A Configurable Mixed-Precision Fused Dot Product Unit for GPGPU Tensor Computation

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

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

Abstract

There has been increasing interest in developing and accelerating mixed-precision Matrix-Multiply-Accumulate operations in GPGPUs for Deep Learning workloads. However, existing open-source RTL implementations of inner dot product units rely on discrete arithmetic units, leading to suboptimal throughput and poor resource utilization. To address these challenges, we propose a scalable mixed-precision dot product unit that integrates floating-point and integer arithmetic pipelines within a singular fused architecture, implemented as part of the open-source RISC-V based Vortex GPGPU's Tensor Core Unit extension. Our design supports low-precision multiplication in FP16/BF16/FP8/BF8/INT8/UINT4 formats and higher-precision accumulation in FP32/INT32, with an extensible framework for adding and evaluating other custom representations in the future. Experimental results demonstrate 4-cycle operation latency at 362.2 MHz clock frequency on the AMD Xilinx Alveo U55C FPGA, delivering an ideal filled pipeline throughput of 5.795 GFlops in a 4-thread per warp configuration.

CCS Concepts

• **Computer systems organization** → *Multicore architectures.*

Keywords

GPGPU, Microarchitecture, Mixed-Precision, Fused Dot Product

1 Introduction

Microbenchmarking NVIDIA Volta Architecture's [12] Warp-Matrix-Multiply-Accumulate (WMMA) instructions have demonstrated that each "Tensor Core" is essentially a grid of 16 mixed-precision Four-Element Dot Product (FEDP) units scheduled in parallel, completing a 4×4×4 matrix multiply accumulate per cycle in a 4-stage pipeline [7, 13]. Fig. 1 illustrates how we adopt a 2×2 grid of FEDP units to form a Tensor Core Unit (TCU) within a Vortex GPGPU [16] sub-core in a 4-thread per warp scheduler configuration.

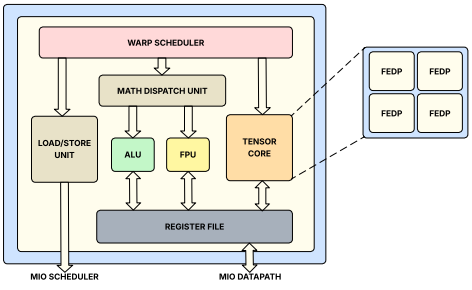

Figure 1: Sub-Core in the Vortex TCU Extended Architecture

To address the lack of high-performance fused dot product implementations in the open-source GPGPU design space:

- We propose a configurable 4-stage fused dot product architecture supporting low-precision (FP16/BF16/FP8/BF8) multiplication with FP32 accumulation; implemented as part of the Vortex GPGPU's in development TCU extension [15]
- We describe a unified pipeline methodology for integrating integer arithmetic within the floating-point datapath, incurring minimal overhead by maximizing resource reuse
- We demonstrate significant performance improvements over an equivalent Berkeley HardFloat [6] based implementation, achieving 362.2MHz maximum operational frequency and 1.448 GFLOPS single-cycle throughput

2 Mixed-Precision Floating-Point Datapath

2.1 Key Arithmetic Submodules

The floating-point dot product pipeline requires several multi-operand additions. Initially, we adopted a 2-operand adder reduction tree structure, which exhibited high latency due to accumulated carry propagation delays. We found **Carry-Save Adders (CSAs)** to be particularly suitable for our use case since they optimally reduce multiple operands without prior carry dependencies. The CSA uses recursively generated 4:2 compressors, with 3:2 compressors (Full Adders) handling remaining 3-operand cases, before final summation via **Kogge-Stone Adders (KSAs)**. KSAs outperform Carry-Look-Ahead designs by sacrificing area efficiency to achieve lower fanout at every stage due to their parallel prefix tree structures. We also implement **Wallace Tree Multipliers** whose partial products are reduced effectively using CSA structures. We decide against incorporating Radix-4 booth recoding in our design, since the bit-pair encoding overhead outweighs the benefit of halving partial products at our target 4-11 bit widths.

2.2 Low-Precision Multiplication, Maximum Exponent and Significand Alignment

The first two pipeline stages perform low-precision multiplication in parallel to finding the maximum exponent for significand alignment. Inputs are packed as FP16/BF16 pairs or FP8/BF8 quads per 32-bit register. Dedicated Wallace tree multipliers process full mantissas (including implicit bits) for each dot product element, with FP8/BF8 formats requiring additional multiplication and summation to maintain pipeline consistency. Format selection multiplexers route the results using 4-bit selectors, hence generating raw E8M25 intermediates for accumulation. Exponent addition and FP32 exponent bias conversion (as required) are performed as follows:

$$EXP_{FP32} = EXP_A + EXP_B + BIAS_{FP32} - (2 \times BIAS_{FP16}) + 1$$

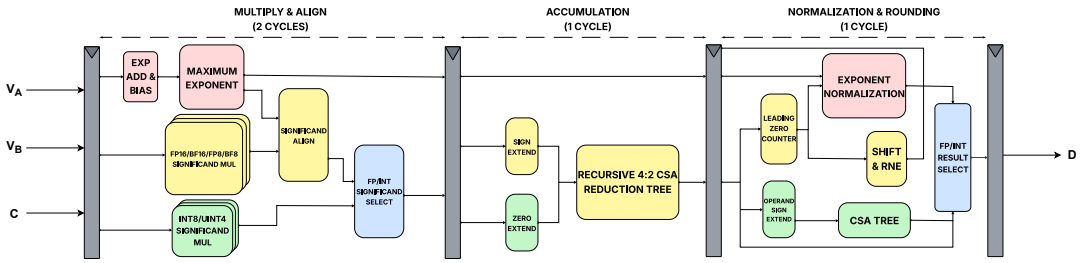

Figure 2: Mixed-Precision Fused Dot Product 4-Stage Pipeline Architecture

Maximum exponent identification builds upon a novel subtractor-based comparator architecture [14] computing all N×N pairwise exponent differences concurrently. Each comparison generates a sign bit indicating relative magnitude, thus forming a difference matrix. The maximum exponent index is determined through combinational logic that finds the element where all left comparisons are negative (1) and all right comparisons are positive (0), producing a one-hot selection vector. This approach provides O(1) depth versus traditional reduction tree comparator schemes while also computing shift amounts by simply negating the stored differences. Product significands are then aligned using these shift amounts and sign-extended before passing to the next stage.

2.3 Accumulation, Normalization and Rounding

Traditional approaches separately accumulate the addend "C" after dot product summation, requiring additional 2-operand alignment, normalization, and rounding that increases both rounding error and critical path delay. Our design integrates addend processing from the first pipeline stage, where C's exponent participates in maximum exponent finding and its significand undergoes alignment and sign-extension alongside product terms. The 25-bit aligned, sign-extended significands and addend are further sign-extended (FP) or zero-extended (INT) to $25 + \log_2 N$-bits to handle signed arithmetic correctly. The multi-operand summation utilizes the recursive 4:2 CSA with 3:2 fallback to produce the accumulation result. Finally, standard Leading Zero Counter (LZC) normalization and Round-to-nearest-even (RNE) rounding is performed.

3 Fusing the Integer Pipeline

Integer dot product operations require multiple arithmetic components already present in the FP datapath [3, 4]. Fusing both pipelines omits the need for an arbiter and scheduling two separate execution units via the same interface. We support INT8 and UINT4 formats, with INT8 inputs undergoing two's complement conversion before multiplication to maintain compact bit-width configurations. Products are sign-extended to 25 bits for accumulator compatibility. Rather than forwarding the complete 32-bit addend C to the final stage, we employ a novel splitting strategy that partitions C addition into two components. The lower 25 bits accumulate in the existing FP accumulation module, while only the upper 7 bits and product sign bits propagate through the pipeline, reducing intermediate register overhead. The final integer result's upper 7 bits are constructed in parallel with FP normalization and rounding, before concatenation with the lower 25-bit accumulation result.

4 Evaluation

We evaluate our FEDP design against equivalent HardFloat [6] and Xilinx DSP IP based discrete implementations, targeting 300MHz clock frequency on the AMD Xilinx Alveo U55C FPGA.

Table 1: Comparison of Timing Reports (FP16/BF16)

Version	Critical Path (ns)	F_{max} (MHz)	Latency (cycles)	Throughput[1] (GFLOPS)
Xilinx DSP	2.965	337.3	42	0.128
HardFloat	3.679	271.8	13	0.334
Proposed	**2.761**	**362.2**	**4**	**1.448**

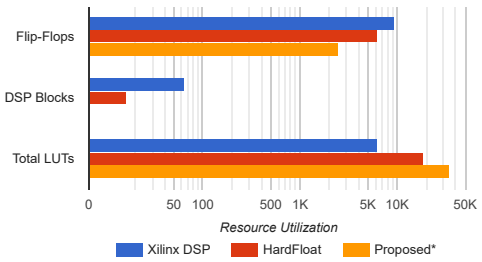

Figure 3: Comparison of Resource Utilization Reports[2]

5 Related Work

Although numerous efforts have optimized mixed-precision Fused-Multiply-Add and dot product units [2, 9, 14, 17], many academic projects involving transprecision computing, including Gemmini [5], Virgo [8], and Rocket Chip [1], still rely on Berkeley HardFloat modules [6]. Similarly, another Tensor Core implementation integrated into the Vortex framework [11] utilized FPnew [10]. These discrete approaches suffer from high latency, accumulated rounding errors, and poor area efficiency compared to our fused architecture.

6 Conclusion

This work presents a configurable high-performance fused dot product microarchitecture. Future work includes exploring shared significand multipliers, sparse-enabled FEDP and energy efficiency.

[1] Single-cycle Throughput = (16 / Latency) × F_{max})
[2] Proposed includes FP8/BF8/INT8/UINT4 support LUT count

References

[1] Krste Asanović, Rimas Avizienis, Jonathan Bachrach, Scott Beamer, David Biancolin, Christopher Celio, Henry Cook, Daniel Dabbelt, John Hauser, Adam Izraelevitz, Sagar Karandikar, Ben Keller, Donggyu Kim, John Koenig, Yunsup Lee, Eric Love, Martin Maas, Albert Magyar, Howard Mao, Miquel Moreto, Albert Ou, David A. Patterson, Brian Richards, Colin Schmidt, Stephen Twigg, Huy Vo, and Andrew Waterman. 2016. *The Rocket Chip Generator.* Technical Report UCB/EECS-2016-17. http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.html

[2] Luca Bertaccini, Gianna Paulin, Matheus Cavalcante, Tim Fischer, Stefan Mach, and Luca Benini. 2024. MiniFloats on RISC-V Cores: ISA Extensions With Mixed-Precision Short Dot Products. *IEEE Transactions on Emerging Topics in Computing* 12, 4 (2024), 1040–1055. https://doi.org/10.1109/TETC.2024.3365354

[3] Tom M. Bruintjes, Karel H. G. Walters, Sabih H. Gerez, Bert Molenkamp, and Gerard J. M. Smit. 2012. Sabrewing: A lightweight architecture for combined floating-point and integer arithmetic. *ACM Trans. Archit. Code Optim.* 8, 4, Article 41 (Jan. 2012), 22 pages. https://doi.org/10.1145/2086696.2086720

[4] Stef Cuyckens, Xiaoling Yi, Nitish Satya Murthy, Chao Fang, and Marian Verhelst. 2025. Efficient Precision-Scalable Hardware for Microscaling (MX) Processing in Robotics Learning. arXiv:2505.22404 [cs.AR] https://arxiv.org/abs/2505.22404

[5] Hasan Genc, Seah Kim, Alon Amid, Ameer Haj-Ali, Vighnesh Iyer, Pranav Prakash, Jerry Zhao, Daniel Grubb, Harrison Liew, Howard Mao, Albert Ou, Colin Schmidt, Samuel Steffl, John Wright, Ion Stoica, Jonathan Ragan-Kelley, Krste Asanovic, Borivoje Nikolic, and Yakun Sophia Shao. 2021. Gemmini: Enabling Systematic Deep-Learning Architecture Evaluation via Full-Stack Integration. In *Proceedings of the 58th Annual Design Automation Conference (DAC).*

[6] John R. Hauser. 2019. Berkeley HardFloat Floating-Point Arithmetic Package, Release 1. https://www.jhauser.us/arithmetic/HardFloat.html. Accessed: September 5, 2025.

[7] Zhe Jia, Marco Maggioni, Benjamin Staiger, and Daniele P. Scarpazza. 2018. Dissecting the NVIDIA Volta GPU Architecture via Microbenchmarking. arXiv:1804.06826 [cs.DC] https://arxiv.org/abs/1804.06826

[8] Hansung Kim, Ruohan Richard Yan, Joshua You, Tieliang Vamber Yang, and Yakun Sophia Shao. 2025. Virgo: Cluster-level Matrix Unit Integration in GPUs for Scalability and Energy Efficiency. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2* (Rotterdam, Netherlands) *(ASPLOS '25)*. Association for Computing Machinery, New York, NY, USA, 1382–1399. https://doi.org/10.1145/3676641.3716281

[9] Qiong Li, Chao Fang, and Zhongfeng Wang. 2023. PDPU: An Open-Source Posit Dot-Product Unit for Deep Learning Applications. In *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, USA, 1–5. https://doi.org/10.1109/ISCAS46773.2023.10182007

[10] Stefan Mach, Fabian Schuiki, Florian Zaruba, and Luca Benini. 2020. Fpnew: An open-source multiformat floating-point unit architecture for energy-proportional transprecision computing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 29, 4 (2020), 774–787.

[11] Abubakr Nada, Giuseppe Maria Sarda, and Erwan Lenormand. 2025. Cooperative Warp Execution in Tensor Core for RISC-V GPGPU. In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 1422–1436. https://doi.org/10.1109/HPCA61900.2025.00107

[12] NVIDIA Corporation. 2017. *NVIDIA Tesla V100 GPU Architecture.* Technical Report. https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf

[13] Md Aamir Raihan, Negar Goli, and Tor M. Aamodt. 2019. Modeling Deep Learning Accelerator Enabled GPUs. In *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 79–92. https://doi.org/10.1109/ISPASS.2019.00016

[14] Jongwook Sohn and Earl E. Swartzlander. 2016. A Fused Floating-Point Four-Term Dot Product Unit. *IEEE Transactions on Circuits and Systems I: Regular Papers* 63, 3 (2016), 370–378. https://doi.org/10.1109/TCSI.2016.2525042

[15] Blaise Tine and Nikhil Rout. 2025. Vortex GPGPU Tensor Core Unit Extension FEDP DRL RTL Backend. https://github.com/vortexgpgpu/vortex/tree/bug_fixes/hw/rtl/tcu/drl.

[16] 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. https://doi.org/10.1145/3466752.3480128

[17] Hao Zhang, Dongdong Chen, and Seok-Bum Ko. 2019. Efficient Multiple-Precision Floating-Point Fused Multiply-Add with Mixed-Precision Support. *IEEE Trans. Comput.* 68, 7 (2019), 1035–1048. https://doi.org/10.1109/TC.2019.2895031