RoadBlock: Rethinking Tensor Core Microarchitecture for Emerging Microscaling Format Support on GPUs

Nikhil Rout
Vellore Institute of Technology, Chennai
nikhilrout97@gmail.com

Blaise Tine
University of California, Los Angeles
blaisetine@cs.ucla.edu

1 Introduction

As model parameters continue to grow dramatically in size, even with kernel engineering optimizations [10, 24] and KV-cache compression [22, 23] techniques, low-precision Quantization-Aware Training (QAT) [2, 7] and Post-Training Quantization (PTQ) [13, 21] schemes remain the most effective solution for serving models [9]. To this end, multiple recent works proposing custom Microscaling (MX) formats and quantization schemes such as AMXFP4 [12], MX+ [11], BlockDialect [8], RaZeR [3], Four Over Six [4], and IF4 [5] have emerged dedicated to pushing the Quantization-Perplexity Pareto Frontier beyond the industry-adopted Open Compute Project (OCP) MX formats [6, 15, 16]. However, the hardware evaluations for the majority of these formats are limited to MAC Unit prototypes intended to be plugged into DNN accelerators' Systolic Array PEs. On the other hand, most block-quantized training and inference recipes and software stacks are being developed from the perspective of NVIDIA Blackwell [14] and AMD CDNA4 [1] GPU Tensor Cores, as they are currently the only commercially available hardware with native MX format acceleration.

To bridge this gap, we propose ***RoadBlock***, an open-source framework[1] for accurate end-to-end modeling and evaluation of custom MX formats and their unconventional quantization schemes in Tensor Core Units (TCUs), built on top of the highly flexible and programmable RISC-V extended Vortex GPGPU [19] platform. Key contributions of this work are summarized as follows:

- RoadBlock enables custom MX format design space exploration by exposing data size (d), scale factor size (w), and block size (k) as configurable parameters, covering compression ratio, block-level granularity, multi-level scaling hierarchies, and their impact on metadata overhead, arithmetic intensity, and compute/memory bandwidth utilization in a GPU Tensor Core setting.
- RoadBlock automatically derives the tile dimensions of scale-factor metadata and indexing using generalized equations for any given format and threads/warp count.
- RoadBlock proposes a lightweight metadata-register TCU-SRAM duplication methodology for transferring metadata from global memory (GMEM) to the TCU Fused Dot Product grid directly, overcoming obstacles present on currently available GPUs.

2 RoadBlock Microarchitecture

Fig. 1 shows a top-level overview of the extensions RoadBlock makes to a Vortex GPGPU SIMT Sub-Core to support custom MX formats in its Tensor Cores.

[1]Source code available at https://github.com/vortexgpgpu/vortex/tree/oscar-roadblock

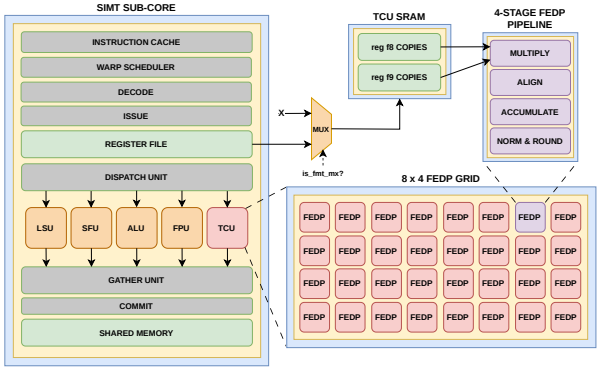

Figure 1: RoadBlock-extended Vortex GPGPU SIMT Sub-Core

MX-enhanced Fused Dot Product (FEDP) Unit. The fundamental dot product for two MX-compliant vectors A and B of length k is calculated as:

$$\mathrm{Dot}(A,B) = X^{(A)}X^{(B)}\sum_{i=1}^{k}\left(P_i^{(A)} \times P_i^{(B)}\right) \tag{1}$$

where $X^{(A)}$, $X^{(B)}$ are block scale factors and $P_i^{(A)}$, $P_i^{(B)}$ are the i-th elements of A and B respectively. RoadBlock adopts and enhances the 4-cycle Ten-Four [17] mixed-precision FEDP microarchitecture for its RTL implementation. Since Ten-Four FEDPs perform early accumulation of the "C" addend alongside the vector products, all required scale-factor computations and shifts are performed and applied to each lane's significand multiplication result in the first `MULTIPLY` pipeline stage itself.

MX Scale Factor Metadata Tiling. Within the constraints of Vortex's RISC-V SIMT programming model, the 32 registers per thread budget is divided between matrices A (`f10-f17`), B (`f21-f28`), and C/D (`f0-f7`), yielding a maximum tile size of 16x16 at NT=32. Additionally, contiguous low-precision elements are packed along the `K` dimension in 32-bit bundles. Fig. 2 demonstrates how RoadBlock derives the required per-thread metadata for a given format. Assuming NVFP4 as the format with finest-granularity (`d=4, w=8, k=16`), applying the formulae results in 64-bits of metadata per thread, requiring two additional registers (arbitrarily chosen as `f8, f9`) for scale factor storage.

MX Metadata Delivery and Storage. Since Vortex TCU WMMA instructions can only have 3 input source operands, MX scale-factor metadata needs to be fetched and delivered to the FEDP grid directly. RoadBlock employs a lightweight TCU-SRAM for this purpose, written by duplicating registers `f8, f9` via a simple MUX that activates

Given

No. Regs/Operand $(NR) = 8$, *No. Threads/Warp* $(NT) \in \{4, 8, 16, 32\}$
d = *element bits*, w = *scale bits*, k = *block size*

Tiling (per-warp work chunk)

$$tile\_cap = NT \times NR = 8\,NT \tag{2}$$

$$\ell_g = \log_2(tile\_cap) = \log_2(8\,NT) = 3 + \log_2(NT) \tag{3}$$

$$tileM = 2^{\ell_g - \lfloor \ell_g/2 \rfloor} = 2^{\lfloor \ell_g/2 \rfloor} = 2^{\frac{3}{2}} \cdot 2^{\frac{\log_2(NT)}{2}} = \sqrt{8\,NT} \tag{4}$$

$$tileK = \frac{tile\_cap}{\max(tileM, tileN)} = \frac{8\,NT}{\sqrt{8\,NT}} = \sqrt{8\,NT} \tag{5}$$

MX block scaling

$$tileK_{fmt\_unpacked} = tileK \times \frac{32}{d} = \frac{32\sqrt{8\,NT}}{d} \tag{6}$$

$$num\_blocks_k = \frac{tileK_{fmt\_unpacked}}{k} = \frac{32\sqrt{8\,NT}}{d\,k} \tag{7}$$

$$scale\_bits_k = num\_blocks_k \times w = \frac{32\,w\sqrt{8\,NT}}{d\,k} \tag{8}$$

MX Metadata

$$A\_metadata = tileM \times scale\_bits_k = \sqrt{8\,NT} \times \frac{32\,w\sqrt{8\,NT}}{d\,k} = \frac{256\,w\,NT}{d\,k} \tag{9}$$

$$Since,\ tileN \in \left\{tileM(NT = 8, 32), \frac{tileM}{2}(NT = 4, 16)\right\} \tag{10}$$

$$B\_metadata \in \left\{\frac{256\,w\,NT}{d\,k}, \frac{128\,w\,NT}{d\,k}\right\} \tag{11}$$

$$total\_metadata = A\_metadata + B\_metadata \in \left\{\frac{512\,w\,NT}{d\,k}, \frac{384\,w\,NT}{d\,k}\right\} \tag{12}$$

$$per\_thread\_metadata = \frac{total\_metadata}{NT} \in \left\{\frac{512\,w}{d\,k}, \frac{384\,w}{d\,k}\right\}\ bits \tag{13}$$

Figure 2: Derivation of required per-thread metadata for custom MX format support on Warp-Tiled Tensor Cores

when the WMMA instruction's format field indicates an MX format. The scoreboard treats `f8, f9` as "hidden source registers", preserving data dependency tracking without requiring any additional operand collector ports or bandwidth. This contrasts with NVIDIA's Blackwell architecture, where a private Tensor Memory (TMEM) serves as both accumulator buffer and scale factor metadata store for `tcgen05.mma` MX instructions — restricting programmers from utilizing the maximum MMA tile size without complicated warp-specialized software pipelining [18, 20]. On the other hand, AMD CDNA4 architecture holds MX scale factor matrices in dedicated VGPRs, trading register pressure for operand decoupling. RoadBlock avoids both these pitfalls by keeping MX metadata in TCU-SRAM and only the accumulator in the register file.

3 RoadBlock Host-Kernel ABI

RoadBlock extends the Vortex WMMA ABI with MX metadata pointer arguments, as shown in Fig. 3, carrying the two scale factor metadata buffers for A and B produced during host-side block-quantization. We also prototyped a version interleaving A and B scale factor metadata with their respective main data tiles into a single buffer, but this introduced significant indexing decode overhead, largely negating the saved kernel launch cost.

Additionally, for formats requiring tensor-wise scaling such as NVFP4, the FP32 tensor-wise scale factor dequantization is handled as WMMA epilogue on the FPU lanes while the TCU only operates on the block-wise E4M3 scale factors.

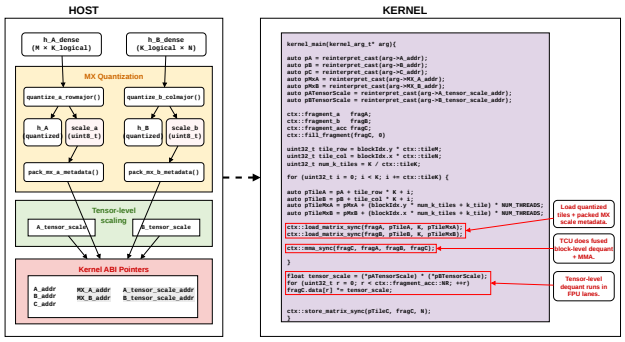

Figure 3: RoadBlock Host-Kernel ABI Flow

4 Preliminary Evaluation

To analyze the overhead incurred by MX formats, we benchmark an SGEMM kernel with `M=N=K=256` across NT∈{4, 8, 16, 32} threads/warp counts using the Vortex cycle-approximate simulator SimX. Early implementation results demonstrate MXFP8 achieves 76–98% of it's ideal 2× throughput gain over FP16/BF16, while NVFP4 only recovers 70–76% of it's theoretical 2× advantage over FP8/BF8, owing to the additional tensor-wise dequantization overhead in Fig. 4.

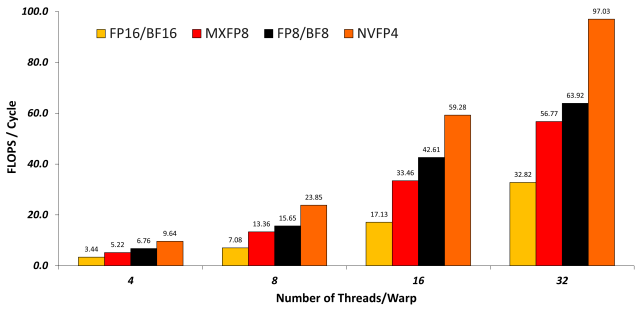

Figure 4: Format-Wise FLOPs/Cycle scaling sweep across NT

5 Future Work & Conclusion

On the hardware side, we plan to prototype the aforementioned emerging formats in RTL to characterize their timing, power, and area efficiency trade-offs. On the software side, we intend on investigating warp-specialized asynchronous pipelined kernels for formats requiring both block-wise and tensor-wise scaling, with the goal of surfacing actionable microarchitecture insights. Finally, as tile sizes grow substantially with the introduction of warpgroup-level WGMMA scheduling in Tensor Cores, the trade-offs between TCU-SRAM and dedicated private TMEM approaches will need to be revisited.

In conclusion, RoadBlock establishes an open-source, configurable framework for exploring and prototyping unconventional MX formats and quantization schemes in a realistic GPU Tensor Core setting, bridging the gap between format proposals and the hardware-software stacks needed to evaluate them end-to-end.

References

[1] AMD. 2025. AMD CDNA 4 Architecture. https://www.amd.com/content/dam/amd/en/documents/instinct-tech-docs/white-papers/amd-cdna-4-architecture-whitepaper.pdf.

[2] Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. 2025. EfficientQAT: Efficient Quantization-Aware Training for Large Language Models. arXiv:2407.11062 [cs.LG] https://arxiv.org/abs/2407.11062

[3] Yuzong Chen, Xilai Dai, Jake Hyun, Chi-Chih Chang, Wonsuk Jang, Yuheng Wu, Thierry Tambe, Jae sun Seo, and Mohamed S. Abdelfattah. 2026. RaZeR: Pushing the Limits of NVFP4 Quantization with Redundant Zero Remapping. arXiv:2501.04052 [cs.LG] https://arxiv.org/abs/2501.04052

[4] Jack Cook, Junxian Guo, Guangxuan Xiao, Yujun Lin, Keith Wyss, Mahdi Nazemi, Asit Mishra, Carlo del Mundo, Tijmen Blankevoort, and Song Han. 2026. Four Over Six: More Accurate NVFP4 Quantization with Adaptive Block Scaling. arXiv:2512.02010 [cs.CL] https://arxiv.org/abs/2512.02010

[5] Jack Cook, Hyemin S. Lee, Kathryn Le, Junxian Guo, Giovanni Traverso, Anantha P. Chandrakasan, and Song Han. 2026. Adaptive Block-Scaled Data Types. arXiv:2603.28765 [cs.CL] https://arxiv.org/abs/2603.28765

[6] Bita Darvish Rouhani, Ritchie Zhao, Venmugil Elango, Rasoul Shafipour, Mathew Hall, Maral Mesmakhosroshahi, Ankit More, Levi Melnick, Maximilian Golub, Girish Varatkar, Lai Shao, Gaurav Kolhe, Dimitry Melts, Jasmine Klar, Renee L'Heureux, Matt Perry, Doug Burger, Eric Chung, Zhaoxia (Summer) Deng, Sam Naghshineh, Jongsoo Park, and Maxim Naumov. 2023. With Shared Microexponents, A Little Shifting Goes a Long Way. In *Proceedings of the 50th Annual International Symposium on Computer Architecture* (Orlando, FL, USA) *(ISCA '23)*. Association for Computing Machinery, New York, NY, USA, Article 83, 13 pages. doi:10.1145/3579371.3589351

[7] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2017. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. arXiv:1712.05877 [cs.LG] https://arxiv.org/abs/1712.05877

[8] Wonsuk Jang and Thierry Tambe. 2025. BlockDialect: Block-wise Fine-grained Mixed Format Quantization for Energy-Efficient LLM Inference. arXiv:2501.01144 [cs.CL] https://arxiv.org/abs/2501.01144

[9] Eldar Kurtić, Michael Goin, and Alexandre Marques. 2026. A First Comprehensive Study of TurboQuant: Accuracy and Performance. vLLM Blog. https://vllm.ai/blog/2026-05-11-turboquant Accessed: 2026-05-15.

[10] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Ion Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th ACM Symposium on Operating Systems Principles (SOSP)*. 611–626. doi:10.1145/3600006.3613165

[11] Jungi Lee, Junyong Park, Soohyun Cha, Jaehoon Cho, and Jaewoong Sim. 2025. MX+: Pushing the Limits of Microscaling Formats for Efficient Large Language Model Serving. In *Proceedings of the 58th IEEE/ACM International Symposium on Microarchitecture (MICRO '25)*. Association for Computing Machinery, New York, NY, USA, 869–883. doi:10.1145/3725843.3756118

[12] Janghwan Lee, Jiwoong Park, Jinseok Kim, Yongjik Kim, Jungju Oh, Jinwook Oh, and Jungwook Choi. 2025. AMXFP4: Taming Activation Outliers with Asymmetric Microscaling Floating-Point for 4-bit LLM Inference. arXiv:2411.09909 [cs.AI] https://arxiv.org/abs/2411.09909

[13] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. In *Proceedings of Machine Learning and Systems*, P. Gibbons, G. Pekhimenko, and C. De Sa (Eds.), Vol. 6. 87–100. https://proceedings.mlsys.org/paper_files/paper/2024/file/42a452cbafa9dd64e9ba4aa95cc1ef21-Paper-Conference.pdf

[14] NVIDIA Corporation. 2024. NVIDIA Blackwell Architecture Technical Brief. https://resources.nvidia.com/en-us-blackwell-architecture/blackwell-architecture-technical-brief.

[15] Open Compute Project. 2023. OCP Microscaling Formats (MX) Specification v1.0. https://www.opencompute.org/documents/ocp-microscaling-formats-mx-v1-0-spec-final-pdf.

[16] Bita Darvish Rouhani, Ritchie Zhao, Ankit More, Mathew Hall, Alireza Khodamoradi, Summer Deng, Dhruv Choudhary, Marius Cornea, Eric Dellinger, Kristof Denolf, Stosic Dusan, Venmugil Elango, Maximilian Golub, Alexander Heinecke, Phil James-Roxby, Dharmesh Jani, Gaurav Kolhe, Martin Langhammer, Ada Li, Levi Melnick, Maral Mesmakhosroshahi, Andres Rodriguez, Michael Schulte, Rasoul Shafipour, Lei Shao, Michael Siu, Pradeep Dubey, Paulius Micikevicius, Maxim Naumov, Colin Verrilli, Ralph Wittig, Doug Burger, and Eric Chung. 2023. Microscaling Data Formats for Deep Learning. arXiv:2310.10537 [cs.LG] https://arxiv.org/abs/2310.10537

[17] Nikhil Rout and Blaise Tine. 2026. Ten-Four: An Open-Source Fused Dot Product Unit for Mixed-Precision GPGPU Tensor Cores. arXiv:2512.00053 [cs.AR] https://arxiv.org/abs/2512.00053

[18] Stuart Sul. 2025. 1.5x faster MoE training with custom MXFP8 kernels. Cursor Blog. https://cursor.com/blog/kernels Accessed: 2026-05-16.

[19] Blaise Tine, Krishna Praveen Yalamarthy, Fares Elsabbagh, and Kim Hyesoon. 2021. Vortex: Extending the RISC-V ISA for GPGPU and 3D-Graphics. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture* (Virtual Event, Greece) *(MICRO '21)*. Association for Computing Machinery, New York, NY, USA, 754–766. doi:10.1145/3466752.3480128

[20] Daniel Vega-Myhre, Matthias Reso, Vasiliy Kuznetsov, Driss Guessous, Simon Fan, Alireza Shamsoshoara, and Chinmay Baikar. 2026. MXFP8 Training for MoEs: 1.3x training speedup vs BF16 for Llama4 Scout on GB200 cluster using TorchAO and TorchTitan. PyTorch Blog. https://pytorch.org/blog/mxfp8-training-for-moes-1-3x-training-speedup-vs-bf16-for-llama4-scout-on-gb200-cluster-using-torchao-and-torchtitan/ Accessed: 2026-05-16.

[21] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2024. SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models. arXiv:2211.10438 [cs.CL] https://arxiv.org/abs/2211.10438

[22] Amir Zandieh, Majid Daliri, Majid Hadian, and Vahab Mirrokni. 2025. TurboQuant: Online Vector Quantization with Near-optimal Distortion Rate. arXiv:2504.19874 [cs.LG] https://arxiv.org/abs/2504.19874

[23] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. 2023. H_2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. arXiv:2306.14048 [cs.LG] https://arxiv.org/abs/2306.14048

[24] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. 2023. SGLang: Efficient Execution of Structured Language Model Programs. In *Advances in Neural Information Processing Systems (NeurIPS)*. doi:10.48550/arxiv.2312.07104