Dynamic, Policy-Based Memory Management for Unified Scratchpads in Deep Learning Accelerators

Tharindu Dasun Pathirage

University Of Westminster

tharindu.20210830@iit.ac.lk

August 17, 2025

Abstract

Specialized hardware accelerators are the engine behind the widespread adoption of Artificial Intelligence (AI), but their efficiency is increasingly challenged by the growing complexity of Deep Learning (DL) models. A primary performance bottleneck is the management of limited on-chip memory. Many contemporary accelerators rely on a rigid memory architecture with static partitions for different data types, a design that fails to accommodate the diverse and changing memory needs of a model's individual layers. This inflexibility results in wasted memory and excessive, energy-intensive data traffic to off-chip DRAM. This proposal details a research initiative to create and validate a novel, software-driven management framework for a unified on-chip scratchpad. Our approach treats all on-chip memory, including space typically reserved for prefetching, as a single, adaptable resource pool. We will formulate a series of distinct memory management policies, each engineered to capitalize on a specific data reuse pattern. A lightweight analytical engine will be developed to select the optimal policy for each layer of a network, dynamically configuring the memory to achieve a specific goal, such as minimizing off-chip transfers or reducing execution latency. The proposed framework's efficacy will be quantified through a simulation-based comparative analysis against a conventional systolic array accelerator with a fixed memory layout. This work is expected to deliver a more efficient and adaptable memory management solution that boosts the performance and energy efficiency of next-generation DL hardware.

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence

CNN Convolutional Neural Network

DL Deep Learning

DSE Design Space Exploration

GLB Global Buffer

ifmap Input Feature Map

ofmap Output Feature Map

MAC Multiply and Accumulate

NPU Neural Processing Unit

PE Processing Element

TPU Tensor Processing Unit

1 Introduction

1.1 Motivation and Rationale

The computational demands of modern Deep Learning have necessitated a shift towards specialized hardware accelerators. The performance of these accelerators is critically dependent on their memory hierarchy, specifically the efficiency of their on-chip scratchpad memories. These small, fast memories are essential for exploiting the considerable data reuse inherent in DL workloads, thereby reducing the need for communication with off-chip DRAM. Accessing off-chip memory is a major bottleneck; it is significantly slower and can consume orders of magnitude more energy than on-chip computation, making its minimization a primary goal in accelerator design.

A complication arises from the nature of DL models themselves. Within a single network, the memory requirements for inputs (ifmaps), weights (filters), and outputs (ofmaps) are not uniform; they fluctuate significantly from one layer to the next. Early convolutional layers might process large ifmaps with few filters, while later layers may do the opposite. This workload heterogeneity creates a fundamental challenge for hardware with static resource allocation.

ifmap buffer	Input Buffer Memory
ofmap buffer	Onput Buffer Memory
filter buffer	Filter Memory for Weight

Figure 1: Memories.

1.2 Problem Statement

This research addresses the architectural mismatch between static on-chip memory partitioning in DL accelerators and the dynamic resource requirements of modern neural networks. The prevailing design philosophy of assigning fixed, separate buffers for inputs, weights, and outputs introduces significant inefficiencies. This rigidity leads to a critical problem with two facets:

1. **Suboptimal Resource Utilization:** A fixed-partition scheme inevitably leads to scenarios where one buffer is overwhelmed, forcing frequent off-chip transfers, while another buffer remains largely unused. This idle memory represents a wasted opportunity to cache more data and improve performance.

2. **Inflexible Handling of Prefetching Space:** To hide the latency of memory accesses, accelerators often employ double buffering, which reserves a static portion of memory exclusively for prefetching data. This space cannot be dynamically reallocated to enhance data reuse, even in layers where memory capacity, not latency, is the primary constraint.

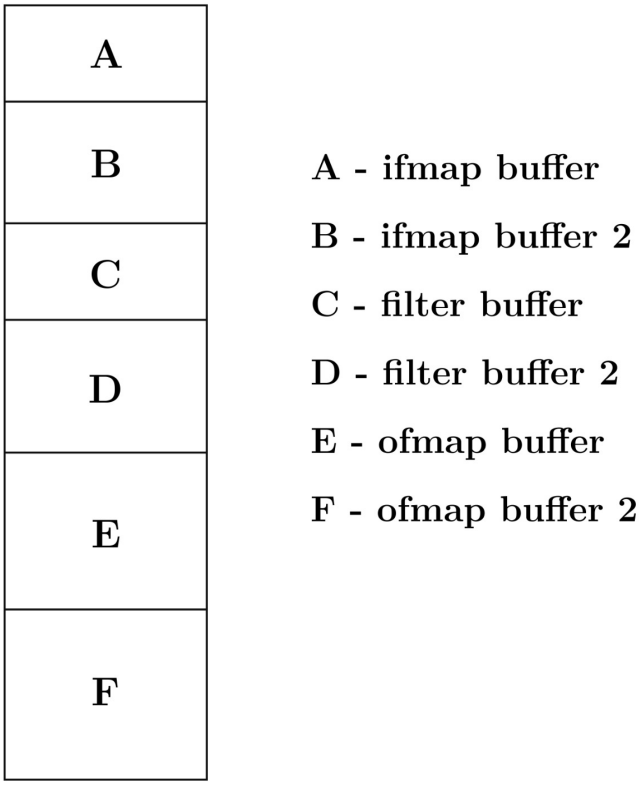

Figure 2: Separate memory sections in current accelerators use.

While some advanced accelerators use a unified memory space, they often lack a sophisticated software layer for dynamic, goal-oriented management, instead relying on complex, pre-computed dataflow maps. The central issue is the absence of a lightweight, adaptive mechanism to intelligently manage the entire on-chip memory pool on a per-layer basis.

1.3 Significance and Potential Contribution

By tackling this problem, this research is poised to make several key contributions to the field of computer architecture and deep learning:

- **A Pathway to Superior Accelerator Performance:** The proposed framework is expected to yield substantial reductions in off-chip memory traffic and overall execution time. Decreasing off-chip access directly lowers the system's energy consumption, a vital consideration for both large-scale data centers and battery-powered edge devices.
- **A Blueprint for Future-Proof Hardware:** By abstracting memory management into a flexible software layer, the underlying hardware becomes more adaptable. This allows a single accelerator design to efficiently execute a wider variety of current and future DL models without needing physical modification.
- **A Novel Framework for Memory Optimization:** The project will produce a tangible set of management policies and a selection algorithm. This framework will empower system designers to reason about and systematically navigate the trade-offs between competing optimization targets, such as prioritizing data locality to save energy versus aggressive prefetching to reduce latency.

1.4 Aims and Objectives

Aim: To develop, model, and validate a dynamic, software-controlled memory management system for a unified on-chip scratchpad, enabling DL accelerators to adapt on a layer-by-layer basis to optimize for performance and energy efficiency.

To realize this aim, the following objectives have been established:

1. To perform a systematic review of on-chip memory architectures, data reuse patterns, and scheduling strategies in existing DL accelerators.
2. To formulate a versatile set of memory management policies, each defining a unique strategy for data tiling and placement within a unified buffer to exploit specific reuse opportunities.
3. To construct a lightweight analytical algorithm capable of analyzing any given DL layer and selecting the most suitable policy to satisfy a user-defined optimization goal and the available memory constraints.
4. To build a simulation environment to model the proposed system and estimate its impact on key performance indicators like off-chip access volume and inference latency.
5. To conduct a rigorous comparative analysis of the proposed dynamic system against a conventional accelerator with static memory partitions to quantify the achievable benefits.

1.5 Research Questions

This study is guided by the following core research questions:

- What strategies can be employed to dynamically partition a unified scratchpad memory to meet the distinct data requirements of individual DL model layers?
- How does the optimization objective (e.g., minimizing memory traffic vs. minimizing latency) influence the selection of memory management policies, and what are the resulting performance trade-offs?
- What is the magnitude of improvement in off-chip access reduction and latency that a flexible, heterogeneous management scheme can offer over traditional, static memory architectures?
- How do factors like total on-chip memory capacity and the precision of data types affect the performance of a dynamic memory management system?

1.6 Scope

This research focuses on the on-chip scratchpad memory subsystem for DL inference accelerators. The core of the work is the development of a software-based management technique for a unified Global Buffer (GLB). The evaluation will be simulation-driven, utilizing a set of standard CNN benchmarks (including ResNet18, MobileNetV2, and others) as workloads. The analysis assumes a layer-by-layer execution model and will investigate inter-layer reuse where the output of one layer is held on-chip for the next. The underlying compute architecture is modeled as a systolic array, a prevalent design in modern accelerators.

1.7 Structure of the Report

The final dissertation will follow a logical structure. Chapter 1 will introduce the project's context, problem, and goals. Chapter 2 will provide a foundation with a comprehensive literature review. Chapter 3 will detail the proposed methodology. Chapter 4 will cover the implementation and experimental design. Chapter 5 will present and critically analyze the findings. Finally, Chapter 6 will offer concluding remarks and discuss potential avenues for future research.

2 Background

2.1 Literature Review

Efficient on-chip memory design is a central theme in DL accelerator research. Existing literature provides a foundation in memory architectures and data scheduling methods.

2.1.1 On-Chip Memory Architectures

The design of on-chip buffers generally follows one of three paradigms:

1. **Separate Buffers:** The most straightforward approach assigns dedicated scratchpads for ifmaps, filters, and ofmaps. While simple to control, this static allocation is inherently inefficient, as the highly variable memory demands of different layers lead to poor utilization.

2. **Semi-Unified Buffers:** An intermediate strategy combines the ifmap and ofmap buffers into a single ”activation” buffer while keeping filters separate. This facilitates inter-layer reuse, as a layer’s output can remain on-chip to become the next layer’s input, but the rigid partition between activations and weights remains.

3. **Global Buffer (GLB):** A fully unified GLB offers maximum flexibility by storing all data types in a common memory space. This architecture is the most promising for adapting to heterogeneous workloads. However, effectively managing a GLB is non-trivial. Prior work has often relied on complex, offline DSE to find a single optimal dataflow for the entire model, a process that is time-consuming and still results in a static configuration.

What is missing from the current body of literature is a lightweight, dynamic software layer that can reconfigure the use of a GLB—including its prefetching space—at the granularity of a single layer, based on a clear optimization objective.

2.1.2 Scheduling and Tiling Techniques

Because entire DL layers rarely fit into on-chip memory, their computations are tiled into smaller blocks. An extensive body of research explores methods for finding optimal tiling parameters to maximize data reuse and thus minimize energy-hungry off-chip memory accesses. These methods often employ techniques like loop transformations or exhaustive DSE to identify efficient configurations. The primary limitation of these approaches is their static nature; they typically derive a single ”best” tiling strategy that is applied uniformly, failing to adapt to the unique characteristics of each layer.

2.2 Theoretical Foundations

This research builds upon established principles in computer architecture and deep learning:

- **Scratchpad Memories:** These are software-managed on-chip RAMs that offer predictable performance, unlike hardware-managed caches. The deterministic access patterns in DL make scratchpads an ideal choice, as they allow for precise offline analysis and planning, which is a prerequisite for our proposed policy selection algorithm.

- **CNN Data Reuse:** Accelerator efficiency is fundamentally tied to exploiting data reuse. Our work will strategically target different forms of reuse:
  - **Intra-layer Reuse:** This involves reusing data within a single layer's computation, such as applying the same filter to multiple locations on an ifmap, or using a single ifmap value for computations with multiple filters.
  - **Inter-layer Reuse:** This occurs when a layer's output (ofmap) is used directly as the next layer's input (ifmap) without being written to off-chip memory.
  - **Global Reuse:** This refers to the reuse of model weights across multiple inference runs. If space permits, keeping weights resident on-chip eliminates the need to reload them for each new input.

3 Methodology

3.1 Research Paradigm

The project will employ a quantitative, simulation-based methodology. This approach enables a flexible and rigorous evaluation of the proposed memory management concepts across numerous models and hardware parameters. We will develop a custom analytical framework to model our proposed system and use an existing, validated simulator for the baseline, ensuring a controlled and reproducible experimental process.

3.2 Optimization Problem Formulation

The memory management technique is designed to solve a constrained optimization problem for each layer. The two primary optimization objectives are:

1. Minimization of Off-Chip Data Transfers.
2. Minimization of Execution Latency.

These objectives are constrained by the finite size of the on-chip GLB. We will investigate two allocation strategies: one that dedicates the entire GLB to active data for reuse, and a second that partitions the space to accommodate prefetching for latency hiding. These constraints are formalized as:

$$GLB \geq I_{Tile} + F_{Tile} + O_{Tile}$$

$$GLB \geq 2 \times I_{Tile} + 2 \times F_{Tile} + 2 \times O_{Tile}$$

Here, I_{Tile}, F_{Tile}, and O_{Tile} denote the respective memory sizes for the tiles of the ifmap, filter, and ofmap. The second equation accounts for double buffering.

3.3 Development of On-Chip Memory Policies

A core component of this research is the creation of distinct memory management policies. Each policy is a specific strategy for tiling data and orchestrating its movement to exploit a particular type of data reuse. The proposed policies are:

- **Policy 1 (Ifmap Reuse Focus):** To capitalize on the extensive reuse of ifmaps, this policy keeps all filters for a layer on-chip while streaming the ifmap in ”sliding window” tiles. This dramatically reduces the number of times ifmap data must be re-read from off-chip memory.
- **Policy 2 (Filter Reuse Focus):** To maximize the reuse of filters, this policy holds the entire ifmap on-chip and processes data one filter at a time. This is most effective in layers where ifmaps are relatively small.
- **Policy 3 (Per-Channel Reuse Focus):** This policy leverages the fact that convolutions operate independently across channels. It processes the data one channel at a time, reducing the amount of filter data that needs to be on-chip at any given moment.
- **Policies 4 & 5 (Partial Reuse Strategies):** These are adaptive versions of Policies 1 and 3, designed for scenarios where the full filter set is too large for the GLB. They tile the filters into smaller blocks, accepting a controlled increase in ifmap re-fetches to make the problem fit within the memory budget.

3.4 Heterogeneous Management Scheme Algorithm

The intelligence of the system resides in an algorithm that generates a *heterogeneous management scheme*—a customized, layer-specific execution plan. The algorithm proceeds in four stages:

1. **Inputs:** It takes a description of the DL network and the specifications of the target accelerator hardware (e.g., GLB size) as input.
2. **Estimation Models:** For every layer in the network, the algorithm uses fast analytical models to calculate the required memory, total off-chip accesses, and latency that would result from applying each of the defined policies.
3. **Analyser and Policy Selection:** A strategic decision-making engine, the ’Analyser’, evaluates the estimated metrics for each policy against the GLB size constraint and the stated optimization goal. It selects the policy that best meets the goal (e.g., the one with the fewest accesses). A secondary metric (e.g., latency) is used to resolve ties.
4. **Output:** The algorithm outputs the complete management scheme, specifying the chosen policy and data tiling for each layer of the model.

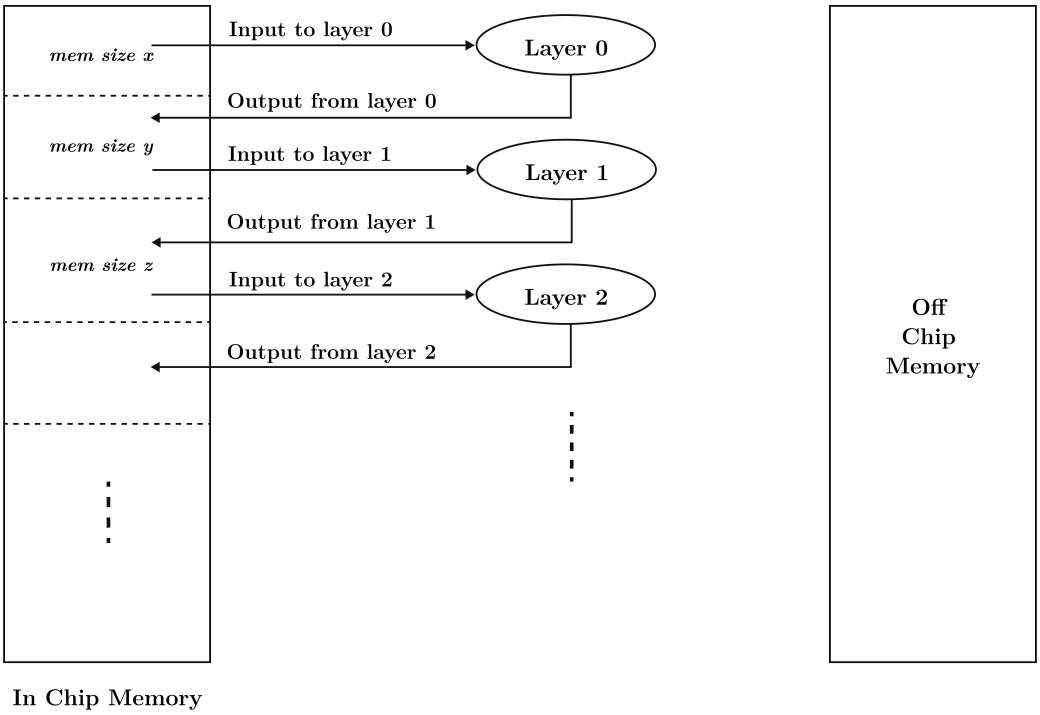

Figure 3: Layer memory sections utilization.

3.5 Simulation and Evaluation Framework

- **Baseline System:** We will use the public-domain SCALE-Sim simulator to model a conventional systolic array accelerator. This baseline will be configured with static, separate buffers for ifmaps and filters. To ensure a fair comparison, we will test several fixed partition ratios (e.g., 25-75%, 50-50%, 75-25%).
- **Proposed System Model:** Our proposed management algorithm will be implemented in a fast, custom-built analytical tool. This allows for rapid generation of management schemes and performance estimates without the overhead of cycle-accurate simulation for every configuration.
- **Experimental Scope:** The evaluation will span multiple GLB sizes, from small (64kB) to large (1024kB), to assess performance under varying memory pressures. A diverse suite of six well-known CNNs will be used as test workloads to ensure the generality of our findings.

3.6 Prototype Specification

The main outcome of the development phase will be a Python-based software prototype. This tool will feature:

- A parser for ingesting DL model specifications.
- An engine containing the analytical performance models for each memory policy.
- The core 'Analyser' module for implementing the intelligent policy selection logic.
- A reporting module to present the generated management schemes and performance predictions.

4 Plan of Work

4.1 Timeline and Deliverables

This project is scheduled to be completed over a period of nine months, structured into several phases with clear milestones, as detailed in Table 1.

Table 1: Project Milestones and Deadlines

Phase	**Key Activities**	**Deliverable**	**Target Date**
1. Foundation	Lit. review, methodology finalization.	Report Chapters 1 & 2	Month 2
2. Implementation	Develop policy engine & analyser.	Functional Software	Month 5
3. Experimentation	Run baseline & proposed simulations.	Complete Raw Results	Month 7
4. Analysis	Interpret data, draft main report.	Full First Draft	Month 8
5. Finalization	Revisions, final submission prep.	Final Report	Month 9

4.2 Project Gantt Chart

A detailed Gantt chart will be provided to visually map the tasks from Table 1 onto the project timeline, illustrating dependencies and durations.

5 Professional, Legal, and Ethical Implications

This research will be executed in accordance with the highest ethical and professional standards, including the IEEE code of conduct.

- **Academic Integrity:** All external sources, including open-source tools (SCALE-Sim) and public model architectures, will be appropriately acknowledged and cited.
- **Data and Software:** The project involves no sensitive or personal data. The simulation data generated will be managed responsibly. The custom software prototype developed will be well-documented to ensure transparency and reproducibility.

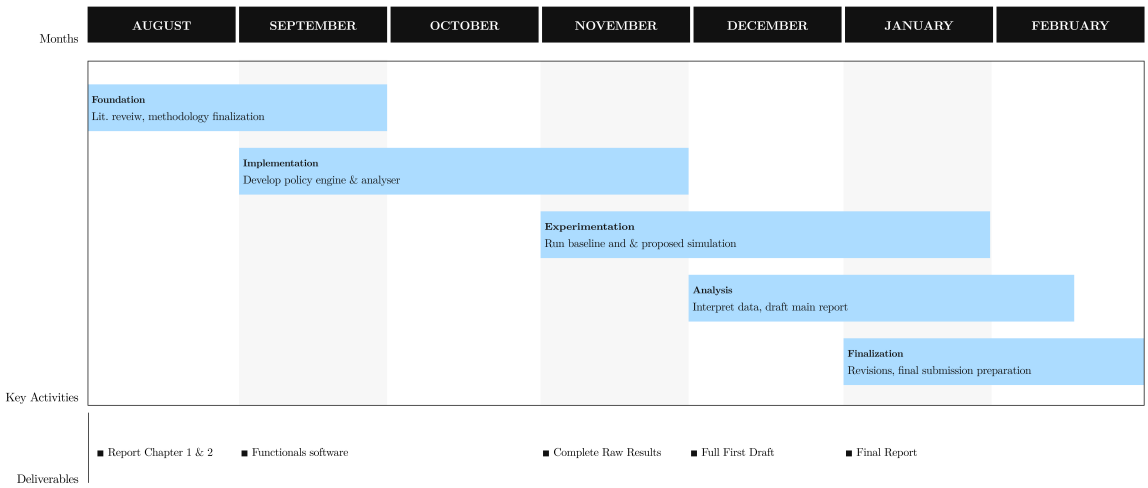

Figure 4: Project Execution Gantt Chart (Placeholder)

- **Societal Impact:** The primary ethical consideration of this work is positive. By developing techniques to reduce the energy consumption of AI computations, this research contributes to the growing field of sustainable and ”green” computing. Improving accelerator efficiency can also democratize access to powerful AI tools by making them viable on low-power devices. The broader societal implications of AI deployment will be acknowledged as important context for this work.

6 Conclusions

The prevailing paradigm of static on-chip memory allocation in DL accelerators is a significant impediment to achieving optimal performance and efficiency. This proposal has outlined a comprehensive plan to research and develop a dynamic, software-driven alternative. By creating a system that intelligently manages a unified memory space through a set of goal-oriented policies, this work seeks to resolve the mismatch between rigid hardware and dynamic workloads. The proposed methodology, based on a combination of analytical modeling and simulation, is designed to rigorously evaluate this approach and quantify its benefits. The expected result is a novel memory management framework that provides a clear path towards building more powerful, adaptable, and energy-efficient hardware for the next generation of artificial intelligence.

References

[1] Chen, Y. H., Krishna, T., Emer, J. S., & Sze, V. (2016). Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE journal of solid-state circuits*, 52(1), 127-138.

[2] Jouppi, N. P., et al. (2017). In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture* (pp. 1-12).

[3] Peemen, M., Setio, A. A., Mesman, B., & Corporaal, H. (2013). Memory-centric accelerator design for convolutional neural networks. In *2013 IEEE 31st international conference on computer design (ICCD)* (pp. 13-19).

[4] Samajdar, A., Joseph, J. M., Zhu, Y., Whatmough, P., Mattina, M., & Krishna, T. (2020). A systematic methodology for characterizing scalability of dnn accelerators using scale-sim. In *2020 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)* (pp. 58-68).

[5] Siu, K., Stuart, D. M., Mahmoud, M., & Moshovos, A. (2018). Memory requirements for convolutional neural network hardware accelerators. In *2018 IEEE International Symposium on Workload Characterization (IISWC)* (pp. 111-121).