ECE 552: Project Proposal
Micro-architectural Support for Packed Sparse Matrix Multiplication on Hybrid Systolic Arrays

David Raibaut

***Abstract*—This project pulls inspiration from two novel architectural methods for increasing NPU efficiency: Hybrid Systolic Arrays (HSA) [1] and Sparse-TPU [2]. HSA provides a way of reducing chip area by combining matrix and vector multiplies into a single systolic array. Sparse-TPU provides hardware support for sparse-packed matrices thus increasing performance and reducing area. I will develop unit tests and benchmarks for the components to ensure functional correctness, compare HSA, sparse-tpu and their novel combination with more traditional methods and analyze the performance and energy efficiency impact.**

I. Introduction and Problem Statement

User demand for Large Language Model (LLM) inference has grown rapidly throughout the past years [3]. On-edge inference has emerged as a powerful alternative to cloud-based solutions, thanks to its lower cost, decreased latency and heightened security with comparable performance for daily tasks. However, the limited memory capacity, energy constraints and sequential query model pose significant challenges for successfully deploying LLMs on edge, where user-facing applications require minimal latency (dictated by the decode stage [1]) and a reduced energy footprint (dictated by external memory accesses (EMA)) [1].

A proposed solution is the use of sparse-matrix LLMs [2], such as SparseLLM [4], SparseGPT [5] and KMZ [6]. These models compress larger trained models to reduce their number of parameters and hence reduce the memory requirements for the model and the number of operations to produce their computation by making their composing matrices sparser.

Other proposals to enhance edge inference efficiency include the use of Hybrid Systolic Arrays (HSA) [1]. These units function as an intermediate point between Matrix Processing Units (MPU, used during pre-fill stage) and Vector Processing Units (VPU, used during decode stage), thus eliminating the need for both. This has been shown to effectively reduce the area and thus energy consumption and maintain comparable latency.

In this project I plan to explore a novel HSA, by including hardware support for packed sparse matrices, essentially exploring the performance-efficiency tradeoff exposited by a combination of two previous methods. I will also implement sparse-matrix support on clock-gated MPUs and VPUs to show the advantage provided by the HSA architecture, and conversely non-sparse matrix supported MPUs, VPUs and HSAs to show the advantage provided by the HSA structure independently.

II. Related Work

There have been multiple proposals for on-edge LLM inference without the use of dedicated hardware accelerators, focusing mostly on CPU optimisation [7], [8], [9], efficient model compression and quantization [10], [11] and even custom frameworks for edge-specific model design [12], [13].

Even then, due to the tight energy and performance efficiency constraints on edge devices, particularly mobile devices such as phones or wearables, most of these techniques incur large tradeoffs, such as requiring models to be very small [7] ($\sim$1B parameters), or finding significant performance degradation at sub-4-bit quantization [10].

Thus, given the rise in user-demand [3] for low-latency high-performing LLMs, the most likely path forward seems to be dedicated hardware accelerators [14] optimised for this specific workload.

In this project I will mostly refer to the HSA [1] and Sparse-TPU [2] as they are the two most relevant sources due to their closeness with the proposed implementation. Other notable areas of research include heterogeneous systems [15], [16], as a way of attempting to reduce the memory bandwidth bottleneck at the cost of architectural flexibility, compressed KV-cache implementations [17] at the cost of added latency and KV cache reuse optimisations [18] at the cost of decreased accuracy.

III. Research Questions

- How can existing HSA hardware be adapted to support packed sparse matrix multiplication?
- How much performance and energy gains does packed sparse matrices provide in practice?
- How much performance and energy gains does HSAs provide in practice?
- How much performance and energy gains does the novel packed sparse matrix HSAs provide in practice?

IV. Proposed Solutions

The proposed solution is that of a HSA structure with added hardware support for sparse-packed matrices. The use of the HSA structure allows for a lower chip area (since separate MPUs and VPUs are not required for the pre-fill and decode stages respectively) thus leading to lower leakage power and overall energy use. Hardware-support for sparse-packed matrices increase performance by significantly reducing the number of operations required to be computed (i.e. the number of MAC

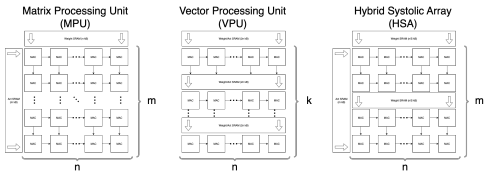

Fig. 1. Comparison of the three structural systolic array architectures

units), at the cost of some accuracy (due to the packing), and thus reducing overall SA latency. Moreover, by clock-gating the individual MAC units, the dynamic power of the whole chip can be reduced thus decreasing energy use too.

In this project I set out to explore whether this novel idea of a HSA with support for sparse-packed matrices (referred to throughout as SHSA) truly leads to the performance and energy improvements over single HSA or single sparse-packing, how much each of these individual improvements contributes and what the accuracy tradeoff is.

Additionally, time-permitting, the energy-performance tradeoff of utilising the equivalent MPU+VPU area for just the (S)HSA structure (by increasing the number of MAC units) will also be evaluated.

V. Evaluation

A. Methodology

The hardware units will be implemented in SystemVerilog [19], and Xilinx Vivado will be used to synthesize them. They will be tested on the AMD Artix A7 FPGA with 63400 LUTs, 126800 FFs, and 4860Kb of SRAM. The following hardware components will be implemented:

- SHSA: Sparse HSA
- HSA: Regular HSA
- SMPU: Sparse MPU
- MPU: Regular MPU
- SVPU: Sparse VPU
- VPU: Regular VPU
- MCU: Multiply aCcumulate Unit
- NPU package [1]

Note: The sparse-packing software algorithm will be taken directly from [2].

Given the memory and logic slice constraints on the Artix A7, I estimate I will be able to fit a 32 by 32 SA unit (see figure 1), along with the accompanying driving logic, however this is subject to change. Assuming this, the activation and weight SRAMs will be 32kB and 16kB each in total.

B. Functional Validation

(a) *Unit testing*: Test benches in SystemVerilog will be provided for each of the aforementioned units

(b) *Software implementation*: A C++ implementation of the units will be realised so benchmarks can be tested and outputs compared with the hardware.

[1]Includes: Accumulate, Activate, Normalise units, Unified Buffer and Systolic datapaths, Instruction queue and control paths, Host-Device interface, External memory-device interface

C. Physical Characteristics

Vivado software provides tools for finding power: both compiler estimates and XADC System Monitor readings will be provided for power.

Vivado also provides tools for assessing the maximal frequency, namely the timing report after synthesis, which includes the 'Worst Negative Slack' (WNS) metric. In the case of a negative WNS (indicating timing violations), an analysis of the critical paths and iterative redesign/testing (including frequency increase) will be performed, until all designs meet the timing requirements. Ideally, all structures should operate at the same frequency, since MAC units and supporting NPU package will be identical as control variables.

Vivado synthesis also provides data for logic cell and BRAM utilisation which will be used as an estimate for chip area.

D. Performance Impact

Performance impact will be compared across multiple metrics:

1) Raw performance (TOPS)
2) Energy efficiency (pJ/Token)
3) Power efficiency (TOPS/W)
4) *Area (mm2)
5) *Accuracy (for sparse-packing, % diff vs non-packed)

All the items marked with an asterisk are considered to be implemented 'time-permitting'. Raw performance will be approximated by estimating the operations per clock cycle, and energy efficiency will be calculated from the power efficiency (which can be estimated using Xilinx's XPower tool).

VI. Milestones

The project is broken down into 5 milestones, spanning 6 weeks a more aggressive deadline so that possible unforeseen circumstances can be accommodated.

1) **Milestone 1: C++ Software Implementation (Week 1):** The initial software implementation in C++ will be realised early on, to facilitate the hardware development later on and ensure functional validation from the get-go thus reducing debugging time.
2) **Milestone 2: MCU, SHSA, HSA, SMPU, MPU, SVPU, VPU Implementation in Verilog (Weeks 2-3):** The smaller modules will be implemented first in Verilog with unit testbenches to accompany them and periodic synthesis on the FPGA to guarantee it works there too.
3) **Milestone 3: NPU Package (Week 4):** Supporting logic for above modules so they can be tested on an FPGA acting as a NPU to the host.
4) **Milestone 4: Functional Validation and Performance Testing (Week 5):** Conduct functional validation and run performance benchmarks (TOPS calculation, *RetNet, *Llama2)
5) **Milestone 5: Final Evaluation and Reporting (Week 6):** Analyze performance and write report.

VII. Division of Labour

All work will be done by me (David Raibaut).

VIII. References

References

[1] C.-T. Chen, H. Mun, J. Meng, M. S. Abdelfattah, and J. sun Seo, "Hybrid systolic array accelerator with optimized dataflow for edge large language model inference," 2025. [Online]. Available: https://arxiv.org/abs/2507.09010

[2] A. A. S. F. D.-H. P. A. R. H. Y. Y. C. R. D. T. M. Xin He, Subhankar Pal, "Sparse-tpu: Adapting systolic arrays for sparse matrices," 2020. [Online]. Available: https://tnm.engin.umich.edu/wp-content/uploads/sites/353/2020/08/2020.6.sparse-tpu_ics2020.pdfhttps://arxiv.org/abs/2507.09010

[3] A. Fradkin, "Demand for llms: Descriptive evidence on substitution, market expansion, and multihoming," 2025. [Online]. Available: https://arxiv.org/abs/2504.15440

[4] G. Bai, Y. Li, C. Ling, K. Kim, and L. Zhao, "Sparsellm: Towards global pruning for pre-trained language models," 2024. [Online]. Available: https://arxiv.org/abs/2402.17946

[5] E. Frantar and D. Alistarh, "Sparsegpt: Massive language models can be accurately pruned in one-shot," 2023. [Online]. Available: https://arxiv.org/abs/2301.00774

[6] H. Kung, B. McDanel, and S. Q. Zhang, "Packing sparse convolutional neural networks for efficient systolic array implementations: Column combining under joint optimization," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 821–834. [Online]. Available: https://doi.org/10.1145/3297858.3304028

[7] H. Zhang and J. Huang, "Challenging gpu dominance: When cpus outperform for on-device llm inference," 2025. [Online]. Available: https://arxiv.org/abs/2505.06461

[8] C. Zhang, X. Zhu, L. Chen, T. Yang, E. Pan, G. Yu, Y. Zhao, X. Wu, B. Li, W. Mao, and G. Han, "Enhancing llm inference performance on arm cpus through software and hardware co-optimization strategies," *Integrated Circuits and Systems*, vol. 2, no. 2, pp. 49–57, 2025.

[9] H. Shen, H. Chang, B. Dong, Y. Luo, and H. Meng, "Efficient llm inference on cpus," 2023. [Online]. Available: https://arxiv.org/abs/2311.00502

[10] S. Cao, L. Ma, and T. Cao, "Advances to low-bit quantization enable llms on edge devices," 2025. [Online]. Available: https://arxiv.org/abs/2311.00502

[11] K. Freund, "How to run large ai models on an edge device," 2023. [Online]. Available: https://cambrian-ai.com/how-to-run-large-ai-models-on-an-edge-device/

[12] ggml org, "llama.cpp," 2025. [Online]. Available: https://github.com/ggml-org/llama.cpp

[13] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee, W.-T. Chang, W. Hua, M. Georg, and M. Grundmann, "Mediapipe: A framework for building perception pipelines," 2019. [Online]. Available: https://arxiv.org/abs/1906.08172

[14] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P. luc Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon, "In-datacenter performance analysis of a tensor processing unit," 2017. [Online]. Available: https://arxiv.org/abs/1704.04760

[15] L. Chen, D. Feng, E. Feng, Y. Wang, R. Zhao, Y. Xia, P. Xu, and H. Chen, "Characterizing mobile soc for accelerating heterogeneous llm inference," 2025. [Online]. Available: https://arxiv.org/abs/2501.14794

[16] M. Seo, X. T. Nguyen, S. J. Hwang, Y. Kwon, G. Kim, C. Park, I. Kim, J. Park, J. Kim, W. Shin, J. Won, H. Choi, K. Kim, D. Kwon, C. Jeong, S. Lee, Y. Choi, W. Byun, S. Baek, H.-J. Lee, and J. Kim, "Ianus: Integrated accelerator based on npu-pim unified memory system," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ser. ASPLOS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 545–560. [Online]. Available: https://doi.org/10.1145/3620666.3651324

[17] A. Liu, J. Liu, Z. Pan, Y. He, G. Haffari, and B. Zhuang, "Minicache: Kv cache compression in depth dimension for large language models," 2024. [Online]. Available: https://arxiv.org/abs/2405.14366

[18] J. Thomson, A. Shah, and L. Tewari, "Introducing new kv cache reuse optimizations in nvidia tensorrt-llm," 2025. [Online]. Available: https://developer.nvidia.com/blog/introducing-new-kv-cache-reuse-optimizations-in-nvidia-tensorrt-llm/

[19] "Ieee standard for systemverilog–unified hardware design, specification, and verification language," *IEEE Std 1800-2023 (Revision of IEEE Std 1800-2017)*, pp. 1–1354, 2024.