction execution for rapid training and inference with negligible loss in accuracy **[3,6]**. In this paper, we will discuss the MI100 architecture focusing on the matrix core functional units that serve to speed up machine learning operations at the micro level. The focus will be on providing a detailed technical description of the operation of these matrix units, their implementation with the instruction set architecture and their optimization for dominant matrix operations used in deep learning **[1,3,6]**.

High-Throughput Computations and Instruction Set Architecture

To meet the needs of deep learning workload acceleration, AMD integrated a new block of matrix instructions with high speed processing into the CDNA instruction set architecture. These new instructions target the matrix core functional units of the MI100 GPU, supporting efficient FMA on vectorized and tensor data structures. AMD's matrix instructions are specified using the Matrix Fused Multiply-Add operation. The MFMA instruction family is concerned with the computation of multiplication, accumulation, and other operand combinations within sub-matrix tiles, taking advantage of data level parallelism. These instructions support multiple precisions which include FP32, FP16 and BF16, and also have hardware mixed precision support, that is, lower precision formats for the inputs and higher precision for the accumulators, ensuring accuracy during the training phase **[3,6]**. The ISA part of MI100 executes using SIMT (Single Instruction, Multiple Threads) model. The MI100's Compute Units embedded with MFMA instructions are matrix cores to compute units at the side of the SIMD (Single Instruction Multiple Data) parallel. Each CU contains four SIMD units, and MFMA operations rely on dedicated hardware for parallel matrix execution across these SIMDs **[4]**. Inside every SIMD lane, Wave Matrix Multiply Accumulate implementation units are integrated for array-wise decomposition of block matrix multiplications. This enables in reducing register overhead along with reducing latency all while providing plenty of unused instructions to be executed, therefore, allowing for high performance for training deep neural networks **[3]**. Every MFMA instruction causes the execution of a precise set of microcode steps for operand fetch, compute, and write back to the register file to be done with the least amount of stall cycles. One of the most unique microarchitectural features is an improvement due to register tiling where the matrix operands stored in vector registers are reused over several FMA cycles. This reduces the access frequency to memory and improves temporal locality. Memory access patterns, especially in the context of deep layers of machine learning models, pose more challenges due to their considerable size **[6]**. These registers are arranged into vectors of 64 elements so that all of them can be used by the 64-thread wavefront. Also, AMD's high-bandwidth Infinity Fabric and HBM2 memory offer separate bidirectional data buses to other modules for efficient data movement or transmission which improves bandwidth of 1.23 TB/s as this is 20% faster than prior AMD generations **[4]**. Continuous data stream without stalling the compute units is critical for high MFLOP performance for machine learning workloads. The microarchitecture of the AMD MI100 is driven by device-level machine learning application constraints, primarily optimizing the instruction set architecture for the MFMA instruction and matrix core interconnections. The operational needs are mainly for training and inference workloads in deep neural networks. This is done by enabling efficient tile-parallel matrix multiplications with SIMT-style parallel, mixed-precision arithmetic at extremely high throughput.

At a micro-architectural level, the MI100 GPU's execution units are fine-tuned to maximize solution throughput. Memory bandwidth is utilized more effectively at the cost of latency and power consumption. The hierarchical memory architecture consisting of registers L1 cache, L2 cache, and high bandwidth HBM2 memory mitigates memory access bottlenecks by parallelizing the large matrix multiplication task. Further, multiple operations are executed simultaneously, requiring concurrent access to weight and activation storage. Partial results during MFMA operations are stored in registers to minimize shared and global memory access, increasing availability **[6]**. During FMA cycles, register tiling is used to enhance operand tile reuse, optimizing data locality while lowering memory bandwidth

demand. This is crucial in executing matrix multiplications over highly dimensional layers in either convolutional or transformer networks **[6]**. Moreover, advanced data movement algorithms that employ prefetching and operand reuse enable dynamic bus management by the compute units. This is performed to maintain load on the execution pipelines as it minimizes idle cycles **[3]**. GPU cores and the shared memory components are interconnected for faster data exchange over memory layers by Infinity Fabric which is a high bandwidth interconnect. To avoid memory stalls as well as contention, this fabric guarantees deterministic data routing paths and parallel SIMD lane synchronized access **[4]**. More precisely, the separate read and write pathways permit data flow in both directions. This is significant for training because weights can be loaded and gradients can be written back computation without pausing processing. Stacking HBM2 with easy access and bank level parallelism allows for merging of computation and memory I/O operations, leading to uninterrupted data streaming into MFMA pipelines. As discussed in **[2]** and **[5]**, the integration of memory and compute resources is beneficial for deep learning models because it reduces the time required for training and increases the throughput per watt used. During dense matrix operations which are characteristic of the transformer attention heads. The design exploits coalescing where adjacent accesses from the same thread are merged to enhance efficiency in accessing memory. This is particularly crucial in deep learning when we consider batch processing because large input tensors have to be dealt with simultaneously. Such as increases the effective throughput but reduces the energy consumed for each operation, which is in line with the AI supercomputing design objectives **[6]**. Therefore, AMD enables efficient and scalable matrix computations with AMD's custom microarchitecture through register tiling, SIMD level synchronization, HBM2 optimization, and integration of Infinity Fabric. These systems are solely in meeting the growing needs for deep learning in terms of high volume inferencing and training within multi-layered neural networks.

Matrix Core Architecture: Optimizing Machine Learning Workloads

The matrix core functional unit is extremely vital to AMD's MI100 GPU architecture, it's a specialized piece of hardware which is carved into each compute unit. This device is created to help with matrix operations fundamental to MLOPs. This executor unit is designed for matrix MFMA and performs the range of actions in the instruction set architecture. Essentially, it works on constructs that maximize information reuse, restrict memory, and use parallelism. Each of the compute units of the MI100 consist of SIMDs and the matrix core logic is integrated into these SIMDs. Matrix cores process matrix tiles using a warp-level execution model while scalar & vector execution paths follow a more traditional approach. The tiles are worked on in a pipelined manner through a specialized WMMA (Wave Matrix Multiply Accumulate) engine that interprets MFMA instructions, sends micro operations, and divides the processes into stages decoding, operand fetch, multiply, accumulate, and write back **[3,6]**. The matrix core utilizes appropriate datapaths and custom ALUs as tile oriented arithmetic units. Each WMMA engine works with small matrix fragments that are stored in registers, which further reduces the need for L1/L2 caches. This structure enables high instruction level parallelism, reducing competition between parallel tasks for registers, and makes use of banked register files with multiple simultaneous read/write ports per thread in a wavefront **[4]**. With the control bypassing, loop unrolling, and operand prefetching, load-balanced SIMD lanes can operate on matrix multiplications for multiple cycles without stalling their lanes. This is crucial for achieving high speed on large matrix multiplications in the innermost layers of deep neural networks. The matrix core's efficient data reuse tactic is perhaps its most distinguishing feature. Register-level tiling allows multiple FMA cycles on the same operand tiles while accessing registers, minimizing memory traffic and taking advantage of fast register access. The operand fetch units are designed to stall and preload entire wavefront-wide tiles into vector registers prior to the compute stage, mitigating the latency spent on repeatedly accessing the memory **[6]**. The MI100 also adds support for mixed-precision accumulation. Inputs can be kept as FP16 or BF16 to lessen the memory requirements, but accumulation takes place in FP32 to maintain the preferred level of stability. The matrix core does this promotion automatically without software needing to touch it. This is useful in deep learning training as the scaling of precision must stay aggressive but precise accuracy is required **[5]**. The MI100, according to AMD technical documentation, manages to provide an approximate 11.5 TFLOPs (FP64) and 184.6 TFLOPs (FP16) of peak performance attributed mostly to the highly parallel structure of the matrix core **[2,4]**. Unlike the scalar and vector execution units which perform general-purpose processing on 1D or 2D data, the matrix core functional unit focuses on optimization for 2D tile operations. Scalar units are preferable in control flow and branching logic while vector units are left to 'pick up the slack' left from traditional parallel computation. Still, the energy efficiency of the matrix cores far surpasses the performance of scalar or vector units when dealing with large matrix multiplications, a common operation in machine learning models. Matrix cores outclass vector lanes for ML workloads in terms of performance-per-watt and performance-per-area. Allen et al. **[6]** for example reported up to three times more efficient matrix-heavy deep learning inference with MFMA capable units compared to vector ALUs. The matrix cores built into AMD GPUs are what enable these GPUs to remain relevant against dedicated AI accelerators such as NVIDIA's Tensor Cores and Graphcore IPUs **[5]**. AMD's MI100 integrates a specialized execution engine referred to as matrix core functional unit which provides parallel processing capabilities suited for the rapid performance and efficiency required in heavy matrix computation. These cores create the foundation for deep learning training and

inference workloads by integrating tile-based execution, mixed-precision compute, and deep pipelining in each SIMD lane. Their architectural design shifts away from general purpose vector units toward SIMD pipelines toward workload accelerations.

As an extension to the former passage, in this section we will discuss in depth on AMD's matrix core and how it aids in high-throughput computation for machine learning workloads. The discussion is based on principles on its scalability, energy consumption, and efficiency in performing large-scale ML training and inference tasks. To add to the parallelism already offered by AMD's matrix cores, the MI100 uses its architecture to facilitate data throughput for machine learning workloads through efficient data management techniques. Perhaps the most important issue in matrix core optimization is the data movement, especially for the large matrices' movements across the various levels of memory. The MI100 matrix cores utilize tile cache and register level tiling with warp execution and tile level register caching that is added to alleviate the impact of fetches on L1 and L2 cache hits **[4]**. In large scale machine learning operations that involve rapid transformations of the high dimensional matrices, this is critical. According to Allen et al **[6]**, managing the tile caches adds the ability to exploit reuse without being limited to the slow deep caches that are present in the modern systems that are built for deep learning. Also, observe that MI100's matrix cores implement dynamic scheduling of compute tasks at the SIMD (Single Instruction, Multiple Data) level, which improves load balancing throughout the GPU. Such scheduling improves task distribution among all SIMD lanes and keeps all SIMD idle times to a minimum during execution, especially during large matrix multiplications where workloads tend to be unevenly distributed. This is important for the training of very large neural networks because it enables the MI100 to perform highly complex, large-batch, neural network training with greater throughput and lower power relative to conventional GPUs. Maintaining an adequate level of energy efficiency for prolonged training sessions that are common in ML workloads is made possible by dynamically scheduling tasks, greatly improving energy efficiency. As noted in the cited work by Peng et al. **[5]**, the scalability of AMD's matrix cores provides sustained performance as the model or dataset size increases, while executing dense matrix operations more energy-efficiently and rapidly than a conventional GPU. With regard to the tasks provided by deep learning, the MI100's support of mixed-precision calculations improves the efficiency of such tasks like the training of neural networks. The MI100's technique of raising precision during accumulation (FP16 or BF16 inputs with FP32 accumulations) handles this challenge by enabling the model to gain accuracy at essential phases of training while still benefitting from lower precision memory. This is a form of mixed precision, emphasized by Wang et al. **[2]** as a reason lower bounds on train time were removed, to enable faster matrix operations without affecting computational stability. Models also do not experience memory bottlenecks common with larger models since the mixed precision mitigates the saturation of memory bandwidth. Moreover, the MI100 matrix cores significantly improve the throughput of matrix operations which enhances the competitiveness of AMD GPUs for real-time ML and AI workloads. The MI100's performance of up to 184.6 TFLOPs (FP16) increases its usefulness for tasks like parallel deep neural network training and inference **[2]**. Besides the hardware features, AMD's MI100 matrix core architecture is augmented with software tools that let developers capitalize on its computational power. One such tool is the AMD Matrix Instruction Calculator **[7]**, which helps engineers optimize the use of MFMA instructions by data types (FP16, BF16, INT8) and target matrix tile sizes. This tool supports the lower programming level of matrix cores by allowing developers to design register tiling, operand positioning, and instruction selection within WMMA engine's compute model. It illustrates to developers how the matrix core MFMA instruction execution unit decodes MFMA instructions

and workload distribution on wavefront threads, enabling efficient register reuse while increasing L1 and L2 cache bandwidth pressure. AMD's is claiming that by coupling such software with tools of this nature, highly-optimizable ML workloads, particularly in the context of training large convolutional networks and transformers, can be accelerated. Wang et al. **[2]** further observed this in benchmarking experiments across deep learning workloads where they reported high throughput and power efficiency improvements while MFMA instructions were utilized optimally. Also, in the case of AMD's compiler stack for ROCm, it integrates with the matrix instructions to construct MFMA-based kernels where balanced workload distribution across matrix cores as SIMD lanes are super utilized and memory stalls are reduced. Compared to the offloading techniques to ALU vectors, cores with MFMA matrix instructions applied have demonstrated much higher scalability in inference and training pipelines with greater parallelism in computation. This is strikingly clear in large-scale benchmarking scenarios like the Open Compass Project, where AMD GPU accelerators showcased competitive performance-per-watt with NVIDIA in machine learning centric workloads **[2,5]**. AMD did not only optimize the hardware with parallelism using the matrix cores, but also built a comprehensive software framework that logically automates workload partitioning and low-level tuning, achieving these benchmarks. While scalar and vector execution units cater to general-purpose tasks, the matrix cores are optimized for ML workload specifics, especially the large matrix multiplications done in neural network layers. As mentioned by Otterness and Anderson **[1]**, this specialization enables AMD GPUs to handle matrix-heavy computations with far greater efficiency compared to general-purpose GPUs, thus, addressing the rapid training cycles required by contemporary machine learning tasks. In addition, the energy efficiency of the matrix cores reported by Allen et al. **[6]**, is remarkable. Matrix cores have a distinctive edge, being able to undertake computations with such low energy usage, especially when scaling machine learning workloads in data centers and edge devices where energy and heat emission are of the utmost importance. AMD's design decisions regarding power distribution across the SIMD lanes also yield superior performance per-watt than older scalar or vector units. This performance allows machine learning models to be trained and deployed at scale without drastic increases in power use, critical for real-time applications in autonomous vehicles, healthcare, finance, and large-scale industries.

To summarize, MFMA matrix cores in AMD's MI100 GPU are built with algorithmic logic meant to address the extreme throughput needs imposed by machine learning workloads. The MI100 matrix cores outperform versatile computer units through advanced techniques like register-level tiling, mixed-precision arithmetic, dynamic task scheduling, and smart data with these metrics are required to scale modern AI applications. Based on these results, AMD has strengthened the MI100's position in the AI accelerator market, where it competes strongly with NVIDIA's Tensor Cores and other acceleration technologies **[5]**.

Matrix Core Performance: Wavefront Scheduling and Control Logic

The performance of AMD's MI100 GPU matrix core functional unit is highly dependent on the matrix instruction scheduling and execution. In addition to instruction scheduling and control logic, another key factor in achieving high throughput for all SIMDs is managing dense matrix workloads for deep learning types. AMD's architecture uses wavefronts, similar to NVIDIA's sets of 32 threads executed simultaneously. To achieve maximum parallelism, the matrix core depends on effective wavefront scheduling. Each compute unit uses SIMD units and a corresponding scheduler tasked with determining the next wavefronts to issue based on operand availability and execution unit readiness. The hardware scheduler uses a strategy such as round-robin aligned with ready wavefronts capable of issuing MFMA instructions first. To mitigate stalls, the scheduler is careful regarding register dependency and data hazard avoidance. With matrix instructions, a tile wide scheduling is often employed, requiring proper synchronization with operand readiness across the wavefront **[3,6]**.The MFMA units work with multi-cycle operations that need control over the core matrix's core pipeline. The control unit administers the instruction issue, operand fetch, and result commit within several clock cycles. Control signals are developed for partial pipeline steps like operand decoding, multiplication, accumulation and writing to the vector register file. The control unit ensures data flow correctness by implementing scoreboard-based hazard detection. This prevents overlapping instructions from prematurely gaining access to a shared destination register, leading to incorrect results. In contrast, correct results are ensured by keeping track of ongoing operations in each SIMD lane **[3]**. The ROCm compiler from AMD unconsciously contributes to the hierarchical shift of matrix operations to MFMA's lower level **[1]**. The compiler detects matrix multiplication and performs adequate emission of MFMA instructions within operated levels of loop unrolling and tile decomposition. Instruction latency, which is hidden by overlapping multiple MFMA operations, is also targeted by software pipelining. One tile is multiplicated with operands for the succeeding tile pre-staged in registers, guaranteeing execution unit activity as well as preventing stalling **[2,6]**.

If data dependencies are minimized, matrix instructions can take advantage of instruction-level parallelism (ILP). The control logic takes advantage of this ILP whenever possible by dynamically reordering instruction issues. For example, if the operand fetch and register ports are free, two MFMA instructions that target different registers can be issued serially. Hazard detection logic guarantees that instructions are not issued if their operands are not accessible in time. This is critical for matrix accumulations where the result of one MFMA instruction is the input to the next. In such cases, the scheduler stifles the machine's ability to execute instruction streams to suppress register overwrites **[3,4]**. The matrix core displays some optimized performance design, but its performance can be limited by a number of issues such as the active wavefronts which are limited by the excessive pressure on the register file and careless scheduling of instruction executions increases latency as well as reduces throughput. It is possible to tackle these issues for instance, the use of operand preloading during computation and breaking down large operations into sub-tiles for registers. In the performance evaluations by Allen et al. **[6]**, show careful scheduling and control optimization with regard to the wavefront processor arrays achieves up to 90% peak efficiency on matrix kernels. With regards to larger matrix operations, there is a need for barrier synchronization when several wavefronts work together towards a single goal **[3]**. In the MI100, there is support for barrier synchronization at the hardware level to manage wavefronts. These

barriers guarantee that every thread stops at a point before proceeding for all participating threads, which is a necessity for the tiled matrix multiplication model that occurs at different control units **[5]**. Moreover, control divergence is a non-issue due to the general lack of conditional branching matrix workloads because these types of workload escalate warp divergence(threads needing to perform separate tasks) and stall execution lanes. Scalar units assume the responsibility of control logic for these cases, enabling computation within a matrix core. Execution control for the MI100's matrix core control architecture is balance oriented, meaning that throughput and latency is high for matrix execution. Real-time scheduling for wavefronts and dynamic hazard avoidance allow AMD control logic to optimally deal with MFMA instructions in real-time. These conditions are crucial to realizing the potential of the matrix core during workloads meant to accelerate machine learning tasks.

Conclusion

The AMD MI100 GPU, based on the CDNA architecture and its matrix multiplication units, demonstrates how far the industry has come in the design of hardware to accelerate machine learning tasks. AMD delivers a computing paradigm with high throughput and low latency for deep learning by embedding the Matrix Fused Multiply-Add (MFMA) instruction set into SIMT execution model, including, but not limited to, parallelism at the level of threads. The matrix cores of MI100 are integrated into compute units with SIMD lanes where efficient tile-based execution is possible, also data reuse through register tiling and mixed precision arithmetic is done which is becoming vital with increasing neural network computation requirements. This paper has focused on the architectural and microarchitectural factors that provide such performance improvements from smooth instruction flow within the WMMA engine to memory hierarchies and data movement through Infinity Fabric and HBM2 memory. The results show that scalable parallel matrix processing in MI100 comes at very low costs in terms of memory access and energy expenditure. These features make MI100 a strong candidate for performing deeper transformers and other complex deep learning model inference and training. The matrix core architecture from AMD showcases what the future of AI hardware looks like. It features highly specialized compute engines configured to perform matrix operations at peak efficiency. Thus, decisions around AI architecture will be fundamental to solving the problems posed by increasingly complex and huge AI models.

References

[1] N. Otterness and J. Anderson (2020), “AMD GPUs as an Alternative to NVIDIA for Supporting Real-Time Workloads,” https://www.cs.unc.edu/~anderson/papers/ecrts20a.pdf (https://www.cs.unc.edu/%7Eanderson/papers/ecrts20a.pdf)

[2] M. Y. Wang, J. Uran, and P. Buitrago (2023), “Deep Learning Benchmark Studies on an Advanced AI Engineering Testbed from the Open Compass Project,” Practice and Experience in Advanced Research Computing, https://dl.acm.org/doi/pdf/10.1145/3569951.3597596. (https://dl.acm.org/doi/pdf/10.1145/3569951.3597596)

[3] “AMD Instinct MI100 instruction set architecture,” (2020), AMD Technical Documentation. https://www.amd.com/content/dam/amd/en/documents/instinct-tech-docs/instruction-set-architectures/instinct-mi100-cdna1-shader-instruction-set-architecture.pdf (https://www.amd.com/content/dam/amd/en/documents/instinct-tech docs/instruction-set-architectures/instinct-mi100-cdna1-shader-instruction-set-architecture.pdf)

[4] “AMD CDNA Architecture”, (2020), The All-New AMD GPU Architecture for the Modern Era of HPC & AI. https://www.conferenceharvester.com/uploads/harvester/VirtualBooths/13396/NKBNOCXO-PDF-2-416892(1).pdf (https://www.conferenceharvester.com/uploads/harvester/VirtualBooths/13396/NKBNOCXO-PDF-2-416892(1).pdf)

[5] H. Peng, C. Ding, T. Geng, S. Choudhury, K. Barker, and A. Li, (2024), “Evaluating Emerging AI/ML Accelerators: IPU, RDU, and NVIDIA/AMD GPUs,” https://research.spec.org/icpe_proceedings/2024/companion/p14.pdf (https://research.spec.org/icpe_proceedings/2024/companion/p14.pdf)

[6] Allen, M., et al. (2022). "Characterizing the Performance, Power Efficiency, and Programmability of AMD Matrix Cores." OSTI Technical Report. https://www.osti.gov/servlets/purl/2345982 (https://www.osti.gov/servlets/purl/2345982)

[7] “ROCm/amd_matrix_instruction_calculator: A tool for generating information about the matrix multiplication instructions in AMD RadeonTM and AMD InstinctTM accelerators,” GitHub, 2025. https://github.com/ROCm/amd_matrix_instruction_calculator (https://github.com/ROCmamd_matrix_instruction_calculator)