ASP-DAC 2025

FactorFlow: Mapping GEMMs on Spatial Architectures through Adaptive Programming and Greedy Optimization

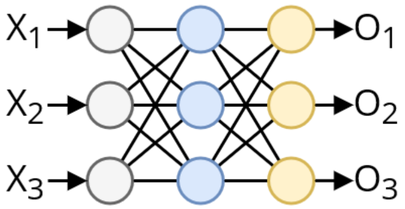
Marco Ronzani
Speaker – PhD Student – marco.ronzani@polimi.it

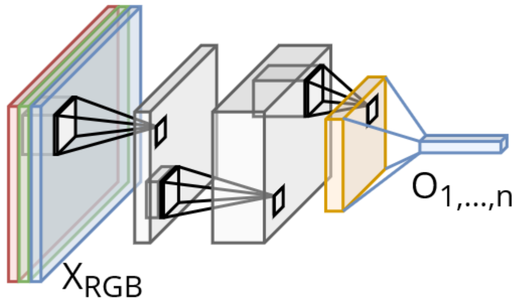
Cristina Silvano
Full Professor – cristina.silvano@polimi.it

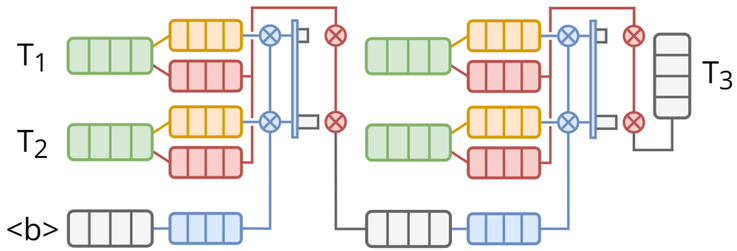
POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Motivation



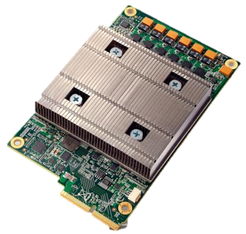


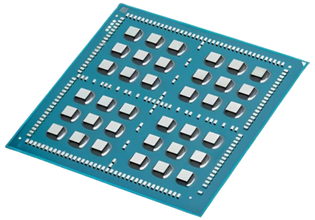


MLP CNN Transformer

MFLOPs GFLOPs TFLOPs per inference

SPATIAL ARCHITECTURES

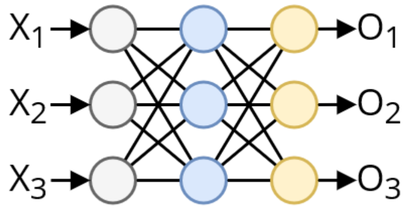


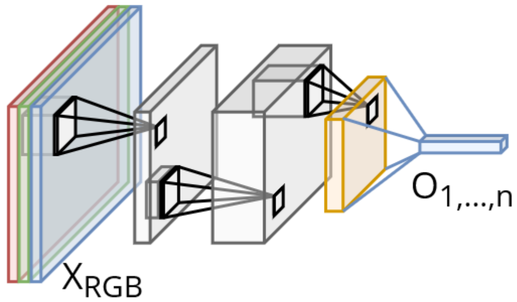


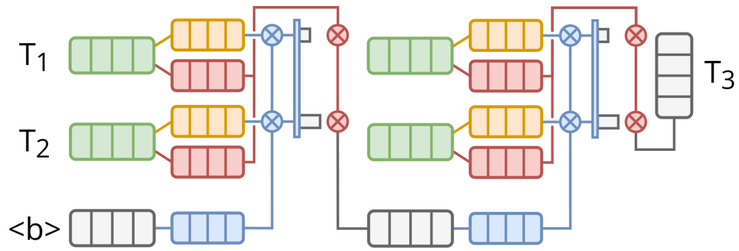
CPU GPU Systolic Array Processing Elements Mesh

General Purpose – High Perf. – Power-Hungry Specialized – High Perf. Per Watt

Source: [13, 14, 15]





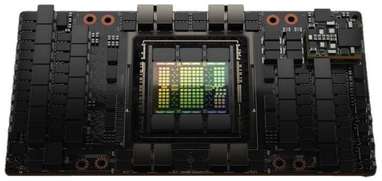


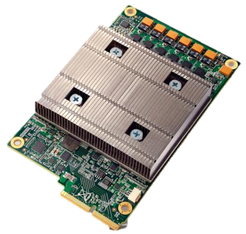
MLP CNN Transformer

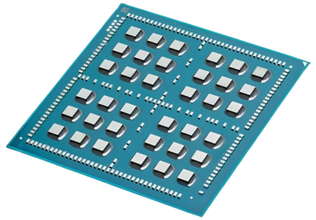
MFLOPs GFLOPs TFLOPs per inference

GOAL:
to minimize the enegy and latency of running AI kernels

MAPPING:
determines the workload's execution on the hardware







CPU GPU Systolic Array Processing Elements Mesh

General Purpose – High Perf. – Power-Hungry Specialized – High Perf. Per Watt

Source: [13, 14, 15]

Contributions

SoA Analysis

Many mapping techniques.

No current mapping tool focuses on GEMMs.

Mapping Formalization

Mathematical formulation of the mapping problem.

Map-space size analysis.

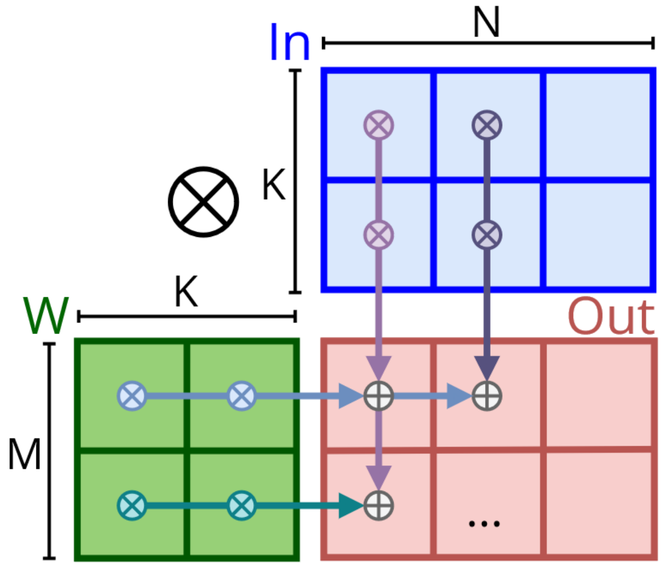
New Mapping Tool: FactorFlow

Three novel robust heuristics to map GEMMs.

Mapping Tools Comparison

FactorFlow finds **1-161x better mappings** in up to **205x less time** than four SoA tools.

General Matrix Multiplication (GEMM)



Definition:

$$Out = W \cdot In + Bias$$

$$In \in \mathbb{M}_{K \times N}$$

$$W \in \mathbb{M}_{M \times K}$$

$$Out,\ Bias \in \mathbb{M}_{M \times N}$$

Nested Loop Form:

```
Out = Bias
for m in [0,M):
  for k in [0,K):
    for n in [0,N):
      Out[m,n] +=
        W[m,k]·In[k,n]
```

Multiply and Accumulate (MAC)

- Each operand is orthogonal to a loop ⇒ **data reuse**
- Regular data dependencies ⇒ **parallelism opportunities**
- Loop order is arbitrary, a loop can be split in multiple copies.

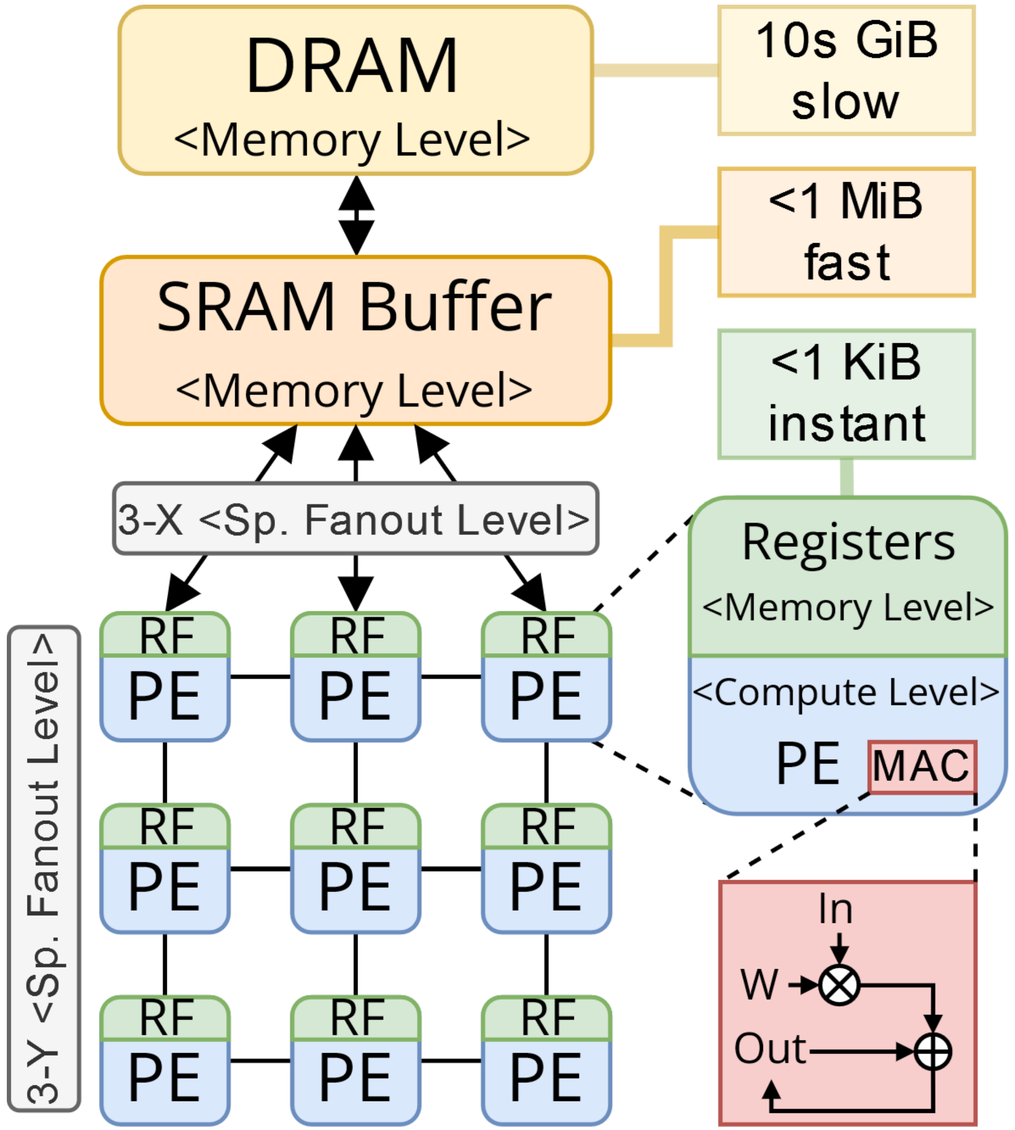
Spatial Architectures (SAs)

Components:

- Array of Processing Elements (PEs)
- Memory hierarchy
- Interconnects

Modeled as a **hierarchy of levels**:

- Memory level
- Spatial fanout level
- Compute level



Performance and Data Reuse

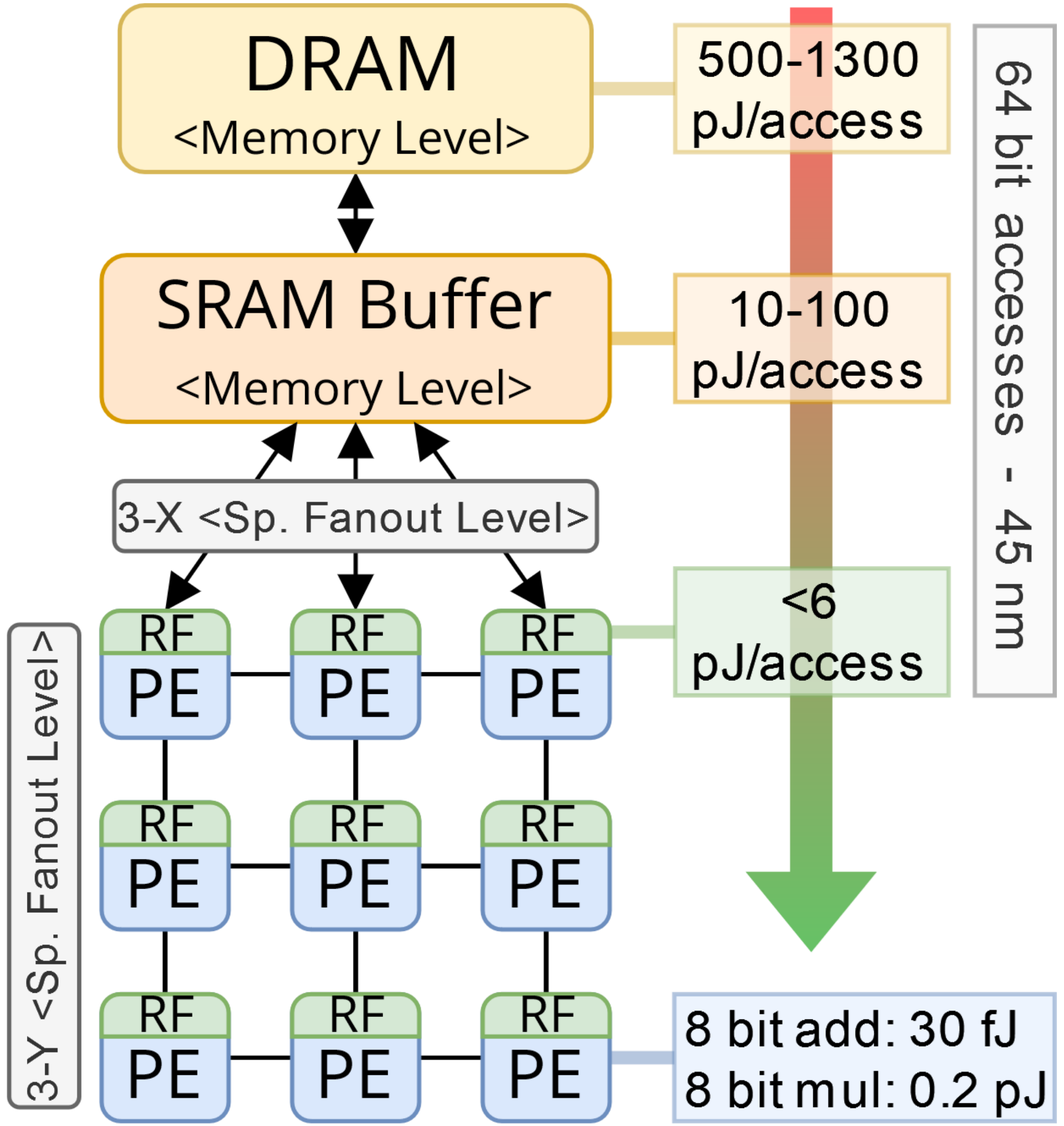
Performance metrics:

- Energy
- Latency
- Energy-Delay Product (EDP)

Energy and **latency** of **memory accesses** dominate those of **compute**.

Memory hierarchy and interconnects ⇒ exploit **data reuse!**

Multiple PEs ⇒ exploit **parallelism!**

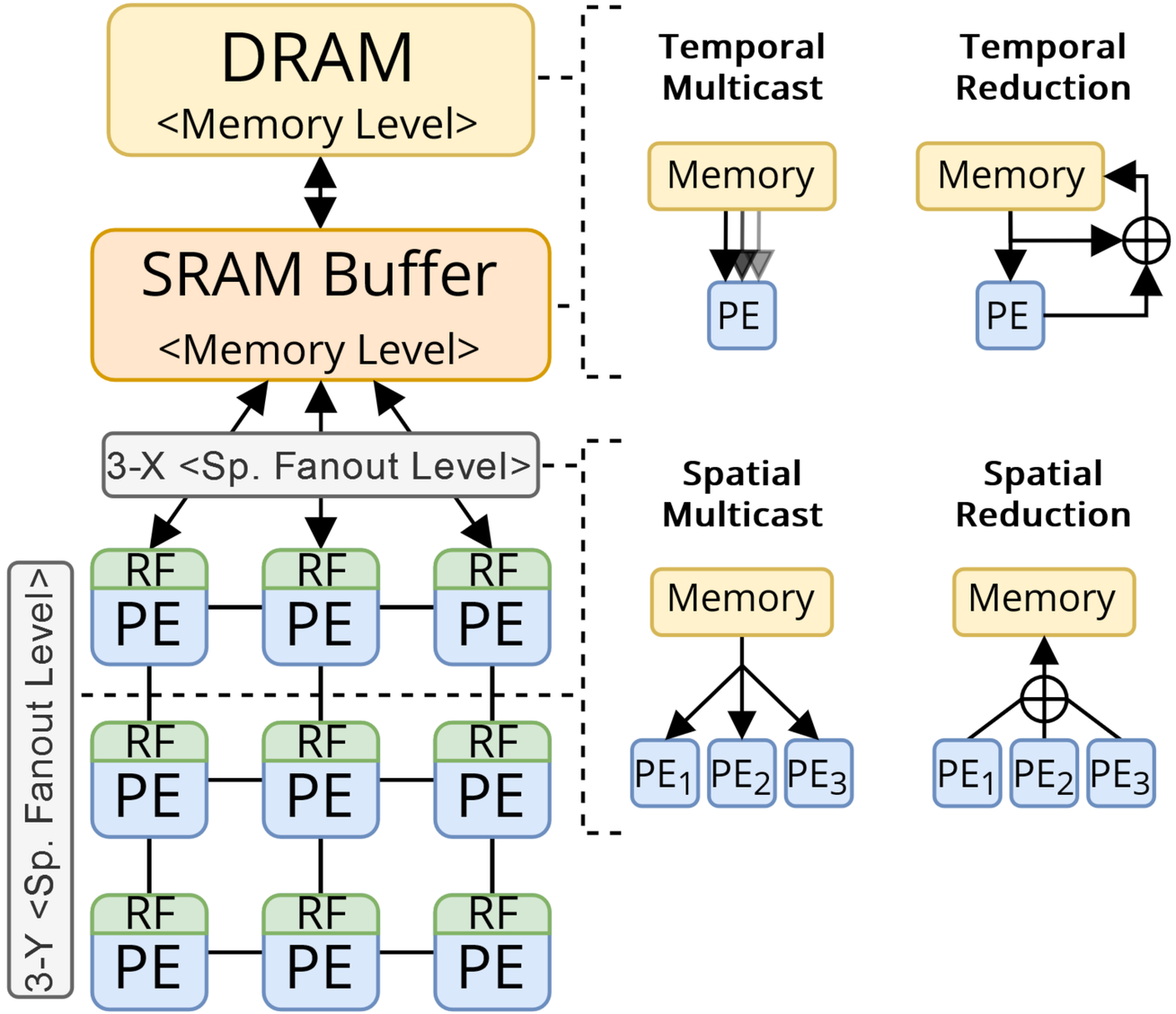


Source: [11]

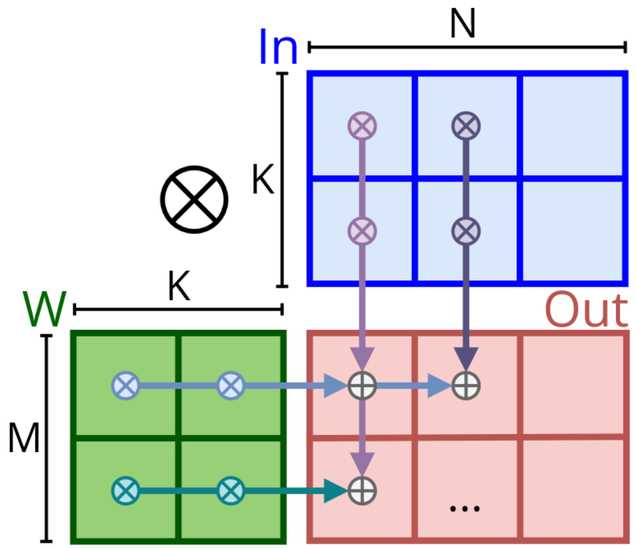
Performance and Data Reuse

- Many types of reuse
- Arbitrary data allocation
- Flexible data movement
- Several orders of computation

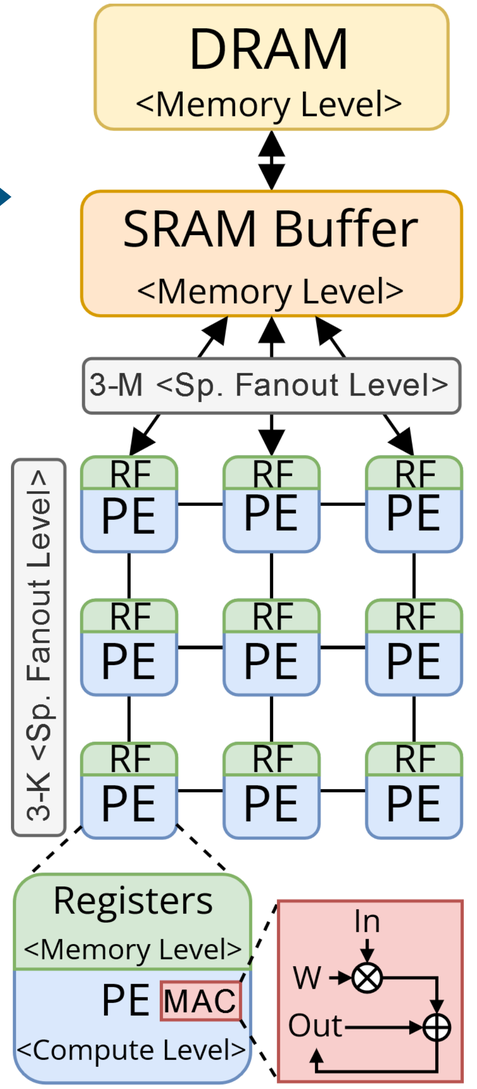
⇒ A **mapping** exploits data reuse and parallelism!



The Mapping Problem



WITH MINIMAL ENERGY AND LATENCY



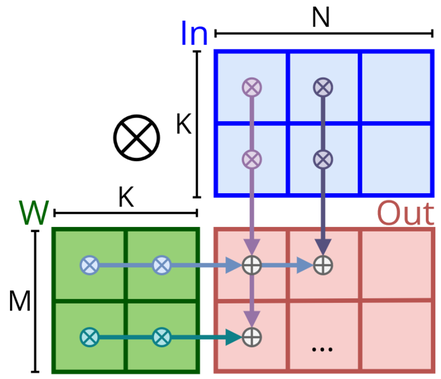

```
for m in [0, M):
  for k in [0, K):
    for n in [0, N):
      <<MAC>>
```

Mapping decisions:

1. Tiling
2. Parallelism strategy
3. Loop ordering

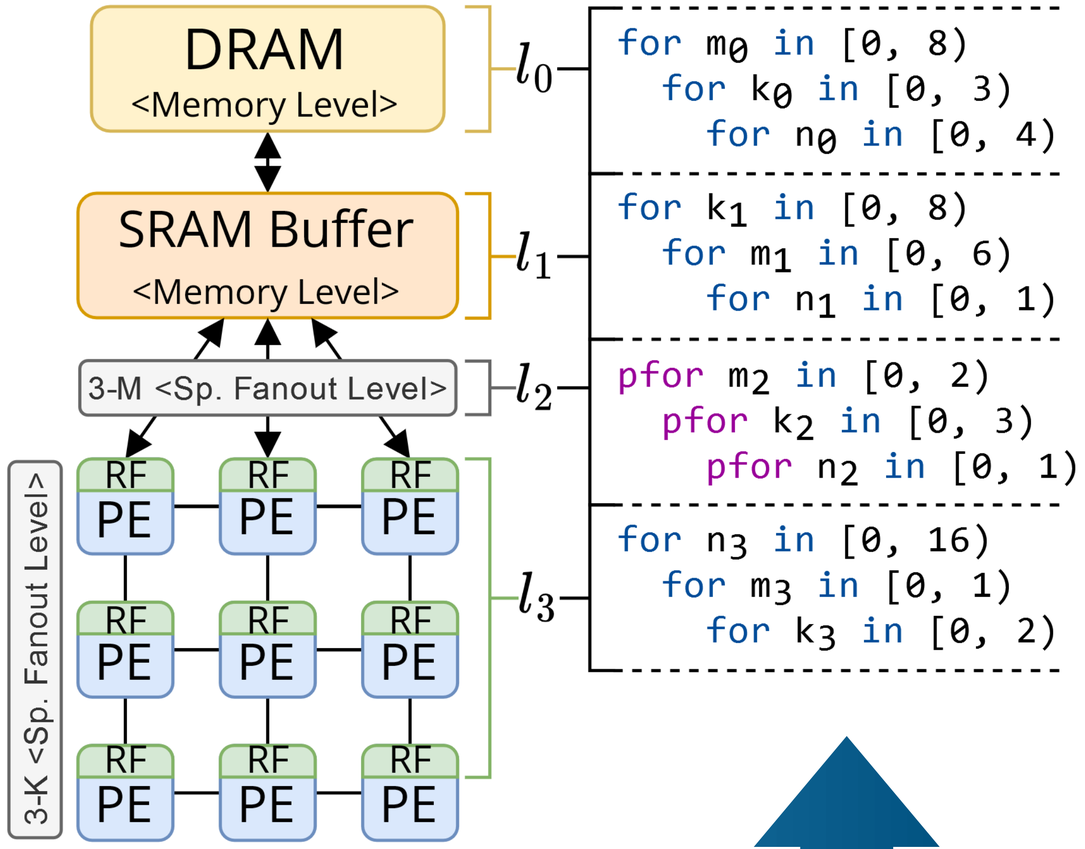
Source: [5, 6]

The Mapping Problem




```
for m in [0, M):
  for k in [0, K):
    for n in [0, N):
      <<MAC>>
```

WITH MINIMAL ENERGY AND LATENCY



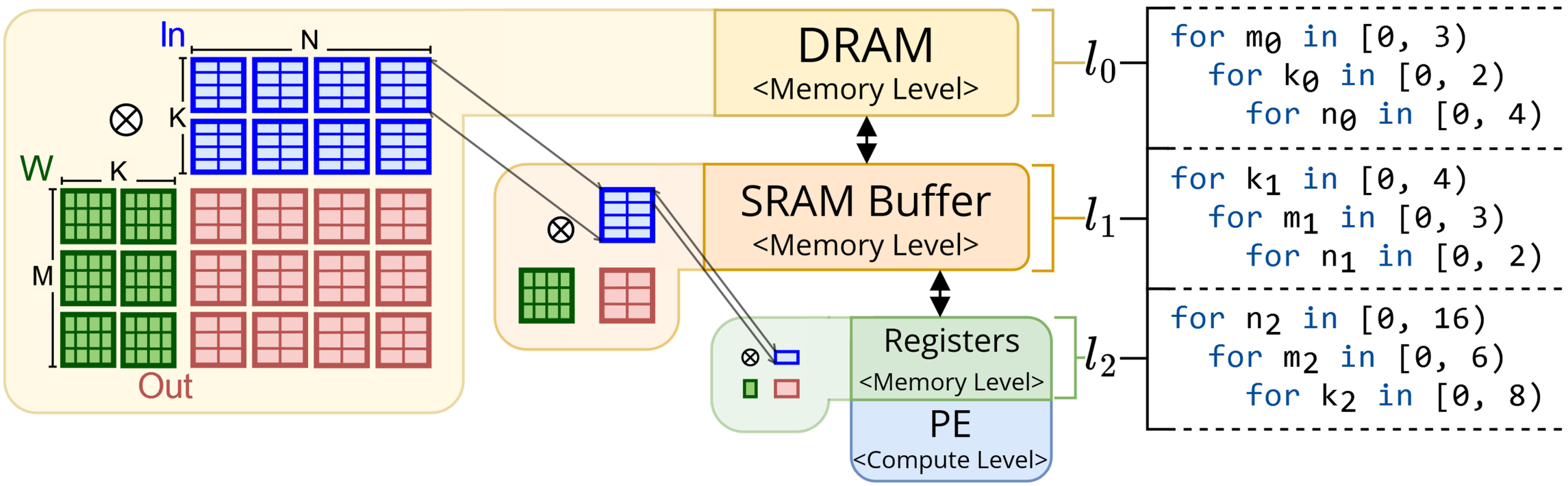
How: by distributing prime factors of total loop iterations to SA levels.

Source: [5, 6]

Tiling

Fits operands on memory levels by tiling them.

Modeled by allocating GEMM iterations to each **memory level**.

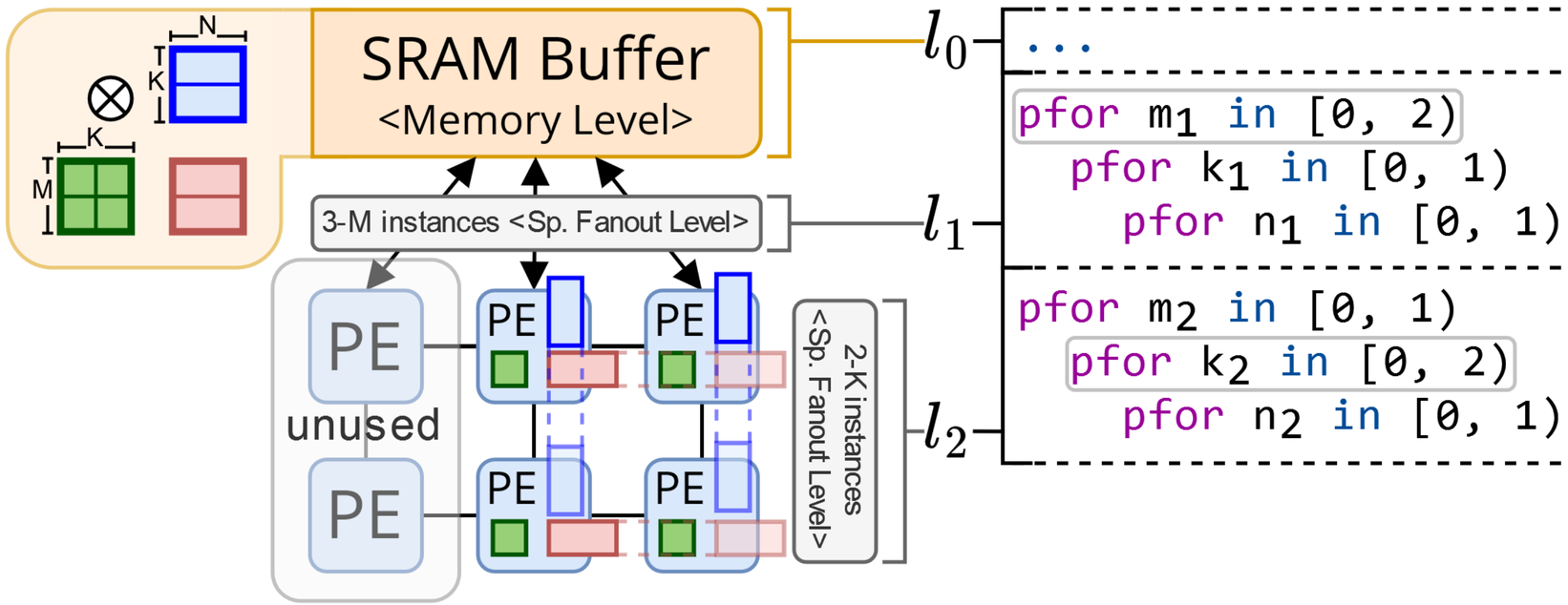


Parallelism Strategy

Unfold some iterations in parallel over the PEs array.

Modeled by replicating GEMM's loops on each **spatial fanout level**.

Let "**pfor**" indicate spatial iterations.



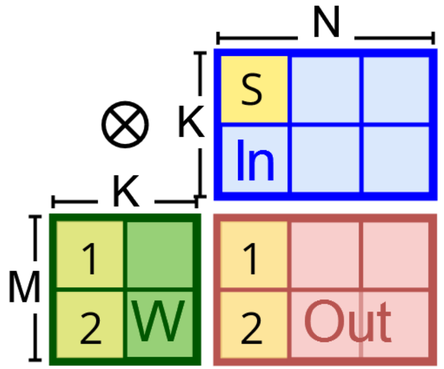
Loop Ordering

The order of each loop triplet dictates a **dataflow.**

The operand orthogonal to the innermost loop is reused.

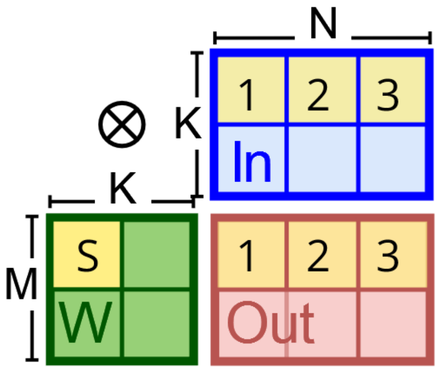
Input Stationary

```
for n in [0, N)
 for k in [0, K)
  for m in [0, M≠1)
```



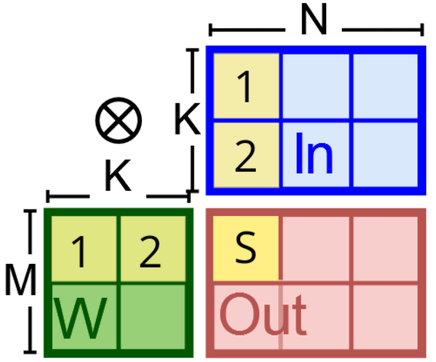
Weight Stationary

```
for m in [0, M)
 for k in [0, K)
  for n in [0, N≠1)
```



Output Stationary

```
for m in [0, M)
 for n in [0, N)
  for k in [0, K≠1)
```



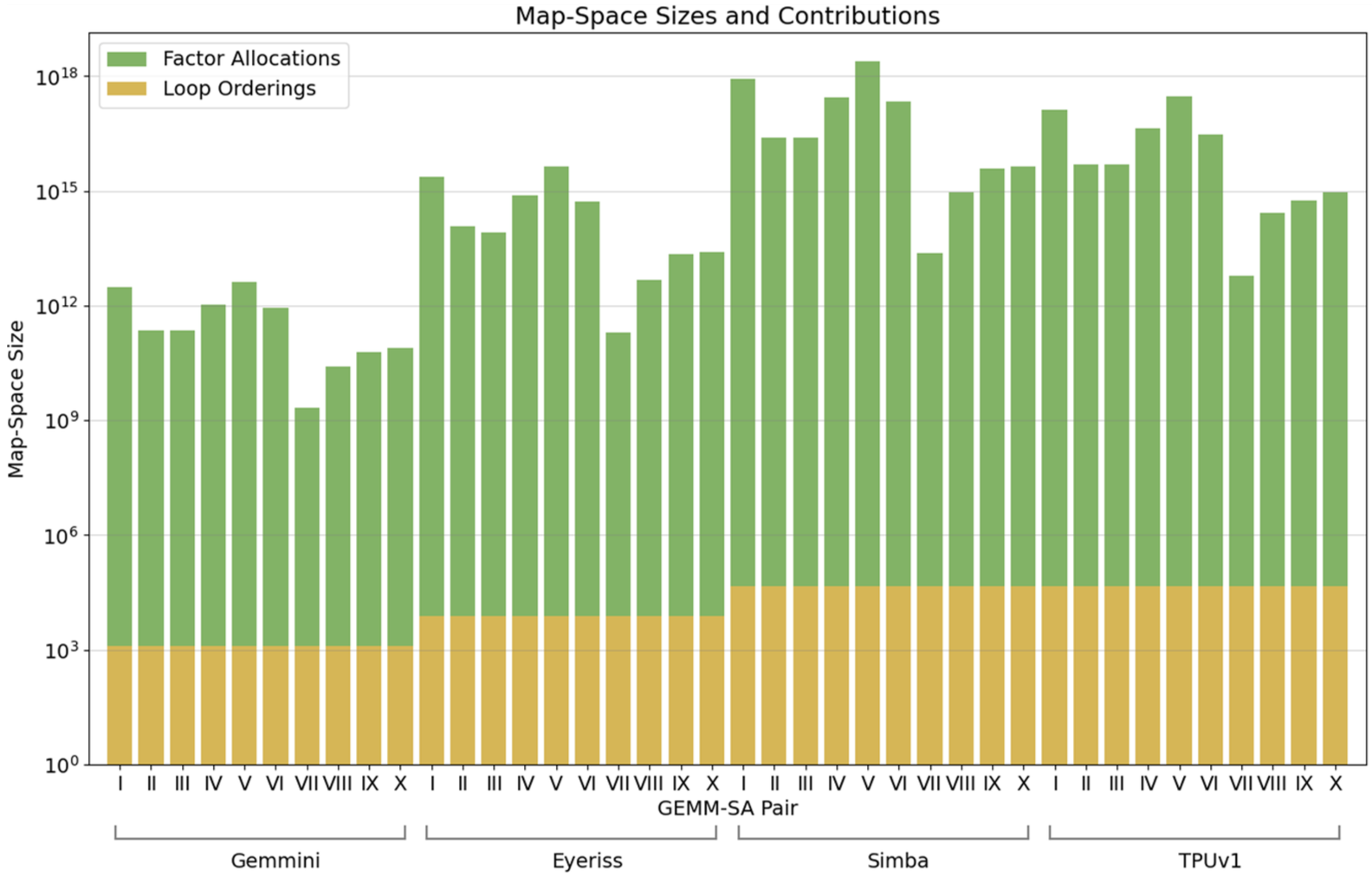
Map-Space Sizes

Map-space: set of all mappings for a GEMM-SA pair.

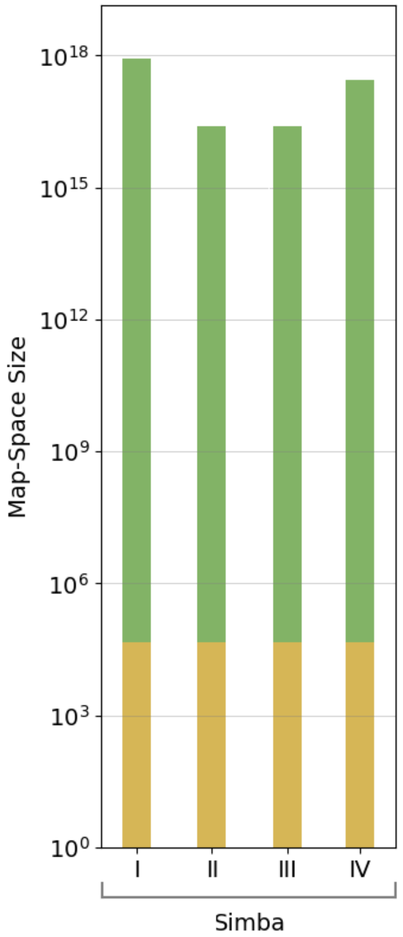
The mapping problem is complex because: **map-spaces are huge**!

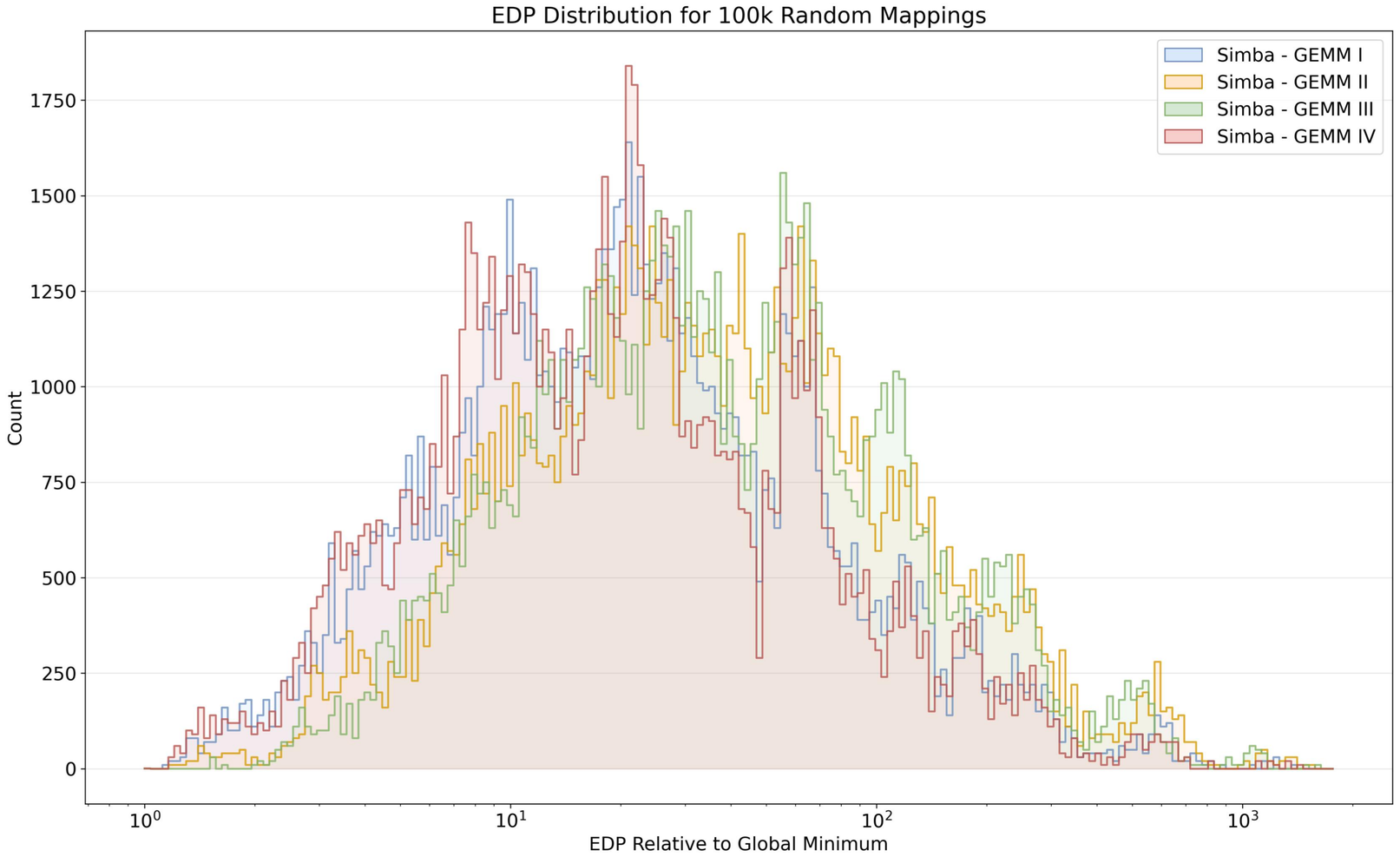
Major size contributor: **factor allocations**!

- tiling
- parallelism strat.



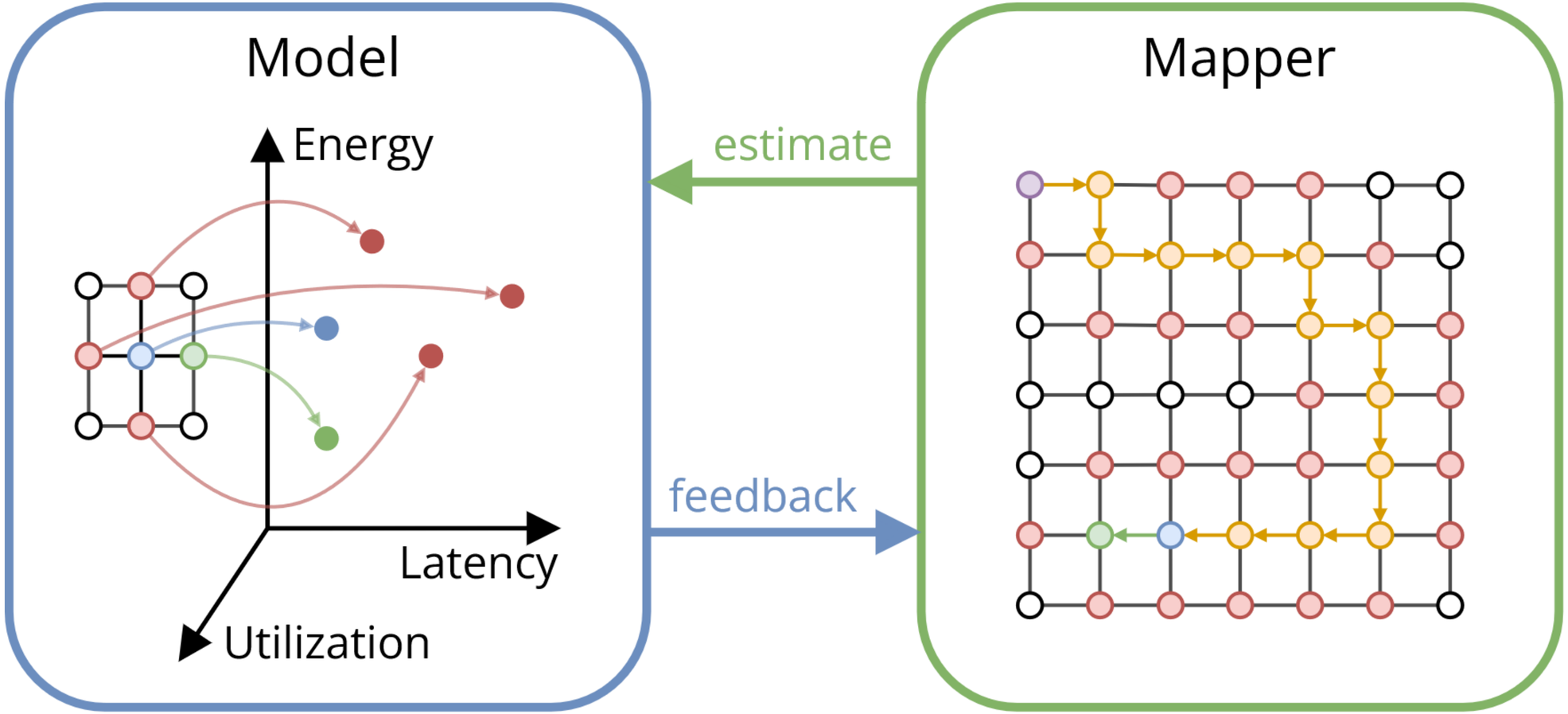
Map-Space Sizes





Near-optimal mappings are rare!

Mapper and Model Paradigm



Objective: **Minimize the EDP**

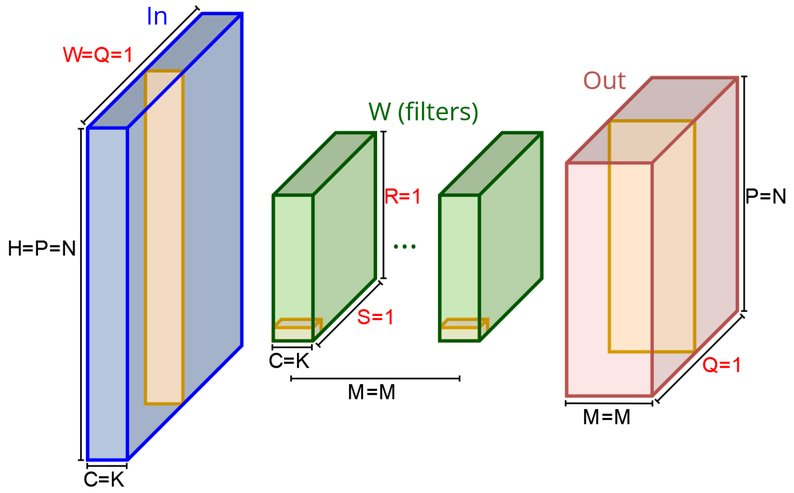
Source: [5]

SoA Mapping Tools

Limitations:

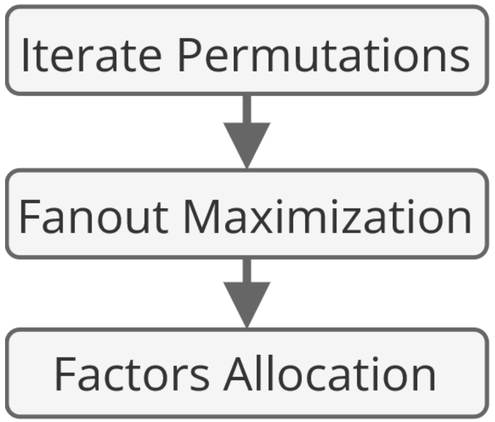
- Focused on **convolutions** (GEMMs as a byproduct)
- **Lack of comparison** with each other

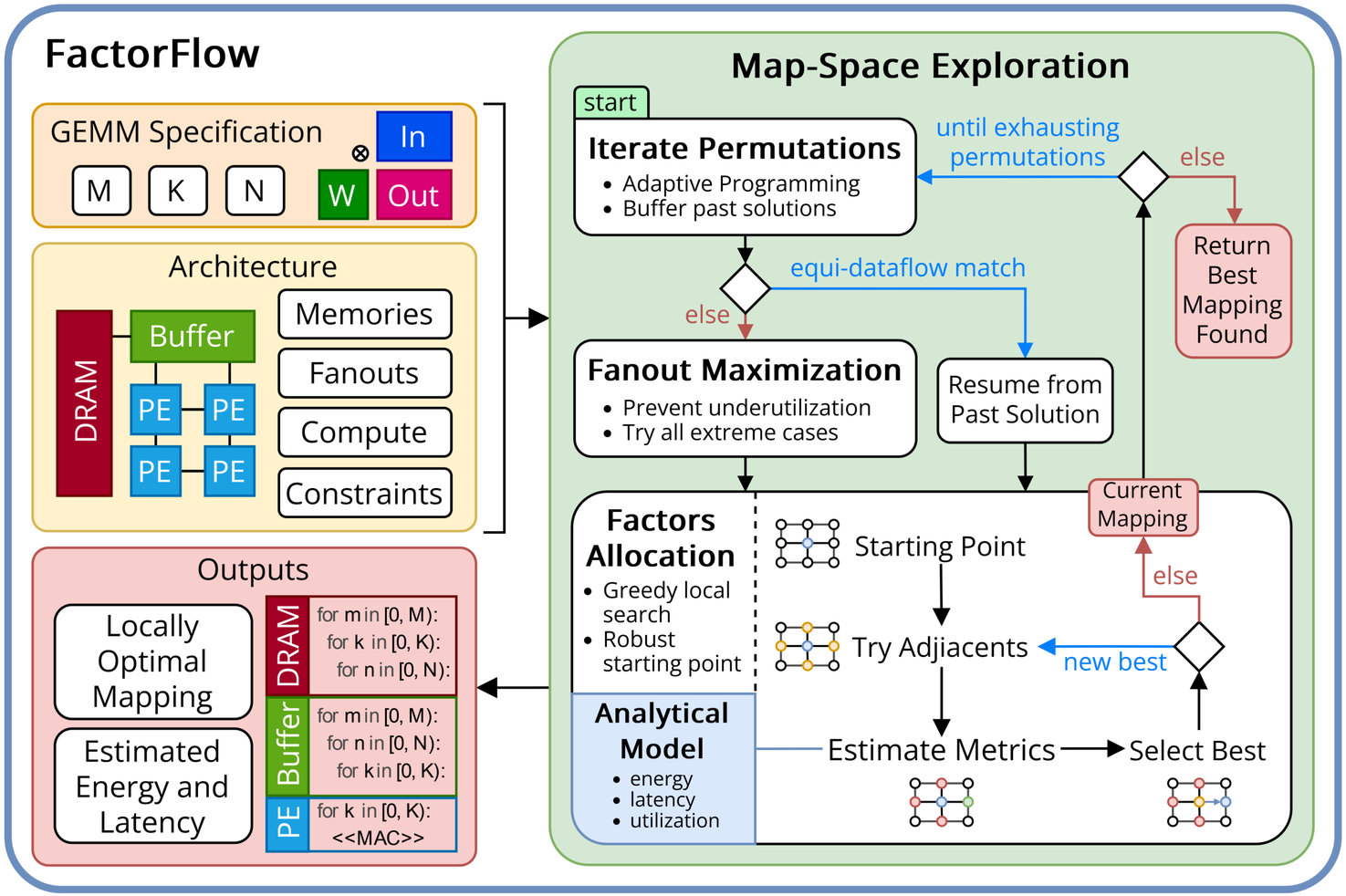
Mapping Tool	Approach	Flexible
Timeloop [5]	random search	yes
GAMMA [6]	genetic algorithm	no
FLASH [7]	exhaustive	no
LOMA [8]	exhaustive pruned	yes
SALSA [9]	simulated annealing	yes
CoSA [10]	mixed integer programming	yes



Our Appoach: FactorFlow

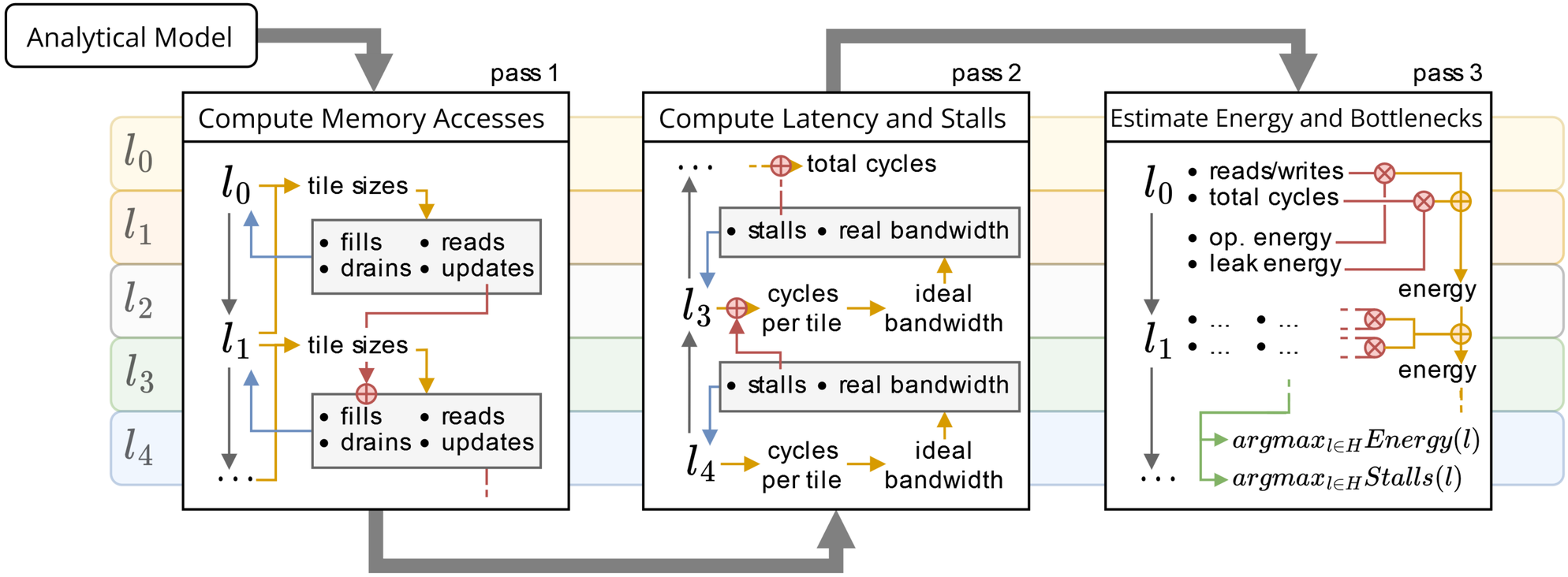
- Specialized for GEMMs
- Comprises a mapper and a model
- Implements three novel heuristics





Analytical Model

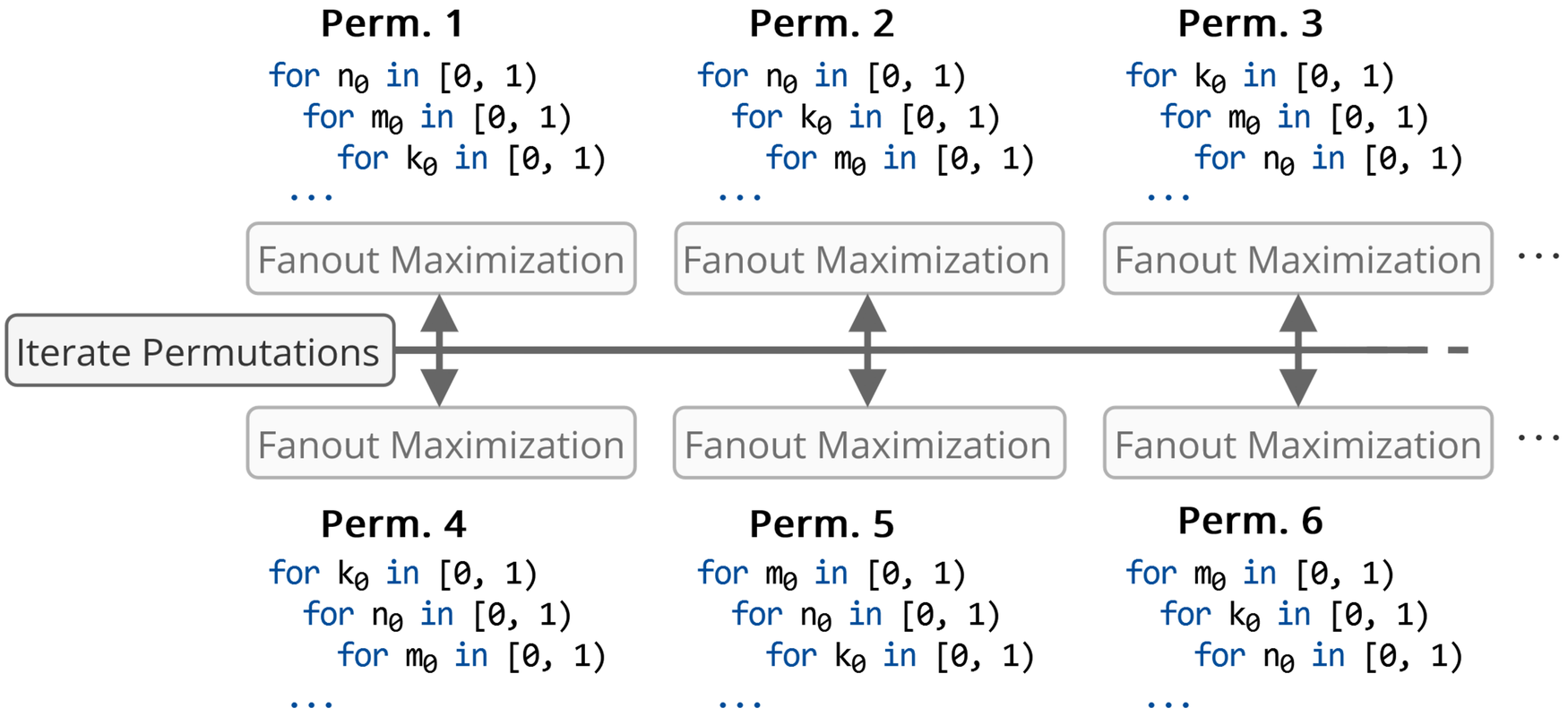
- Fully flexible, can model most SAs
- Functionally equivalent to Timeloop [5]
- Three passes over the SA hierarchy
- Max measured execution time: 1 ms



Step 1: Iterate Permutations

Exhaustively try loop permutations. Total permutations: $6^{\#levels}$

Target decision: loop ordering



Step 1: Iterate Permutations

Exhaustively try loop permutations.

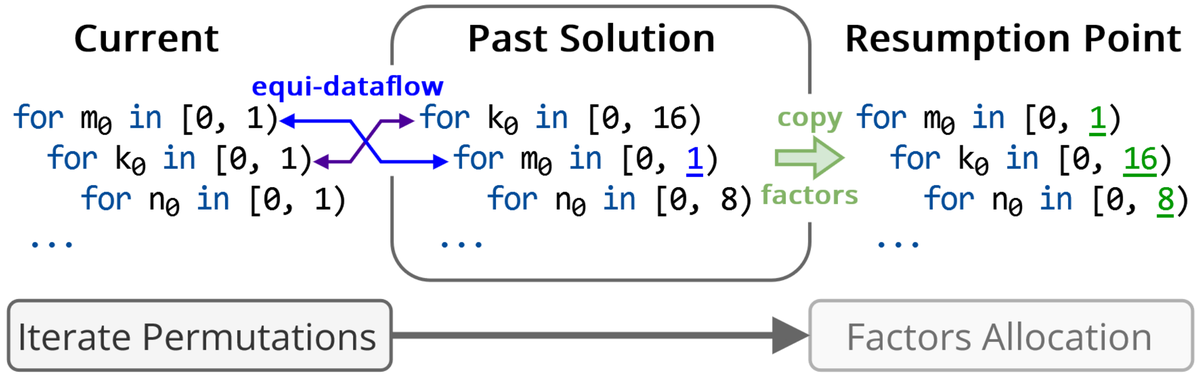
Total permutations: $6^{\#levels}$

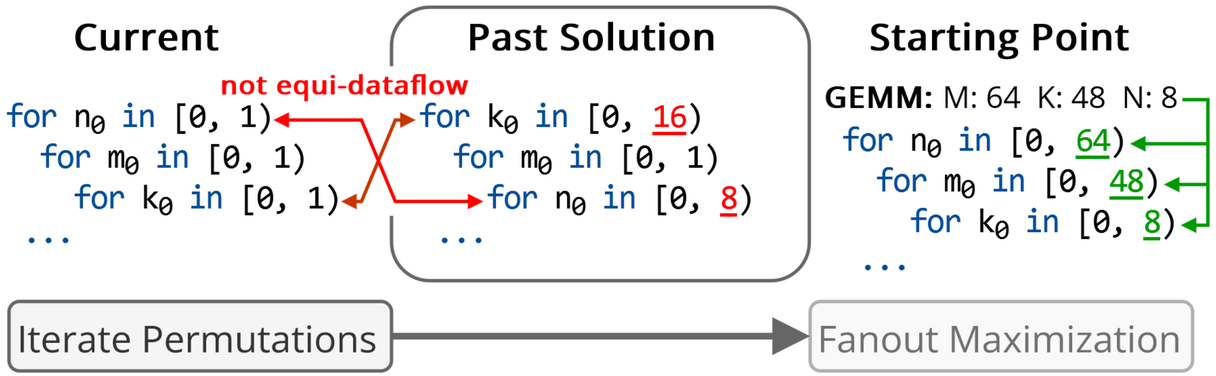
Equi-dataflow: same relative order of loops with >1 iteration, same EDP.

Adaptive programming: speedup exploration of equi-dataflow permutations by restarting from a past solution's factors allocation.

Must buffer past solutions.

Equi-dataflow matches are likely, as are loops with 1 iteration.



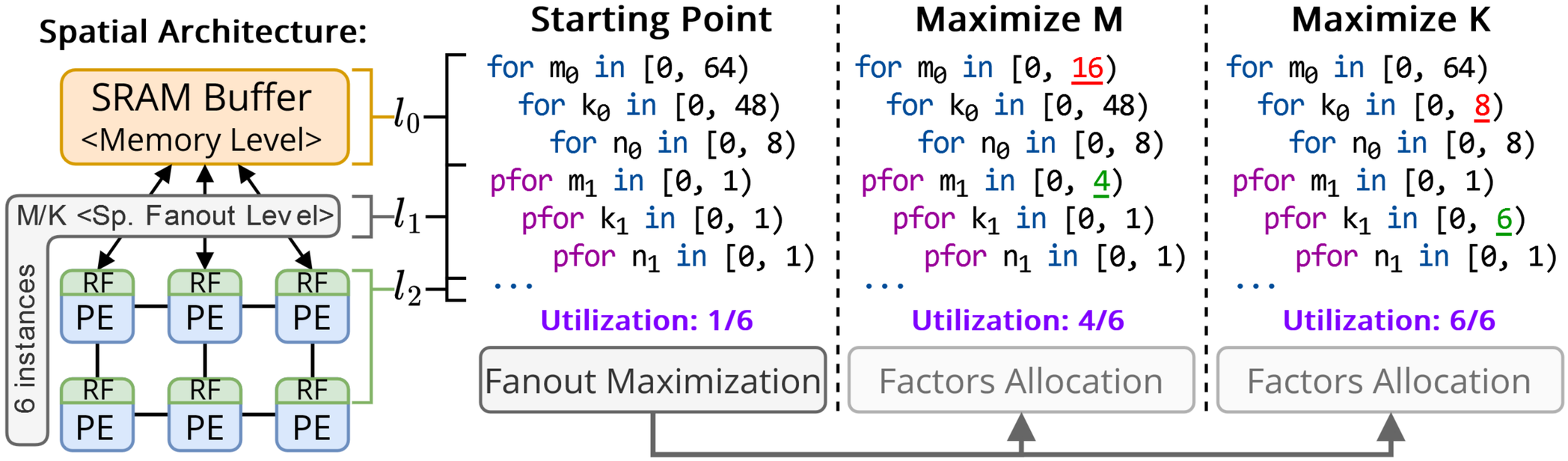


Step 2: Fanout Maximization

Higher utilization increases reuse and parallelism.

Target decision: parallelism strategy

Try all mappings saturating instances with different spatial dimensions.



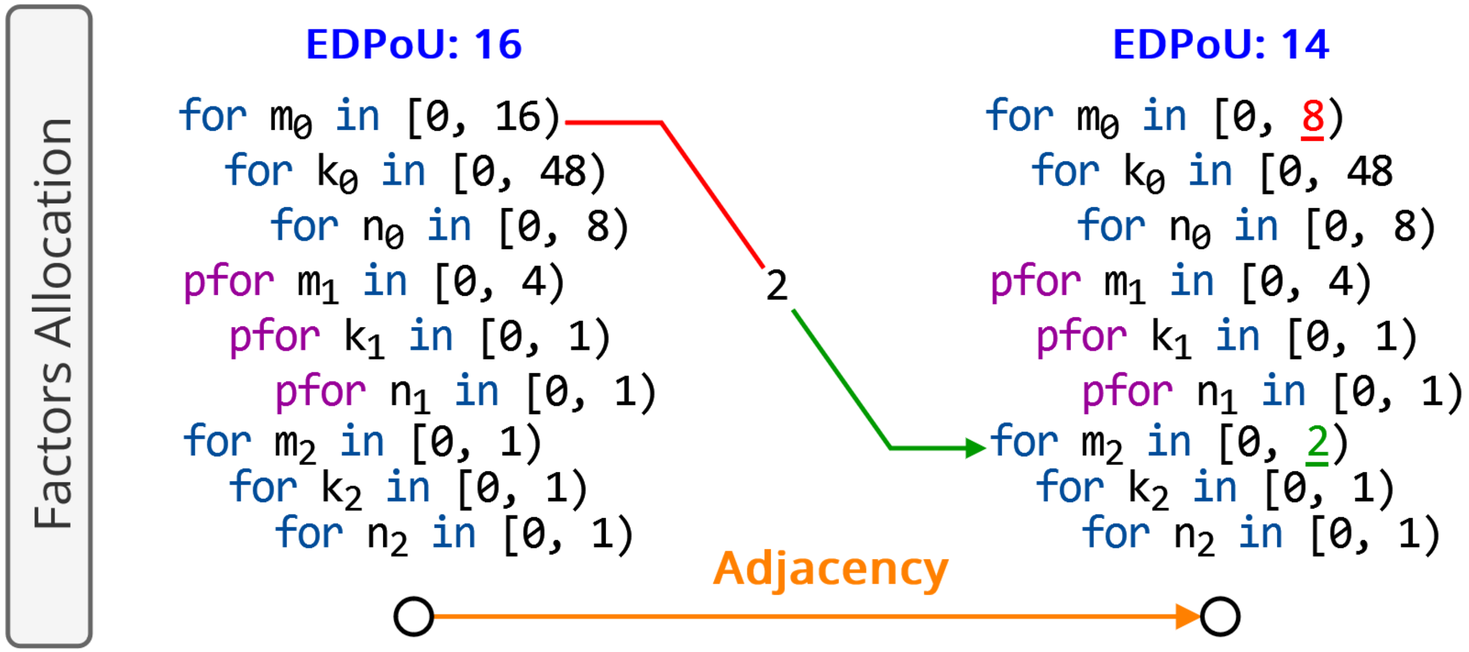
Step 3: Greedy Descent Factors Allocation

Target decision:tiling

Local search between adjacent mappings, reach local optimality.

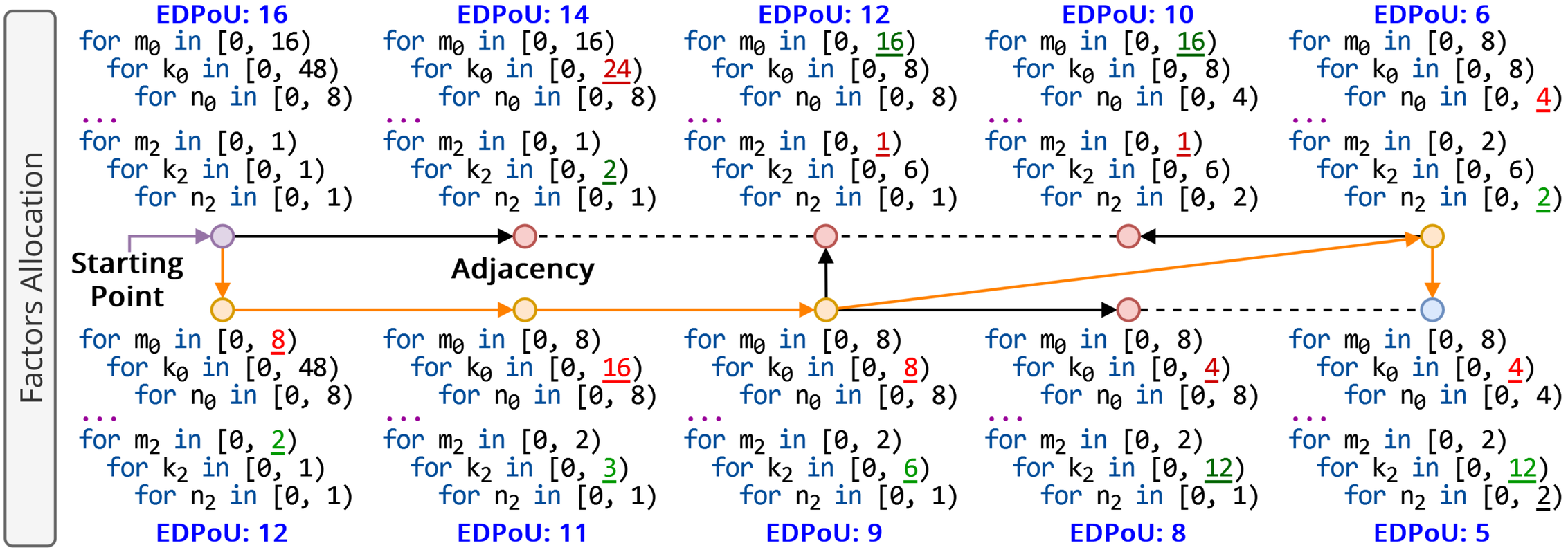
Adjacency: two mappings differing by a single moved prime factor.

Starting point: all unused prime factors on the first level.



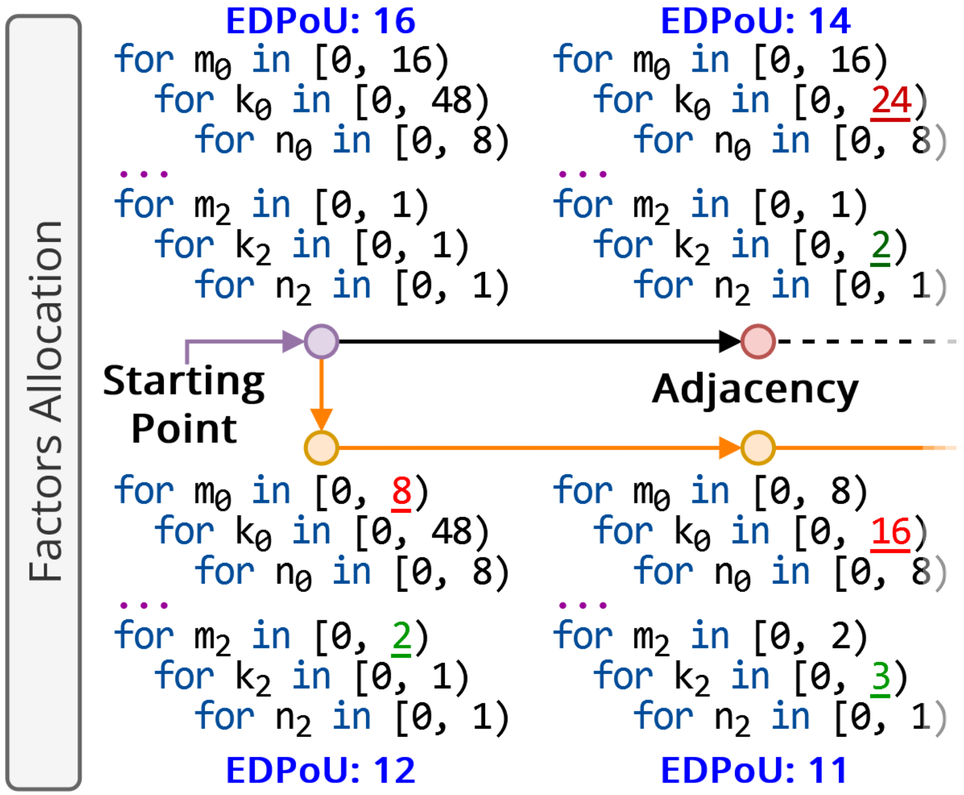
Step 3: Greedy Descent Factors Allocation

Local search between adjacent mappings, reach local optimality.



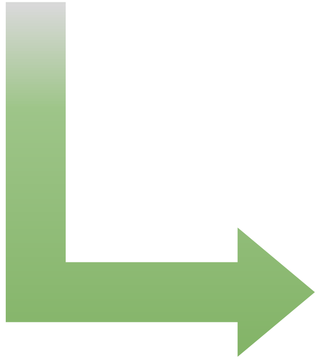
Step 3: Greedy Descent Factors Allocation

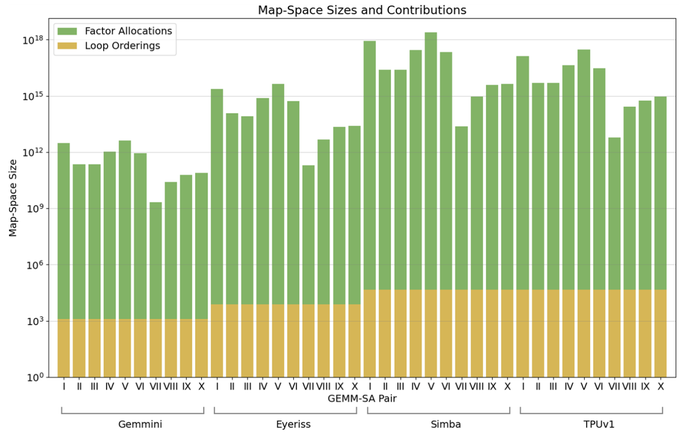
Local search between adjacent mappings, reach local optimality.



Empirically, local optimality often leads to global optimality as well.

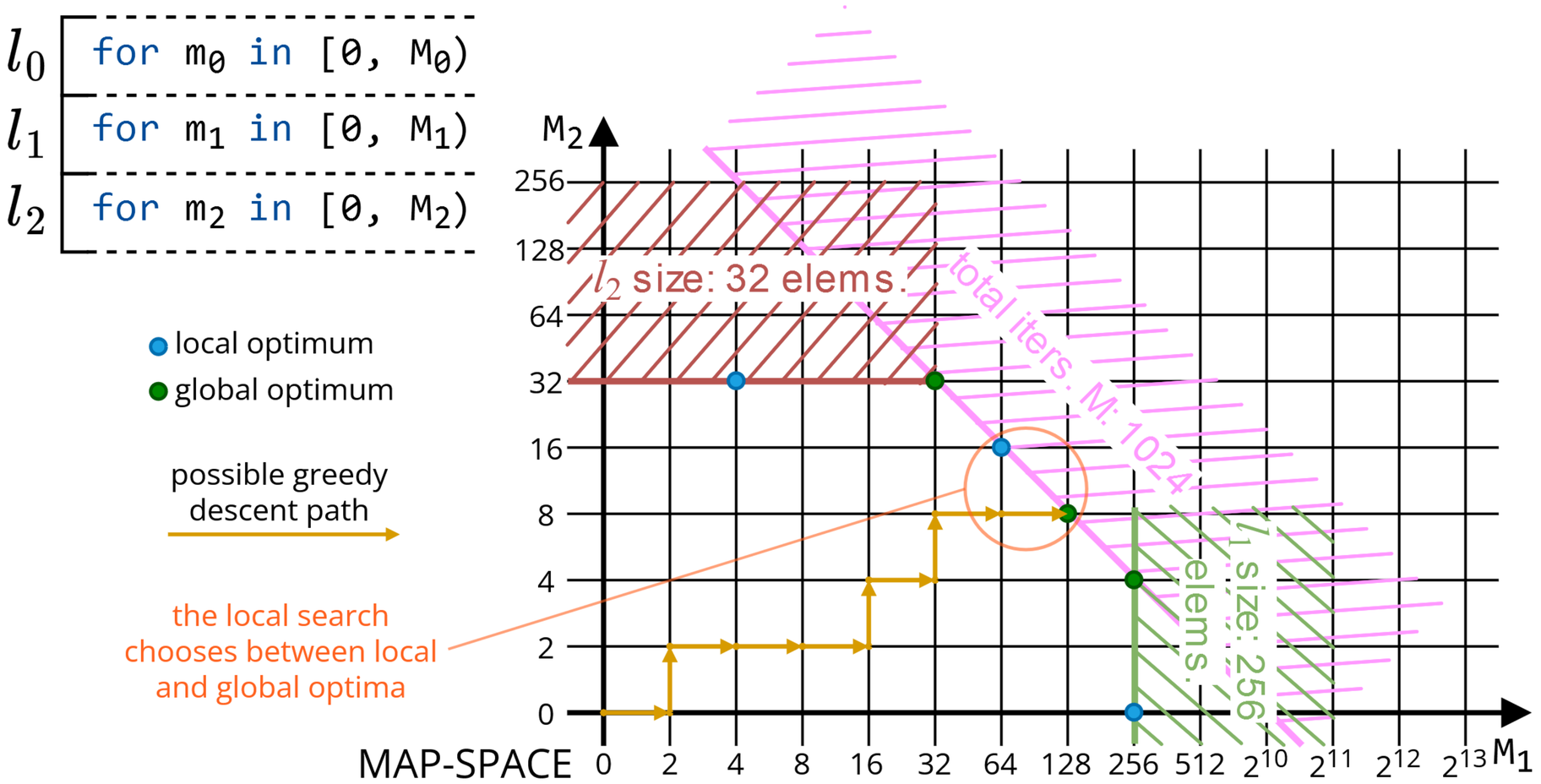
Fast and effective handling of the most complex part of the map-space.





Step 3: Greedy Descent Factors Allocation

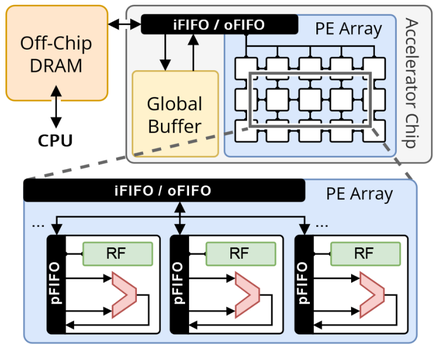
Intuition: where to find optimal mappings.



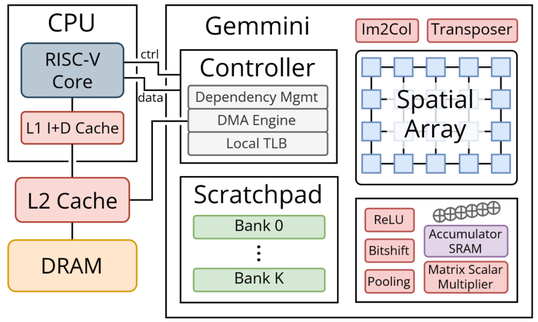
⇒ one lattice point, one mapping. ⇒ remaining iterations are on l_0.

Experimental Setup

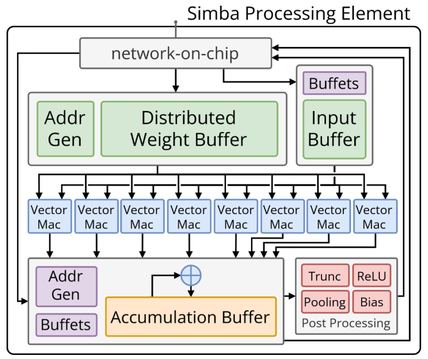
4 SoA Spatial Architectures



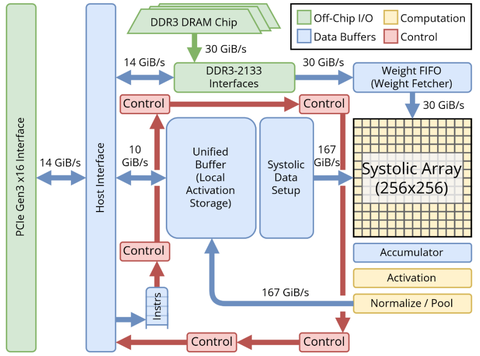
Eyeriss [1]



Gemmini [2]



Simba [3]



TPUv1 [4]

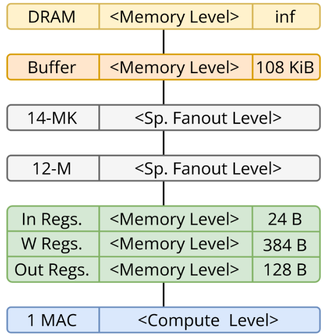
10 GEMMs

GEMM	BERT Transformer [12]				Scientific Applications [7]					
	I	II	III	IV	V	VI	VII	VIII	IX	X
M	3072	4096	64	4096	8192	1024	8	8	8192	512
K	1024	64	4096	1024	8192	8192	8192	1024	1024	256
N	4096	4096	4096	4096	8192	1024	8	8192	8	256

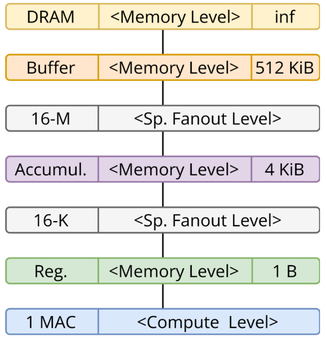
⇒ 40 diverse map-spaces

Experimental Setup

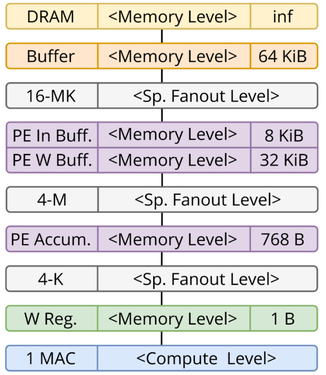
4 SoA Spatial Architectures



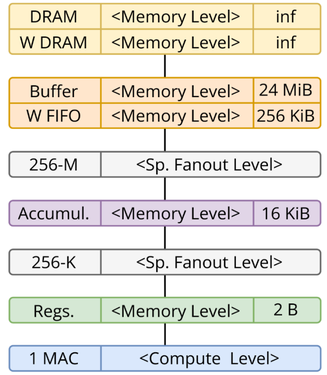
Eyeriss [1]



Gemmini [2]



Simba [3]



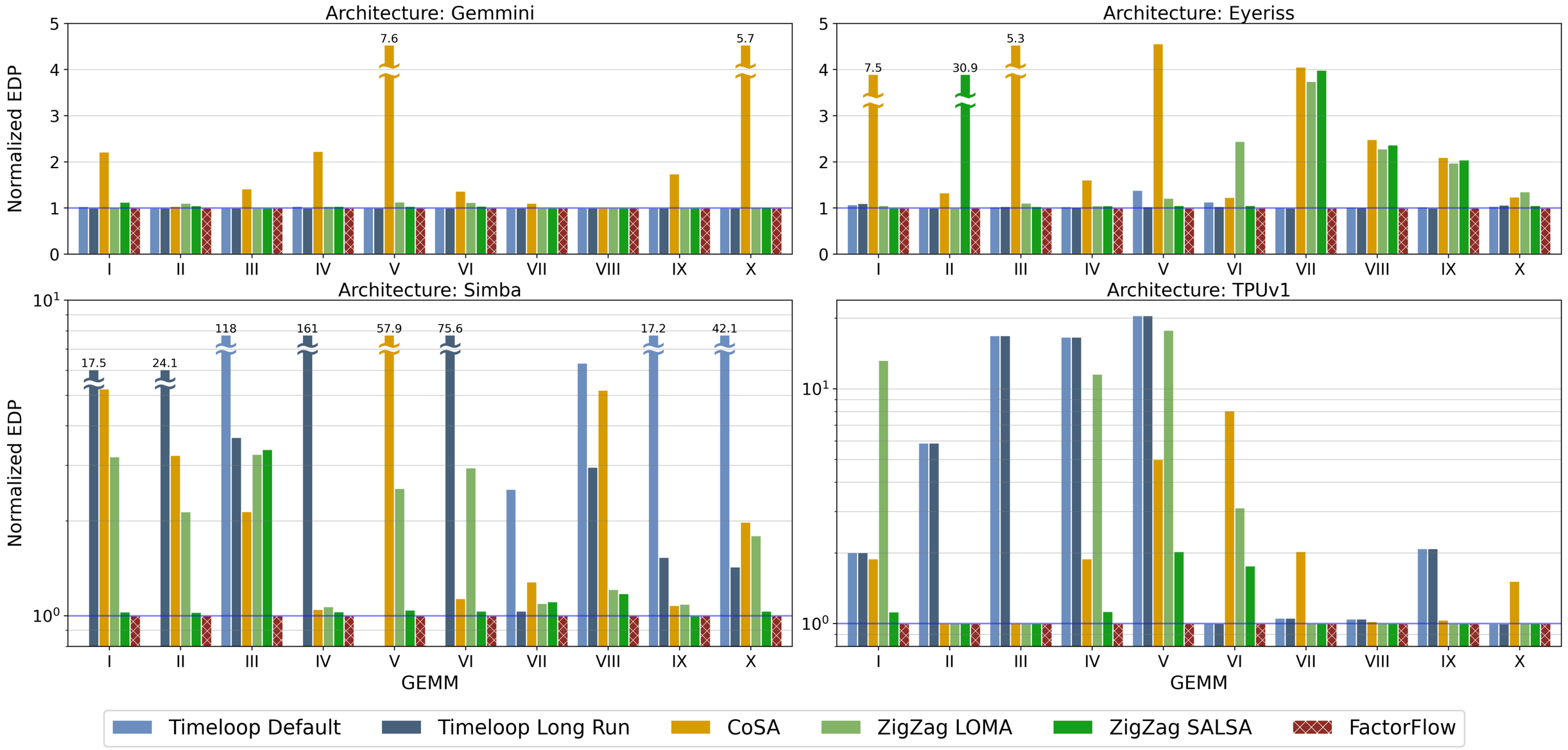
TPUv1 [4]

10 GEMMs

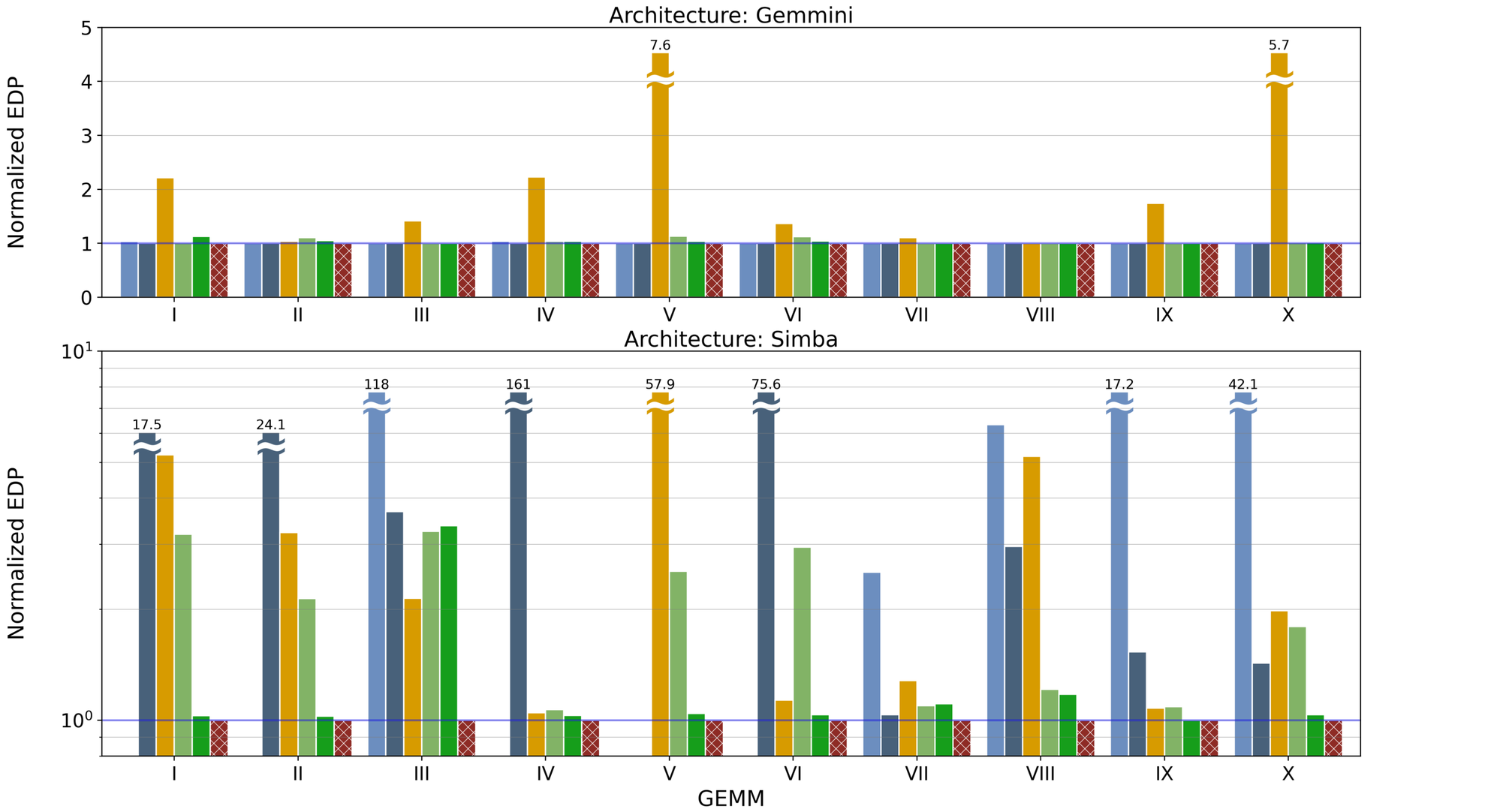
GEMM	BERT Transformer [12]				Scientific Applications [7]					
	I	II	III	IV	V	VI	VII	VIII	IX	X
M	3072	4096	64	4096	8192	1024	8	8	8192	512
K	1024	64	4096	1024	8192	8192	8192	1024	1024	256
N	4096	4096	4096	4096	8192	1024	8	8192	8	256

⇒ 40 diverse map-spaces

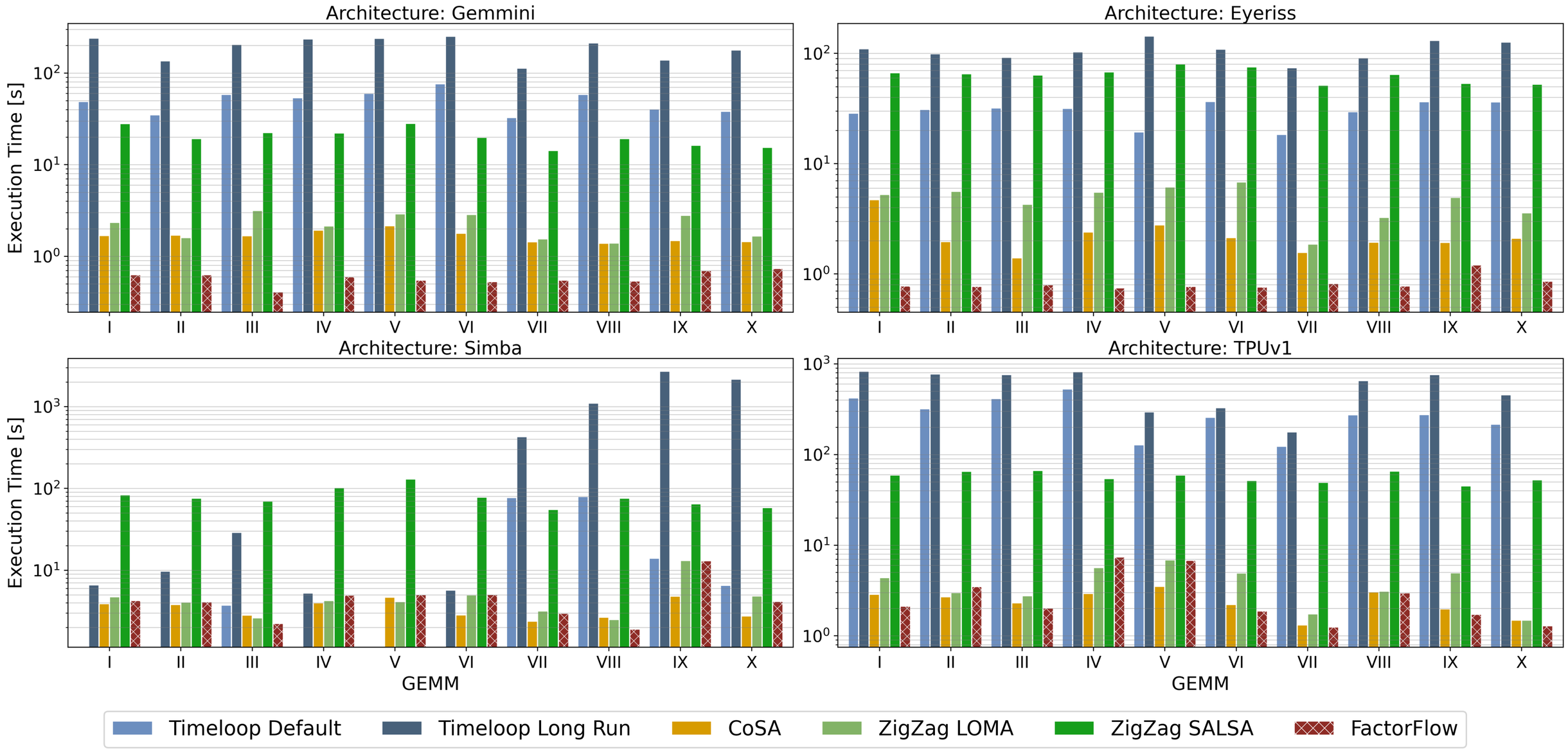
Comparison Results: EDP



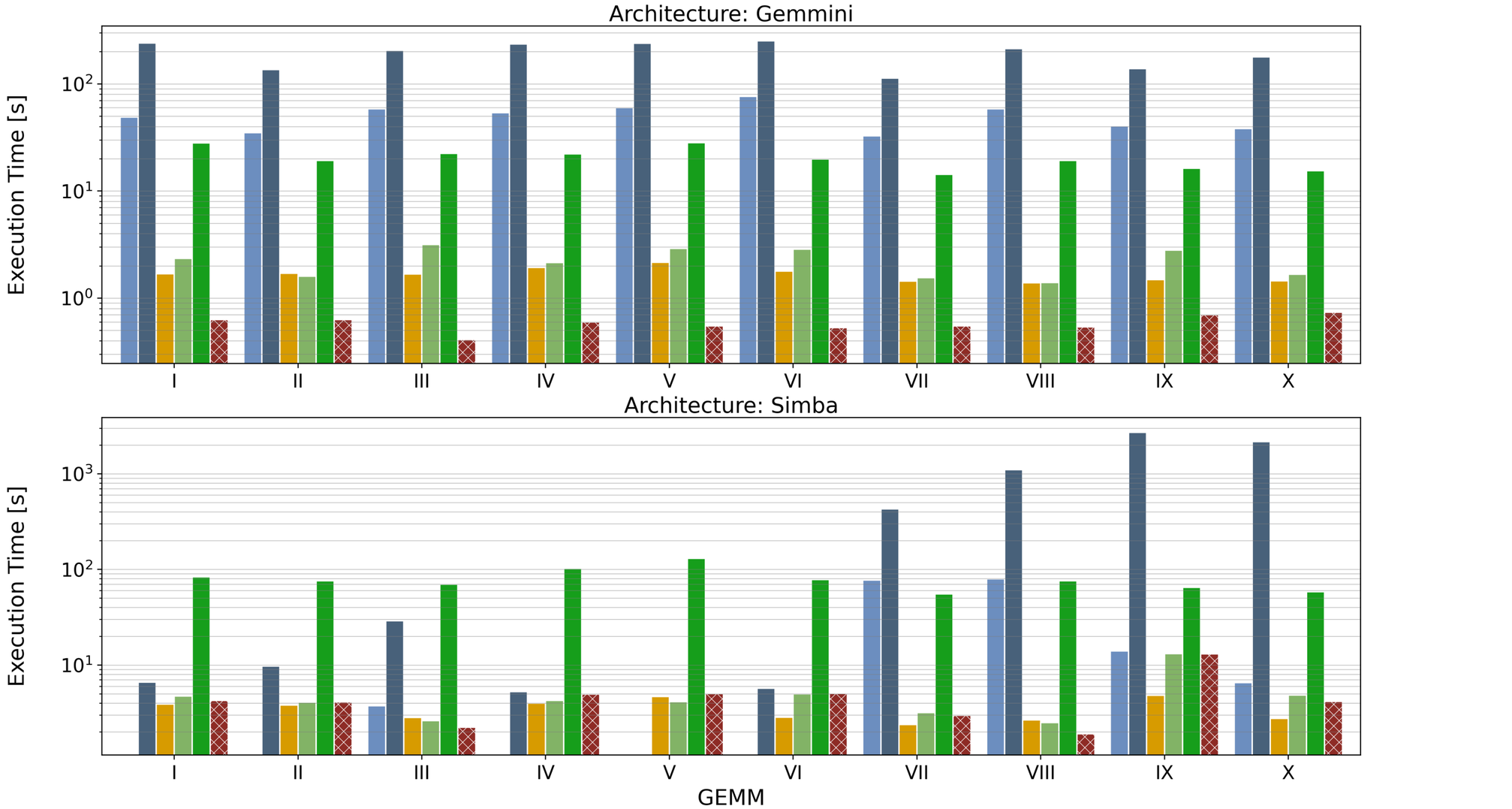
Comparison Results: EDP



Comparison Results: Execution Time



Comparison Results: Execution Time



Comparison Results: Global Optima

FactorFlow EDP difference w.r.t. Global Optima

	I	II	III	IV	V	VI	VII	VIII	IX	X
Gemmini	OPT	OPT	OPT	OPT	OPT	OPT	OPT	OPT	OPT	OPT
Eyeriss	OPT	OPT	OPT	OPT	2.67%	OPT	OPT	OPT	OPT	OPT
Simba	OPT	OPT	OPT	OPT	1.93%	2.28%	OPT	OPT	OPT	OPT
TPUv1	0.94%	OPT	OPT	OPT	OPT	OPT	OPT	OPT	OPT	OPT

⇒ 36/40 global optima found.

Conclusions

FactorFlow **consistently finds equal or better mappings** for GEMMs.

FactorFlow's heuristics are considerably **faster** than previous techniques.

Future Developments

FactorFlow redesign to target **convolutions** (mostly done).

Extension to accelerators based on **in-memory computing.**

FactorFlow is open-source, find it on GitHub:
https://github.com/EMJzero/FactorFlow

References

1. Yu-Hsin Chen, Joel Emer, and Vivienne Sze. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. In 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), pages 367–379, 2016.
2. Hasan Genc, Yakun Sophia Shao, et al. Gemmini: Enabling systematic deep-learning architecture evaluation via full-stack integration. In Proceedings of the 58th Annual Design Automation Conference (DAC), 2021.
3. Y. S. Shao, J. Clemons, et al. Simba: Scaling deep-learning inference with multi-chip-module-based architecture. In Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO '52, page 14–27, New York, NY, USA, 2019.
4. N. P. Jouppi, et al. In-datacenter performance analysis of a tensor processing unit. SIGARCH Comput. Archit. News, 45(2):1–12, jun 2017.
5. Angshuman Parashar, Priyanka Raina, Yakun Sophia Shao, Yu-Hsin Chen, Victor A. Ying, Anurag Mukkara, Rangharajan Venkatesan, Brucek Khailany, Stephen W. Keckler, and Joel Emer. Timeloop: A systematic approach to dnn accelerator evaluation. In 2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pages 304–315, 2019.
6. Sheng-Chun Kao and Tushar Krishna. Gamma: Automating the hw mapping of dnn models on accelerators via genetic algorithm. In 2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD), pages 1–9, 2020.
7. G. E. Moon, H. Kwon, G. Jeong, P. Chatarasi, S. Rajamanickam, and T. Krishna. Evaluating spatial accelerator architectures with tiled matrix-matrix multiplication, 2021.
8. A. Symons, L. Mei, and M. Verhelst. Loma: Fast auto-scheduling on dnn accelerators through loop-order-based memory allocation. In 2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS), pages 1–4, 2021.
9. V. J. Jung, A. Symons, L. Mei, M. Verhelst, and L. Benini. Salsa: Simulated annealing based loop-ordering scheduler for dnn accelerators. In 2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS), pages 1–5, 2023.
10. Qijing Huang, Minwoo Kang, Grace Dinh, Thomas Norell, Aravind Kalaiah, James Demmel, John Wawrzynek, and Yakun Sophia Shao. Cosa: Scheduling by constrained optimization for spatial accelerators. In 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA), pages 554–566, 2021.
11. N. P. Jouppi, D. H. Yoon, M. Ashcraft, M. Gottscho, T. B. Jablin, G. Kurian, J. Laudon, S. Li, P. Ma, X. Ma, T. Norrie, N. Patil, S. Prasad, C. Young, Z. Zhou, and D. Patterson. Ten lessons from three generations shaped google's tpuv4i. In Proceedings of the 48th Annual International Symposium on Computer Architecture, ISCA '21, page 1–14. IEEE Press, 2021.
12. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.
13. Sehoon Kim, Coleman Hooper, Thanakul Wattanawong, Minwoo Kang, Ruohan Yan, Hasan Genc, Grace Dinh, Qijing Huang, Kurt Keutzer, Michael W. Mahoney, Yakun Sophia Shao, and Amir Gholami. Full stack optimization of transformer inference: a survey, 2023.
14. Cristina Silvano, et al. A survey on deep learning hardware accelerators for heterogeneous hpc platforms, 2023.
15. Fabrizio Ferrandi, at al. A survey on design methodologies for accelerating deep learning on heterogeneous architectures, 2023.

Thank You for Your Attention!