MICRO 2025 Session 7B: Tools and Simulators

PyTorchSim: A Comprehensive, Fast, and Accurate NPU Simulation Framework

Wonhyuk Yang[*], Yunseon Shin[*], Okkyun Woo[*], Geonwoo Park[§], Hyunkyu Ham, Jeehoon Kang[†¶], Jongse Park[¶], Gwangsun Kim

POSTECH
Parallel System Architecture Lab.

[‖] [†] FuriosaAI

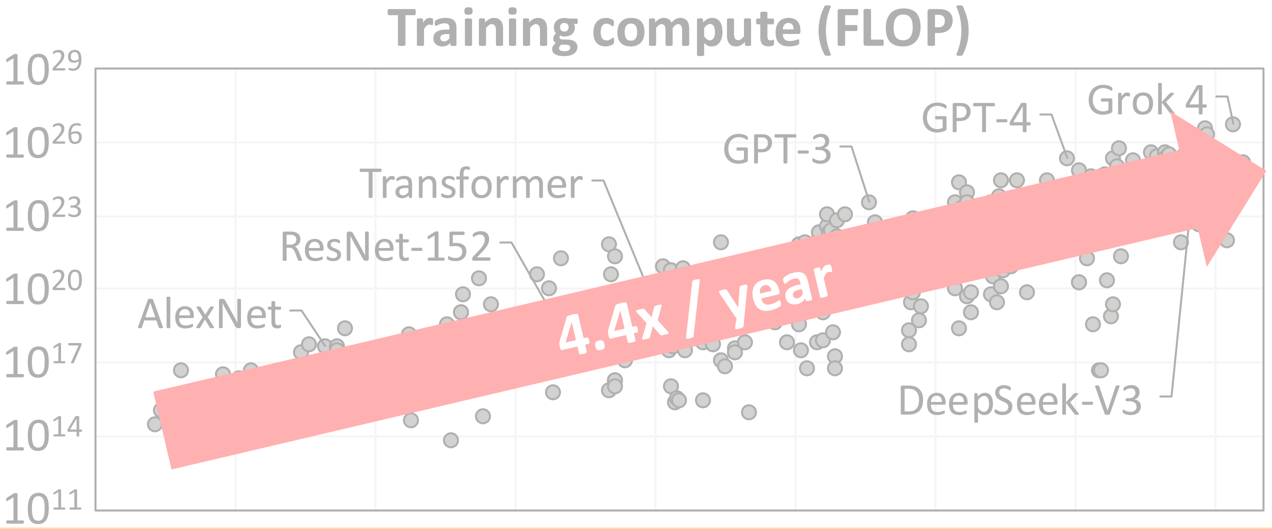
[¶] KAIST

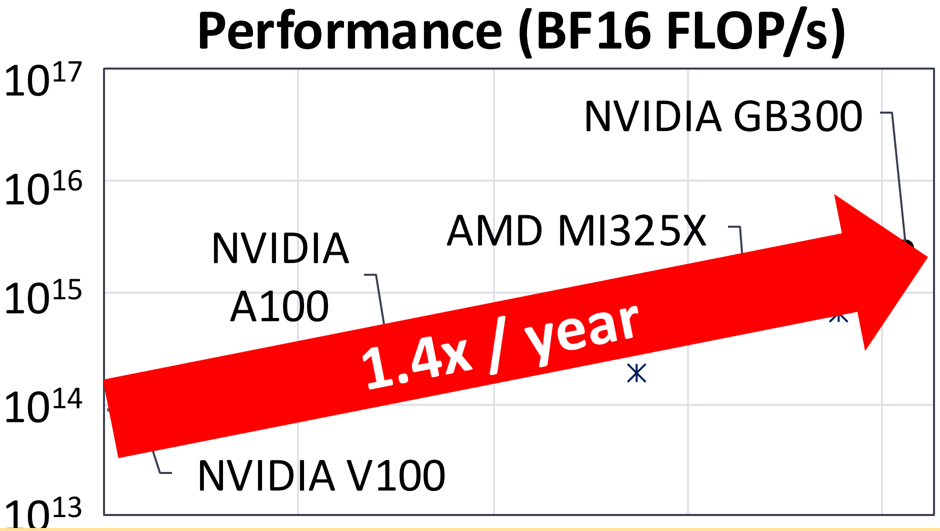
[*] Equal contribution

[§] Currently with Samsung Electronics

AI Model & Hardware Trends

- **Deep Neural Network (DNN)** are growing exponentially in size
- **Hardware performance** improves relatively slowly





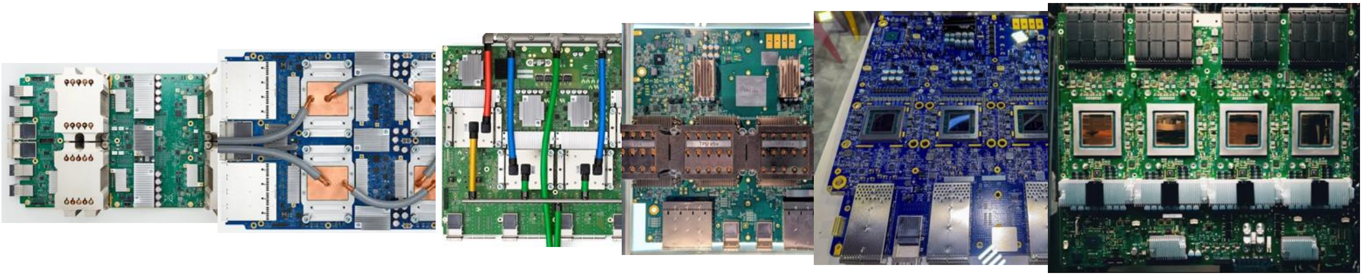
We need **high-performance** & **efficient hardware**

Evolution of Neural Processing Units

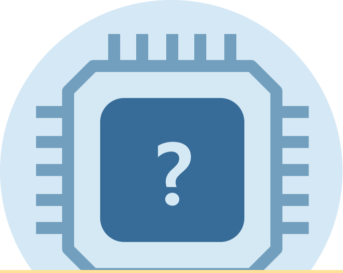
Neural Processing Units (NPUs) are designed to address this challenge

- Efficient dataflow units (e.g., systolic arrays, adder-tree)

TPU v2 TPU v3 TPU v4 TPU v5 TPU v6 TPU v7



Future NPUs



Inferentia2

Trainium2

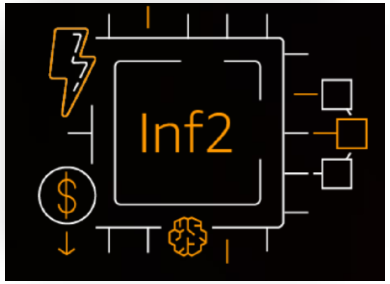
Intel Gaudi3

NPU simulators play a crucial role in designing **NPUs**

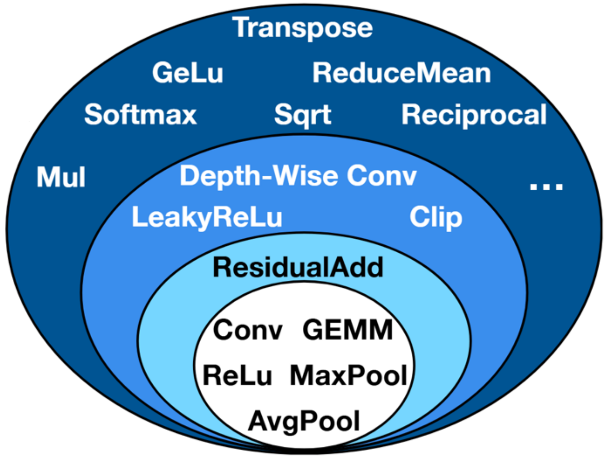
NPU Simulator Requirements – AI Model

- Both **inference** and **training** are important
- **(Vector) operations** are becoming increasingly diverse
- **Sparsity** is now widely exploited for efficiency
  - Latency varies with input data (i.e., **data-dependent timing behavior**)

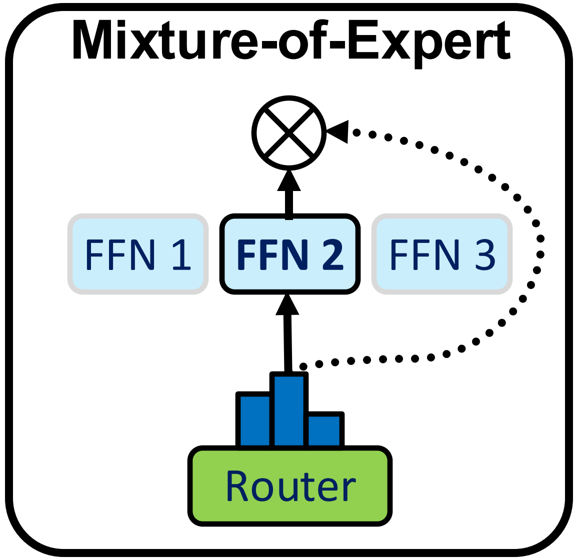
Inferentia2
(Inference)



Trainium2
(Training)

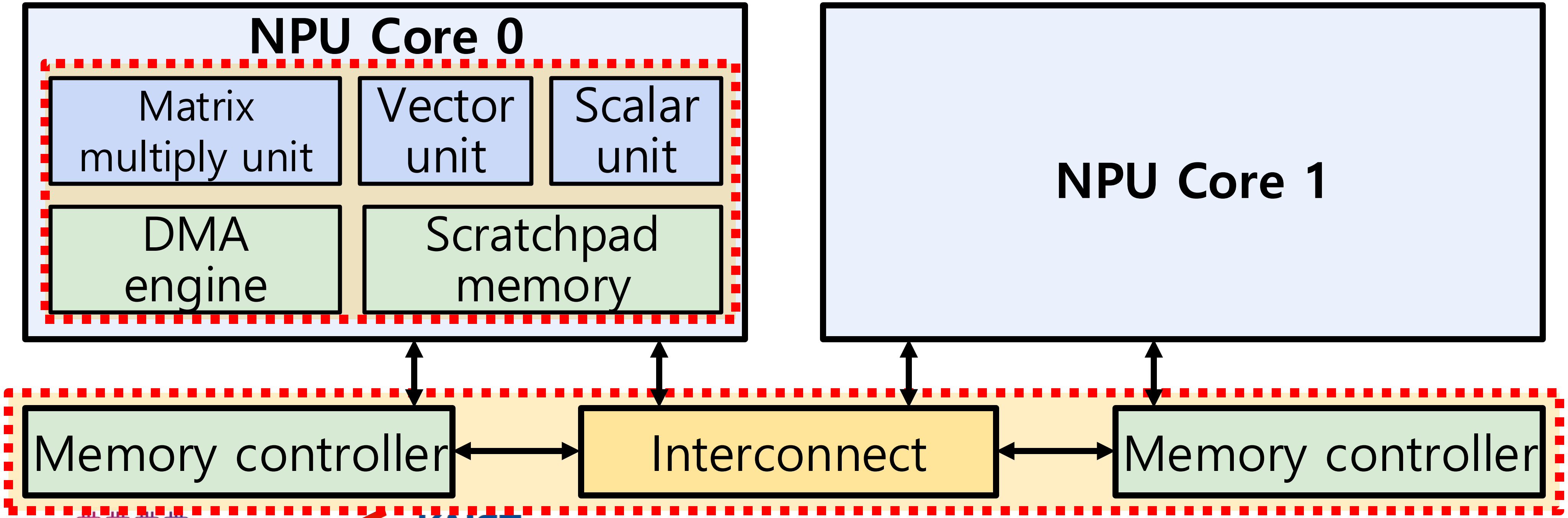


Source: Tandem Processor
[Ghodrati, Soroush, et al,. ASPLOS 2024]



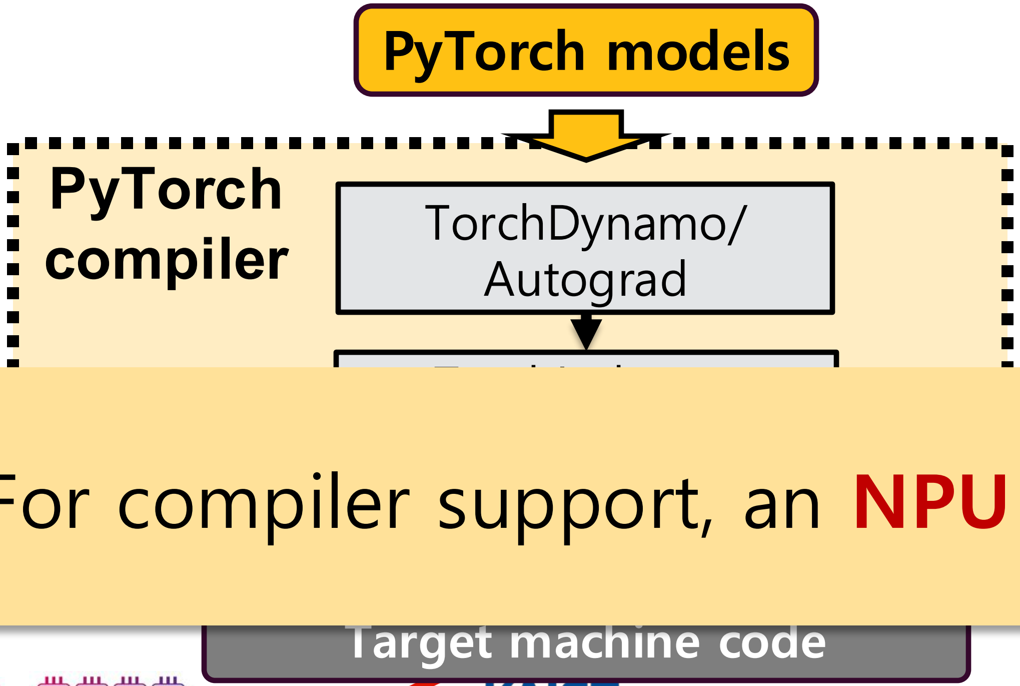
NPU Simulator Requirements – NPU Hardware

- Need to accurately model key components of modern NPUs
- **NPU core**: matrix, vector, and scalar units and scratchpad memory
- **Shared resources**: interconnect and DRAM
  - Detailed simulation often necessary to model inter-core contention

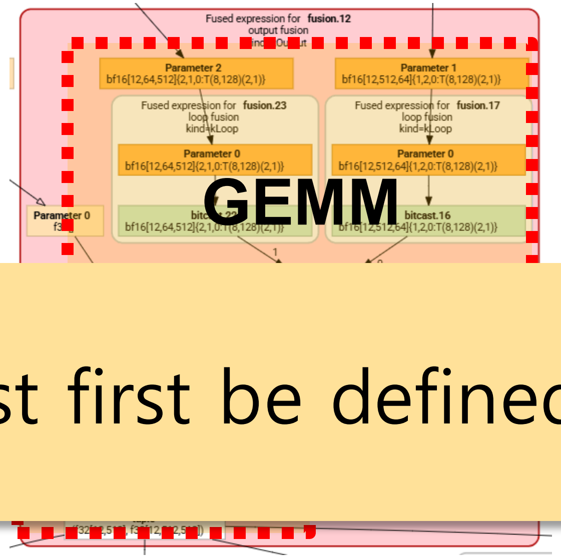


NPU Simulator Requirements – Compiler

- Necessary to bridge AI models and NPU hardware
- Lowers DNN models into machine code
- Apply various optimizations
- Enables full-model simulation for both inference and training



Operation fusion


Fused expression for fusion.12
output fusion
Parameter 2
bf16[12,64,512]{2,1,0:T(8,128)(2,1)}
Parameter 1
bf16[12,512,64]{1,2,0:T(8,128)(2,1)}
Fused expression for fusion.23
loop fusion
kind=kLoop
Fused expression for fusion.17
loop fusion
kind=kLoop
Parameter 0
bf16[12,64,512]{2,1,0:T(8,128)(2,1)}
Parameter 0
bf16[12,512,64]{1,2,0:T(8,128)(2,1)}
GEMM
Parameter 0
bitcast.16
bf16[12,64,512]{2,1,0:T(8,128)(2,1)}
bf16[12,512,64]{1,2,0:T(8,128)(2,1)}

For compiler support, an **NPU ISA** must first be defined

NPU ISA

- No de-facto standard ISA for NPUs (unlike CPUs and GPUs)

- RISC-V can be a strong candidate for an NPU ISA
  - Generality:
    - Vector-length agnostic design
    - A generic interface (VCIX) for dataflow units
  - Extensibility: Reserved opcode space for custom instructions
  - Openness: Contributions from both academia and industry
  - SW Infrastructure: Rich software ecosystem (e.g., compilers, simulators)

Existing NPU Models

- **Various NPU simulators** and **analytical models** have been proposed

Timeline

2019

2020

2021

2022

SMAUG, MAESTRO

Timeloop

Stonne

mNPUSim,

None of them meets all the key requirements

1. Full AI model support ✗

2. AI compiler support ✗

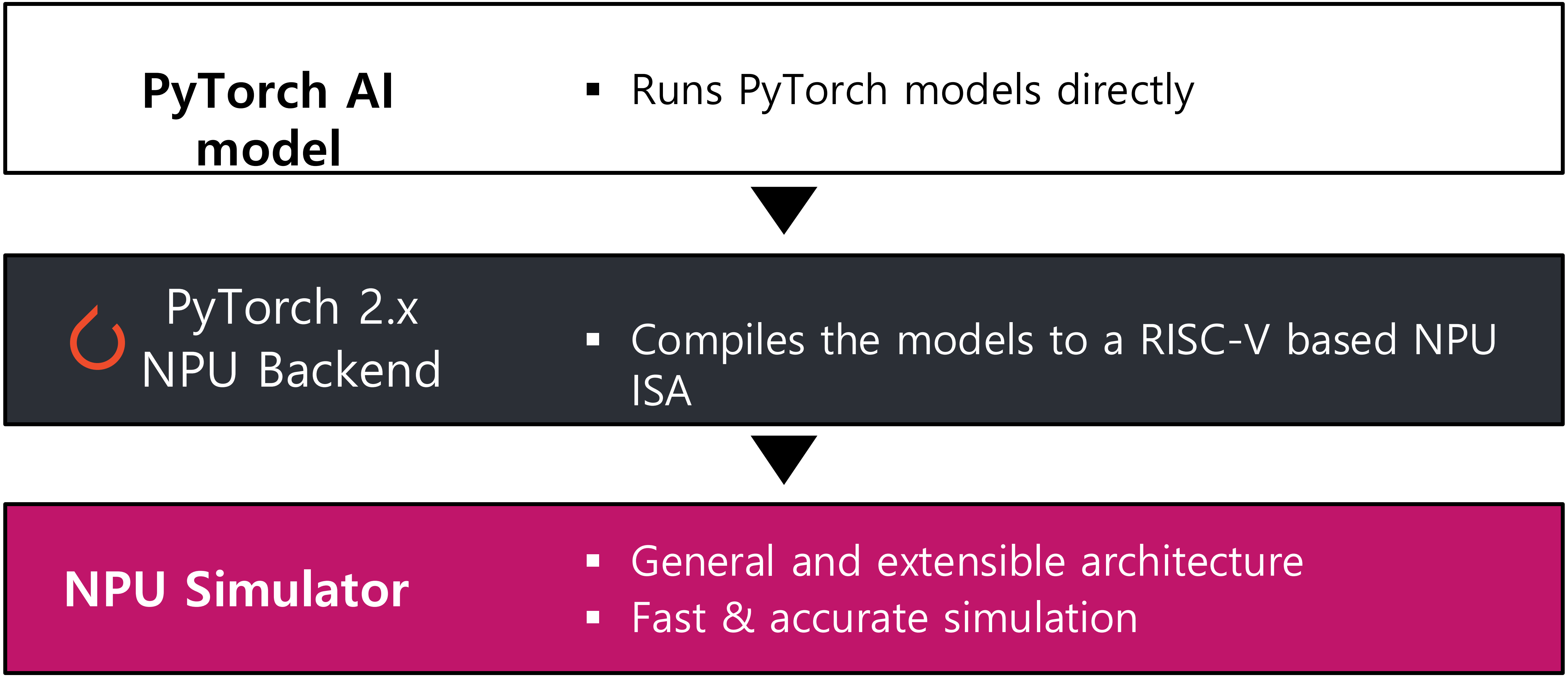
3. NPU core & shared resource △

We propose **PyTorchSim**, which can better satisfy the key requirements for NPU simulation

Contents

- Background / Motivation
- **PyTorchSim**
  - PyTorchSim overview
  - NPU core modeling
  - Compilation flow
- Methodology & Evaluation
- Case study
  - Impact of DNN Training Hyperparameter
  - Scheduling for Chiplet-based NPUs
- Summary

PyTorchSim Framework: Overview



PyTorch AI model

- Runs PyTorch models directly

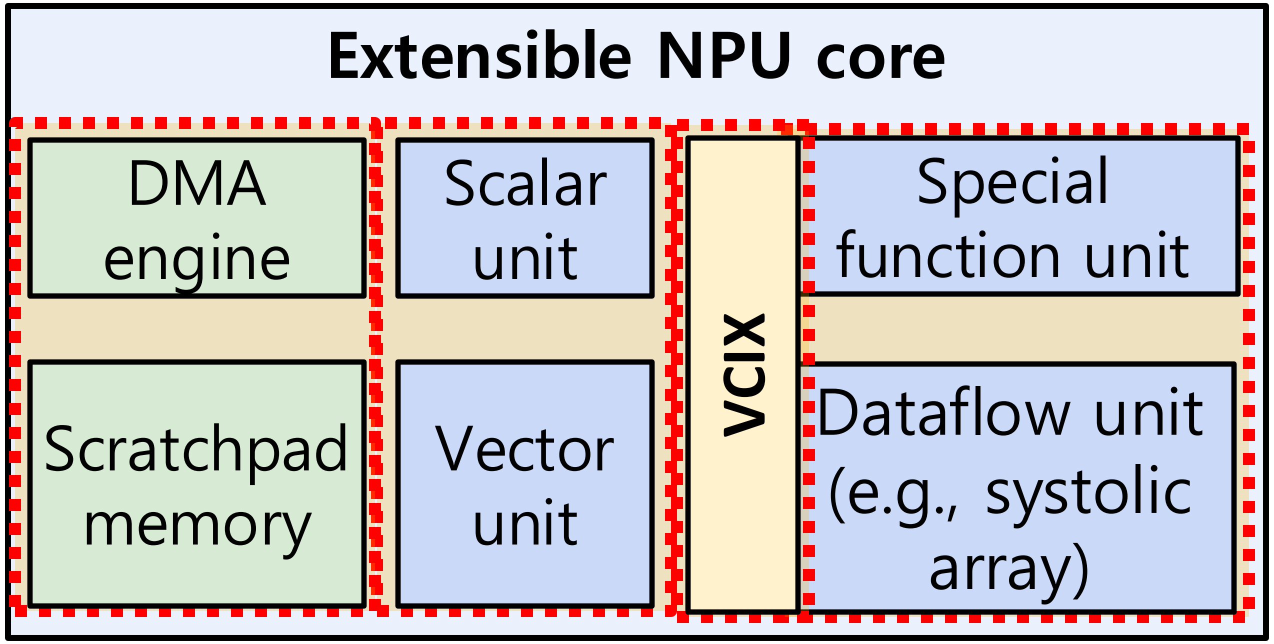
PyTorch 2.x NPU Backend

- Compiles the models to a RISC-V based NPU ISA

NPU Simulator

- General and extensible architecture
- Fast & accurate simulation

Modeling Common Building Blocks in NPU Cores



Extensible NPU ISA

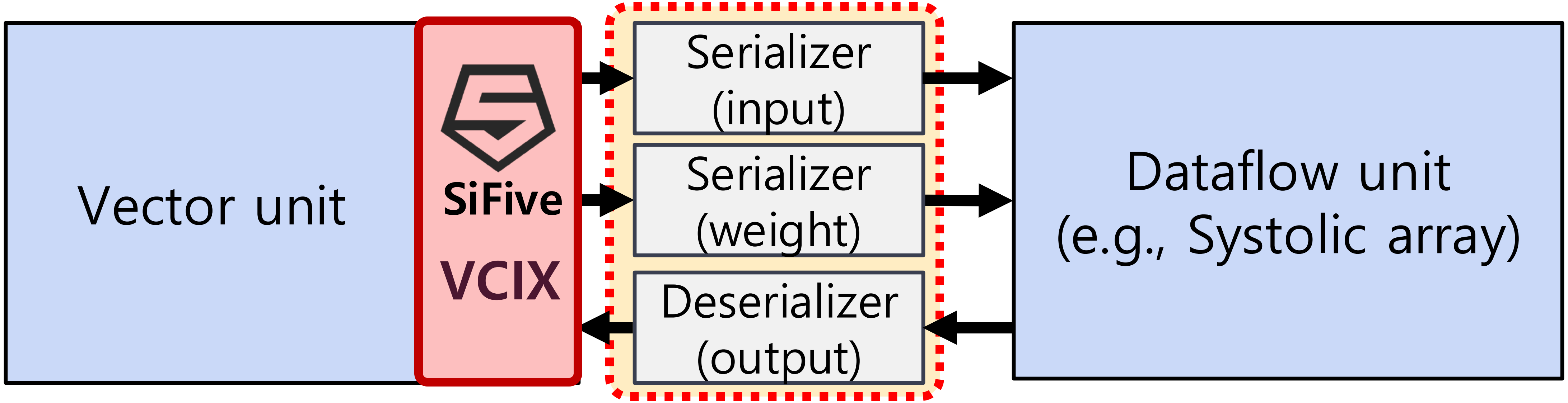
- **Scalar unit**
- **Vector unit**

} **RISC-V + Vector extension**

- **DMA engine**
- **SFU.**
- **Dataflow unit**

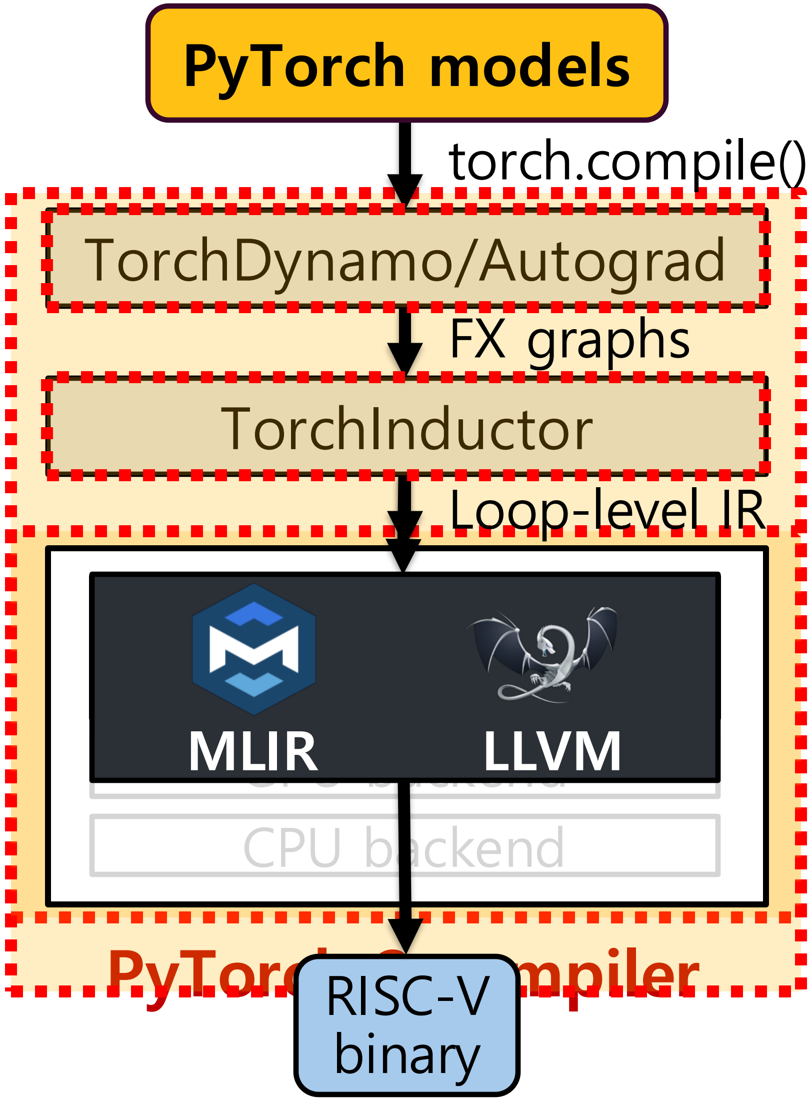
} **Custom Instructions**

VCIX: A Generic Interface for Dataflow Units



We adopt VCIX, a generic interface for diverse dataflow units

PyTorchSim Compilation Flow

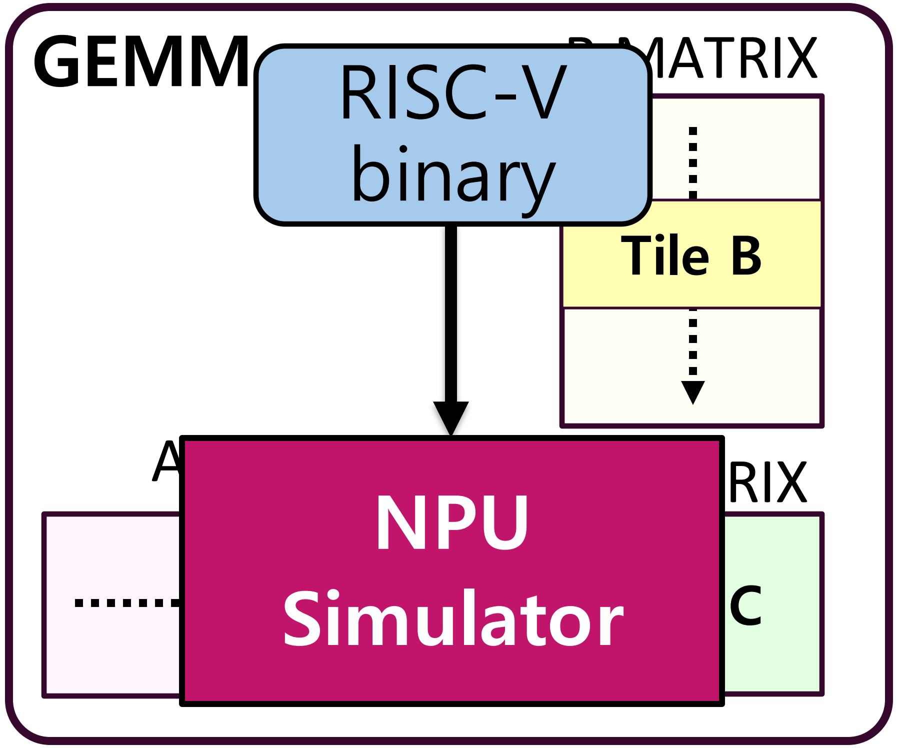


- Benefits of integration with PyTorch 2
  - **Simulate existing PyTorch models** without any need for manual conversion
  - Supports **training simulation** through PyTorch's auto-differentiation
  - Leverage existing **target-independent optimizations**

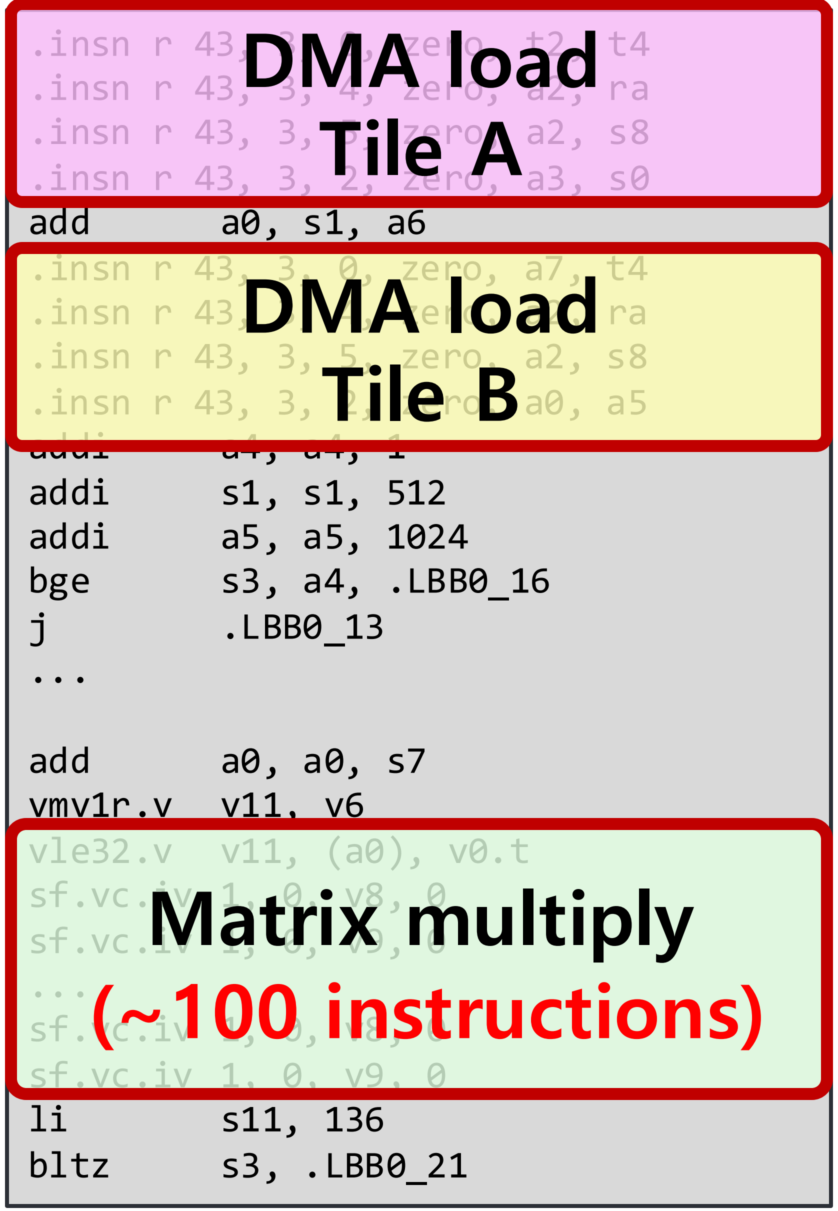
- **NPU backend** generates a **RISC-V** binary
  - Applies target-dependent optimizations

Instruction-Level Simulation (ILS)

- Simulate every single instruction one-by-one
- **Inherently slow**



RISC-V binary (GEMM)

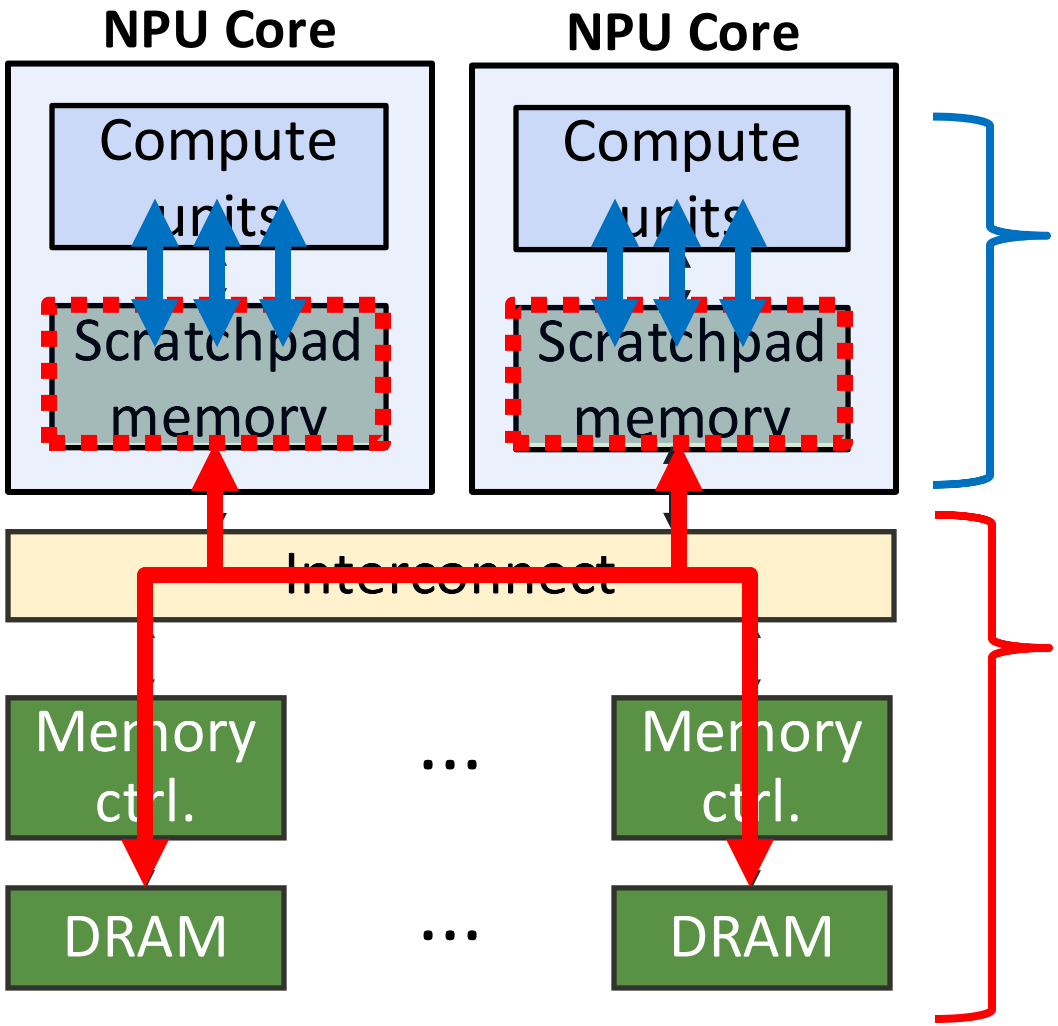


```
.insn r 43, 3, 0, zero, t2, t4
.insn r 43, 3, 4, zero, a2, ra
.insn r 43, 3, 5, zero, a2, s8
.insn r 43, 3, 2, zero, a3, s0
add         a0, s1, a6
.insn r 43, 3, 0, zero, a7, t4
.insn r 43, 3, 4, zero, a2, ra
.insn r 43, 3, 5, zero, a2, s8
.insn r 43, 3, 2, zero, a0, a5
addi        a4, a4, 1
addi        s1, s1, 512
addi        a5, a5, 1024
bge         s3, a4, .LBB0_16
j           .LBB0_13
...

add         a0, a0, s7
vmv1r.v     v11, v6
vle32.v     v11, (a0), v0.t
sf.vc.iv    1, 0, v8, 0
sf.vc.iv    1, 0, v9, 0
...
sf.vc.iv    1, 0, v8, 0
sf.vc.iv    1, 0, v9, 0
li          s11, 136
bltz        s3, .LBB0_21
```

DMA load Tile A
DMA load Tile B
Matrix multiply
(~100 instructions)

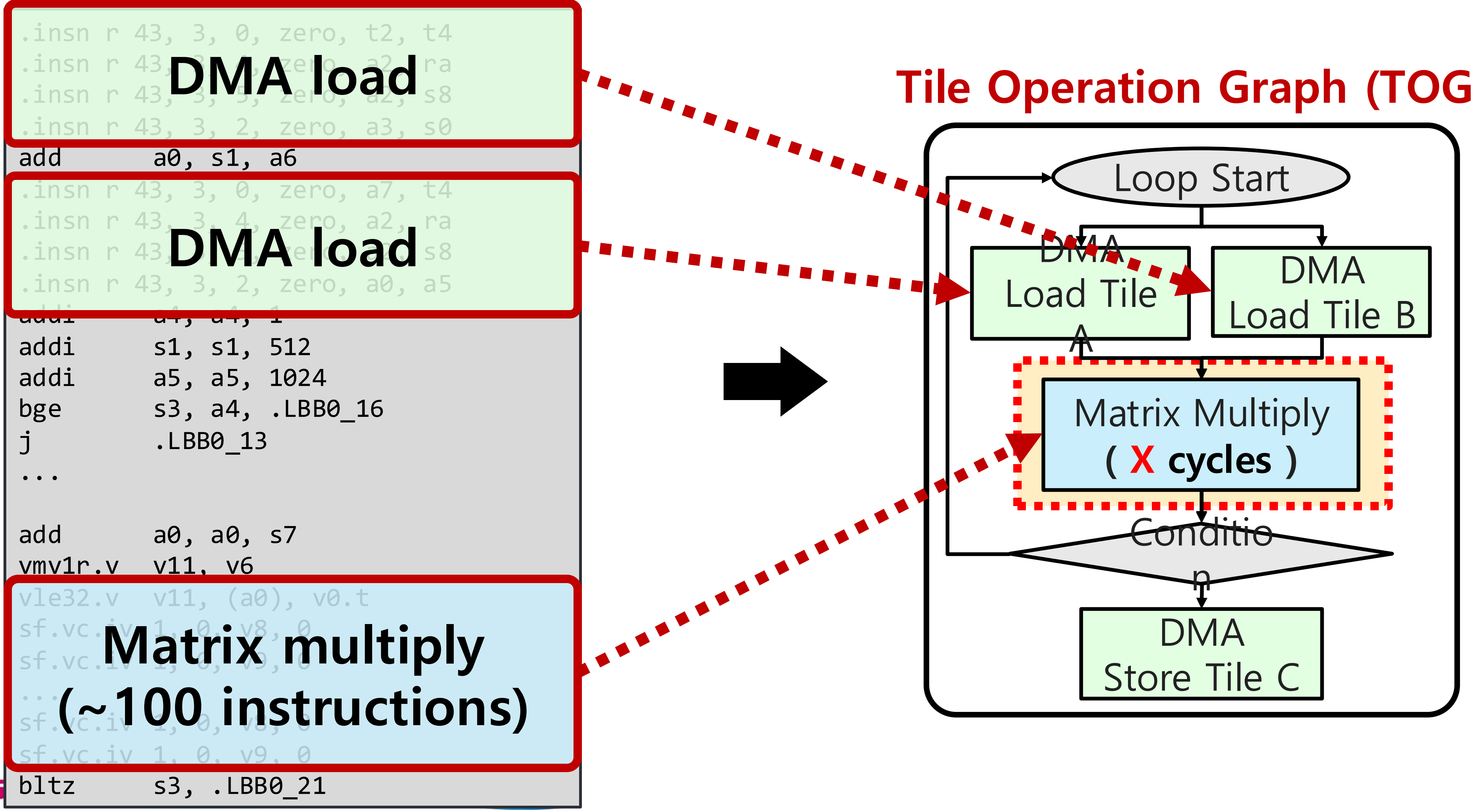
Key Insight behind NPU Execution



- Intra-core **compute latency** is *deterministic*
- **NoC & DRAM latencies** remain ***non-deterministic*** due to inter-core contention and traffic

Tile Operation Graph (TOG)

RISC-V binary (GEMM)



```
.insn r 43, 3, 0, zero, t2, t4
.insn r 43, 3, 4, zero, a2, ra
.insn r 43, 3, 5, zero, a2, s8
.insn r 43, 3, 2, zero, a3, s0
add       a0, s1, a6
.insn r 43, 3, 0, zero, a7, t4
.insn r 43, 3, 4, zero, a2, ra
.insn r 43, 3, 5, zero, a2, s8
.insn r 43, 3, 2, zero, a0, a5
addi      a4, a4, 1
addi      s1, s1, 512
addi      a5, a5, 1024
bge       s3, a4, .LBB0_16
j         .LBB0_13
...

add       a0, a0, s7
vmv1r.v   v11, v6
vle32.v   v11, (a0), v0.t
sf.vc.iv  1, 0, v8, 0
sf.vc.iv  1, 0, v9, 0
...
sf.vc.iv  1, 0, v8, 0
sf.vc.iv  1, 0, v9, 0
bltz      s3, .LBB0_21
```

Tile Operation Graph (TOG)

Tile-Level Simulation (TLS) Execution Flow

Tile-Level Simulation works in a two-step flow

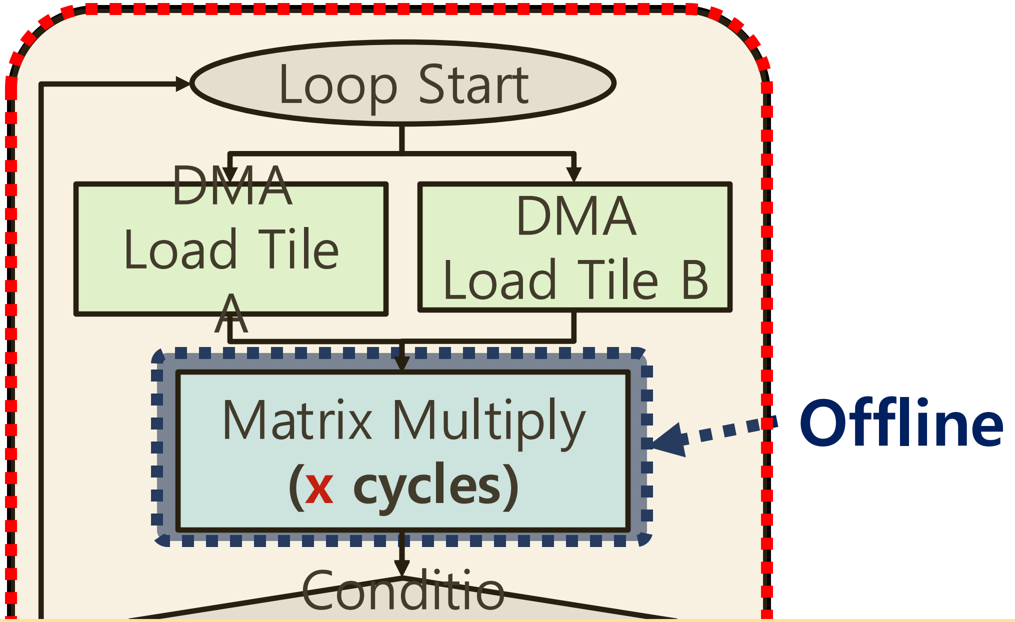
Offline (compile time)

- Obtain the deterministic compute latency in the TOG

Online (simulation time)

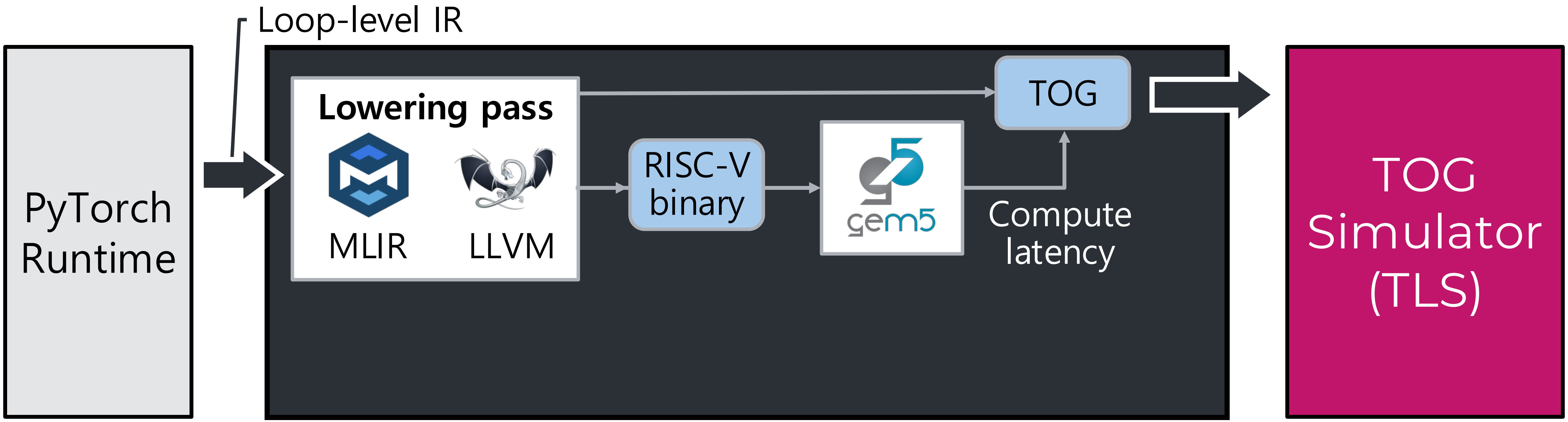
- Reuse the obtained compute latency

Tile Operation Graph (TOG)



TLS can achieve both **high simulation speed** and **accuracy**

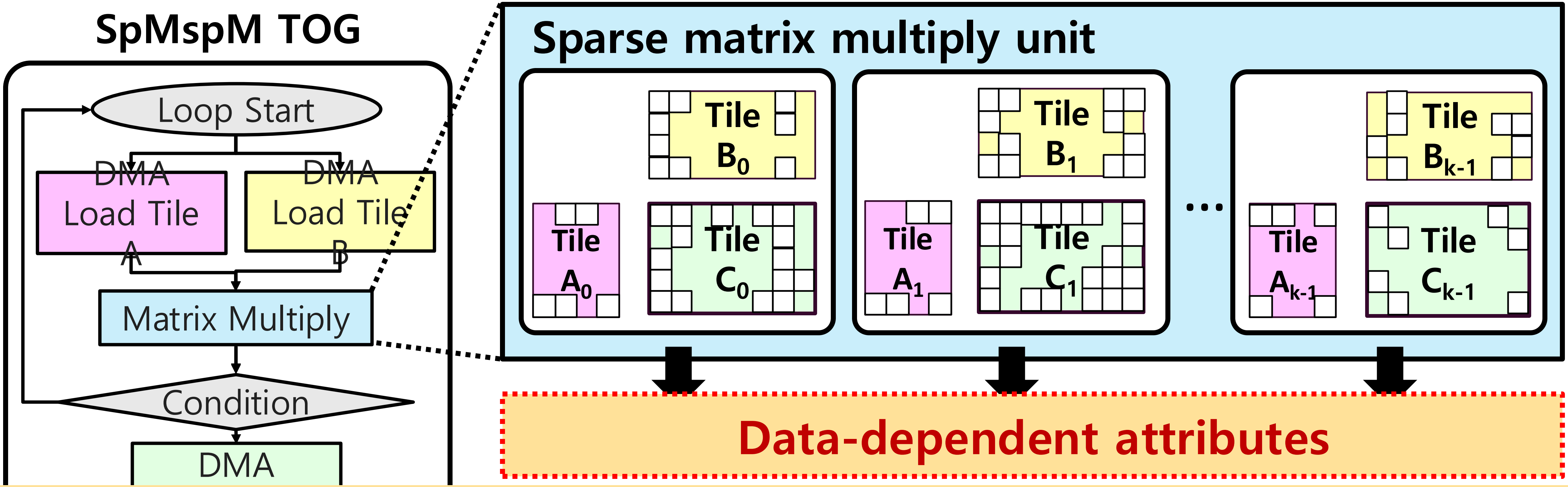
PyTorchSim TLS Compilation Flow



- **Gem5 (timing simulator)** models the compute latency for TLS

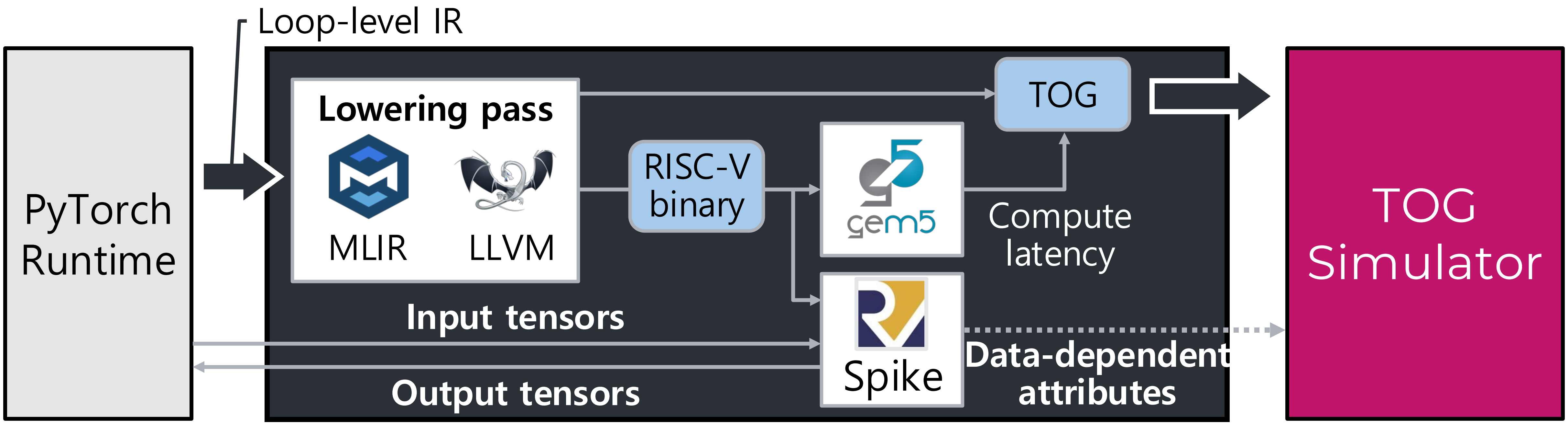
Challenge: Tile operation latency can depend on input data!

Data-Dependent Timing Behavior Example



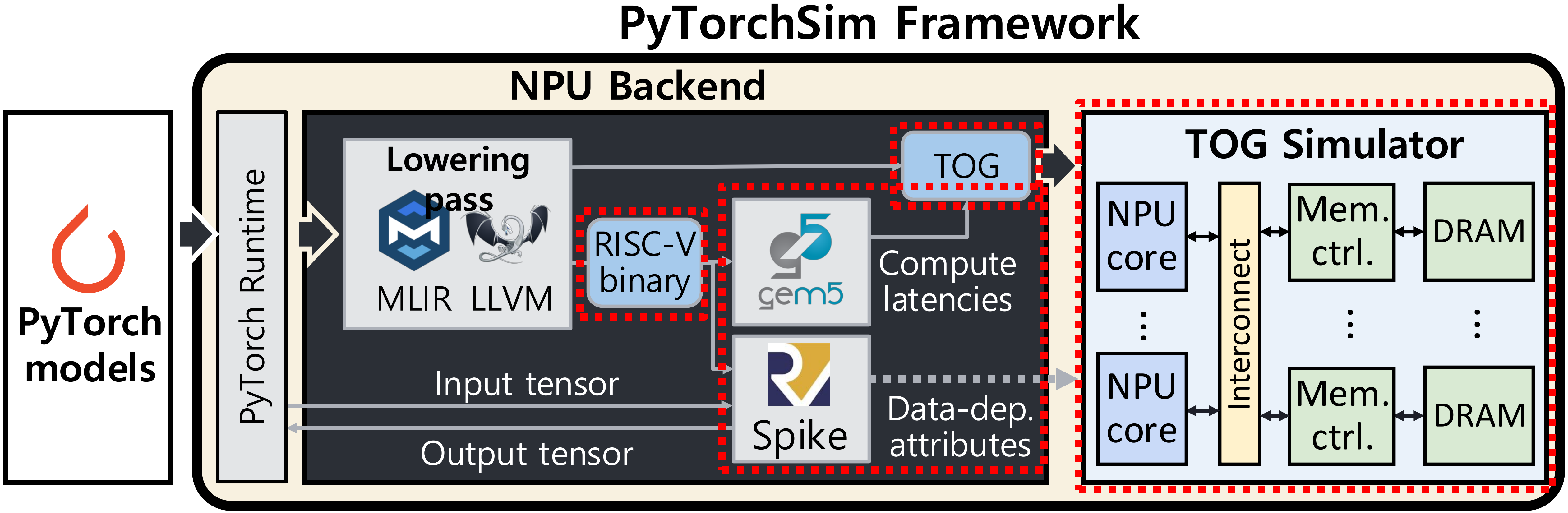
For each specific pair of input tiles,
the compute latency is still deterministic

PyTorchSim TLS Compilation Flow



- **Spike (functional simulator)** executes the binary with PyTorch input:
  - To obtain data-dependent attributes
  - To validate correctness of compiled binary

Putting It All Together



Evaluation Methodology

Accuracy validation

- **Target**: **Real Google TPU v3**
- **Baselines**: SCALE-Sim v3, mNPUSim, Timeloop, MAESTRO

	Cores & clock	Systolic arrays	Vector lanes (# of ALUs)	Scratchpad	DRAM
TPU v3	1 core @ 940 MHz	(128x128) x 2	128 (16 ALUs each)	32 MB	4 HBM2 (960 GB/s)

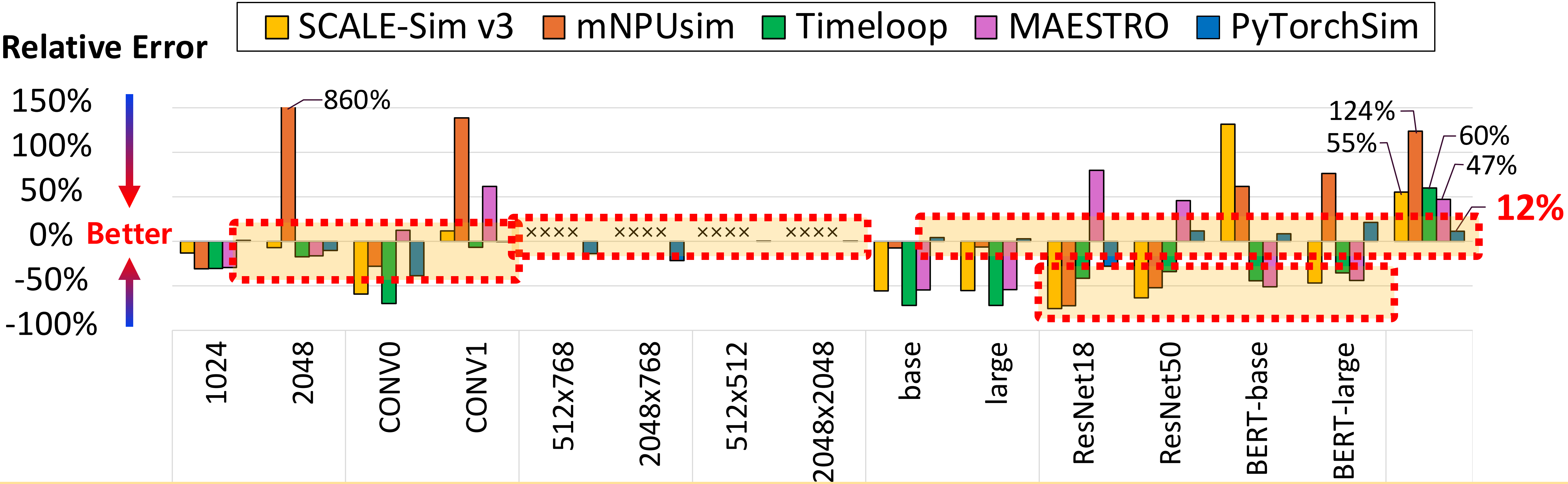
Simulation speed

- **Baselines:** Accel-Sim, mNPUsim
  - We selected Accel-Sim for its rich features and GPUs' dominance in deep learning

Workloads

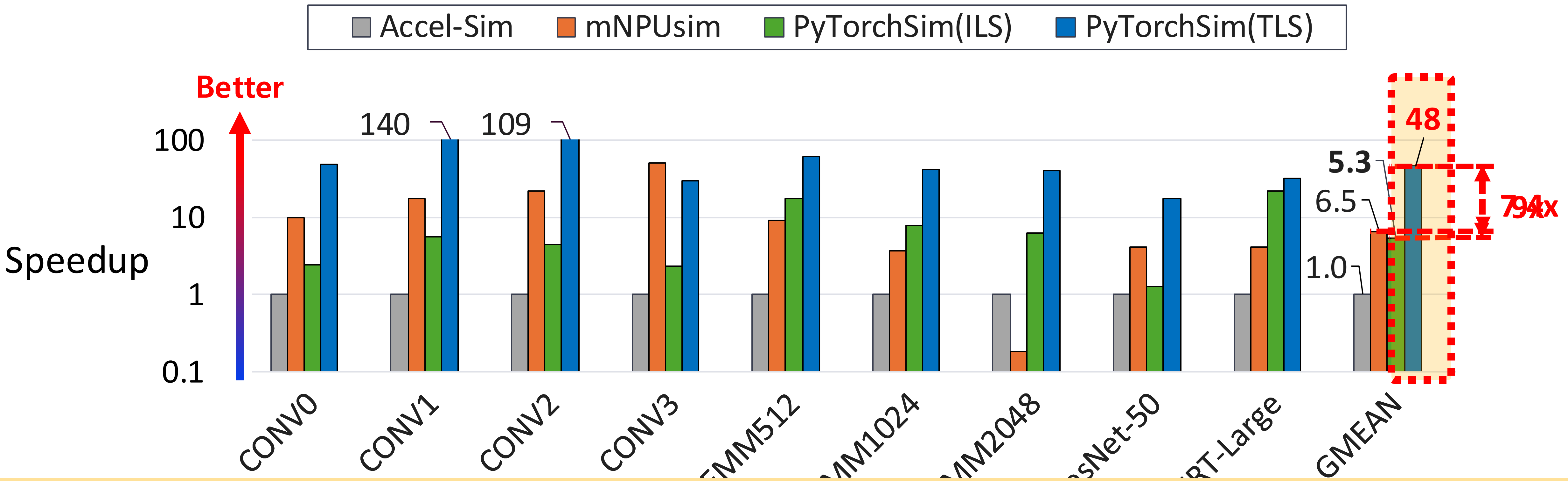
- **Kernels:** GEMM, convolution, layer normalization, softmax, attention
- **Full models:** ResNet18/50, Bert-base/large

Evaluation: Validation against Real TPU v3



PyTorchSim can accurately simulate full models end-to-end

Evaluation: Speedup

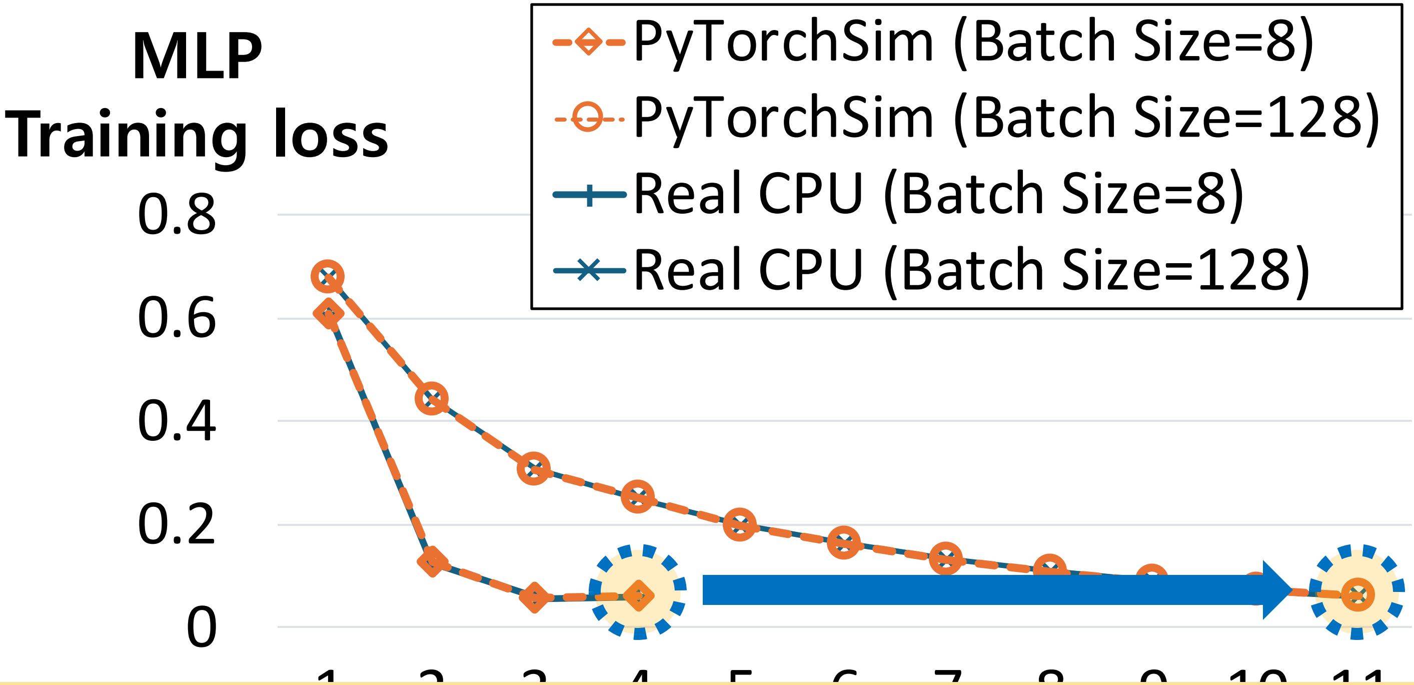


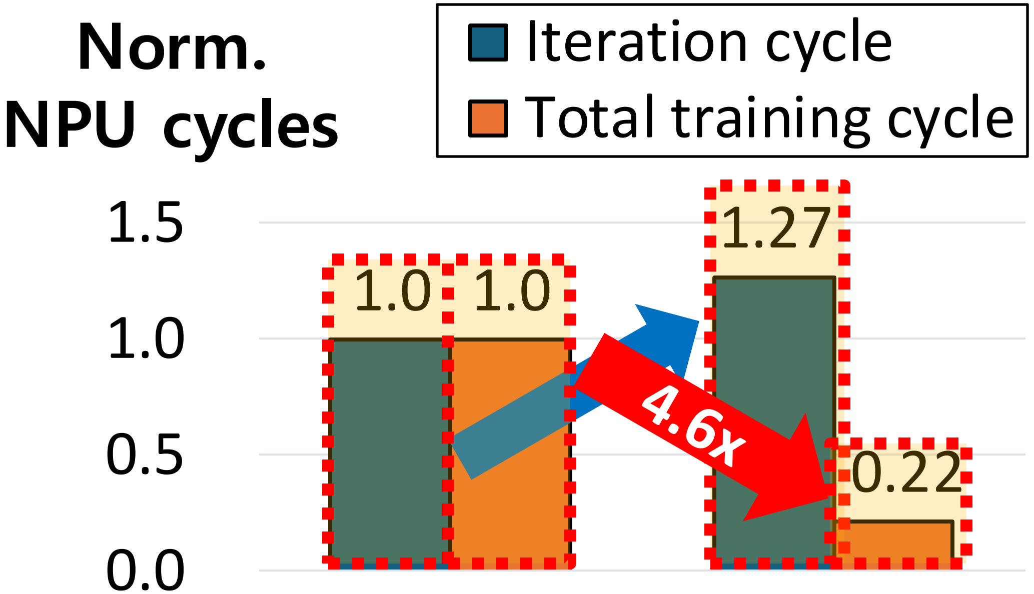
PyTorchSim achieves significant simulation speedup

Contents

- Background / Motivation
- PyTorchSim
  - Overview
  - NPU core modeling
  - Compiler workflow
- Methodology & Evaluation
- **Case Studies (CS)**
  - Impact of DNN training hyperparameter
  - Impact of data placement for chiplet-based NPUs
- Summary

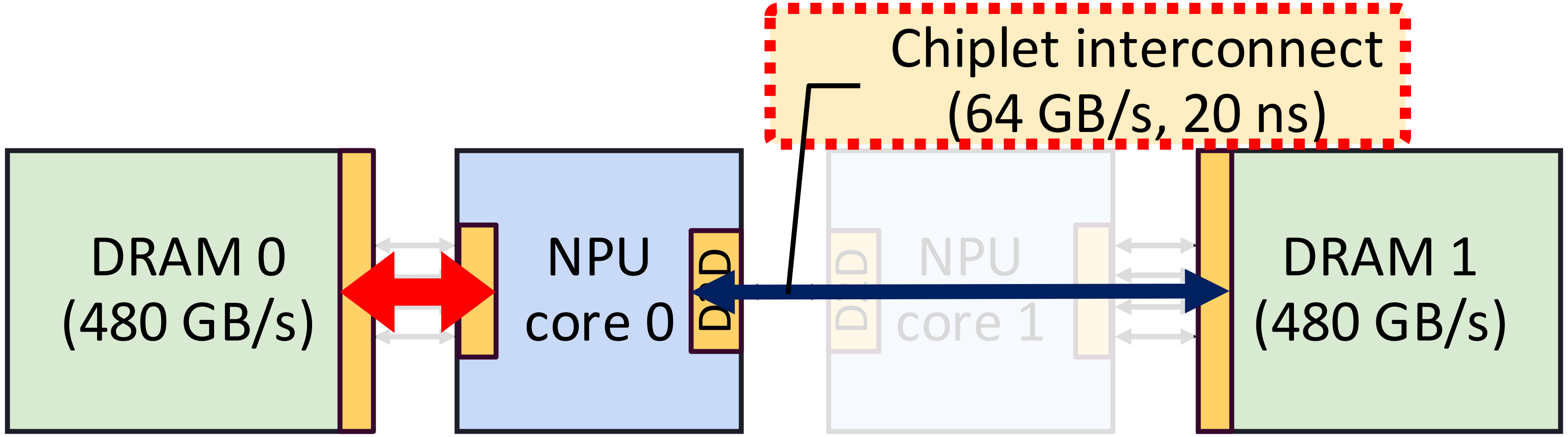
CS1: Impact of DNN Training Hyperparameter





PyTorchSim enables studying training time behaviors of systems

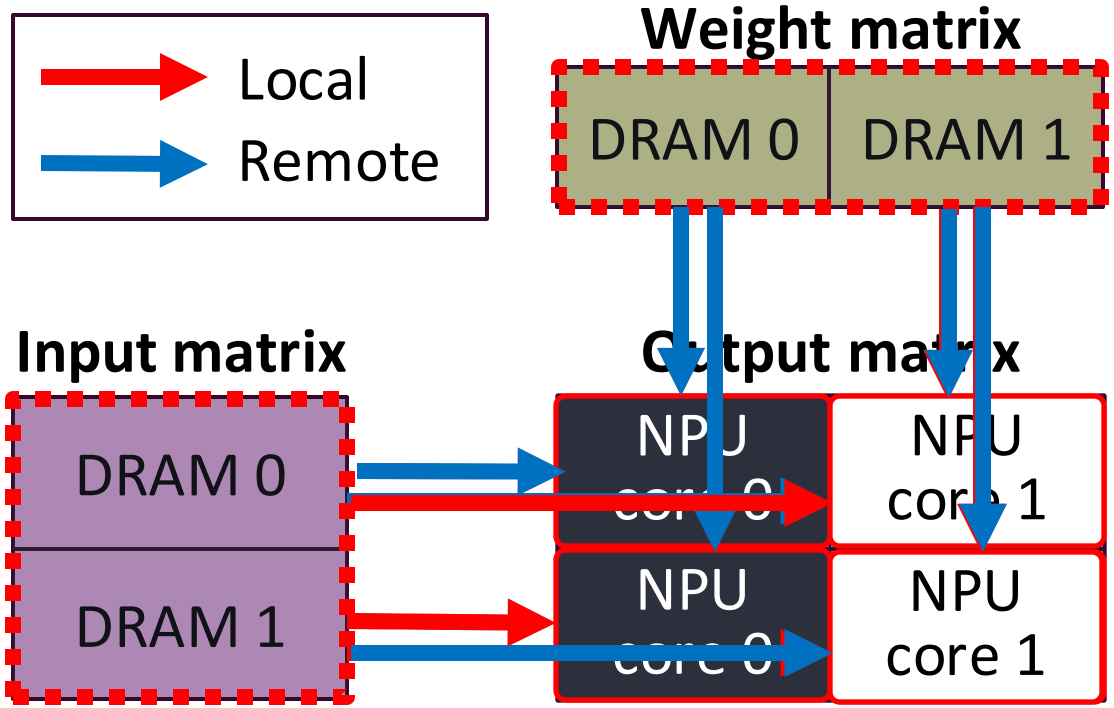
CS2: Impact of Data Placement for Chiplet-based NPUs

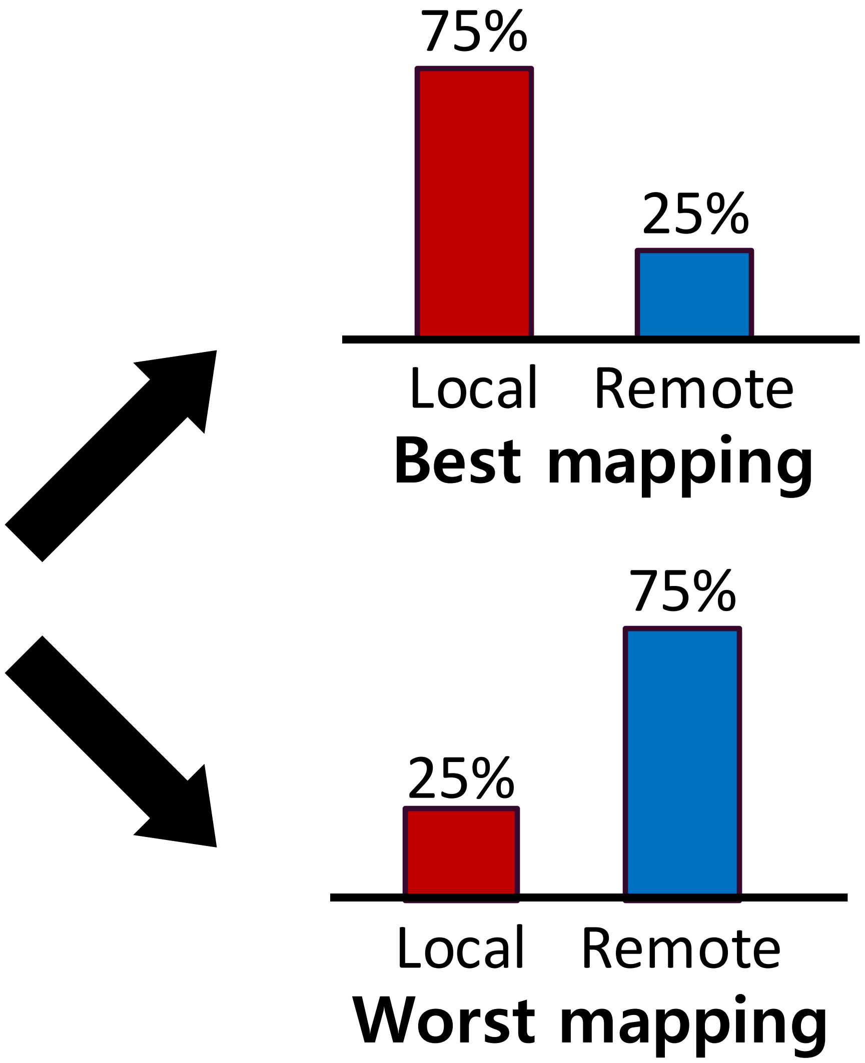


Local traffic
(High bandwidth)

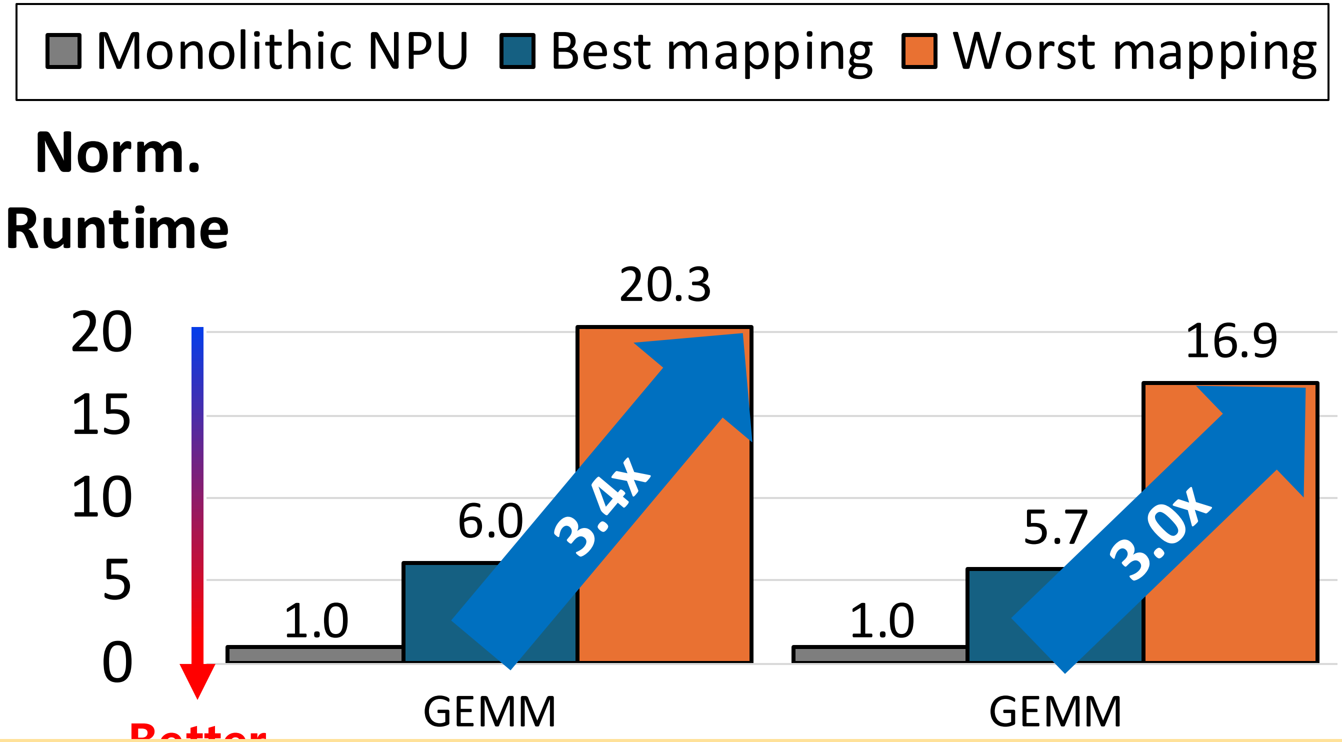
Remote traffic
(Low bandwidth)

CS2: Impact of Data Placement for Chiplet-based NPUs





CS2: Impact of Data Placement for Chiplet-based NPUs



- High remote traffic becomes the severe bottleneck

PyTorchSim enables the study of diverse NPU architectures

More Results and Discussions in the Paper

Case studies

- Heterogeneous dense-sparse NPU
- DNN inference with multi-model tenancy
- Compiler optimization impact

Discussions

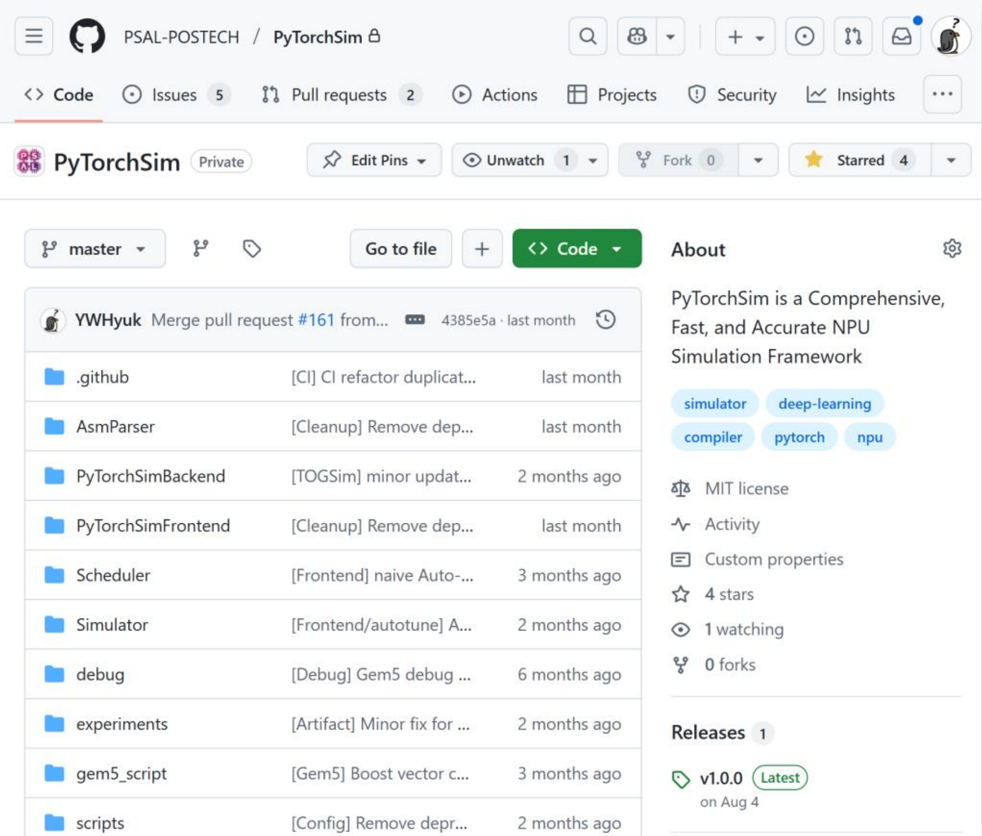
- Extension for other deep learning frameworks
- Extension for modeling GPUs and large-scale systems

Summary

- AI workloads require a comprehensive, fast, and accurate simulator
- To address this, we propose **PyTorchSim**
  - **High-speed, high-accuracy** through our **Tile-Level Simulation (TLS)**
  - **Integration with PyTorch 2 compilation flow**
  - **General and extensible architecture** through a **RISC-V-based NPU ISA**
- Compared to prior NPU simulators, PyTorchSim:
  - Enables various case studies not supported by prior simulators
  - Significantly improves **accuracy** and **simulation speed**

PyTorchSim is Open Source

- Contributions are welcome!



PSAL-POSTECH / PyTorchSim

YWHyuk Merge pull request #161 from... 4385e5a · last month

.github	[CI] CI refactor duplicat...	last month
AsmParser	[Cleanup] Remove dep...	last month
PyTorchSimBackend	[TOGSim] minor updat...	2 months ago
PyTorchSimFrontend	[Cleanup] Remove dep...	last month
Scheduler	[Frontend] naive Auto-...	3 months ago
Simulator	[Frontend/autotune] A...	2 months ago
debug	[Debug] Gem5 debug ...	6 months ago
experiments	[Artifact] Minor fix for ...	2 months ago
gem5_script	[Gem5] Boost vector c...	3 months ago
scripts	[Config] Remove depr...	2 months ago

About

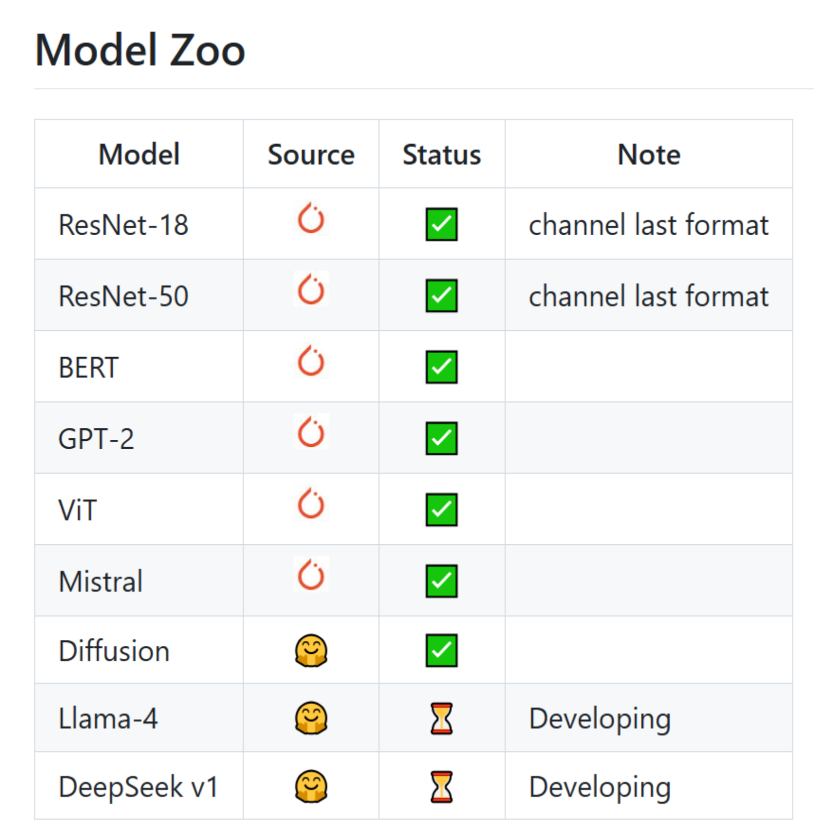
PyTorchSim is a Comprehensive, Fast, and Accurate NPU Simulation Framework

simulator deep-learning compiler pytorch npu

MIT license
Activity
Custom properties
4 stars
1 watching
0 forks

Releases 1

v1.0.0 Latest
on Aug 4



Model Zoo

Model	Source	Status	Note
ResNet-18	🔥	✅	channel last format
ResNet-50	🔥	✅	channel last format
BERT	🔥	✅	
GPT-2	🔥	✅	
ViT	🔥	✅	
Mistral	🔥	✅	
Diffusion	🤗	✅	
Llama-4	🤗	⏳	Developing
DeepSeek v1	🤗	⏳	Developing

GitHub Link