Rectangular Packing for High-utilization Analog In-memory Compute Mappings

Lawrence Roman A. Quizon

April 2025

Contents

Abstract

The use of Artificial intelligence (AI) offers great benefits in terms of scalability, application space and viability for extreme edge devices. However, extreme edge devices are constrained to work with extremely low amounts of memory and energy. For this purpose, developments ranging from the emergence of more efficient AI algorithms all the way to the design and fabrication of more efficient application-specific ICs have been emerging in the recent years.

Analog in-memory computing shows itself as the most area and energy-efficient solution for AI inference in extreme edge devices. However, typical analog in-memory computing (AIMC) architectures require large amounts of energy for writing the weights to the memory cells. Reusing the written weights as much as possible is imperative to achieve high energy efficiency in AIMC.

Existing works on AIMC acceleration tend to ignore model utilization entirely or target a specific model in an ad-hoc manner. Since AIMC arrays may need to be designed for generality, there is a need to design a utilization mapping and tool flow that can optimize for a set of target models. We take into account a set of models and derive a minimum highest-utilization viable mapping at a specific latency constraint.

Matrices are typically mapped one-to-one to a memory array, causing very little array utilization when layers with small footprint are mapped. We propose a new AIMC architecture that can achieve high utilization by allowing multiple layers to be mapped to the same AIMC array, a restriction that many DNN compiler works self-impose. We also propose a new mapping algorithm that can achieve high utilization by packing multiple models into the same AIMC array. We show that our proposed architecture and mapping algorithm can achieve high utilization while maintaining low latency and energy consumption.

1 Introduction

The use of deep neural networks (DNNs) in extreme edge devices such as wireless sensor nodes (WSNs) will greatly benefit the scalability and application space of such nodes. AI can be applied to solve problems with clustering, data routing, and most importantly it can be used to reduce the volume of data transmission via data compression or making conclusions from data within the node itself [1].

However, since devices in the extreme edge are constrained to work with extremely low amounts of memory and energy, even the simplest AI models are difficult to execute with typical sequential processors. WSNs have memories in the order of kB and clock speeds in the order of kHz to MHz due to energy constraints, rendering them unable to run state-of-the art AI applications.

As of the recent decade, two main solutions to the growing compute needs of AI have emerged: the development of more efficient AI algorithms and the design and fabrication of more efficient application-specific integrated circuits (ASICs) []. The former has led to the emergence of new AI algorithms that are more efficient in terms of memory and energy usage, such as neural network quantization [2], neural architecture search (NAS) [3], and depthwise convolutions [4]. The latter has led to the emergence of hardware accelerators that take advantage of data usage patterns in DNN computational graphs to reduce memory accesses. Memory accesses are the most energy-expensive and time-consuming part of AI processing [5].

One of the most promising hardware solutions to edge AI is in-memory computing (IMC) [6]. IMC allows very high energy savings compared to other approaches by eliminating half of the memory accesses required by performing computations using the memory array. Analog IMC (AIMC) with memristors has proven to be fast and efficient at multiply-and-accumulate operations (MACs) which are by far the most common operation used by AI software. Additionally, since memristors are nonvolatile memory devices, they are particularly robust to energy interruptions from ultra-low power situations in WSN.

DNN compilers are responsible for taking a DNN model and mapping it to the hardware [7]. By analyzing the computational graph of the DNN, the DNN compiler can essentially create an equivalent computational graph- a "mapping"- that is compatible with and optimal for the hardware to map the DNN to the hardware accelerator. ASIC DNN accelerators emerged faster than the accompanying DNN compilers in the past decade (2015-2025). While the gap has mostly closed for compiling for digital accelerators [8], the same cannot be said for analog accelerators [9]. The majority of papers on analog accelerators are still tested with handcrafted mappings [10–12] or suboptimal single-layer mappings. We find that there are three main challenges for DNN compilation and mapping for AIMC accelerators.

Firstly, an AIMC DNN compiler must amortize the write energy cost over as many MAC operations as possible in the AIMC architecture, moreso than in regular digital accelerators. Analog IMC banks typically report much higher write energy costs than read energy costs [6] than digital accelerators. DNN compilers for digital accelerators typically treat feature maps and weights almost equally, prioritizing whichever causes the highest amount of data reuse [13]. In contrast, there exists a large discrepancy in write energy to the AIMC memory array (typically used for weights) vs the cost of moving data to and from peripheral buffers (typically used for activations) due to the use of special memory cells and other peripheral elements like the ADC and adder trees. Not only that, but AIMC also incentivizes doubling down on weight reuse due to its essentially zero read energy cost for weights once they are in the AIMC memory array.

Secondly, not all the multiply-accumulate layers of modern DNNs benefit from AIMC. AIMC arrays only support general matrix multiplies (GEMMs). Generally, the solution is typically to transform other operations into matrix operations which work for most DNN layers. However, modern DNNs also employ layers that use special types of convolutions like depthwise convolutions [4] that end up transforming into to impractically large sparse matrices or thousands of extremely small (1x9) matrices. Both of these cases are inefficient for AIMC architectures. The former case leads to a large number of writes and the latter case leads to an impossibly large latency due to the matrices' inability to share their inputs. Certain dataflow architectures for digital accelerators like the row-stationary dataflow support depthwise convolutions natively. Due to that, heterogenous accelerators that contain both digital and AIMC processing elements have been designed [11, 14].

Thirdly, while efficient mapping of single DNN layers is mostly solved [8] and interlayer mapping for digital accelerators has made significant progress [15], optimizing mappings of multiple DNN layers on AIMC architectures is still an open problem. In DNN compilation, some layers map into matrices that are larger than the AIMC memory array while others are much smaller than the memory array size. All existing DNN compiler works with interlayer scope avoid mapping the more than a single layer into an AIMC memory array or digital processing element (PE) array at the same time to simplify the work. In the rest of this work, we will refer to this as "one-to-one restriction".

In this work, we introduce QRAcc, a heterogenous DNN accelerator architecture with a row-stationary digital accelerator and charge-redistribution SRAM AIMC bank that can support the workloads of modern DNNs designed for the edge at high utilization. We also introduce MARP, a DNN compiler codesigned with QRAcc that optimizes DNN mappings in heterogenous AIMC architectures by solving rectangular bin packing algorithms to pack DNN layers as tightly as possible into the AIMC accelerator. We show that MARP can achieve optimal memory utilization for a given heterogenous accelerator architecture for the modern DNN workloads in MLPerfTiny []. MARP is the first DNN compiler to optimize for AIMC architectures with interlayer mappings without being restricted to single-layer mappings. We verify the results of MARP with implementations and measurements of energy

efficiency in QRAcc implemented in 22nm FDSOI technology. We find that different settings for MARP shows design tradeoffs that optimize for different aspects such as utilization, latency, and energy efficiency. Lastly, we provide optimal QRAcc configurations for the MLPerfTiny workloads.

2 Background

3 Methodology

Bibliography

[1] M. A. Alsheikh, S. Lin, D. Niyato, and H.-P. Tan, "Machine learning in wireless sensor networks: Algorithms, strategies, and applications," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1996–2018, 2014.

[2] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2704–2713.

[3] J. Lin, W.-M. Chen, Y. Lin, C. Gan, S. Han, *et al.*, "Mcunet: Tiny deep learning on iot devices," *Advances in neural information processing systems*, vol. 33, pp. 11 711–11 722, 2020.

[4] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[5] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC)*, IEEE, 2014, pp. 10–14.

[6] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory devices and applications for in-memory computing," *Nature nanotechnology*, vol. 15, no. 7, pp. 529–544, 2020.

[7] M. Li *et al.*, "The deep learning compiler: A comprehensive survey," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 3, pp. 708–727, 2020.

[8] L. Mei, P. Houshmand, V. Jain, S. Giraldo, and M. Verhelst, "Zigzag: Enlarging joint architecture-mapping design space exploration for dnn accelerators," *IEEE Transactions on Computers*, vol. 70, no. 8, pp. 1160–1174, 2021.

[9] P.-Y. Chen, X. Peng, and S. Yu, "Neurosim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 12, pp. 3067–3080, 2018.

[10] W. Wan *et al.*, "Neurram: Rram compute-in-memory chip for efficient versatile and accurate ai inference,"

[11] A. Garofalo *et al.*, "A heterogeneous in-memory computing cluster for flexible end-to-end inference of real-world deep neural networks," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 12, no. 2, pp. 422–435, 2022.

[12] C. Zhou *et al.*, "Ml-hw co-design of noise-robust tinyml models and always-on analog compute-in-memory edge accelerator," *IEEE Micro*, vol. 42, no. 6, pp. 76–87, 2022.

[13] L. Mei and M. Verhelst, "Design space exploration of deep learning accelerators," 2023.

[14] P. Houshmand *et al.*, "Diana: An end-to-end hybrid digital and analog neural network soc for the edge," *IEEE Journal of Solid-State Circuits*, vol. 58, no. 1, pp. 203–215, 2022.

[15] L. Mei, K. Goetschalckx, A. Symons, and M. Verhelst, "Defines: Enabling fast exploration of the depth-first scheduling space for dnn accelerators through analytical modeling," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, IEEE, 2023, pp. 570–583.