Efficient Programming on Heterogeneous Accelerators

Peipei Zhou

2025/05/09, UD Seminar

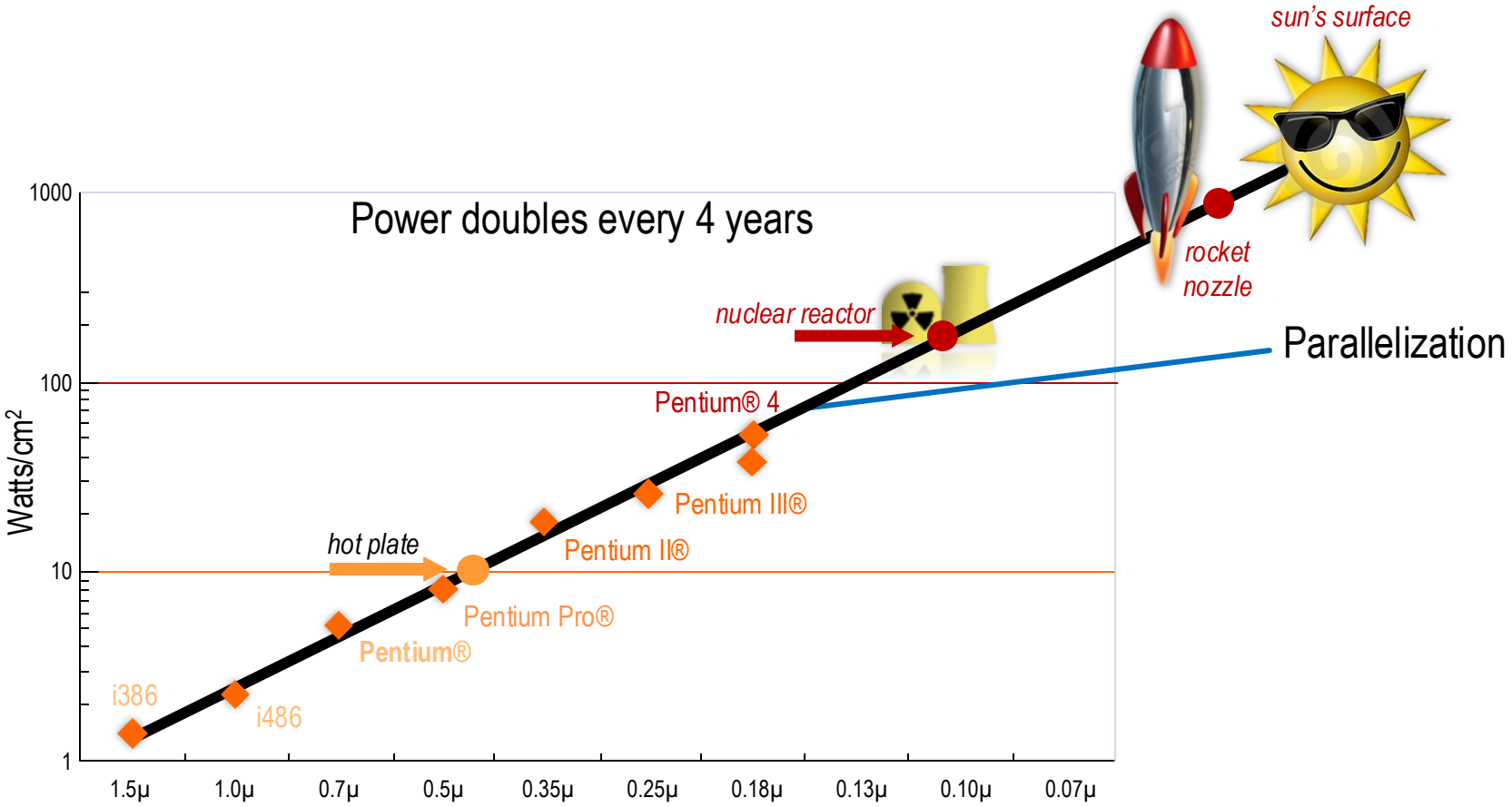
Peipei Zhou, Assistant Professor

- Experience:
  - 2008-2012: ECE B.Sc. @ Southeast University
  - 2012-2019: CS Ph.D. @ UCLA with Prof. Jason Cong
  - 2019-2021: Staff Software Engr. @ Enflame, AI ASIC Startup
  - 2021/09-2024/08: Assistant Professor @ Pitt-ECE
  - 2024/09/01-Now: Assistant Professor @ Brown
- Research:
  - **A**rchitecture: FPGA, AI ASIC, GPU, Chiplet, etc.
  - **A**bstraction: Compiler Support
  - **A**pplication: Artificial Intelligence, Genome (Medical), etc.
- Teaching:
  - G: Reconfigurable Computing in Deep Learning, 2022 Spring, 2023 Spring, 2024 Fall
  - UG: Computer Organization and Architecture, 2022 Fall, 2023 Fall, 2024 Spring
  - UG: Embedded System Design, 2023 Spring

Peipei Zhou's Group
https://peipeizhou-eecs.github.io/

Why Customized Computing?

Challenge with Processor Design – Power Barrier:
From Parallelism to Customization and Specialization

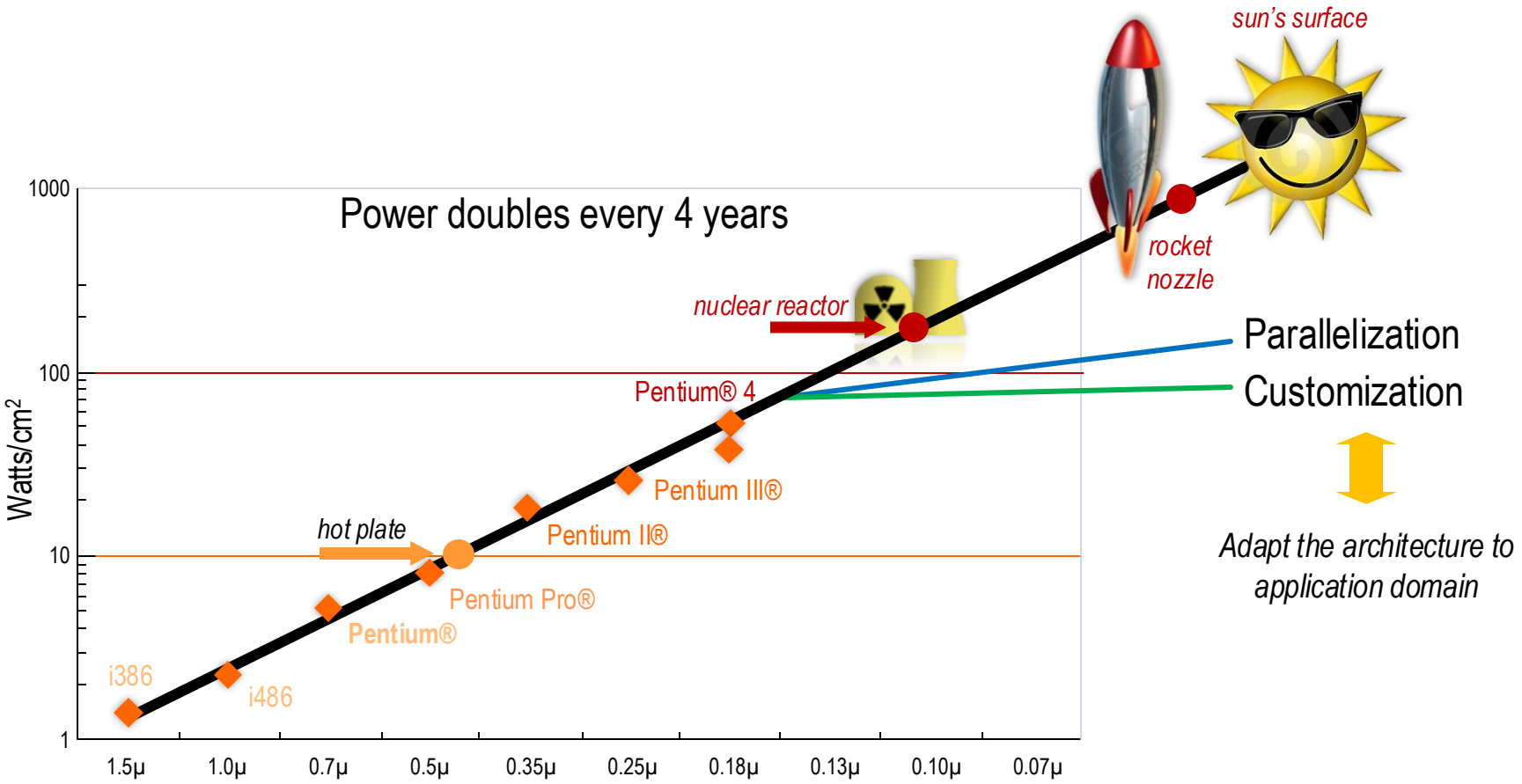


Based on Fred Pollack (Intel) and Michael Taylor (UW)

"New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies", Fred J. Pollack (Intel Fellow)

Why Customized Computing?

Challenge with Processor Design – Power Barrier: From Parallelism to Customization and Specialization



Based on Fred Pollack (Intel) and Michael Taylor (UW)

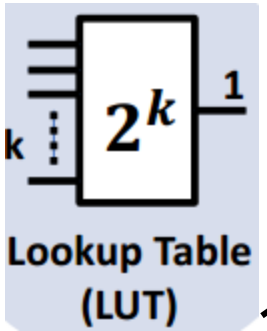
"New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies", Fred J. Pollack (Intel Fellow)

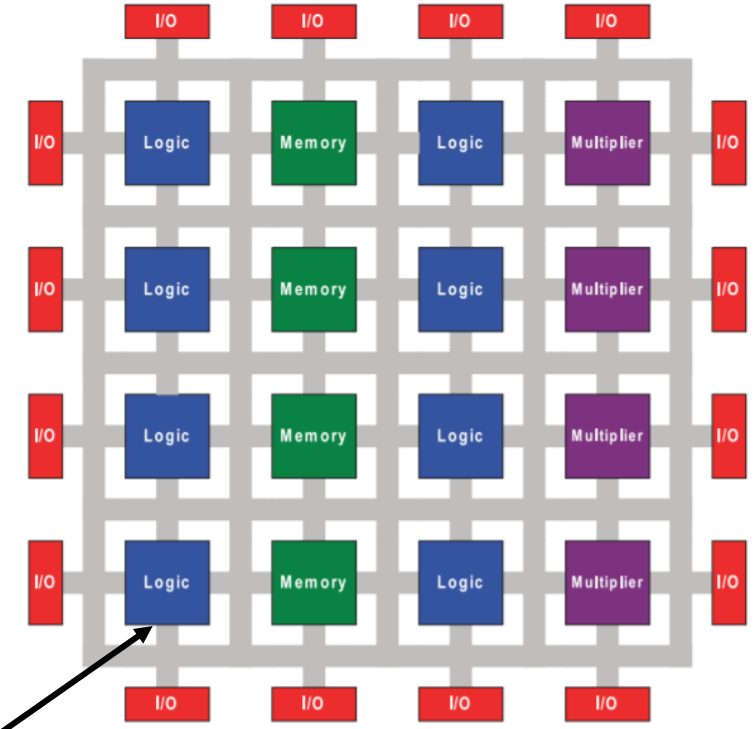
Customized Computing on FPGAs

- FPGA (Field Programmable Gate Arrays)
  - Energy efficiency, 10-100X than CPUs

Customized Computing on FPGAs

- FPGA (Field Programmable Gate Arrays)
  - Energy efficiency, 10-100X than CPUs

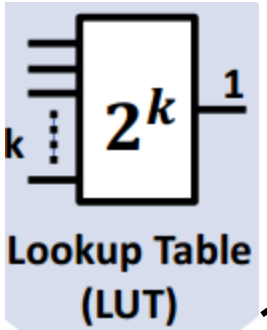


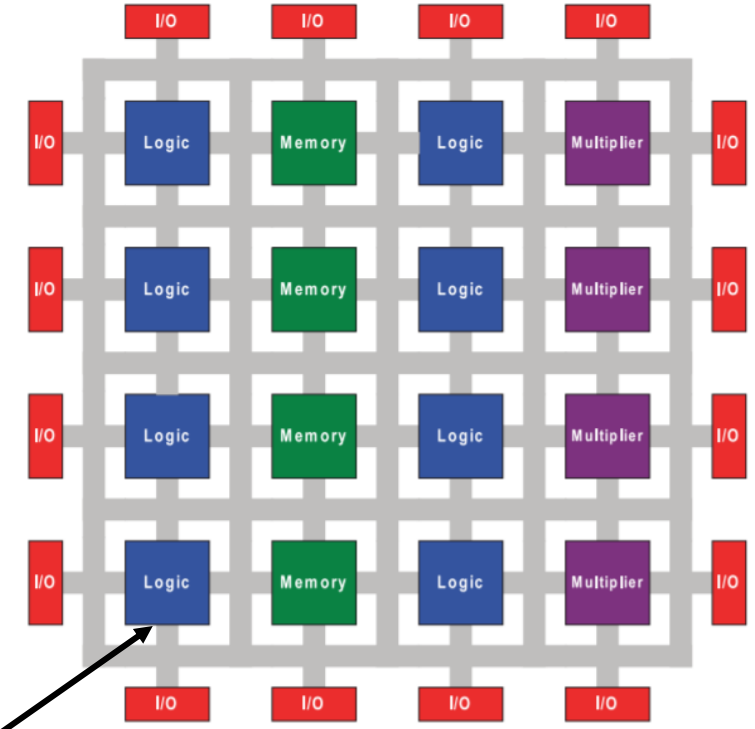


Source: I. Kuon, R. Tessier, J. Rose. FPGA Architecture: Survey and Challenges. 2008.

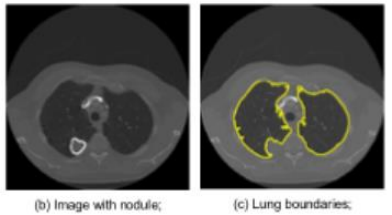
Customized Computing on FPGAs

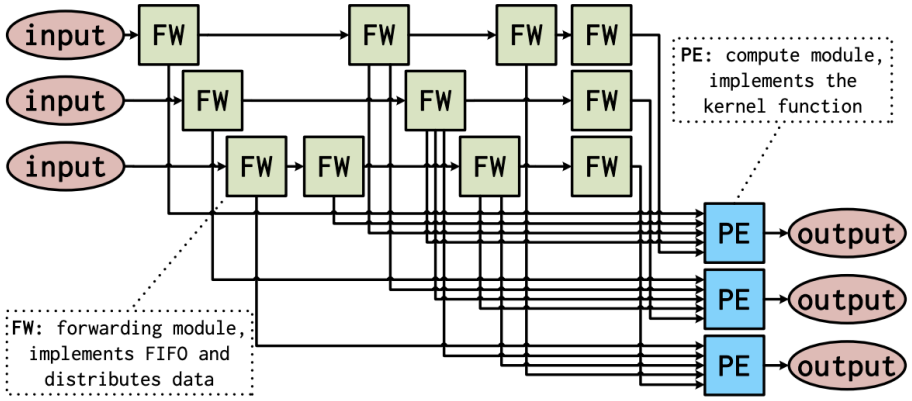
- FPGA (Field Programmable Gate Arrays)
  - Energy efficiency, 10-100X than CPUs
  - Domain-Specific Accelerators (DSA) for various applications: genomics, image/video, GNN, etc.



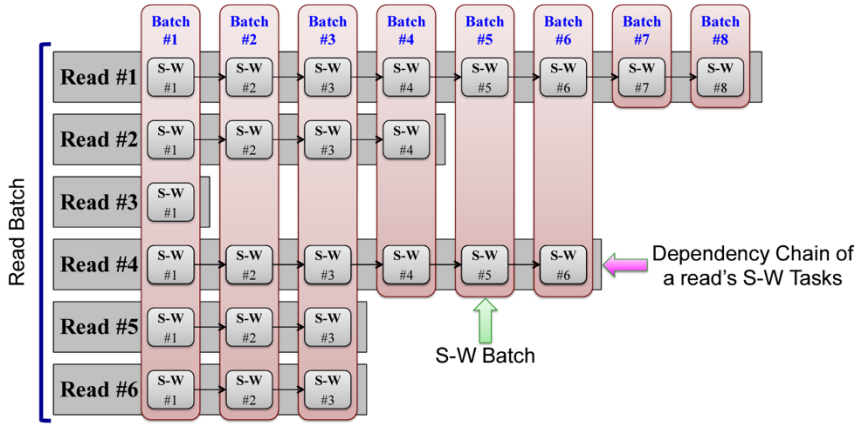


Source: I. Kuon, R. Tessier, J. Rose. FPGA Architecture: Survey and Challenges. 2008.





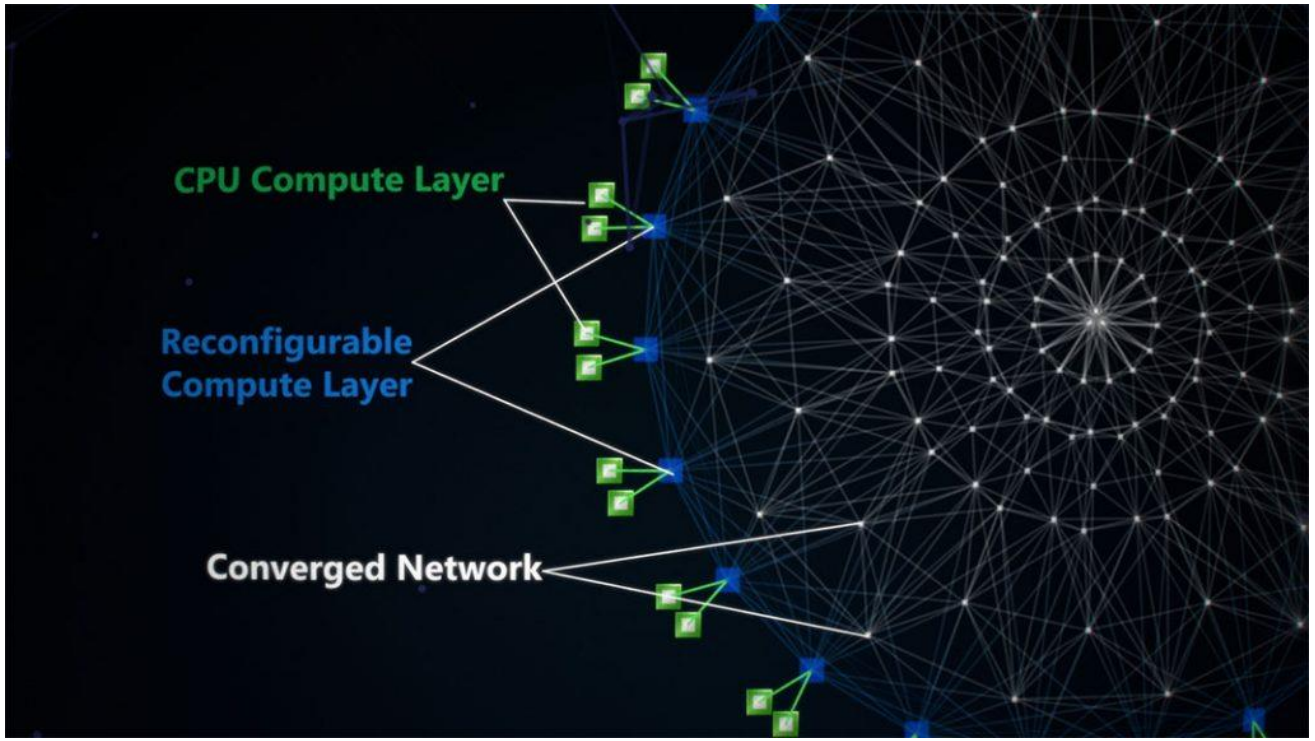
Stencil for Image/Video Processing [ICCAD'18 SODA]

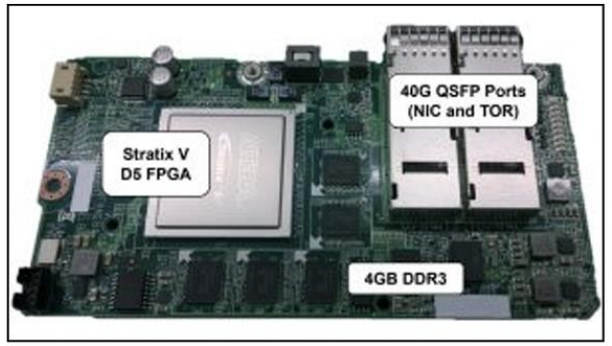


Genomics Acceleration [FPGA'21 Mocha]

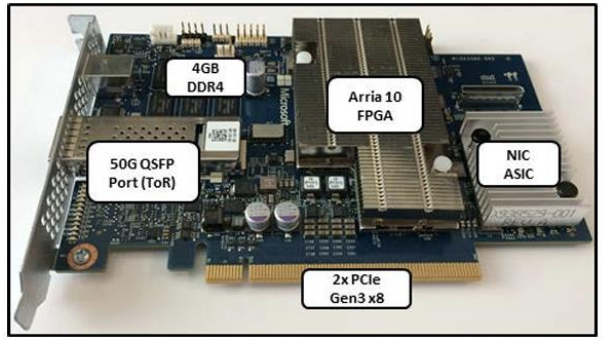
Use of FPGAs trade-off performance for design cost, flexibility, and time-to-silicon

Customized Computing on FPGAs Deployed in Cloud Scale



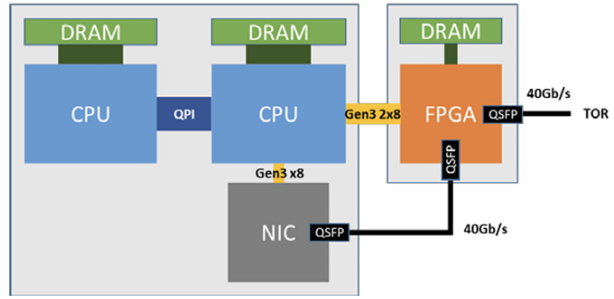


(a) Azure SmartNIC Gen1, 40GbE w/ external NIC



(b) Azure SmartNIC Gen2, 50GbE w/ on-board NIC

Microsoft programmable hardware/software "fabric" on more than 5,760 Microsoft datacenter servers running Intel Xeon processors and Altera FPGA chips.
Apps: Bing Search, Azure Machine Learning, SmartNIC



(c) *bump-in-the-wire* architecture

ISCA'14, IEEE Micro'15, MICRO'16, NSDI'18, NeurIPS'20

Microsoft

Customized Computing on FPGAs

Can Every Programmer Easily Design DSAs?

Customized Computing on FPGAs

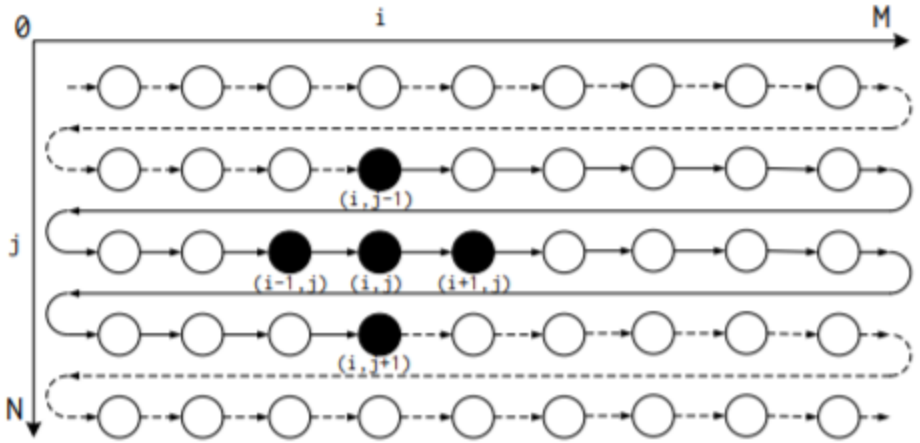
Can Every Programmer Easily Design DSAs?
Yes! We can create circuits from C/C++ with high-level synthesis (HLS)

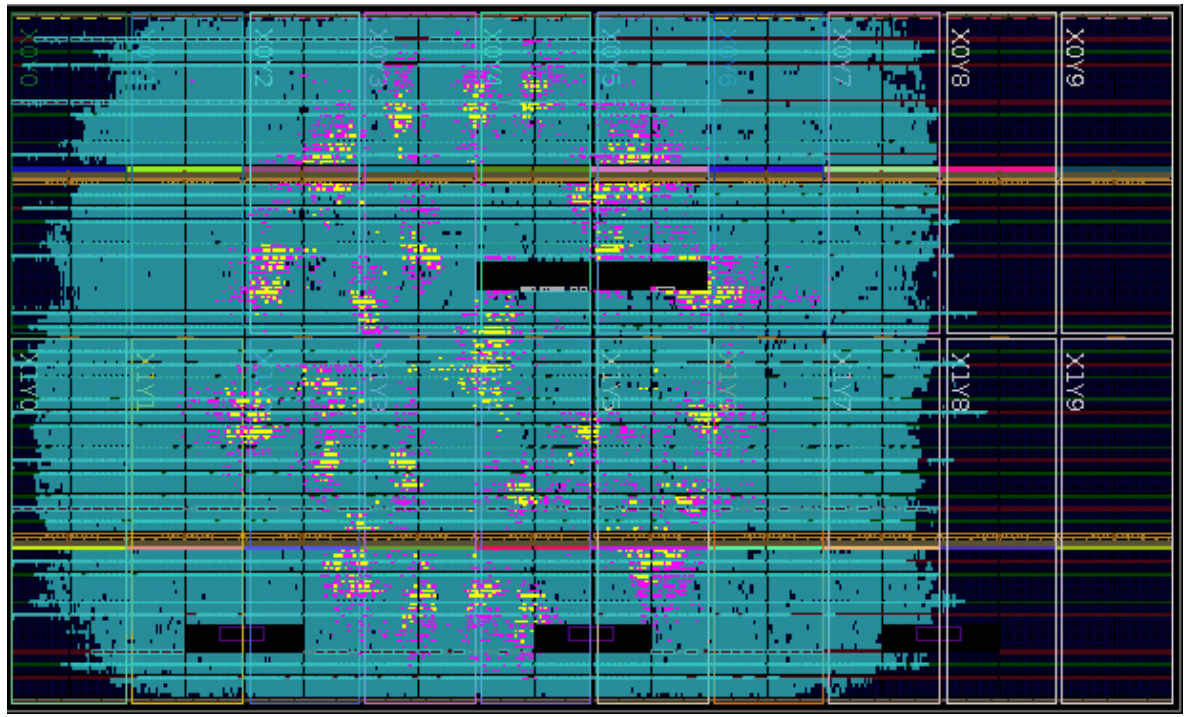
Customized Computing on FPGAs

Can Every Programmer Easily Design DSAs?
Yes! We can create circuits from C/C++ with high-level synthesis (HLS)

Example Code:

```
void blur(float input[N][M], float output[N][M])
{
    for(int j = 1; j < N-1; ++j)
        for(int i = 1; i < M-1; ++i)
            output[j][i] = (  input[j-1][i]+
                input[j][i-1]+input[j  ][i]+
                input[j][i+1]+input[j+1][i])*0.2f;
}
```

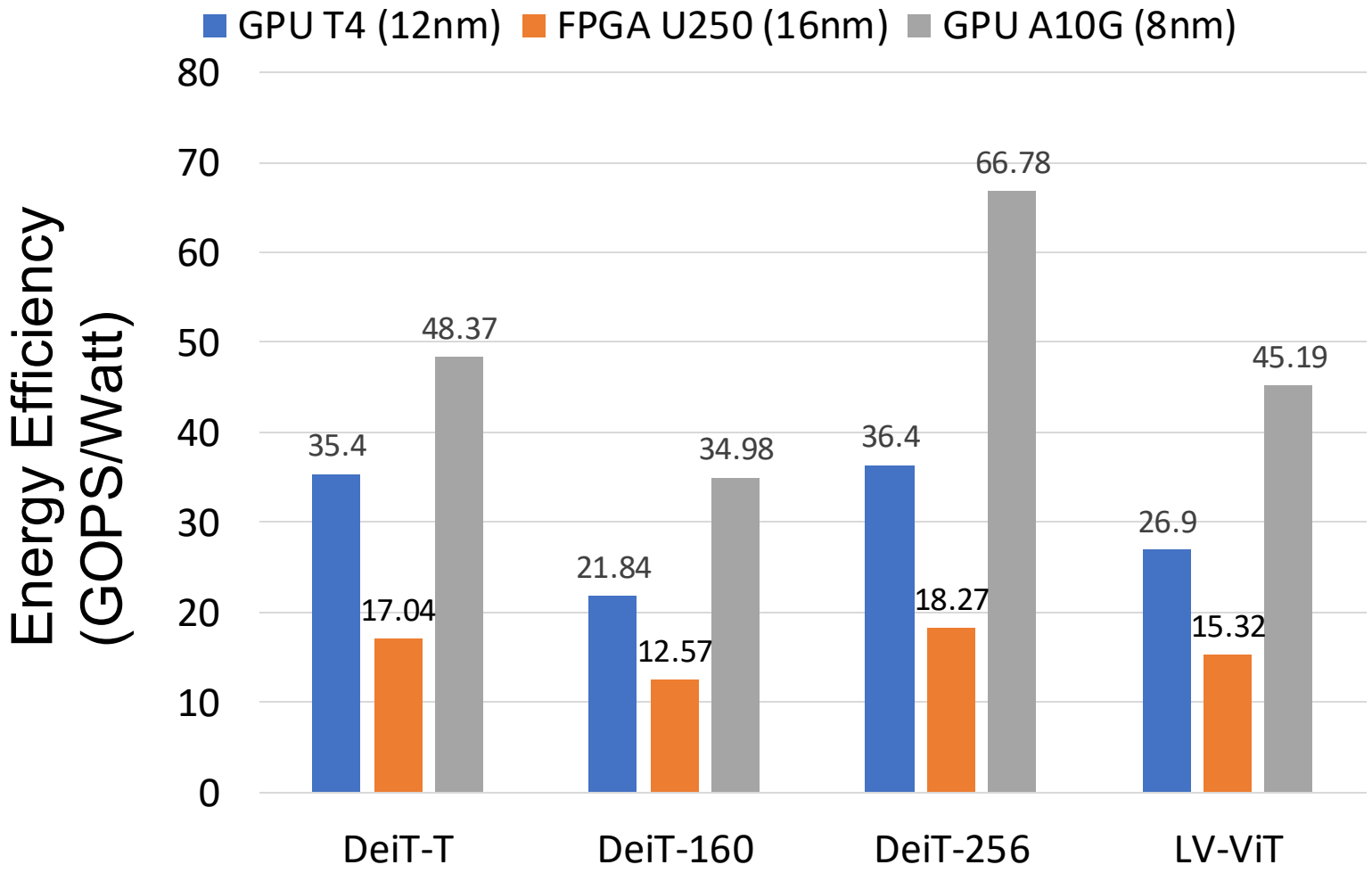




Use of FPGAs trade-off performance for design cost, flexibility, and time-to-silicon

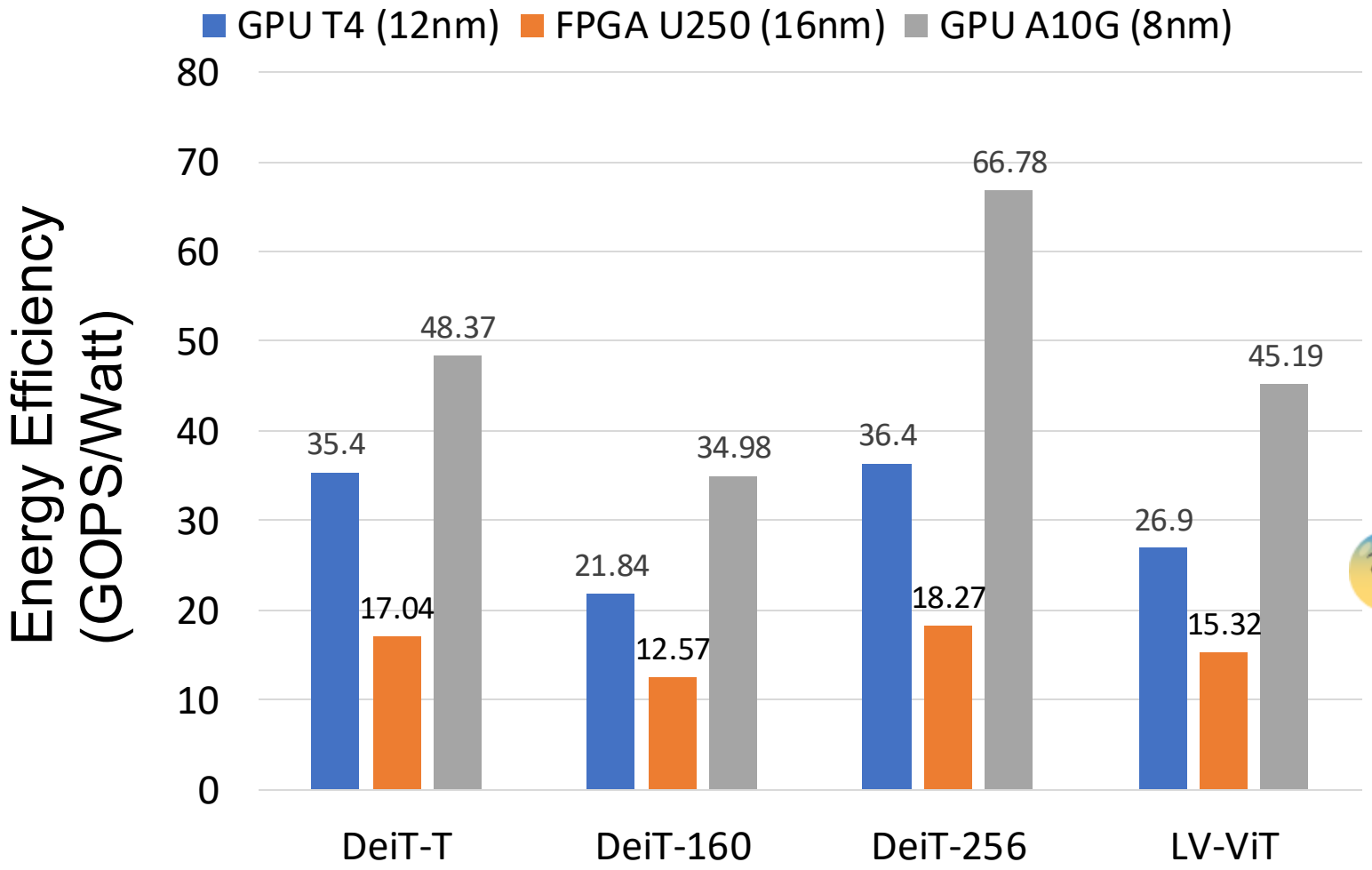
Re-examine FPGA vs. GPU vs. CPU: Deep Learning Inference

Vision Transformer, INT8 inference, batch = 6
Energy efficiency comparison



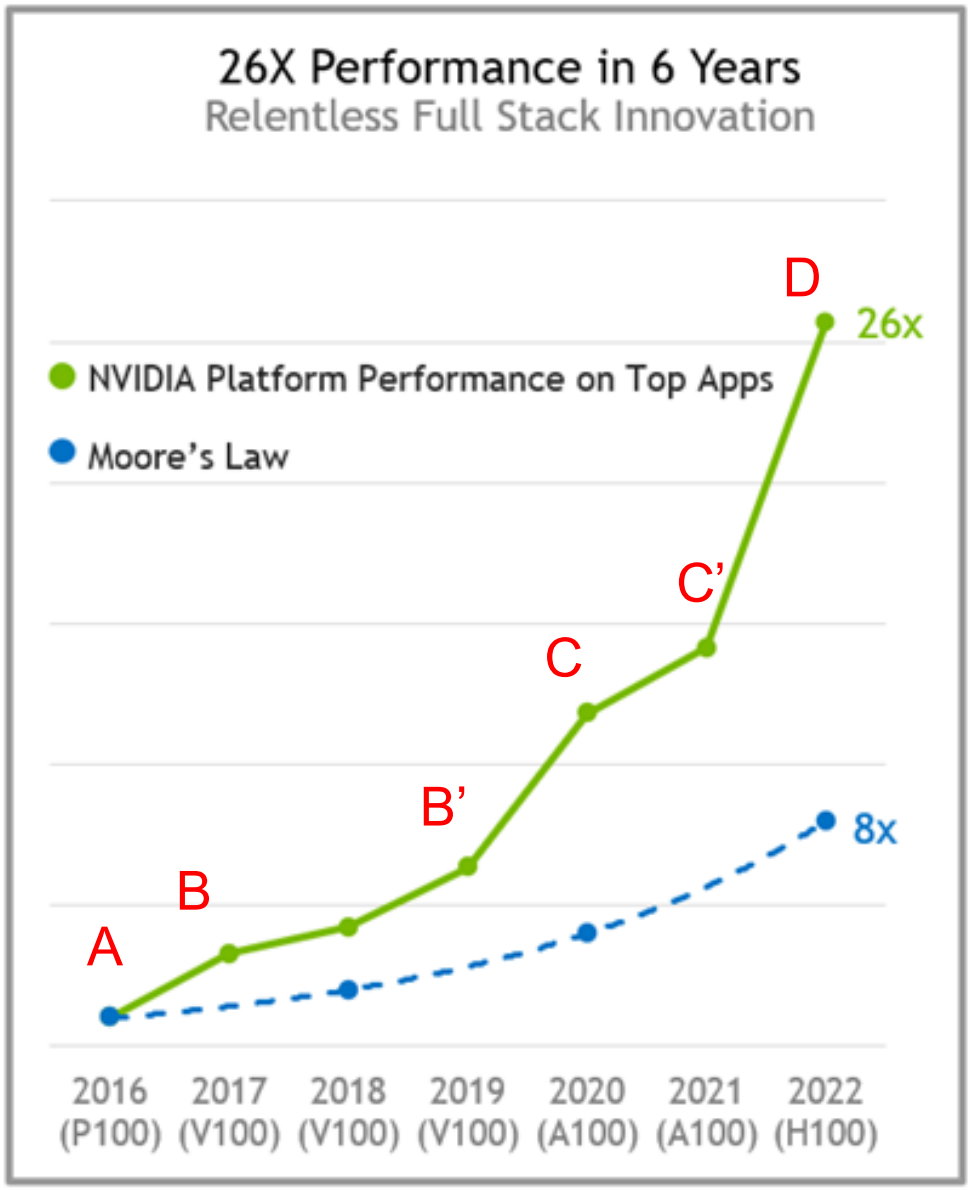
Re-examine FPGA vs. GPU vs. CPU: Deep Learning Inference

Vision Transformer, INT8 inference, batch = 6
Energy efficiency comparison

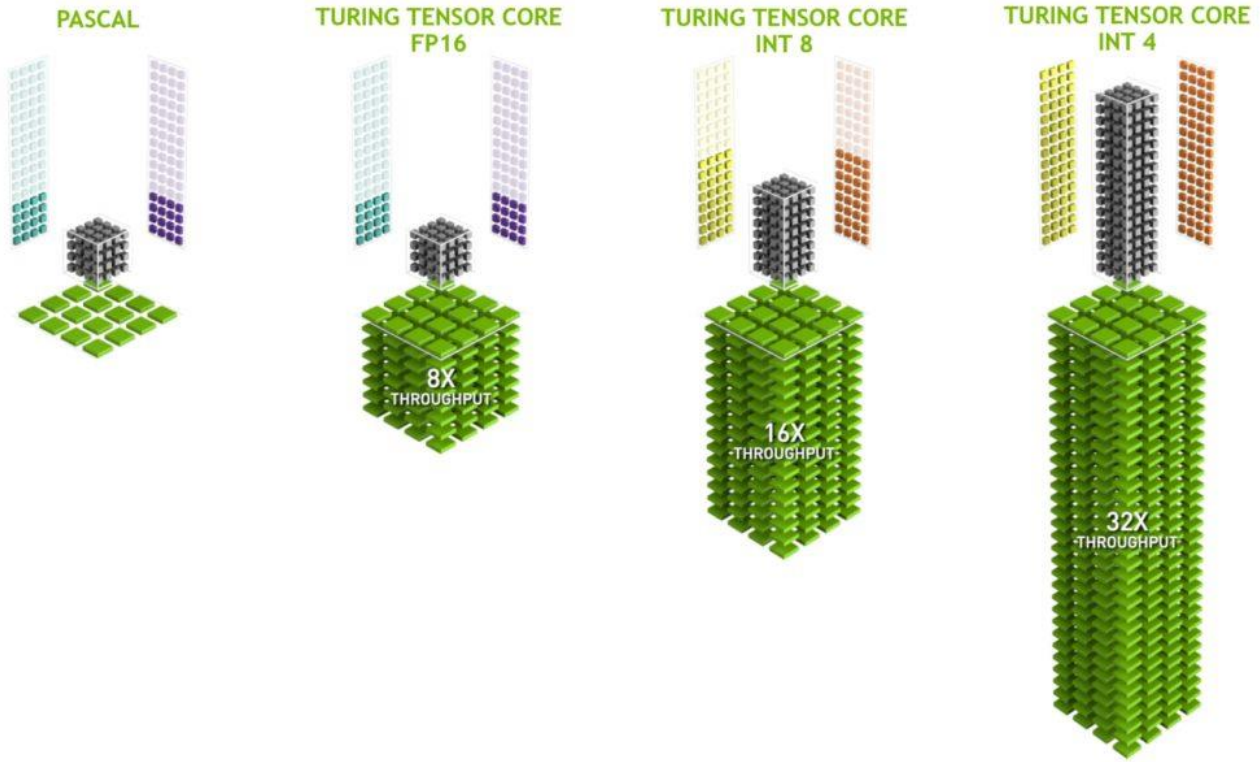


Why FPGA becomes the most inefficient platform in deep learning inference?

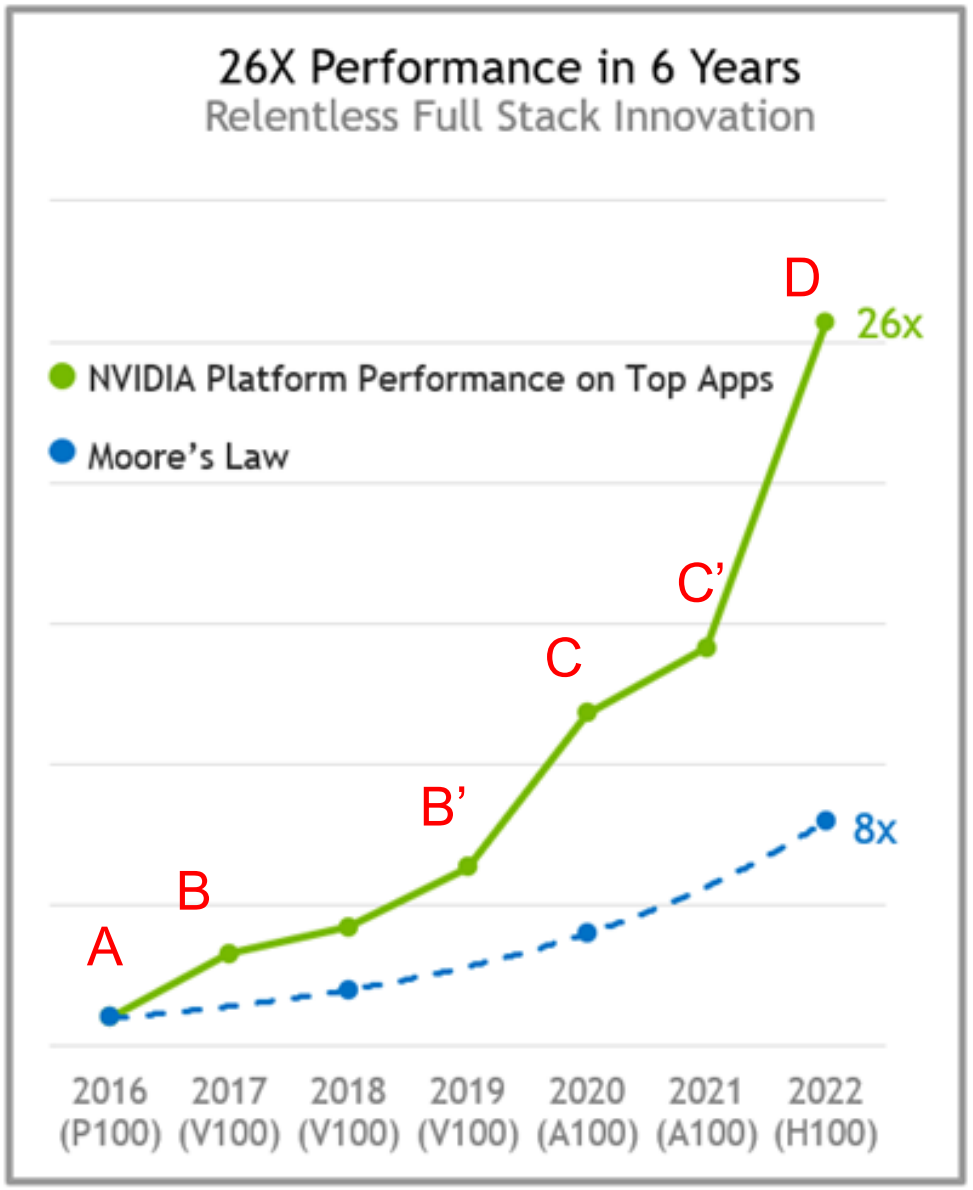
Re-examine FPGA vs. GPU vs. CPU: Deep Learning Inference



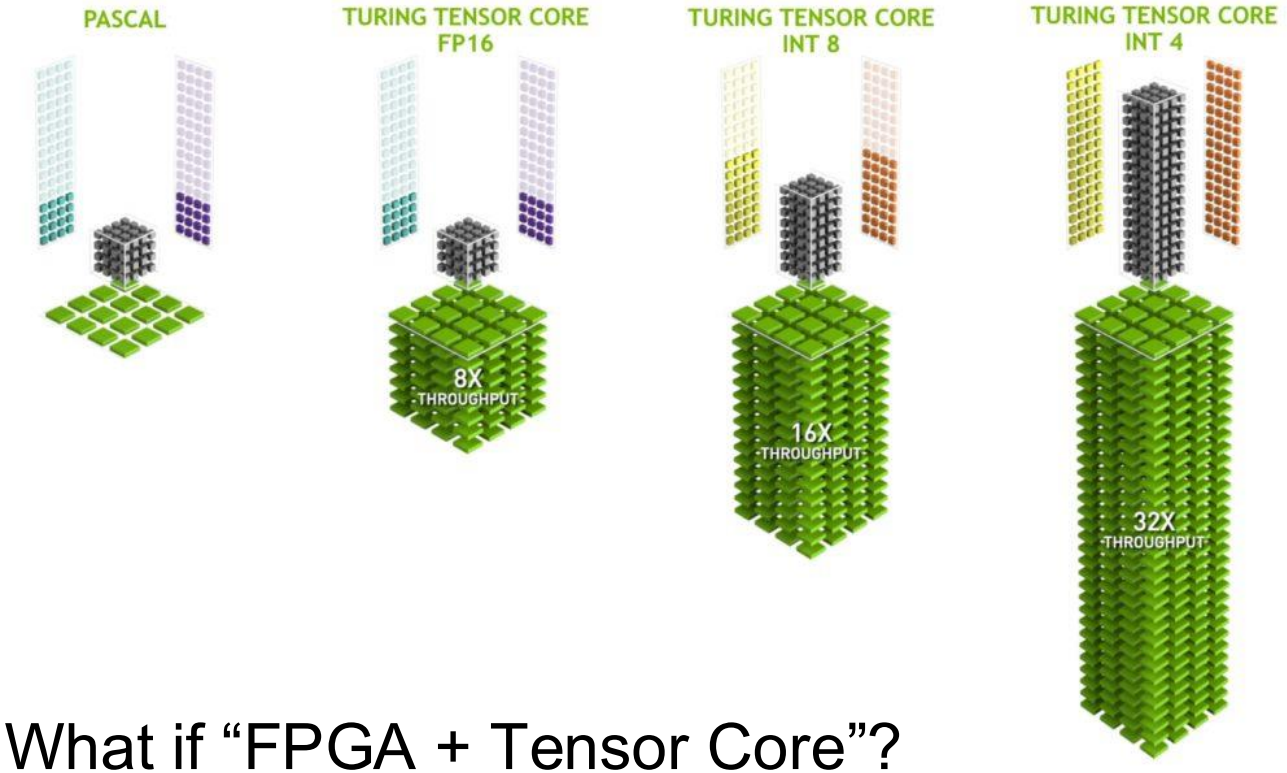
A -> B, Volta GPU introduces Tensor Core
B'-> C, Tensor Core Upgrade
C'-> D, Tensor Core Upgrade



Re-examine FPGA vs. GPU vs. CPU: Deep Learning Inference



A -> B, Volta GPU introduces Tensor Core
B'-> C, Tensor Core Upgrade
C'-> D, Tensor Core Upgrade

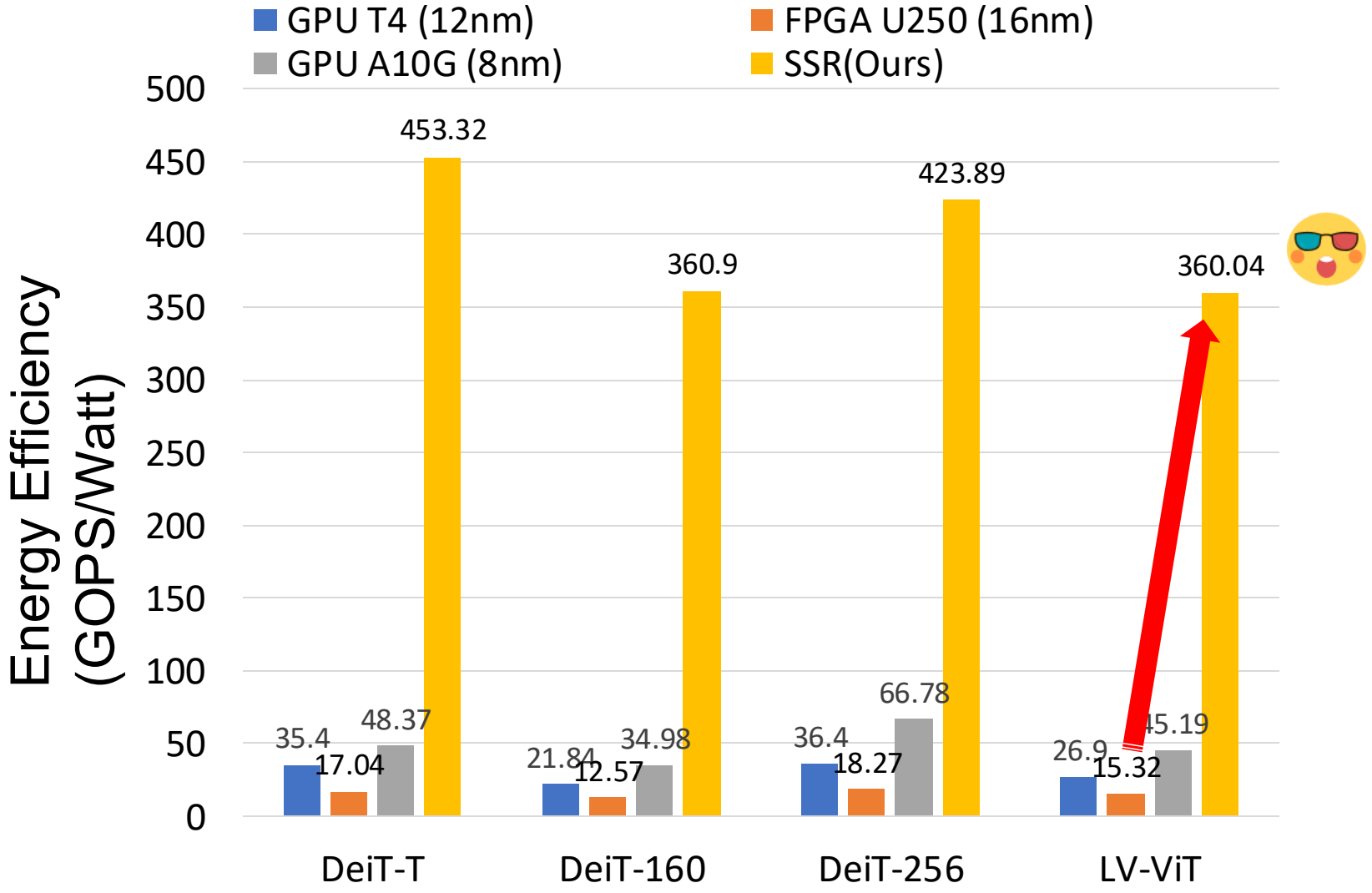


What if "FPGA + Tensor Core"?

“Game Changer” in Deep Learning Inference

Vision Transformer, INT8 inference, batch = 6

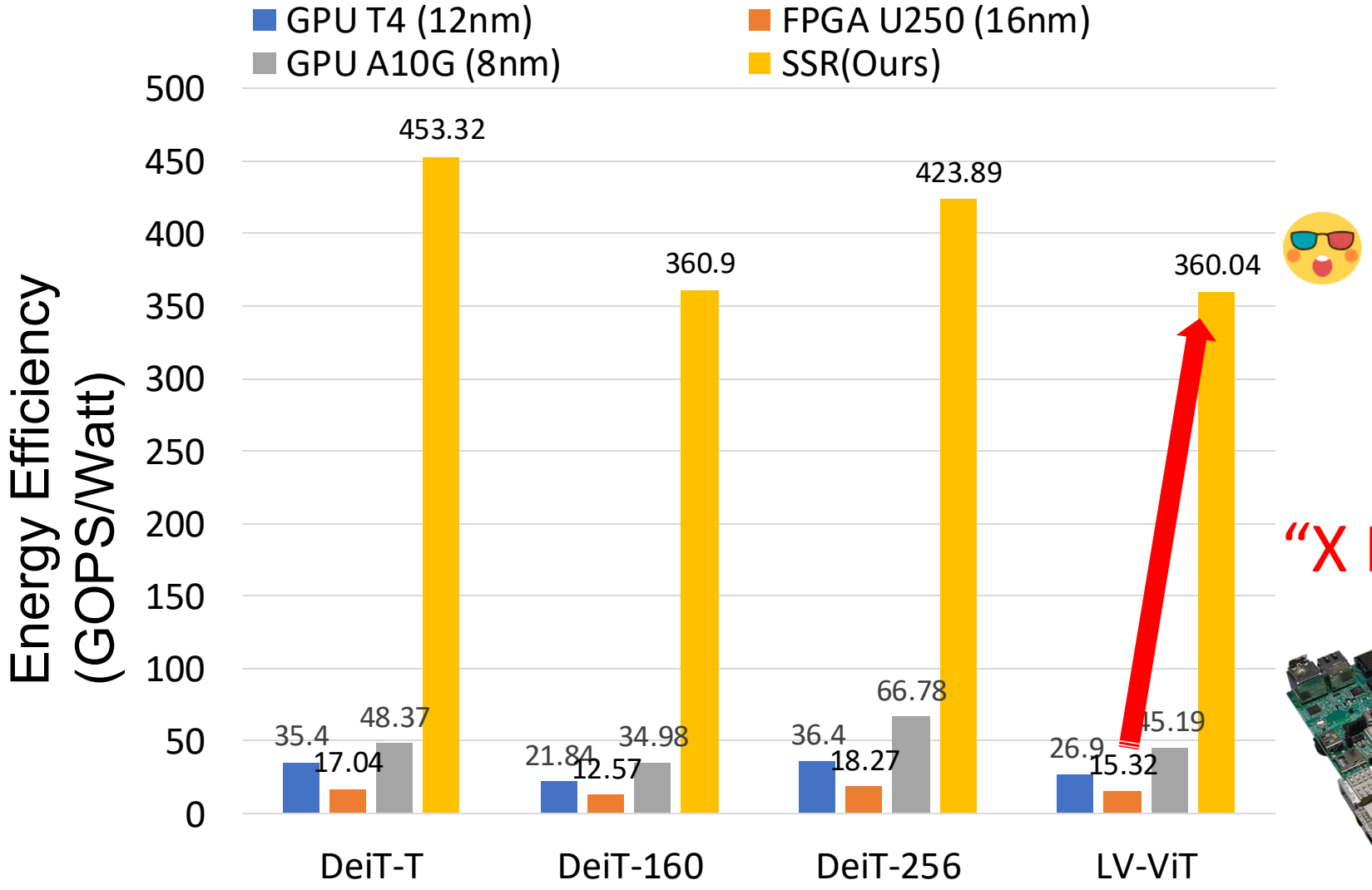
Energy efficiency comparison



“Game Changer” in Deep Learning Inference

Vision Transformer, INT8 inference, batch = 6

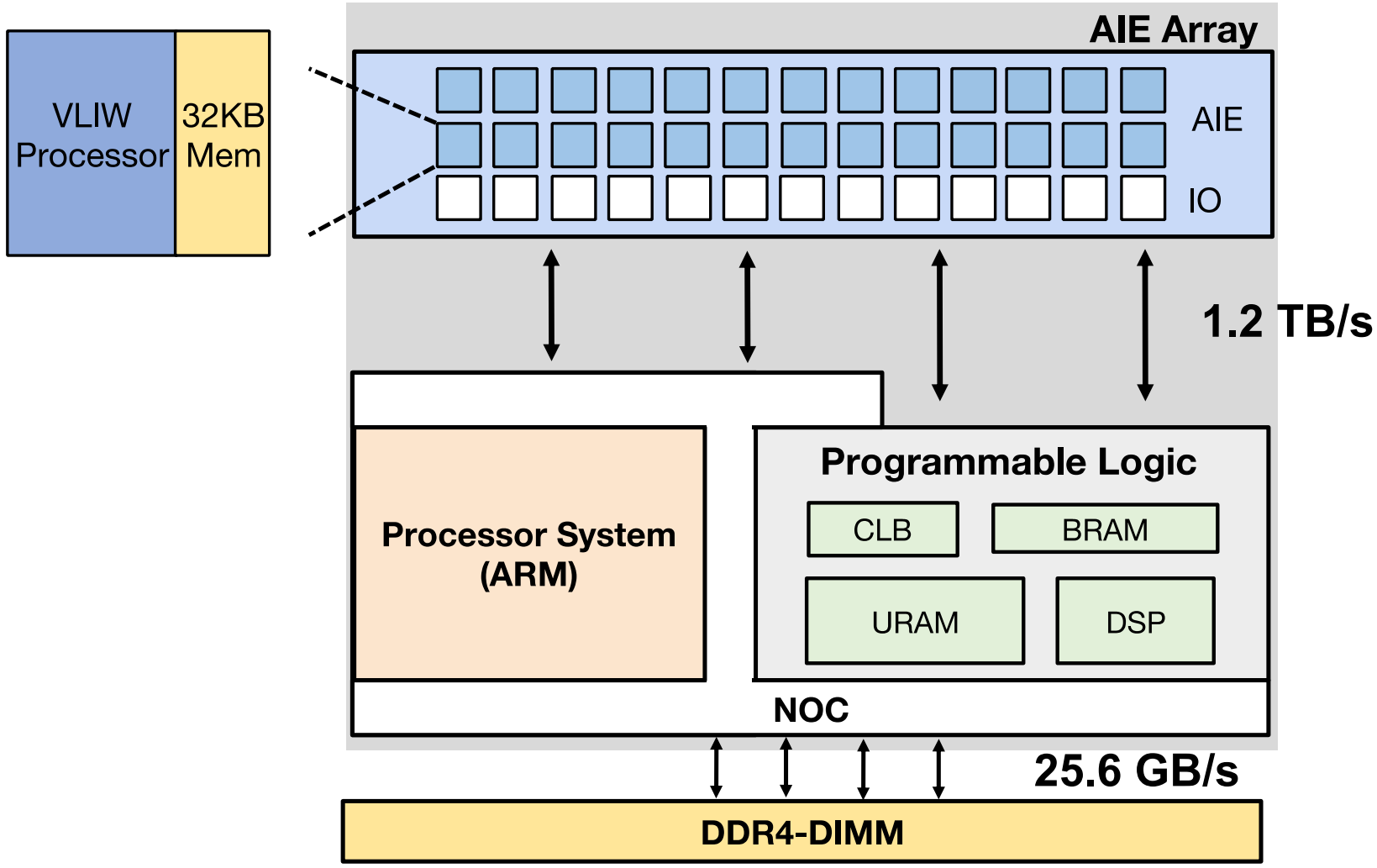
Energy efficiency comparison



“X FPGA”

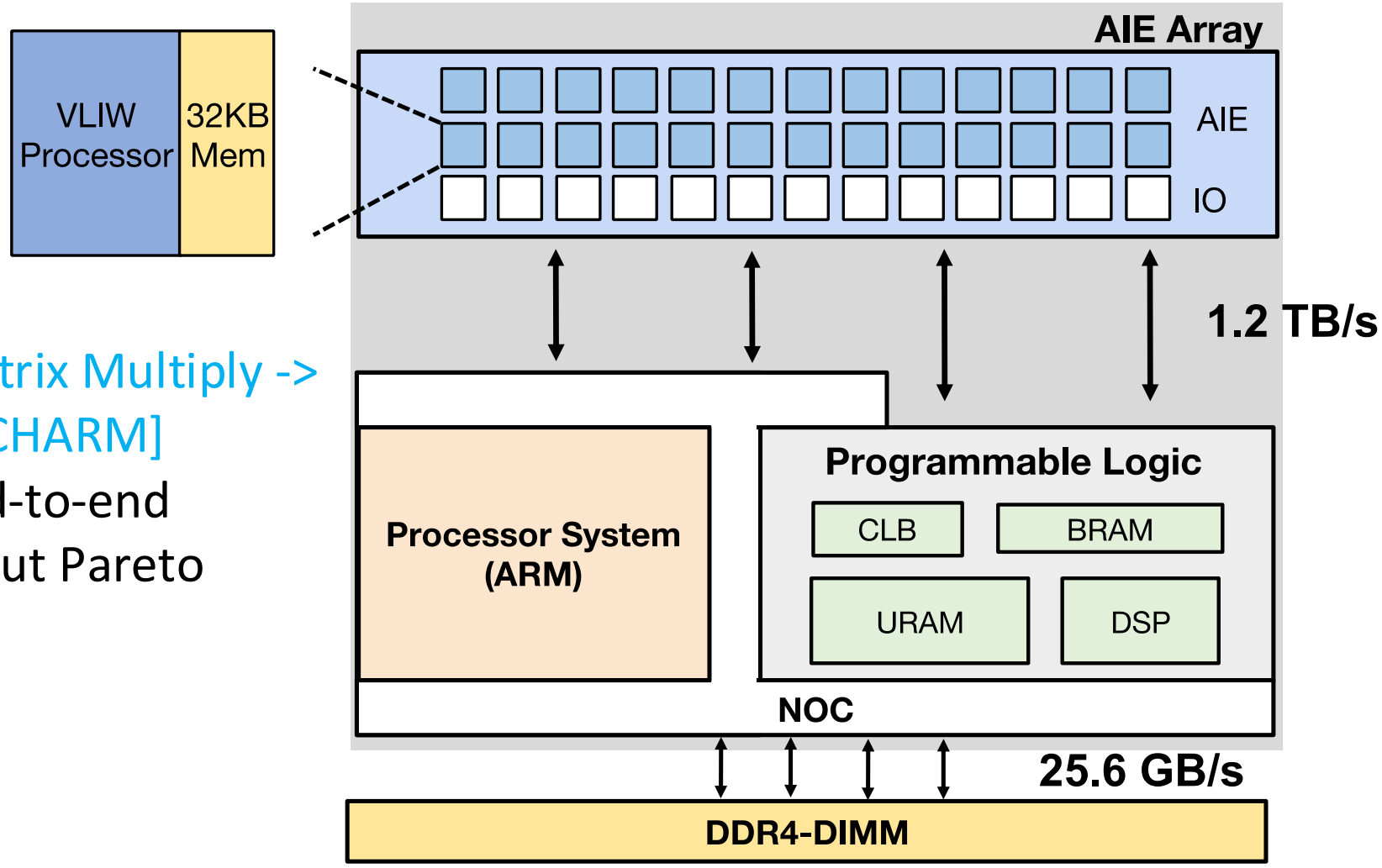
“X FPGA” => Versal ACAP Architecture

AMD ACAP
Heterogeneous SoC
FPGA + AI Tensor Cores + CPU



“X FPGA” => Versal ACAP Architecture

AMD ACAP
Heterogeneous SoC
FPGA + AI Tensor Cores + CPU

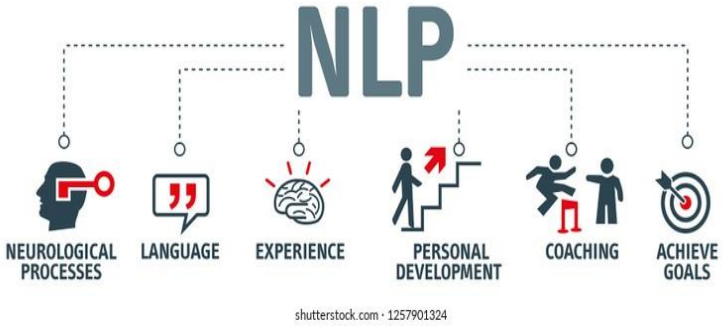


- Composing Diverse Accs for Matrix Multiply -> Throughput Optimal [FPGA’23 CHARM]
- Composing Diverse Accs for End-to-end Applications, Latency-Throughput Pareto [FPGA’24 SSR]

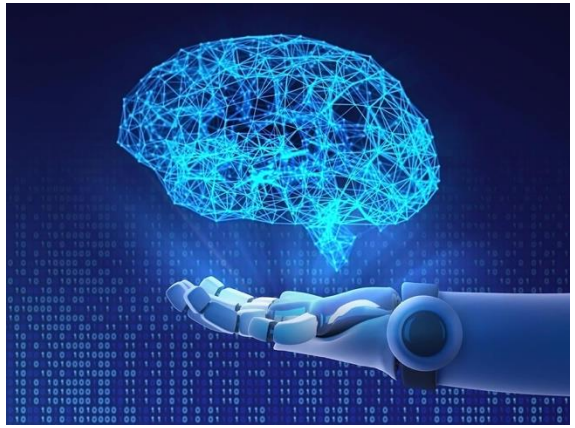
Background Story

Image Recognition

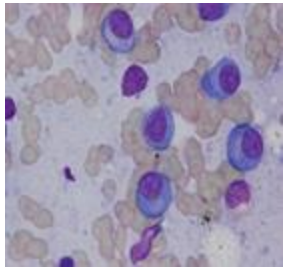
Natural Language processing



Recommendation Systems



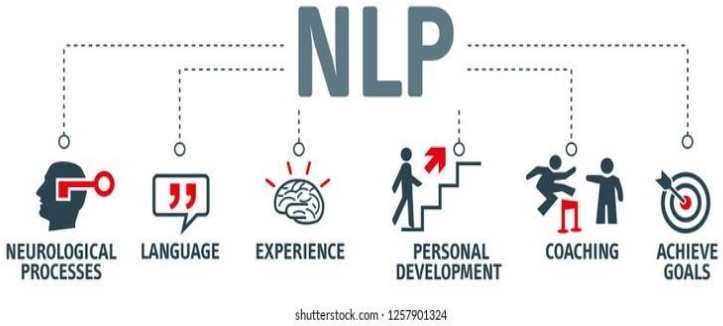
Health: Medical Image Segmentation



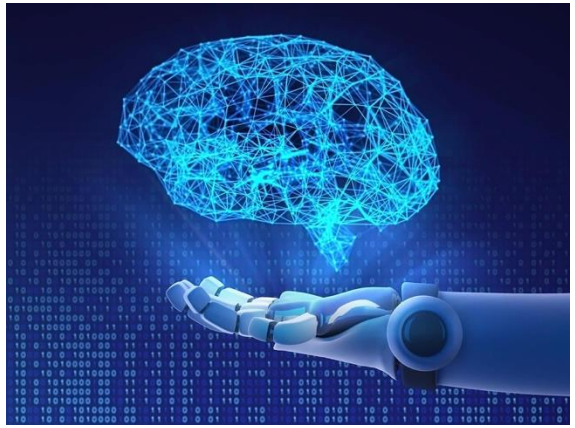
Background Story

Image Recognition

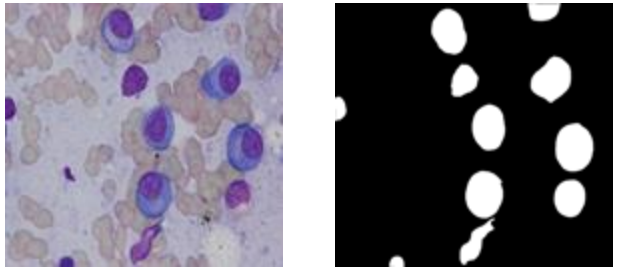
Natural Language processing



Recommendation Systems



Health: Medical Image Segmentation



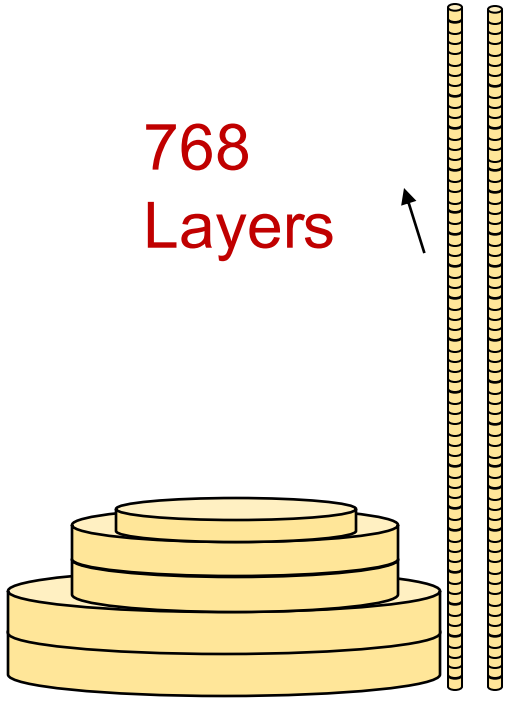
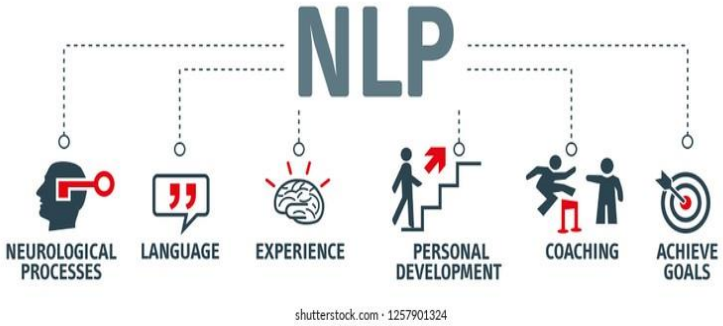


Image Classification: Vision Transformer

Background Story

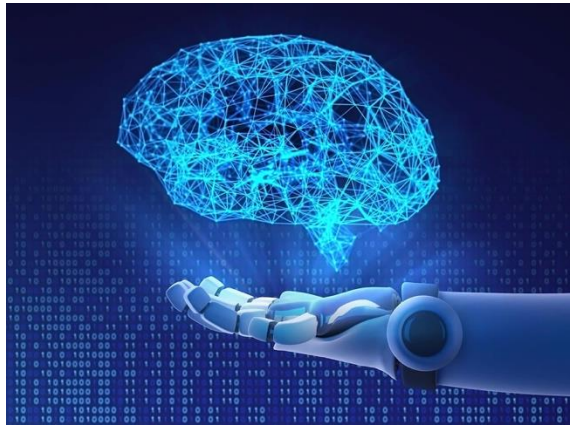
Image Recognition

Natural Language processing

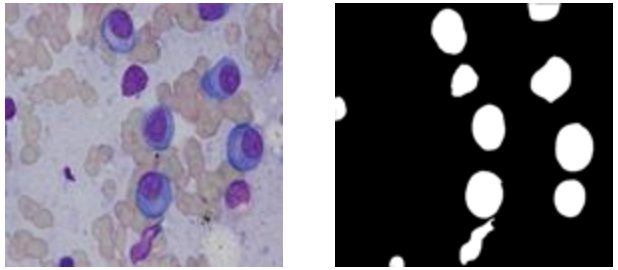


Before CHARM: 49.5 GFLOPS

Recommendation Systems



Health: Medical Image Segmentation



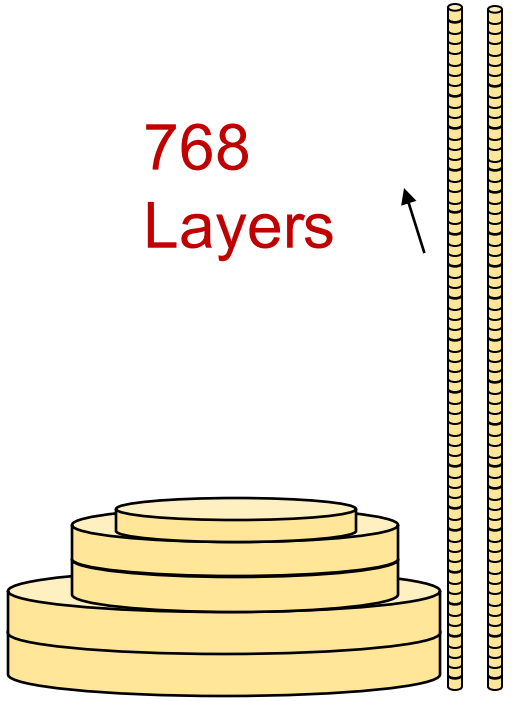
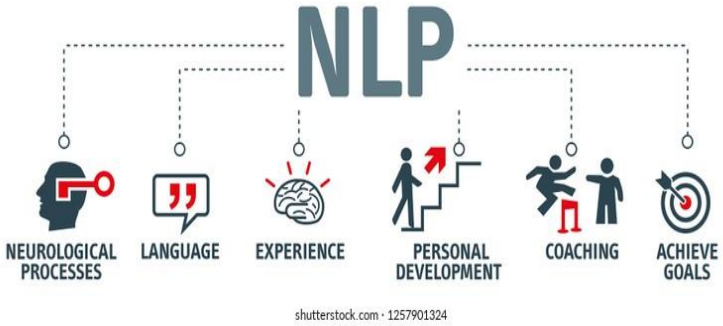


Image Classification: Vision Transformer

Background Story

Image Recognition

Natural Language processing

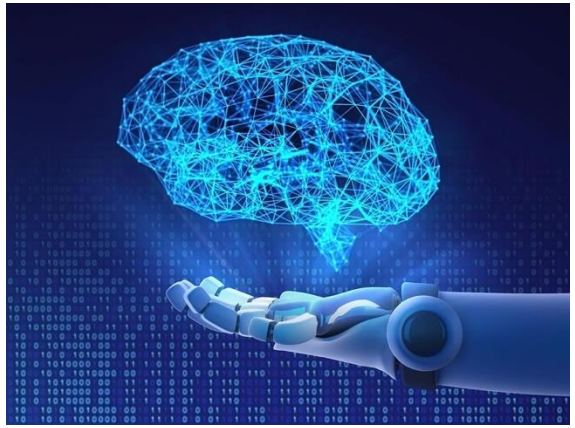


After CHARM: 1609 GFLOPS

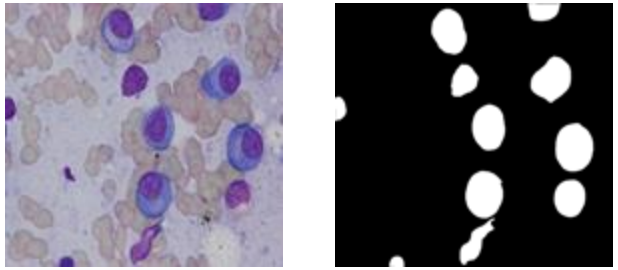
32.5x

Before CHARM: 49.5 GFLOPS

Recommendation Systems



Health: Medical Image Segmentation



768 Layers

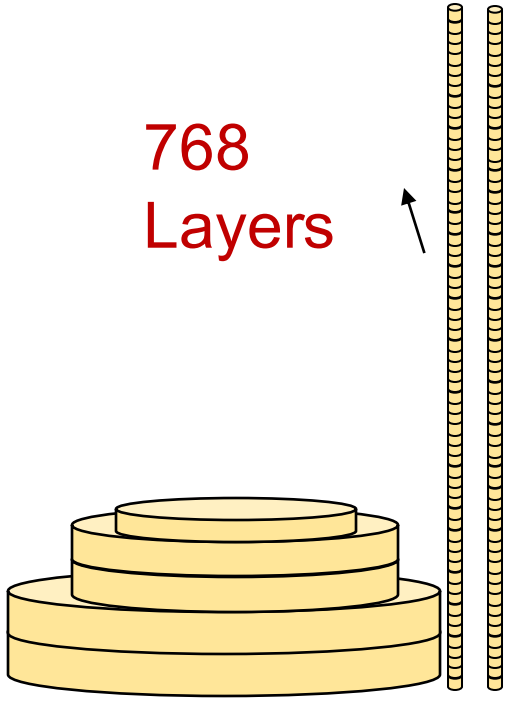


Image Classification: Vision Transformer

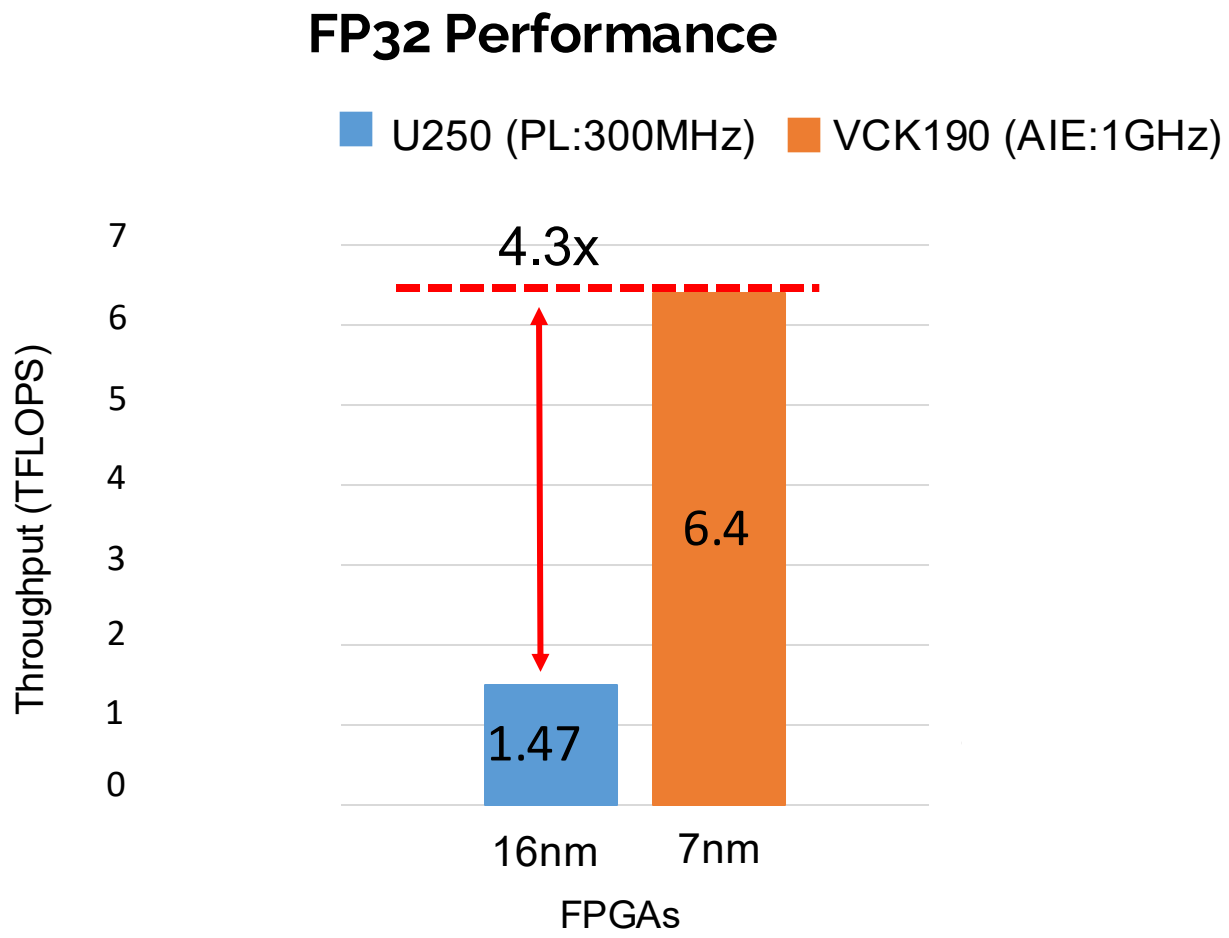
Challenge 1

Challenge 1

- **The computation scaling vs. off-chip communication scaling**

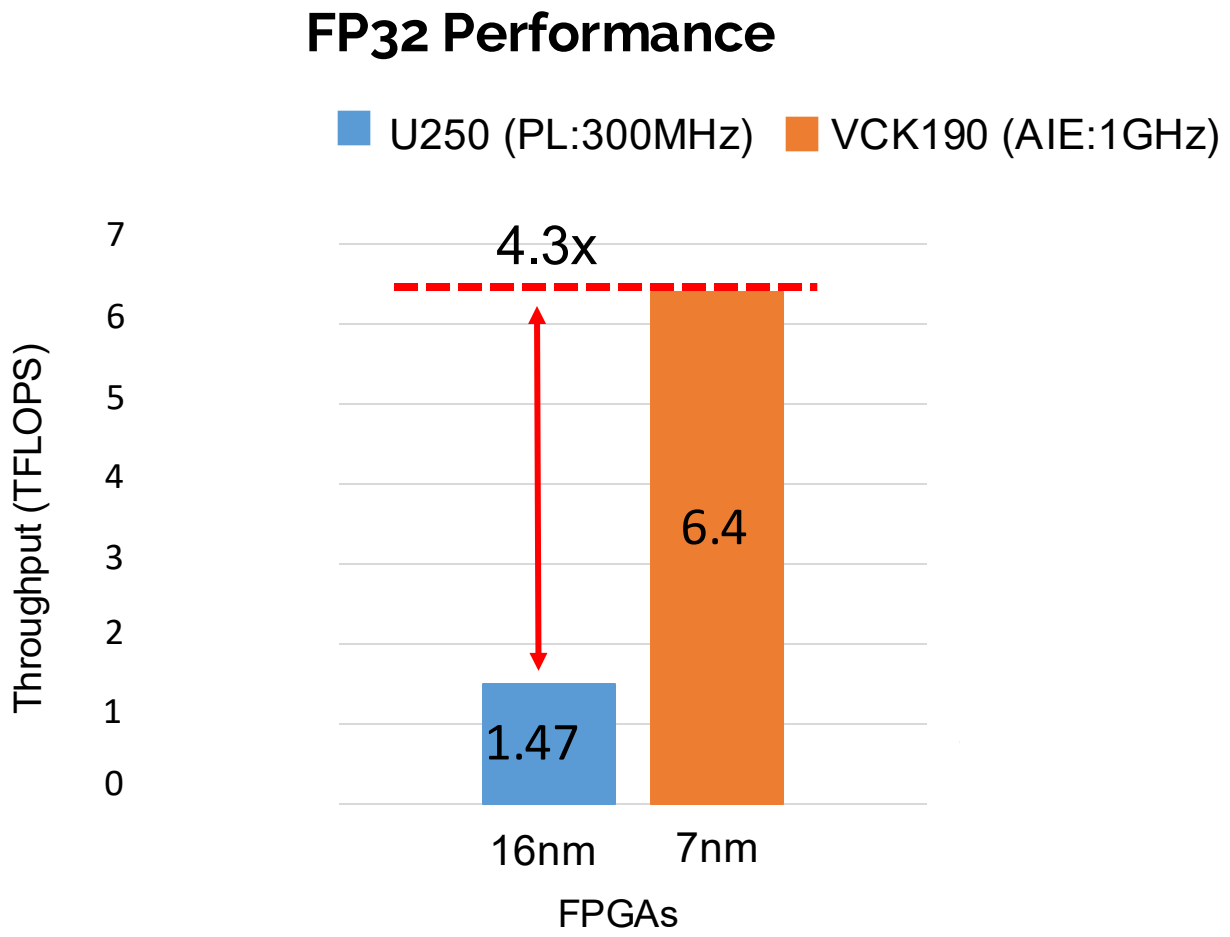
Challenge 1

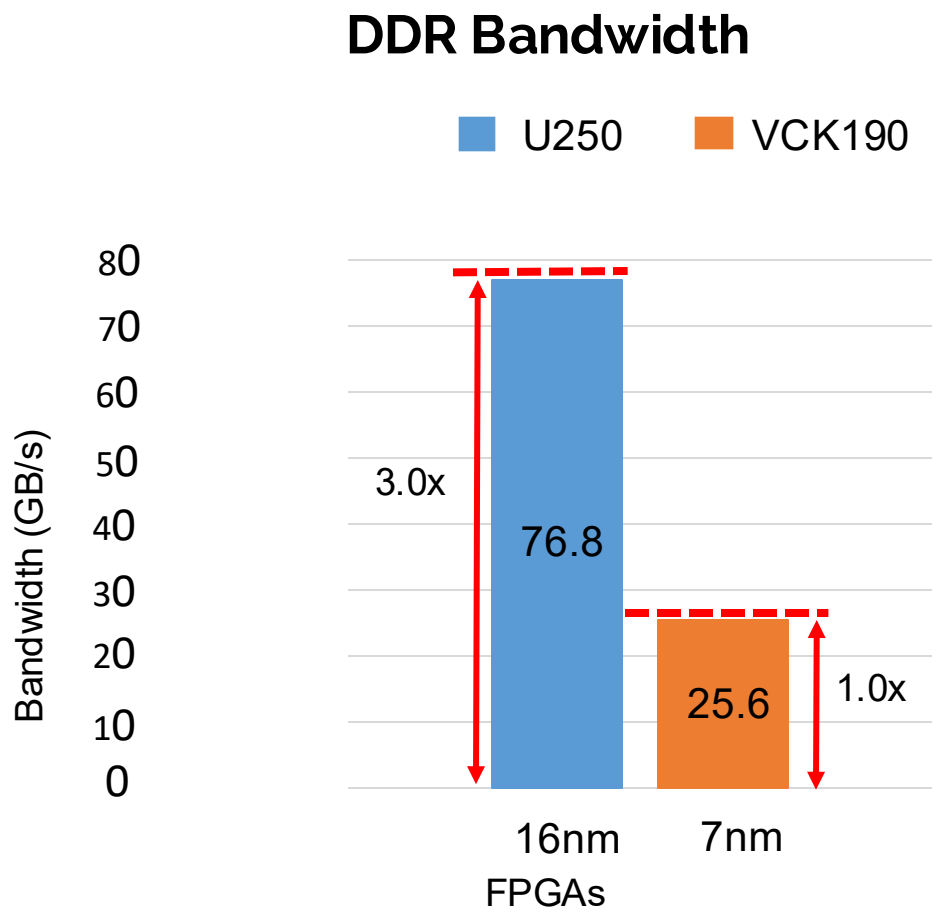
- **The computation scaling vs. off-chip communication scaling**



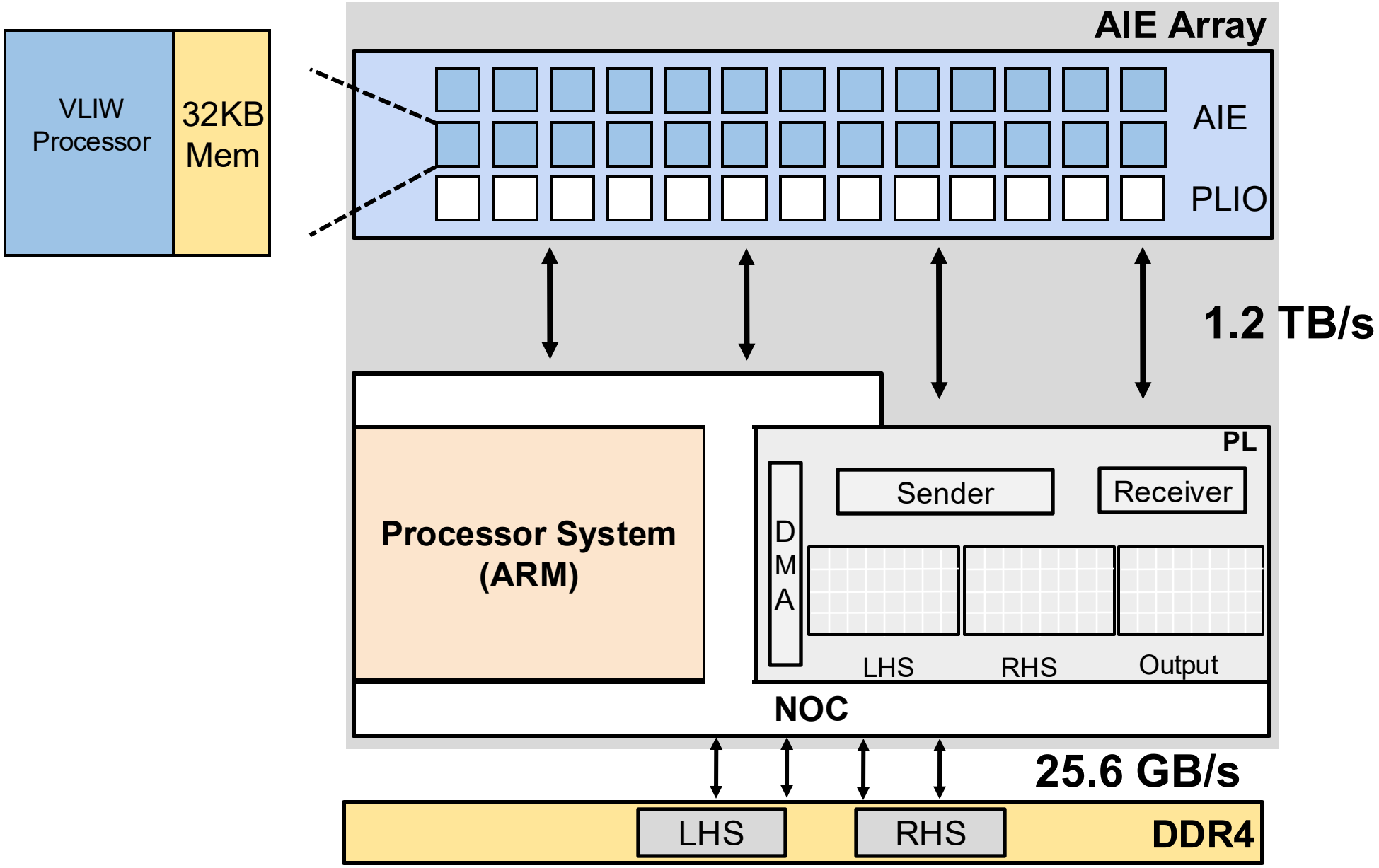
Challenge 1

- **The computation scaling vs. off-chip communication scaling**

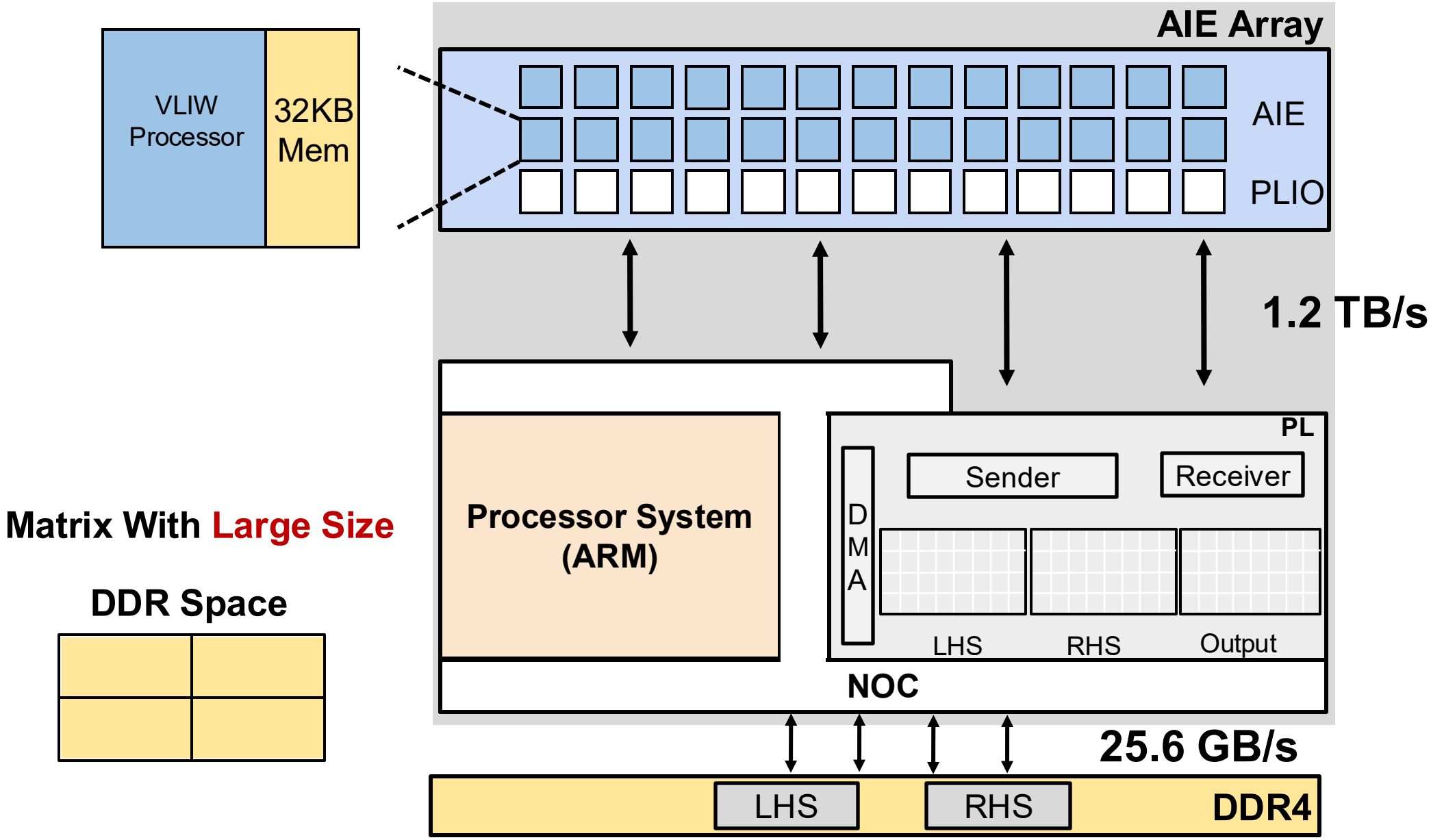




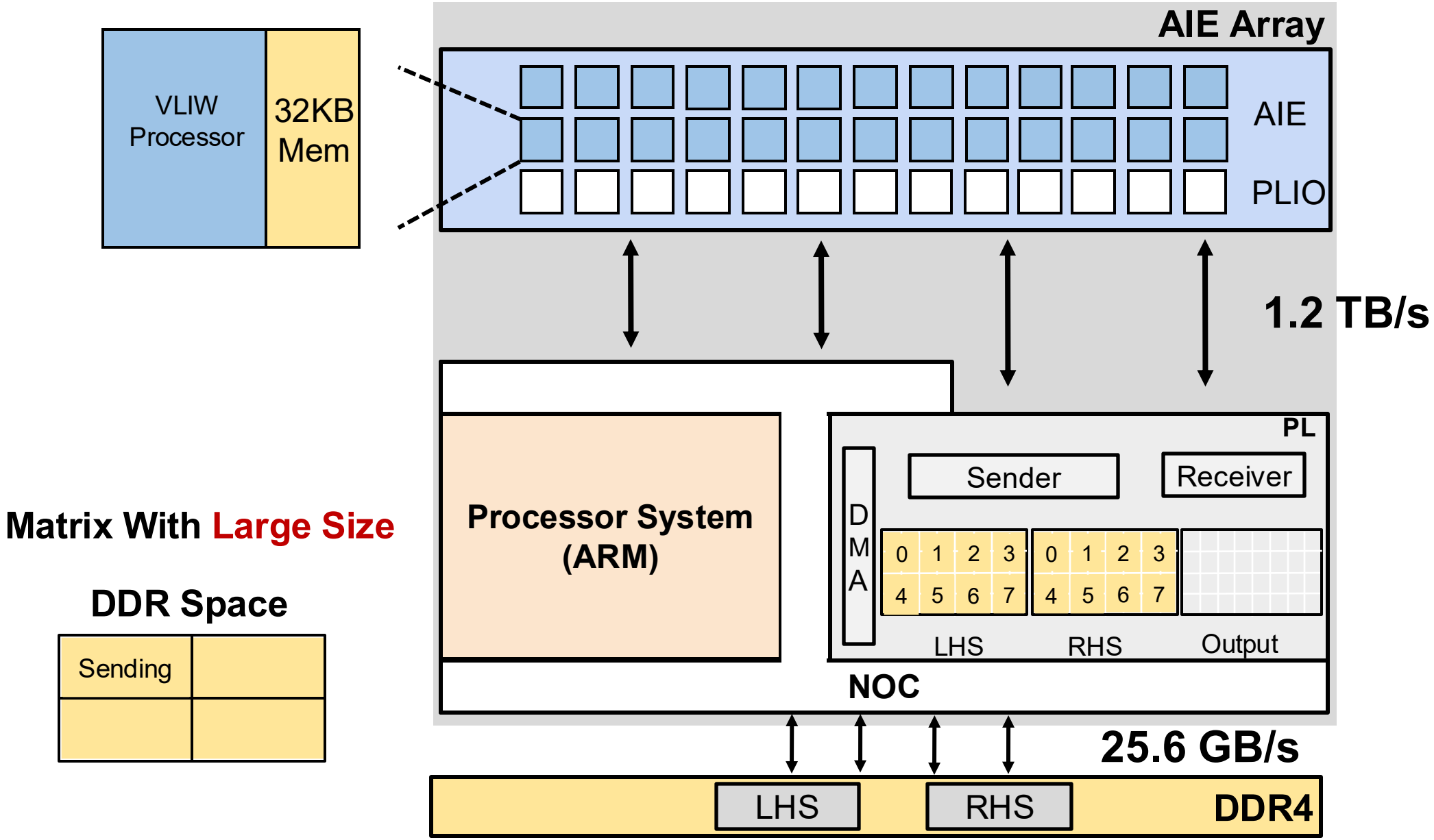
Versal Programming Model



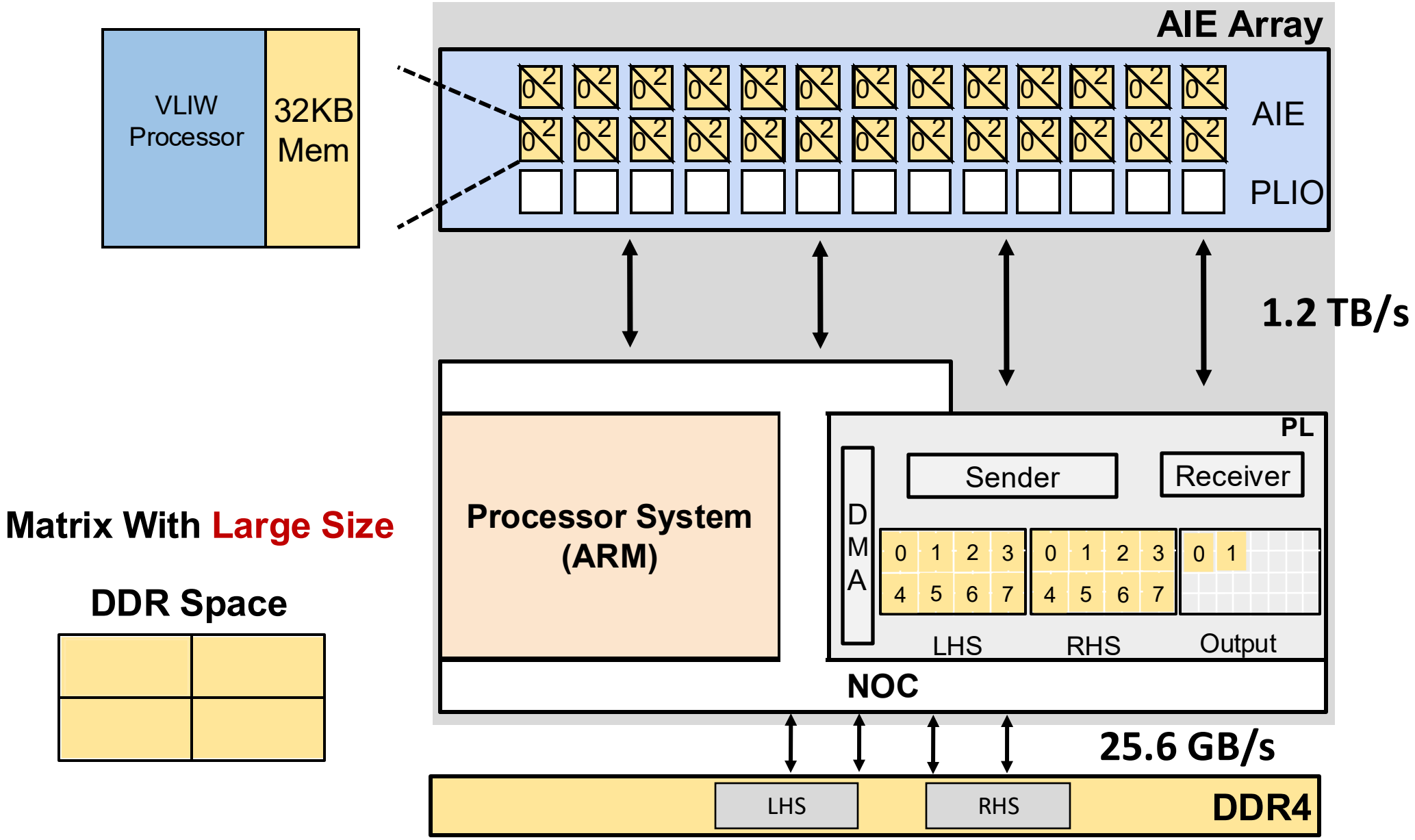
Versal Programming Model



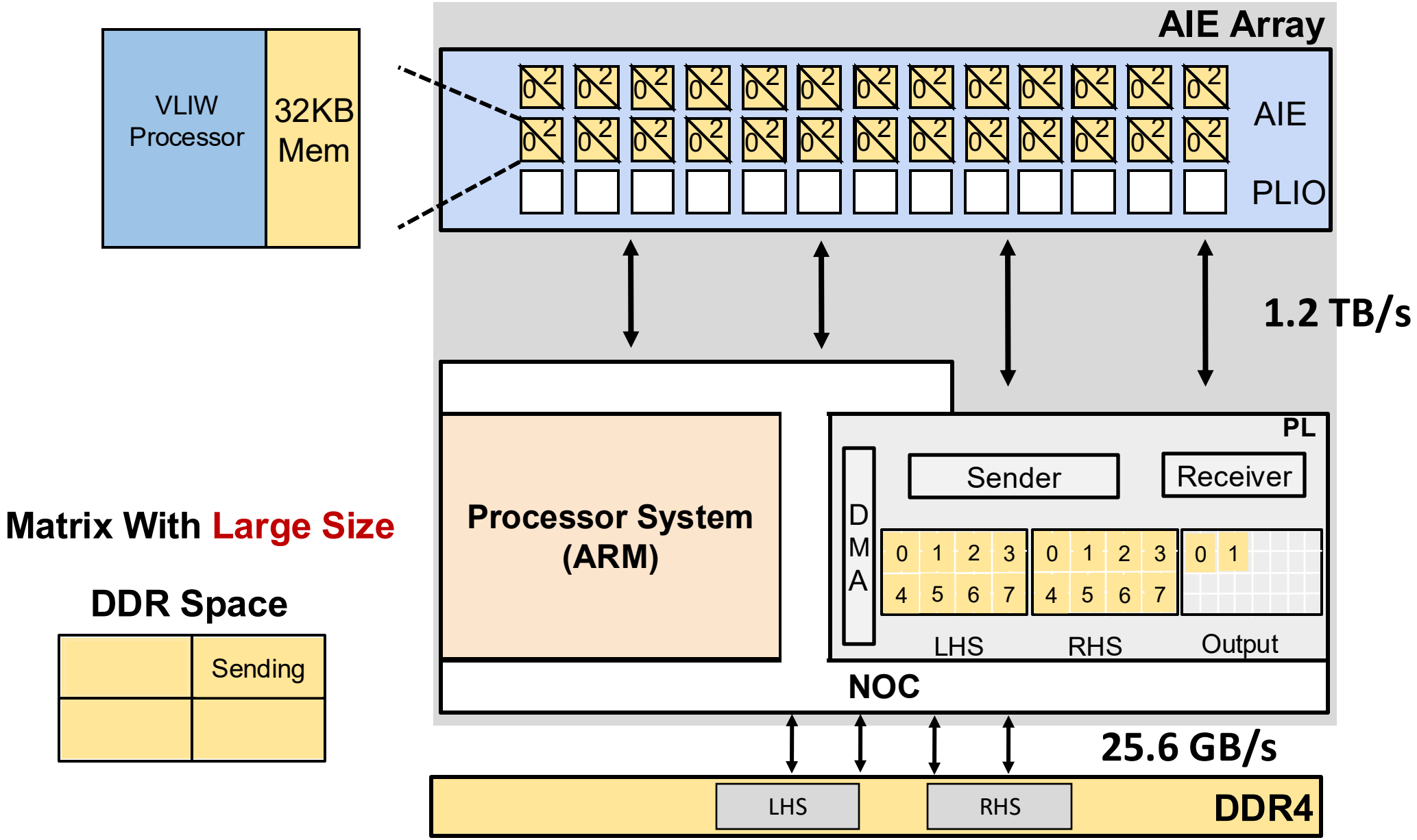
Versal Programming Model



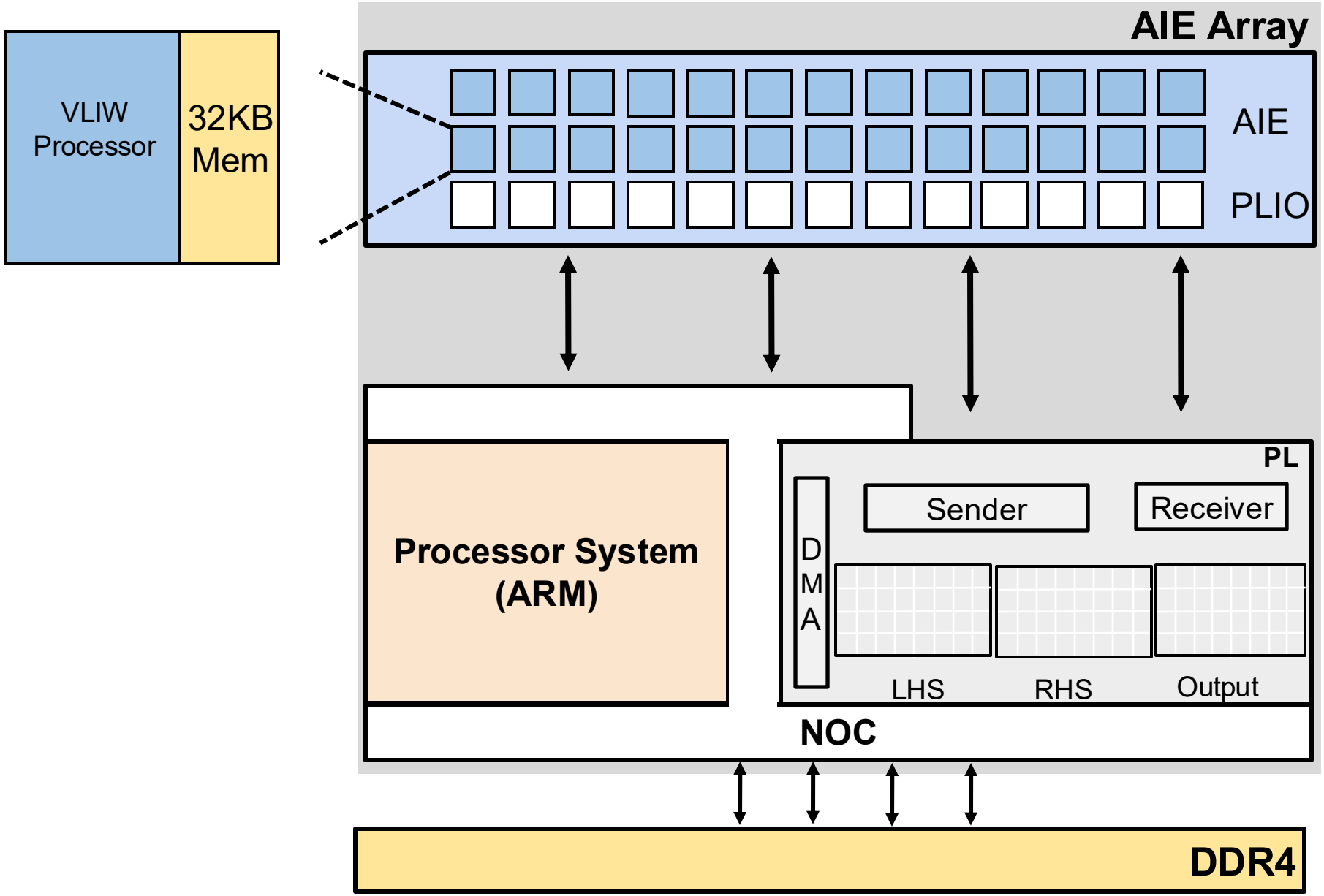
Versal Programming Model



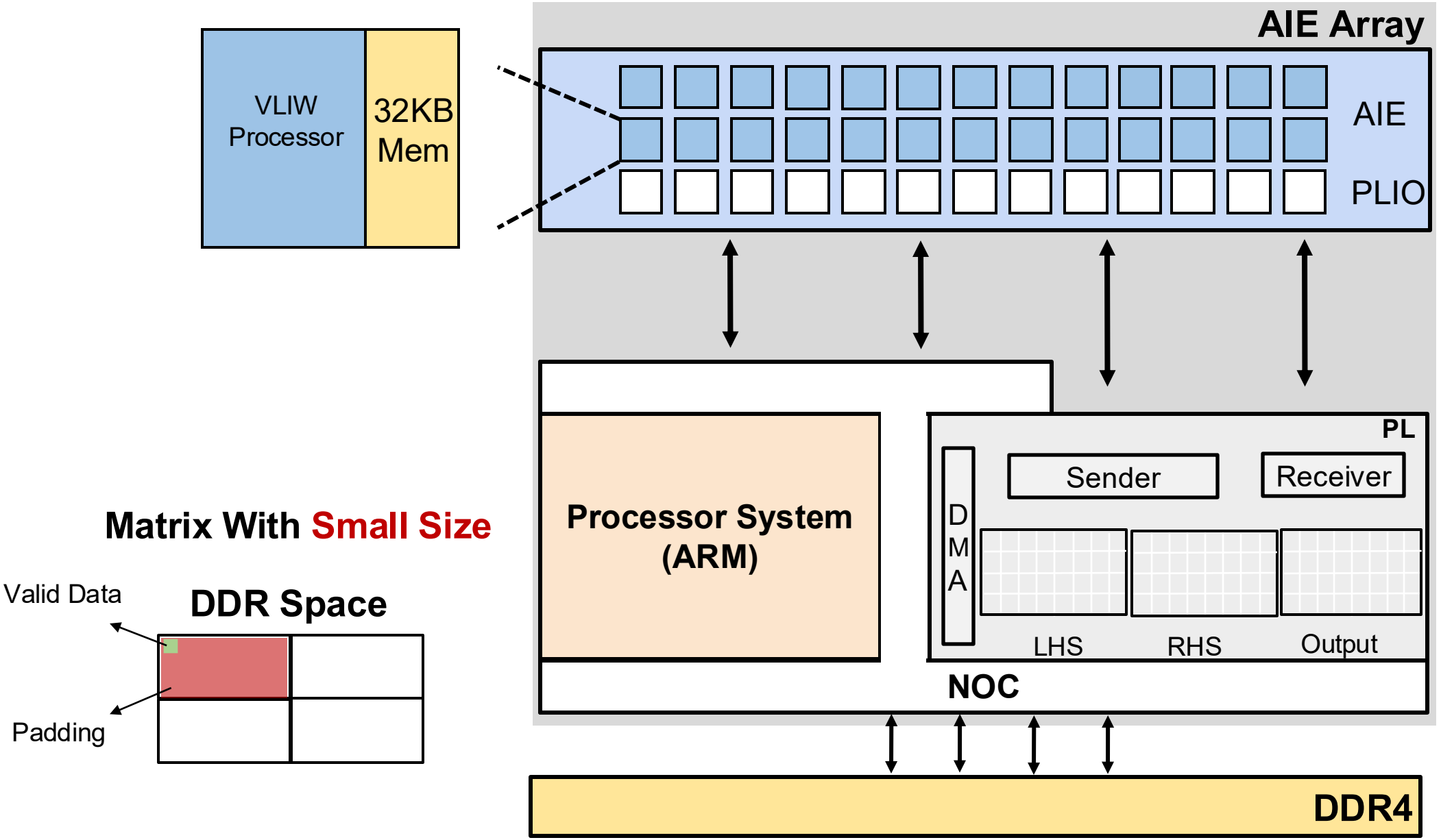
Versal Programming Model



Versal Programming Model

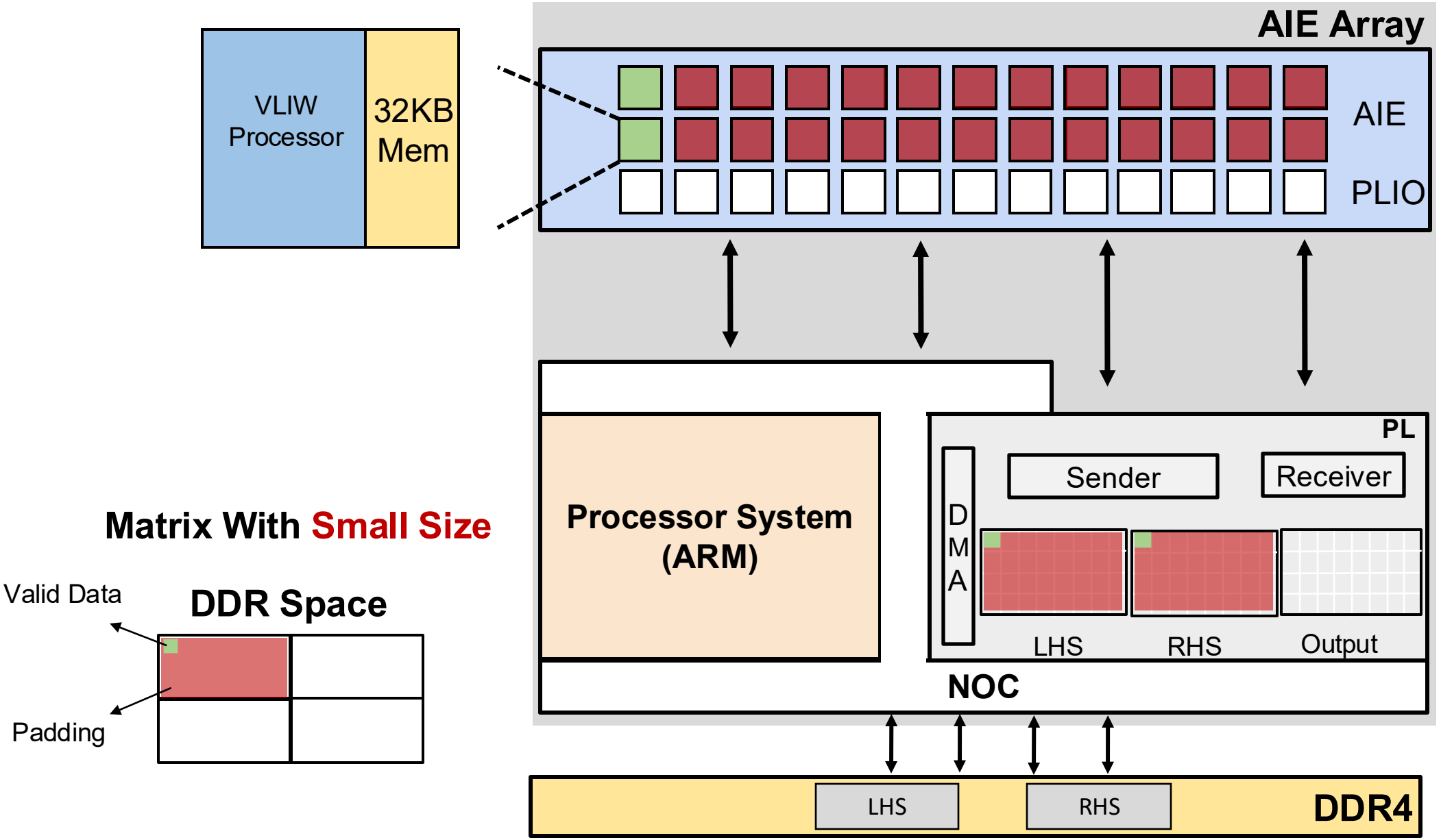


Versal Programming Model

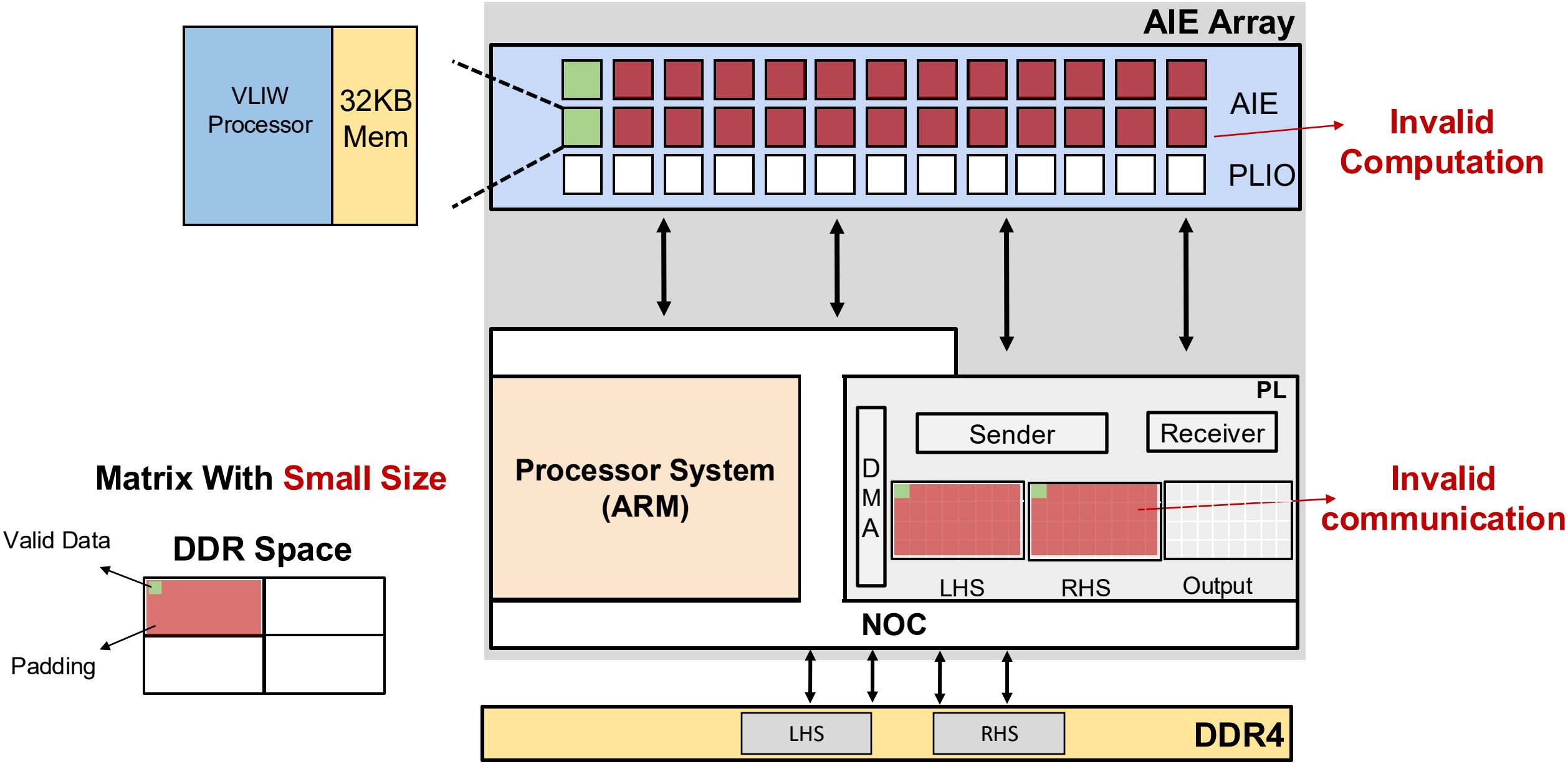


Matrix With Small Size

Versal Programming Model



Versal Programming Model



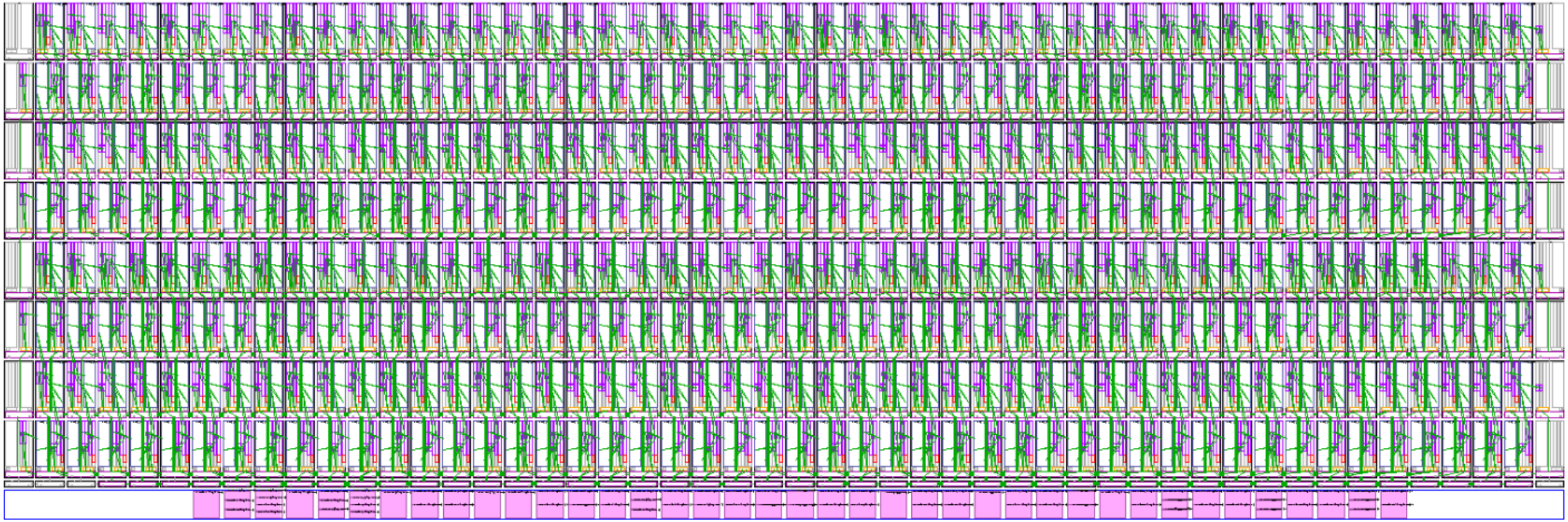
Challenge 2

- **Huge computation resources in one monolithic accelerator** vs. **MM layers of small sizes**

Challenge 2

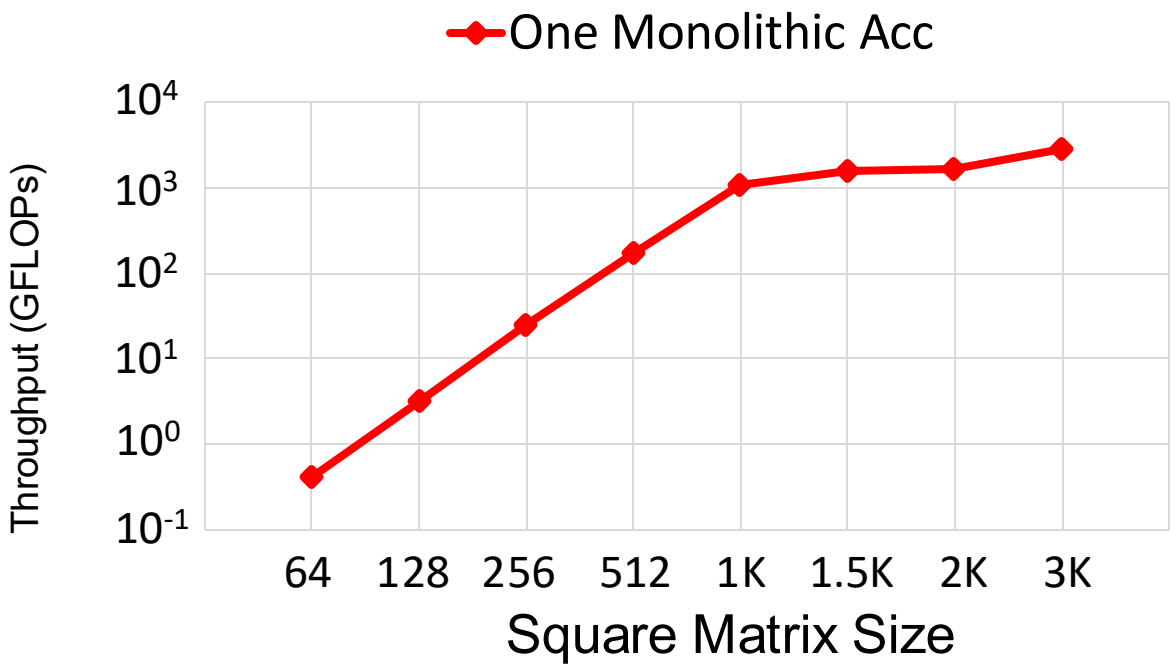
- **Huge computation resources in one monolithic accelerator** vs. **MM layers of small sizes**

  Matrix Multiply on 384 AIEs



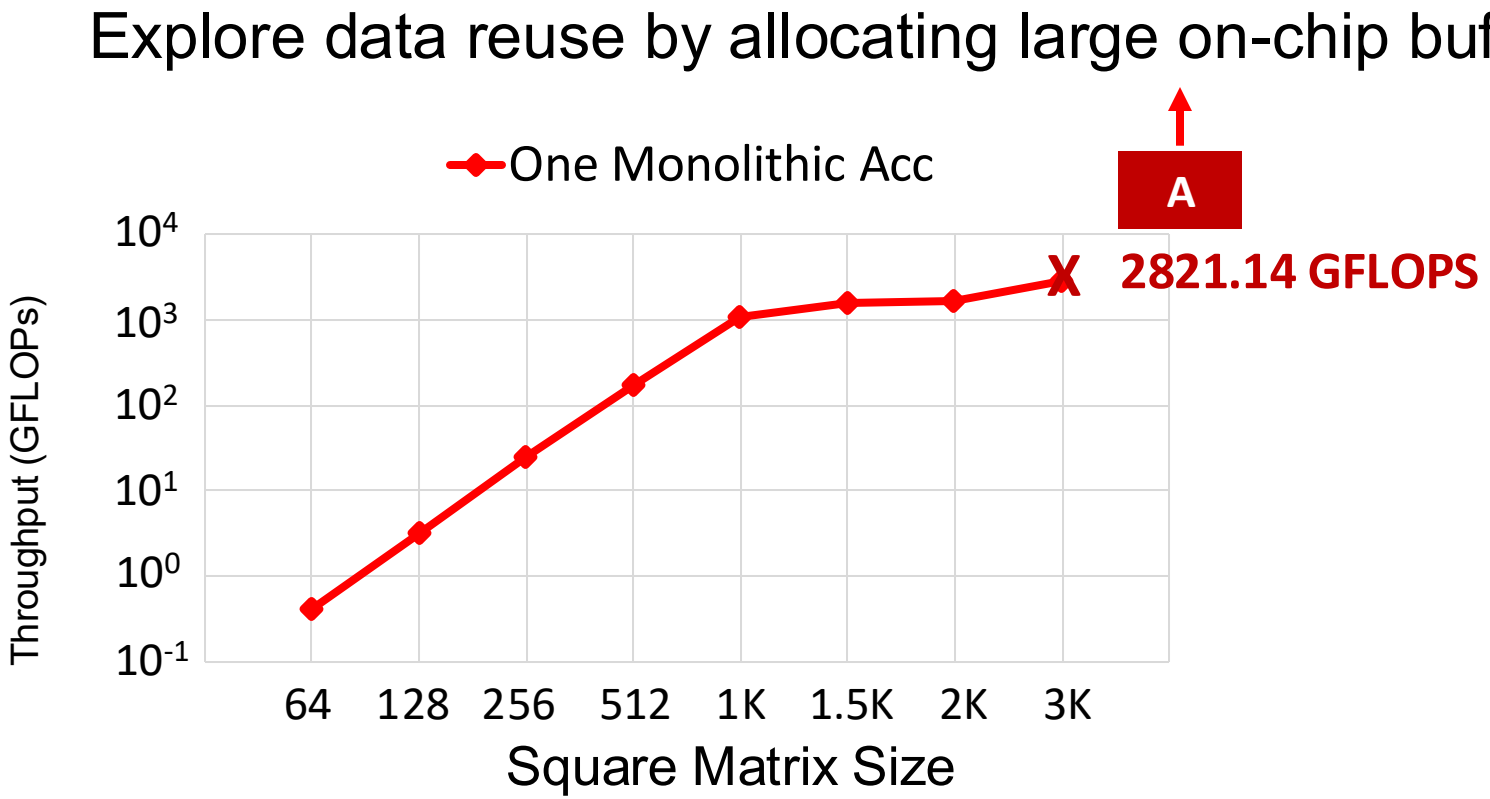
Challenge 2

- **Huge computation resources in one monolithic accelerator** vs. **MM layers of small sizes**



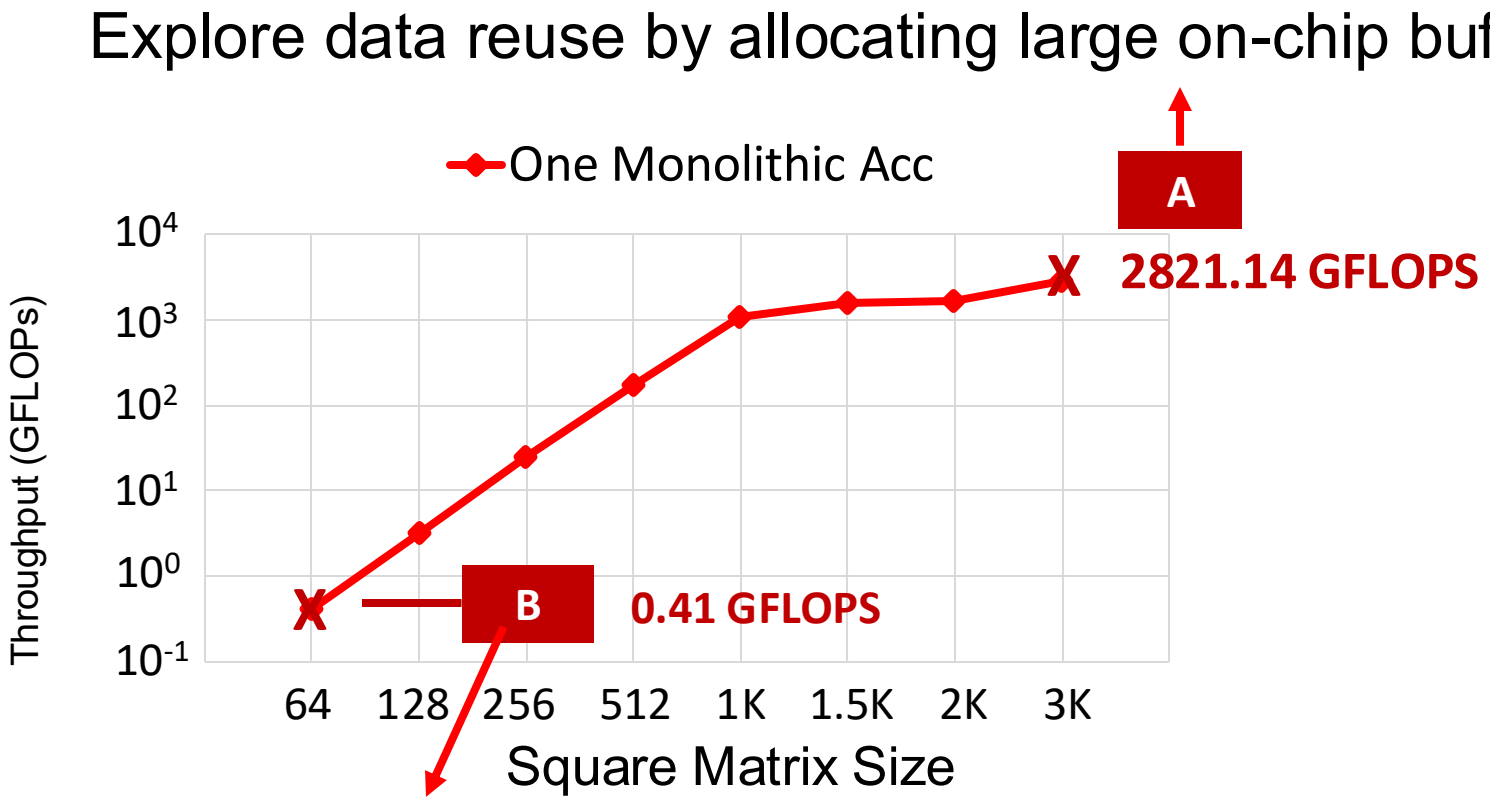
Challenge 2

- **Huge computation resources in one monolithic accelerator** vs. **MM layers of small sizes**



Challenge 2

- **Huge computation resources in one monolithic accelerator** vs. **MM layers of small sizes**



Shape mismatch, waste on both computation and communication

Motivation Example

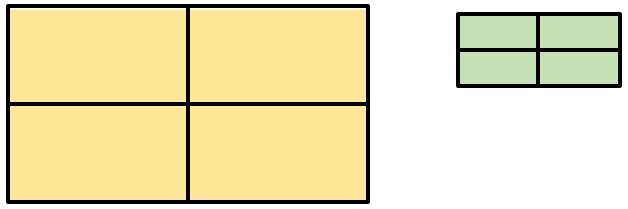
- **Composing Diverse Accs**

Motivation Example

- **Composing Diverse Accs**

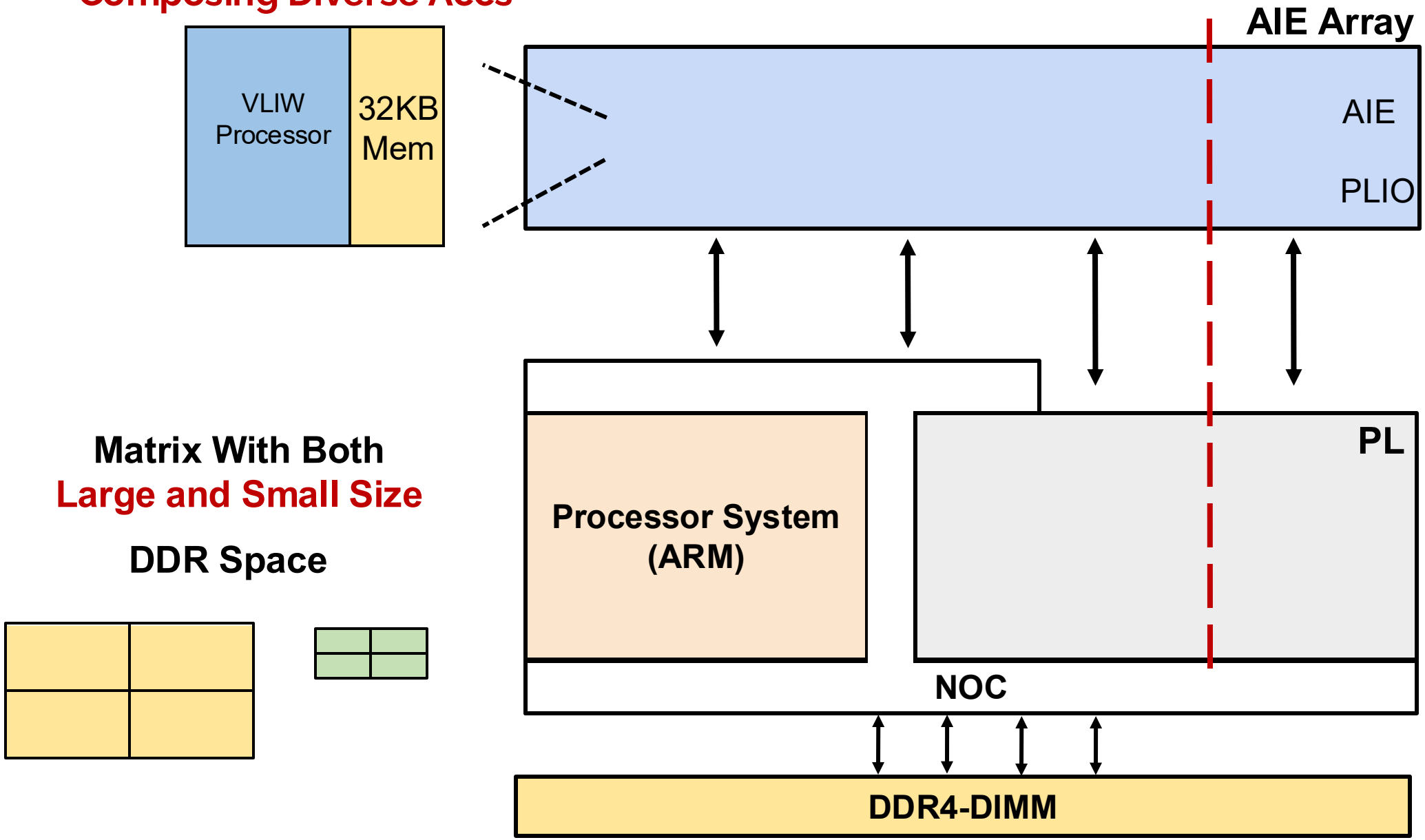
Matrix With Both
Large and Small Size

DDR Space



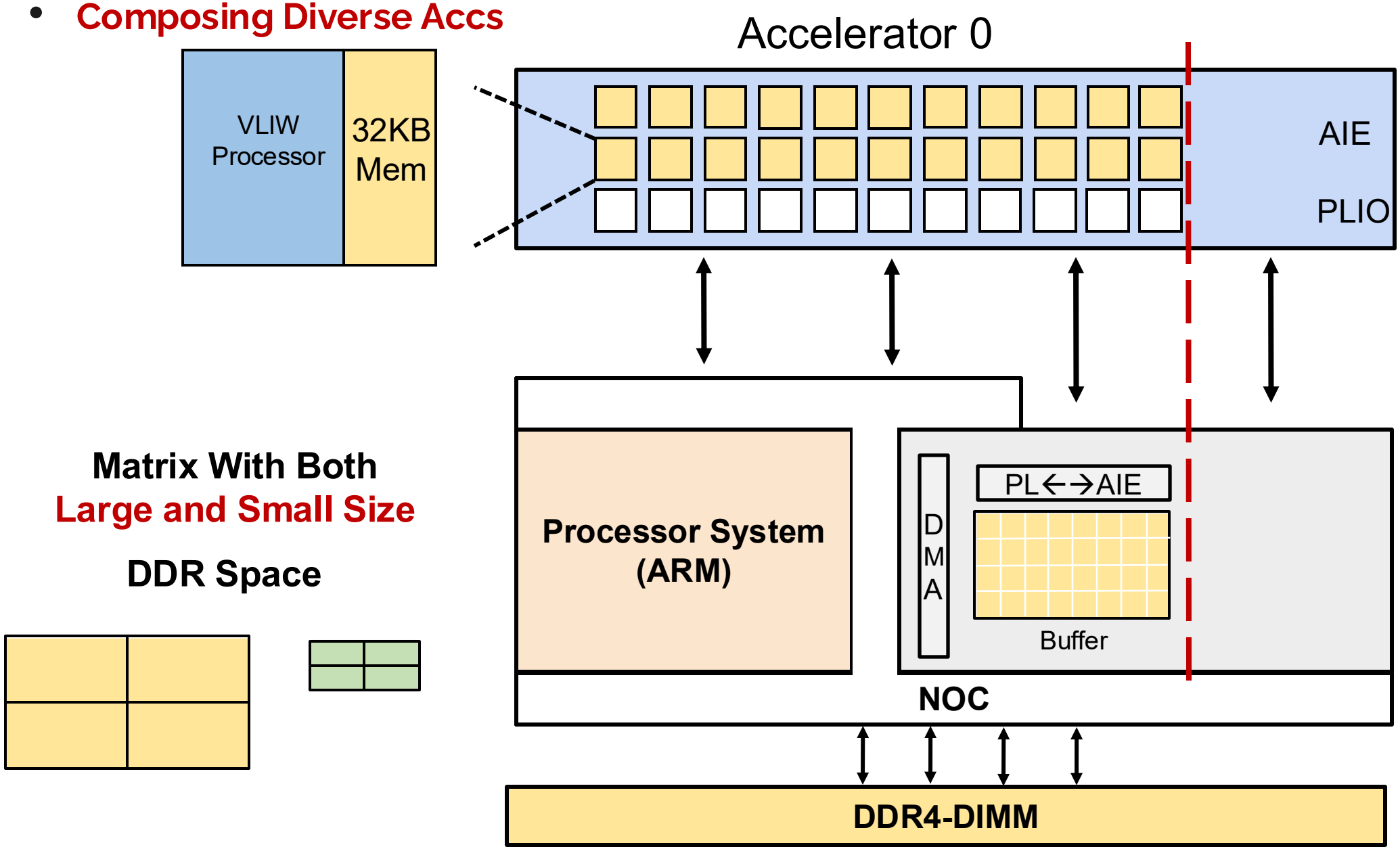
Motivation Example

- **Composing Diverse Accs**



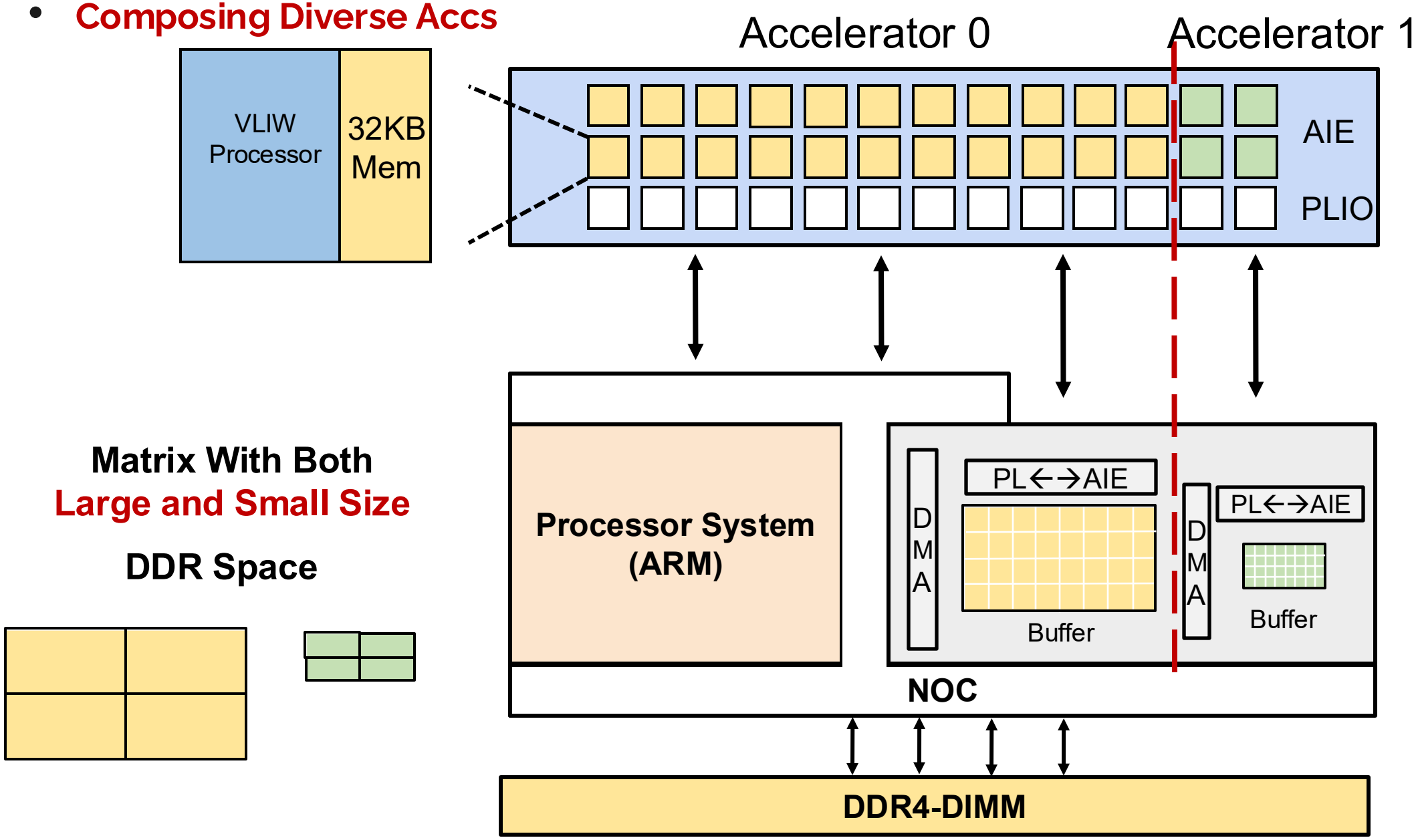
Motivation Example

- **Composing Diverse Accs**



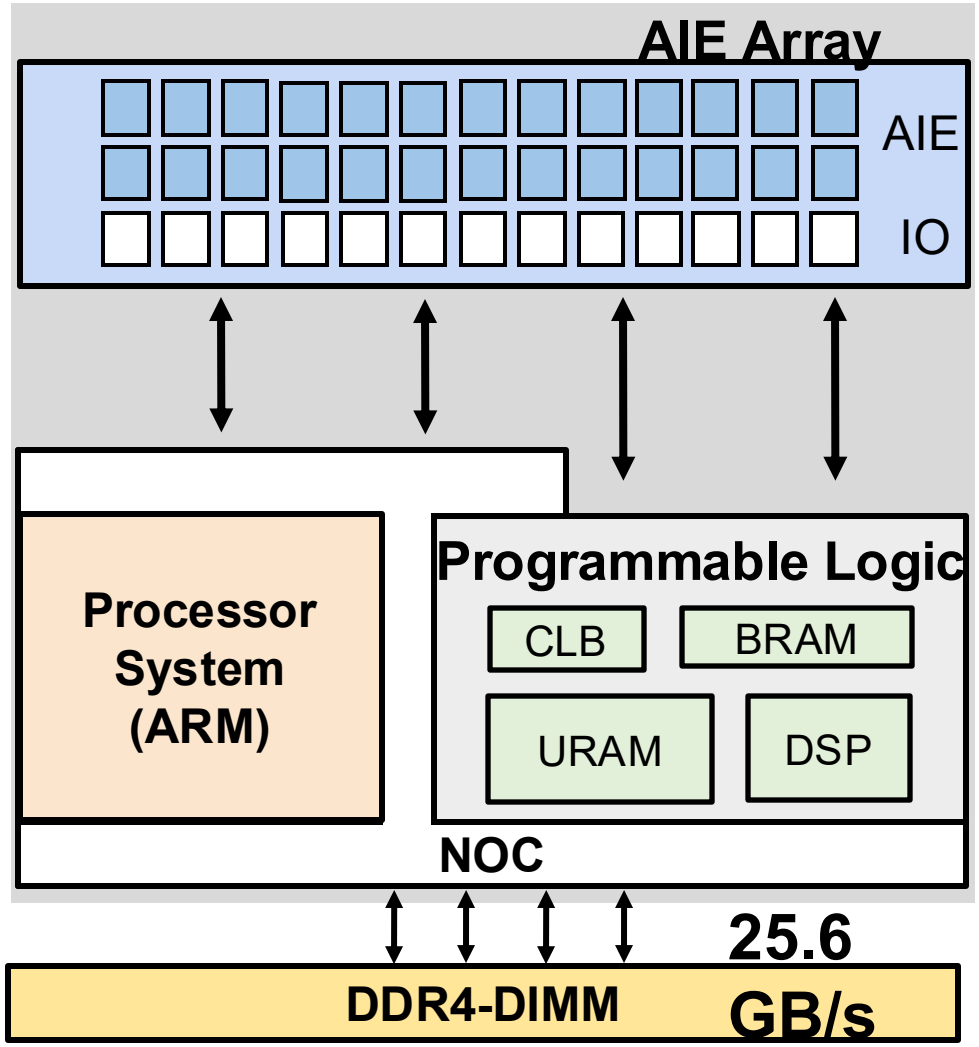
Motivation Example

- **Composing Diverse Accs**



Challenge 3

High level complexity



Increasing Programming Efforts

AIE C/C++	HLS C/C++	Host C/C++
900 LOC	1000 LOC	300 LOC

CHARM: Composing Heterogeneous Accelerators for Matrix Multiply on Versal ACAP Architecture

Challenges

Solutions

CHARM: Composing Heterogeneous Accelerators for Matrix Multiply on Versal ACAP Architecture

Challenges

- **Off-chip bandwidth doesn't scaling as fast as computation**

Solutions

- **Hugely explore the on-chip reuse**

CHARM: Composing Heterogeneous Accelerators for Matrix Multiply on Versal ACAP Architecture

Challenges

- Off-chip bandwidth doesn't scaling as fast as computation
- High parallelism makes one-size-fit-for-all solution inefficient

Solutions

- Hugely explore the on-chip reuse
- Two-step Accelerator Composing Algorithm

CHARM: Composing Heterogeneous Accelerators for Matrix Multiply on Versal ACAP Architecture

Challenges

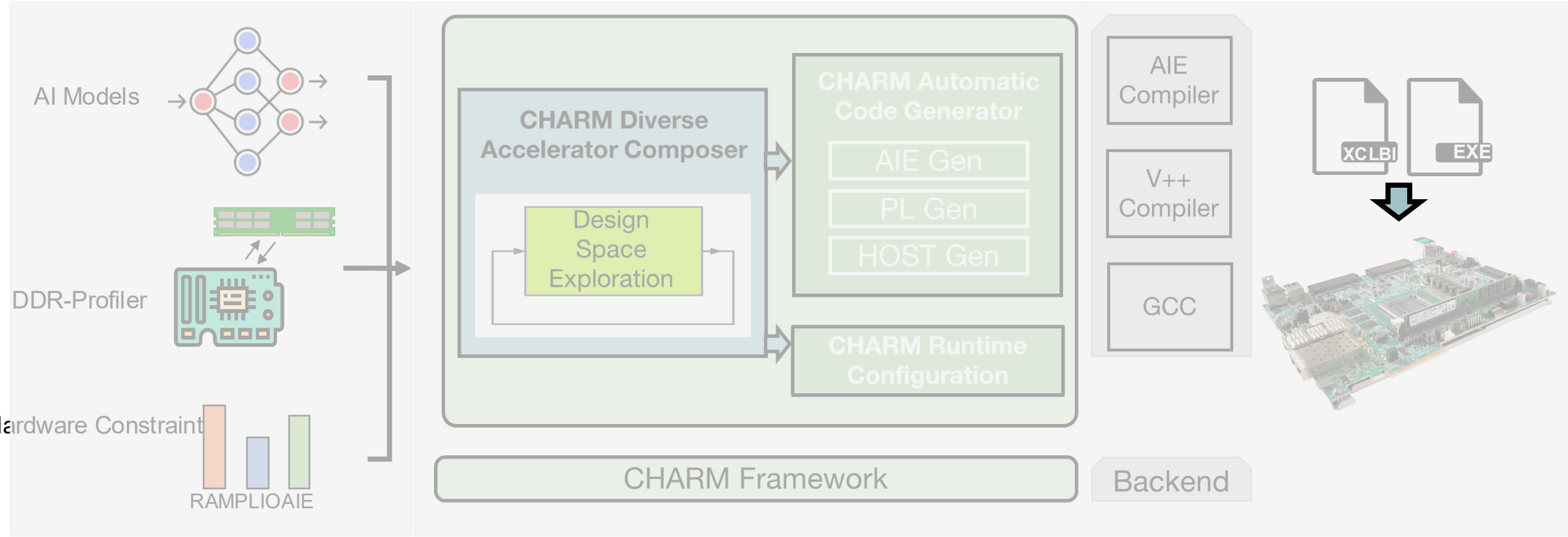
- Off-chip bandwidth doesn't scaling as fast as computation
- High parallelism makes one-size-fit-for-all solution inefficient
- Heterogeneity makes it non-trivial to program a system like Versal with good performance

Solutions

- Hugely explore the on-chip reuse
- Two-step Accelerator Composing Algorithm
- White-box open-sourced Framework
  https://github.com/arc-research-lab/CHARM

CHARM Compilation Flow

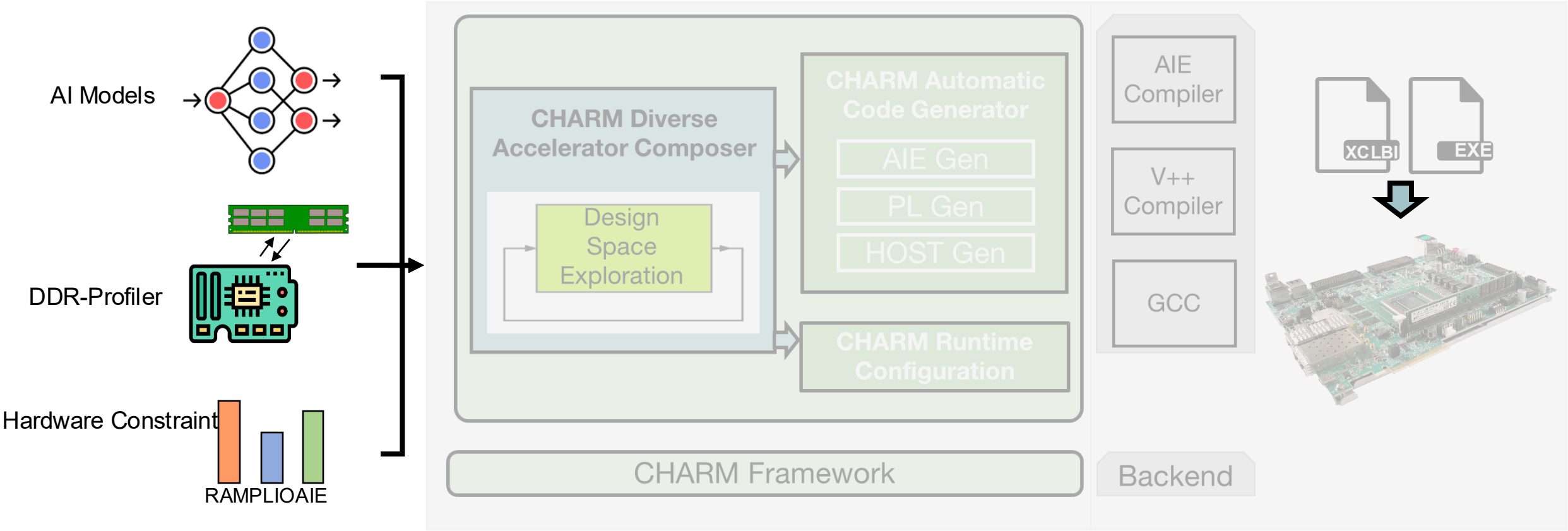
CHARM Input and Output



CHARM Compilation Flow

CHARM Input and Output

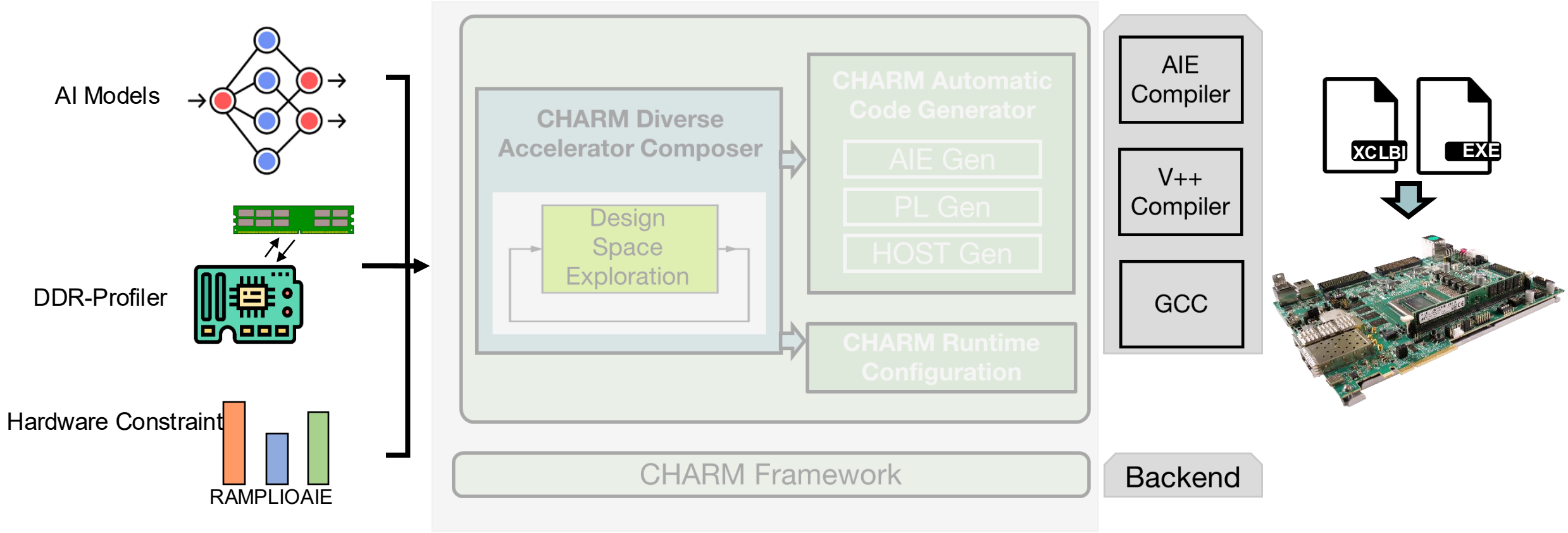
- **Input**
  1) MM-based AI Model ;
  2) Profiling of Off-chip Bandwidth ;
  3) Hardware resource constrains ;



CHARM Compilation Flow

CHARM Input and Output

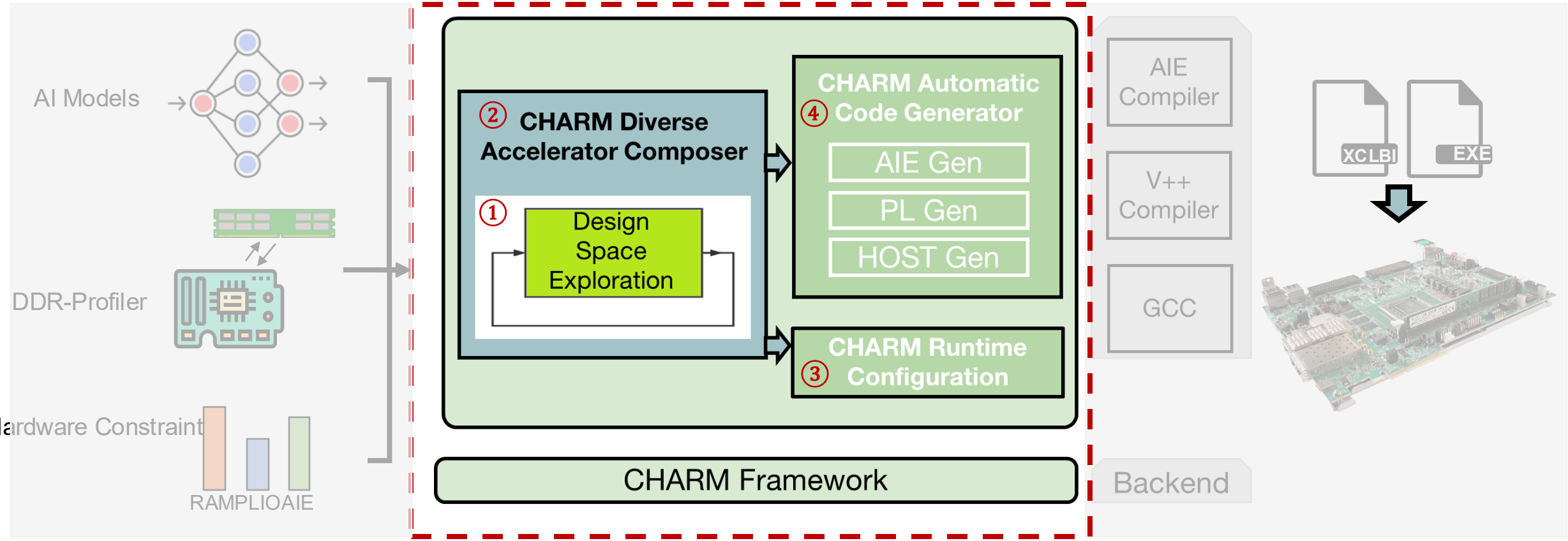
- **Input**
  1) MM-based AI Model ;
  2) Profiling of Off-chip Bandwidth ;
  3) Hardware resource constrains ;
- **Output**
  1) Bitstream Running on the AIE and PL (AIE/V++ Compiler)
  2) Executable Running on ARM CPU (GCC Cross Compilation)



CHARM Framework Overview

CHARM Components

① **Single Accelerator Design Space Exploration**
② **Diverse Accelerator Composer**
③ **Runtime Configuration**
④ **Automatic Code Generator**



CHARM Framework Overview

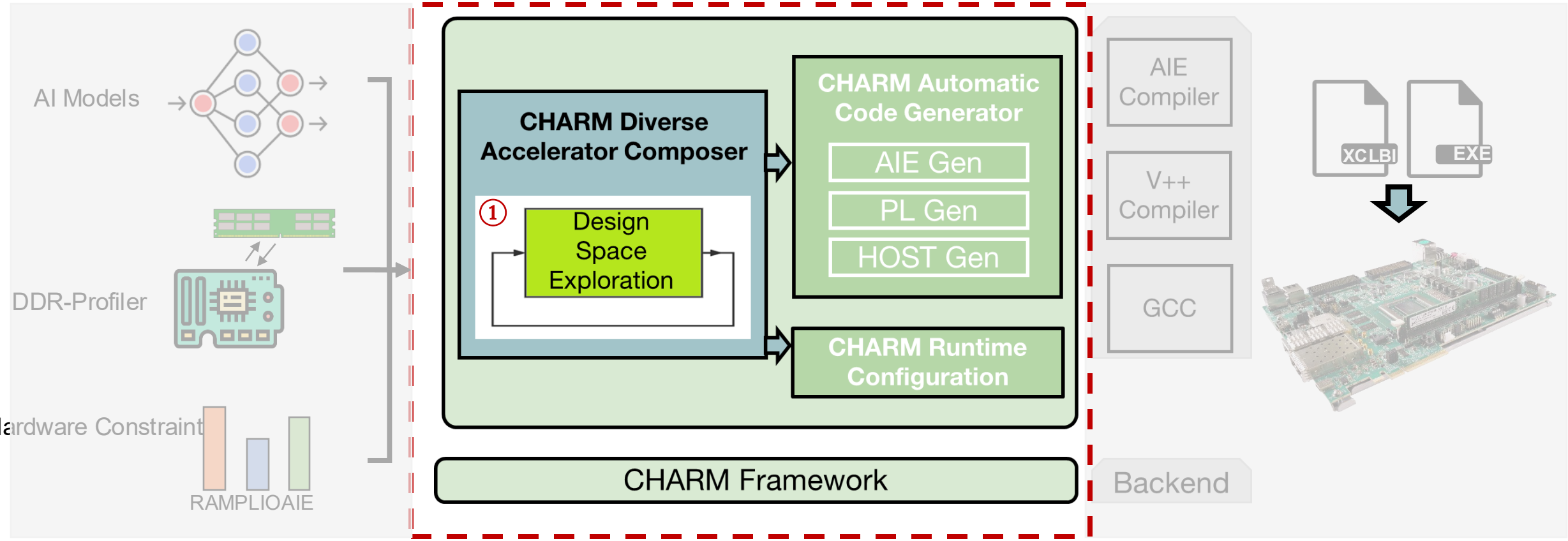
CHARM Components

① Single Accelerator Mathematical-Modeling Based Design Space Exploration

1) Fully-pipelined AIE Processor Design

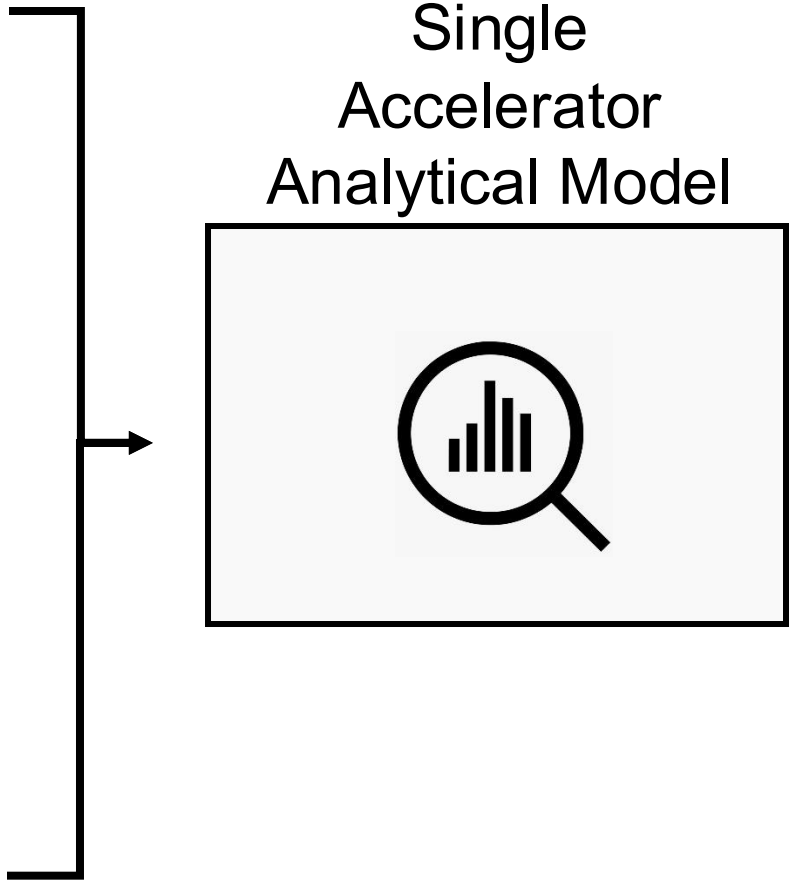
2) Hugely IO Reused AIE Array Design

3) On-chip Buffer Reused PL Design



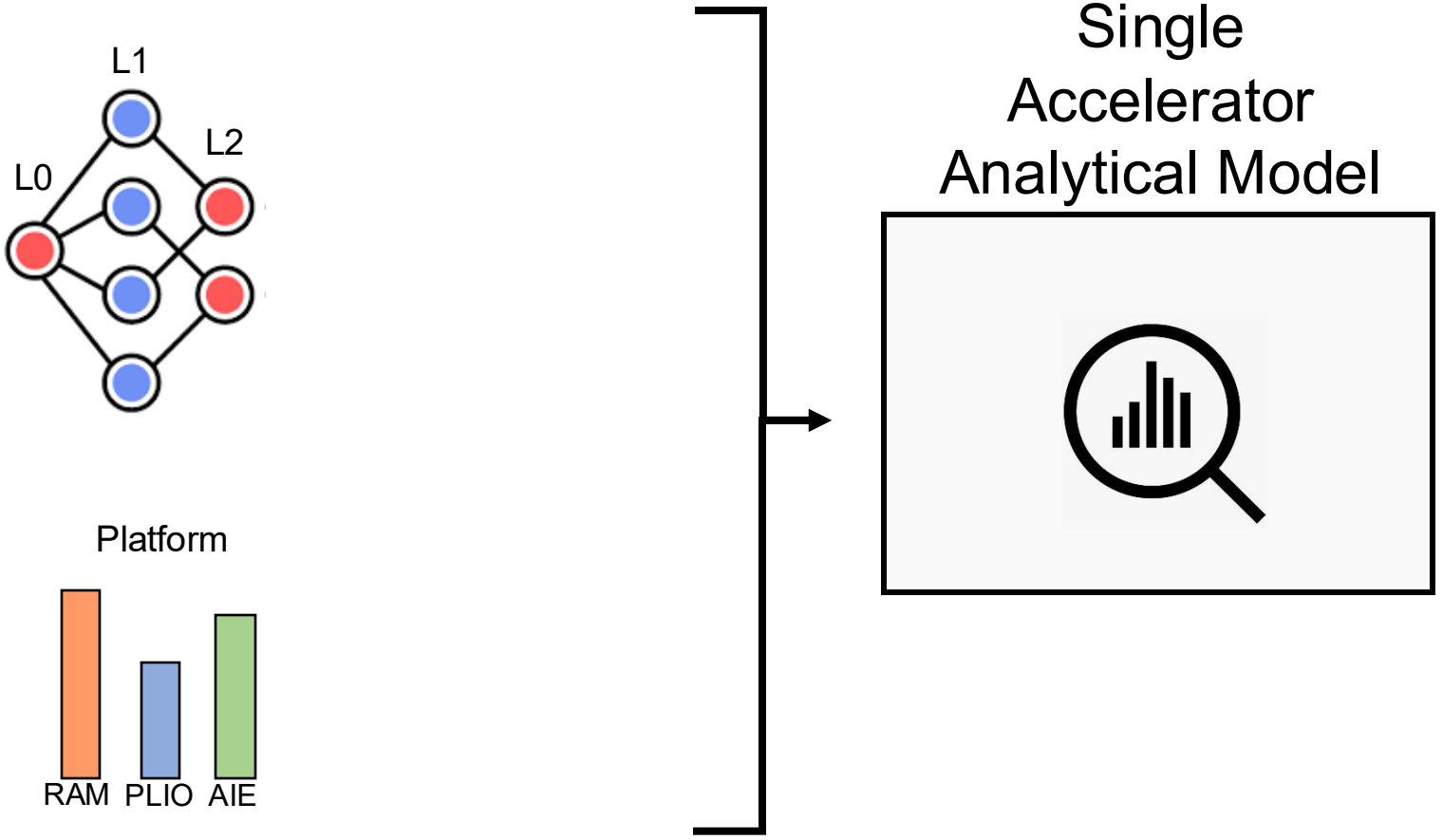
CHARM Framework Components

① Single Accelerator Design Space Exploration



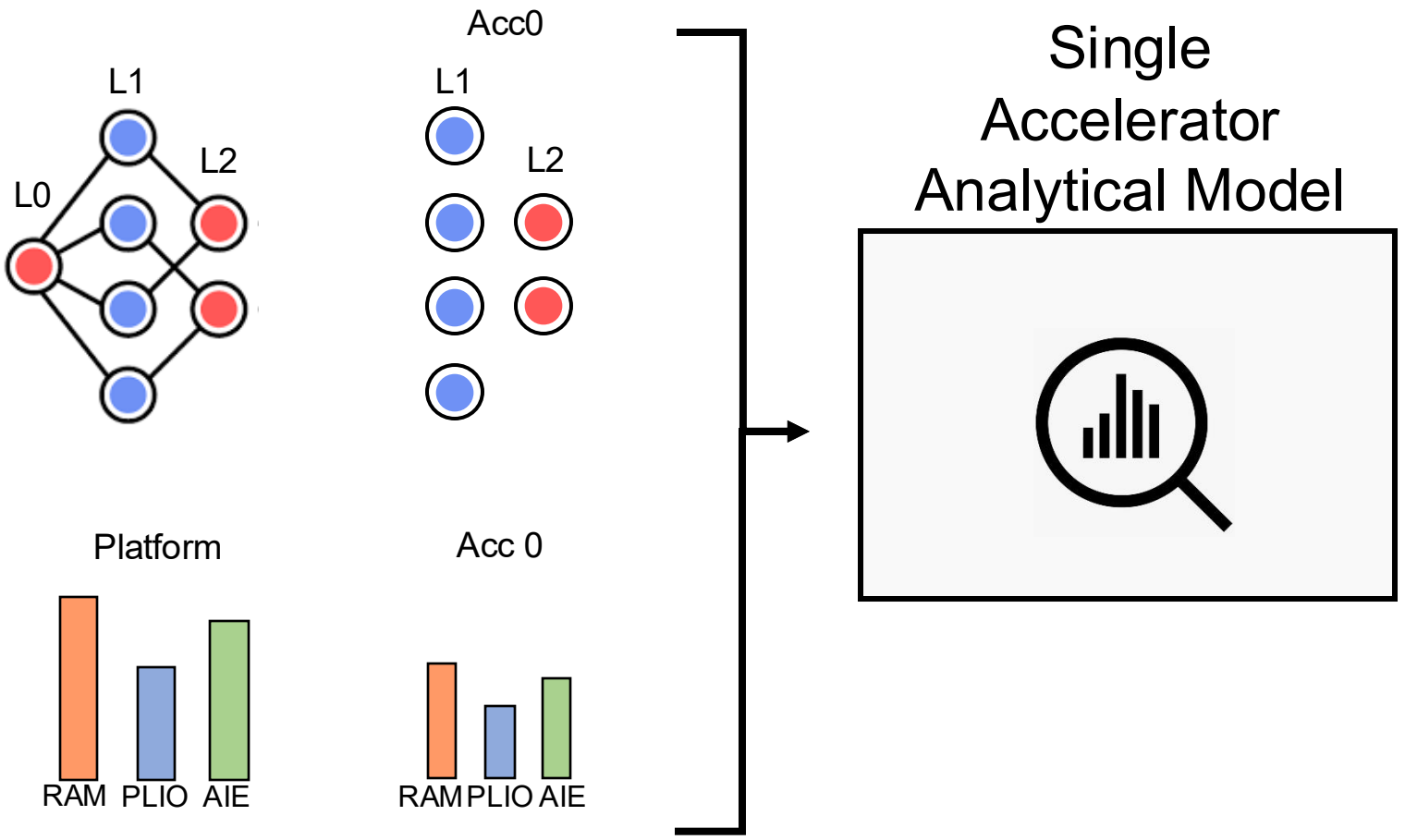
CHARM Framework Components

① Single Accelerator Design Space Exploration



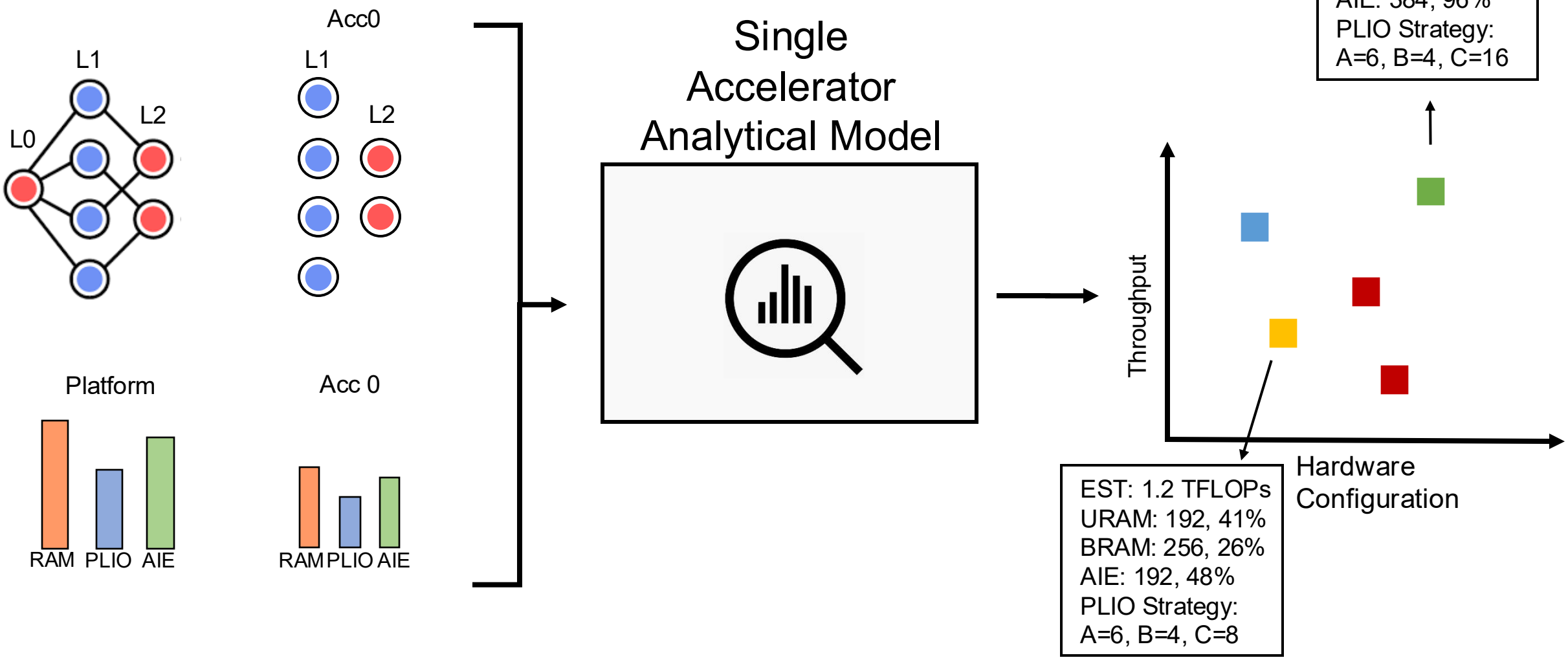
CHARM Framework Components

① Single Accelerator Design Space Exploration



CHARM Framework Components

① Single Accelerator Design Space Exploration



CHARM: Diverse Accelerator Composing

- **Two-step Search Algorithm**

CHARM: Diverse Accelerator Composing

- **Two-step Search Algorithm**

  How to assign **M layers** (1) in one AI Model to **N accelerators** (2)?

  1st Step: Workload Assignment

  2nd Step: Hardware Resource Partitioning

CHARM: Diverse Accelerator Composing

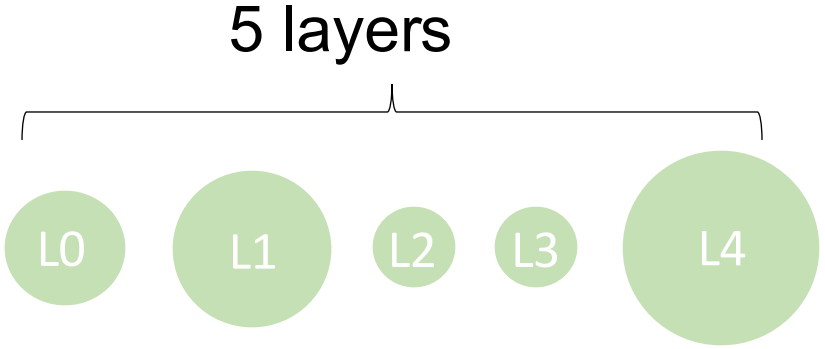
- **Two-step Search Algorithm**

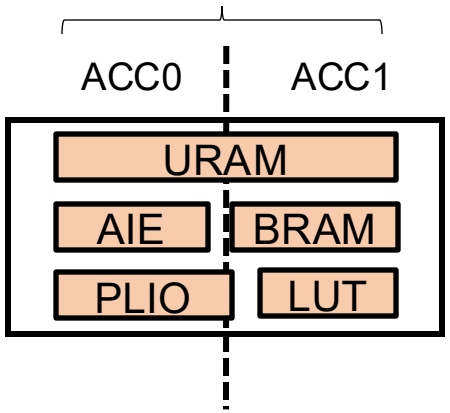
How to assign **M layers** in one AI Model to **N accelerators**?

① ②

1st Step: Workload Assignment

2nd Step: Hardware Resource Partitioning





CHARM: Diverse Accelerator Composing

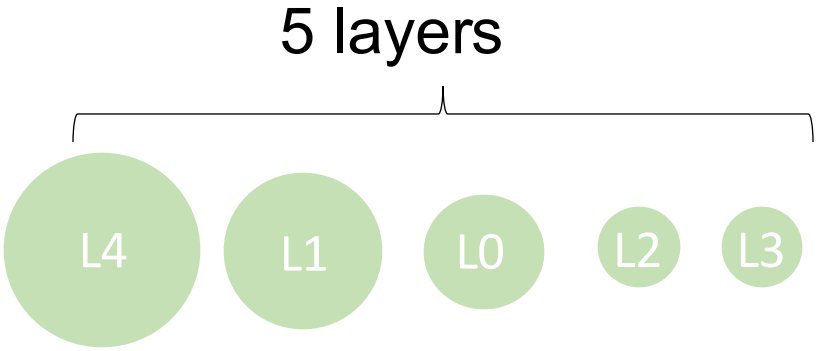
- **Two-step Search Algorithm**

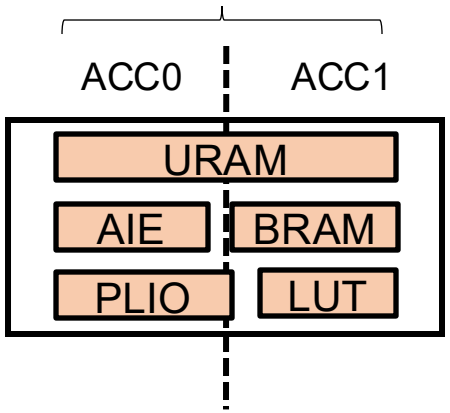
How to assign **M layers** in one AI Model to **N accelerators**?

① ②

1st Step: Workload Assignment

2nd Step: Hardware Resource Partitioning





CHARM: Diverse Accelerator Composing

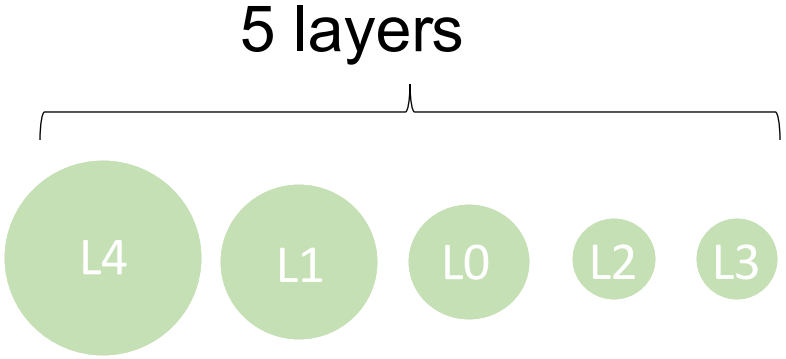
- **Two-step Search Algorithm**

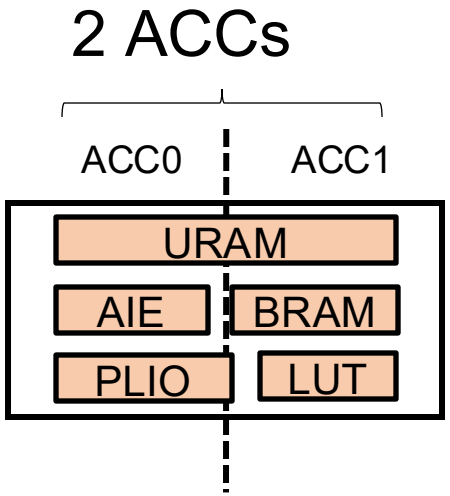
How to assign **M layers** in one AI Model to **N accelerators**?

① ②

1st Step: Workload Assignment

2nd Step: Hardware Resource Partitioning





Single Accelerator Analytical Model

CHARM: Diverse Accelerator Composing

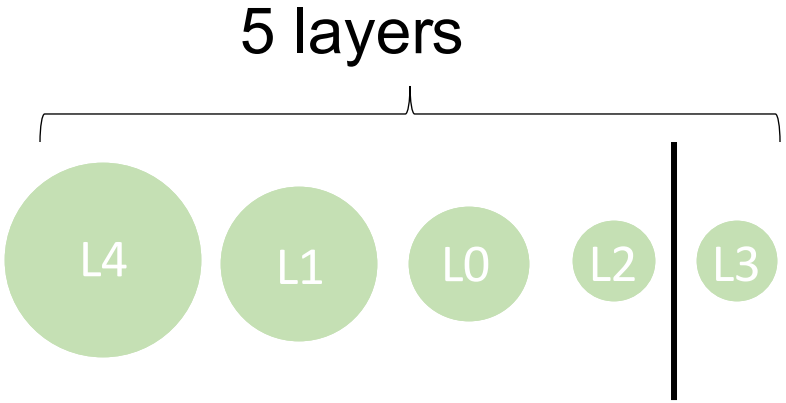
- **Two-step Search Algorithm**

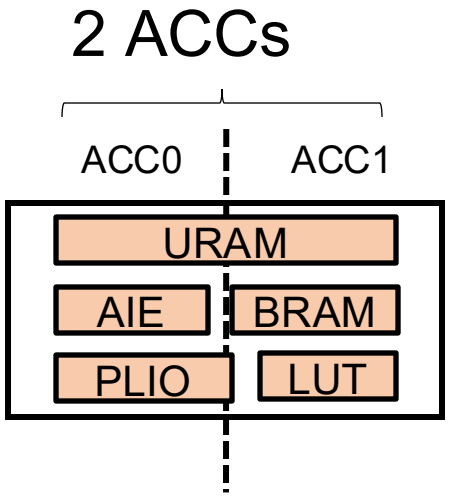
How to assign **M layers** in one AI Model to **N accelerators**?

① ②

1st Step: Workload Assignment

2nd Step: Hardware Resource Partitioning





Single Accelerator Analytical Model

CHARM: Diverse Accelerator Composing

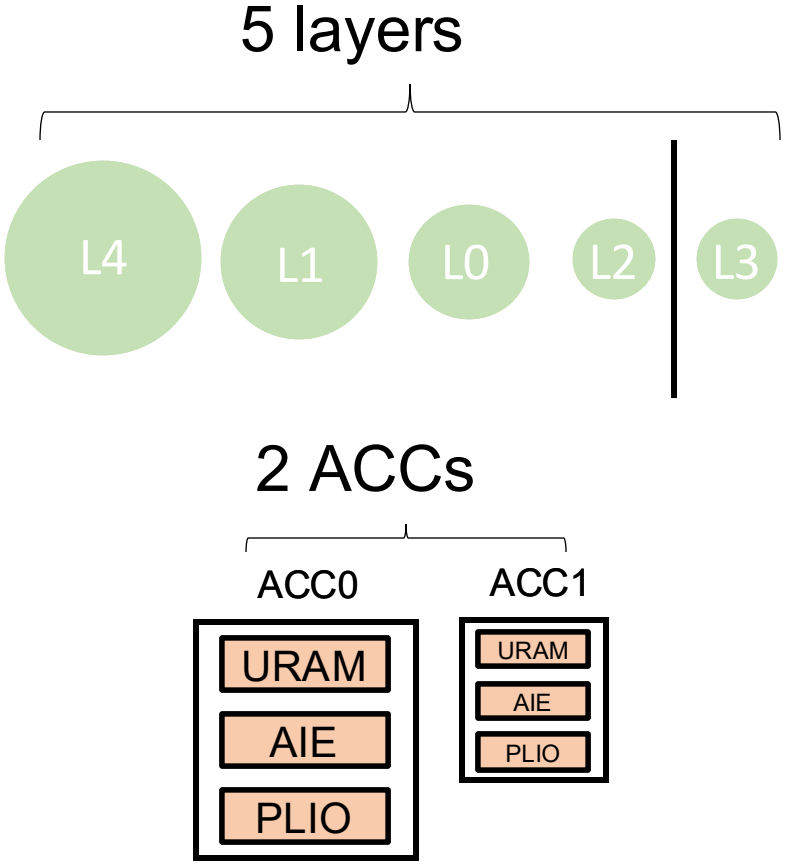
- **Two-step Search Algorithm**

How to assign **M layers** in one AI Model to **N accelerators**?

① ②

1st Step: Workload Assignment

2nd Step: Hardware Resource Partitioning



Single Accelerator Analytical Model

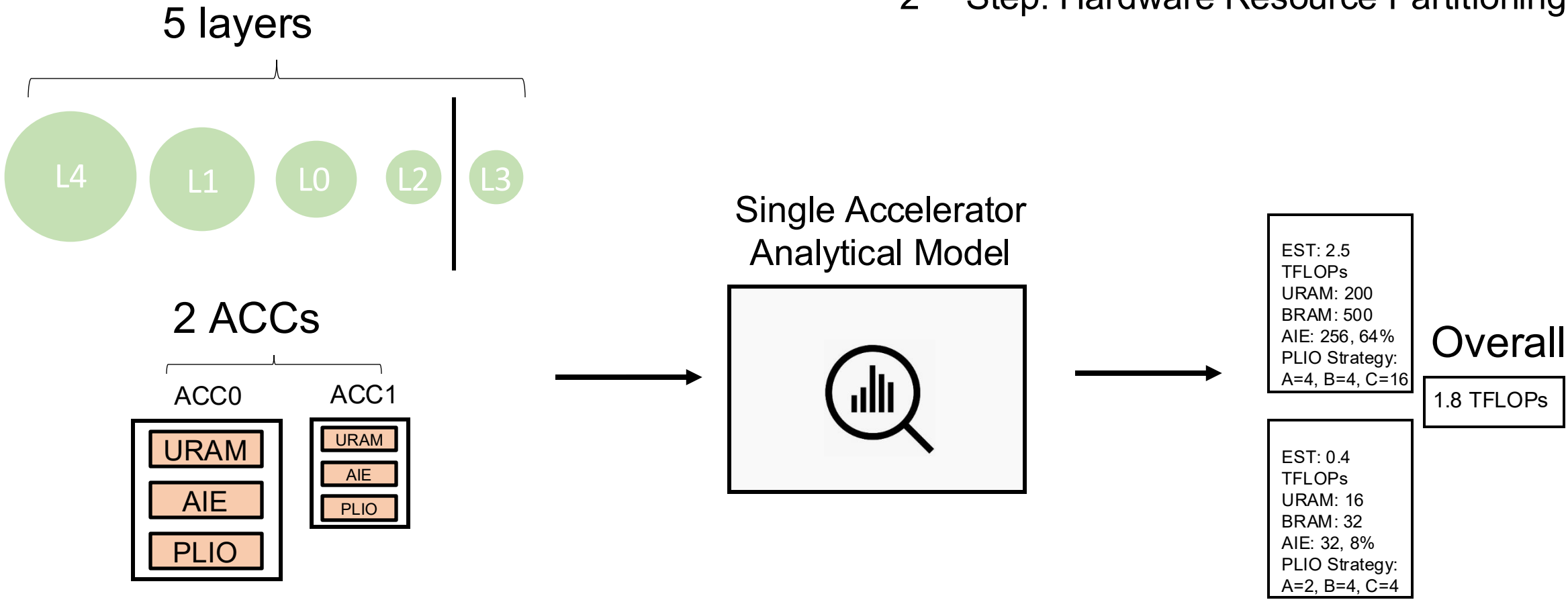
CHARM: Diverse Accelerator Composing

- **Two-step Search Algorithm**

How to assign **M layers** (1) in one AI Model to **N accelerators** (2)?

1st Step: Workload Assignment

2nd Step: Hardware Resource Partitioning



CHARM: Diverse Accelerator Composing

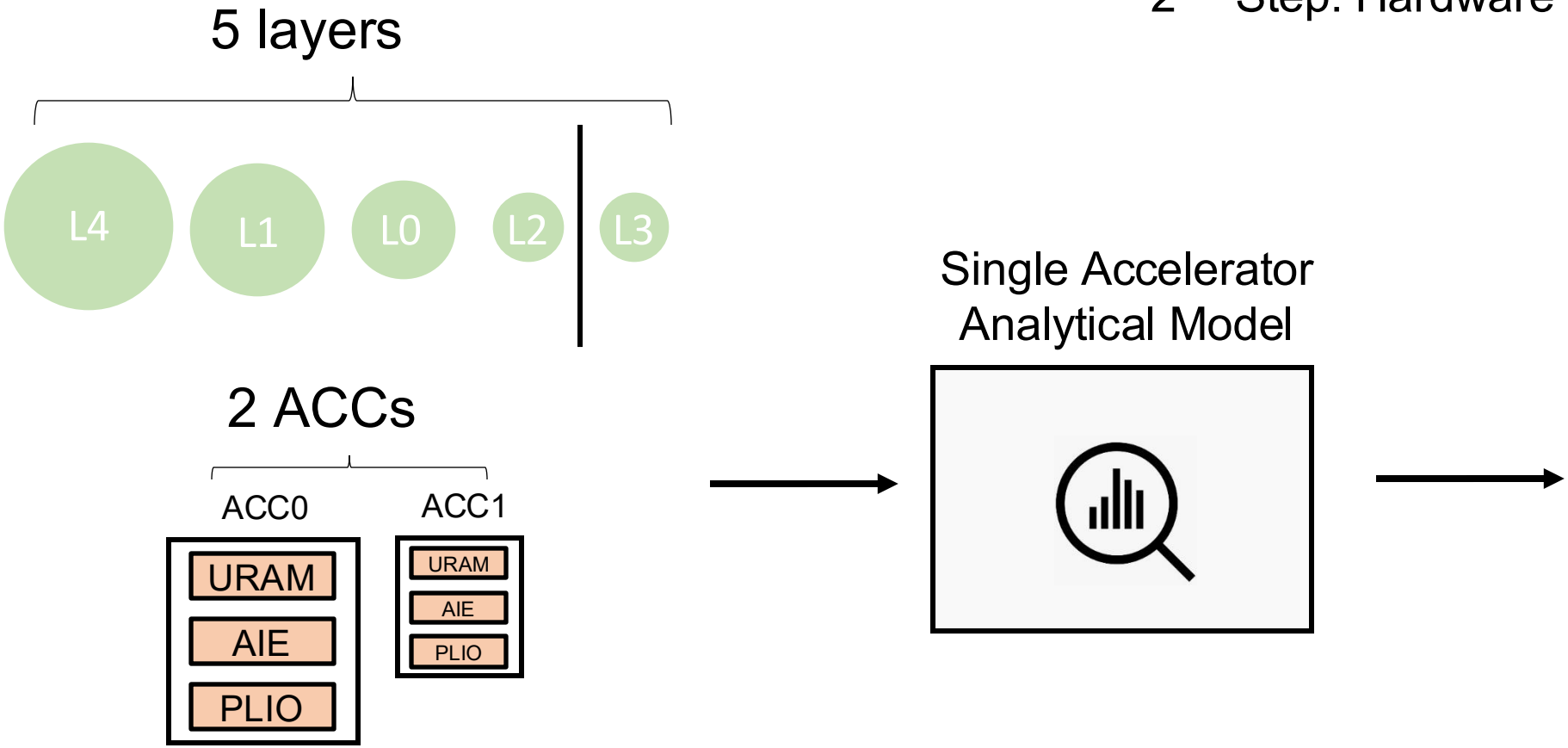
- **Two-step Search Algorithm**

How to assign **M layers** in one AI Model to **N accelerators**?

① ②

1st Step: Workload Assignment

2nd Step: Hardware Resource Partitioning



CHARM: Diverse Accelerator Composing

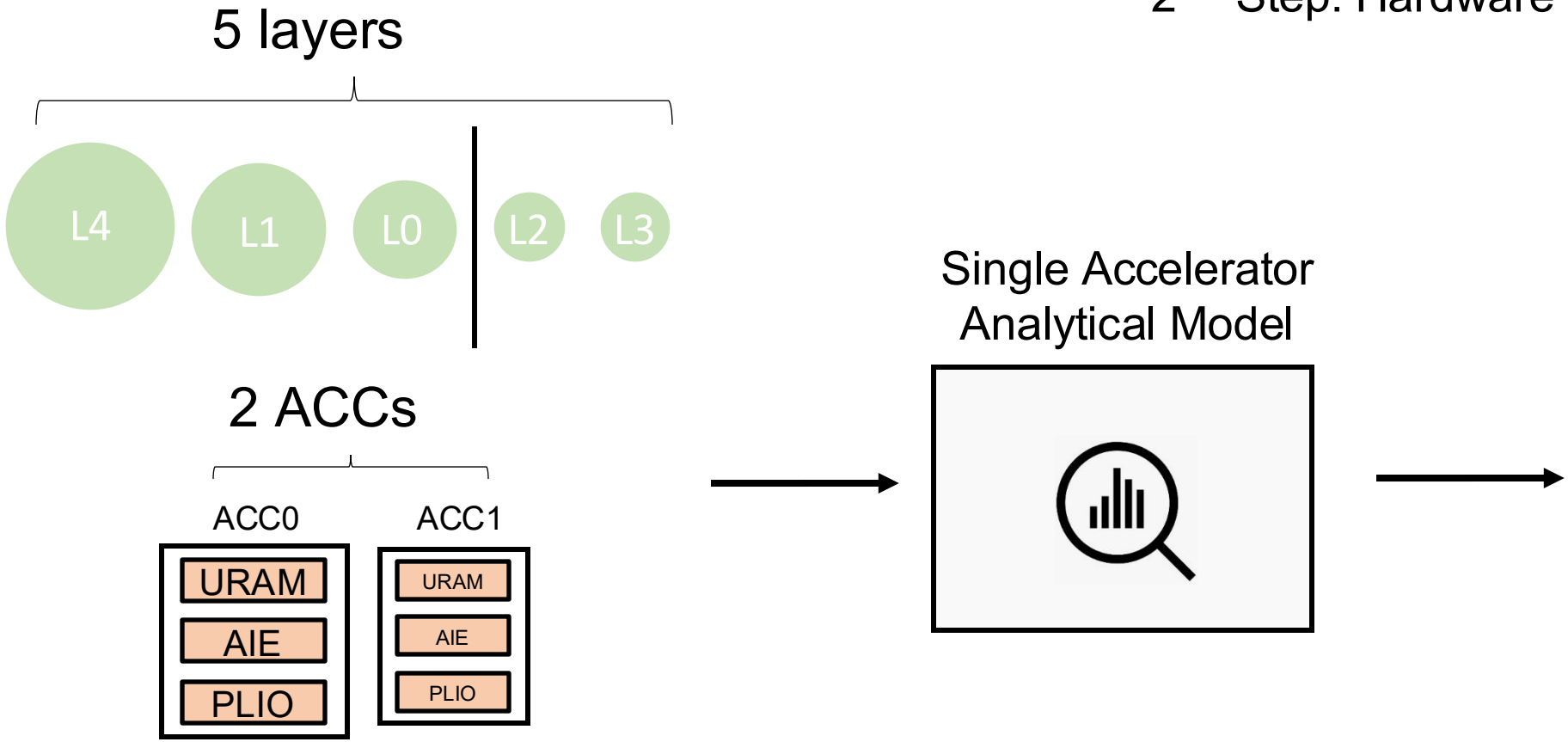
- **Two-step Search Algorithm**

How to assign **M layers** in one AI Model to **N accelerators**?

① ②

1st Step: Workload Assignment

2nd Step: Hardware Resource Partitioning



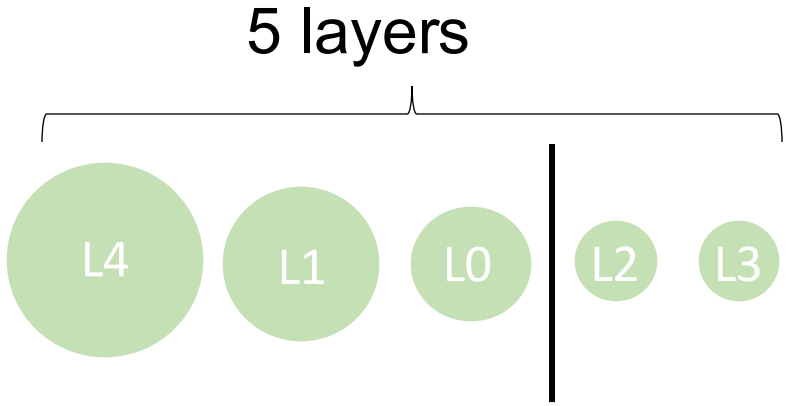
CHARM: Diverse Accelerator Composing

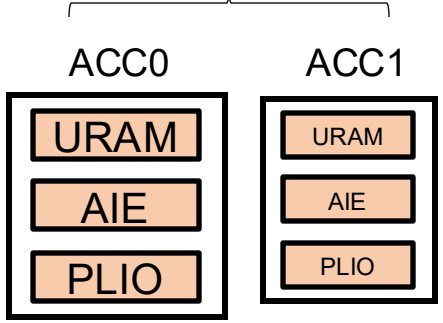
- **Two-step Search Algorithm**

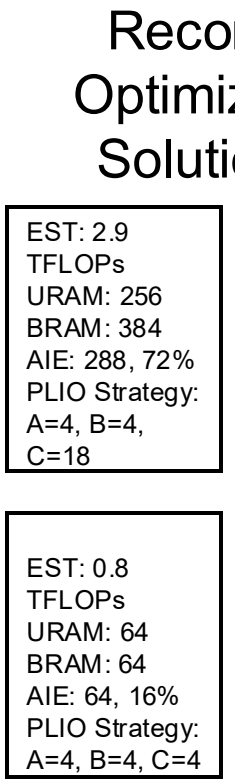
How to assign **M layers** (1) in one AI Model to **N accelerators** (2)?

1st Step: Workload Assignment

2nd Step: Hardware Resource Partitioning







Overall

2.0 TFLOPs

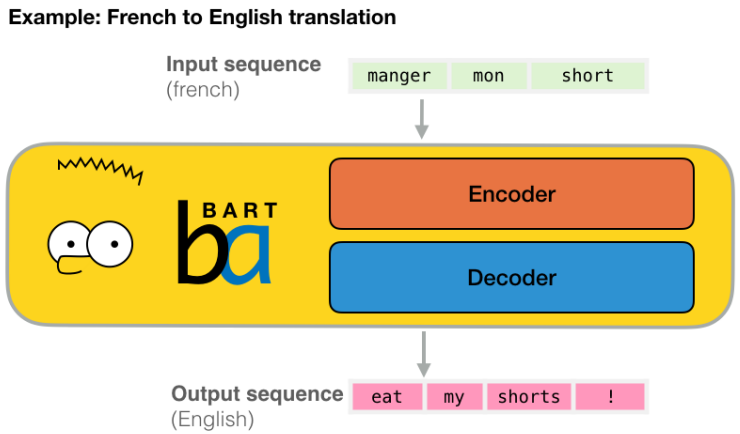
Experiment Results

- Applications

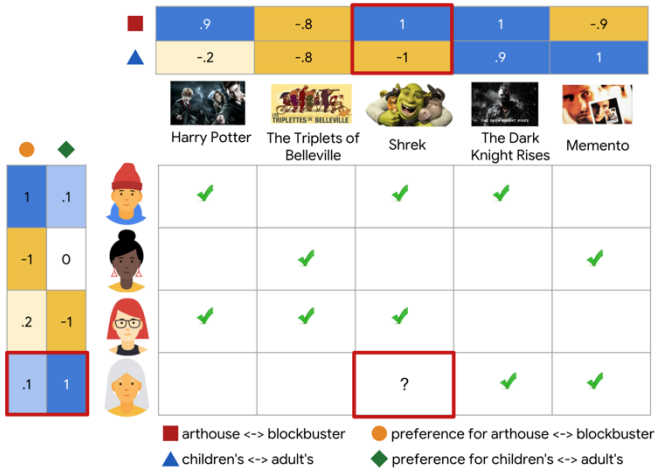
Experiment Results

- Applications

Bert-Large
(Natural Language Processing)

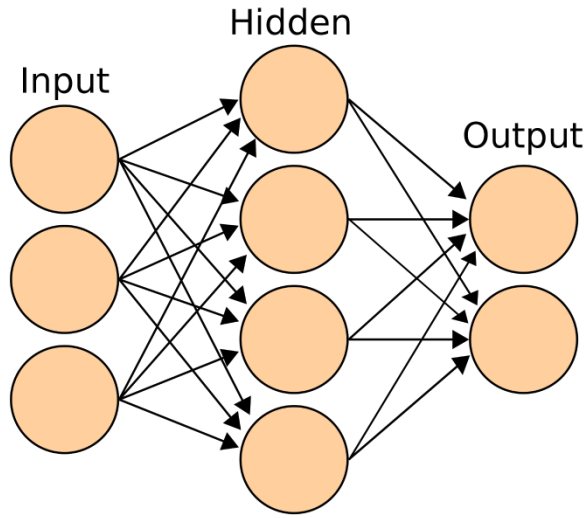


Neural Collaborative Filtering
(Recommendation)



Vision Transformer (Classification)

Multilayer Perceptrons
(Regression, Classification)



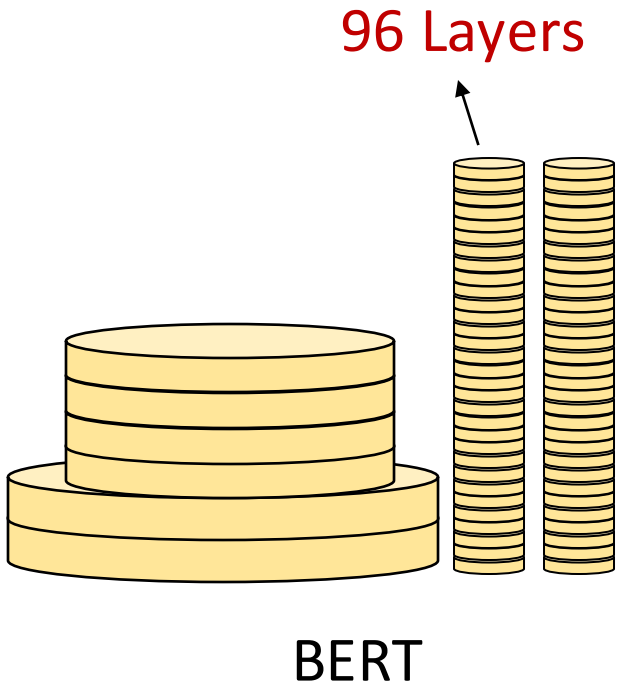
Experiment Results

- Applications Characteristics

Experiment Results

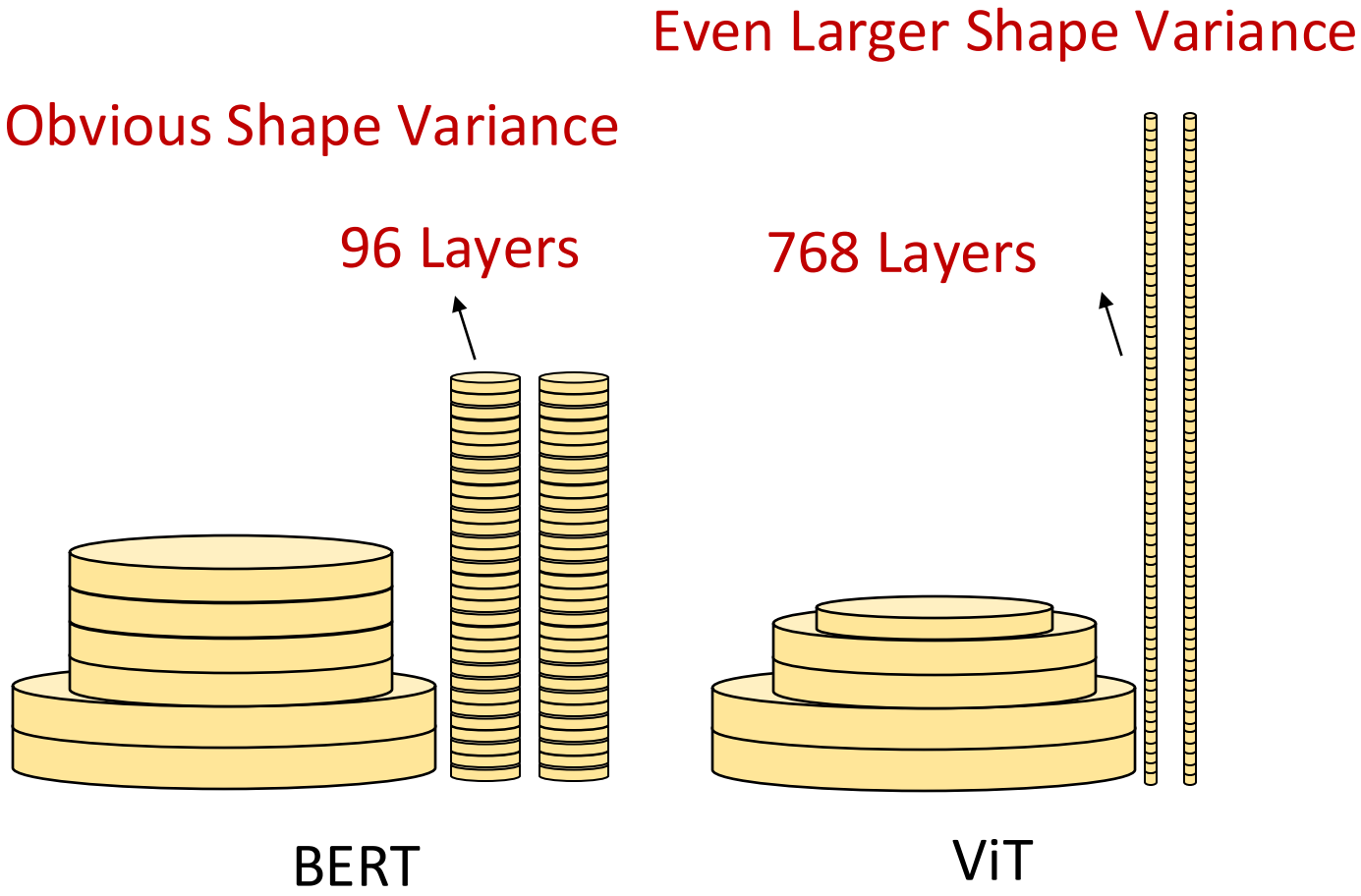
- Applications Characteristics

Obvious Shape Variance



Experiment Results

- Applications Characteristics



Experiment Results

- Applications Characteristics

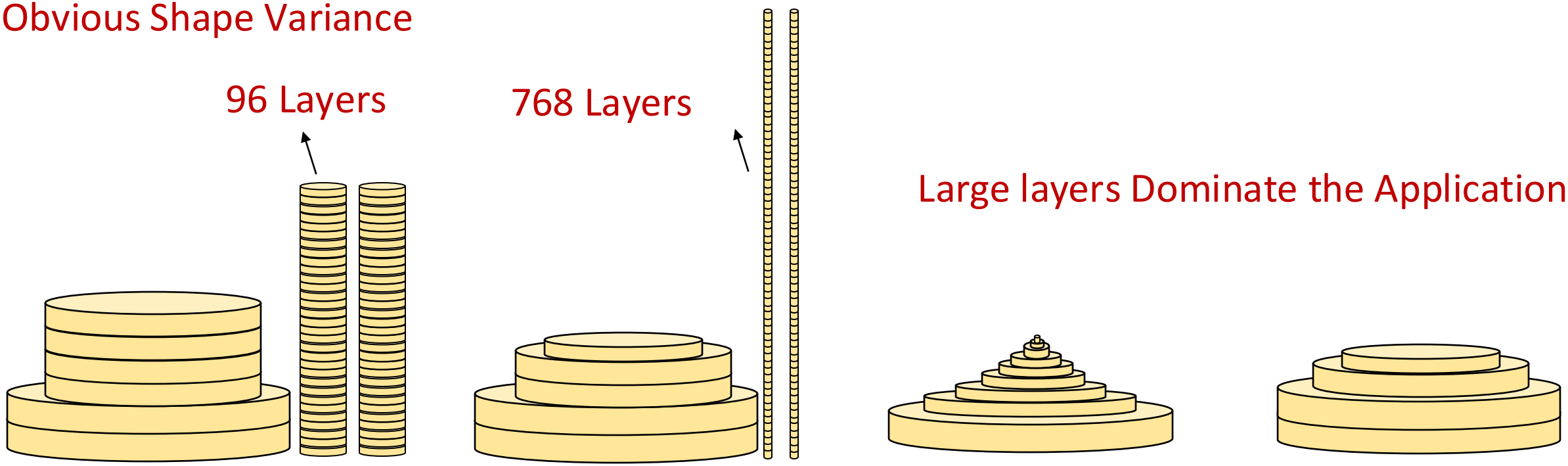
Even Larger Shape Variance

Obvious Shape Variance

96 Layers

768 Layers

Large layers Dominate the Application



BERT

ViT

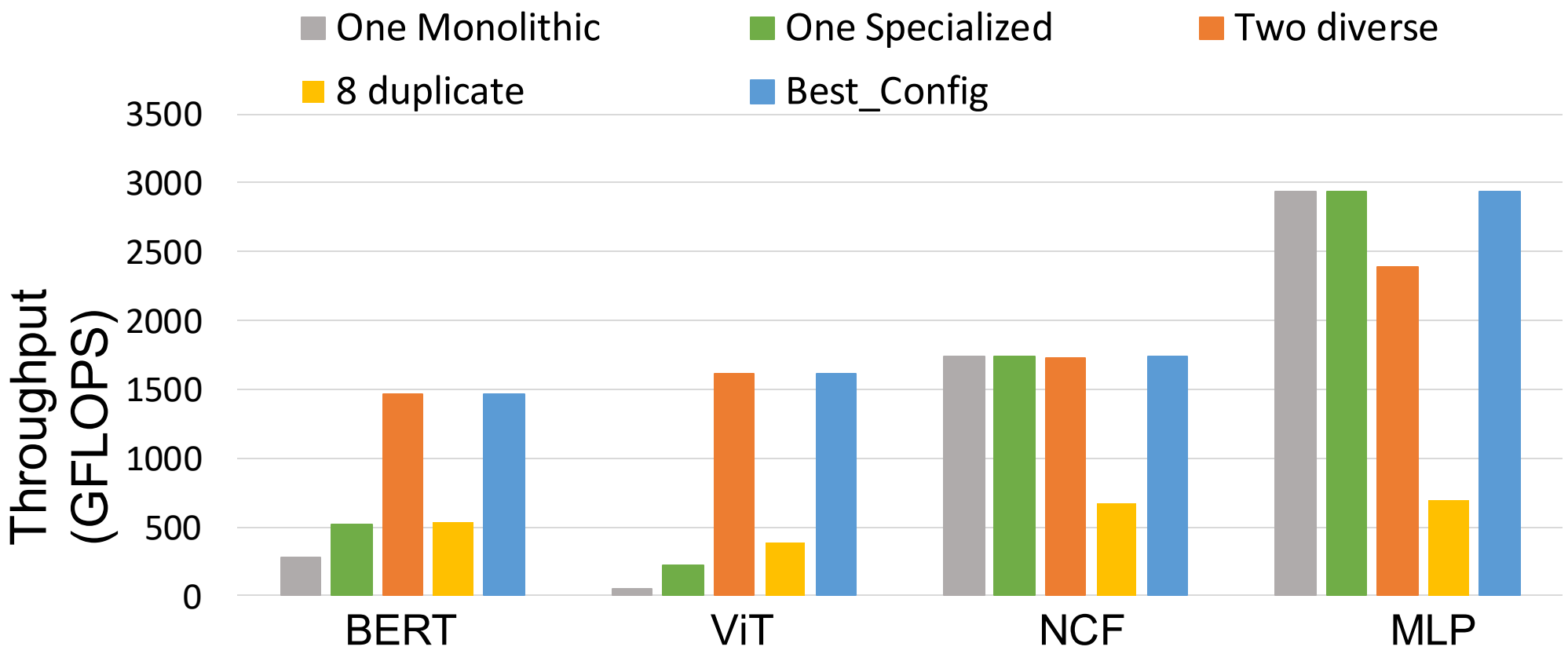
NCF

MLP

Experiment Results

- **Composing Diverse Accelerators**

  Throughput Comparison Among Different CHARM Strategies



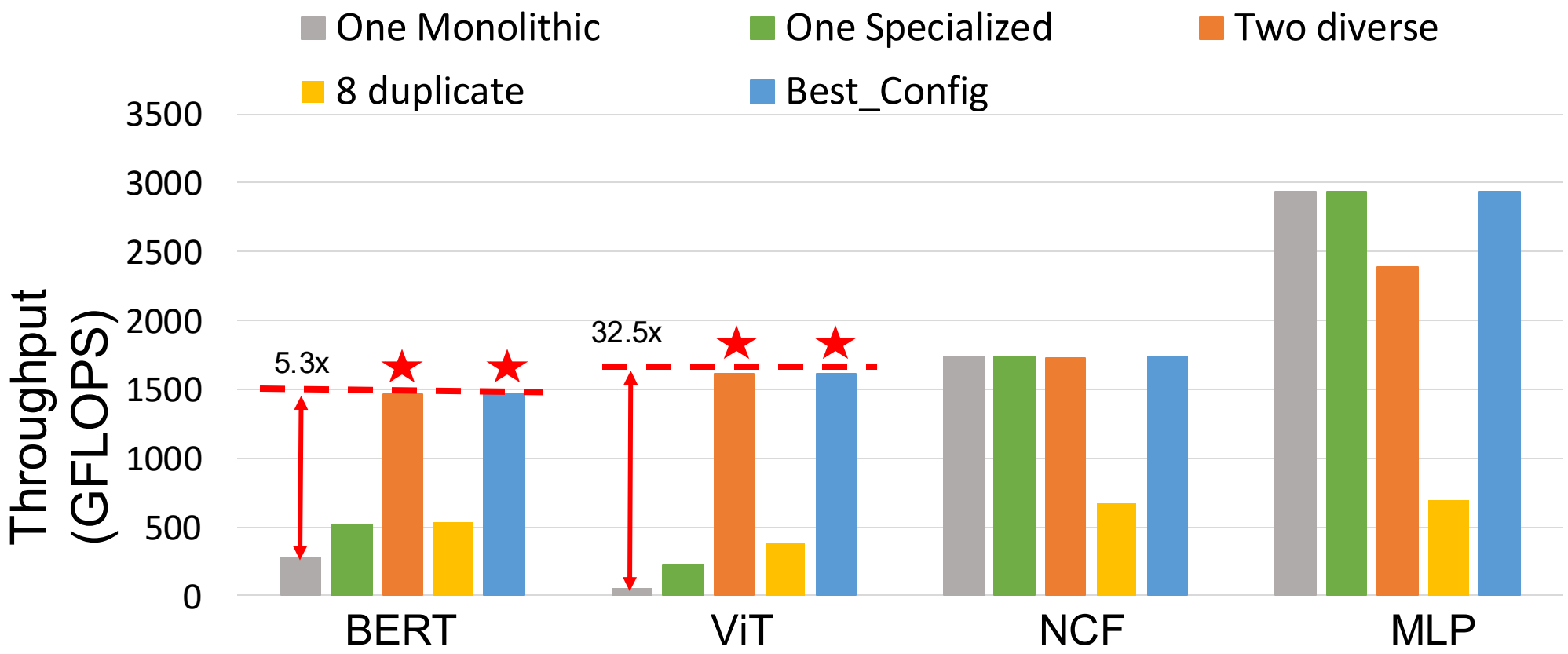
Experiment Results

- **Composing Diverse Accelerators**

  Throughput Comparison Among Different CHARM Strategies

BERT: 1.46 TFLOPS, 5.3x gain over one monolithic design

ViT: 1.61 TFLOPS, 32.5x gain over one monolithic design



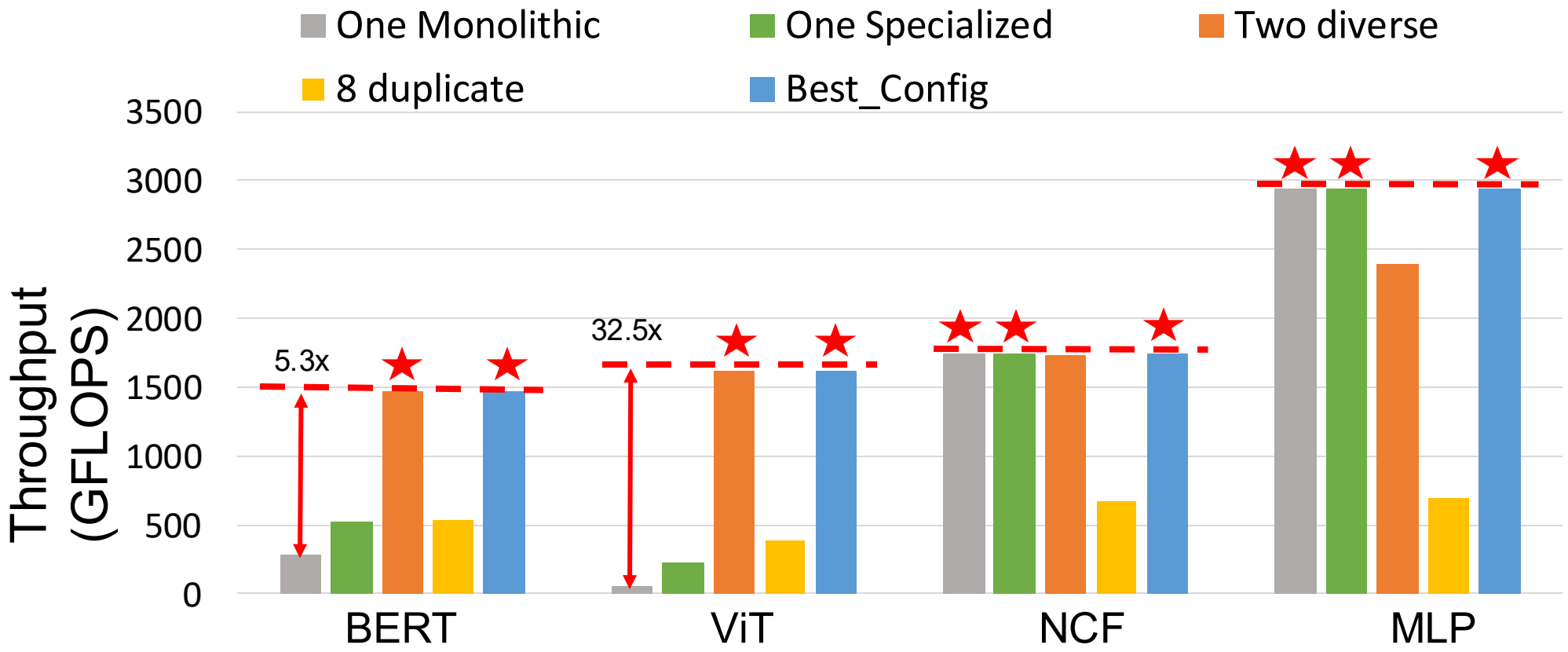
Experiment Results

- **Composing Diverse Accelerators**

  Throughput Comparison Among Different CHARM Strategies

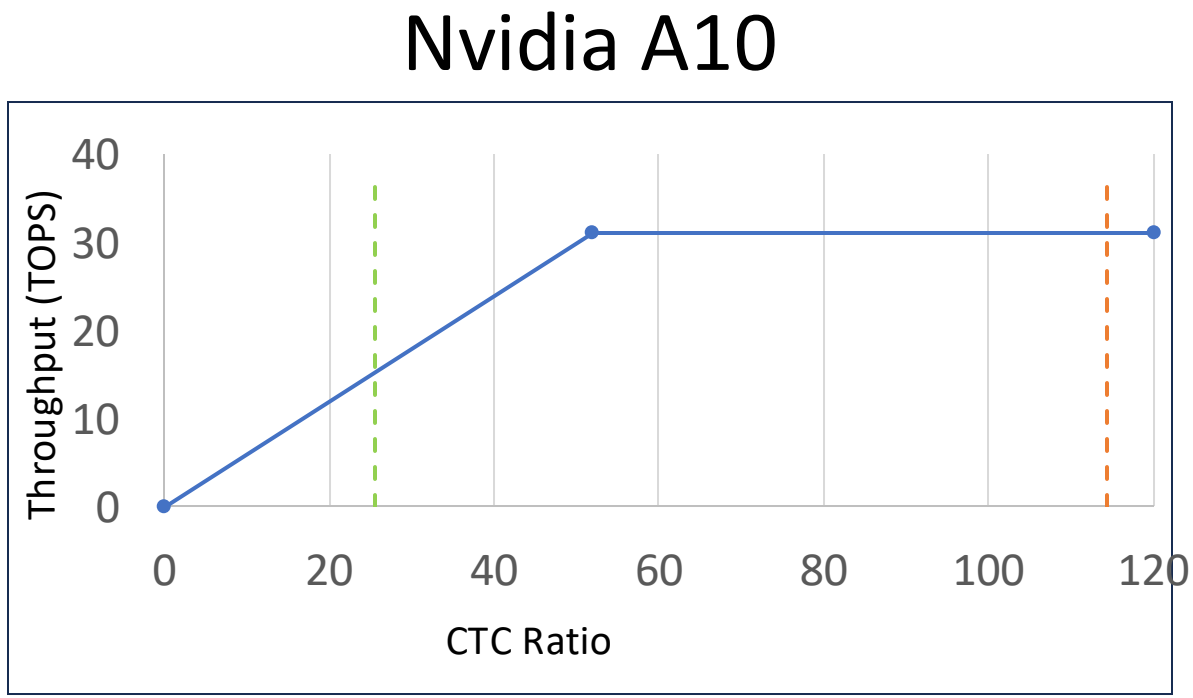
BERT: 1.46 TFLOPS, 5.3x gain over one monolithic design NCF: 1.74 TFLOPS for one accelerator design

ViT: 1.61 TFLOPS, 32.5x gain over one monolithic design MLP: 2.94 TFLOPS for one accelerator design

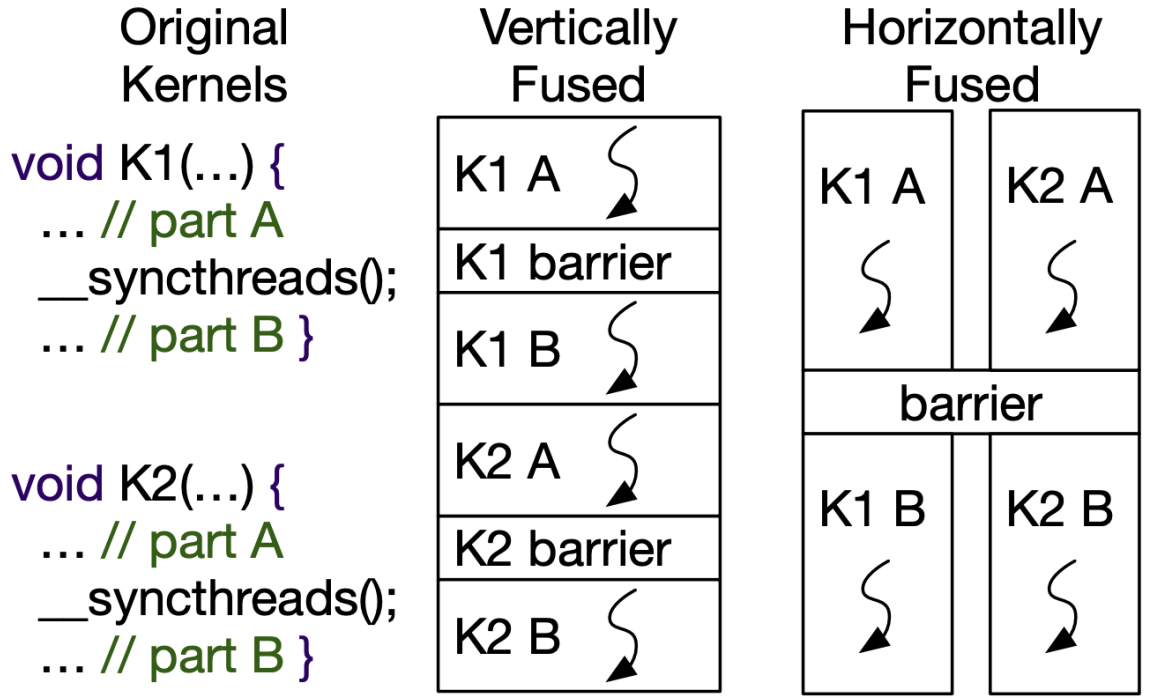


Apply CHARM to Other Platforms, GPUs?

- **Provided spatial flexibility, combine communication-bound MM with computation-bound MM**



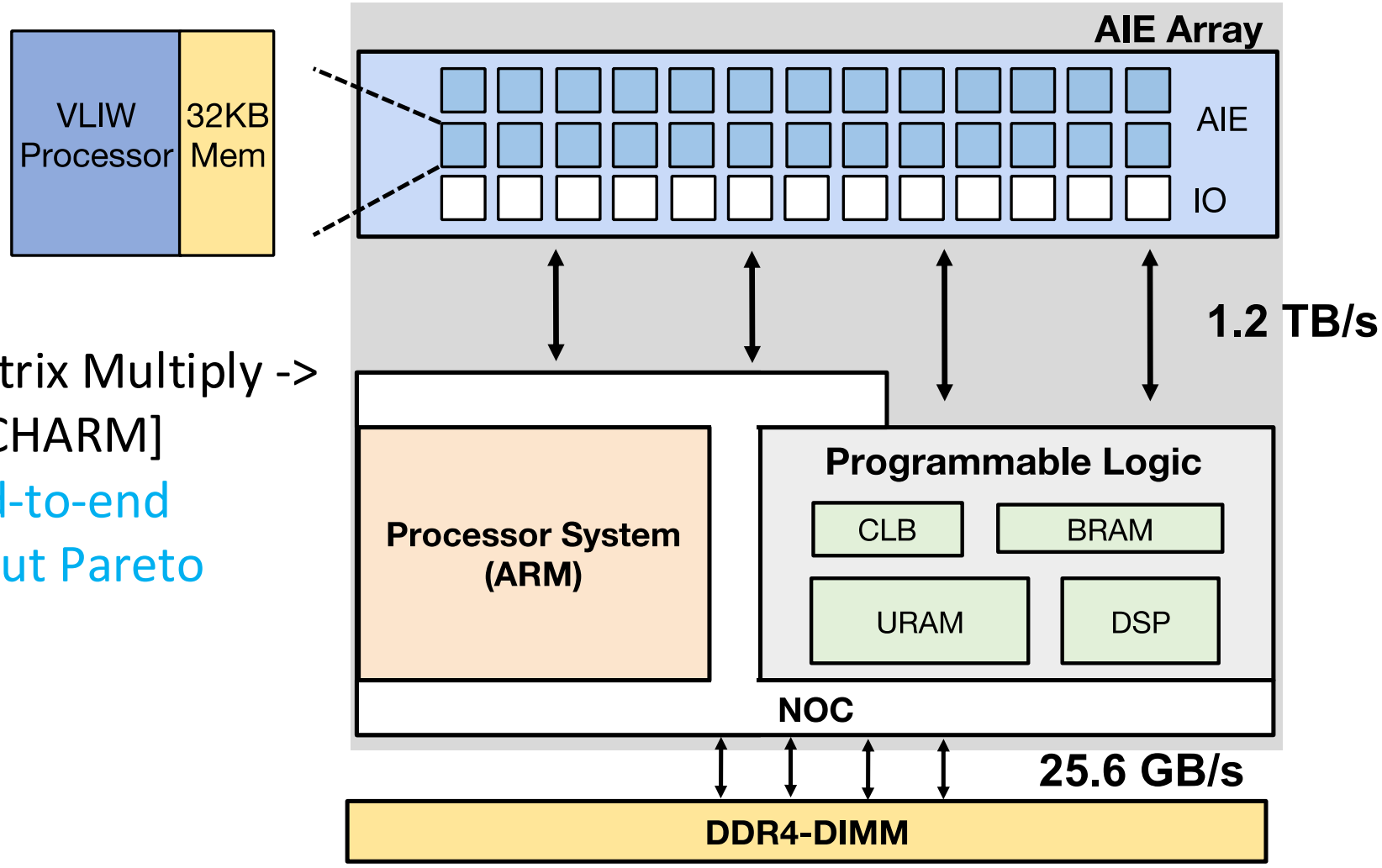
1.37x Gain



Vertical and horizontal kernel fusion in GPU.

“X FPGA” => Versal ACAP Architecture

AMD ACAP
Heterogeneous SoC
FPGA + AI Tensor Cores + CPU

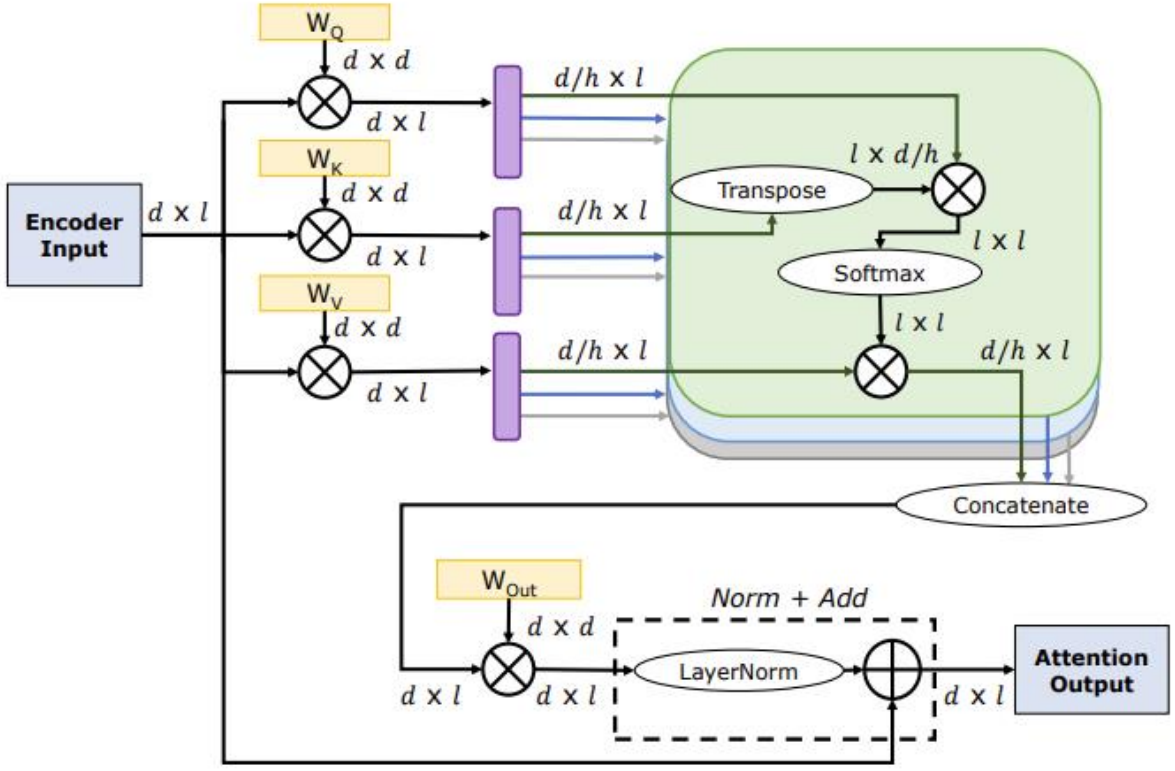


- Composing Diverse Accs for Matrix Multiply -> Throughput Optimal [FPGA’23 CHARM]
- Composing Diverse Accs for End-to-end Applications, Latency-Throughput Pareto [FPGA’24 SSR]

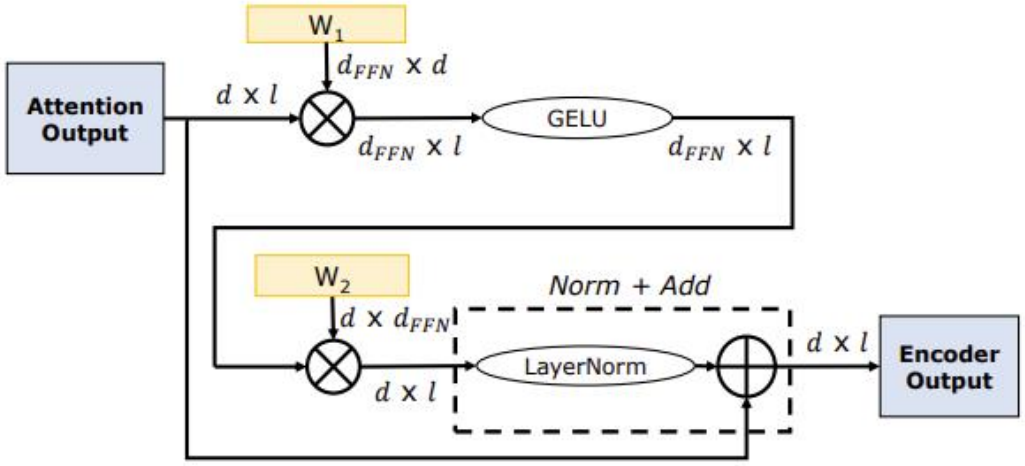
End-to-end Deep Learning, e.g., Transformer

- Element-wise & Non-linear Kernels (CTC <=1)

Non-MM kernels <1% total FLOPS,
~ 50% execution time in GPU



Muti-Head Attention (MHA) Module

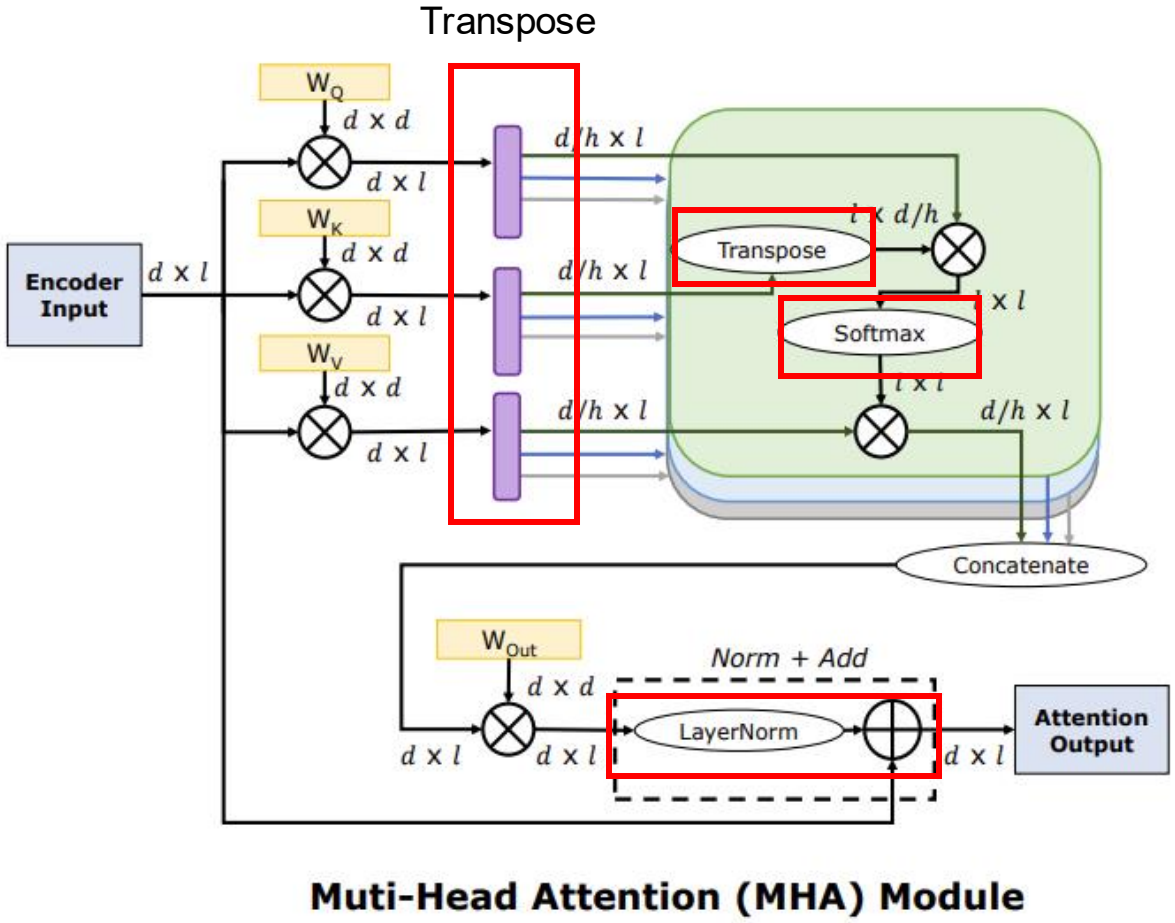


Feed-Forward Network (FFN) Module

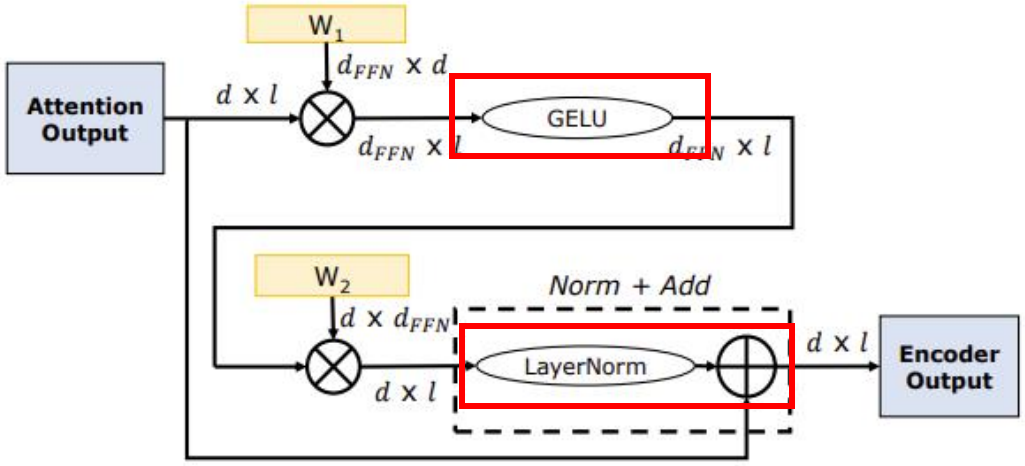
End-to-end Deep Learning, e.g., Transformer

- Element-wise & Non-linear Kernels (CTC <=1)

Non-MM kernels <1% total FLOPS,
~ 50% execution time in GPU

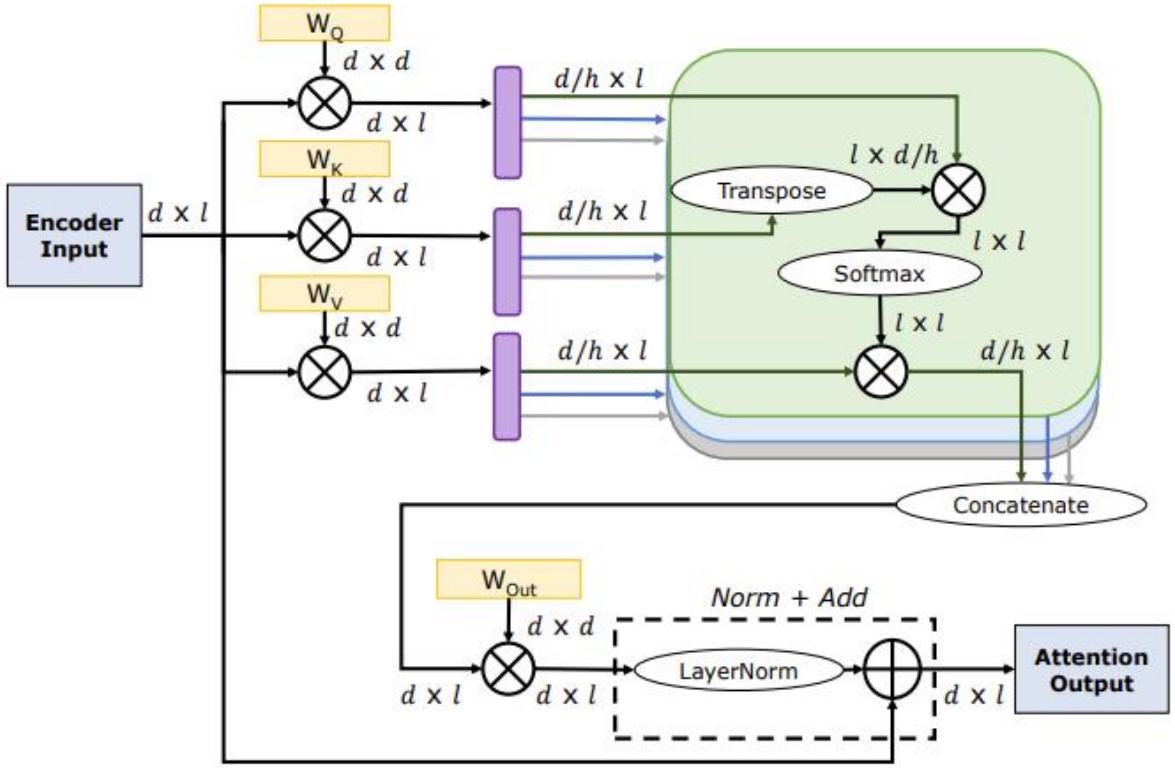


Muti-Head Attention (MHA) Module

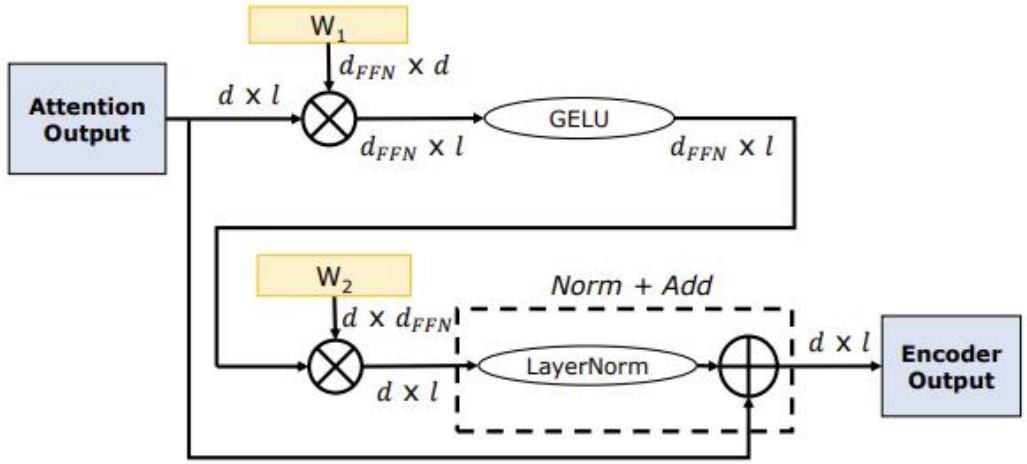


Feed-Forward Network (FFN) Module

End-to-end Deep Learning, e.g., Transformer



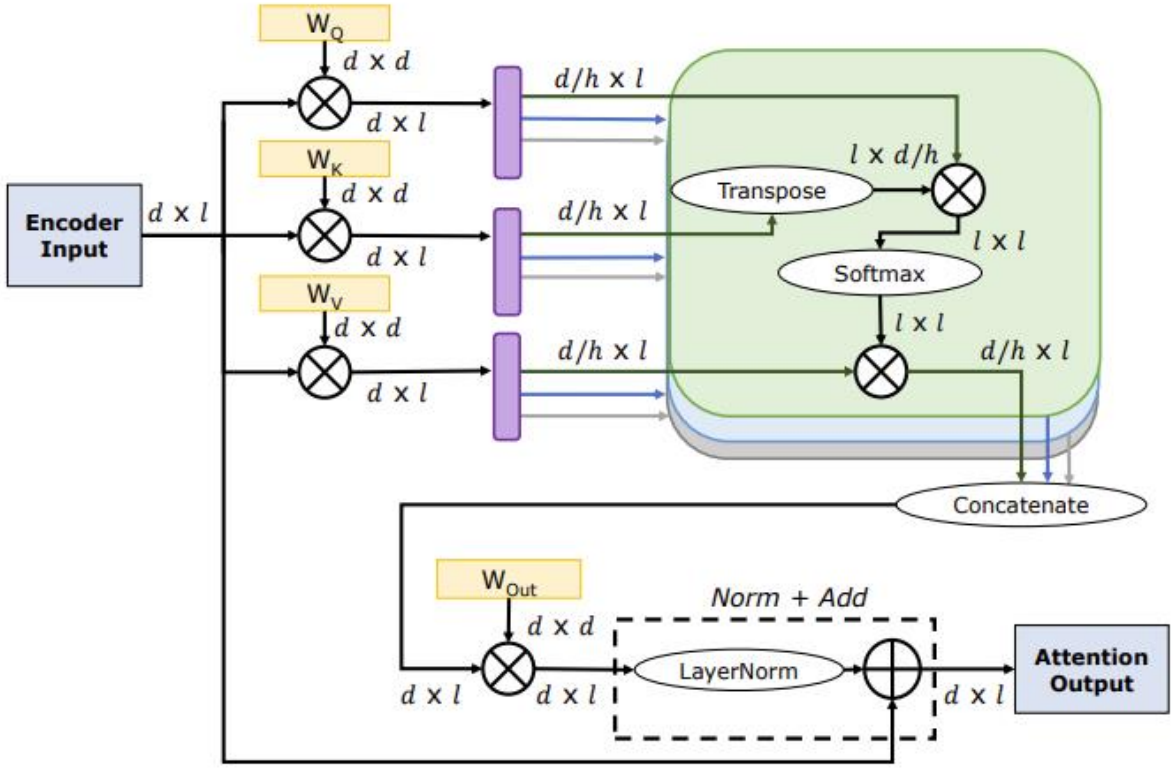
Muti-Head Attention (MHA) Module



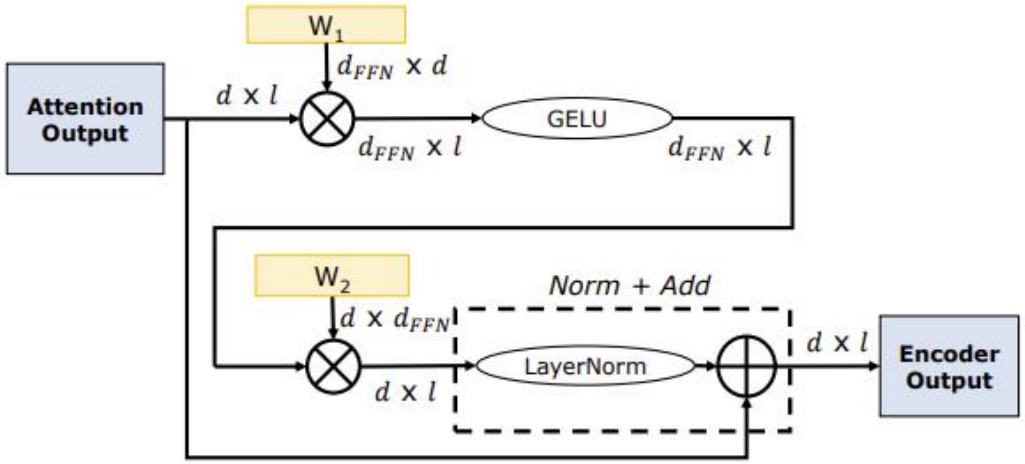
Feed-Forward Network (FFN) Module

End-to-end Deep Learning, e.g., Transformer

- Computation-Intensive Kernels



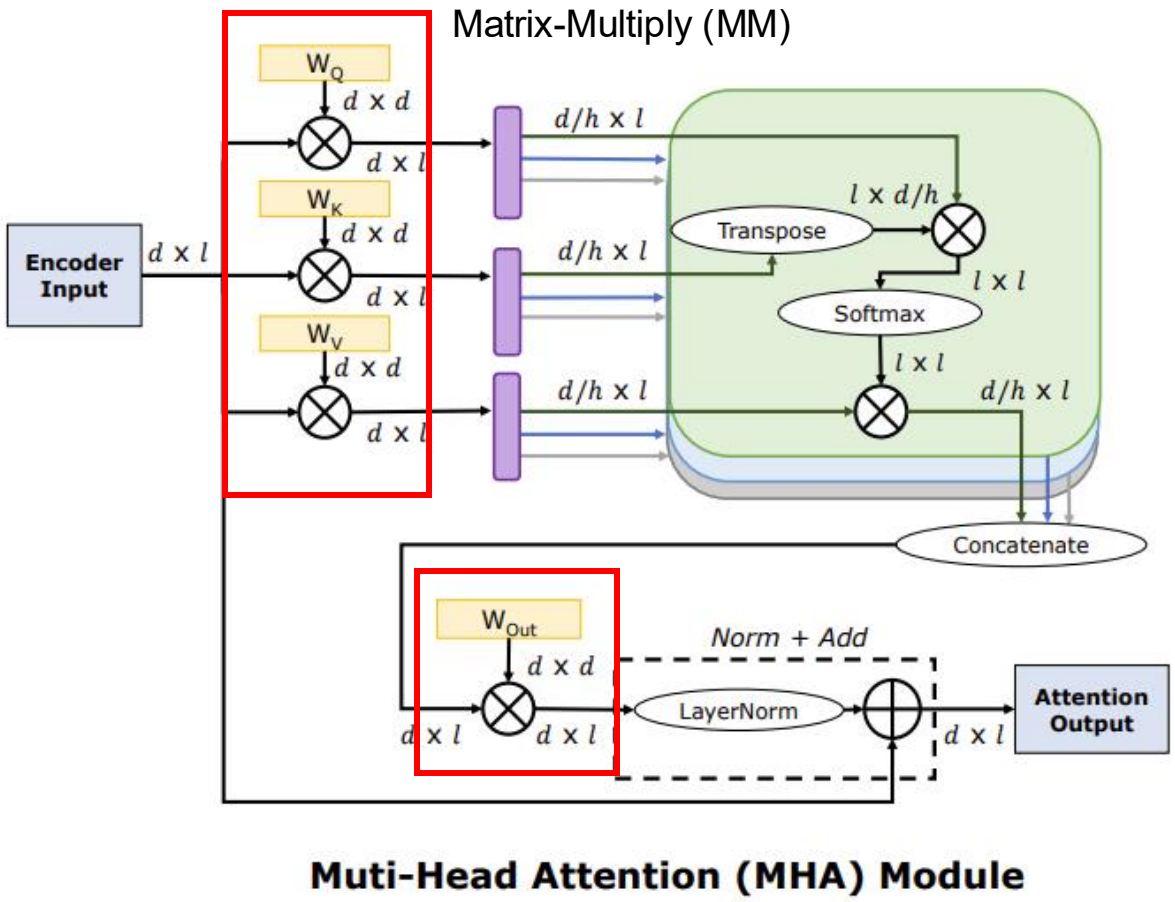
Muti-Head Attention (MHA) Module

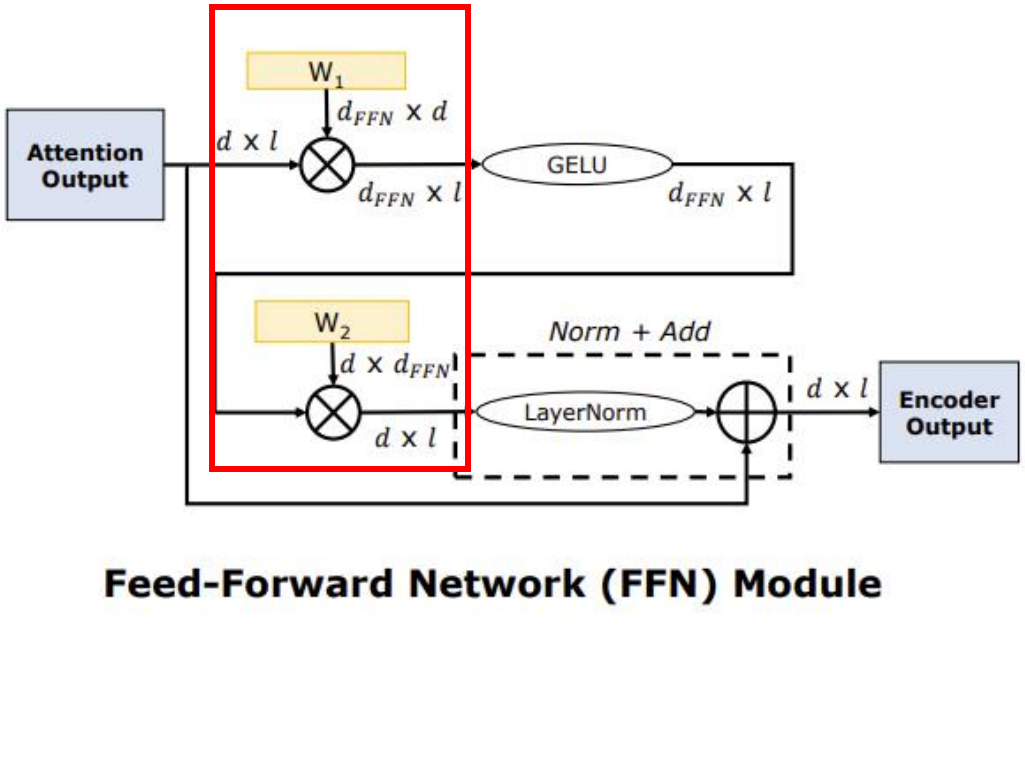


Feed-Forward Network (FFN) Module

End-to-end Deep Learning, e.g., Transformer

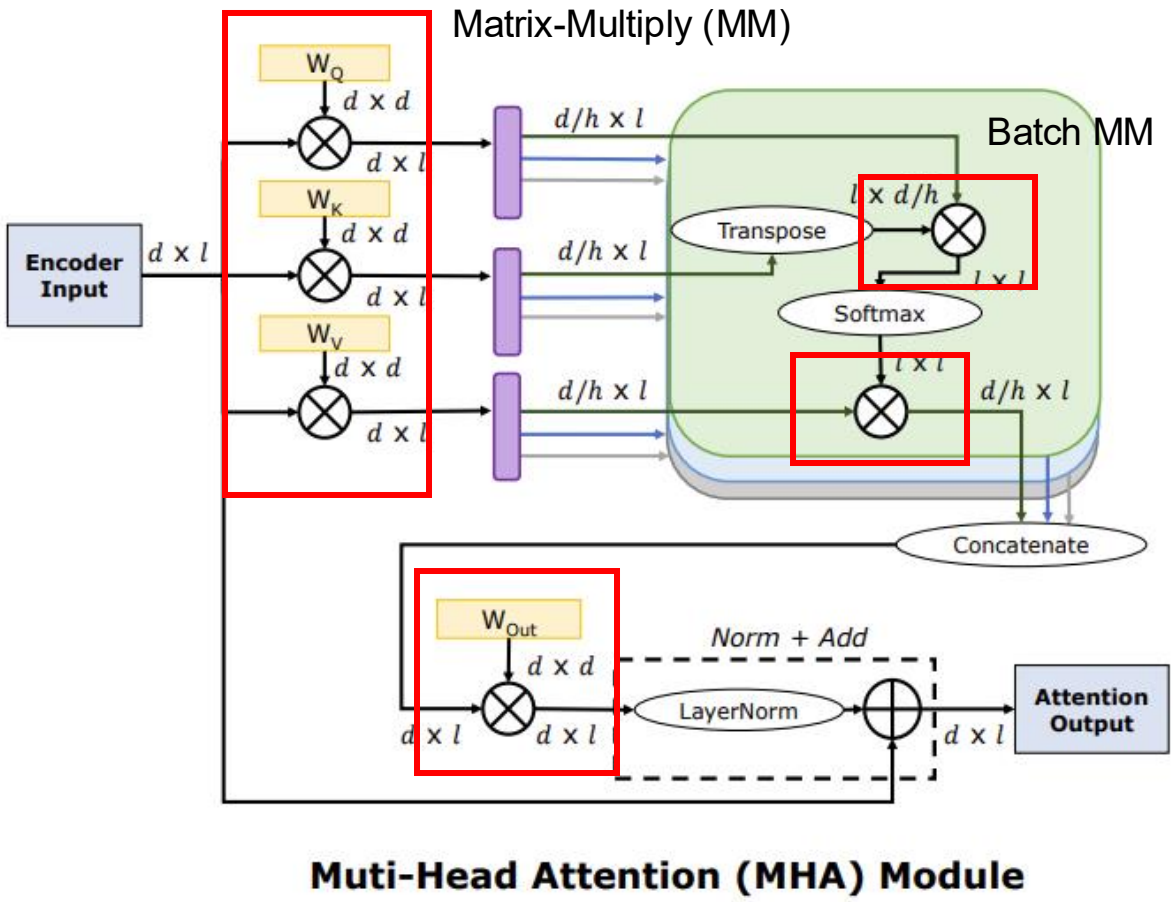
- Computation-Intensive Kernels



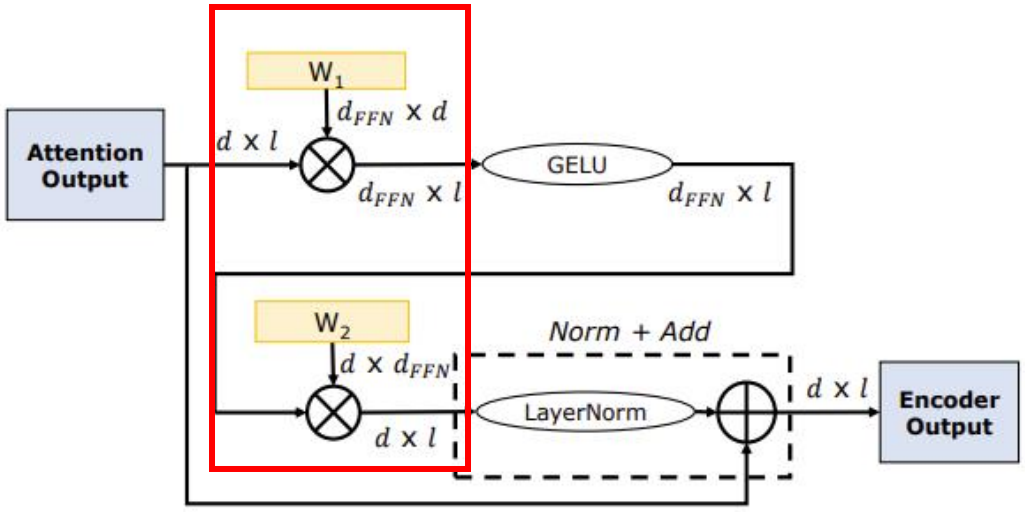


End-to-end Deep Learning, e.g., Transformer

- Computation-Intensive Kernels



Muti-Head Attention (MHA) Module



Feed-Forward Network (FFN) Module

Latency & Throughput Driven Scenarios

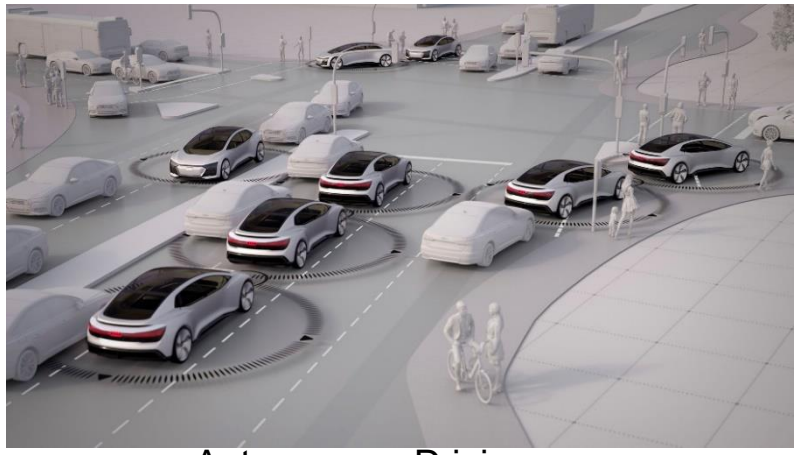
Latency & Throughput Driven Scenarios

Latency-Driven

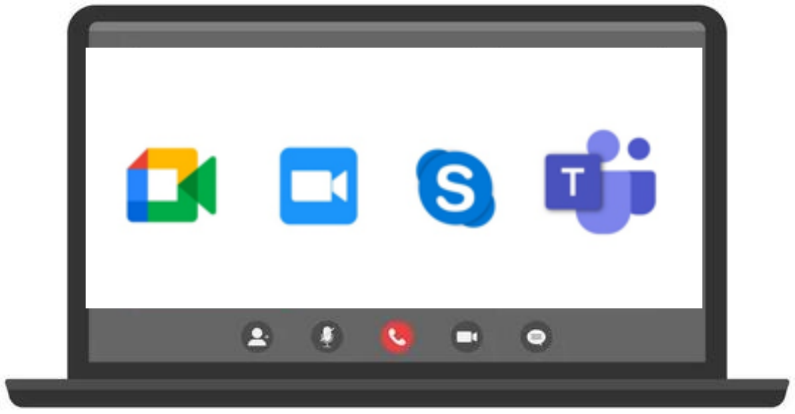
Throughput-Driven

Latency & Throughput Driven Scenarios

Latency-Driven



Autonomous Driving

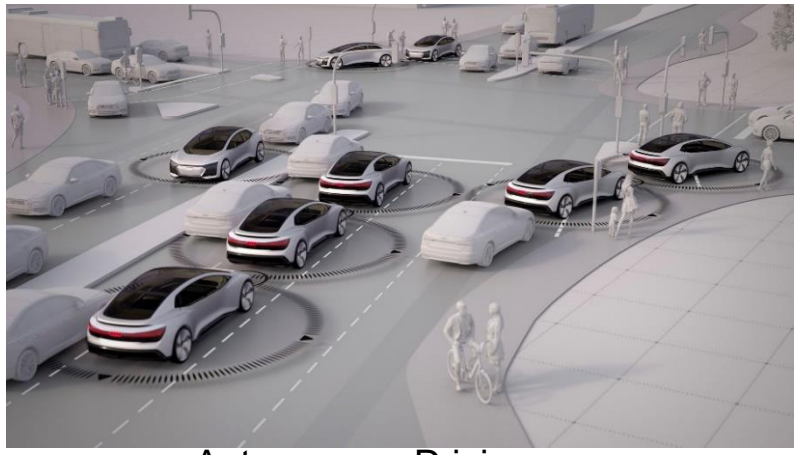


Video Conferencing

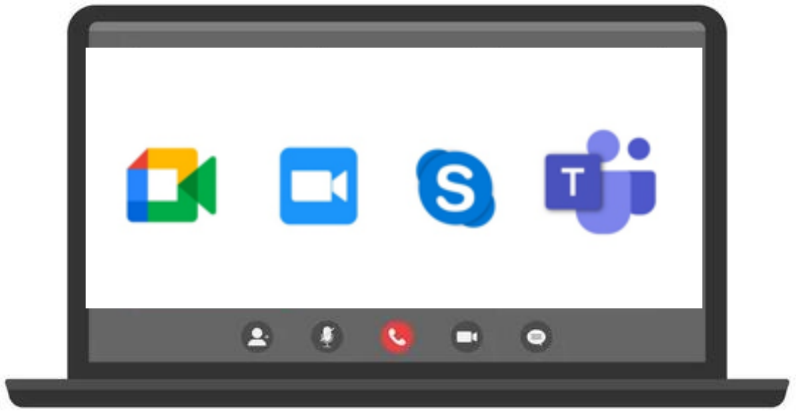
Throughput-Driven

Latency & Throughput Driven Scenarios

Latency-Driven

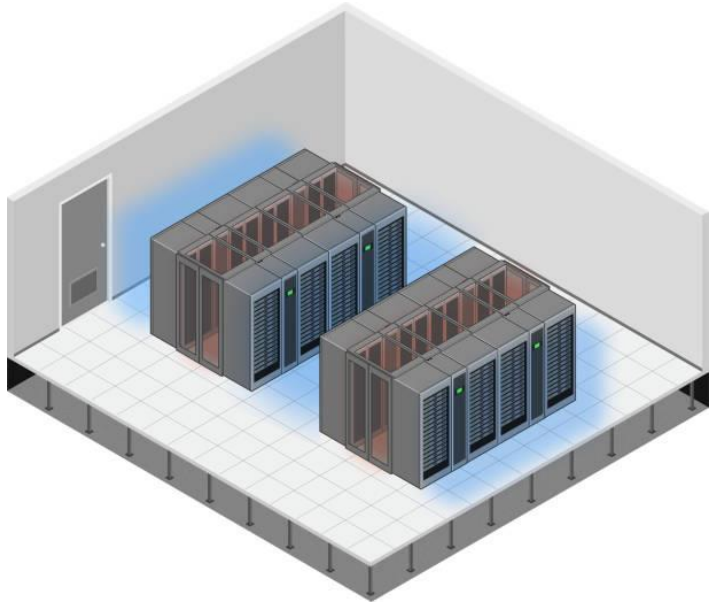


Autonomous Driving



Video Conferencing

Throughput-Driven



Google Microsoft aws ...

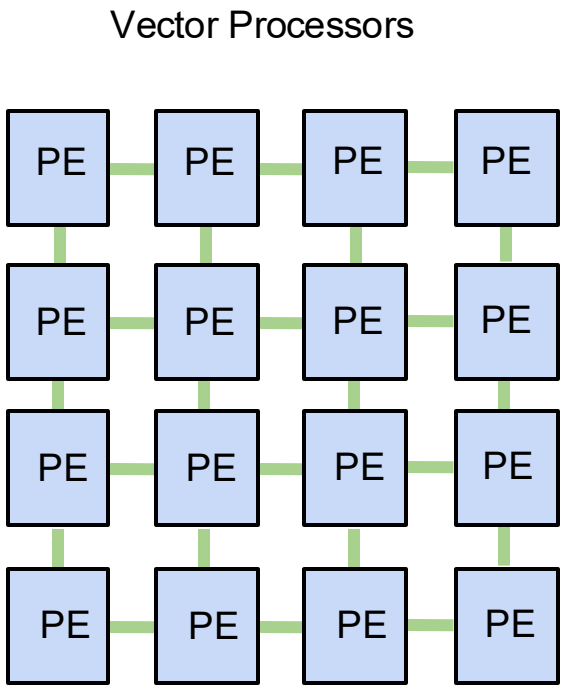
Data Center Services

Motivation: Sequential or Spatial Arch?

- Sequential-only Accelerator → Hardware Under-utilization

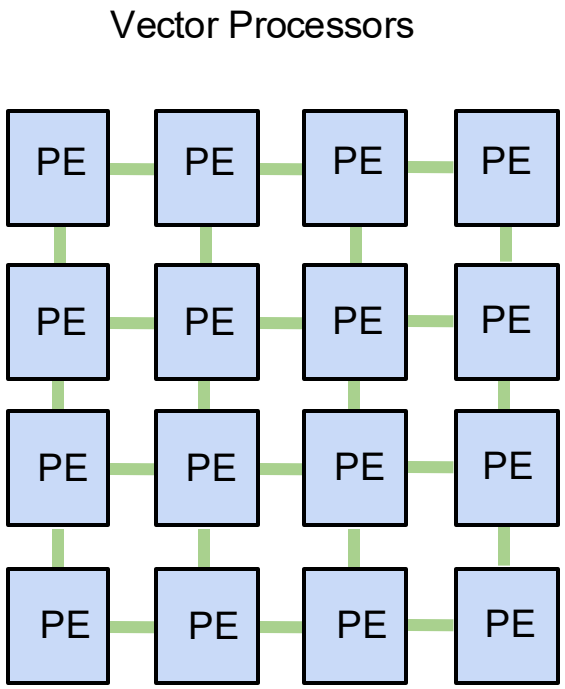
Motivation: Sequential or Spatial Arch?

- Sequential-only Accelerator → Hardware Under-utilization



Motivation: Sequential or Spatial Arch?

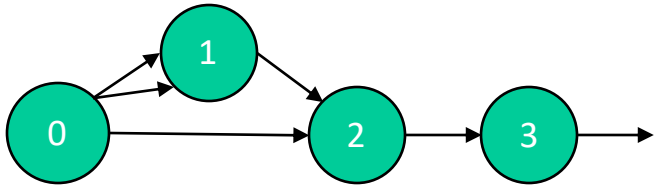
- Sequential-only Accelerator → Hardware Under-utilization
  - Mismatch between shape of the layers & size of the accelerator



Motivation: Sequential or Spatial Arch?

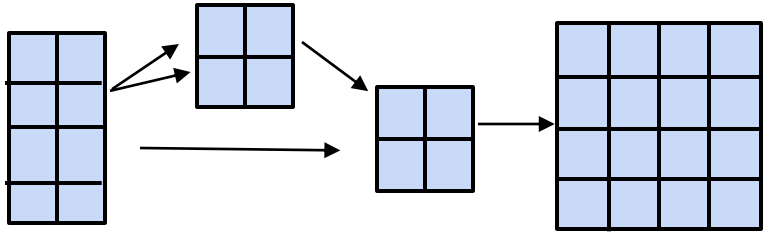
- Sequential-only Accelerator → Hardware Under-utilization
  - Mismatch between shape of the layers & size of the accelerator

Directed acyclic graph (DAG)

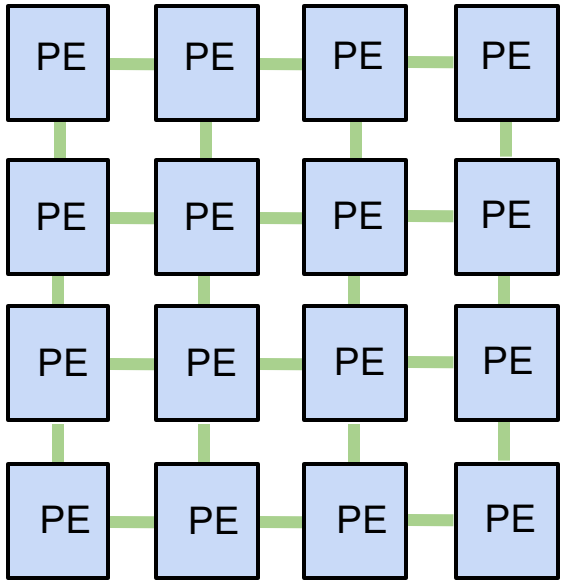


Corresponding Workloads

8 : 4 : 4 : 16



Vector Processors



Scheduling

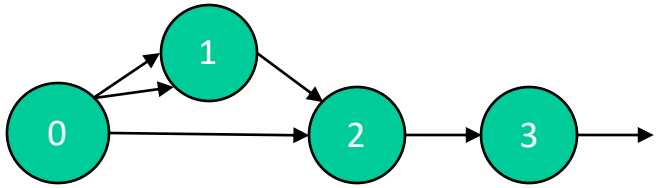
ACC0

Timeline

Motivation: Sequential or Spatial Arch?

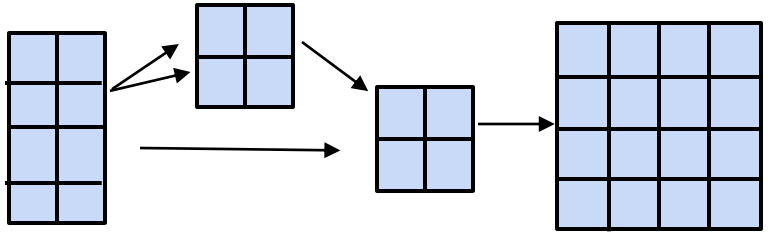
- Sequential-only Accelerator → Hardware Under-utilization
  - Mismatch between shape of the layers & size of the accelerator

Directed acyclic graph (DAG)

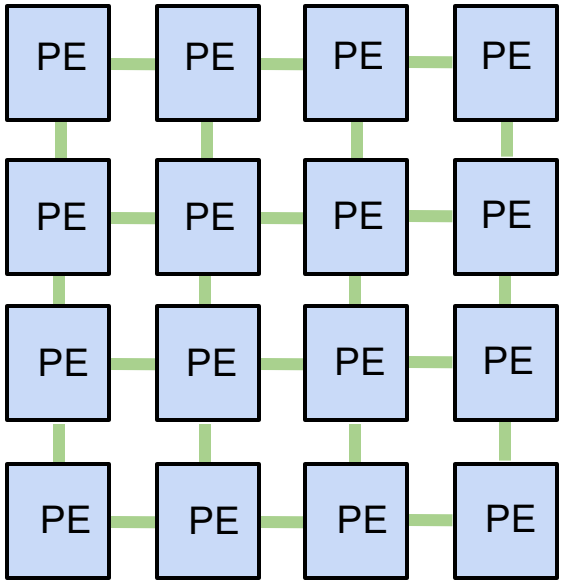


Corresponding Workloads

8 : 4 : 4 : 16



Vector Processors



Scheduling Layer 0

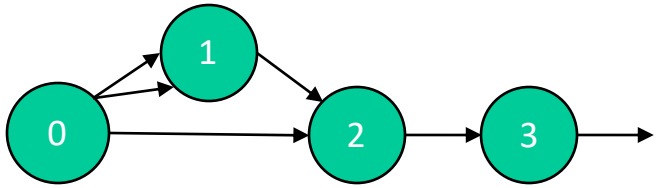
ACC0

Timeline

Motivation: Sequential or Spatial Arch?

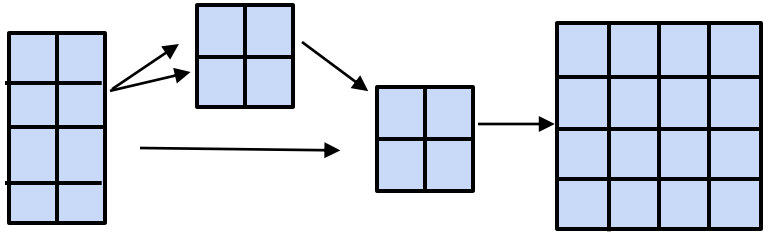
- Sequential-only Accelerator → Hardware Under-utilization
  - Mismatch between shape of the layers & size of the accelerator

Directed acyclic graph (DAG)

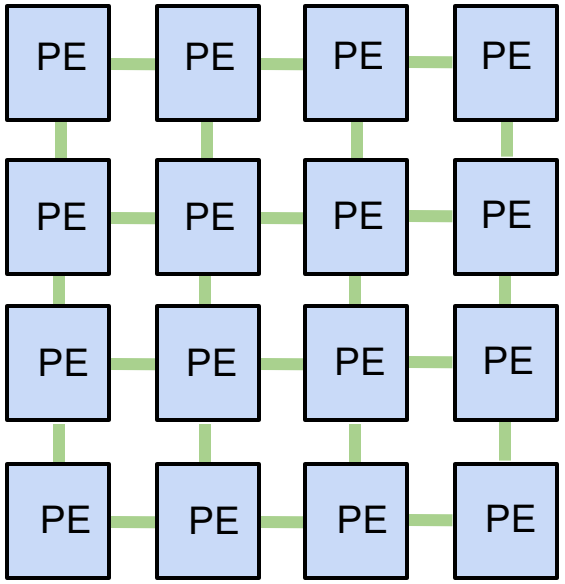


Corresponding Workloads

8 : 4 : 4 : 16



Vector Processors



Scheduling Layer 0

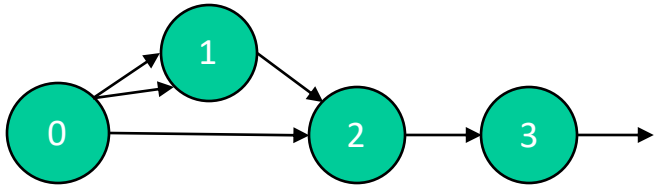
ACC0

Timeline

Motivation: Sequential or Spatial Arch?

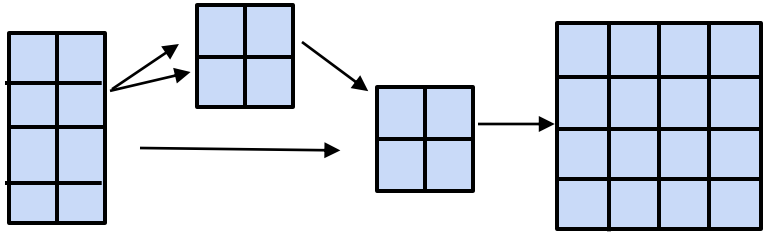
- Sequential-only Accelerator → Hardware Under-utilization
  - Mismatch between shape of the layers & size of the accelerator

Directed acyclic graph (DAG)

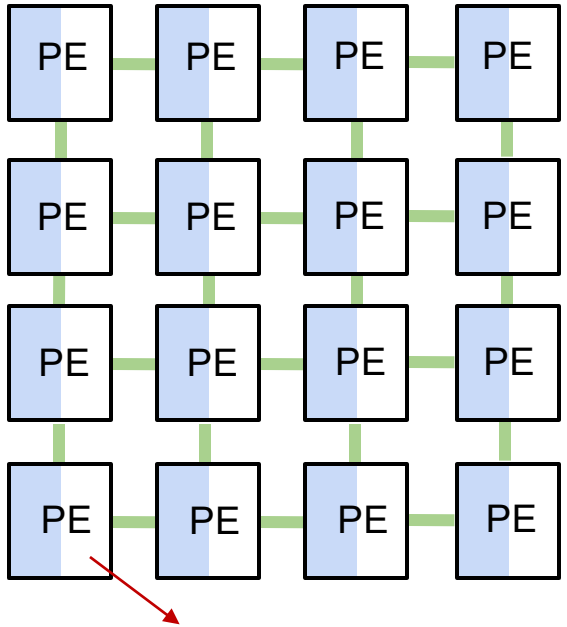


Corresponding Workloads

8 : 4 : 4 : 16

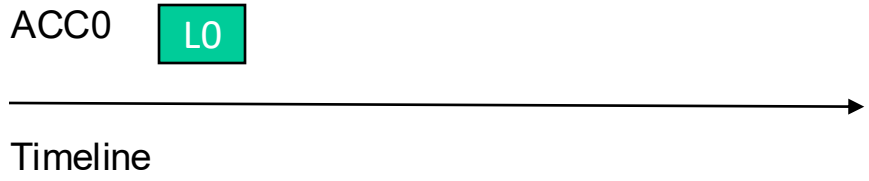


Vector Processors



Padding → Invalid Computation

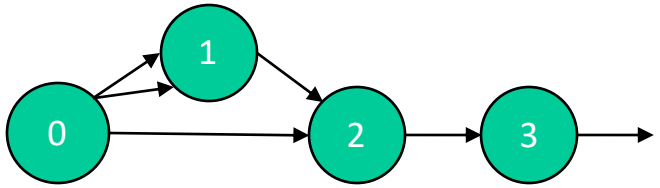
Scheduling Layer 0



Motivation: Sequential or Spatial Arch?

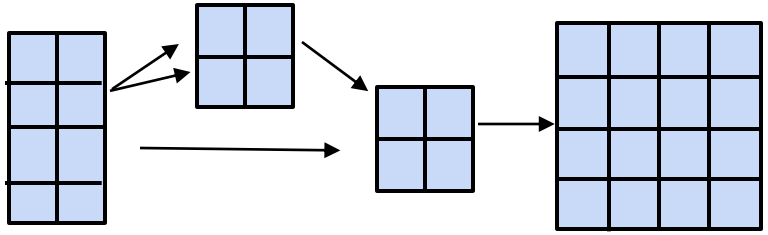
- Sequential-only Accelerator → Hardware Under-utilization
  - Mismatch between shape of the layers & size of the accelerator

Directed acyclic graph (DAG)

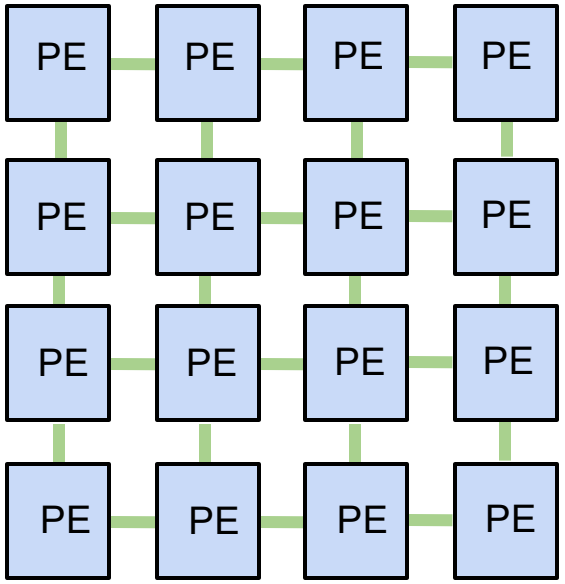


Corresponding Workloads

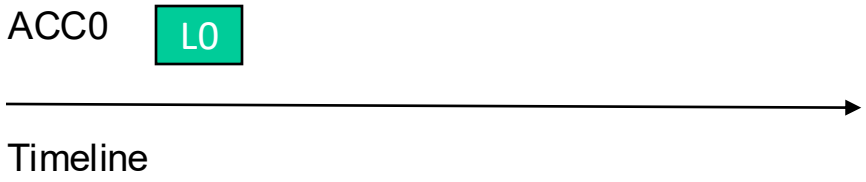
8 : 4 : 4 : 16



Tiled Architecture



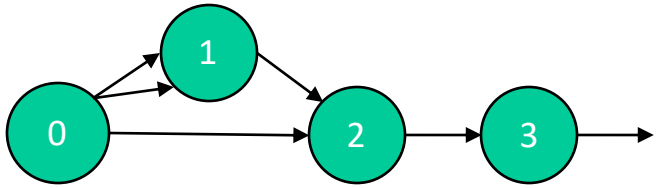
Scheduling Layer 1-2



Motivation: Sequential or Spatial Arch?

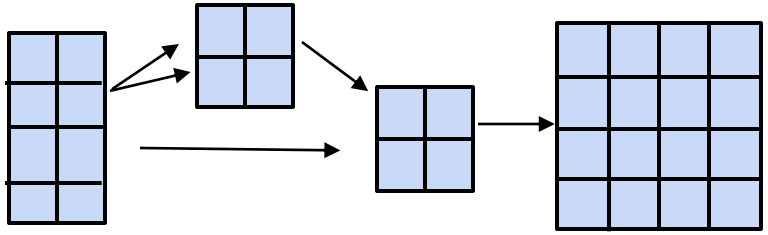
- Sequential-only Accelerator → Hardware Under-utilization
  - Mismatch between shape of the layers & size of the accelerator

Directed acyclic graph (DAG)

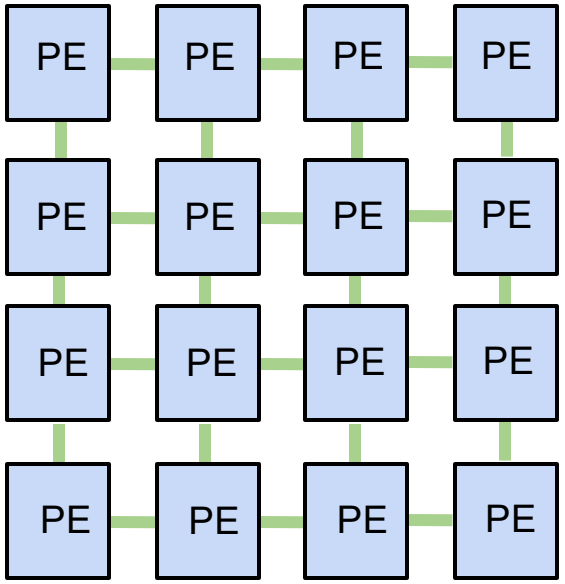


Corresponding Workloads

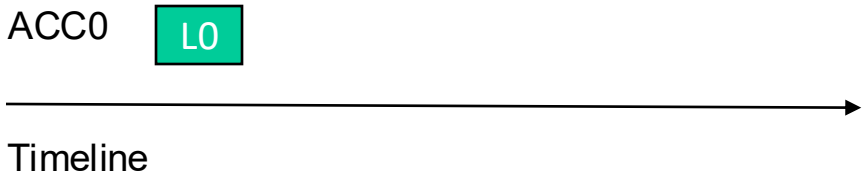
8 : 4 : 4 : 16



Tiled Architecture



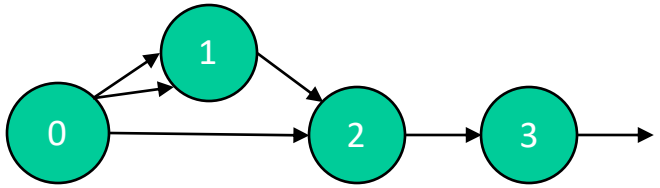
Scheduling Layer 1-2



Motivation: Sequential or Spatial Arch?

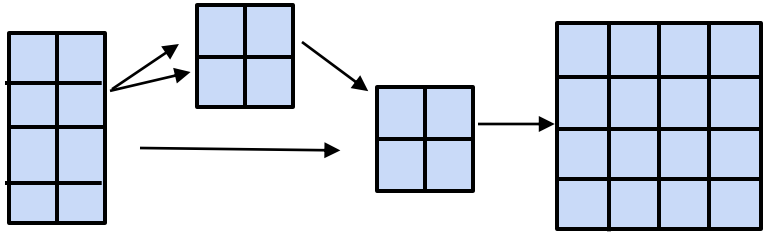
- Sequential-only Accelerator → Hardware Under-utilization
  - Mismatch between shape of the layers & size of the accelerator

Directed acyclic graph (DAG)

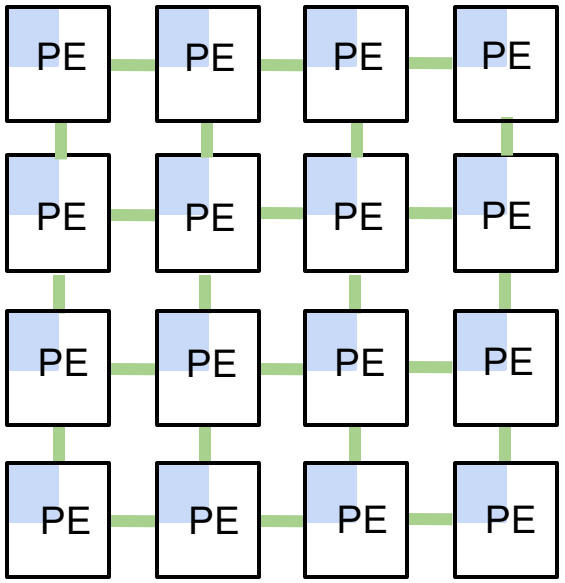


Corresponding Workloads

8 : 4 : 4 : 16



Tiled Architecture

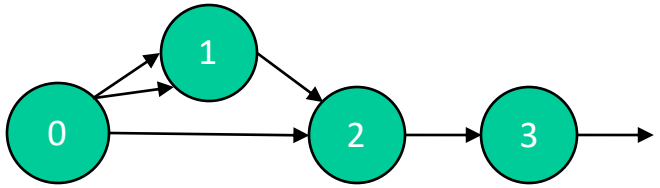


Scheduling Layer 1-2

Motivation: Sequential or Spatial Arch?

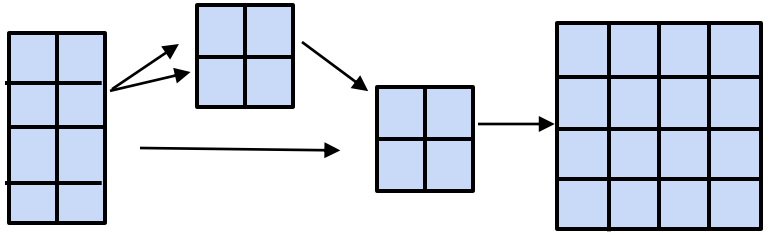
- Sequential-only Accelerator → Hardware Under-utilization
  - Mismatch between shape of the layers & size of the accelerator

Directed acyclic graph (DAG)

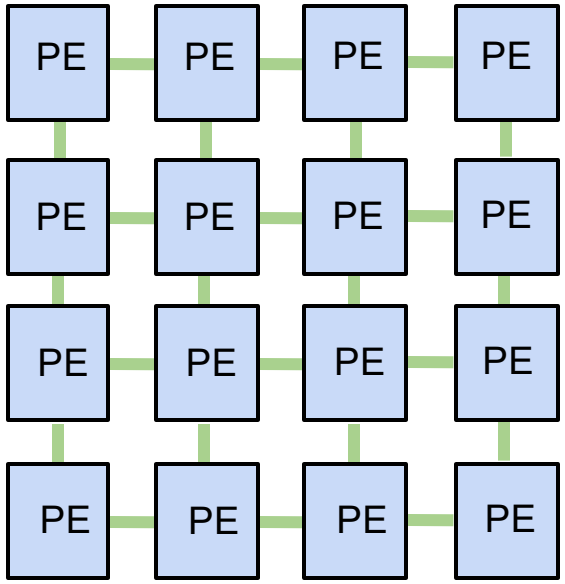


Corresponding Workloads

8 : 4 : 4 : 16



Vector Processors

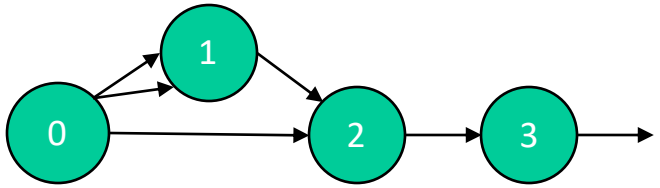


Scheduling Layer 3

Motivation: Sequential or Spatial Arch?

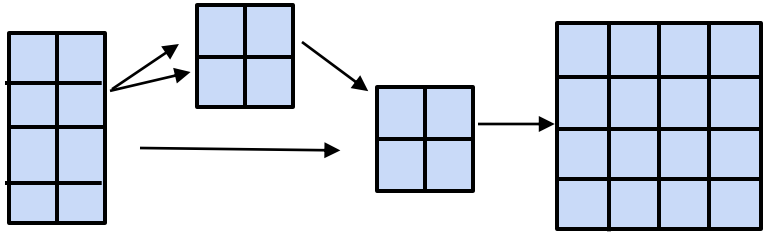
- Sequential-only Accelerator → Hardware Under-utilization
  - Mismatch between shape of the layers & size of the accelerator

Directed acyclic graph (DAG)

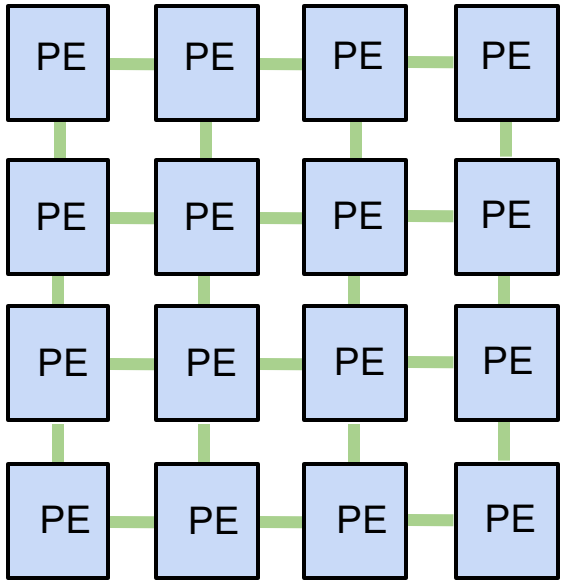


Corresponding Workloads

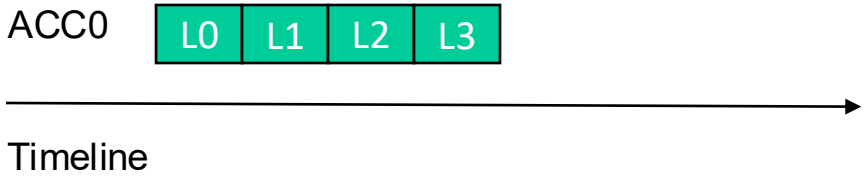
8 : 4 : 4 : 16



Vector Processors



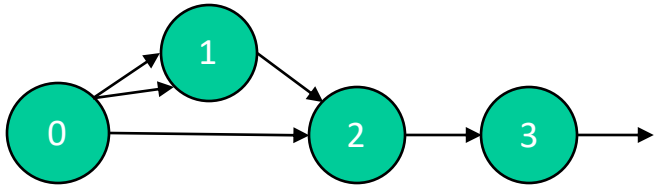
Scheduling Layer 3



Motivation: Sequential or Spatial Arch?

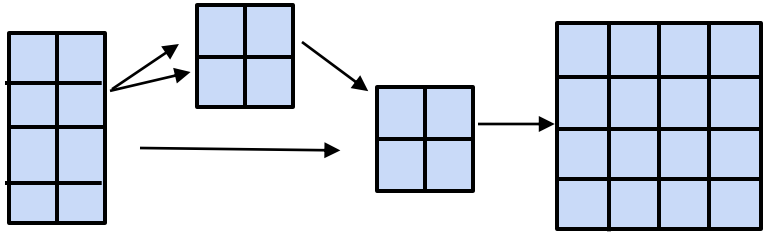
- Spatial-only Accelerator → Longer Latency (Multiple batches)

Directed acyclic graph (DAG)

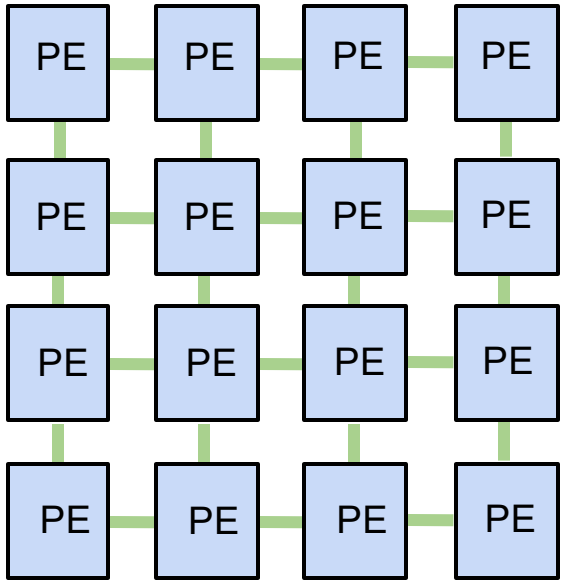


Corresponding Workloads

8 : 4 : 4 : 16



Vector Processors

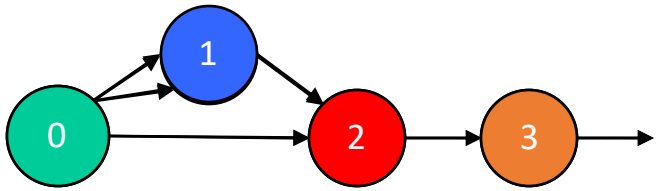


Scheduling

Motivation: Sequential or Spatial Arch?

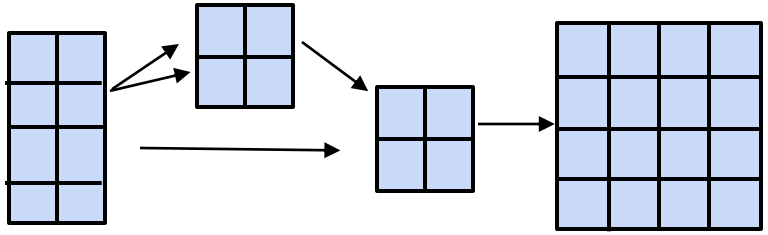
- Spatial-only Accelerator → Longer Latency (Multiple batches)

Directed acyclic graph (DAG)

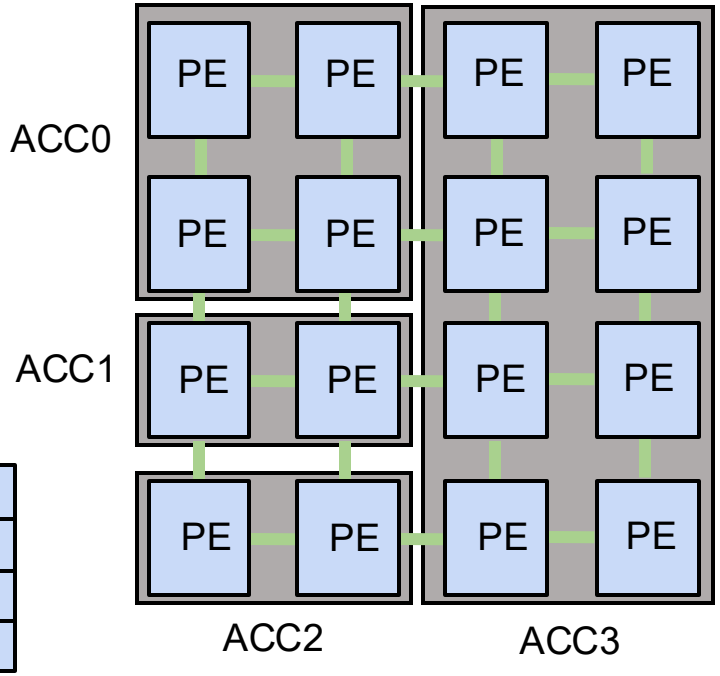


Corresponding Workloads

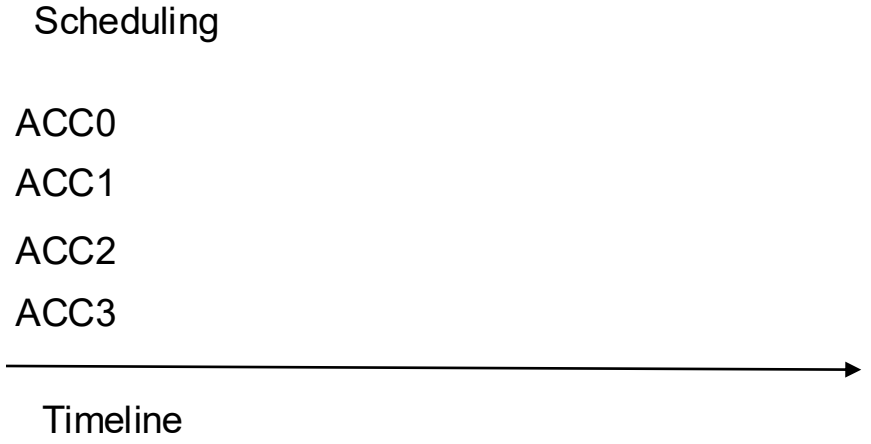
8 : 4 : 4 : 16



Vector Processors



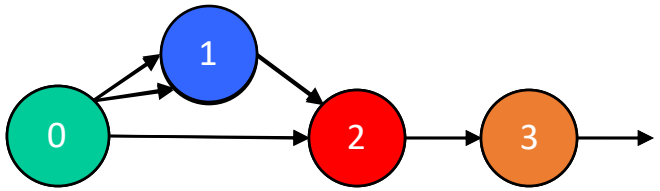
Scheduling



Motivation: Sequential or Spatial Arch?

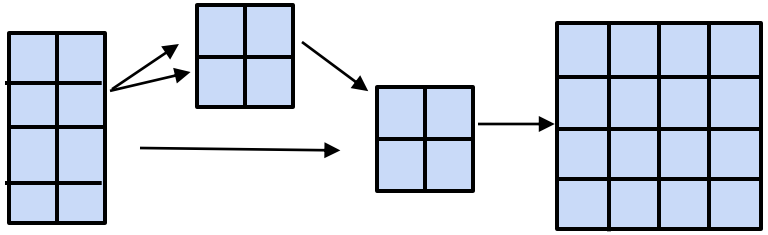
- Spatial-only Accelerator → Longer Latency (Multiple batches)

Directed acyclic graph (DAG)

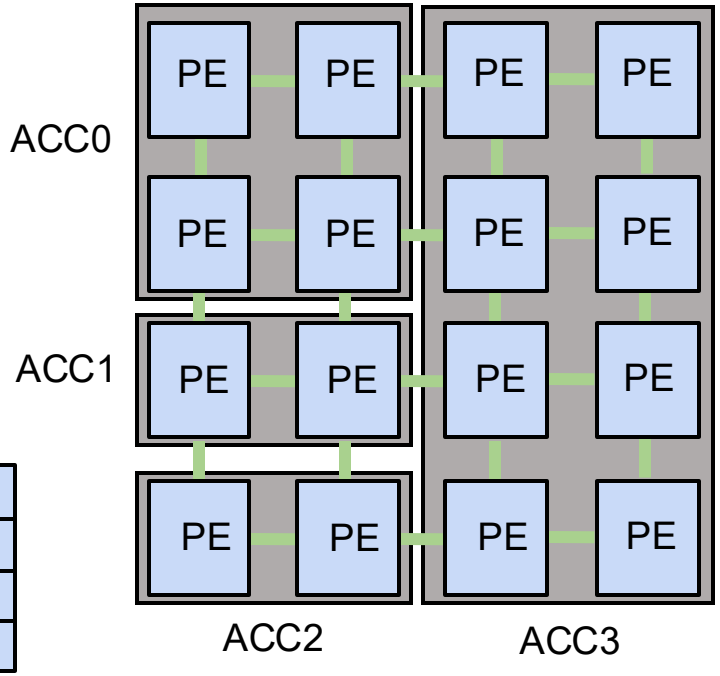


Corresponding Workloads

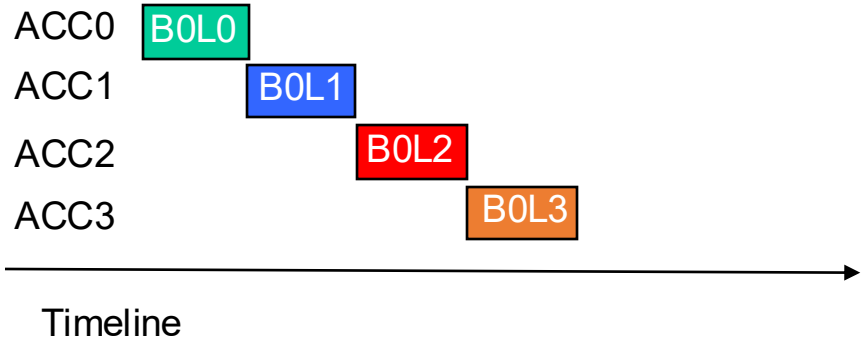
8 : 4 : 4 : 16



Vector Processors



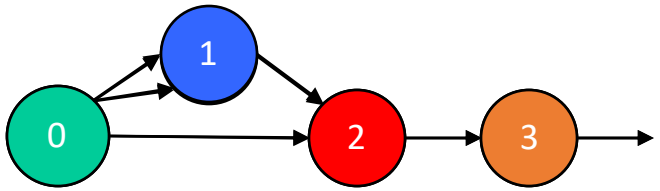
Scheduling Batch 0



Motivation: Sequential or Spatial Arch?

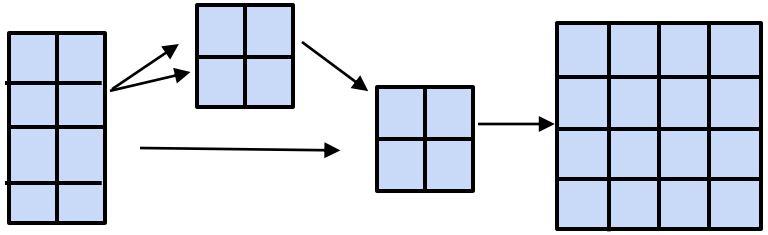
- Spatial-only Accelerator → Longer Latency (Multiple batches)

Directed acyclic graph (DAG)

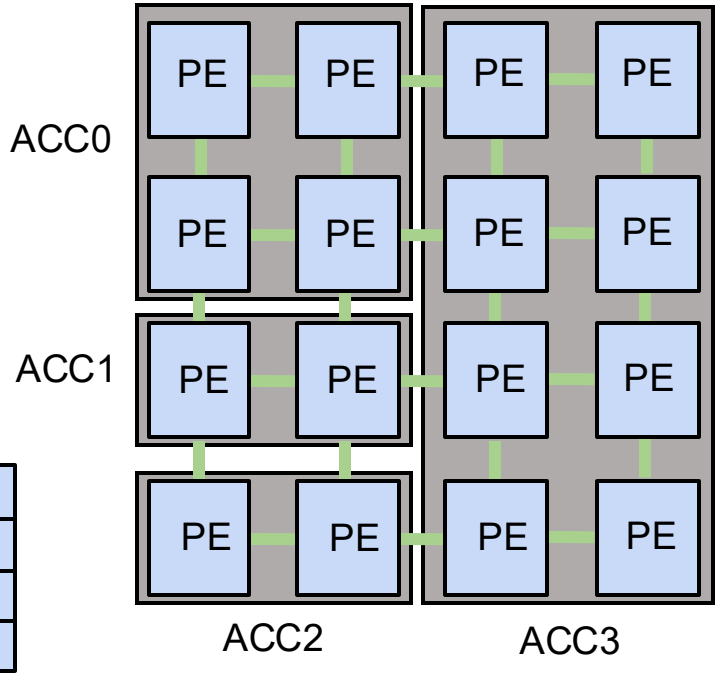


Corresponding Workloads

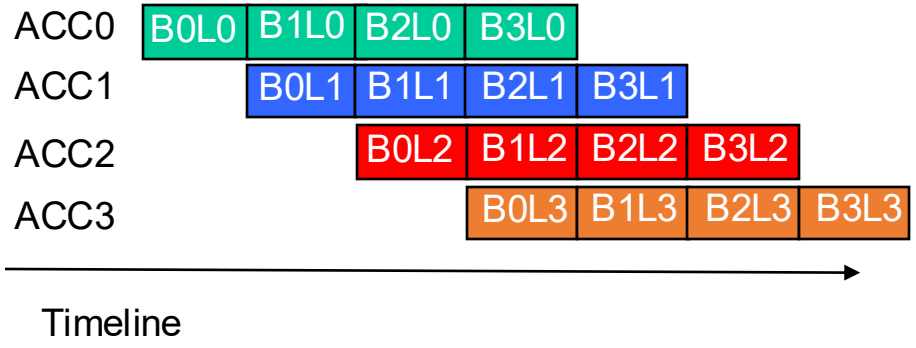
8 : 4 : 4 : 16



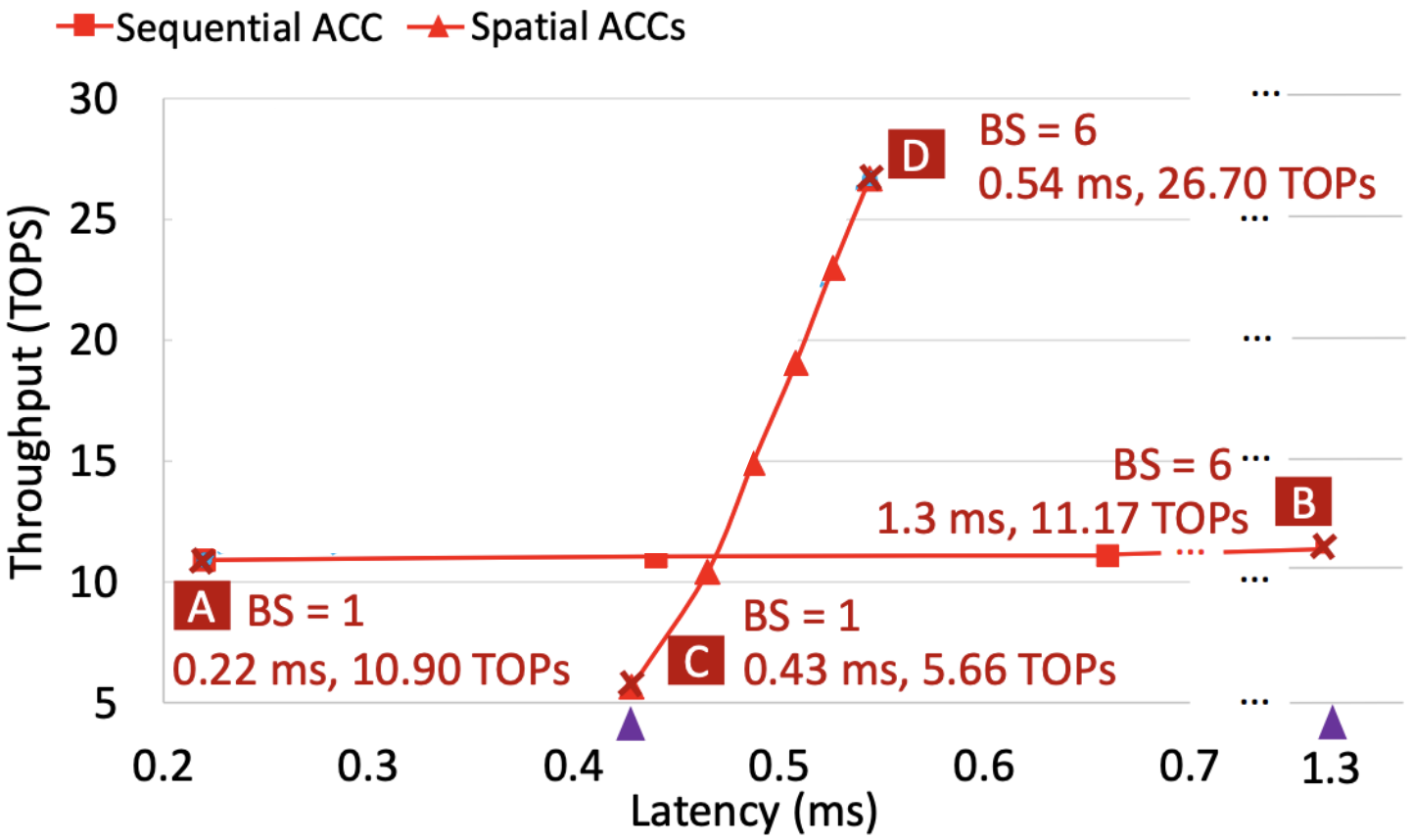
Vector Processors

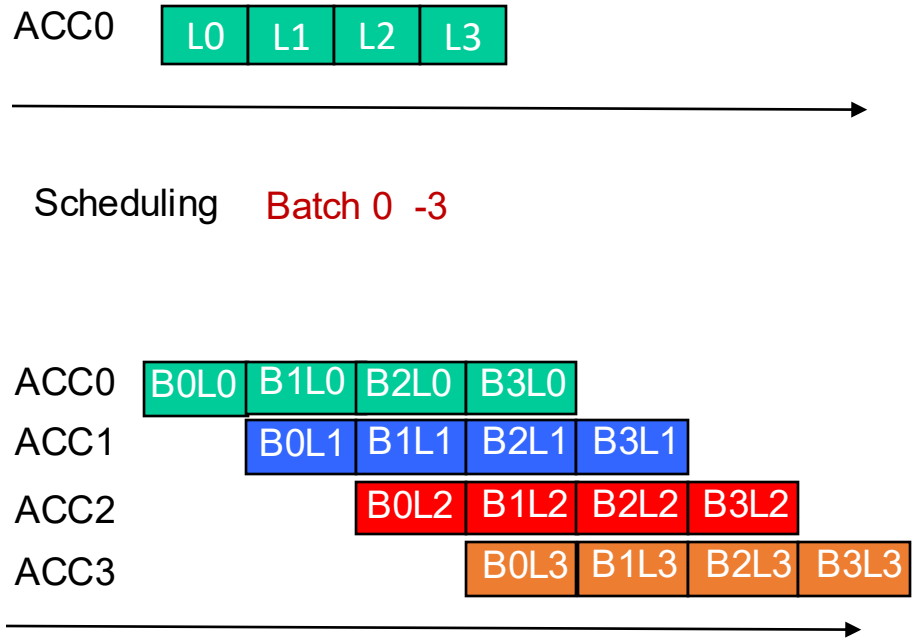


Scheduling Batch 0 -3

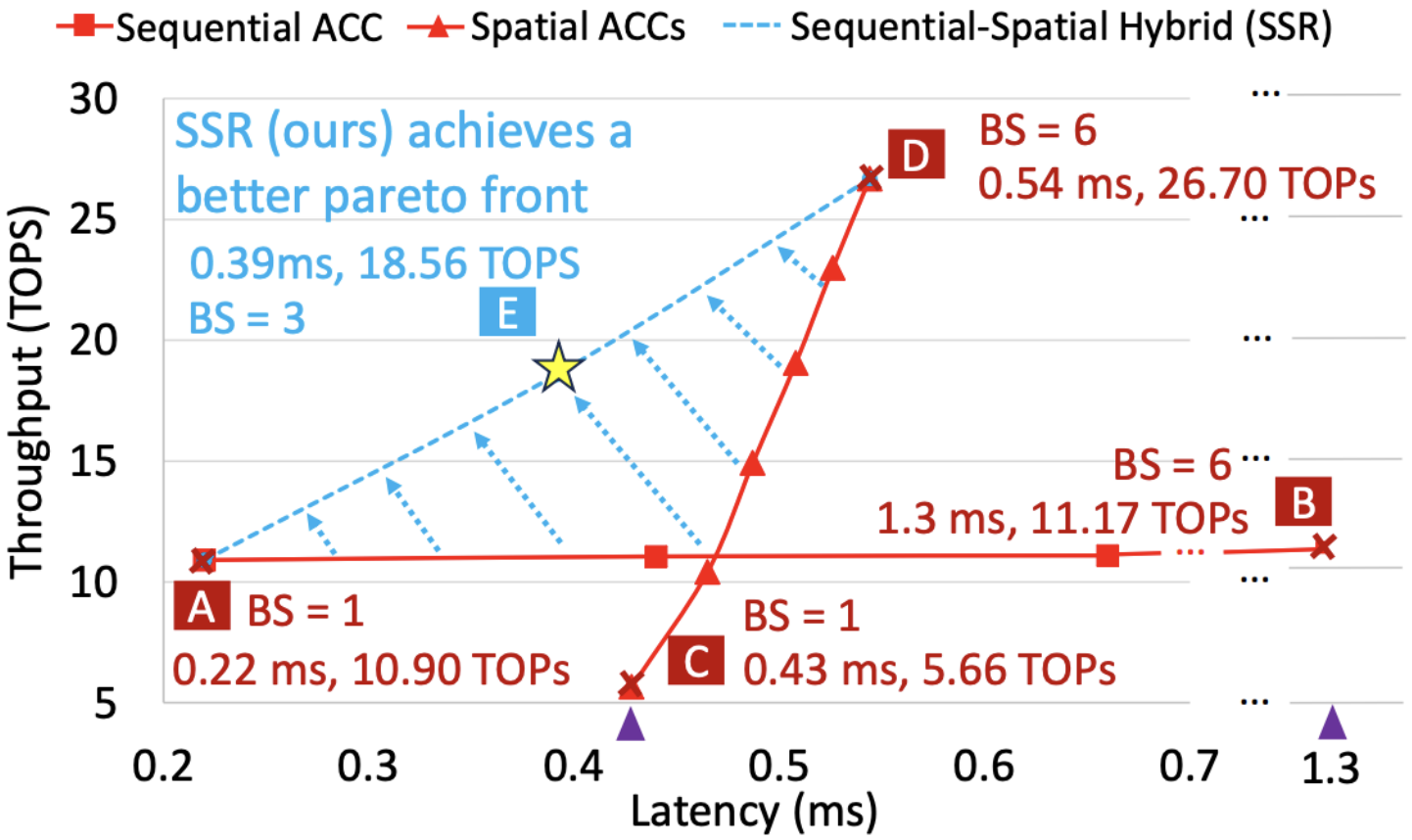


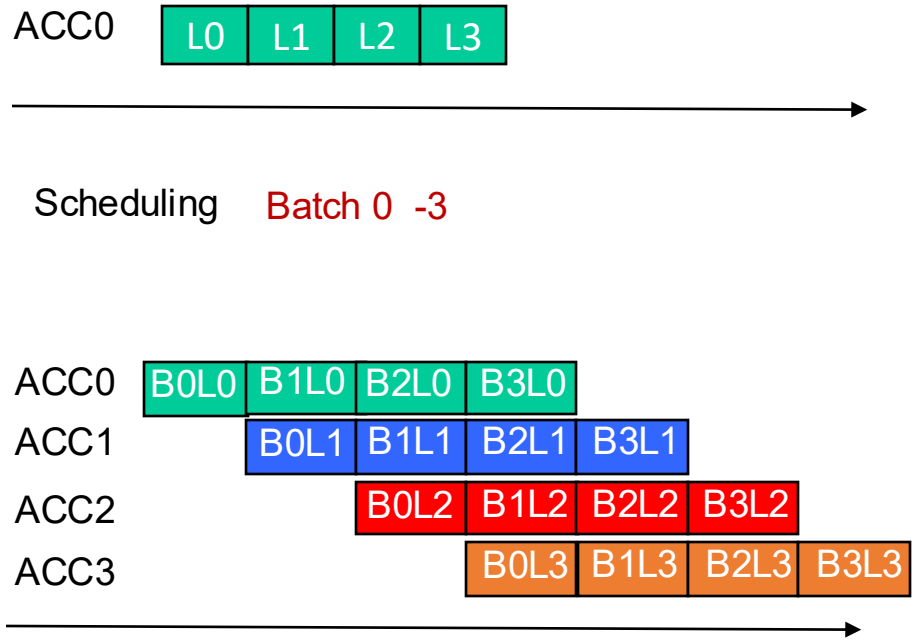
Motivation: Sequential or Spatial Arch?





Motivation: Sequential or Spatial Arch?

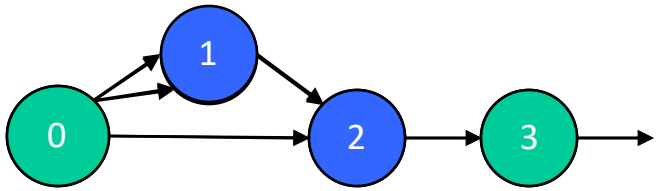




Solution: Sequential + Spatial Hybrid

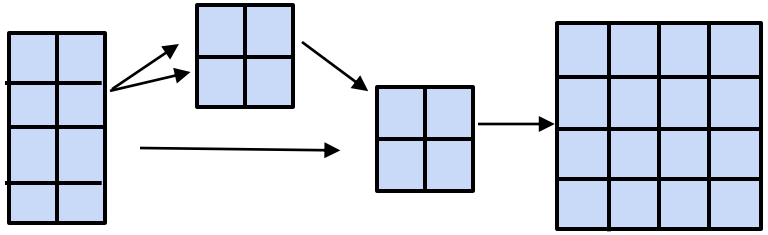
- SSR-Hybrid → Pareto front in Latency throughput tradeoff

Directed acyclic graph (DAG)

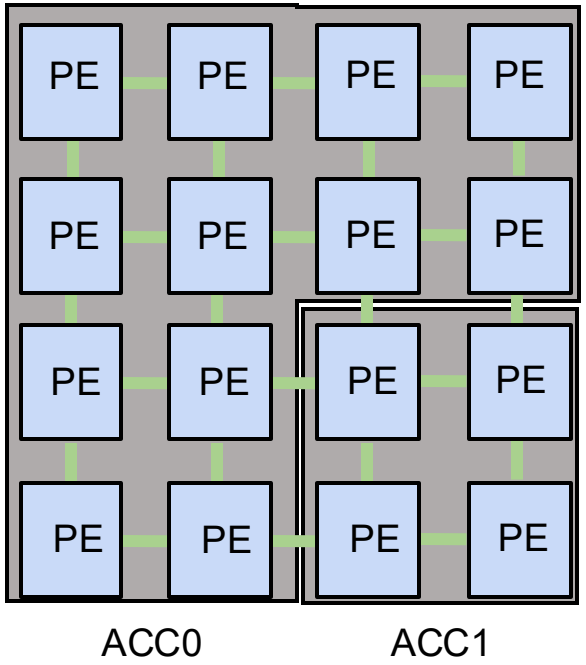


Corresponding Workloads

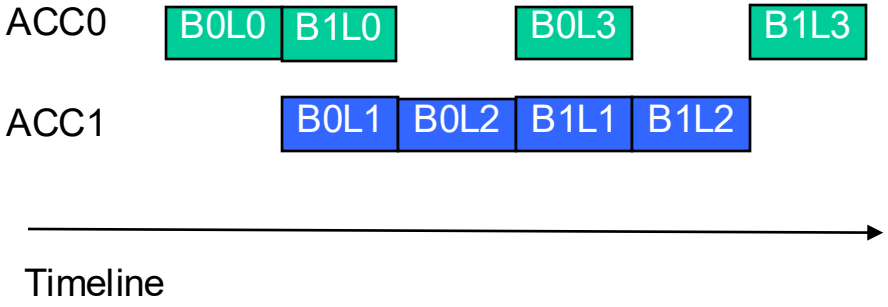
8 : 4 : 4 : 16



Vector Processors

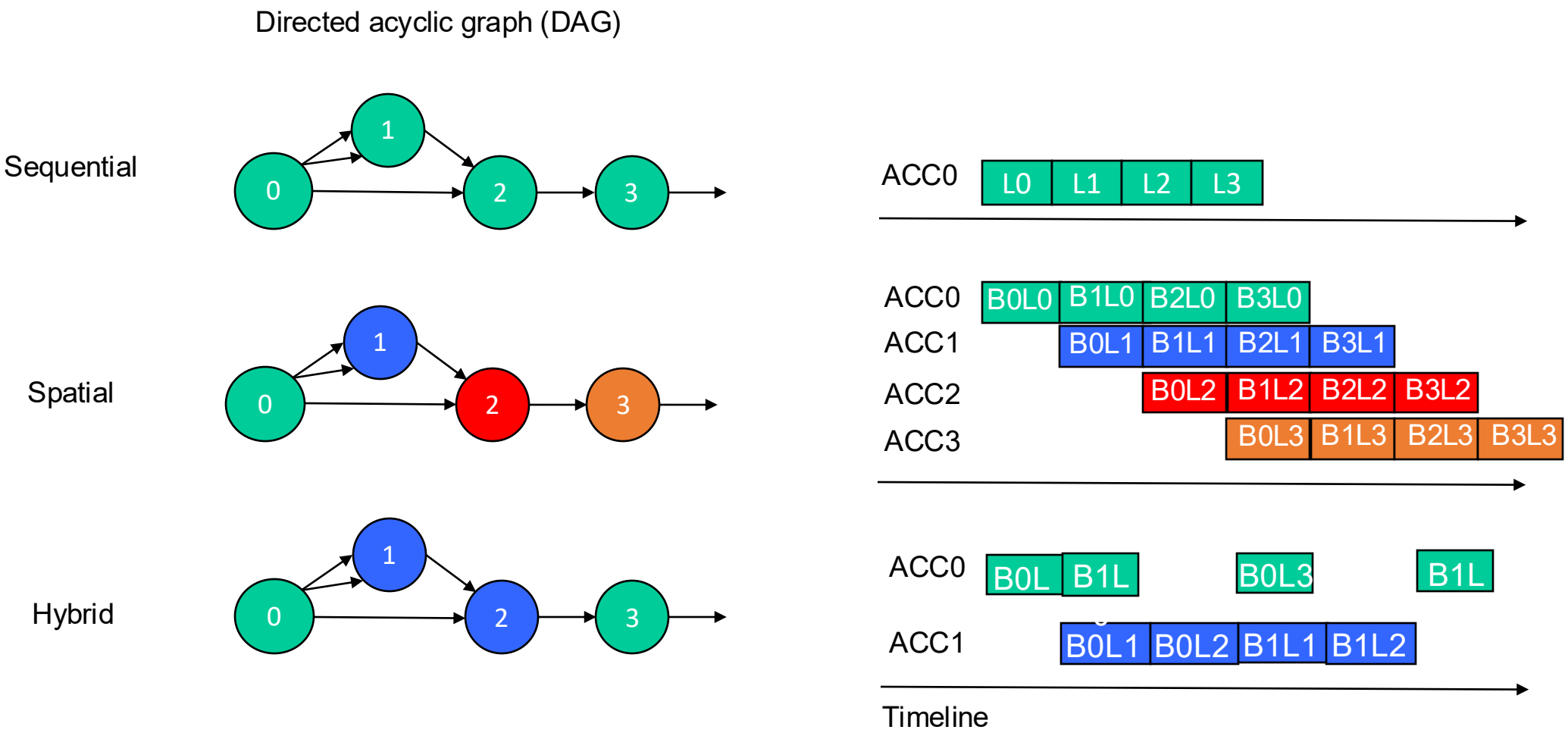


Scheduling Batch 0-1



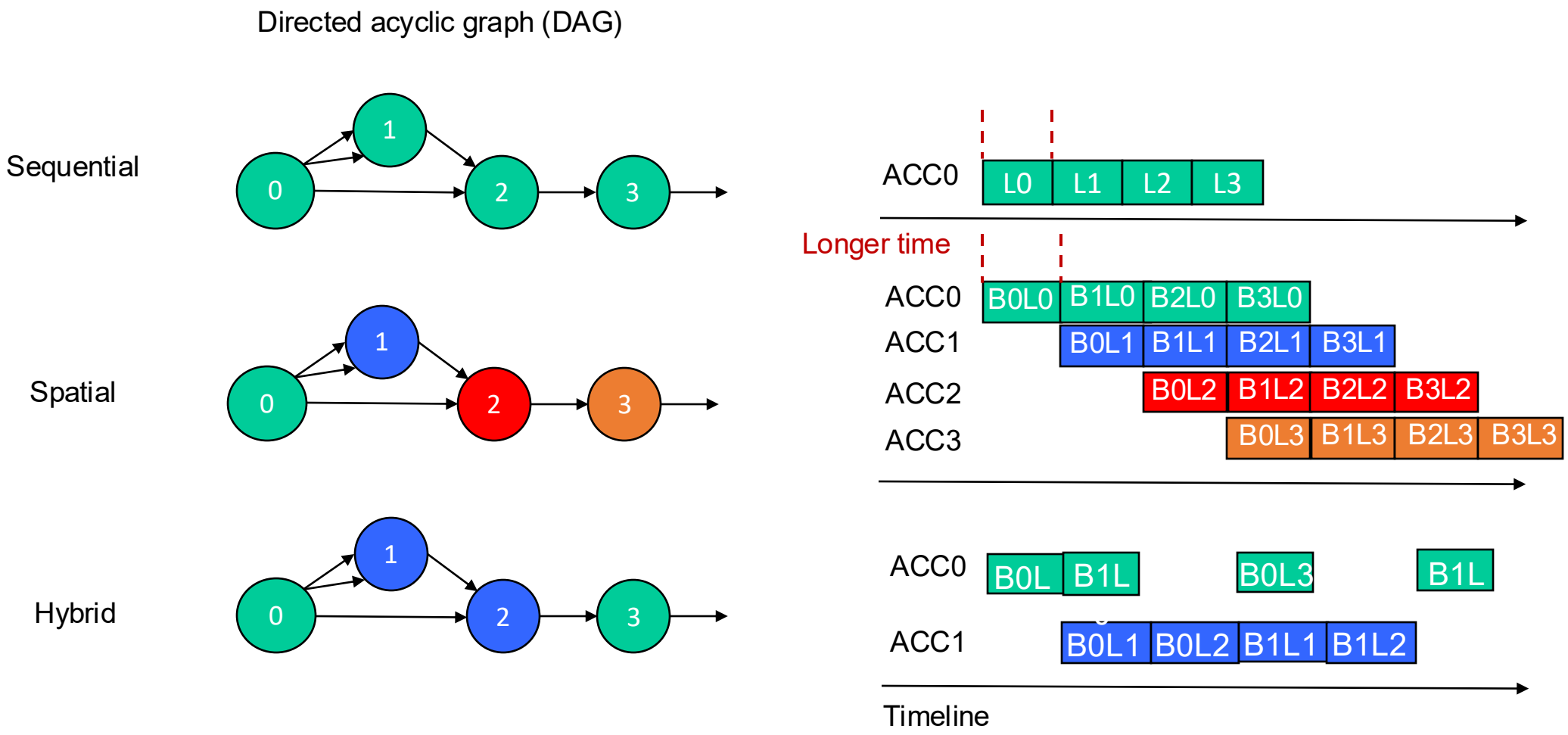
SSR-Hybrid Solution

- SSR-Hybrid → Pareto front in Latency throughput tradeoff



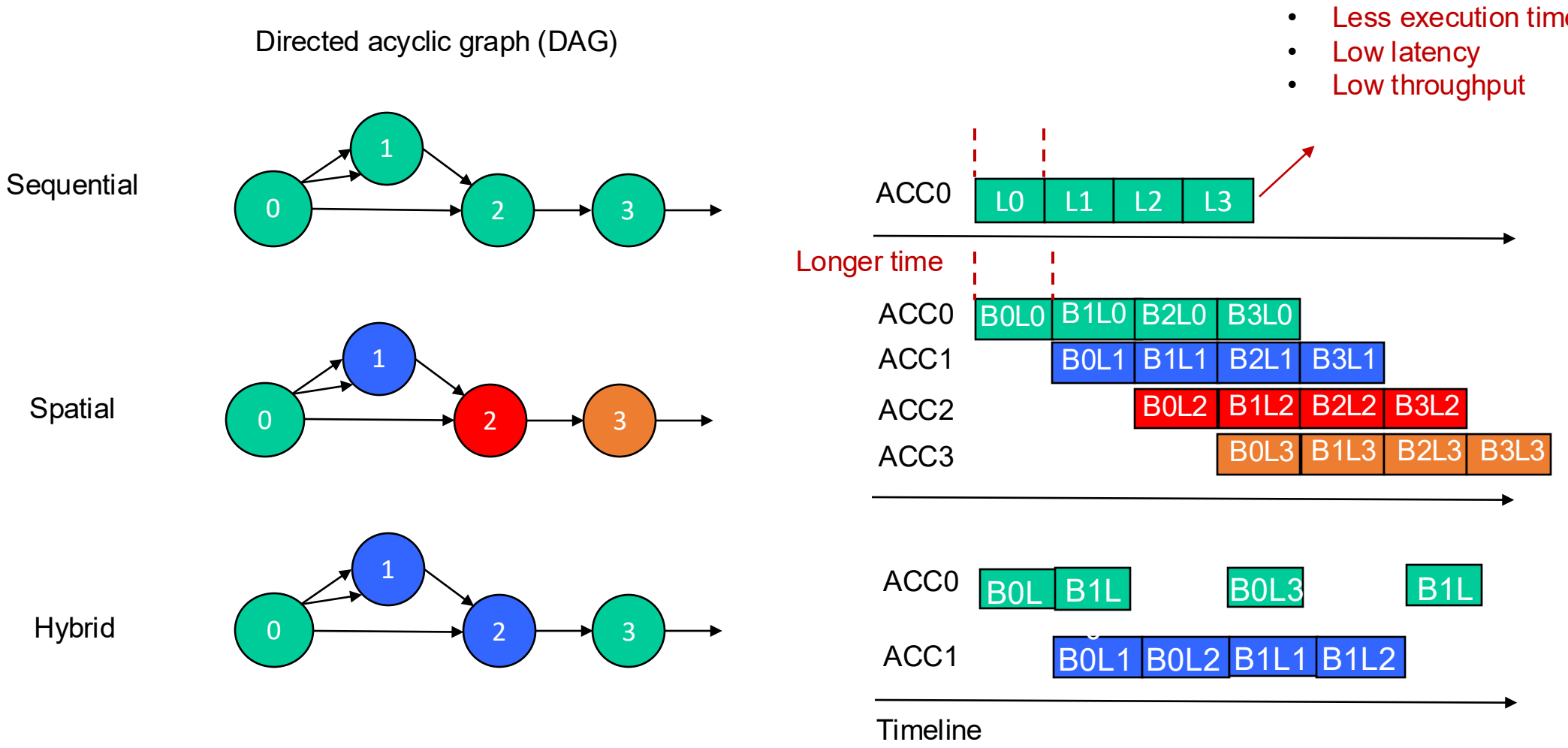
SSR-Hybrid Solution

- SSR-Hybrid → Pareto front in Latency throughput tradeoff



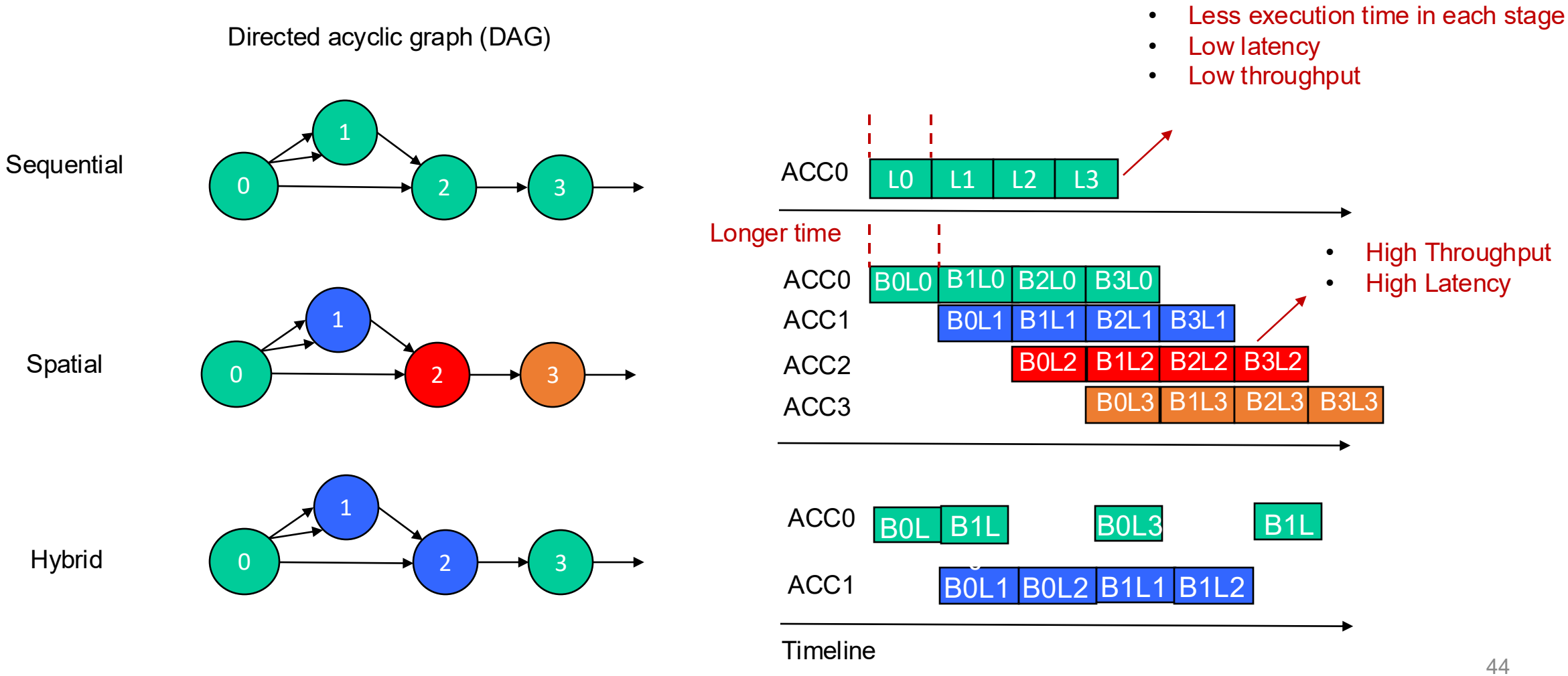
SSR-Hybrid Solution

- SSR-Hybrid → Pareto front in Latency throughput tradeoff



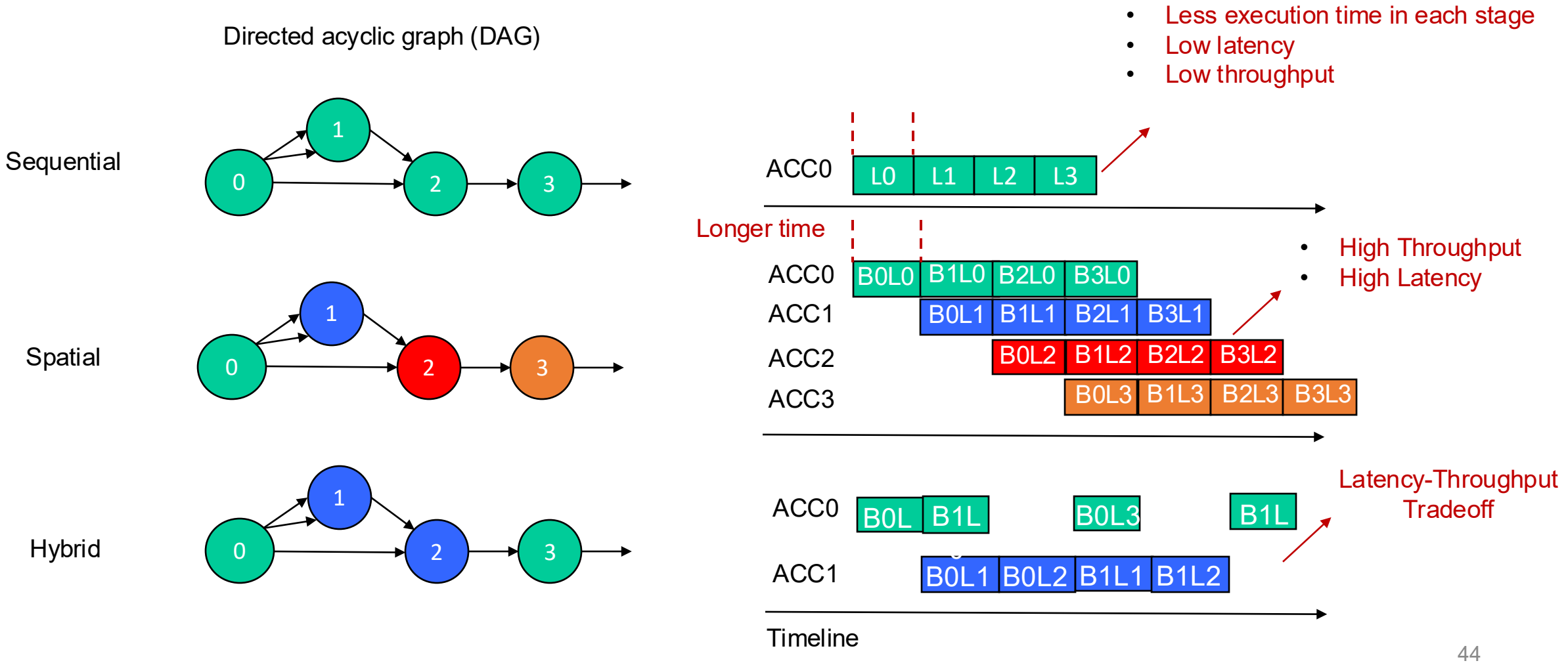
SSR-Hybrid Solution

- SSR-Hybrid → Pareto front in Latency throughput tradeoff



SSR-Hybrid Solution

- SSR-Hybrid → Pareto front in Latency throughput tradeoff



SSR Architecture, Designs & Software

Challenges

Solutions

SSR Architecture, Designs & Software

Challenges

- Spatial-Only → High Latency
  Sequential-Only → Low Throughput

Solutions

- Feature: Multi Spatial Accelerators;
  Apply hybrid architecture to explore latency throughput tradeoff

SSR Architecture, Designs & Software

Challenges

- Spatial-Only → High Latency
  Sequential-Only → Low Throughput

- Non-linear & elementwise kernels takes Non-negligible time

Solutions

- Feature: Multi Spatial Accelerators; Apply hybrid architecture to explore latency throughput tradeoff

- Feature: Enable Inter-Acc aware on-chip data forwarding and fine-grained pipeline

SSR Architecture, Designs & Software

Challenges

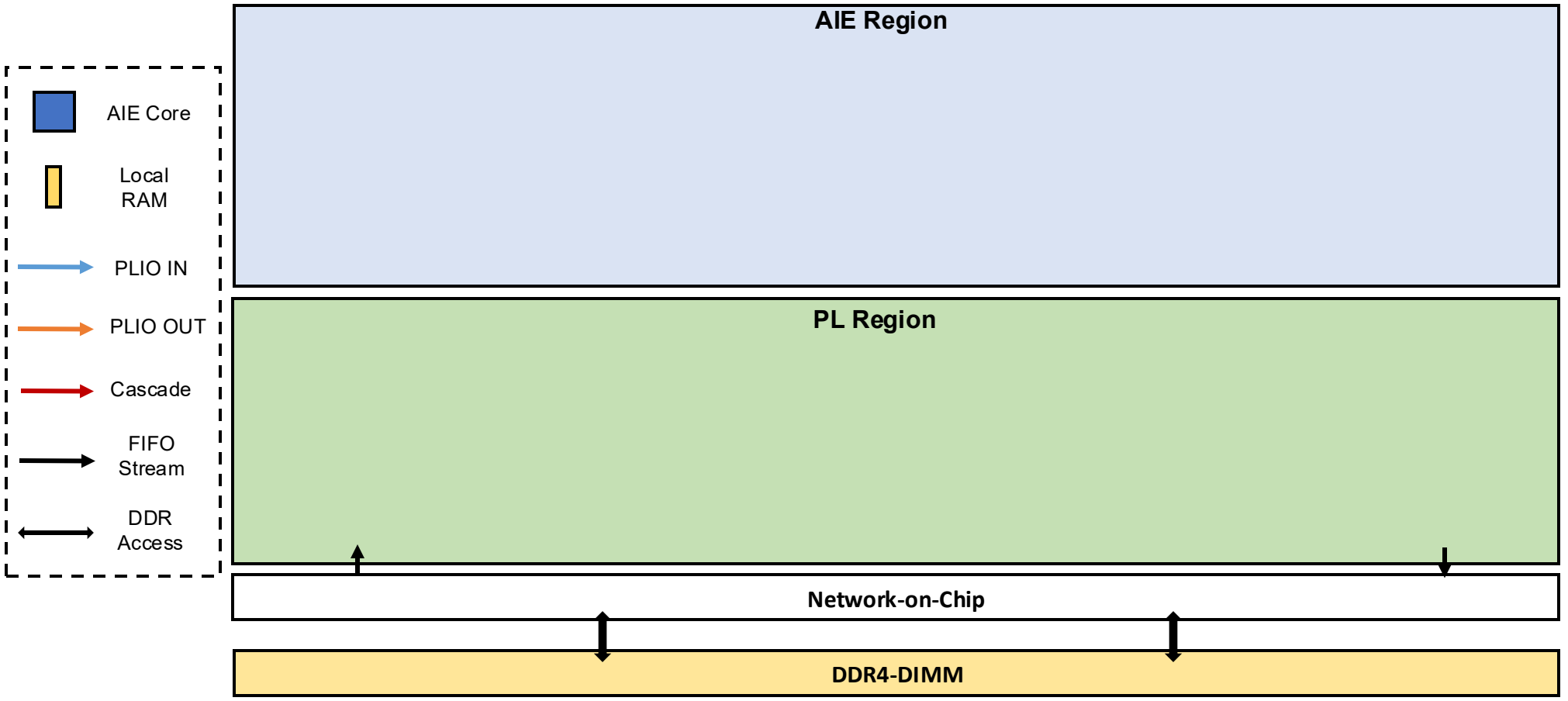
- Spatial-Only → High Latency
  Sequential-Only → Low Throughput
- Non-linear & elementwise kernels takes Non-negligible time
- The large design space & heterogeneity of ACAP motivates an automation tool

Solutions

- Feature: Multi Spatial Accelerators; Apply hybrid architecture to explore latency throughput tradeoff
- Feature: Enable Inter-Acc aware on-chip data forwarding and fine-grained pipeline
- SSR two-phase DSE tool & automatic workflow for implementation on ACAP
  https://github.com/arc-research-lab/SSR

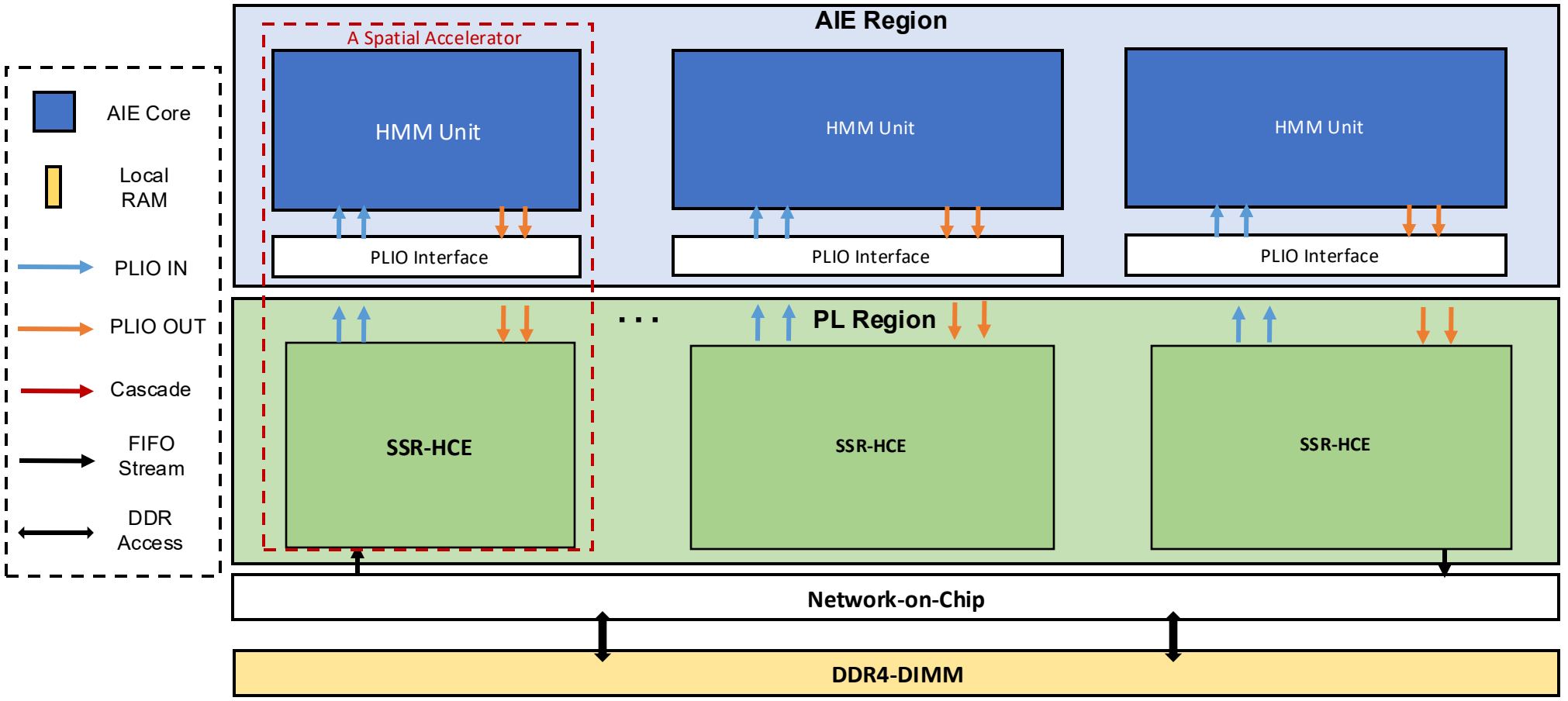
SSR Design Methodology

- SSR Architecture



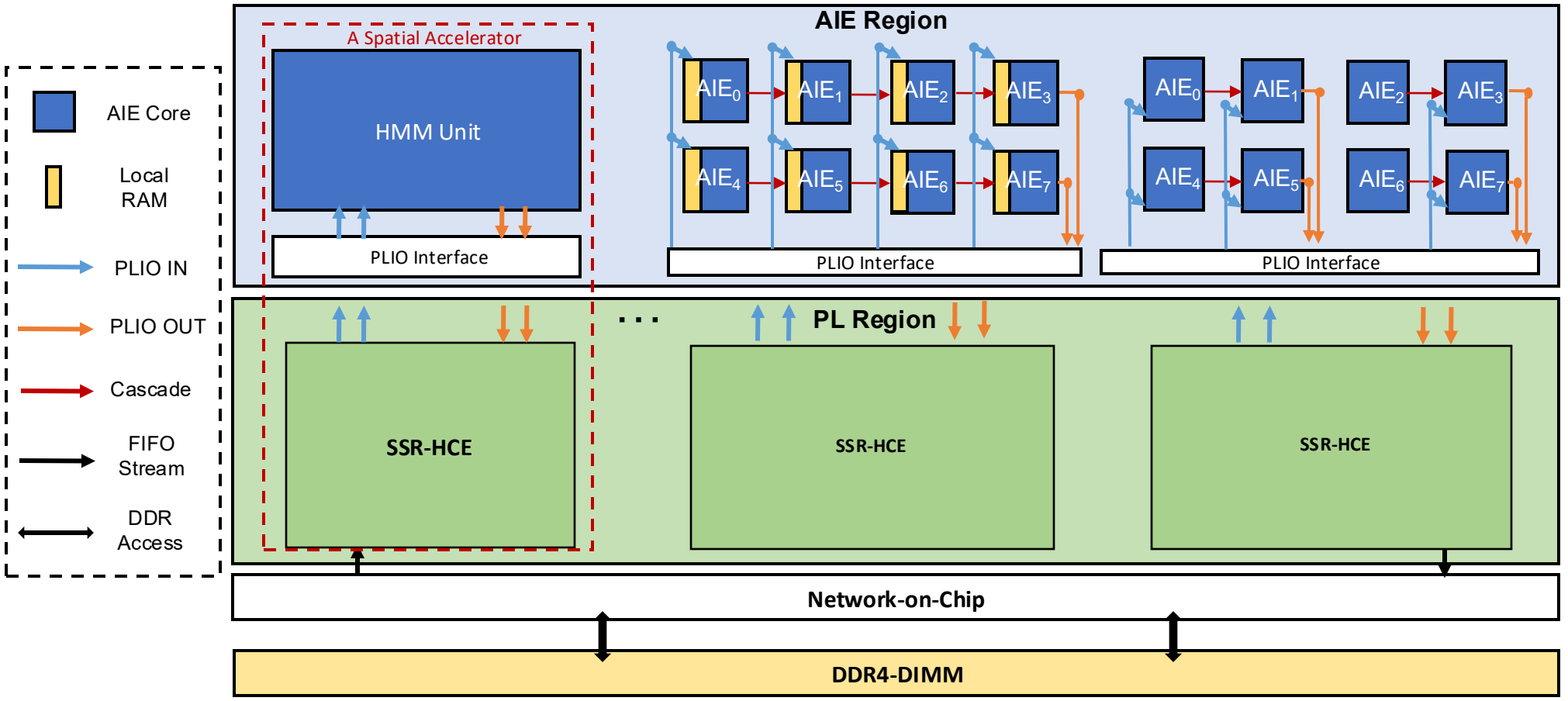
SSR Design Methodology

- SSR Architecture



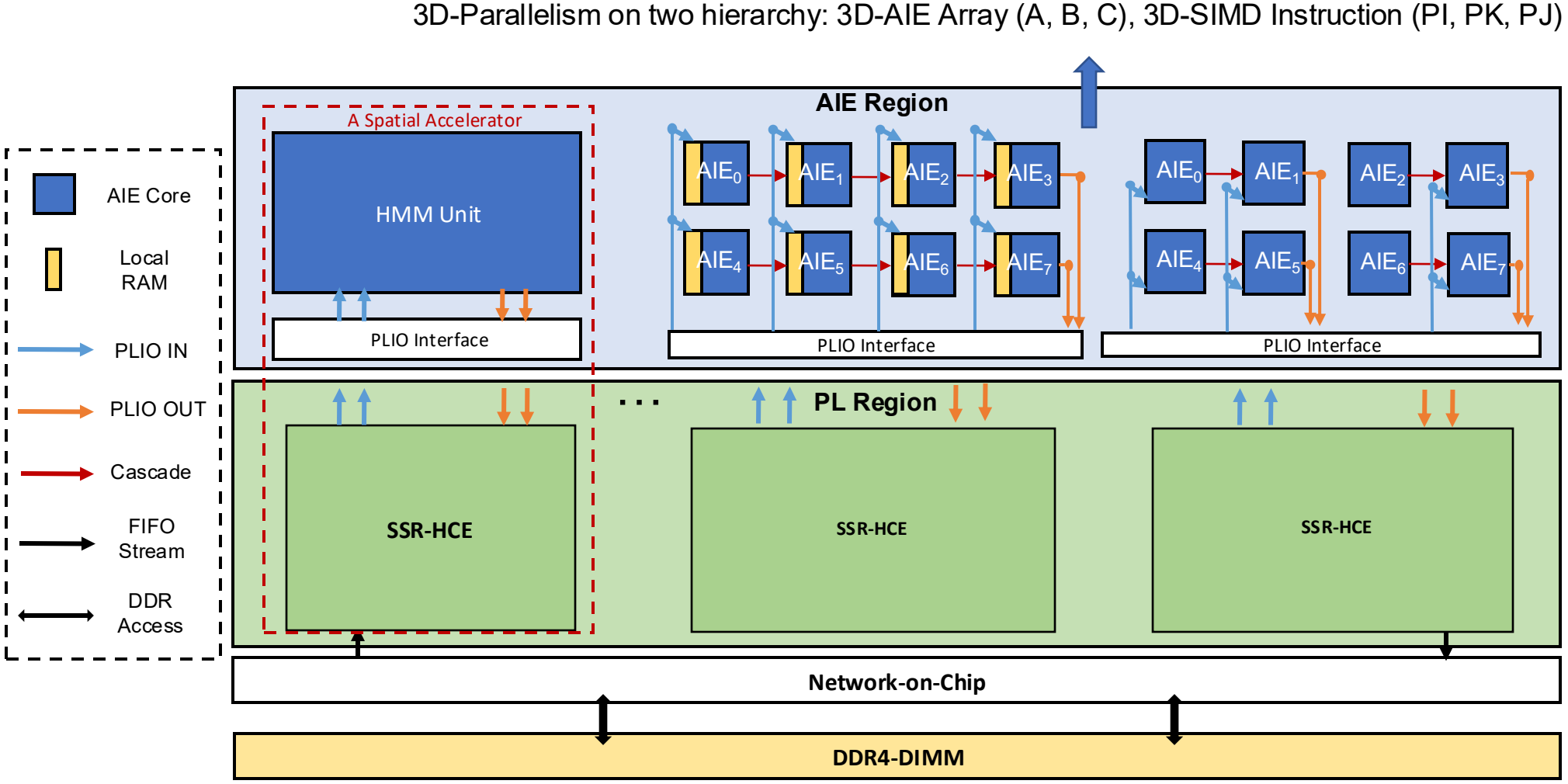
SSR Design Methodology

- SSR Architecture



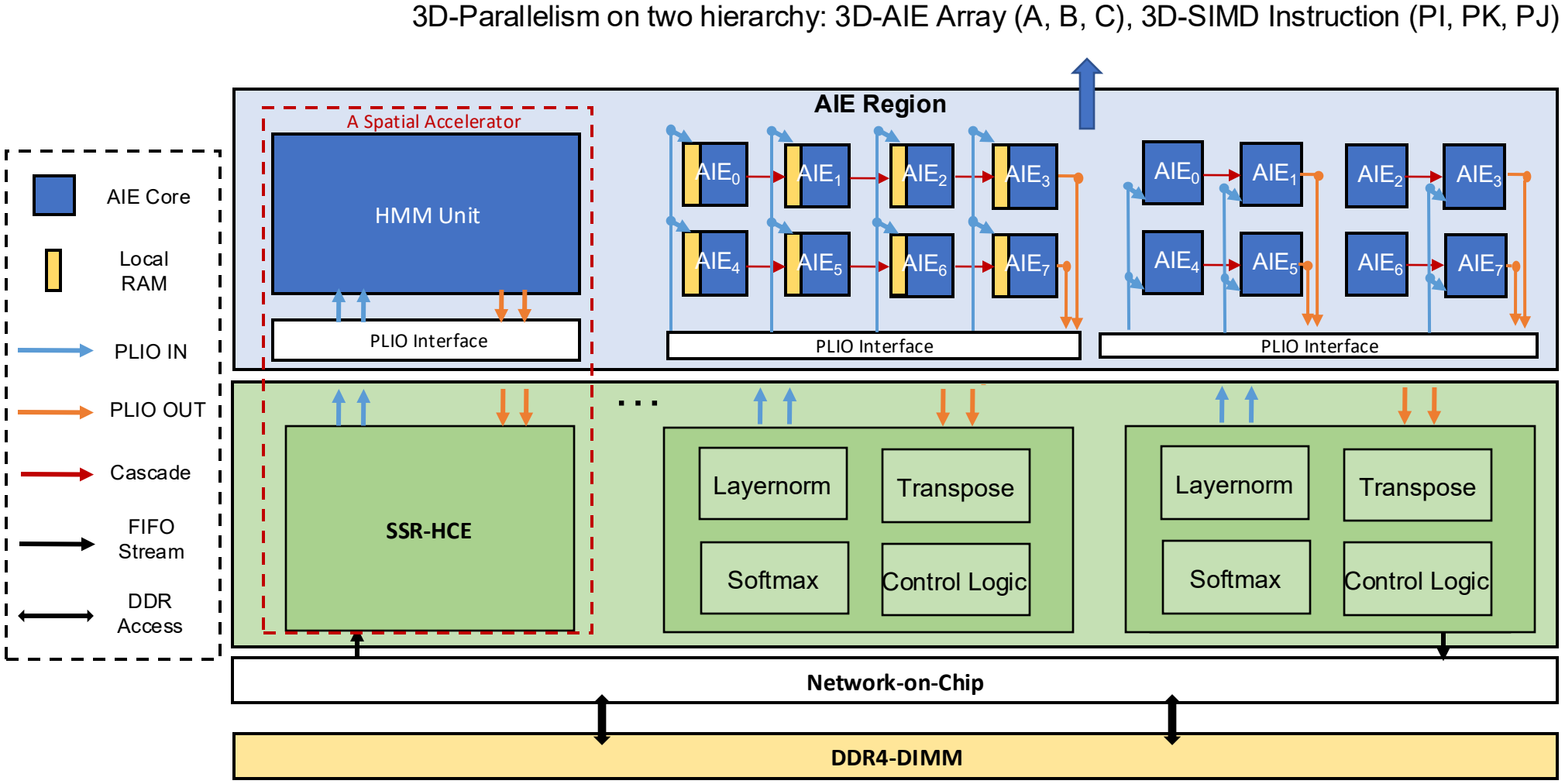
SSR Design Methodology

- SSR Architecture



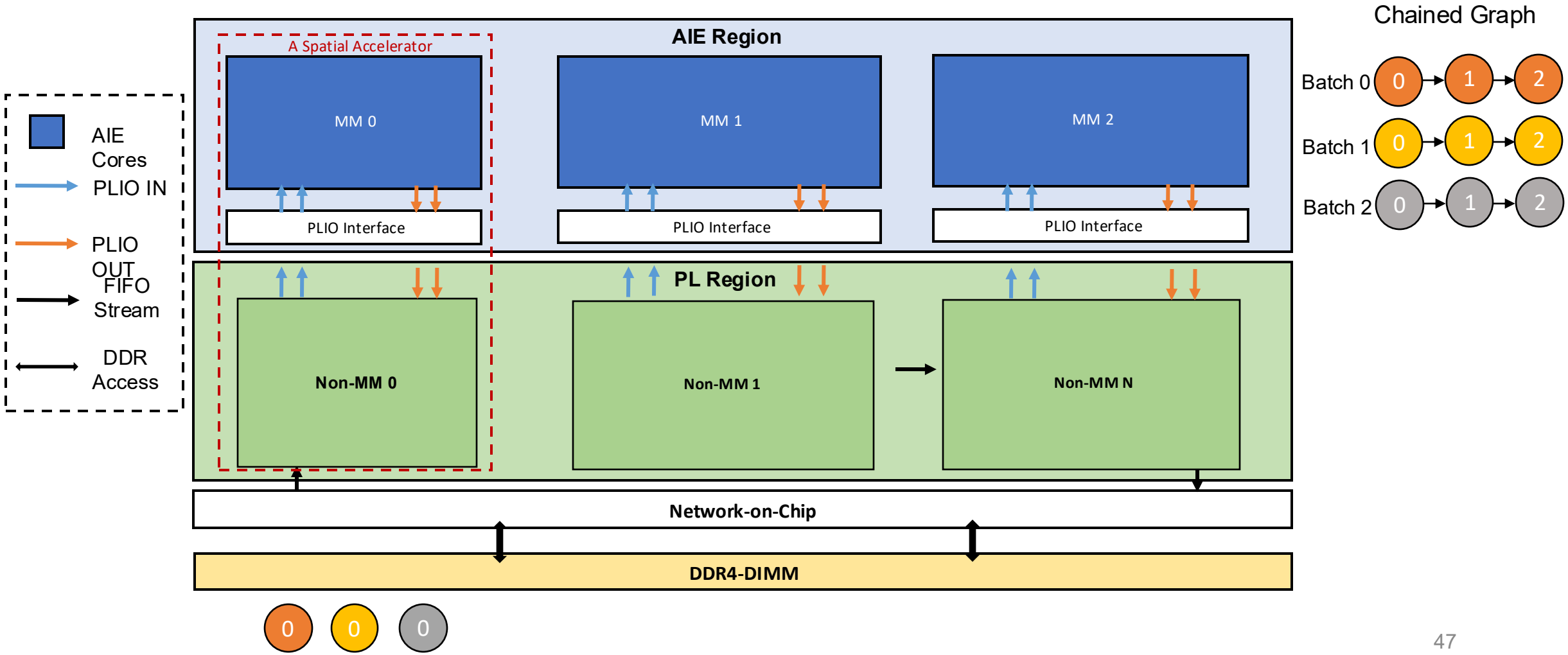
SSR Design Methodology

- SSR Architecture



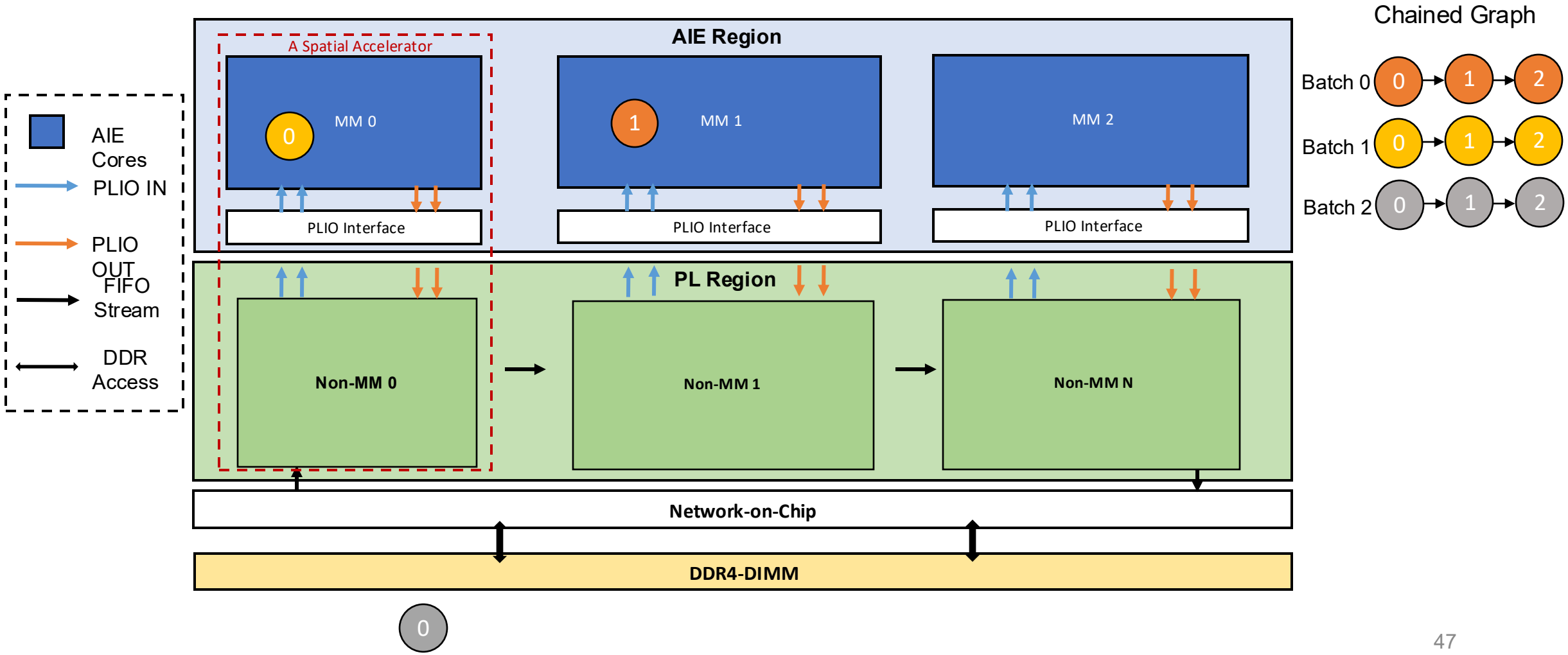
SSR Design Methodology

- SSR Programming Model & Dataflow



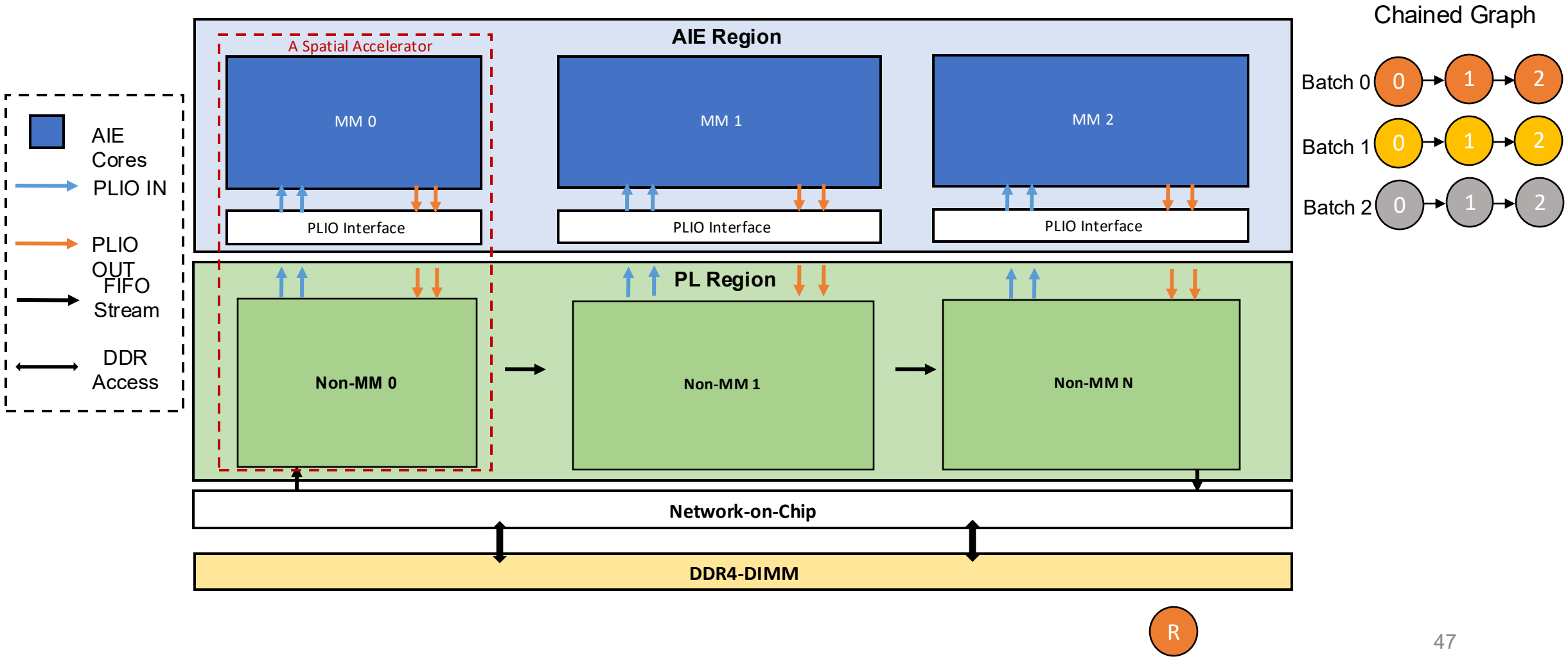
SSR Design Methodology

- SSR Programming Model & Dataflow



SSR Design Methodology

- SSR Programming Model & Dataflow

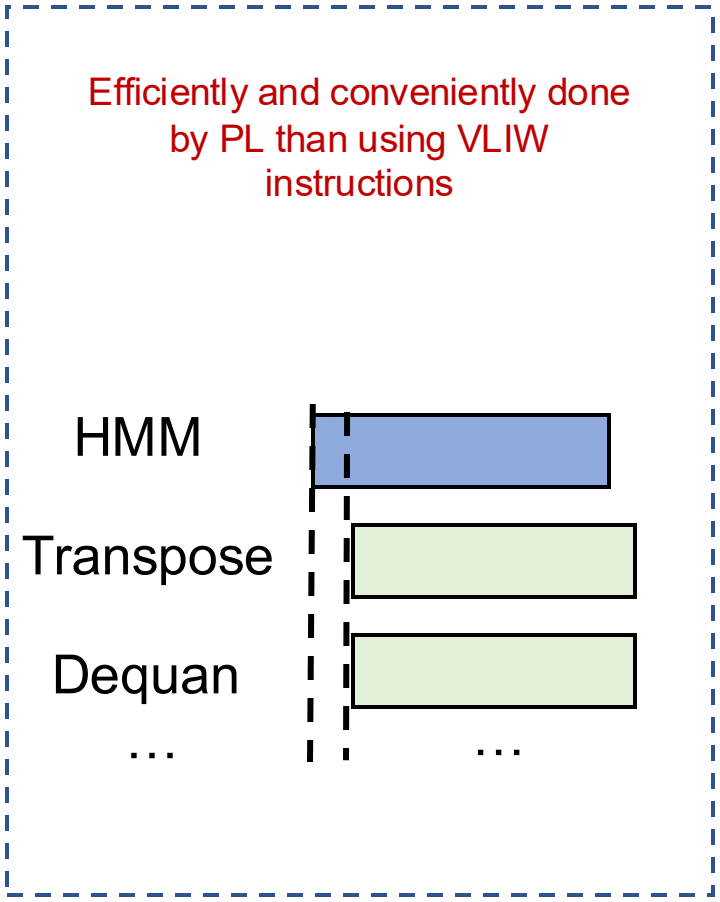


SSR Design Methodology

- Fine-grained pipeline for element-wise and non-linear kernels

SSR Design Methodology

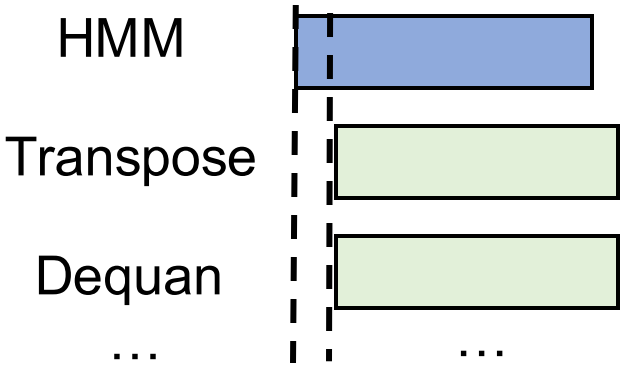
- Fine-grained pipeline for element-wise and non-linear kernels
  - Element-wise Kernels

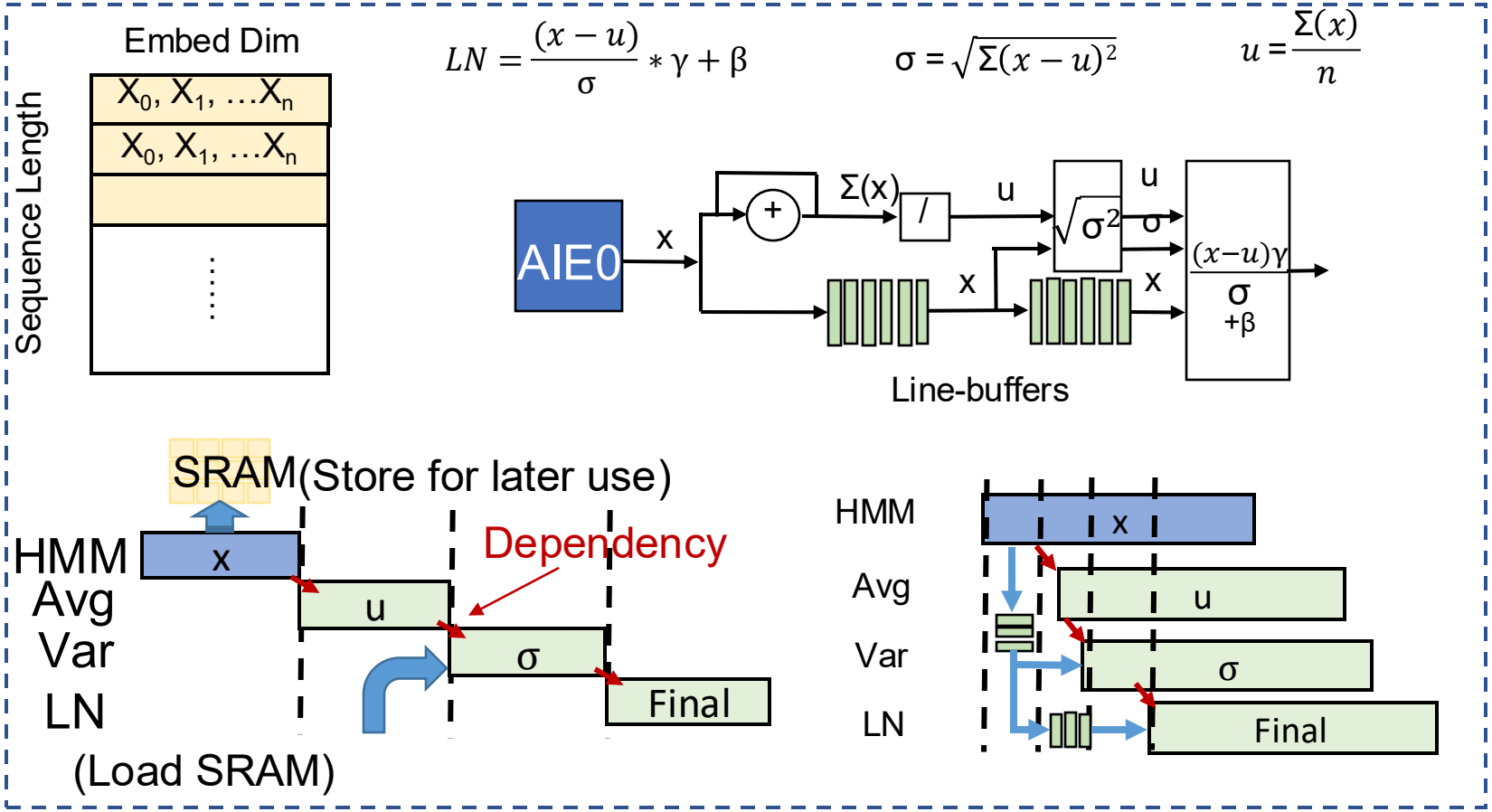


SSR Design Methodology

- Fine-grained pipeline for element-wise and non-linear kernels
  - Element-wise Kernels
  - Non-linear Kernels

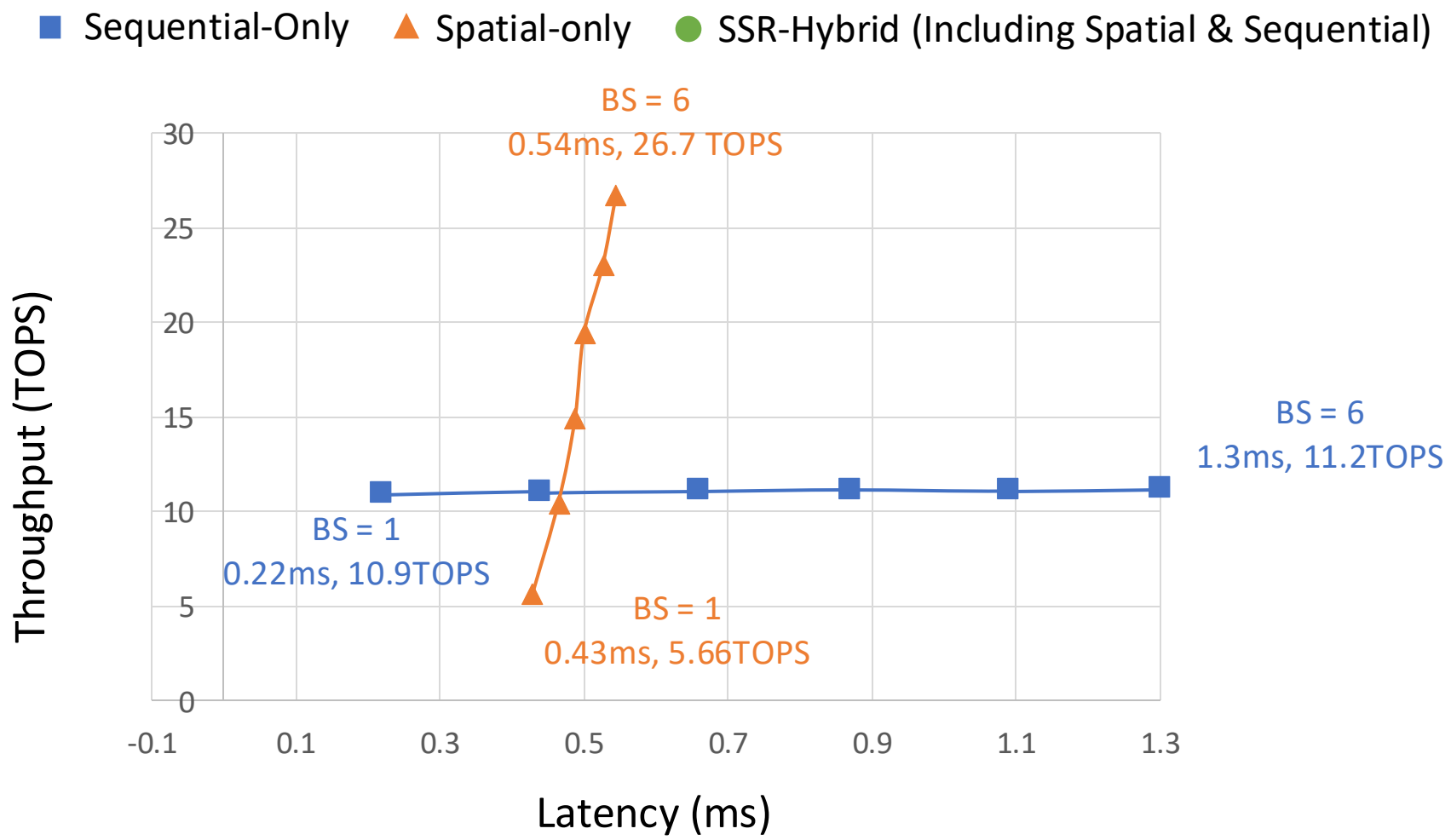
Efficiently and conveniently done by PL than using VLIW instructions





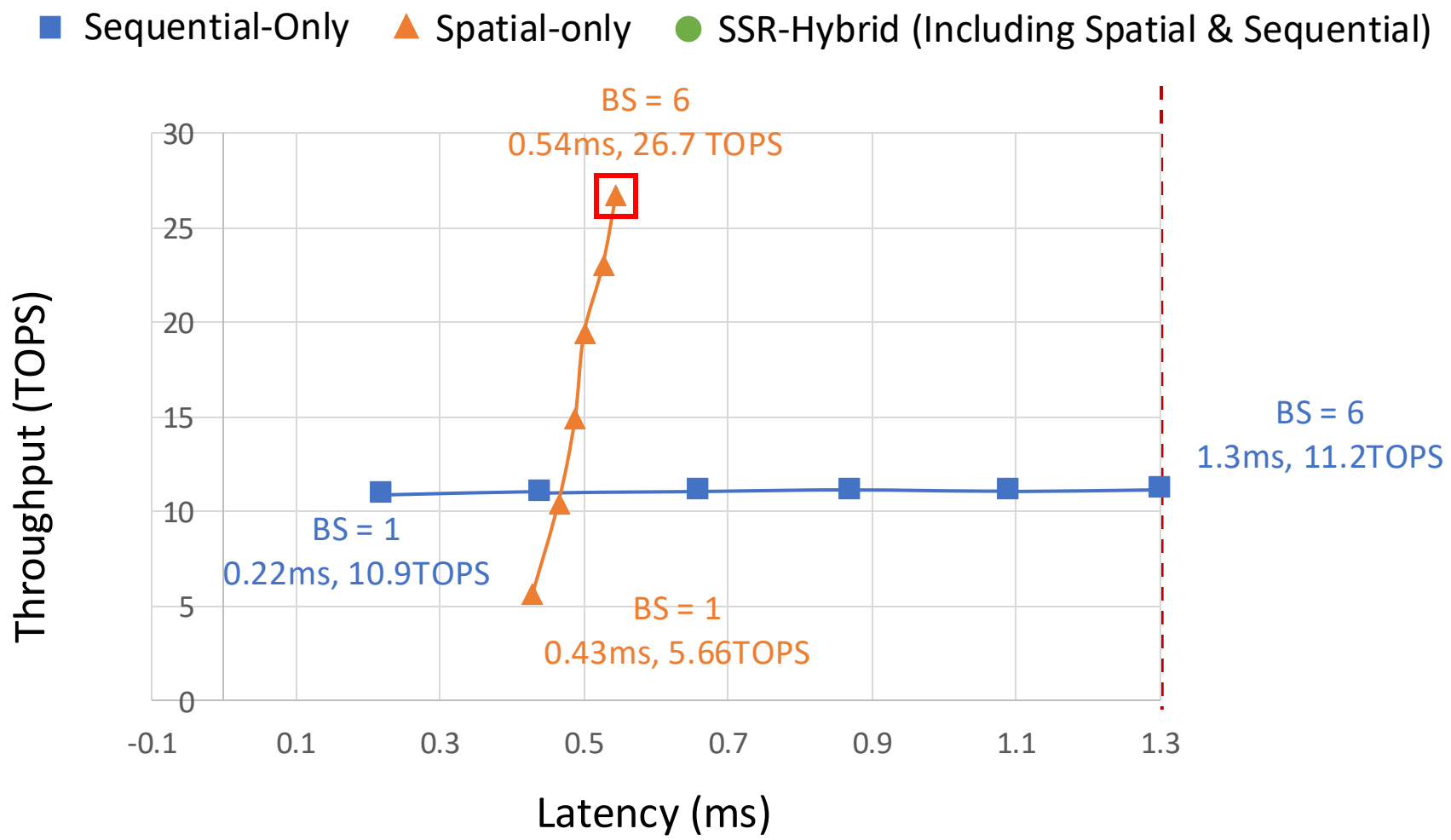
Experiment Results

- Latency & Throughput Tradeoff Analysis of DeiT-T across Different Strategies



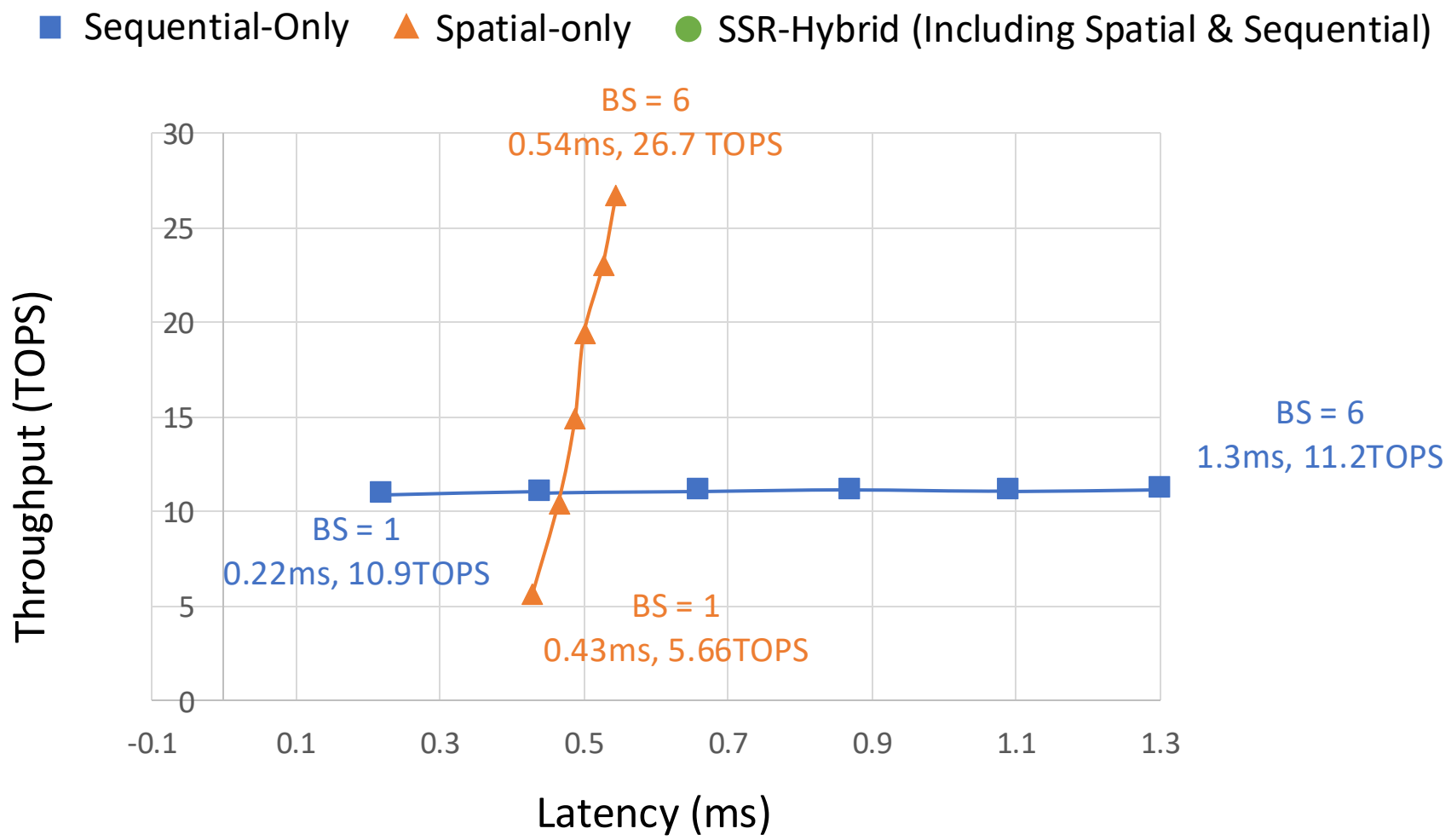
Experiment Results

- Latency & Throughput Tradeoff Analysis of DeiT-T across Different Strategies



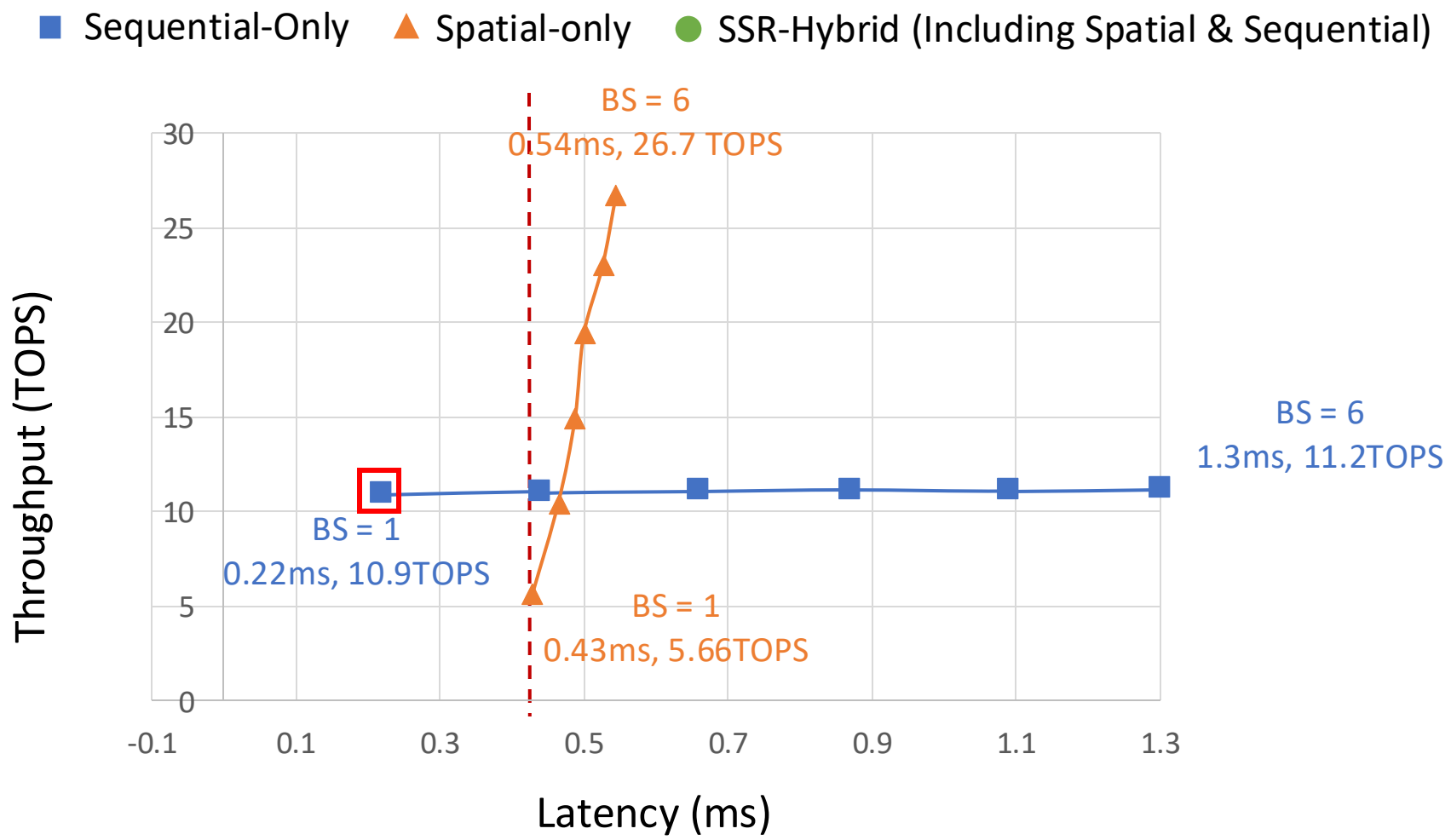
Experiment Results

- Latency & Throughput Tradeoff Analysis of DeiT-T across Different Strategies



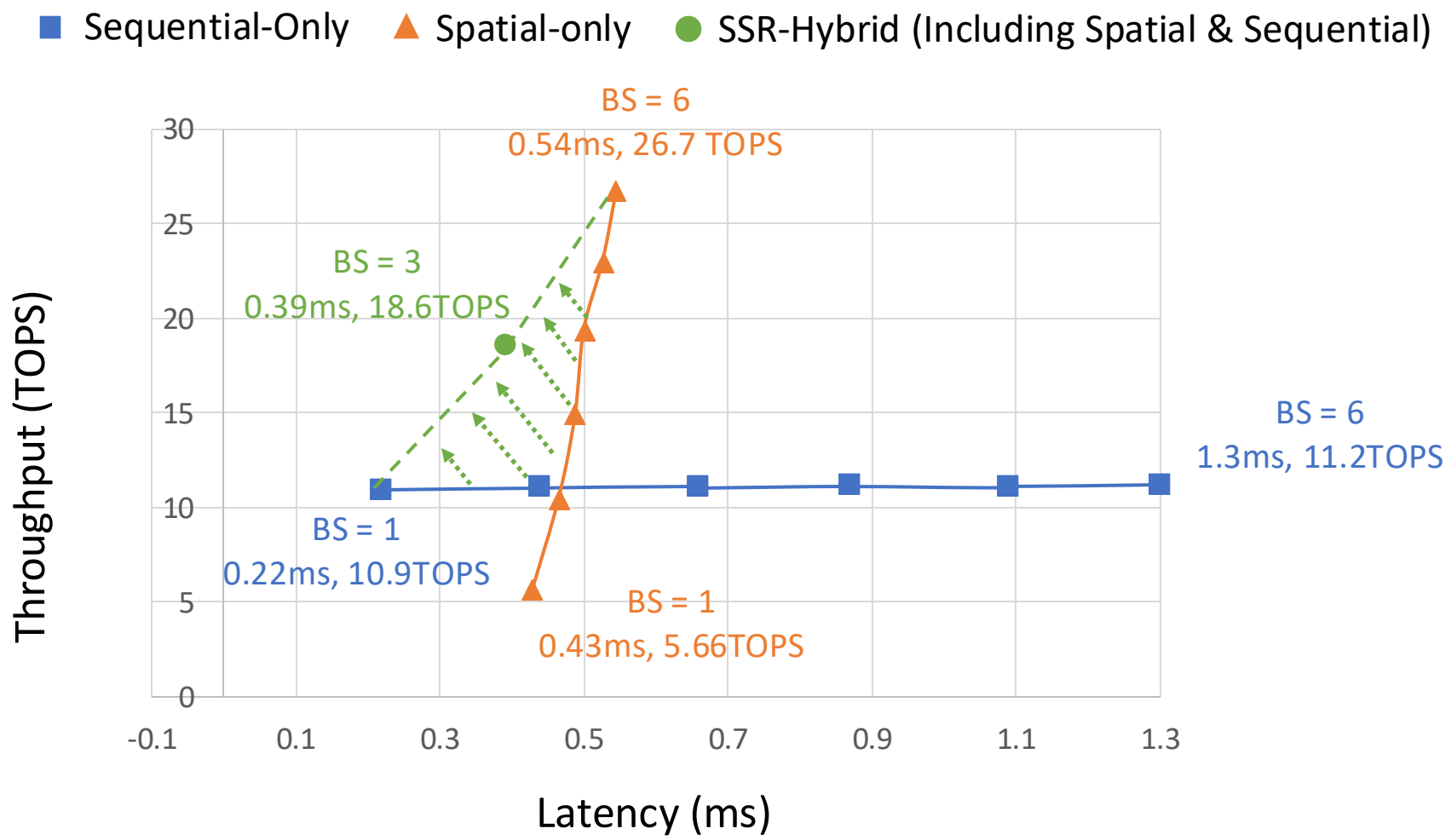
Experiment Results

- Latency & Throughput Tradeoff Analysis of DeiT-T across Different Strategies



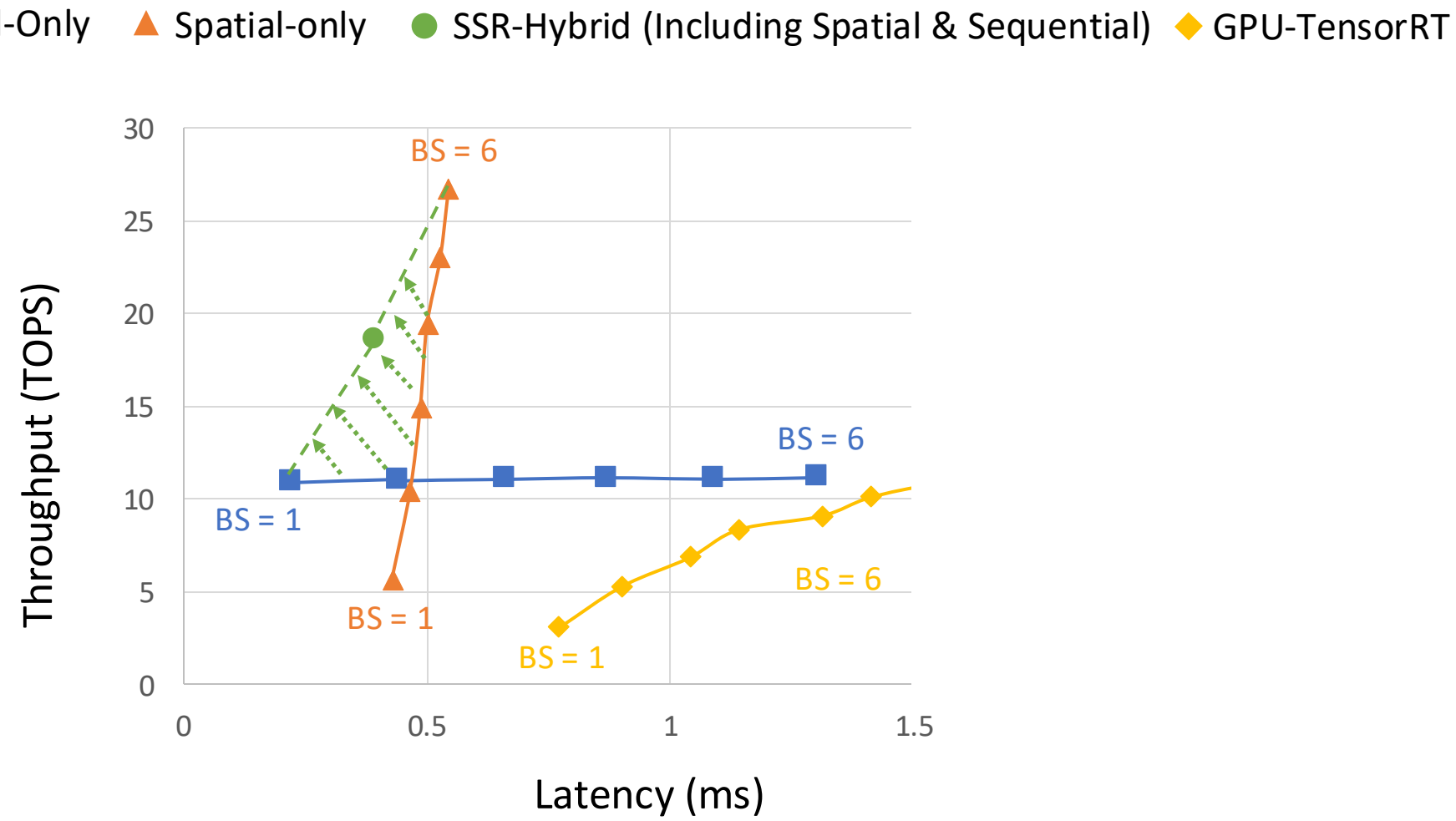
Experiment Results

- Latency & Throughput Tradeoff Analysis of DeiT-T across Different Strategies



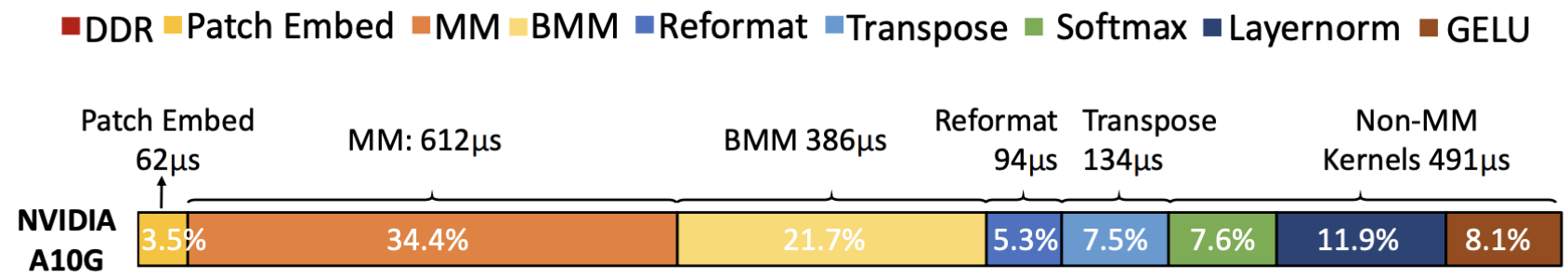
Experiment Results

- SSR (ours) vs TensorRT on GPU A10G

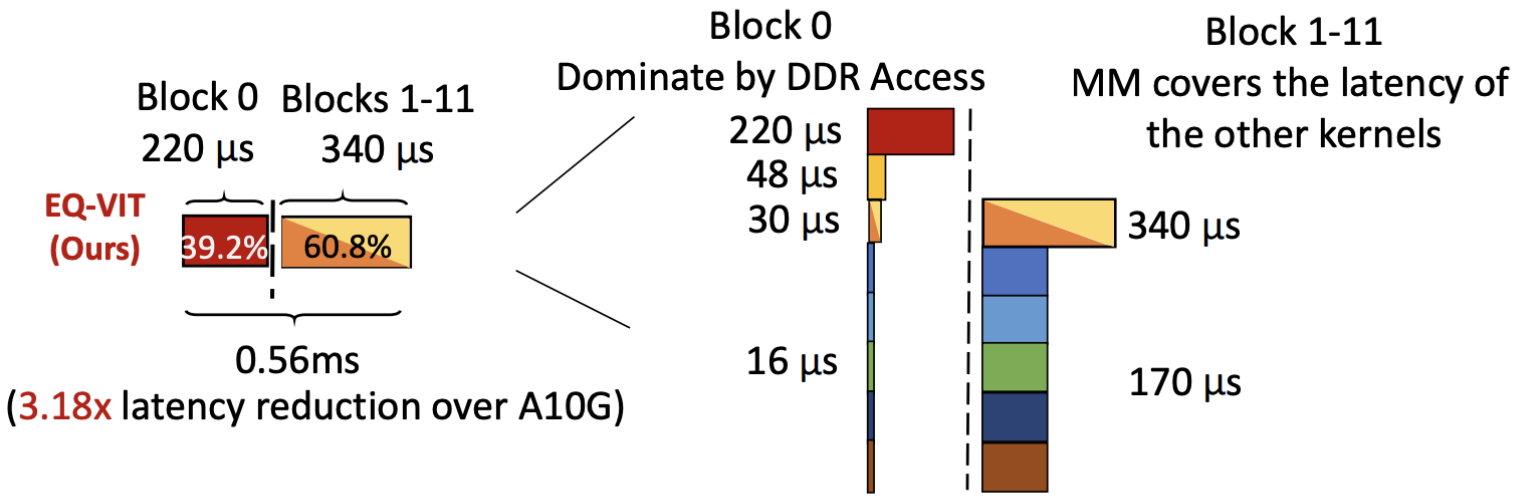


Experiment Results Further Analysis

- SSR (ours) vs TensorRT on GPU A10G



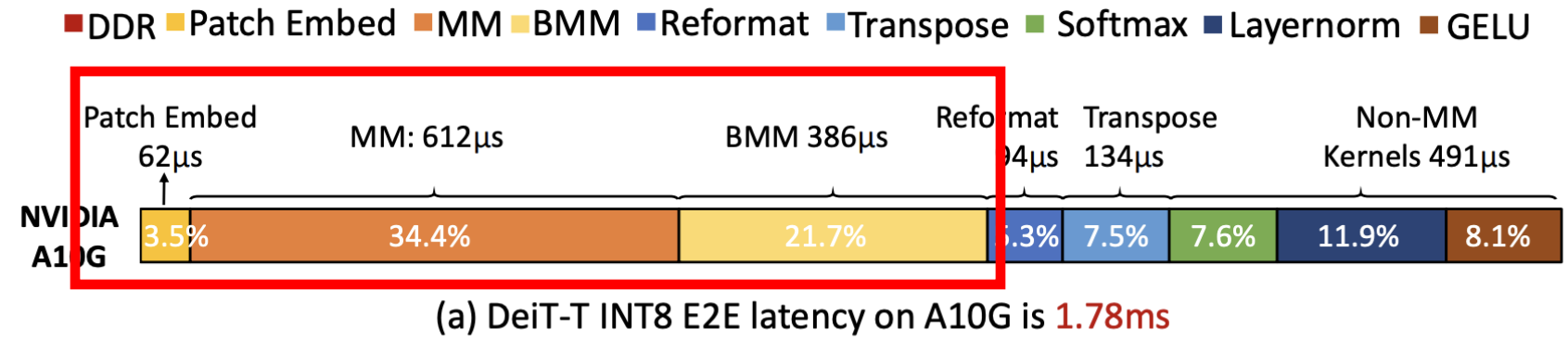
(a) DeiT-T INT8 E2E latency on A10G is 1.78ms



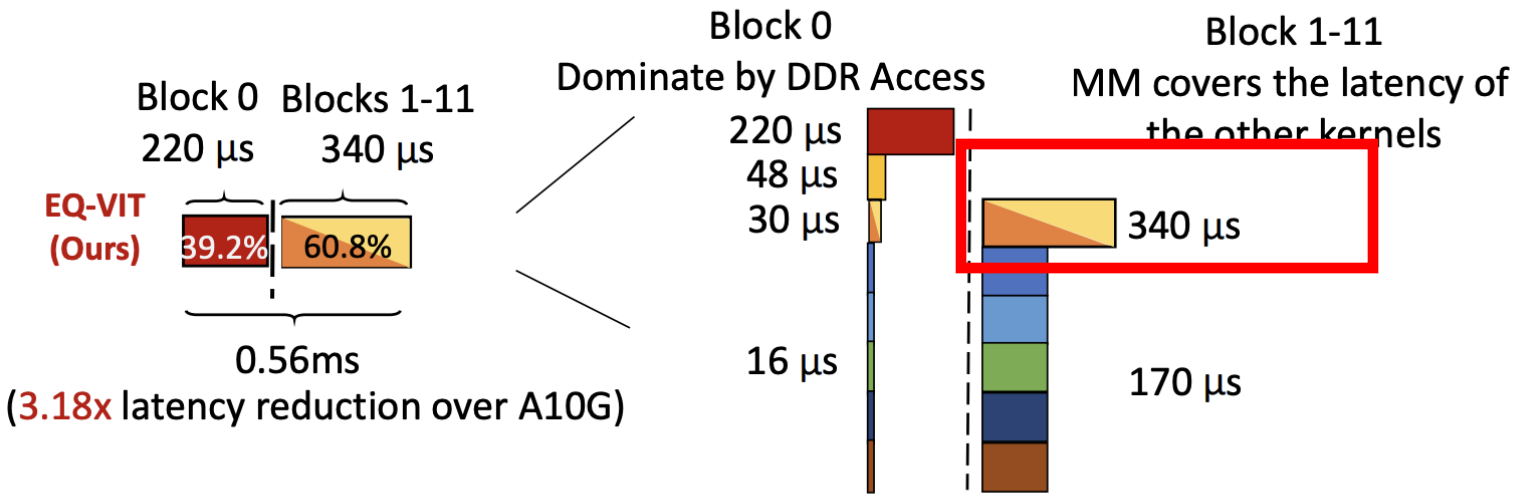
(b) DeiT-T INT8 E2E latency on EQ-ViT (Ours) is 0.56ms

Experiment Results Further Analysis

- SSR (ours) vs TensorRT on GPU A10G



(a) DeiT-T INT8 E2E latency on A10G is 1.78ms

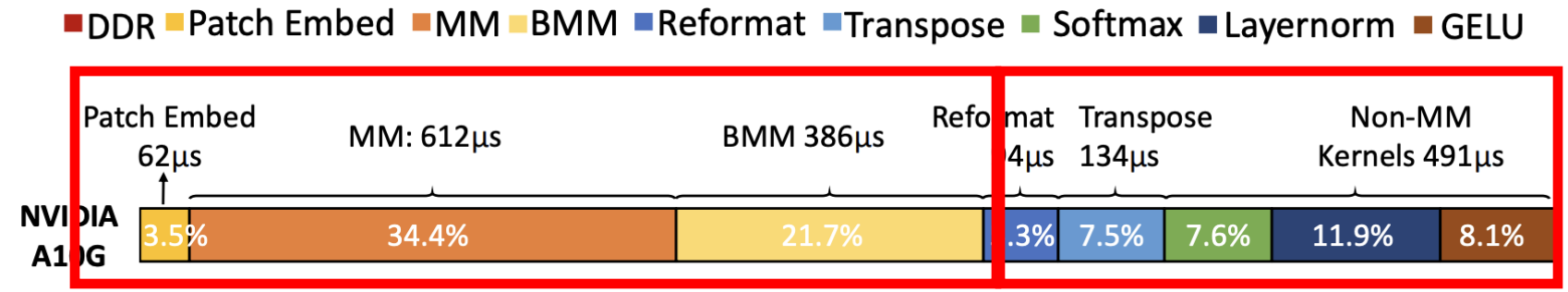


(b) DeiT-T INT8 E2E latency on EQ-ViT (Ours) is 0.56ms

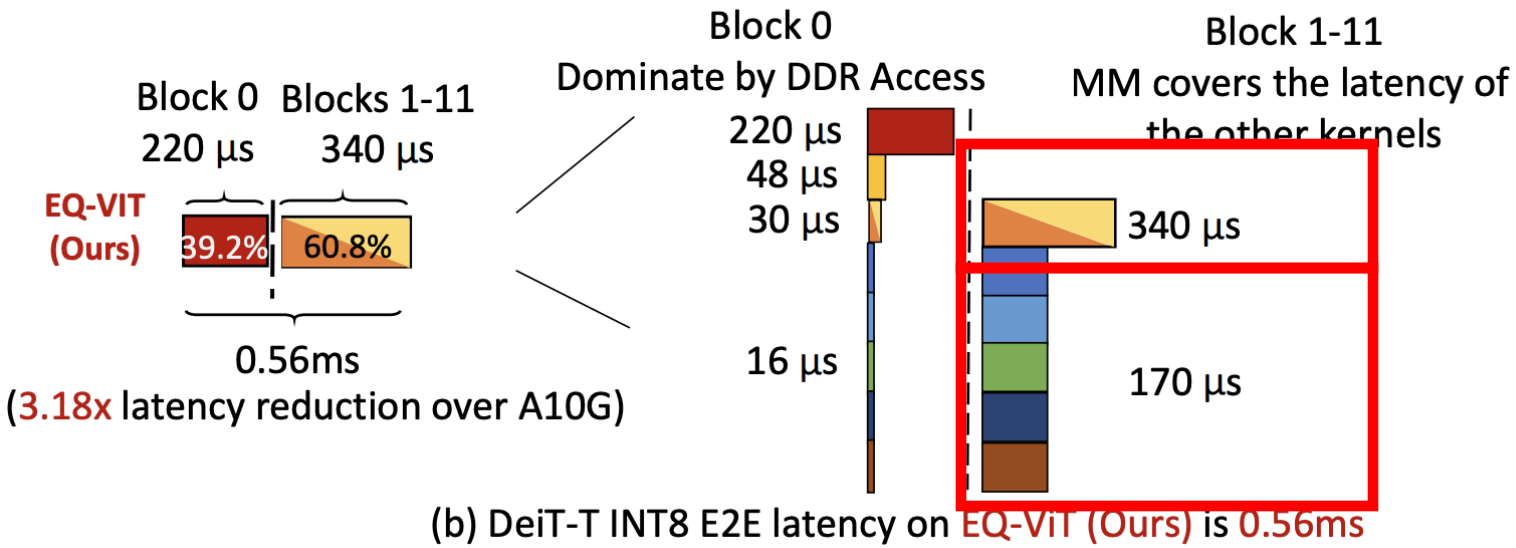
Multiple Spatial Accs -> Improve tensor core utilization

Experiment Results Further Analysis

- SSR (ours) vs TensorRT on GPU A10G



(a) DeiT-T INT8 E2E latency on A10G is 1.78ms



(b) DeiT-T INT8 E2E latency on EQ-ViT (Ours) is 0.56ms

Multiple Spatial Accs -> Improve tensor core utilization

On-chip Forwarding & Fine Grained Pipeline -> Overlap execution of MM & non-MM kernels

Apply SSR to Other Platforms, GPUs?

Mapping features	FPGA+SOTA Impl. (HeatViT)	FPGA+EQ-ViT Optimizations	GPU+ SOTA Impl. (TensorRT)	GPU+EQ-ViT Optimizations	ACAP+SOTA Impl. (CHARM)	ACAP+EQ-ViT Optimization
Quantization	yes	yes	partial	partial ->yes	no	yes (4.2x)
On-Chip Forwarding	no	yes	no	arch limit	no	yes (3.4x)
Multi Spatial Accelerators	no	yes	no	arch limit	yes	yes (2.3x)
Fine-grained Pipelining	no	yes	no	arch limit	no	yes (2.7x)
Utilize AI-optimized PEs	no	arch limit	yes	yes	yes	yes
Estimated latency after EQ-ViT	7.3ms	3.9ms	1.8ms	1.05ms	50ms (1x)	0.561ms (89x)

Apply SSR to Other Platforms, GPUs?

Mapping features	FPGA+SOTA Impl. (HeatViT)	FPGA+EQ-ViT Optimizations	GPU+ SOTA Impl. (TensorRT)	GPU+EQ-ViT Optimizations	ACAP+SOTA Impl. (CHARM)	ACAP+EQ-ViT Optimization
Quantization	yes	yes	partial	partial ->yes	no	yes (4.2x)
On-Chip Forwarding	no	yes	no	arch limit	no	yes (3.4x)
Multi Spatial Accelerators	no	yes	no	arch limit	yes	yes (2.3x)
Fine-grained Pipelining	no	yes	no	arch limit	no	yes (2.7x)
Utilize AI-optimized PEs	no	arch limit	yes	yes	yes	yes
Estimated latency after EQ-ViT	7.3ms	3.9ms	1.8ms	1.05ms	50ms (1x)	0.561ms (89x)

- Can we design “GPU” better than current GPU?

Apply SSR to Other Platforms, GPUs?

Mapping features	FPGA+SOTA Impl. (HeatViT)	FPGA+EQ-ViT Optimizations	GPU+ SOTA Impl. (TensorRT)	GPU+EQ-ViT Optimizations	ACAP+SOTA Impl. (CHARM)	ACAP+EQ-ViT Optimization
Quantization	yes	yes	partial	partial ->yes	no	yes (4.2x)
On-Chip Forwarding	no	yes	no	arch limit	no	yes (3.4x)
Multi Spatial Accelerators	no	yes	no	arch limit	yes	yes (2.3x)
Fine-grained Pipelining	no	yes	no	arch limit	no	yes (2.7x)
Utilize AI-optimized PEs	no	arch limit	yes	yes	yes	yes
Estimated latency after EQ-ViT	7.3ms	3.9ms	1.8ms	1.05ms	50ms (1x)	0.561ms (89x)

- Can we design “GPU” better than current GPU?

- Shall we introduce FPGA or reconfigurable architecture in broader GPU architecture to improve latency?

Apply SSR to Other Platforms, GPUs?

Mapping features	FPGA+SOTA Impl. (HeatViT)	FPGA+EQ-ViT Optimizations	GPU+ SOTA Impl. (TensorRT)	GPU+EQ-ViT Optimizations	ACAP+SOTA Impl. (CHARM)	ACAP+EQ-ViT Optimization
Quantization	yes	yes	partial	partial ->yes	no	yes (4.2x)
On-Chip Forwarding	no	yes	no	arch limit	no	yes (3.4x)
Multi Spatial Accelerators	no	yes	no	arch limit	yes	yes (2.3x)
Fine-grained Pipelining	no	yes	no	arch limit	no	yes (2.7x)
Utilize AI-optimized PEs	no	arch limit	yes	yes	yes	yes
Estimated latency after EQ-ViT	7.3ms	3.9ms	1.8ms	1.05ms	50ms (1x)	0.561ms (89x)

- Can we design “GPU” better than current GPU?
- Shall we introduce FPGA or reconfigurable architecture in broader GPU architecture to improve latency?
- If FPGA is too fine-grained, what is the least reconfigurability needed in future architecture to balance performance and adaptability? ⇔ <=> Microarchitecture support to enable flexible mapping & execution model

CHARM & SSR Impacts

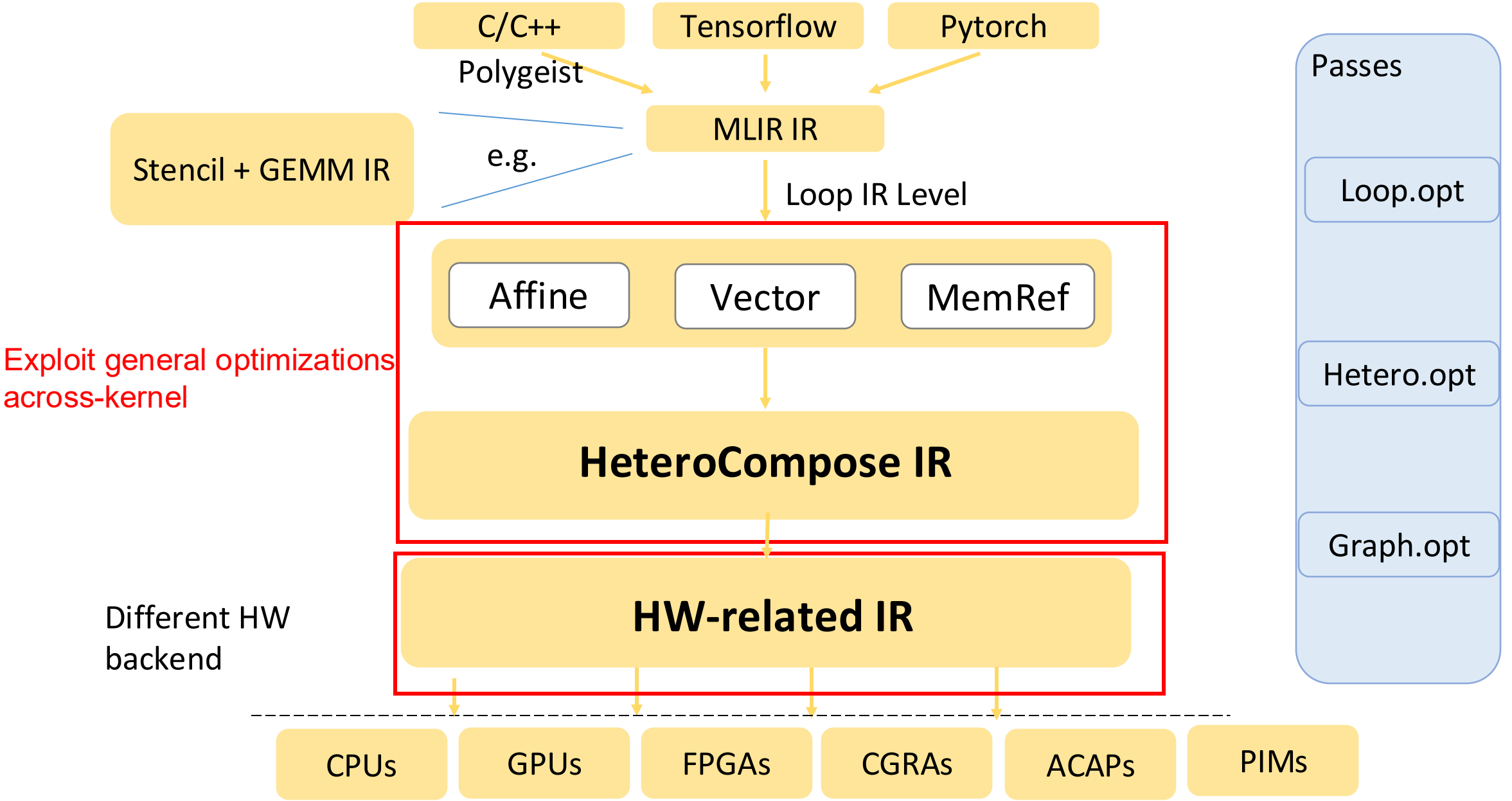
- Impacts (Application): Smart Health, Smart Transportation, AIoT & IoS,
- Impacts (Architecture): Heterogeneous Domain Specific Architecture
- Impacts (Programming): GitHub Open-source, No.1 Downloaded (>2300) Paper in FPGA'23

ART: Accelerator Customization for Real-Time System

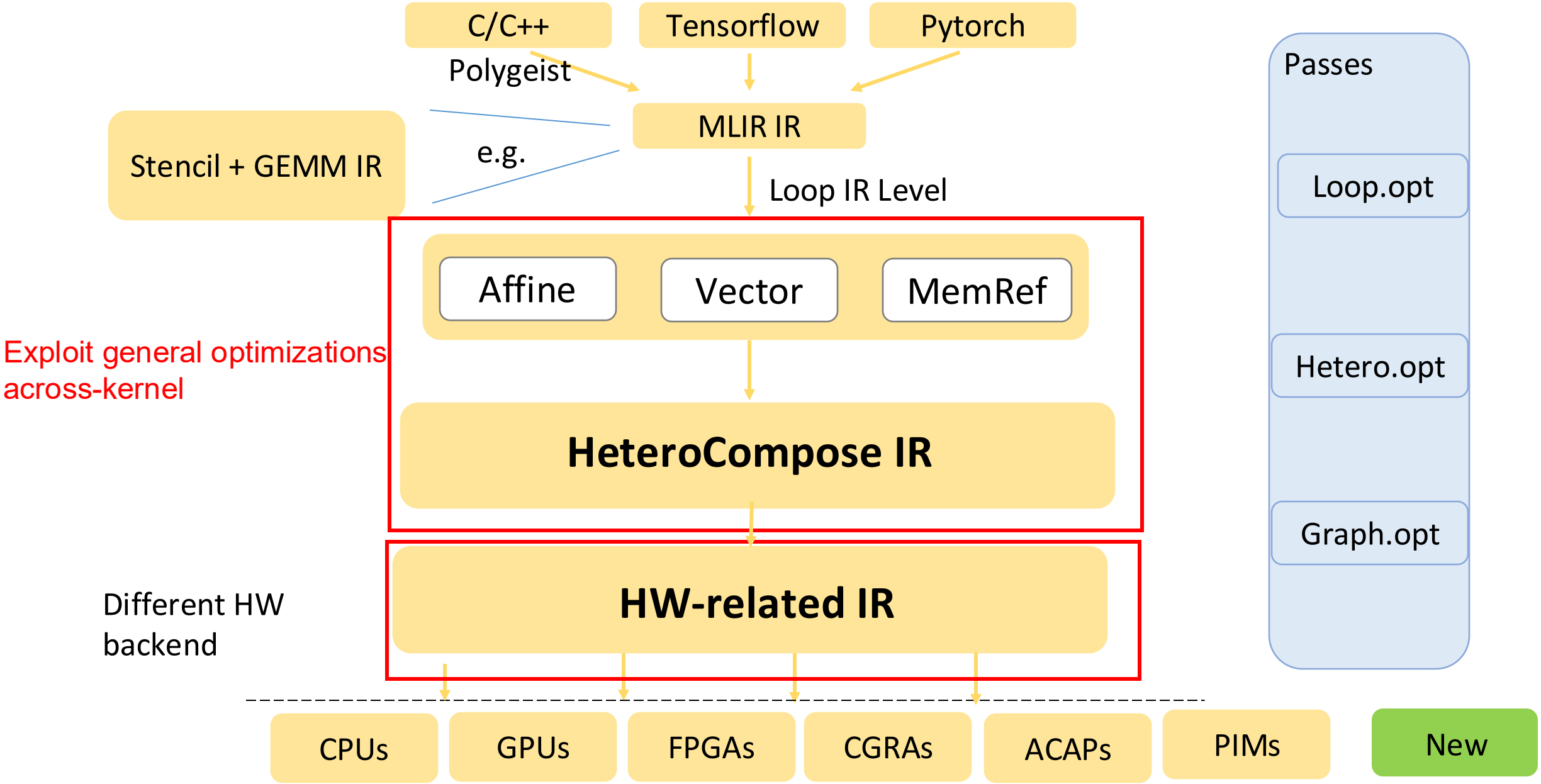
Collaboration with the CAR Lab, University of Delaware, Wayne State University

NSF #2213701: Collaborative Research: CCRI: New: A Scalable Hardware and Software Environment Enabling Secure Multi-party Learning
NSF #2217003: Collaborative Research: PPoSS: LARGE: Co-designing Hardware, Software, and Algorithms to Enable Extreme-Scale Machine Learning Systems

Ongoing Research: MLIR-based HeteroCompose Compilation Framework



Ongoing Research: MLIR-based HeteroCompose Compilation Framework



Thank you & Questions?