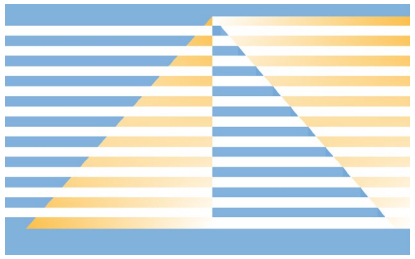
PID-Comm:

Fast and Flexible Collective Communication Framework for Commodity Processing-in-DIMM Devices

Si Ung Noh[1], **Junguk Hong**[1], Chaemin Lim[2], Seongyeon Park[1], Jeehyun Kim[2], Hanjun Kim[3], Youngsok Kim[2] and Jinho Lee[1]

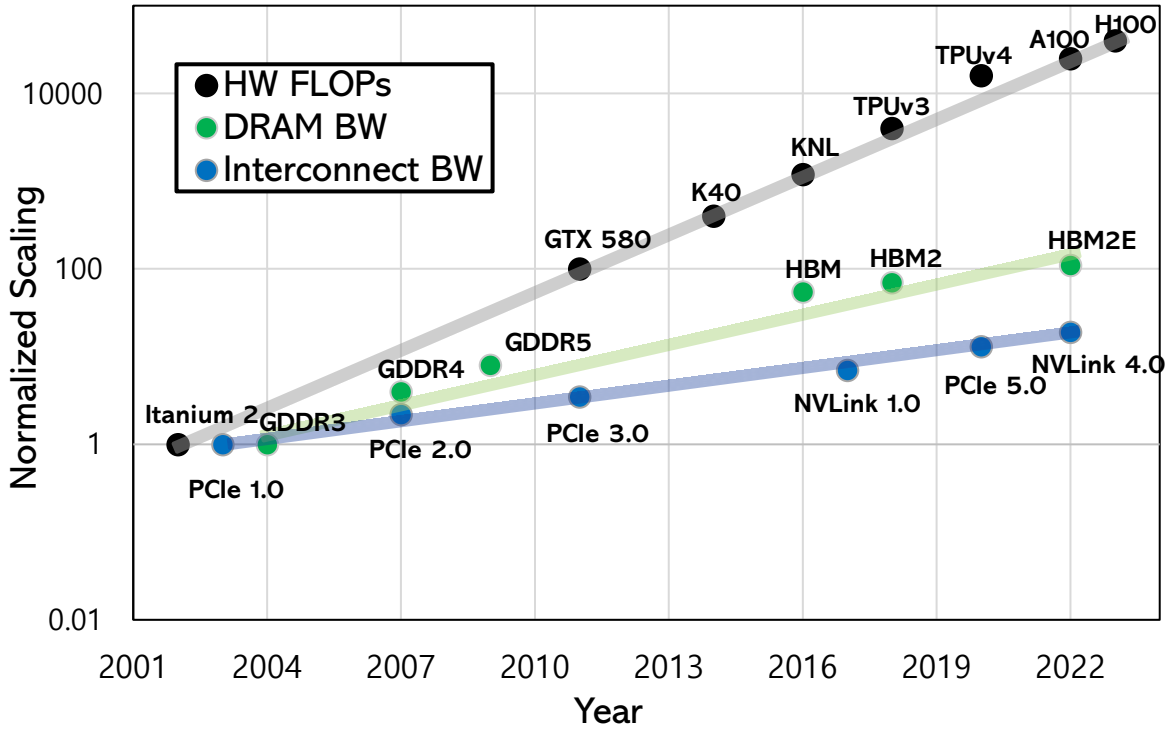
[1]: Department of Electrical and Computer Engineering, Seoul National University
[2]: Department of Computer Science, Yonsei University
[3]: School of Electrical and Electronic Engineering, Yonsei University



*The 51st ACM International Symposium on Computer Architecture (**ISCA**)*

The Memory Wall

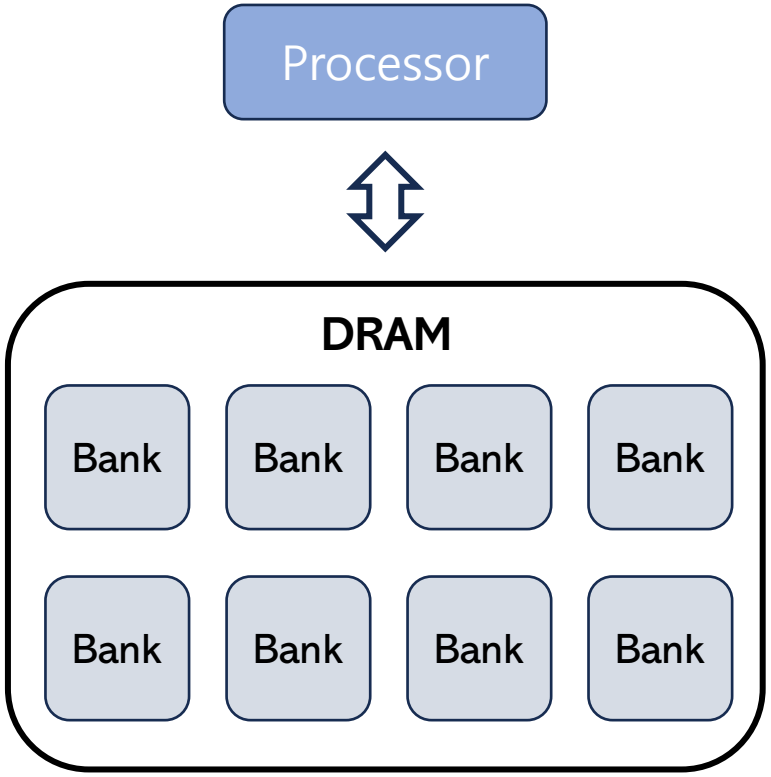
- Processor performance is outpacing memory interconnect bandwidth
- AI applications require high memory bandwidth
- Memory is becoming the dominant bottleneck



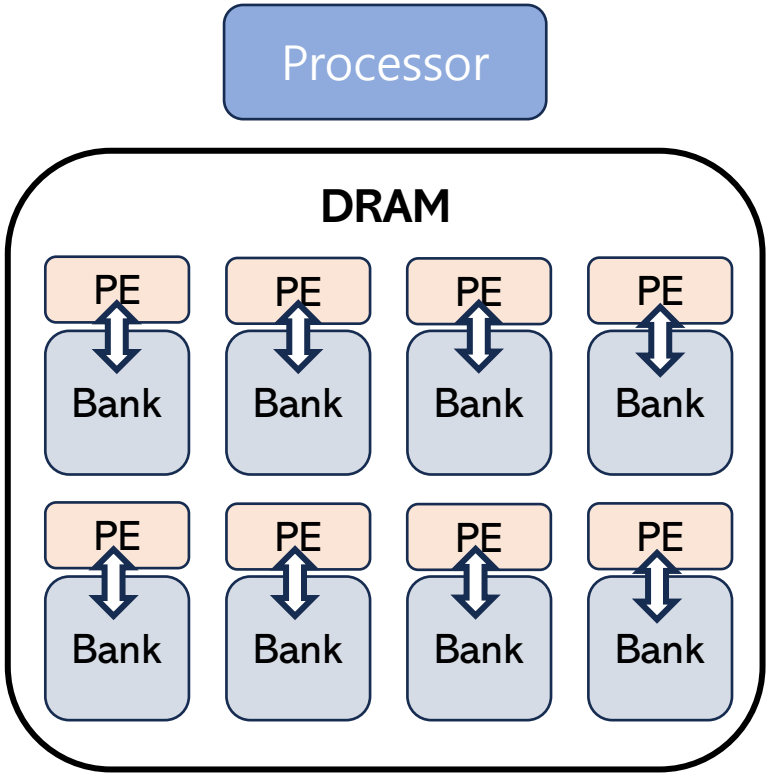
[+]Source: Amir Gholami, Zhewei Yao, Sehoon Kim, Coleman Hooper, Michael W. Mahoney, Kurt Keutzer. 2024 AI and Memory Wall. In IEEE Micro

Processing-in-Memory (PIM)

- PIM is a promising solution to the memory bottleneck
- Achieves higher memory bandwidth by offloading computation to memory



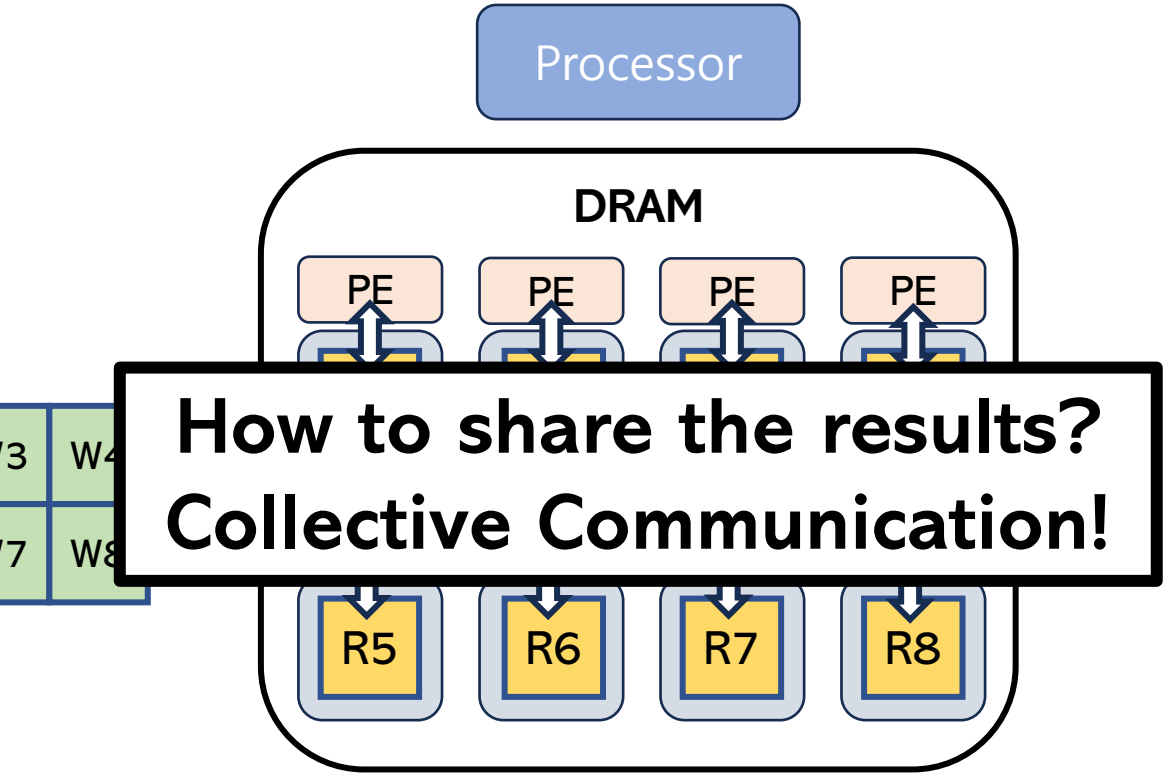
Conventional Systems



PIM-enabled Systems
(e.g., UPMEM DIMMs)

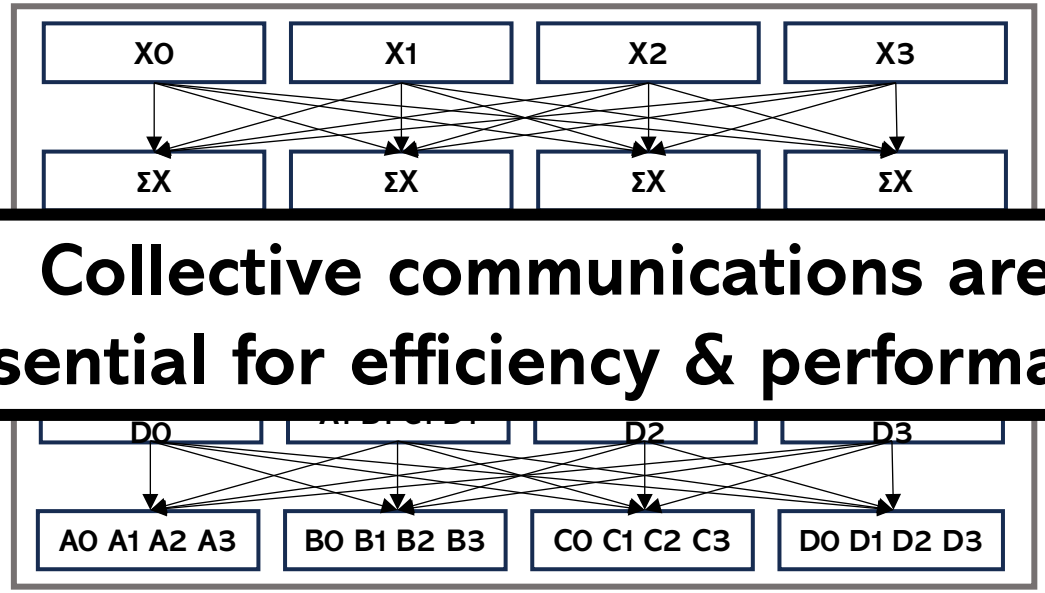
Inter-PE Communications

- Workload is distributed to each PEs (nodes) for computation
- For efficient sharing of intermediate results, collective communications are essential



PIM-enabled Systems
(e.g., UPMEM DIMMs)

How to share the results?
Collective Communication!



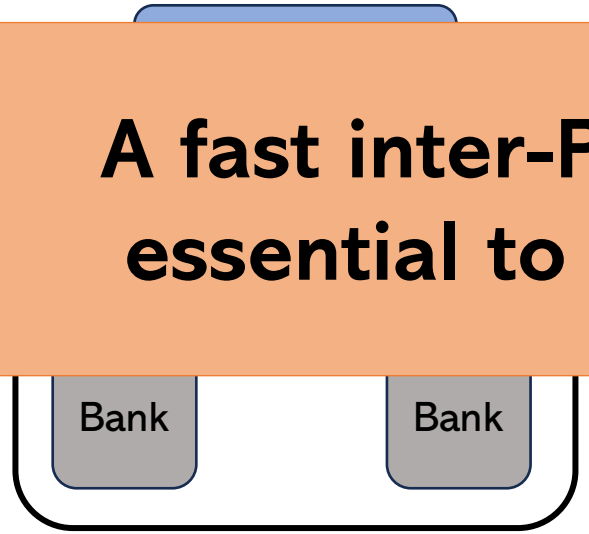
AlltoAll (AA)

Collective communications are essential for efficiency & performance

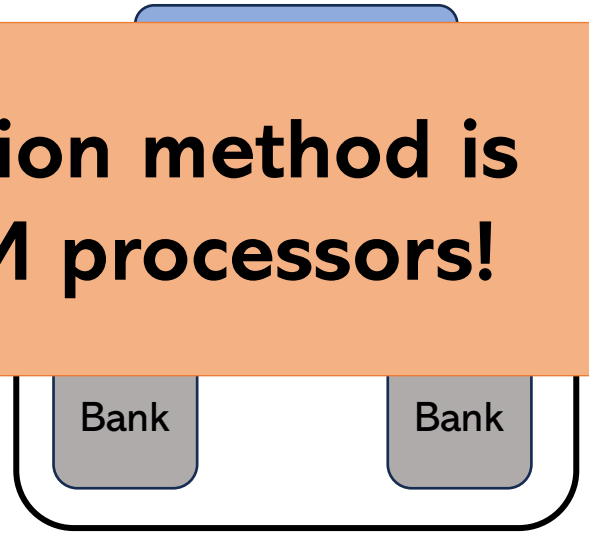
Inter-PE Communications

- No direct path between each DRAM processing elements (PEs)
- Host processor becomes the medium for inter-PE communication
- Inter-PE communications are becoming the bottleneck of major applications

A fast inter-PE communication method is essential to well-utilize PIM processors!



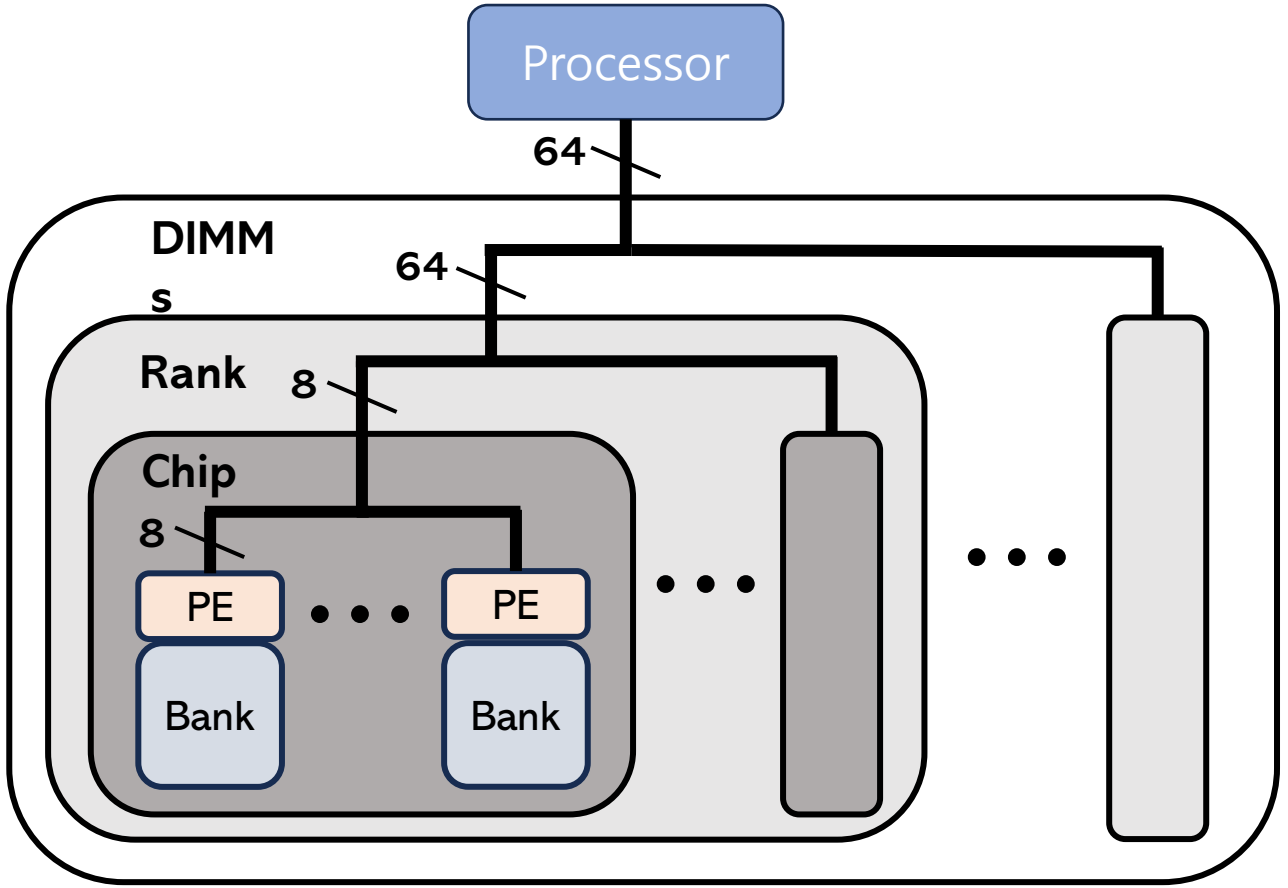
Lack of a direct path between PEs



Existing path passes the host processor

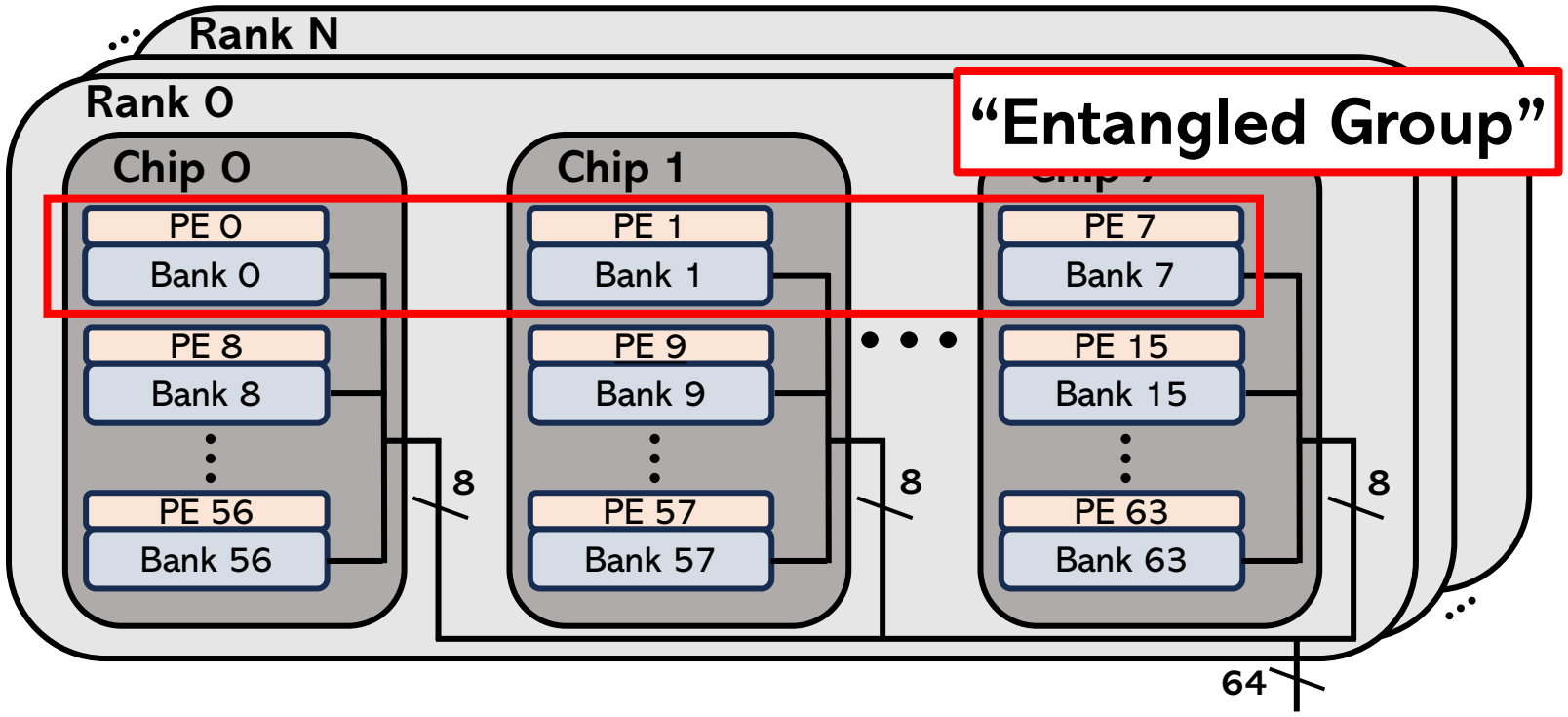
PIM-enabled DIMMs

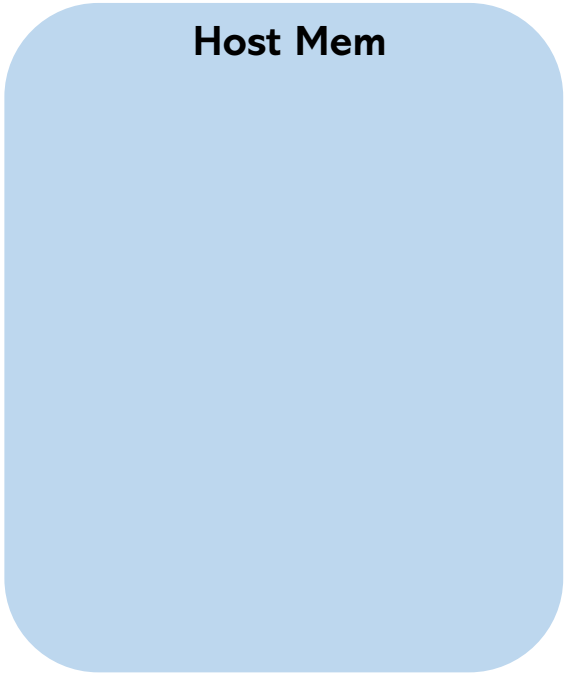
- Refers to PIM implemented on dual in-line memory modules (UPMEM DIMMs)
- Shares the same hierarchy as DDR4 DRAMs



PIM-enabled DIMMs

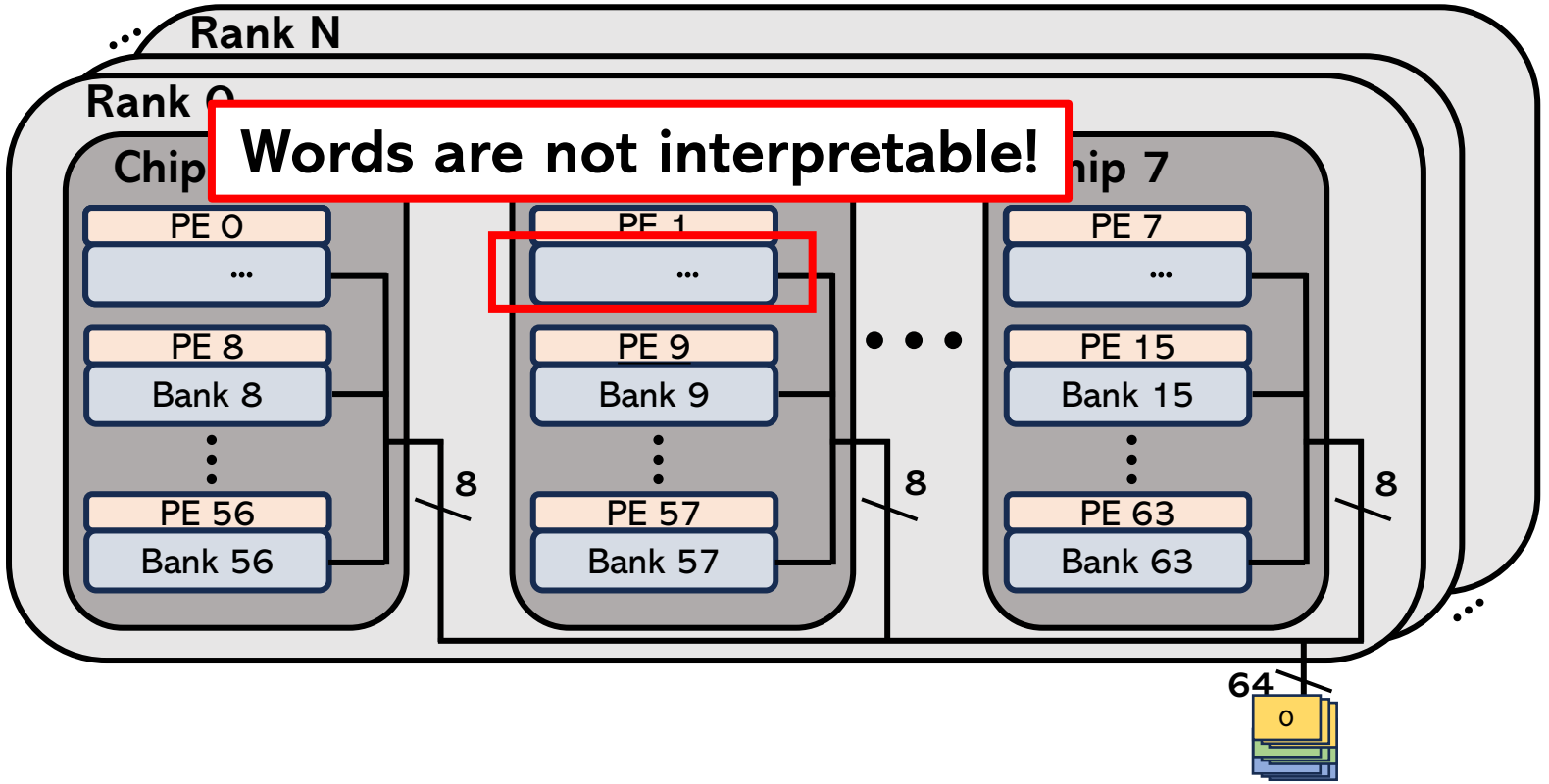
- 8-bit bus per chip to form 64-bit channel bus
- The same bank of each chip in a rank are accessed at once
- We name the group of banks accessed together as “Entangled Group”

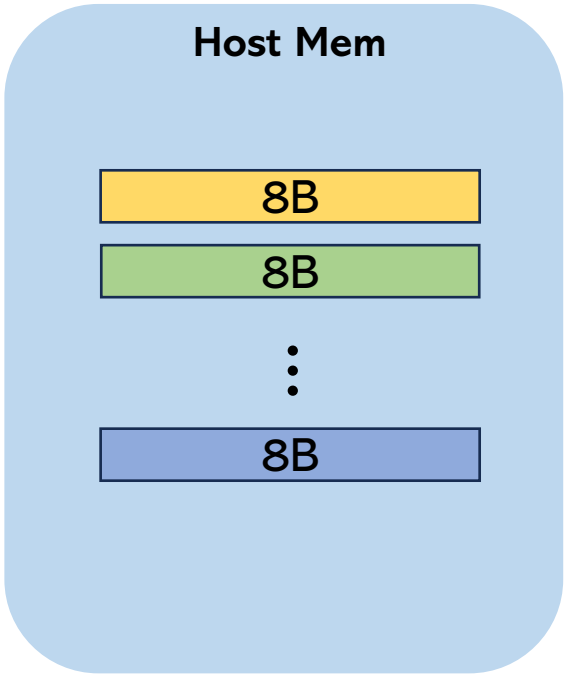




PIM-enabled DIMMs

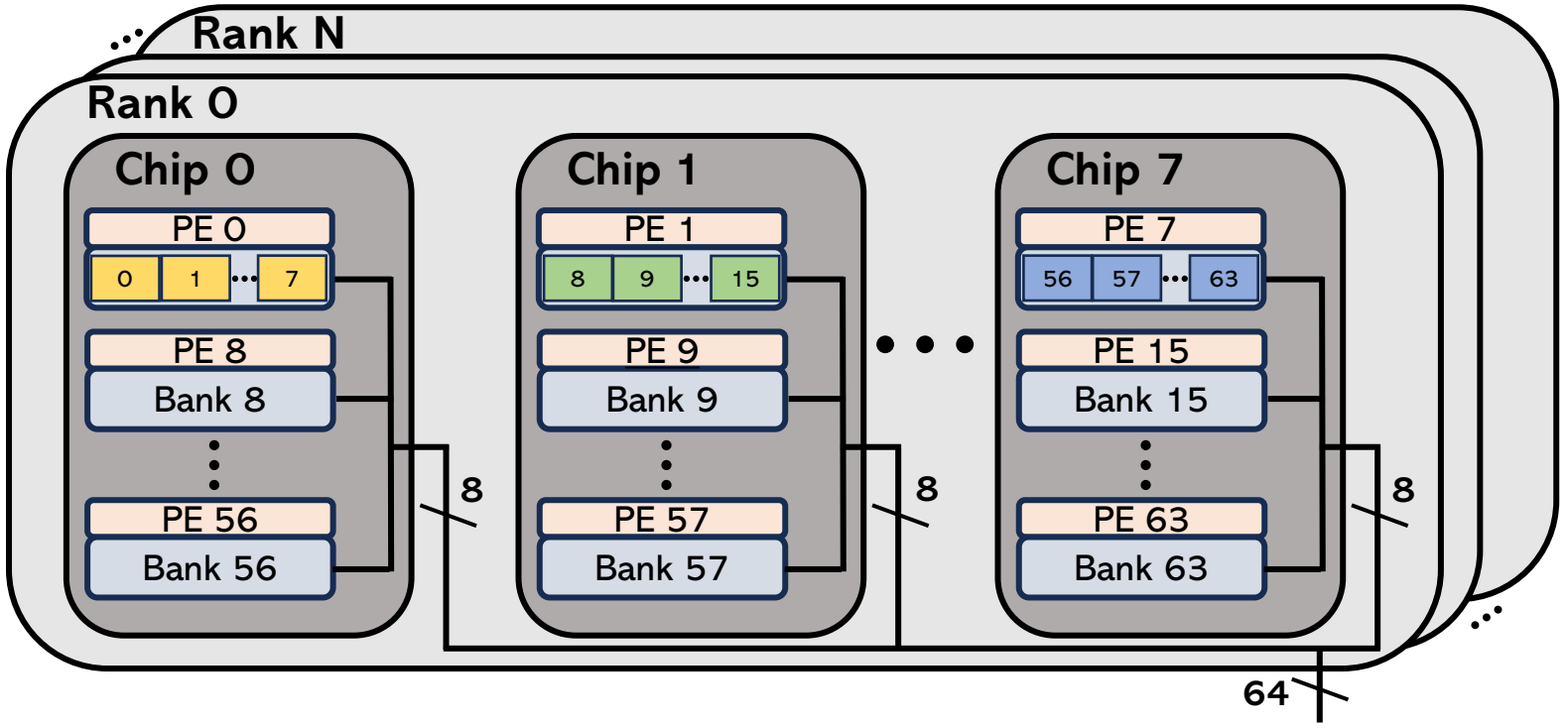
- 8-bit bus per chip to form 64-bit channel bus
- The same bank of each chip in a rank are accessed at once
- We name the group of banks accessed together as “Entangled Group”

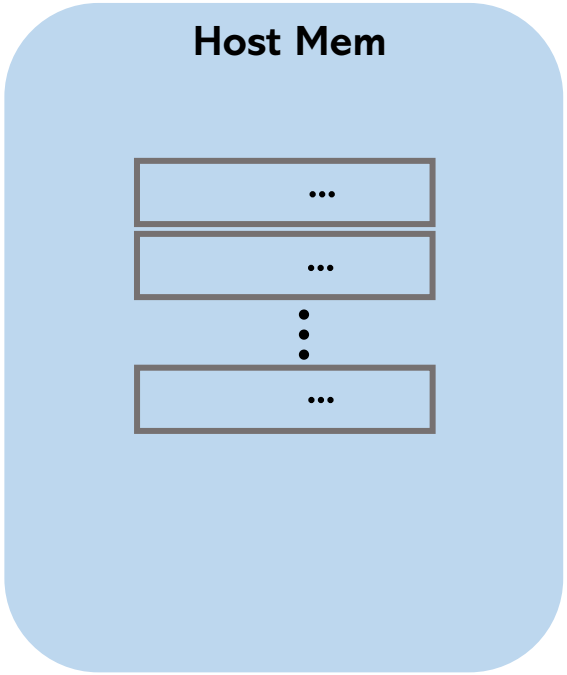




PIM-enabled DIMMs

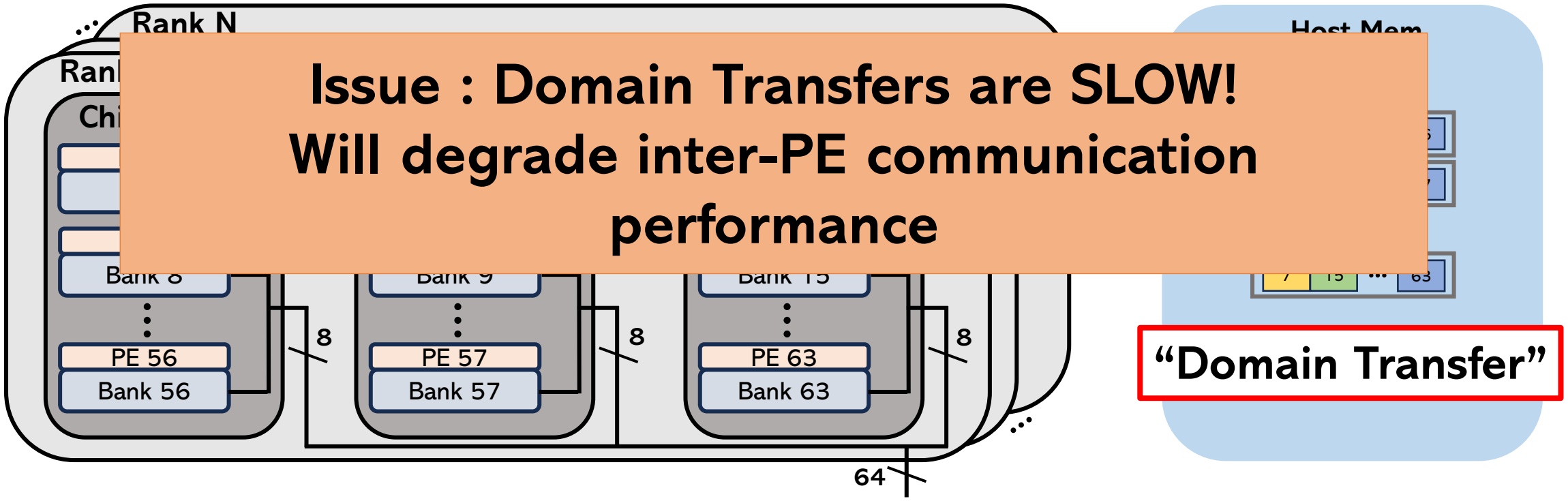
- 8-bit bus per chip to form 64-bit channel bus
- The same bank of each chip in a rank are accessed at once
- We name the group of banks accessed together as “Entangled Group”





PIM-enabled DIMMs

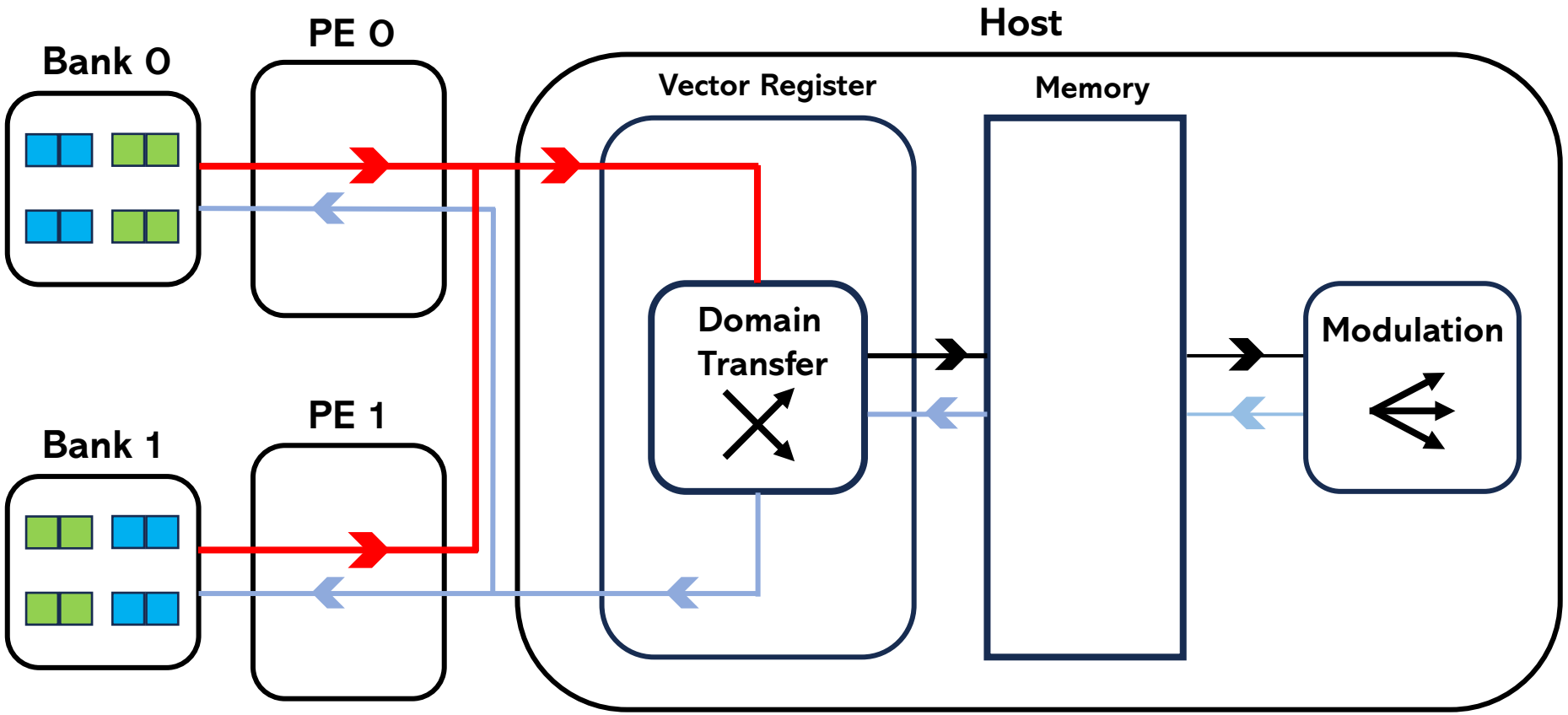
- 8-bit bus per chip to form 64-bit channel bus
- The same bank of each chip in a rank are accessed at once
- We name the group of banks accessed together as “Entangled Group”



Issue : Domain Transfers are SLOW!
Will degrade inter-PE communication performance

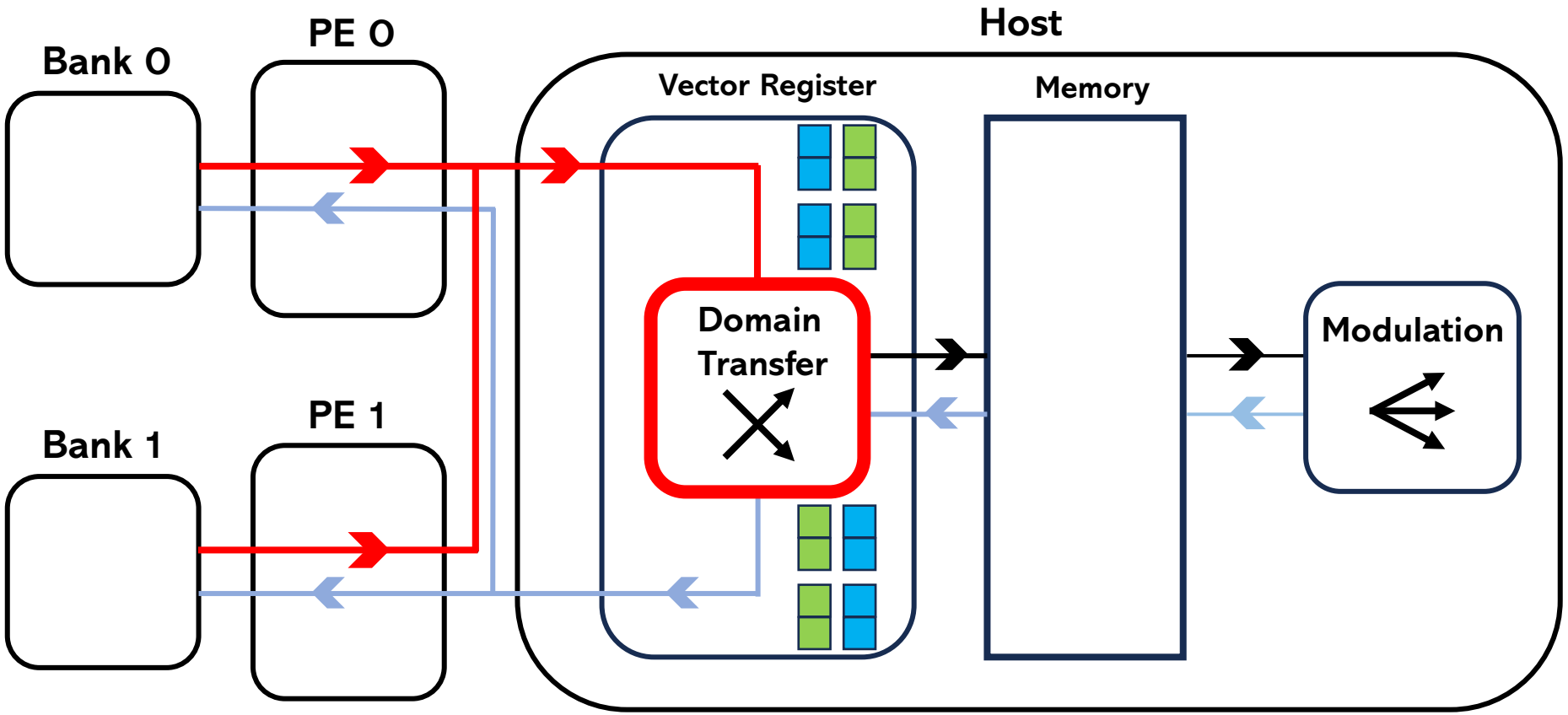
Conventional Inter-PE Communication

- Data domain-transferred in the vector register and saved in host memory
- Data sent back to bank after modulation in the host processor



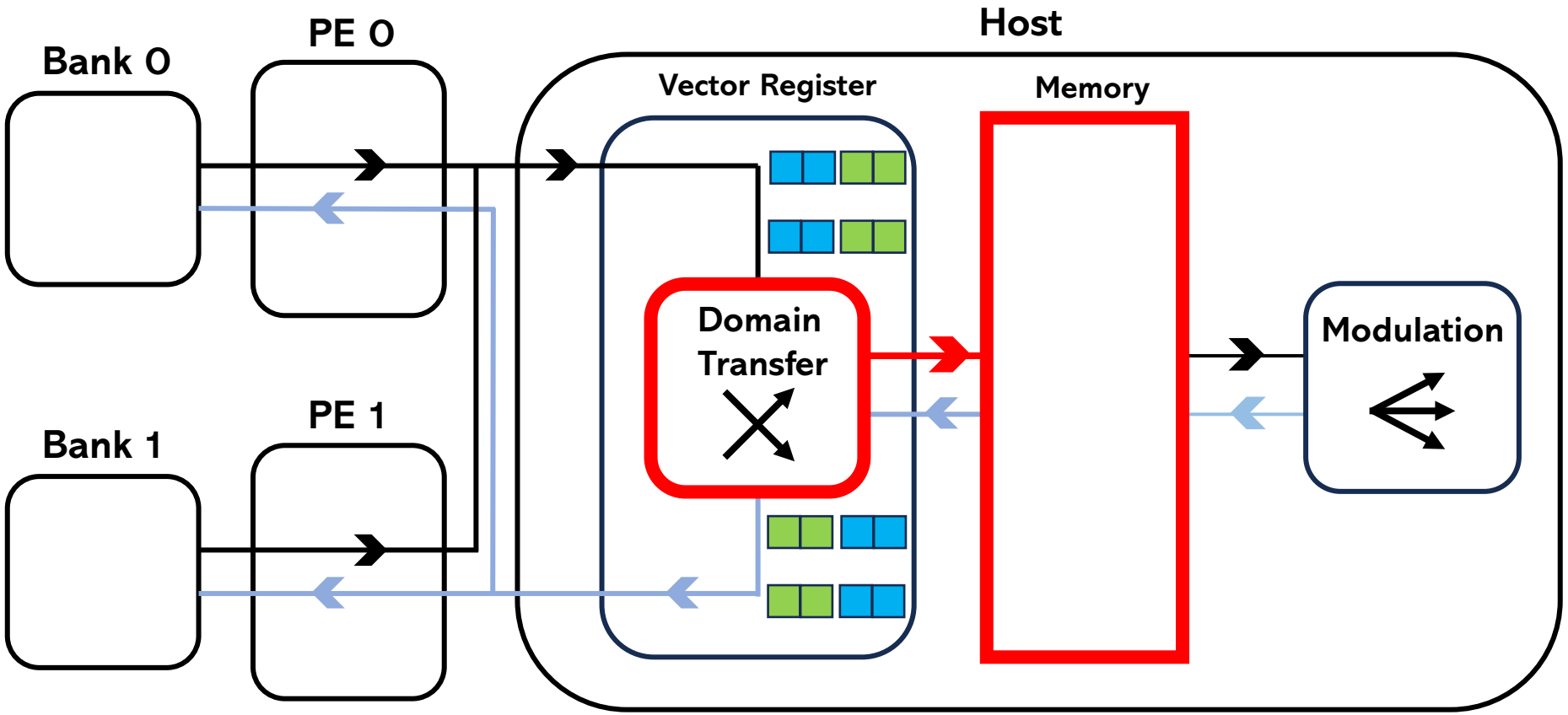
Conventional Inter-PE Communication

- Data domain-transferred in the vector register and saved in host memory
- Data sent back to bank after modulation in the host processor



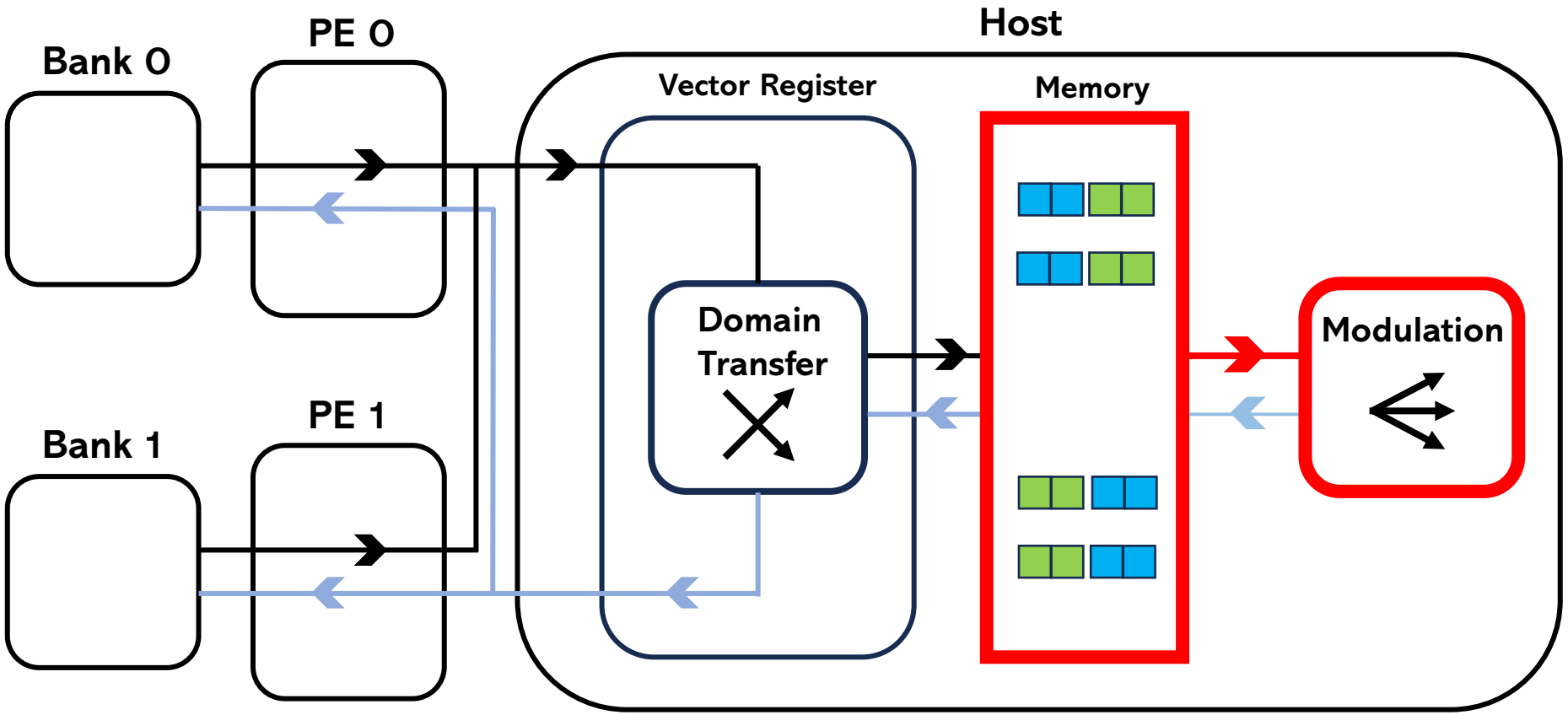
Conventional Inter-PE Communication

- Data domain-transferred in the vector register and saved in host memory
- Data sent back to bank after modulation in the host processor



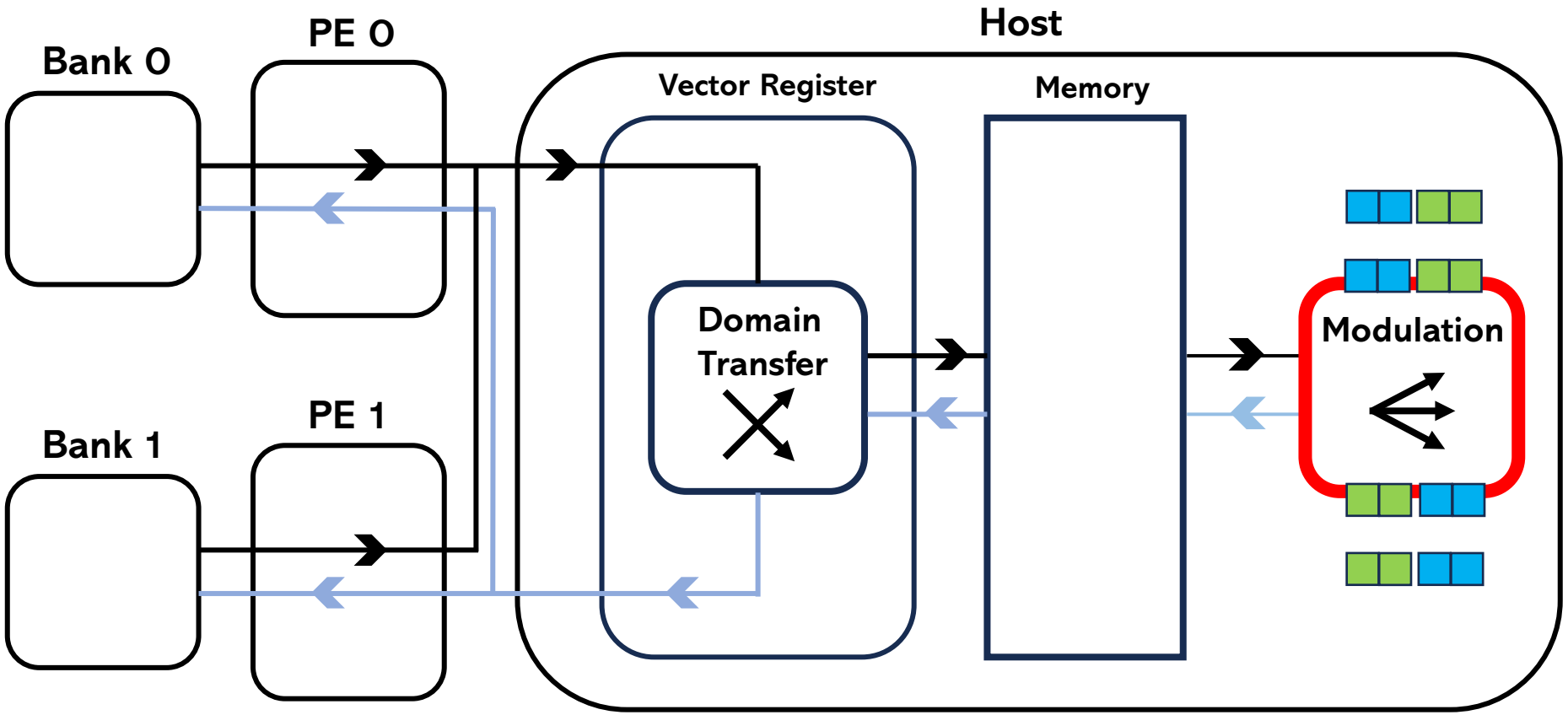
Conventional Inter-PE Communication

- Data domain-transferred in the vector register and saved in host memory
- Data sent back to bank after modulation in the host processor



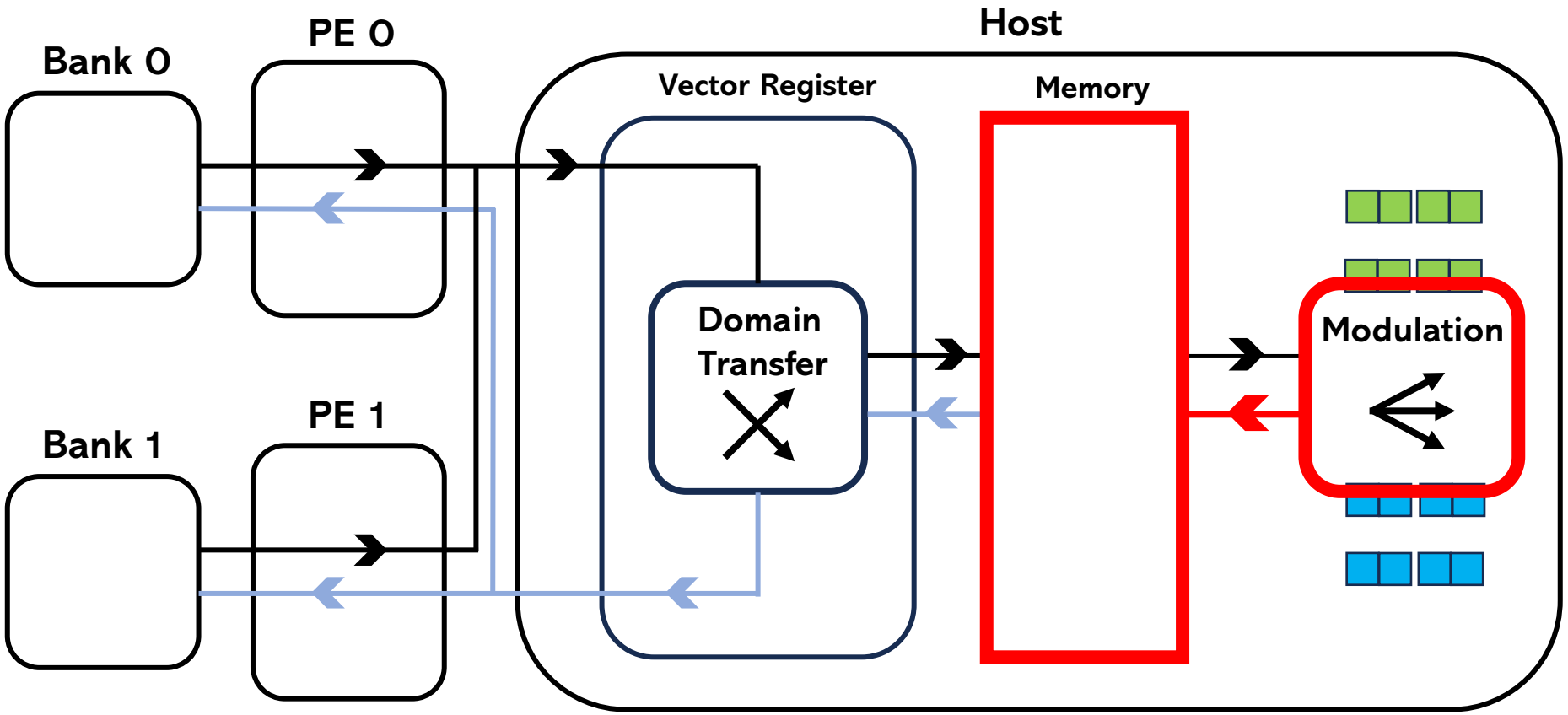
Conventional Inter-PE Communication

- Data domain-transferred in the vector register and saved in host memory
- Data sent back to bank after modulation in the host processor



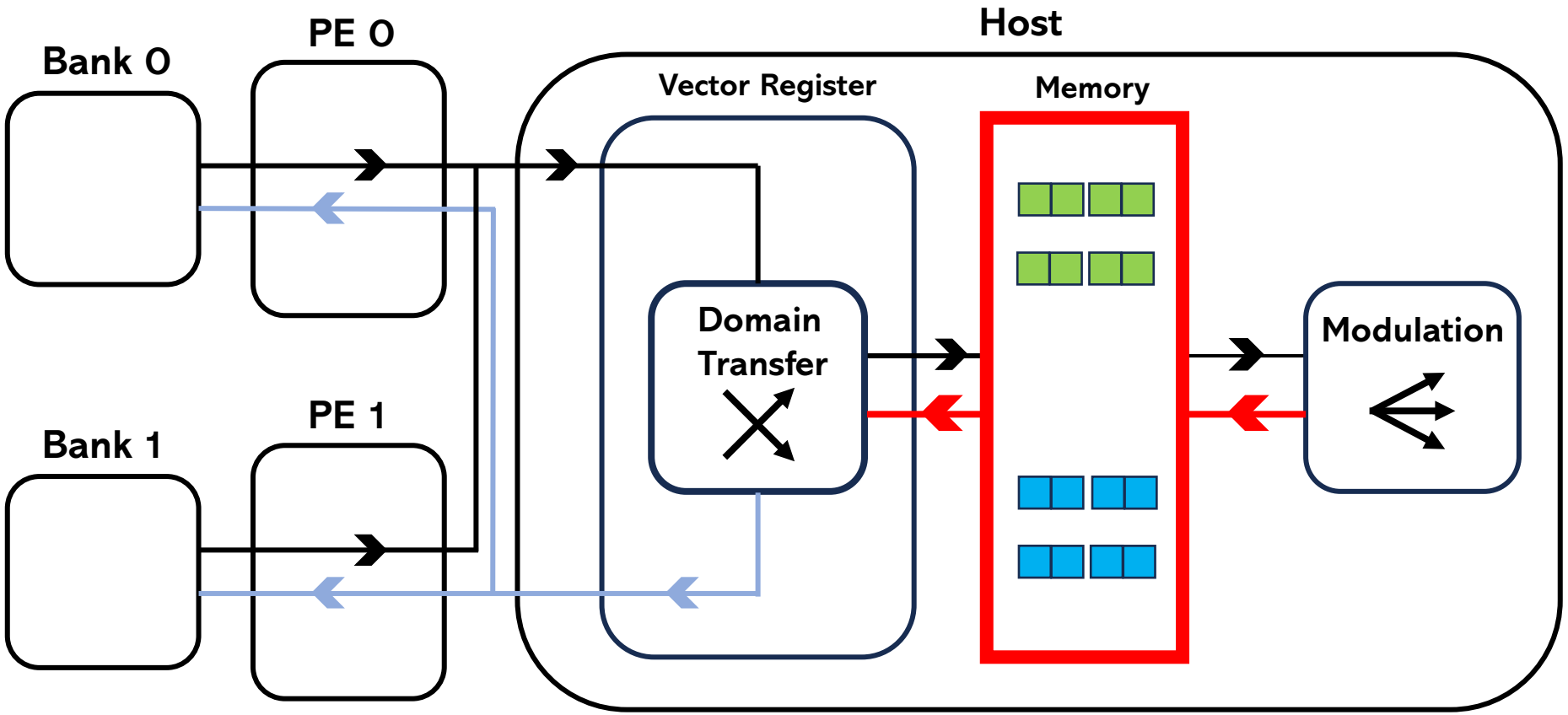
Conventional Inter-PE Communication

- Data domain-transferred in the vector register and saved in host memory
- Data sent back to bank after modulation in the host processor



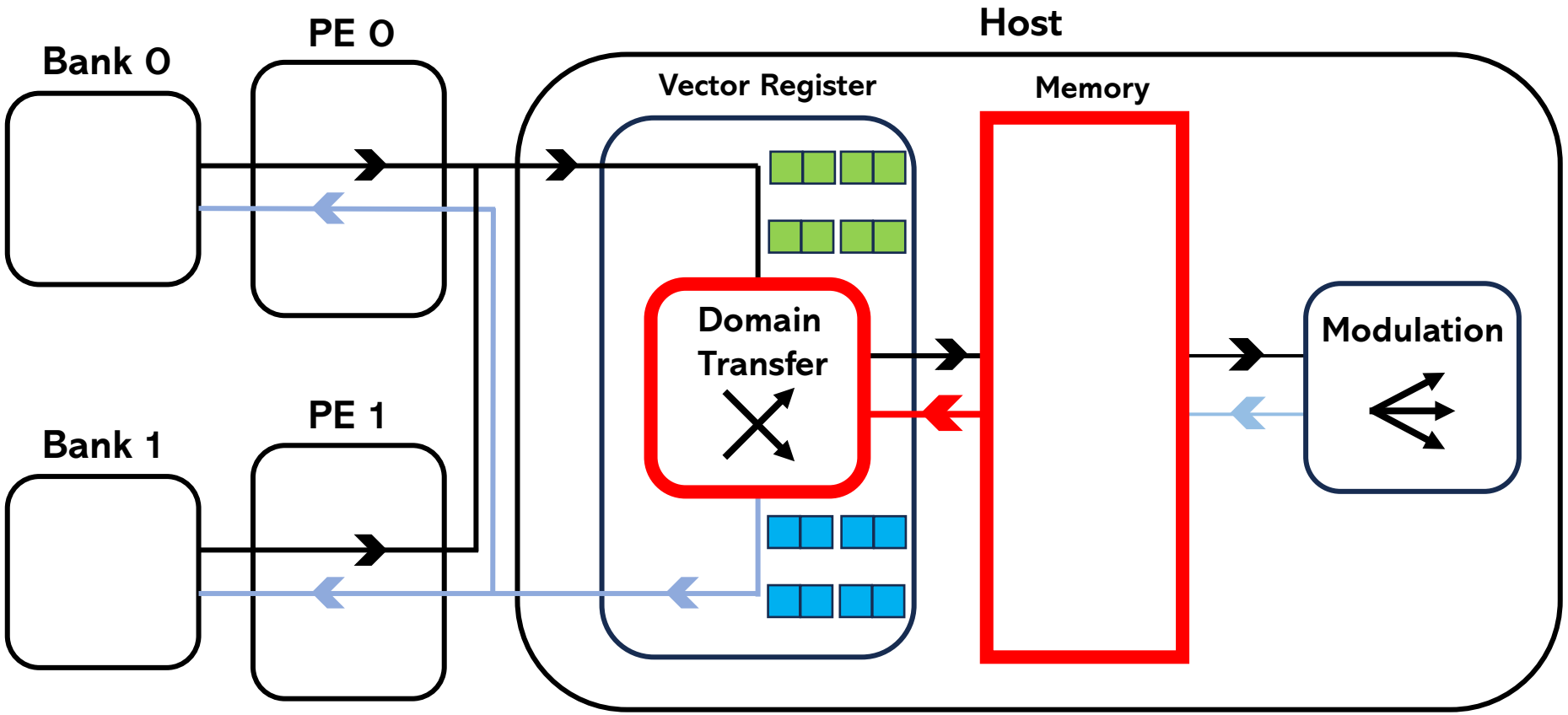
Conventional Inter-PE Communication

- Data domain-transferred in the vector register and saved in host memory
- Data sent back to bank after modulation in the host processor



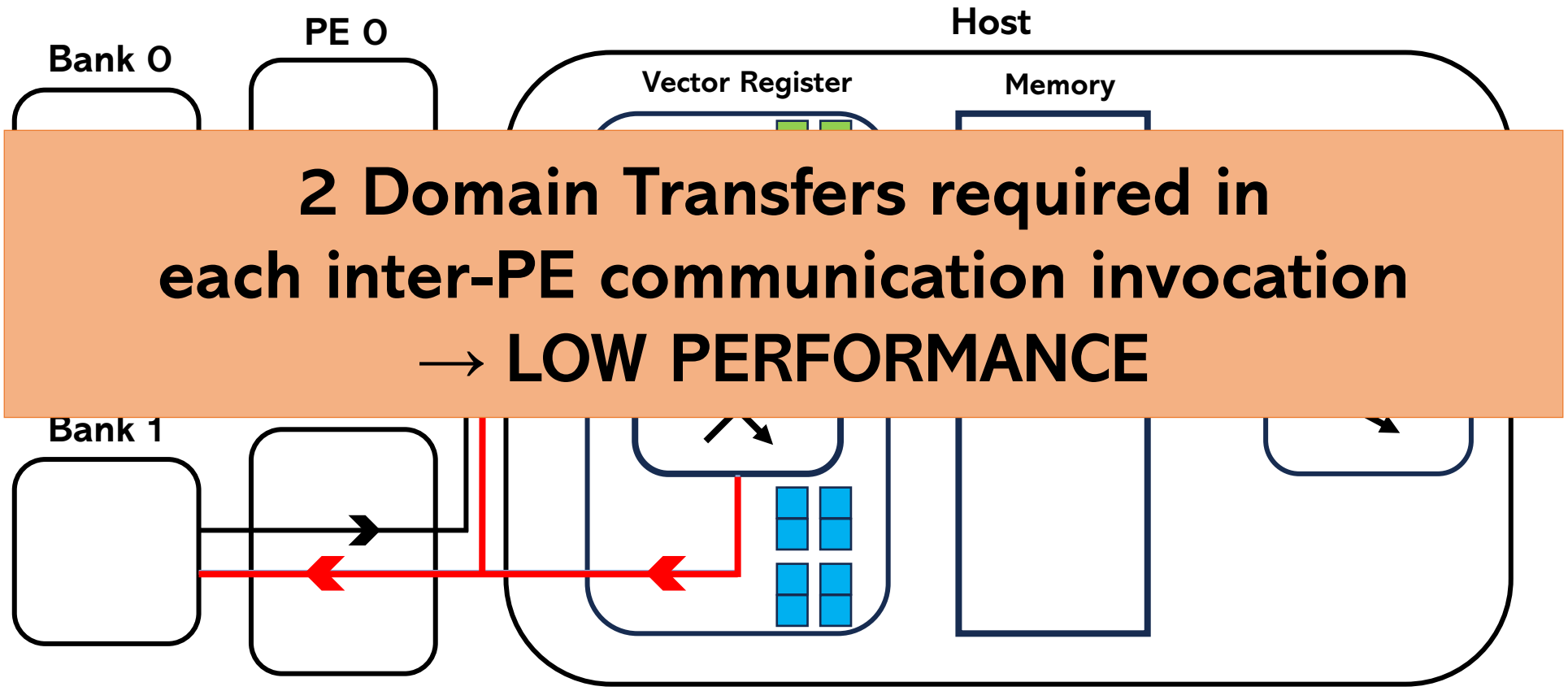
Conventional Inter-PE Communication

- Data domain-transferred in the vector register and saved in host memory
- Data sent back to bank after modulation in the host processor



Conventional Inter-PE Communication

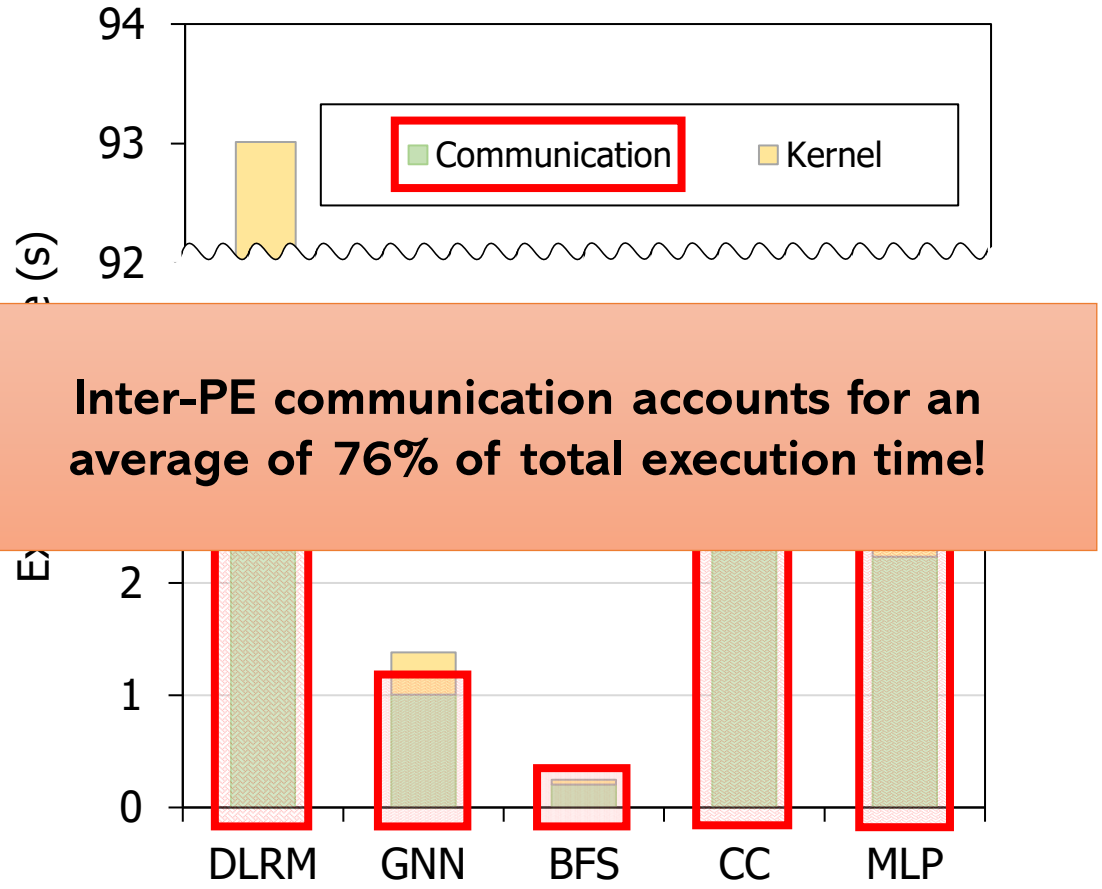
- Data domain-transferred in the vector register and saved in host memory
- Data sent back to bank after modulation in the host processor



2 Domain Transfers required in each inter-PE communication invocation → LOW PERFORMANCE

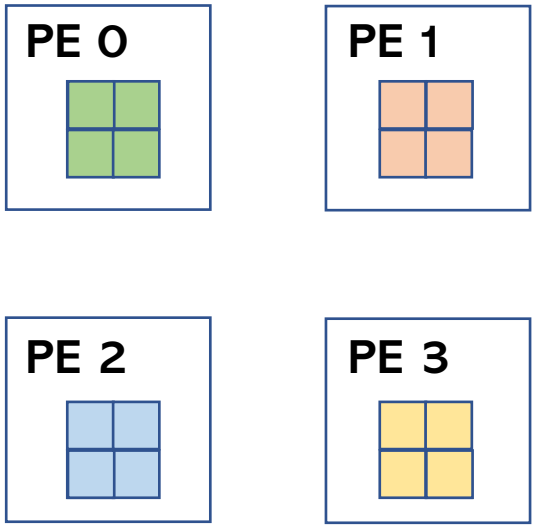
Overview

- Conventional inter-PE communication is slow, but there is room for enhancement
- We aim to make them fast, and make them support flexible use



Inter-PE communication accounts for an average of 76% of total execution time!

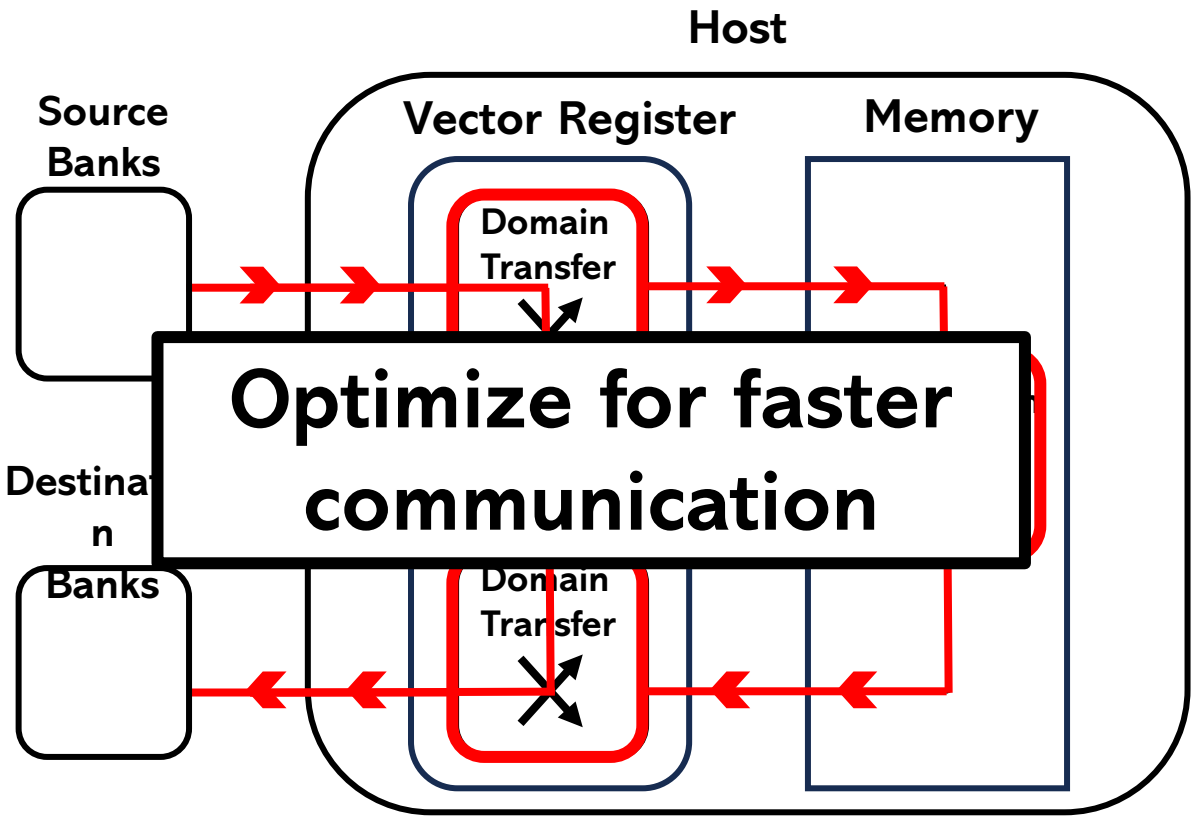
Execution time breakdown of benchmark applications



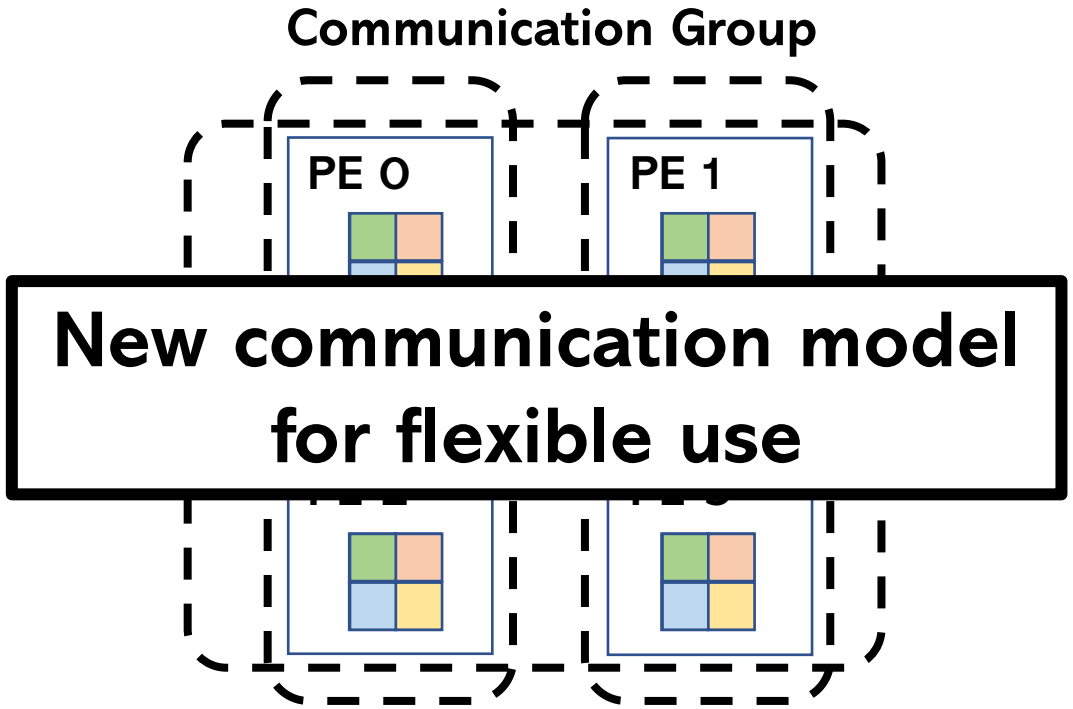
Flexible communication between PEs

Overview

- Conventional inter-PE communication is slow, but there is room for enhancement
- We aim to make them fast, and make them support flexible use

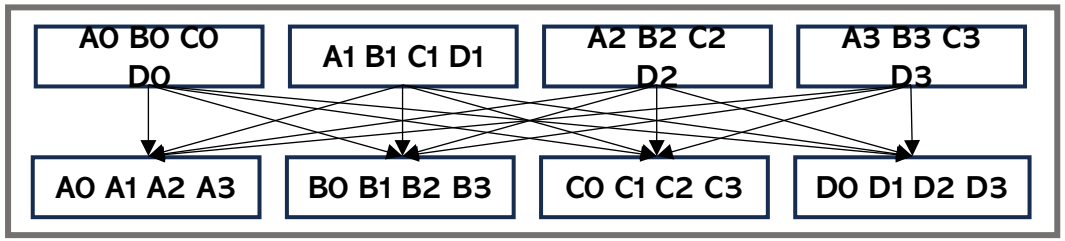


Optimization of inter-PE communication

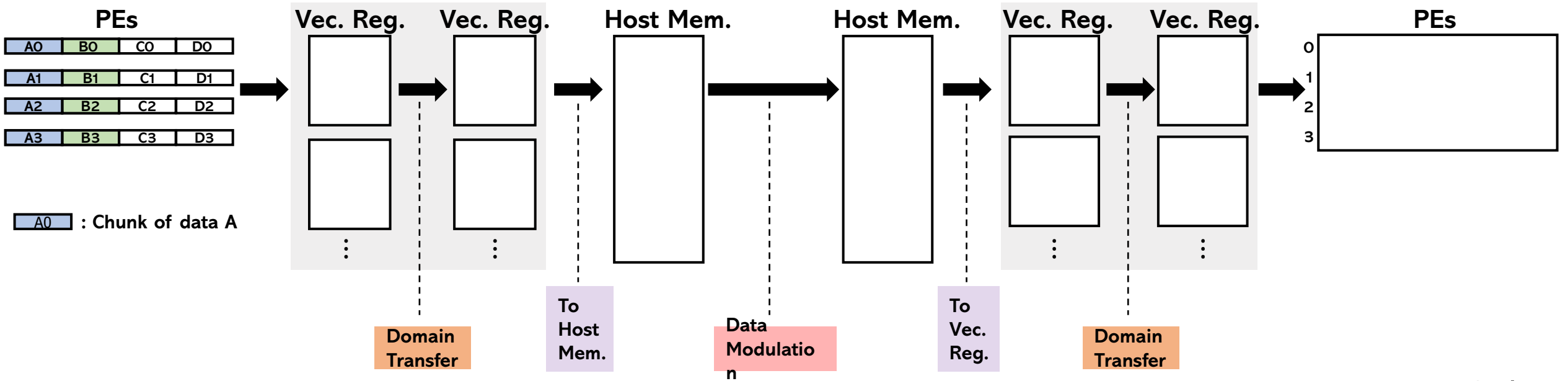


Flexible communication between PEs

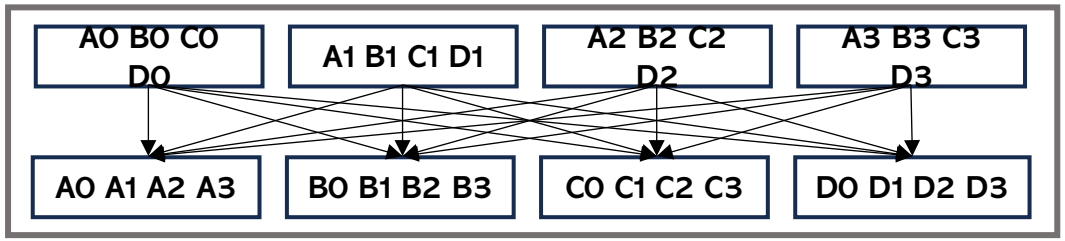
Conventional Inter-PE communication



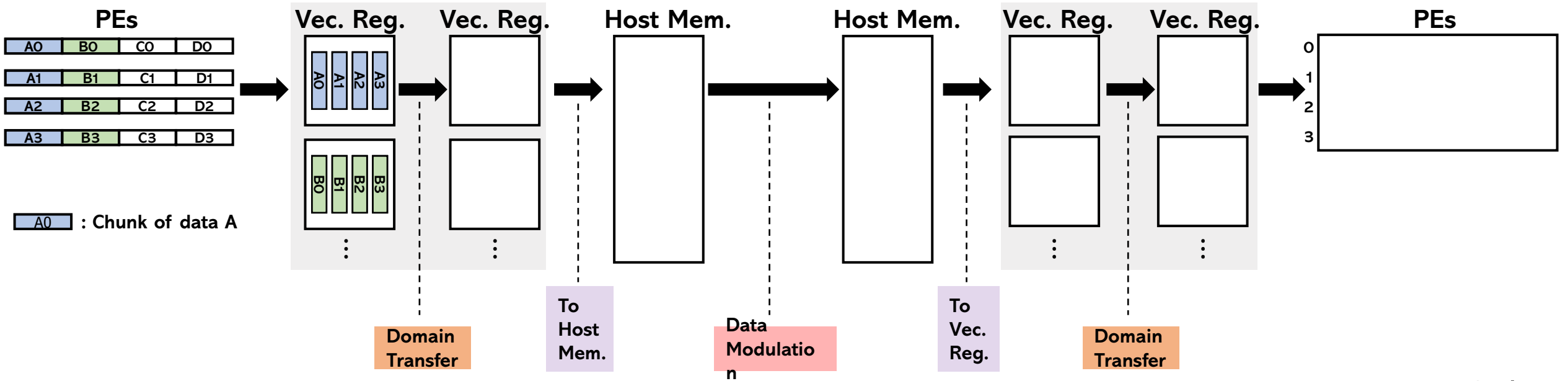
AlltoAll (AA)



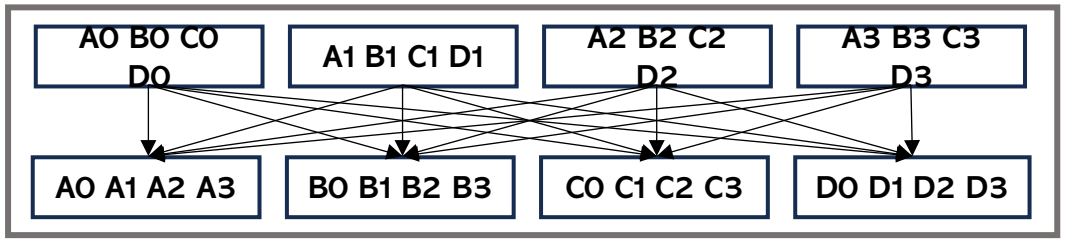
Conventional Inter-PE communication



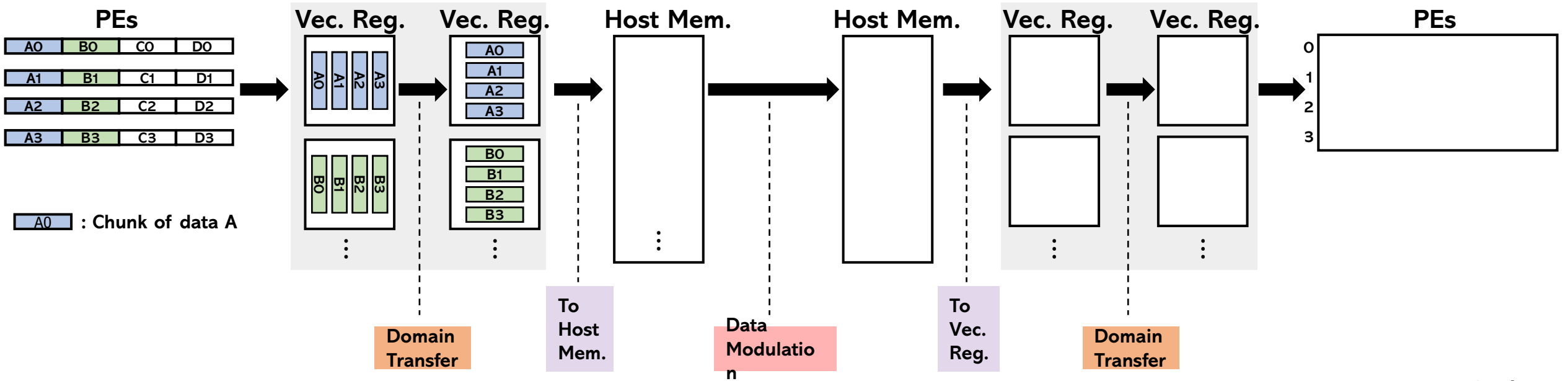
AlltoAll (AA)



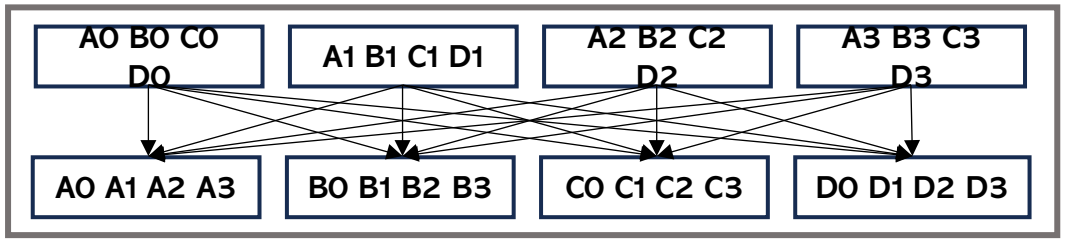
Conventional Inter-PE communication



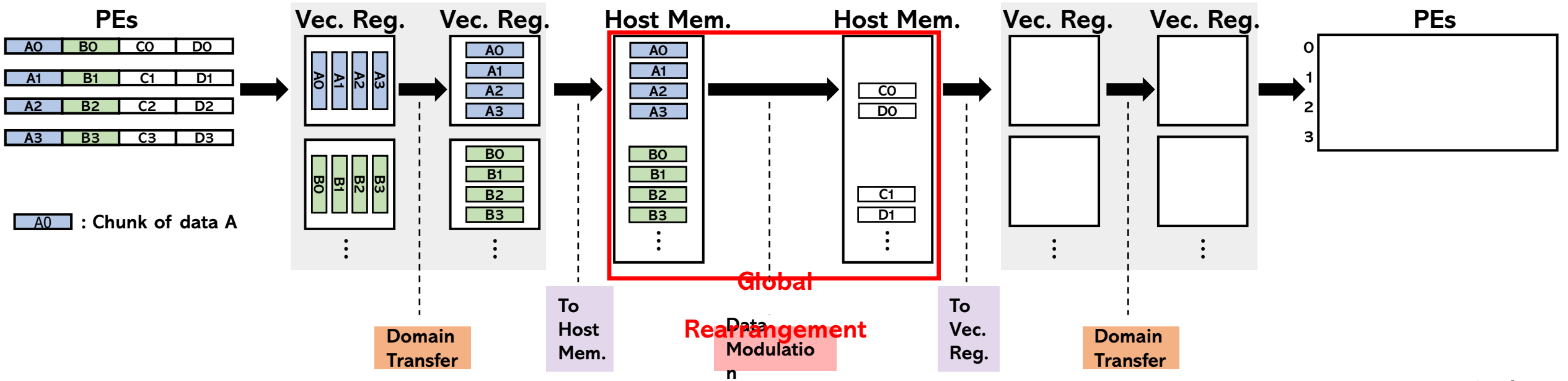
AlltoAll (AA)



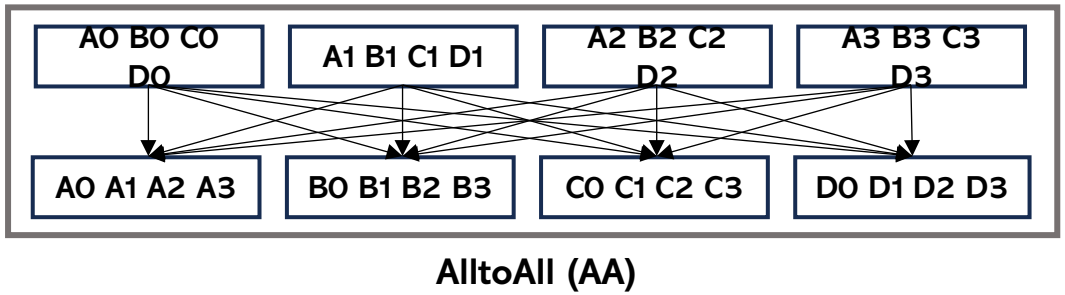
Conventional Inter-PE communication



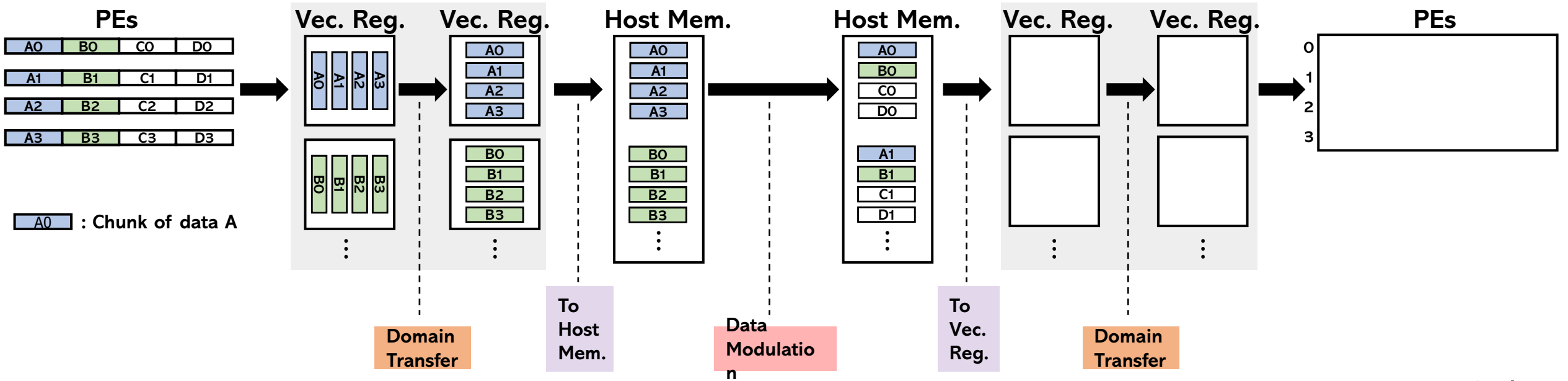
AlltoAll (AA)



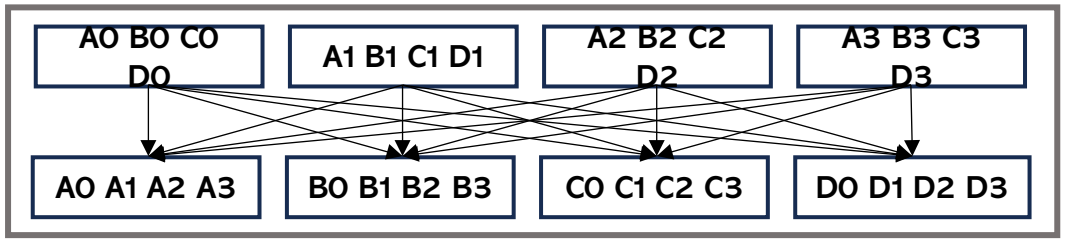
Conventional Inter-PE communication



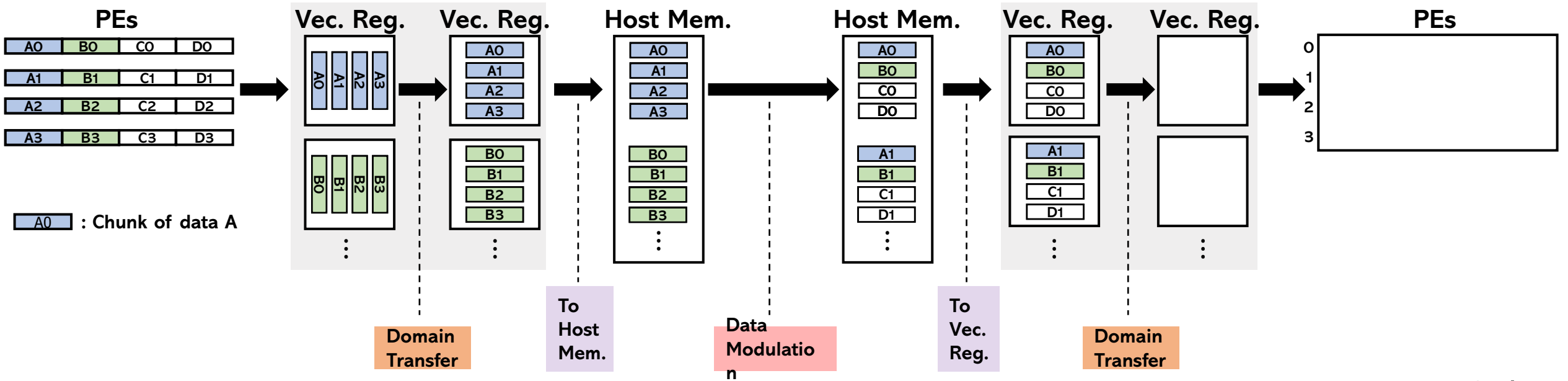
AlltoAll (AA)



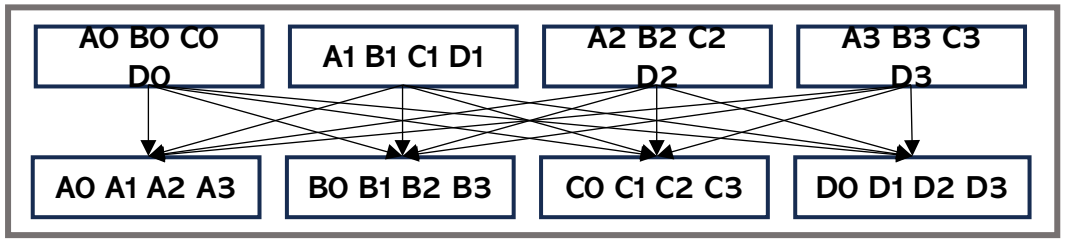
Conventional Inter-PE communication



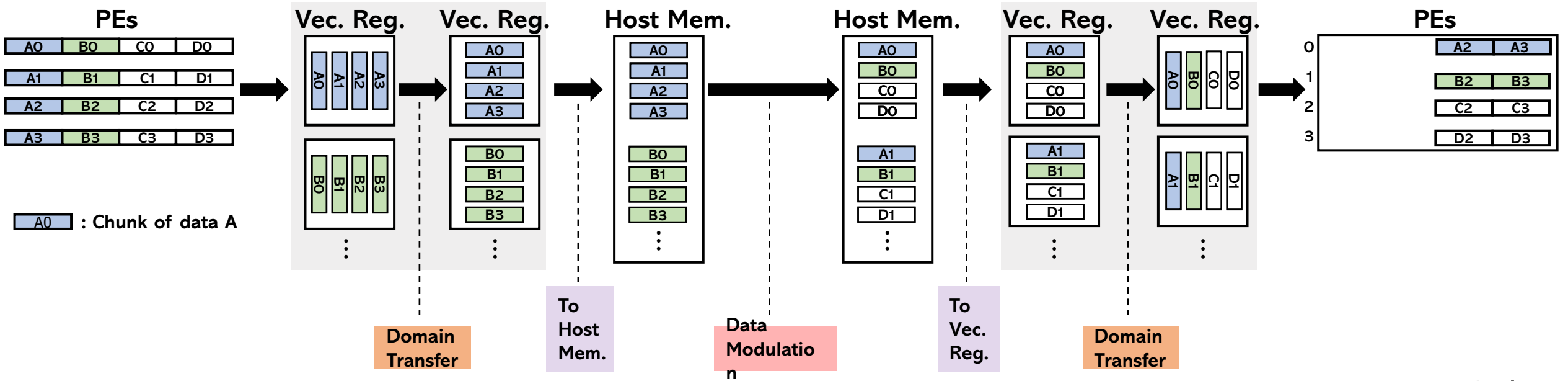
AlltoAll (AA)



Conventional Inter-PE communication

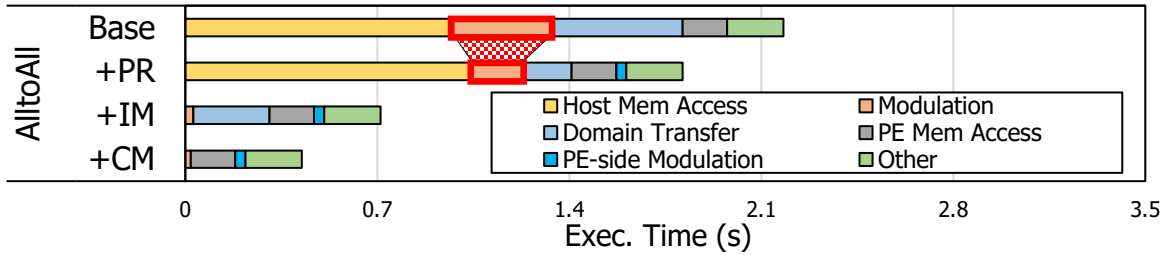


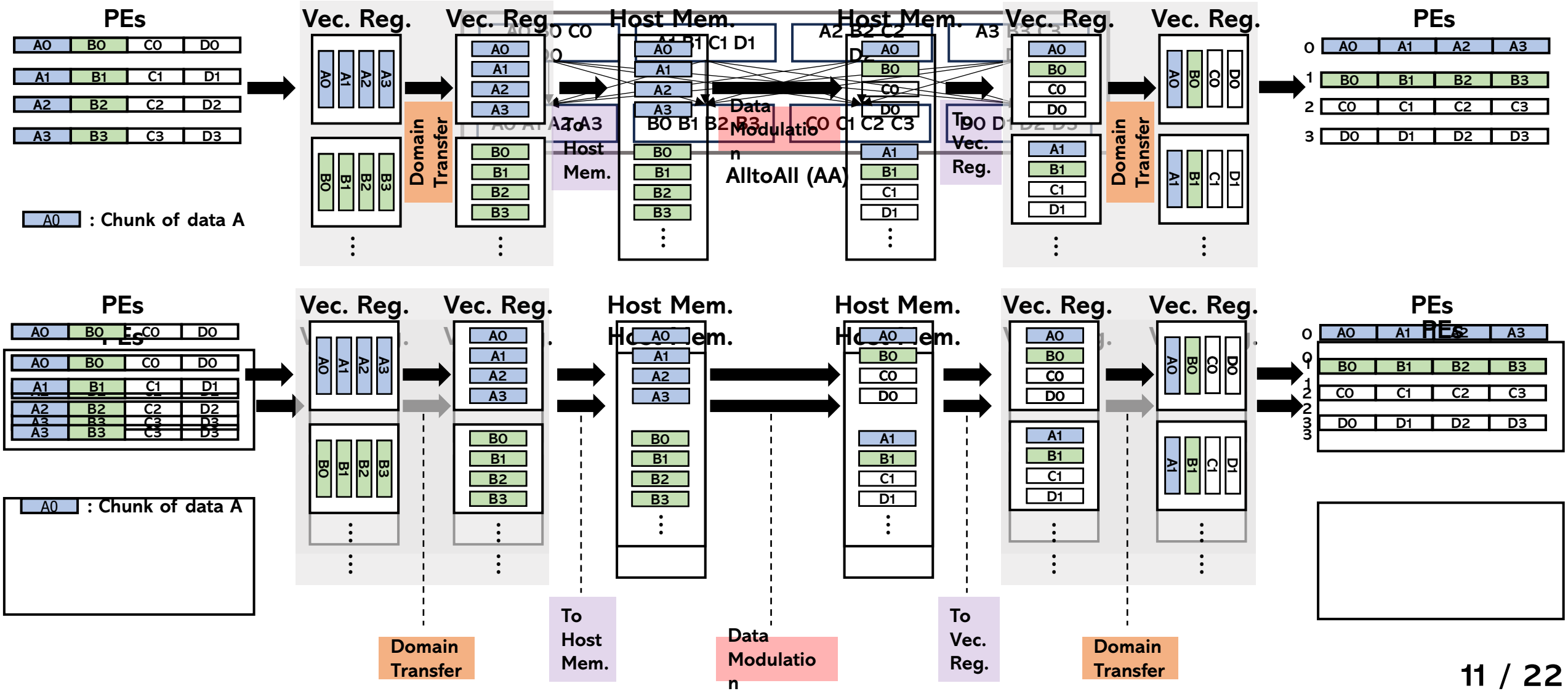
AlltoAll (AA)



1. PE-assisted Reordering

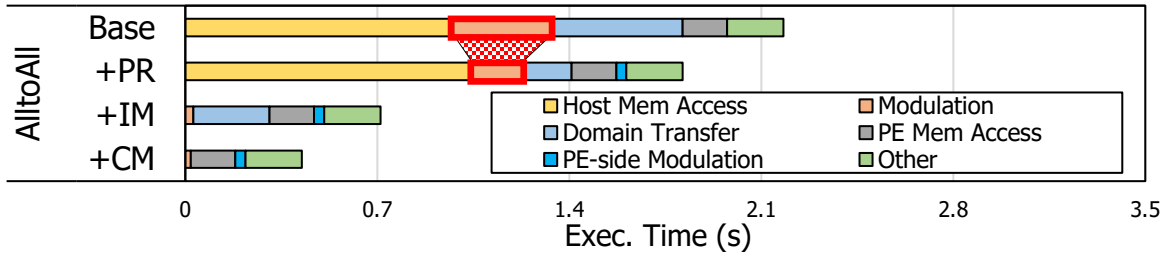
- Reorder data inside PEs to enhance data locality

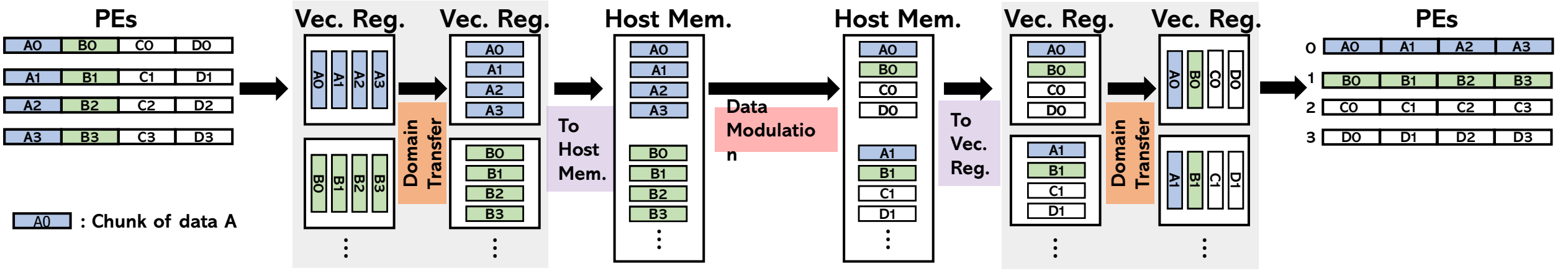


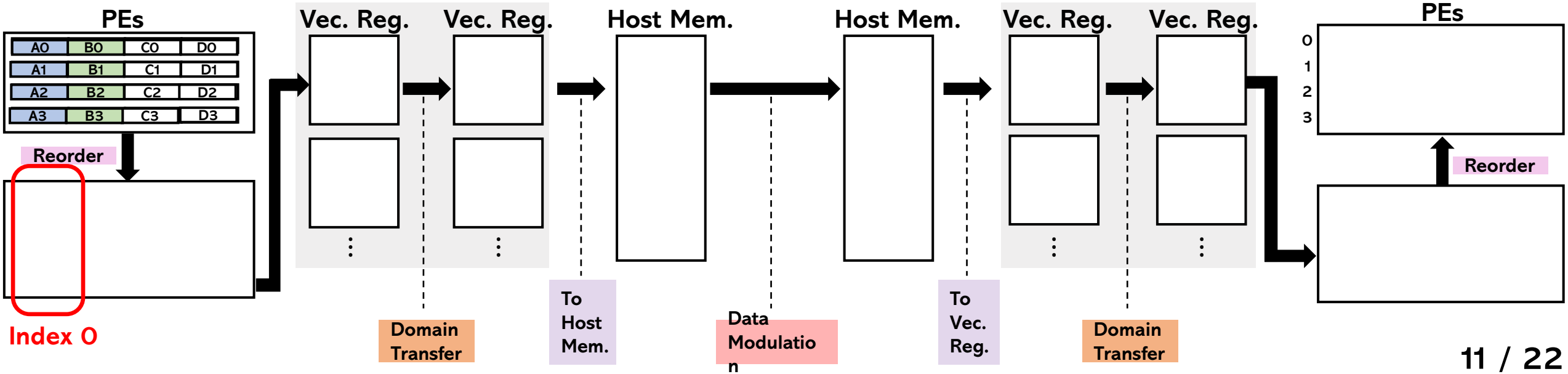


1. PE-assisted Reordering

- Reorder data inside PEs to enhance data locality

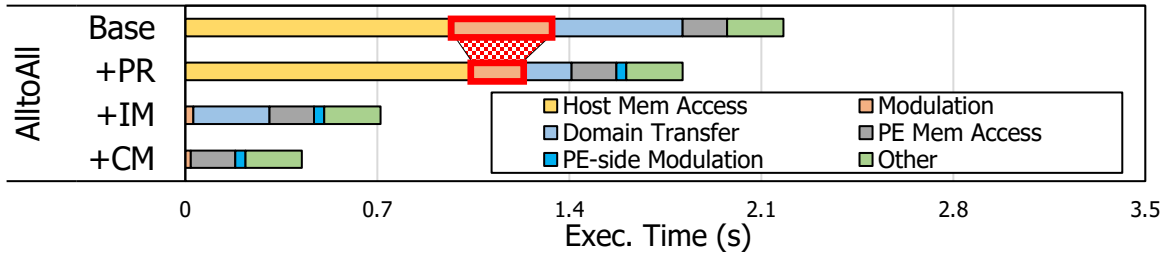


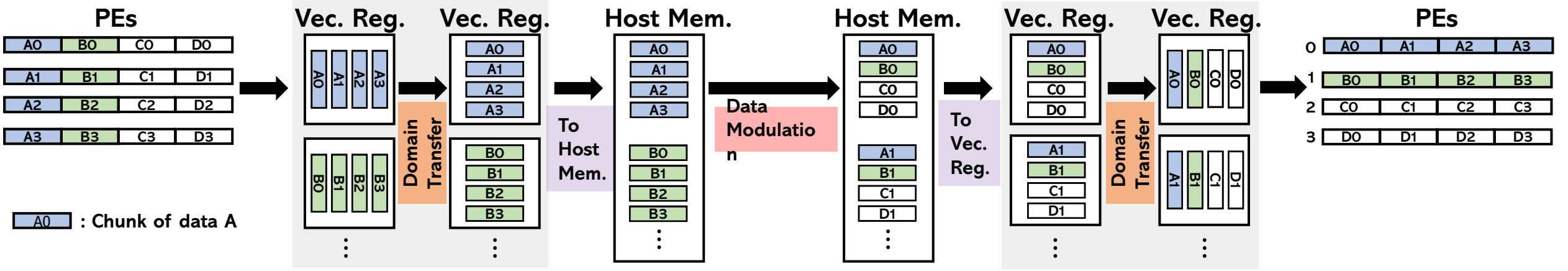


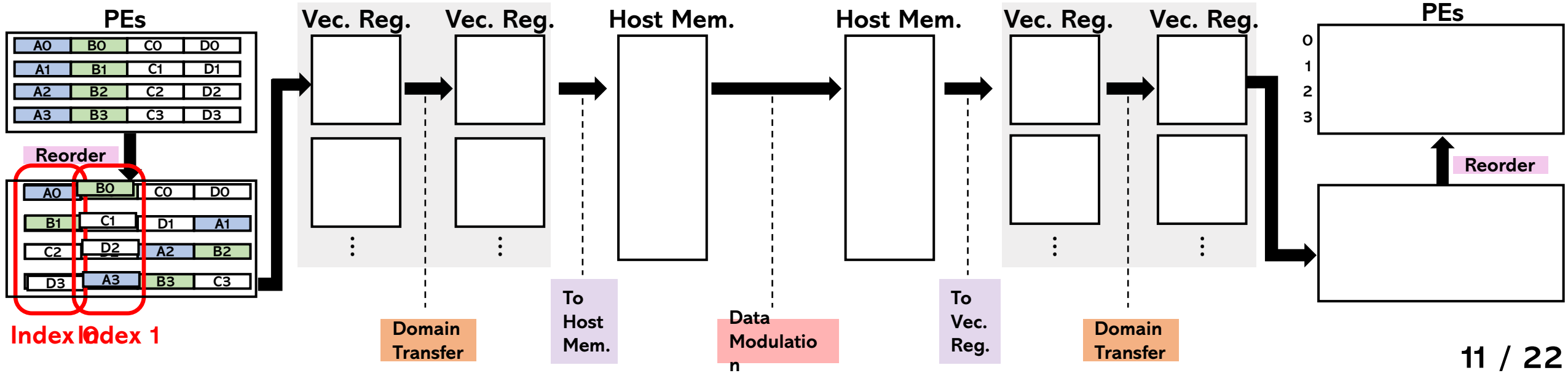


1. PE-assisted Reordering

- Reorder data inside PEs to enhance data locality

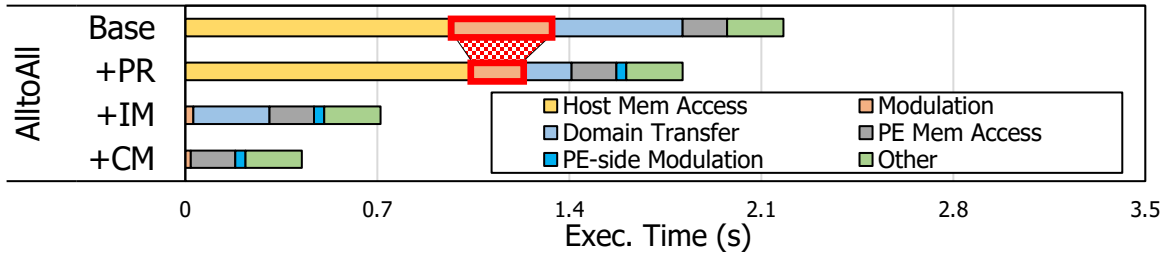


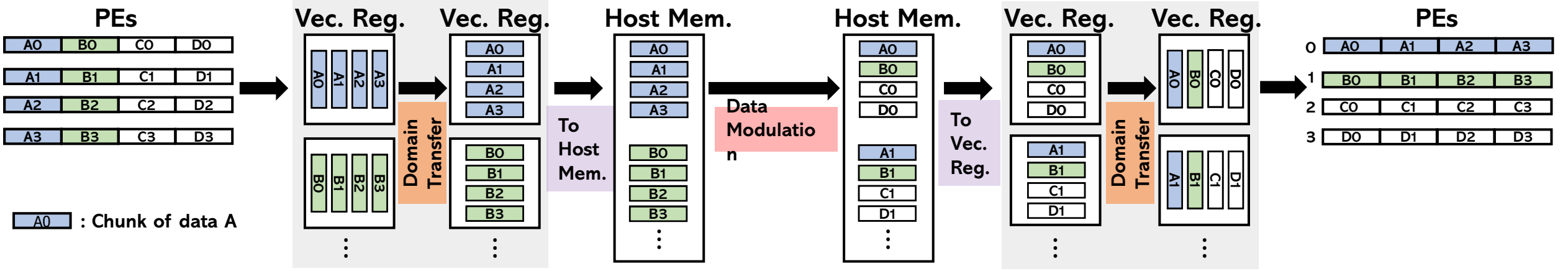


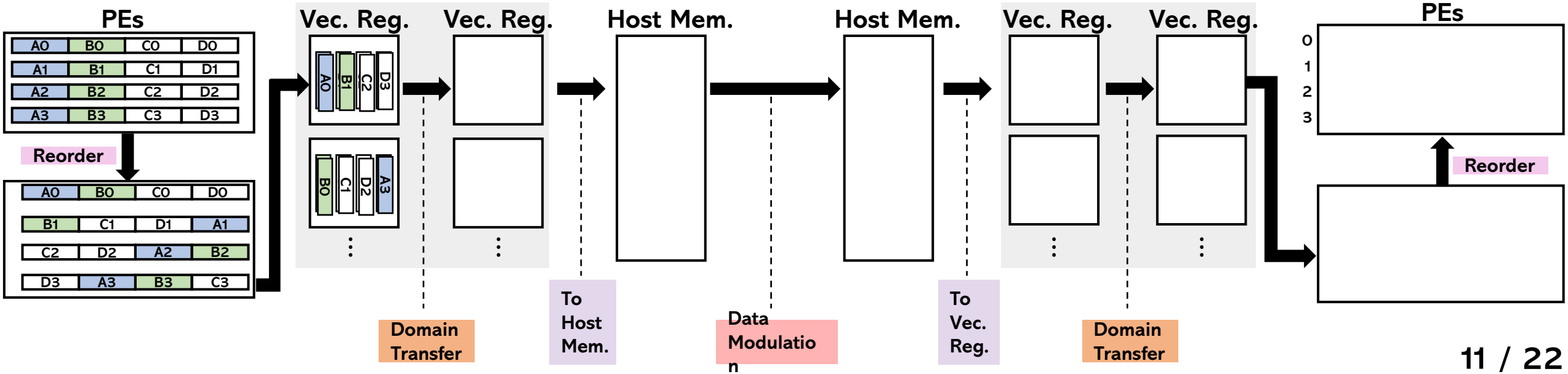


1. PE-assisted Reordering

- Reorder data inside PEs to enhance data locality

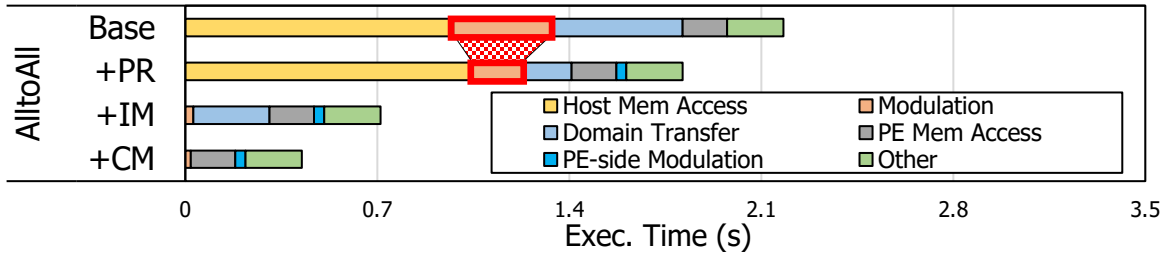


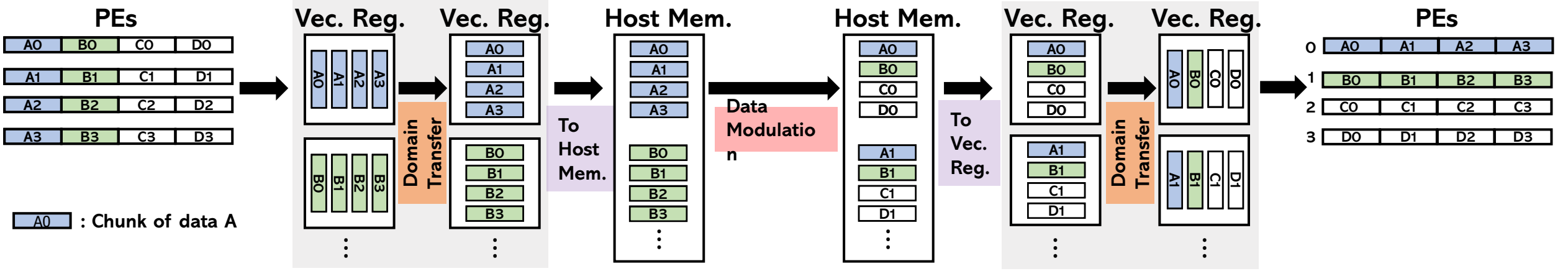


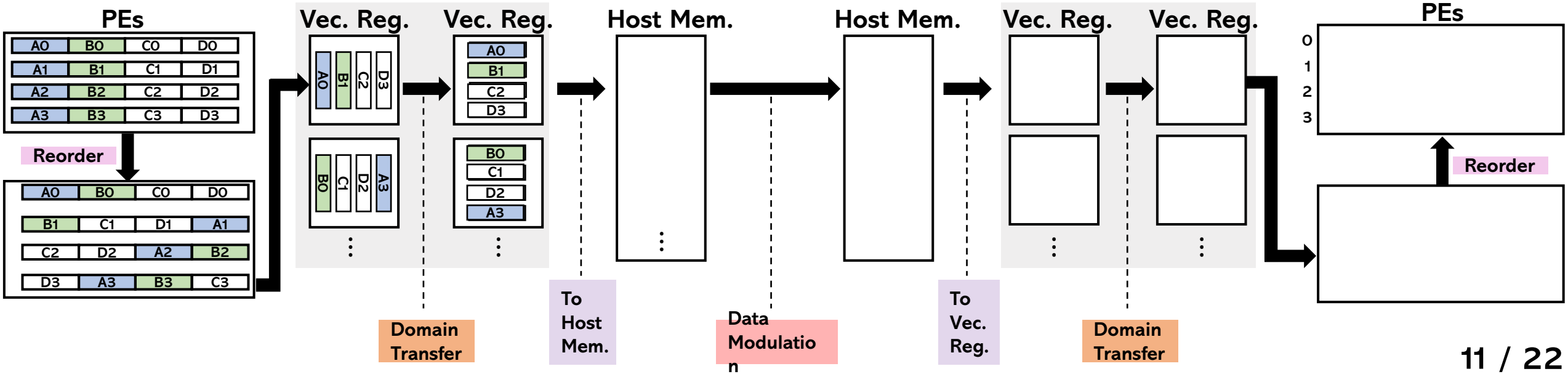


1. PE-assisted Reordering

- Reorder data inside PEs to enhance data locality

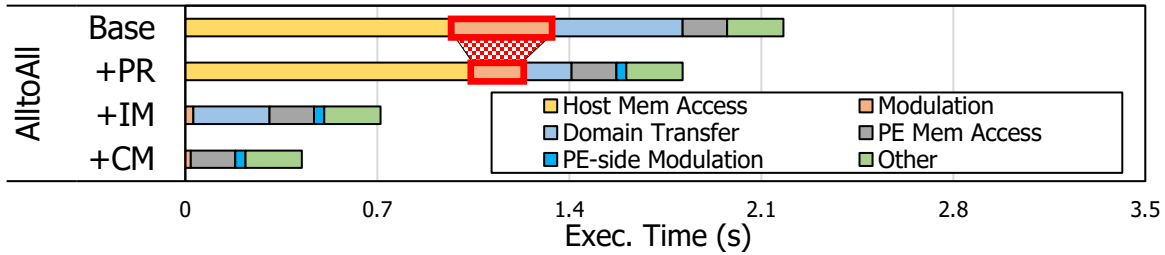


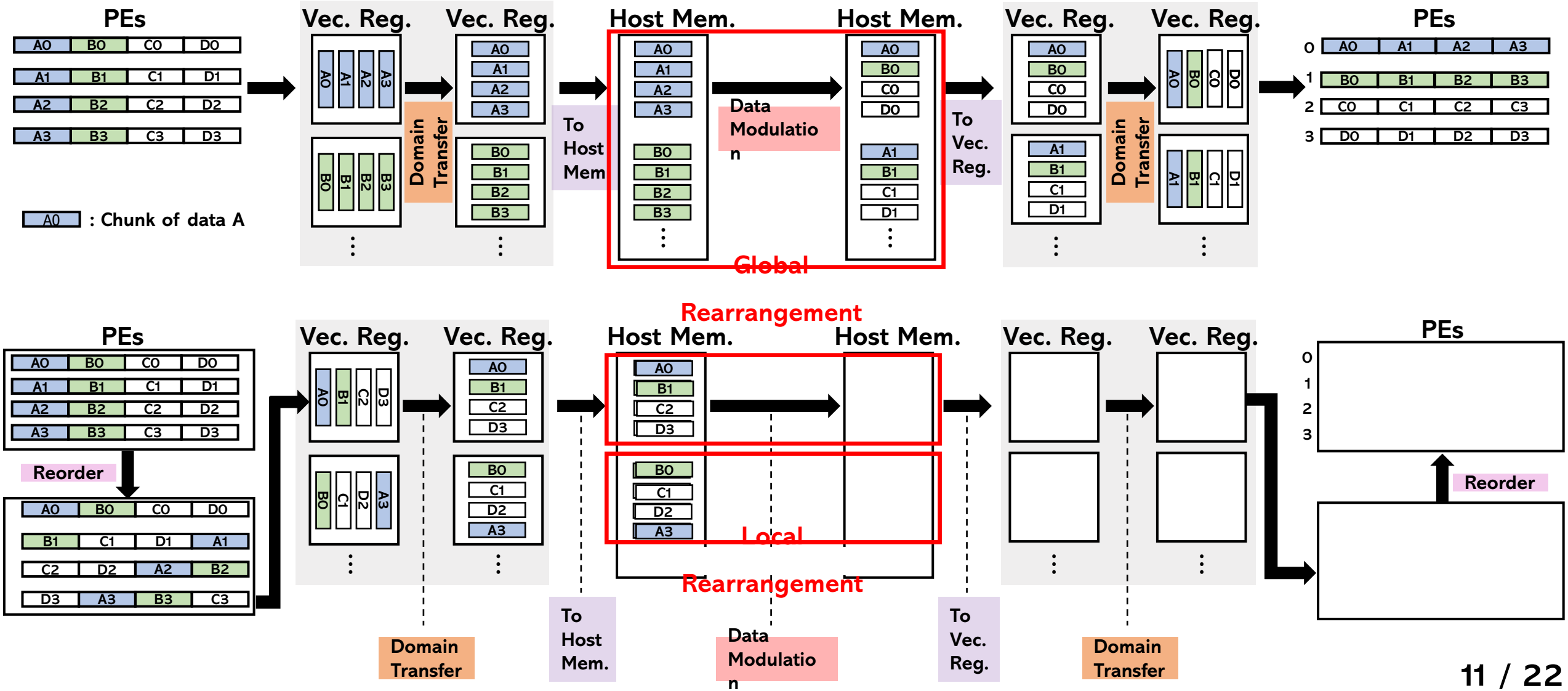




1. PE-assisted Reordering

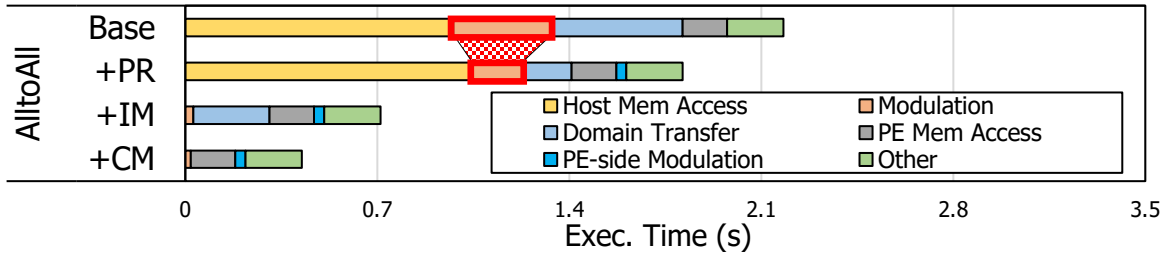
- Reorder data inside PEs to enhance data locality

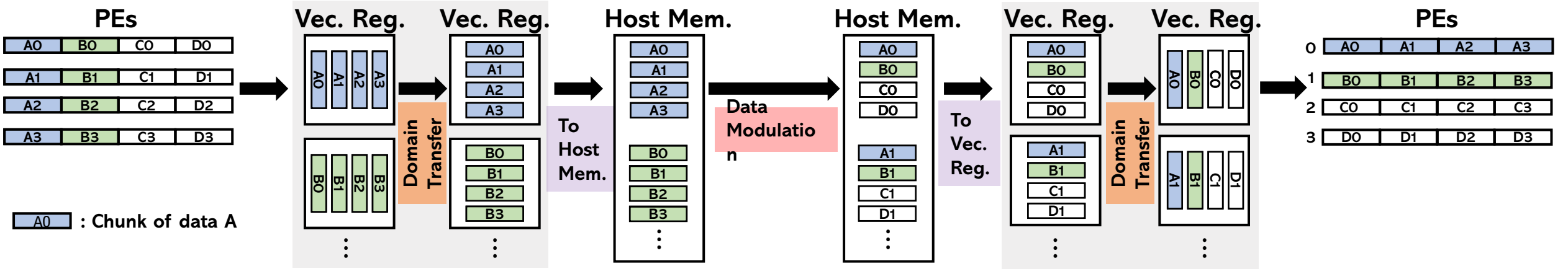


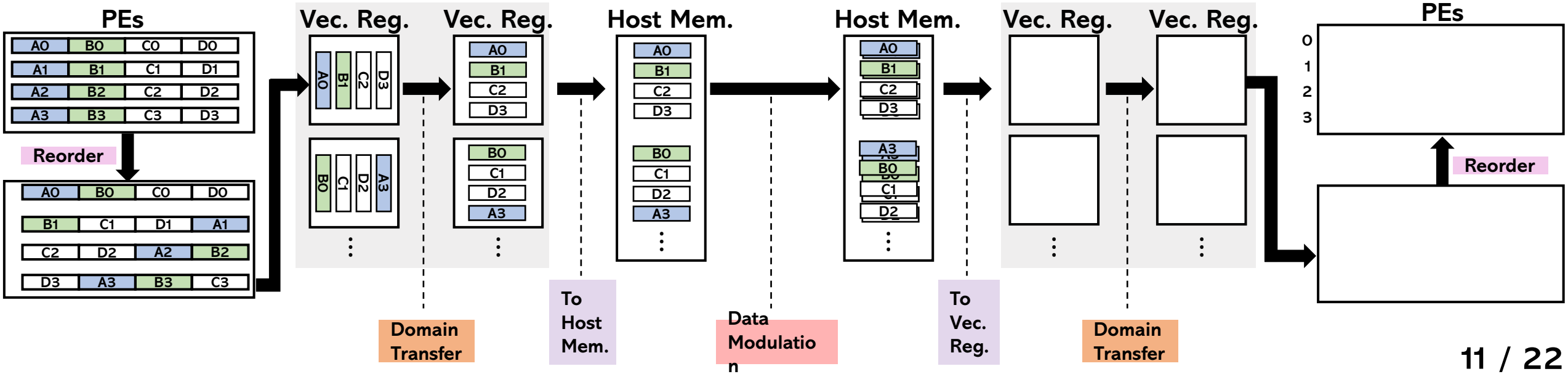


1. PE-assisted Reordering

- Reorder data inside PEs to enhance data locality

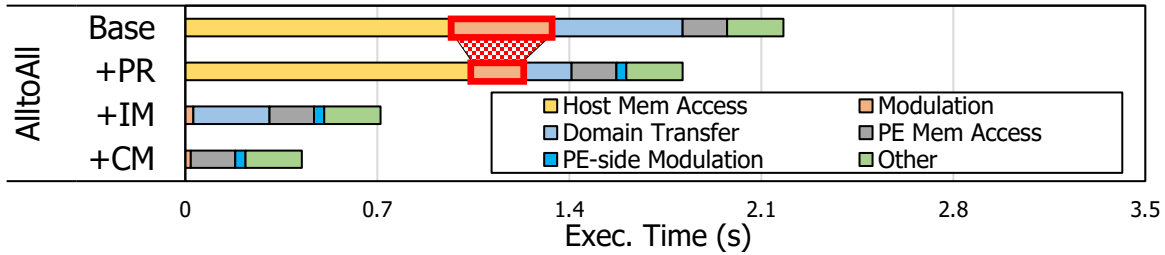


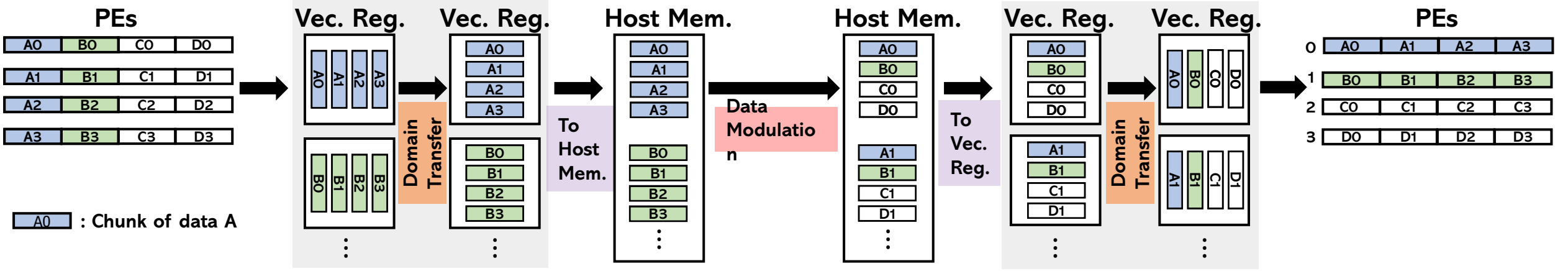


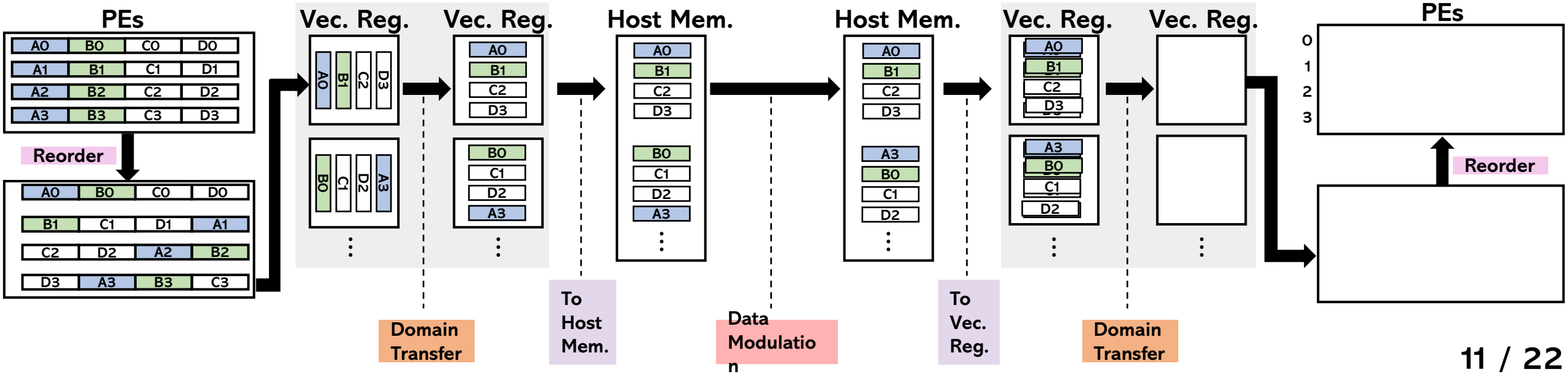


1. PE-assisted Reordering

- Reorder data inside PEs to enhance data locality

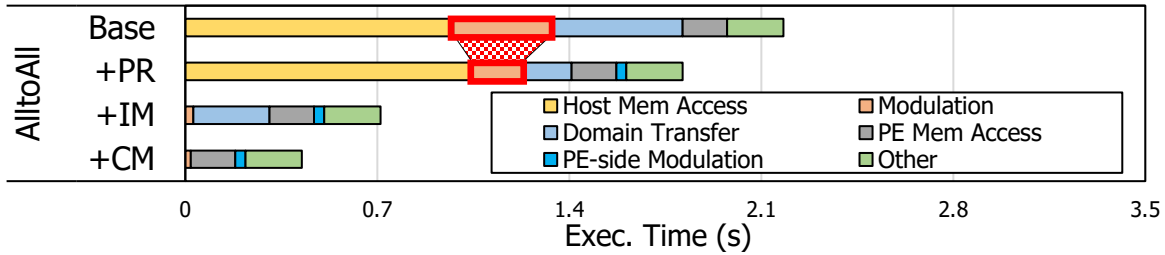


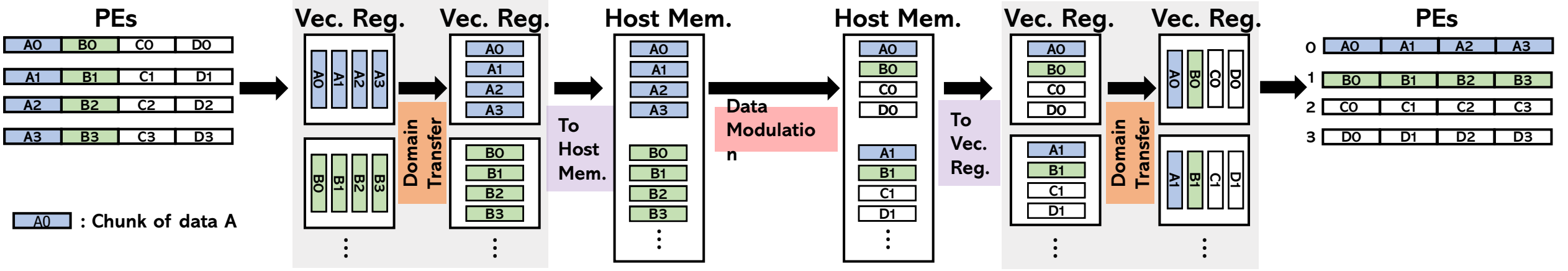


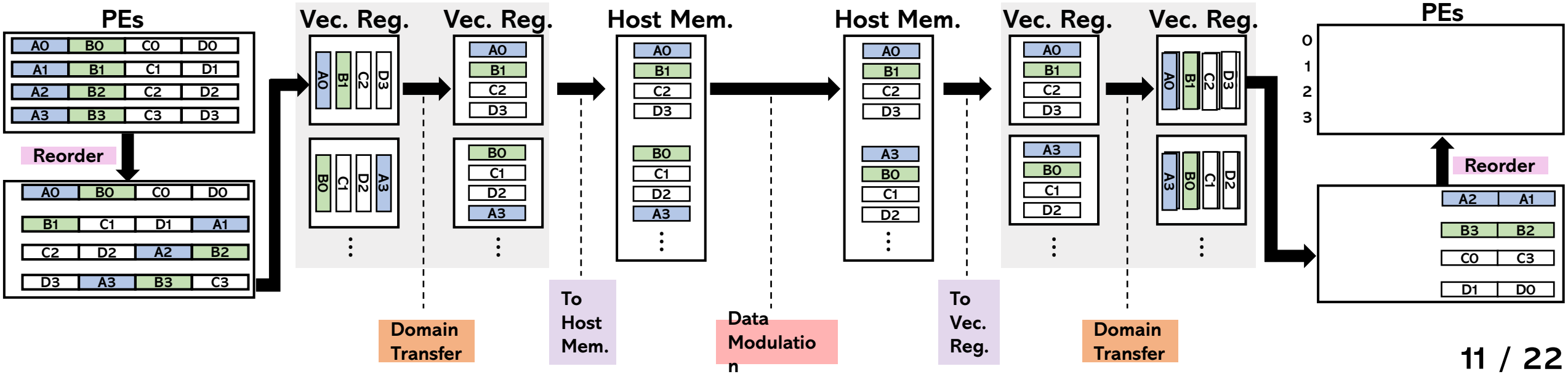


1. PE-assisted Reordering

- Reorder data inside PEs to enhance data locality

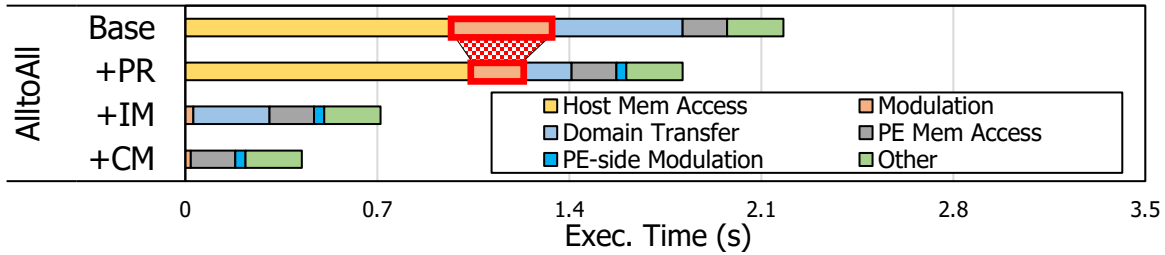


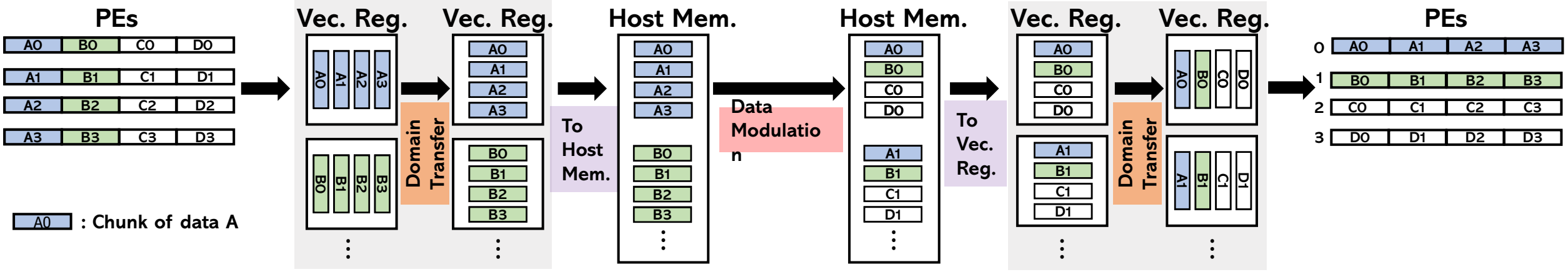


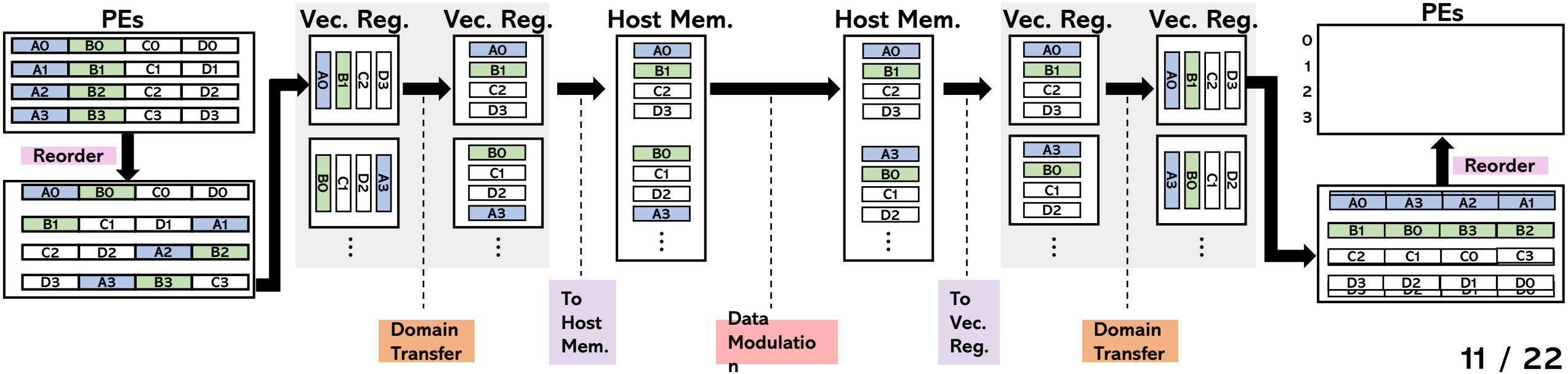


1. PE-assisted Reordering

- Reorder data inside PEs to enhance data locality

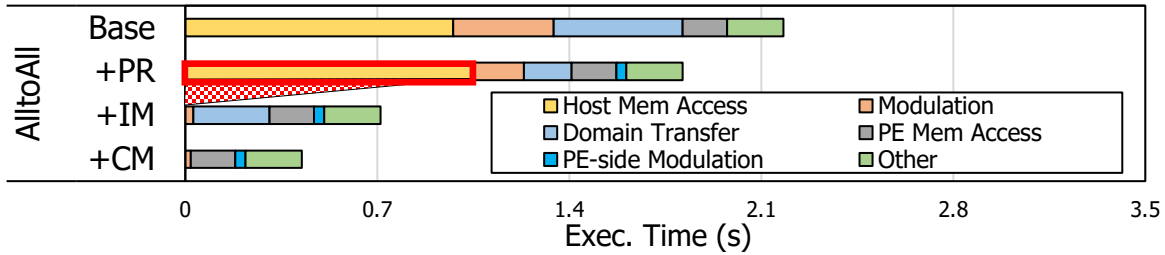


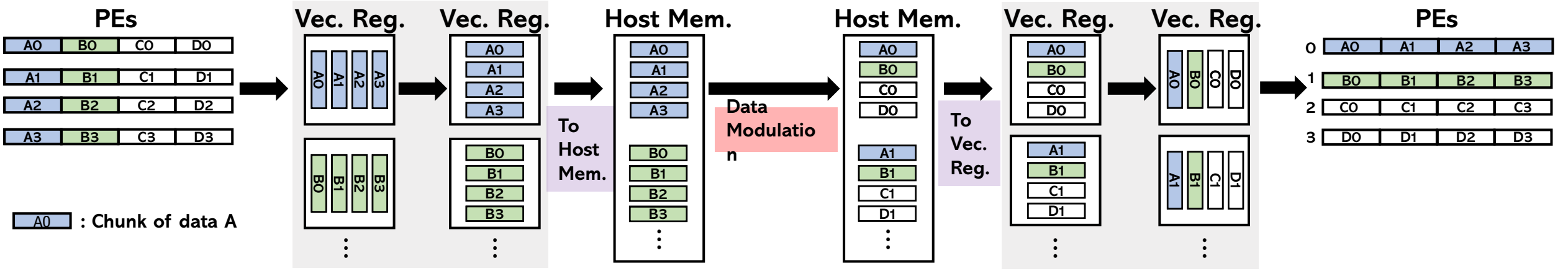




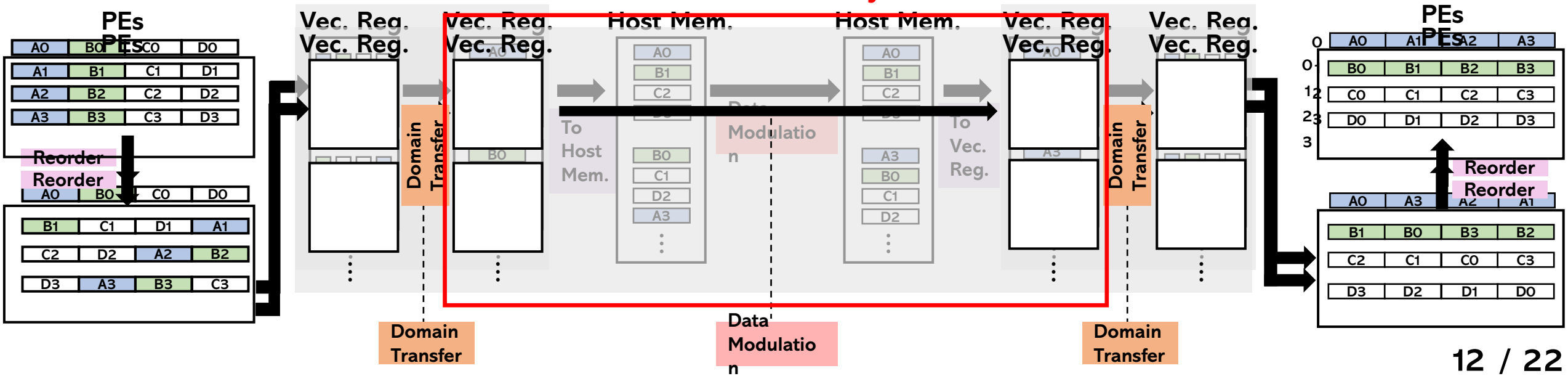
2. In-register Modulation

- Modulate data within the vector register



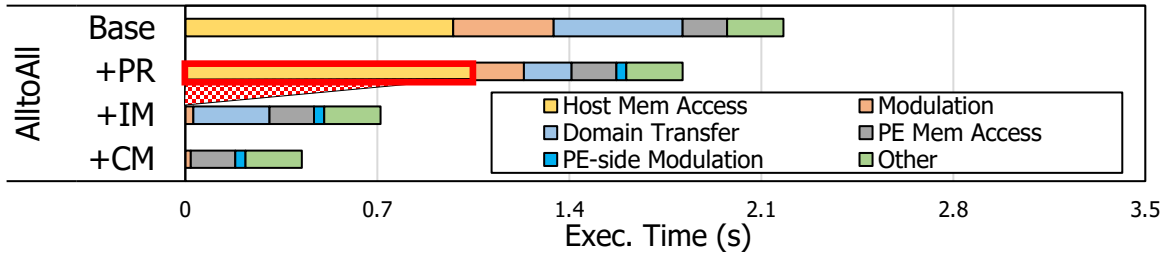


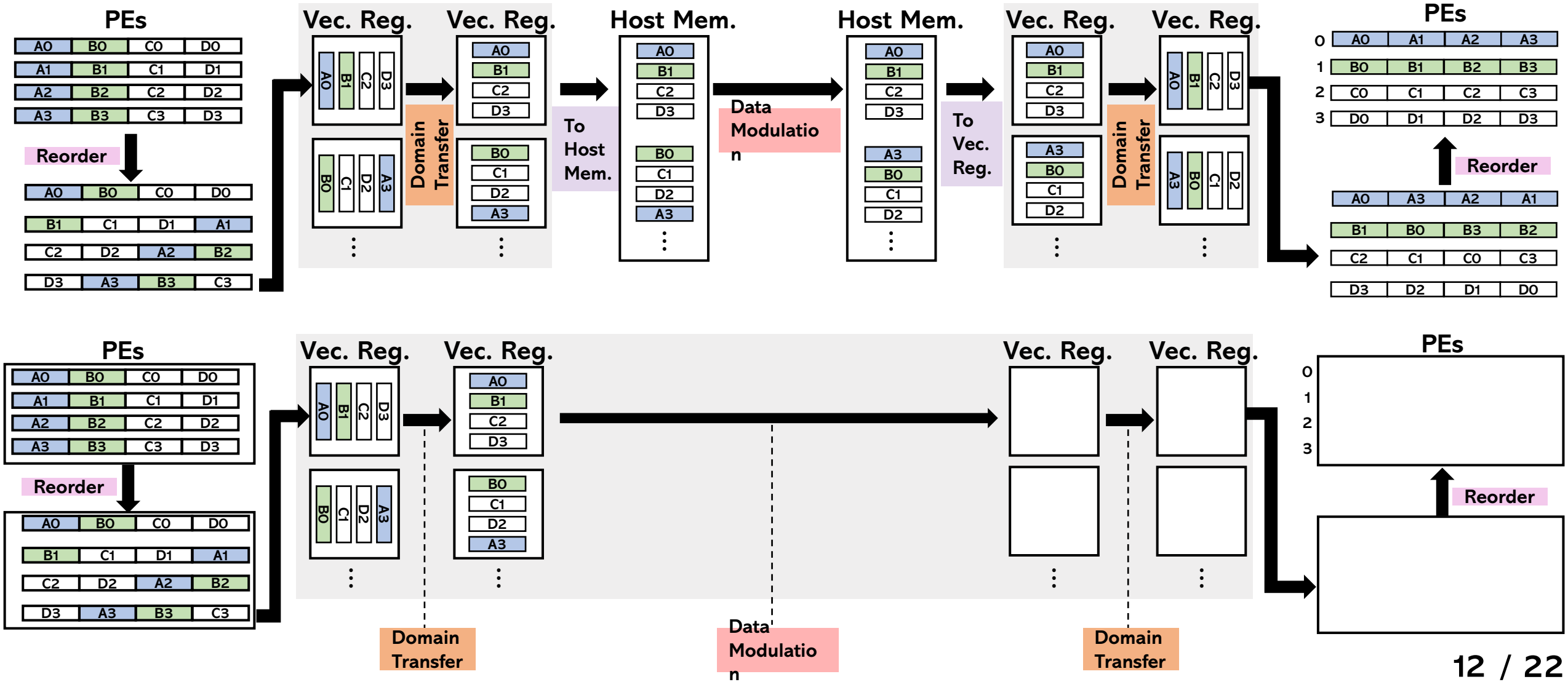
Remove host memory access



2. In-register Modulation

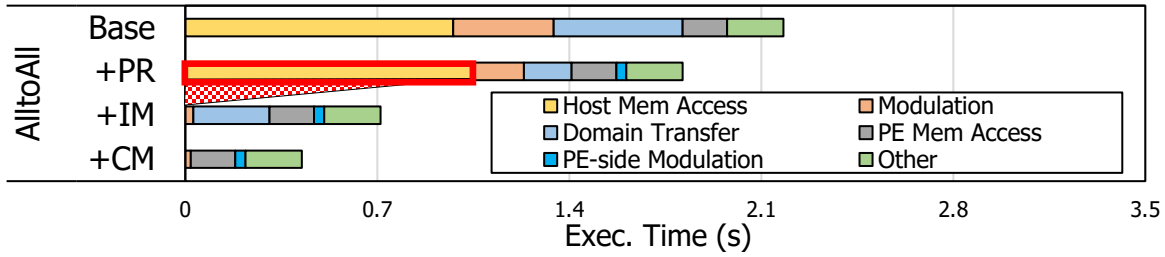
- Modulate data within the vector register

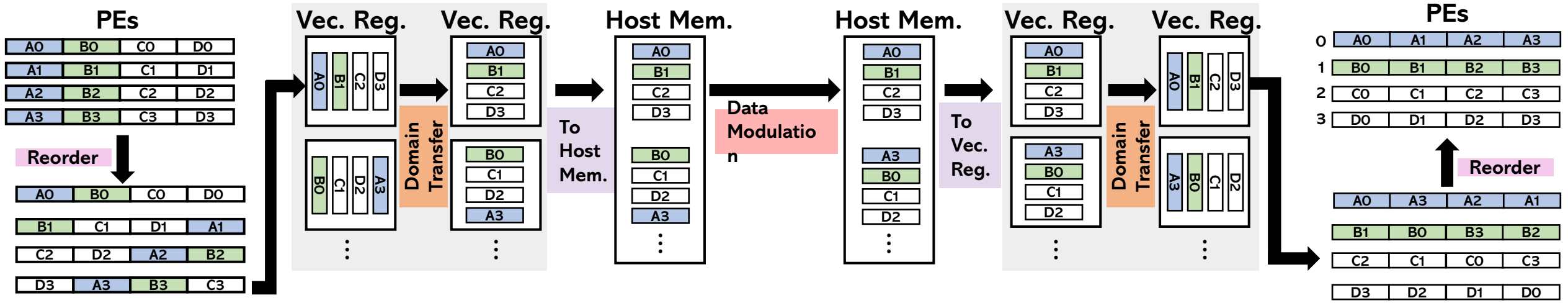


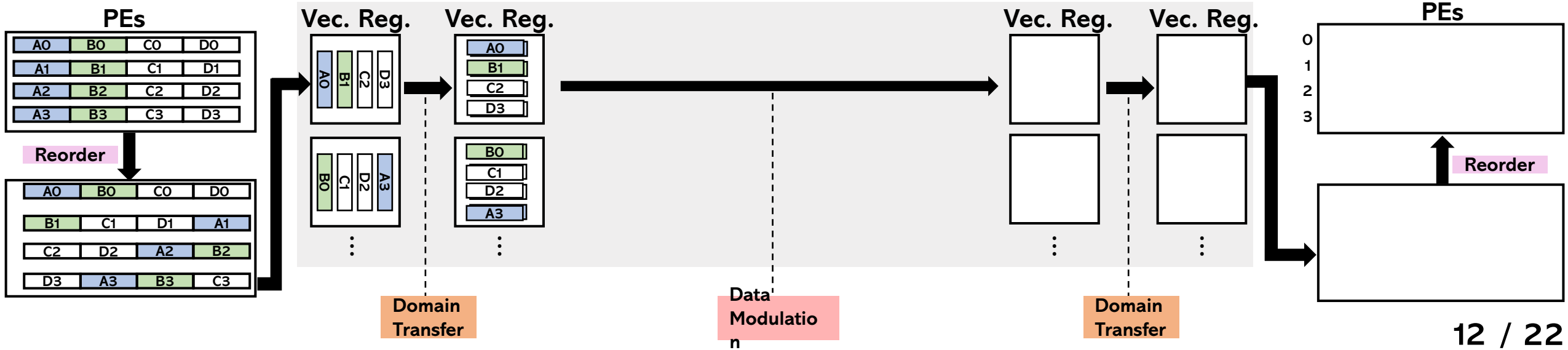


2. In-register Modulation

- Modulate data within the vector register

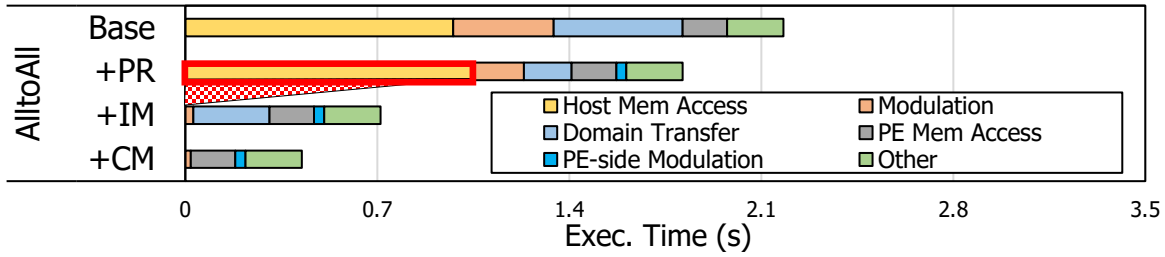


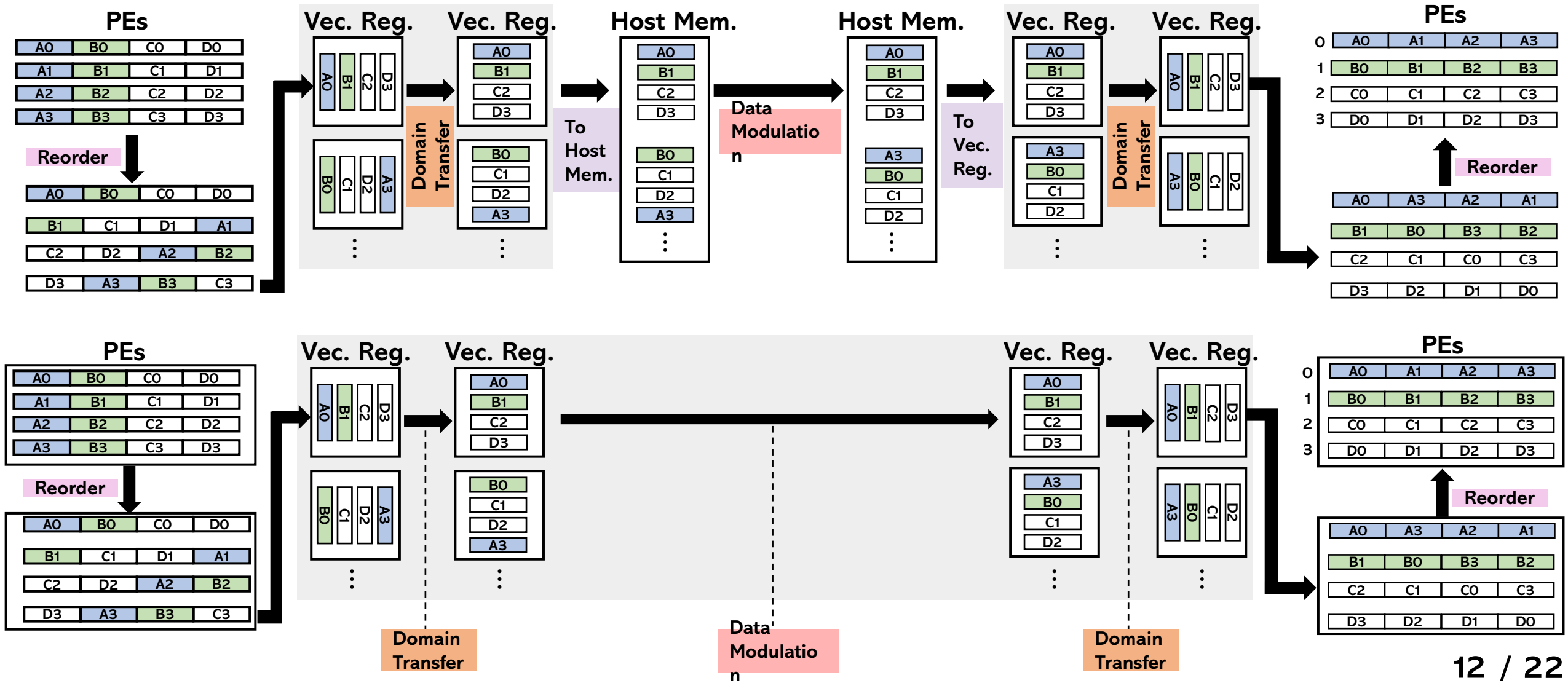




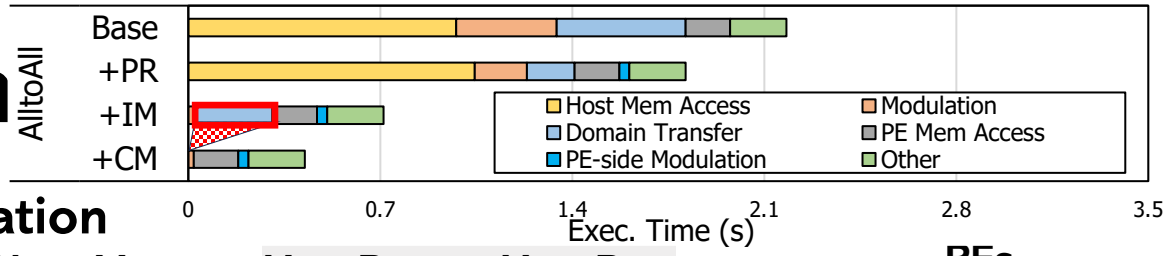
2. In-register Modulation

- Modulate data within the vector register

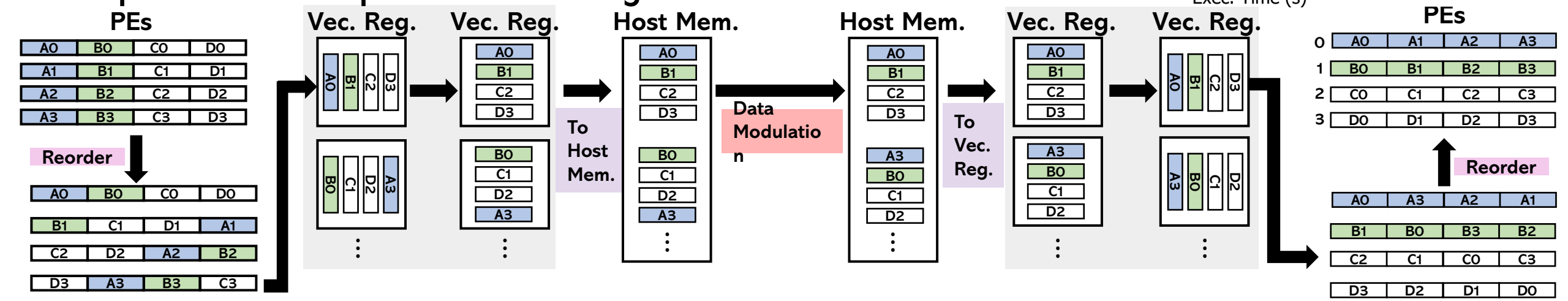


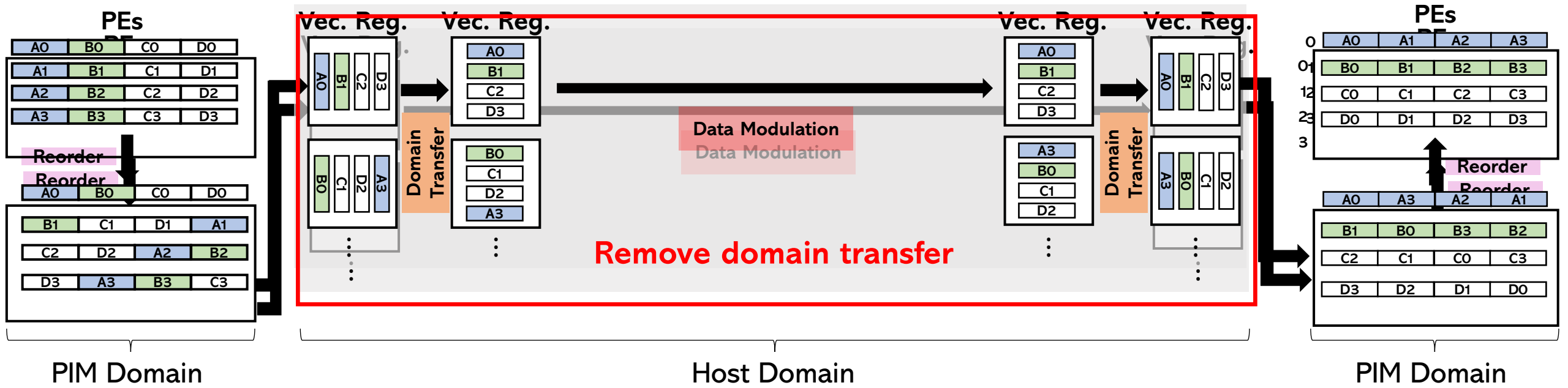


3. Cross-domain Modulation

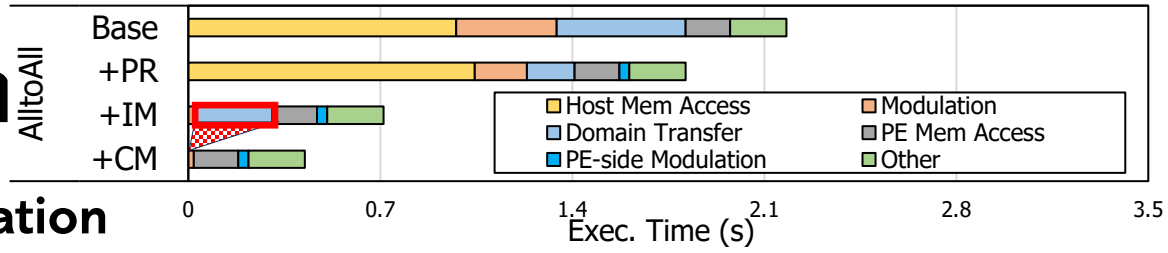


- Replace inefficient procedures with light bitwise rotation

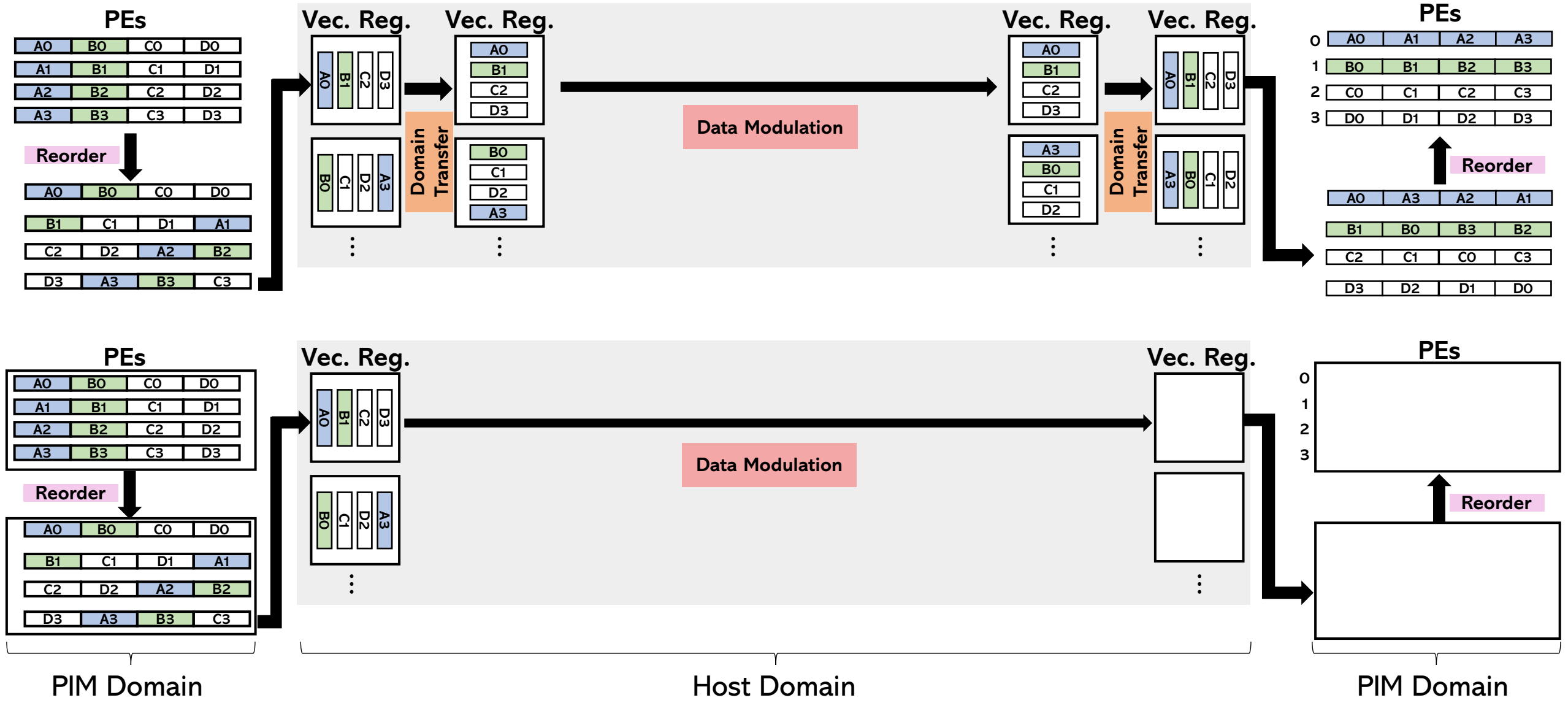




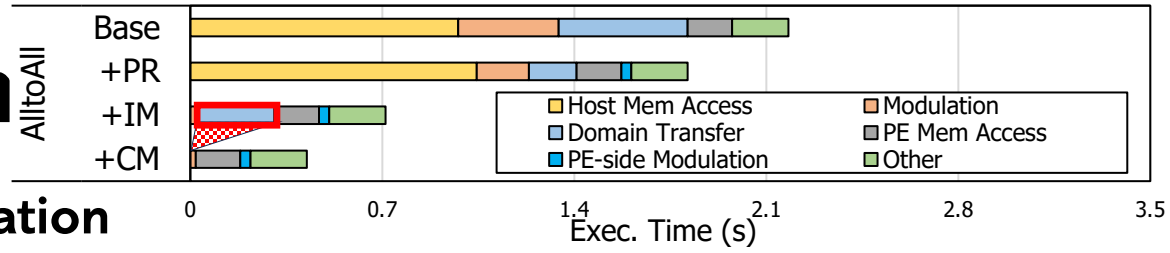
3. Cross-domain Modulation



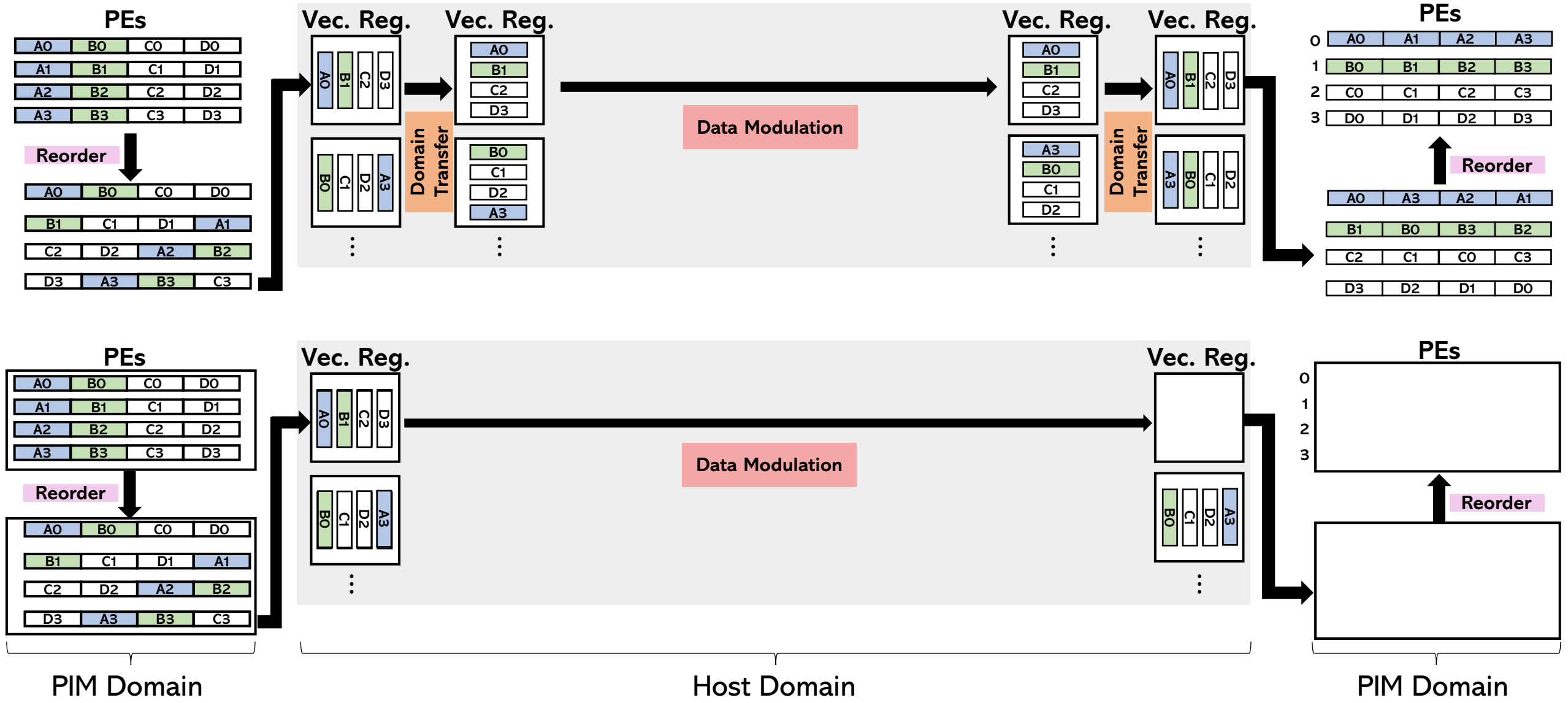
- Replace inefficient procedures with light bitwise rotation



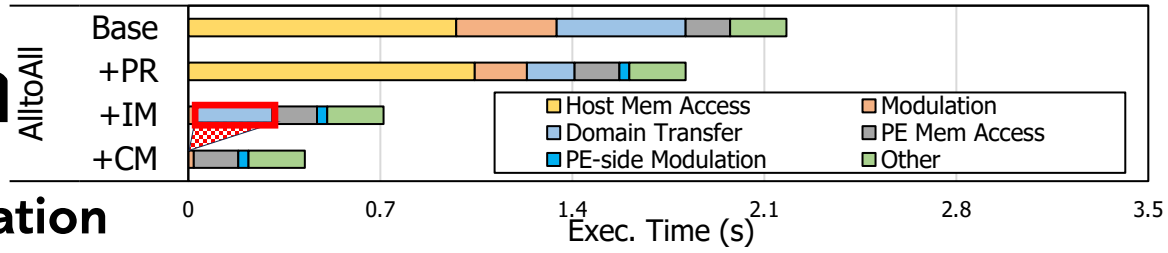
3. Cross-domain Modulation



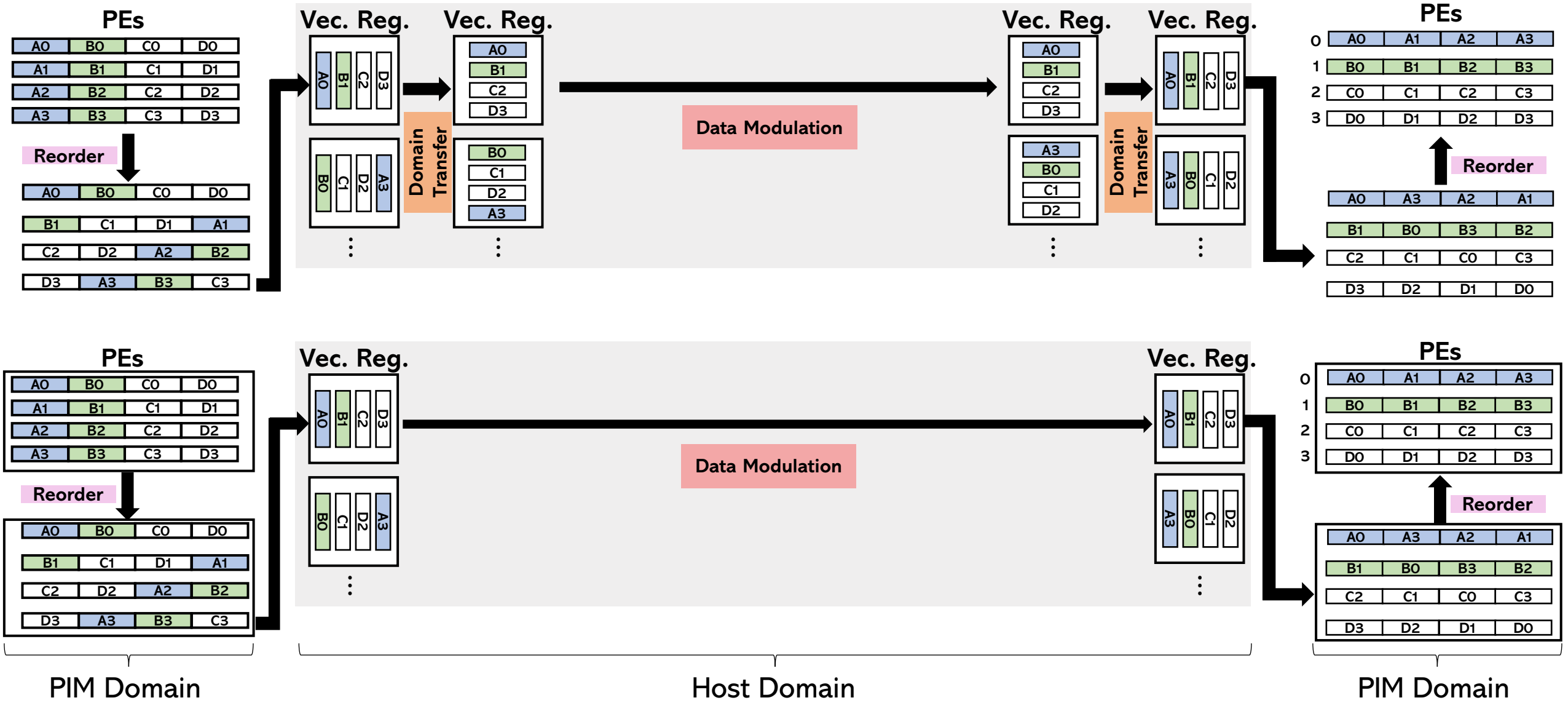
- Replace inefficient procedures with light bitwise rotation



3. Cross-domain Modulation

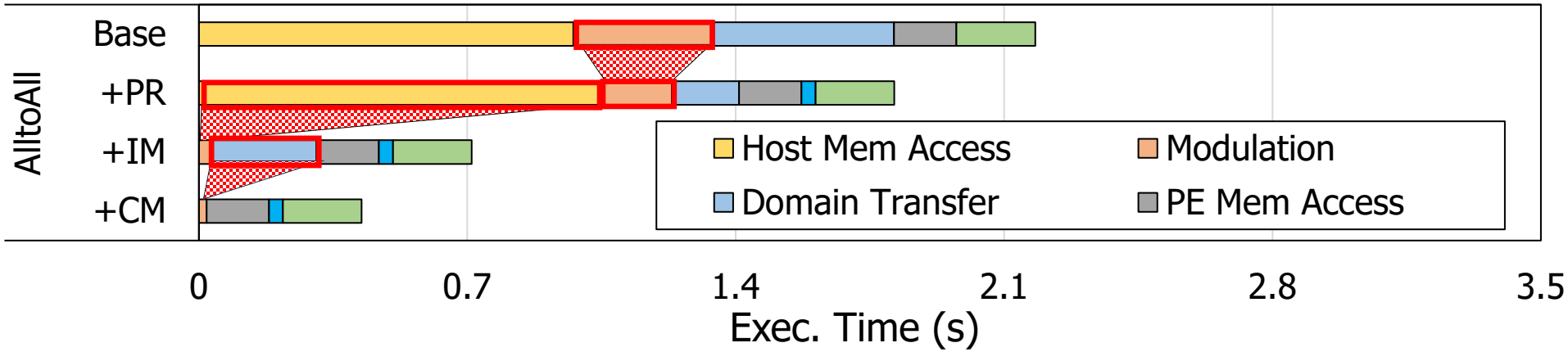


- Replace inefficient procedures with light bitwise rotation



PID-Comm's Optimization

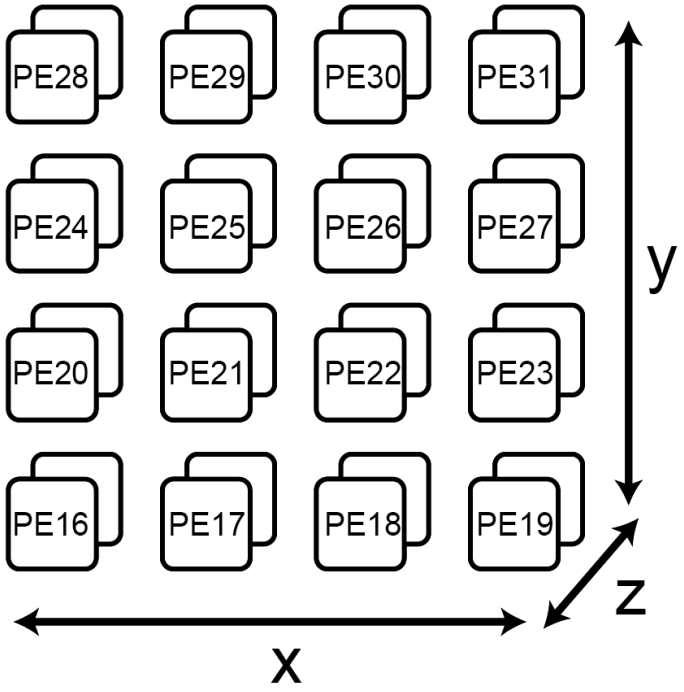
- **Three optimization techniques reduce their target bottleneck**
  - PE-assisted Reordering (PR) targets data modulation
  - In-register Modulation (IM) targets host memory access
  - Cross-domain Modulation (CM) targets domain transfer



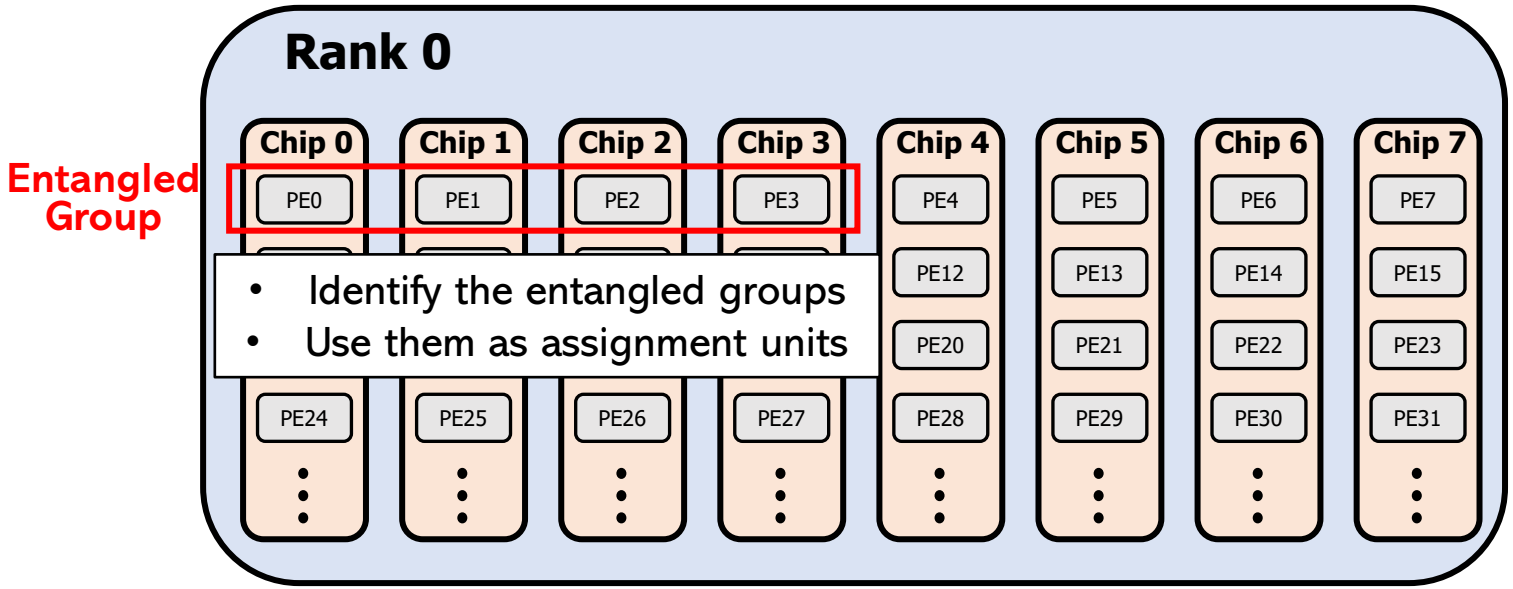
- **PID-Comm is fast but for practical use we need a model that supports**
  - Diverse Communication Groups
  - Multi-instance invocation

4. The Hypercube Model

- **Maintaining both optimal performance and high flexibility**
- **Mapping virtual hypercube to physical banks**
  - Follow the DRAM hierarchy in the order of chip-bank-rank-channel

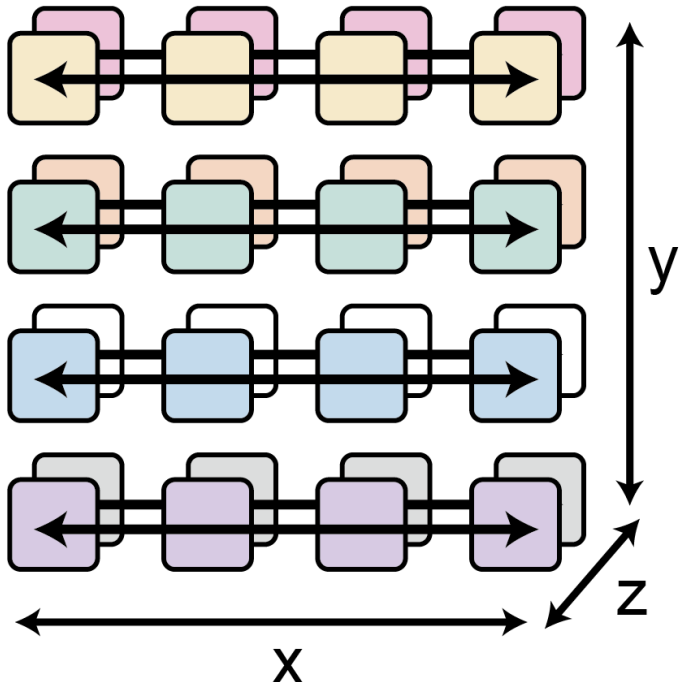


Hypercube (4, 4, 2)

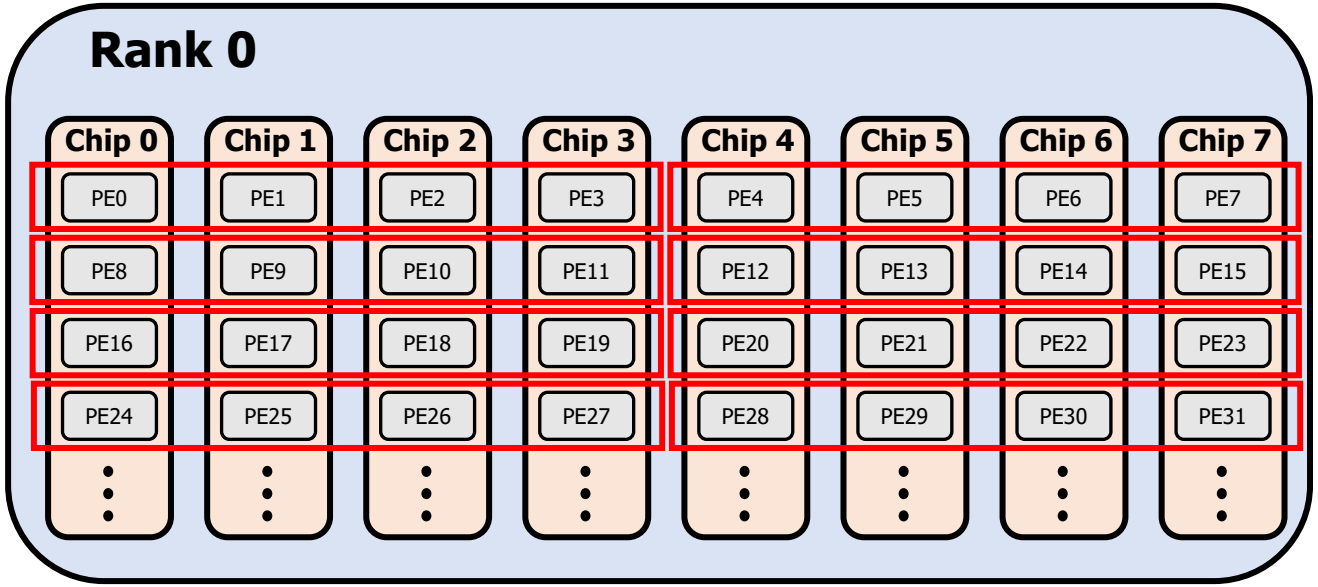


4. The Hypercube Model

- Allows multiple communication invocations
- Support diverse communication groups



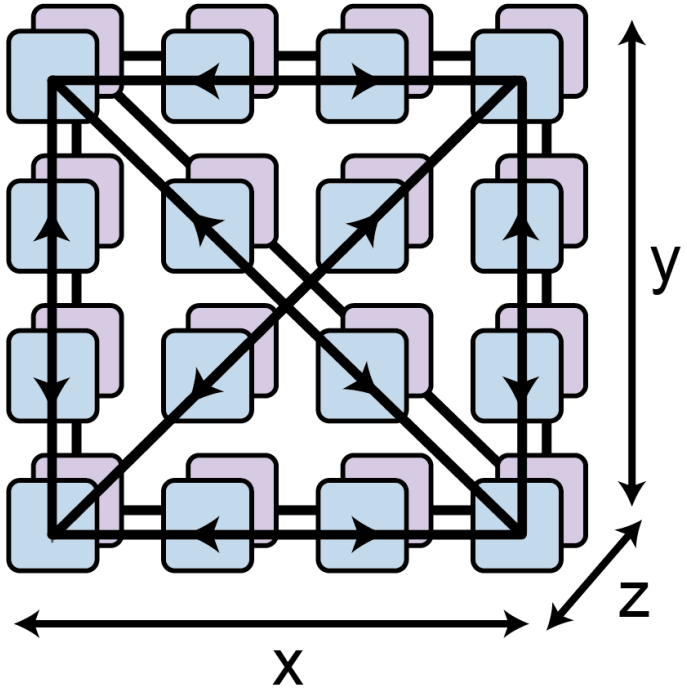
Hypercube (4, 4, 2)



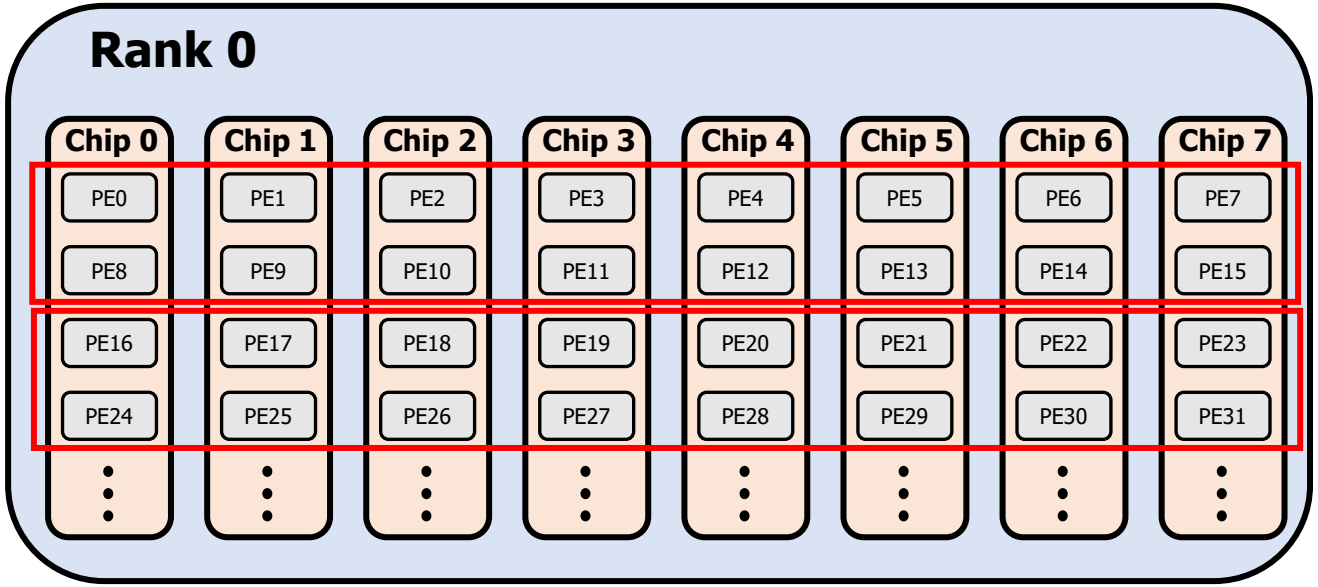
Communication group configuration.

4. The Hypercube Model

- Allows multiple communication invocations
- Support diverse communication groups



Hypercube (4, 4, 2)



Communication group configuration.

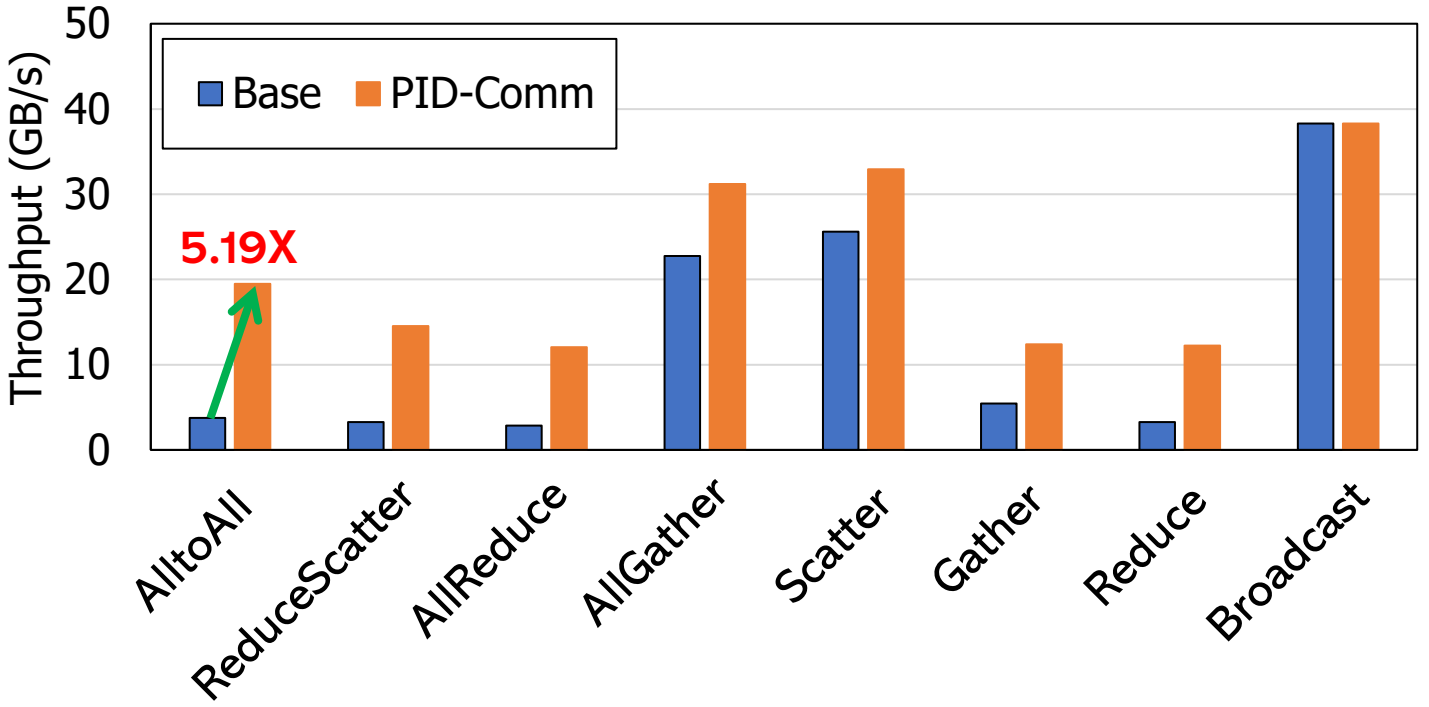
Environment

- **Experimental Setup**
  - Intel Xeon Gold 5125 CPU (Double socket, 10 cores each)
  - 4 Channels of UPMEM DIMMs (1024 PEs)
- **Benchmark Applications**
- **Baseline**
  - SimplePIM (for AllGather, AllReduce, Scatter, Gather, Broadcast)
  - UPMEM SDK based implementation (for AlltoAll, ReduceScatter, Reduce)

App.	Hyper. Dim.	Communication Primitives							
		AlltoAll	Reduce Scatter	All Reduce	All Gather	Scatter	Gather	Reduce	Broad cast
DLRM	3	✓	✓			✓	✓		✓
GNN	2			✓	✓	✓	✓		
BFS	1			✓		✓		✓	
CC	1			✓		✓		✓	
MLP	1		✓			✓		✓	

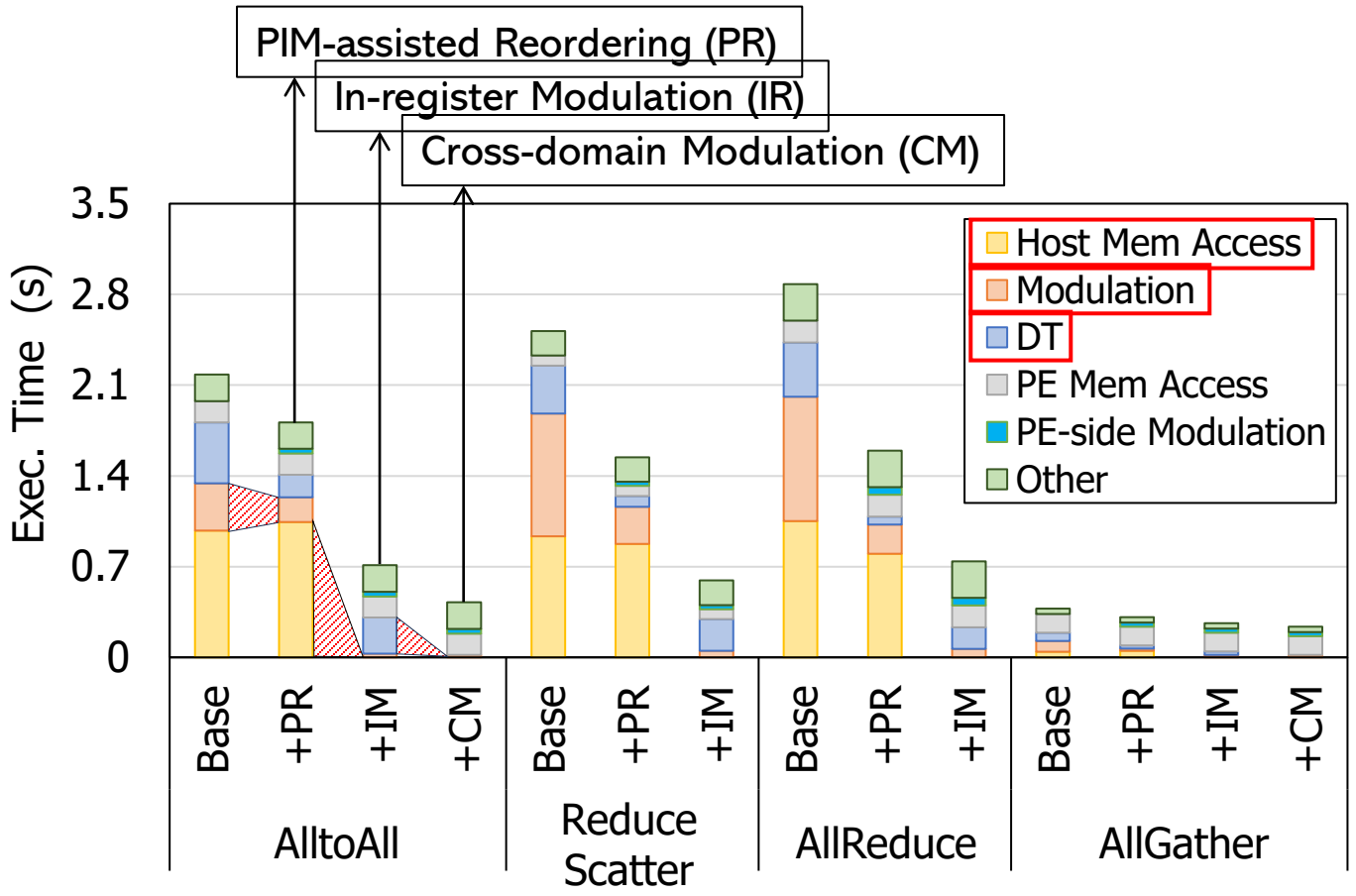
Performance of Primitives

- Up to 5.19x higher throughput compared to PIM baseline
- Geomean Speedup of 2.83X



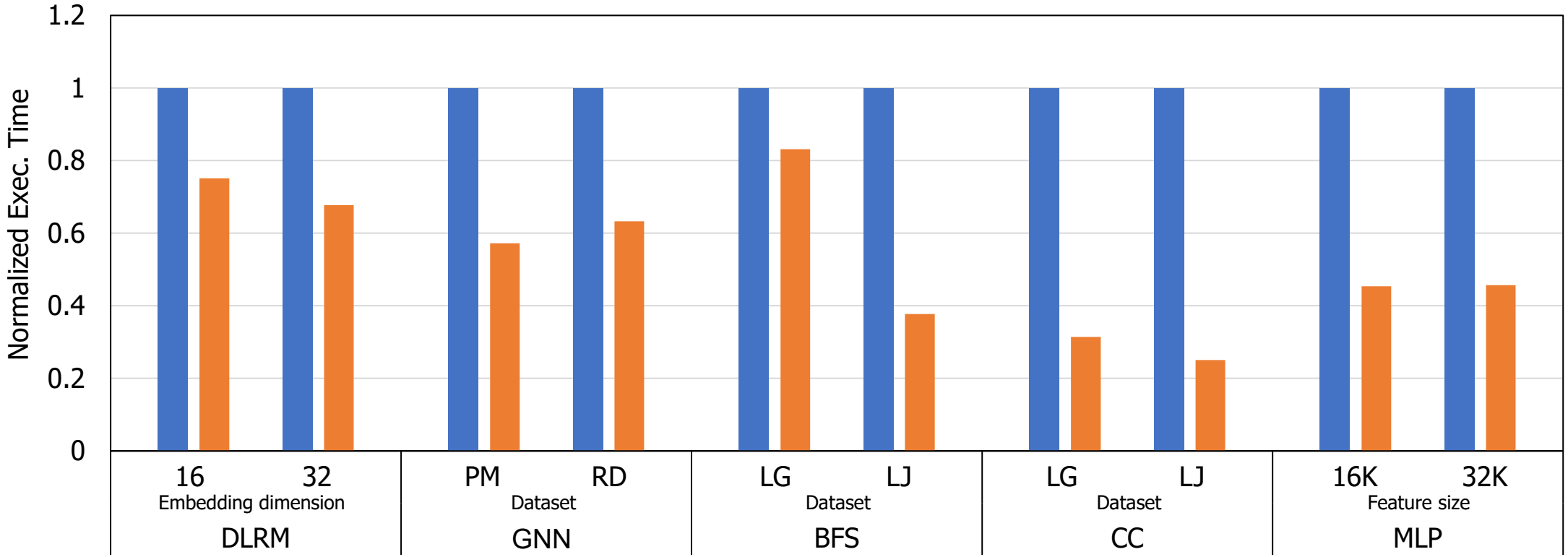
Ablation Study & Breakdown

- Average speedup of 1.48x, 2.03x, and 1.42x for +PR, +IM, and +CM



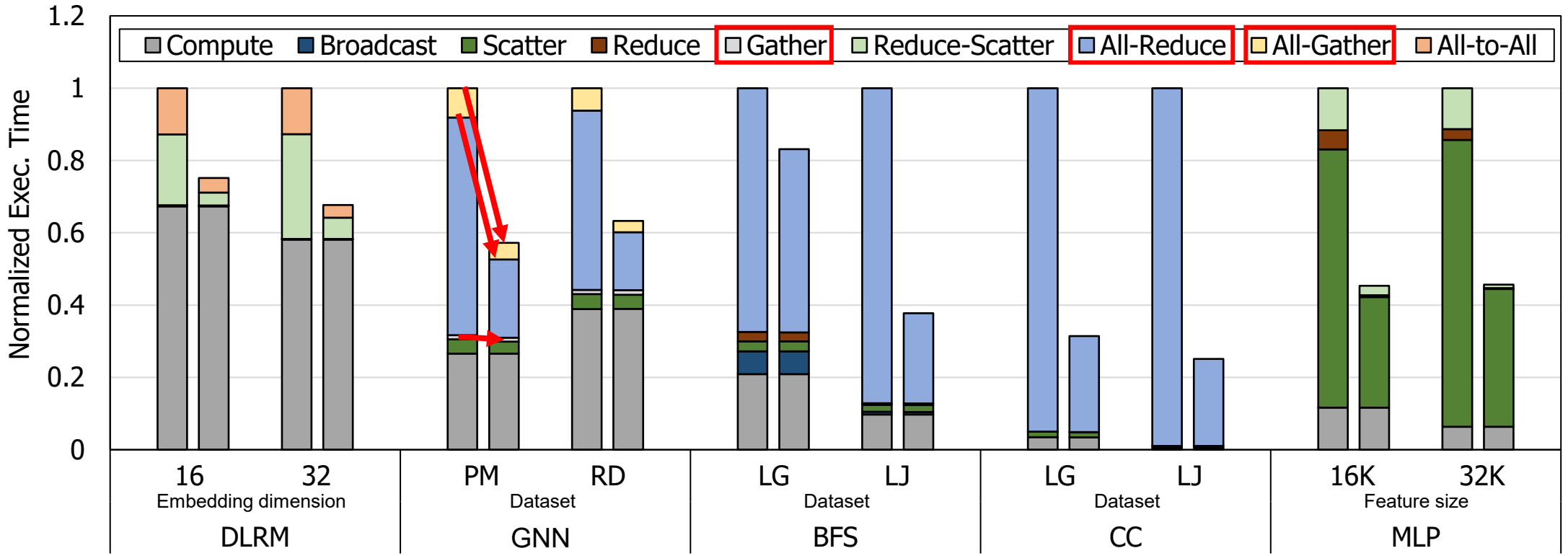
Benchmark Applications

- Evaluated on different datasets / embedding dimensions / feature sizes
- Up to 3.99x speedup compared to conventional communication schemes
- Geo-mean speedup of 1.99x

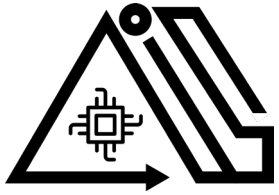


Benchmark Applications

- Evaluated on different datasets / embedding dimensions / feature sizes
- Up to 3.99x speedup compared to conventional communication schemes
- Geo-mean speedup of 1.99x



6. Conclusion



PID-Comm is..

1. **The 1st full-fledged collective communication library for PIM-enabled DIMMs**
   - Supports 8 types of communication primitives (sharing the scope of NCCL)

2. **Provides**
   - Micro-level optimizations to accelerate inter-PE communications
   - A hypercube communication model for flexible communications

3. **Primitives outperform PIM baseline by 2.83x in geomean**

4. **Open source: https://github.com/AIS-SNU/PID-Comm**

Thank you.

Si Ung Noh @ Seoul National University
Email: siung98@snu.ac.kr
Junguk Hong @ Seoul National University
Email: junguk16@snu.ac.kr