ETH zürich ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

10 years of

Making open source RISC-V based computing architectures

F. Kağan Gürkaynak kgf@iis.ee.ethz.ch
and the PULP team

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

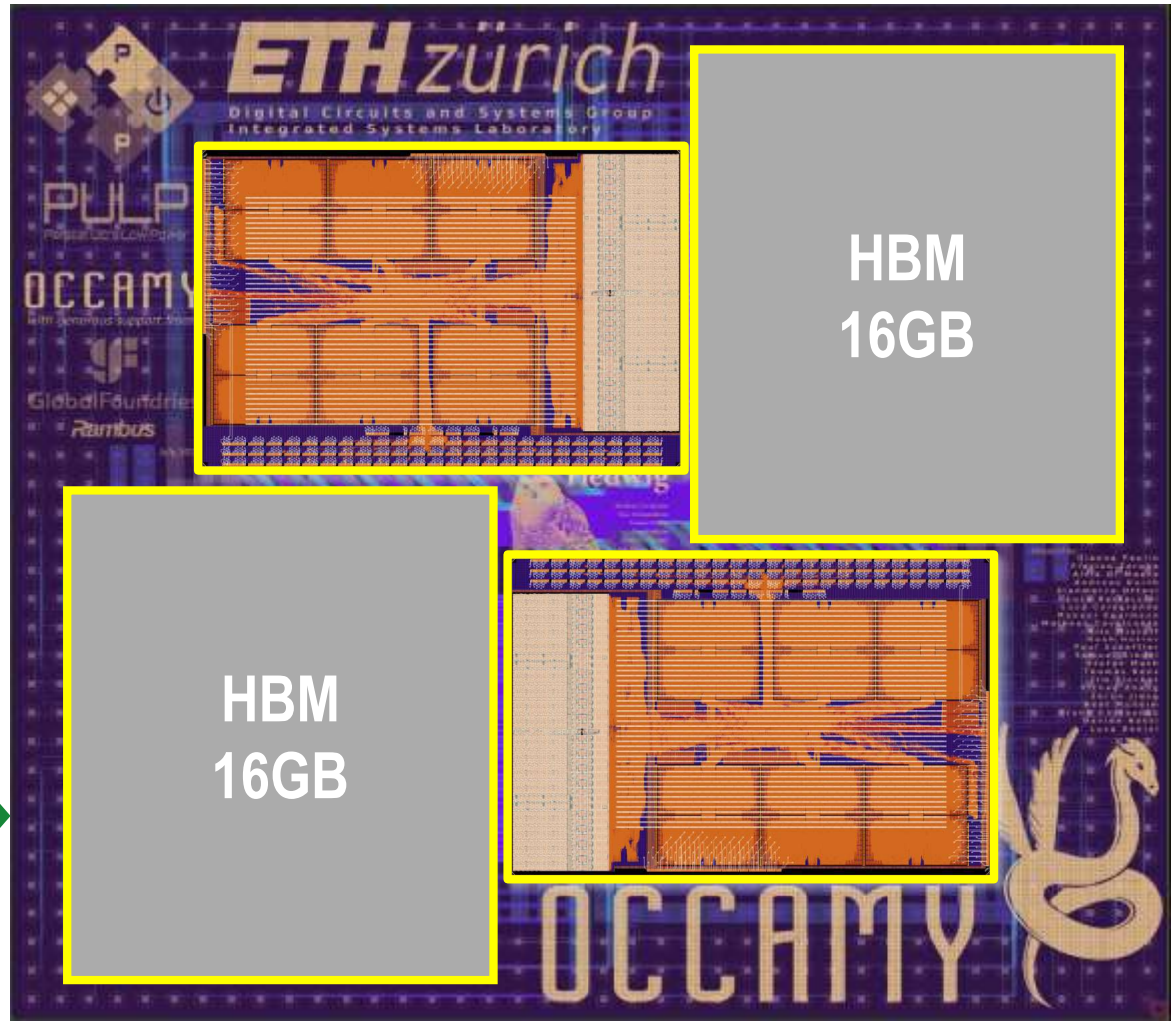
PULP Platform
Open Source Hardware, the way it should be!

Our latest design Occamy: 0.75 TFLOP/s, 400+ cores

- Chiplet based design
- 2x Compute chiplets (Occamy)
  - 216+1 RISC-V cores
  - GF12LPP
  - Running at 1 GHz
- 2x 16GByte HBM memories
- Silicon Interposer (Hedwig)
- Finished in less than 15 months

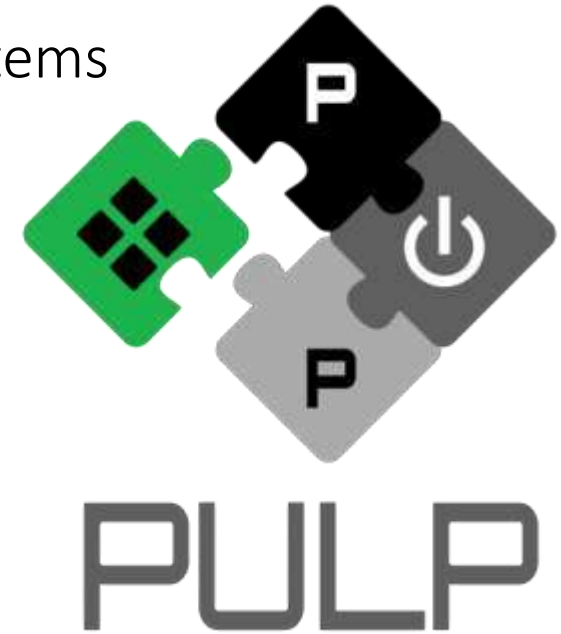
How did we manage this?

  - Taped out on 1st of July 2022



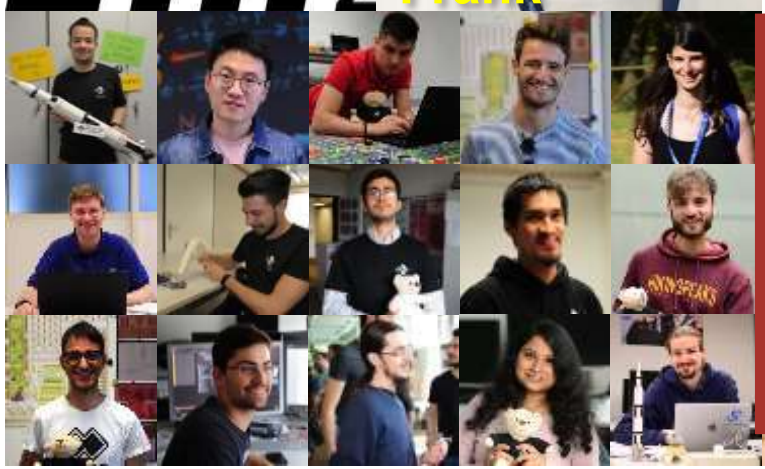
Our goal when we started in 2013: energy efficiency

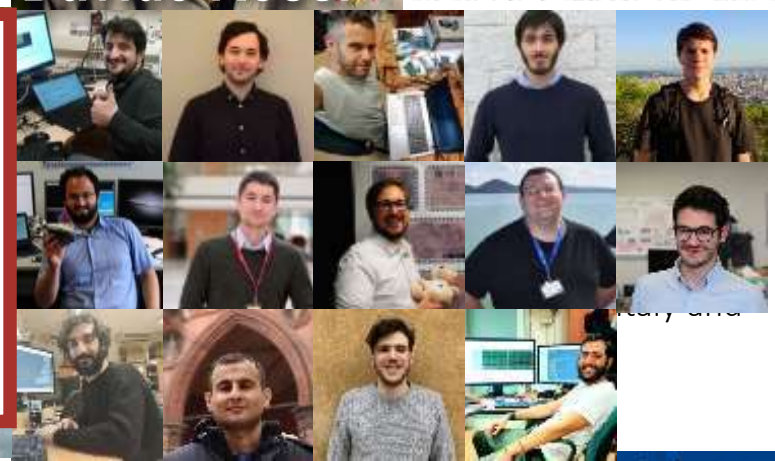
- Investigating new computing architectures
  - Efficient over a wide range from IoT applications to HPC systems
- Key points
  - Parallel processing
  - Near threshold computing
  - Efficient switching between operating modes
  - Making best use of technology
  - Heterogeneous acceleration
- **Parallel Ultra Low-Power** (PULP) platform was born



Who are we and who is behind PULP?

Prof. Luca Benini



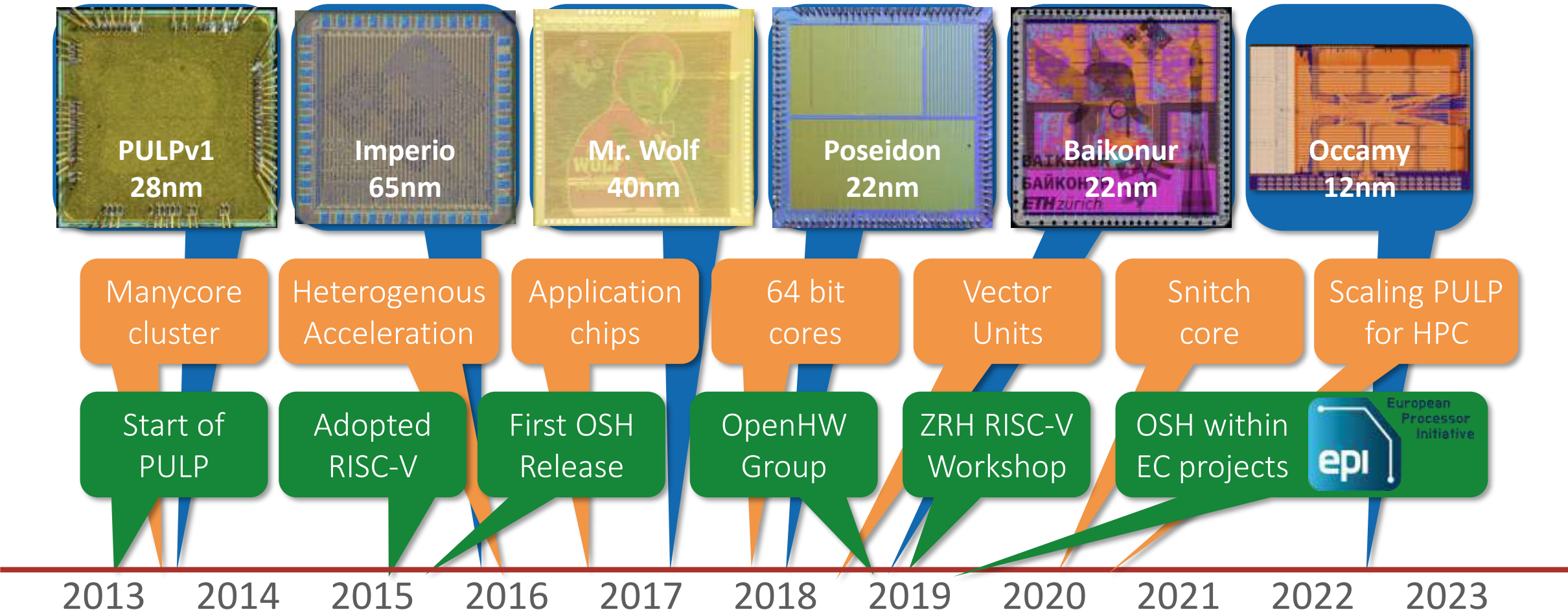


In total about 60 people work on projects related to PULP in Zurich and Bologna

https://pulp-platform.org/team.html

2012)

Timeline of Parallel Ultra Low Power (PULP) project



You need a 'core' to build computer architectures

- Initially we did not want to design our own processors
  - Wanted to use available processors (ARC, ARM..)
  - It proved difficult to design systems that we could share with our collaborators
- Then we used OpenRISC cores (2013-2015)
  - We had to completely redesign and optimize these cores

- We moved to RISC-V starting in 2015
  - Adapted the decoder of our optimized OpenRISC core
  - Make use of a growing SW development environment
  - ETH is one of the founding members of the RISC-V foundation

Our research is not implementing RISC-V cores

- We develop efficient programmable architectures
  - Processor cores of various capabilities are required for that
- We need efficient implementations of cores for our research
  - To produce relevant results, our cores have to perform as good as other solutions
  - We ended up spending quite an effort to make sure they perform really well
- Processor core alone is not enough
  - You need peripherals, interconnect solutions, programming support...
- **PULP Platform** provides us a playground for our research
  - And we share it as open source

RISC-V cores developed by PULP team

32 bit			64 bit
Low Cost Core	**Core with DSP support**	**Core for Streaming**	**Linux capable Core**
▪ **IBEX** ▪ Zero-riscy ▪ Micro-riscy ▪ **2 options** ▪ RV32-ICM ▪ RV32-CE *ARM Cortex-M0+*	▪ **CV32E40P** ▪ RV32-ICMXF ▪ SIMD ▪ HW loops ▪ Bit manipulatio *ARM Cortex-M4F* ▪ Fixed point	▪ **Snitch** ▪ RV32-IMAFD ▪ SmallInt core ▪ Sig. 64bit FPU *novel* ▪ Extensions for	▪ **Ariane/ CVA6** ▪ RV64-ICMAFD *ARM Cortex-A55*

Maintained By LowRISC

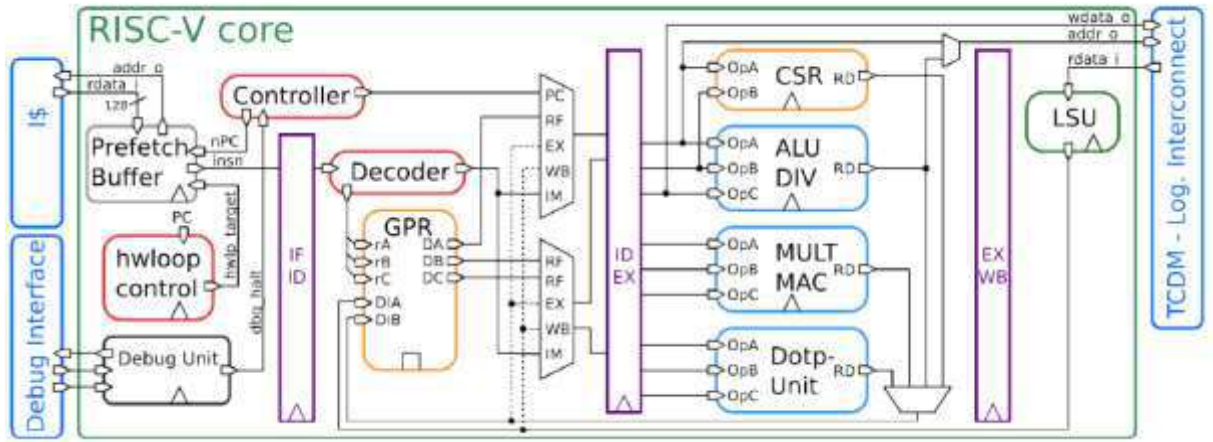
Maintained By OpenHW

Brand New Core

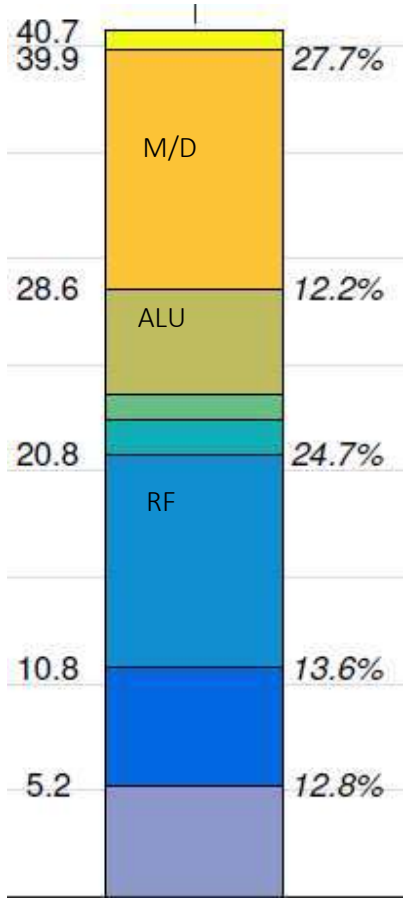
Maintained by OpenHW

CV32E40P –aka RI5CY – RISC-V Core for edge AI

3-cycle ALU-OP, 4-cyle MEM-OP→IPC loss: LD-use, Branch



40 kGE
70% RF+DP



[Gautschi et al. TVLSI 2017]

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

V1 Baseline RISC-V RV32IMC (not so good for ML)

HW loops

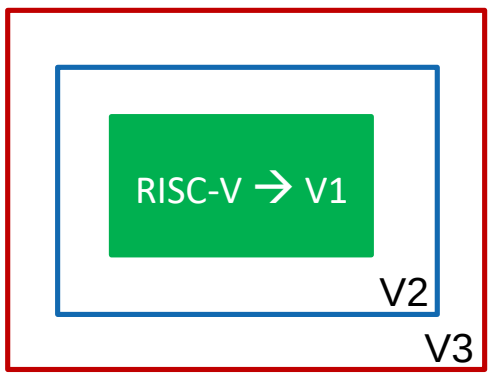
V2 Post modified Load/Store

Mac

V3 SIMD 2/4 + DotProduct + Shuffling

Bit manipulation unit

Lightweight fixed point



XPULP extensions: 25 kGE → 40 kGE (1.6x)

PULP-NN: Xpulp ISA exploitation

8-bit Convolution

HW Loop

LD/ST with post increment

8-bit SIMD sdotp

8-bit sdotp + LD

RV32IMC

N

```
addi   a0,a0,1
addi   t1,t1,1
addi   t3,t3,1
addi   t4,t4,1
lbu    a7,-1(a0)
lbu    a6,-1(t4)
lbu    a5,-1(t3)
lbu    t5,-1(t1)
mul    s1,a7,a6
mul    a7,a7,a5
add    s0,s0,s1
mul    a6,a6,t5
add    t0,t0,a7
mul    a5,a5,t5
add    t2,t2,a6
add    t6,t6,a5
bne    s5,a0,1c000bc
```

RV32IMCXpulp

N/4

```
lp.setup
p.lw  w1, 4(a0!)
p.lw  w2, 4(a1!)
p.lw   x1, 4(a2!)
p.lw   x2, 4(a3!)
pv.sdotsp.b   s1, w1, x1
pv.sdotsp.b   s2, w1, x2
pv.sdotsp.b   s3, w2, x1
pv.sdotsp.b   s4, w2, x2
end
```

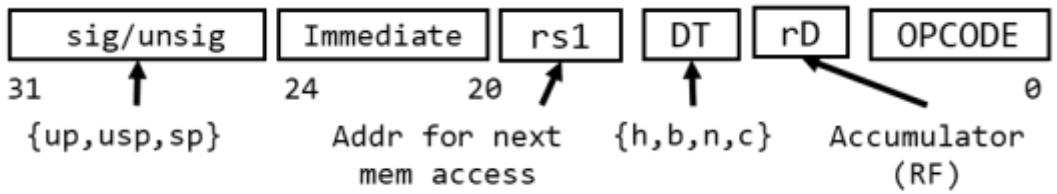
can we remove?

Yes! DOTP+LW

N/4

```
Init NN-RF (outside of the loop)
lp.setup
pv.nnsdotup.h  s0,ax1,9
pv.nnsdotsp.b  s1, aw2, 0
pv.nnsdotsp.b  s2, aw4, 2
pv.nnsdotsp.b  s3, aw3, 4
pv.nnsdotsp.b  s4, ax1, 14
end
```

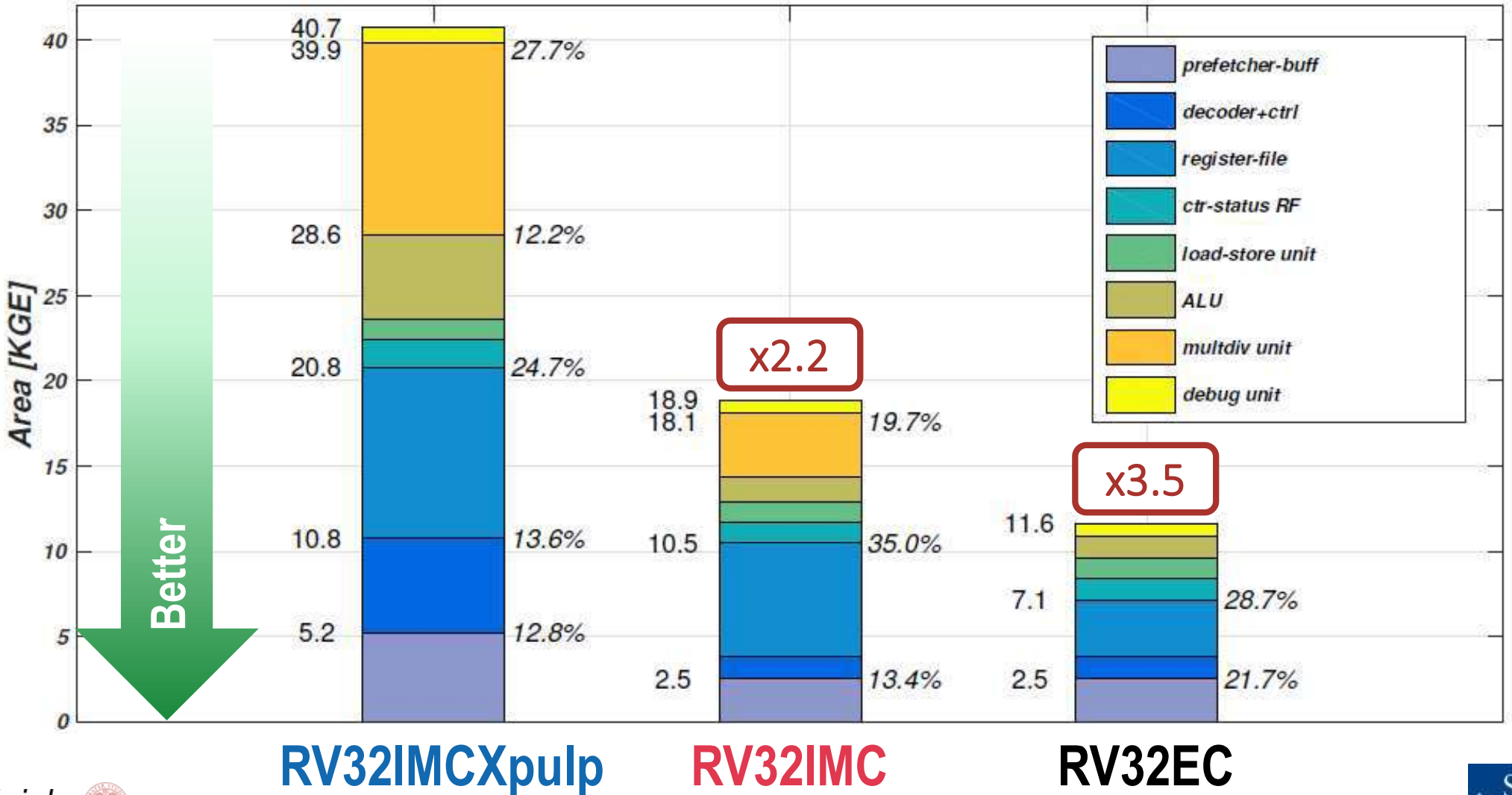
pv.nnsdot{up,usp,sp}.{h,b,n,c} rD, rs1, Imm



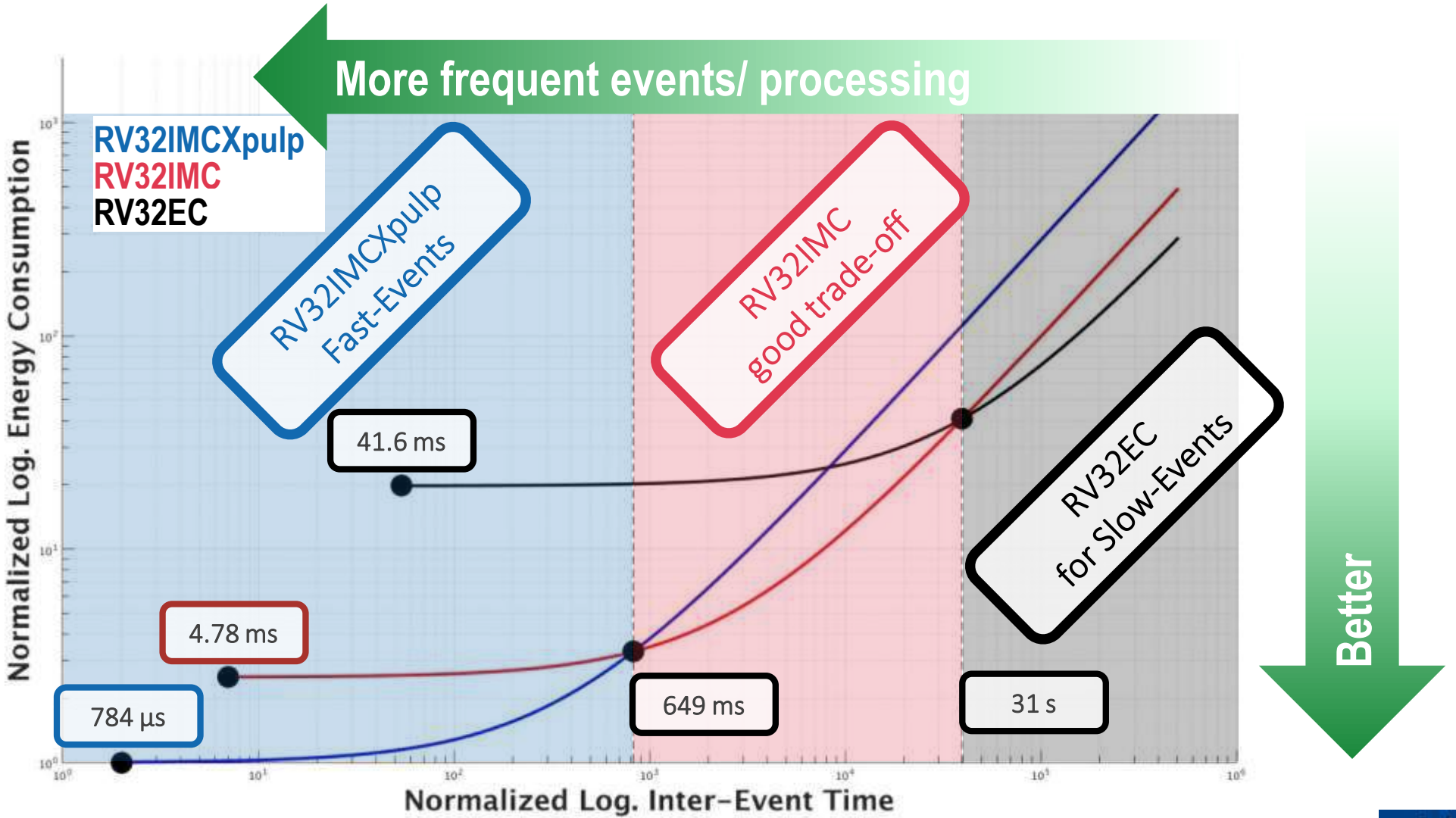
9x less instructions than RV32IMC

14.5x less instructions at an extra 3% area cost (~600GEs)

Different cores for different area/power budgets



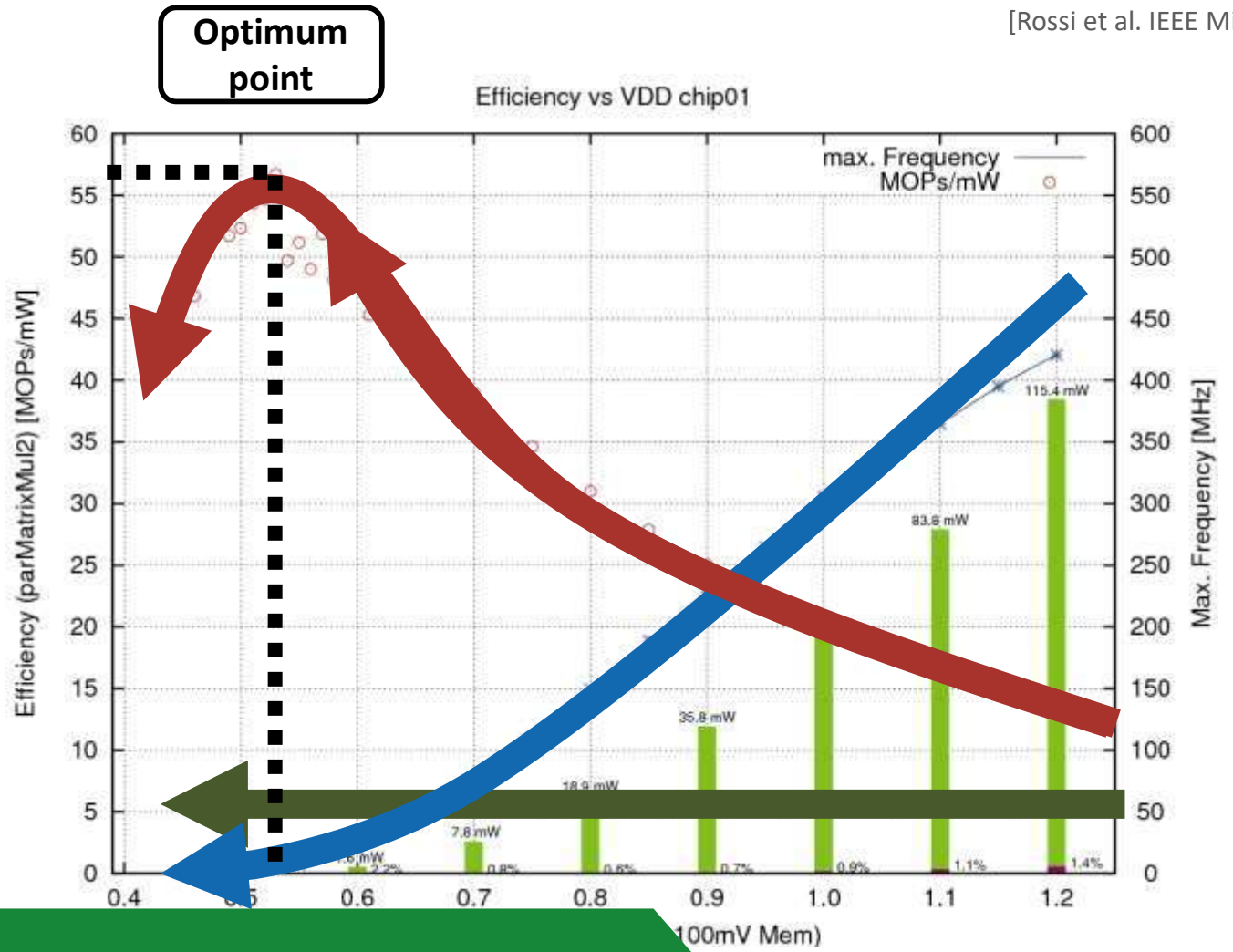
Energy Efficiency: 2D-Convolution @55MHz, 0.8V



Scaling performance: the case for parallelization

- As **VDD** decreases, **operating speed** decreases as well.
- However **efficiency** increases→ more work done per Joule
  - Until leakage effects start to dominate
- Put more units in parallel to get performance up and keep them busy with a parallel workload

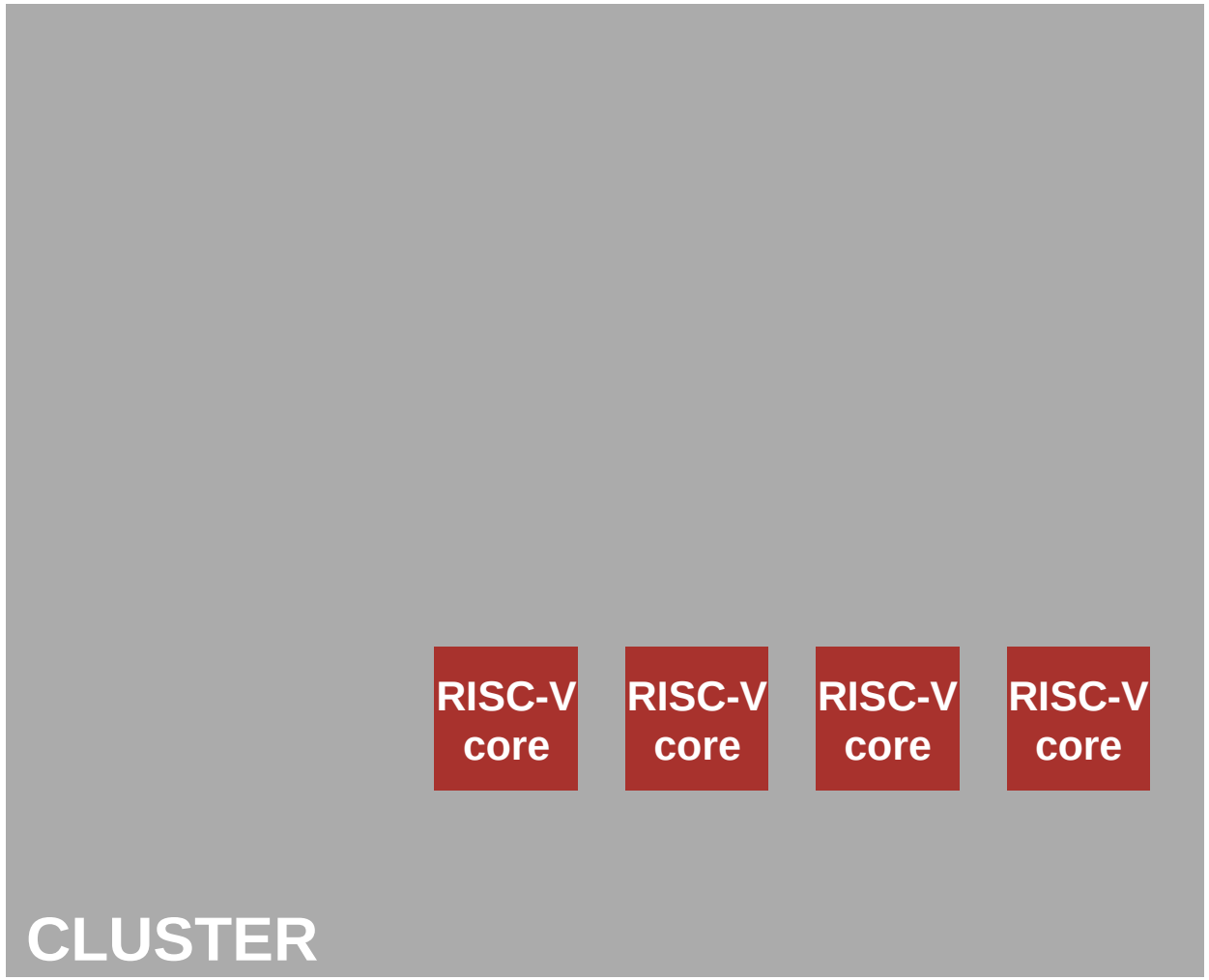
[Rossi et al. IEEE Micro 2017]



N cores running at lower VDD is more energy efficient

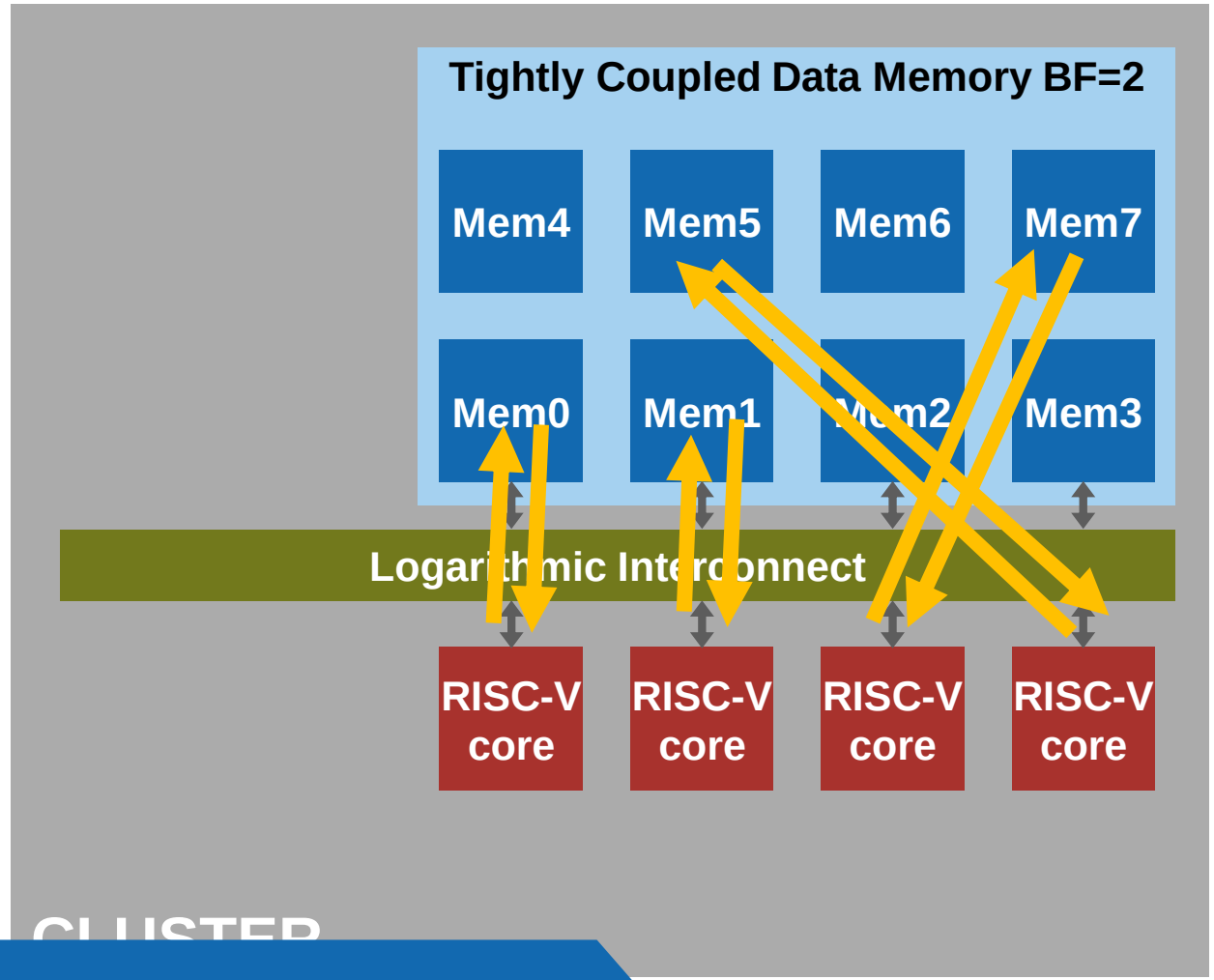
Sabancı Üniversitesi

Let's design a cluster with multiple (4-16) RISC-V cores



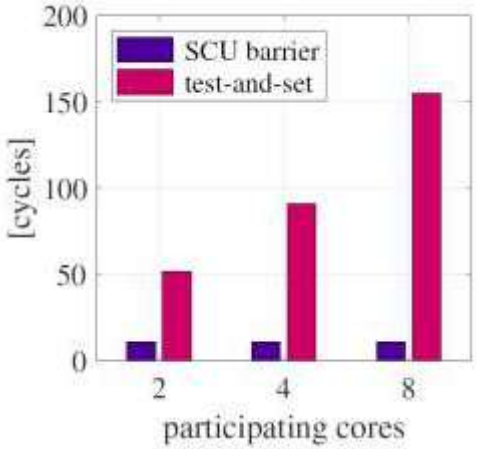
Low-Latency shared Tightly Coupled Data Memory (TCDM)

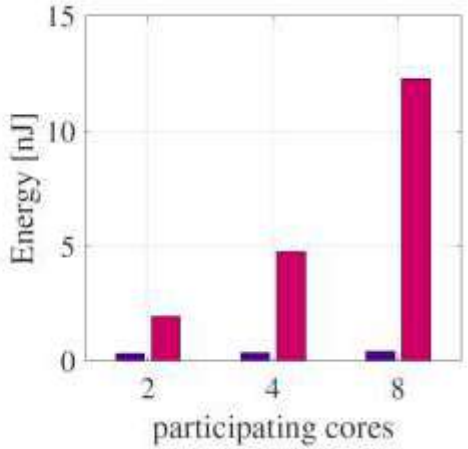
- Parallel memory access with low contention
  - Multi-banked, address-interleaved L1
- Fast interconnect with physical design awareness
  - Logarithmic depth of combinational switchboxes

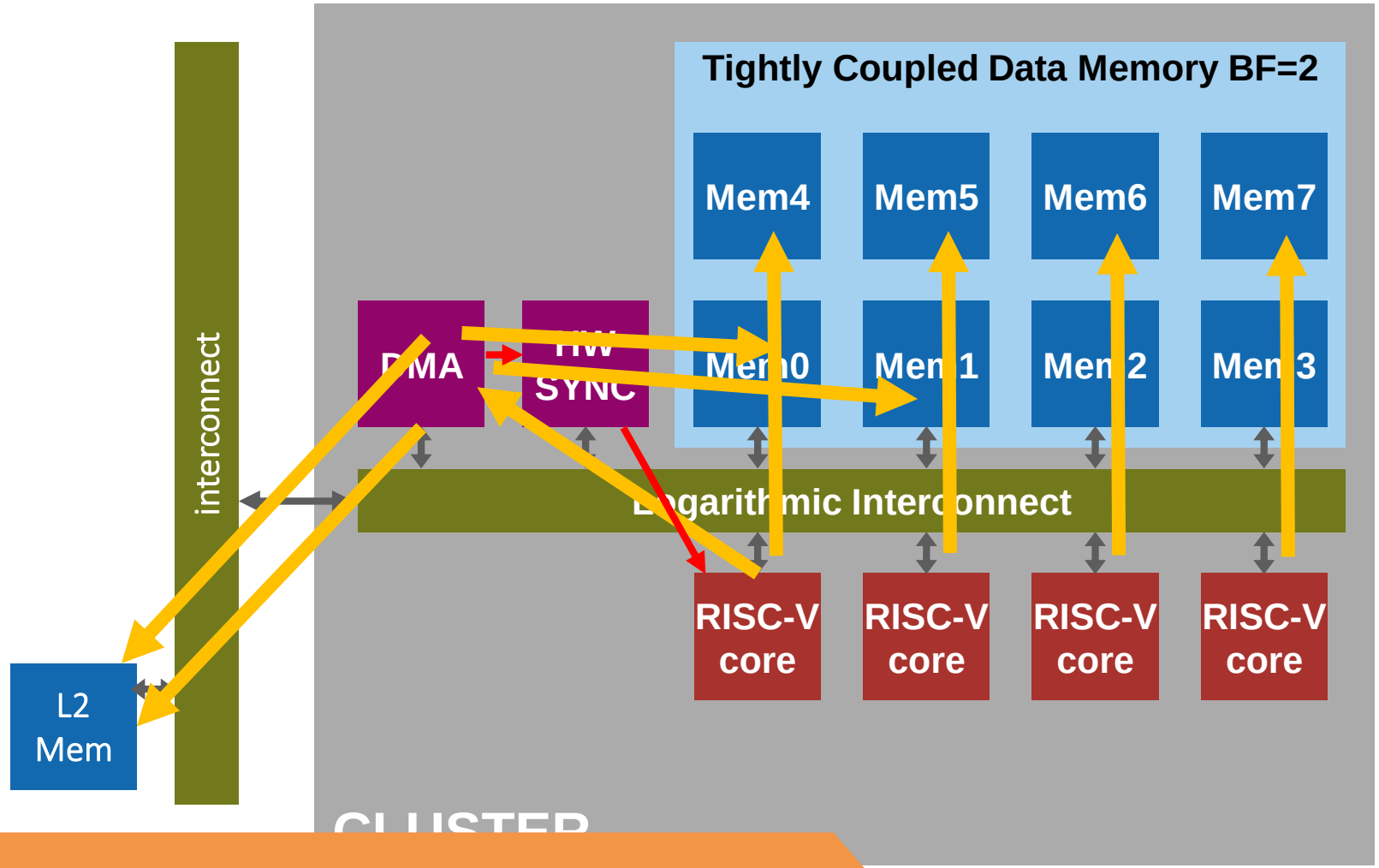


Trade-off between memory size and latency

DMA based, non-blocking mem copy with fast sync.







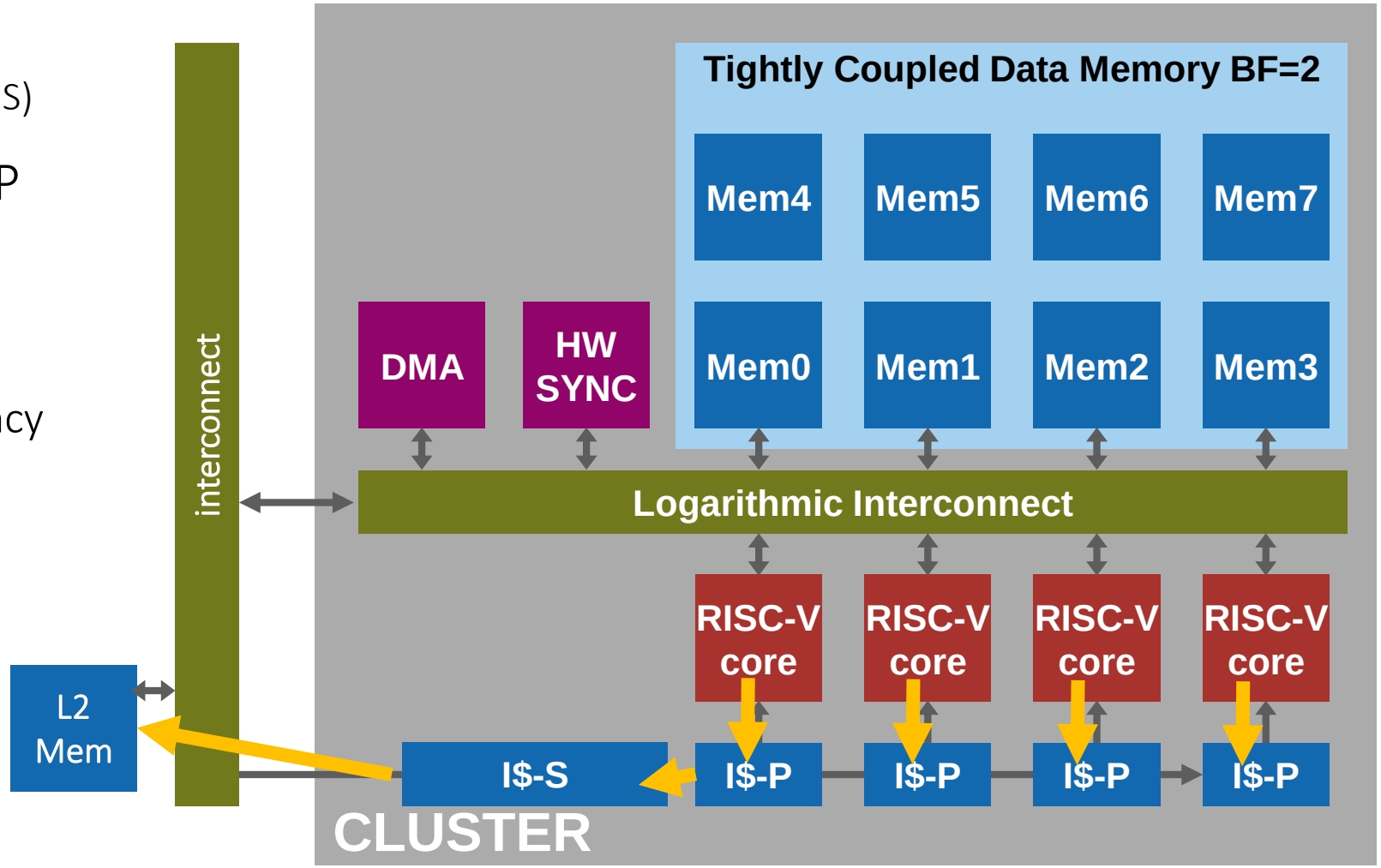
~15x latency and energy reduction for a barrier

[Glaser TPDS20]

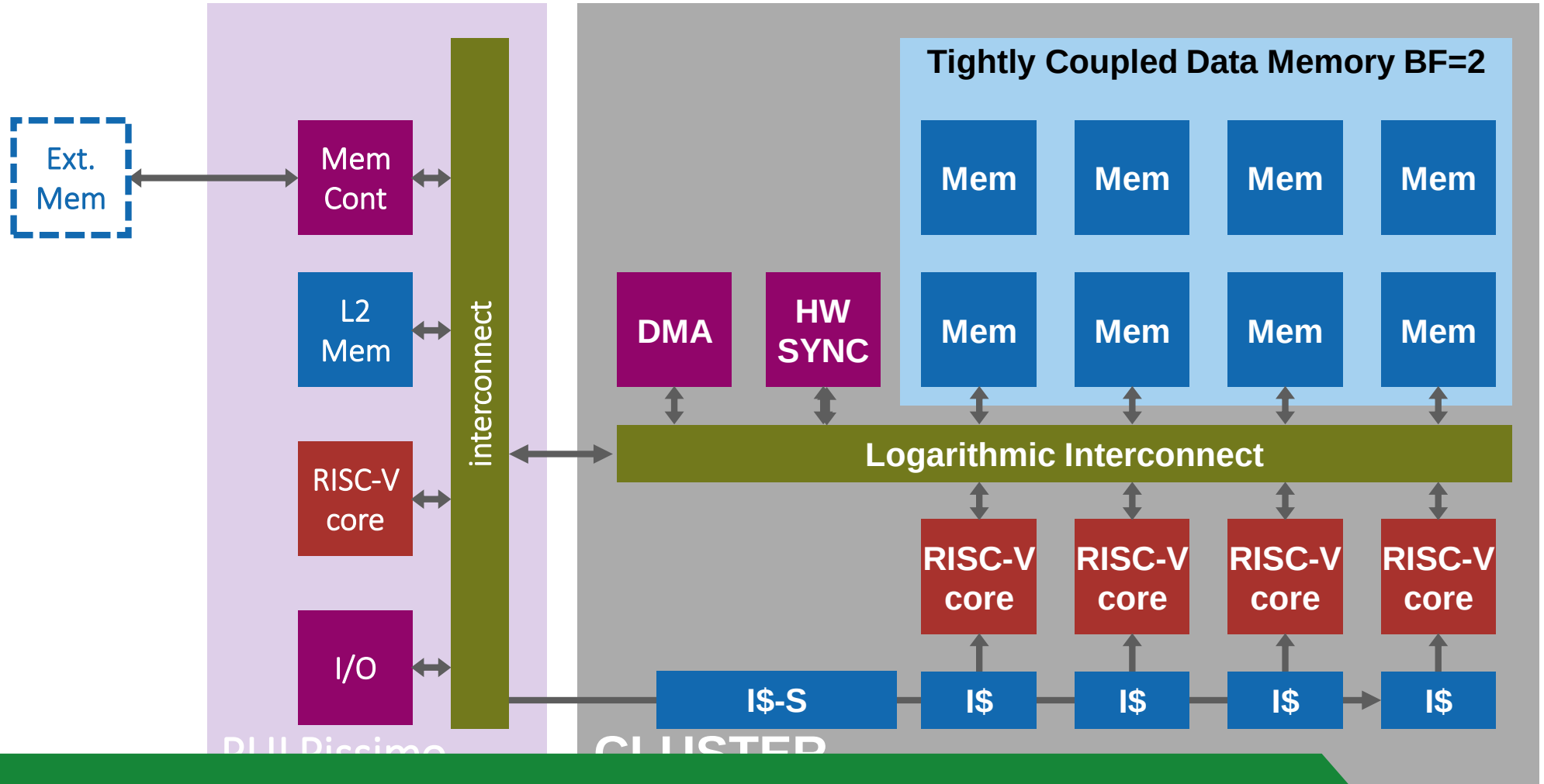
Sabancı Üniversitesi

Shared instruction cache with private “loop buffer”

- Two-level I$
  - Private (P) + Shared (S)
- Most IFs from I$-P
  - Low IF energy
- I$-S for capacity
  - Reduces miss latency



Host for sequential, I/O + Data-Parallel Accelerator Cluster

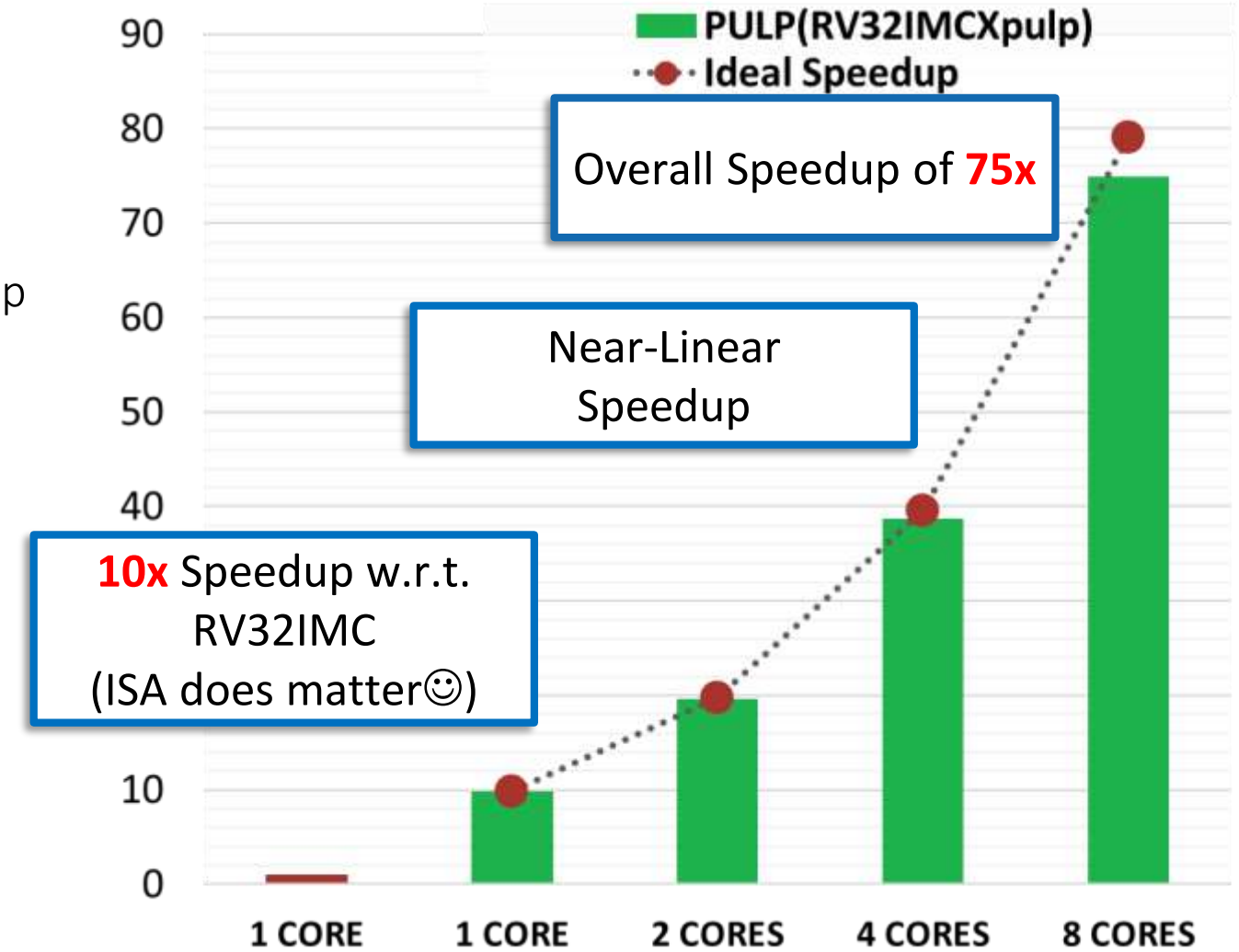


https://github.com/pulp-platform/pulp

Sabancı Üniversitesi

Combining ISA extension + Efficient parallel execution

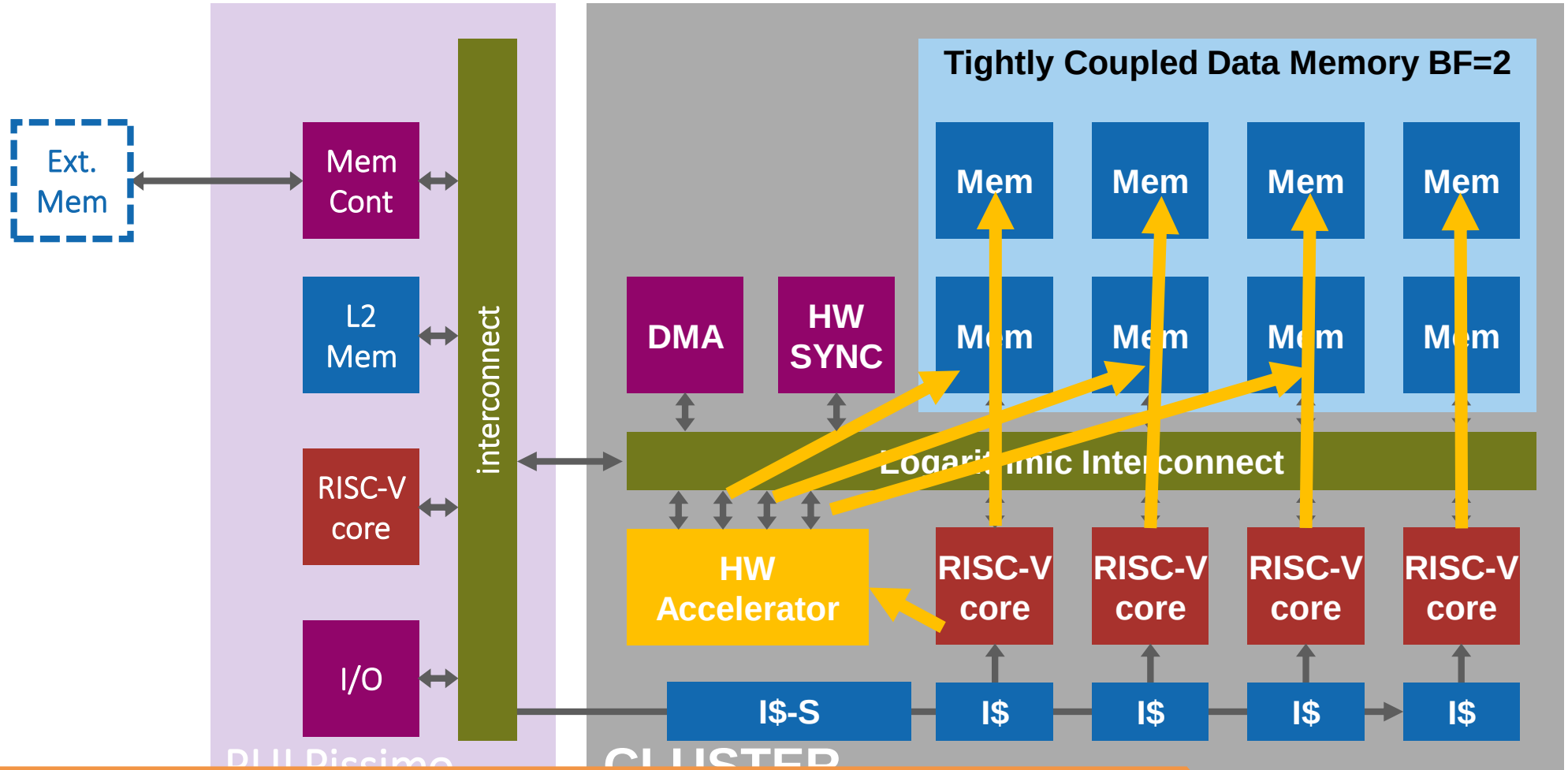
- 8-bit convolution
  - Open source DNN library
- 10x through xPULP
  - Extensions bring real speedup
- Near-linear speedup
  - Scales well for regular workloads
- 75x overall gain
- 7-8 GMACs
  - 250MHz
  - 4 MAC/Cycle (8bit)
  - 8 Cores



[Garofalo et al. Philos. Trans. R. Soc 20]

More GOPS, Less Power?

What's next? Tightly-coupled accelerators

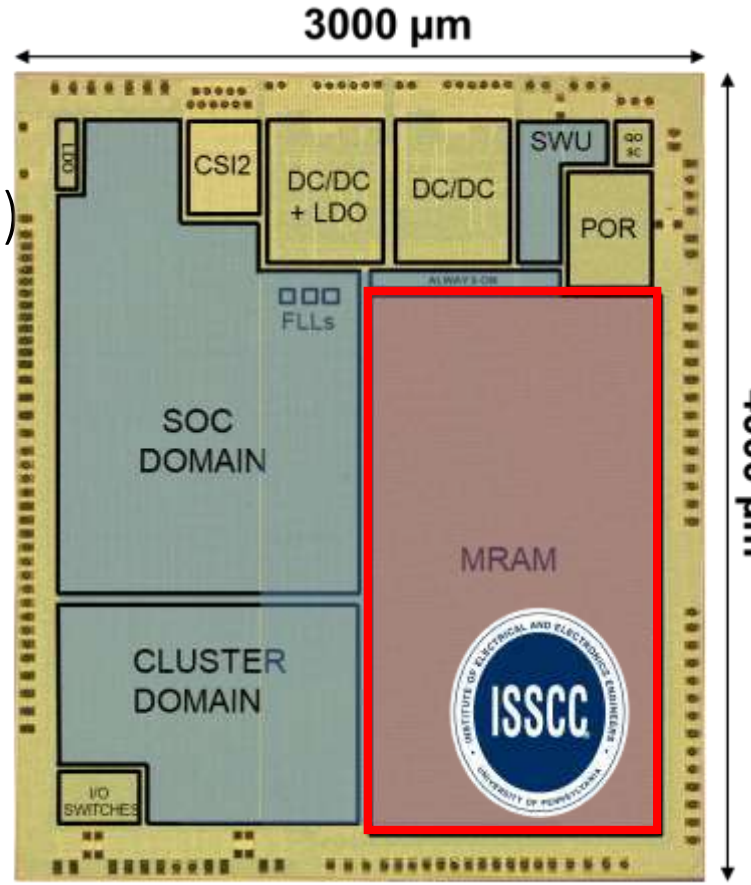


Acceleration with flexibility: zero-copy HW-SW cooperation

Sabancı Üniversitesi

All together in VEGA: Extreme Edge IoT Processor

- RISC-V cluster (9 cores +1)
  - 614GOPS/W @ 7.6GOPS (8bit DNNs),
  - 79GFLOPS/W @ 1GFLOP (32bit FP)
- Multi-precision HWCE(4b/8b/16b)
  - 3×3×3 MACs with normalization / activation:
  - 32.2GOPS and 1.3TOPS/W (8bit)
- 1.7 µW cognitive unit for autonomous wake-up from retentive sleep mode
- On chip voltage/frequency regulation



In cooperation with GREENWAVES TECHNOLOGIES

Technology	22nm FDSOI
Chip Area	12mm^2
SRAM	1.7 MB
MRAM	4 MB
VDD range	0.5V - 0.8V
VBB range	0V - 1.1V
Freq. Range	32 kHz - 450 MHz
Power Range	1.7 µW - 49.4 mW

[D. Rossi, ISSCC21]

Fully-on chip DNN inference with 4MB MRAM

Sabancı Üniversitesi

IoT processor like Vega enable autonomous UAVs

Multiple, complex, heterogeneous tasks at high speed and robustness **fully on board**

Object detection

Obstacle avoidance & Navigation

Environment exploration

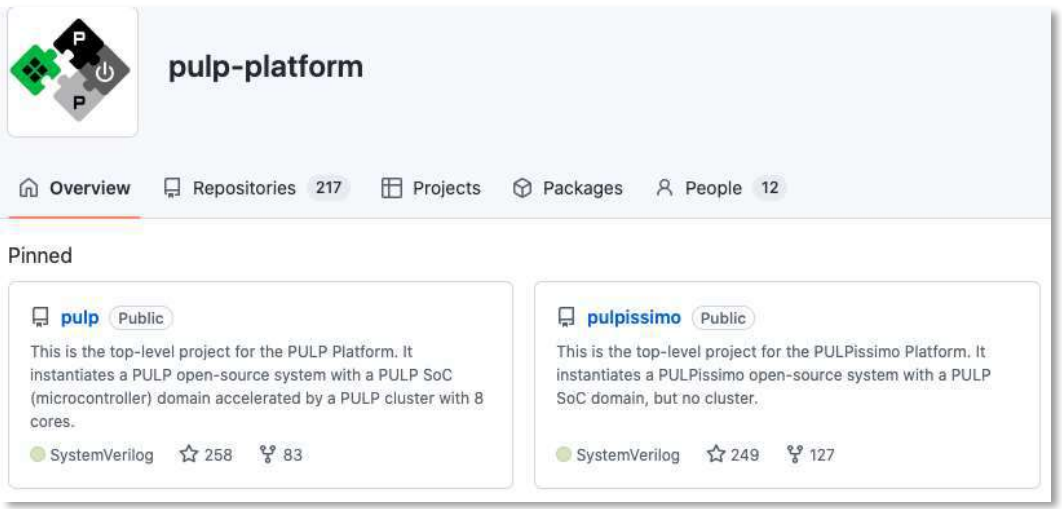
Multi-GOPS workload at extreme efficiency → P_{max} 100mW

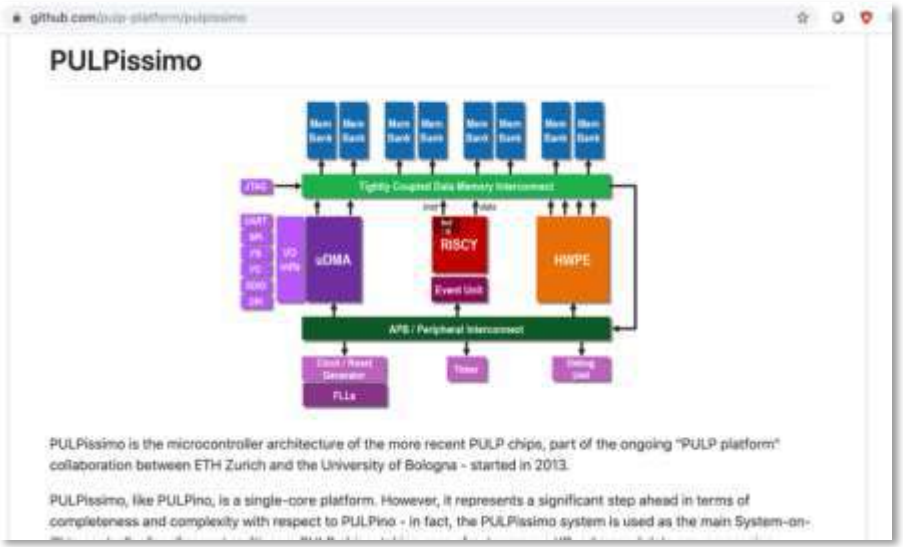
PULP uses a permissive open source license

- All our development is on GitHub
  - HDL source code, testbenches, software development kit, virtual platform

https://github.com/pulp-platform

- Allows anyone to use, change, and make products without restrictions.





For us open source is a necessity to manage our projects!

- Modern IC design is complex and expensive
  - We need partners to help and collaborate
  - We need support (IPs, donations) to realize designs

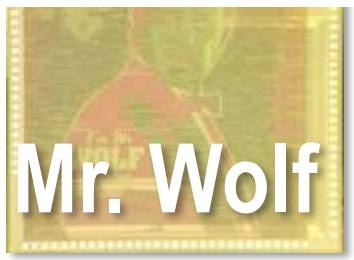
Open Source to the rescue

- Makes it easy to collaborate with external partners (both industrial and academic)
  - Less paperwork/NDAs to get started
  - Partners see/are aware of what we provide
- What we do can be re-used (permissive licensing) by our partners
- Results can be more easily verified

Open source collaboration scheme explained

Direct research collaborators on PULP

- Politecnico di Torino
- University of Cambridge
- USI Lugano
- TU Kaiserslautern
- University of Cagliari
- IBM Research Zurich
- EPF Lausanne
- CSEM Neuchatel
- Princeton University
- Technische Universität Graz
- CEA-Leti Grenoble
- Fraunhofer-Gesellschaft
- Sapienza Università di Roma



commits

commits

GitHub

Academic users we are aware of

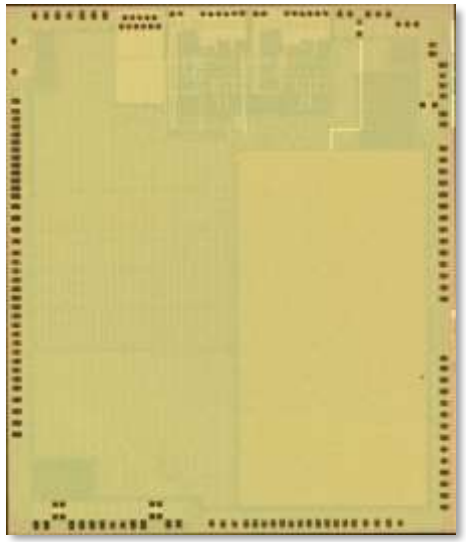
- Università di Genova
- Politecnico di Milano
- Fondazione Bruno Kessler
- Lund University
- Stanford University
- UC Los Angeles
- UC San Diego
- Columbia University
- Universitat Bar-Ilan
- İstanbul Teknik Üniversitesi
- NCTU Hsinchu
- University of Zagreb, FER
- TUT Tampere
- RWTH Aachen
- IST University of Lisboa
- UFRN Rio Grande do Norte
- TU Darmstadt
- Universität Bremen
- Hongik University Seoul
- IIT Kharagpur
- LIRMM Montpelier
- University of Stuttgart
- University of Tübingen
- TU Münich
- FORTH Hellas
- Kyoto University
- Chalmers Göteborg
- FAU Erlangen-Nürnberg
- NTNU Trondheim

The open model led to successful industry collaborations

QuickLogic®

Arnold (GF22)

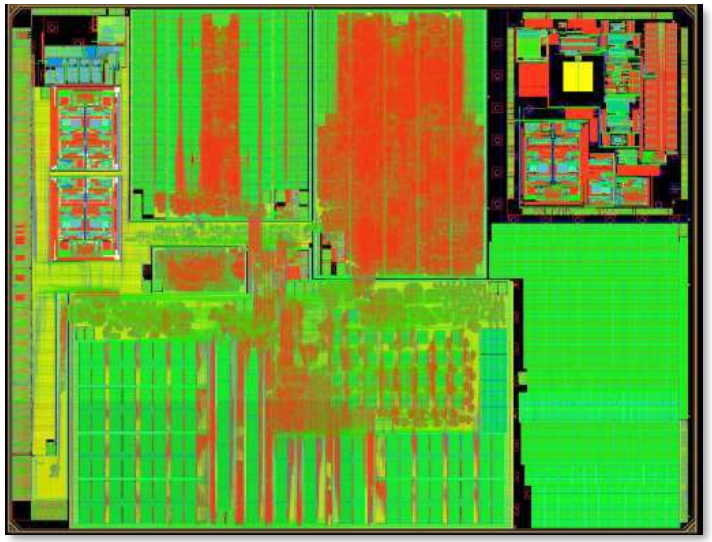
eFPGA with RISC-V core



GREENWAVES TECHNOLOGIES

Vega (GF22)

IoT Processor with ML acceleration



DOLPHIN INTEGRATION
The enabler of low-power Systems-on-Chip

Marsellus (GF22)

IoT Processor with low power modes and event based computing

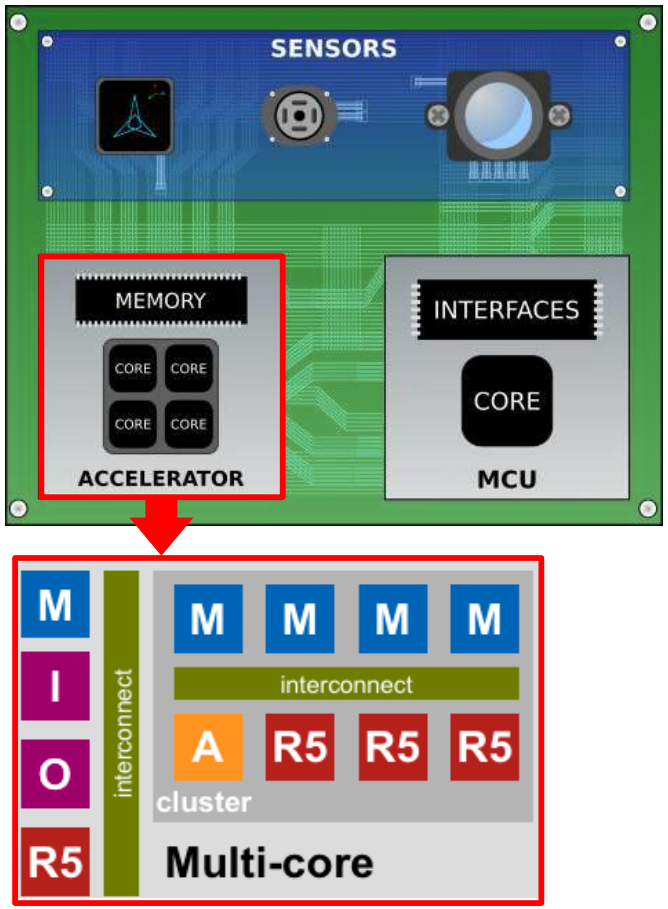
Meta numem

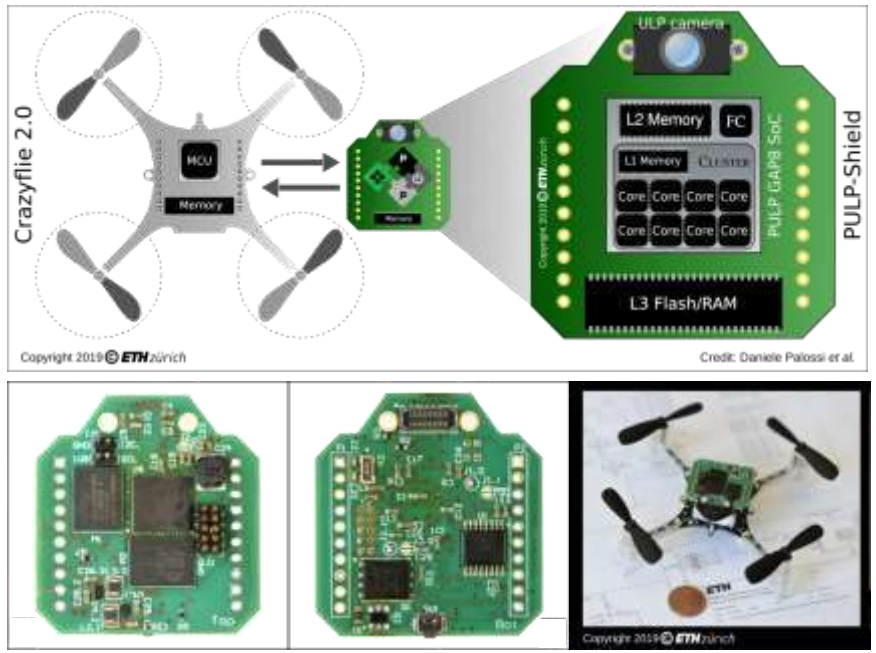
Siracusa (TSMC16)

IoT Processor with NVM technology

The PULP-Based Commercial Nanodrone Platform

ULP heterogeneous model [1] → **PULP-Shield [2]** → **AI-deck**





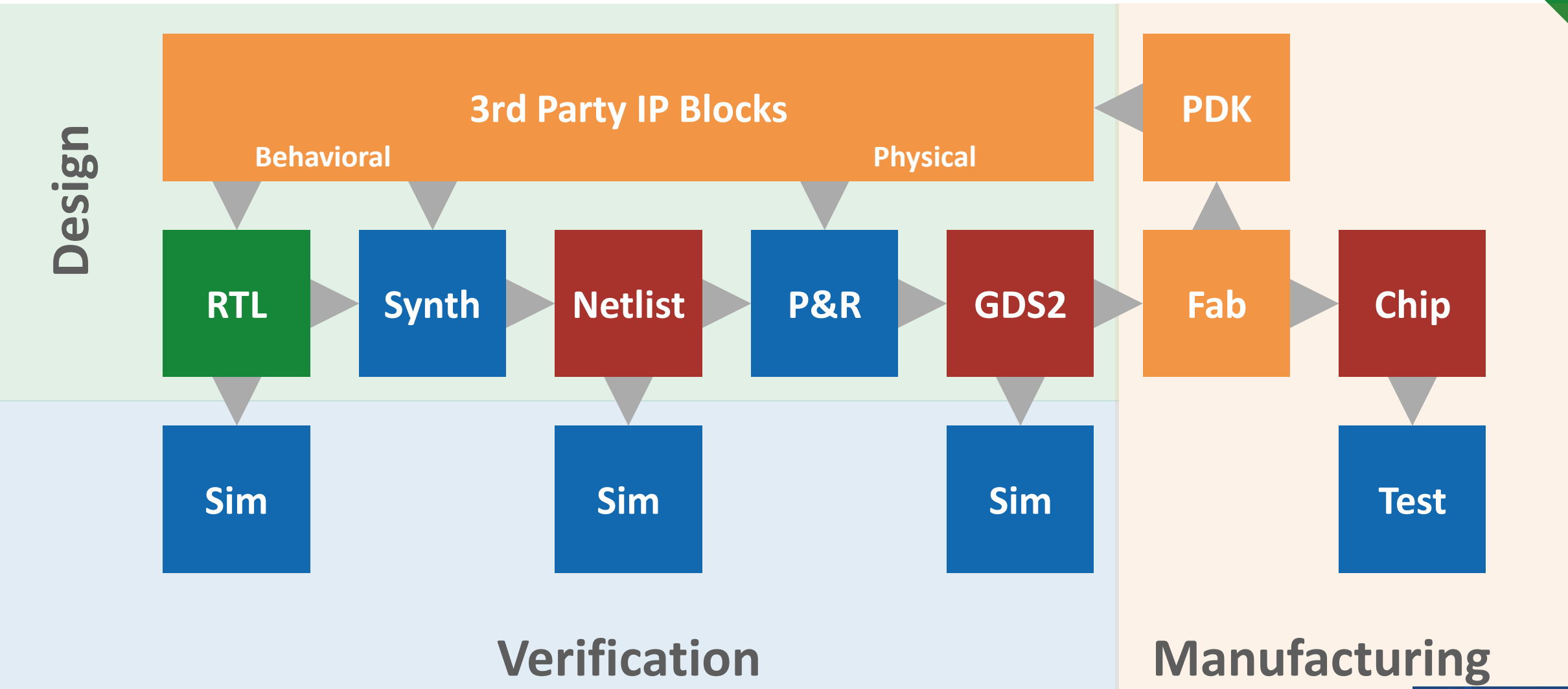
- ~ 5 g – 30x28 mm
- PULP GAP8 SoC
- Off-chip DRAM/Flash
- QVGA ULP Camera
- **Open source hardware**

GREENWAVES TECHNOLOGIES
ALTIUM DESIGNER
GitHub

- ~ 8 g – 40x28 mm
- PULP GAP8 SoC
- Off-chip DRAM/Flash
- QVGA ULP Camera
- WiFi module

[1] F. Conti, D. Palossi, A. Marongiu, D. Rossi, and L. Benini. "Enabling the heterogeneous accelerator model on ultra-low power microcontroller platforms." IEEE DATE, *2016.*
[2] D. Palossi, F. Conti, and L. Benini "An open source and open hardware deep learning-powered visual navigation engine for autonomous nano-UAVs." IEEE DCOSS, *2019.*

The complicated relationship of Open Source Hardware



State of Open Source for Hardware: Rapid Developments

TYPE	EXAMPLES	STATUS
Open Specifications	RISC-V	Established
Architectures	PULP	Quite mature
Implementations in RTL	Snitch, Hero	Many

State of Open Source for Hardware: Rapid Developments

TYPE	EXAMPLES	STATUS
Open Specifications	RISC-V	Established
Architectures	PULP	Quite mature
Implementations in RTL	Snitch, Hero	Many
Open source Hard IP	FLL, DDR PHY..	Very Limited

State of Open Source for Hardware: Rapid Developments

TYPE	EXAMPLES	STATUS
Open Specifications	RISC-V	Established
Architectures	PULP	Quite mature
Implementations in RTL	Snitch, Hero	Many
Open source Hard IP	FLL, DDR PHY..	Very Limited
Process Design Kits	Skywater 130nm	Just Started

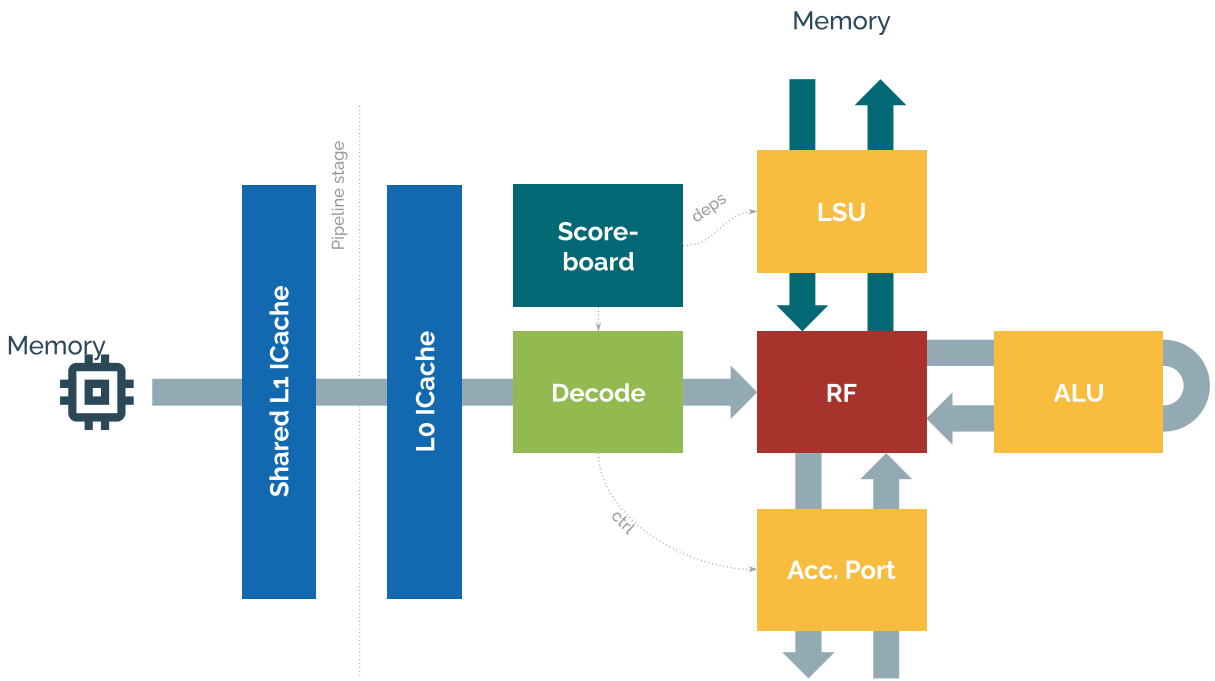
State of Open Source for Hardware: Rapid Developments

TYPE	EXAMPLES	STATUS
Open Specifications	RISC-V	Established
Architectures	PULP	Quite mature
Implementations in RTL	Snitch, Hero	Many
Open source Hard IP	FLL, DDR PHY..	Very Limited
Process Design Kits	Skywater 130nm	Just Started
Open Source Tools	Open Lane	On its way

Why is RISC-V so special: Freedom to Explore and **Fail!**

- The ISA provides a contract between HW and SW
  - As long as you stick to the ISA, you can develop HW and SW independently
  - All RISC-V research in HW can continue to rely on growing SW ecosystem for RISC-V
- RISC-V comes with plenty of options for extensions
  - There are reserved encoding spaces for instruction set extensions
- Being able to change everything gives great flexibility
  - Do you want 33 registers, or a 48 bit accumulator.. No problem
  - You need to bring the SW support for your addtions.

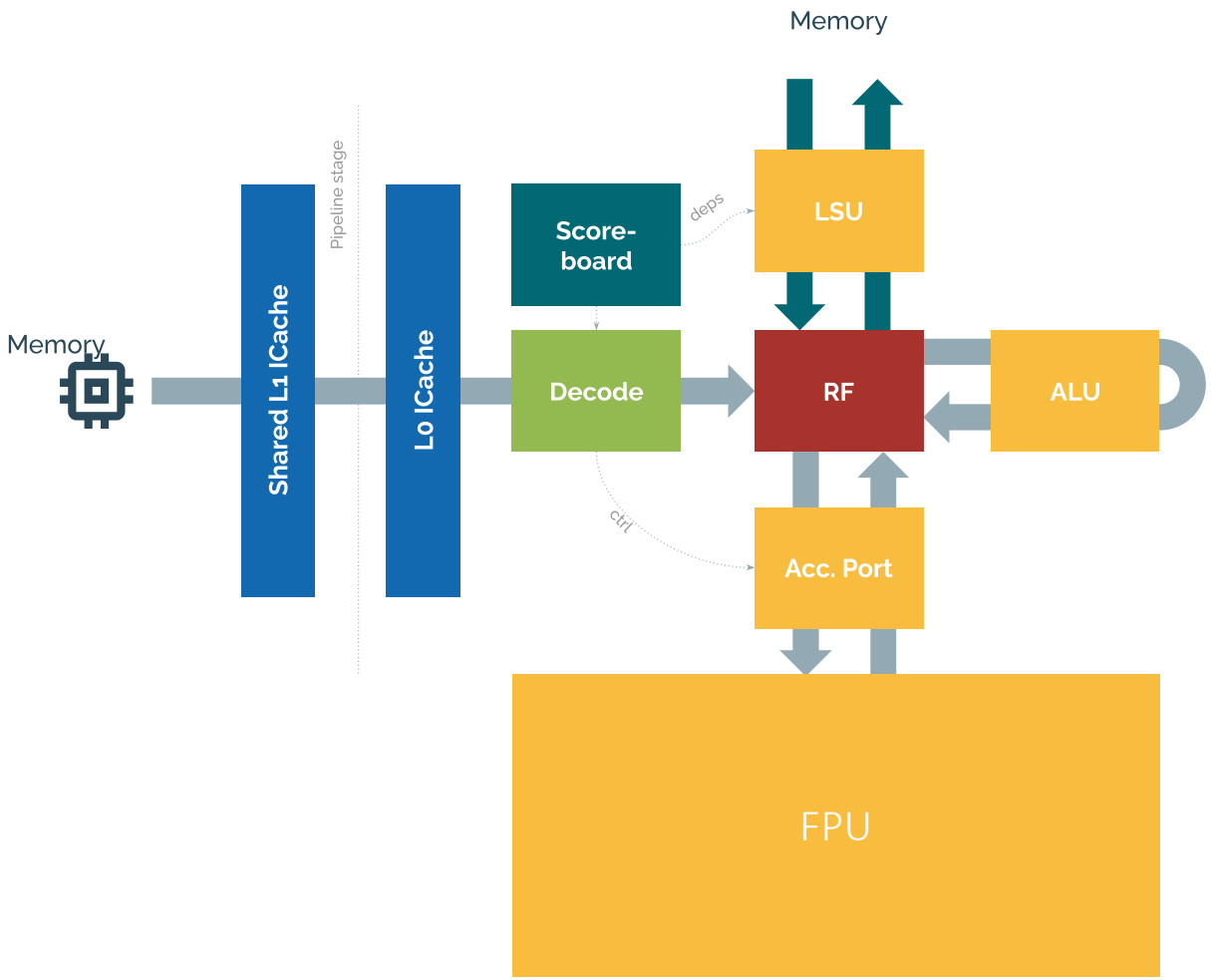
What if we had a tiny 32b core



Introducing SNITCH

- Start with a simple RISC-V core
- Focus on key features:
  - Lightweight microarchitecture
  - Extensibility: Performance through ISA extensions
  - Latency tolerant
  - Competitive frequency
- Around 15-25 kGE

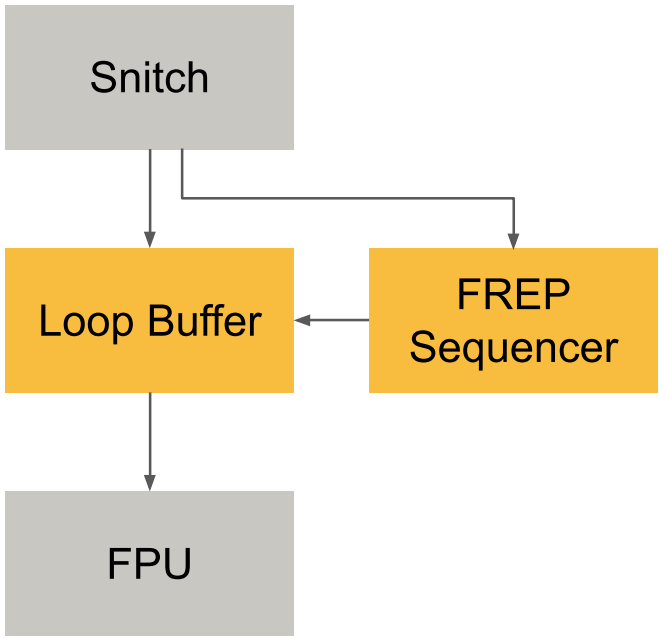
What if we had a tiny 32b core and add a big 64b FPU



Introducing SNITCH

- Start with a simple RISC-V core
- Focus on key features:
  - Lightweight microarchitecture
  - Extensibility: Performance through ISA extensions
  - Latency tolerant
  - Competitive frequency
- Around 15-25 kGE
- Capable 64b FPU with many extensions

What if we add a Floating-point Repetition Buffer? (FREP)



Remove control flow overhead

- Programmable micro-loop buffer
- Sequencer steps through the buffer, independently of the FPU
- Integer core free to operate in parallel: **Pseudo-dual issue**
- High area- and energy-efficiency

```
  mv     r0, zero
loop:
  addi   r0, 1
  fmadd  r2, ssr0, ssr1
  bne    r0, r1, loop
```

→

```
  frep   r1, 1
loop:
  fmadd  r2, ssr0, ssr1
```

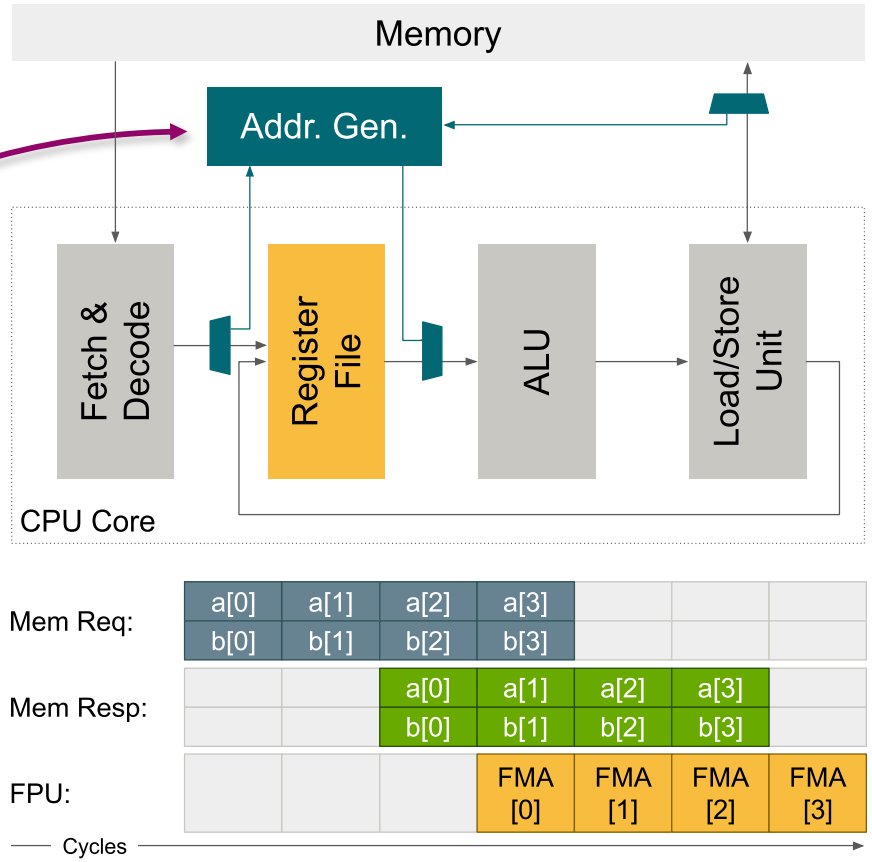
RISC-V

Allows custom instruction set extensions

What if we could stream data to from FPU directly? (SSR)

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

```
loop:                          scfg 0, %[a], ldA
fld r0, %[a]                   scfg 1, %[b], ldB
fld r1, %[b]          ——→      loop:
fmadd r2, r0, r1               fmadd r2, ssr0, ssr1
```



- Increase FPU/ALU utilization by ~3x up to 100%
- SSRs ≠ memory operands
  - Perfect prefetching, latency-tolerant

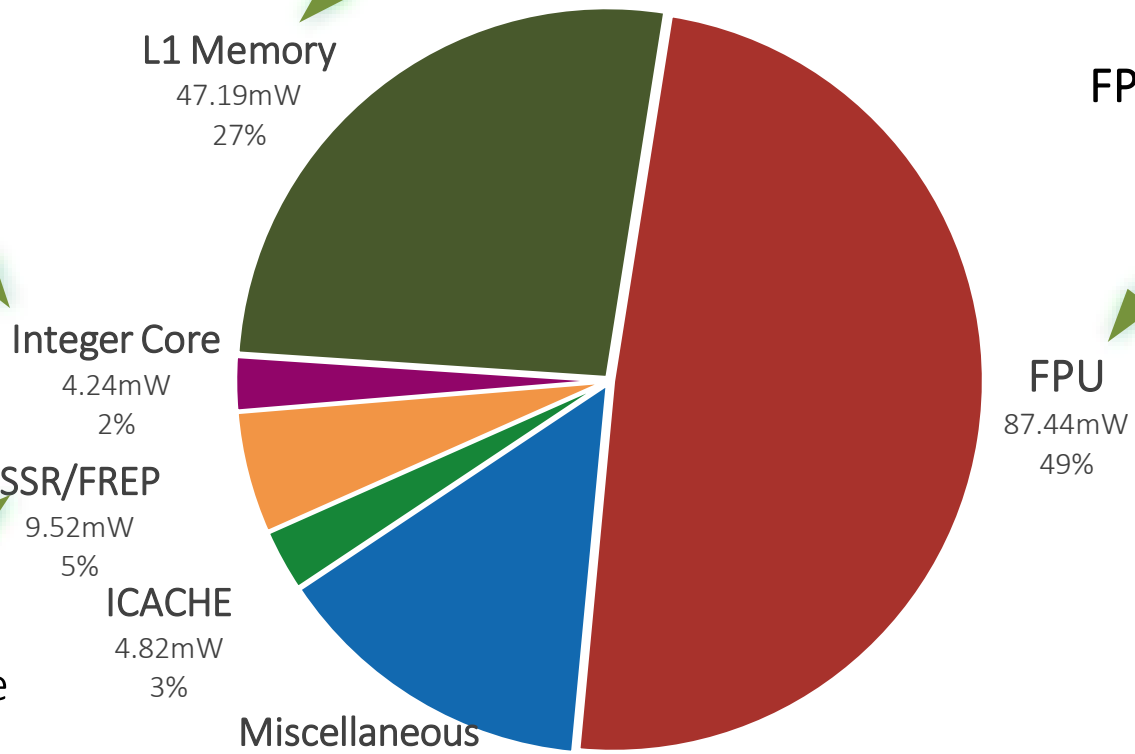
We have a processor that maximizes FPU efficiency

In an 8-core cluster

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

Integer core uses
2% of power

FPU uses 50% of power

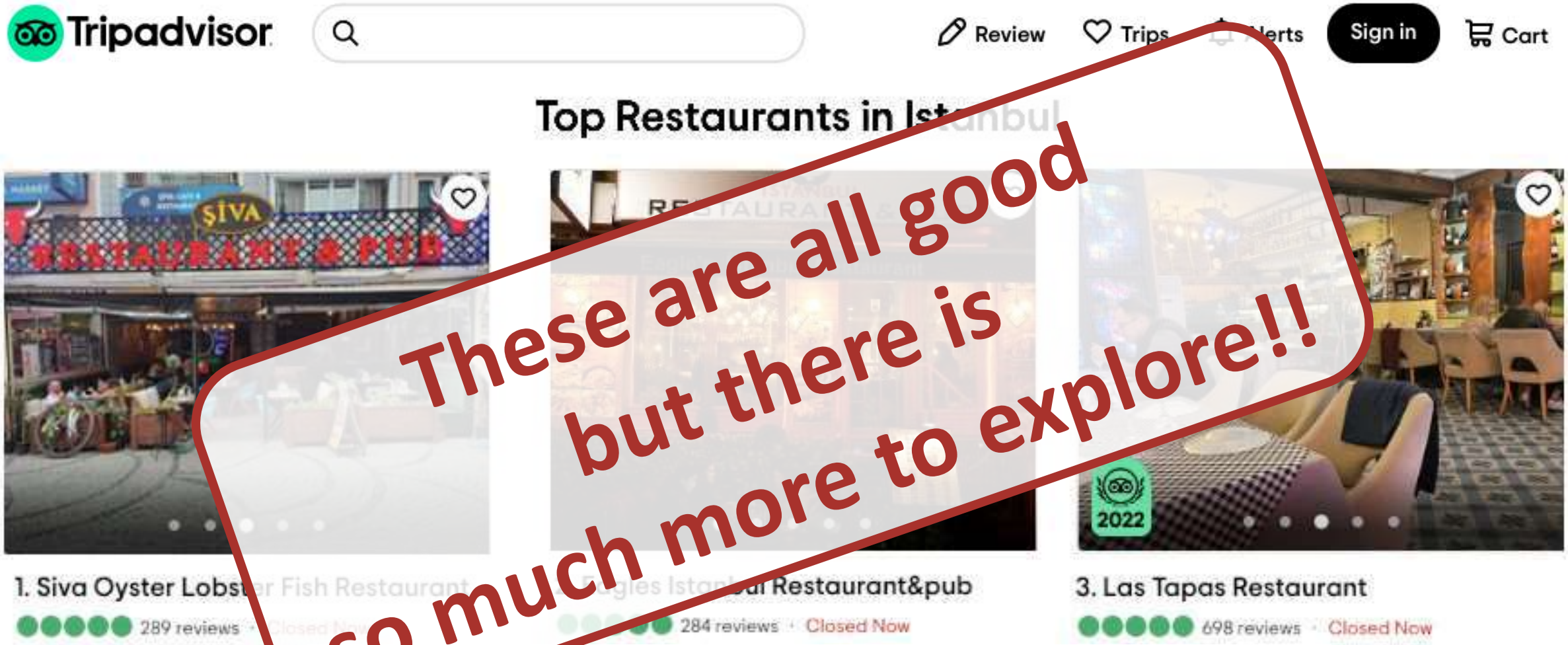


SSR/FREP hardware
uses 5% of power

Spending energy where it counts the most == Efficiency

Sabancı Üniversitesi

What is the most common/accepted way of doing things?



These are all good but there is so much more to explore!!

Heterogeneous + Parallel… Why?

- Processors can do two kinds of useful work:

Decide (jump to different program part)

- Modulate flow of **instructions**
- Mostly sequential decisions:
  - Don't work too much
  - Be clever about the battles you pick (latency is king)
- **Lots of decisions**
  Little number crunching

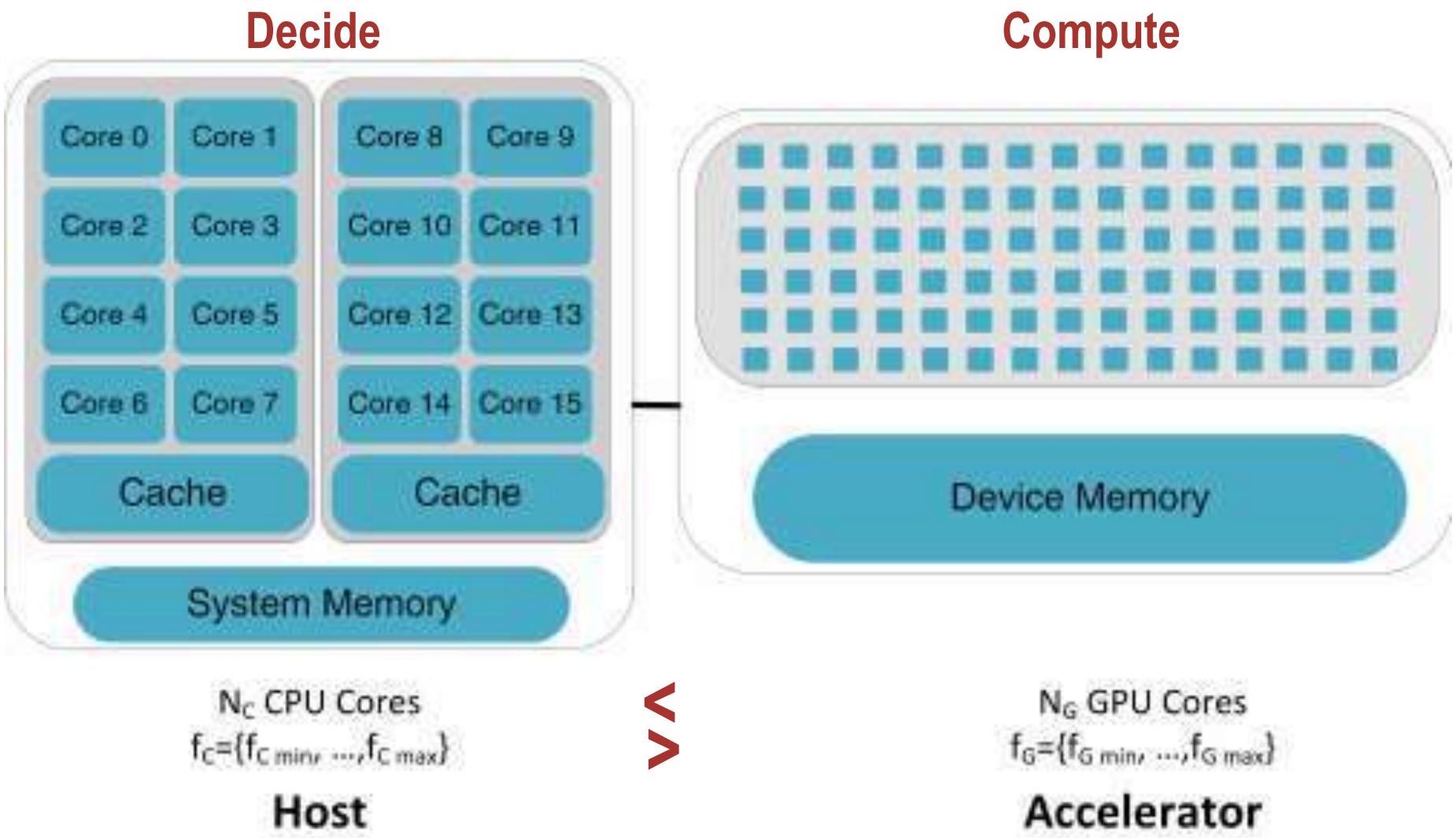
Compute (plough through numbers)

- Modulate flow of **data**
- Embarassingly data parallel:
  - Don't think too much
  - Plough through the data (throughput is king)
- **Few decisions**
  Lots of number crunching

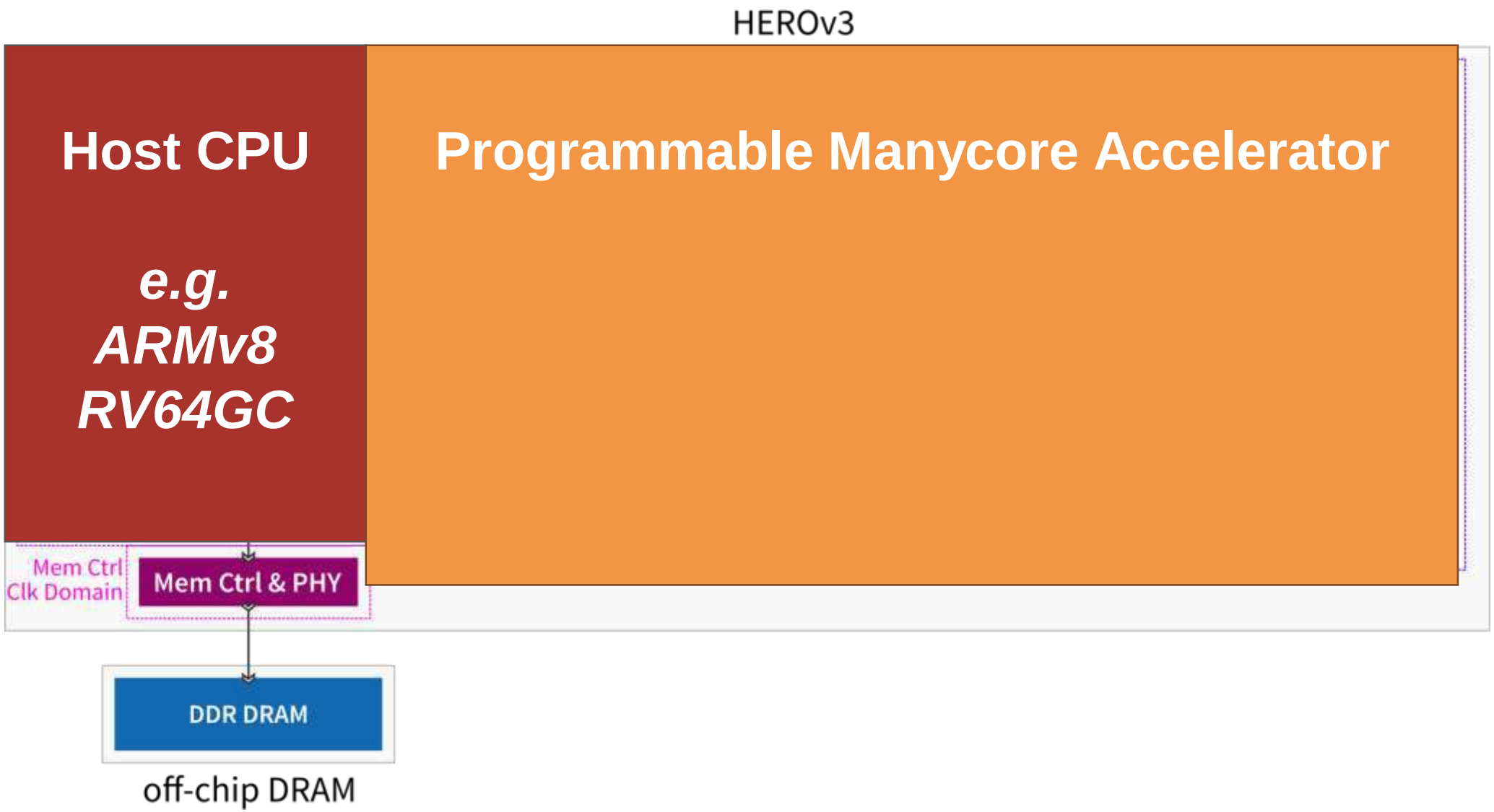
- Today's workloads are dominated by "Compute":
  - Tons of data, few (as fast as possible) decisions based on the computed values,
  - Data-Oblivious Algorithms (ML, or better DNNs are so!)
  - Large data footprint + sparsity

How to design an efficient "Compute" fabric?

Efficient Architecture: Heterogeneous + Parallel



Here is one idea: Use PULP clusters as an accelerator



Introducing Occamy

Dual Chiplet System Occamy:

- Technology: GF12LP+
- Area: 73mm^2

Interposer Hedwig:

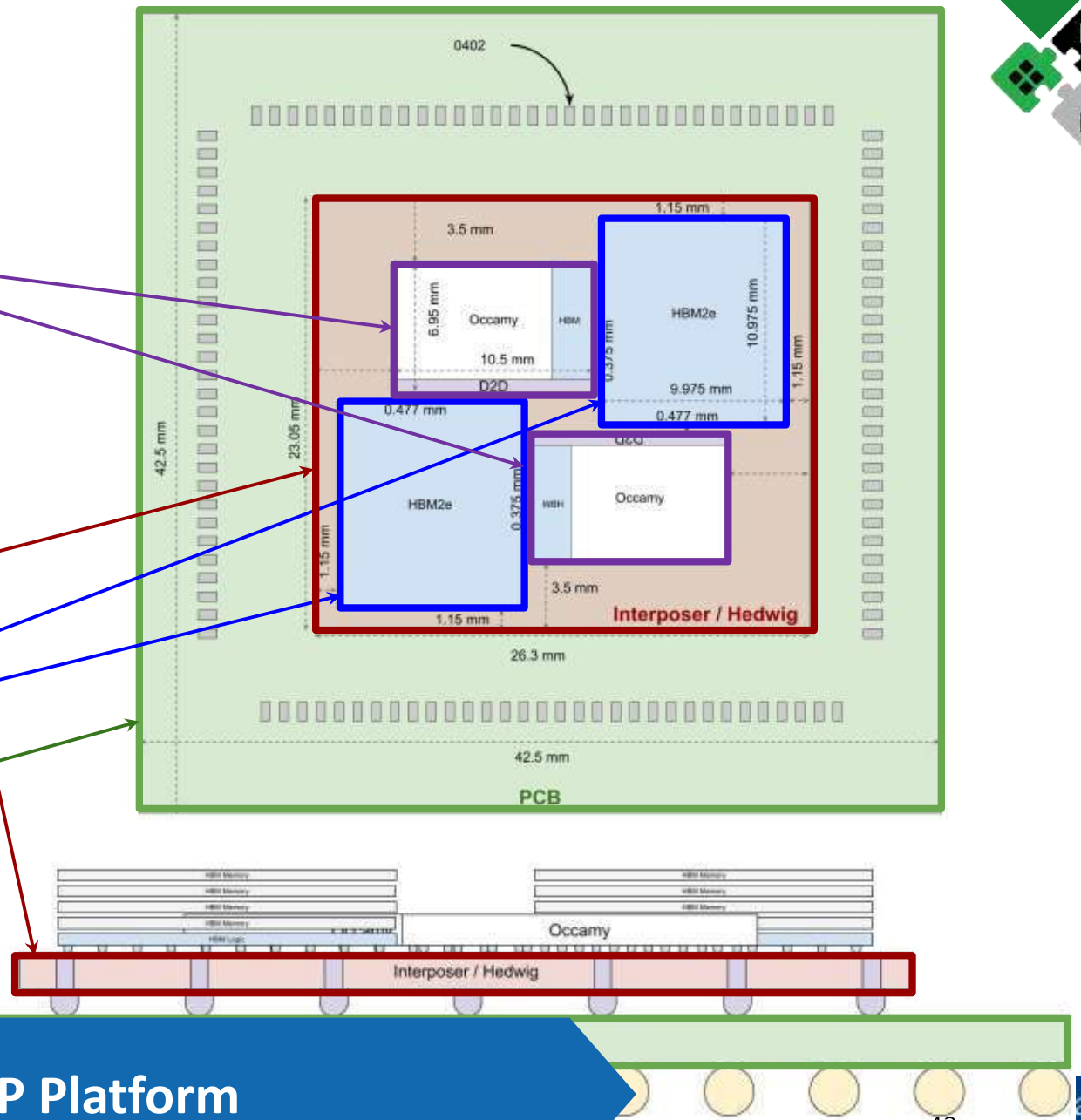
- Technology: 65nm, passive (only BEOL)
- Area: 26.3mm x 23.05mm

HBM2e:

- 16GB HBM2e (Micron)

Fan-out PCB:

- Low-CTE (~3)



Very ambitious project based on PULP Platform

Occamy, ambitious project: needs strong partners

GlobalFoundries™

AVERY design systems

Rambus

EUROPRACTICE IC SERVICE

Micron®

SYNOPSYS®

Fraunhofer IZM

Snitch Cluster, eight + one core for DMA + 128kB memory

8 Snitch compute cores

- Single-stage, small Integer control core

9th Core: DMA

- 512 bit data interface
- Efficient data movement

128 kB TCDM

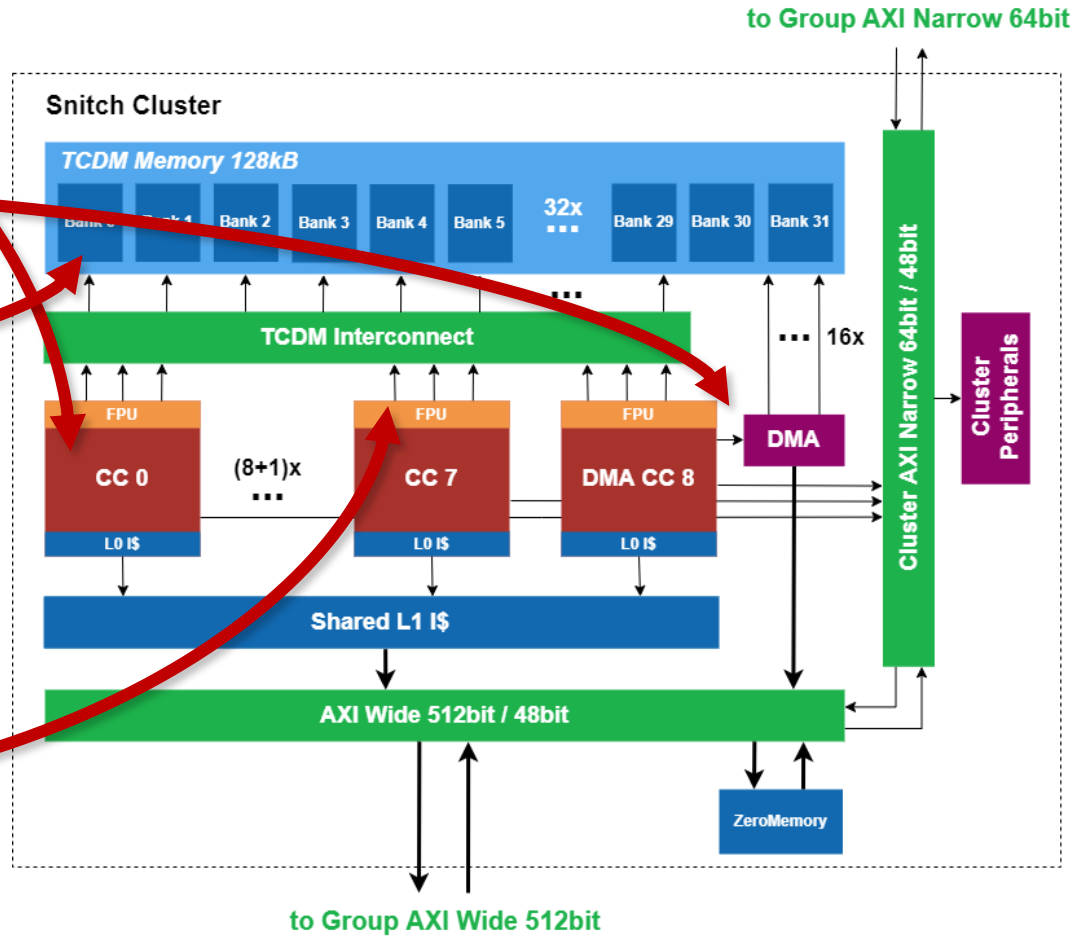
- Scratchpad for predictable memory accesses
- 32 Banks

Custom ISA extensions

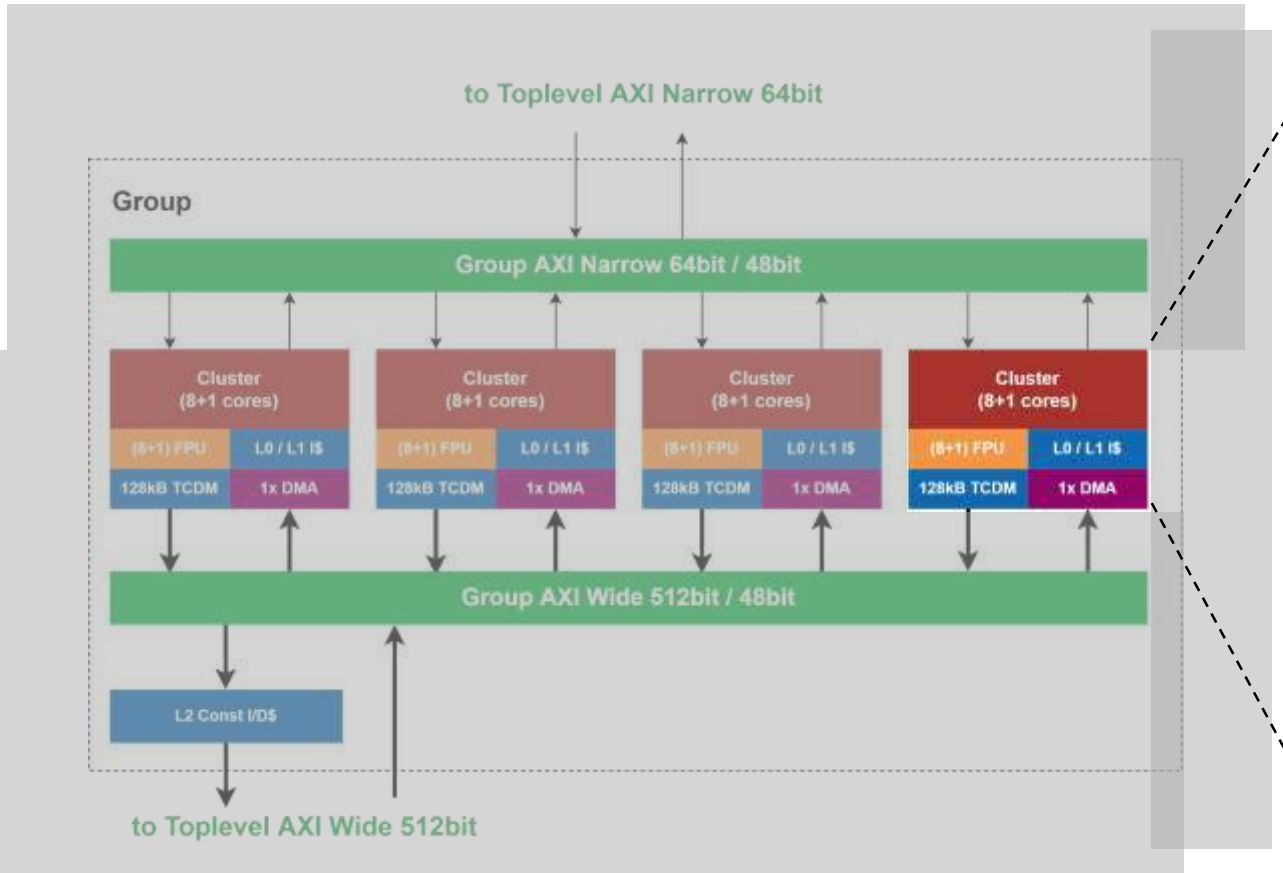
- Xfrep, Xssr
- New: Xissr sparsity support

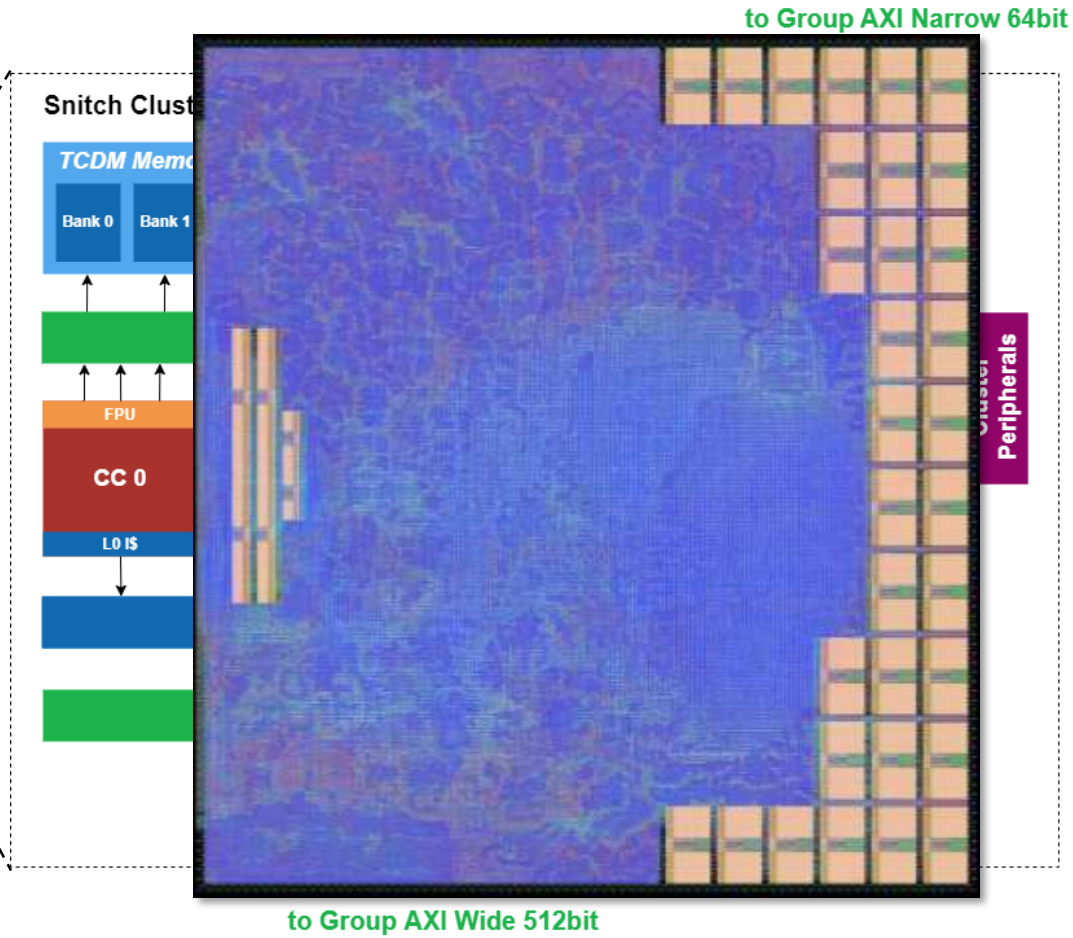
1 FPU per Snitch core

- Decoupled and heavily pipelined
- Multi-format FPU (+SIMD)
- New: Minifloat support + SDOTP

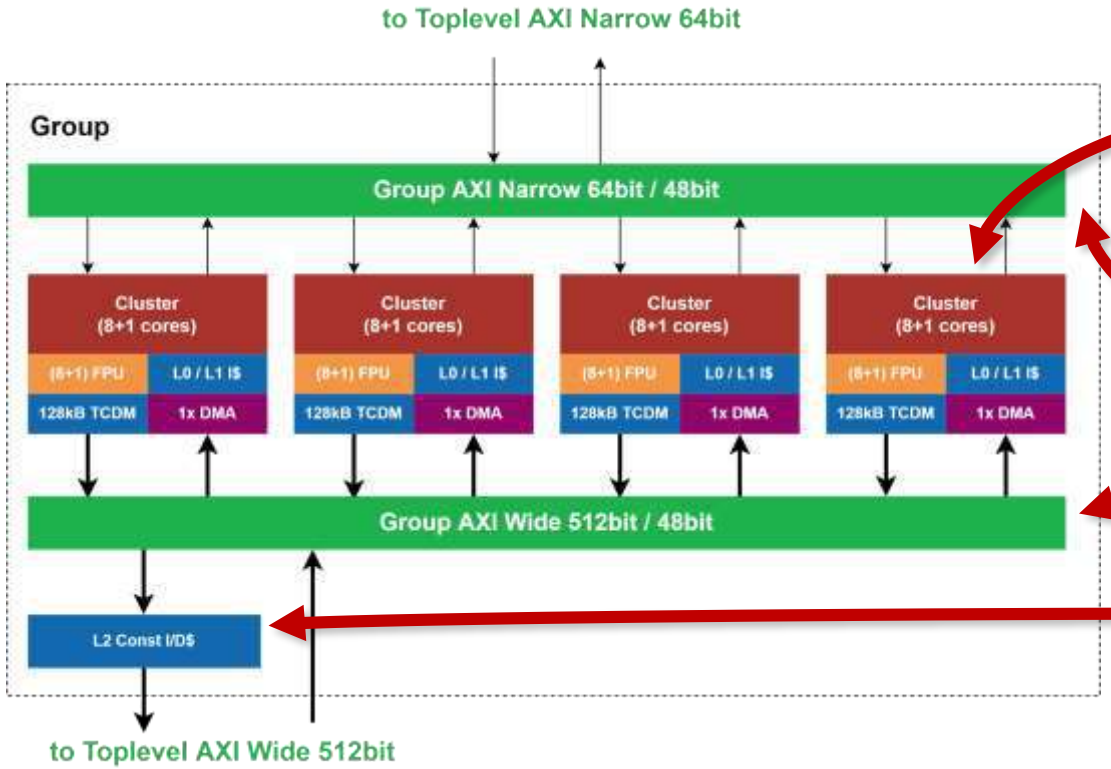


Multiple Snitch clusters form a group





Snitch Group in Occamy: 4 Clusters



4 Clusters per Group

- Single-stage, small Integer control core

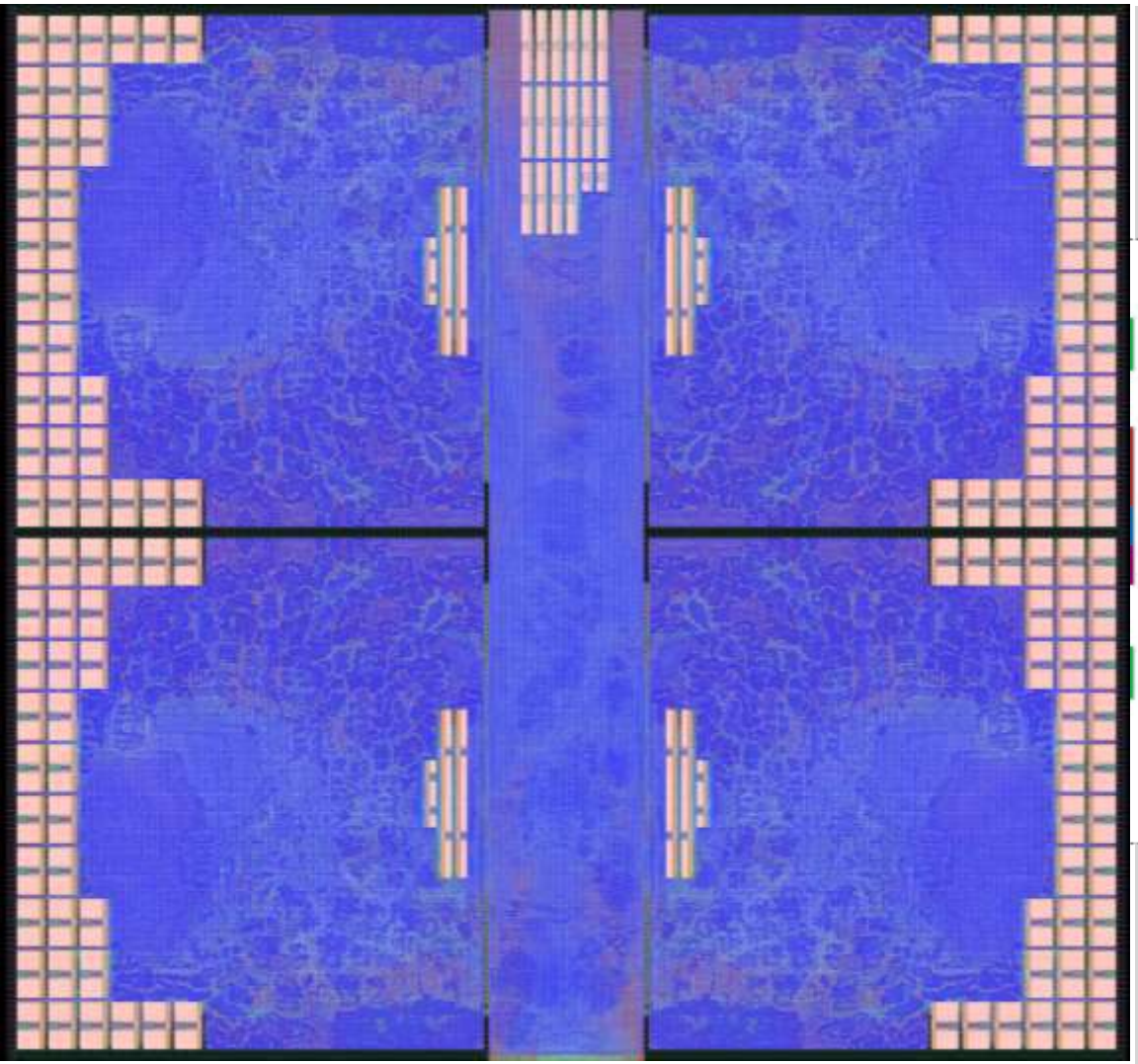
2 AXI Busses

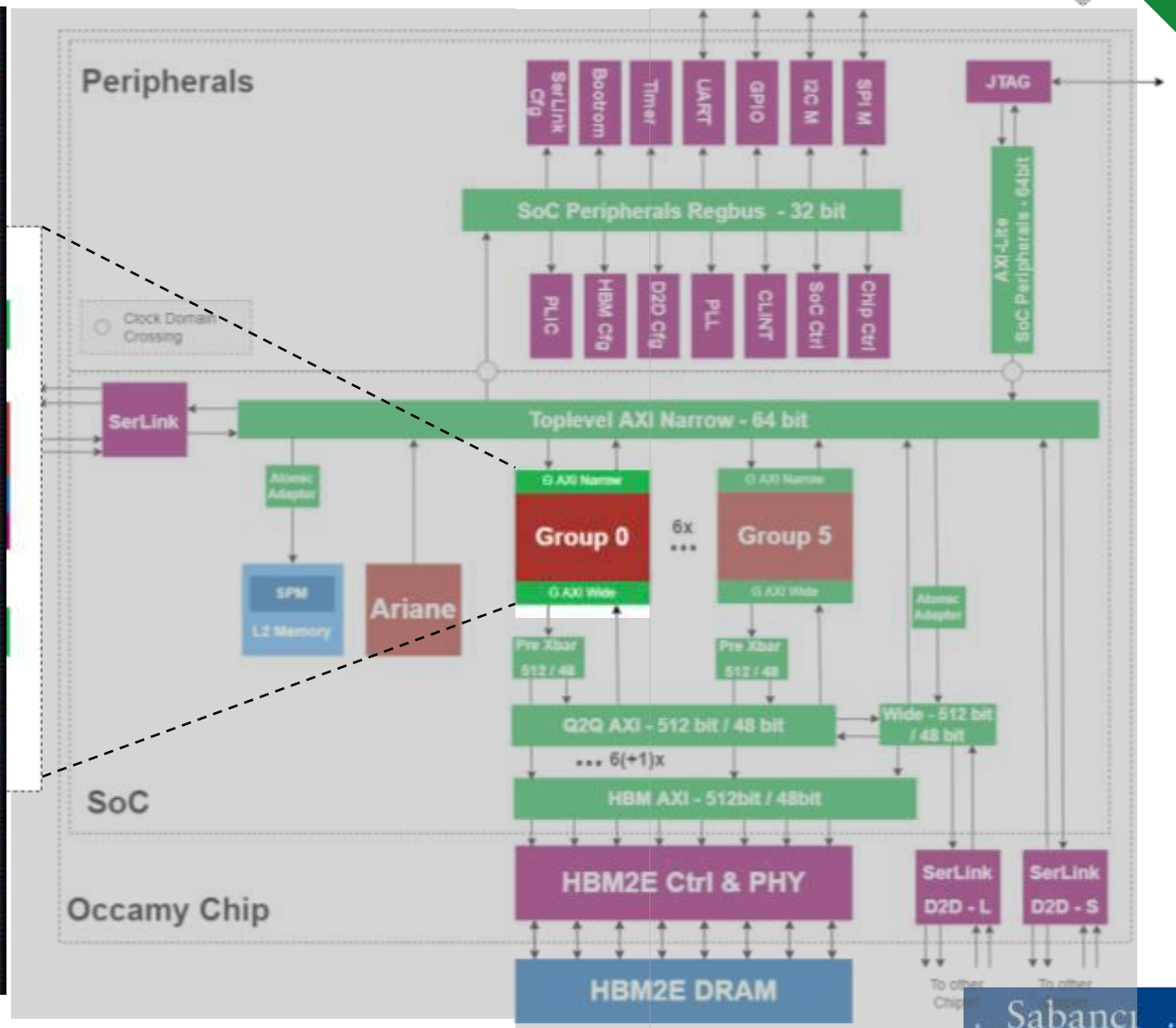
- 64-bit narrow interface: config
- 512-bit wide interface: DMA

Constant Cache

- D/I-Cache hierarchy

Total of Six Snitch Groups in Occamy





Occamy – Finally all together

2 AXI Busses

- 64-bit narrow interface: config
- 512-bit wide interface: DMA

Peripherals

- Complex address space management

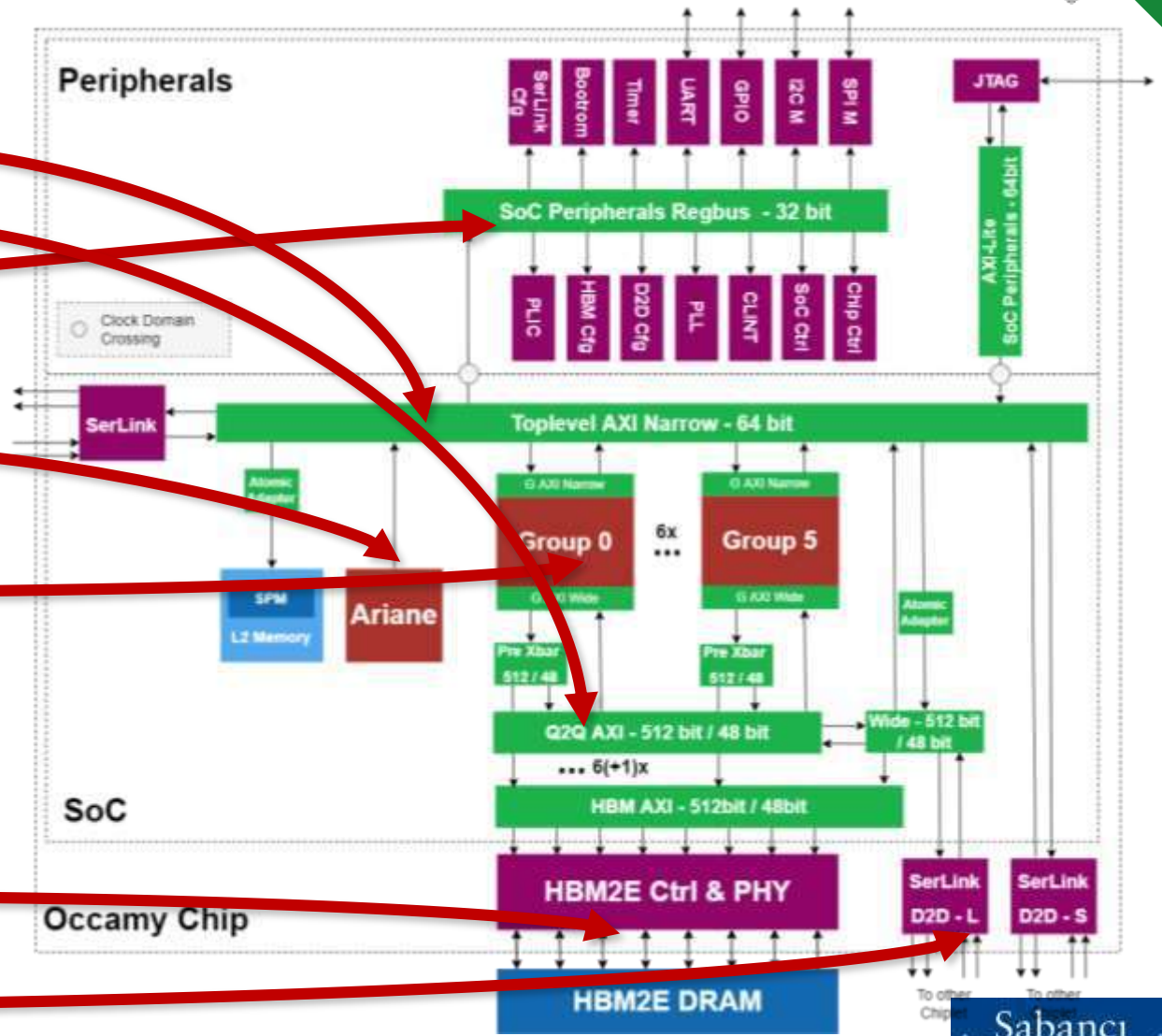
Linux-capable manager core CVA6

6 Groups: 216 cores/die

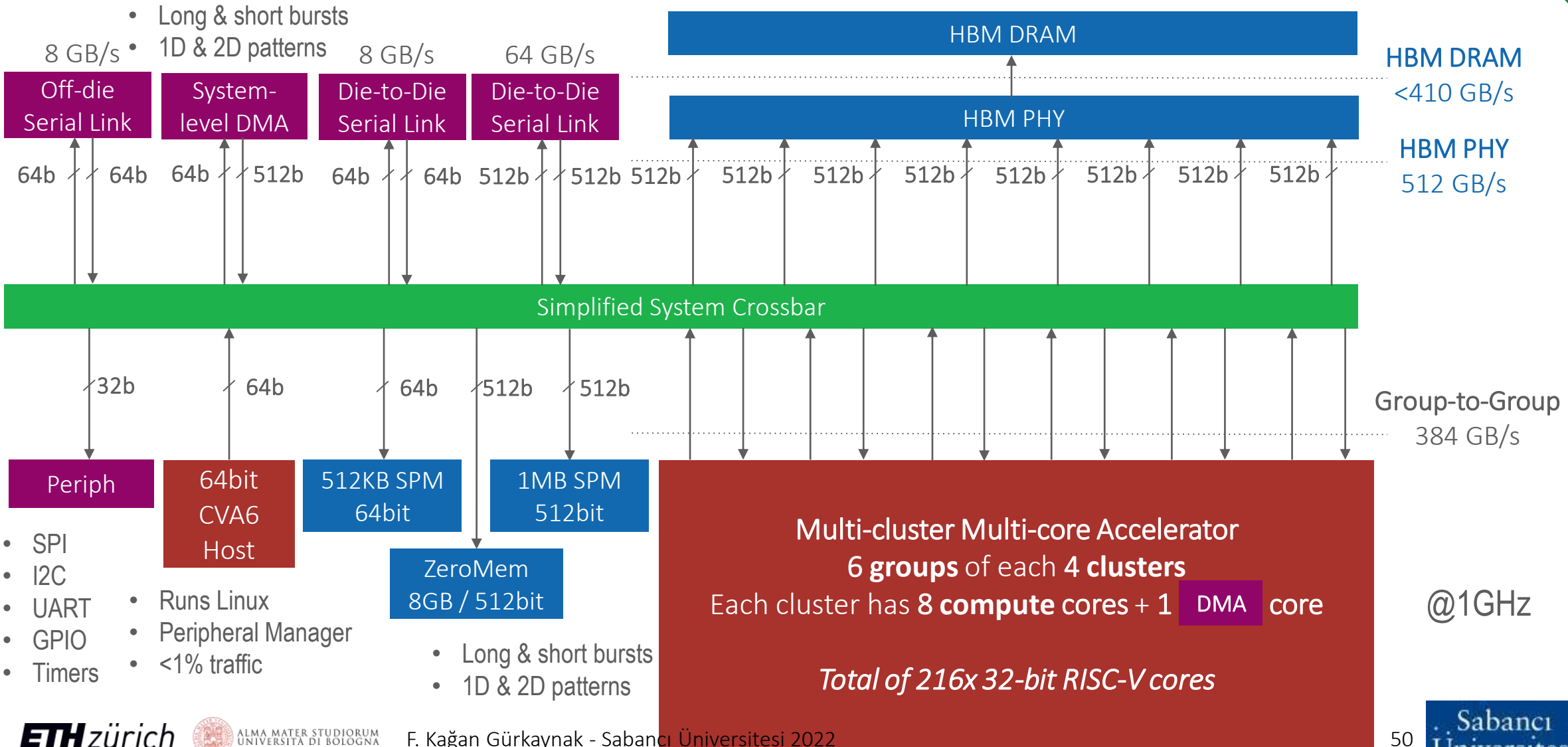
- 4 cluster / group:
  - 8 compute cores / cluster
  - 1 DMA core / cluster
- 512bit Constant Cache

8-channel **HBM2e** (16GB)

D2D serial link

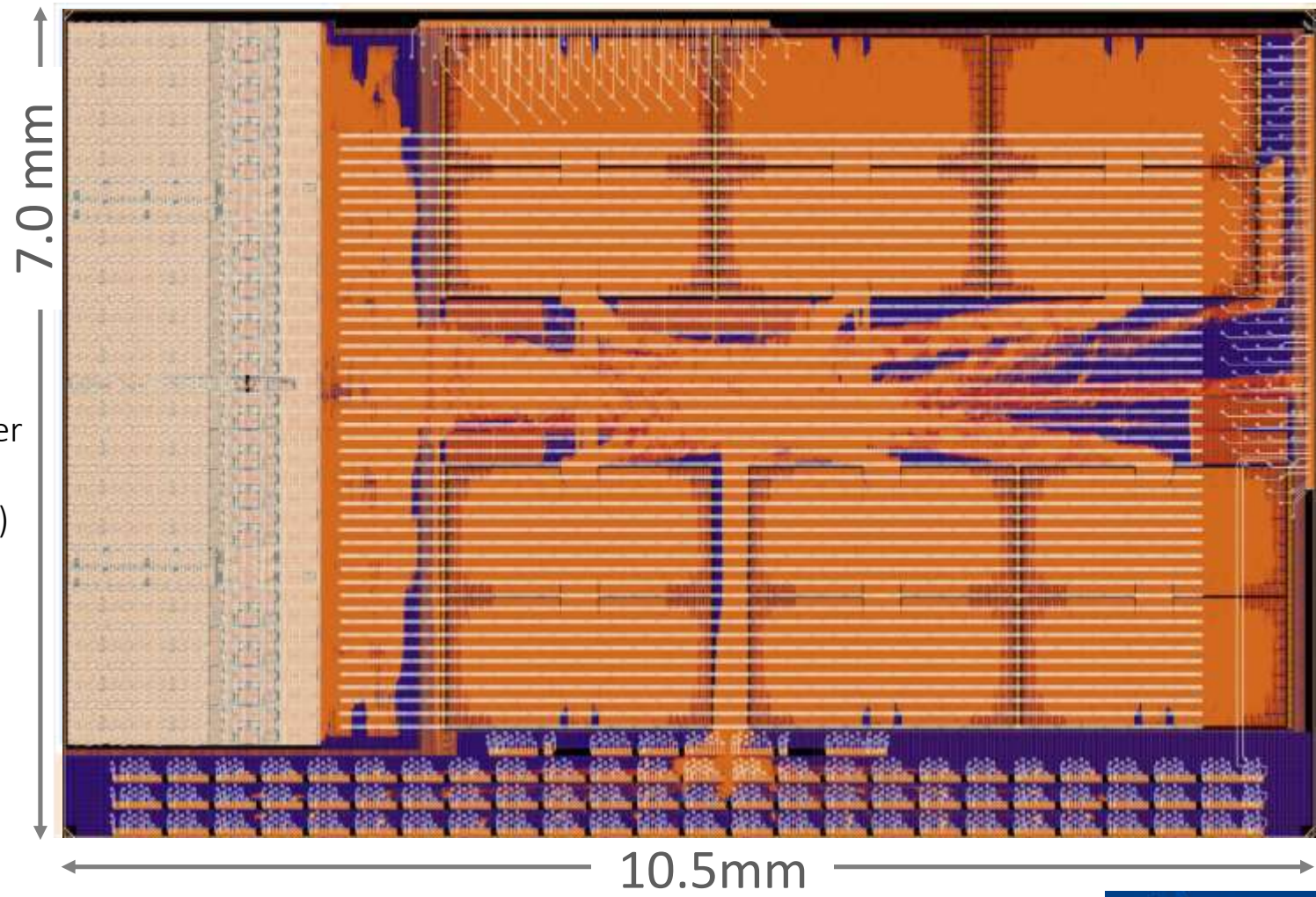


Occamy Chiplet, balancing bandwidth and compute

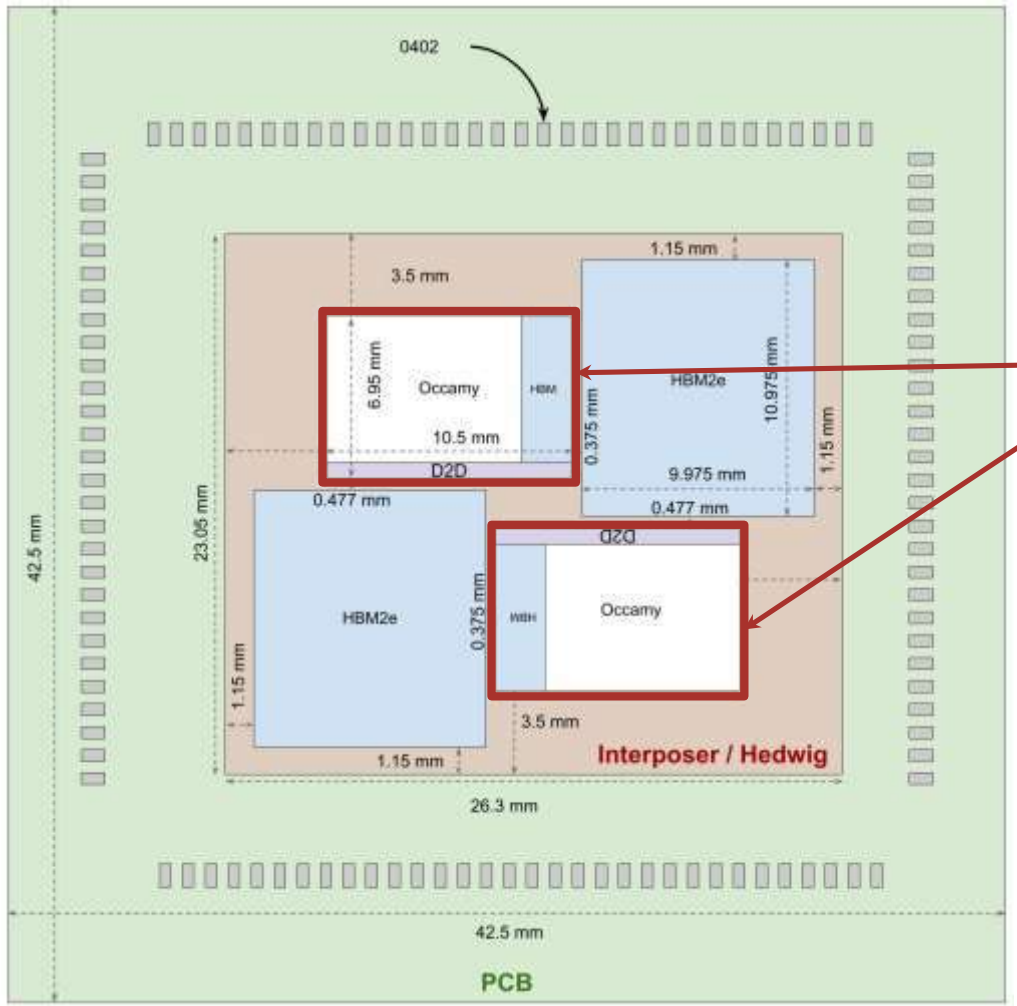


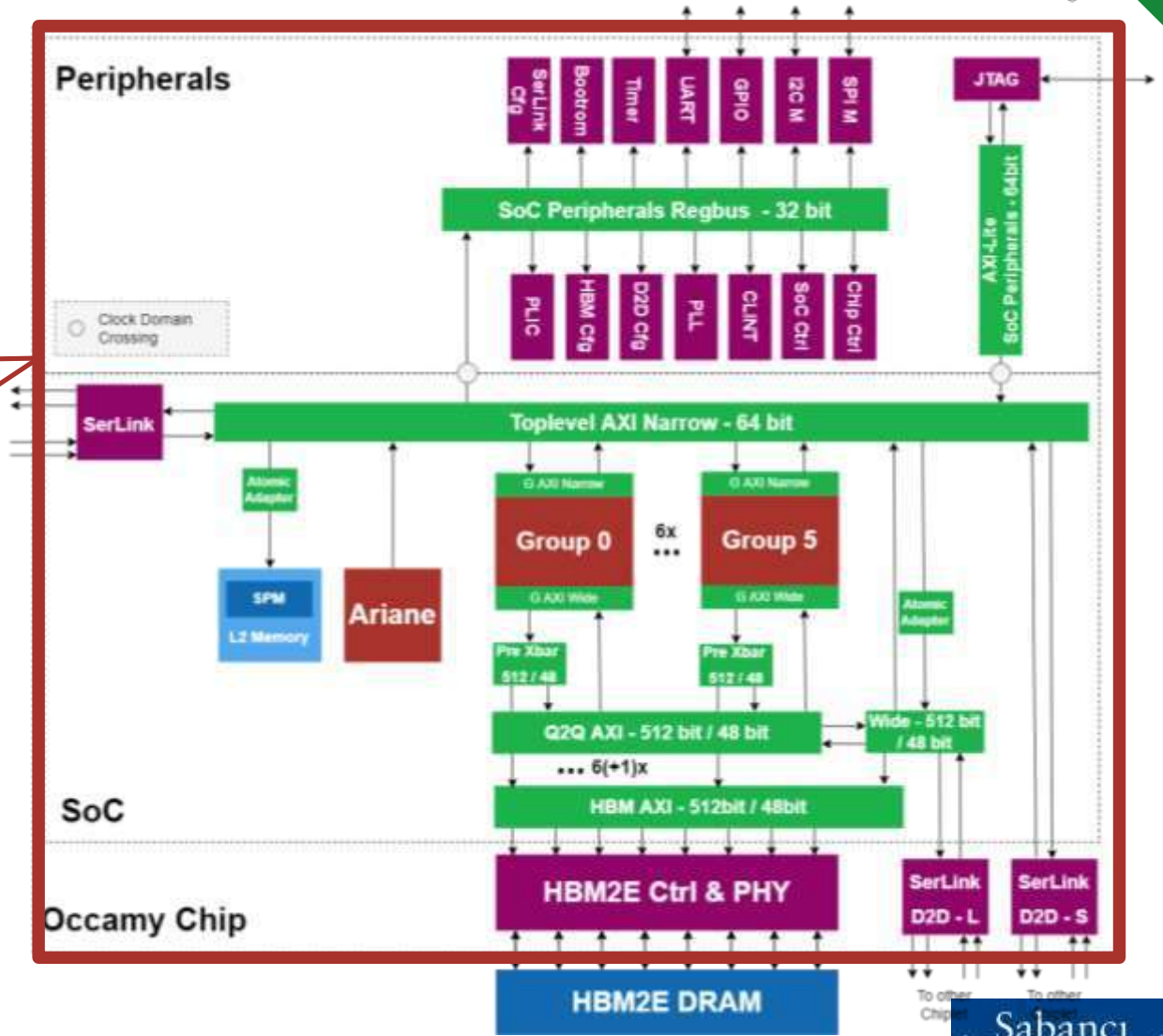
The heart: Occamy Chiplet: 384 GDFlop/s Engine

- GF12, target 1GHz (typ)
- 2 AXI NoCs (multi-hierarchy)
  - 64-bit for configuration/service
  - 512-bit with “interleaved” mode
- Peripherals
- Linux-capable manager core CVA6
- 6 Quadrants: 216 cores/chiplet
  - 4 cluster / quadrant:
    - 8 compute +1 DMA core / cluster
    - 1 multi-format FPU / core (**FP64,x2 32, x4 16/alt, x8 8/alt**)
- 8-channel HBM2e (8GB) **512GB/s**
- D2D link (Wide, Narrow) **70+2GB/s**
- System-level DMA
- SPM (2MB wide, 512KB narrow)



Occamy system with two compute dies and two HBM2e





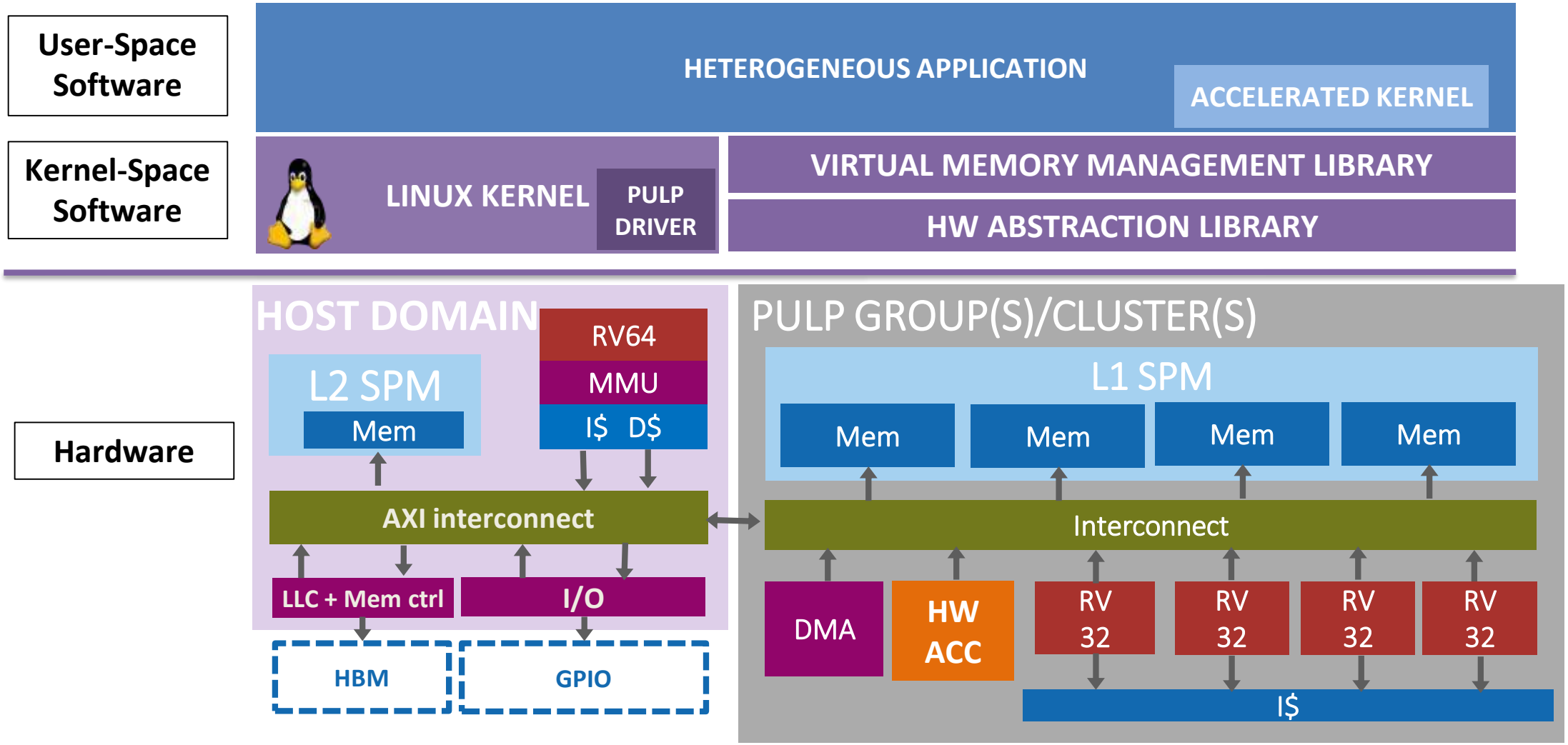
Programming Model

```
void main() {
  unsigned repetition = 2, bound = 4, stride = 8;
  static int data[8] = {1,2,3,4,5,6,7,8};

  __builtin_ssr_setup_1d(0, repetition, bound, stride, data);
  static volatile double d = 42.0;

  __builtin_ssr_enable();

  __builtin_ssr_push(0, d);
  volatile double e;
  e = __builtin_ssr_pop(0);
  __builtin_ssr_disable();
}
```

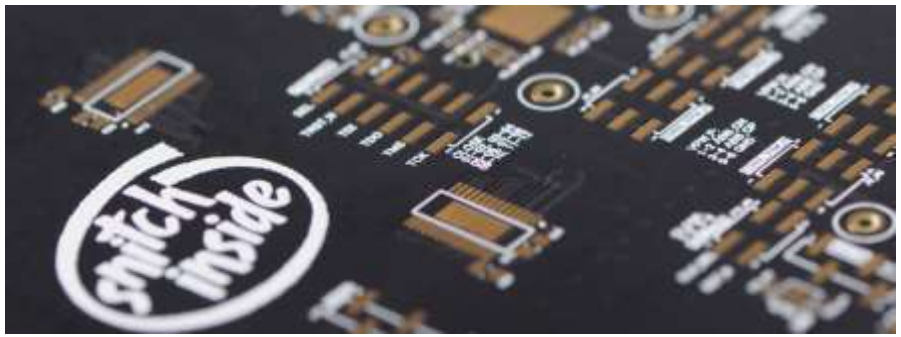
- Multiple layers of abstraction:
  - Hand-tuned assembly
  - LLVM intrinsics
  - FREP inference
  - High-level frameworks:
    DaCE: spcl.inf.ethz.ch/Research/DAPP/
    Pytorch+Dory: tiling of neural networks
- Bare-metal runtime
- Basic OpenMP runtime

OpenMP

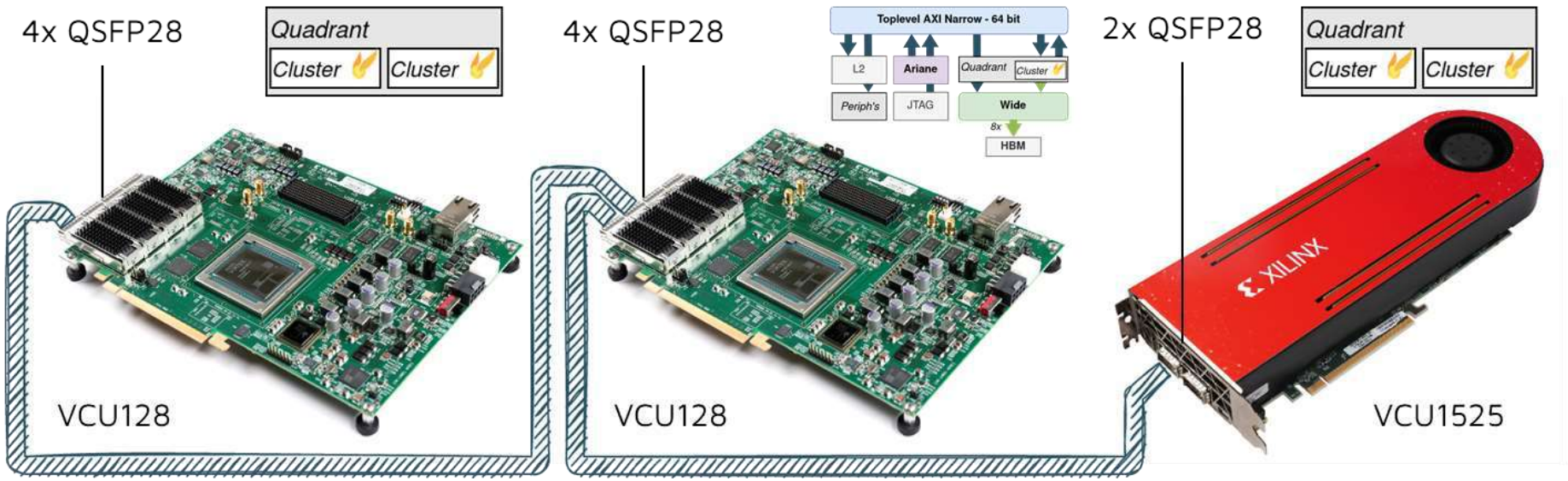
Software Stack



Prototyping and Emulation



- **“Quad-chiplet” prototype** board with FPGA interface
- Occamy mapped onto 2x VCU128 (with HBM) + 1x VCU1525
  - 1x CVA6
  - 2-4x 9-core Snitch cluster



The adventure is just starting..

- We work on energy efficient computing architectures
- Practically everything is permissively licensed and open source
  - We use the open source repositories for all our chips (more than 50)
- IoT applications (Edge computing at mW range)
  - PULPopen (Mr. Wolf, Vega, GAP8...) Cluster based many core architectures
- HPC applications
  - Snitch based heterogeneous accelerators (Occamy, mempool)
  - Vector processing based systems (Ara, Spatz)

Much more to come...

PULP

Parallel Ultra Low Power

Luca Benini, Ahmad Mirsalari, Alessandro Capotondi, Alessandro Nadalini, Alessandro Ottaviano, Alessio Burrello, Alfio Di Mauro, Andrea Borghesi, Andrea Cossettini, Angelo Garofalo, Arpan Prasad, Chi Zhang, Corrado Bonfanti, Cristian Cioflan, Cyril Koenig, Daniele Palossi, Davide Rossi, Fabio Montagna, Florian Glaser, Francesco Conti, Georg Rutishauser, Germain Haugou, Gianna Paulin, Giuseppe Tagliavini, Hanna Müller, Jannis Schoenleber, Lorenzo Lamberti, Luca Bertaccini, Luca Colagrande, Luca Valente, Maicol Caini, Manuel Eggimann, Manuele Rusci, Marco Bertuletti, Marco Guermandi, Matheus Cavalcante, Matteo Perotti, Mattia Sinigaglia, Michael Rogenmoser, Moritz Scherer, Moritz Schneider, Nazareno Bruschi, Nils Wistoff,, Paul Scheffler, Philipp Mayer, Robert Balas, Samuel Riedel, Segio Mazzola, Sergei Vostrikov, Simone Benatti, Thomas Benz, Thorir Ingolfsson, Tim Fischer, Victor Javier Kartsch Morinigo, Victor Jung, Viviane Potocnik, Vlad Niculescu, Xiaying Wang, Yichao Zhang, Yvan Tortorella, Frank K. Gürkaynak, all our past collaborators

and many more that we forgot to mention

http://pulp-platform.org @pulp_platform

Related publications and open-source repositories

Publications:

- Manticore @HotChips: https://ieeexplore.ieee.org/abstract/document/9296802
- SSRs: https://ieeexplore.ieee.org/abstract/document/9474230
- ISSRs: https://ieeexplore.ieee.org/abstract/document/9474230
- Snitch core & FREP: https://ieeexplore.ieee.org/abstract/document/9216552
- MiniFloat-NN: https://arxiv.org/pdf/2207.03192.pdf (accepted ARITH `22)
- SoftTiles: https://arxiv.org/pdf/2209.00889.pdf (accepted ISVLSI `22)

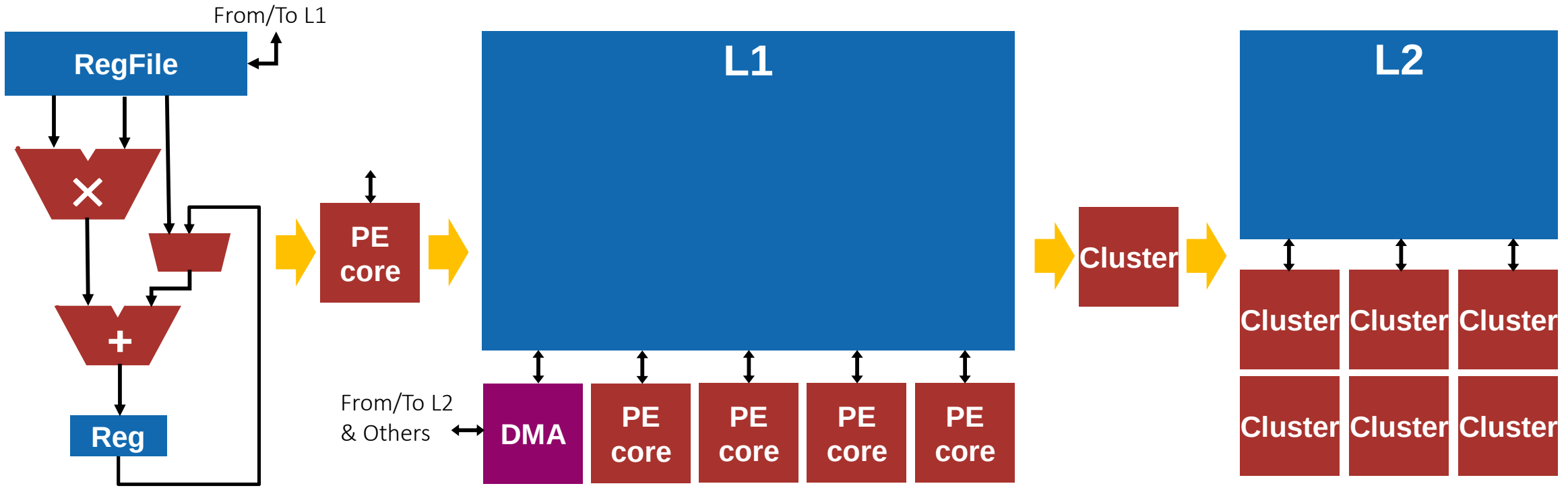
Repositories open-source:

- Snitch/Occamy: github.com/pulp-platform/snitch

See also our main web page:

- https://pulp-platform.org

Computing: Memory Everywhere



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

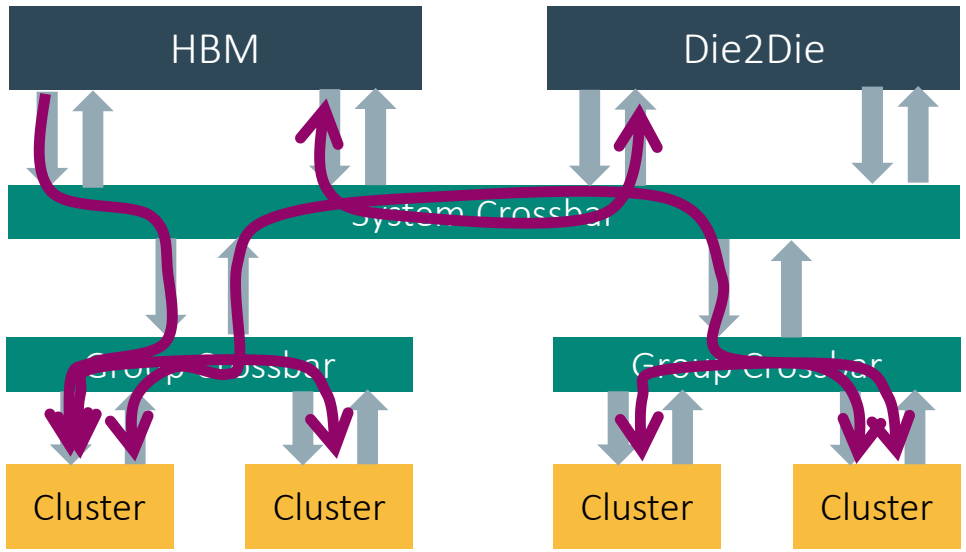
Additional Slides

Efficient Data Mover: hide L2, main memory latency

- 64-bit AXI DMA
- Tightly coupled with Snitch (<10 cycles configuration)
- Operates on wide 512-bit data-bus
- Hardware support to autonomously copy 2D shapes
- Higher-dimensionality can be 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);
```

Occamy 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**

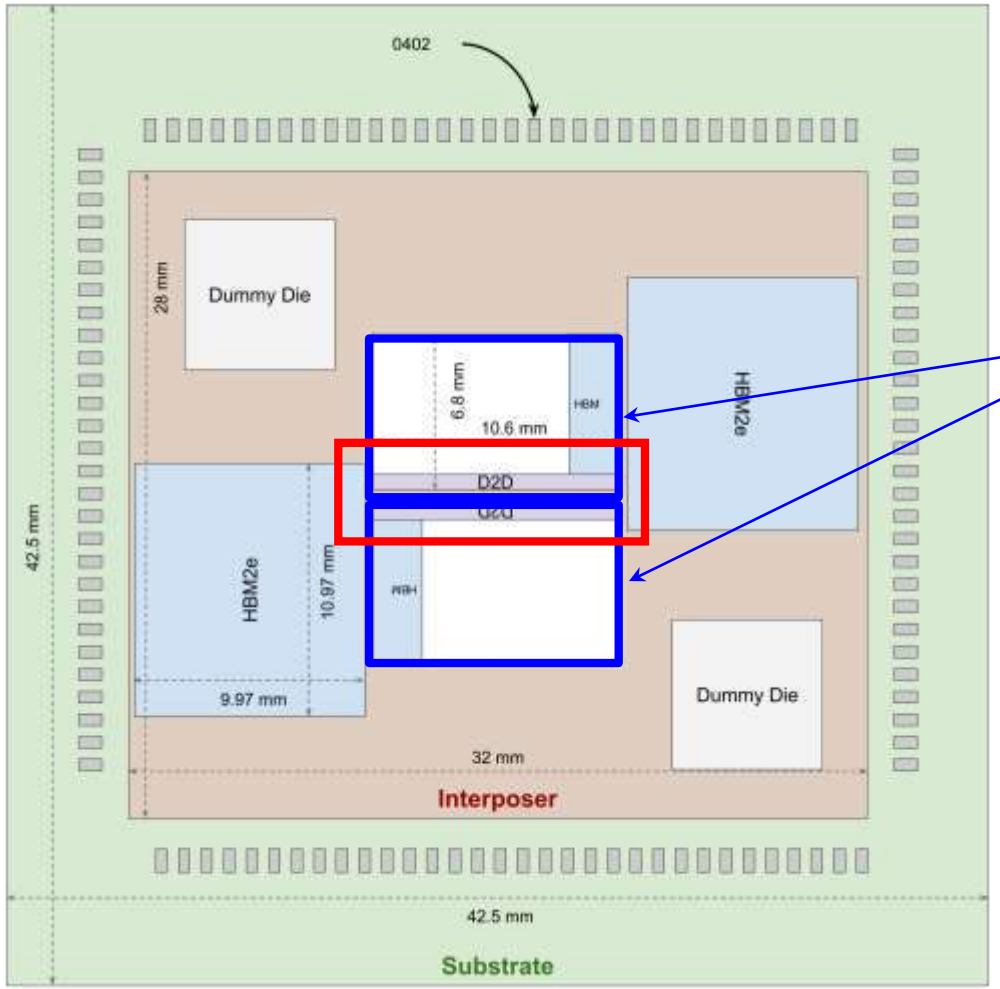
Smartly distribute workload

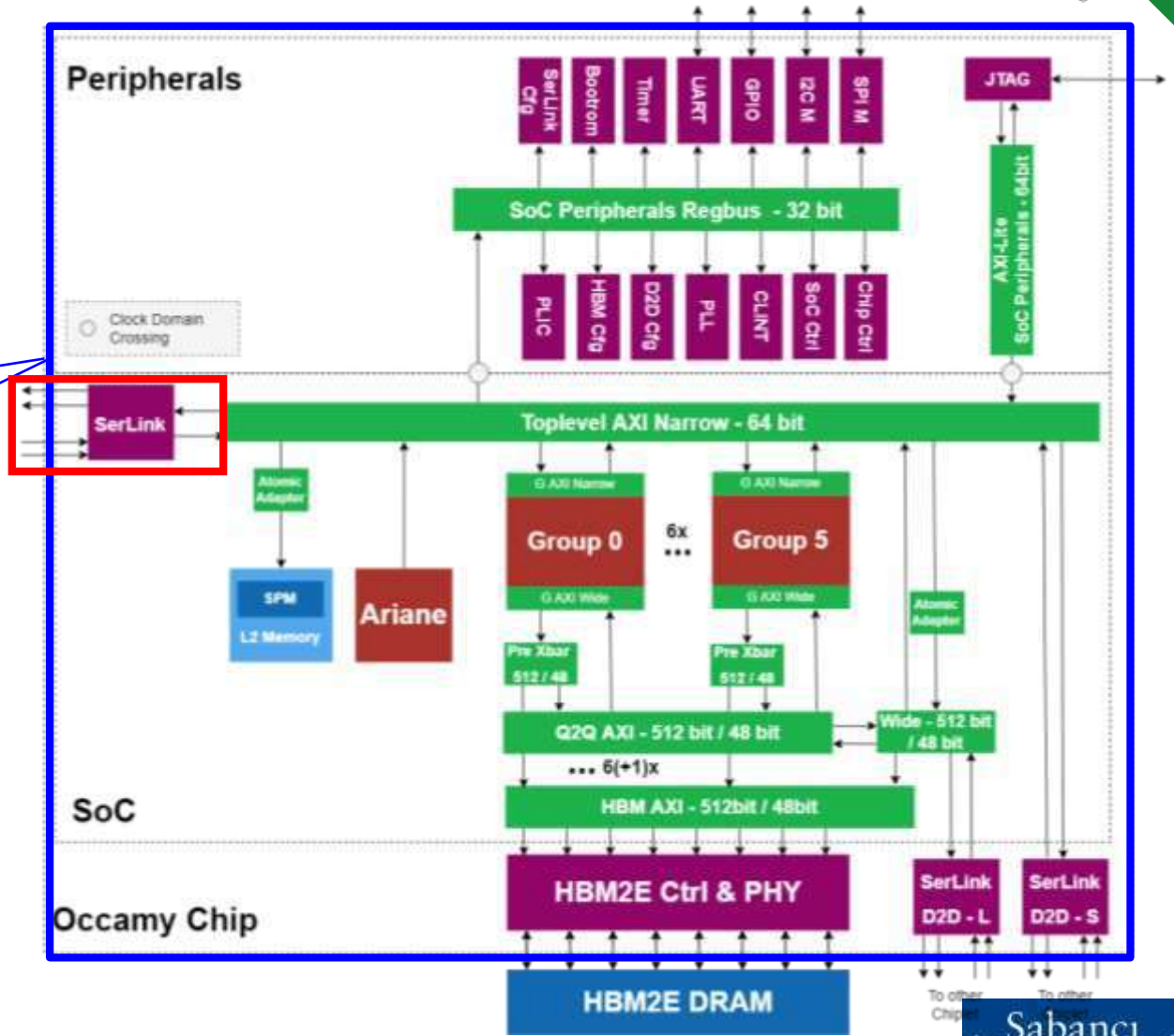
- **Clusters**: *DORY* framework for tiling strategy [1]
- **Chiplets**: E.g. Layer pipelining

Big trend today!

[1] Burrello, Alessio, et al. "Dory: Automatic end-to-end deployment of real-world dnns on low-cost iot mcus." IEEE Transactions on Computers 70.8 (2021): 1253-1268.

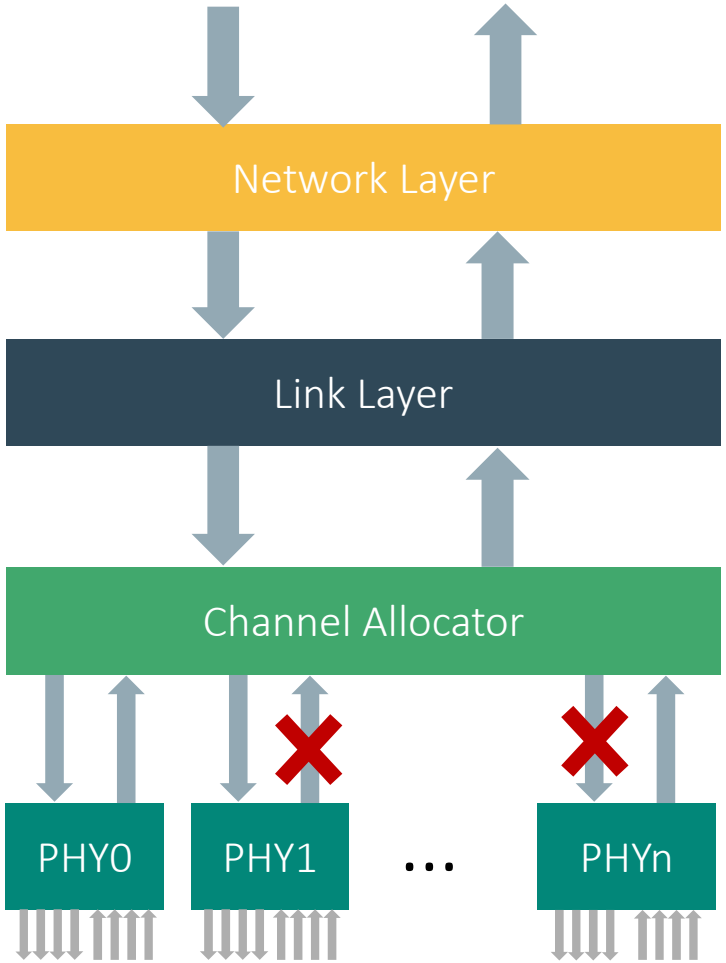
Occamy's C2C Link





C2C Link

Scalable and fault tolerant Chiplet2Chiplet Link



Network Layer

- Full AXI4 interface
- AXI4 to AXI stream converter

Data Link Layer

- Credit-based flow control
- RX synchronisation

Channel Allocator

- Chops and reshuffles payload
- Fault tolerance mechanisms

Physical layer (decoupled)

- Source-synchronous DDR sampling
- **Scalable** number of channels
  (38 x 8-bit full-duplex DDR channels in Occamy)
- Compatible with fast serial phy (e.g. HBI)