



POLITECNICO
MILANO 1863

**SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE**

EXECUTIVE SUMMARY OF THE THESIS

Modeling and Exploration of AI Accelerators based on Digital In-Memory-Computing

LAUREA MAGISTRALE IN COMPUTER SCIENCE AND ENGINEERING-INGEGNERIA INFORMATICA

Author: VALERIA DE GENNARO

Advisor: PROF. CRISTINA SILVANO

Co-advisors: MARCO RONZANI, CRISTIAN ZAMBELLI

Academic year: 2024-2025

1. Introduction

The rapid growth of Artificial Intelligence (AI), recently amplified by Generative AI and Transformer-based models, has transformed multiple domains, including computer vision, natural language processing, and autonomous systems. In particular, image recognition models enable artificial systems to learn visual patterns directly from raw pixel data for tasks like classification, detection, and segmentation. At the core of deep learning lies the multiply-accumulate (MAC) operation. In convolutional neural networks, MAC operations exhibit strong data reuse, as convolution kernels are repeatedly applied to input feature maps. However, accelerating such workloads remains challenging due to the memory-computation bottleneck [3], where frequent data transfers between memory and processing units severely impact energy and latency.

In-Memory Computing (IMC) offers a promising solution by performing computations directly within the on-chip memory arrays [1]. IMC architectures reduce data movement and exploit inherent data reuse, with each cell storing operands and contributing to MAC operations. In particular, digital IMC based on static RAM

technologies is scalable, CMOS-compatible, and highly energy-efficient [1].

Despite these advantages, IMC-based accelerators present complex architectural optimization challenges. Exhaustively exploring design configurations is infeasible due to the exponential growth of the design space. To address this, Design Space Exploration (DSE) and Design of Experiments (DoE) methodologies are employed to efficiently identify Pareto-optimal trade-offs between energy, latency, and resource constraints. The main contributions of this thesis can be summarized as follows:

- Literature review and analysis of the current state of the art on accelerators based on In-Memory Computing technology and related Design Space Exploration methodologies.
- Modeling and characterization of a configurable digital SRAM-based IMC-module suitable for AI accelerators.
- Integration of the SRAM-based IMC-module in a new template architecture for the FactorFlow existing mapping framework.
- Definition of the design space parameters of the template architecture to be explored

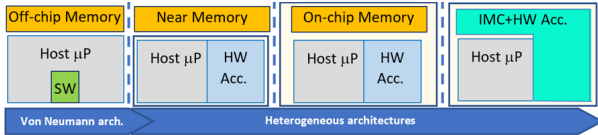


Figure 1: Evolution of memory–compute integration strategies.

and related optimization metrics.

- Development of a multi-objective evolutionary algorithm (MOEA)-based exploration approach, leveraging genetic algorithms (GAs) and DoE techniques to identify Pareto-optimal design solutions within constrained design spaces.
- Experimental comparison of different algorithmic configurations and optimization strategies, including randomized and baseline approaches.
- Experimental evaluation to demonstrate the effectiveness of the proposed exploration methodology to identify Pareto-optimal configurations of the IMC-based template architecture to execute Deep Learning workloads.

2. Background

In-Memory Computing (IMC). Modern deep-learning accelerators have increased computational throughput, but a key bottleneck remains: the **energy cost of data movement**. Most energy is spent transferring weights and activations between processing units and off-chip memory, especially in convolutional workloads. Thus, compute improvements alone are no longer enough—the **memory-transfer overhead** has become the main limitation.

As shown in Fig. 1 [7], architectures have evolved from *near-memory computing* (placing NVM close to processing elements) to tighter on-chip integration (e.g., SRAM or NVM arrays). This trend culminates in **IMC** architectures, where computation occurs **inside the memory arrays**. By embedding simple operations directly in the memory fabric, IMC drastically reduces data movement and achieves much higher energy efficiency, making it especially suitable for data-intensive tasks such as neural network inference and signal processing.

Design Space Exploration (DSE). DSE provides methods and tools to search large de-

sign spaces and identify Pareto-optimal solutions. A preliminary *Design of Experiments (DoE)* phase reduces cost by sampling the space using techniques such as **Latin Hypercube Sampling (LHS)** [6], which ensures uniform, non-clustered coverage.

Non-Dominated Sorting Genetic Algorithm (NSGA-II). This thesis employs NSGA-II for multi-objective optimization. The algorithm evolves candidate solutions through selection, crossover, and mutation. It ranks solutions via non-dominated sorting and uses crowding distance to maintain diversity. Its elitist strategy preserves high-quality solutions, and its $\mathcal{O}(N^2)$ sorting makes it effective for large-scale DSE problems.

3. Related works

Several tools have been proposed for *modeling, mapping, and exploration* of accelerator architectures. **CIMLoop** [5] is an open-source framework for modeling and exploring IMC-based DNN accelerators, addressing key challenges of *flexibility, accuracy, and speed*. Among these tools, **FactorFlow** [8] provides analytical modeling and automated mapping of computational kernels onto Spatial Architectures (SAs). Initially developed for GEMMs and later extended to convolutions, it represents the target SA as a hierarchy of **Memory, Fanout, and Compute** levels, enabling efficient performance estimation and mapping optimization. In DSE, **DiGamma** [4] applies a domain-aware genetic algorithm for hardware–mapping co-optimization of DNN accelerators under design constraints. It integrates an **Optimization Block** and an **Evaluation Block**, the latter leveraging the *MAESTRO* framework for performance evaluation.

4. Proposed Work

The methodology proposed in this thesis builds upon and extends the *FactorFlow* framework, a state-of-the-art tool for mapping computational workloads onto spatial architectures. The proposed extensions enable *FactorFlow* to model IMC-based accelerators, overcoming its original limitations and enabling an exploration process that seamlessly integrates computation–memory co-design.

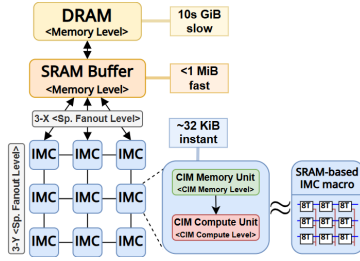


Figure 2: Template architecture including IMC integration through the newly introduced CIM Memory and CIM Compute levels.

4.1. Template Architecture

The **template architecture** used in this thesis builds on **spatial architectures (SAs)**, composed of multidimensional arrays of Processing Elements (PEs) that perform MAC operations. To support IMC, the **FactorFlow** framework was extended with two new levels (Fig. 2): **CIM Memory** and **CIM Compute**. The former models the IMC cell’s storage component, while the latter models its in-memory digital MAC computation. Although real IMC intrinsically combines storage and compute, we separate them to align with *FactorFlow*’s design constraints. Each CIM Memory Level is paired with a corresponding CIM Compute Level, preserving the memory–compute coupling of IMC within FactorFlow’s hierarchical abstraction.

4.2. IMC Modeling and Characterization.

The IMC models used in this thesis rely on the datasets of an IMC module designed by the group of Prof. Zambelli at the University of Ferrara. The IMC array sizes span from 4×4 to 32×32 , and include various sparsity levels and multiple supply voltages (V_{DD}). Figure 3 shows the DIMC array used in this thesis, composed of 8T-SRAM IMC cells. A memory-mapped I/O interface is considered, but not explicitly simulated. All circuits were designed and evaluated by using the 22 nm HP PTM.

Raw data are interpolated and fitted with polynomial models for *read*, *write*, and *hold* operations across different V_{DD} values, array sizes, sparsity levels, and switching activities. To improve accuracy, the **array adder tree** has been added at the fanout level, with parameters derived from empirical measurements. Total energy is computed from 4-bit and 8-bit CLA sim-

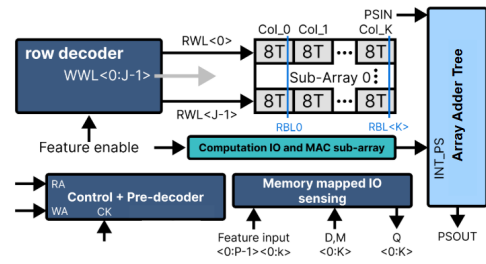


Figure 3: Simulated digital In-Memory Computing (IMC) array.

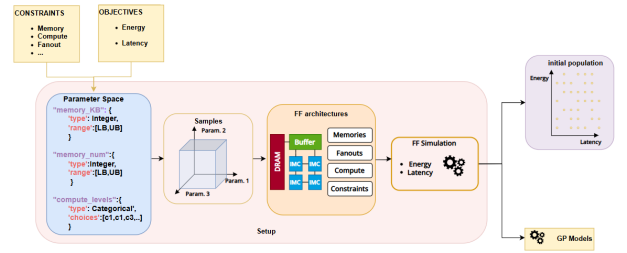


Figure 4: Overview of the setup phase.

ulations as $E_{total} = E_{adder} \cdot \lceil (N_{op} - 1) / (arity - 1) \rceil \cdot N_{iter}$. The same holds for the latency computation.

4.3. Problem Setup

1) Parameter Space Definition The framework defines a multi-dimensional design space spanning architectural and computational parameters. Key parameters include: **Memory Capacity (KB)**, representing available on-chip storage (off-chip DRAM assumed infinite, following *FactorFlow* [8]); **Compute Units**, describing IMC array dimensions; **Fanout Levels**, defining hierarchical parallelism; **Register Count**, impacting reuse and throughput; and **Parallelism Dimension**, specifying the loop axis executed concurrently. Each parameter is assigned a value range or a categorical set, with some—such as memory size, compute allocation, and register distribution—generated dynamically at runtime as continuous values in the range $[0, 1]$. Combined with *energy* and *latency* objectives, these parameters define the **design vector** explored by the framework.

2) Sampling Strategy: Latin Hypercube Sampling is performed through a **sampler** that, given N samples, generates an (N, d) normalized matrix in $[0, 1]$. Normalized entries are then mapped to actual parameter domains: integer values are scaled and rounded, categorical pa-

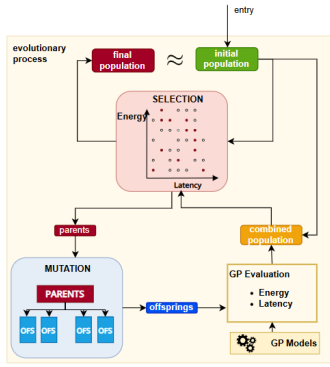


Figure 5: Evolutionary exploration workflow.

parameters selected by index, and continuous ones kept unchanged. Dependent parameters (e.g., memory or compute splits) are post-processed to ensure coherent assignments, such as valid proportions or consistent resource usage.

3) Initial Evaluation and Training The initial LHS-generated architectures are evaluated with *FactorFlow*. Since the designs are not initially in *FactorFlow*’s hierarchical format, each configuration is first translated into a compatible representation. After translation, the framework computes energy and latency, forming the initial population. To avoid repeated full evaluations during search, one Gaussian Process (GP) regressor is trained per objective as a surrogate model. These surrogates provide fast performance estimates for new candidates. At the end of this phase, the full initial population and the trained GP models are ready for the evolutionary exploration stage.

4.4. Multi-Objective Evolutionary Algorithm

With the parameter space defined and GP surrogates trained, exploration proceeds using NSGA-II, a multi-objective evolutionary algorithm. The GP surrogates allow rapid evaluation of many candidate solutions while maintaining population diversity. Key elements include the **population** (P_t) at generation t , **survivors/elites** retained across generations, **parents** (p) generating **offspring** (Q_t) via crossover and mutation, and **fitness**, determined through Pareto dominance and crowding distance. Occasional full simulations are performed on offspring to validate surrogate predictions and provide additional training data for model refinement.

1) Selection In multi-objective optimization, a single “best” solution does not exist, as improving one objective may degrade another. NSGA-II handles this by sorting solutions into dominance-based *Pareto fronts*: the first front F_1 contains all non-dominated solutions, the second front F_2 contains solutions dominated only by those in F_1 , and so forth. Within each front, a *crowding distance* quantifies how isolated each solution is in the objective space: $d_i^m = (f_m(i+1) - f_m(i-1)) / (f_m^{\max} - f_m^{\min})$, summed across objectives to guide selection. Survivors are then selected from the sorted population and used as parents for generating offspring via *tournament selection*.

2) Mutation and Offspring Generation Each parent p produces offspring through stochastic *mutation*, which introduces variability to maintain diversity and prevent premature convergence. Mutation adjusts each parameter based on the **mutation power** (σ), controlling the balance between *diversity* and *refinement quality* of candidate solutions. Equally important is the number of offspring per generation, which controls the balance between *exploration* and *exploitation*: more offspring promote exploration and may overlook some high-quality existing solutions, while fewer offspring favor exploitation by refining the current population.

3) Iteration and Termination Once the offspring population Q_t is generated and evaluated, it is merged with the parent population P_t to form $R_t = P_t \cup Q_t$. Non-dominated sorting and crowding-distance ranking are then applied to R_t , and the top N solutions are retained for the next generation: $P_{t+1} = \text{Survivor_Selection}(R_t, N)$. This process is repeated for a fixed number of generations G . Over successive generations, the population evolves toward convergence, gradually approximating the Pareto-optimal front that captures the best trade-offs between energy and latency. At the end of the exploration, the final Pareto front is obtained by selecting the first front F_1 from the last population, containing the optimal configurations. Each point on this front represents a valid architecture with a distinct compromise between energy efficiency and computational speed, enabling informed design choices according to the target application’s requirements.

Optimization The algorithm was enhanced with several strategies to improve efficiency. *Mutation adaptation* dynamically adjusts the mutation power based on the observed progress of the evolutionary process: if the improvement of a hypervolume indicator is low, the mutation is increased; otherwise, it is reduced to remain near the optimal zone. *Mutation decay* increases the mutation by 15% at each iteration to maintain exploration. *Simulated annealing-based adaptation* updates the mutation power according to a probabilistic rule governed by an *annealing temperature* T_t , which decreases progressively over generations. Finally, *retraining* the GP models mitigates prediction errors when encountering previously unseen configurations.

5. Experimental Results

5.1. Experimental Setup

The proposed methodology is evaluated using 5 **VGG-16** layers as workloads, testing different combinations of baseline NSGA-II and the proposed optimizations. The comparison metrics include more than just visual inspection of the Pareto front: **Hypervolume**: measures the volume of the objective space *dominated* by a Pareto-optimal set $P(Y)$ and bounded by a reference point v_{ref} , representing the Lebesgue measure of the dominated region. **ADRS**: quantifies how closely the Pareto front obtained by an algorithm approximates a reference front. Here, the reference is defined as the union of all points explored by all algorithms, so ADRS captures the relative distance between each algorithm’s solutions [2].

5.2. Results

The evaluation focuses on the most effective configuration identified during experimentation: the **homogeneous exploration** setup. In this setting, 250 initial samples are evaluated using *FactorFlow*, followed by 125 offspring per generation across 40 generations. Mutation adaptation is enabled, retraining is disabled, and a 1:1 offspring-survivor ratio is applied to balance exploration of new candidates with exploitation of high-quality ones.

Figure 6a illustrates the distribution of explored points and highlights a consistent domination gap in favor of the **NSGA-II** configuration

combining both *Decay* and *Simulated Annealing*. The trend is reinforced by the **hypervolume** metric in Figure 6b, which shows a rapid early increase followed by stabilization, indicating convergence of the search process.

The **ADRS** metric (Figure 6c) further confirms these findings: the NSGA-II variant with Decay and SA achieves $\text{ADRS} = 0$, thus defining the reference Pareto front and dominating all competing approaches. Overall, the homogeneous exploration scenario validates the proposed DSE methodology, demonstrating strong performance across metrics and underscoring the benefits of combining Decay and SA-inspired perturbations to simultaneously improve convergence and maintain diversity within the population.

Using *FactorFlow*, we modeled a Colonnade-inspired reference architecture with two fanout stages, each a 32×32 IMC array including partial-sum registers to capture accumulation and internal dataflow. Figure 6d shows the resulting Pareto front. The optimized NSGA-II configuration achieves up to **99% lower energy** and **41–44% reduced latency** compared to the baseline, demonstrating that the proposed exploration framework can efficiently identify high-performance IMC designs, even under strict architectural constraints, and significantly outperform traditional Colonnade-style solutions.

6. Conclusions

This thesis work successfully demonstrates the effectiveness of integrating accurate, In-Memory Computing (IMC)-aware evaluation models directly into a Design Space Exploration (DSE) framework. This tight integration moves beyond conventional, high-level abstractions, thereby enabling a far more accurate and hardware-aware exploration process that leads to high-quality, non-trivial optimizations. As a result, the proposed methodology consistently identifies novel accelerator architectures that populate a superior Pareto front, achieving **energy-latency** trade-offs that traditional methods would miss. This validated approach provides a robust and scalable foundation for the next generation of IMC accelerator co-design. Future work could focus on enhancing this foundation by developing an automated parser for

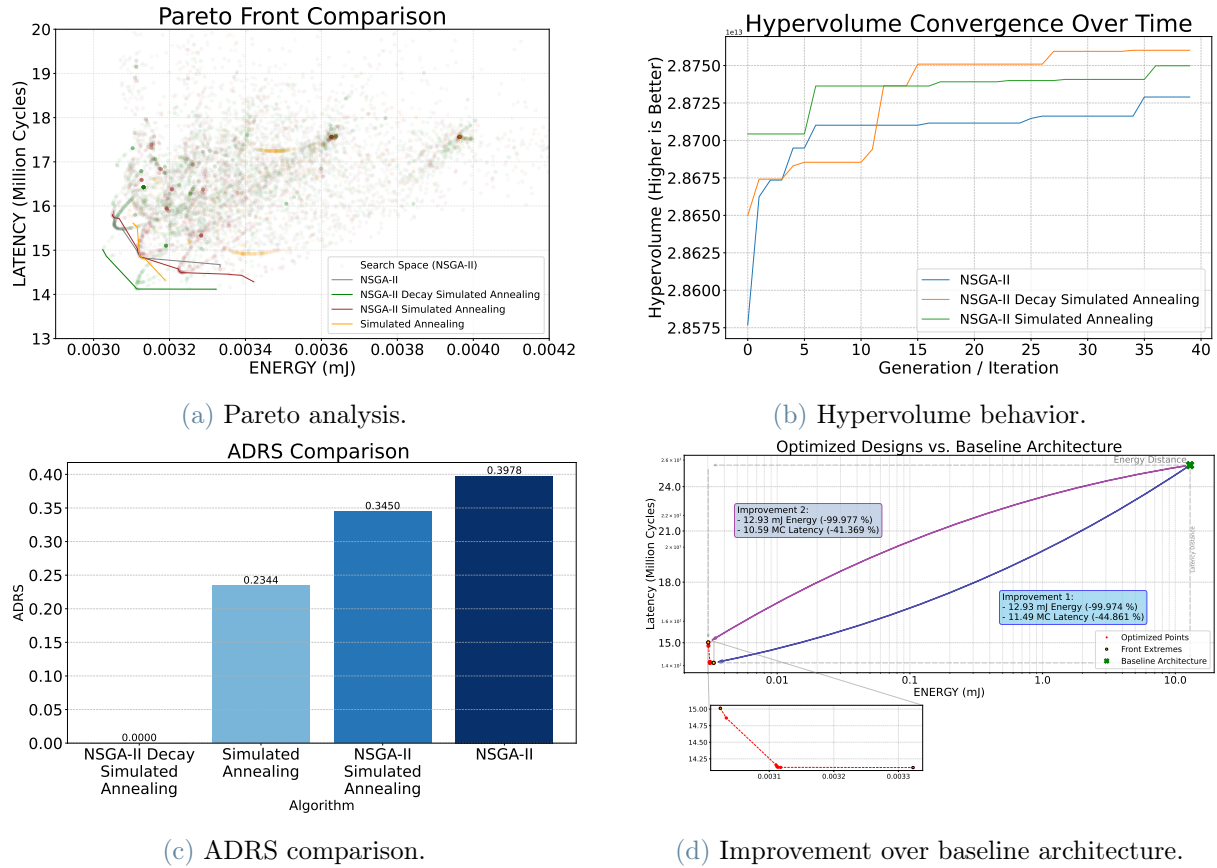


Figure 6: Exploration results. (a) Pareto fronts, (b) Hypervolume convergence, (c) ADRS comparison, and (d) Architecture improvement.

existing macro libraries, such as those from CiM-Loop. Another valuable improvement concerns the **extension of the IMC characterization dataset**, together with the increase of the design space, adding more exploration parameters.

References

- [1] G. Desoli et al. 16.7 a 40-310tops/w SRAM-based all-digital up to 4b in-memory computing multi-tiled NN accelerator in FD-SOI 18nm for deep-learning edge applications. In *2023 IEEE ISSCC*. ISSN: 2376-8606.
- [2] G. Palermo et al. ReSPIR: A response surface-based pareto iterative refinement for application-specific design space exploration. *28(12):1816–1829*.
- [3] Onur Mutlu et al. A modern primer on processing in memory, 2025.
- [4] Sheng-Chun Kao et al. DiGamma: Domain-aware genetic algorithm for HW-mapping optimization for DNN accelerators. In *2022 DATE*. IEEE.
- [5] T. Andrulis et al. CiMLoop: A flexible, accurate, and fast compute-in-memory modeling tool. *arXiv*. Publisher: arXiv Version Number: 2.
- [6] C. Devon Lin and Boxin Tang. Latin hypercubes and space-filling designs, 2022.
- [7] S. Perri, C. Zambelli, D. Ielmini, and C. Silvano. Digital In-Memory Computing to Accelerate Deep Learning Inference on the Edge. In *2024 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 130–133. IEEE, 2024.
- [8] M. Ronzani and C. Silvano. FactorFlow: Mapping GEMMs on spatial architectures through adaptive programming and greedy optimization. In *Proceedings of the 30th Asia and South Pacific Design Automation Conference*, pages 706–712. ACM.