



PULP
Parallel Ultra Low Power

PULP: an Open Hardware Platform

The story so far

NIPS summer school 2018 - Perugia

19.07.2018



Frank K. Gürkaynak

Davide Rossi

¹*Department of Electrical, Electronic
and Information Engineering*

<http://pulp-platform.org>

PRECOMP
Open Transprecision Computing


Eurolab4HPC

ETH zürich

²*Integrated Systems Laboratory*

What will we cover today

- 14:30 - 16:00 / Part I / Frank K. Gürkaynak
What is PULP and what are its building blocks
 - Brief history of the project and its goals
 - Open source HW, why and why not
 - Components of a PULP based System
 - RISC-V cores of PULP project
 - PULP systems and implementations
- 16:00 - 16:30 / Coffee Break
- 16:30 - 18:00 / Part II / Davide Rossi
Secrets of PULP based systems
- Tomorrow / Hands on sessions with PULP systems



Eurolab4HPC



Parallel Ultra Low Power (PULP)

- Project started in **2013** by Luca Benini
- A collaboration between University of Bologna and ETH Zürich
- Key goal is

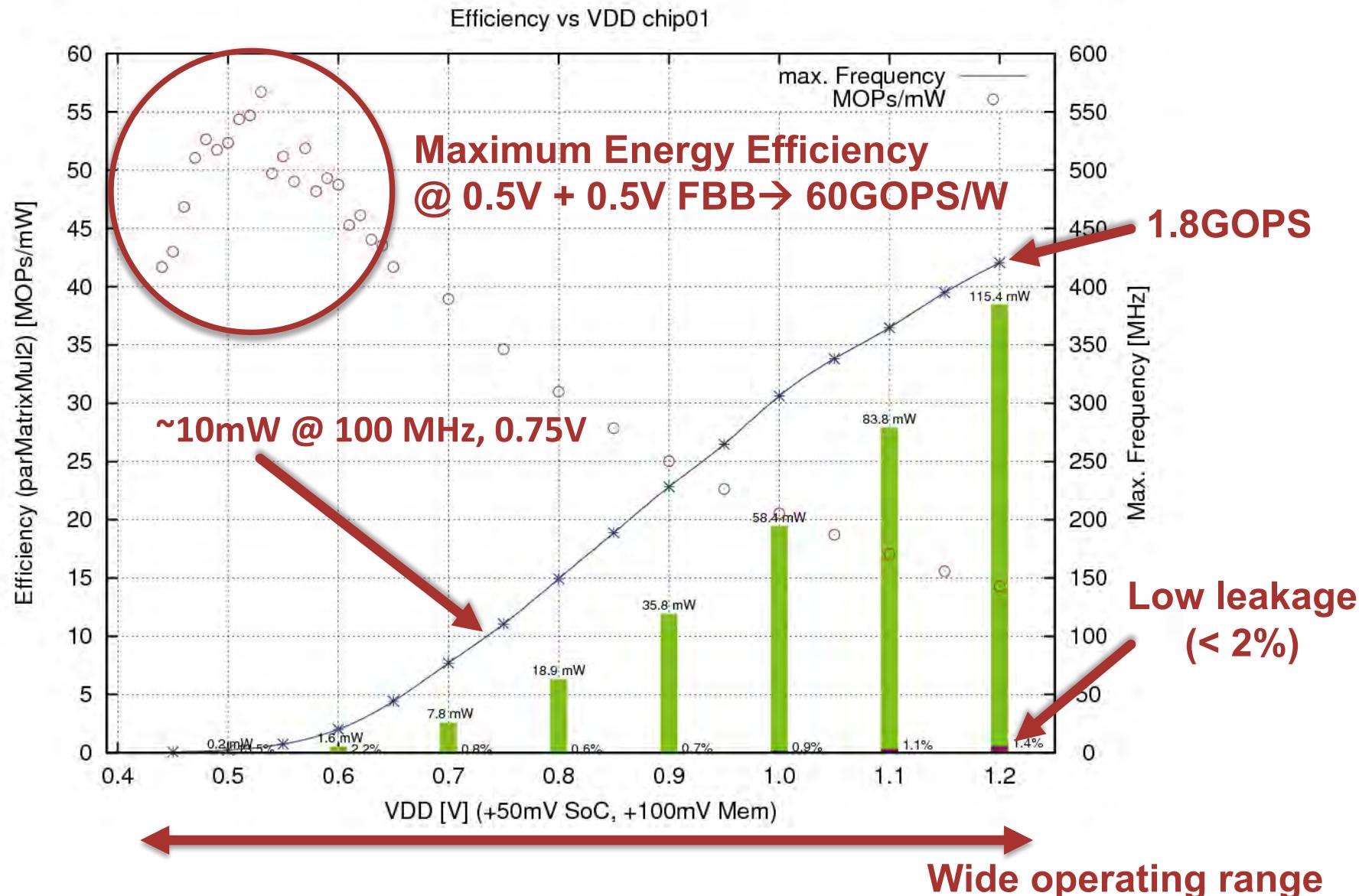
**How to get the most BANG
for the ENERGY consumed
in a computing system**

- We were able to start with a clean slate, no need to remain compatible to legacy systems.



Energy efficiency is the key driver in the PULP project

Measurement results from PULPv1 a four core cluster in ST FDSOI
28nm technology running a parallel matrix multiplications



To reach our goal we work on the following

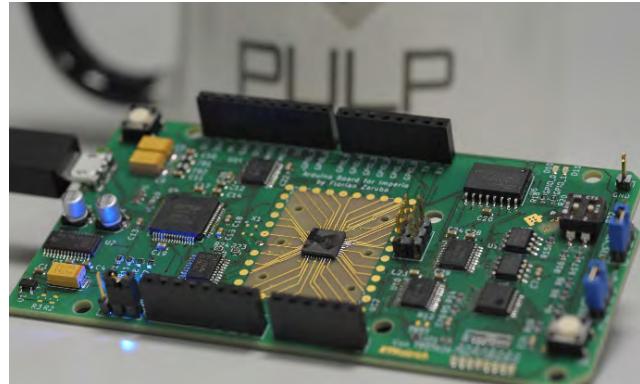
- **We concentrate on programmable systems**
 - Can not have custom hardware, need to be flexible
 - We need to make the system accessible to application developers
- **Scalable over a wide operating range**
 - Work just as well when processing 0.001 GOPS as 1000 GOPS
- **Don't waste** idle energy
 - Eliminate sources where cores and systems are idly wasting energy
- Make **good use** of options provided by the **technology**
 - Body biasing, power gating, library, memory selection etc..
- Take advantage of **heterogeneous acceleration**
 - Allow an architecture where accelerators can be added efficiently

PULP is maintained by a large group

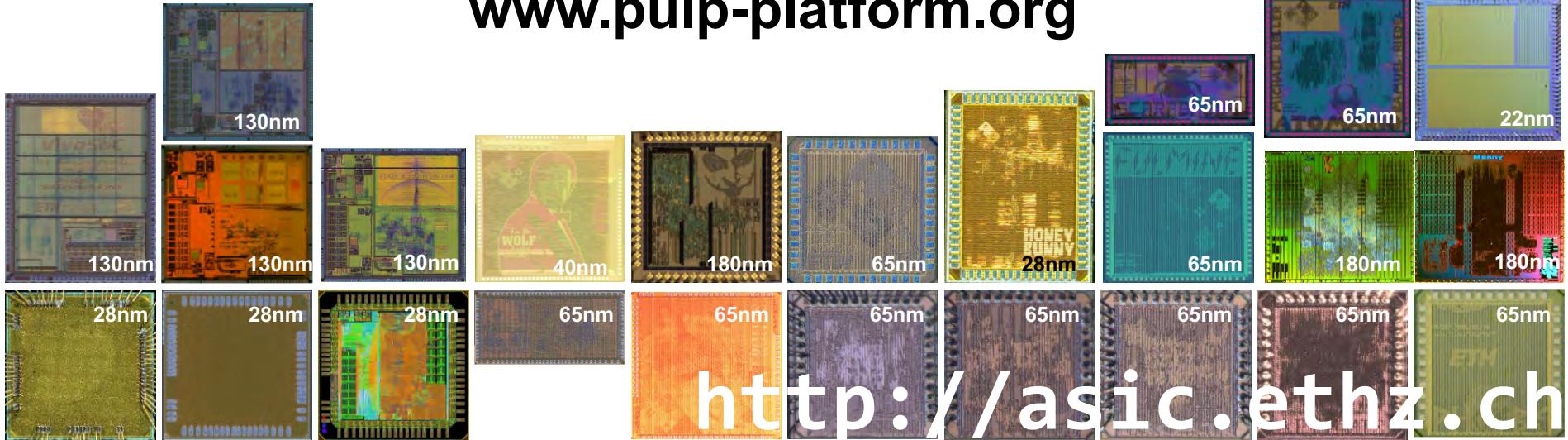
- Luca Benini holds a dual appointment in ETH Zürich and Bologna
- Total team about 50-60 members (60% in Zürich, 40% in Bologna)
 - 1 Professor
 - 3 Assistant Professors and 2 Senior Scientists
 - 8 Post Doctoral researchers
 - 30+ Ph.D. students
 - 6 Technical staff and 2 embedded staff in industrial partners
- Most of this team works on projects that are related to PULP
 - Core group of 15 designers that concentrate on PULP development
 - Many others contribute to application development, software support
- In ETH Zürich, the Microelectronics Design Center supports the project
 - Permanent staff of 4
 - Maintains design flows for ASIC and FPGA



We have designed over 20 PULP based ASICs already

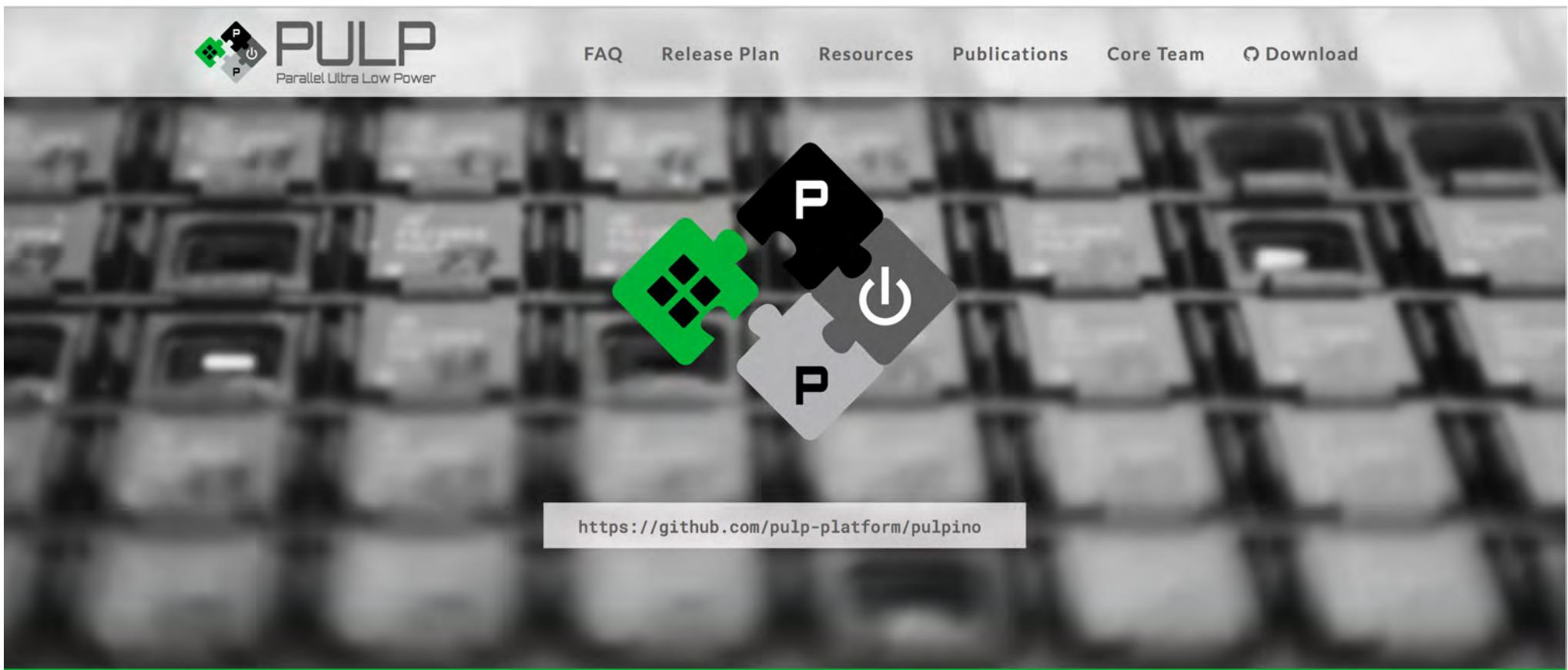


www.pulp-platform.org



<http://asic.ethz.ch>

We firmly believe in Open Source movement



- **First launched in February 2016 (github)**
 - HDL code, testbenches, testcases, SW, scripts, debug support.
 - We use solderpad (v0.51) license (Apache like license adapted for HW)

SOLDERPad
<http://www.solderpad.org/licenses/>



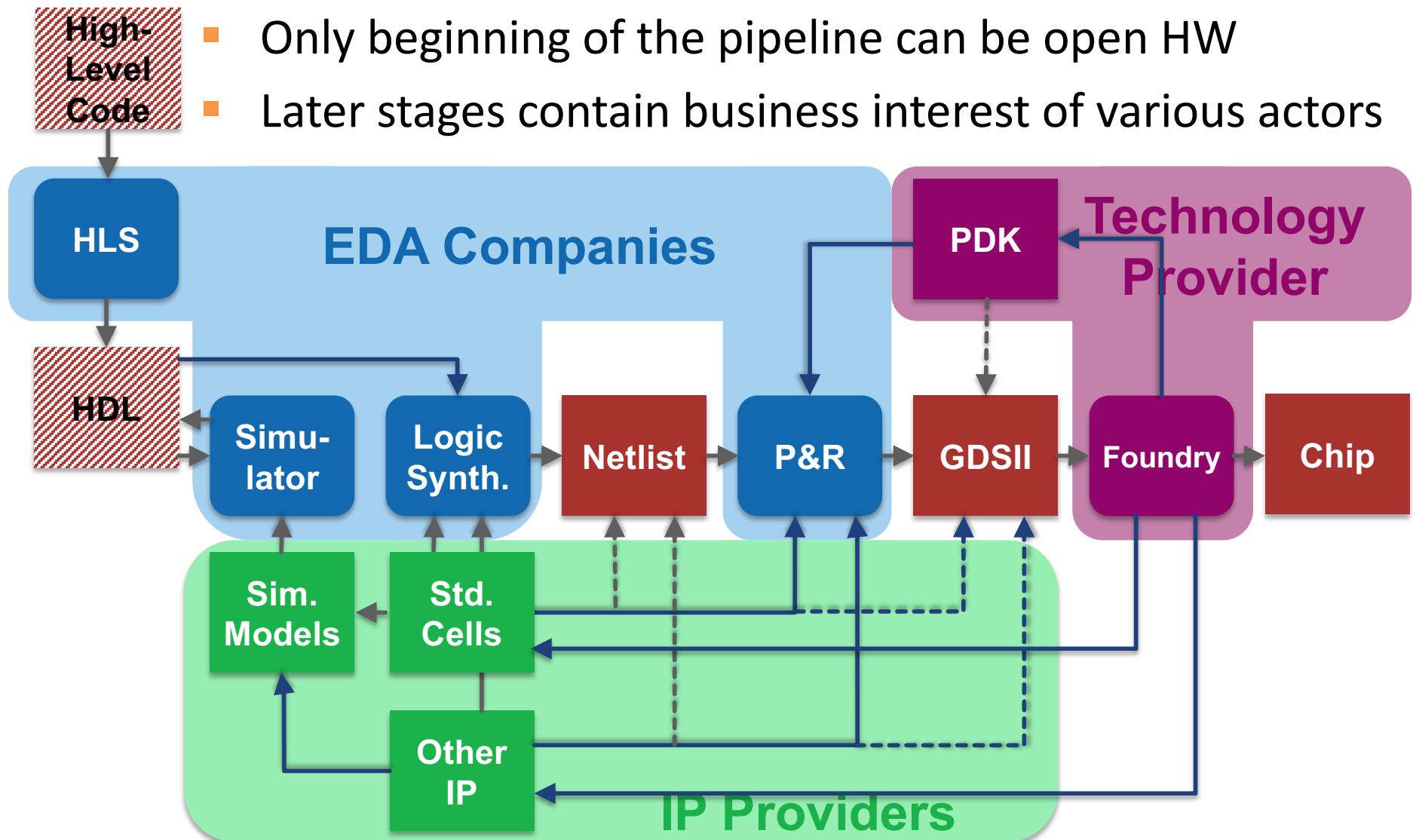
Open Hardware is a necessity, not an ideological crusade

- **The way we design ICs has changed, big part is now infrastructure**
 - Processors, peripherals, memory subsystems are now considered infrastructure
 - Very few (if any) groups design complete IC from scratch
 - High quality building blocks (IP) needed
- **We need an easy and fast way to collaborate with people**
 - Currently complicated agreements have to be made between all partners
 - In many cases, too difficult for academia and SMEs
- **Hardware is a critical for security, we need to ensure it is secure**
 - Being able to see what is really inside will improve security
 - Having a way to design open HW, will not prevent people from keeping secrets.

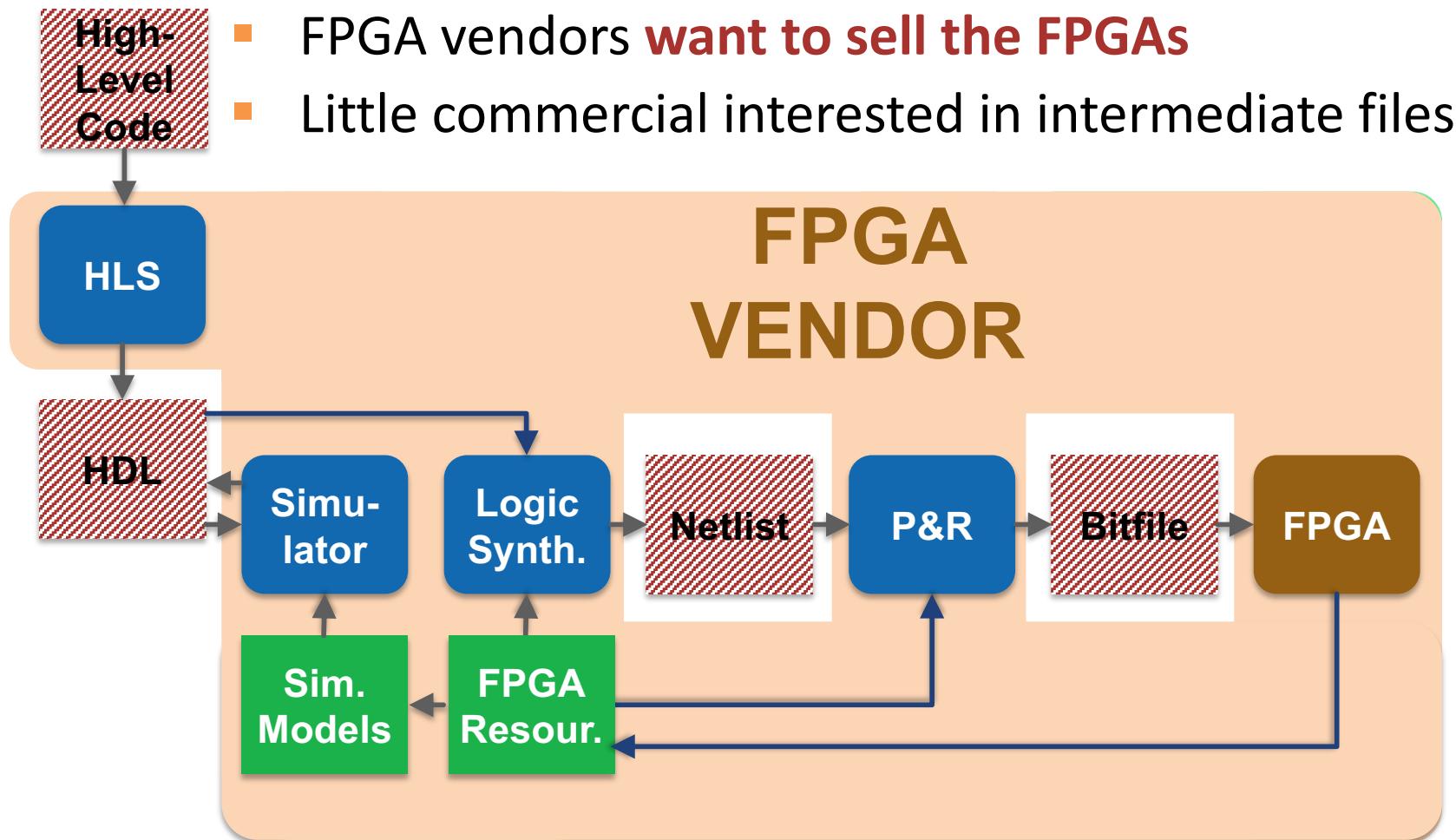
Hardware design flows for PCB/FPGA/ASIC are different

- **The differences complicate things**
 - Not a uniform way to discuss the issue, depends on the design flow
 - This talk about ASIC flow (the most complex one).
- **Different actors, different revenue streams**
 - Not every actor in the current design flow, earns money the same way
 - EDA companies are long complaining that they are out of the loop
Their income is not based on the amount of ICs produced.
 - **Not everybody will be happy** if open hardware will be more common
 - Important to understand the relationships
- **Interestingly most intermediate file formats are open, readable**
 - Verilog, Liberty, SPICE, EDIF, CIF, LEF, DEF, OA (open access)

ASIC Design Flow and Main Actors



FPGA Design Flow and Main Actors



- Most vendors allow bitfiles to be published -> will sell more FPGAs

We provide PULP with SOLDER Pad License

- Similar to Apache/BSD, adapted specifically for Hardware
- Allows you to:
 - Use
 - Modify
 - Make products and sell them

without restrictions.

SOLDER Pad

<http://www.solderpad.org/licenses/>

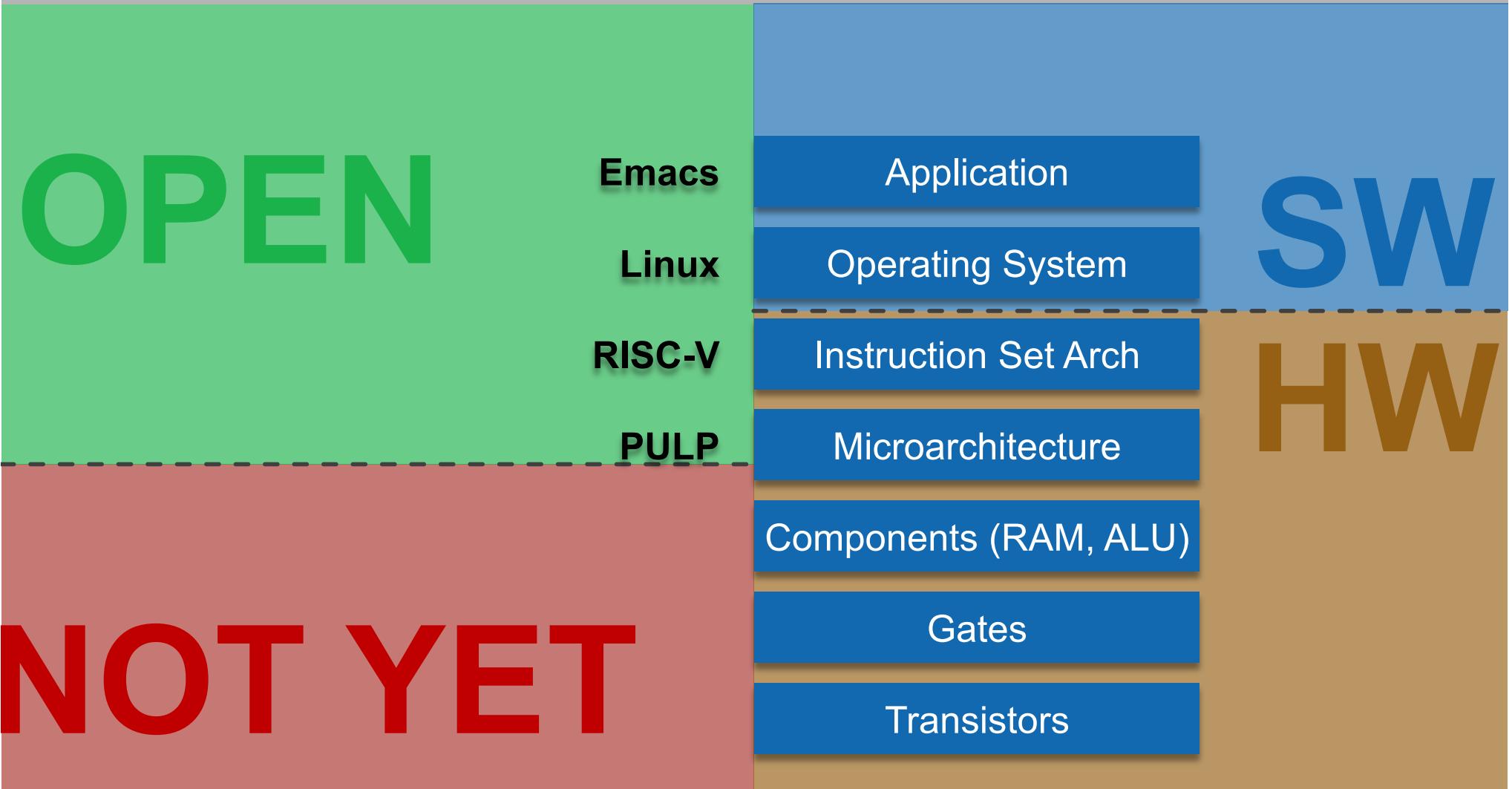
- Note the difference to **GPL**
 - Systems that include PULP do not have to be open source (Copyright not Copyleft)
 - They can be released commercially
 - LGPL may not work as you think for HW



At the moment, open HW can (mostly/only) be HDL code

- **The following are ok:**
 - RTL code written in HDL, or a high-level language for HLS flow
 - Testbenches in HDL and associated makefiles, golden models
- **How about support scripts for different tools?**
 - Synthesis scripts, tool startup files, configurations
- **And these are currently no go :**
 - Netlists mapped to standard cell libraries
 - Placement information (DEF)
 - Actual Physical Layout (GDSII)

Where are we now?



PULP Open-Source Releases and External Contributions

Releases

1

February 2016

First release of **PULPino**, our single-core microcontroller

2

May 2016

Toolchain and compiler for our RISC-V implementation (**RI5CY**), DSP extensions

3

August 2017

PULPino updates, new cores Zero-riscy and Micro-riscy, **FPU**, toolchain updates

4

February 2018

PULPissimo, ARIANE, PULP

5

March 2018

Open PULP



Community Contributions



1

June 2017

Porting of **Verilator** and **BEEBS**

benchmarks to PULPino

<https://github.com/embecosm/ri5cy>



2

September 2017

Porting of **ARM CMSIS** to PULPino

<https://github.com/misaleh/CMSIS-DSP-PULPino>



3

November 2017

Numerous **Bug fixes** to RI5CY in PULPino

<https://github.com/pulp-platform/riscv>



4

December 2017

STING: Open-Source Verification Environment for PULPino

<http://valtrix.in/programming/running-sting-on-pulpino>



We try to leverage open source as much as possible

Programming Model



Virtualization Layer



Compiler Infrastructure



Processor & Hardware IPs



Low-Power Silicon Technology



Silicon and Open Hardware fuel PULP success

Companies with announced products, business
Companies that use PULP internally or for training
Companies exploring opportunities

Companies that are using/evaluating PULP	
▪ GreenWaves Technologies	▪ SIAE Microelectronica
▪ Dolphin	▪ Advanced Circuit
▪ IQ Analog (14nm chips)	Pursuit
▪ Embecosm	▪ Shanghai Xidian
▪ lowRISC	Technology
▪ Mentor Graphics	▪ SCS Zurich
▪ Cadence Design Systems	▪ IMT technologies
▪ ST Microelectronics (IT,F)	▪ Microsemi
▪ Micron	▪ Arduino
▪ Google	▪ RacylCs
▪ IBM	
▪ NXP	

Research Centers/Universities using PULP	
▪ Stanford	▪ Zagreb FER
▪ Cambridge	▪ Universita di Genova
▪ UCLA	▪ Istanbul Technical U.
▪ CEA/LETI	▪ RWTH Aachen
▪ EPFL	▪ Lund
▪ National Chia Tung University	▪ USI – Lugano
▪ Politecnico di Milano	▪ Bar-Ilan
▪ Politecnico di Torino	▪ TU-Kaiserslautern
▪ Universita Roma I	▪ TU-Graz
▪ Instituto Superior Tecnico – U. de Lisboa	▪ UC San Diego
▪ Fondazione Bruno Kessler	▪ CSEM
	▪ IBM Research



The PULP family explained

RISC-V Cores

Peripherals

Interconnect

Platforms

Accelerators

We have developed several optimized RISC-V cores

RISC-V Cores

RI5CY

32b

**Micro
riscy**

32b

**Zero
riscy**

32b

Ariane

64b

How we started with open source processors

- Our research was not developing processors...
- ... but we needed good processors for systems we build for research
- **Initially (2013) our options were**
 - Build our own (support for SW and tools)
 - Use a commercial processor (licensing, collaboration issues)
 - Use what is openly available (OpenRISC,..)
- **We started with OpenRISC**
 - First chips until mid-2016 were all using OpenRISC cores
 - We spent time improving the microarchitecture
- **Moved to RISC-V later**
 - Larger community, more momentum
 - Transition was relatively simple (new decoder)



RISC-V Instruction Set Architecture

- Started by UC-Berkeley in 2010
- Open Standard governed by RISC-V foundation
 - ETHZ is a founding member of the foundation
 - Necessary for the continuity
 - Extensions are still being developed
- Defines 32, 64 and 128 bit ISA
 - No implementation, just the ISA
 - Different RISC-V implementations (both open and close source) are available
- At IIS we specialize in efficient implementations of RISC-V cores

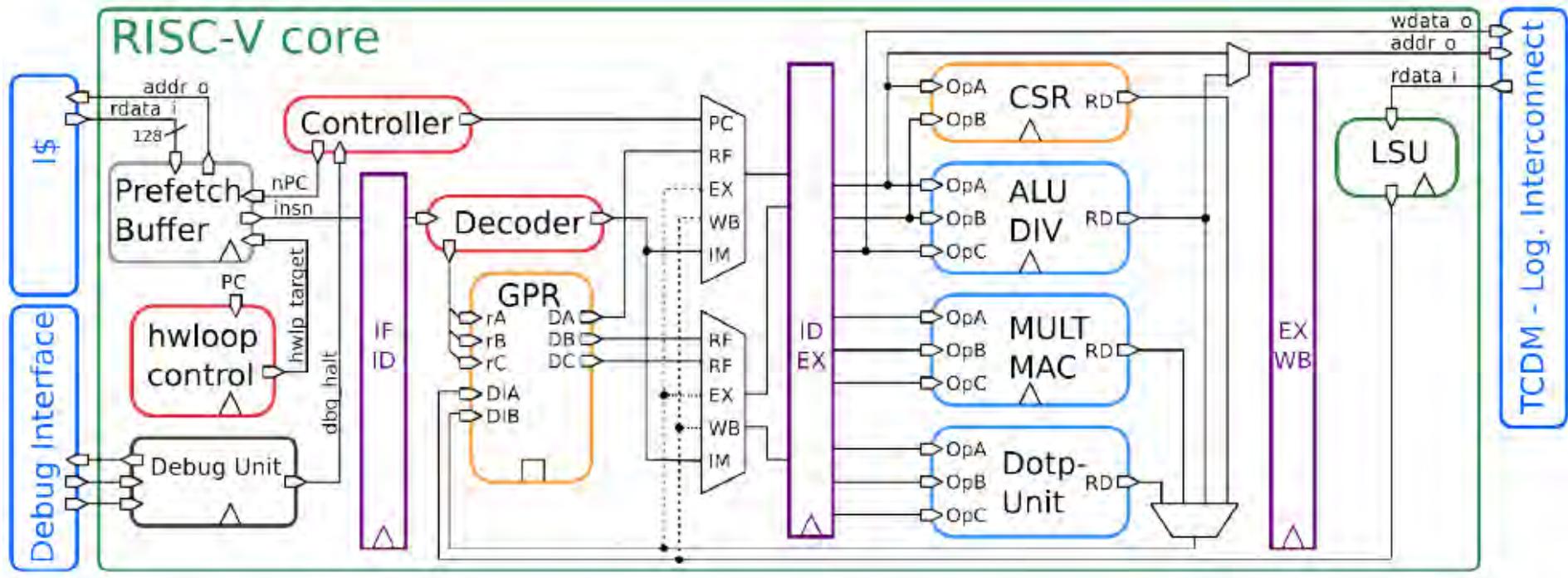
Spec separated into “extensions”

- I Integer instructions
- E Reduced number of registers
- M Multiplication and Division
- A Atomic instructions
- F Single-Precision Floating-Point
- D Double-Precision Floating-Point
- C Compressed Instructions
- X Non Standard Extensions

Our RISC-V cores

32 bit				64 bit
Low Cost Core	Core with DSP enhancements	Floating-point capable Core	Linux capable Core	
<ul style="list-style-type: none">■ Zero-riscy■ RV32-ICM■ Micro-riscy■ RV32-CE	<ul style="list-style-type: none">■ RI5CY■ RV32-ICMX<ul style="list-style-type: none">■ SIMD■ HW loops■ Bit manipulation■ Fixed point	<ul style="list-style-type: none">■ RI5CY+FPU■ RV32-ICMFX	<ul style="list-style-type: none">■ Ariane■ RV64-ICM	
ARM Cortex-M0+	ARM Cortex-M4	ARM Cortex-M4F	ARM Cortex-A55	

RI5CY – Our workhorse 32-bit core



- 4-stage pipeline, optimized for energy efficiency
- 40 kGE, 30 logic levels, Coremark/MHZ 3.19
- Includes various extensions (X) to RISC-V for DSP applications

RI5CY – ISA Extensions improve performance

```
for (i = 0; i < 100; i++)  
    d[i] = a[i] + b[i];
```

Baseline

```
mv    x5, 0  
mv    x4, 100  
Lstart:  
    lb    x2, 0()  
    lb    x3, 0()  
    addi x10, x1  
    addi x11, x1  
    add   x2, x3  
    sb    x2, 0()  
    addi x4, x4  
    addi x12, x1  
    bne   x4, x5
```

Auto-incr load/store

```
mv    x5, 0  
mv    x4, 100  
Lstart:  
    lb    x2, 0()  
    lb    x3, 0()  
    addi x10, x1  
    addi x11, x1  
    add   x2, x3  
    sb    x2, 0()  
    addi x4, x4  
    addi x12, x1  
    bne   x4, x5, Lstart
```

HW Loop

```
lp.setupi 100, Lend  
    lb    x2, 0(x10!)  
    lb    x3, 0(x11!)  
    addi x4, x4  
    add   x2, x3, x2  
    sb    x2, 0(x12!)  
Lend:    sb    x2, 0(x12!)
```

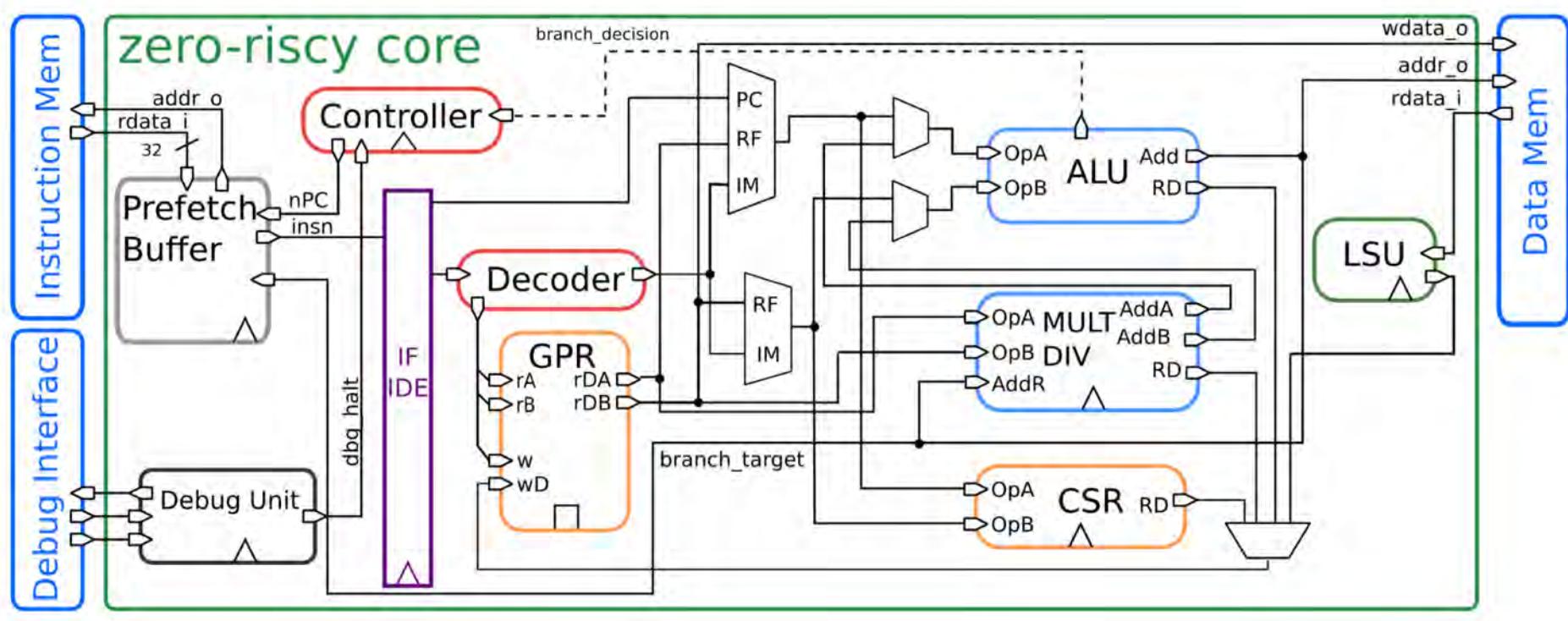
Packed-SIMD

```
lp.setupi 25, Lend  
    lw    x2, 0(x10!)  
    lw    x3, 0(x11!)  
    pv.add.b x2, x3, x2  
Lend:    sw    x2, 0(x12!)
```

Why we designed newer 32-bit cores

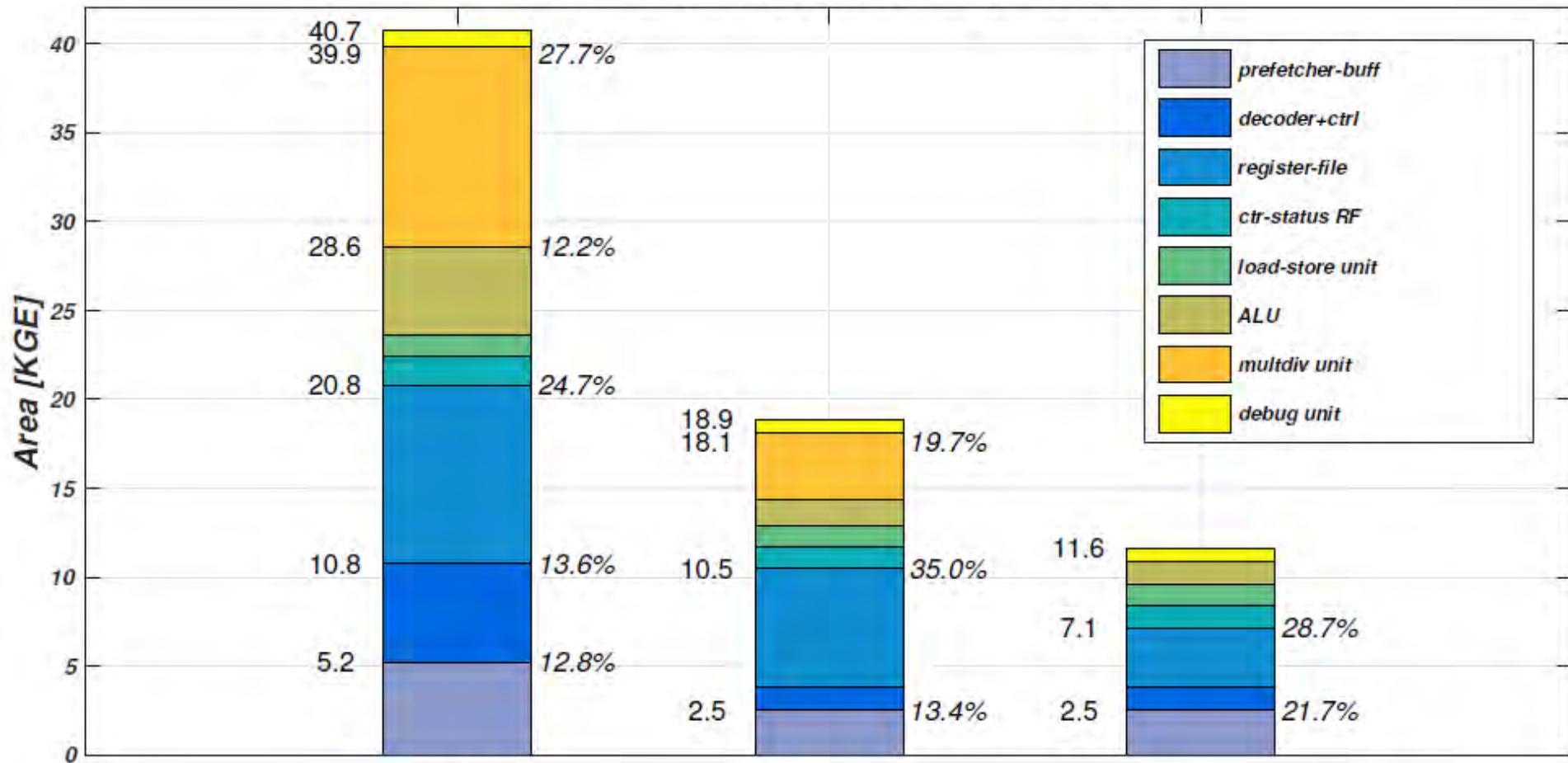
- RI5CY was built for **energy efficiency** for DSP applications
 - Ideally all parts of the core are running all the time doing something useful
 - This does not always mean it is low-power
 - The core is rather large (> 40 kGE without FPU)
- People asked us about a **simple and small core**
 - Not all processor cores are used for DSP applications
 - The DSP extensions are mostly idle for control applications
 - **Zero-Riscy** was designed to as a **simple and efficient** core.
- Some people wanted the **smallest possible RISC-V core**
 - It is possible to further reduce area by using 16 registers instead of 32 (E)
 - Also the multiplier can be removed saving a bit more
 - **Micro-Riscy** is a parametrized variation of Zero-Riscy with **minimal area**

Zero/Micro-riscy, small area core for control applications



- Only 2-stage pipeline, simplified register file
- **Zero-Riscy** (RV32-ICM), 19kGE, 2.44 Coremark/MHz
- **Micro-Riscy** (RV32-EC), 12kGE, 0.91 Coremark/MHz
- Used as SoC level controller in newer PULP systems

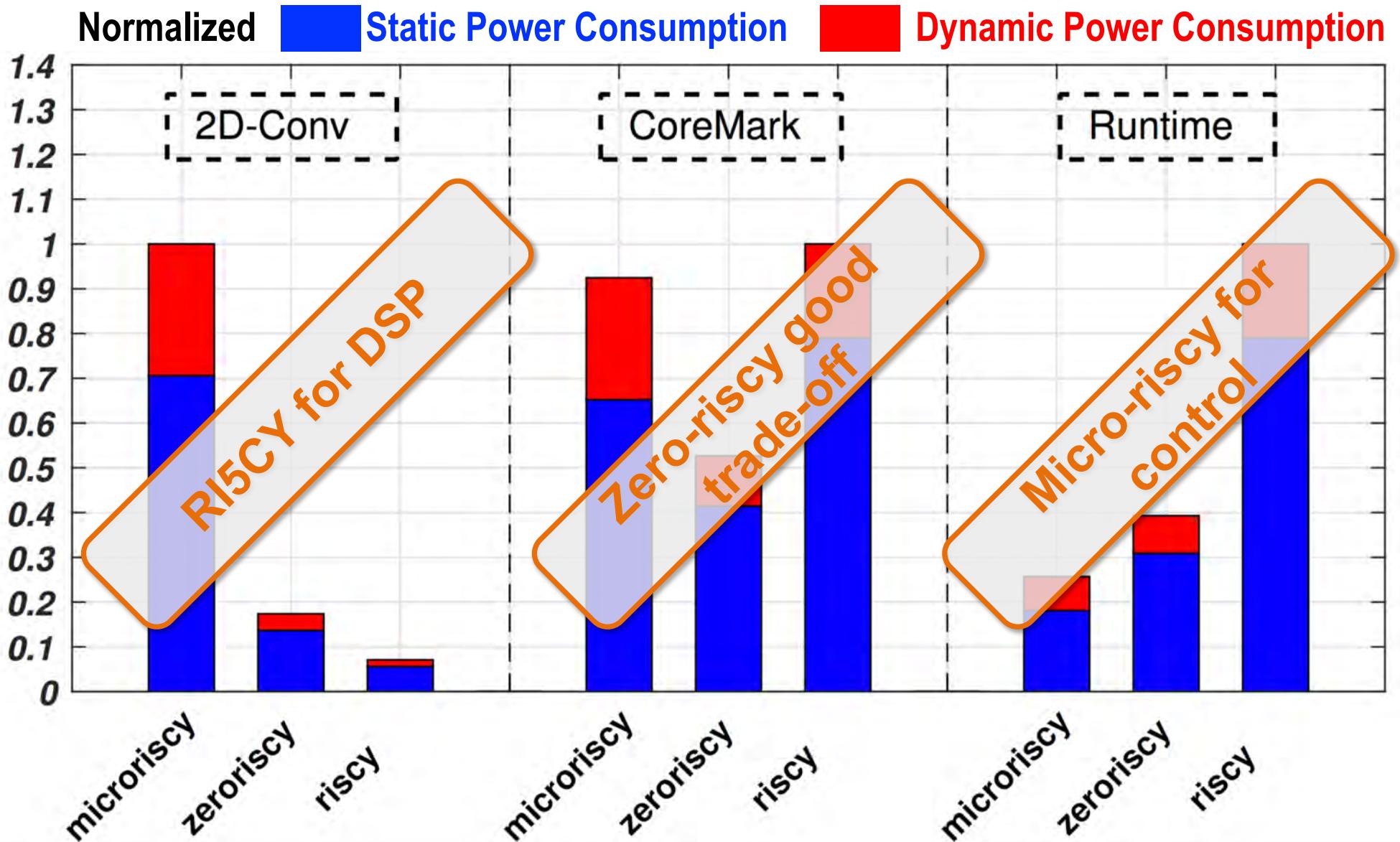
Different 32-bit cores with different area requirements



RI5CY

Zero-riscy Micro-riscy

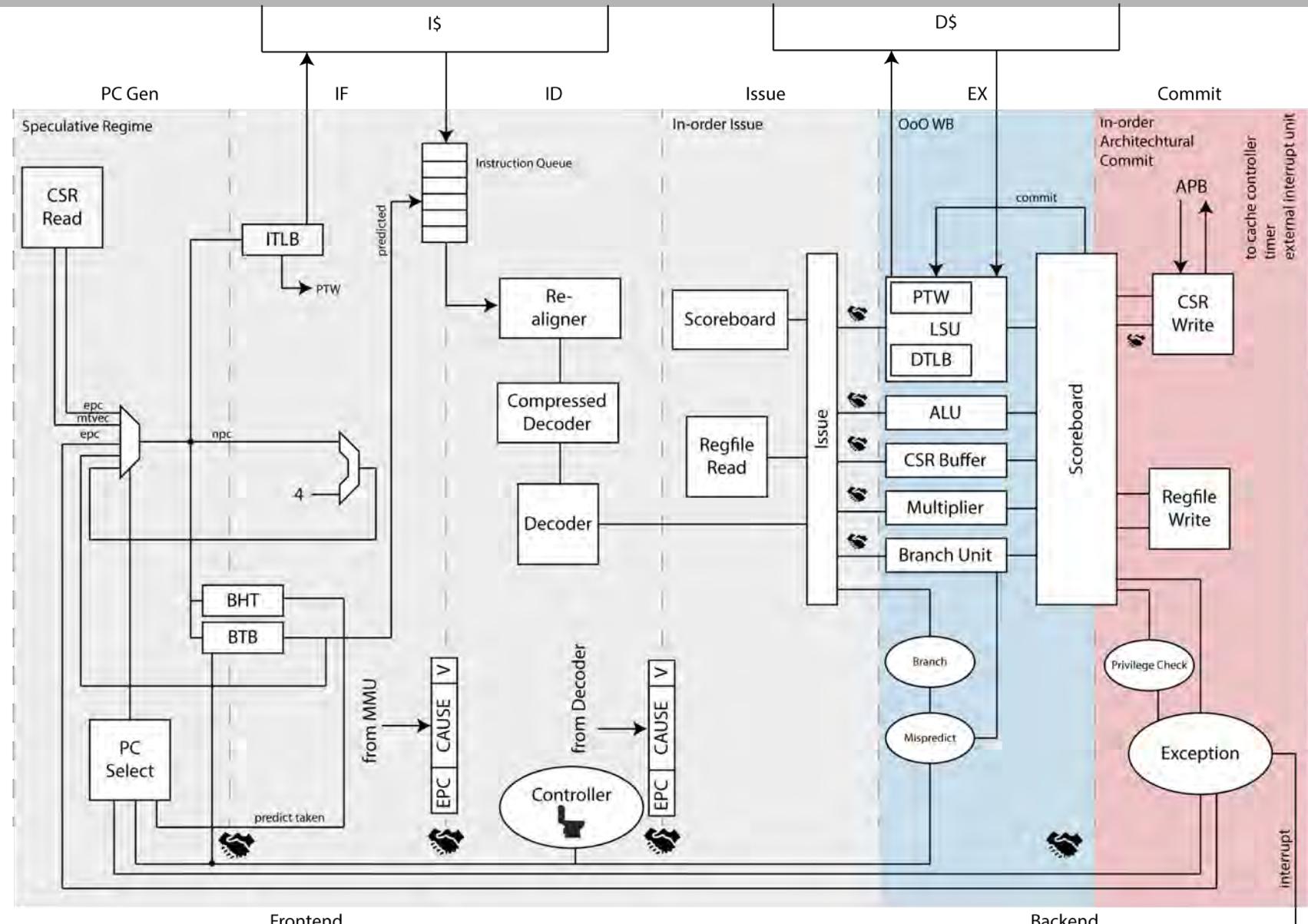
Different cores for different types of workload



Finally the step into 64-bit cores

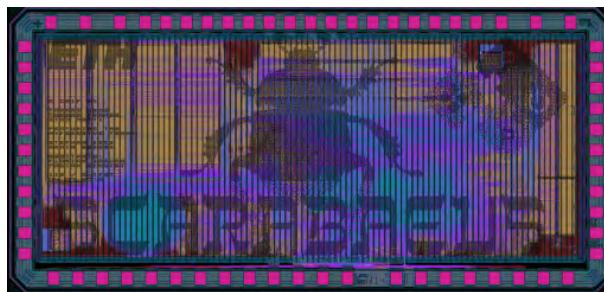
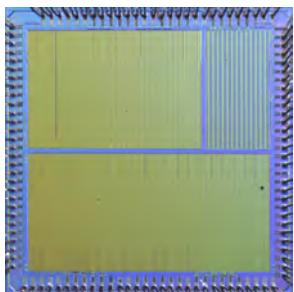
- For the first 4 years of the PULP project we used only 32bit cores
 - Luca once famously said “*We will never build a 64bit core*”.
 - Most IoT applications work well with 32bit cores.
 - A typical 64bit core is much more than 2x the size of a 32bit core.
- **But times change:**
 - Using a 64bit Linux capable core allows you to share the same address space as main stream processors.
 - We are involved in several projects where we (are planning to) use this capability
 - There is a lot of interest in the security community for working on a contemporary open source 64bit core.
 - Open research questions on how to build systems with multiple cores.

ARIANE: Our Linux Capable 64-bit core

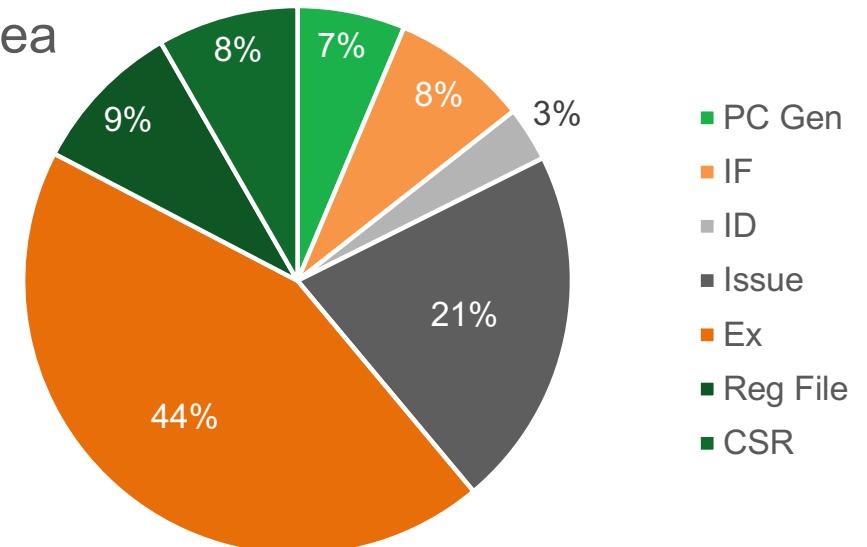


Main properties of Ariane

- Tuned for high frequency, 6 stage pipeline, integrated cache
 - In order issue, out-of-order write-back, in-order-commit
 - Supports privilege spec 1.11, M, S and U modes
 - Hardware Page Table Walker
- Implemented in GF22nm (Poseidon), and UMC65 (Scarabaeus)
 - In 22nm: 910 MHz worst case conditions (SSG, 125/-40C, 0.72V)
 - 8-way 32kByte Data cache and 4-way 32kByte Instruction Cache
 - Core area: 175 kGE

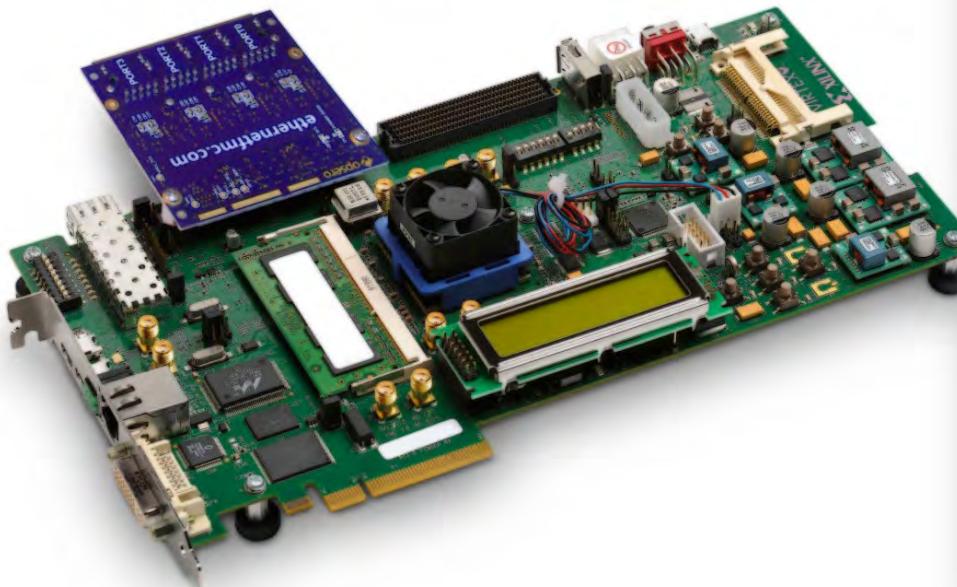


Area



Ariane mapped to FPGA boots Linux

- Full FPGA implementation
 - Xilinx Vertex 7 – VC707
 - **Core:** 50 – 100 MHz
 - **Core:** 15 kLUTs
 - 1 GB DDR3
 - Mapping to simpler boards underway



.shh+
.yMMMMMMm+
.yMMMMMMMN+`
.yMMMMMdooooosmMMMN+`
-yMMMMMMMy +sss. -MMMMMN+`
-hMMMMMMMy /ooo. -MMMMMMMNNo`
-hMMMMMMMMMy osssymMMMMMMMMMNNo`
NMNNNNNNNNNNMy. mMMMMMMMMMMMMMMMs.
-osss/. -dMMMMMMMMMMMMMMMMMMMMMMMMMMMyys:
-osssssss+. `/ss+- :dMMN+:+dMMMMMMMMMMMMMMdyyyyyyys:
:osssssssssssss+ssssso :hy` : NMNNNNNNNNNNNNNNNyyyyyyyyyys:
:ossssssdMDssssssssss/. sMMNNNNNNNNNNNNNNNyhhyyyyyyyyys:
:osssssssyMMMMNyssssssss+. odMMMMMMMMMMMMMMdMMMMMyyy:-:yyyyyyyyys:
:ossssssyddyymMMyynds+ssssss+. -hMMMMMMMMddMMMMMyyy:-:yyyyyyyyys:
.ossssssydMMMDyyymMMMdssssssss+ -hMMMMMMMMddMMMMMyyy:-:yyyyyyyyys:
.ossssssydMMMDyyymMMMdssssssss+ -hMMMMMMMMddMMMMMyyy:-:yyyyyyyyys:
:ossssssyddyymMMyynds+ssssss+. `` `:-:::/syyyyyyy` .yyy//yys` .yyyyys:
:ossssssydMMMMNyyo` '+st.:-`-:::/syyyyyo` .+sys+` .syyyyys:
:ossssssdMDso` ` ` ..:::/oyyyyyy+-....:+yyyyys:
:osssssssssss/. ` ` ..:::/oyyyyyy+-....:+yyyyys:
-osssssssssss: ` ` ..:::/oyyyyyy+-....:+yyyyys:
-osssssss+. ` ` ..:::/oyyyyyy+-....:+yyyyys:
-osss/. ` ` ..:::/oyyyyyy+-....:+yyyyys:
.}` ` ..:::/hhhhh+-....:+/..:
`-:::..:sMyssmm:.....:
`-:::..:sMdhhhNm:.....:
`-:::..:sMsoooo:.....:
`-:::..:od/.....:
`-:::..:.....:
`-:::..:.....:
`-:::..:.....:

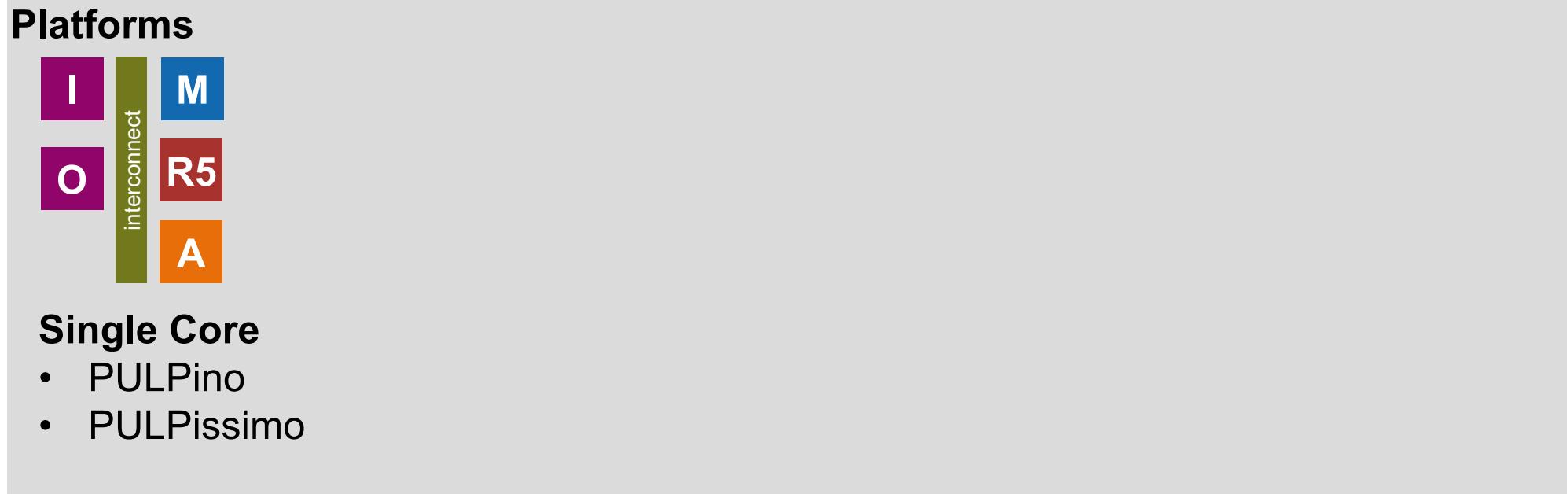
Only processing cores are not enough, we need more

RISC-V Cores				Peripherals		Interconnect
RI5CY 32b	Micro riscy 32b	Zero riscy 32b	Ariane 64b	JTAG	SPI	Logarithmic interconnect
				UART	I2S	APB – Peripheral Bus
				DMA	GPIO	AXI4 – Interconnect

Accelerators	HWCE (convolution)	Neurostream (ML)	HWCrypt (crypto)	PULPO (1 st order opt)

The pulp-platforms put everything together

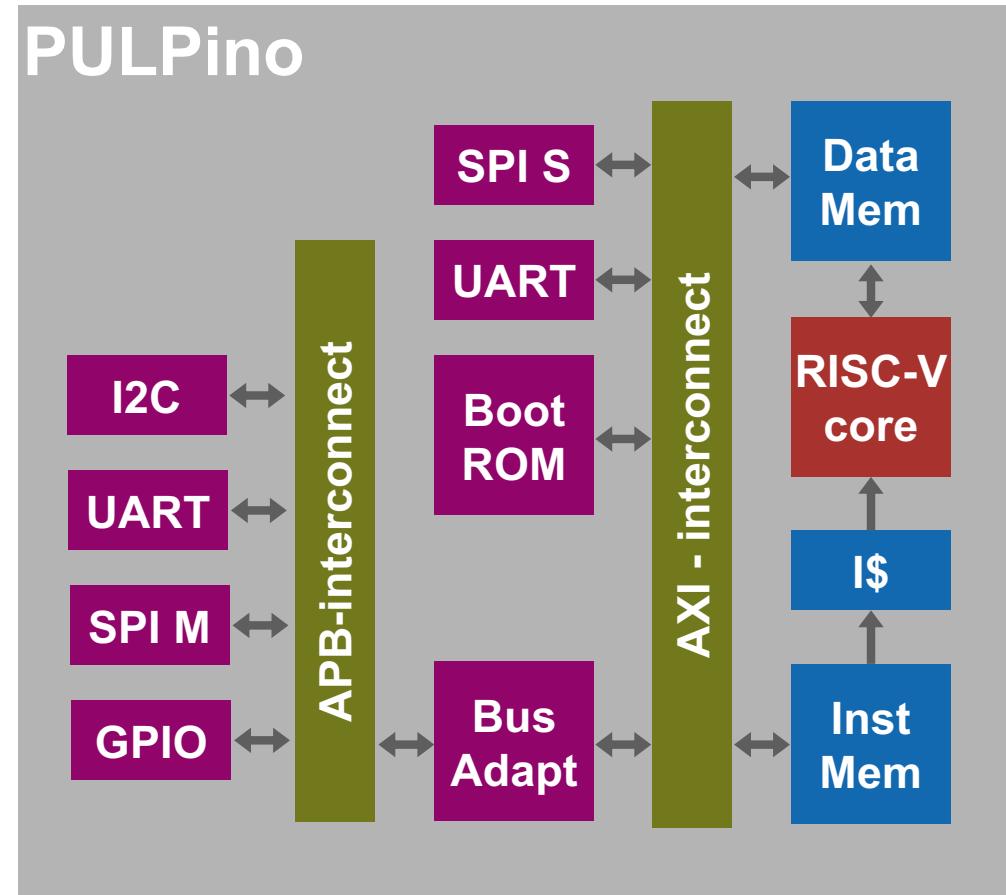
RISC-V Cores				Peripherals		Interconnect
RI5CY 32b	Micro riscy 32b	Zero riscy 32b	Ariane 64b	JTAG	SPI	Logarithmic interconnect
				UART	I2S	APB – Peripheral Bus
				DMA	GPIO	AXI4 – Interconnect



Accelerators	HWCE (convolution)	Neurostream (ML)	HWCrypt (crypto)	PULPO (1 st order opt)

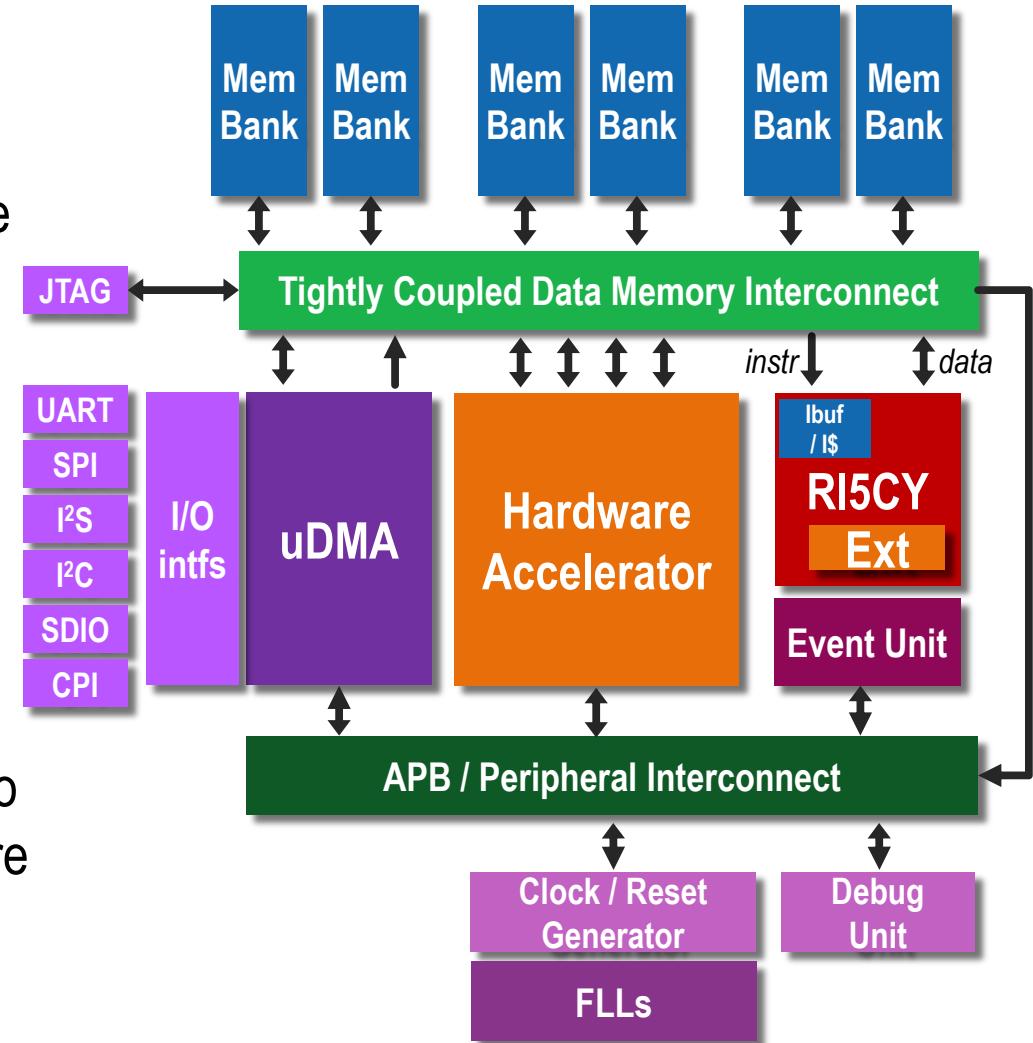
PULPino our first single core platform

- **Simple design**
 - Meant as a quick release
- **Separate Data and Instruction memory**
 - Makes it easy in HW
 - Not meant as a Harvard arch.
- **Can be configured to work with all our 32bit cores**
 - RI5CY, Zero/Micro-Riscy
- **Peripherals copied from its larger brothers**
 - Any AXI and APB peripherals could be used



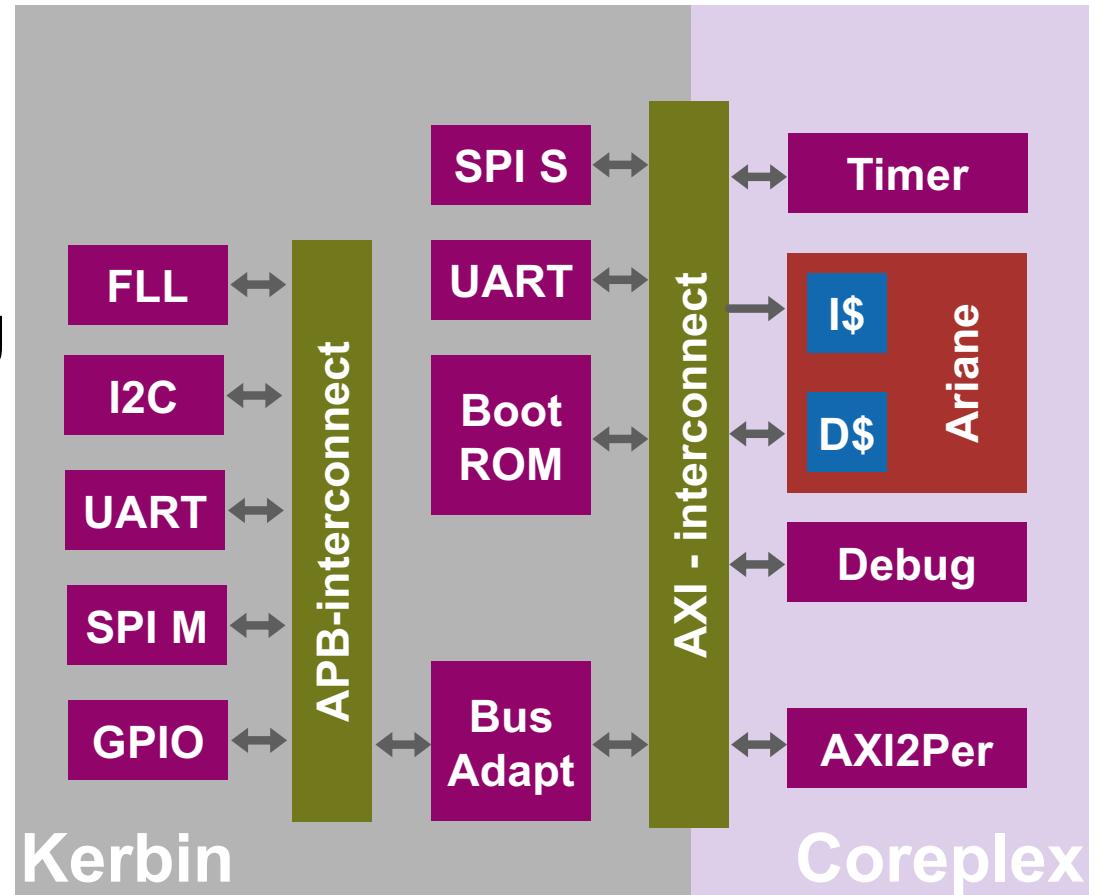
PULPiSSimo the improved single core platform

- **Shared memory**
 - Unified Data/Instruction Memory
 - Uses the multi-core infrastructure
- **Support for Accelerators**
 - Direct shared memory access
 - Programmed through APB bus
 - Number of TCDM access ports determines max. throughput
- **uDMA for I/O subsystem**
 - Can copy data directly from I/O to memory without involving the core
- **Used as a fabric controller in larger PULP systems**



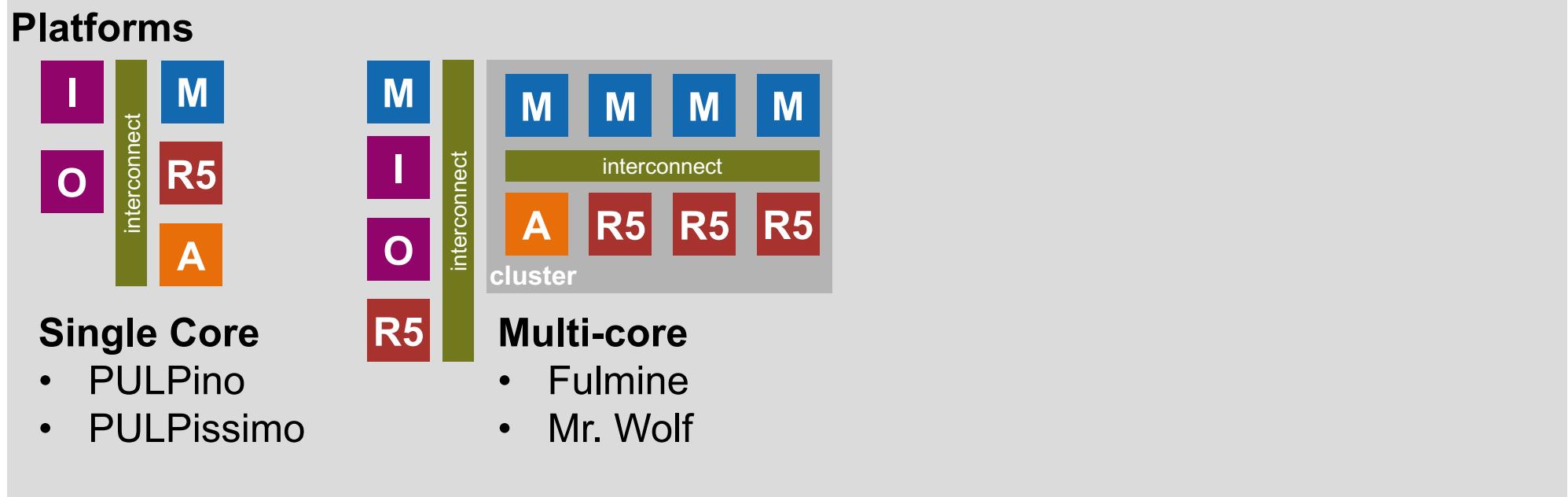
Kerbin the single core support structure for Ariane

- Still work in progress
 - Current version is very simple
 - Useful for in-house testing
 - A more advanced version will likely be developed soon.



Finally for HPC applications we have multi-cluster systems

RISC-V Cores				Peripherals		Interconnect
RI5CY 32b	Micro riscy 32b	Zero riscy 32b	Ariane 64b	JTAG	SPI	Logarithmic interconnect
				UART	I2S	APB – Peripheral Bus
				DMA	GPIO	AXI4 – Interconnect

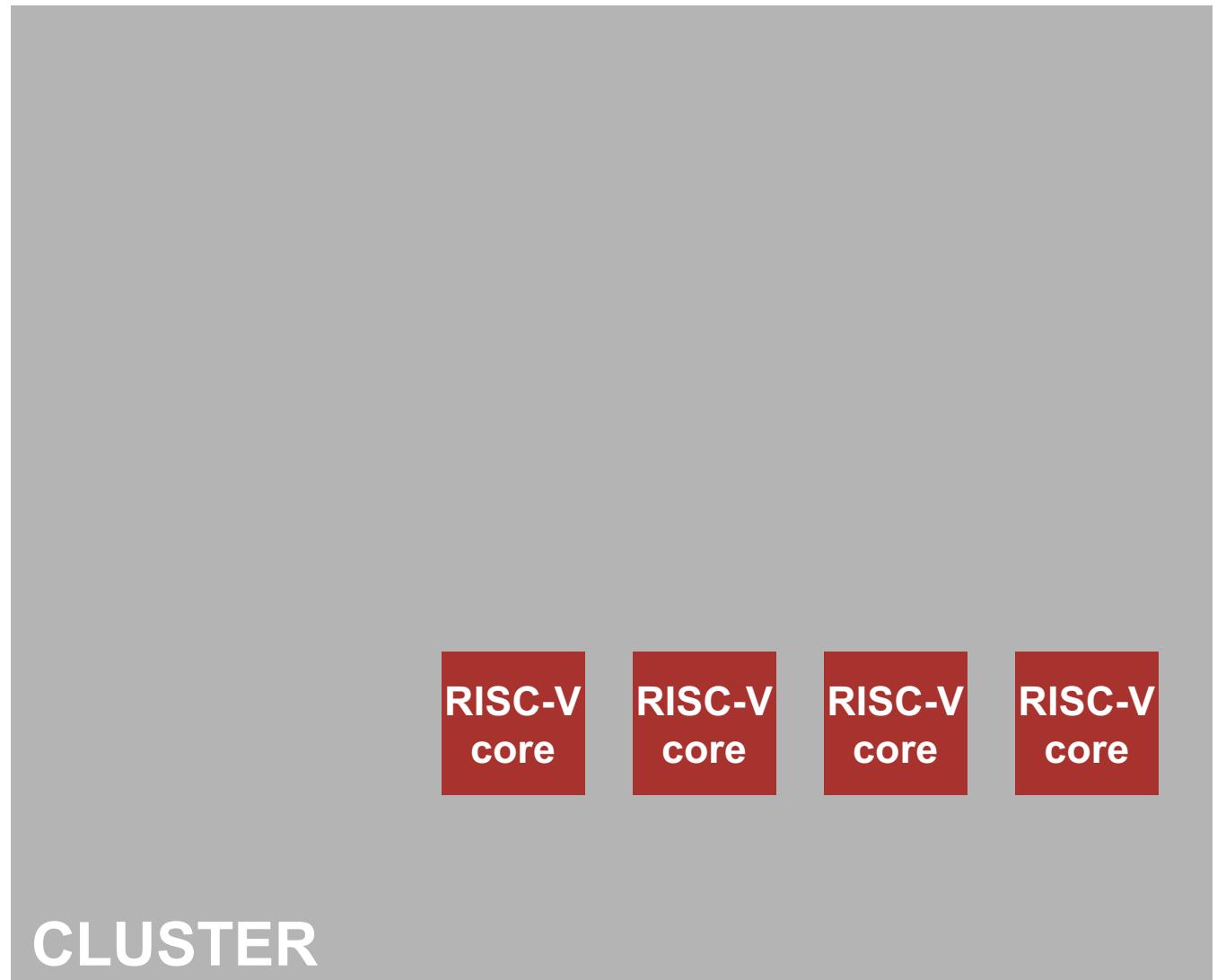


Accelerators			
HWCE (convolution)	Neurostream (ML)	HWCrypt (crypto)	PULPO (1 st order opt)

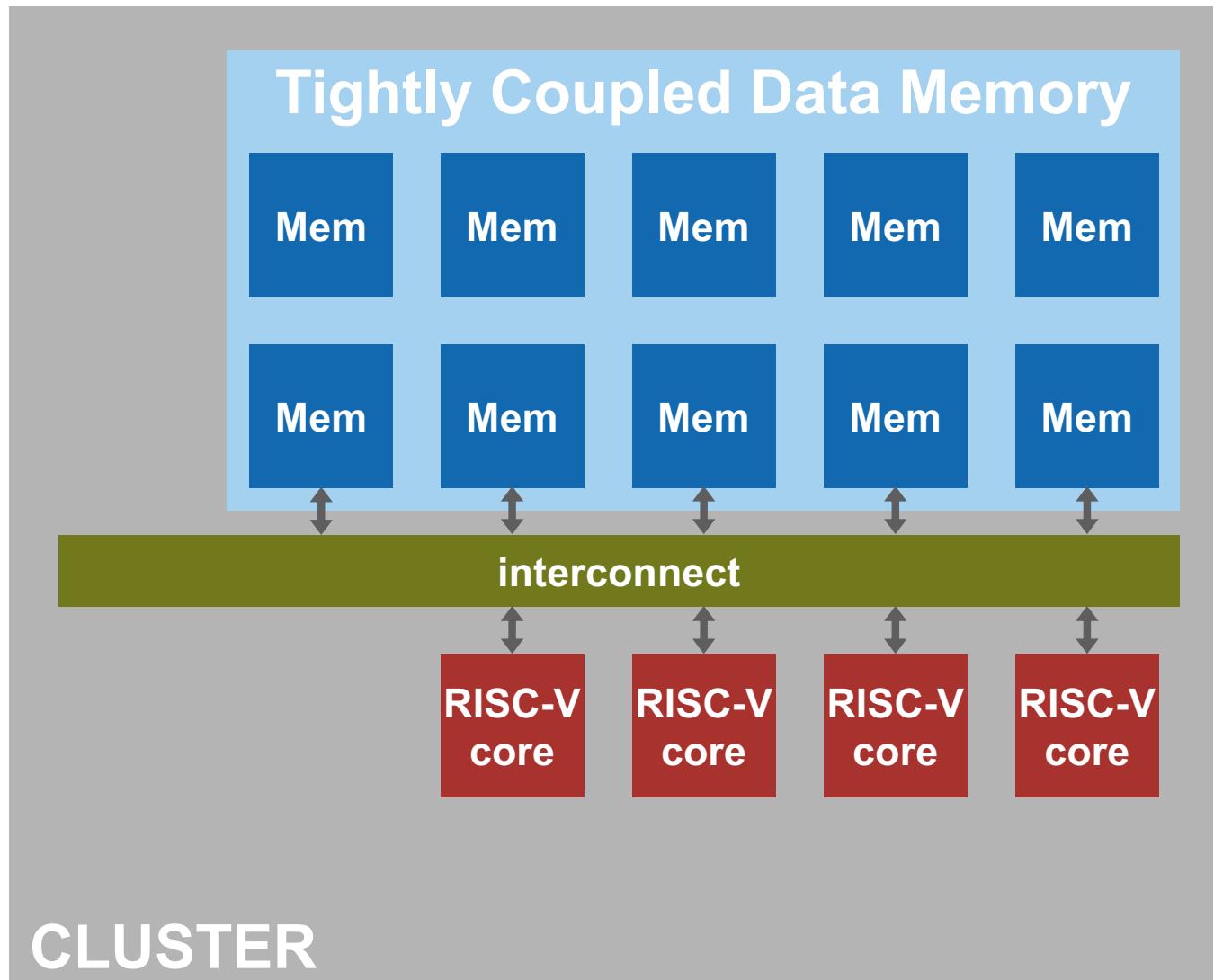
The main components of a PULP cluster

- **Multiple RISC-V cores**
 - Individual cores can be started/stopped with little overhead
 - DSP extensions in cores
- **Multi-banked scratchpad memory (TCDM)**
 - **Not a cache**, there is no L1 data cache in our systems
- **Logarithmic Interconnect allowing all cores to access all banks**
 - Cores will be stalled during contention, includes arbitration
- **DMA engine to copy data to and from TCDM**
 - Data in TCDM managed by software
 - Multiple channels, allows pipelined operation
- **Hardware accelerators with direct access to TCDM**
 - No data copies necessary between cores and accelerators.

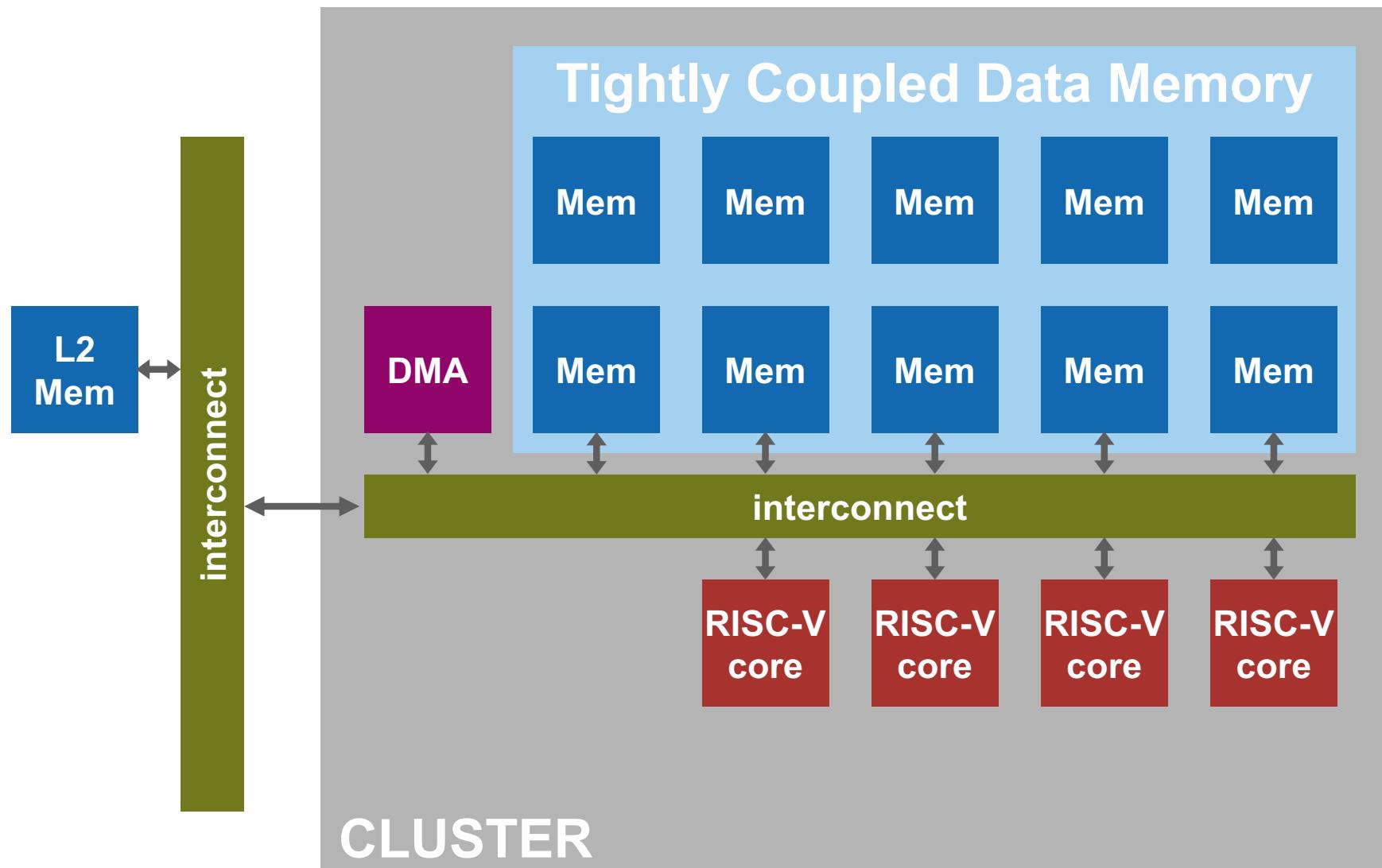
PULP cluster contains multiple RISC-V cores



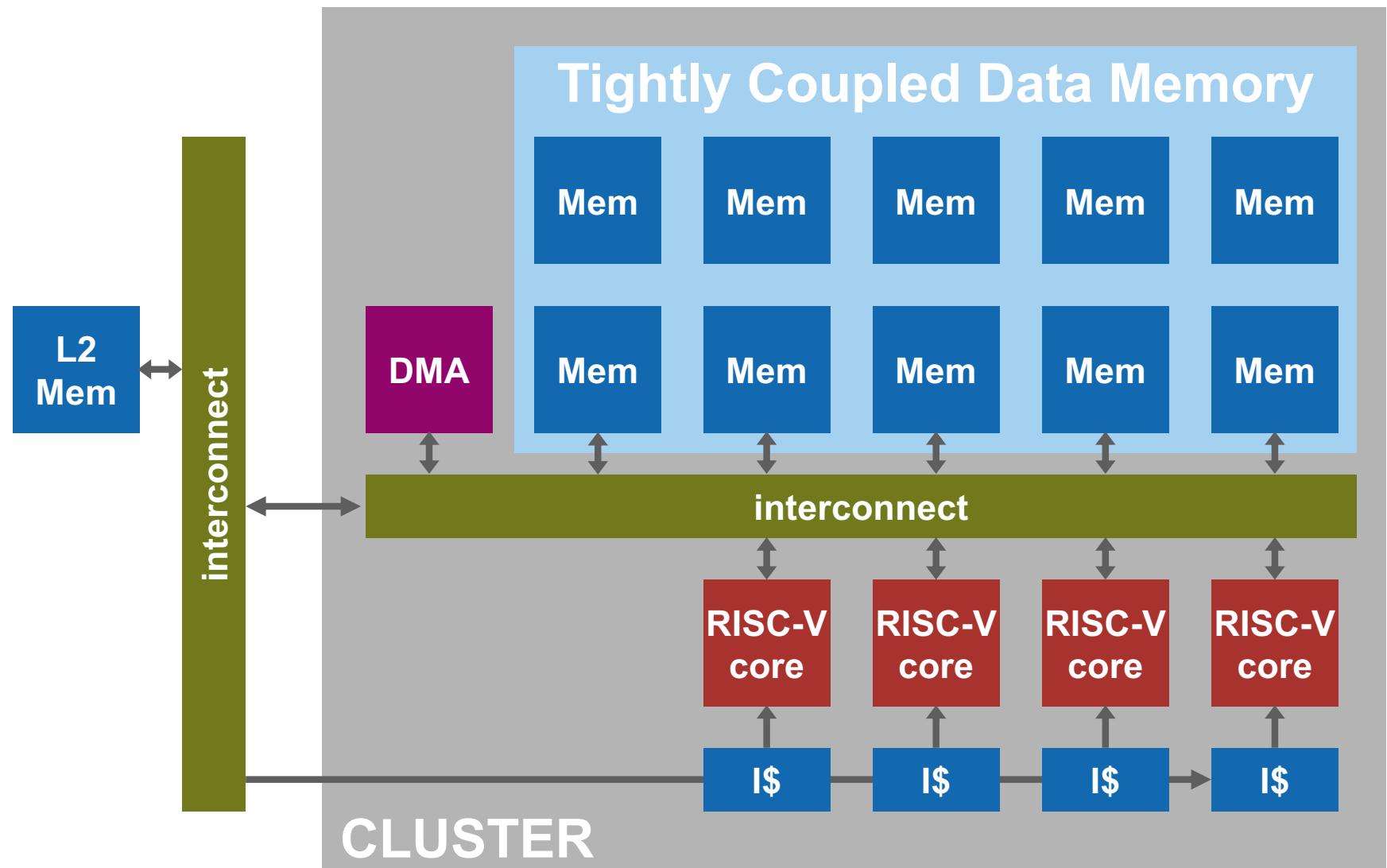
All cores can access all memory banks in the cluster



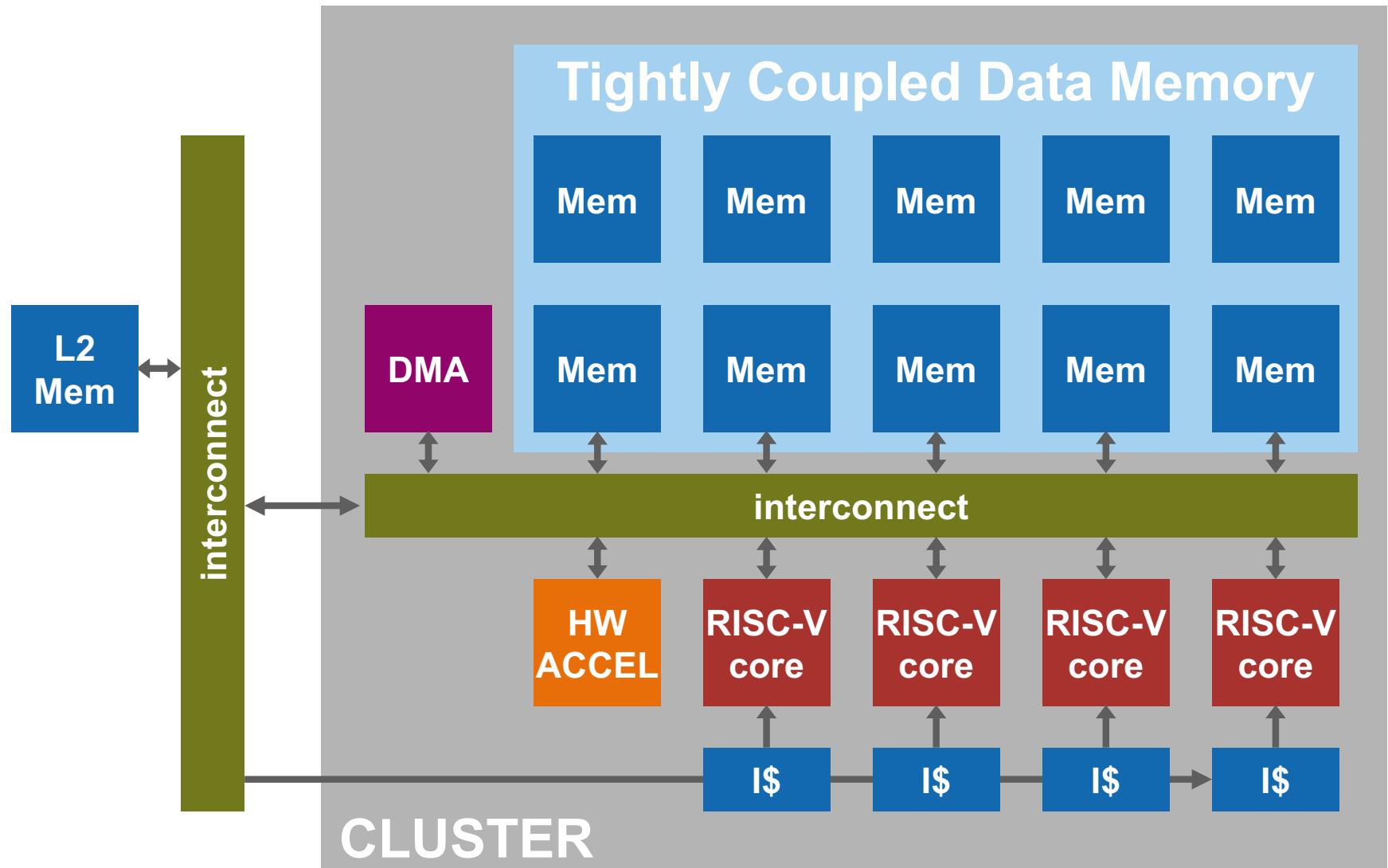
Data is copied from a higher level through DMA



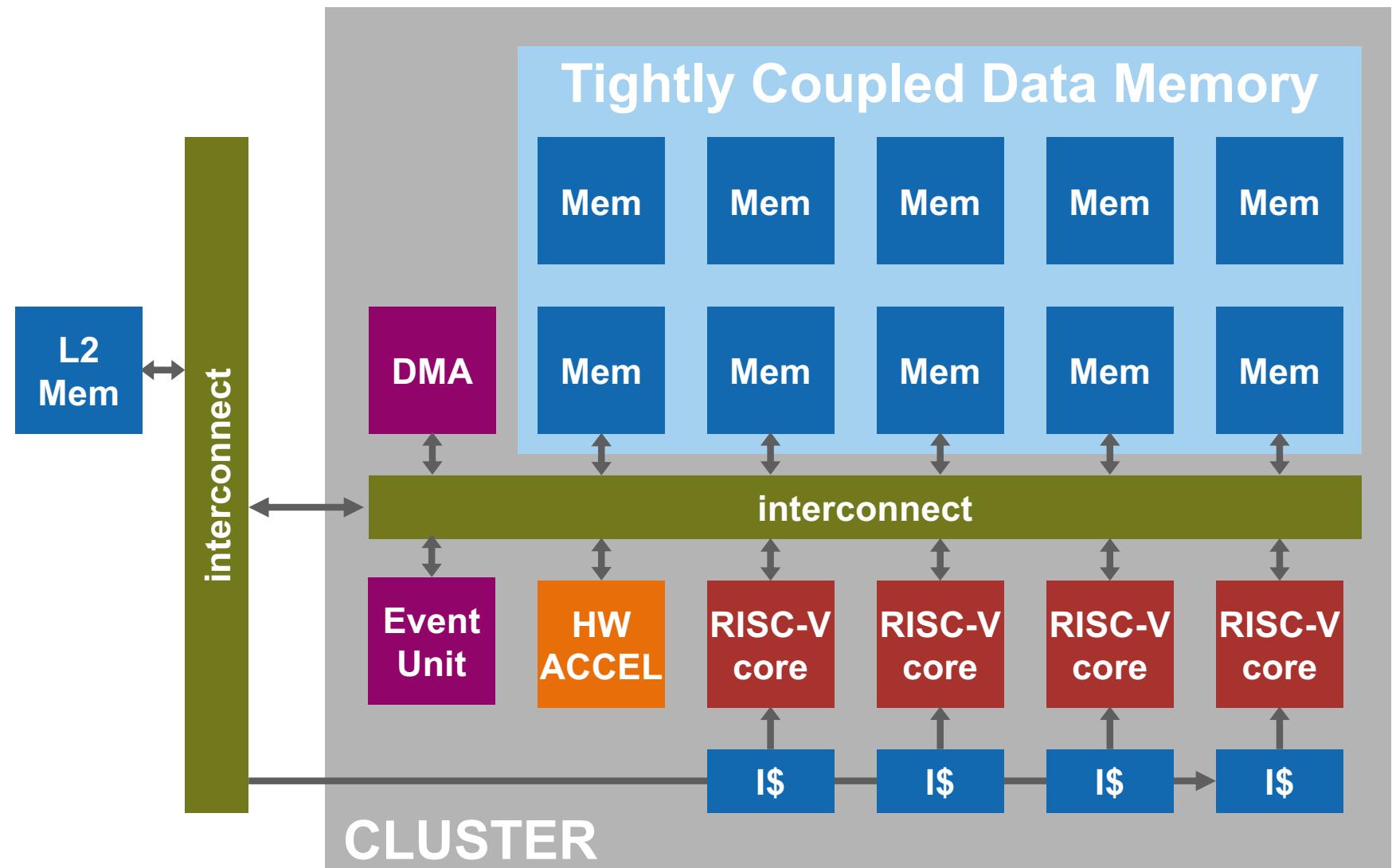
There is a (shared) instruction cache that fetches from L2



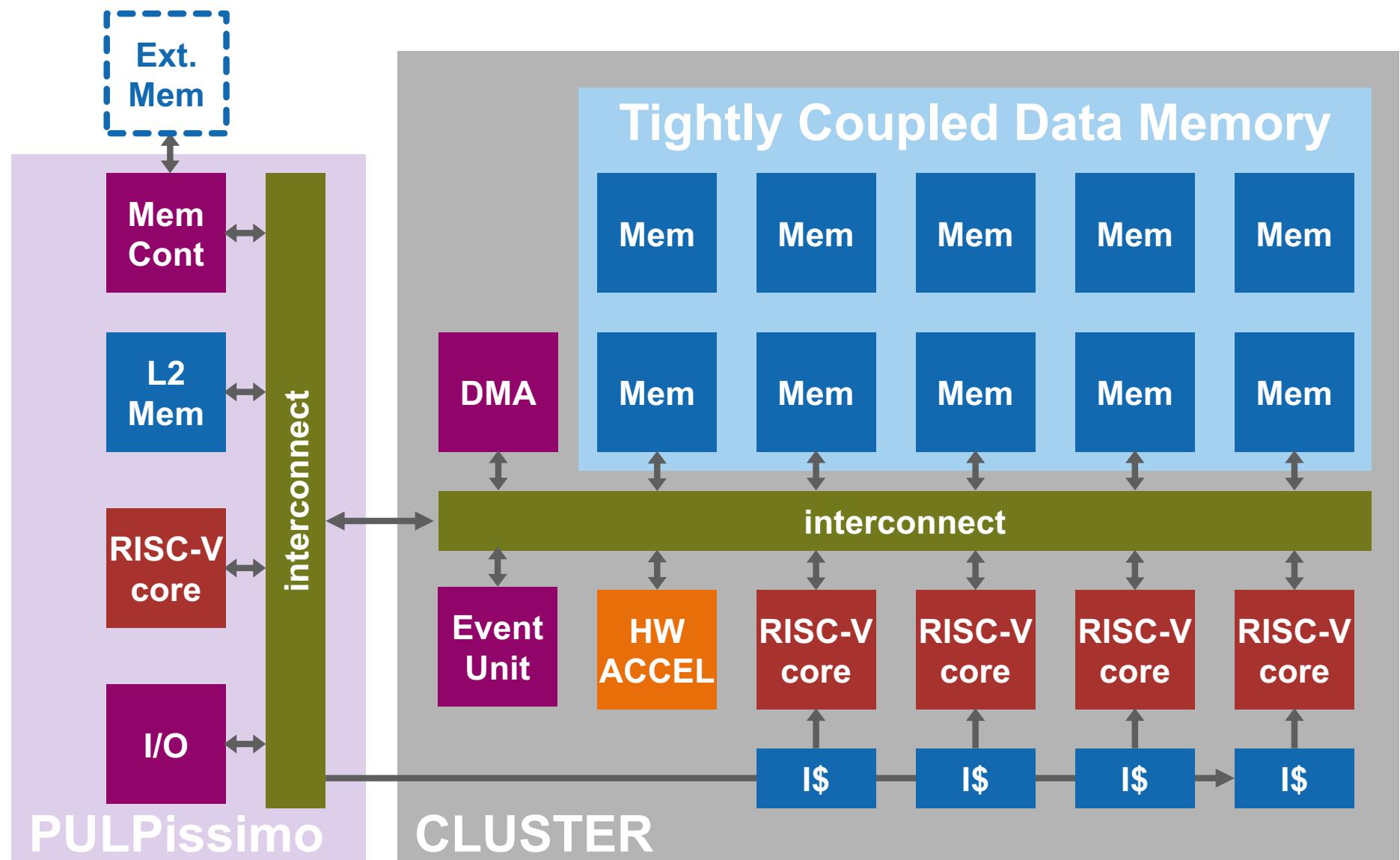
Hardware Accelerators can be added to the cluster



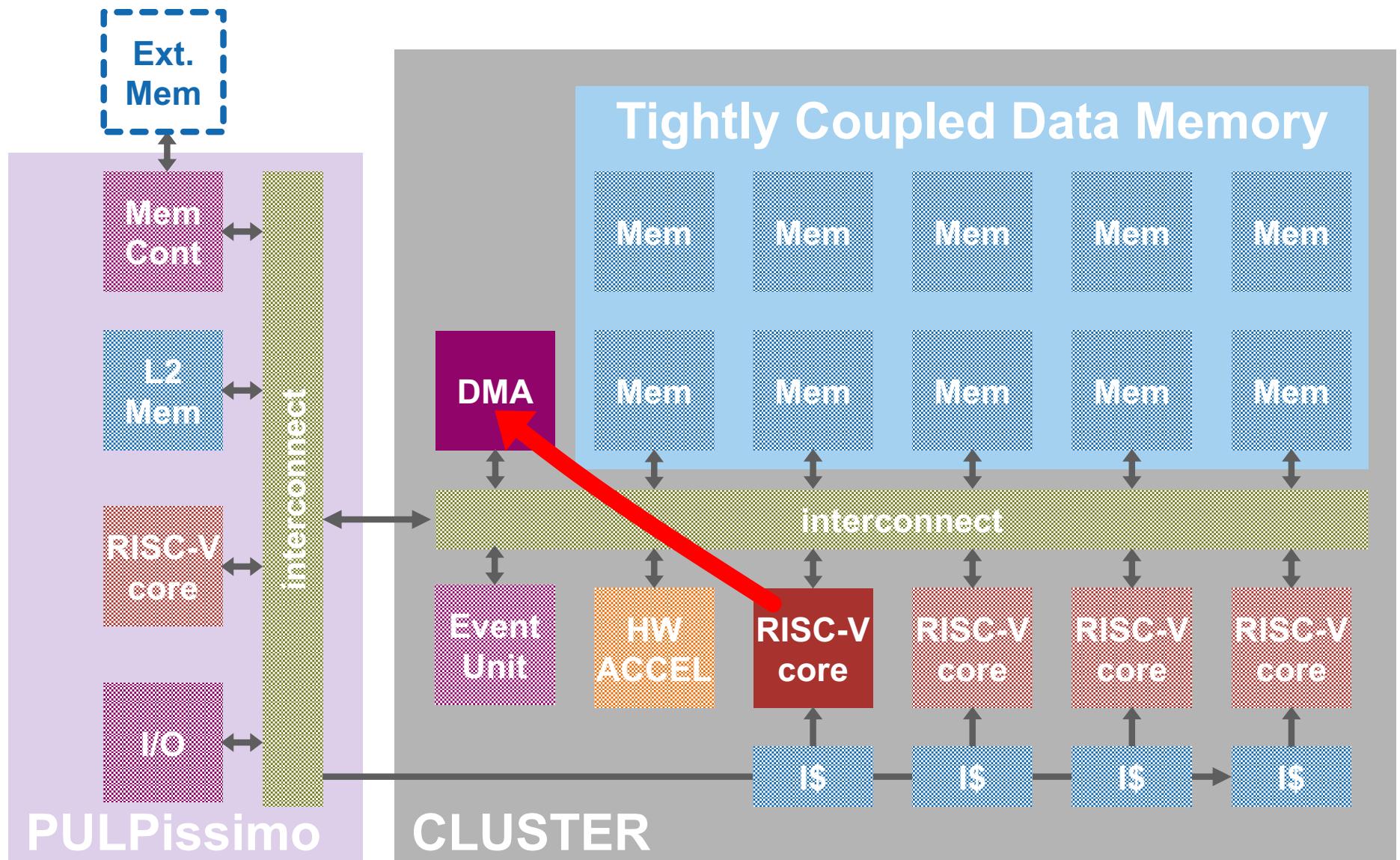
Event unit to manage resources (fast sleep/wakeup)



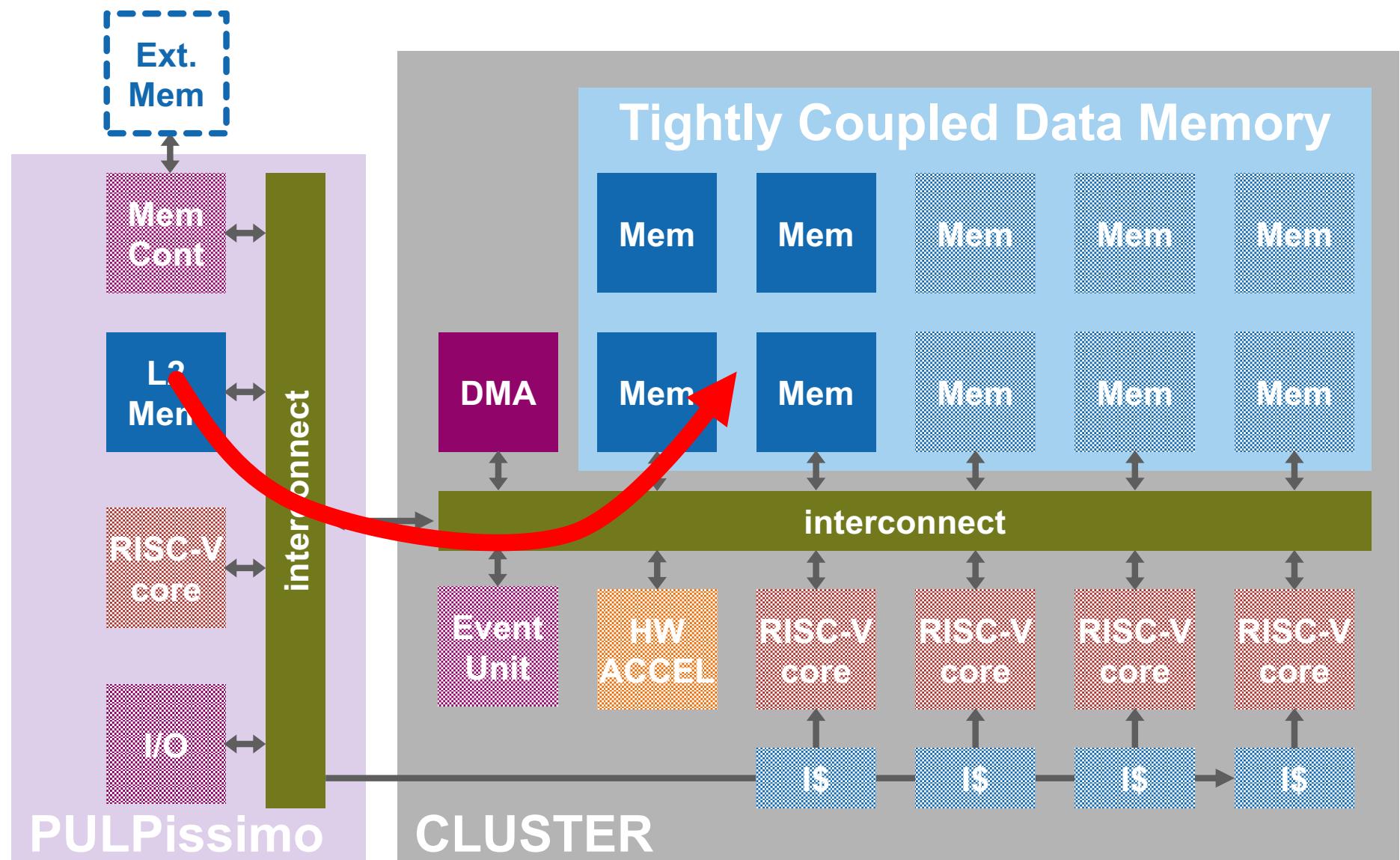
An additional microcontroller system (PULPissimo) for I/O



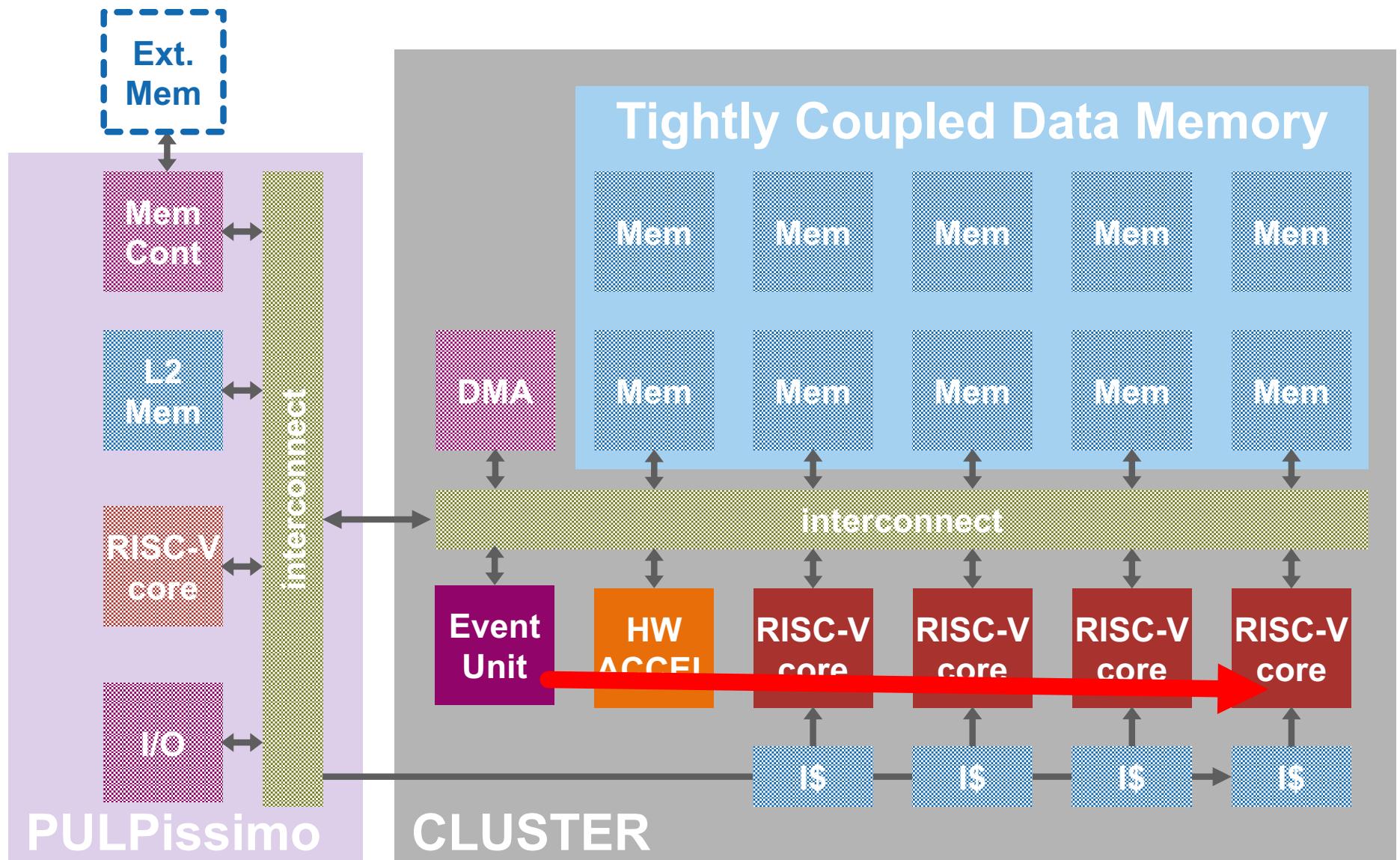
How do we work: Initiate a DMA transfer



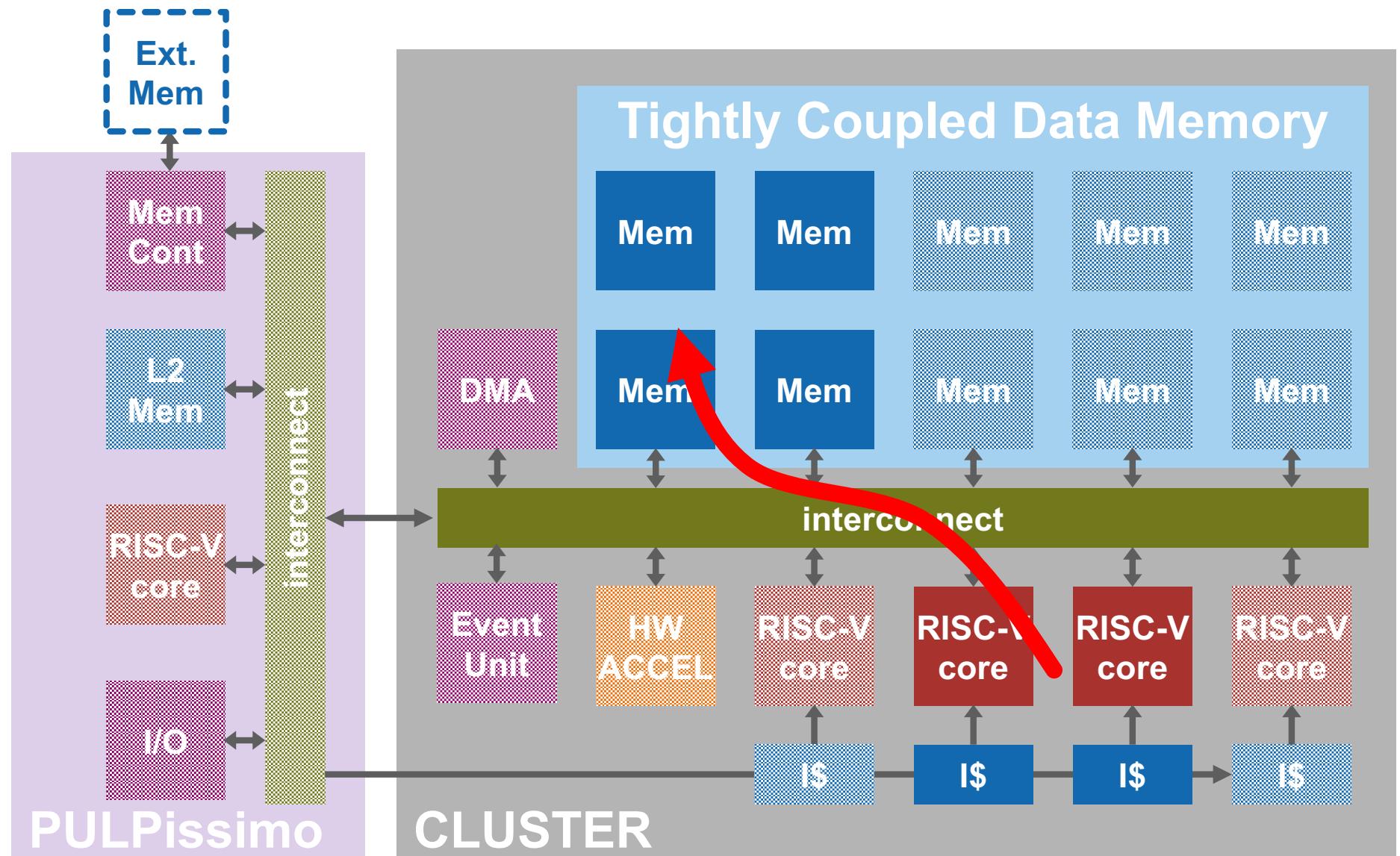
Data copied from L2 into TCDM



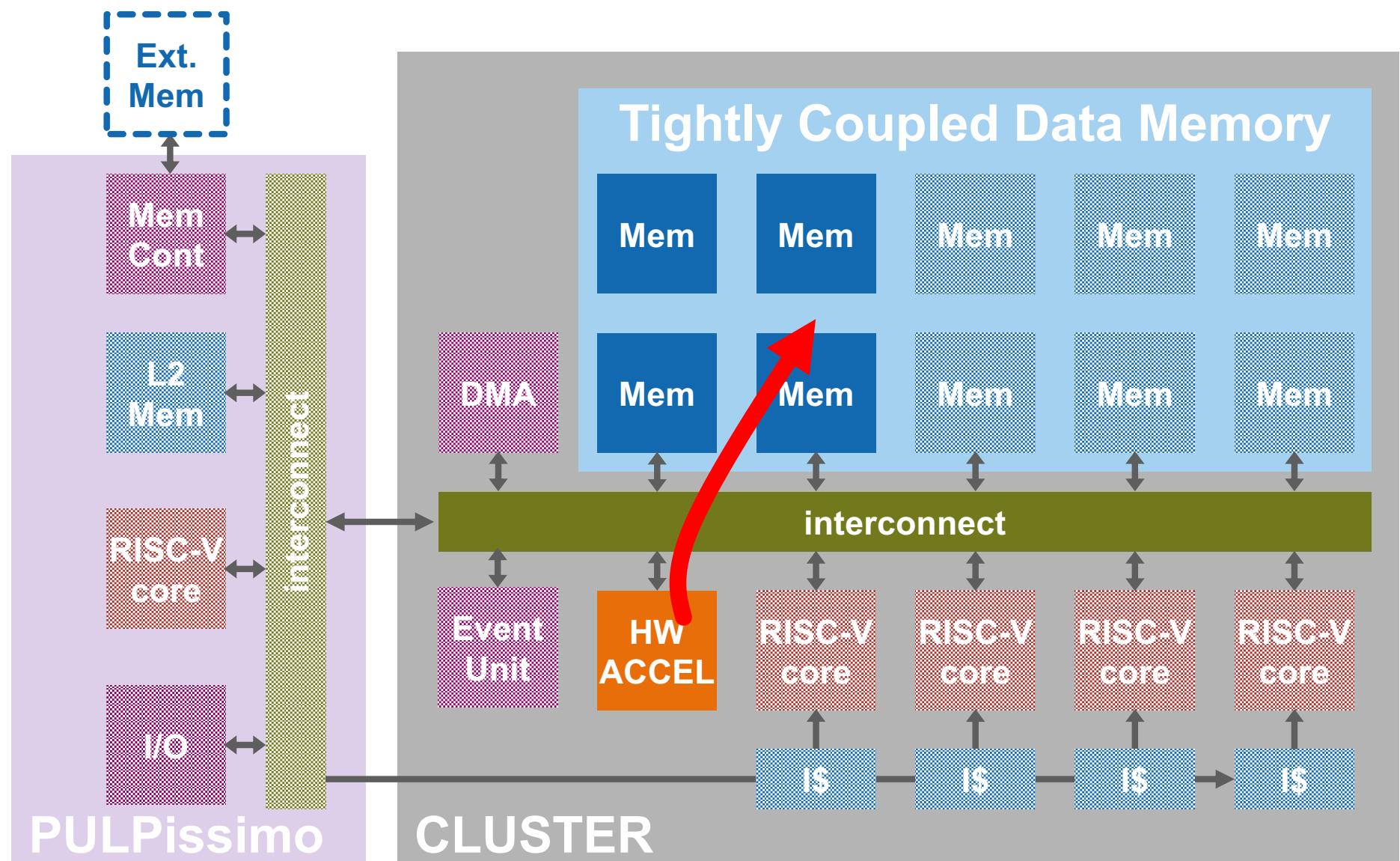
Once data is transferred, event unit notifies cores/accel



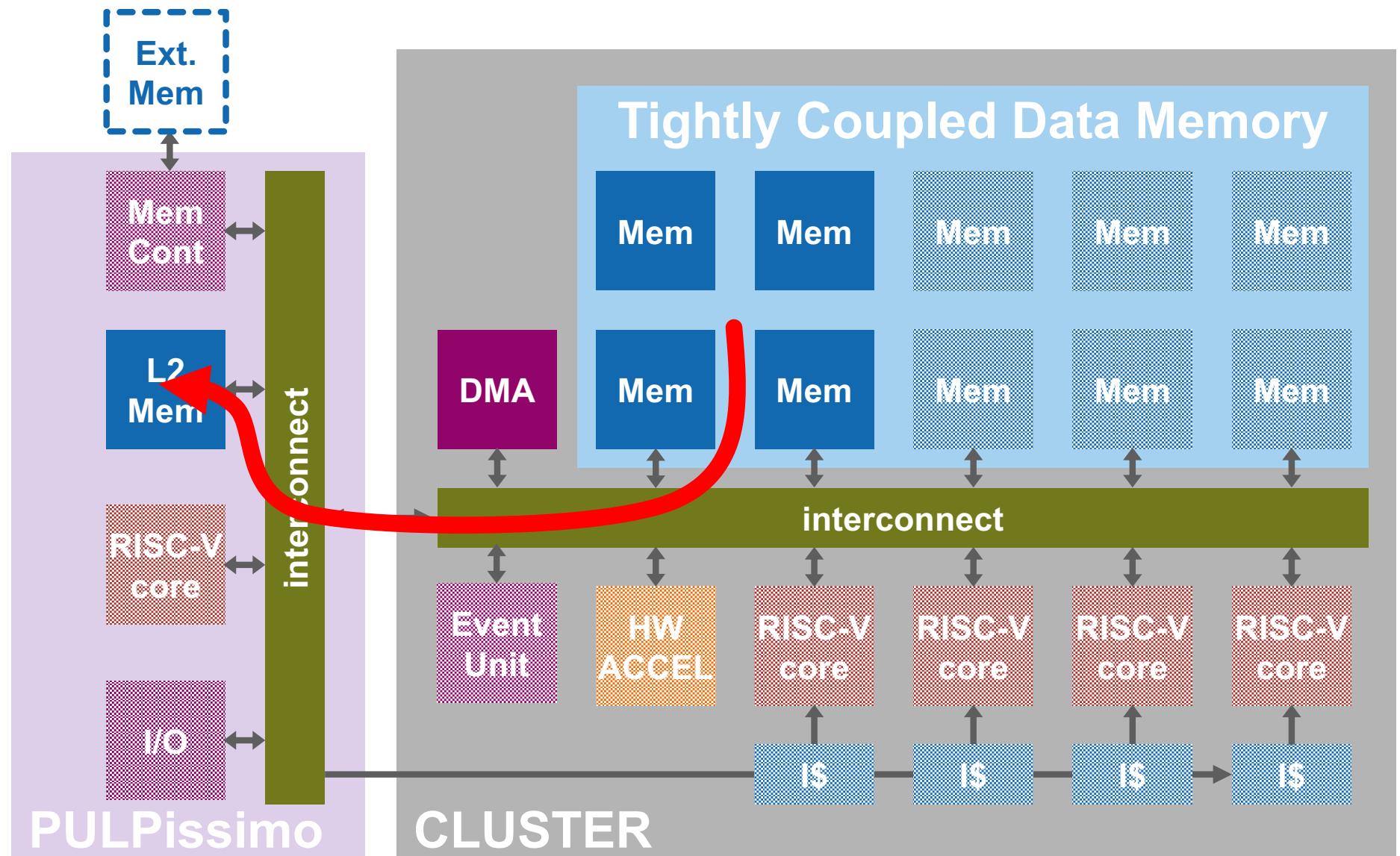
Cores can work on the data transferred



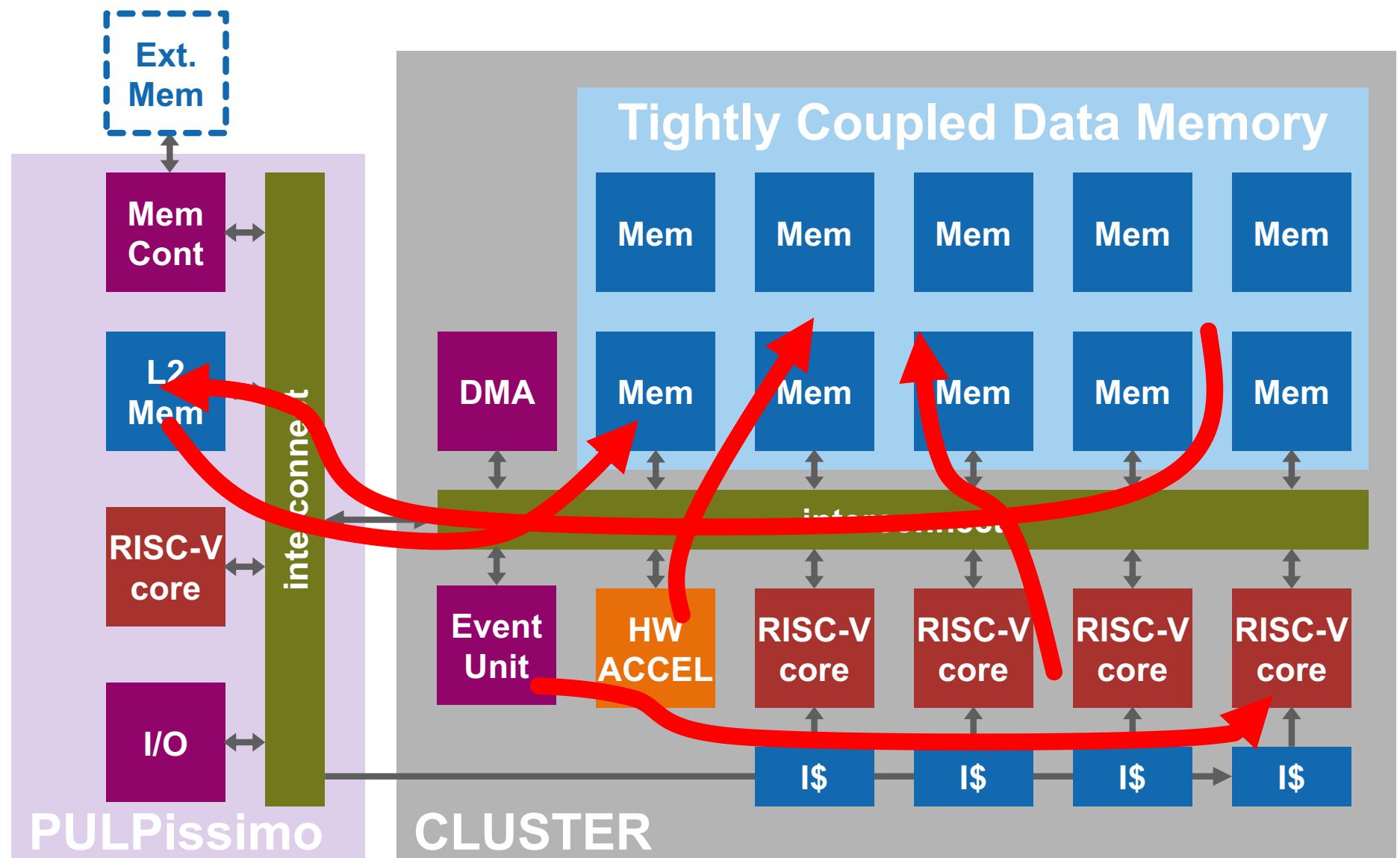
Accelerators can work on the same data



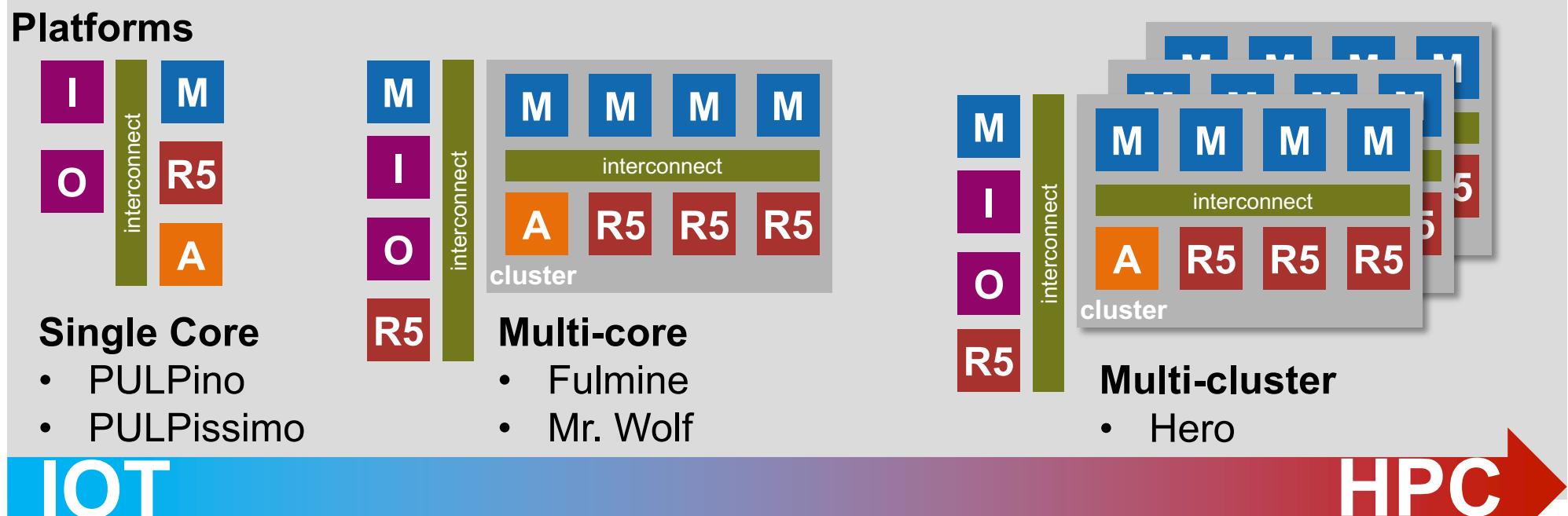
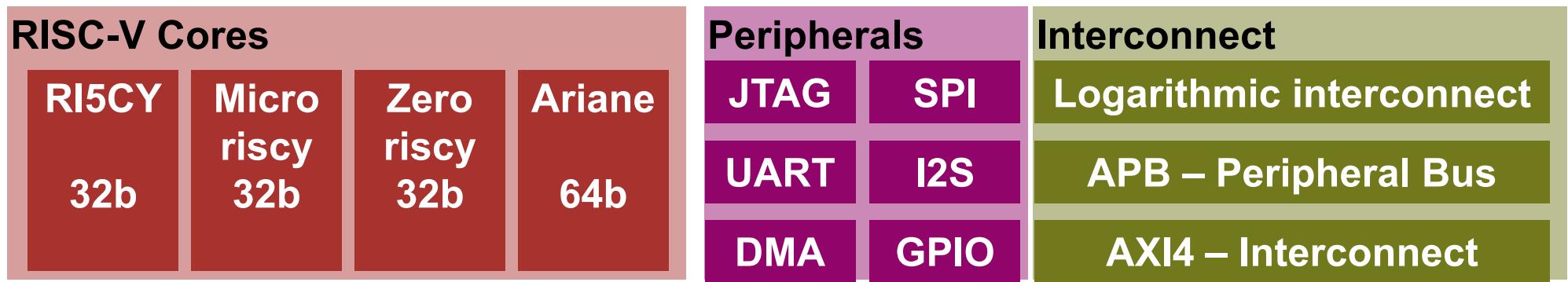
Once our work is done, DMA copies data back



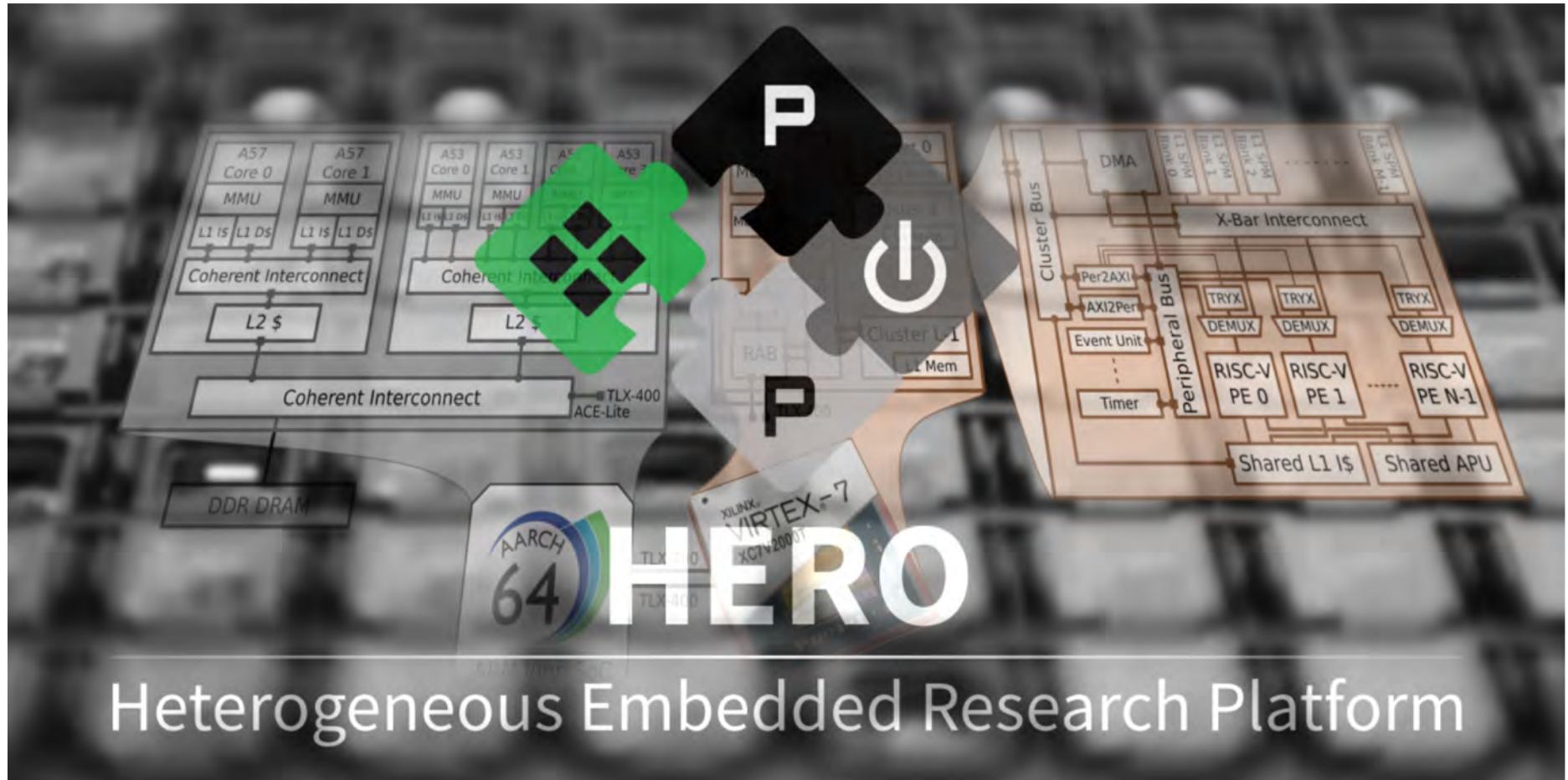
During normal operation all of these occur concurrently



Finally for HPC applications we have multi-cluster systems



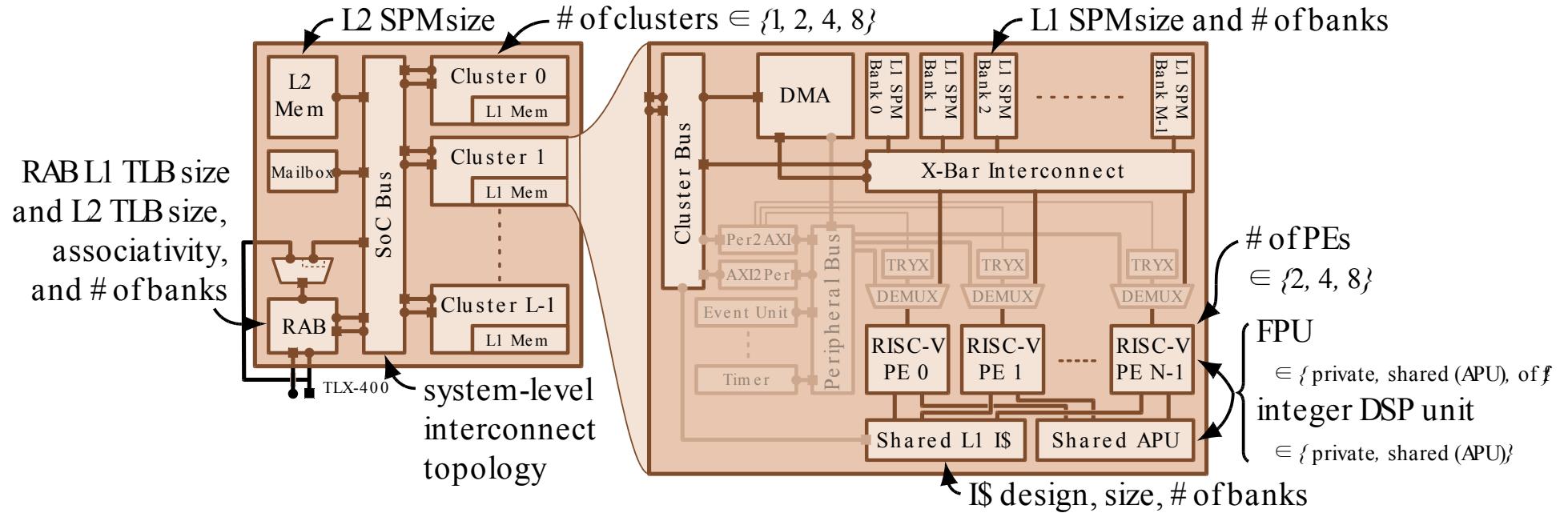
We will release HERO soon



- Multi-cluster PULP system on FPGA connected to ARM cores
- Will be made available in 3Q2018 (soon).

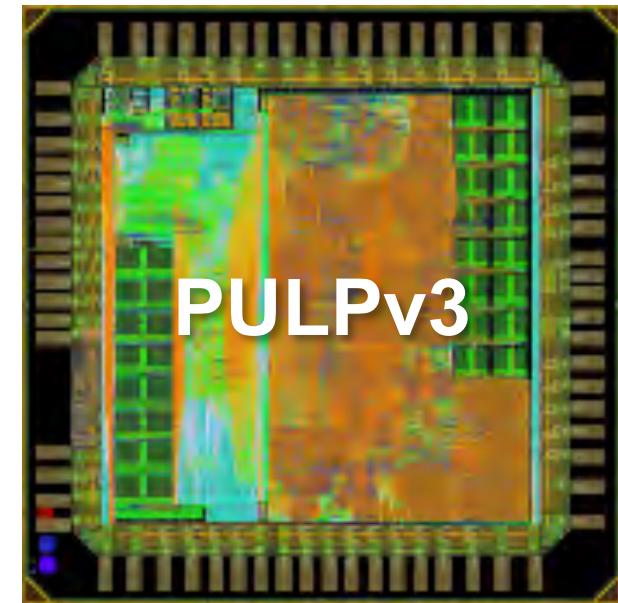
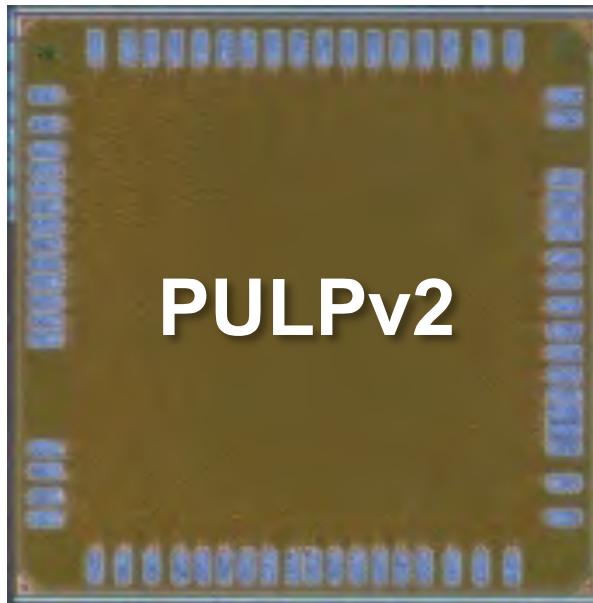
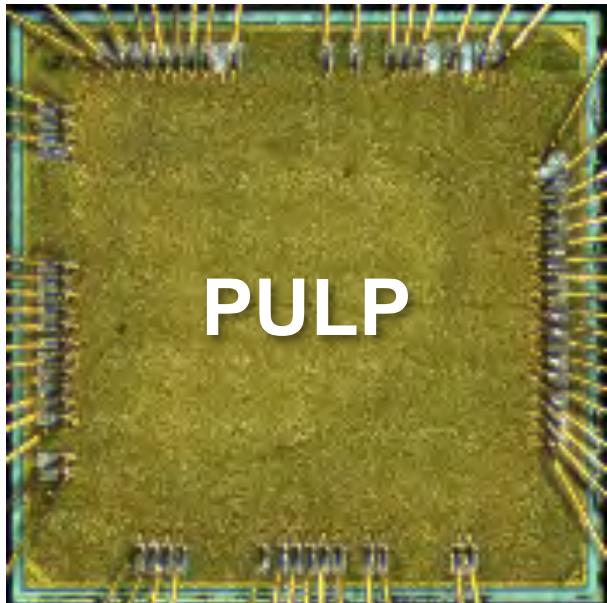


HERO is modifiable and expandable



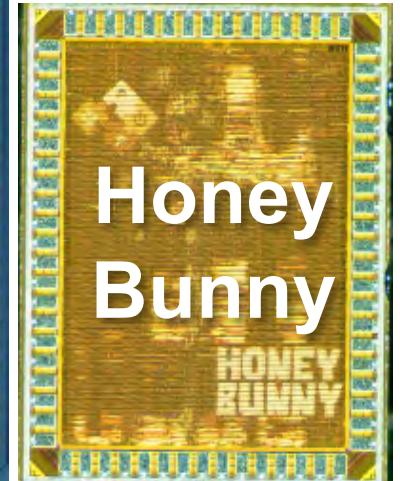
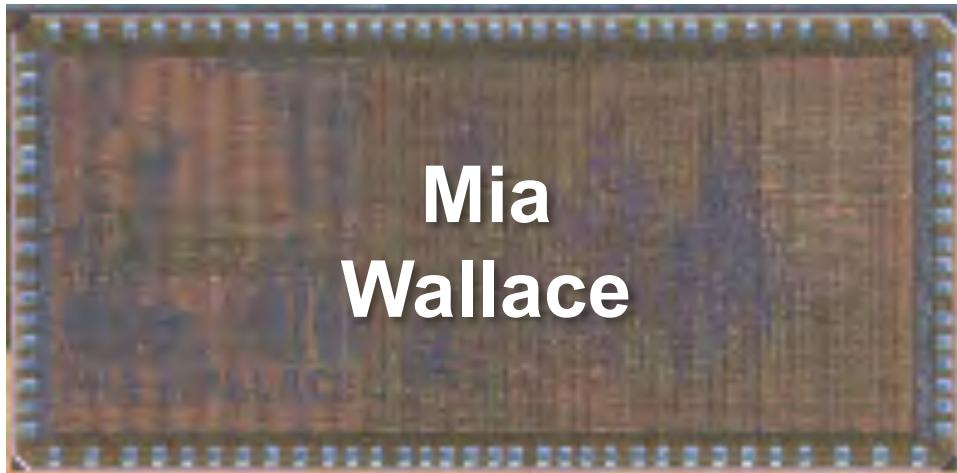
- Up to 8 clusters, each with 8 cores
- All components are open source and written in System Verilog
- Standard interfaces (mostly AXI)
- New components can easily be added to the memory map

Brief illustrated history of selected ASICs



- All are 28 FDSOI technology, RVT, LVT and RVT flavor
- Uses OpenRISC cores
- Chips designed in collaboration with STM, EPFL, CEA/LETI
- PULPv3 has ABB control

The first system chips, meant for designing boards



- First multi-core systems that were designed to work on development boards. Each have several peripherals (SPI, I2C, GPIO)
- **Mia Wallace** and **Fulmine** (UMC65) use OpenRISC cores
- **Honey Bunny** (GF28 SLP) uses RISC-V cores
- All chips also have our own FLL designs.

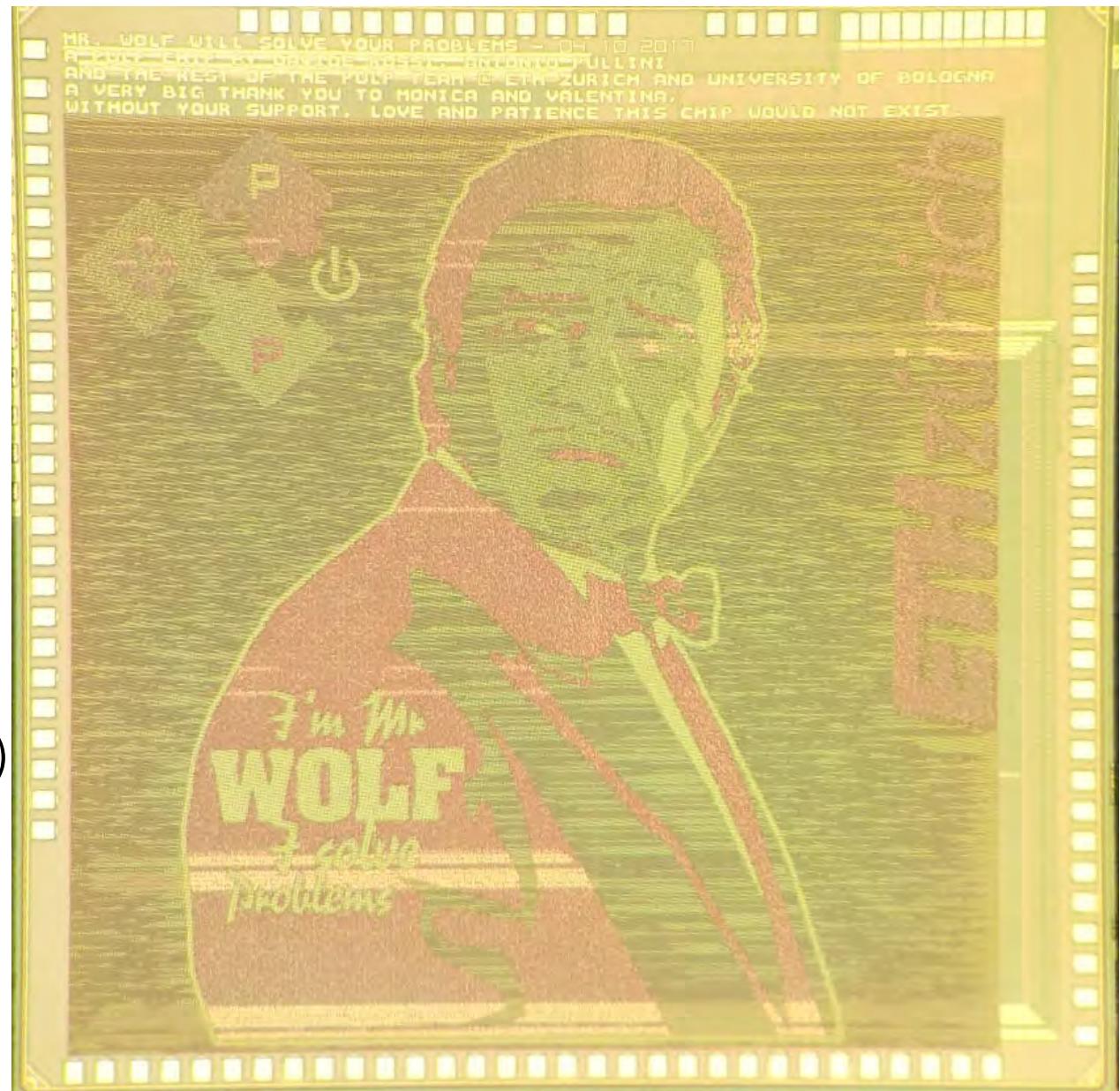
Combining PULP with analog front-end for Biomedical apps



- Designed in collaboration with the Analog group of Prof. Huang
- All chips with SMIC130 (because of analog IPs)
- First three with OpenRISC, VivoSoC3 with RISC-V

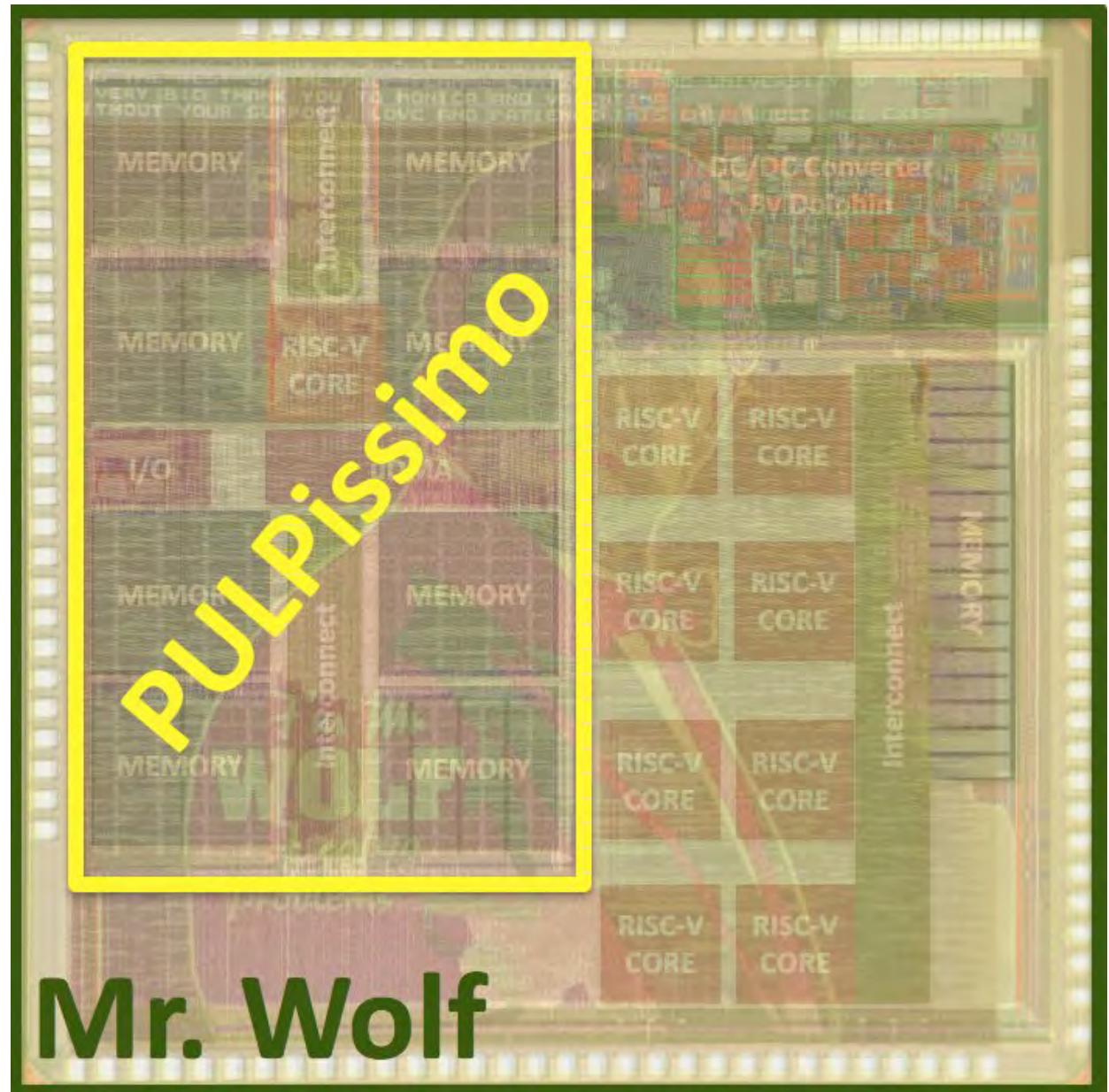
Mr. Wolf our latest system chip will solve problems

- TSMC 40LP - 9mm²
- 64 pin QFN package
- Cluster with
 - 8x RI5CY
 - 2x shared FPUs
 - 64 kByte TCDM
- SoC domain with
 - micro-riscy core to control power modes
 - DC-DC converter (Dolphin) to regulate power
 - 512 kBytes L2

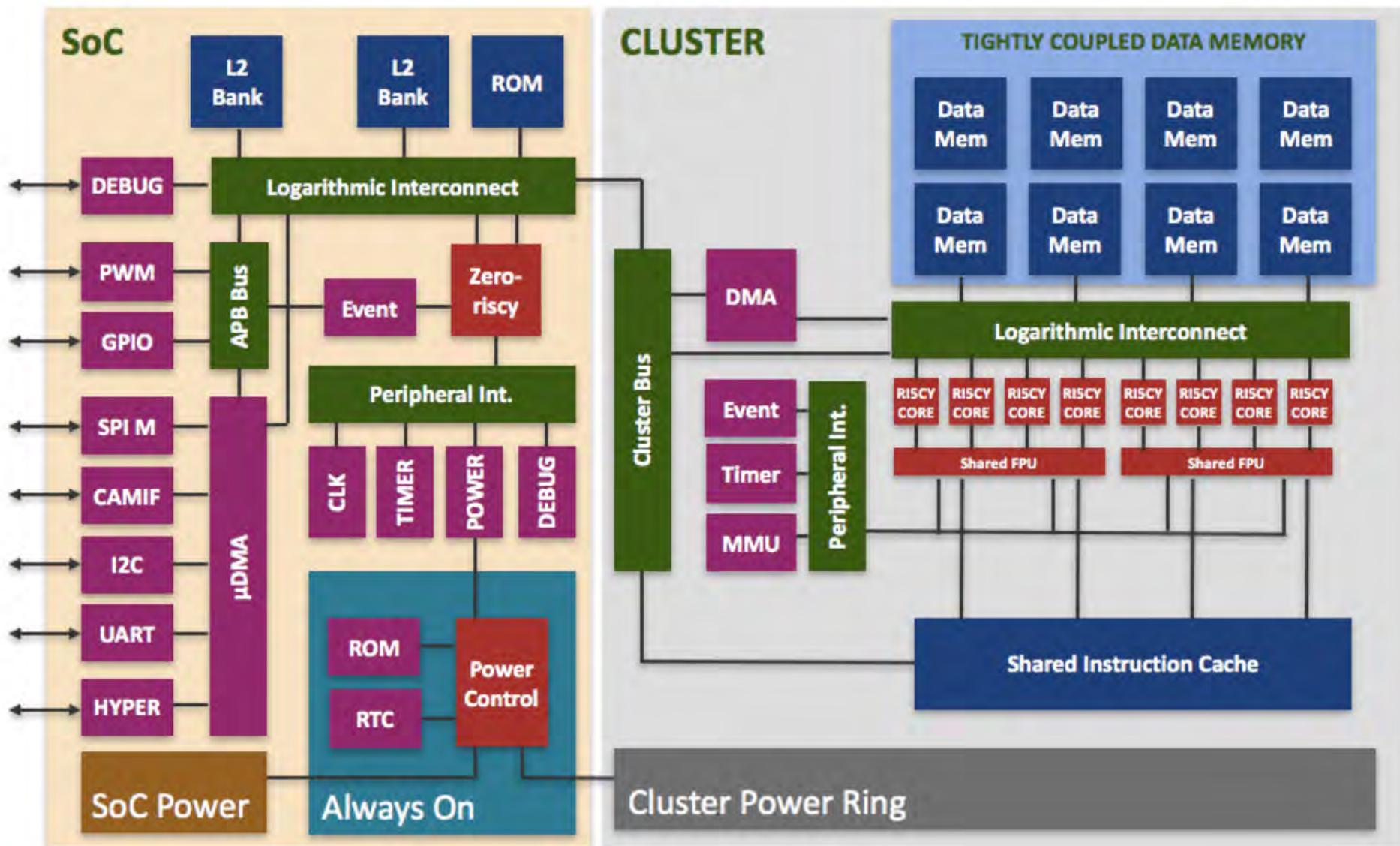


Mr. Wolf will (hopefully) solve problems

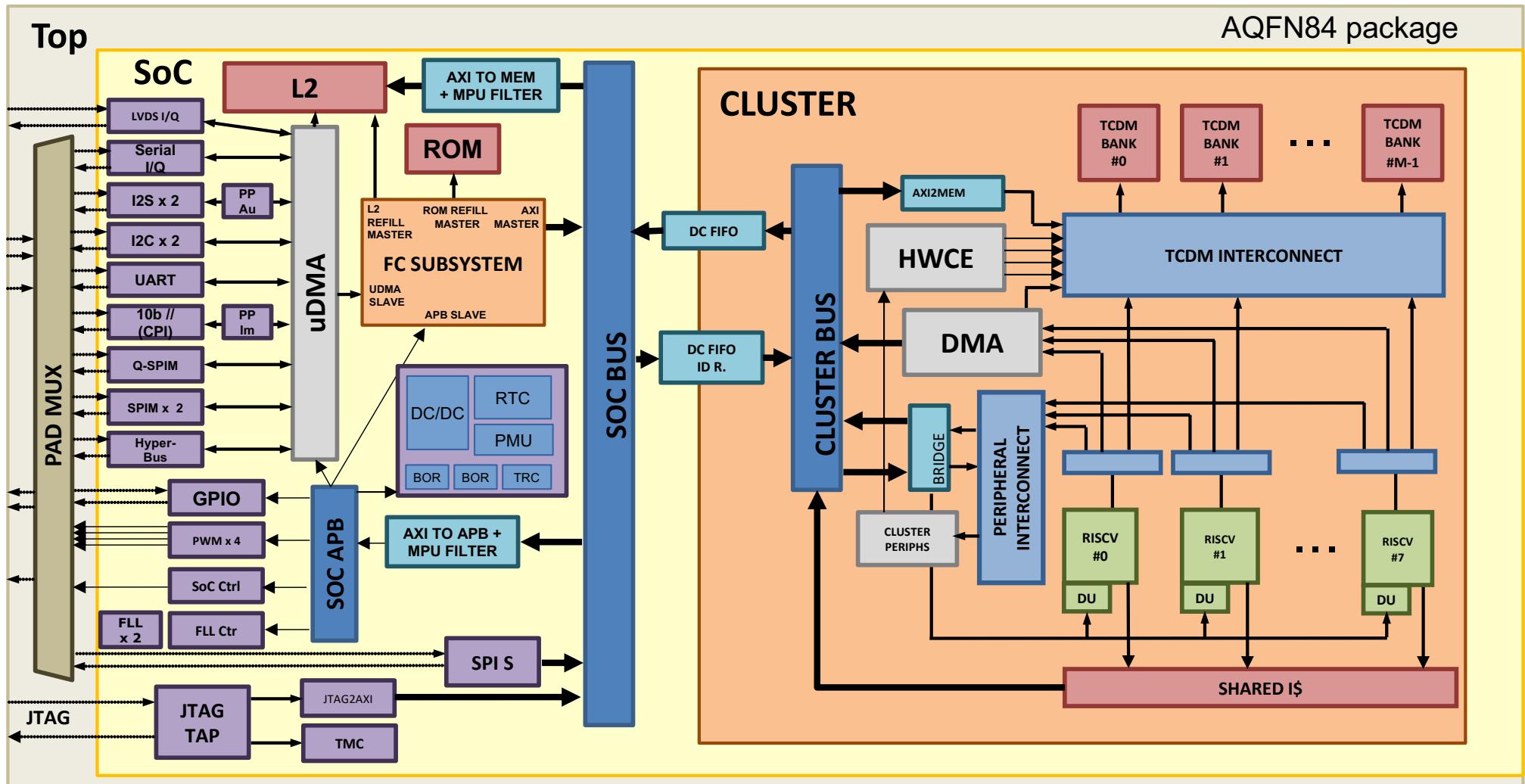
- TSMC 40LP - 9mm²
- 64 pin QFN package
- Cluster with
 - 8x RI5CY
 - 2x shared FPUs
 - 64 kByte TCDM
- SoC domain with
 - micro-riscy core to control power modes
 - DC-DC converter (Dolphin) to regulate power
 - 512 kBytes L2



PULP in the IoT domain: Latest version Mr. Wolf

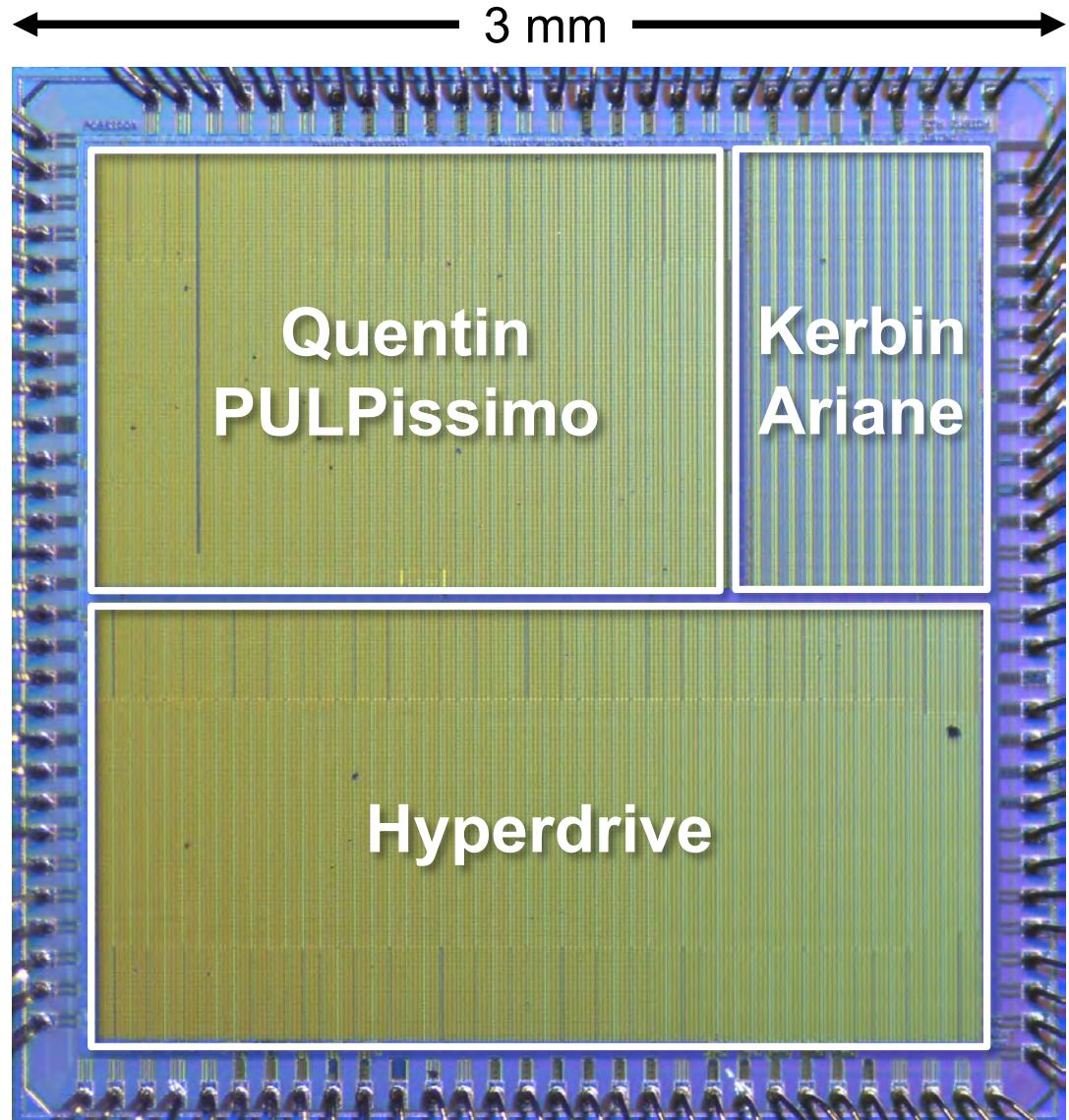


GAP8: commercial big brother of PULP from Greenwaves



Poseidon: Our latest chip in GF 22FDX

- Taped out Jan 2018
- Arrived back June 2018
- **Quentin**
 - PULPissimo implementation
 - RI5CY+ FPU + HWCE
 - 512 kByte RAM
- **Kerbin**
 - Ariane core + Caches
 - Uses the memory infrastructure of Quentin
- **Hyperdrive**
 - Binary CNN Accelerator

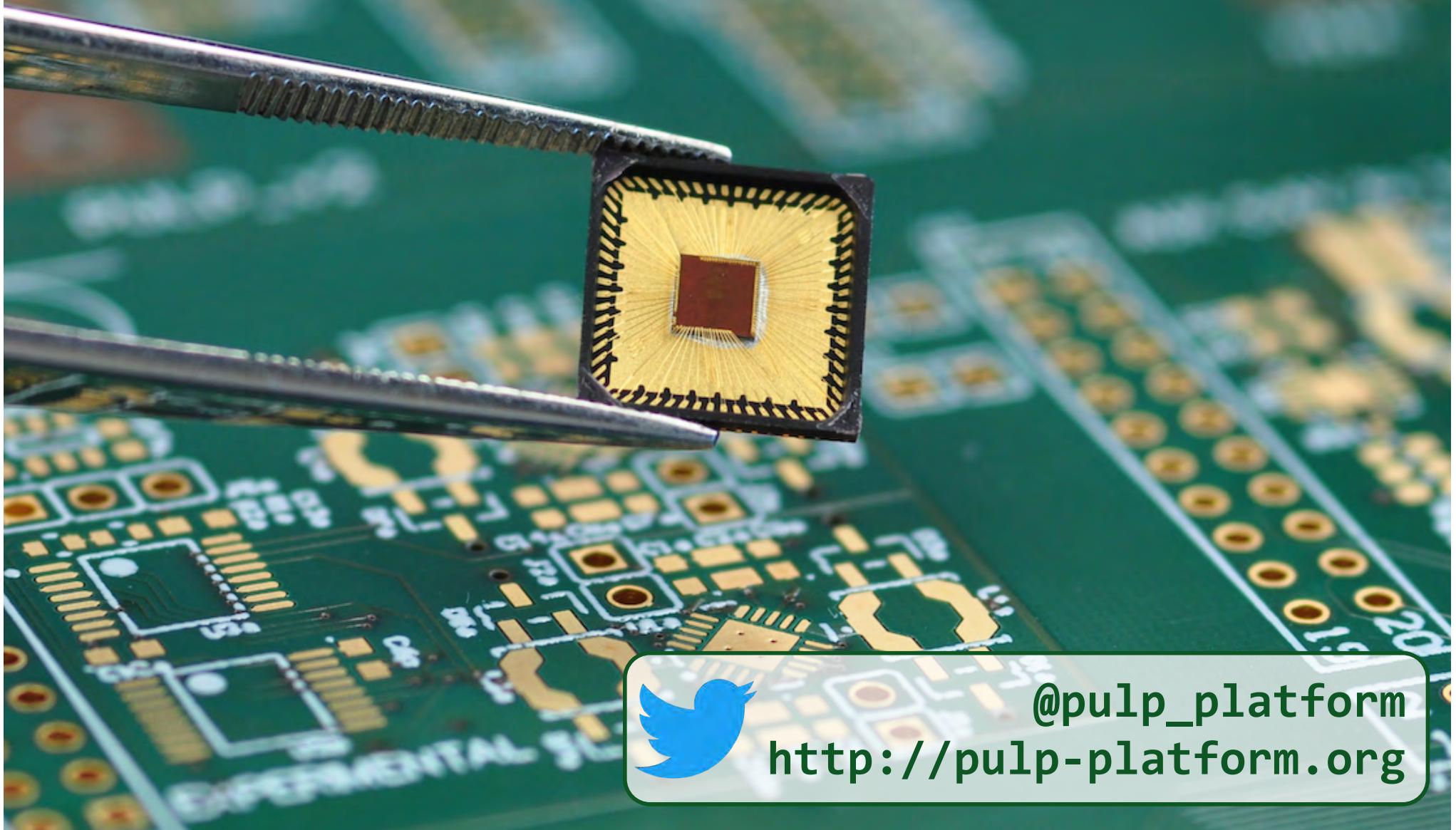


What is next

- **For the workshop:**
 - Davide will give details on how we do thing in PULP next
 - Tomorrow there will be hands on sessions for GAP8 and Multicuster PULP
- **For us:**
 - Plenty of chips in pipeline (Kosmodrom, Vega, Villalobos)
 - Multi-core Ariane
 - Atomic instructions, 64bit FPU, Vector Unit
- **For you:**
 - There is plenty to be done
 - Let us know if you want to/can contribute



QUESTIONS?



@pulp_platform
<http://pulp-platform.org>