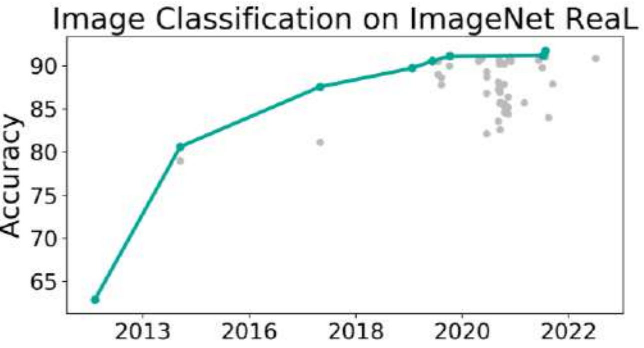
Toward Gen.AI Pervasive Intelligent Systems
An Open RISC-V platform Approach

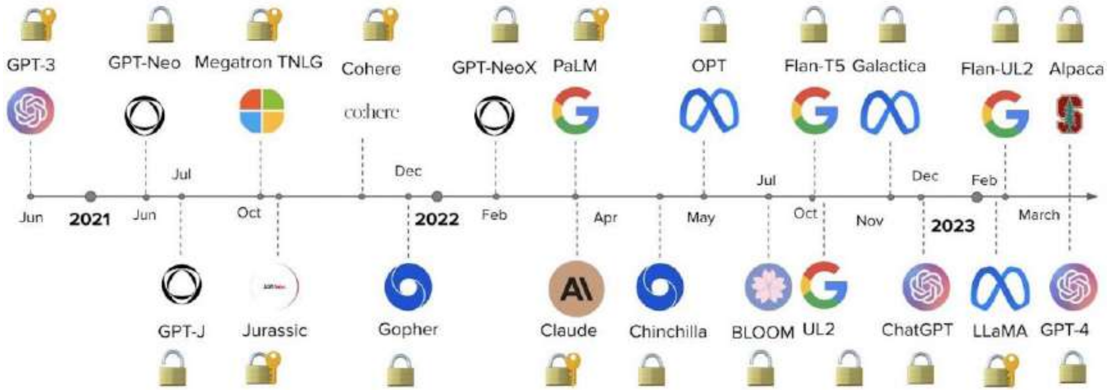
Luca Benini lbenini@iis.ee.ethz.ch

PULP Platform
Open Source Hardware, the way it should be!

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

Perception → Gen.AI → Pervasive Gen.AI





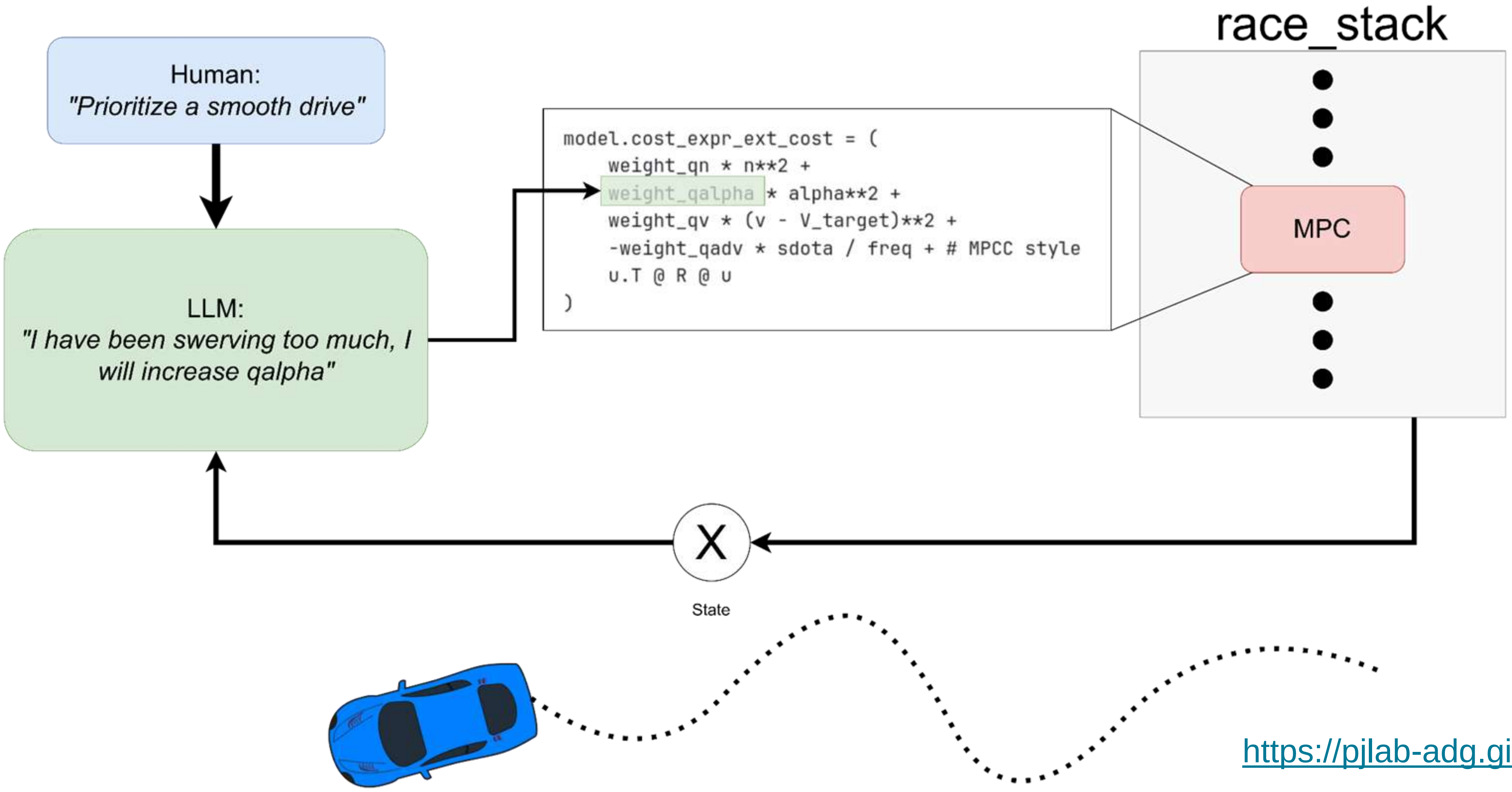
Precise

Interactive, creative

Efficient, RT-safe, secure

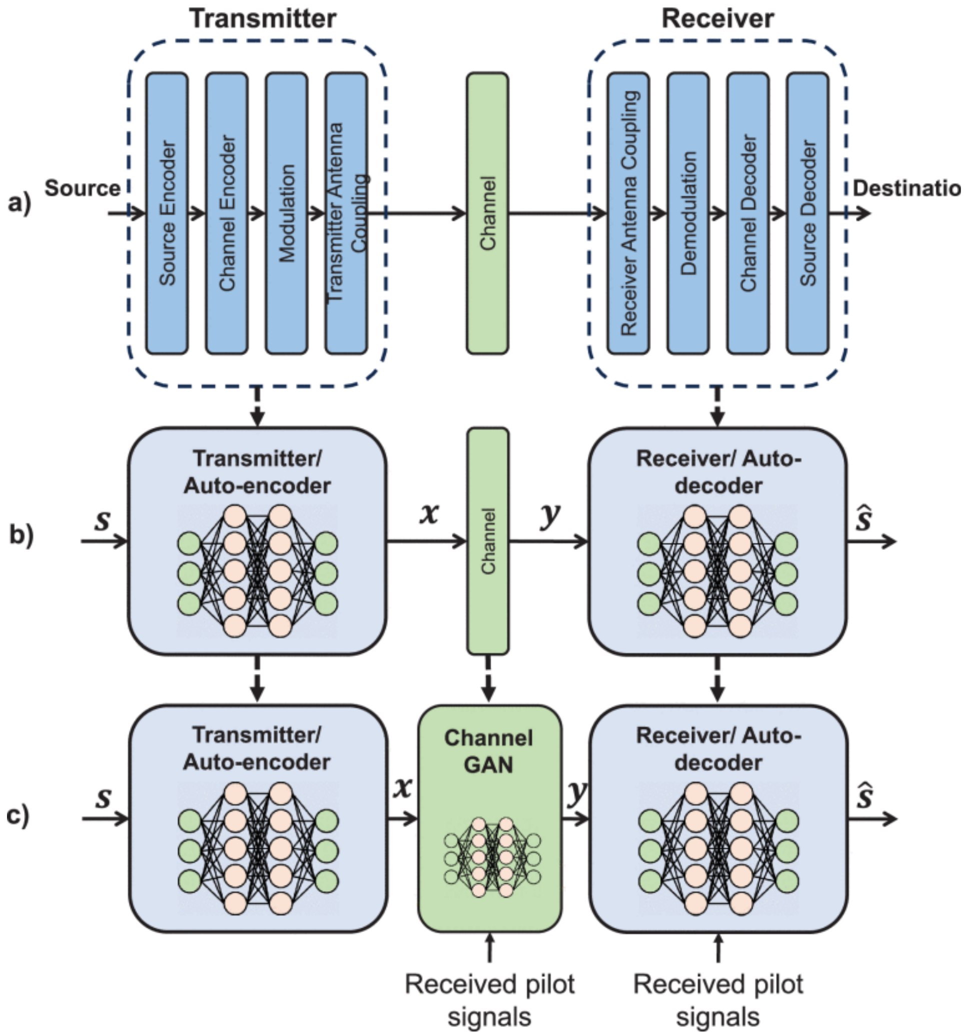
Pervasive Gen.AI: Robots

LLM Reasoning on Human Commands & Robot Observations

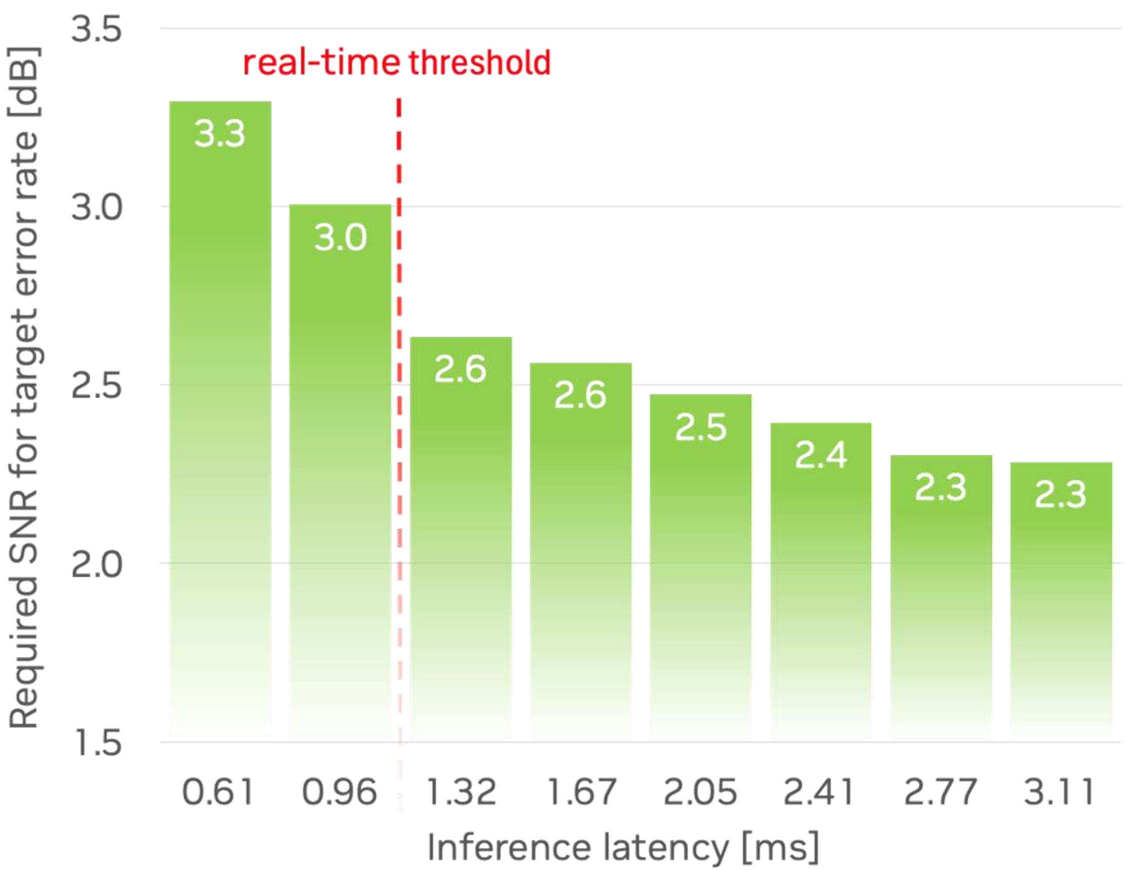


https://pjlab-adg.github.io/DiLu/

Pervasive Gen.AI: AI native Phy for RAN



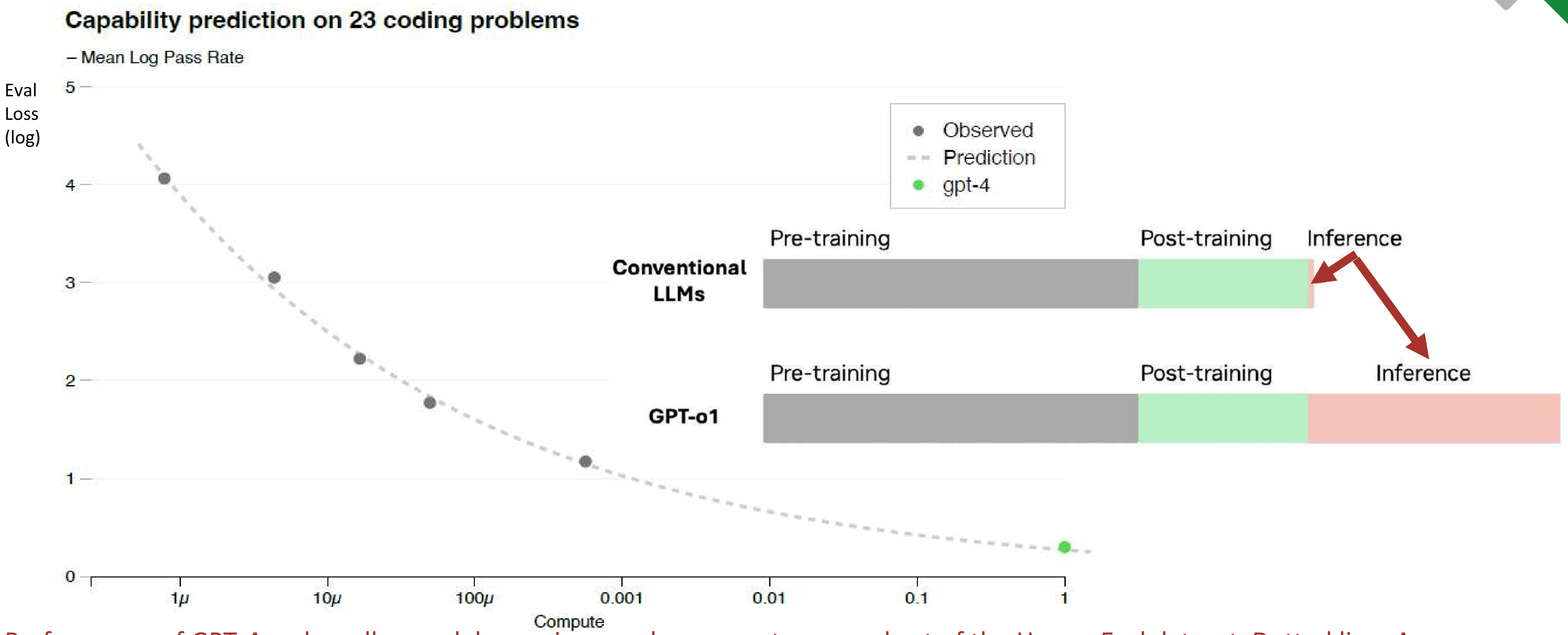
H. Ye, L. Liang, G. Y. Li and B. -H. Juang, "Deep Learning-Based End-to-End Wireless Communication Systems With Conditional GANs as Unknown Channels," IEEE Transactions on Wireless Communications, 19.5, (2020)



https://developer.nvidia.com/blog/real-time-neural-receivers-drive-ai-ran-innovation/

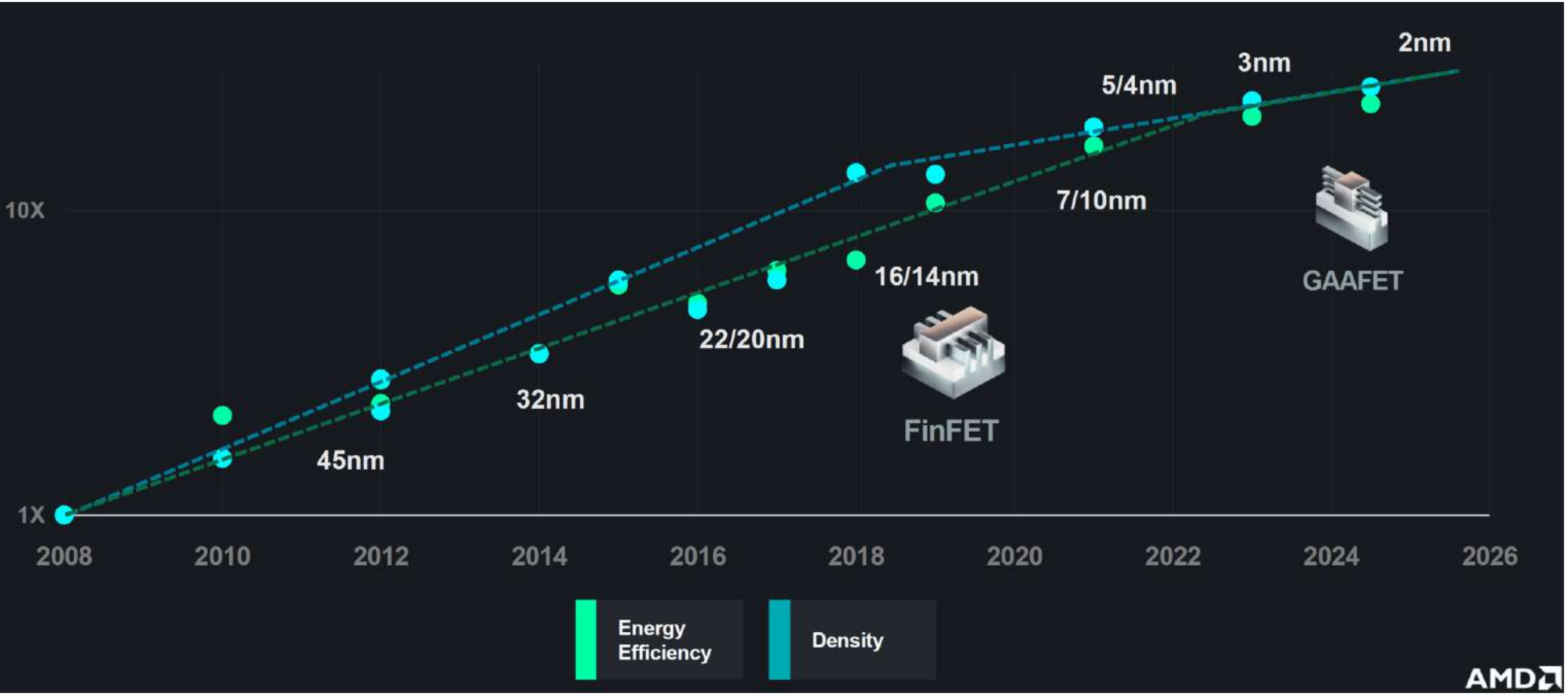
Pervasive Gen.AI Challenge

OpenAI'23 arXiv:2303.08774



Performance of GPT-4 and smaller models: y-axis mean log pass rate on a subset of the HumanEval dataset. Dotted line: A power law fit to smaller models (excluding GPT-4) → Accurately predicts GPT-4's performance. x-axis is training compute (log)

Technology is not Enough



[AMD HotChips24]

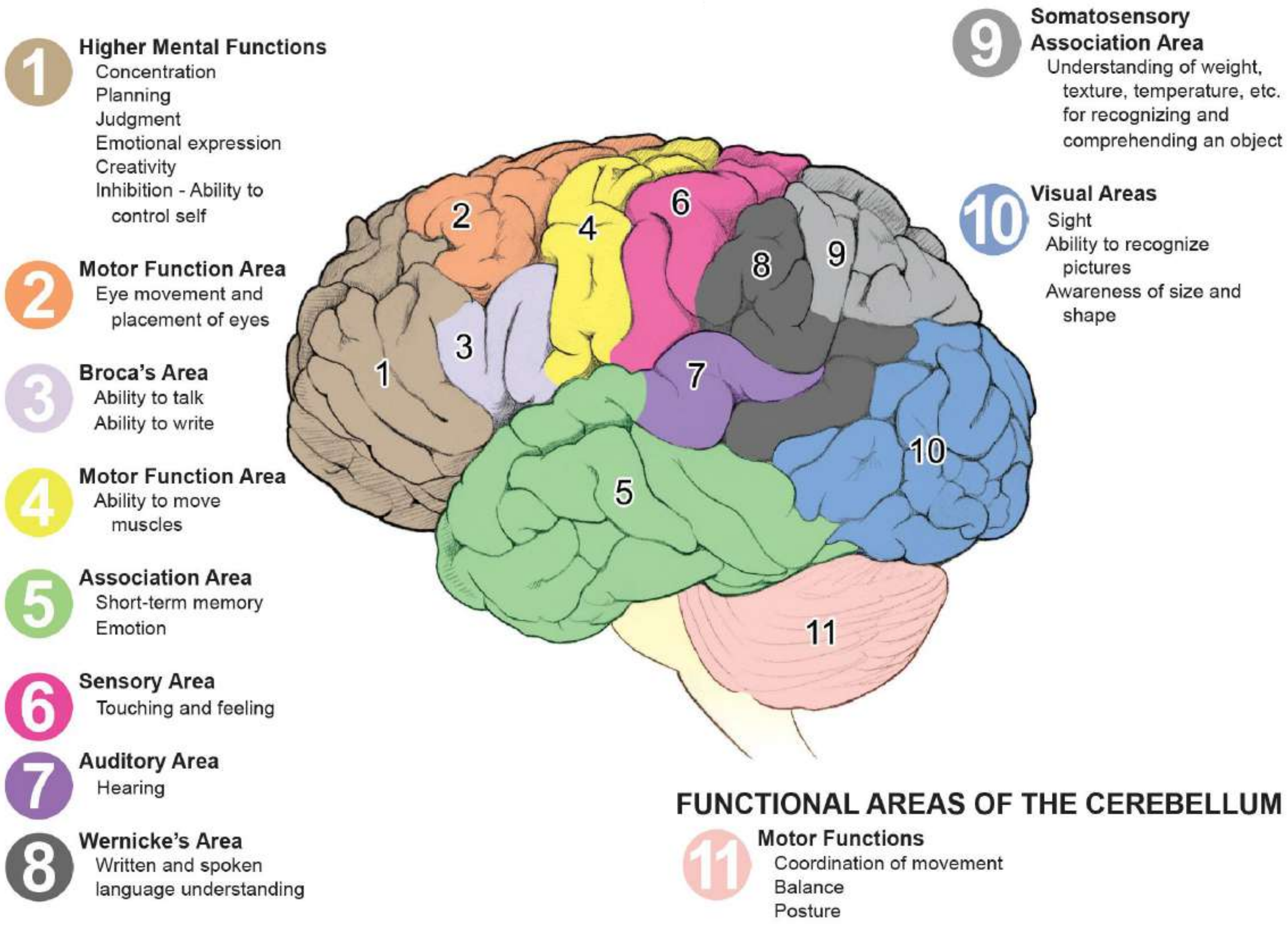
Efficient

On-car Computing
P_{MAX} < 1.5 kW

Model complexity
10× every ~2.5 years
Moore's Law
10x every 12 years!

Efficiency through Heterogeneity: Multi-Specialization

Brain-inspired: Multiple areas, different structure different function!



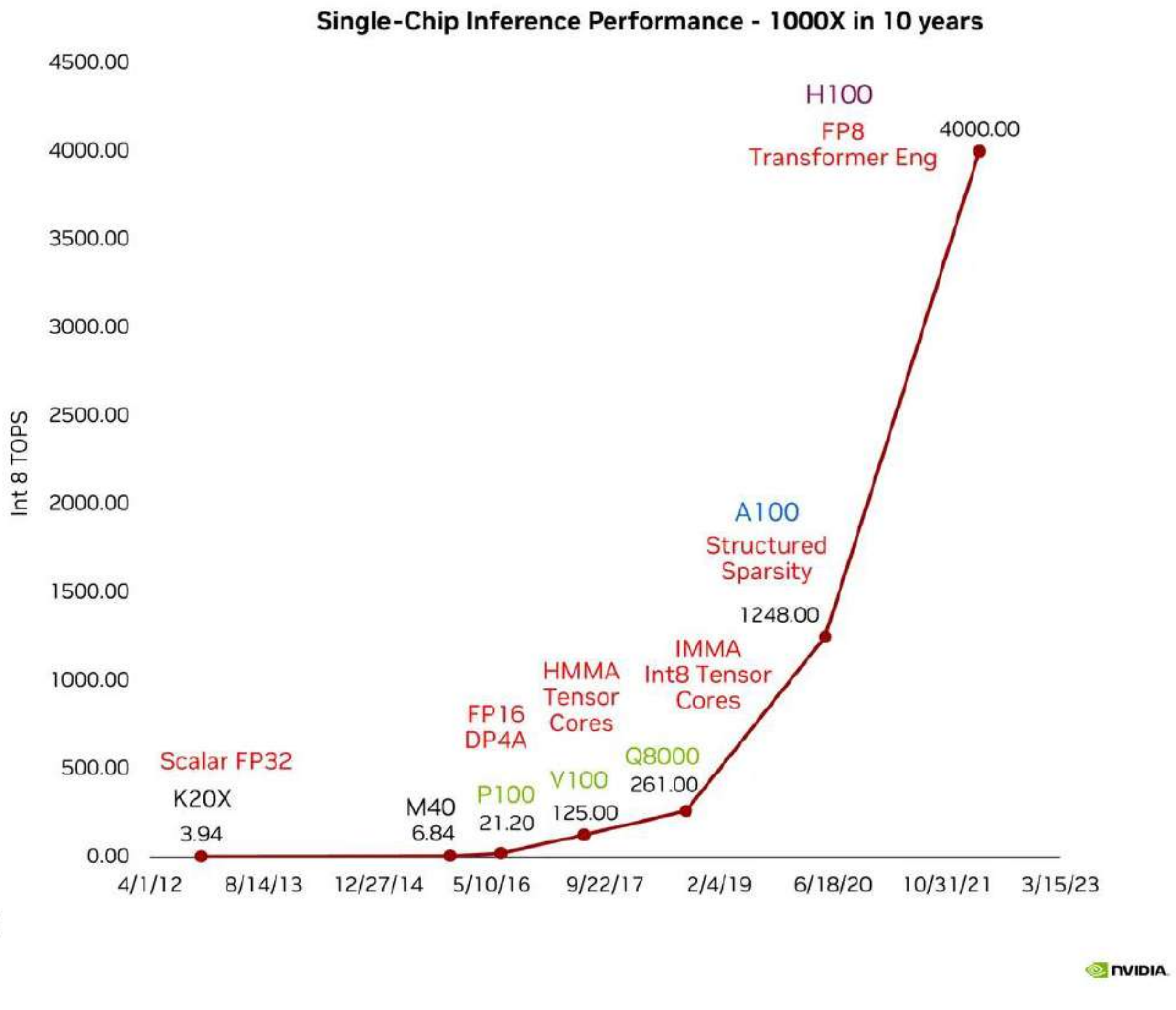
Hailo-10H M.2 Key M ET Generative AI Acceleration Module (40TOPs, few TOPs/W)

Looking up to the Leader

Dally HotChips 2023

Gains from

- Number Representation
  - FP32, FP16, Int8
  - (TF32, BF16)
  - ~16x
- Complex Instructions
  - DP4, HMMA, IMMA
  - ~12.5x
- Process
  - 28nm, 16nm, 7nm, 5nm
  - ~2.5x
- Sparsity
  - ~2x
- Model efficiency has also improved – overall gain > 1000x

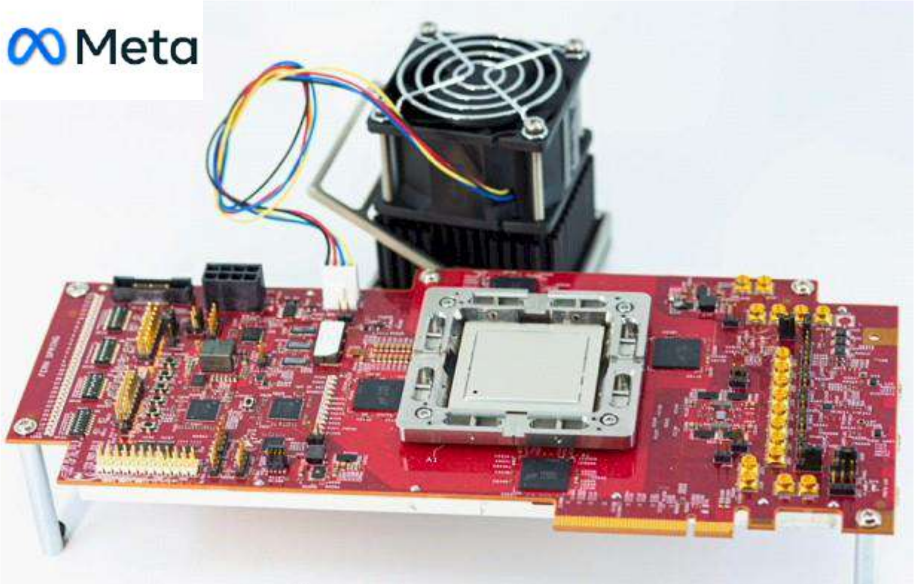


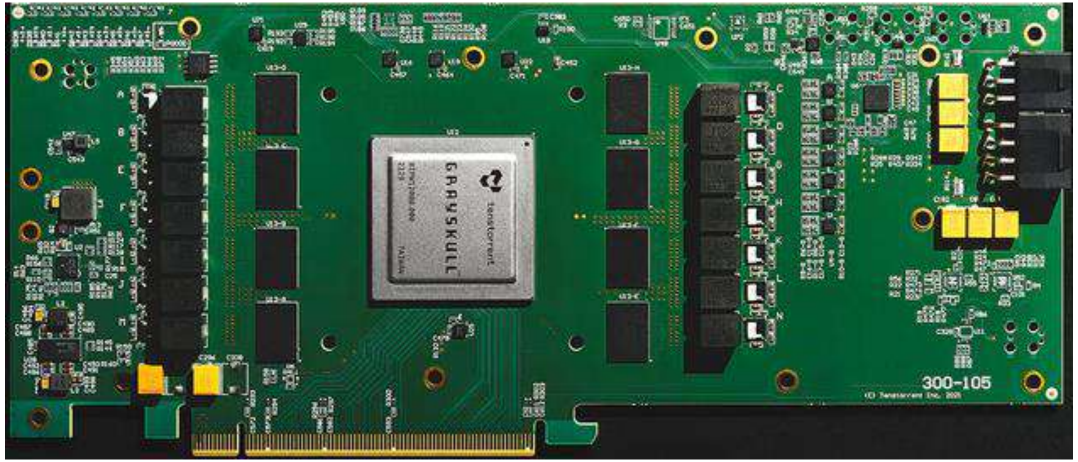
Why NVIDIA owns the Market?

- It’s the software → flexibility, fast evolution!
- Is there a way to Escape “NVIDIA gravity”?
- Need a standard to combat a monopoly

RISC-V®

RISC-V: The Free and Open RISC Instruction Set Architecture





RISC-V is a key enabler → max agility, enabling SW build-up, without vendor lock-in

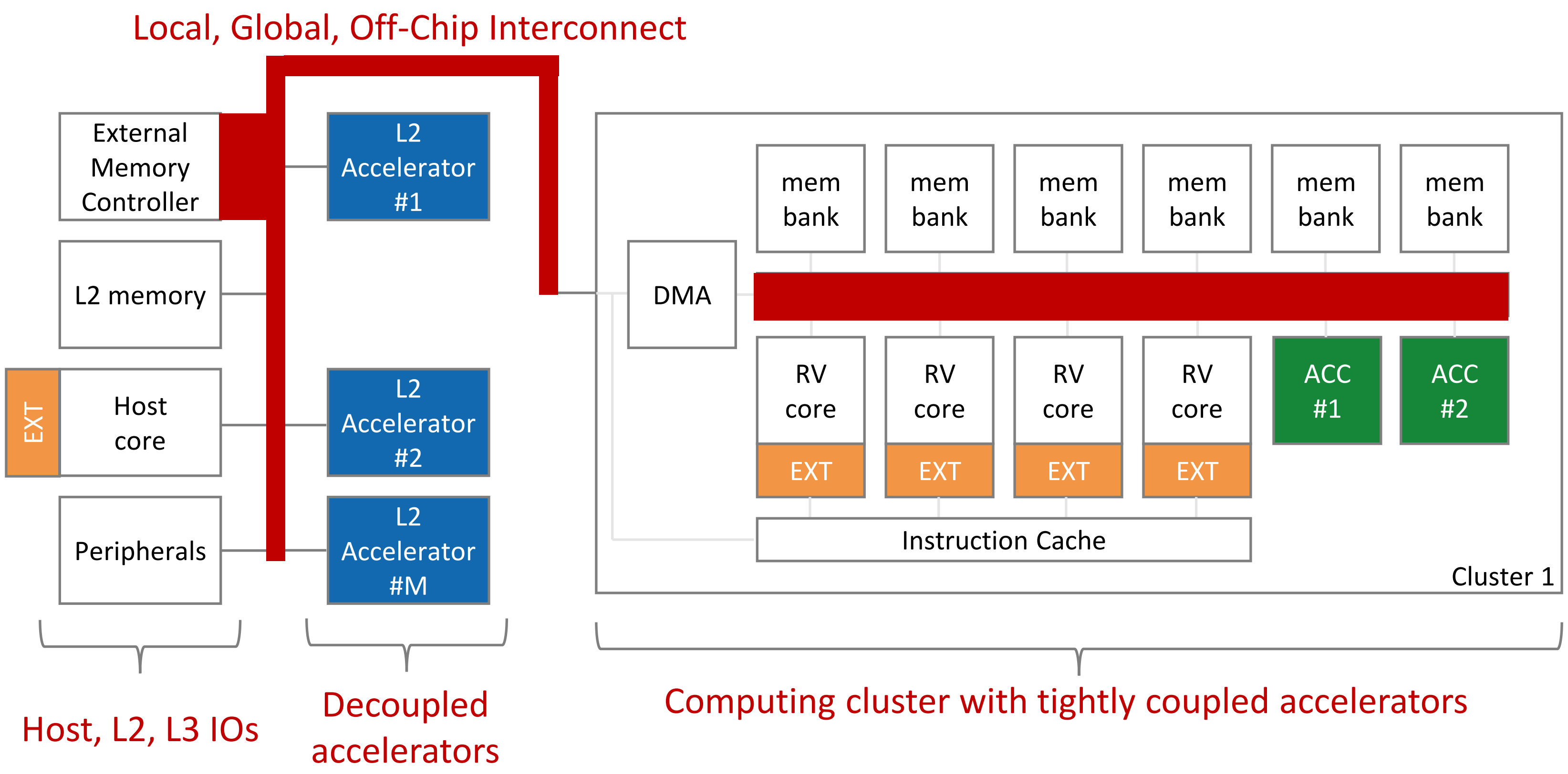
Heterogeneous, Multiscale Accelerated Computing

Multiple Scales of acceleration

Extensions to processor cores
- Explore new extensions
- Efficient implementations

Shared-memory Accelerators
- Domain specific
- Local memory

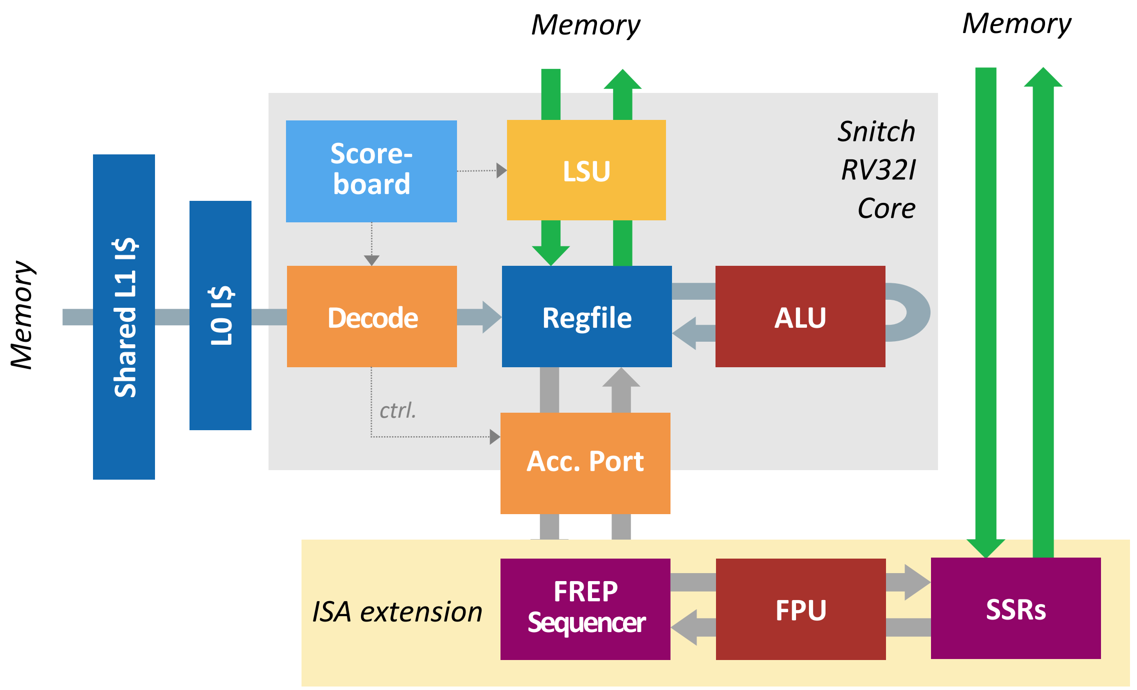
Multiple Decoupled Accelerators
- Communication
- Synchronization



Specialize interconnects too! Local, global, package, system

Snitch Core: Tiny, Latency Tolerant, Extensible RV PE

- ***Snitch*: tiny (20KGE), extensible RV core**
  - *Extensible* through accelerator port
  - *Latency-tolerant* through scoreboard
    → can issue ~10 non-blocking memOPs
- **Paired with *ISA extension subsystem***
- ***Native streaming support***
  - Load/store elision
  - Reduction of I$ pressure

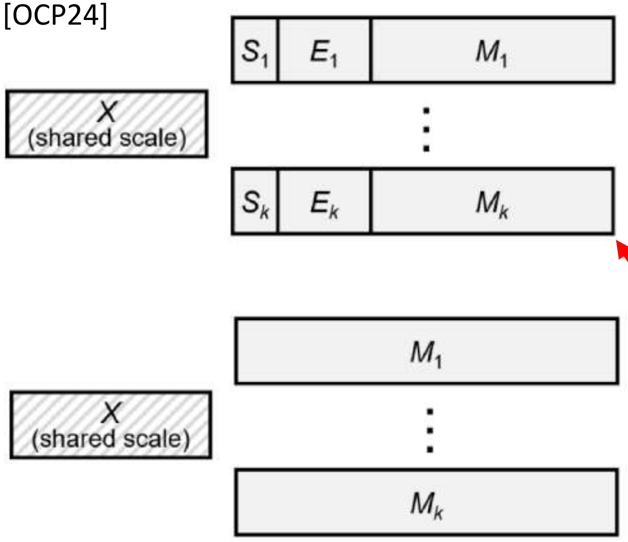


ISA Extension: quantization Galore

Extension for Low-Bitwidth INT (binay, ternary, crumble, nibble, byte) and FP

- Tensor unit support (being standardized now – two versions: “attached” vs. “integrated”)
- OCP ***Microscaling*** Formats (MX) → RVV ISA is a good match
  - Version 1.0 published Sept 2023 - Proponents: AMD, Arm, Intel, Meta, Microsoft, NVIDIA, Qualcomm
- Polynomial Approximation (PACE – stay tuned)

[OCP24]

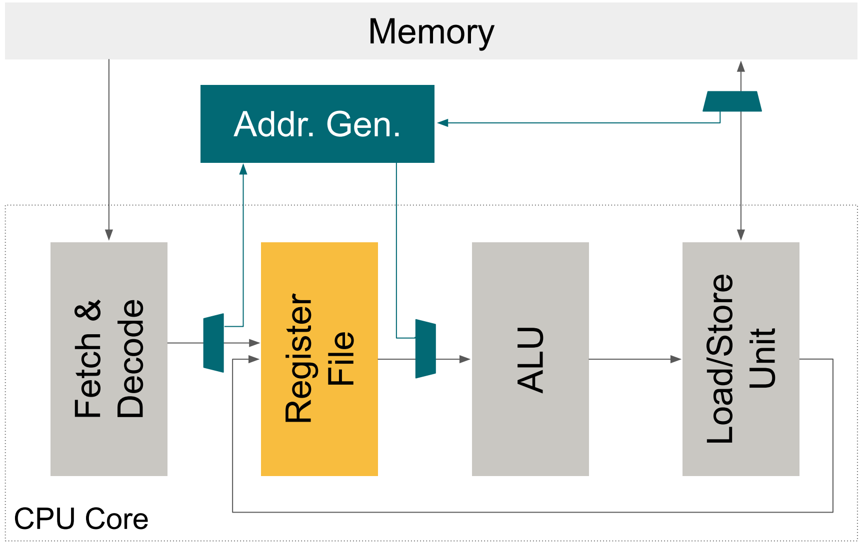


[SemiAnalysis24]

MX Number Formats							
Format	Sign	Exponent	Mantissa	Total bits	Block level exponent	Total bits per block of 32	Total bits per block of 64
FP64	1	11	52	64	0	2048	4096
FP32	1	8	23	32	0	1024	2048
TF32	1	8	10	19	0	608	1216
FP16	1	5	10	16	0	512	1024
BF16	1	8	7	16	0	512	1024
INT8	1	0	7	8	0	256	512
FP8 (e5)	1	5	2	8	0	256	512
FP8 (e4)	1	4	3	8	0	256	512
INT4	1	0	3	4	0	128	256
MXFP8 (e5)	1	5	2	8	**8**	**264**	**520**
MXFP8 (e4)	1	4	3	8	**8**	**264**	**520**
MXFP6 (e3)	1	3	2	6	**8**	**200**	**392**
MXFP6 (e2)	1	2	3	6	**8**	**200**	**392**
MXFP4 (e2)	1	2	1	4	**8**	**136**	**264**
MXINT8	1	0	7	8	**8**	**264**	**520**

SSR & FREP: Streaming Extension

- **SSR**: Link register read/writes into implicit LD/ST
  - Extension around the core's register file
  - Address generators (2-3KGE/SSR)
  - Configured out of inner loop (LD/ST elision)
  - Staggering: generators prefetch from memory (latency tolerant!)
- **FREP**: L0 instruction buffer (no I$ access)
  - Pseudo-dual issue (Int pipeline can proceed in parallel)
  - No boundary checking for loop (similar HW loop in DSPs)
- **Boost FPU utilization → 100% (once setup is amortized)**



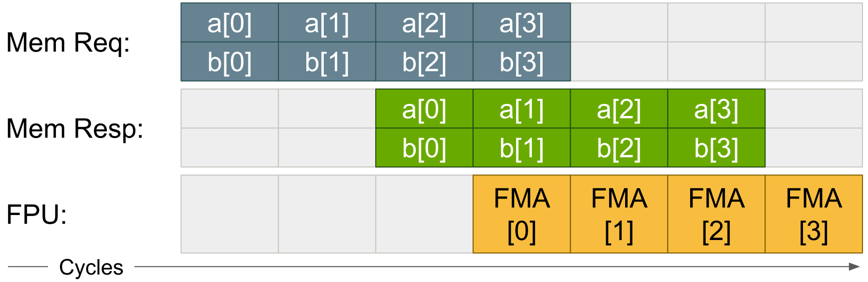
dotp: 30% FPU

```
loop:
fld r0, %[a]
fld r1, %[b]
fmadd r2, r0, r1
```

→

dotp: 90% FPU

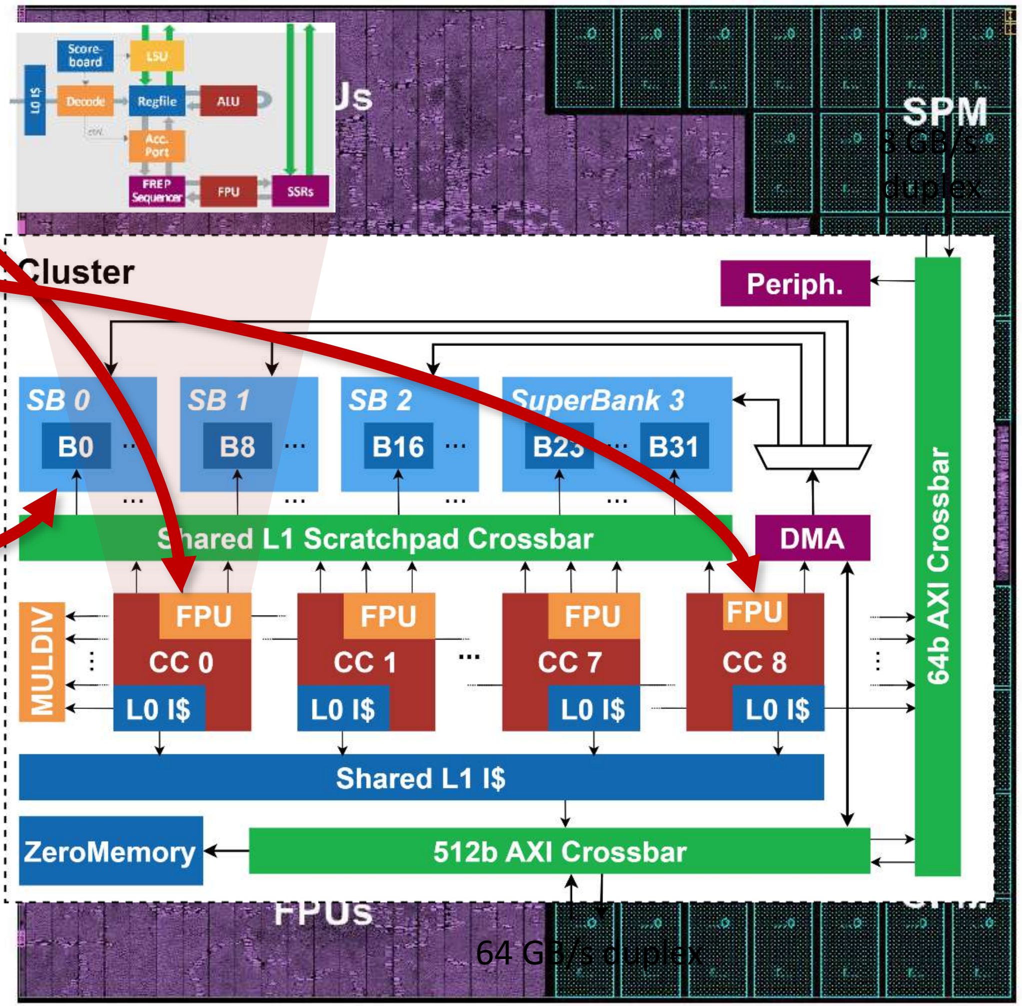
```
scfg 0, %[a], ldA
scfg 1, %[b], ldB
loop:
fmadd r2, ssr0, ssr1
```



Latency Tolerance: Less expensive than OoO (CPU) and Multi-threading (GPU)

Snitch Cluster: The Fundamental Compute Block

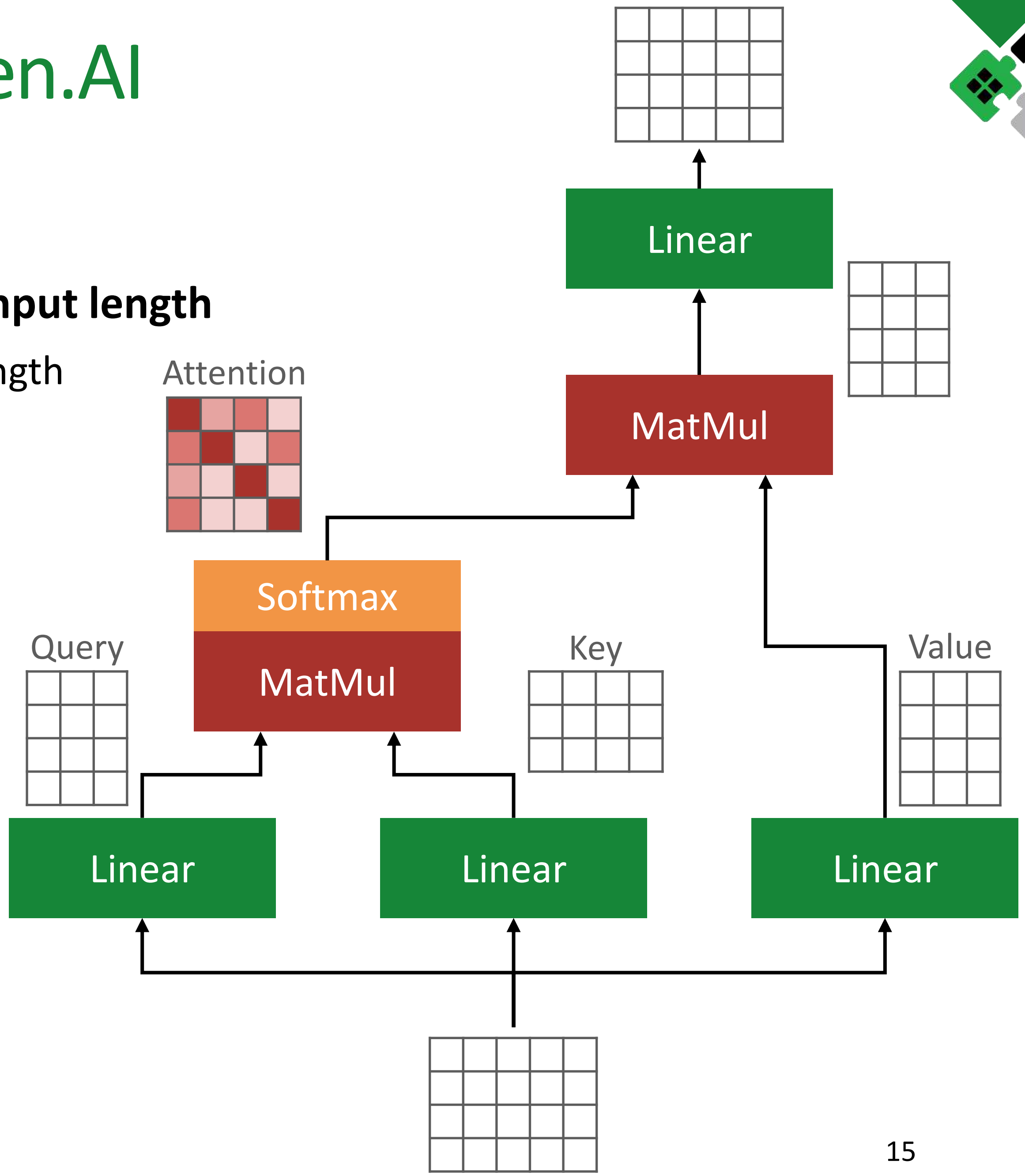
- **8 Snitch compute cores**
  - SIMD 64b FPU with SSRs & FREP
- **9th Core: DMA engine**
  - 512b interface to interconnect
  - HW support for autonomous ≤ 2D transfers, higher dimensions through SW
  - *Latency-tolerance block transfers* (100s of cycles)
- **128 KiB TCDM**
  - 32-bank, low-latency shared scratchpad
  - Double-buffer large chunks with DMA
- **Shared TDCDM, I-cache and peripherals**
- **Shared DMA (10% overhead) for global latency tolerance**



Specializing the Cluster for Gen.AI

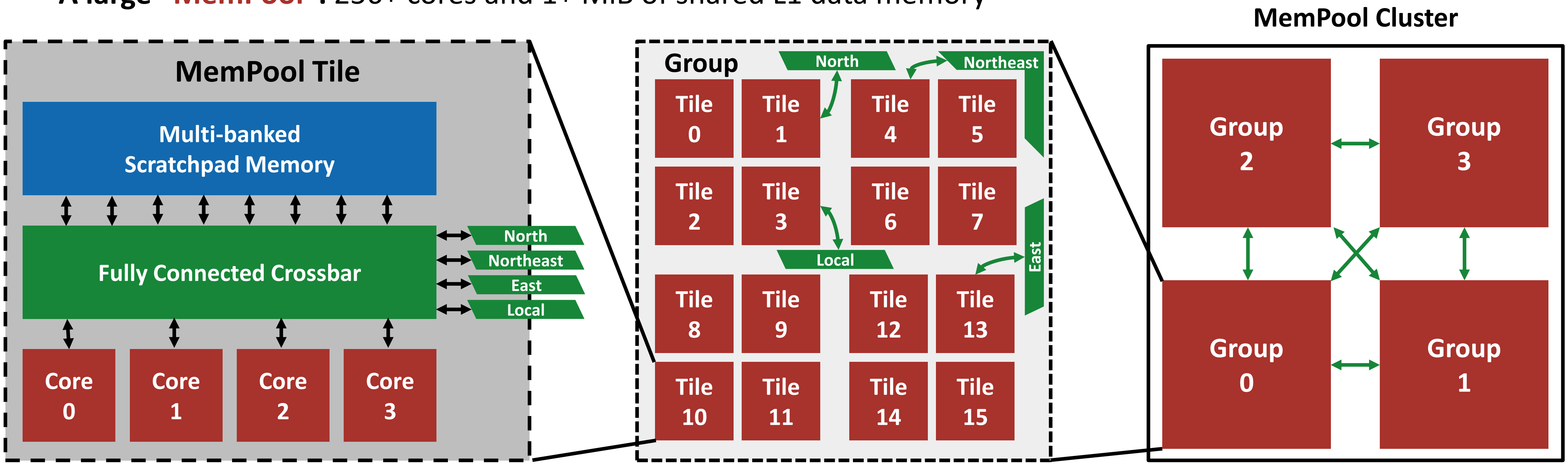
- **Attention is key**
- **Attention matrix is a square matrix of order input length**
  - Quadratic memory requirement vs. sequence length
  - No asymmetry between operands (“weightless”)
- **MatMul & Softmax dominate**

$$\mathrm{Softmax}(\mathbf{x})_i = \frac{e^{x_i - \max(\mathbf{x})}}{\sum_j^n e^{x_j - \max(\mathbf{x})}}$$



Matmul Benefits from Large Shared-L1 clusters

- **Why?**
  - Better global latency tolerance if $\mathbf{L1_{size} > 2\times L2_{latency} \times L2_{bandwidth}}$ (Little's law + double buffer)
  - Smaller data partitioning overhead
  - Larger Compute/Boundary bandwidth ratio: $\mathbf{N^3/N^2}$ for MMUL grows linearly with N!
- **A large "MemPool":** 256+ cores and 1+ MiB of shared L1 data memory



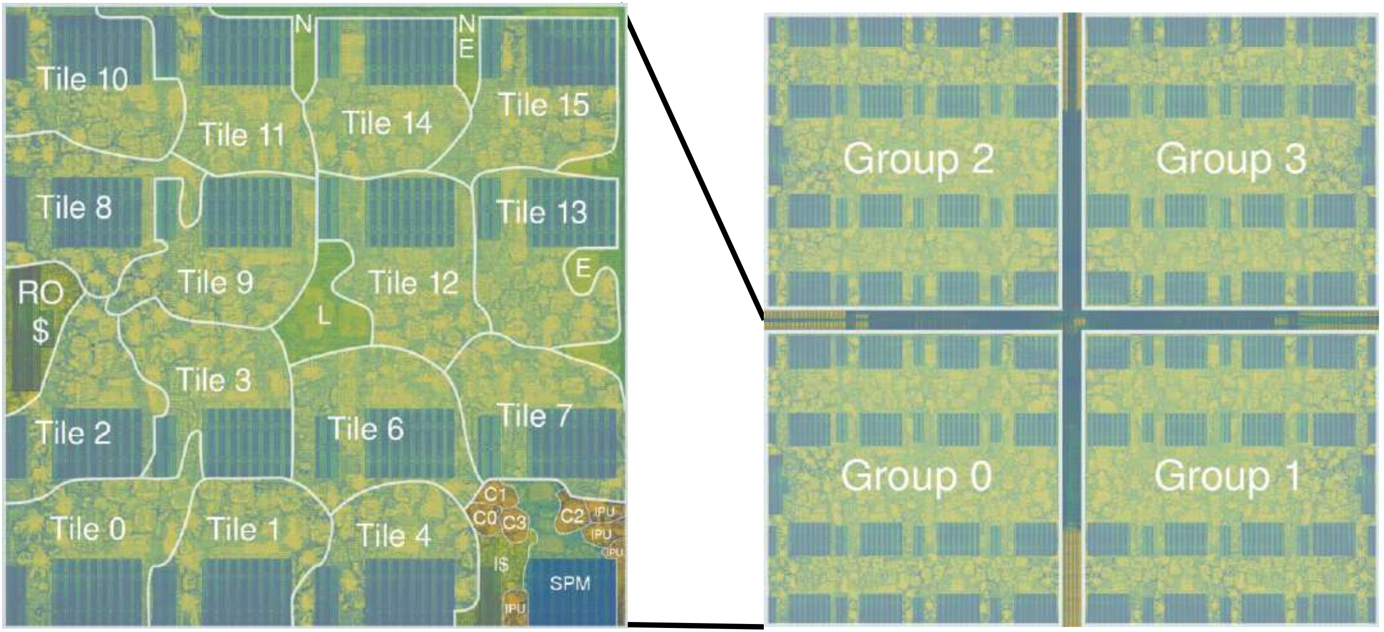
MemPool Cluster: A physical-aware design

- **A Scalable Manycore Architecture with Low-Latency Shared L1 Memory**
  - 256+ cores
  - 1+ MiB of shared L1 data memory
  - ≤ 8 cycle latency (Snitch can handle it)
- **Hierarchical design**
- **Implemented in GF22**
  - Targeting 500 MHz (SS/0.72V/125°C)
  - Reaching 600 MHz (TT/0.80V/25°C)
  - Targeting iso-frequency with PULP
- **Cluster area of 13 mm²**
  - 5 mm diagonal
  - Round trip in 5 cycles
- **Terapool: 1024 Cores!**

MemPool
&
Terapool

MemPool Group

MemPool Cluster



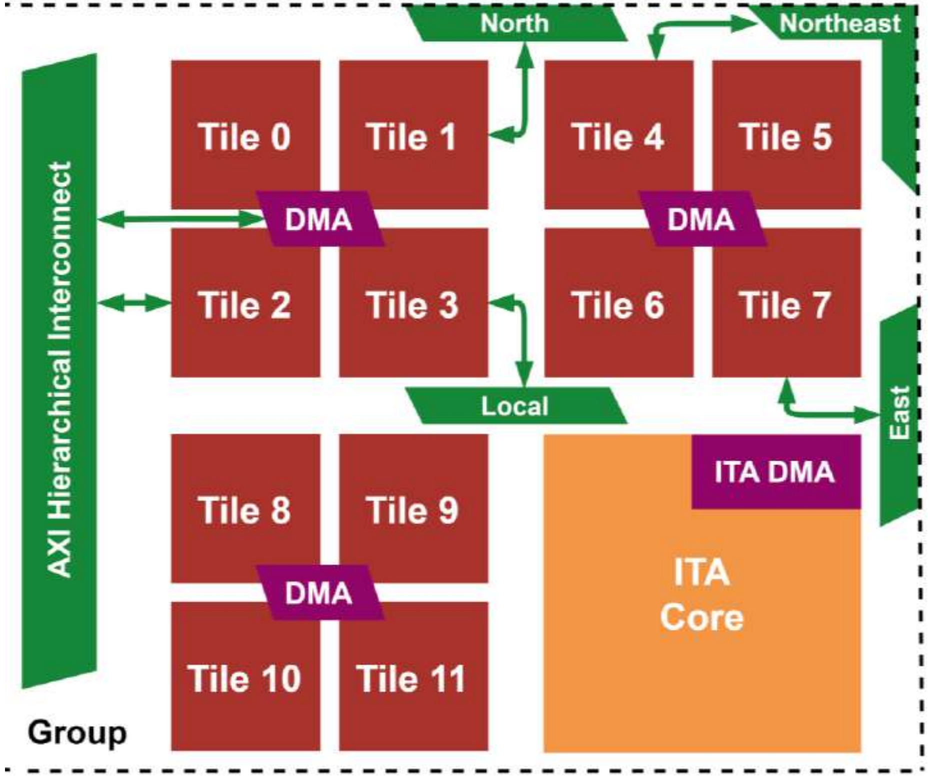
MemPool + Integer Transformer Accelerator (ITA)

Tightly coupled Acceleration Enginee

- Matmul & Softmax
- Reduce pressure on memory and interconnect

Collaborative Execution

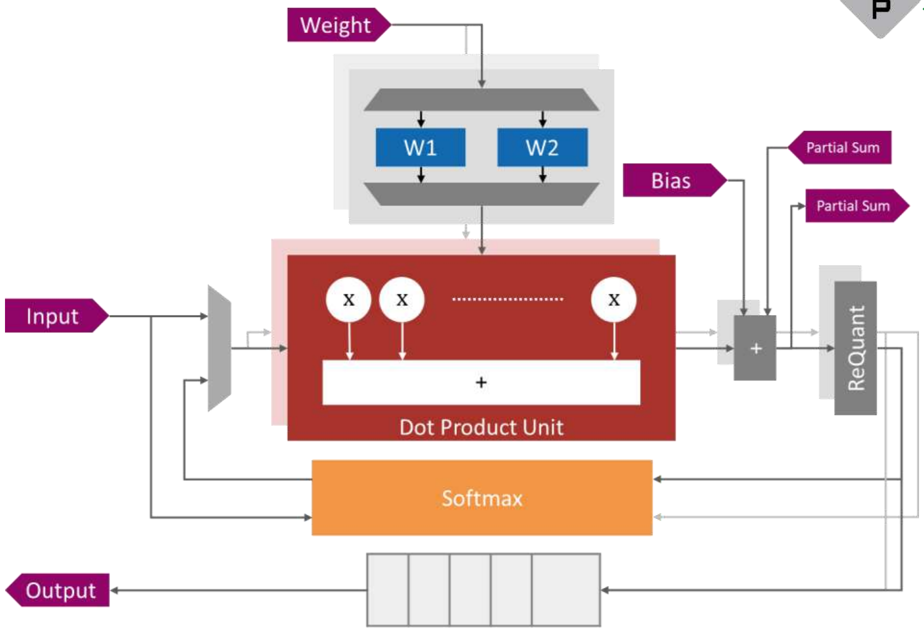
- Cores prepare activations for the next attention head
- Final head accumulation computed in cores
- Nonlinearity in cores (PACE)

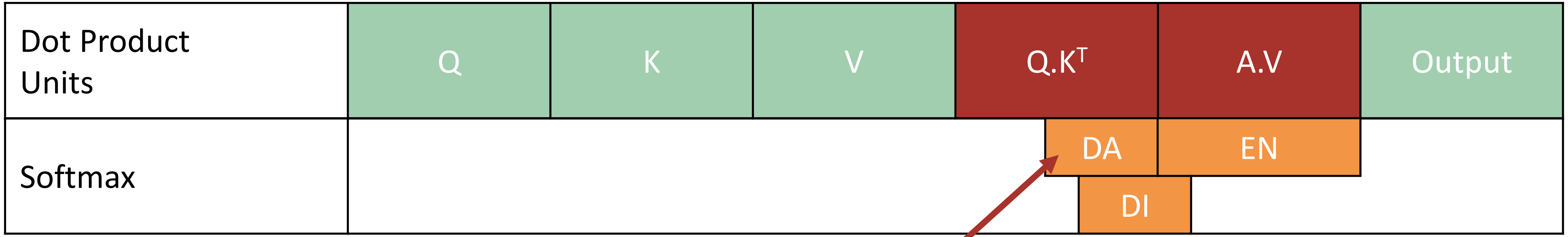


MemPool + Integer Transformer Accelerator (ITA)

Integer Attention Accelerator

- 8-bit inputs, weights & outputs
- Builtin data marshaling & pipelined operation
- Streaming partial Softmax **adding no additional latency**
- Fused Q× K^T, Softmax and A × V computation
- Support for hardware-aware Softmax approximation in QuantLib





$$e^{a_i - a_{max_{n+1}}} = e^{a_i - a_{max_n}} \cdot e^{a_{max_n} - a_{max_{n+1}}}$$

Extending ITA to MXTA

Dot Product between a pair of blocks
(Dot)

↓

Accumulation over multiple blocks
(DotGeneral)

↓

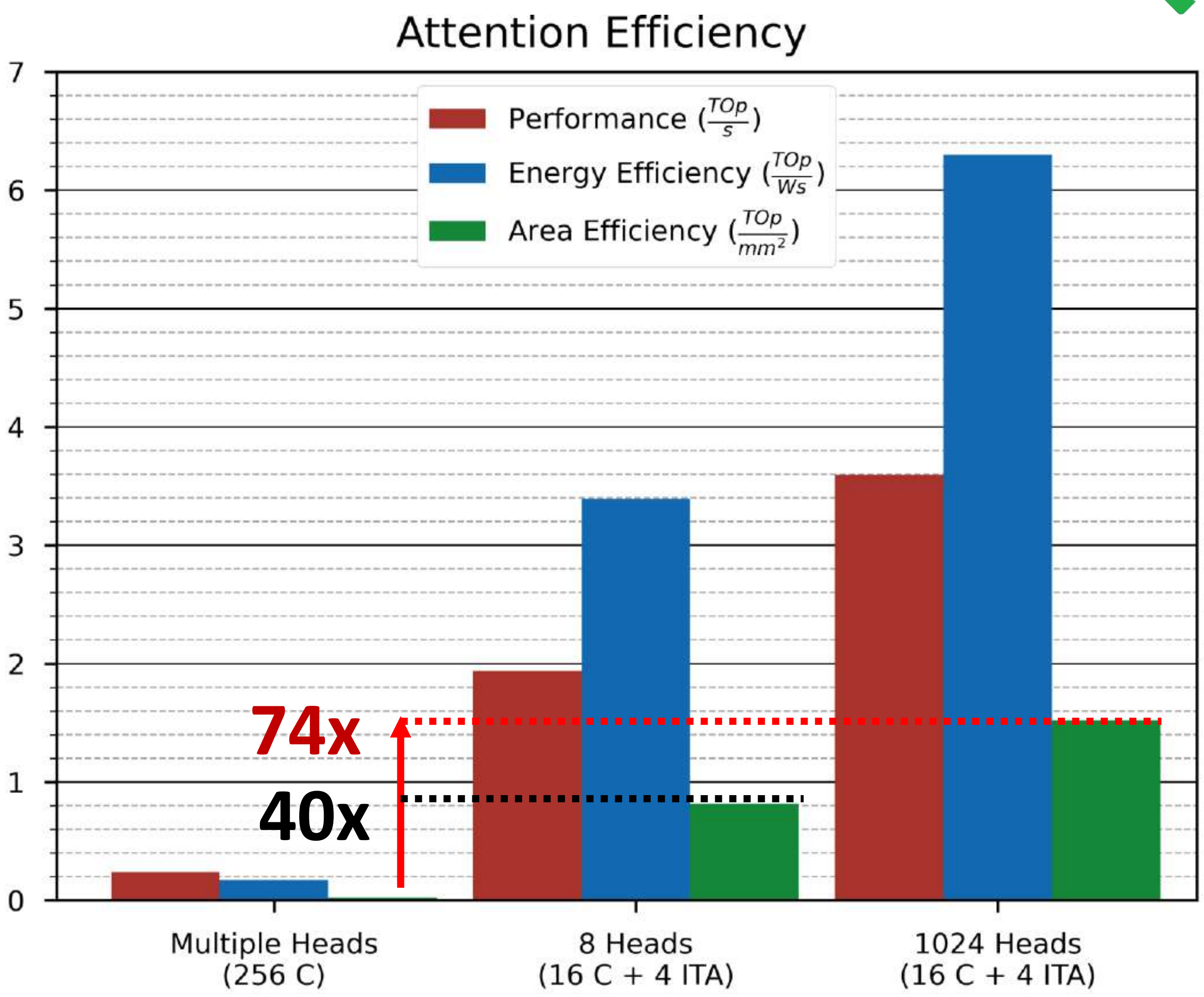
Conversion of output to block format

Attention on ITA

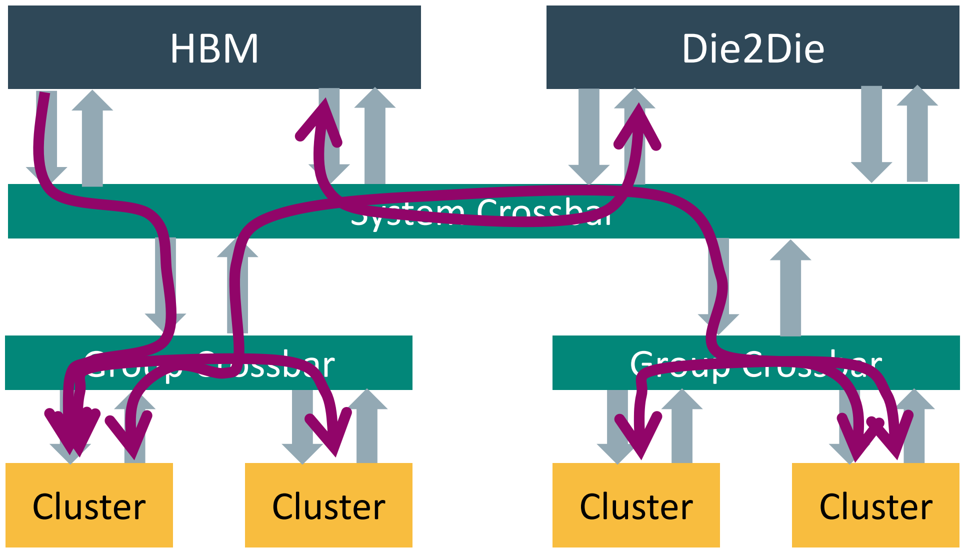
Performance
increase of **15x**

Energy Efficiency
increase of **36x**

Area Efficiency
increase of **74x**



Scaling UP: Efficient and Flexible Data Movement



Problem: HBM Accesses are critical in terms of

- Access energy
- Congestion
- High latency

Instead reuse data on lower levels of the memory hierarchy

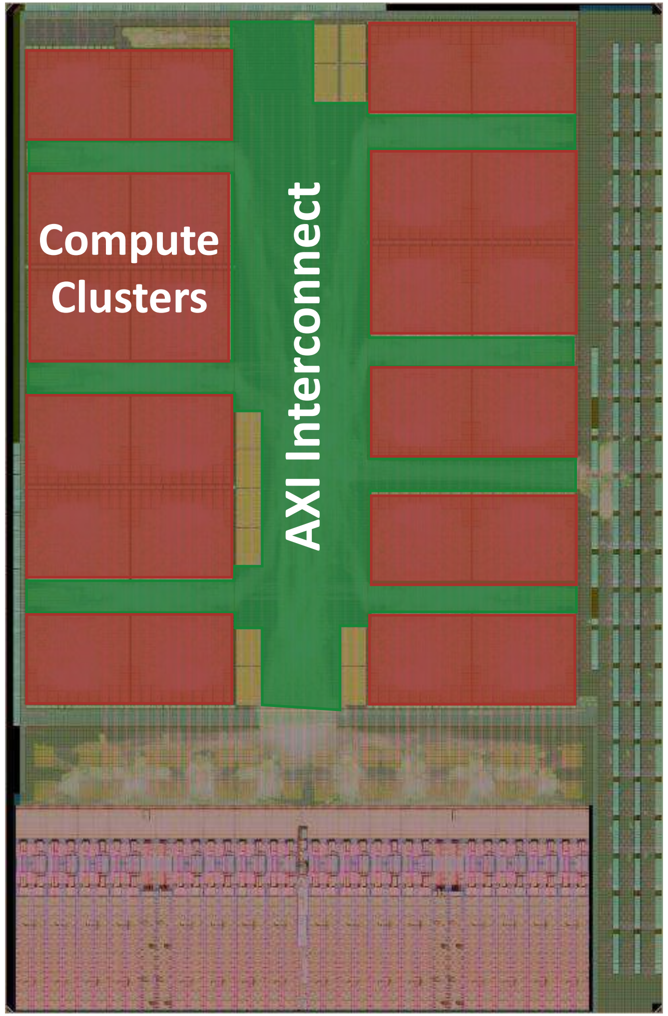
- Between **clusters**
- Across **groups**

Smartly distribute workload

- **Clusters**: Tiling, Depth-First
- **Chiplets**: E.g. Layer pipelining

Big trend!

Addressing interconnect scalability

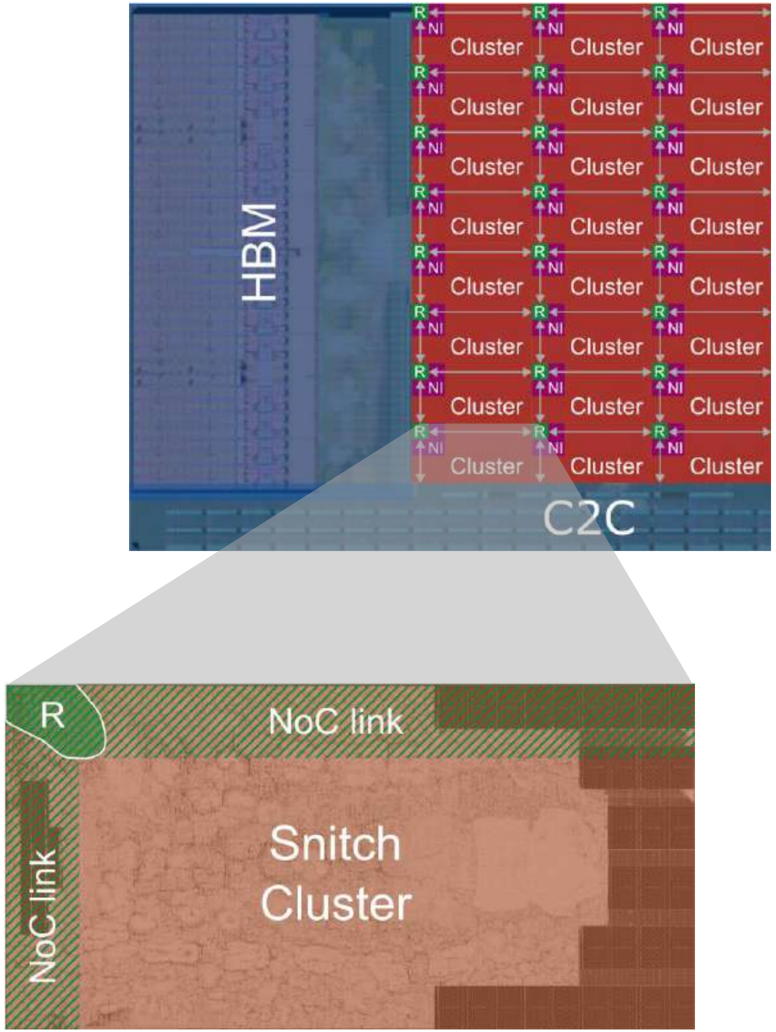


- **Fat-tree was very challenging in Implementation**
  - AXI has severe scalability issues
  - Top-level Xbar had to be split up
  - Still, interconnect takes up almost **40%***
- **Working on NoC solution, *FlooNoC***
  - Fully AXI4 compatible
  - Solves AXI4 **scalability issues**
  - Designed with awareness of physical design
  - **Wide** & **physical** channels

*HBM & C2C excluded

Replacing the AXI interconnect with a NoC

- **Potential for big area/performance gains**
  - Only **~10%** interconnect area
  - **66%** more clusters, same floorplan
  - *High Bandwidth*: **629Gbps/link**
  - *High Energy-Efficiency*: **0.19pj/B/hop**

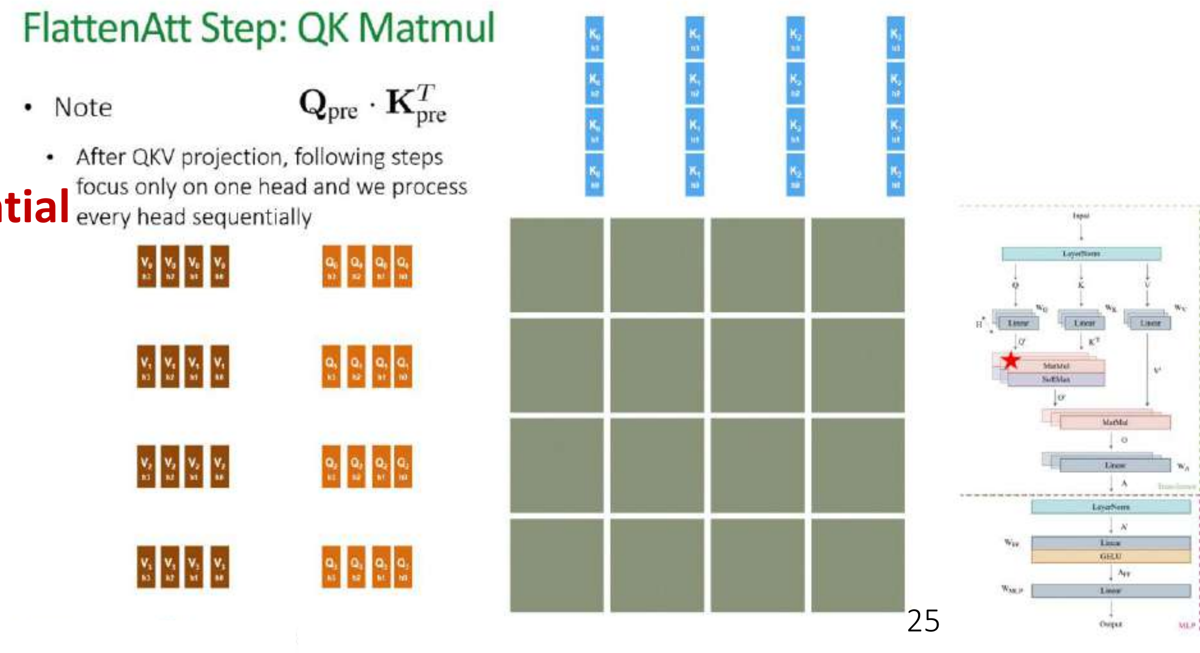


MHA Mapping on NoC: FlattenAttention

- Proposed Dataflow Schedule of MHA
  - We leverage all-cluster L1 for single head attention – Minimize I/O complexity
  - Gen.AI specialized NoC
    - Matrix transpose engine for transposition of (K -> K^T)
    - Collective operations on NoC

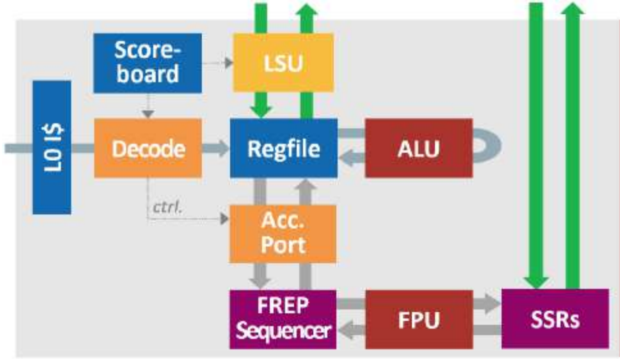
Total Runtime(ms)	
Baseline: Flash Attention for Each Head on Each Cluster	14.4
Flatten Attention (w/o NoC collective)	17.7
Flatten Attention (w/ NoC collective)	**4.6**

- Benchmark & Results
  - 16x16 Clusters (8TFLOPS, 256kB L1), 2TB/s HBM
  - One layer MHA of Llama3-70B (seq=4K, batch=8)
  - **Efficient collective operation support on NoC is essential**
    - **3x speedup to baseline**

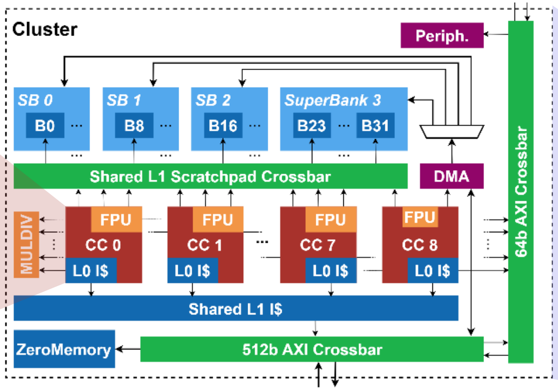


Scaling UP: From Chip to chiplets

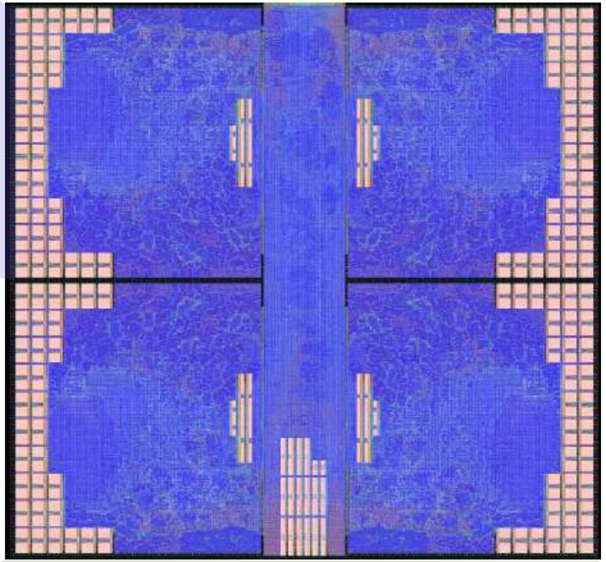
Snitch Core



Snitch Cluster

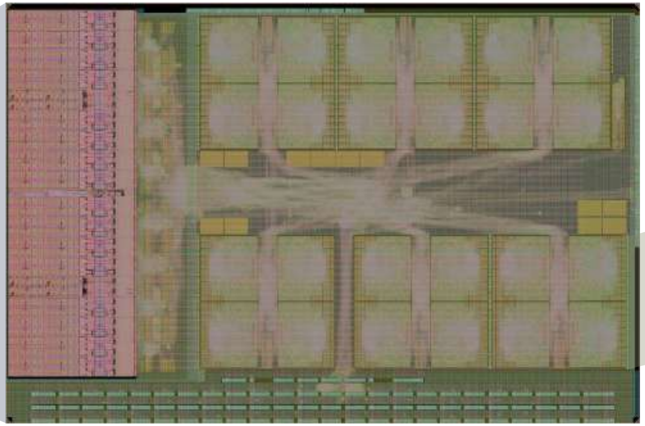


Occamy Group

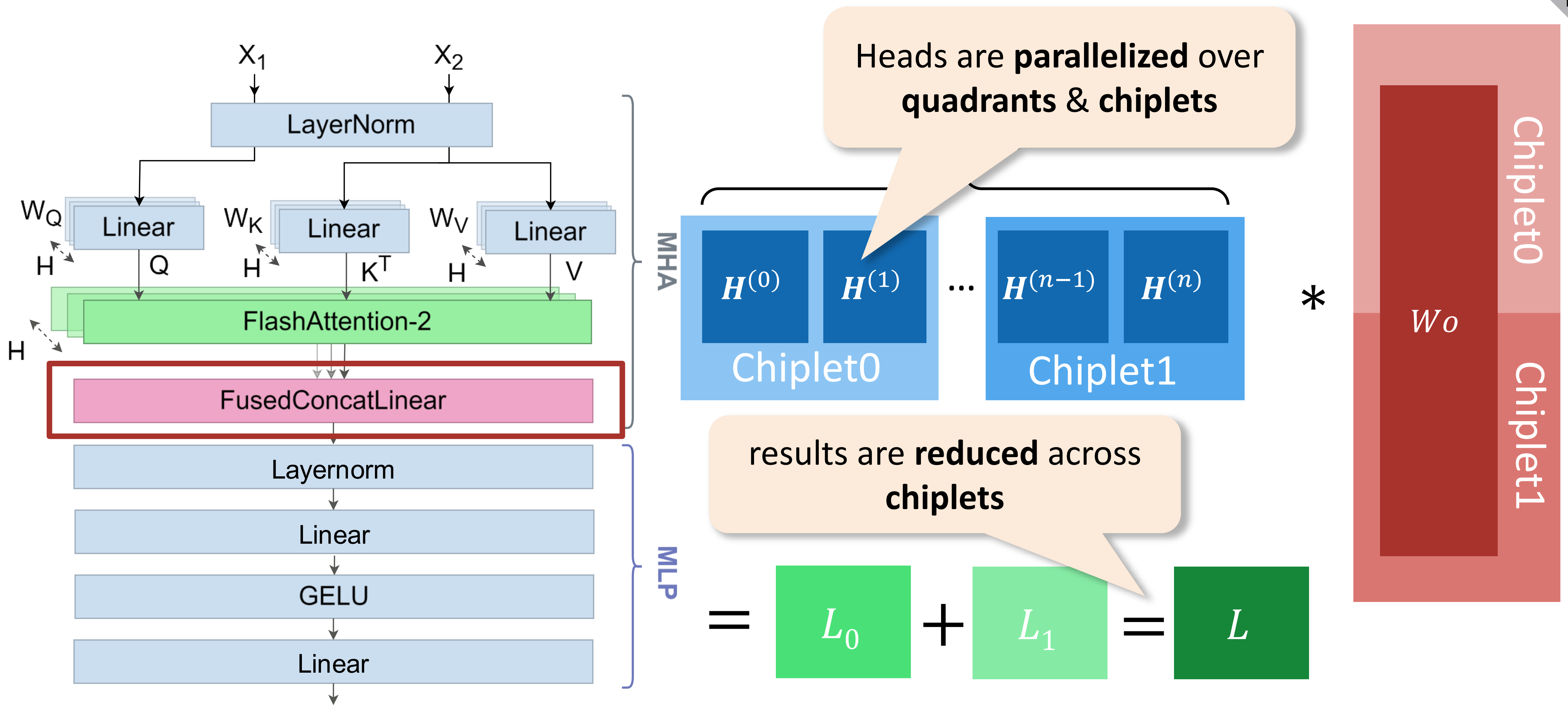


Occamy System

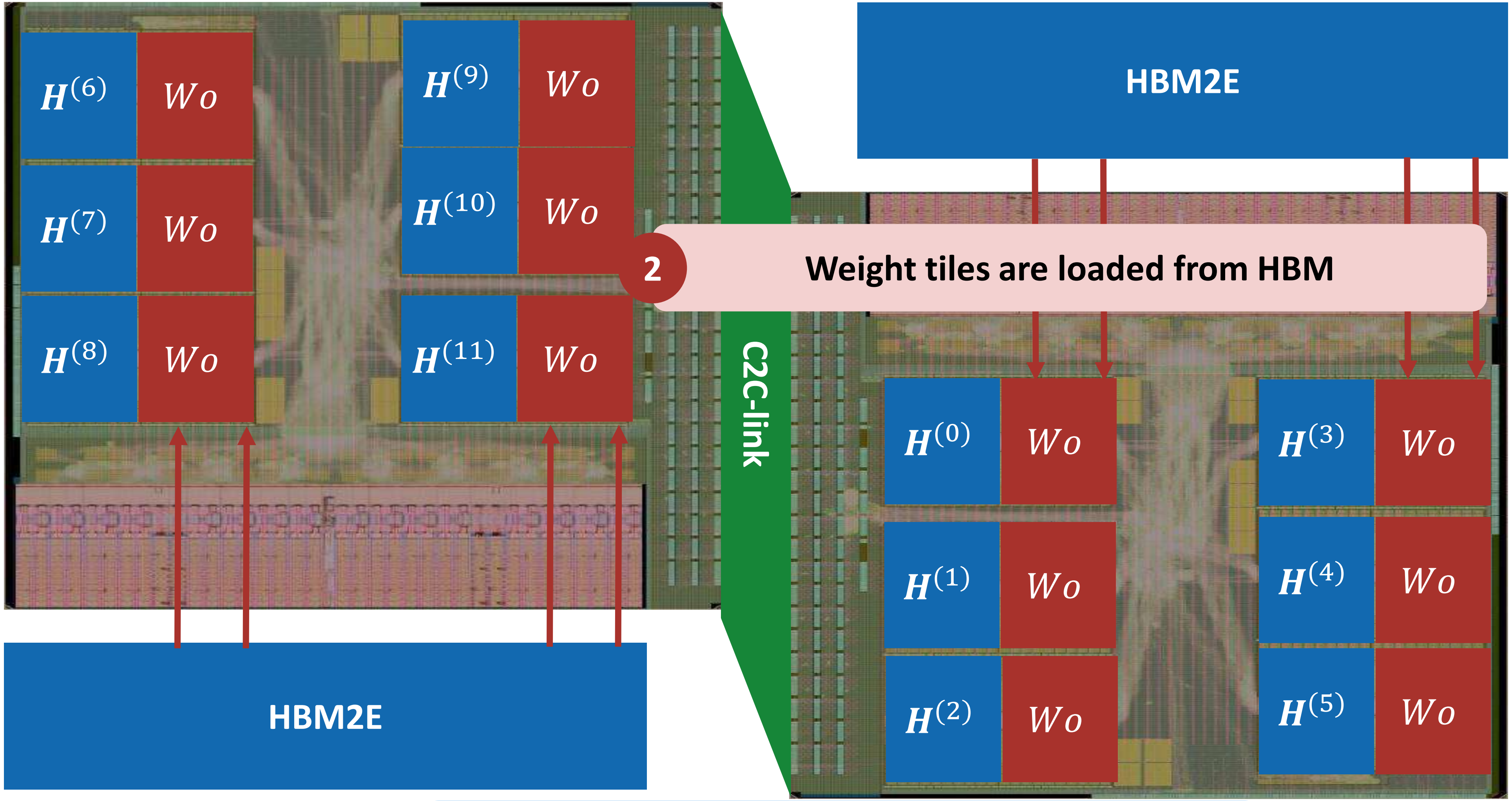
Occamy Chiplet



Not Only Layer-by-Layer distribution across Chiplets!



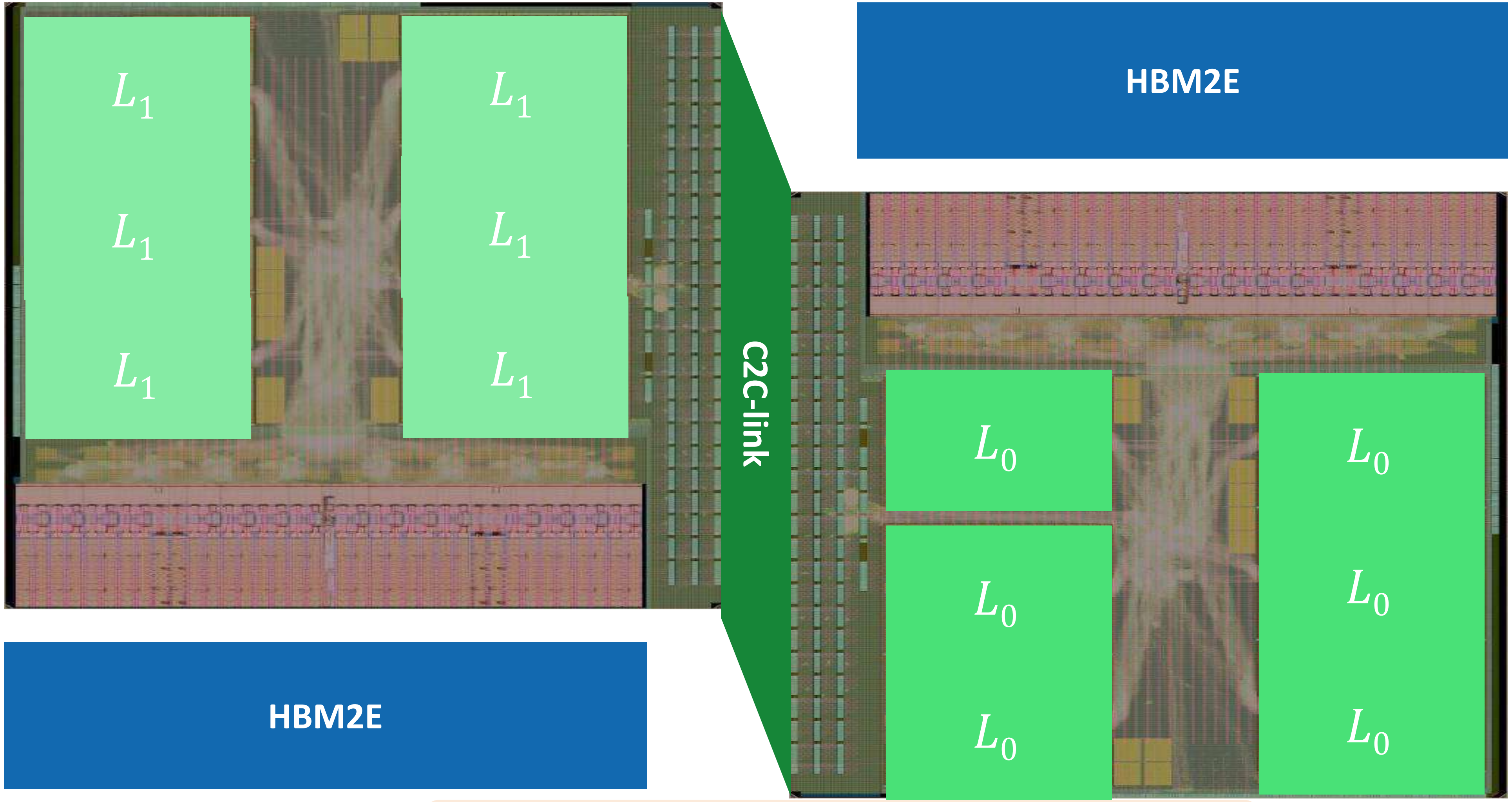
Linear Projection & Head Concatenation Fusion



2 Weight tiles are loaded from HBM

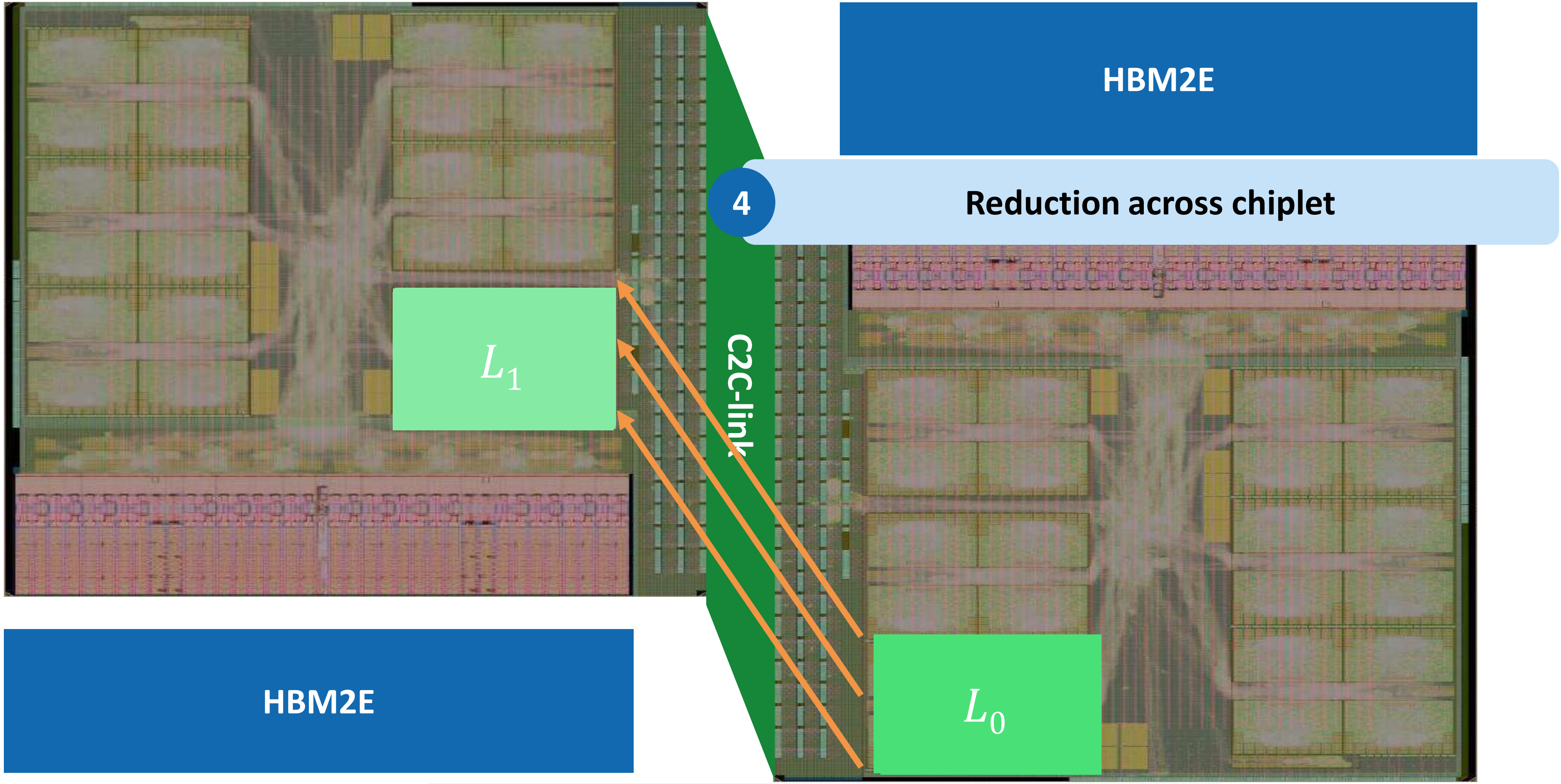
1 Heads are already stored in clusters

Linear Projection & Head Concatenation Fusion



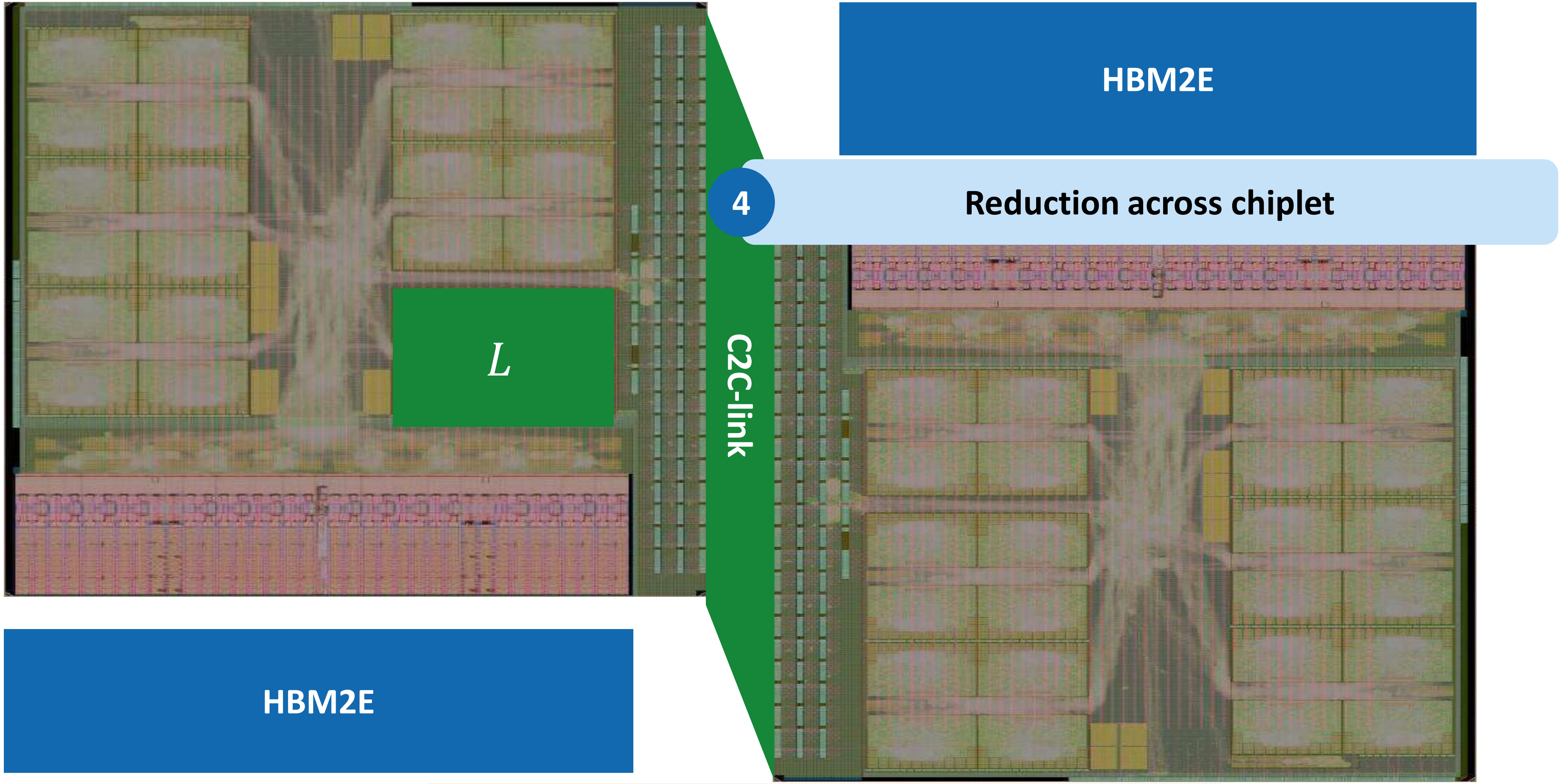
3 **Logarithmic-tree result reduction on-chiplet**

Linear Projection & Head Concatenation Fusion



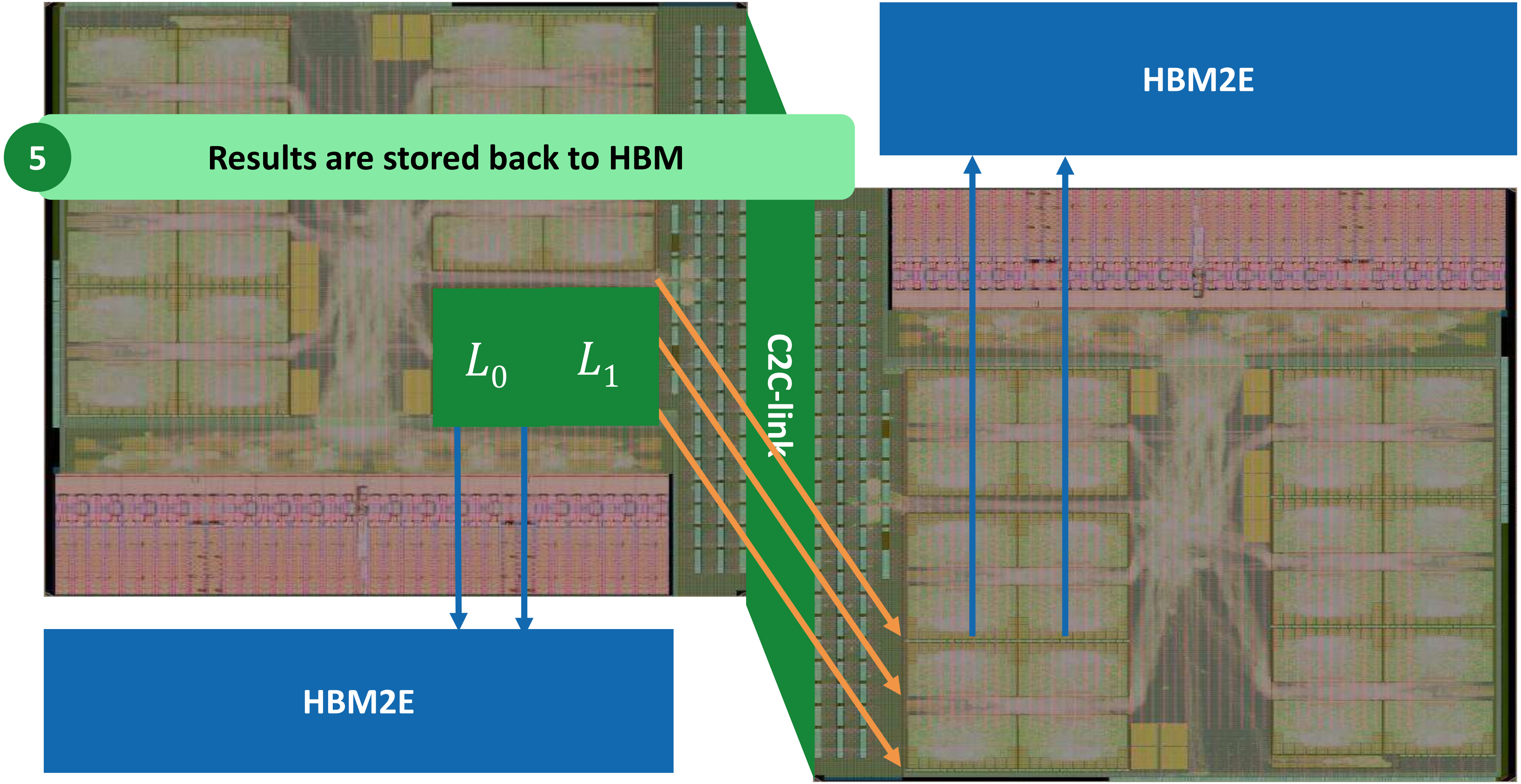
3 **Logarithmic-tree result reduction on-chiplet**

Linear Projection & Head Concatenation Fusion

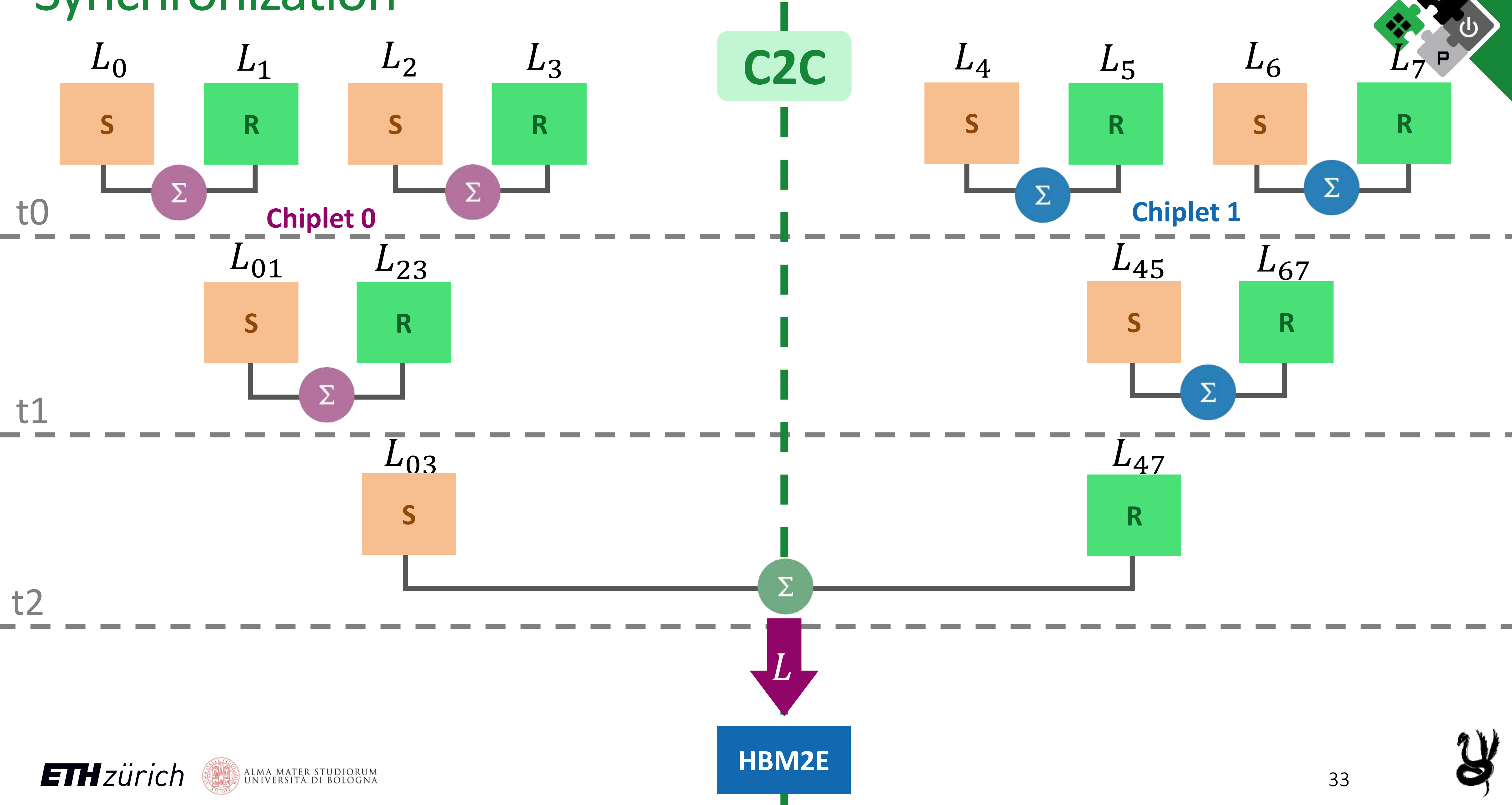


3 Logarithmic-tree result reduction on-chiplet

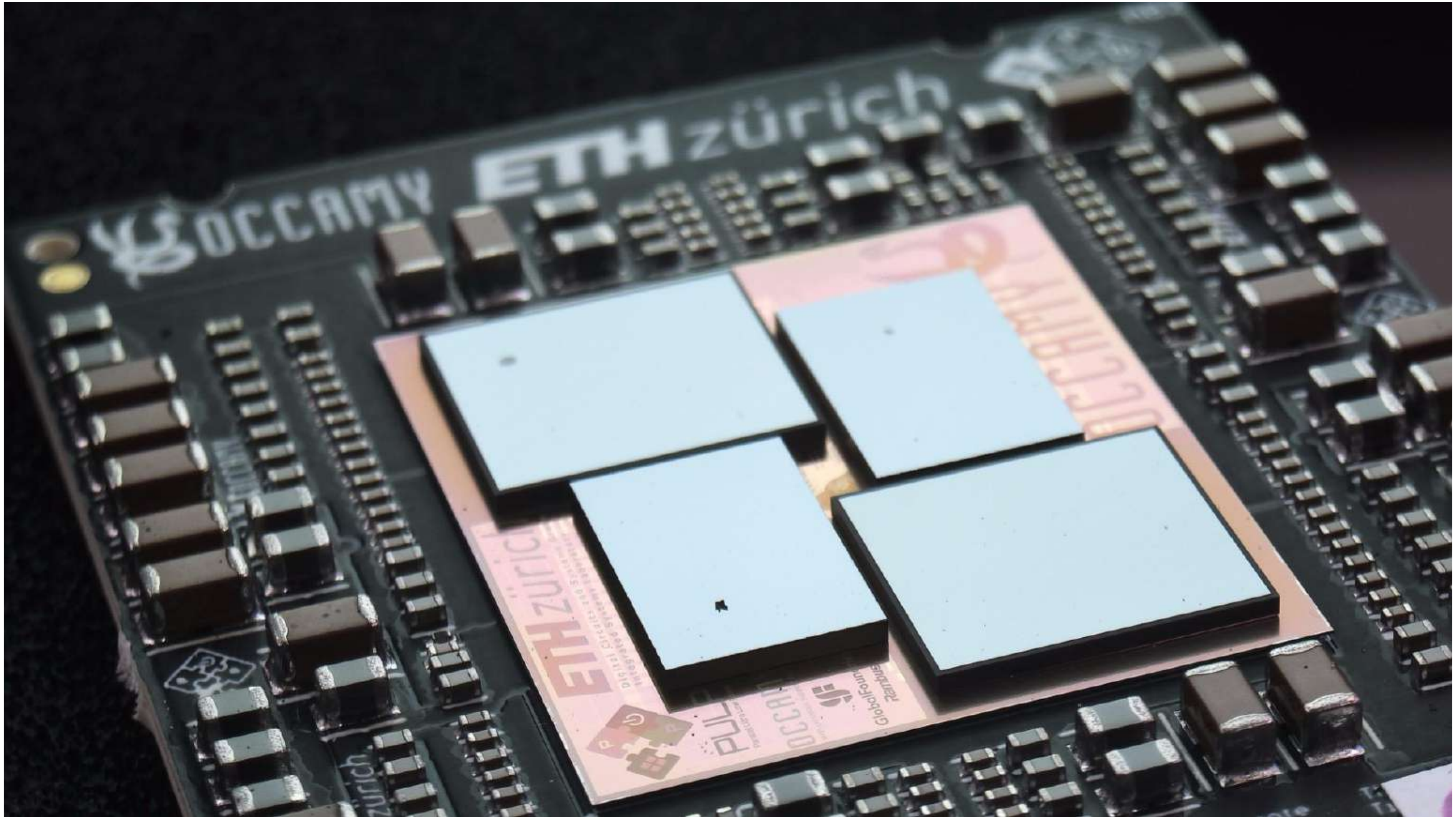
Linear Projection & Head Concatenation Fusion



Synchronization

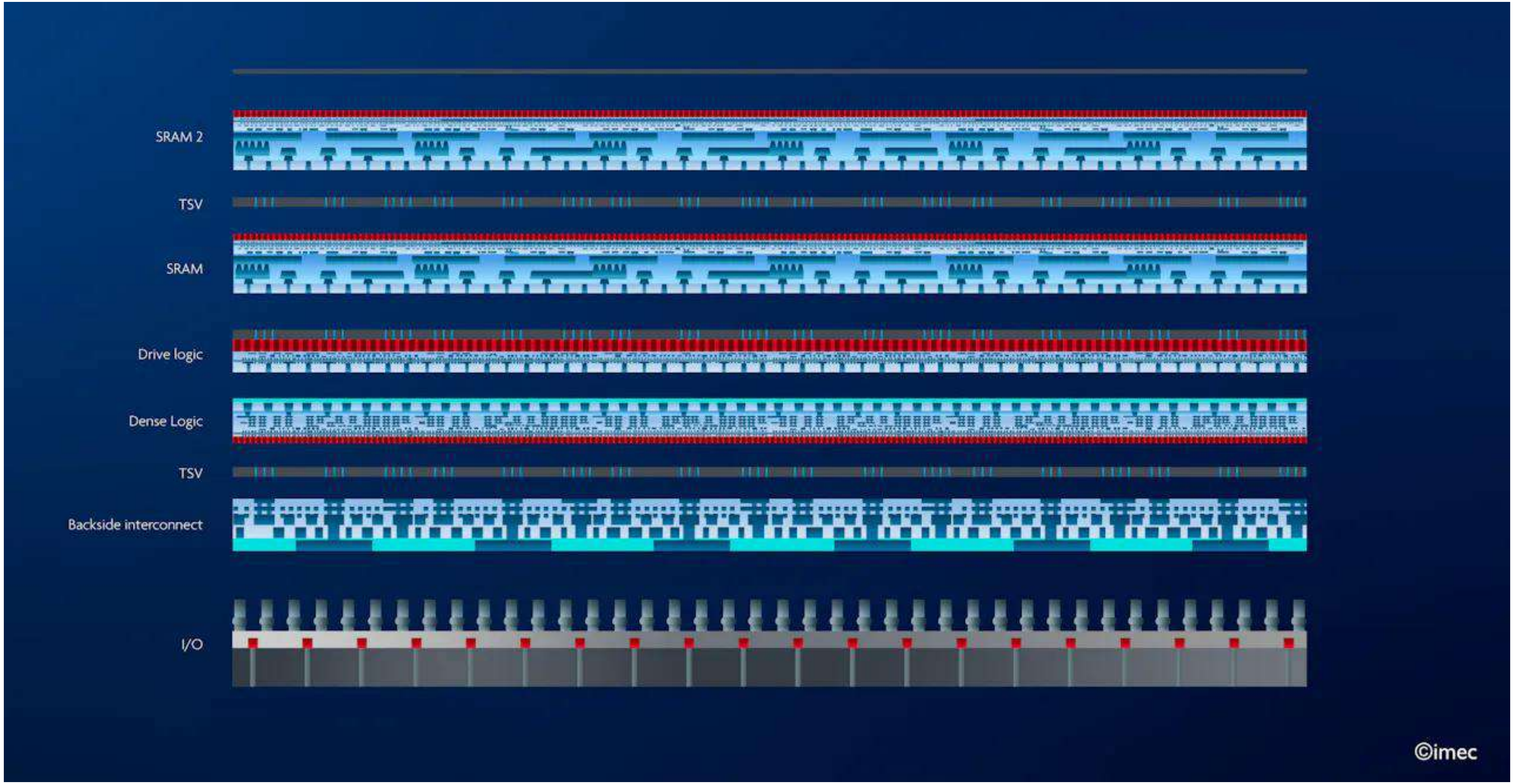


What next?



OCCAMY
ETH zürich
OCCAMY
ETH zürich
PULP
OCCAMY
GlobalFoun
Rambus

What next?



SRAM 2
TSV
SRAM
Drive logic
Dense Logic
TSV
Backside interconnect
I/O
©imec

Thank You!