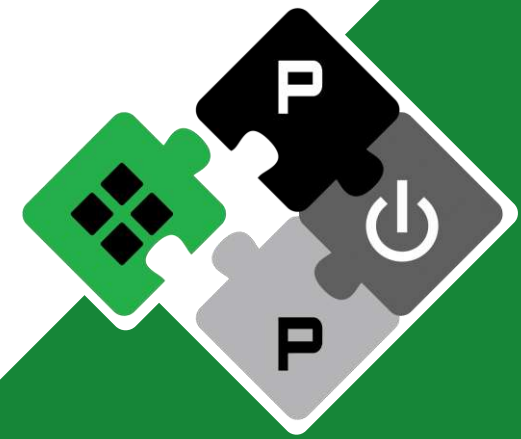
Open RISC-V Platforms for Energy-Efficient Scalable Computing

Luca Benini lbenini@ethz.ch, luca.Benini@unibo.it



@pulp_platform

pulp-platform.org

youtube.com/pulp_platform

PULP Platform

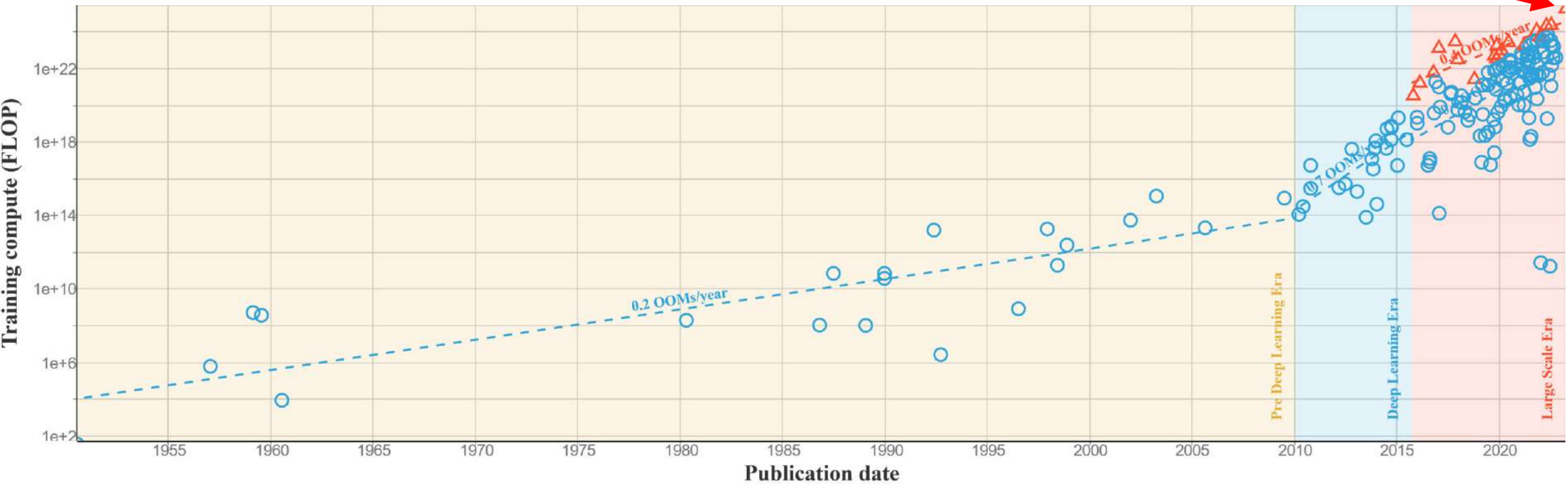
Open Source Hardware, the way it should be!

Computing is Power Bound: from the Cloud…

Largest datacenter <150MW

GPT-4 (OpenAI'23)
Training Compute: 2.1E+25 (FLOP)

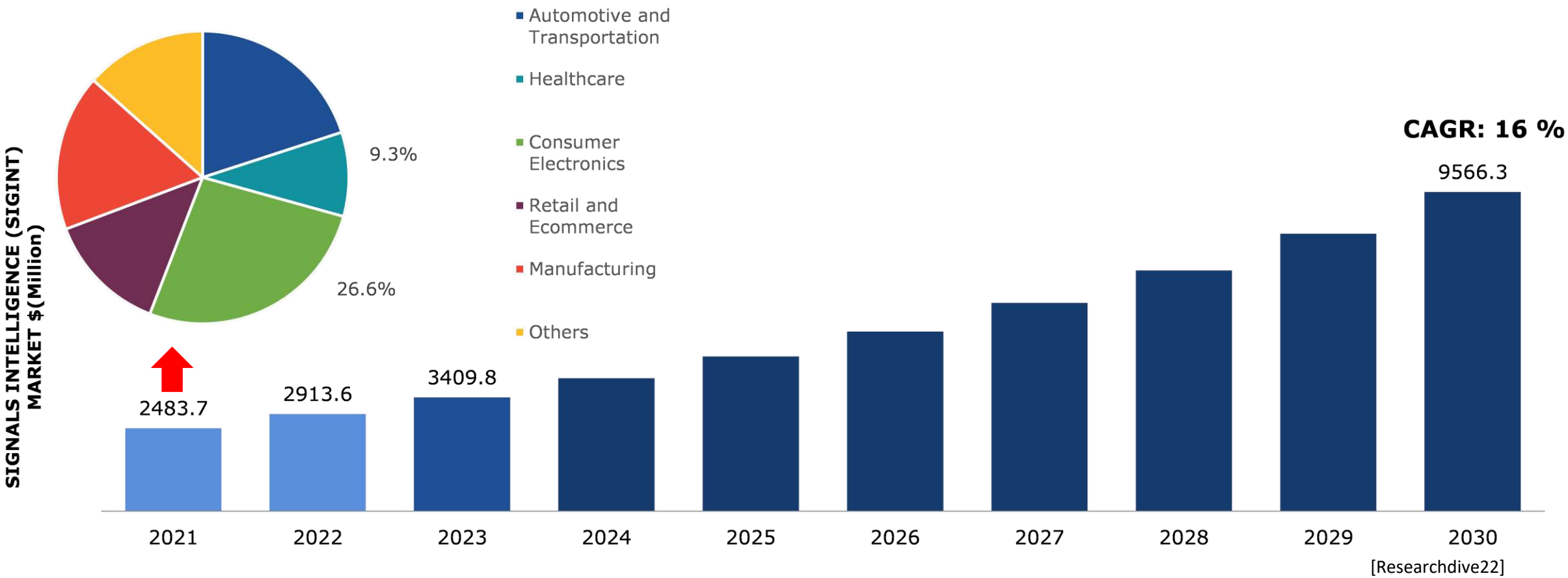
Sevilla 22: arXiv:2202.05924, epochai.org



Machine Learning: 10x every 2 years

…To the Edge

AI capabilities in the power envelope of an MCU: 10-mW peak (1mW avg)

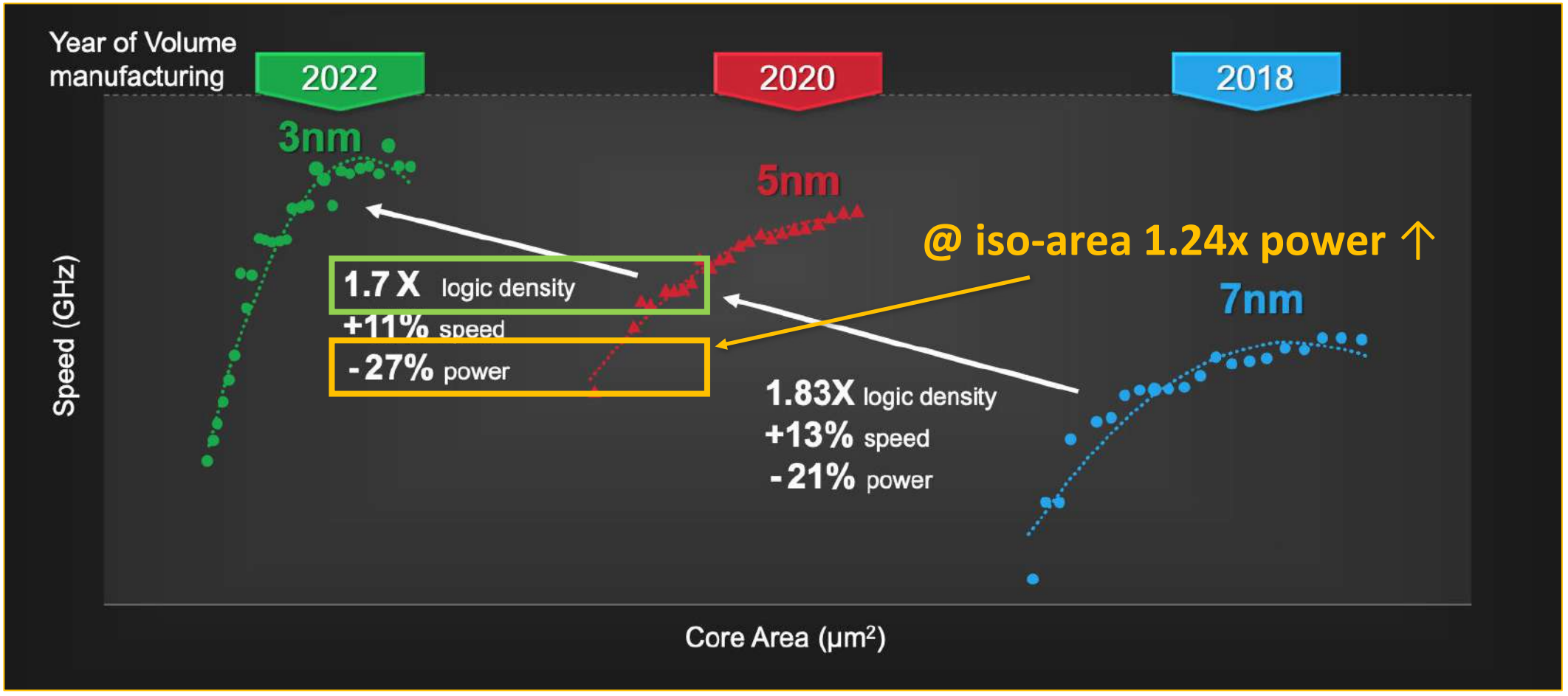


[Researchdive22]

Machine Learning: 10x every 2 years

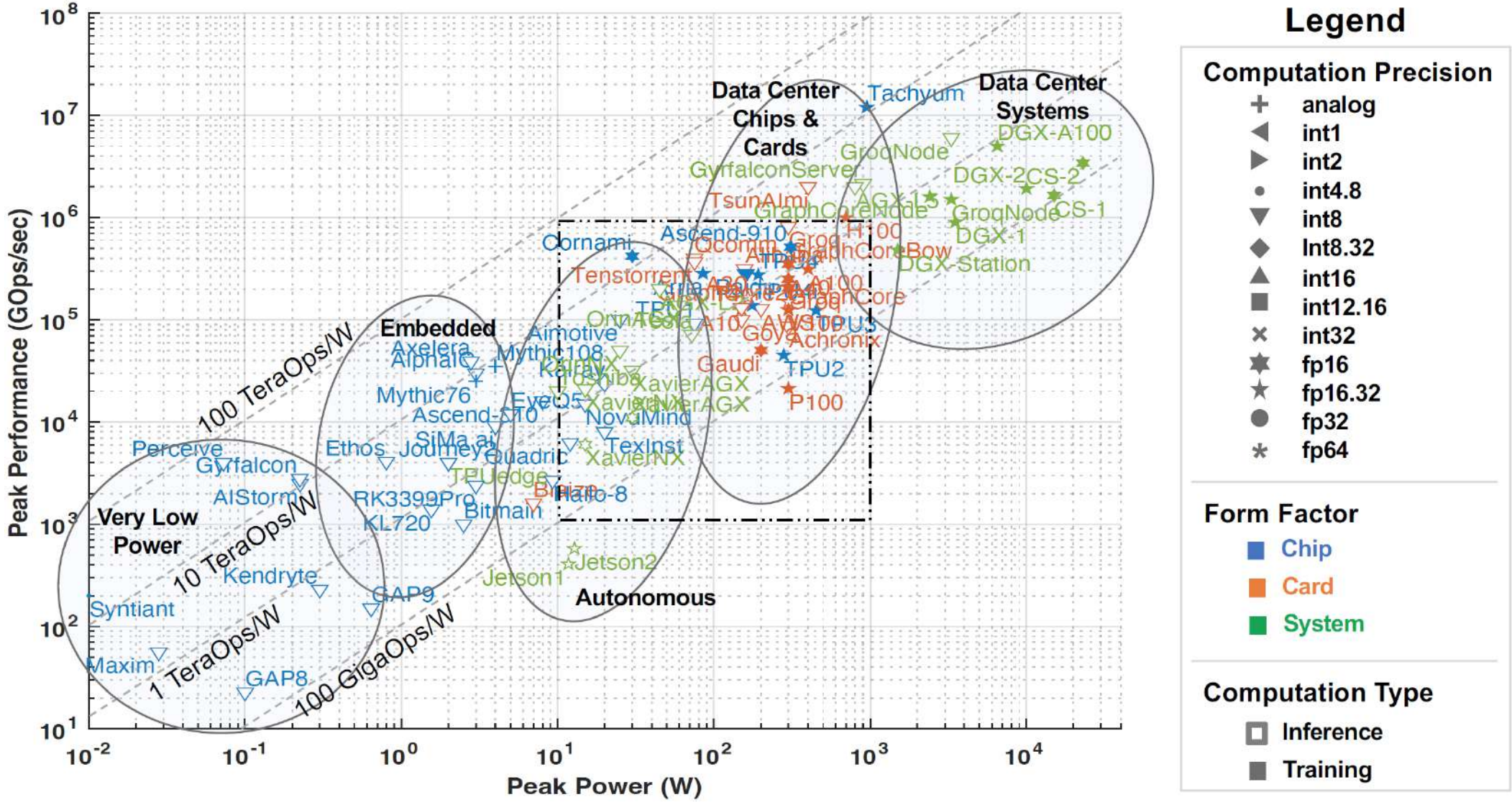
CMOS Scaling is definitely not enough

[TSMC, ISSCC21]



Energy Efficiency $\left(\frac{1}{\text{Power}\cdot\text{Time}}\right)$ → **10x every 12 years…**

Necessity is the Mother of Invention



Reuther 22: arXiv:2210.04055

Legend

Computation Precision

- analog
- int1
- int2
- int4.8
- int8
- Int8.32
- int16
- int12.16
- int32
- fp16
- fp16.32
- fp32
- fp64

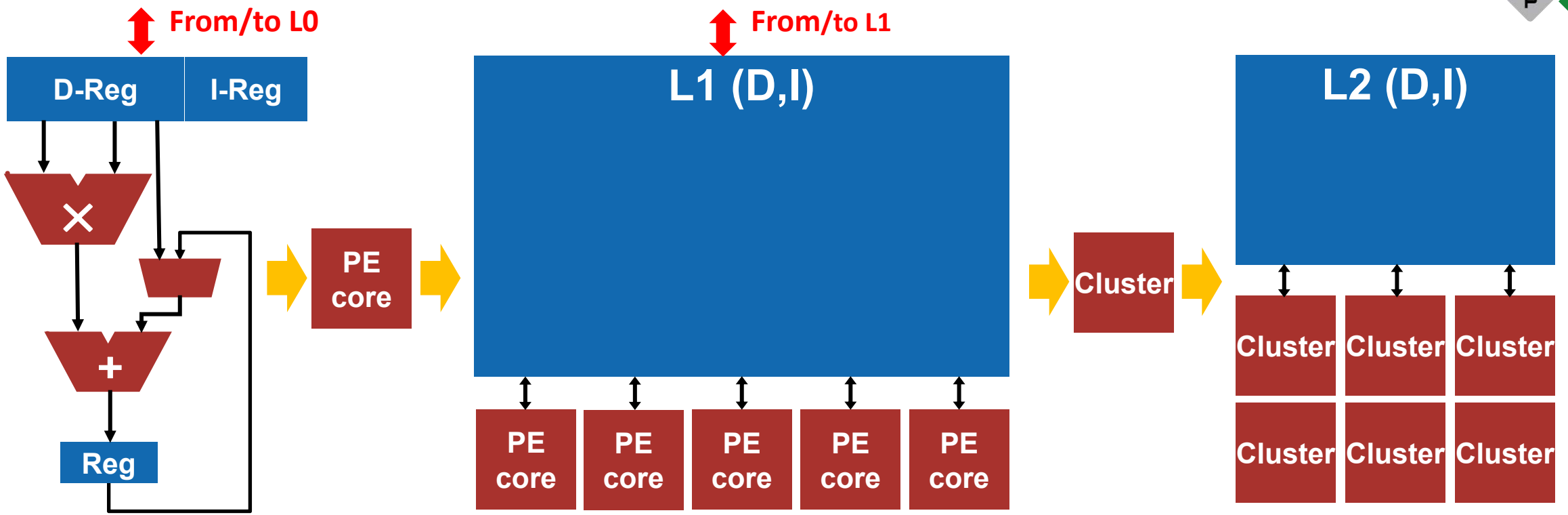
Form Factor

- Chip
- Card
- System

Computation Type

- Inference
- Training

Efficiency: compute & Move



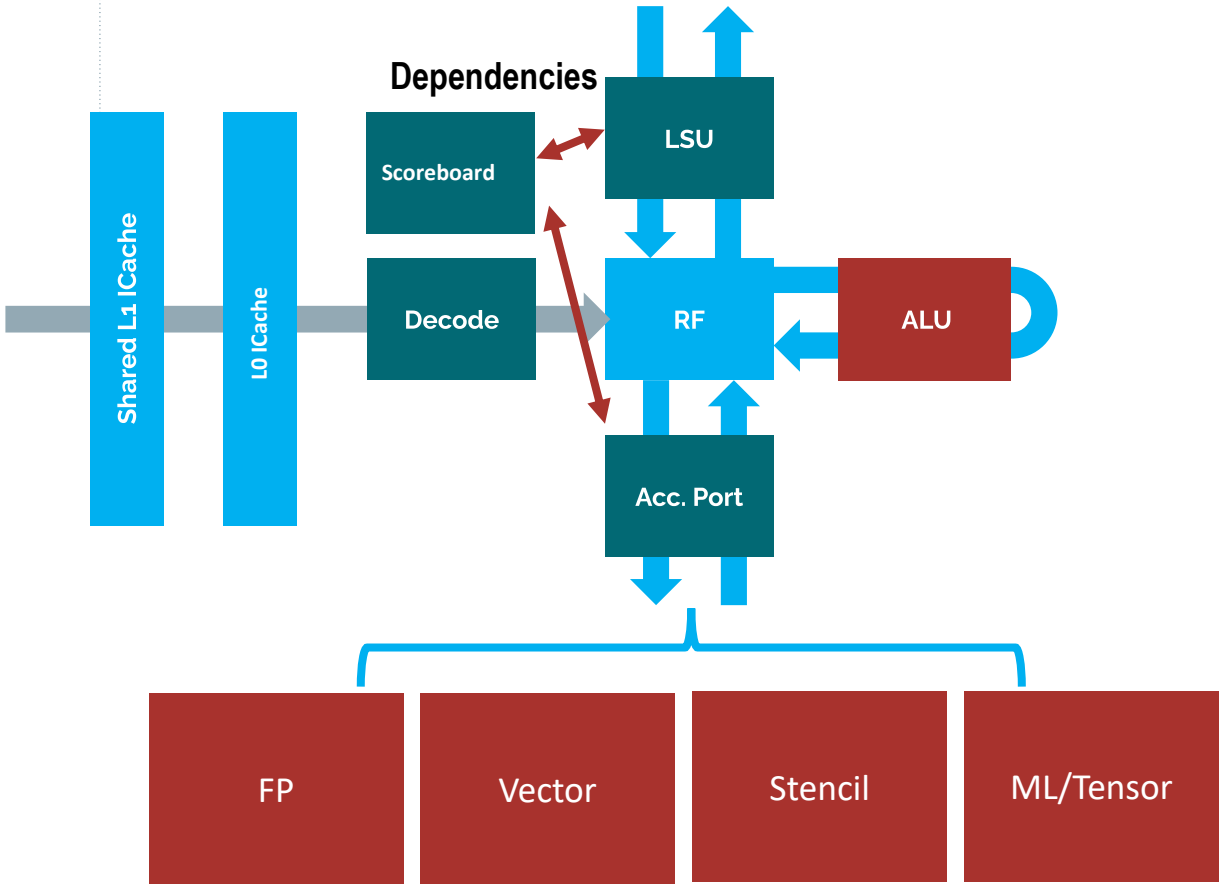
L0:Operand Memory
Latency=1
Density=1
Private

L1: Tightly Coupled DM
Latency<10
Density≈10
Shared

L2: Main Memory
Latency>100
Density≈100
Shared, Remote

PE: The power of ISA Extension

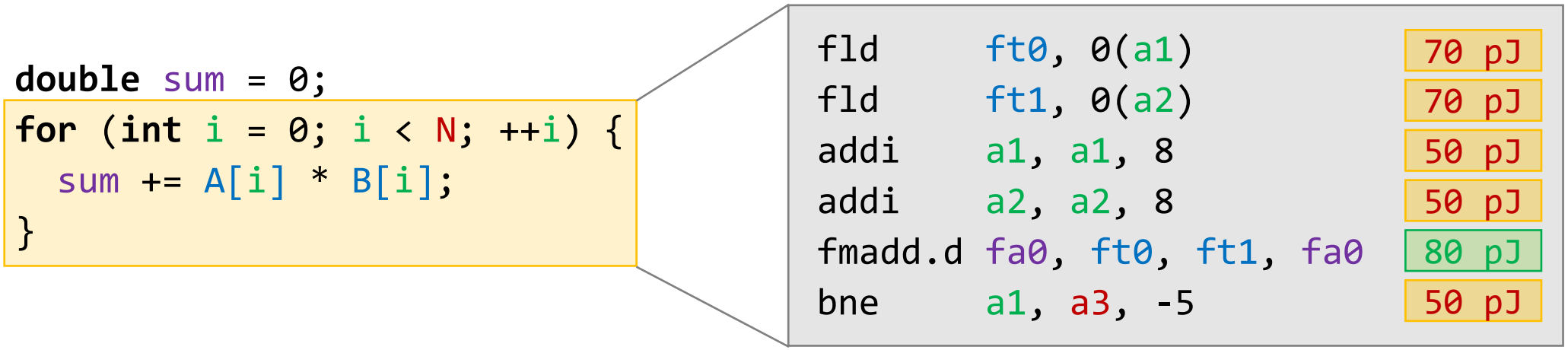
RISC-V® **ISA is extensible *by construction* (great!)**



- ***Snitch* core: around 20KGE**
  - Speed via simplicity (1GHZ+)
  - L0 Icache/buffer for low energy fetch
  - Shared L1 for instruction reuse (SPMD)
- **Extensible→ "Accelerator" port**
  - Minimal baseline ISA (RISC-V)
  - Extensibility: Performance through ISA extensions (via accelerator port)
- **Latency-tolerant →Scoreboard**
  - Tracks instruction dependencies
  - Much simpler than OOO support!

F. Zaruba, F. Schuiki, T. Hoefler and L. Benini, "Snitch: A Tiny Pseudo Dual-Issue Processor for Area and Energy Efficient Execution of Floating-Point Intensive Workloads," in *IEEE Transactions on Computers*, vol. 70, no. 11, pp. 1845-1860, 1 Nov. 2021

Snitch PE: ISA Extension for efficient “Compute”



Memory access, **operation**, iteration control – can we do better?

Note: memory access (>1 cycle even for L1) → need latency tolerance for LD/ST

Stream Semantic Registers & FREP

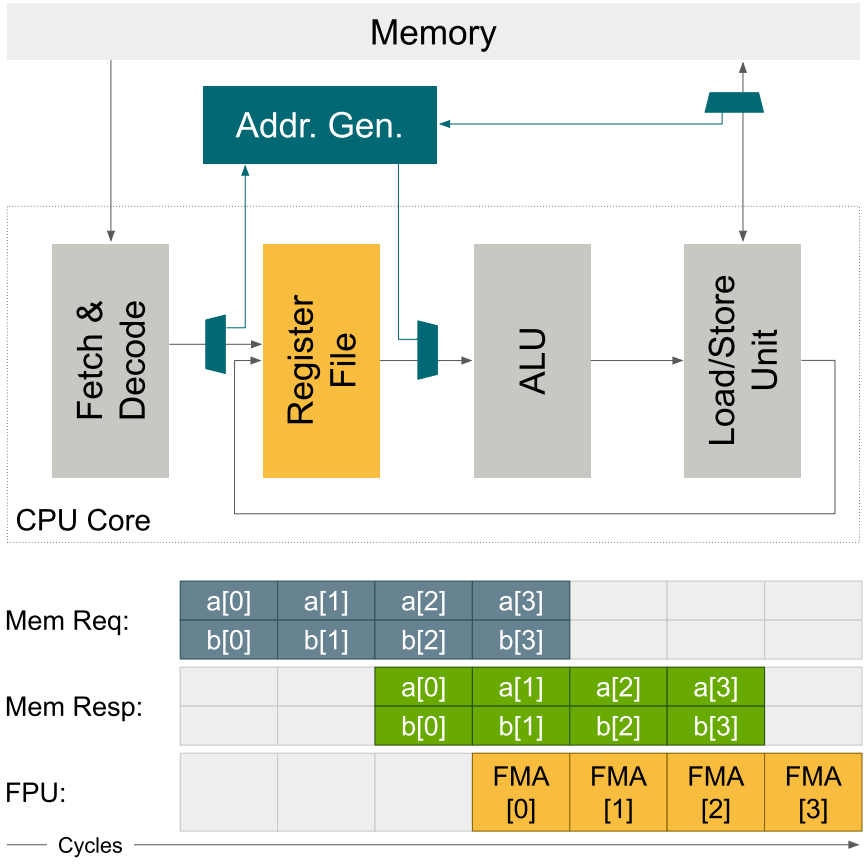
- Intuition: High FPU utilization ≈ high energy-efficiency
- Idea: Turn register read/writes into implicit memory loads/stores.
- Extension around the core's register file
- Address generation hardware

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

→

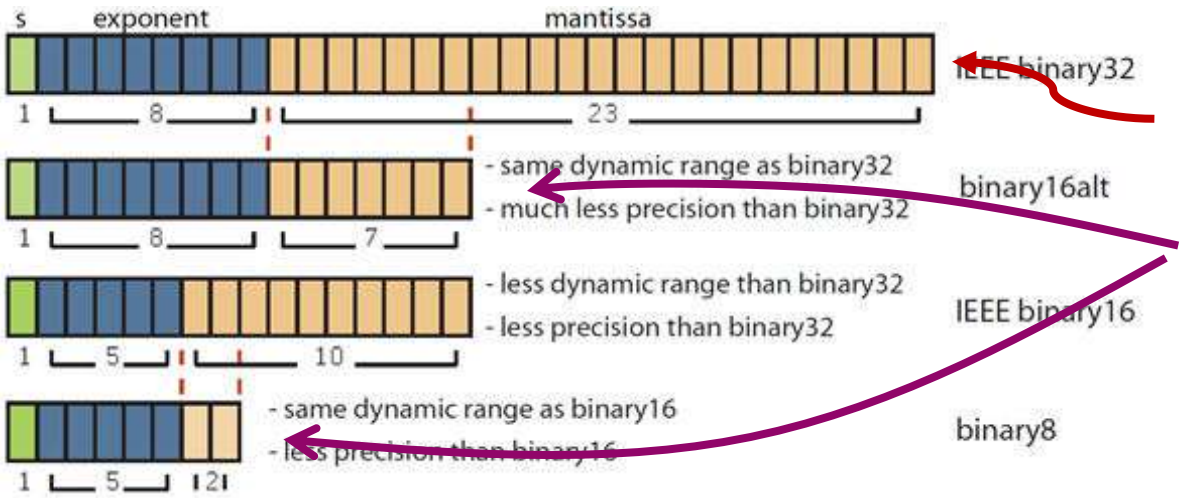
```
scfg 0, %[a], ldA
scfg 1, %[b], ldB
loop:
fmadd r2, ssr0, ssr1
```

- Increase FPU/ALU utilization by ~3x up to 100%
- SSRs ≠ memory operands
  - Perfect prefetching, latency-tolerant
  - 1-3 SSR (2-3KG/SSR)
- **FREP: floating point repetition buffer**



RISC-V ISA Extension for Target Workload

Efficient DNN inference & training



Inference ≠ Training Quantization

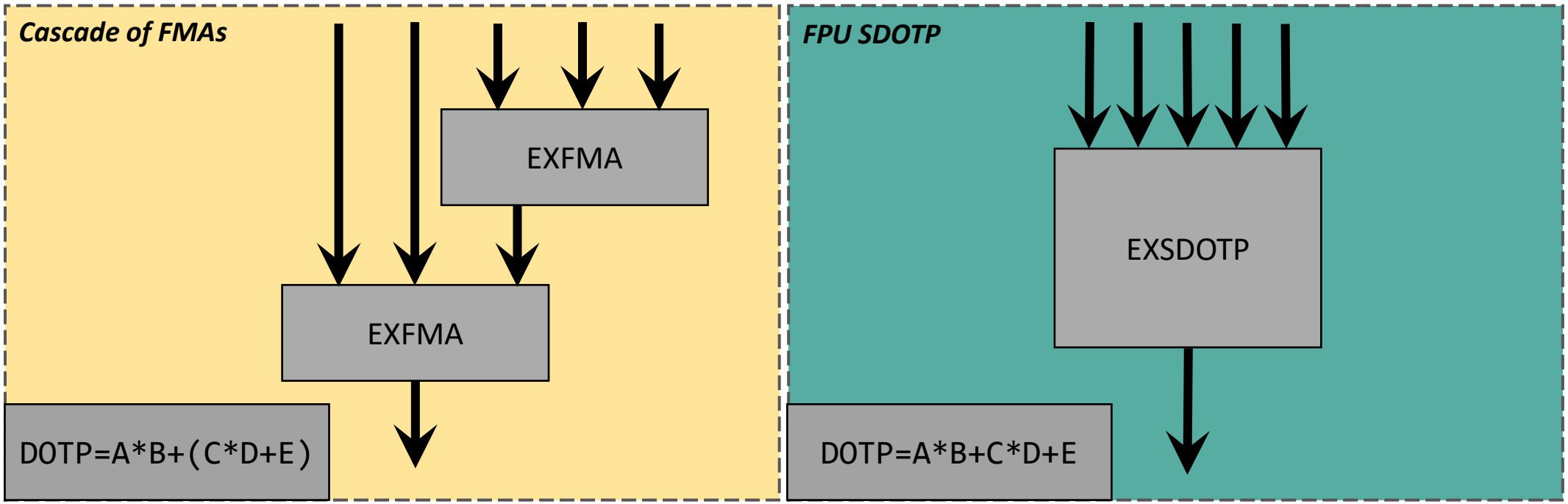
- Inference: INT8 quantization is SoA
- Training: High dynamic range needed for weights and weight updates

`fp32` is still standard for DNN training workloads. Low precision training with **`bf18`** and **`fp8`**

Support a wide variety of FP formats and instructions:

- Standard: **`fp64`**, **`fp32`**, **`fp16`**, **`bf16`**
- Low precision: **`fp8`**, **`altfp8`**
  - `fp8` (1-4-3): forward prop.
  - `altfp8` (1-5-2): backward prop.
- **Expanding ops needed e.g. accumulation**

Cascade of EXFMAs vs EXSDOTP

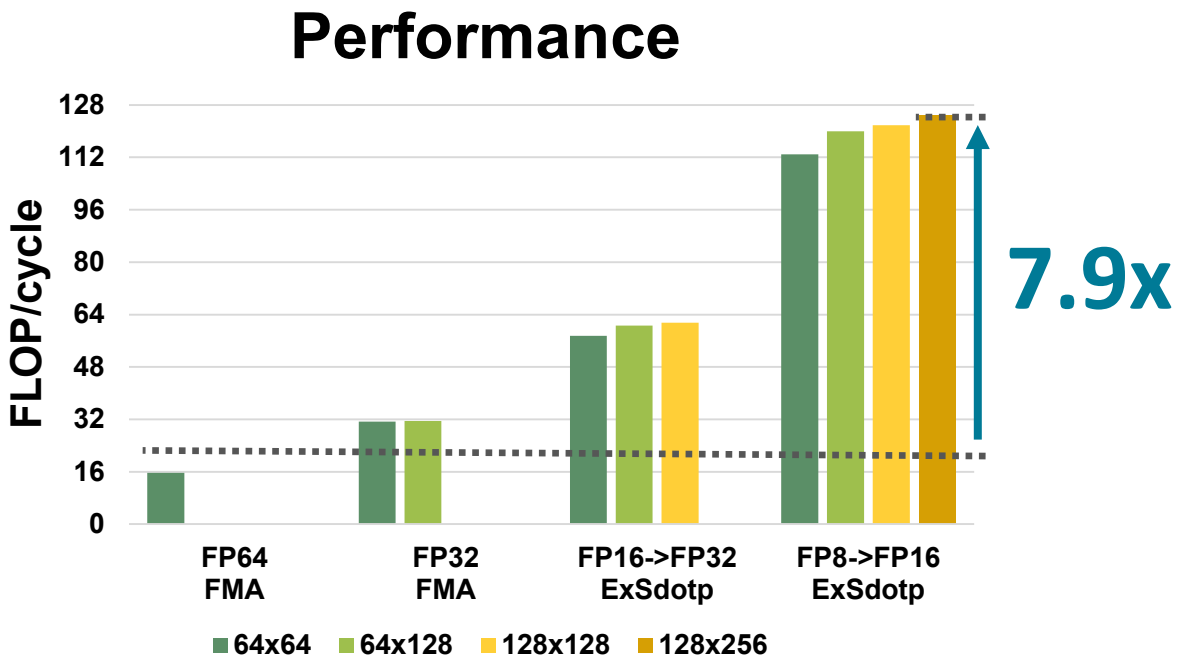


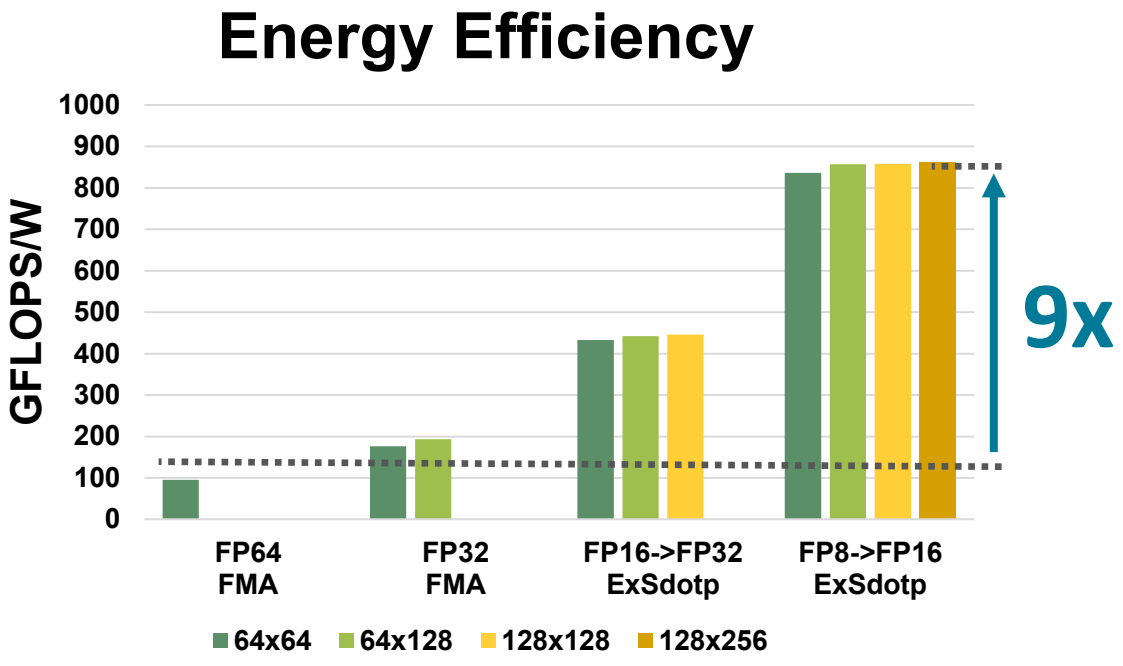
Non-distributive FP addition → **Precision Loss**

- **Fused** EXSDOTP (i.e. lossless)
- Single normalization and rounding step
- **Smaller** area and **shorter critical path**
- Product by-pass to compute **fused three-term addition** (vector inner sum)
- **Stochastic rounding** supported (+3% area)

ExSdotp Enables Performance & Efficiency Improvements

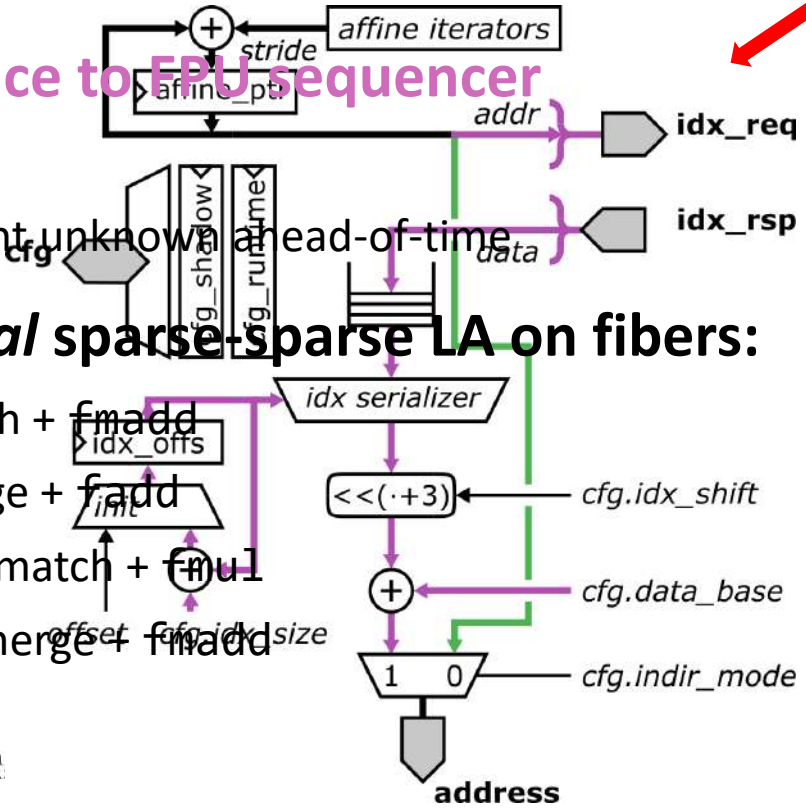
- **Snitch enhanced with multi-format, mixed-precision FPU**
- 12nm tech, 1 GHz (TT, 0.8V, 25°C)

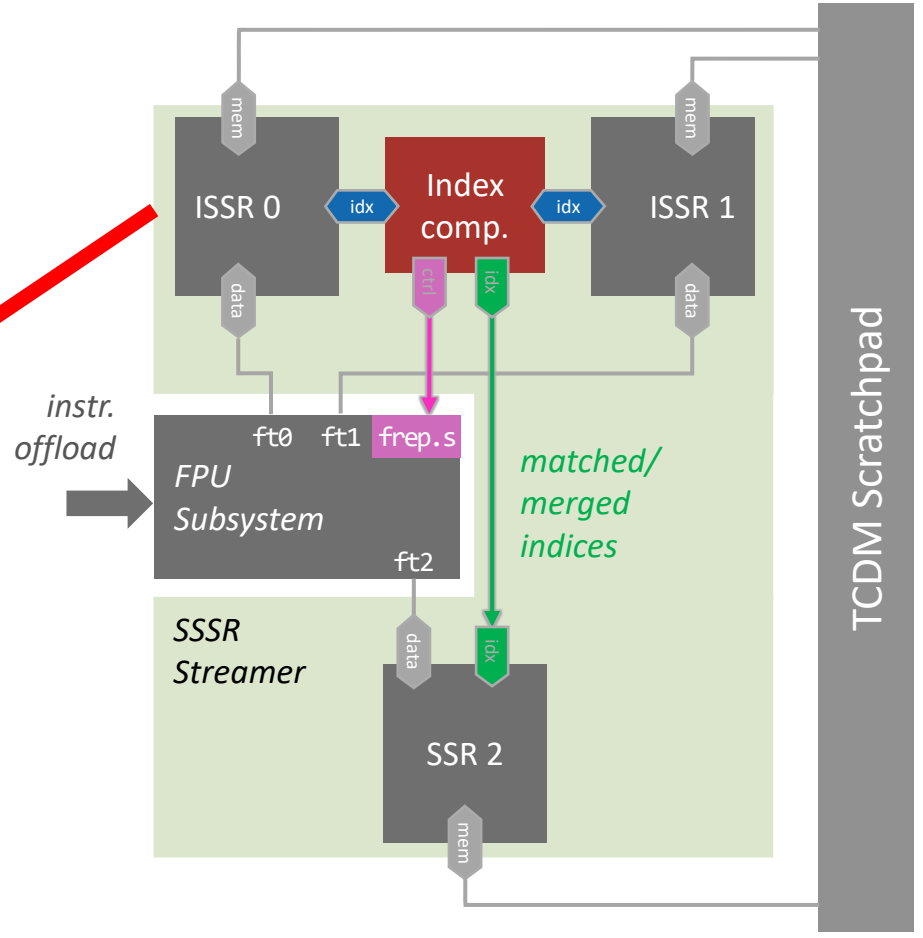




What About Sparsity? Indirect **SSR Streamer**

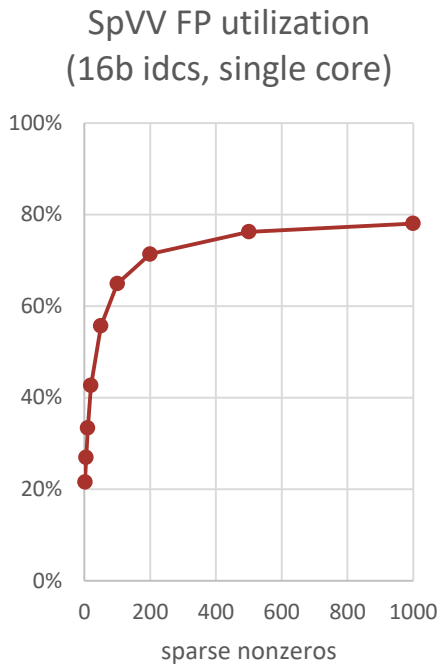
- **Based on existing 3-SSR streamer**
  1. Extend 2 SSRs to ISSRs
  2. Add *index comparison unit* between ISSRs
  3. Forward result indices to 3rd SSR
- **Control interface to FPU sequencer (`frep.s`)**
  - Result index count unknown ahead-of-time
- **Enables *general* sparse-sparse LA on fibers:**
  - *dotp*: index match + ~~fmadd~~
  - *vadd*: index merge + ~~fadd~~
  - *elem-mul*: index match + ~~fmul~~
  - *vec-mac*: index merge + ~~fmadd~~

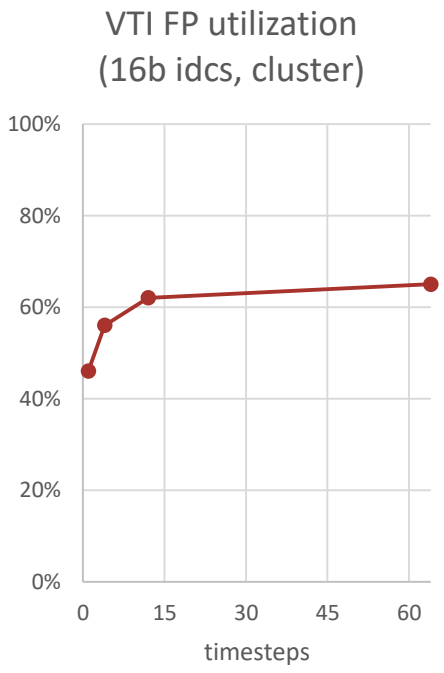




ISSR Performance Benefits

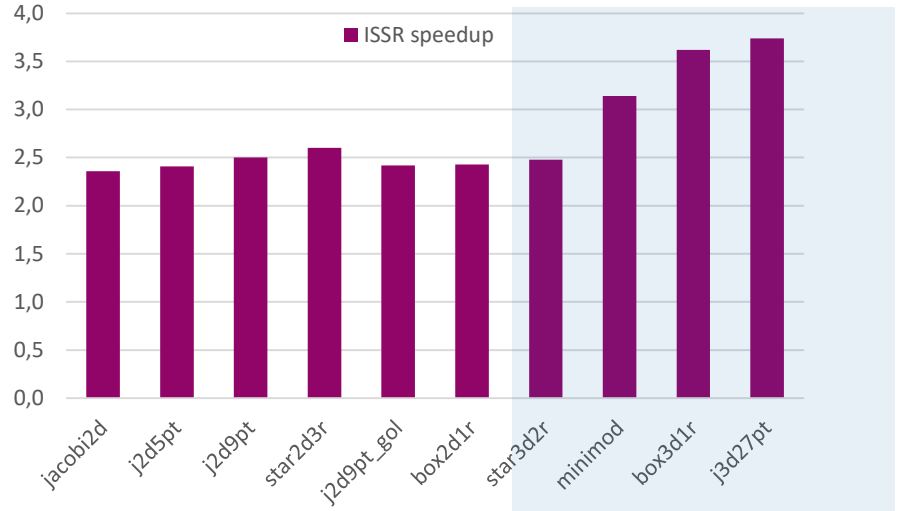
- **Notable single-core speedups over RV baseline**
  - *CsrMV*: up to **7.0**× faster, **79%** FP util.
  - *SpV+SpV*: up to **9.8**× faster / higher FP util.
  - *SpV·SpV*: up to **7.7**× faster / higher FP util.
  - *VTI (3D stencil code):* up to **2.9**× faster, **78%** FP util.
- **Significant benefits in multicore cluster:**
  - *CsrMV :* up to **5.0**× faster, **2.9x** less energy
  - *CsrMSpV :* up to **5.8**× faster, **3.0x** less energy
  - *VTI:* up to **2.7**× faster
- **Notably higher peak FP utilizations than SoA CPUs (69×), GPUs (2.8×) on CsrMV**

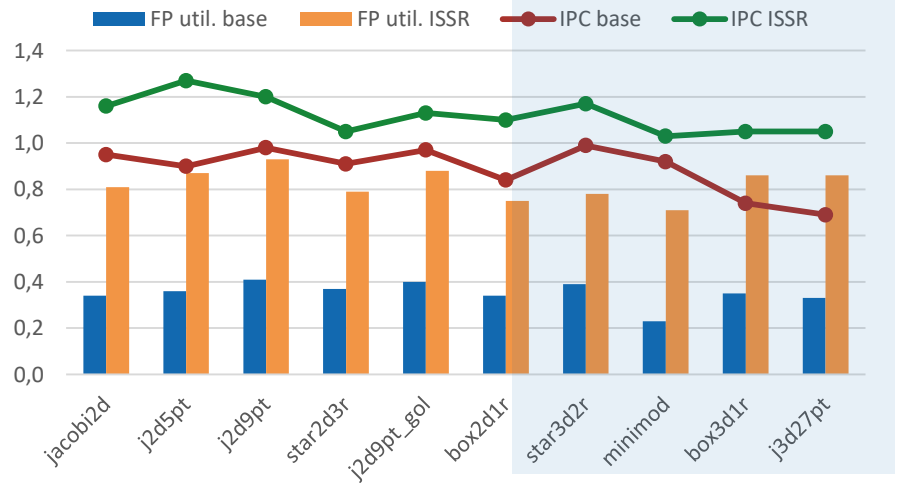




ISSR Performance on Stencils

- Various 2D/3D stencils on 8-worker-core cluster
  - FP64, $64^2/16^3$ grid chunks, up to 4× unroll
  - Tuned LLVM RV32G baseline vs ISSR-enhanced kernels
- Geomean **2.7**× speedups, **82%** FP utilization
  - ISSR **IPC consistently >1** as ISSRs enable pseudo-dual-issue
- Baseline perf. *degrades* for large (3D) stencils
  - Cannot maintain unroll *and* keep reusable inner-loop data in register file
  - ISSR streams **avoid this bottleneck**: 2.5× 2D → 3.2× 3D geomean speedup





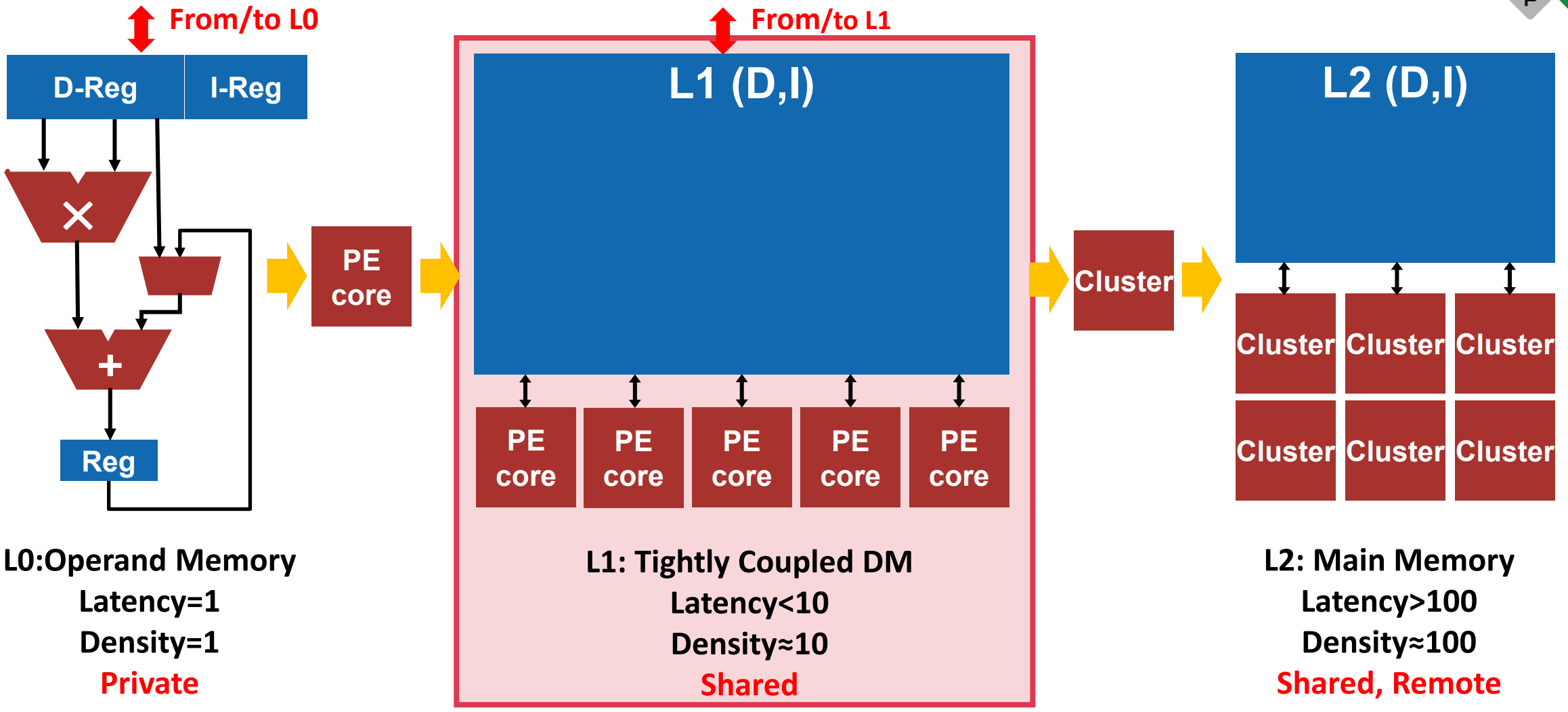
Efficient PE (snitch) architecture in perspective

1. **Minimize control overhead → Simple, shallow pipelines**
2. **Reduce VNB → amortize IF: SSR-FREP + SIMD (Vector processing)**
3. **Hide memory latency → non-blocking (indexed) LD/ST+dependency tracking**
4. **Highly expressive, domain-specific instruction extensions (thanks, RISC-V!)**

European Processor Initiative

THE EUPILOT

Compute Efficiency: the Cluster (PEs + On-chip TCDM)



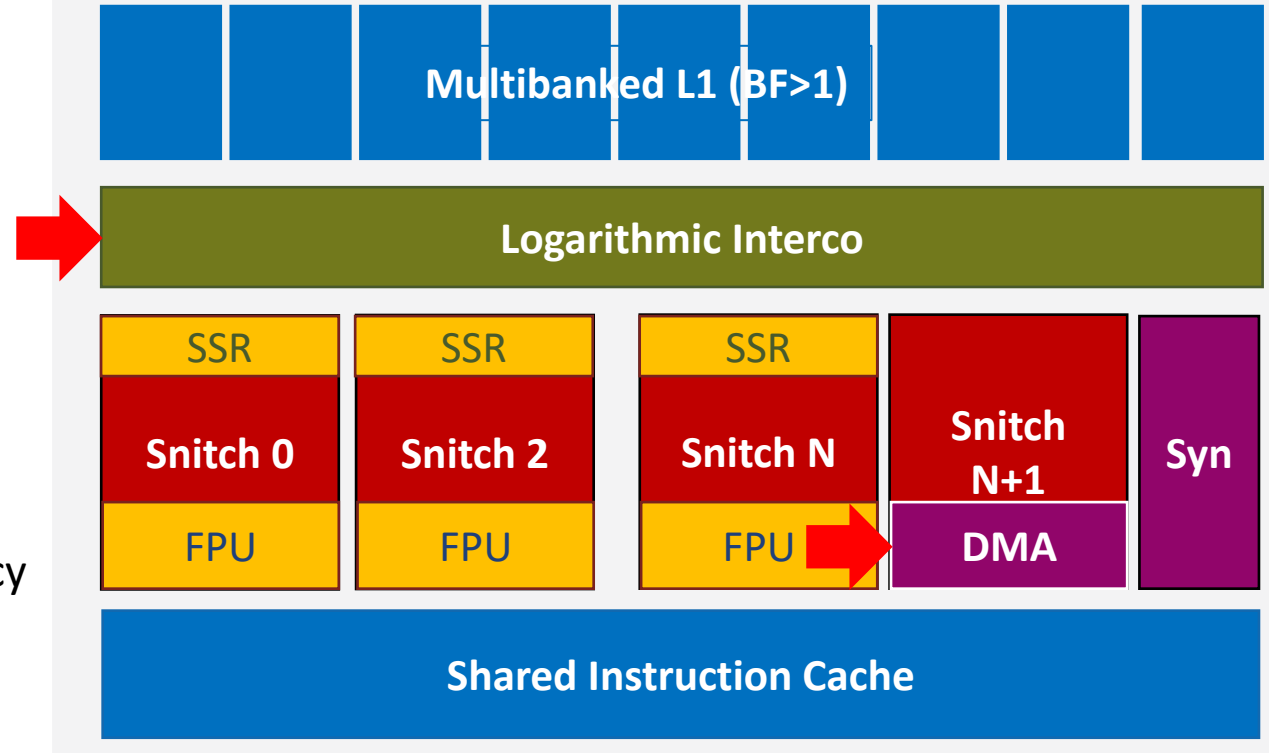
L0:Operand Memory
Latency=1
Density=1
Private

L1: Tightly Coupled DM
Latency<10
Density≈10
Shared

L2: Main Memory
Latency>100
Density≈100
Shared, Remote

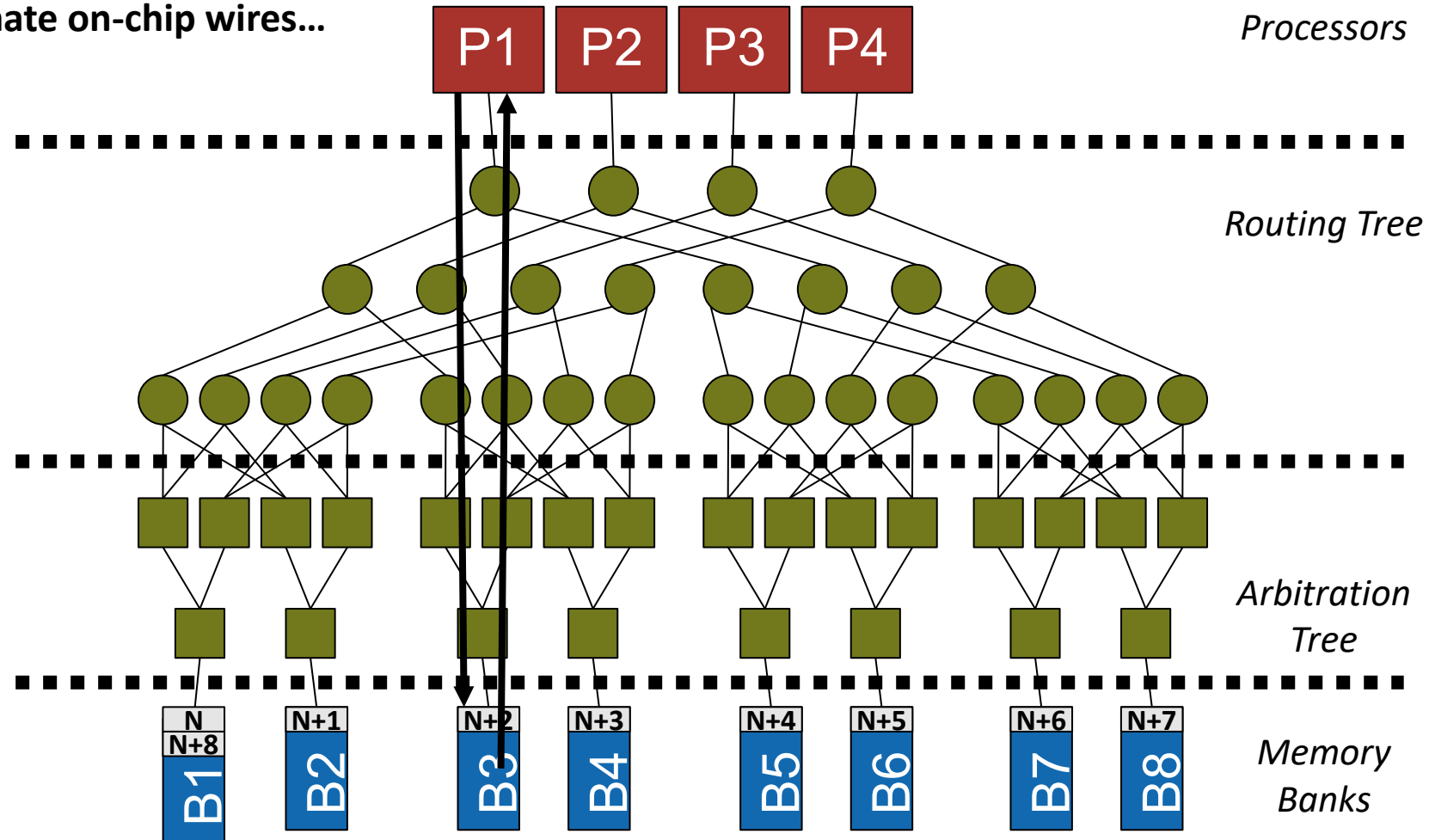
Snitch Cluster Architecture

- **Efficient PE**
  - Hide L1 TCDM “residual” latency
  - RV + Domain-specifi ISA extensions
- **Low latency access TCDM**
  - Multi-banked architecture
  - Fast logarithmic interconnect
- **DMA for data movement**
  - Double buffered copy in/out to hide L2 latency
- **Fast synchronization**
  - Atomics
  - Barriers



High speed logarithmic interconnect

Do not underestimate on-chip wires…



World-level bank interleaving «emulates» multiported mem

@1GHz, 8-16 PEs, Latency: 2 cycles + stalls for banking conflicts

Efficient *Explicit* Global Data Mover

- 512-bit AXI DMA – double-buffered transfers
- Tightly coupled with Snitch (<10 cycles configuration)
- Operates on wide 512-bit data-bus
- Hardware support to copy 2-4-dim shapes
- Higher-dimensionality handled by SW
- Intrinsics/library for easy programming

```
// setup and start a 1D transfer, return transfer ID
uint32_t __builtin_sdma_start_oned(
  uint64_t src, uint64_t dst, uint32_t size, uint32_t cfg);
// setup and start a 2D transfer, return transfer ID
uint32_t __builtin_sdma_start_twod(
  uint64_t src, uint64_t dst, uint32_t size,
  uint32_t sstrd, uint32_t dstrd, uint32_t nreps, uint32_t cfg);
// return status of transfer ID tid
uint32_t __builtin_sdma_stat(uint32_t tid);
// wait for DMA to be idle (no transfers ongoing)
void __builtin_sdma_wait_for_idle(void);
```

Where does the Energy go?

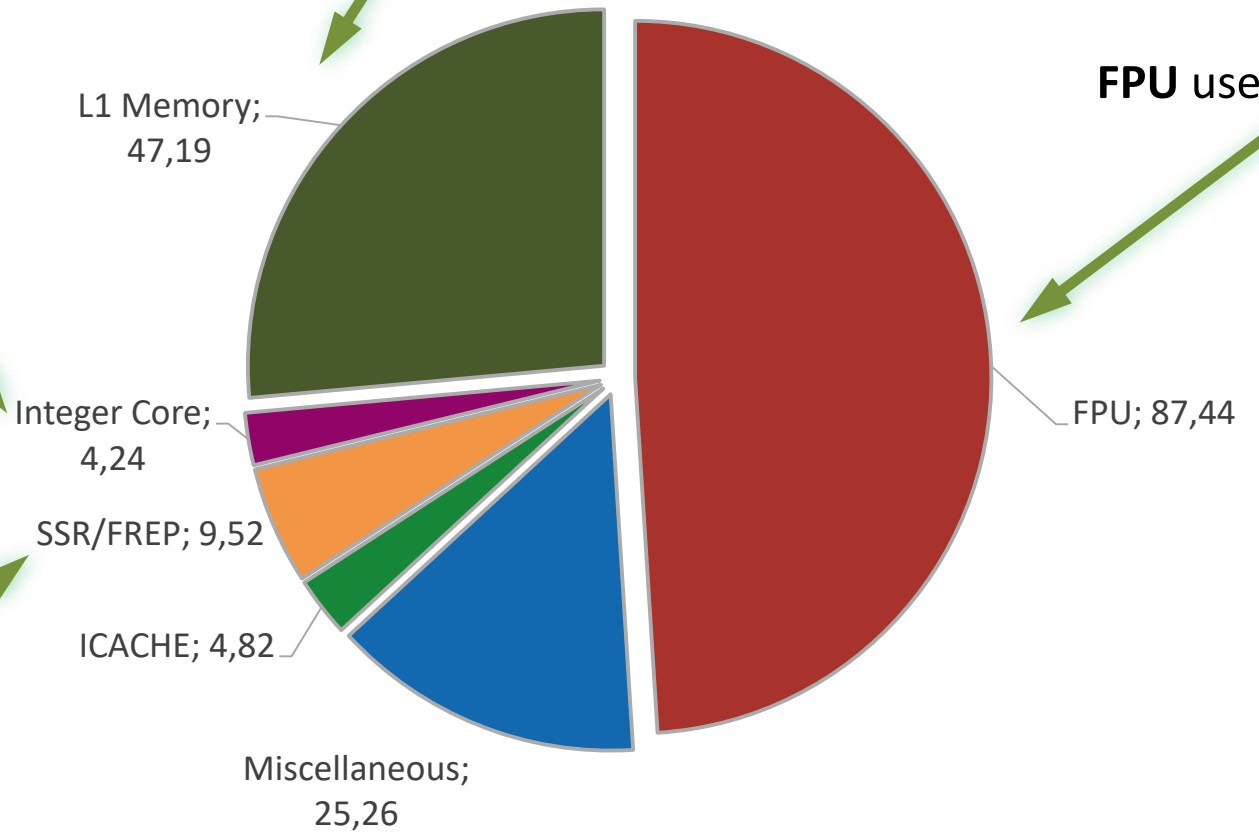
In an 8-core cluster

Inevitable to have local memory
(e.g., GPU/GPU L1 cache, vector register file)

FPU uses **50%** of power

Integer core uses
2% of power

SSR/FREP hardware
uses **5%** of power

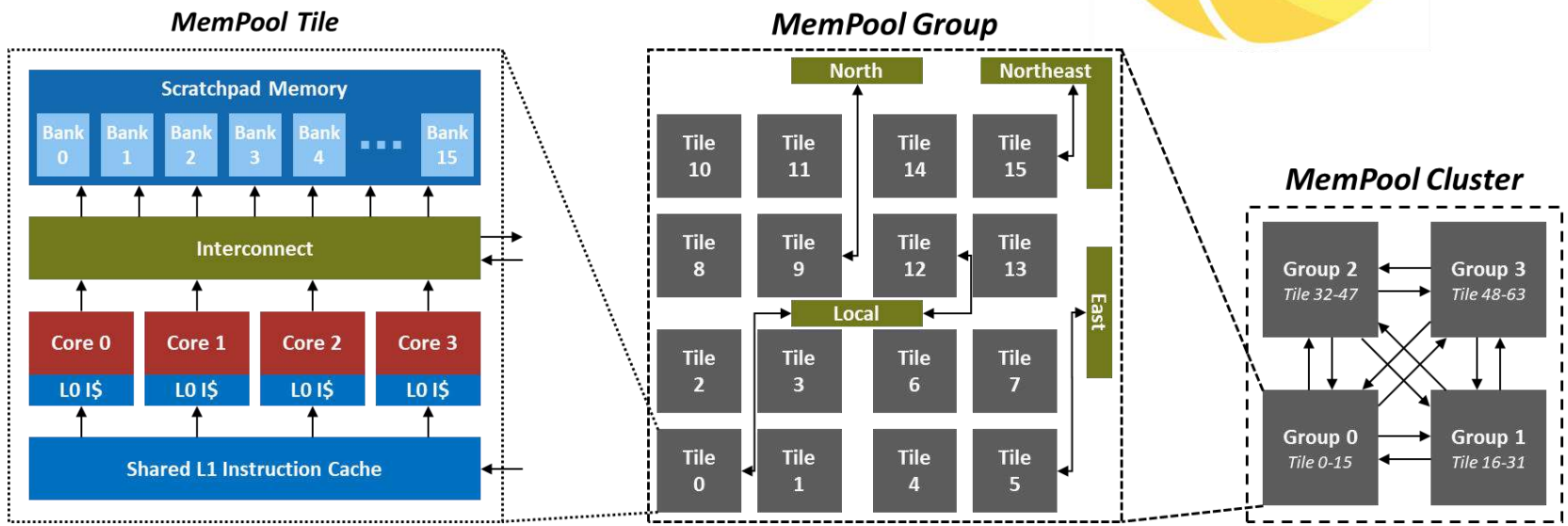


Spending energy where it contributes to the result → **High Efficiency**

Can we make it Bigger?

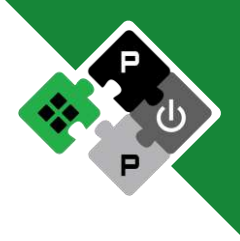
- **Why?**
  - Better global latency tolerance if $L1_{size} > 2*L2_{latency}*L2_{bandwidth}$ (Little's law + double buffer)
  - Easier to program (data-parallel, functional pipeline…)
  - Smaller data partitioning overhead
- **A large "MemPool"**
  - 256+ cores
  - 1+ MiB of shared L1 data memory
  - ≤ 10 cycle latency (Snitch can handle it)
- **Physical-aware design**
  - WC Frequency > 500 Mhz
  - Targeting iso-frequency with small cluster

MemPool



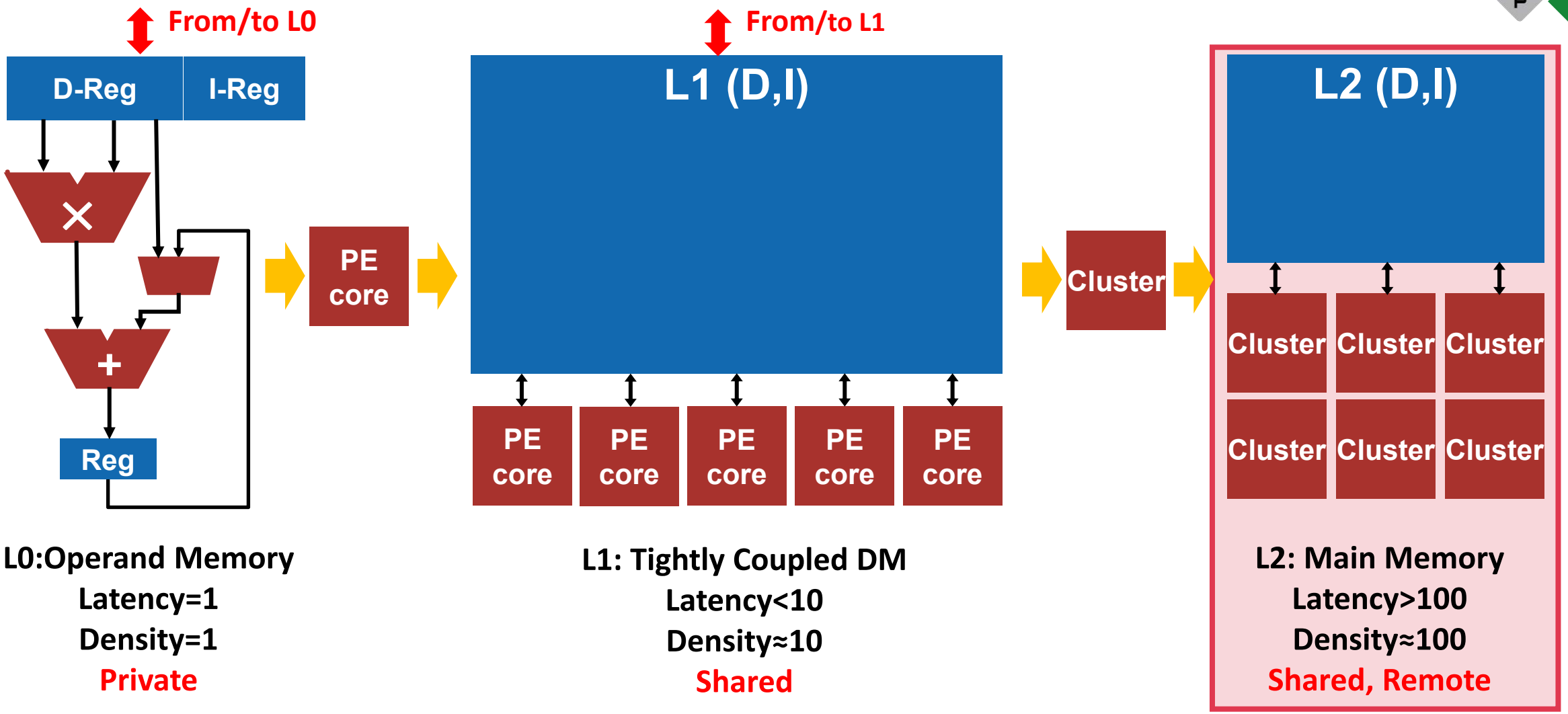
Butterfly Multi-stage Interconnect 0.3req/core/cycle, 5 cycles

Efficient Cluster architecture in perspective



1. **Memory pool – efficient sharing of L1 memory**
2. **Fast and parsimonious synchronization**
3. **Data Mover + Double buffering – explicitly managed block transfers at the boundary**
4. **More cores and more memory per cluster for more latency tolerance!**

Compute Efficiency: the Chip(let) (Clusters+Off-die Mem)

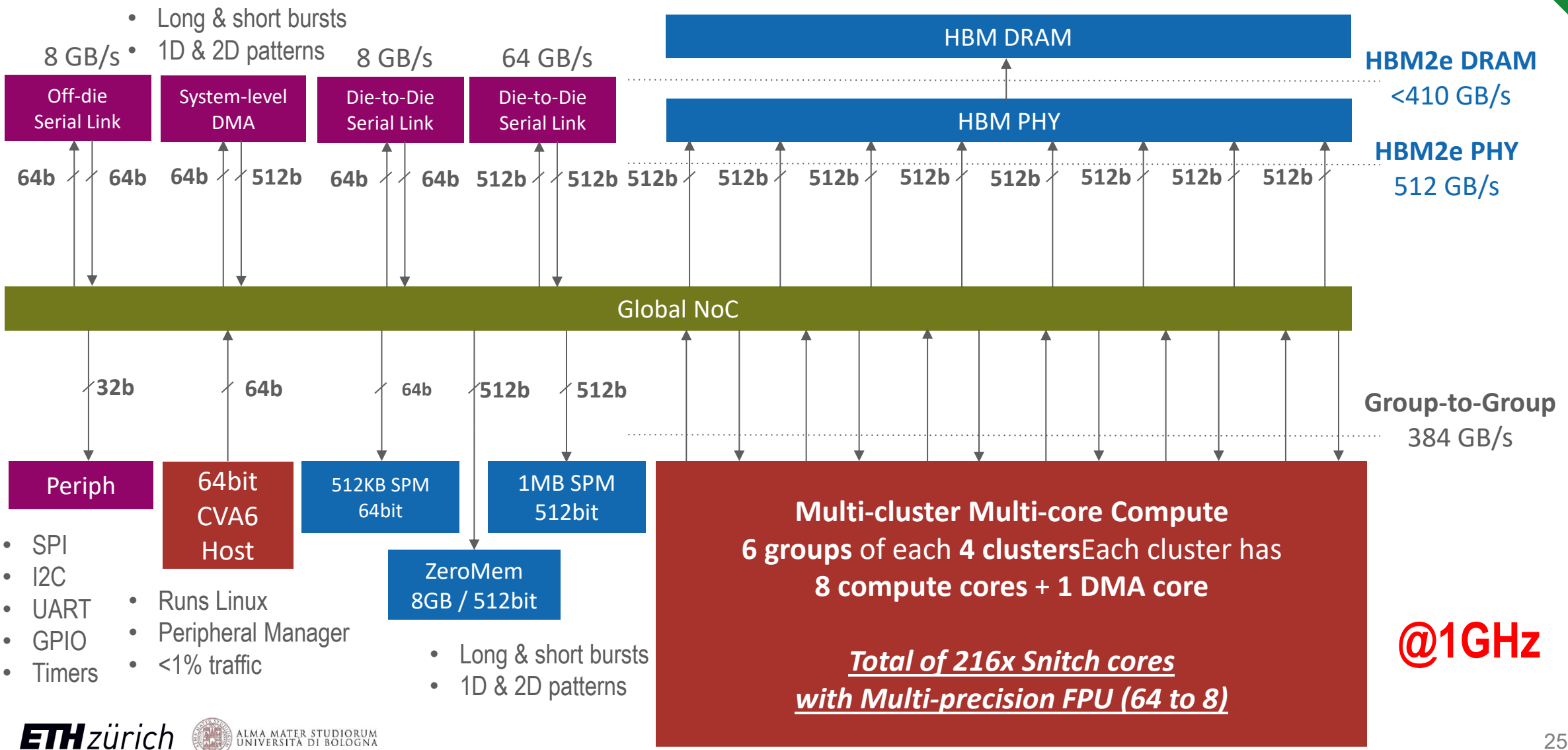


L0:Operand Memory
Latency=1
Density=1
Private

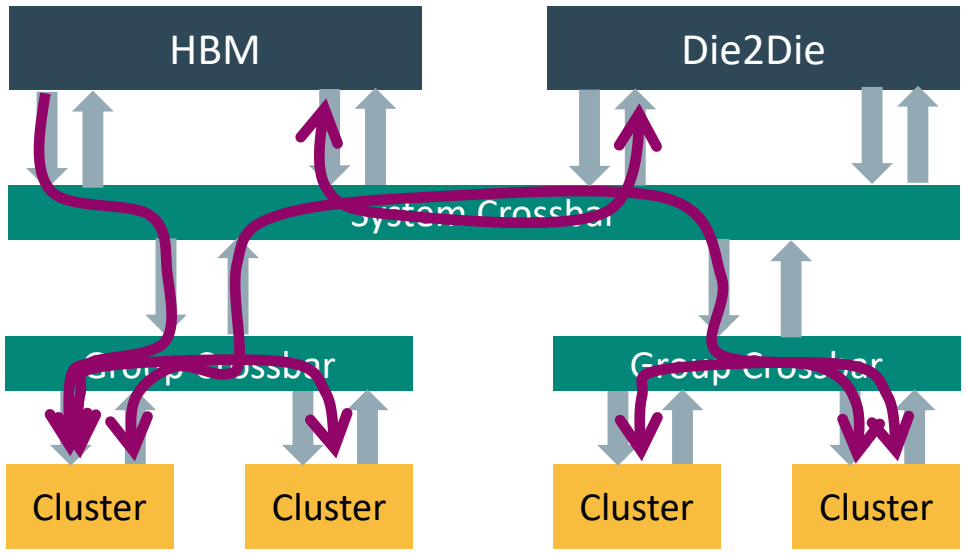
L1: Tightly Coupled DM
Latency<10
Density≈10
Shared

L2: Main Memory
Latency>100
Density≈100
Shared, Remote

Occamy: RISC-V goes HPC Chiplet!



NoC: 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!

High-Performance, General-Purpose

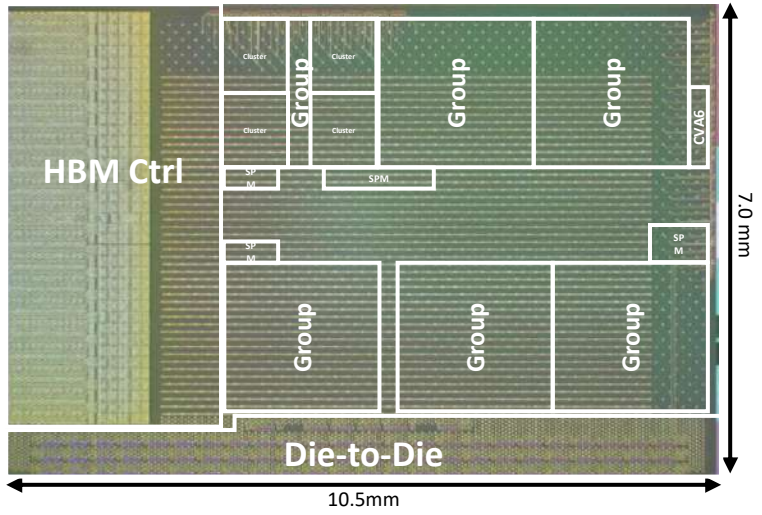
*Our **scalable architecture** is **general-purpose** and **high-performance***

Peak chiplet performance @1GHz:

- FP64: 384 GFLOp/s
- FP32: 768 GFLOp/s
- FP16: 1.536 TFLOp/s
- FP8: 3.072 TFLOp/s

Preliminary measured results:

- Dense Kernels:
  - GEMMS: ≥ 80% FPU utilization (also for SIMD MiniFloat)
  - Conv2d: ≥ 75% PFU utilization (also for SIMD MiniFloat)
- Stencils Kernels: ≤ **60%** FPU utilization
- Sparse Kernels: ≤ **50%** FPU utilization

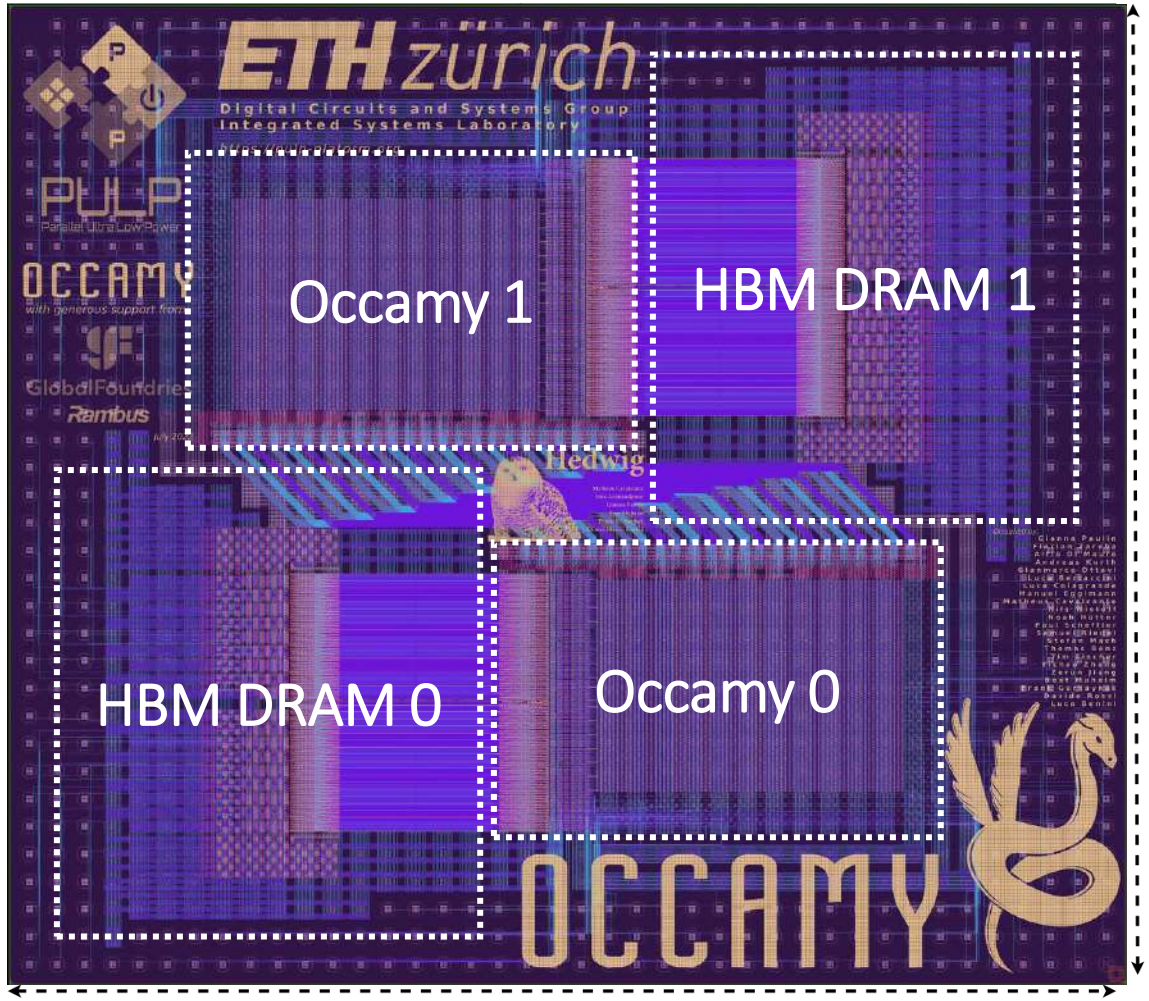


Chiplet taped out: 1st July 22

Silicon Interposer: Hedwig (65nm, passive, GF)

Taped out: 15th of October 2022

- **Interlocked die arrangement**
  - Prevent bending, increase stability
- **Compact die arrangement**
  - No *dummy dies* or *stitching* needed
- **Fairly low I/O pin count due to no high-bandwidth periphery**
  - Off-package connectivity: ~200 wires
  - Array of **40 x 35 (-1) C4s** (total of 1'399 C4 bumps)
    - Diameter: 400µm, Pitch: 650µm
- **Die-to-Die: ~600 wires**
- **HBM: ~1700 wires**



ETH zürich ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

Approaching 1T(DP)-FLOP

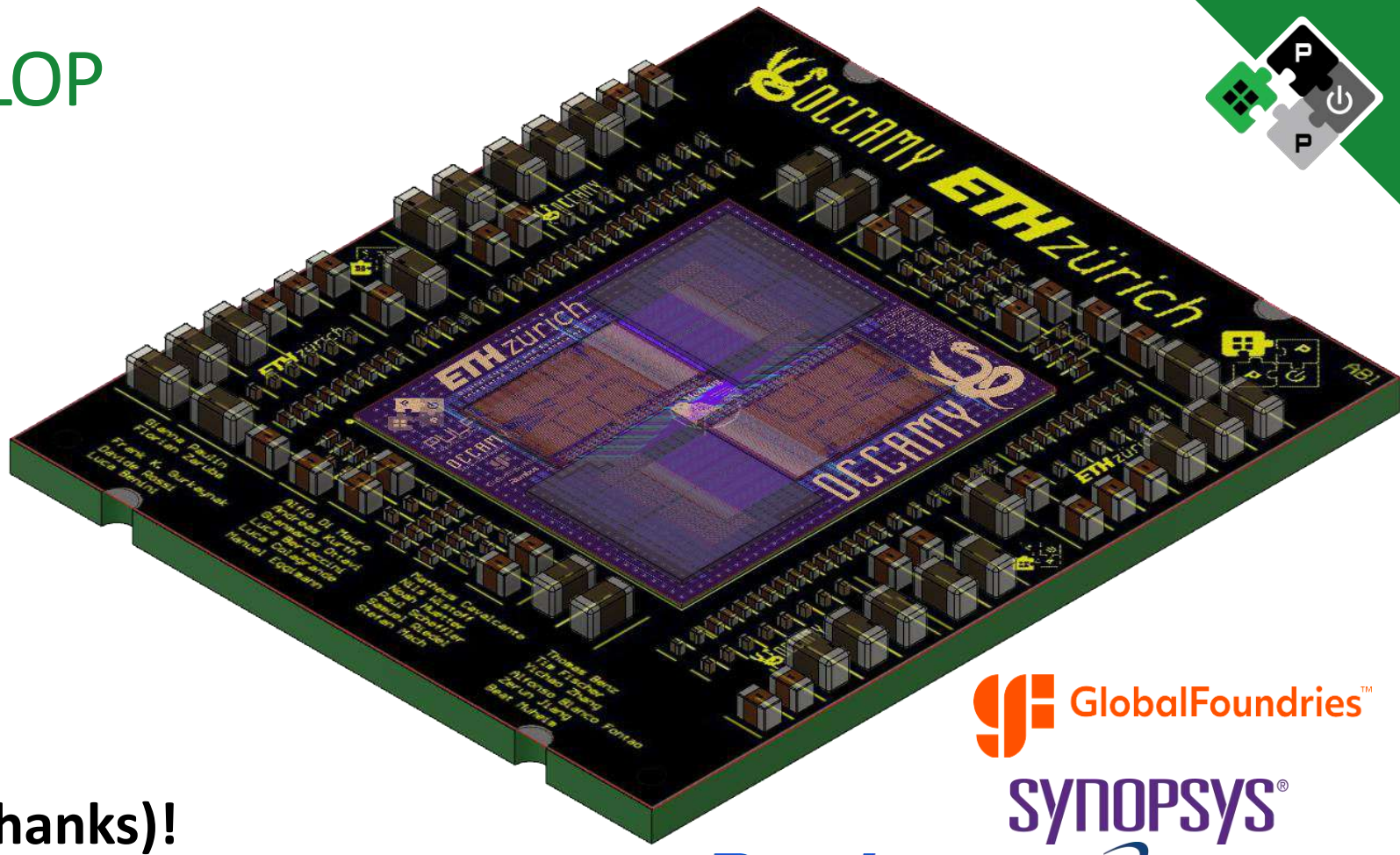
Dual Chiplet System Occamy:

- >430+ RV Cores
- 0.8 T DP-FLOP/s (no overclocking)
- 32GB of HBM2e DRAM
- Low tens of W (est.)

Aggressive 2.5D Integration

Carrier PCB:

- RO4350B (Low-CTE, high stability)
- 52.5mm x 45mm



Industry partners are key (thanks)!

GlobalFoundries™ SYNOPSYS® Rambus Micron® EUROPRACTICE IC SERVICE Fraunhofer IZM

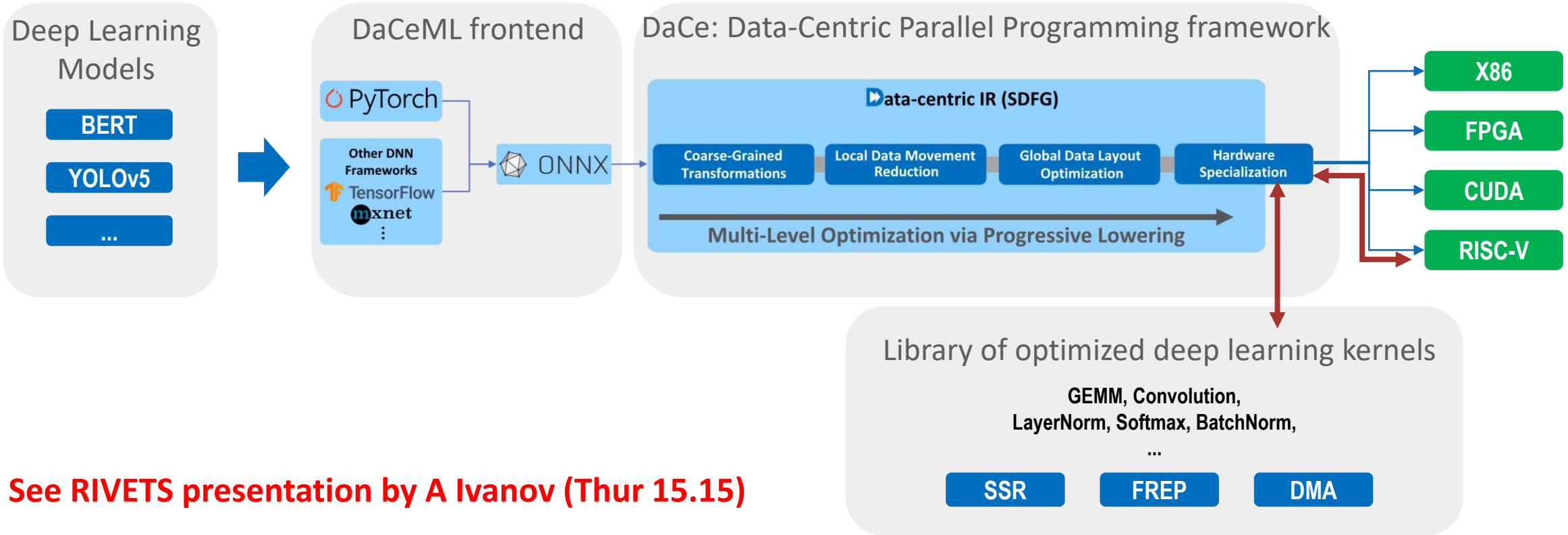
github.com/pulp-platform/snitch

github.com/pulp-platform/serial_link

Programming Occamy: DACE

Highly expressive DSL family – high-level transformations, support for explicitly managed memory

DaCeML: Data-Centric Machine Learning



See RIVETS presentation by A Ivanov (Thur 15.15)

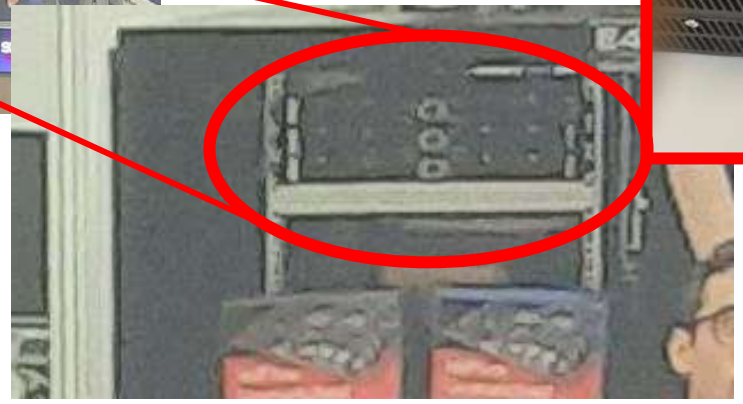
Efficient Chiplet architecture in Perspective

1. **Multi-cluster single-die scaling → strong latency tolerance, modularity**
2. **NoC for flexible Clus2Clus, Clus2Mem, C2C traffic → reduce pressure to Main memory**
3. **Top level NoC Routes to “local main memory” / “global main memory” balanced BW**
4. **Modular chiplet architecture: HBM2e, NoC-wrapped C2C, multi-chiplet ready**
5. **DSL and tooling for programming**

System Level: Monte Cimone, the first RISC-V Cluster

4x E4 RV007 1U Custom Server Blades:

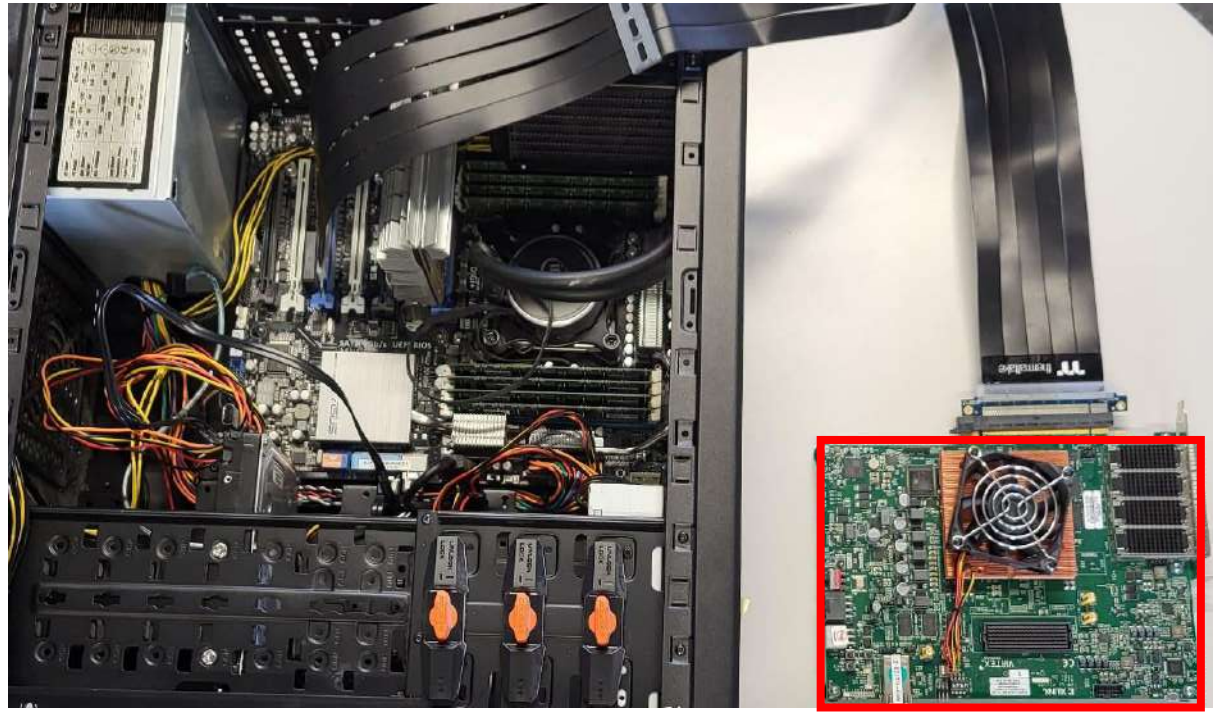
- **2x SiFive U740 SoC** with **4x U74 RV64GCB** cores
- 16GB of **DDR4**
- 1TB node-local **NVME storage**
- **PCIe expansion** card **w/InfiniBand HCAs**
- **Ethernet + IB parallel networks**



Designed for HPC “pipe cleaning”

Preparing for Occamy: Accelerator on PCIe cards

- Currently using FPGA-mapped “tiny Occamy”
  - VCU128 with HBM
- Supporting hybrid usage
  - Boot directly on standalone CVA6
  - Do not boot and let the Host control the cluster
  - HW probing by on-board device tree overlays
- High SW stack re-usability for both modes
  - Same Linux drivers to map the cluster
  - Same OpenMP offloading runtime



RISC-V®

Targeting HPC and Automotive
(European JUs focus on RISC-V)

KDT JU
KEY DIGITAL TECHNOLOGIES JOINT UNDERTAKING

EuroHPC Joint Undertaking

ETH zürich

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

PULP

Parallel Ultra Low Power

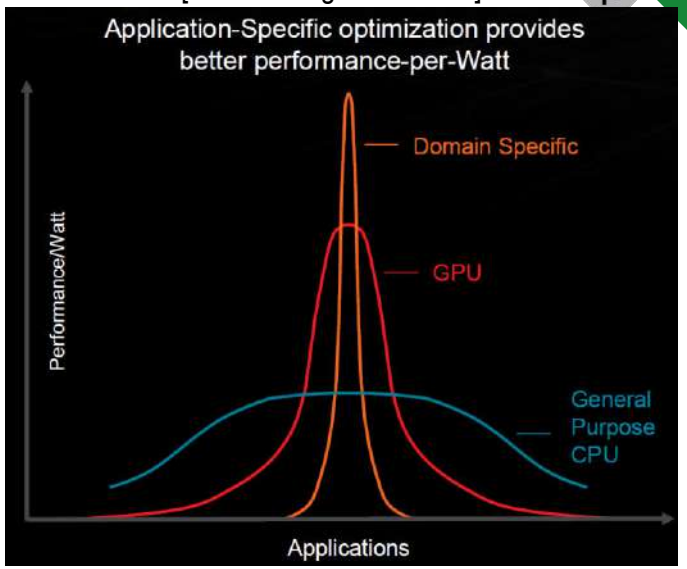
Luca Benini, Alessandro Capotondi, Alessandro Ottaviano, Alessio Burrello, Alfio Di Mauro, Andrea Borghesi, Andrea Cossettini, Andreas Kurth, Angelo Garofalo, Antonio Pullini, Arpan Prasad, Bjoern Forsberg, Corrado Bonfanti, Cristian Cioflan, Daniele Palossi, Davide Rossi, Fabio Montagna, Florian Glaser, Florian Zaruba, Francesco Conti, Georg Rutishauser, Germain Haugou, Gianna Paulin, Giuseppe Tagliavini, Hanna Müller, Luca Bertaccini, Luca Valente, Manuel Eggimann, Manuele Rusci, Marco Guermandi, Matheus Cavalcante, Matteo Perotti, Matteo Spallanzani, Michael Rogenmoser, Moritz Scherer, Moritz Schneider, Nazareno Bruschi, Nils Wistoff, Pasquale Davide Schiavone, Paul Scheffler, Philipp Mayer, Robert Balas, Samuel Riedel, Segio Mazzola, Sergei Vostrikov, Simone Benatti, Stefan Mach, Thomas Benz, Thorir Ingolfsson, Tim Fischer, Victor Javier Kartsch Morinigo, Vlad Niculescu, Xiaying Wang, Yichao Zhang, Frank K. Gürkaynak,
all our past collaborators and many more that we forgot to mention

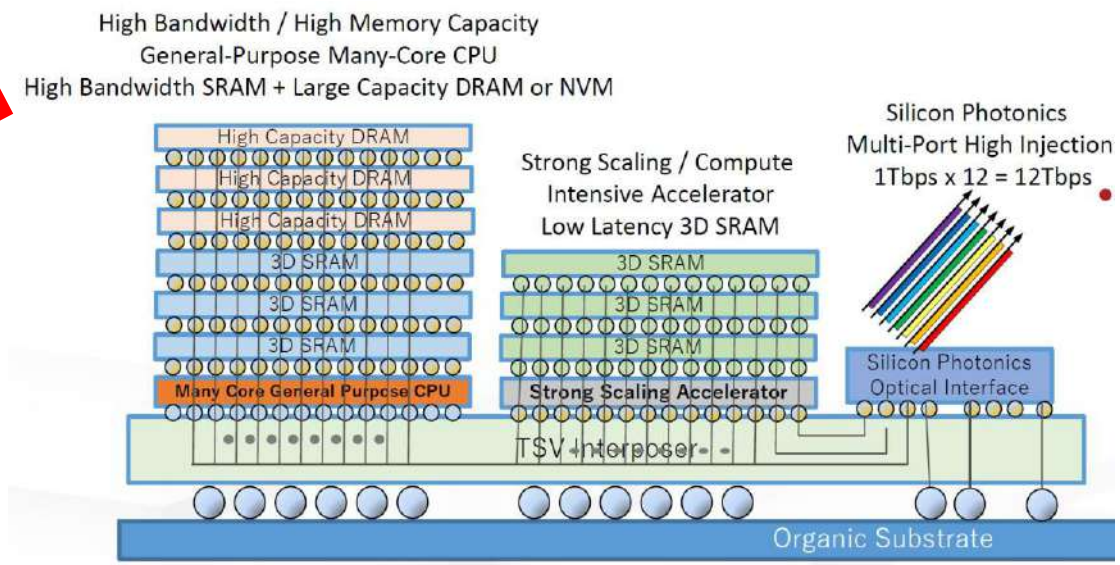
http://pulp-platform.org @pulp_platform

Conclusion

- Efficient, RT, Safe Secure: PE, Cluster, SoC, System
- Key ideas
  - Deep PE optimization → extensible ISAs (RISC-V!)
  - Low-overhead work distribution. Latency hiding → large “mempools”
  - Heterogeneous architecture →host+accelerator(s)
- Game-changing technologies
  - “Commoditized” chiplets: 2.5D, 3D
  - Computing “at” memory (DRAM mempool)
  - Coming: optical IO and smart NICs, swiches
- Challenges:
  - High performance RV Host
  - RV HPC software ecosystem?
  - Access to technology!

[AMD Naffziger ISCAS22]





[RIKEN Matsuoka MODSIM22]