Spatz: a compact vector processing unit for high-performance and energy-efficient shared-L1 clusters

2022 International Conference on Computer-Aided Design (ICCAD 2022)

Matheus Cavalcante matheus@iis.ee.ethz.ch

Domenic Wüthrich

Matteo Perotti

Samuel Riedel

Luca Benini

PULP Platform

Open Source Hardware, the way it should be!

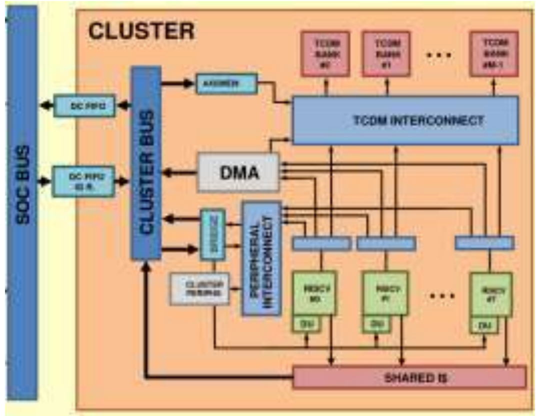
@pulp_platform

pulp-platform.org

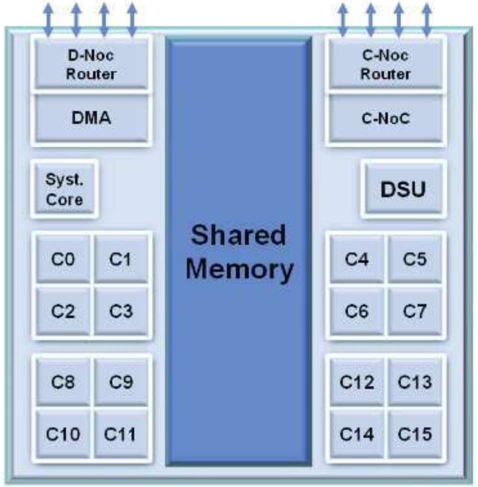
youtube.com/pulp_platform

The shared-L1 cluster: a ubiquitous building block

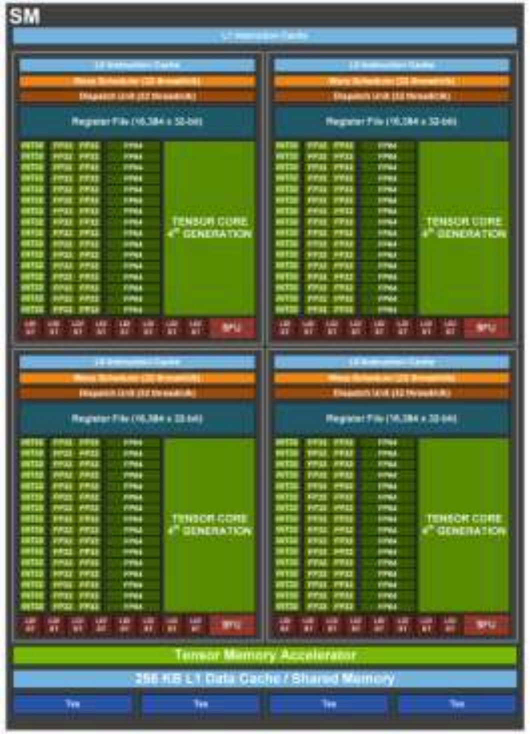
- Cluster of PEs sharing tightly-coupled L1 SPM through a low-latency interconnect
  - Common architectural pattern!



Embedded domain:
GreenWaves' GAP8



Many-core SoCs:
Kalray's MPPA-256

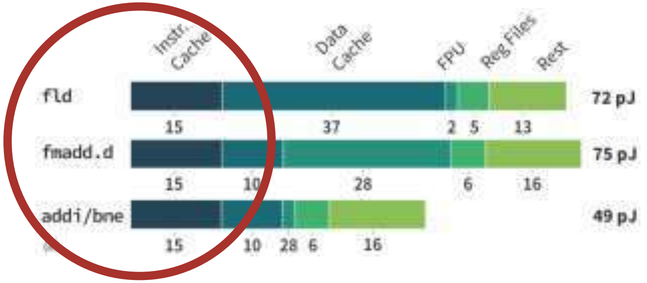

SM
L1 Instruction Cache
Register File (16,384 x 32-bit)
TENSOR CORE 4th GENERATION
SFU
Tensor Memory Accelerator
256 KB L1 Data Cache / Shared Memory
Tex

GPUs:
NVIDIA's H100 SMs

- The efficiency of this building block is key for building efficient large-scale systems

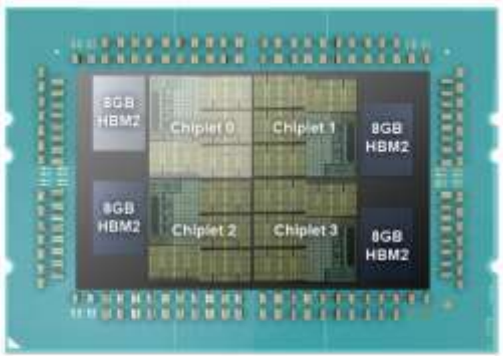
Pushing the shared-L1 cluster to its limit

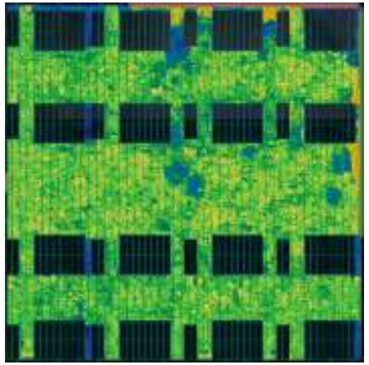
- **How small should the PE of a shared-L1 cluster be?**
- Evidence that a **tiny core** driving a large functional unit (FPU) is highly-efficient
  - OpenHW's CVA6 (Ariane) application-class core running a dot product
    - Out of the 317 pJ of the hot loop, **only 28 pJ are used for the actual computation**
    - Von Neumann Bottleneck (VNB)!



  - Snitch: single-stage tiny 22 kGE RV32IMAFD core driving a multi-precision 64-bit FPU
    - Emergence of large Snitch-based systems

Manticore:
4096-core chiplet-based prototype architecture



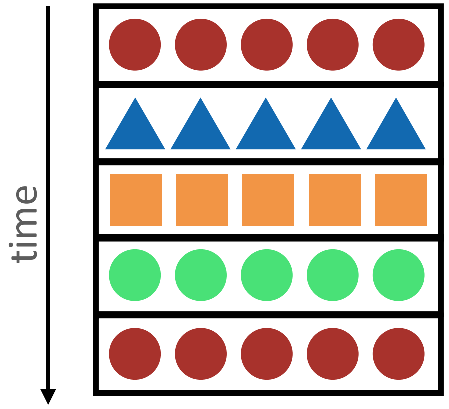


MemPool:
256-core single shared-L1 32-bit integer cluster sharing 1 MiB of L1 within 5 cycles of latency

Vector Processors: fighting back the von Neumann Bottleneck

- One of the most efficient approaches to tackle the VNB

- Exploiting the DLP of modern workloads by amortizing the instruction fetch and decode energy cost over many cycles of computation



- Since their inception, vector processors are tied with **supercomputers**
  - Cray-1, A64FX, Hwacha, Ara, Vitruvius…
    - Large vector units tied with application-class processors
    - Support for tricks that exploit **ILP** on top of **DLP** (OoO execution, renaming, speculation) which lead to *efficiency loss*

- What about a tiny embedded vector machine that focus on the DLP?
  - Arm Helium/MVE (M-Profile Vector Extension) on the Cortex-M cores
  - RISC-V's Zve32x and Zve64x subsets of the Vector Extension (RVV)

Spatz: a tiny and mighty vector processor unit

- *What about a tiny embedded vector machine that focus on the DLP?*
  - Fighting back the von Neumann Bottleneck with a flock of tiny vector machines

- **Spatz:** a compact 32-bit vector processing unit based on RVV Zve32x
  - **Generic:** Spatz interfaces with the scalar core with CORE-V's generic accelerator interface
  - **Parametric:** Configurable number of parallel functional units and VRF size
  - **High-performance:** Spatz must be performant on key data-parallel kernels, as a VPU and as the PE of a large many-core system
  - **Physically-driven implementation:** Small footprint, high operating frequency and energy-efficiency are mandatory for scalability

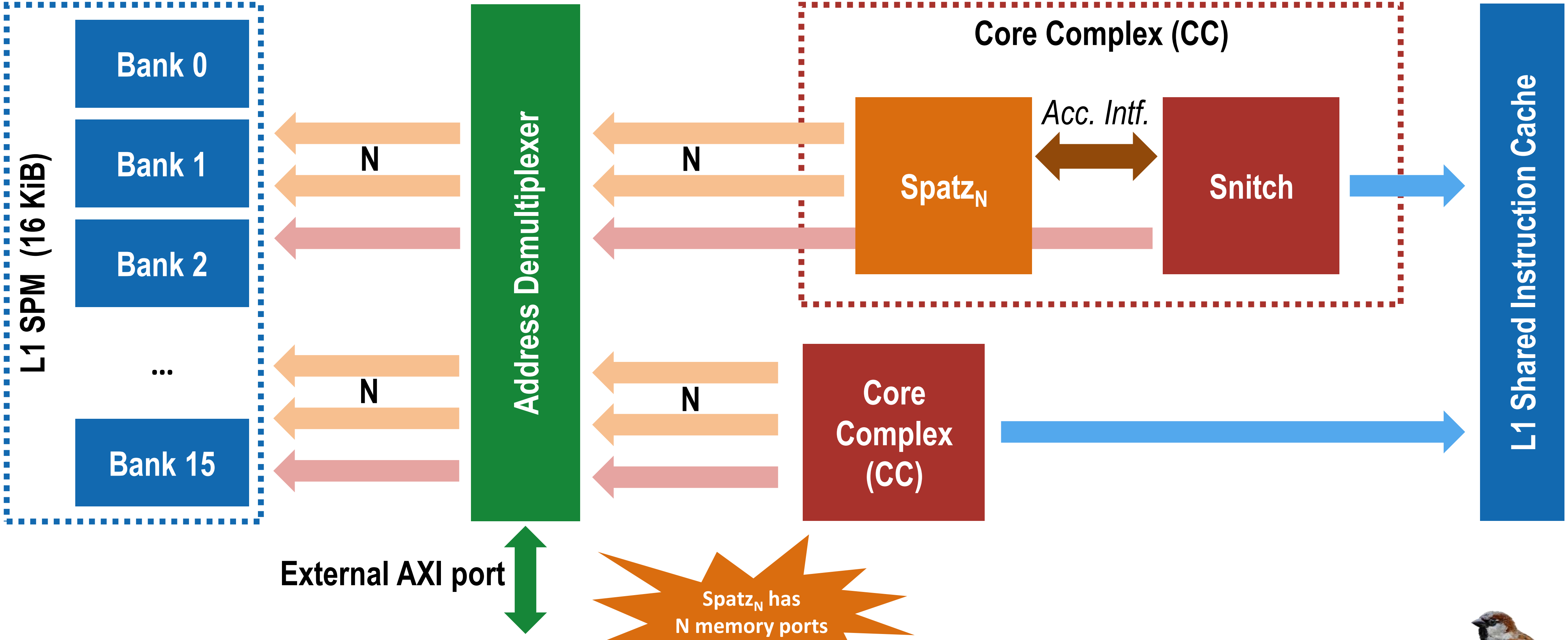
Spatz' microarchitecture

Vector ISAs: the RISC-V Vector Extension

- Embedded vector ISAs and VPUs are a recent development in the vector scene
- Arm Helium/M-Profile Vector Extension (MVE)
  - Optional extension to the Cortex-M55 core
- RISC-V Vector Extension (RVV) ISA
  - Open-source, modular, free to use and adapt
  - The RISC-V Vector Extension (RVV) is the largest subset of the RISC-V ISA
    - Larger than all other RISC-V ISA subsets combined
    - Not the best ISA for a lean VPU ☺
  - Smallest embedded profile of RVV: Zve32x
    - Integer operations on 8, 16, and 32 bits
    - Step in the right direction, but not the tiny vector ISA we wanted:
      - Precise exceptions: **mandatory**
      - Reductions: **mandatory**
      - Fixed-point: **mandatory**
      - Vector register-gather: **mandatory**

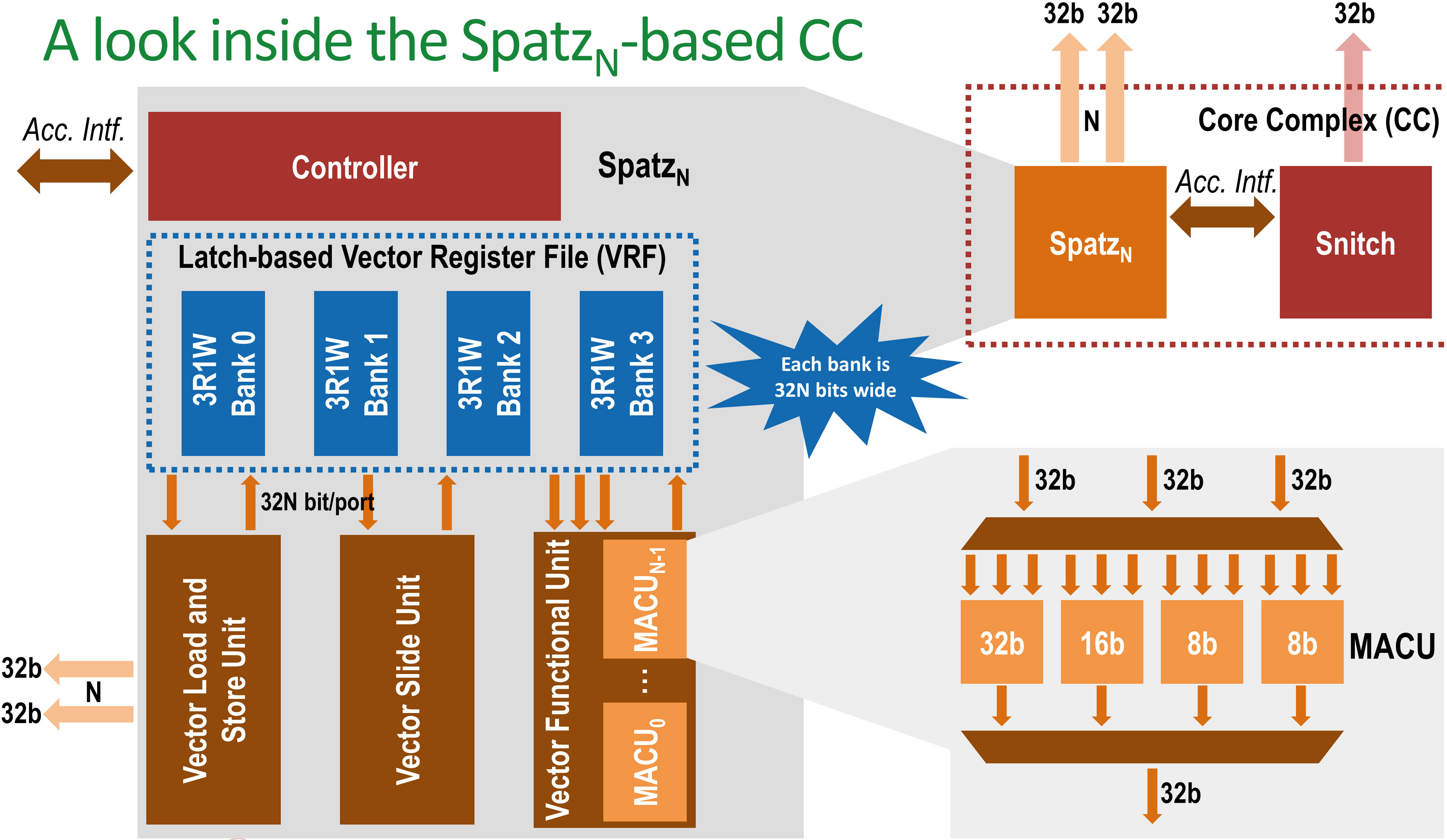
No support for those for now

Spatz integration within a small shared-L1 cluster



Spatz$_N$ has N memory ports

A look inside the Spatz$_N$-based CC



Spatz as a processing element

Evaluation methodology and configurations

- We consider two **Spatz** configurations: **Spatz$_2$** and **Spatz$_4$**

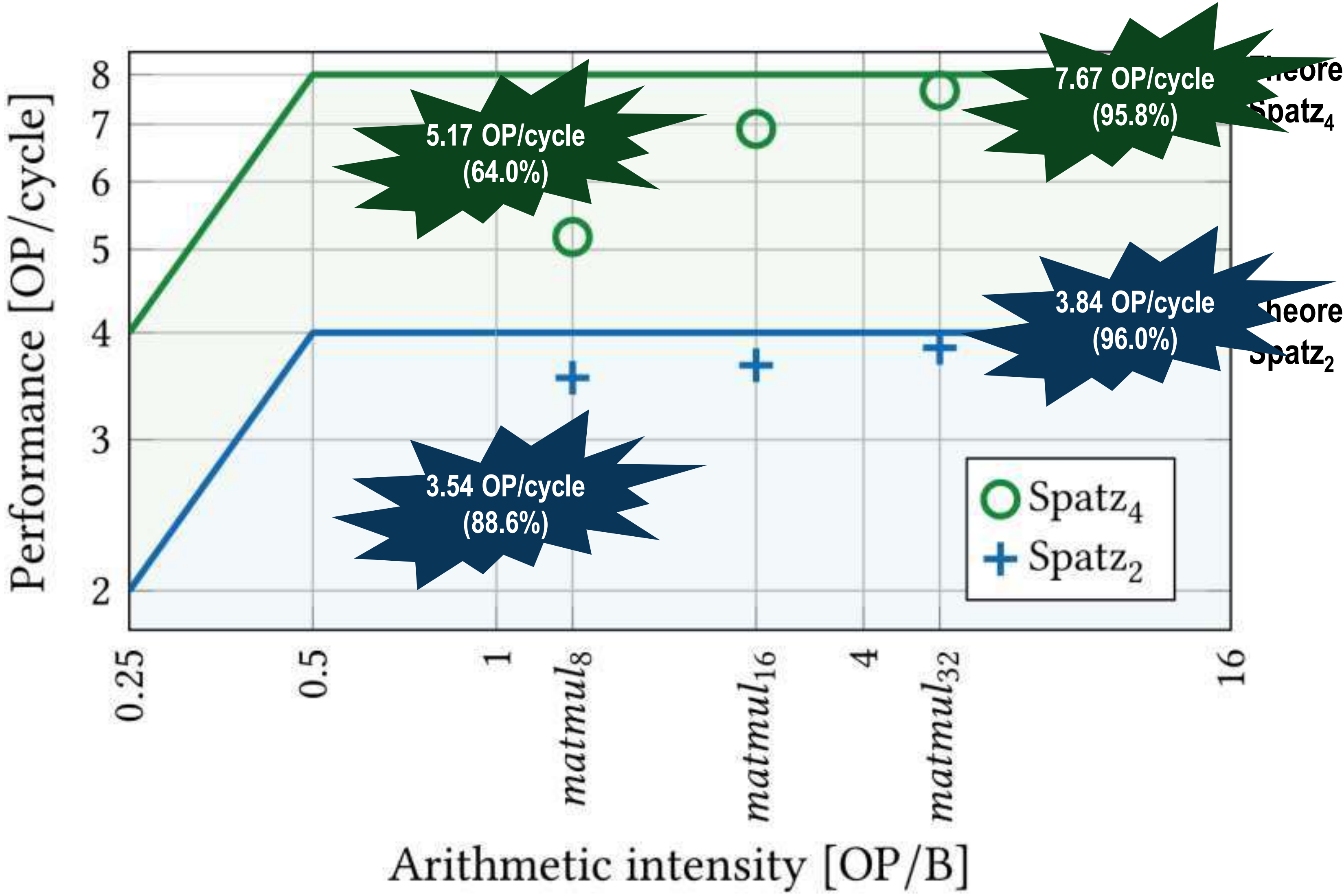
Configuration	Spatz$_2$	Spatz$_4$
#MACUs	2	4
Vector length [bit]	256	512

128 b/MACU

0.5 OP/B

- We evaluate Spatz on
  - *matmul$_n$:* vectorized matrix multiplication of two n × n matrices
  - *conv2d$_f$:* 2D convolution of a large image with a f × f kernel
- Cycle-accurate simulation on the Verilated model of Spatz

Spatz' performance as a PE on data-parallel kernels: matmul



5.17 OP/cycle (64.0%)

7.67 OP/cycle (95.8%)

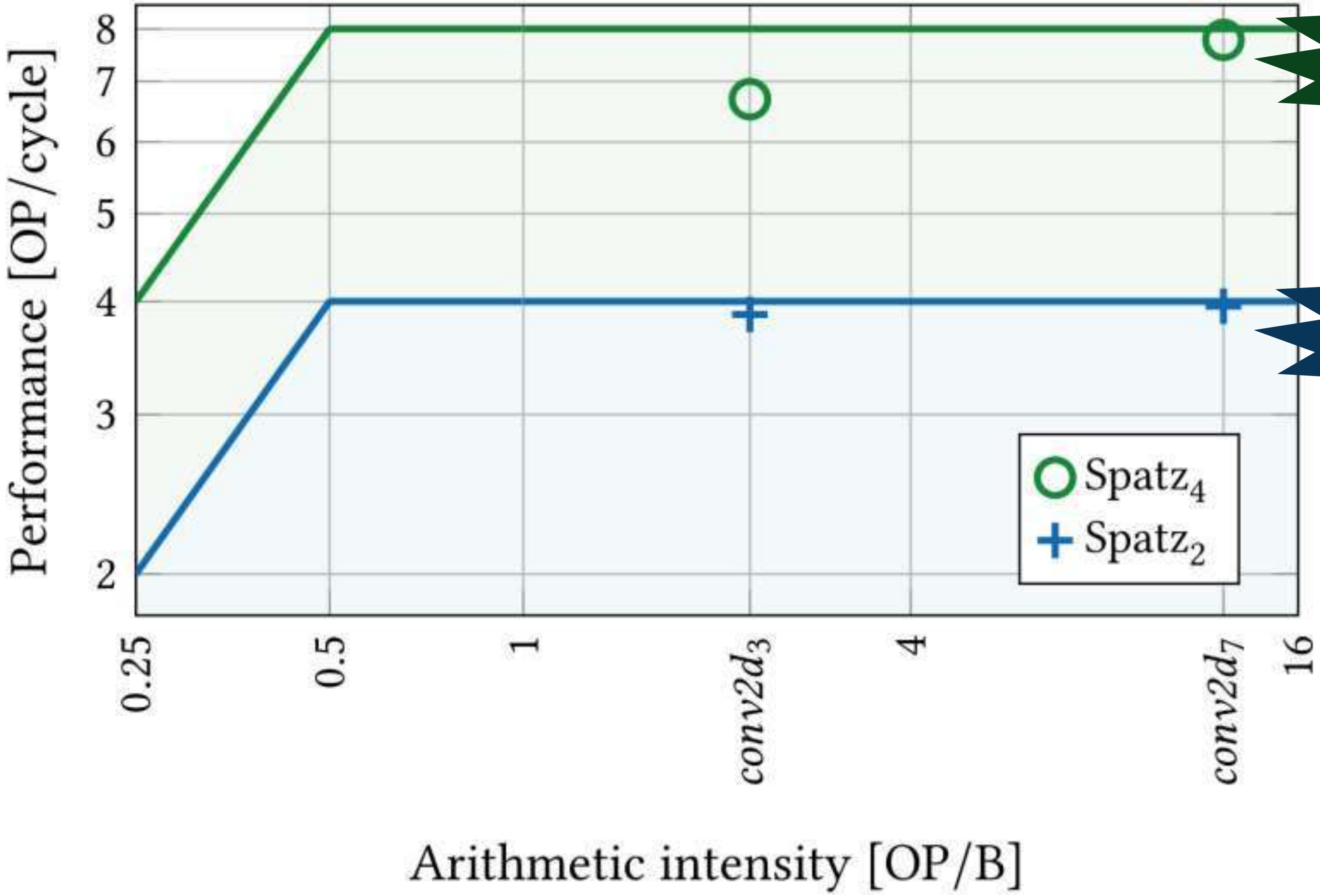
3.54 OP/cycle (88.6%)

3.84 OP/cycle (96.0%)

Theoretical peak performance Spatz$_4$

Theoretical peak performance Spatz$_2$

Spatz' performance as a PE on data-parallel kernels: conv2d

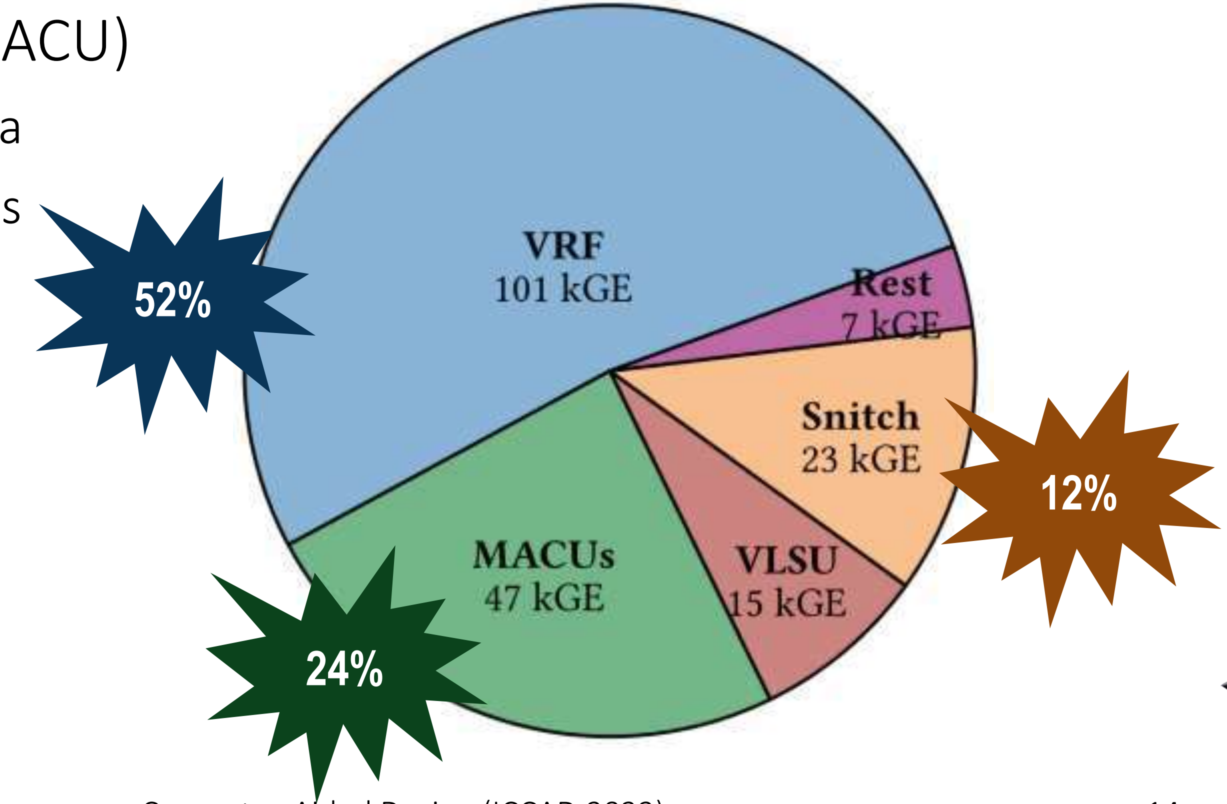


7.78 OP/cycle (97.2%)

3.95 OP/cycle (98.8%)

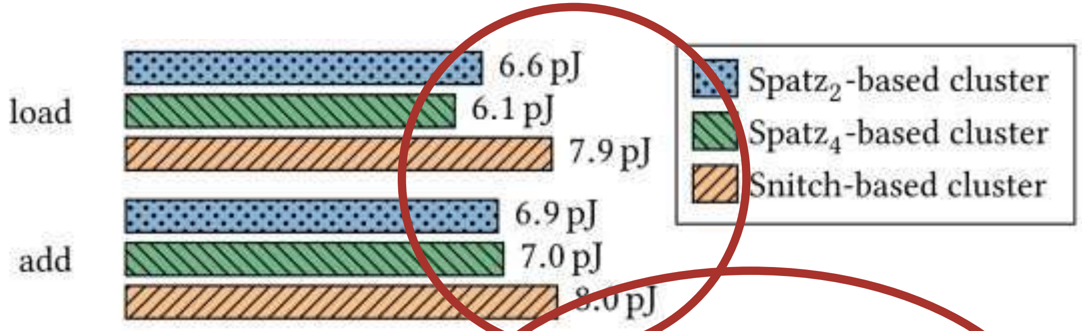
Spatz$_2$ area breakdown

- Spatz was synthesized and implemented with *Synopsys Fusion Compiler 2022.03*
- GlobalFoundries' 22FDX FD-SOI technology
  - Frequency target of 500 MHz in worst-case conditions (SS/0.72V/125°C)
- Spatz$_2$'s area: 193 kGE (97 kGE/MACU)
  - The 1 KiB VRF occupies most of its area
  - Followed by the two parallel MAC units
  - And the Snitch core



Spatz as a PE: energy consumption per elementary operation

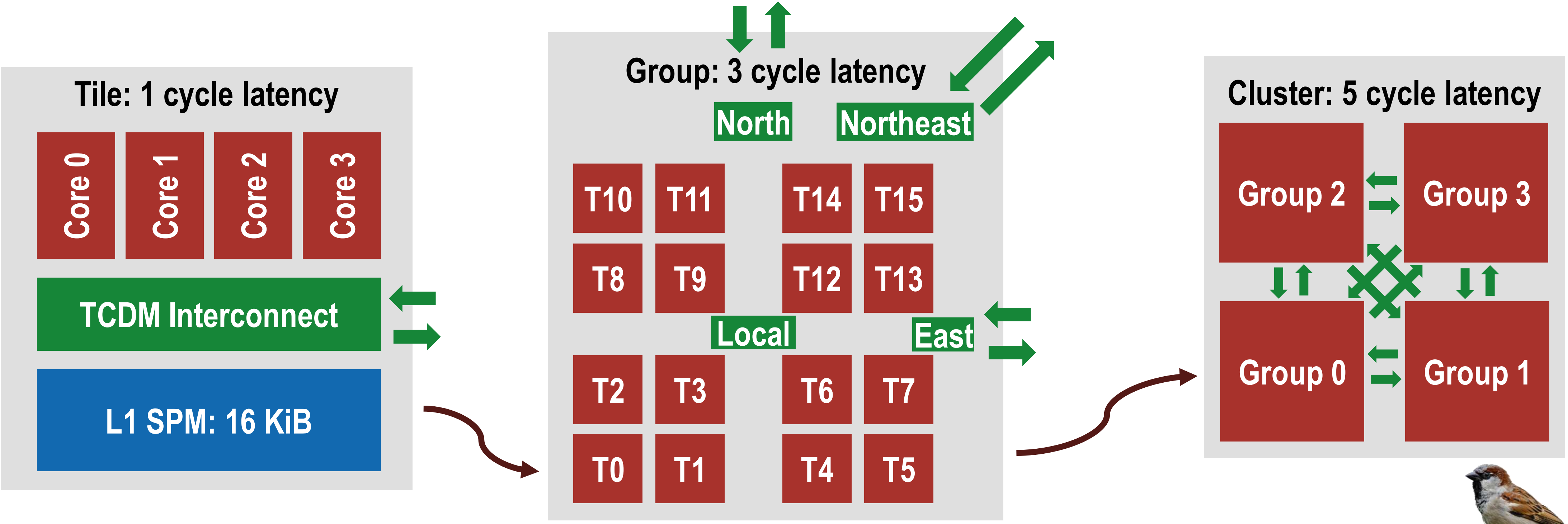
- Consider a small shared-L1 computing cluster containing either:
  - **Spatz$_2$-based:** 2 Snitch + Spatz$_2$ cores
  - **Spatz$_4$-based:** 1 Snitch + Spatz$_4$ cores
  - **Snitch-based:** 4 scalar Snitch cores
- What is the cost of elementary operations in each of those clusters?



Spatz-based MemPool cluster

MemPool: the shared-L1 cluster revisited

- MemPool: scaled-up shared-L1 cluster
  - 256 Snitch cores sharing 1 MiB of L1 SPM within at most five cycles of zero-load latency
    - VNB prone design!
  - Hierarchical interconnect and architecture: tile, group, cluster



Tile: 1 cycle latency

Core 0

Core 1

Core 2

Core 3

TCDM Interconnect

L1 SPM: 16 KiB

Group: 3 cycle latency

North

Northeast

T10 T11 T14 T15

T8 T9 T12 T13

Local

East

T2 T3 T6 T7

T0 T1 T4 T5

Cluster: 5 cycle latency

Group 2

Group 3

Group 0

Group 1

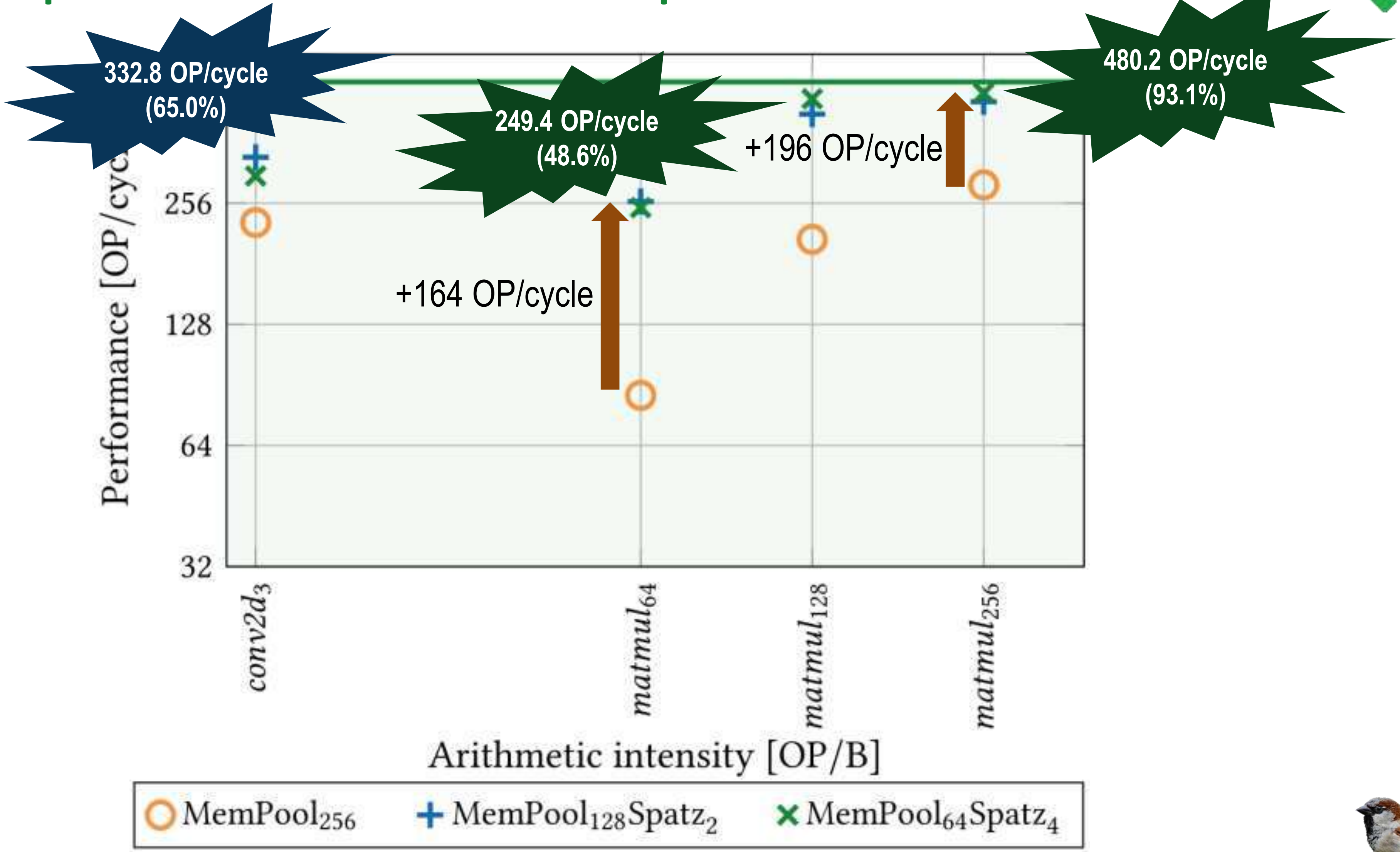
Snitch-based and Spatz-based MemPool configurations

- We replace **Snitch** with **Spatz** as the PE of MemPool

- We end up with three configurations
  - **MemPool$_{256}$:** Configuration with 256 Snitch cores
  - **MemPool$_{128}$Spatz$_2$:** Configuration with 128 Snitch + Spatz$_2$ cores
  - **MemPool$_{64}$Spatz$_4$:** Configuration with 64 Snitch + Spatz$_4$ cores

- All configurations have the same theoretical peak performance: 512 OP/cycle

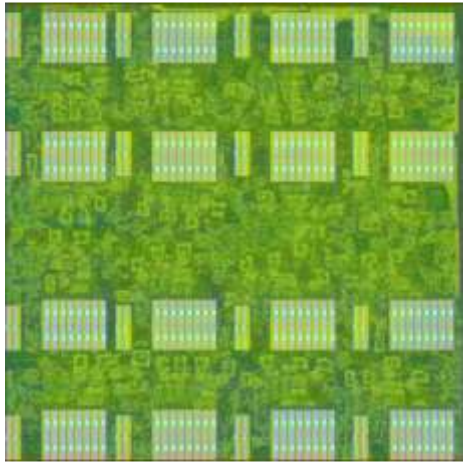
MemPool's performance on data-parallel kernels

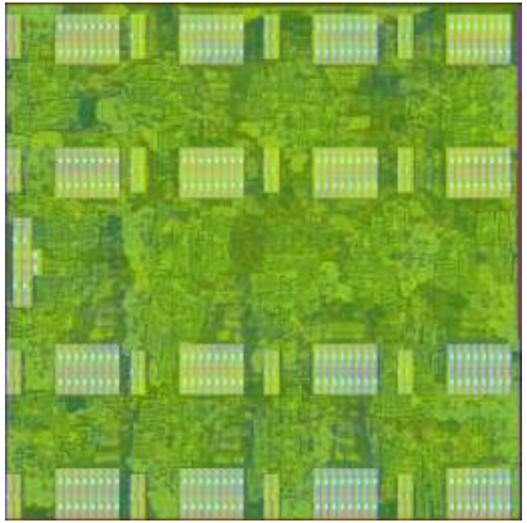


Physical implementation of the MemPool groups

Design	MemPool$_{256}$	MemPool$_{128}$Spatz$_2$	MemPool$_{64}$Spatz$_4$
Area [mm^2]	15.8	21.0	20.1

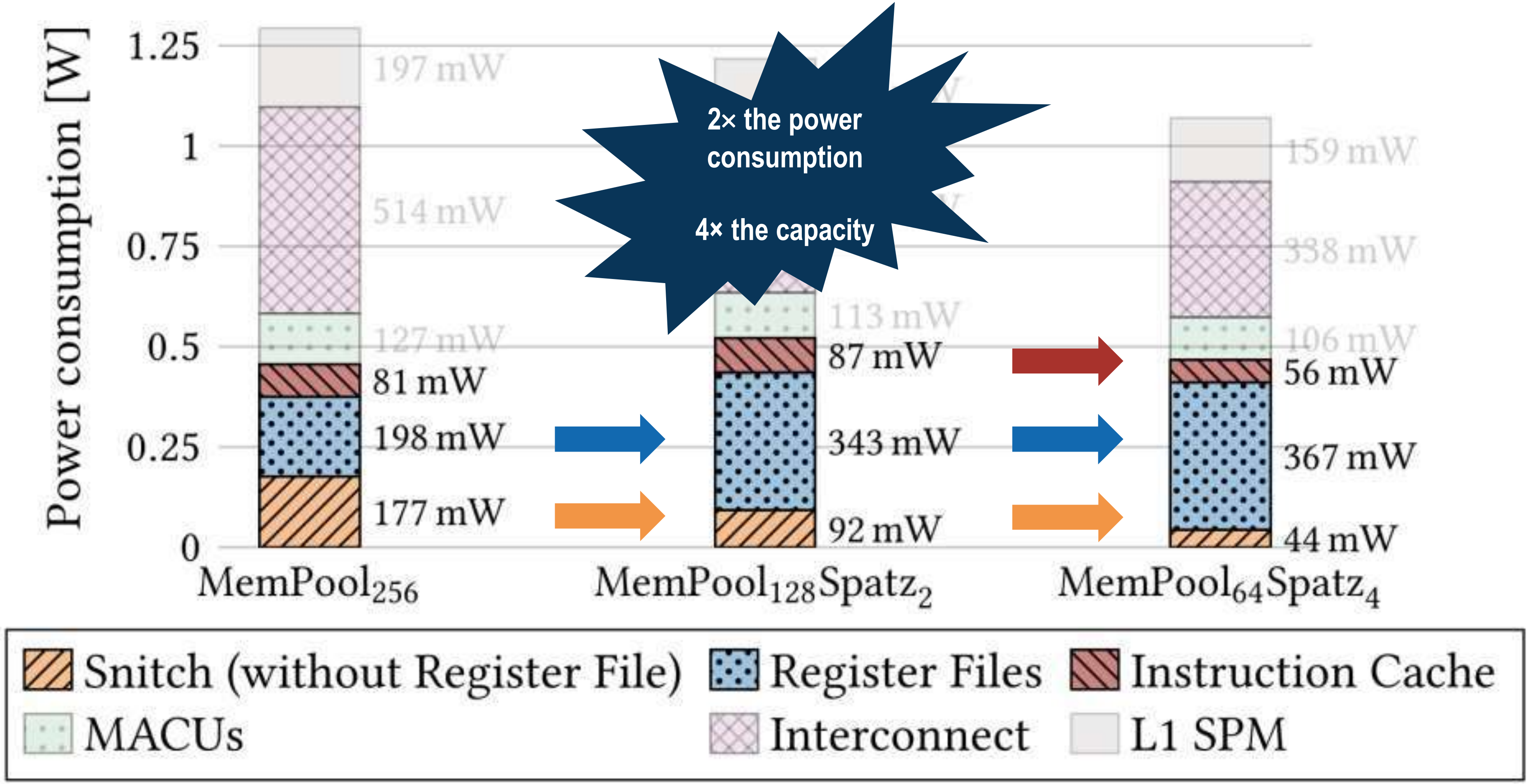
MemPool$_{256}$ group





MemPool$_{64}$Spatz$_4$ group

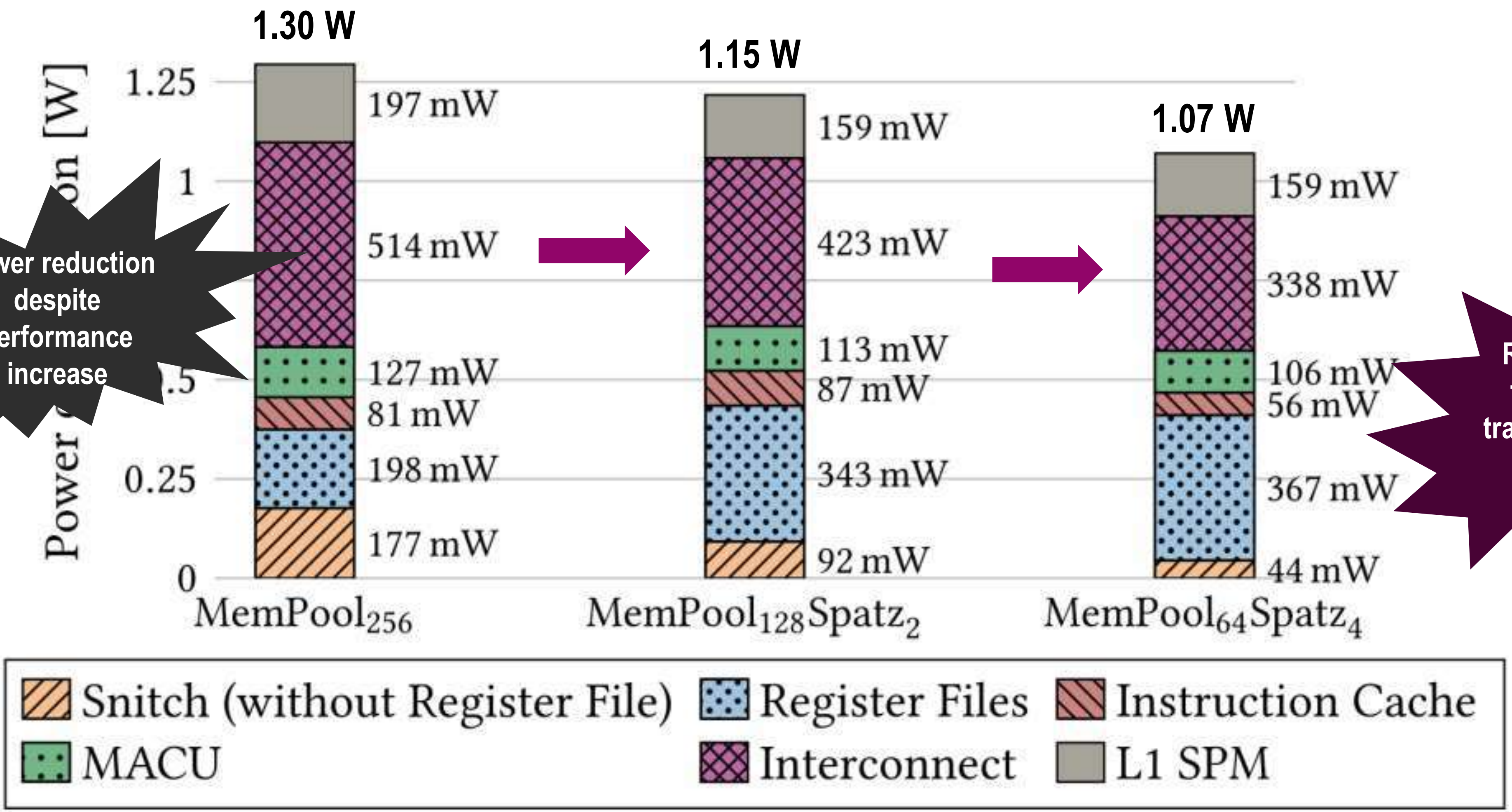
Power breakdown of MemPool running a matrix multiplication



2× the power consumption

4× the capacity

Power breakdown of MemPool running a matrix multiplication



Power reduction despite performance increase

Reduction in the L1 SPM traffic thanks to the VRF

Spatz as the PE of a large-scale shared-L1 cluster

- Spatz greatly improves MemPool's **performance**
  - While **reducing** its power consumption

Design	MemPool$_{256}$	MemPool$_{128}$Spatz$_{2}$	MemPool$_{64}$Spatz$_{4}$
Peak performance [GOPS]	167	270	**285**
Power consumption [W]	1.30	1.15	**1.07**
Energy Efficiency [GOPS/W]	128	234	**266**

More than 2× the efficiency of MemPool$_{256}$

- Even taking the increased area into account, Spatz is an extremely viable approach for building efficient PEs
  - For an **area increase of 27%**, we increased the **peak performance by 70%** and the **energy efficiency by 107%**

Conclusion and Future Work

Tiny vector processors for high-performance and efficient large-scale manycore systems

- Our objective: optimize the energy efficiency of the PE of a shared-L1 cluster
- **Spatz$_4$:** a compact RVV Zve32x-based VPU
  - Parametric design with 4 parallel MACUs and a 2 KiB latch-based centralized VRF
  - Compact vector units are highly efficient:
    - While Spatz$_4$ needs 7.9 pJ to execute an elementary macc, Snitch needs 13.1 pJ
- Spatz$_4$ as MemPool's PE:
  - MemPool$_{64}$Spatz$_4$ reaches up to **480 OP/cycle (94%)** running matmul$_{256}$
  - Spatz amortizes much of Snitch's power consumption
    - Snitch alone (without MACUs) is responsible for 29% of MemPool$_{256}$'s power consumption
  - For an **area increase of 27%** wrt. MemPool$_{256}$, we **increased** MemPool$_{64}$Spatz$_4$'s **energy efficiency by 107%**, reaching **266 GOPS/W**

What is left for Spatz?

- VRF is most of Spatz' power consumption
  - Can we improve this architecturally or through an advanced implementation flow?
- What about floating-point capable systems?
  - FPUs consume much more power than an integer MACU
    - That hides the VNB problem associated with the scalar core power consumption
  - How attractive would Spatz be in such an environment?
- Tensor support for Spatz
  - What is the cost to add support for a small tensor ISA in Spatz?
- Open-sourcing and first tape-out
  - Both ongoing projects, happening soon! ☺

ETH zürich

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Spatz: a compact vector processing unit for high-performance and energy-efficient shared-L1 clusters

2022 International Conference on Computer-Aided Design (ICCAD 2022)

Matheus Cavalcante matheus@iis.ee.ethz.ch
Domenic Wüthrich
Matteo Perotti
Samuel Riedel
Luca Benini

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

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