STEEL: Sparsity-Aware Fused Attention for Energy-Efficient Long-Sequence Inference on AMD's XDNA™ NPU

Integrated Systems Laboratory (ETH Zürich)

&

AMD Research and Advanced Development (RAD)

Victor J.B Jung	jungvi@iis.ee.ethz.ch
Gagandeep Singh	gagandeep.singh@amd.com
Joseph Melber	joseph.melber@amd.com
Kristof Denolf	kristof.denolf@amd.com
Francesco Conti	f.conti@unibo.it
Luca Benini	lbenini@iis.ee.ethz.ch

PULP Platform

Open Source Hardware, the way it should be!

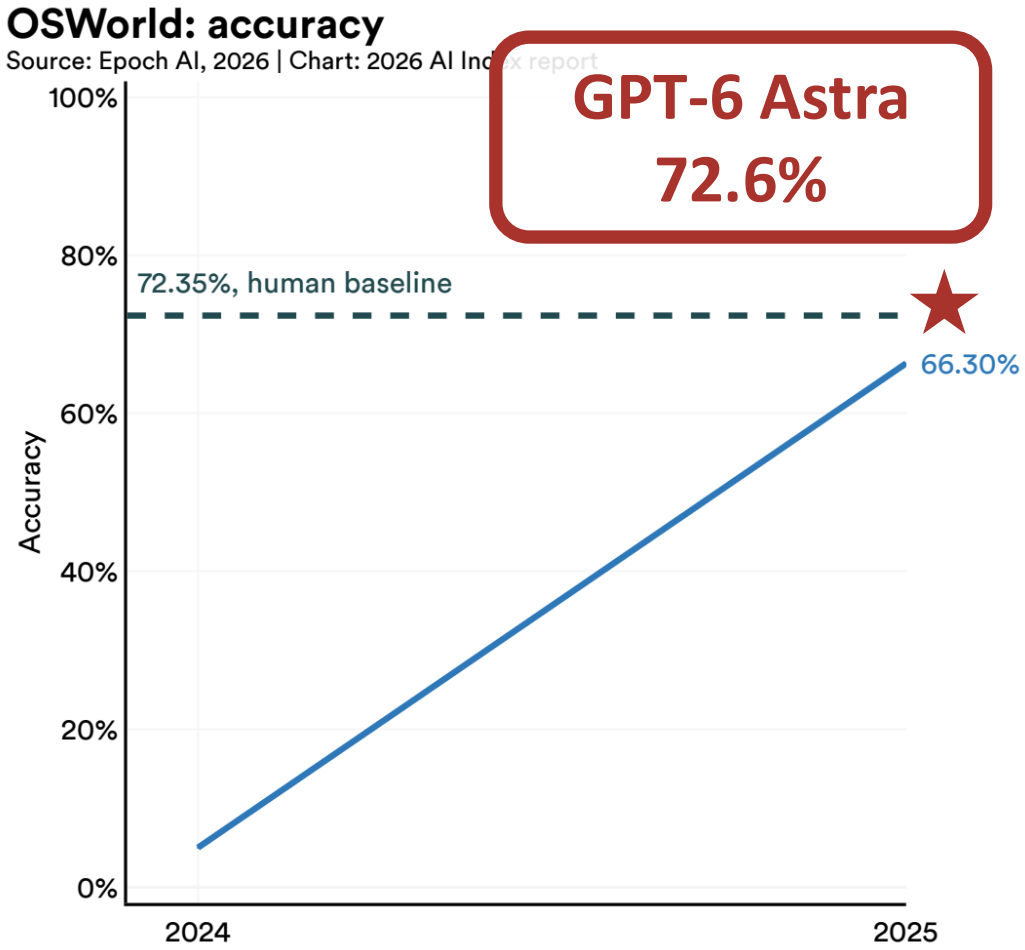
@pulp_platform

pulp-platform.org

youtube.com/pulp_platform

The Agentic Workflow Boom

- **Billion-parameter Transformers now run on laptop-class SoCs**
- **AI Agents are moving into core operating-system workflows [1]**
- **Up to 40% of apps will embed AI Agents [2]**



[1] Stanford HAI, "The 2026 AI Index Report"
[2] Gartner, Inc., "Gartner Predicts 40% of Enterprise Apps Will Feature Task-Specific AI Agents by 2026, Up from Less Than 5% in 2025"

Local Agents are Necessary

- **Local Agents are getting better every year**
- **Executing Agents locally brings:**

Reliability **Privacy**

Local execution is only **viable** if **AI Agents** are **Energy Efficient**

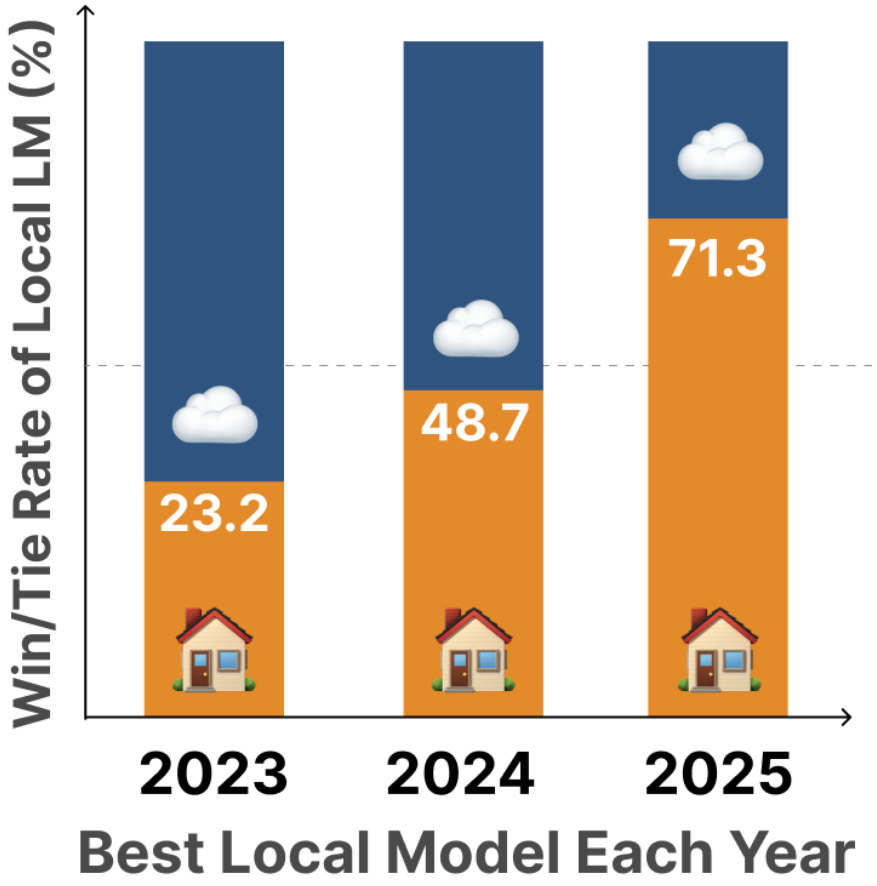


Figure from [1]

[1] Saad-Falcon, Jon, et al. "Intelligence per watt: Measuring intelligence efficiency of local ai."

Attention Dominates Agentic LLM Execution

- **Prefill processes the input prompt and build the context needed for generation**
- **Long contexts prefill execution are important for:**
  - Codebases analysis
  - Document parsing
- **Prefill attention cost grows quadratically with sequence length**
- **On GPU: Many well-optimized FlashAttention kernels exist**

How to map Flash Attention onto specialized hardware?

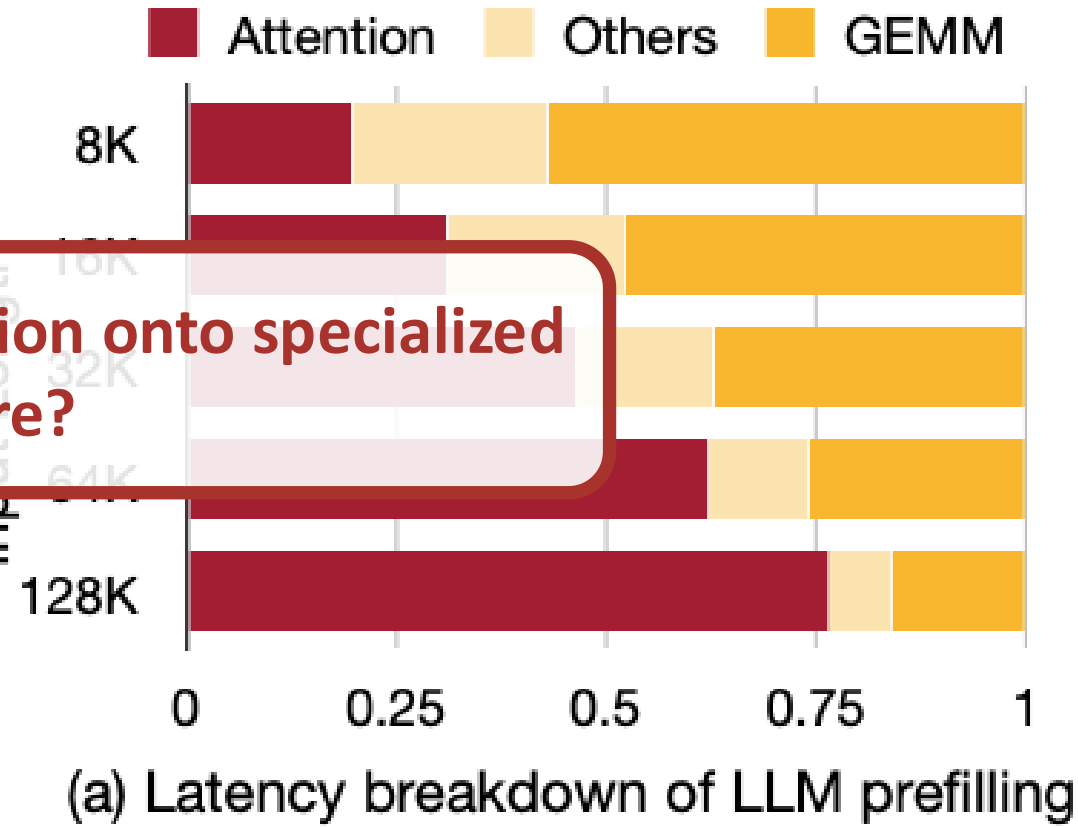


Figure from [1]

[1] Yang, Shang, et al. "Lserve: Efficient long-sequence llm serving with unified sparse attention." MLSys 2025.

NPUs: Energy-Efficient, but Hard to Program

Scratchpad-based spatial dataflow architecture

↓

No cache logic & higher arithmetic unit density

Pros ↓ Higher energy-efficiency

↓ Cons Harder to program

Idea & Contributions

Embedded agentic flow needs efficient prefill attention on NPU!

- First open-source FlashAttention for the XDNA™2 NPU
- Dataflow formulation over three-stage pipeline
- Mask-aware pipeline placement to minimize stalls
- Comparison of STEEL against Zen5 CPU and RDNA 3.5 GPU

38% speedup

9.2x more energy-efficient

1.75x more energy-efficient

Standard Attention

$S_{Q,KV}$: Sequence length of Q or KV
d : Head dimension

$$A = \frac{QK^T}{\sqrt{d}}, P = softmax(A), O = PV$$

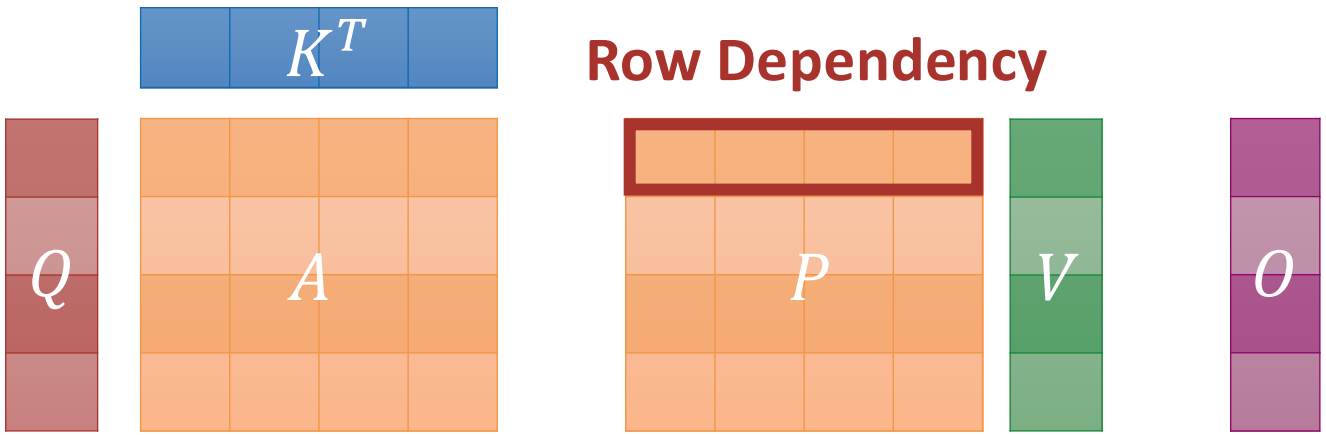
With

$$Q, O \in \mathbb{R}^{S_q \times d}$$
$$K, V \in \mathbb{R}^{S_{kv} \times d}$$
$$A, P \in \mathbb{R}^{S_q \times S_{kv}}$$

Idea: Use online Softmax to tile through the whole attention

$$softmax(x_i) = \frac{e^{x_i - \max(x)}}{\sum_{j=1}^{n} e^{x_j - \max(x)}}$$

$$S_{Q,KV} \gg d$$



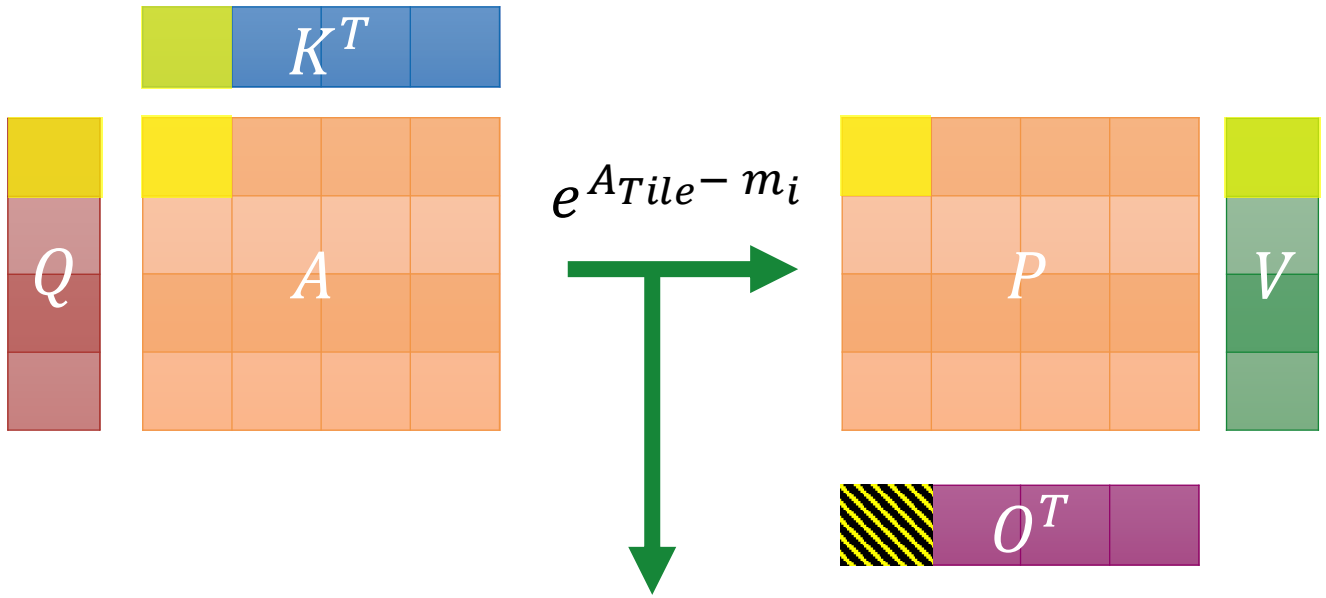
FlashAttention First Iteration

Softmax is simplified

No more **row dependency**

m_i: running max of scores

l_i: running Softmax denominator



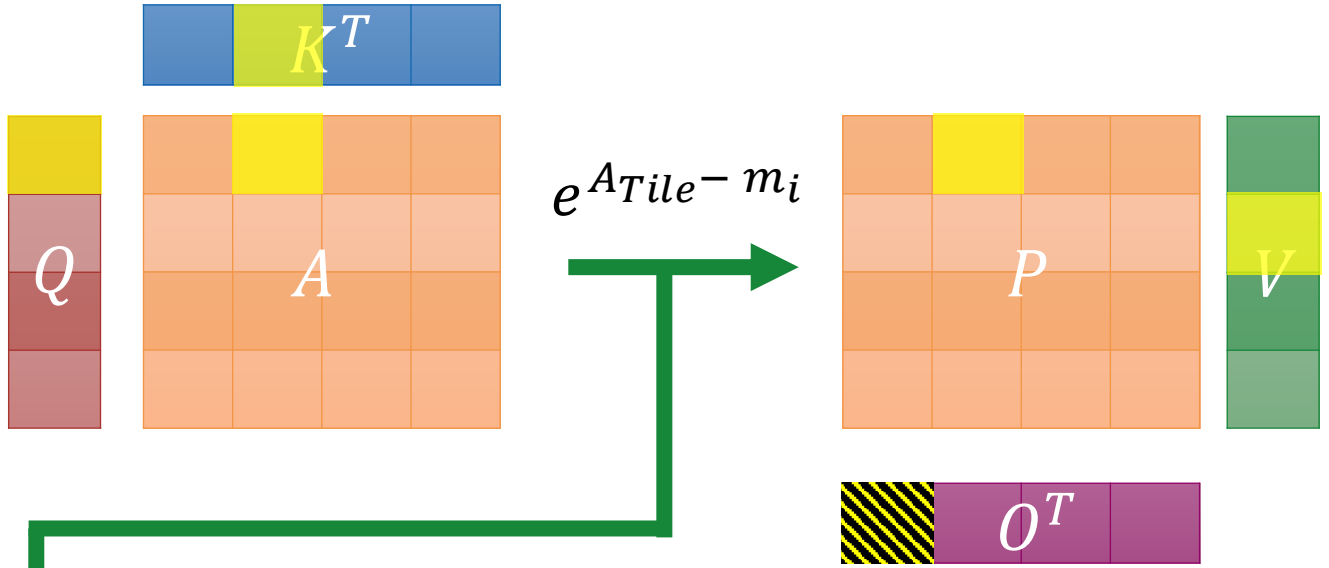
$$m_i = \text{rowmax}(A_{Tile})$$

$$l_i = rowsum(P_{Tile})$$

FlashAttention Recurrency

Update m and l based on the previous value

Last iteration:
Rescale O_{Tile} with l_i^{-1}



$$m_i = \max(m_{i-1}, \text{rowmax}(A_{Tile}))$$
$$l_i = e^{m_{i-1}-m_i} l_{i-1} + rowsum(P_{Tile})$$

$e^{m_{i-1}-m_i}$

The XDNA™2 NPU

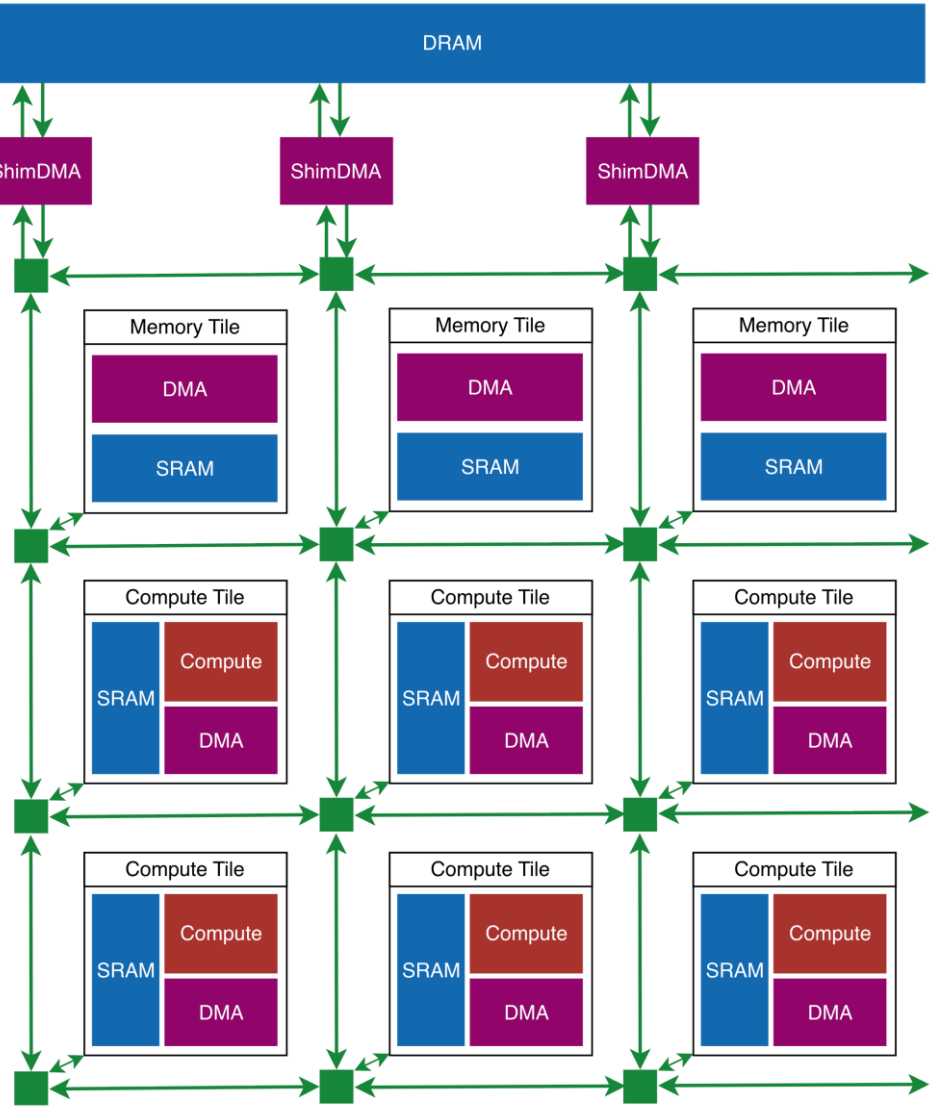
Composed of 3 different tiles

Shim Tile

Memory Tile

Compute Tile

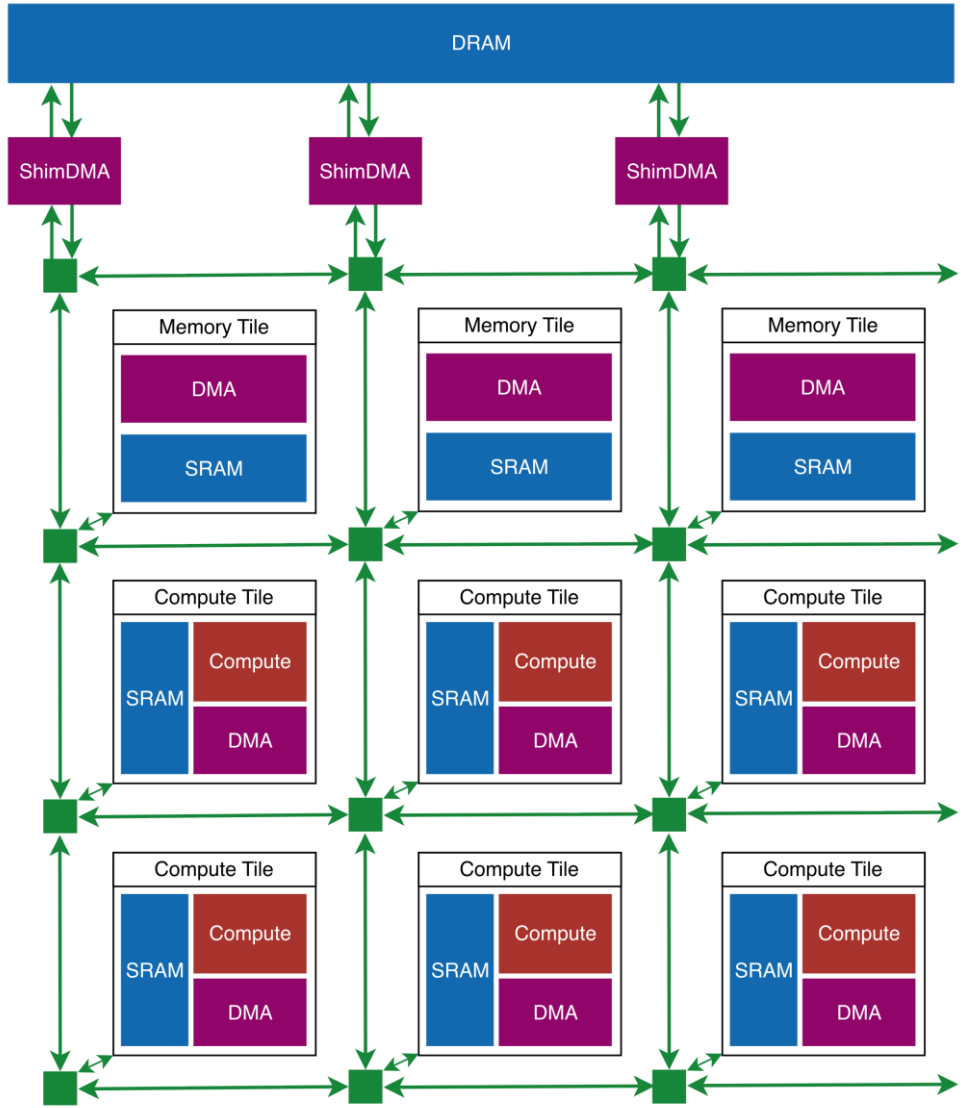
Interconnected via a NoC



The XDNA™2 NPU

Shim Tile

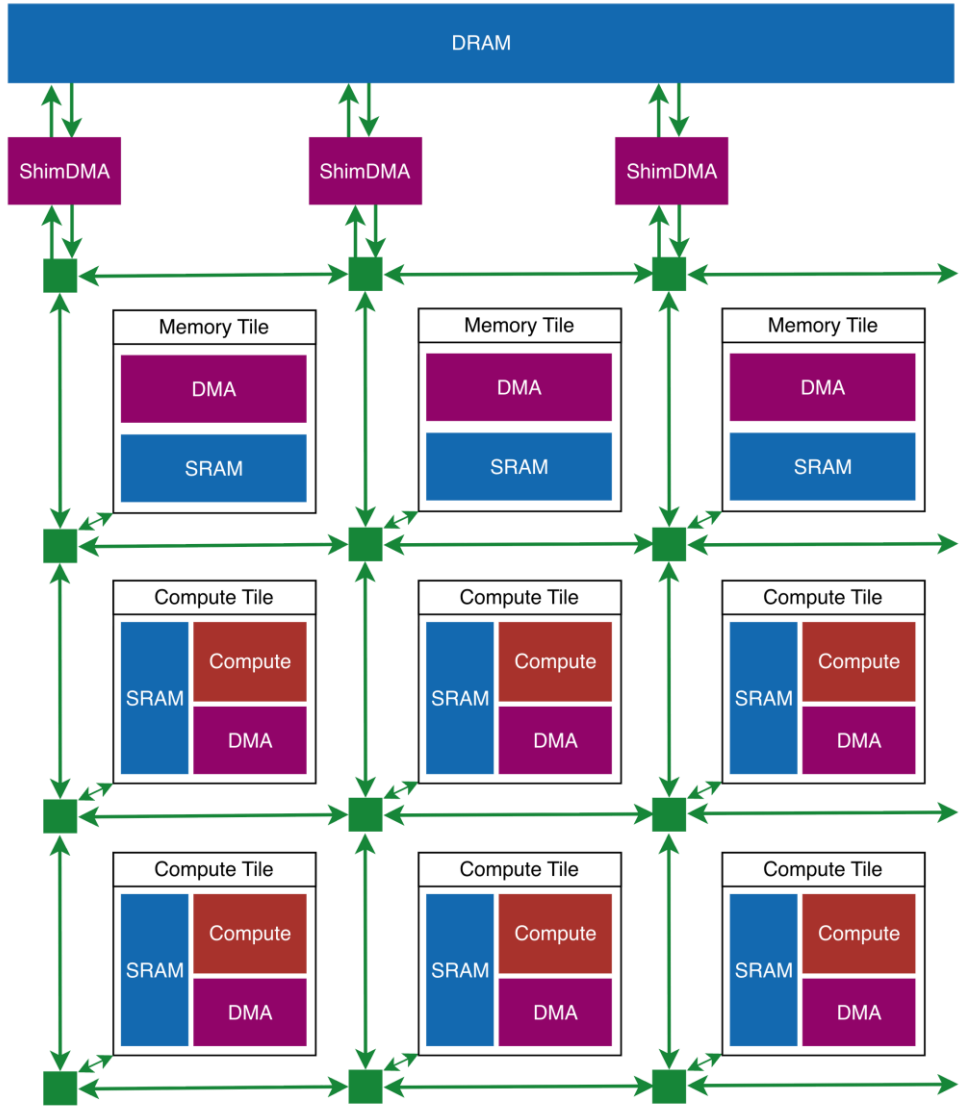
- Interface to off-chip DRAM
- DMA supporting 4D layout transformation



The XDNA™2 NPU

Memory Tile

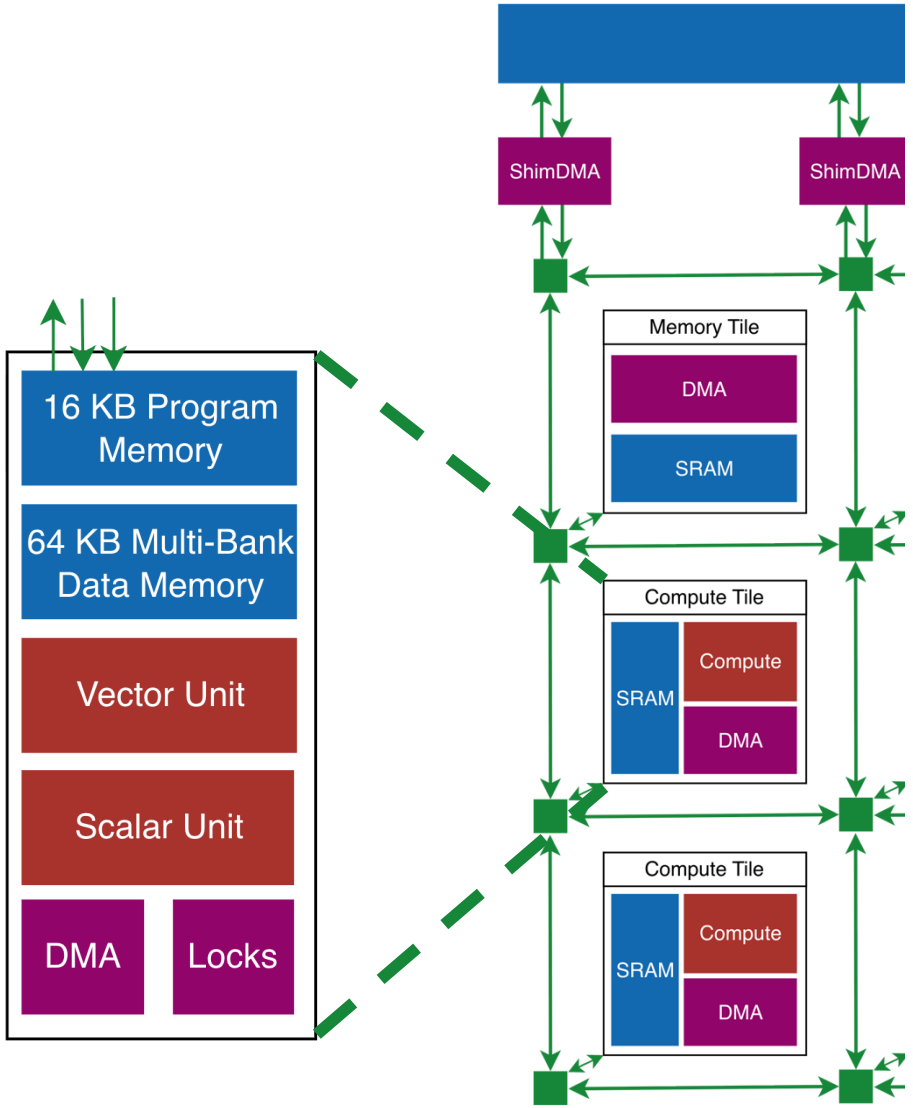
- 512 KB scratchpad memory (L2)
- DMA supporting 4D layout transformation



The XDNA™2 NPU

Compute Tile

- Slow scalar datapath
  - Used for control code handling
- Efficient VLIW vector processor
  - 64 bfloat16 MAC/Cycle
- 64 KB scratchpad memory (L1)
- 16 KB program memory
- DMA supporting 3D layout transformation

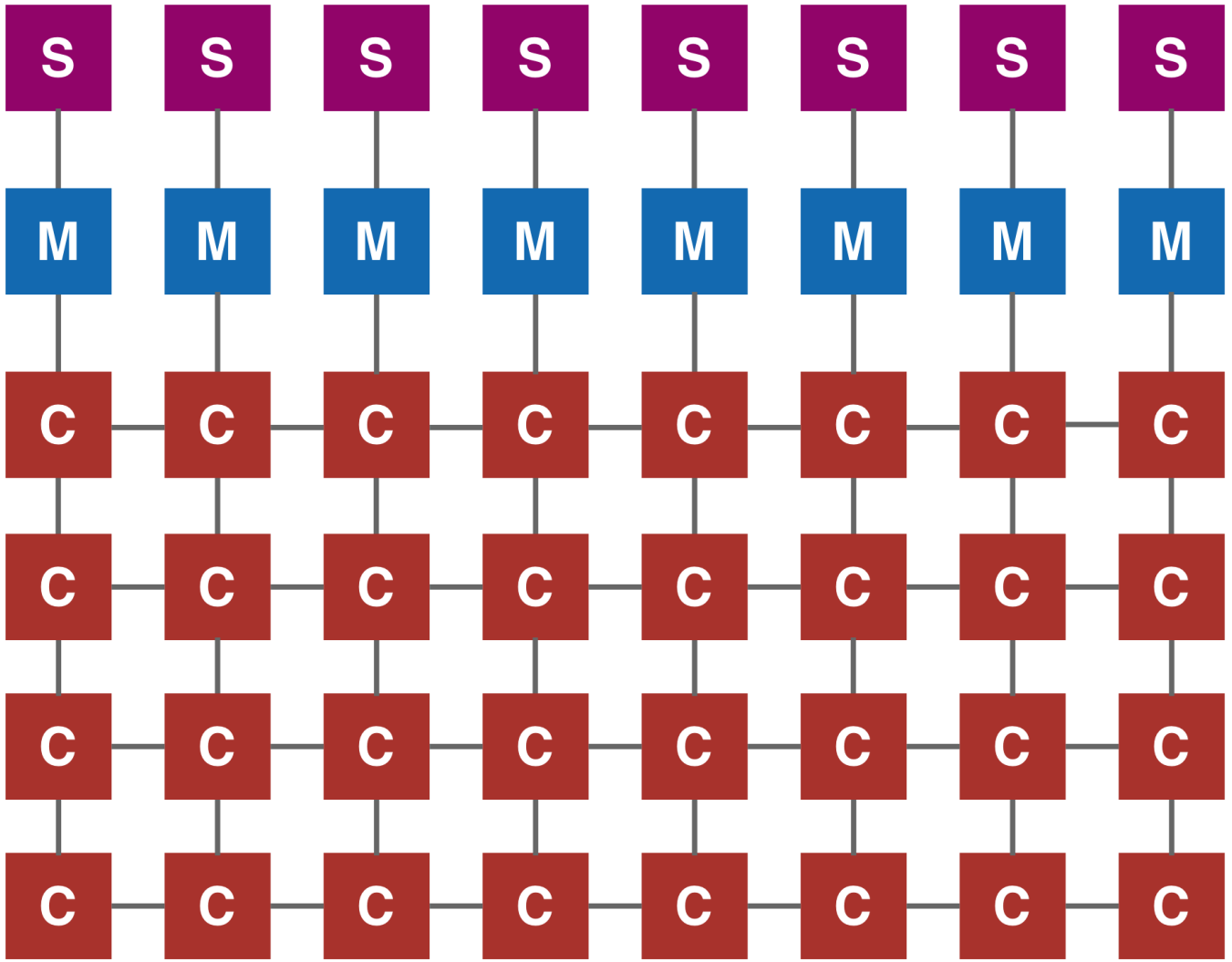


The XDNA™2 NPU Zoomed Out

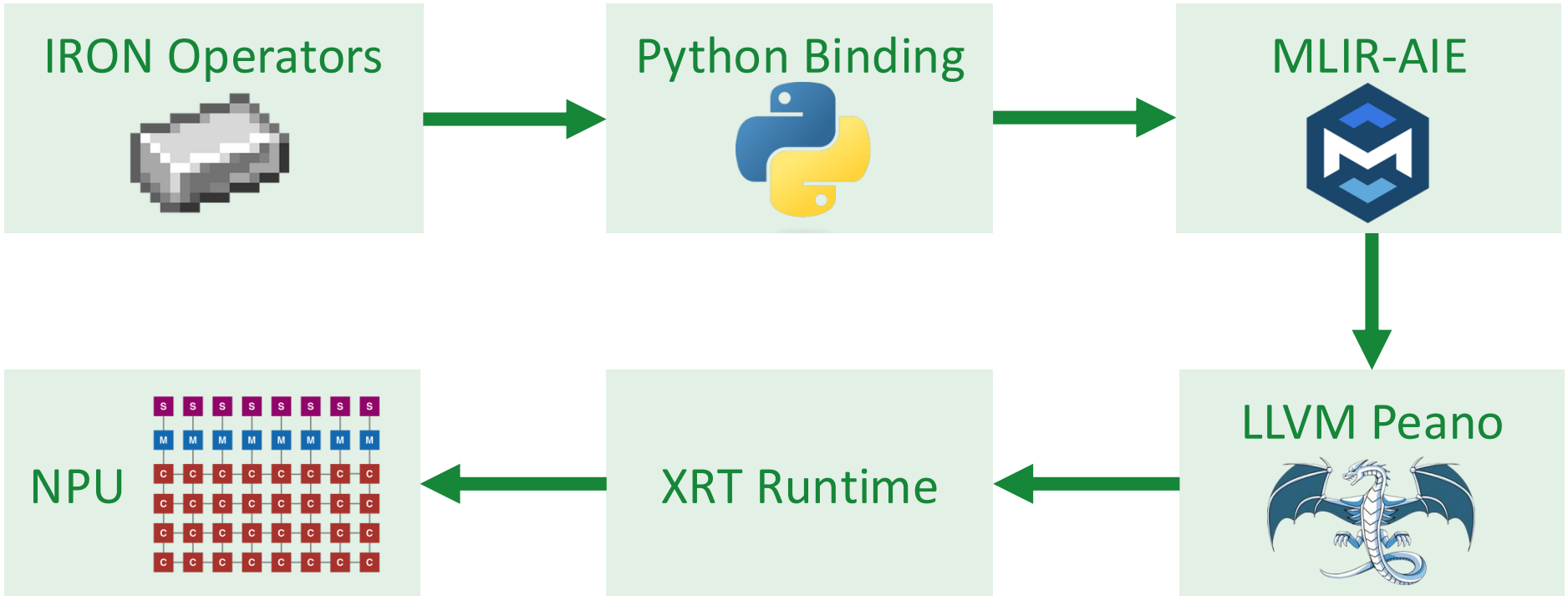
XDNA™2 has 8 columns

Each column has:

- 4 **Compute Tiles** C
- 1 **Memory Tile** M
- 1 **Shim Tile** S



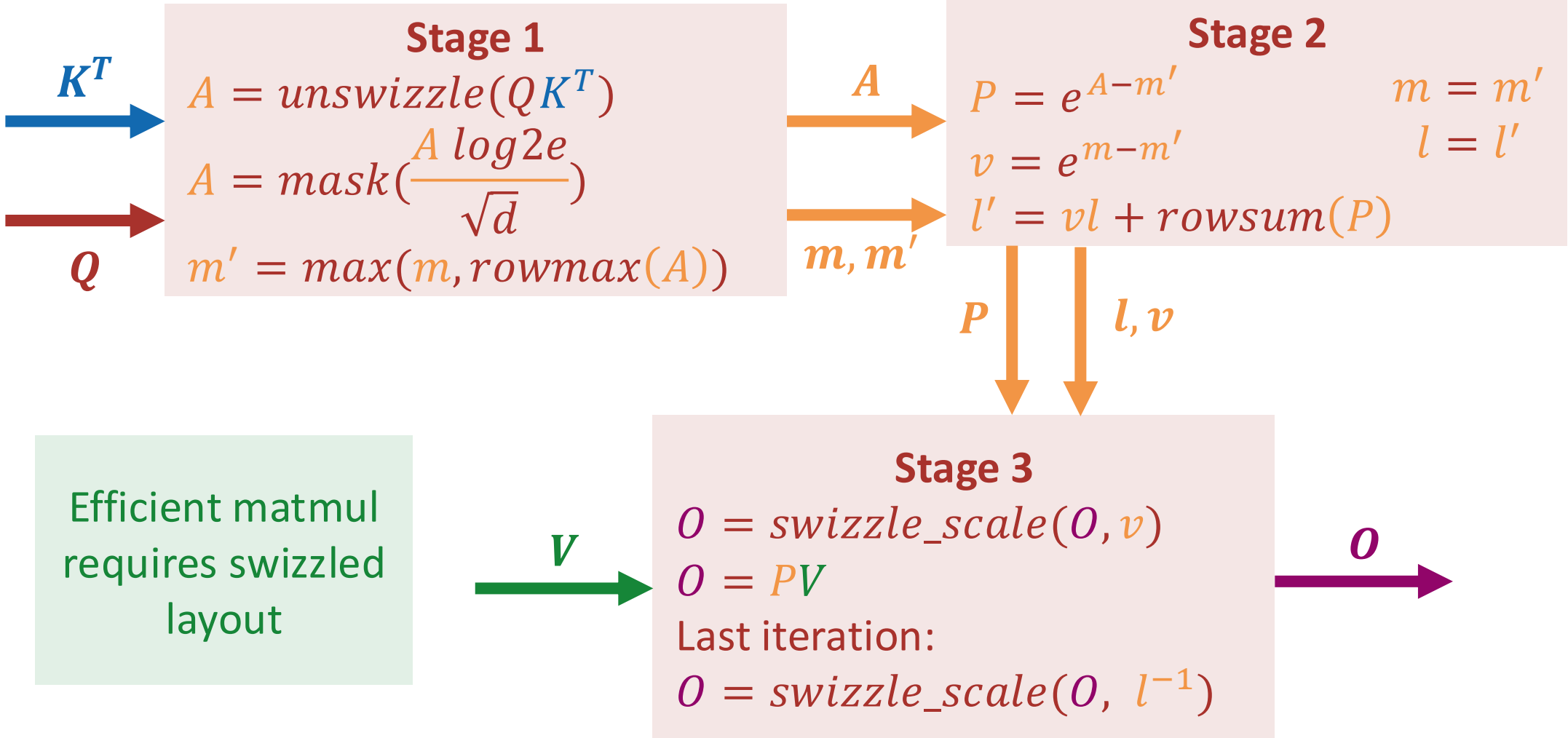
The XDNA2 Programming Stack



github.com/amd/iron

github.com/xilinx/mlir-aie

The STEEL Pipeline

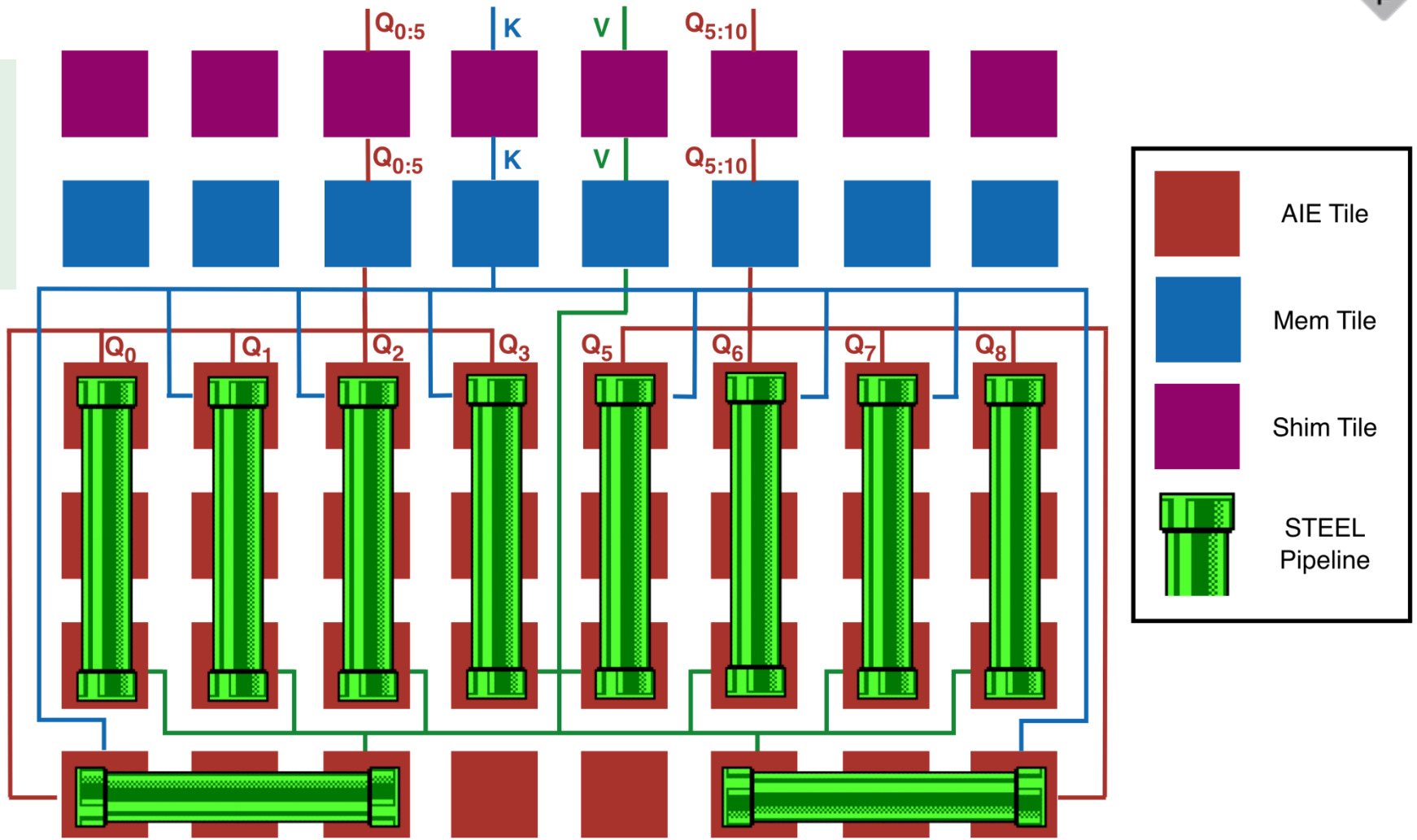


Mapping Pipelines onto the PE Array

We place 10 pipelines onto the PE array

MemTile has **6** I/O ports

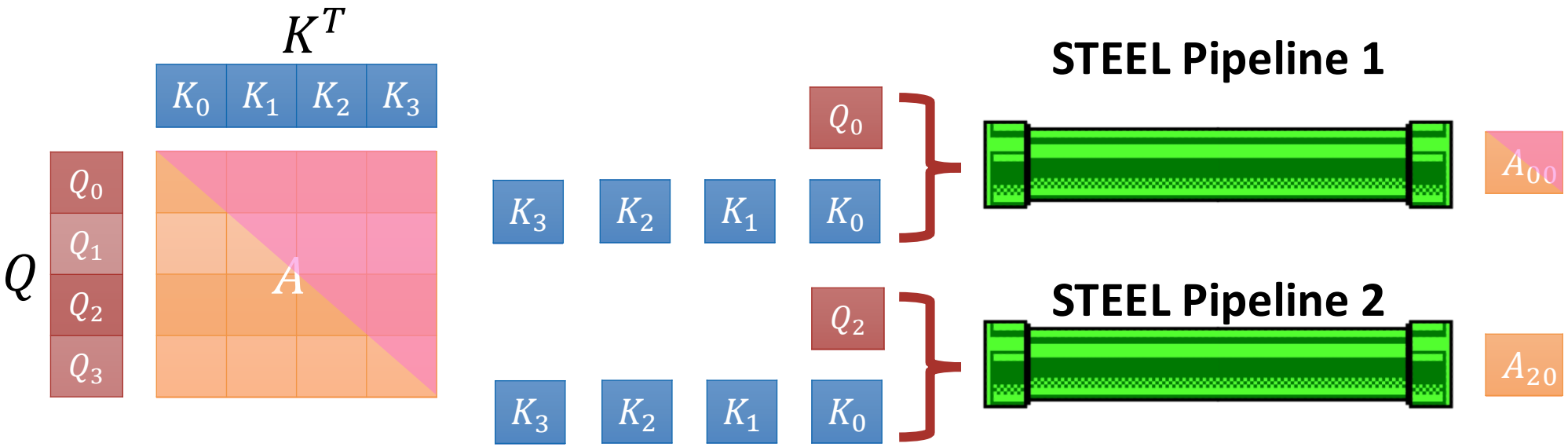
42/48 MemTile ports are used



The Causal Mask Breaks the Balance

Causal Mask:

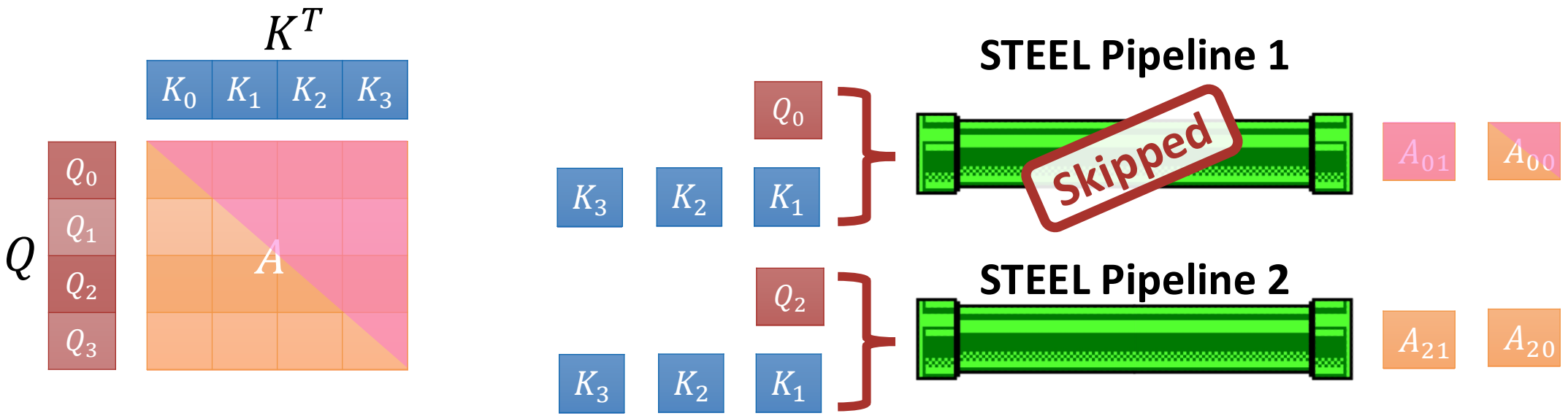
LLM causal mask covers half of A -> Some blocks are skipped



The Causal Mask Breaks the Balance

Causal Mask:

LLM causal mask covers half of A -> Some blocks are skipped

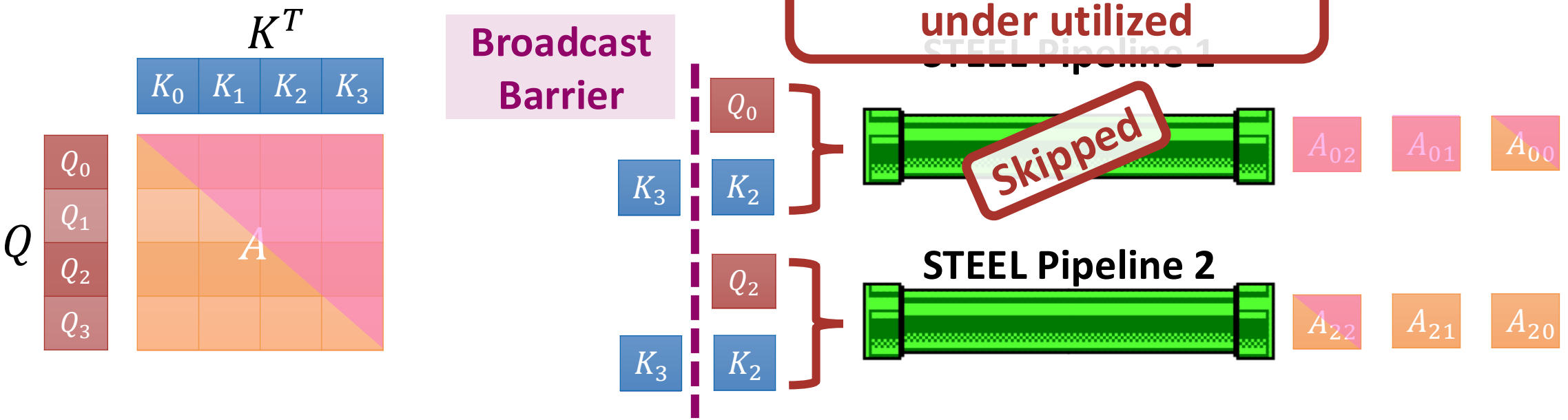


Consequence: Each STEEL pipeline get a different amount of work

The Causal Mask Breaks the Balance

Causal Mask:

LLM causal mask covers half of A => Some blocks are skipped

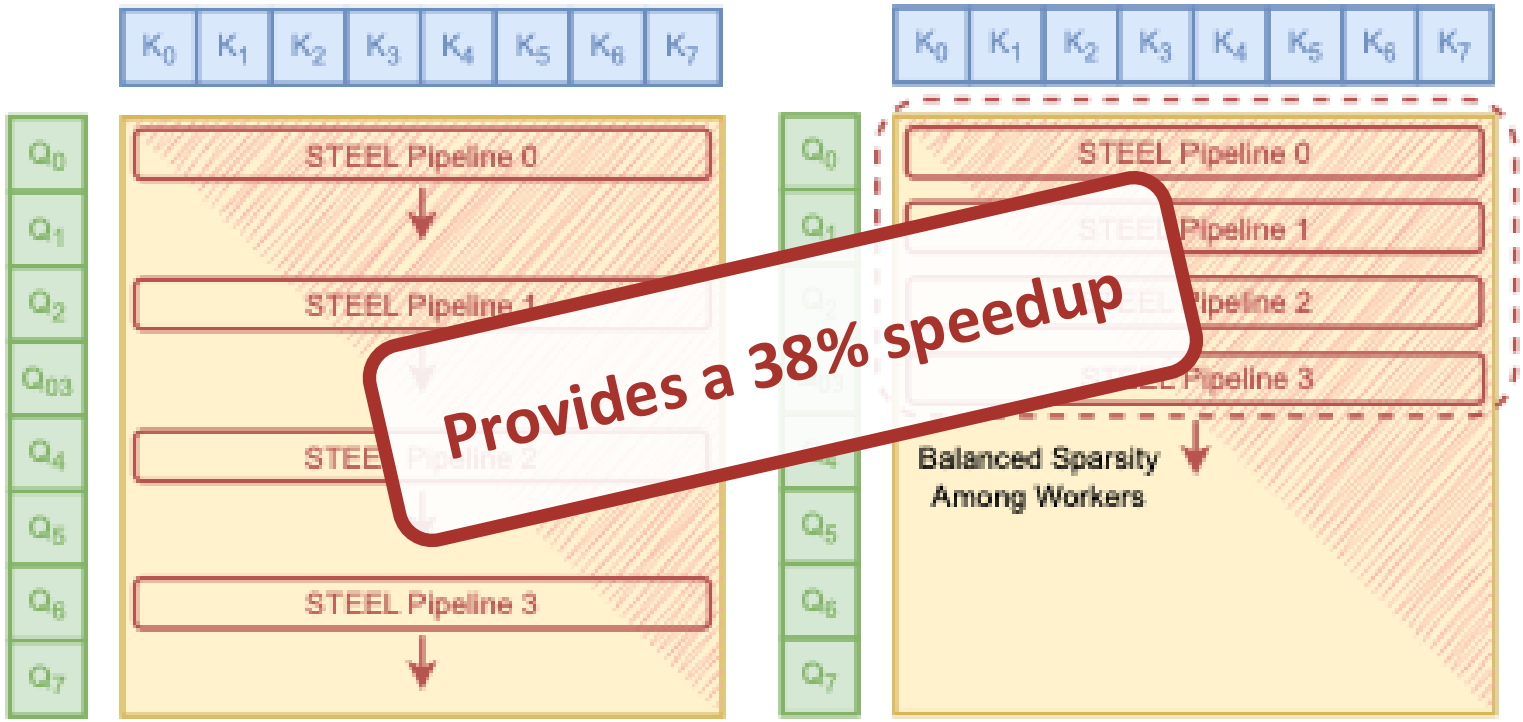


Some STEEL Pipelines are under utilized

Consequence: Each STEEL pipeline get a different amount of work

Mask-Aware Pipeline Placement

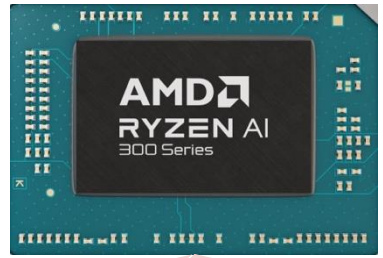
Solution: Distribute consecutive rows across pipelines

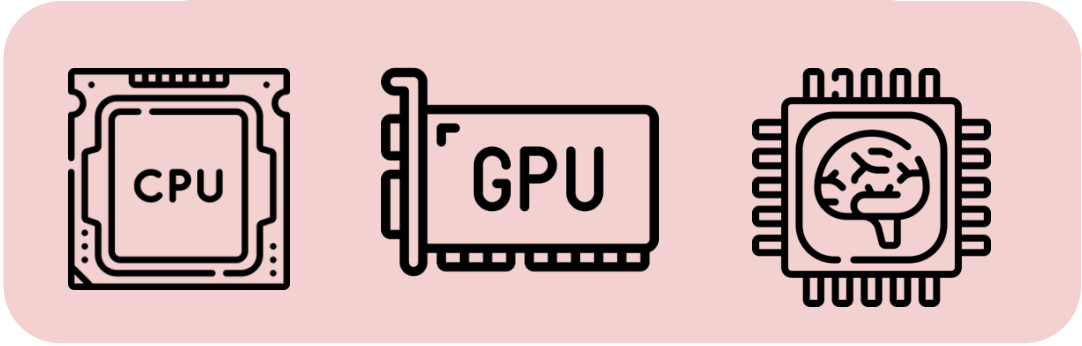


Evaluation Setup

Hardware Setup

- **AMD Ryzen™ AI 9 HX 370**
- 16 Zen5 CPUs
- RDNA 3.5 GPU
- XDNA™2 NPU
- Present on mini-PCs and laptops





Software Setup

- PyTorch 2.1 + ROCm 6.4
- TorchLib C++ API for testbench
- FlashAttention Triton GPU kernel
- 25 warm-up iterations
- 50 measured iterations

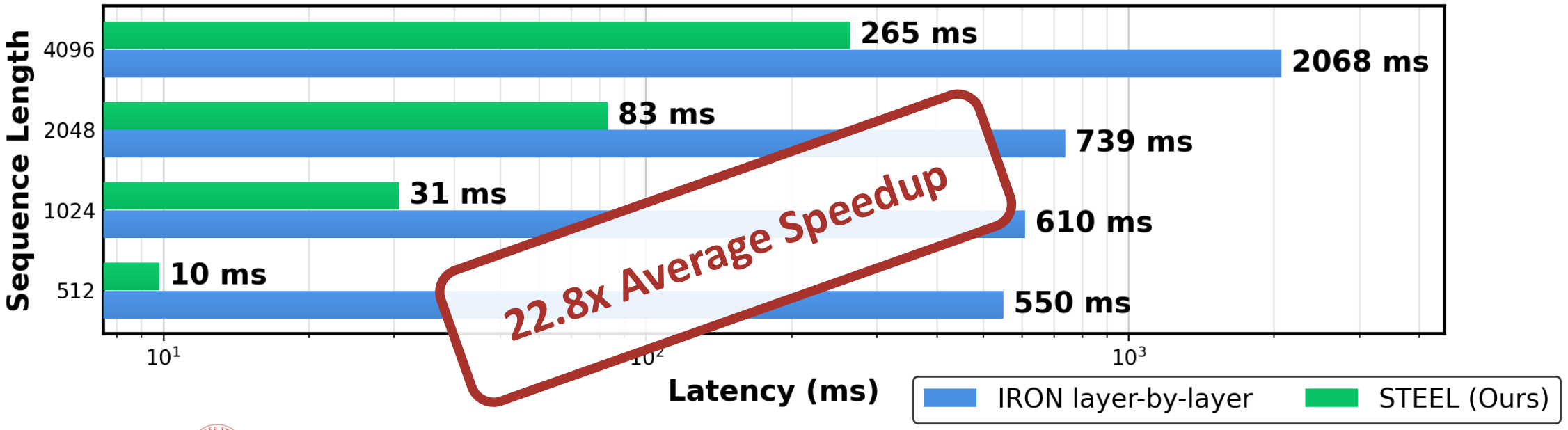
STEEL vs Layer-by-Layer Attention

Attention config from BERT

- **IRON baseline: Separate GEMM, Softmax, and Scale kernels**
- **Fusion benefits:**

Less context switches from kernel load

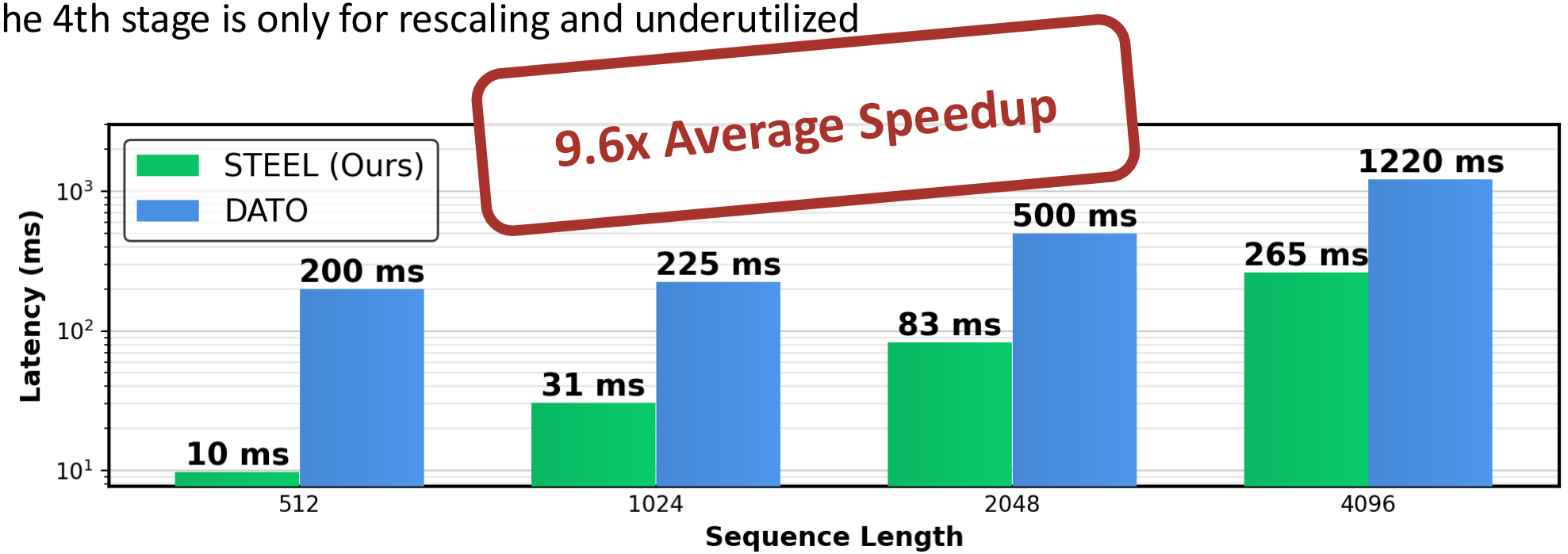
Reduce off-chip data movement



STEEL vs State of the Art (DATO) on XDNA™1

- **DATO [1] -> SotA baseline targeting XDNA™1**
- **We port STEEL to XDNA™1 to compare**
- **DATO has a 4-stages FlashAttention pipeline**
  - The 4th stage is only for rescaling and underutilized

STEEL is portable across NPU generations



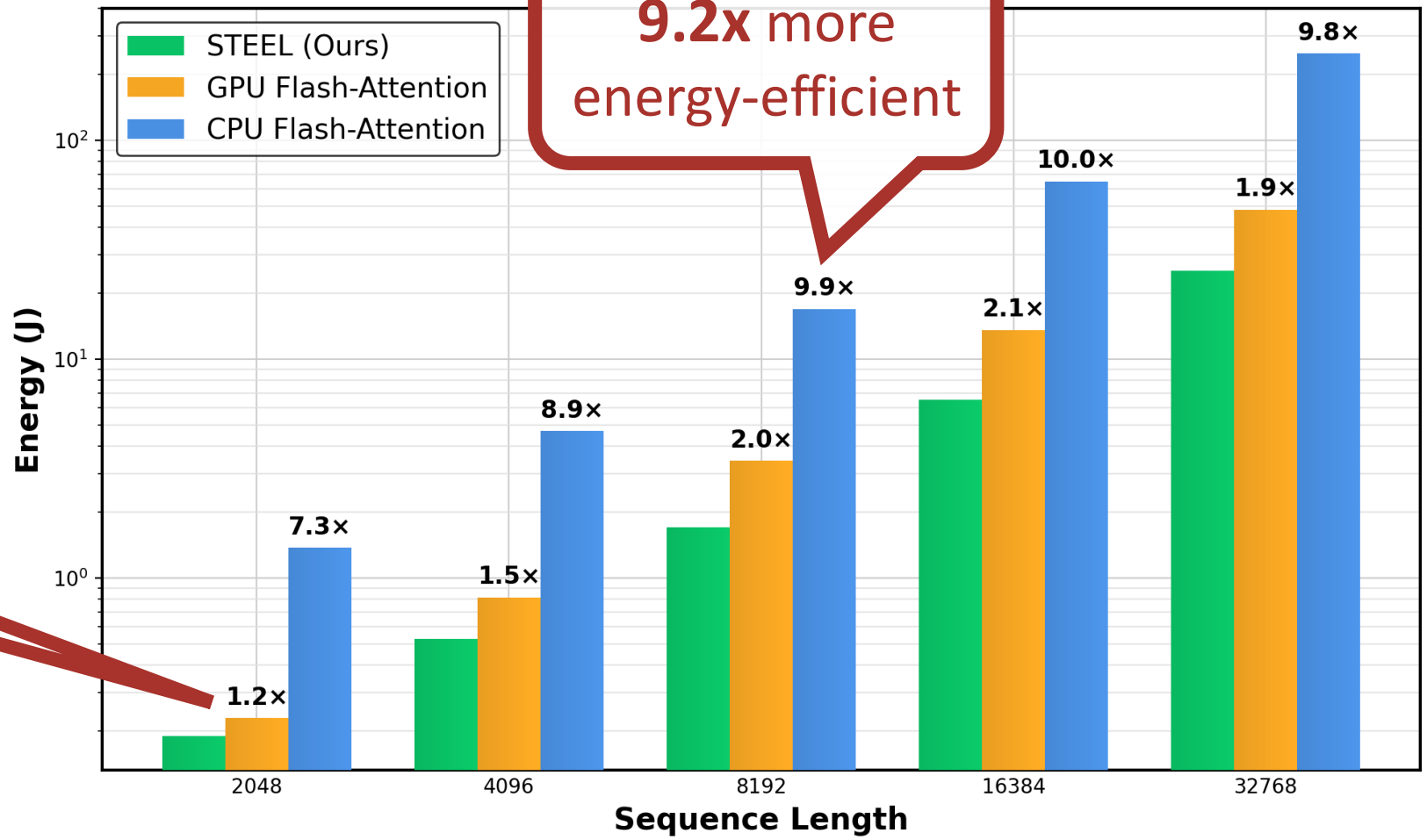
[1] Fang, Shihan, et al. "Dato: A task-based programming model for dataflow accelerators." arXiv:2509.06794 (2025).

Energy: XDNA™2 NPU *vs* RDNA 3.5 GPU *vs* Zen5 CPU

Attention config from Llama3.1-1B

1.75x more energy-efficient

9.2x more energy-efficient



Conclusion & Take-Home Message

We demonstrated that **NPUs can push energy-efficiency of agentic AI at the edge**

But we need **careful mapping decisions** to see benefits against GPUs

`github.com/amd/iron`

`github.com/xilinx/mlir-aie`

Future work: Can we automatically explore and evaluate these mapping decisions?

PULP Platform

Open Source Hardware, the way it should be!

Victor J.B. Jung jungvi@iis.ee.ethz.ch
Gagandeep Singh gagandeep.singh@amd.com
Joseph Melber joseph.melber@amd.com
Kristof Denolf kristof.denolf@amd.com
Francesco Conti f.conti@unibo.it
Luca Benini lbenini@iis.ee.ethz.ch

Institut für Integrierte Systeme – ETH Zürich
Gloriastrasse 35
Zürich, Switzerland

DEI – Università di Bologna
Viale del Risorgimento 2
Bologna, Italy

AMD RAD
3100 Logic Drive
Longmont, CO, United States

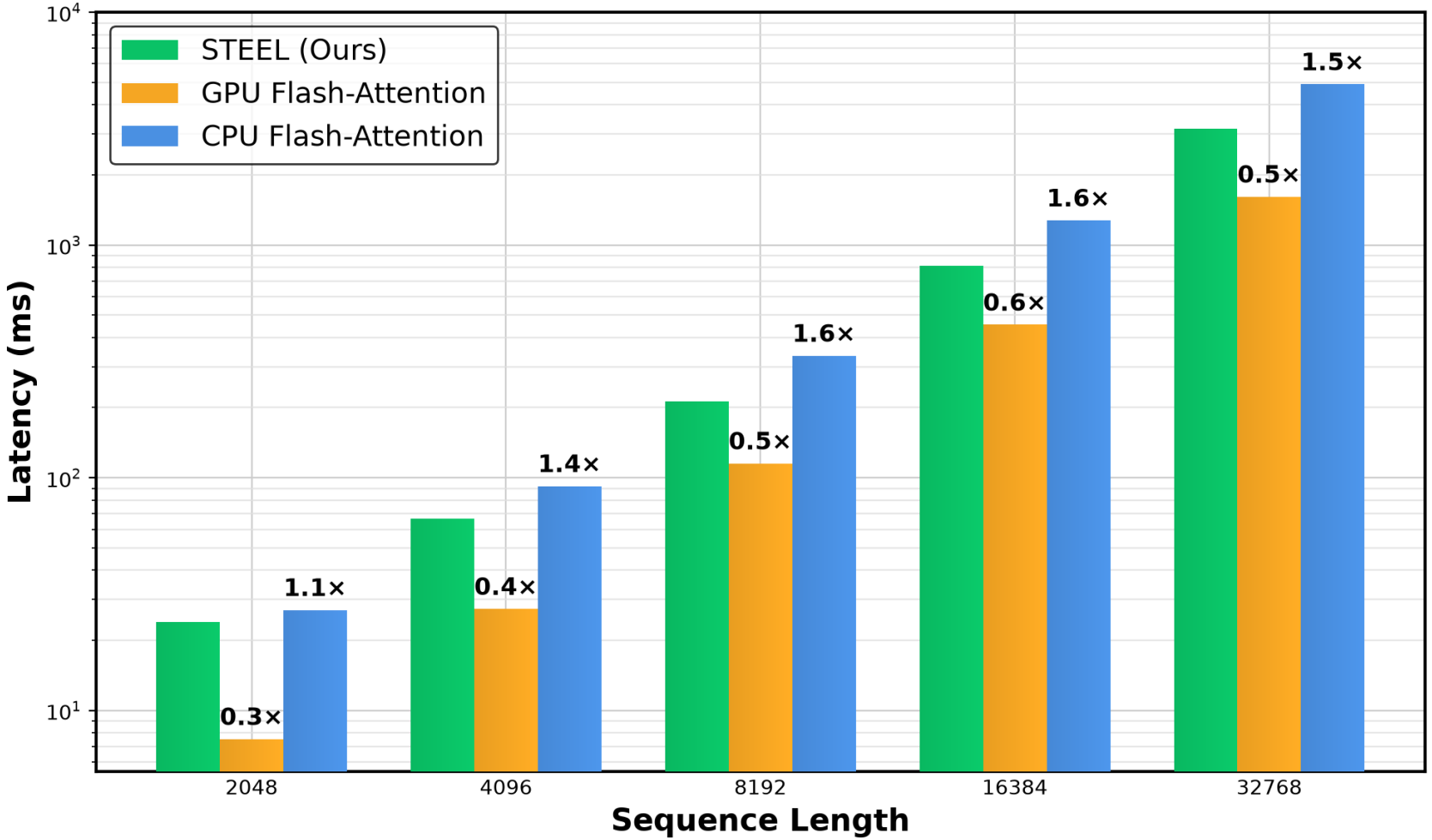
@pulp_platform
pulp-platform.org
youtube.com/pulp_platform

Appendix: Latency: XDNA™2 NPU *vs* RDNA 3.5 *vs* Zen5

RDNA 3.5
24 FP16 TFLOP/s

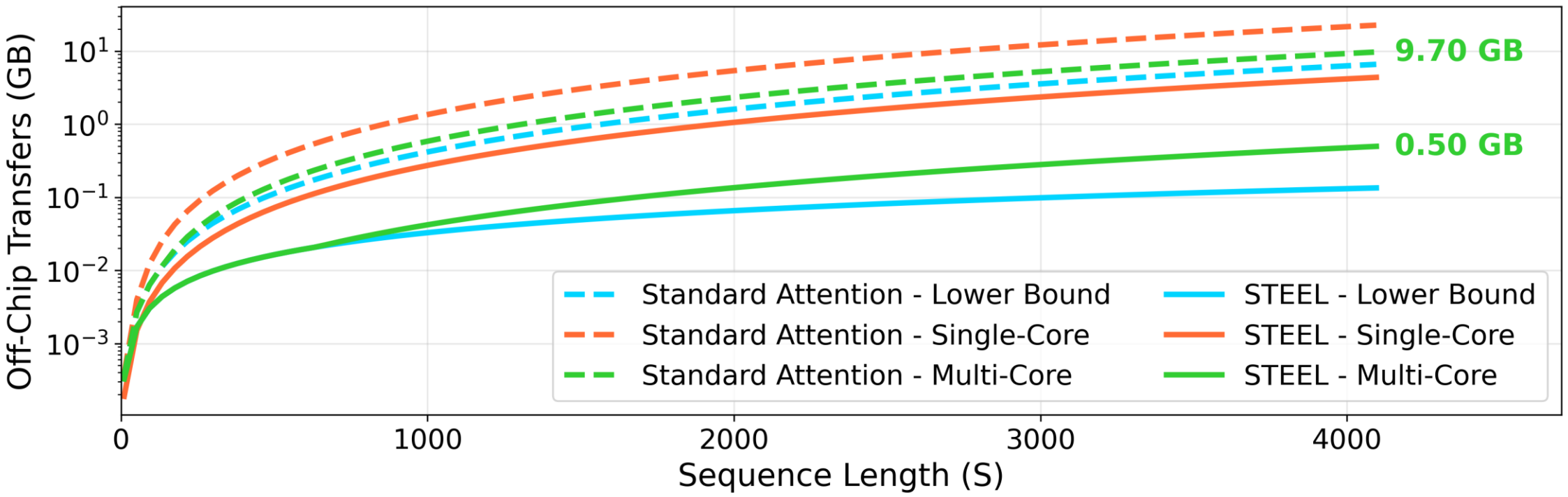
XDNA™2
6.25 FP16 TFLOP/s

XDNA™2 packs
3.8x less compute



Appendix: Flash-Attention Off-Chip Traffic Reduction

Multi-core enables broadcasting → Reduces off-chip transfers



Appendix: Swizzled layout

Required to use the vector unit efficiently

Performed by DMA or AIE Tile