Late Breaking Results: Approximated LUT-based Neural Networks for FPGA Accelerated Inference

Xuqi Zhu, Jiacheng Zhu, Huaizhi Zhang, Tamim M. Al-Hasan, Klaus D. McDonald-Maier and Xiaojun Zhai
School of Computer Science and Electronic Engineering, University of Essex
Wivenhoe Park, Colchester CO4 3SQ, United Kingdom
E-mail: {xz18173, jz23222, hz24245, tamim.almulla, kdm, xzhai}@essex.ac.uk

***Abstract*—This work presents LUT-MU, an approximated LUT-based Matrix Multiplication (MM) architecture designed for FPGA-based Neural Network (NN) inference across. The proposed architecture maximises the utilisation of on-chip memory bandwidth through dedicated memory distribution and pipeline design, addressing performance limitations inherent to LUT-based MM. Experimental evaluation demonstrates that LUT-MU achieves a four-fold improvement in NN inference throughput whilst reducing hardware resource consumption by 80% with only a 5% decline in accuracy. These results validate that our optimisation approach successfully resolves the performance constraints caused by the limited arithmetic intensity and memory bandwidth, enabling the LUT-MU to serve as a foundation for efficient NN acceleration systems.**

***Index Terms*—Approximate computing, NN acceleration, FPGA**

I. Introduction

Neural Networks (NNs) have witnessed remarkable evolution in recent years, with architectures becoming increasingly complex and parameter-intensive. While this evolution has driven significant performance improvements, it has created new challenges for accelerator designs on resource-constrained devices. A considerable bottleneck arises from Matrix Multiplication (MM) operations, which are fundamental building blocks of NN inference and impose considerable computation overhead [1]. To address this, researchers have explored alternative algorithms to reduce the complexity of MM. For instance, Blalock et al. [2] introduced MADDNESS, a LUT-based MM algorithm that utilises the Product Quantisation (PQ) method [3]. This approach eliminates the need for online multiplications between a random input vector and a known weight matrix by approximating the input vector using prototypes clustered offline from the training dataset. Subsequently, Tang et al. [4] highlighted how the non-differentiable nature of PQ functions disrupts the backpropagation of loss signals, causing a notable drop in accuracy. To address this limitation, Jannis et al. [5] introduced a soft encoding method to substitute the non-derivable function of the PQ during the training phase, enabling NNs employing MADDNESS to be retrained effectively via backpropagation, thereby restoring inference accuracy.

The researches [5]–[8] have demonstrated that MADDNESS achieves comparable accuracy in neural network inference while significantly reducing computational overhead compared to General MM (GeMM). However, it requires massive memory traffic to process equivalent workloads compared to GeMM, resulting in lower Arithmetic Intensity (AI), defined as operations per byte of memory traffic. Furthermore, data dependencies and

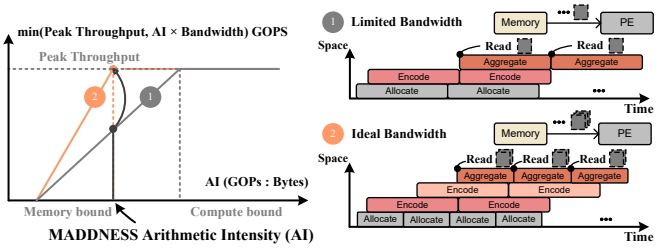

Fig. 1. The challenge of MADDNESS-based NN accelerator.

the incoherent memory access pattern (discussed in Section II) in MADDNESS reduce the effective bandwidth. Lower AI and reduced effective bandwidth restrict memory-bound throughput when implementing MADDNESS on general processor architectures, as illustrated in Fig. 1.

To address the challenge of suboptimal throughput in the MADDNESS-based NN inference, this paper presents LUT-MU, a novel LUT-based MM unit. This work's main contributions are: 1) Proposing tailored memory allocation and pipeline optimisations to overcome bandwidth reduction from data dependencies and incoherent memory access in MADDNESS, boosting LUT-MU throughput. 2) A scalable LUT-MU architecture that mitigates higher resource demands from these pipeline strategies, providing a flexible trade-off between accuracy and resource utilisation.

II. Bottlenecks & Trade-off Design

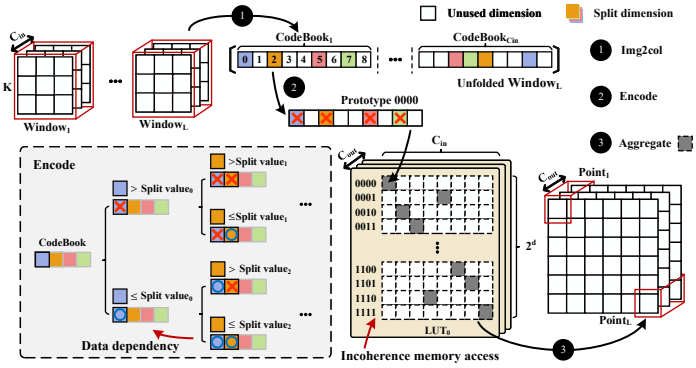

Fig. 2. The bottleneck of achieving ideal bandwidth.

Although MADDNESS brought an innovative way to implement NN inference with matmul-free computation, its considerable memory traffic demands exceed the effective bandwidth, posing a challenge for designing efficient hardware accelerators. As illustrated in Fig. 2, there are three distinct phases to compute convolution via MADDNESS: 1) use im2col to

978-3-9826741-0-0/DATE25/© 2025 EDAA

map the input matrix into L unfolded windows, each containing C_{in} codebooks; 2) sequentially load contents on split dimensions into decision tree-based encode function to obtain the prototypes ID; and 3) fetch and aggregate the corresponding partial results (i.e., grey cells) scattered randomly across the LUT to obtain the approximate results. Data dependencies in the encode function and the incoherent memory access pattern in the aggregate phase result in a stretched Initiation Interval (II) in LUT-MU, thereby restricting the read frequency and reducing the effective bandwidth, as shown in Fig. 1.

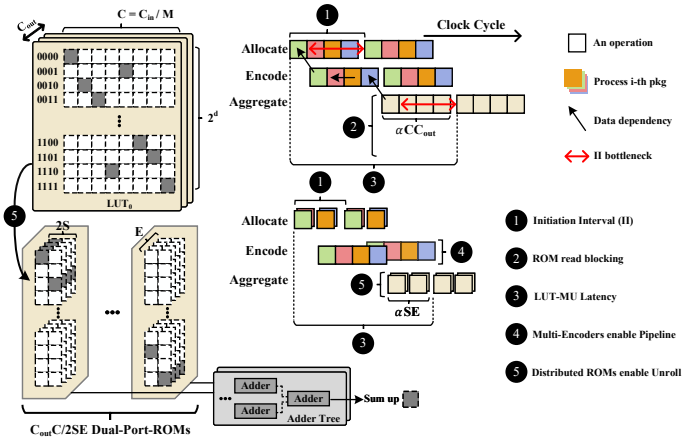

Fig. 3. Memory allocation and pipeline optimisation.

However, we observe that the encoding process is independent for each codebook, allowing d packages to be assembled and loaded in one operation, thus letting the allocator and encoder process successive unfolded windows in a pipelined manner. For each input, only one prototype is chosen per codebook (i.e., each LUT column contains a grey cell). Consequently, the LUT can be tiled along C and C_{out} and distributed across $C_{\text{out}} \times C/(2 \times S \times E)$ ROMs, as shown in Fig. 3, alleviating read conflicts from random access to a single memory unit. Although the pipeline design improves bandwidth, it increases hardware resource usage. To address this, we group every M input feature-map channel as a codebook and include additional prototypes per codebook to mitigate resolution loss (i.e., the number of split dimensions per unfolded window $Cd/(C_{\text{in}}K^2)$). This strategy reduces the size of the encoder and adder tree in LUT-MU while minimising accuracy loss.

III. Evaluation

To validate the effectiveness of the proposed memory allocation and pipeline design in improving throughput and mitigating resource demands, we investigate the throughput and resource utilisation of the LUT-MU under four LUT configurations ($C \times 2^d$) and three partition factors (E, S). The performance data illustrated at the bottom of Fig. 4 demonstrates the impact of distributed ROM configurations on system throughput. As the number of distributed ROMs increases (i.e., $E = 4, 2, 1$), the throughput significantly improves, rising from 1.5×10^4 GOPS to 6×10^4 GOPS. This enhancement validates the effectiveness of our memory allocation strategy and pipeline architecture in optimising bandwidth utilisation for improved throughput performance. From another perspective, the occupancy of FF and LUT decreased by approximately 80% when

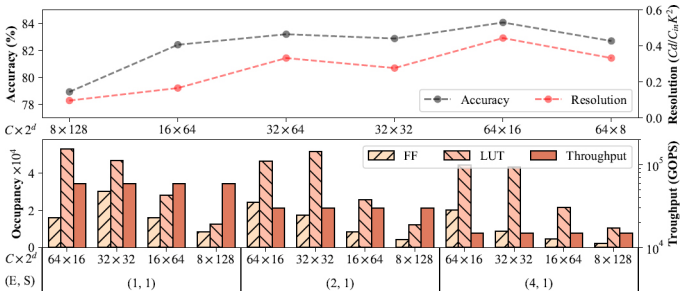

Fig. 4. Top: Classification accuracy of 4-bit quantised LUT-MU-based ResNet-9 on CIFAR-10 across varying resolutions ($Cd/C_{in}K^2$); Bottom: LUT-MU FPGA resource utilisation (LUT & FF occupancy) and throughput for different LUT shape ($C \times 2^d$) and partition (E, S) configuration.

the LUT shape changed from 64×16 to 8×128, while the inference accuracy dropped by 5% due to resolution loss. This demonstrates that configuring the LUT shape efficiently mitigates increased resource demands, providing a flexible trade-off between accuracy and resource utilisation.

IV. Conclusion

We presented LUT-MU, a versatile Look-Up Table-based MM framework that advances the state of NN acceleration. Our targeted optimisation strategies in memory allocation and pipeline design effectively address bandwidth constraints from data dependencies and non-sequential memory access, overcoming throughput limitations of prior LUT-based solutions. However, the proposed optimisation strategies introduce additional resource overhead to achieve the ideal bandwidth. Although reshaping the LUT can alleviate resource demands, it inevitably results in the accuracy degradation of MADDNESS-based NN inference. This motivates further exploration of effective techniques to improve the throughput and reduce memory demands without losing additional accuracy.

Acknowledgment

This work is supported by the UK Engineering and Physical Sciences Research Council through grants, EP/X015955/1, EP/X019160/1 and EP/V000462/1, EP/V034111/1. For the purpose of open access, the author has applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising.

References

[1] X. Zhu *et al.*, "Bayesian optimization for efficient heterogeneous mpsoc based dnn accelerator runtime tuning," in *2023 FPL*, 2023, pp. 355–356.
[2] D. Blalock and J. Guttag, "Multiplying matrices without multiplying," in *ICML*. PMLR, 2021, pp. 992–1004.
[3] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE TPAMI*, vol. 33, no. 1, pp. 117–128, Jan 2011.
[4] X. Tang, *et al.*, "Lut-nn: Empower efficient neural network inference with centroid learning and table lookup," in *MobiCom*, 2023, pp. 1–15.
[5] J. Schönleber, *et al.*, "Stella nera: Achieving 161 top/s/w with multiplier-free dnn acceleration based on approximate matrix multiplication," *arXiv preprint arXiv:2311.10207*, 2023.
[6] S.-Z. Lin *et al.*, "Lutin: Efficient neural network inference with table lookup," in *Proceedings of the 29th ACM/IEEE ISLPED*, 2024, pp. 1–6.
[7] N. Gupta *et al.*, "Tabconv: Low-computation cnn inference via table lookups," in *CF*. ACM, 2024, pp. 180–188.
[8] H. Fuketa *et al.*, "Multiplication-free lookup-based cnn accelerator using residual vector quantization and its fpga implementation," *IEEE Access*, 2024.