



**PULP**  
Parallel Ultra Low Power

# RISC-V Tutorial

HiPEAC 2019, Valencia

21.01.2019

**Frank K. Gürkaynak, Michael Schaffner,  
Fabian Schuiki, Andreas Kurth**



<sup>1</sup>*Department of Electrical, Electronic  
and Information Engineering*

<http://pulp-platform.org>

**PRECOMP**  
Open Transprecision Computing

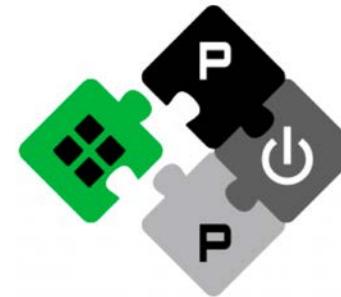
Eurolab4HPC

**ETH zürich**

<sup>2</sup>*Integrated Systems Laboratory*

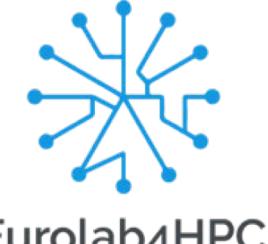
# Welcome to the RISC-V Tutorial @ HiPEAC

- We are from ETH Zürich in Switzerland
- ... and part of the **PULP** team
  - <http://pulp-platform.org>



## A word from your sponsors

- **Eurolab4HPC (800962)**
  - with the commitment to make Europe excel in academic research and innovation in HPC technology
  - <https://www.eurolab4hpc.eu/>
- **OPRECOMP (732631)**
  - aims to build an innovative, reliable foundation for computing based on transprecision analytics
  - <http://oprecomp.eu/>



Eurolab4HPC



# What's on the menu

- **10:00 – 11:00 Frank K. Gürkaynak – Introduction**
  - 11:00 – 11:30 Coffee Break
- **11:30 – 12:00 Andreas Kurth – Software Development Kit**
- **12:00 – 13:00 Fabian Schuiki – PULP and CAPI (Hands on demo)**
  - 13:00 – 14:00 Lunch
- **14:00 – 15:00 Andreas Kurth – Hero (live demo)**
- **15:00 – 15:30 Michael Schaffner – Ariane + Open Piton (intro)**
  - 15:30 – 16:00 Break
- **16:00 – 16:30 Michael Schaffner – Ariane + Open Piton (live demo)**
- **16:30 – 17:30 Q&A, demos**

# RISC-V is ‘just’ the processor core, it works in a system!

- **RISC-V describes an open ISA**
  - Implementations of RISC-V can be both open and closed source
  - There are many implementation options (32/64, FP, Vector, Atomics..)
  - Implementations can be in different languages (Chisel/VHDL/Verilog)...
  - ... and target different platforms (FPGA/ASIC)
- **The PULP team (ETH Zürich and University of Bologna)**
  - Are founding members of the RISC-V foundation
  - Develop **optimized implementations of RISC-V** cores in SystemVerilog
  - Provide the cores using a **permissive open source** license on GitHub
- **The core alone does not do much, it needs a system to work**
  - PULP project provides several **open source platforms** built on RISC-V cores
  - ... and a number of open source peripherals to build these systems

# 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**

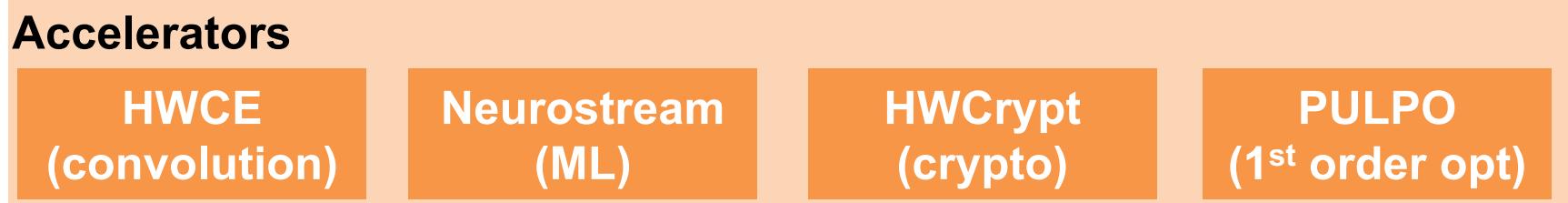
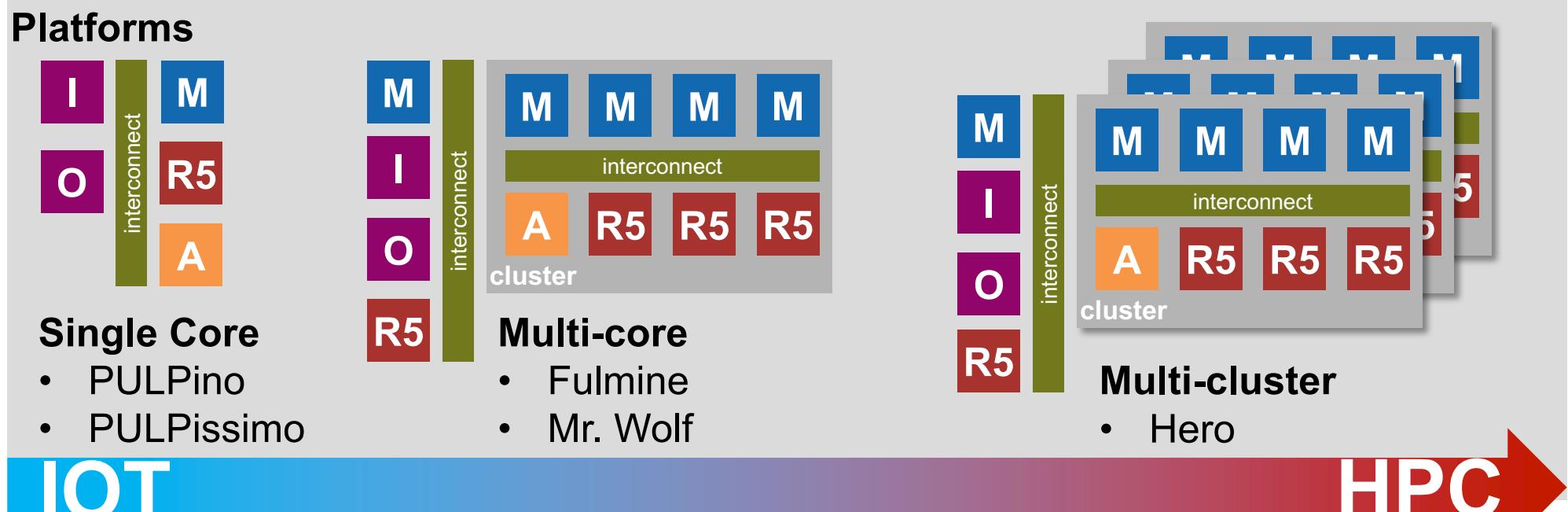
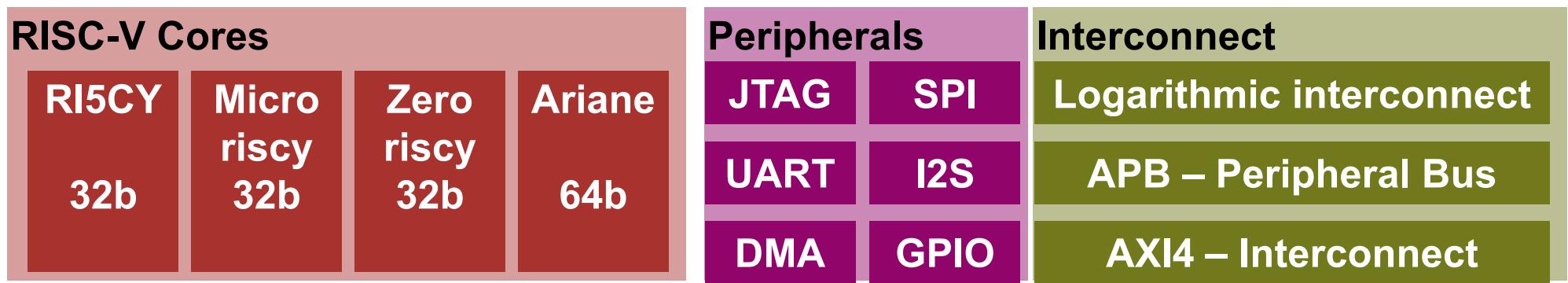
**64b**

# 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 <sup>st</sup> order opt)

# All these components are combined into platforms



# Today we will discuss three HPC applications of PULP

- **Power PULP** (Fabian Schuiki)
  - Developed as part of H2020 project OPRECOMP
  - Allows clusters of RISC-V cores to be connected to a Power8 system over CAPI
- **HERO** (Andreas Kurth)
  - Open Heterogeneous Research Platform
  - Connects a PULP based cluster on FPGA with an ARM core running Linux
- **Ariane + OpenPiton** (Michael Schaffner)
  - Ariane is our 64bit RISC-V processor
  - OpenPiton was developed by Princeton Parallel Group (using Sparc Cores)
  - The recent release of OpenPiton (and Ariane) allow them to work together

**Everything we talk about today is available as open source**



# One SDK, many ways to work with PULP based systems

- **To get applications running we need quite a bit of help**
  - Cross compilation, sometimes for multiple platforms (RISC-V, ARM..)
  - Download the design
  - Run and collect the output
- **The PULP SDK has been developed to help in this regard**
  - Andreas will talk more about this after the break
- **The SDK allows a common interface to run your applications on:**
  - a virtual platform running on your own system (**Power PULP, Hands on**)
  - an RTL simulator (**Ariane + OpenPiton, Live demo**)
  - an FPGA implementation target (**Hero, Live demo**)
  - the actual ASIC that contains your design (**demonstrations at the end of session**)

# Parallel Ultra Low Power (PULP)

- Project started in **2013** by Luca Benini
- A collaboration between University of Bologna and ETH Zürich
  - Large team. In total we are about 60 people, not all are working on PULP
- 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.

# 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

# Extensions still being worked on by RISC-V foundation

- No or partial work done yet on those extensions
- Possible to contribute as a foundation member in **task-groups**
- Dedicated task-groups
  - Formal specification
  - Memory Model
  - Marketing
  - External Debug Specification
- For Bit-manipulation we provide our own solution → part of the task group

**Q** Quad-Precision Floating-Point

**L** Decimal Floating-Point  
(IEEE 754-2008)

**B** Bit-Manipulation

**T** Transactional Memory

**P** Packed-SIMD

**J** Dynamically Translated Languages

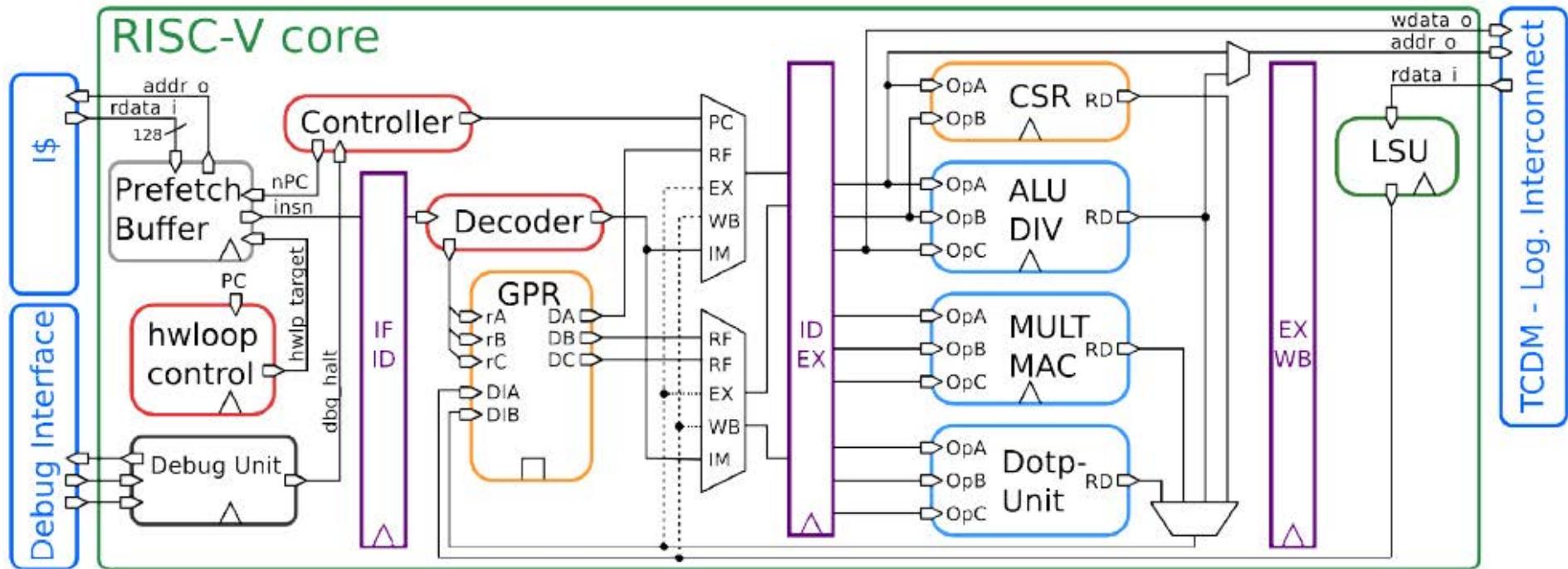
**V** Vector Operations

**N** User-Level Interrupts

# Our RISC-V family explained

32 bit				64 bit
Low Cost Core	Core with DSP enhancements	Floating-point capable Core	Linux capable Core	
<ul style="list-style-type: none"><li>■ Zero-riscy</li><li>■ RV32-ICM</li><li>■ Micro-riscy</li><li>■ RV32-CE</li></ul>	<ul style="list-style-type: none"><li>■ RI5CY</li><li>■ RV32-ICMX<ul style="list-style-type: none"><li>■ SIMD</li><li>■ HW loops</li><li>■ Bit manipulation</li><li>■ Fixed point</li></ul></li></ul>	<ul style="list-style-type: none"><li>■ RI5CY+FPU</li><li>■ RV32-ICMFX</li></ul>	<ul style="list-style-type: none"><li>■ Ariane</li><li>■ RV64-IMAFDCX</li><li>■ Full privilege specification</li></ul>	
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!)  
    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!)
```

# It is possible to enhance RISC-V with custom extensions

**RISC-V:** “To support development of proprietary custom extensions, portions of the encoding space are guaranteed to never be used by standard extensions.”

- **The ‘X’ extension can be used by everyone freely**
  - Offers great flexibility
  - Of course these custom extensions are not automatically supported by tools
  - You have to add patches / new tools so that these can be utilized
- **Even if your tools do not support extensions, the cores will work**
  - The tools will just not generate code that takes advantage of the extensions
  - But the cores with extensions will remain compatible to standard RISC-V
- **The goal is to work so that ‘good’ extensions become standard**
  - Requires being active in the RISC-V foundation task groups
  - ETH Zürich is actively involved in V, P and B at the moment

# Our extensions to RI5CY (with additions to GCC)

- Post-incrementing load/store instructions
- Hardware Loops (lp.start, lp.end, lp.count)
- ALU instructions
  - Bit manipulation (count, set, clear, leading bit detection)
  - Fused operations: (add/sub-shift)
  - Immediate branch instructions
- Multiply Accumulate (32x32 bit and 16x16 bit)
- SIMD instructions (2x16 bit or 4x8 bit) with scalar replication option
  - add, min/max, dotproduct, shuffle, pack (copy), vector comparison

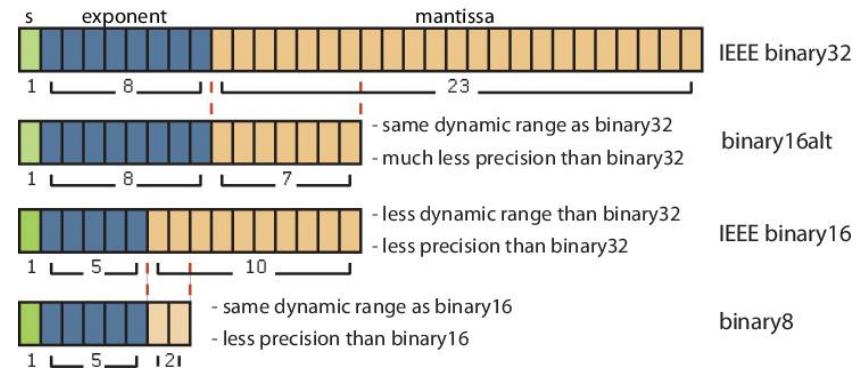
For 8-bit values the following can be executed in a single cycle (`pv.dotup.b`)

$$Z = D_1 \times K_1 + D_2 \times K_2 + D_3 \times K_3 + D_4 \times K_4$$

# What About Floating Point Support?

- F (single precision) and D (double precision) extension in RISC-V
- Uses separate floating point register file
  - specialized float loads (also compressed)
  - float moves from/to integer register file
- Fully IEEE compliant
- RI5CY support for F
- Ariane for F and D
  - Pipelined for Fused Multiply Add (FMA):
  - FMA 64: 4 stages
  - FMA 32: 3 stages
  - FMA 16: 2 stages
  - Iterative for divisions and square roots

## Alternative FP format support



## Unified FP/Integer register file

- Not standard
- up to 15 % better performance
  - Re-use integer load/stores (post incrementing Id/st)
  - Less area overhead
  - Useful if pressure on register file is not very high (true for a lot of applications)

# Memory Protection (PMU and MMU)

## Physical Memory Protection (PMP)

- Protect the physical memory when the core runs in U or S privilege level
- Up to 16 entries for address filtering
- Configuration held in 4 CSRs  
**pmpcfg[0-3]**
- Whether Store (W), Load (R) and Fetch (X) is allowed
- Address matching modes:
  - Naturally aligned power-of-2 regions (NAPOT) or aligned 4 Byte (NA4)
  - Boundaries >, < (TOR)
- Implemented in **RISCV**

## Supervisor Memory Translation and Protection (for Linux-like systems)

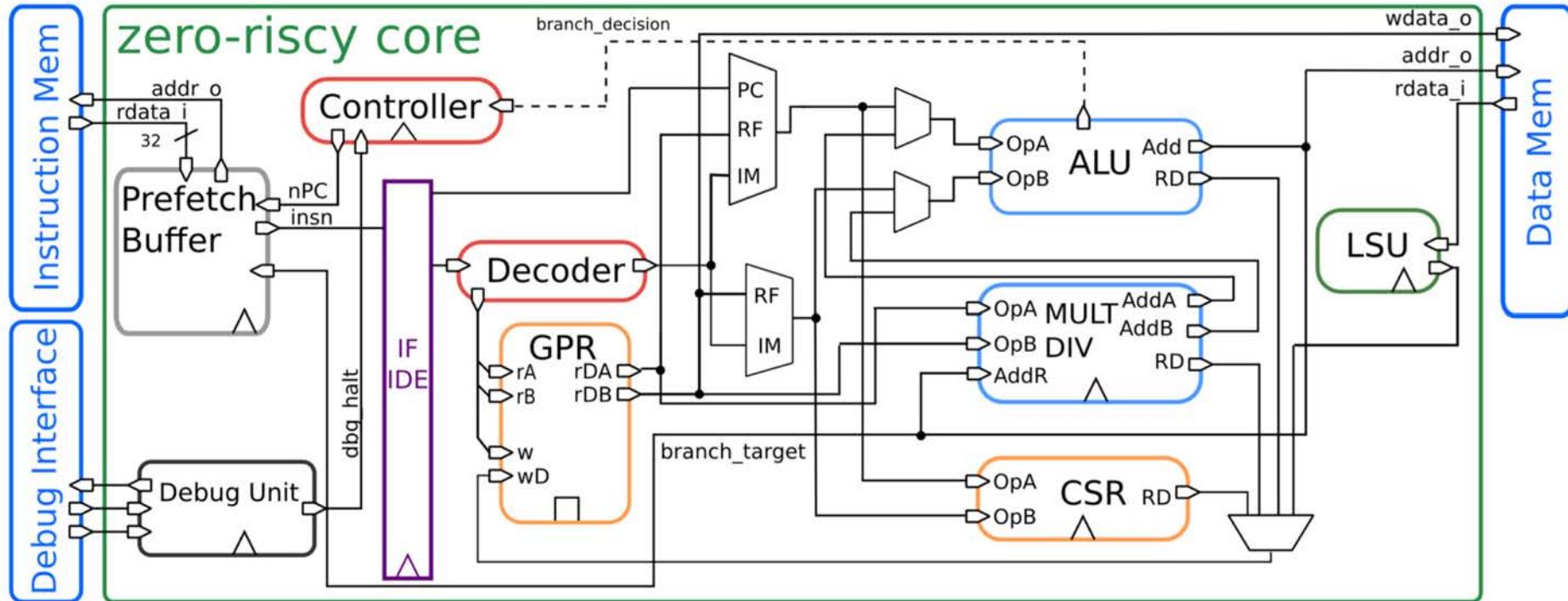
- Effectively needs TLBs
- Register to configure base page number (**satp**)
- Translation Mode (32, 39, 48 virtual addressing)
- Address Space Identifier (**ASID**)
- Implemented in **Ariane**



# Why we designed other 32-bit cores after RI5CY?

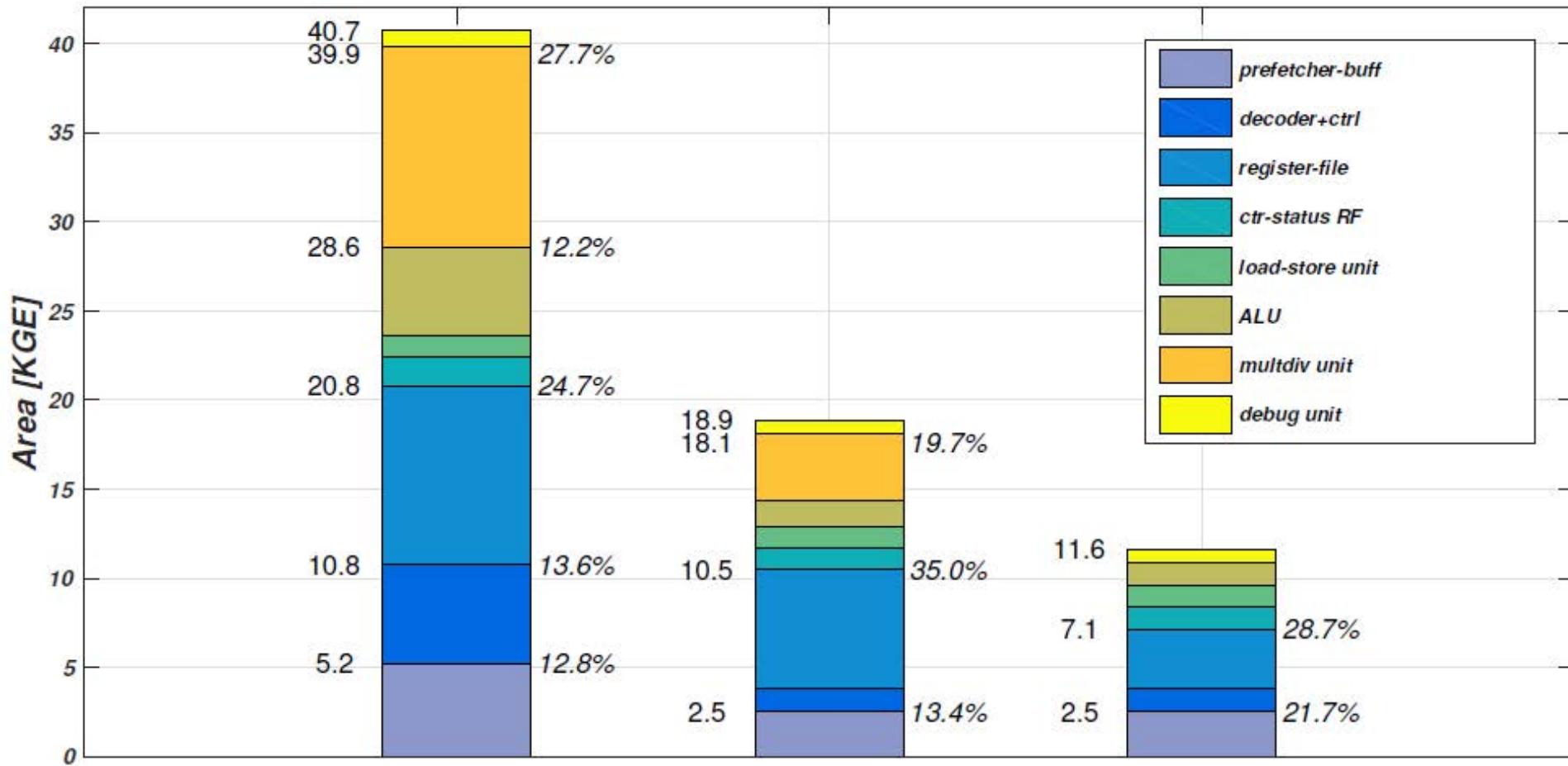
- RI5CY was built for **energy efficiency** for DSP applications
  - Ideally all parts of the core are running all the time doing something useful
  - This does not always mean it is low-power
  - The core is rather large (> 40 kGE without FPU)
- People asked us about a **simple and small core**
  - 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

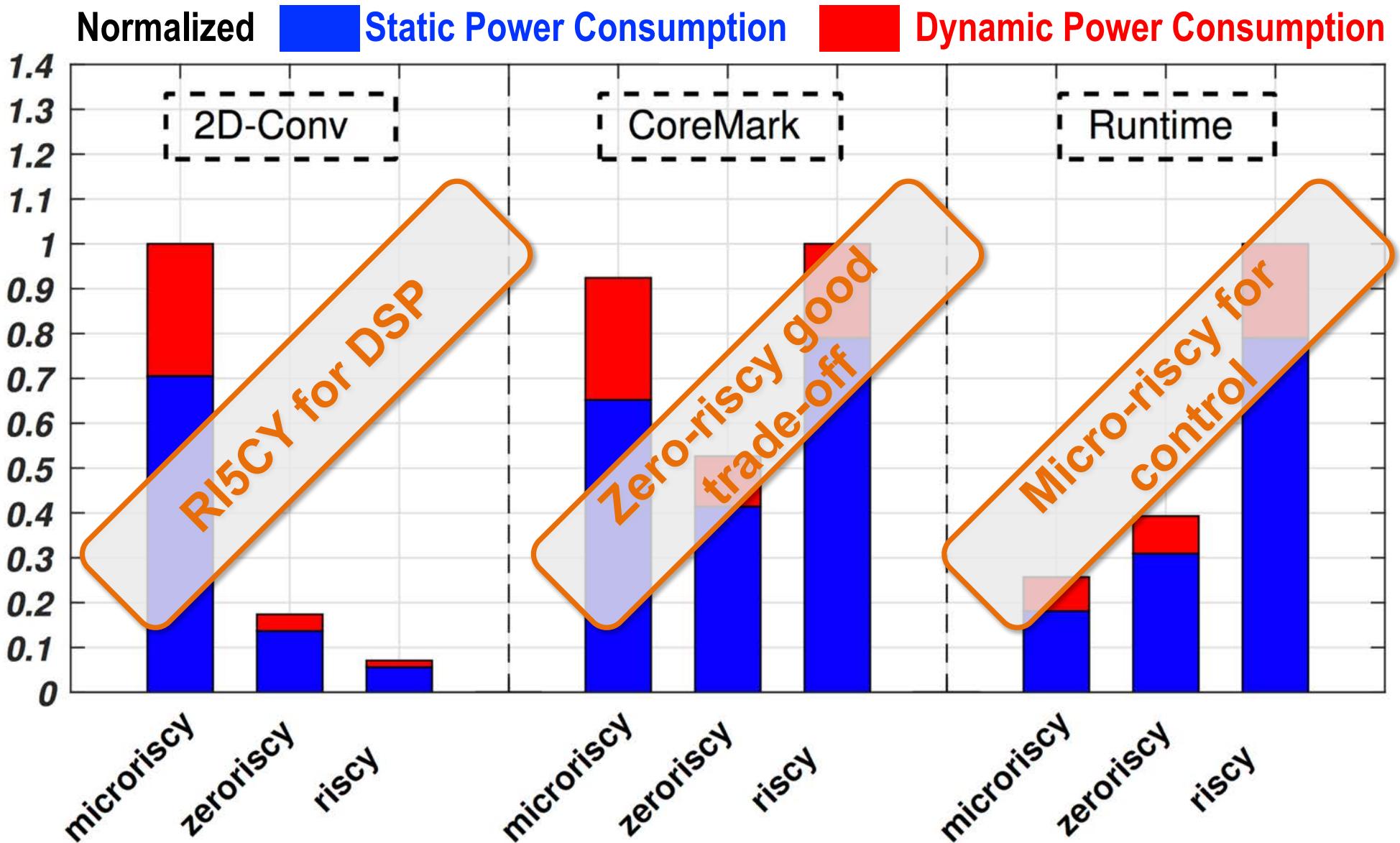
# Different 32-bit cores with different area requirements



RI5CY

Zero-riscy Micro-riscy

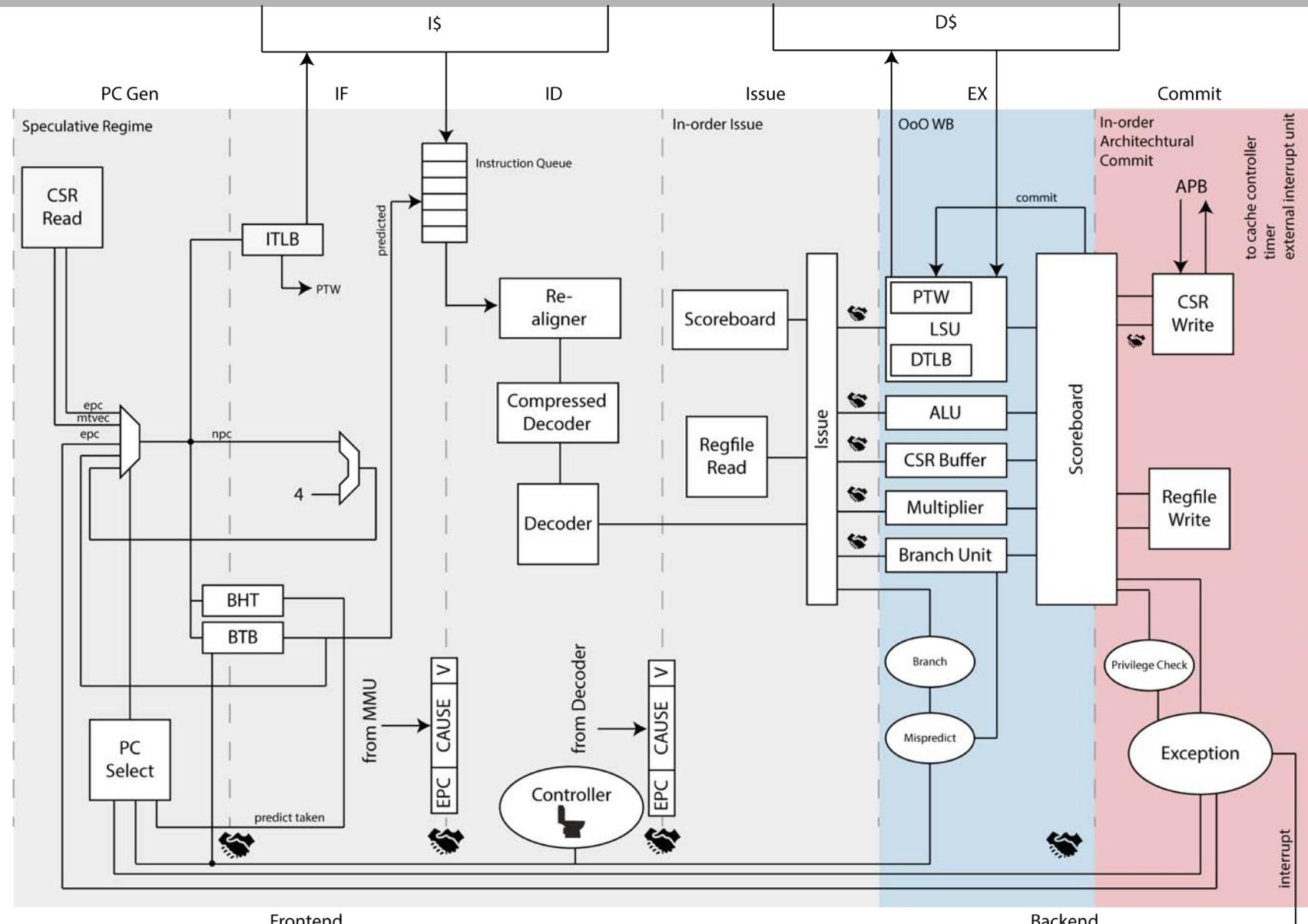
# Different cores for different types of workload



# Finally the step into 64-bit cores

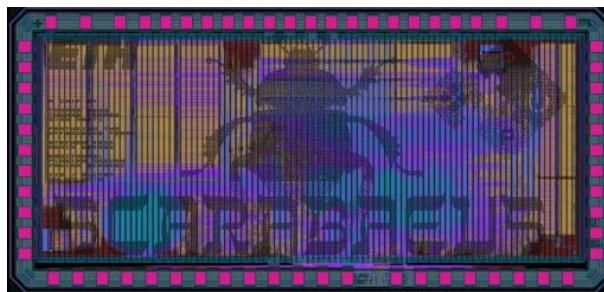
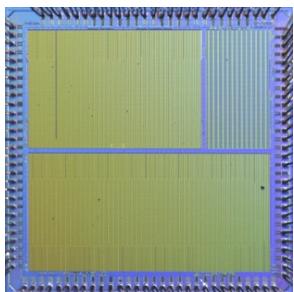
- For the first 4 years of the PULP project we used only 32bit cores
  - Luca once famously said “*We will never build a 64bit core*”.
  - Most IoT applications work well with 32bit cores.
  - A typical 64bit core is much more than 2x the size of a 32bit core.
- **But times change:**
  - Using a 64bit 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

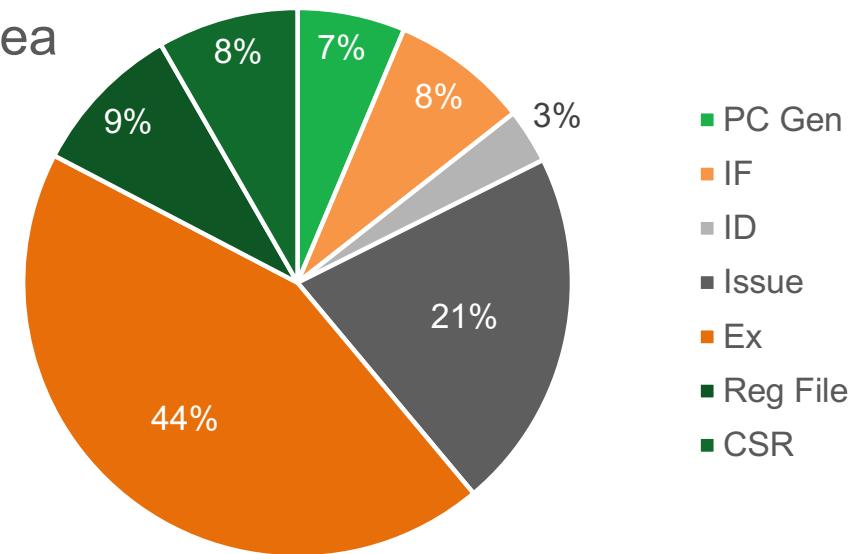


# Main properties of Ariane

- 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



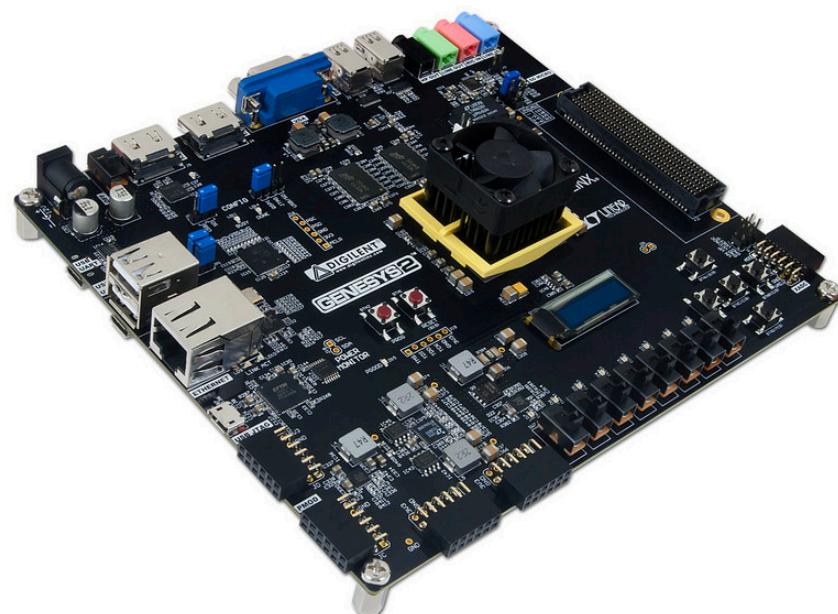
Area



# Ariane mapped to FPGA boots Linux

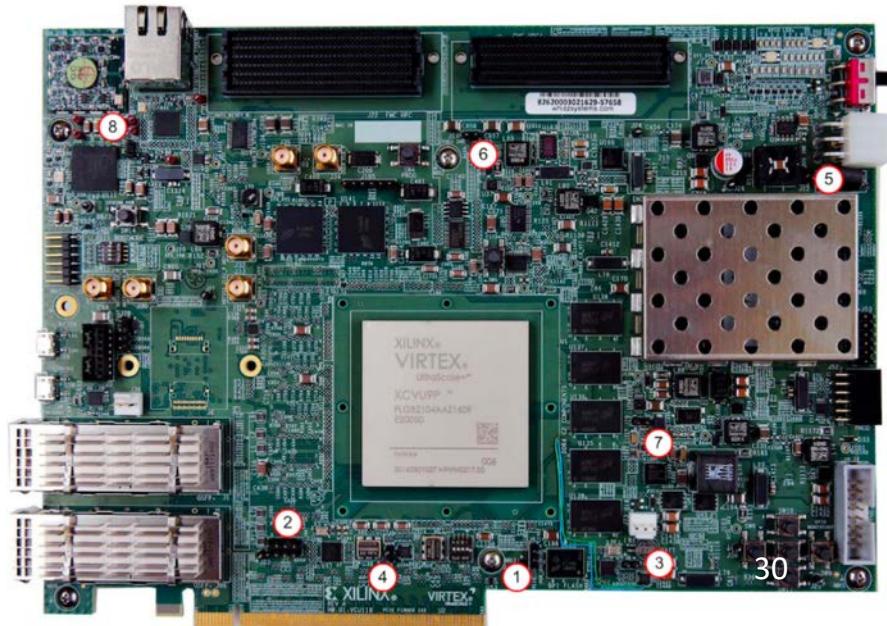
# Digilent Genesys 2

- **Core:** 50 MHz
  - **Core:** 50 kLUTs (20%)
  - 1 GiB DDR3

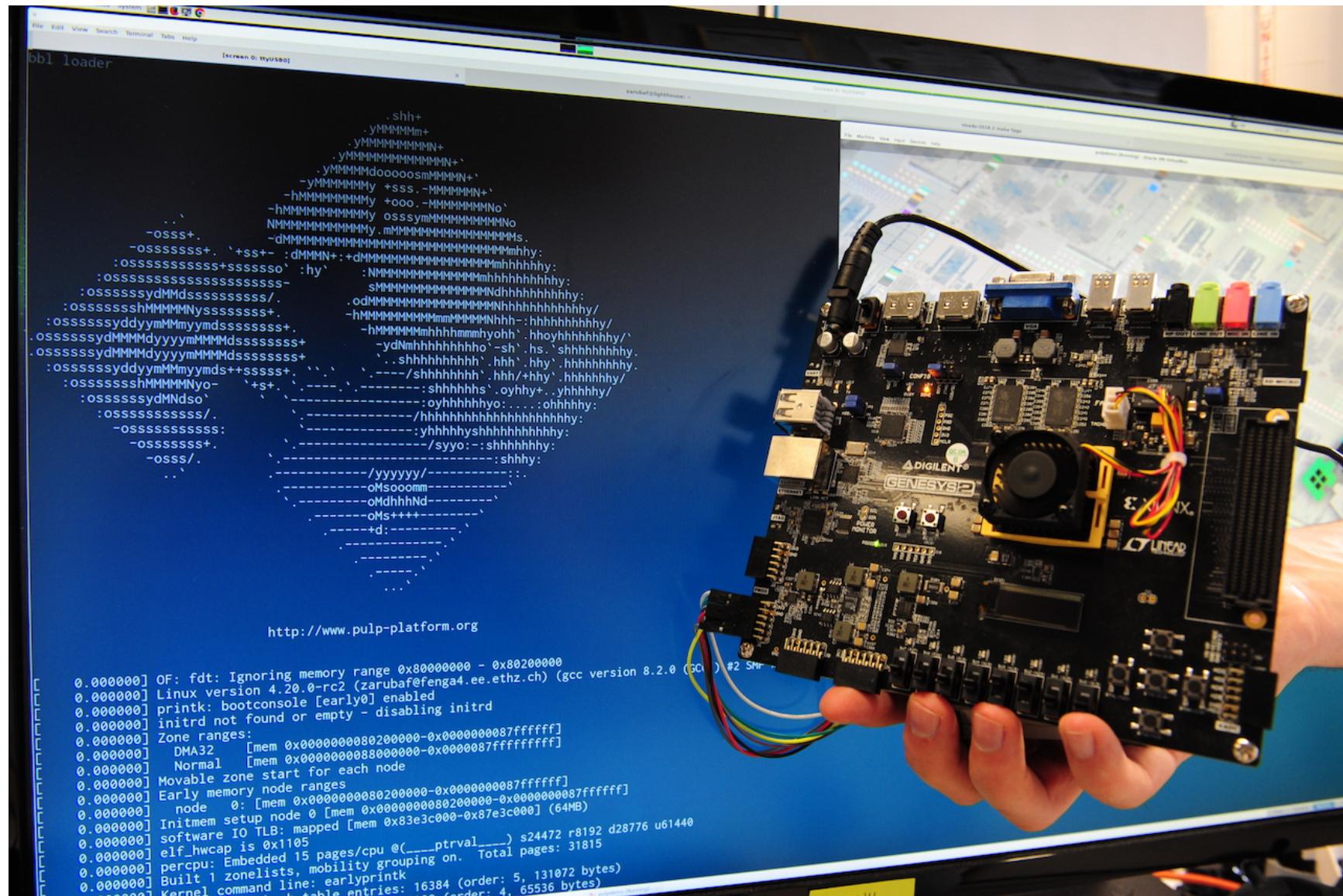


# Xilinx VCU 118

- **Core:** 100 MHz
  - **Core:** 50 kLUTs (3%)
  - 32 GiB DDR4
  - PCIe Interface



# Ariane booting Linux on a Digilent Genesys 2 board



# 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

## Platforms



## Single Core

- PULPino
- PULPissimo

## Accelerators

HWCE  
(convolution)

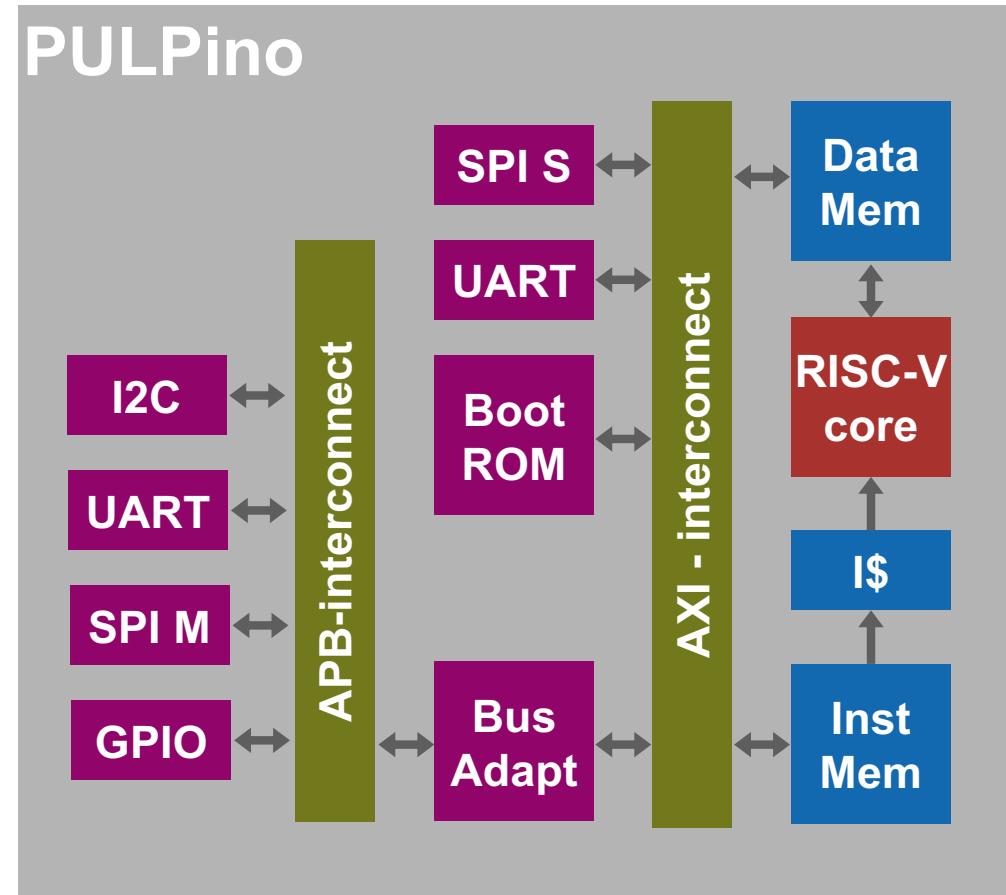
Neurostream  
(ML)

HWCrypt  
(crypto)

PULPO  
(1<sup>st</sup> order opt)

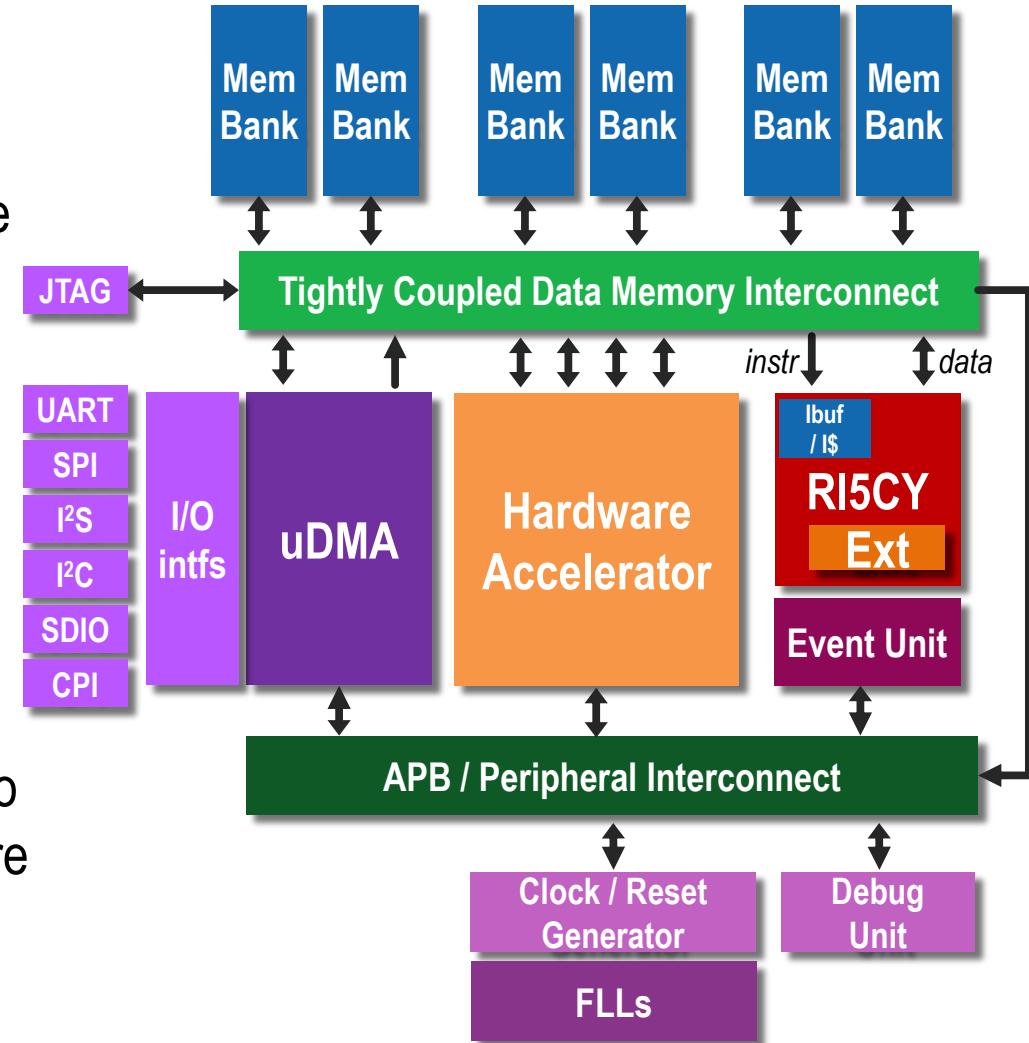
# PULPino our first single core platform

- **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



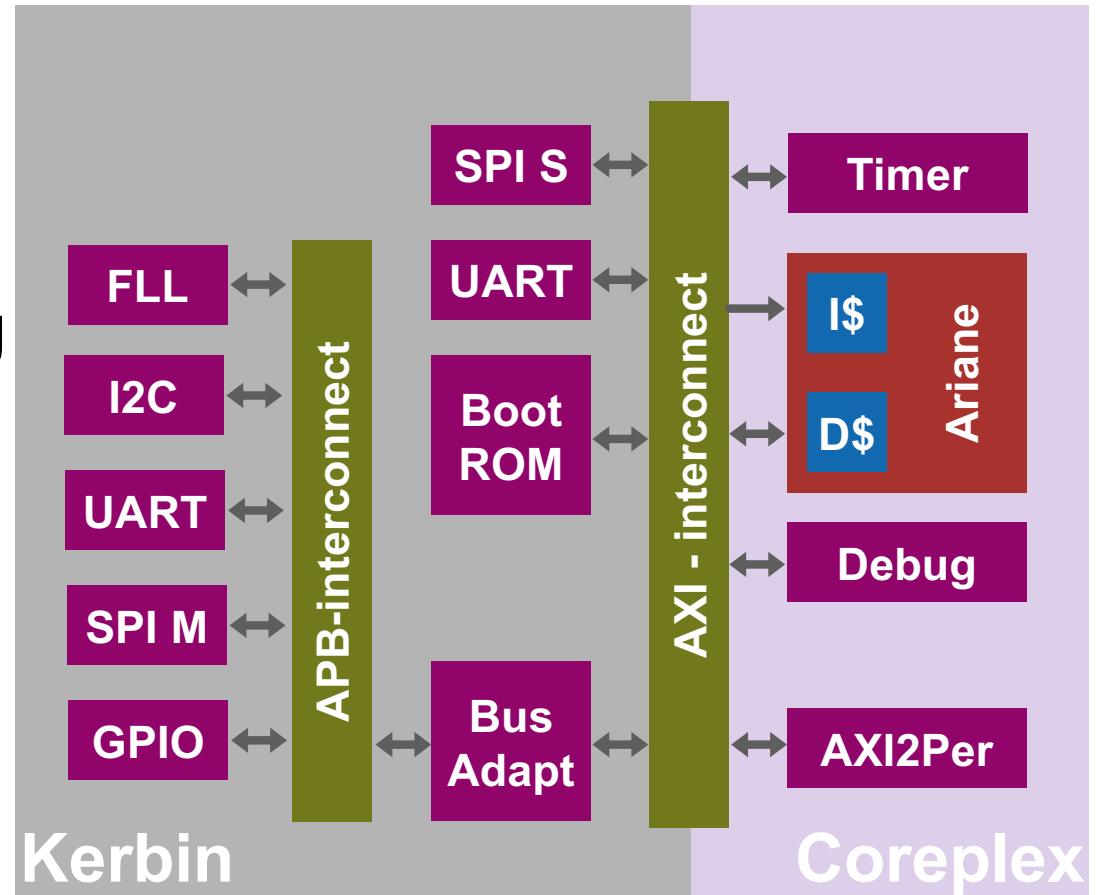
# PULPiSSimo the improved single core platform

- **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**



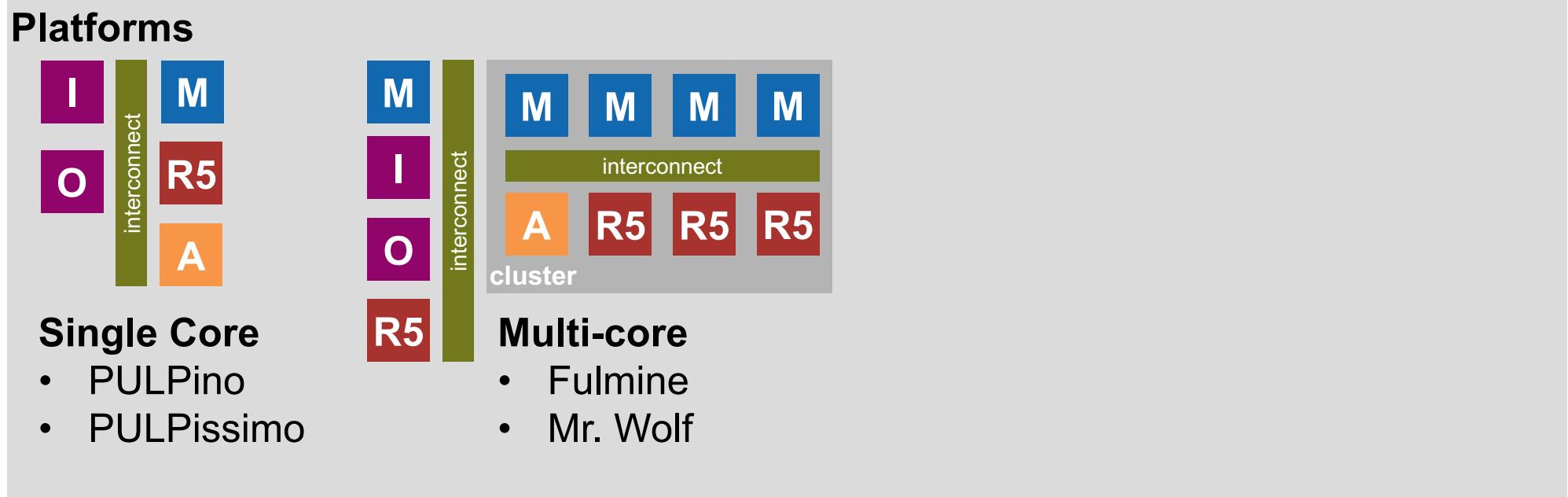
# Kerbin the single core support structure for Ariane

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



# The main PULP systems we develop are cluster based

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

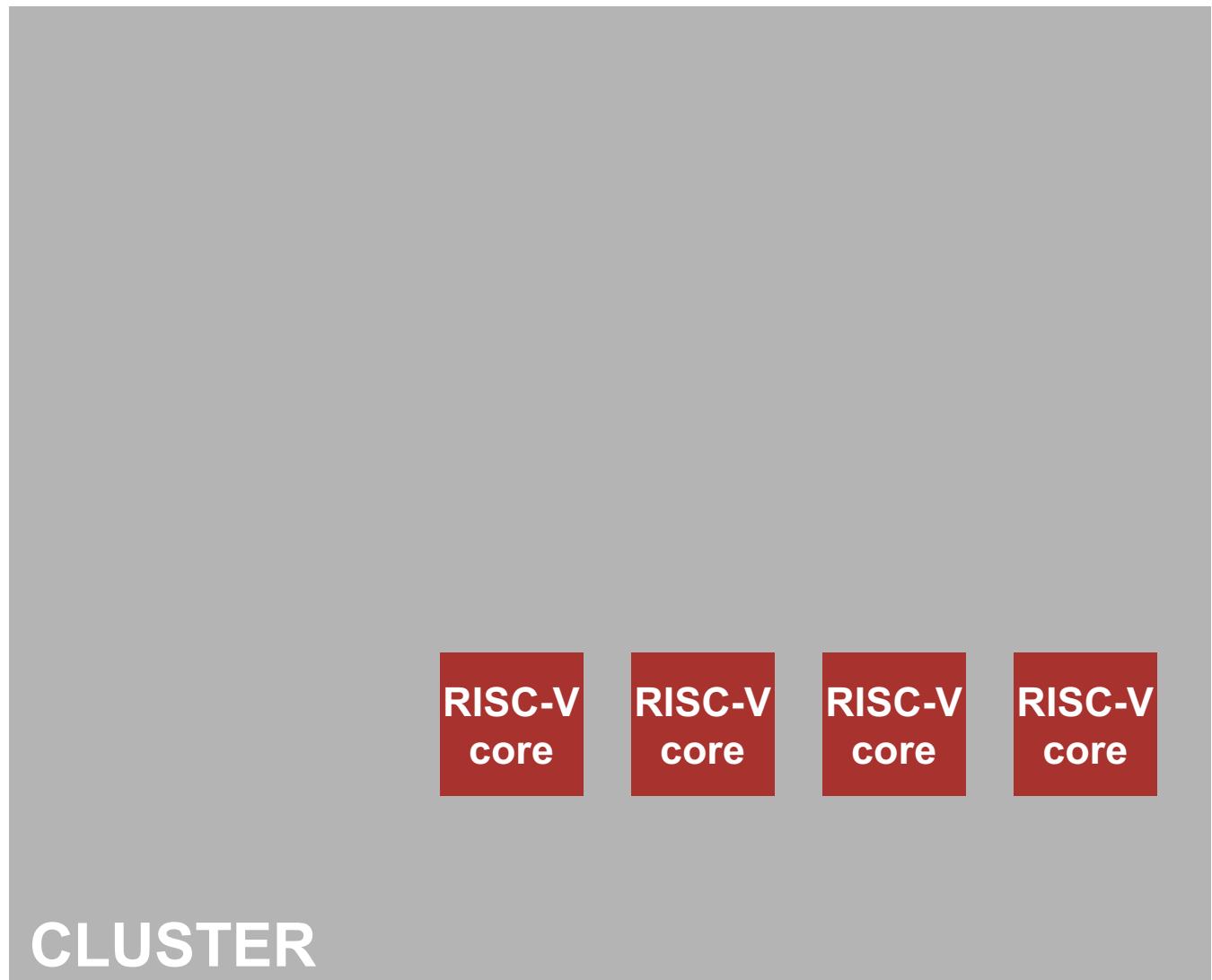


Accelerators	HWCE (convolution)	Neurostream (ML)	HWCrypt (crypto)	PULPO (1 <sup>st</sup> order opt)

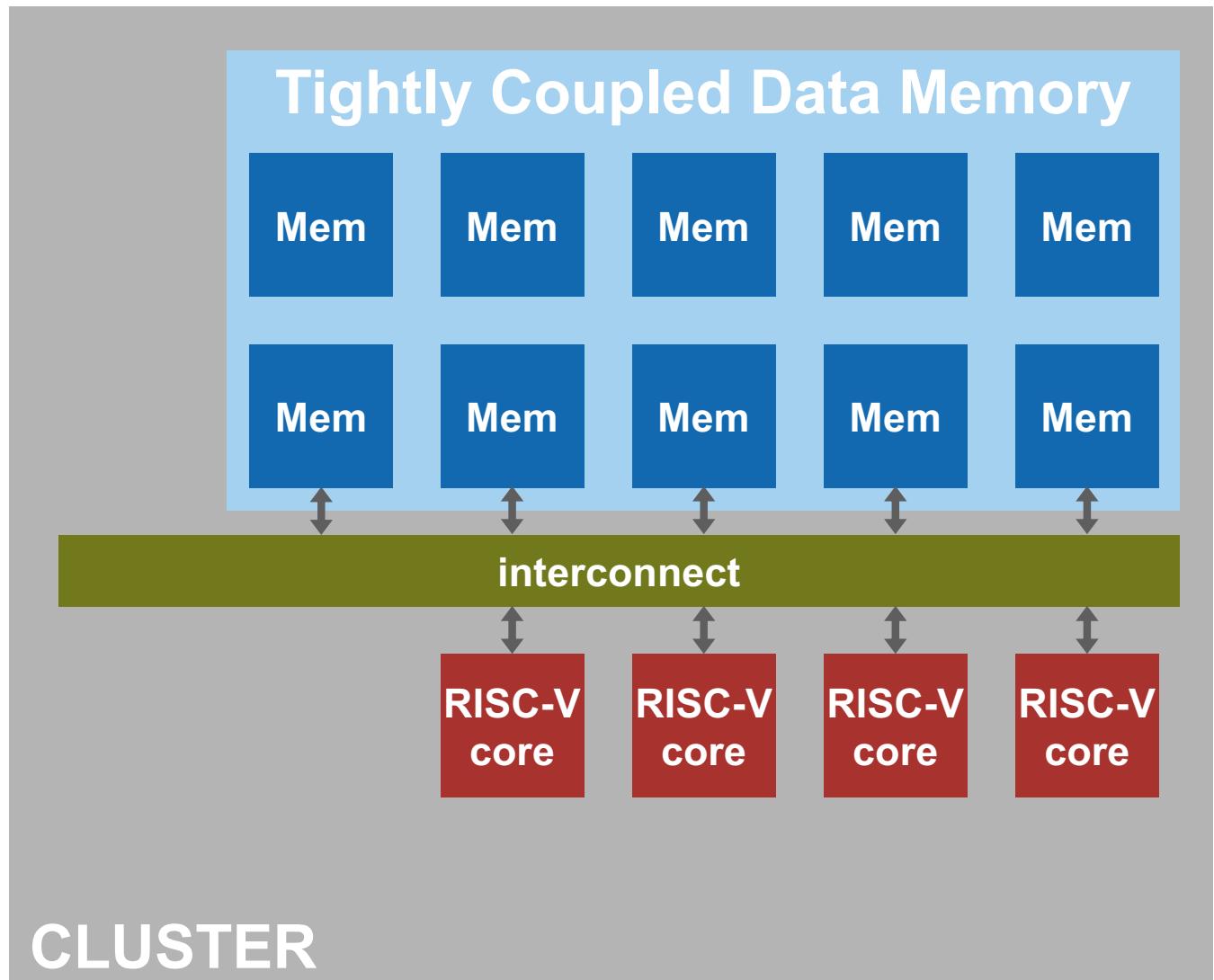
# The main components of a PULP cluster

- **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.

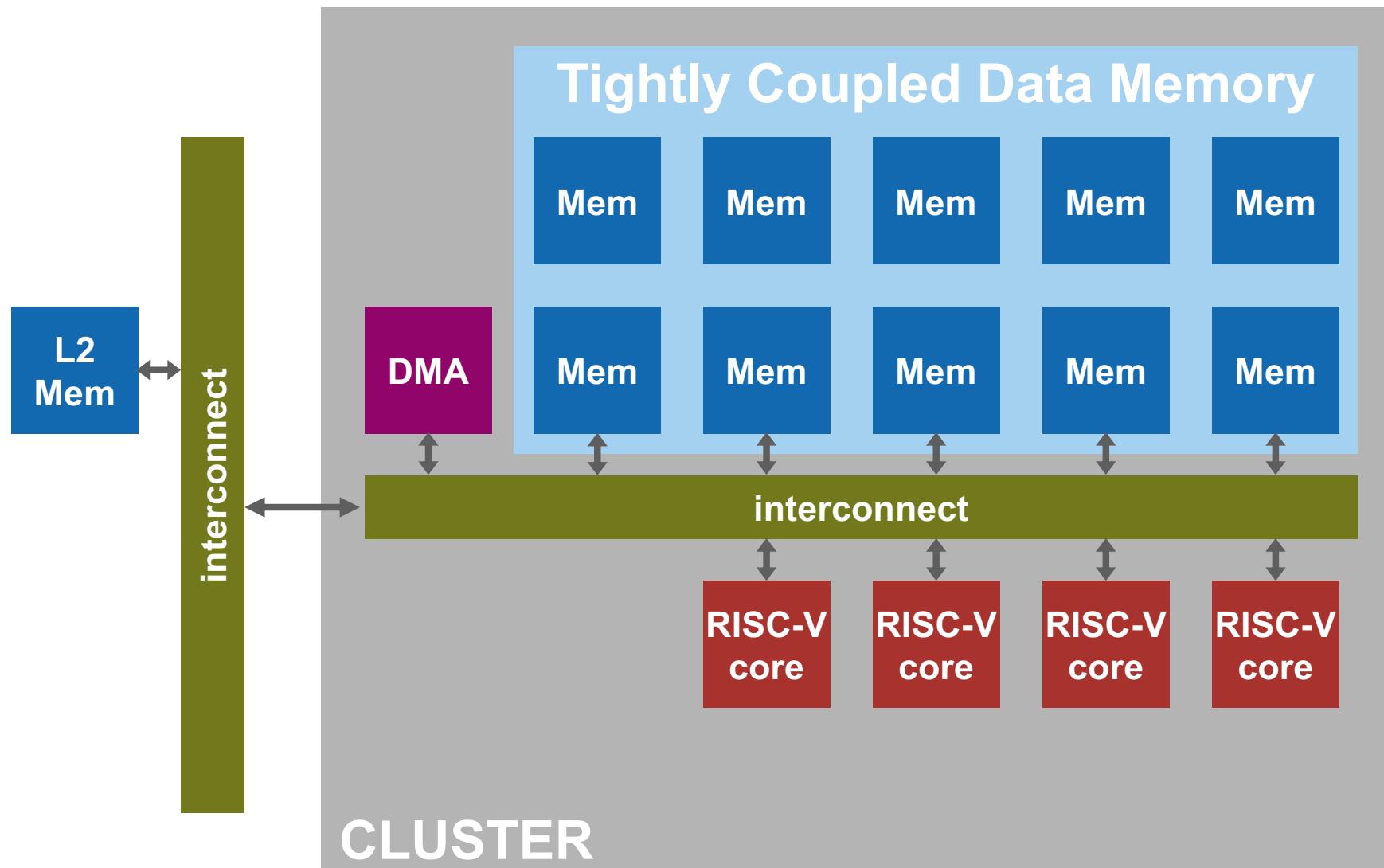
# PULP cluster contains multiple RISC-V cores



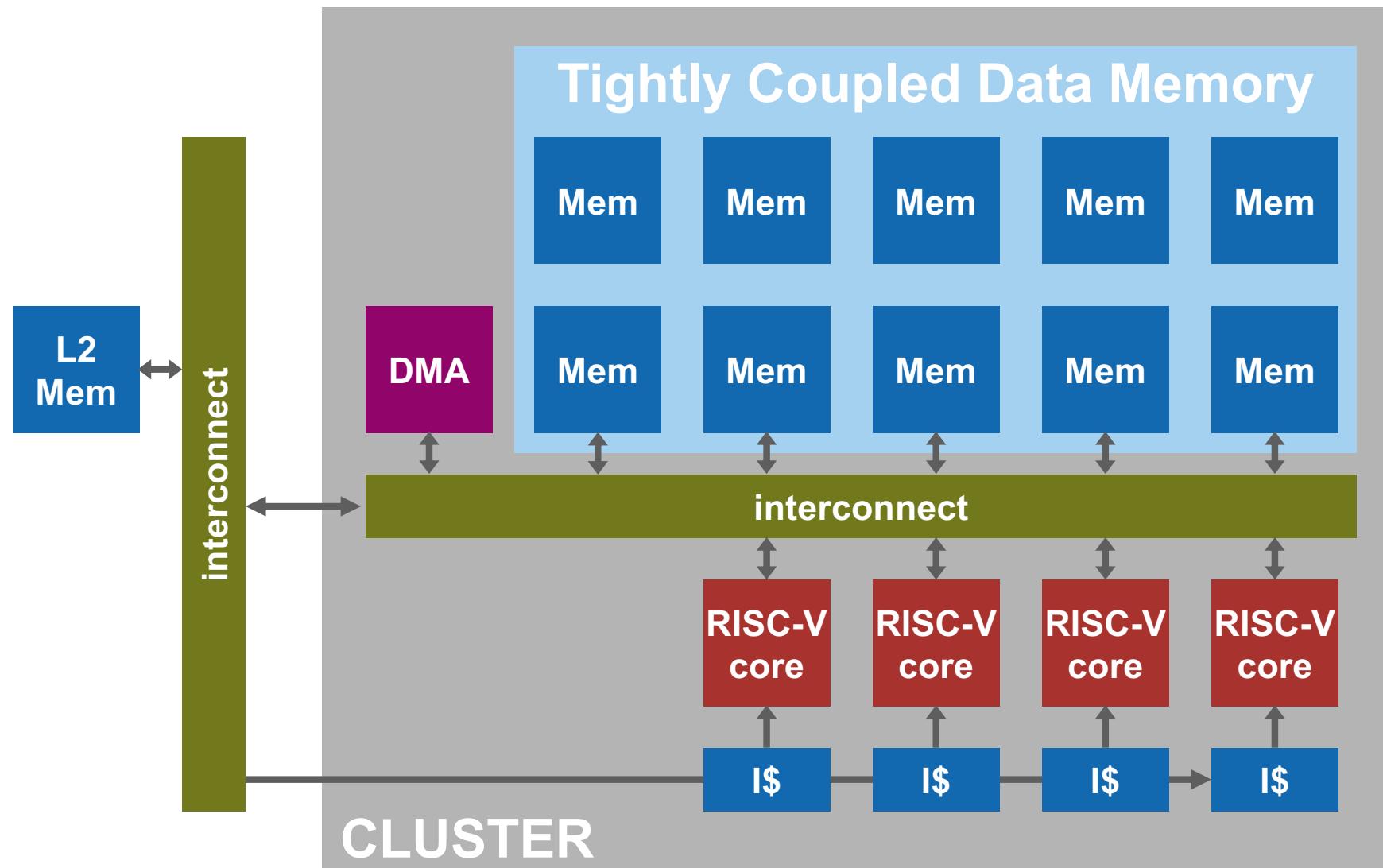
# All cores can access all memory banks in the cluster



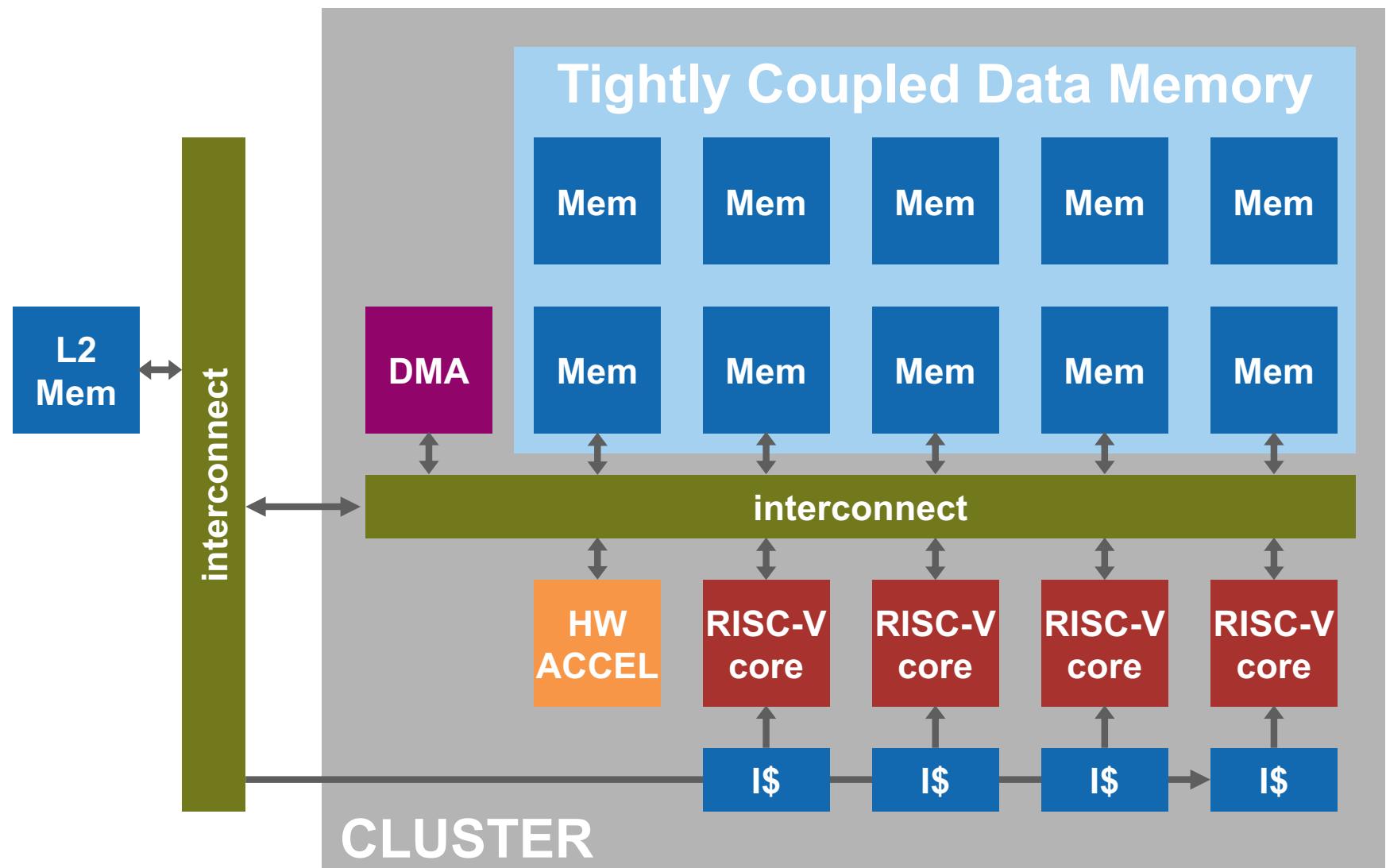
# Data is copied from a higher level through DMA



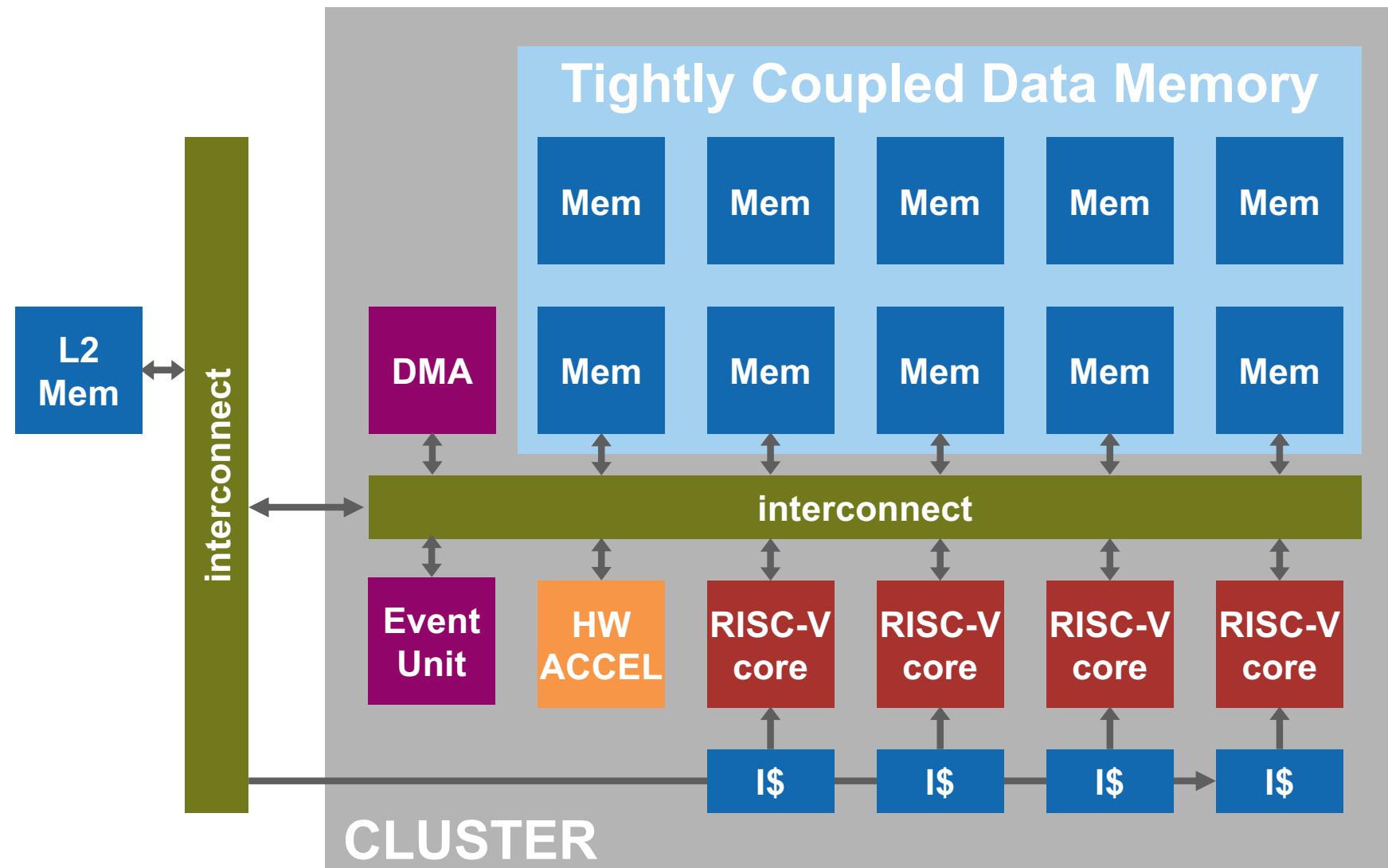
There is a (shared) instruction cache that fetches from L2



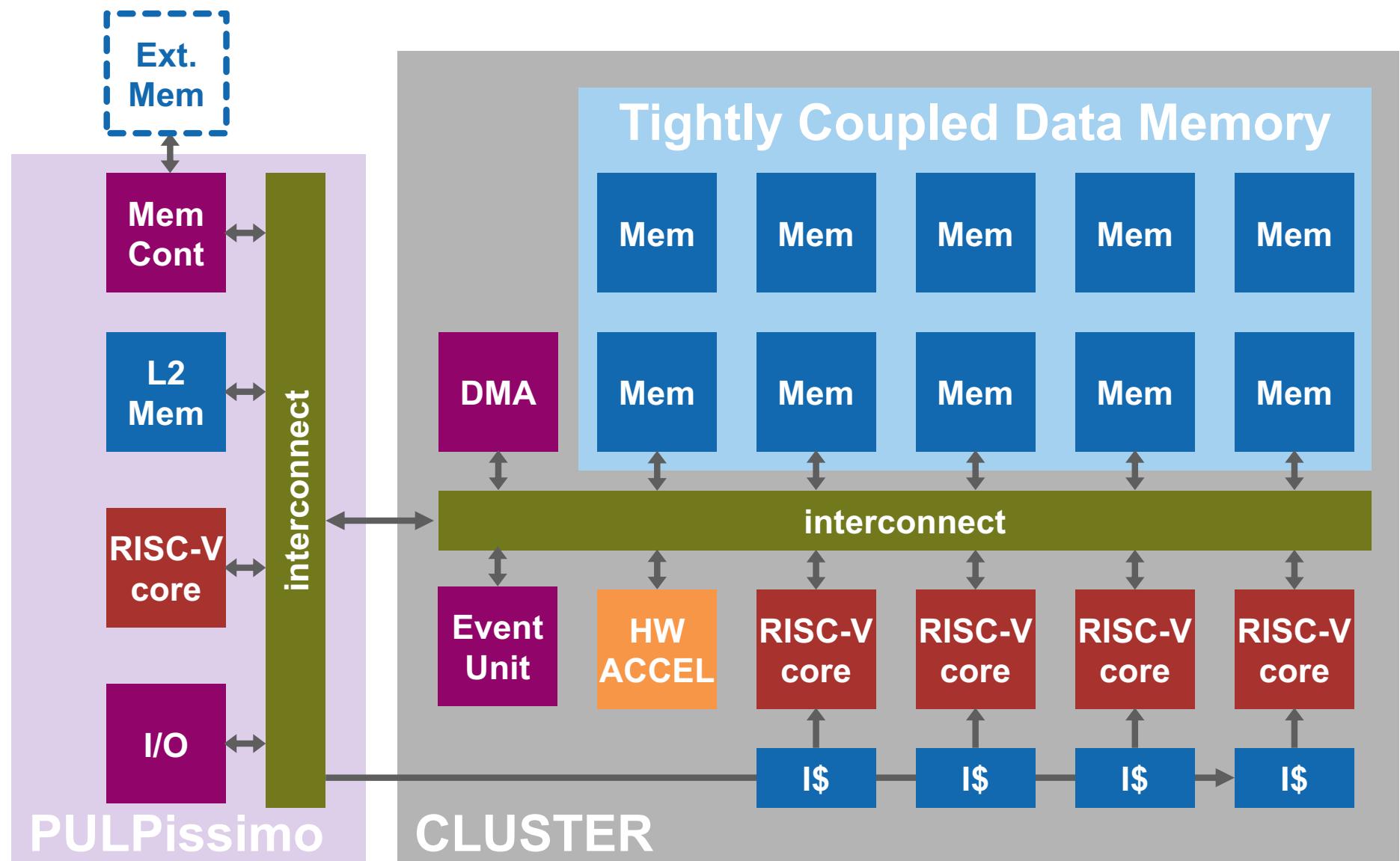
# Hardware Accelerators can be added to the cluster



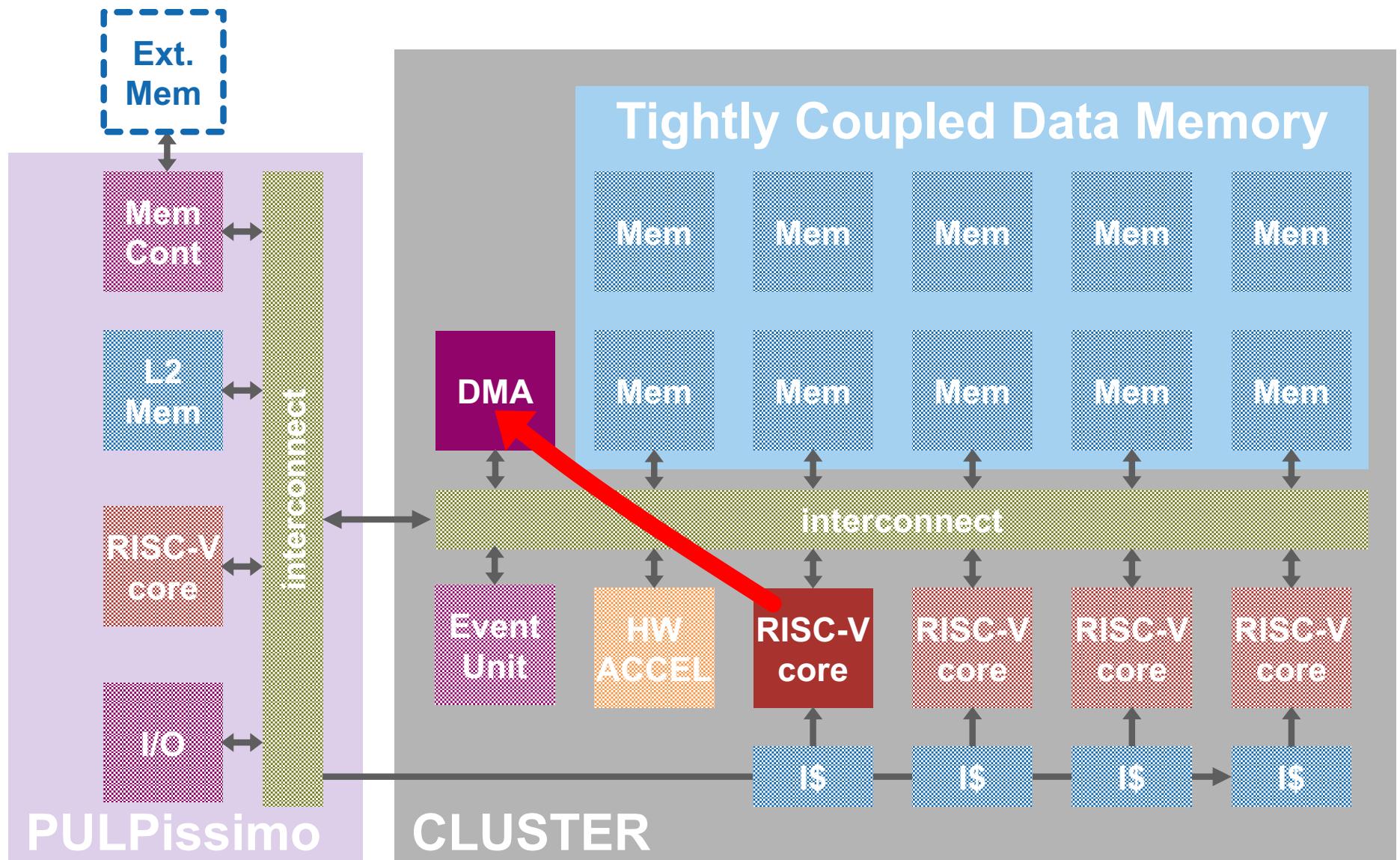
# Event unit to manage resources (fast sleep/wakeup)



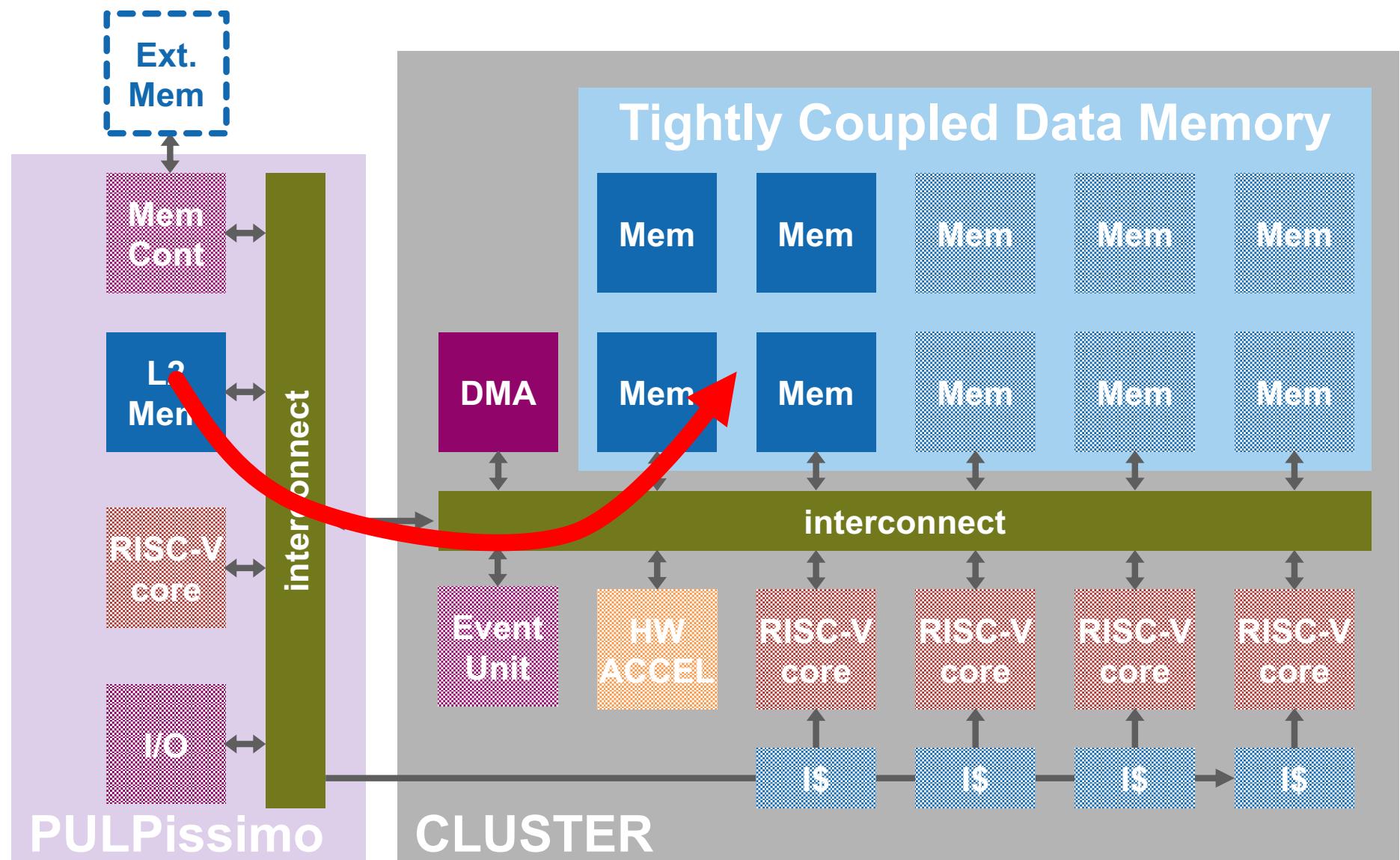
# An additional microcontroller system (PULPissimo) for I/O



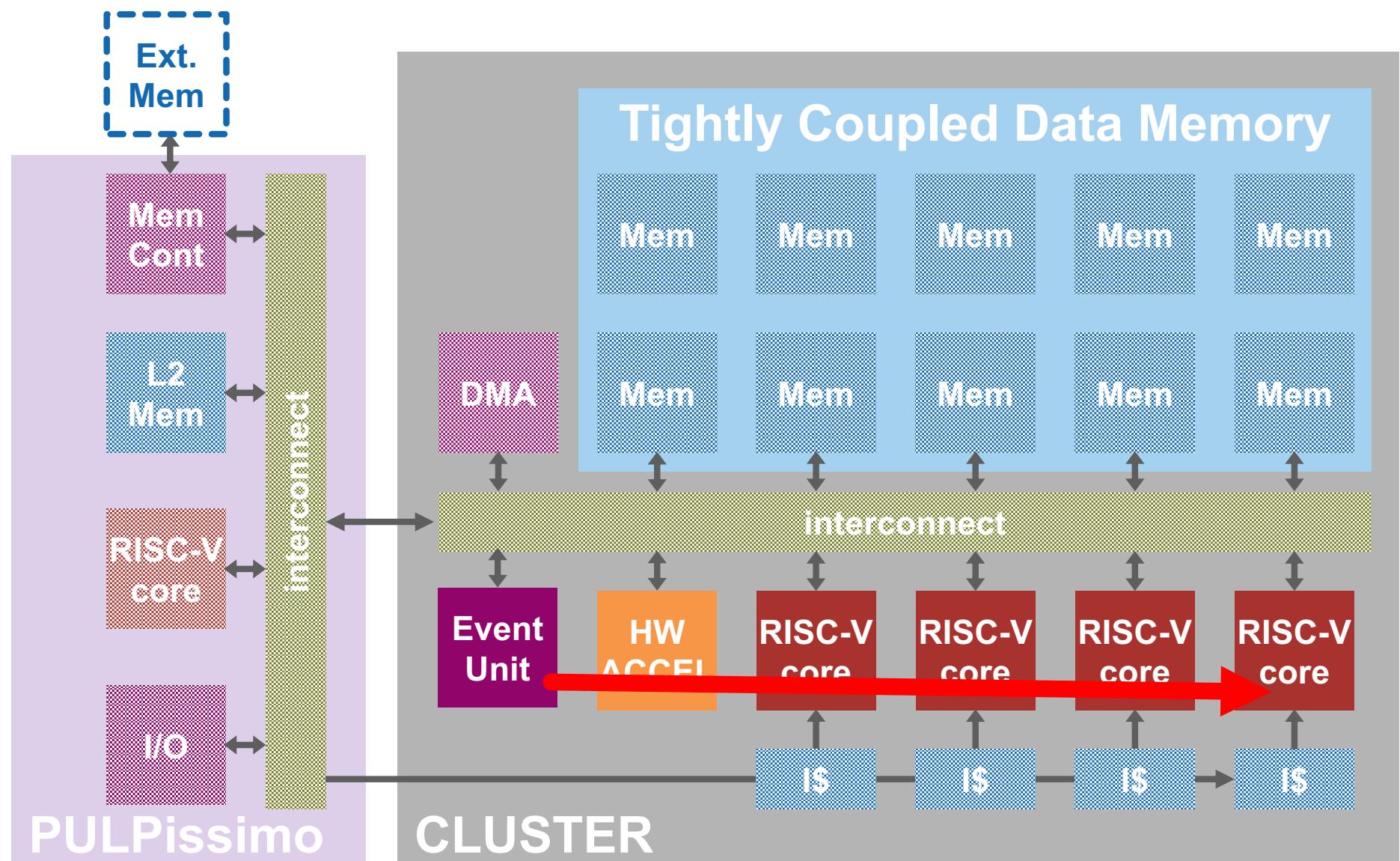
# How do we work: Initiate a DMA transfer



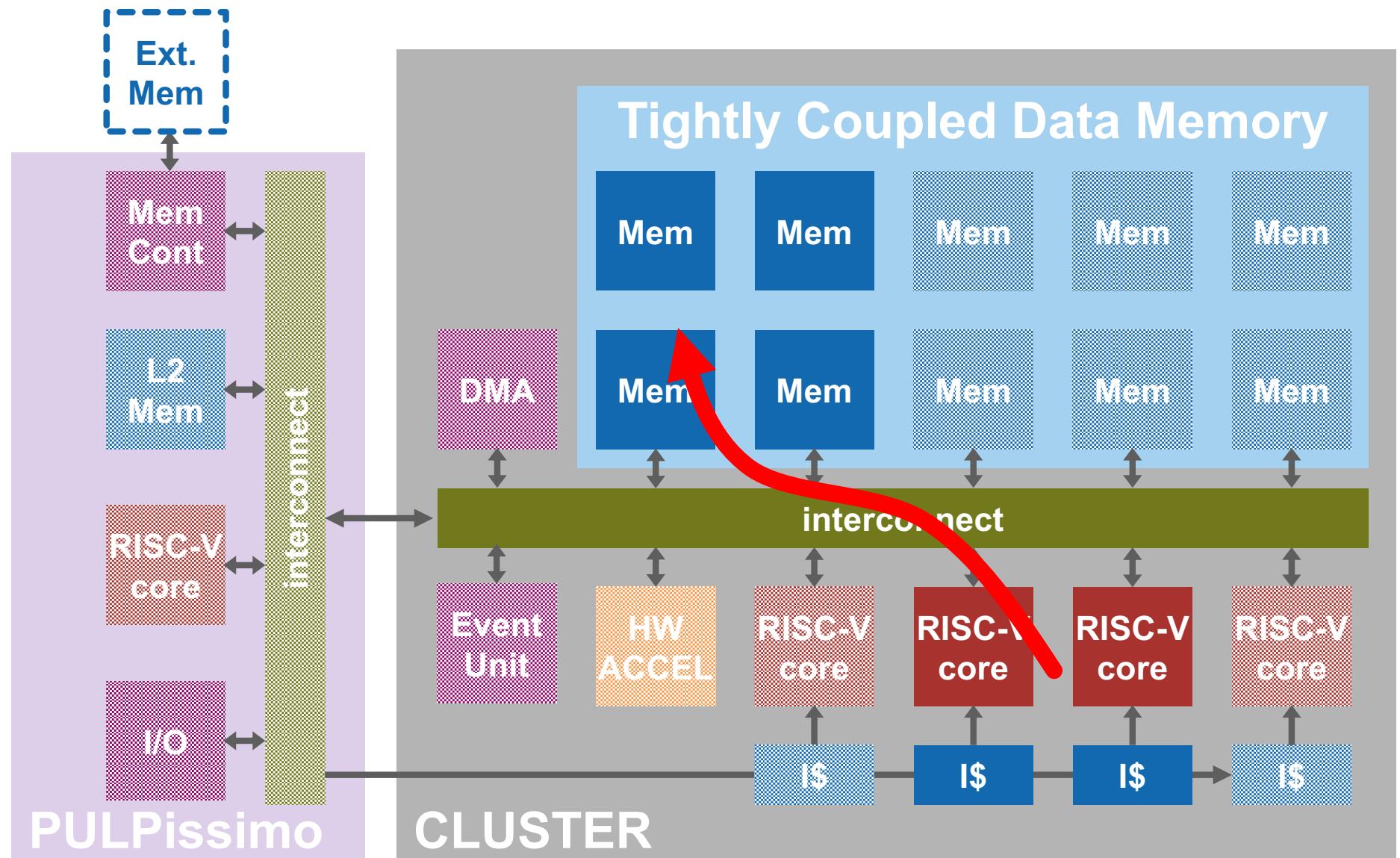
# Data copied from L2 into TCDM



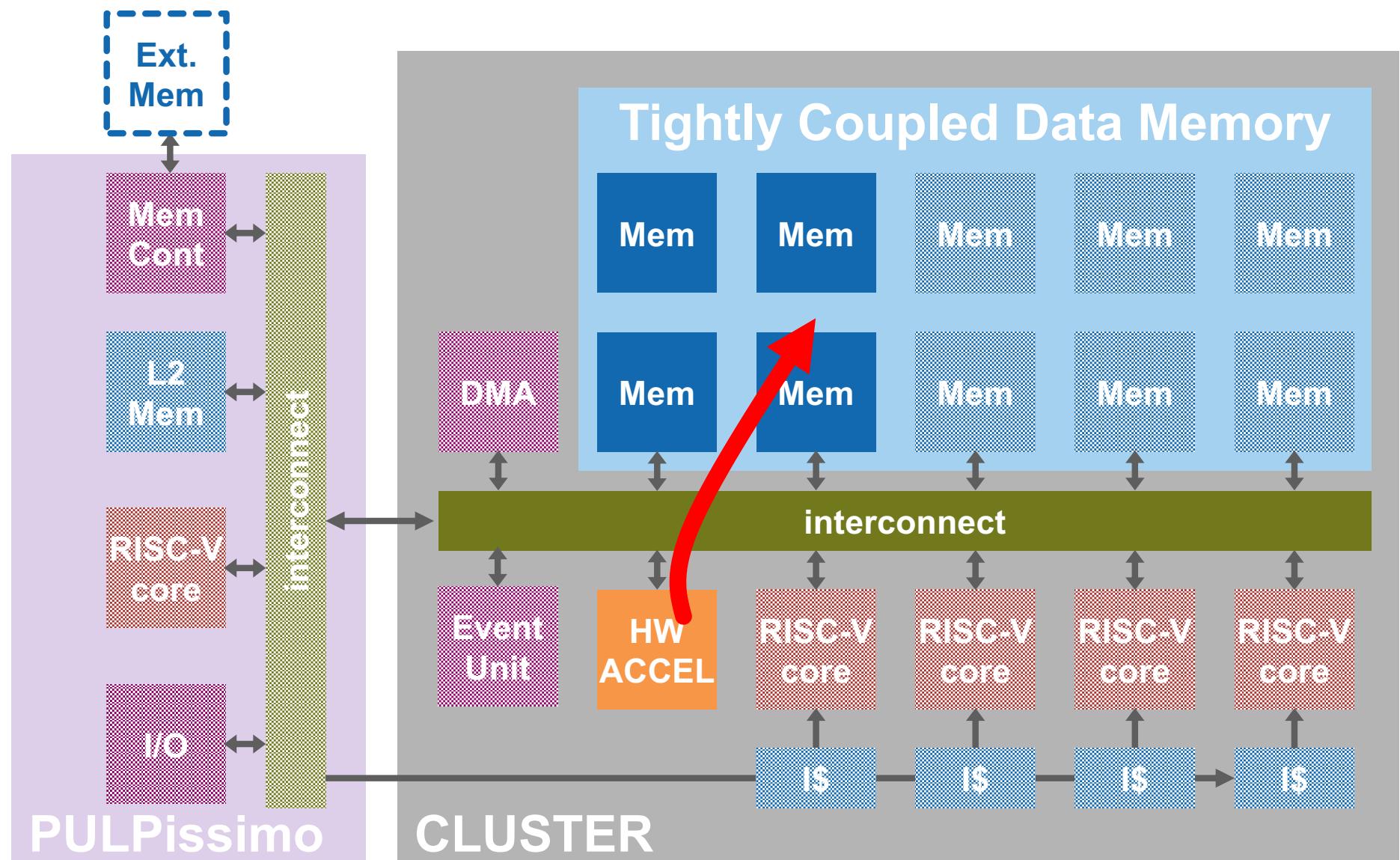
# Once data is transferred, event unit notifies cores/accel



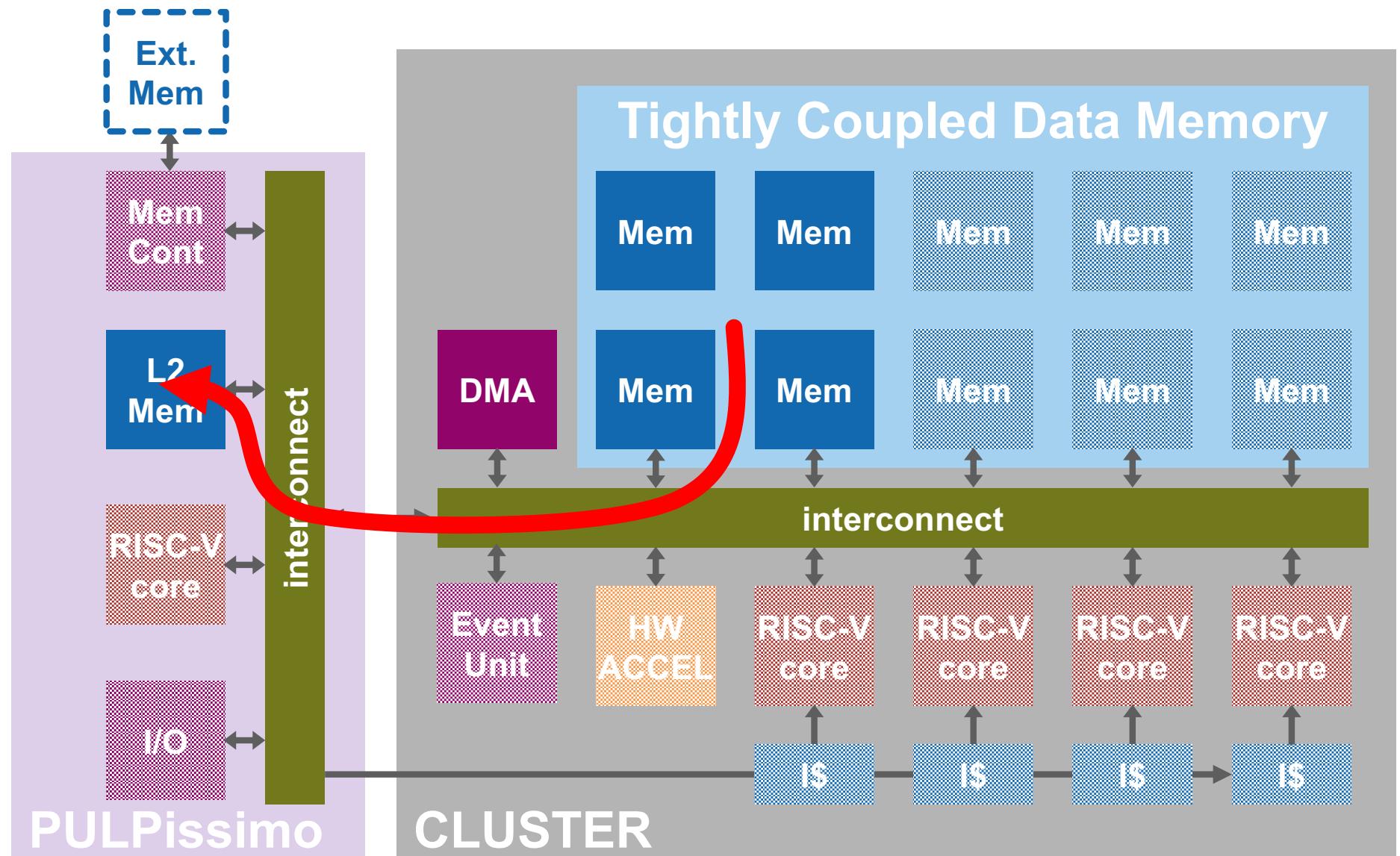
# Cores can work on the data transferred



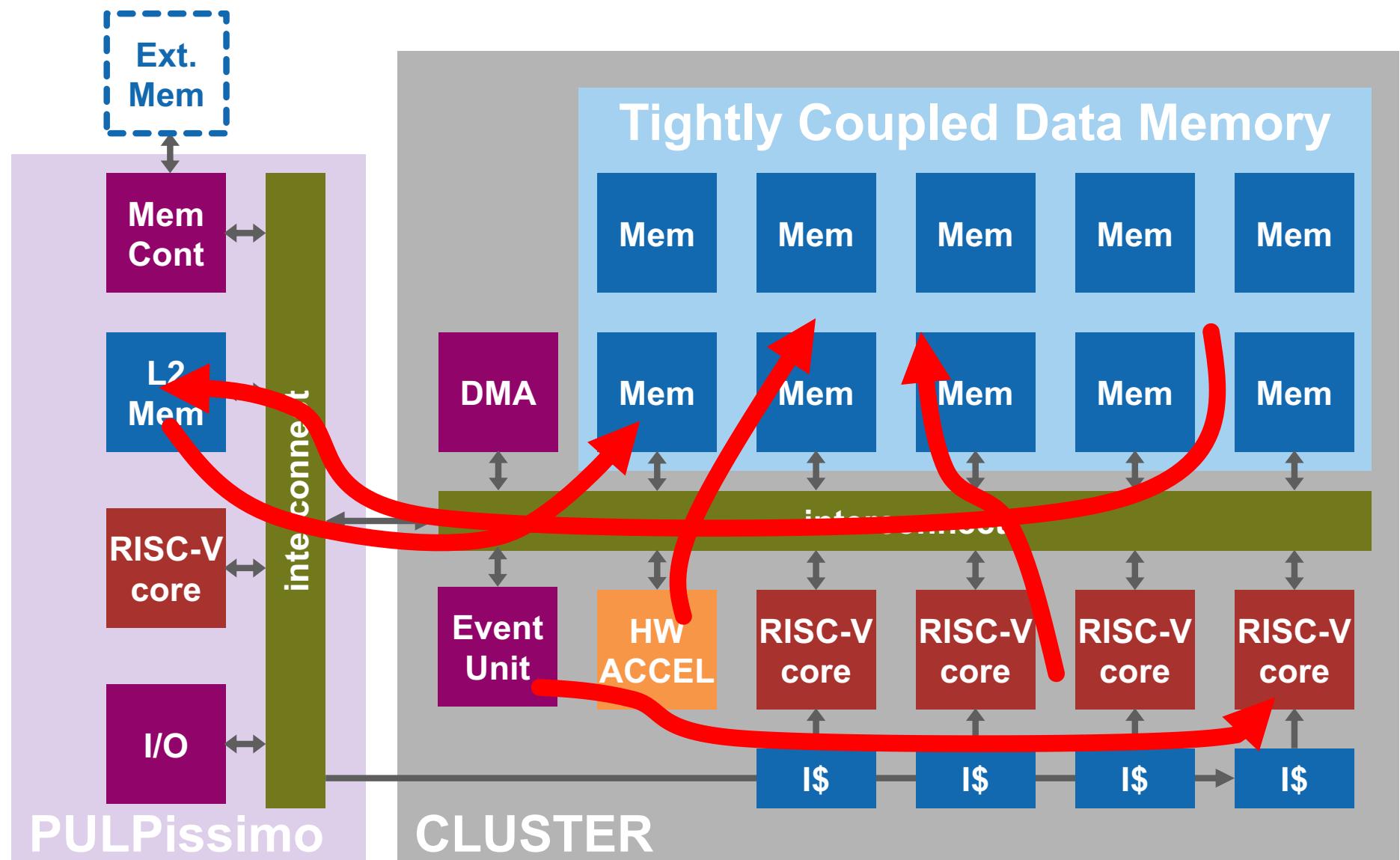
# Accelerators can work on the same data



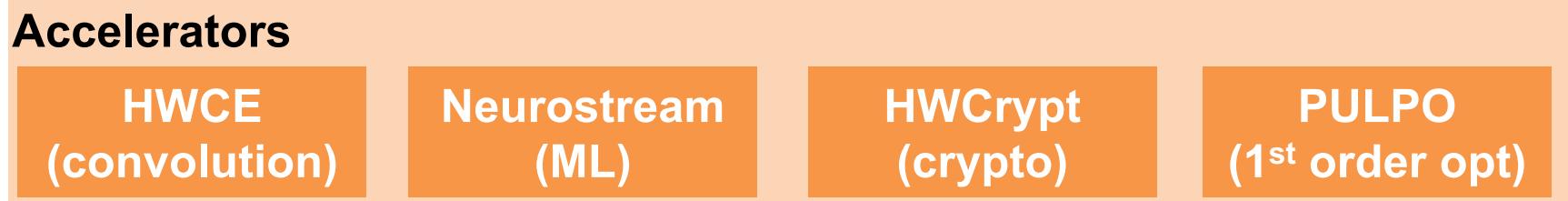
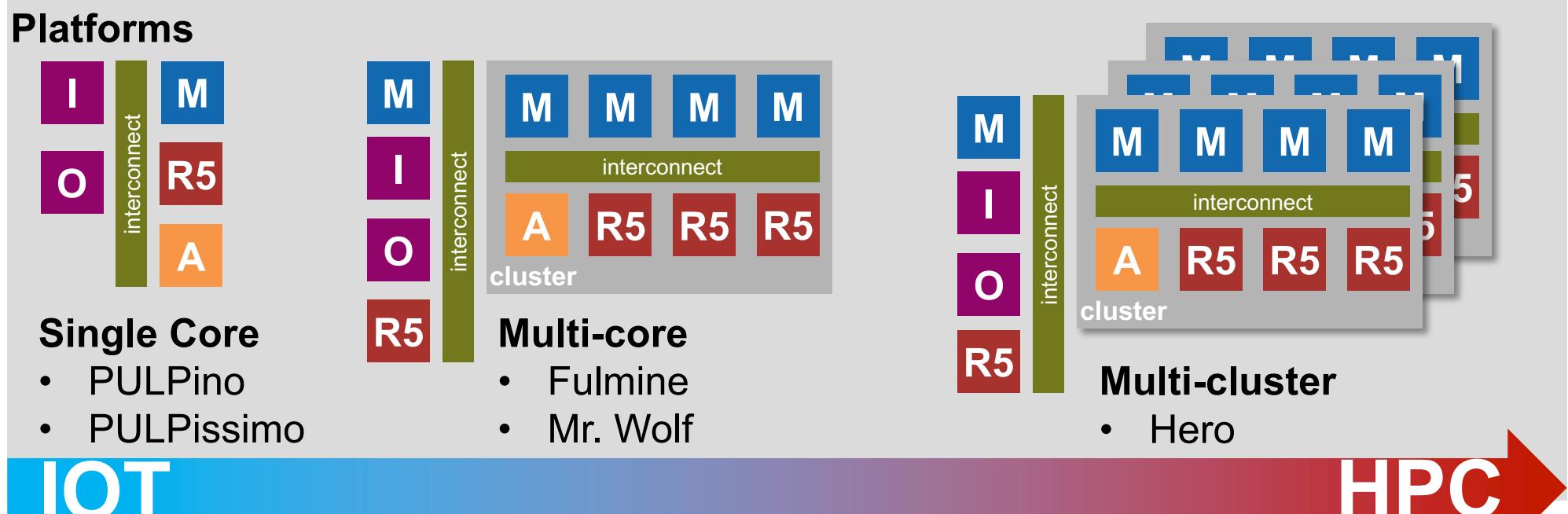
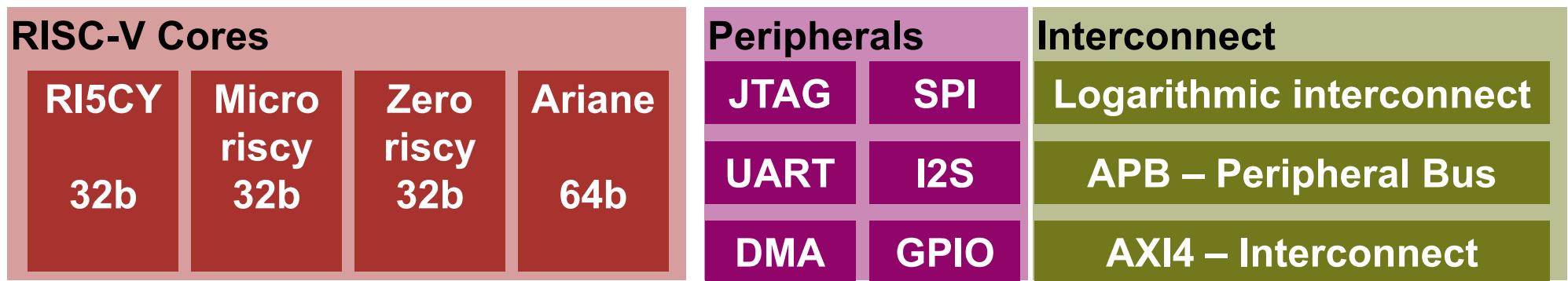
# Once our work is done, DMA copies data back



During normal operation all of these occur concurrently



# All these components are combined into platforms

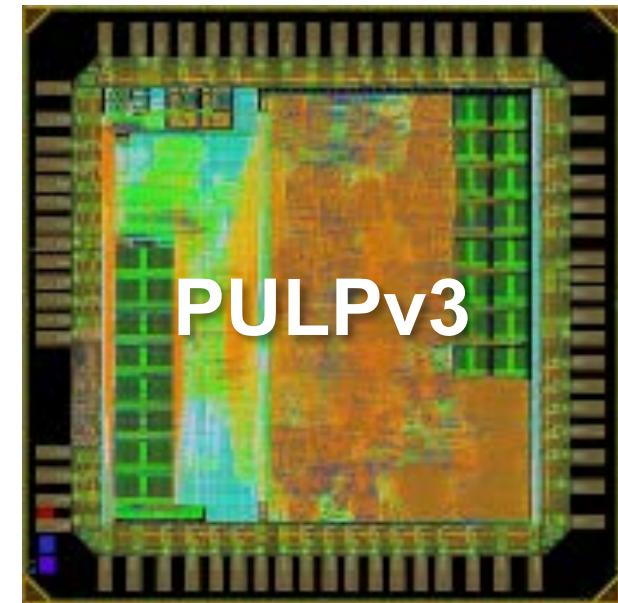
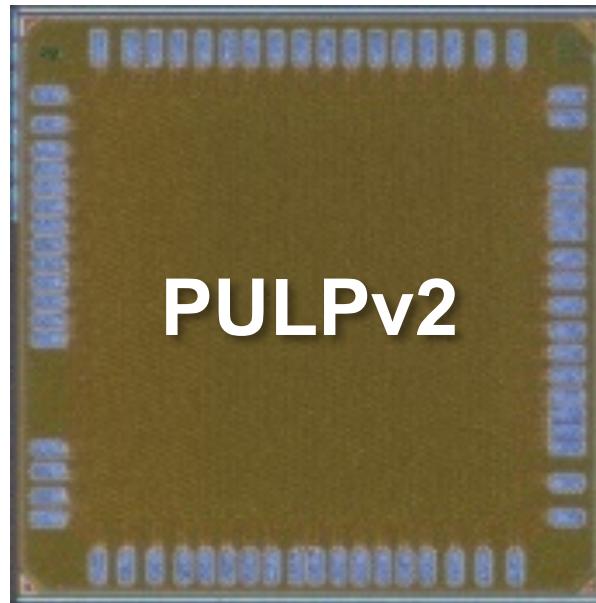
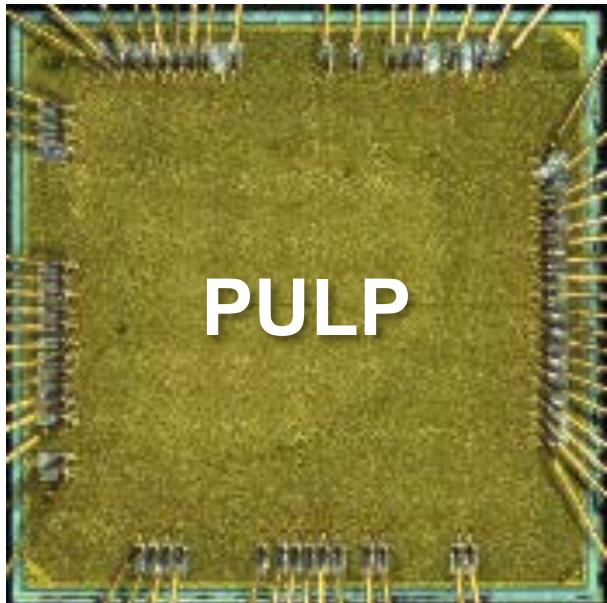


# Multi-cluster systems for HPC applications

- These systems are meant to work as accelerators for larger systems
- Optimized for data processing
- We will cover three such implementations in this workshop
  - **HERO**
  - **PowerPULP**
  - **Ariane + Open Piton**
- Both ETH Zürich and University of Bologna involved in EPI
  - Main contributions in Accelerator stream
  - Close collaboration with several groups
  - The idea is not necessary to use PULP
  - Leverage what we have developed so far

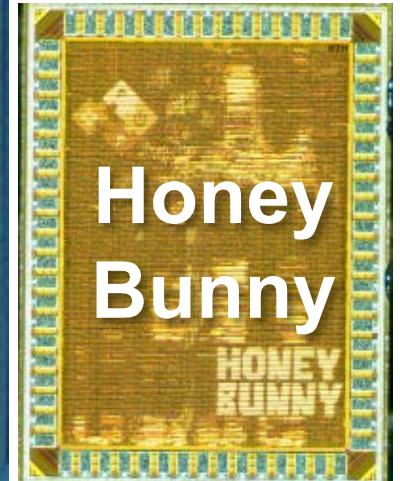
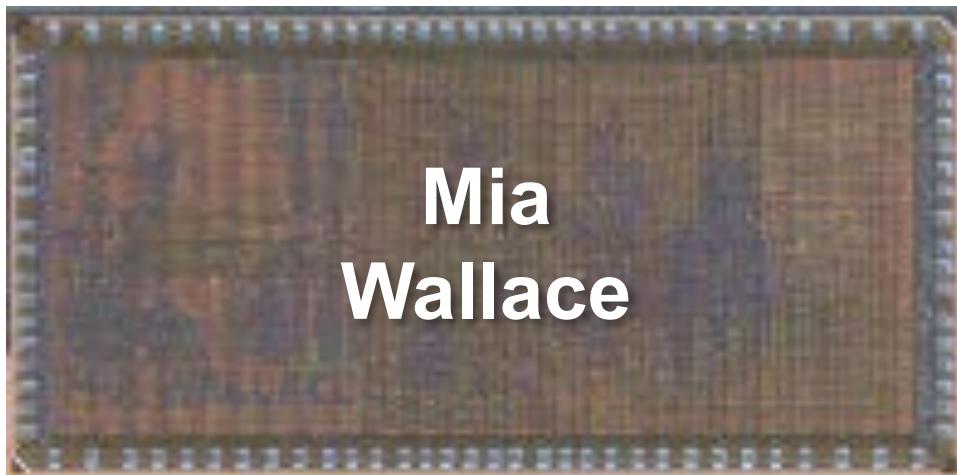


# Brief illustrated history of selected ASICs



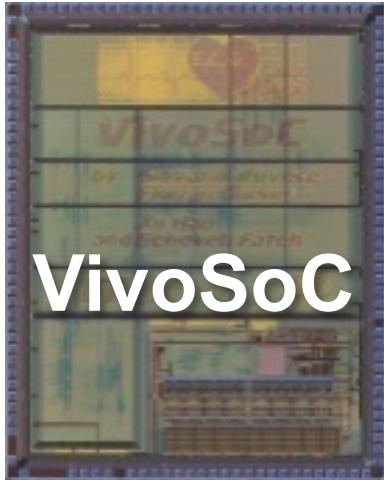
- All are 28 FDSOI technology, RVT, LVT and RVT flavor
- Uses OpenRISC cores
- Chips designed in collaboration with STM, EPFL, CEA/LETI
- PULPv3 has ABB control

# The first system chips, meant for designing boards

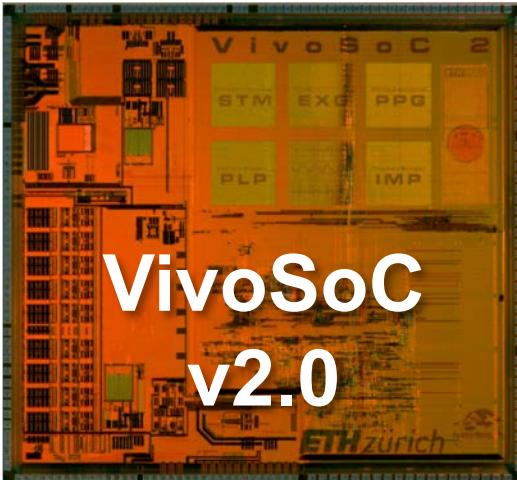


- First multi-core systems that were designed to work on development boards. Each have several peripherals (SPI, I2C, GPIO)
- **Mia Wallace** and **Fulmine** (UMC65) use OpenRISC cores
- **Honey Bunny** (GF28 SLP) uses RISC-V cores
- All chips also have our own FLL designs.

# Combining PULP with analog front-end for Biomedical apps



VivoSoC



VivoSoC  
v2.0



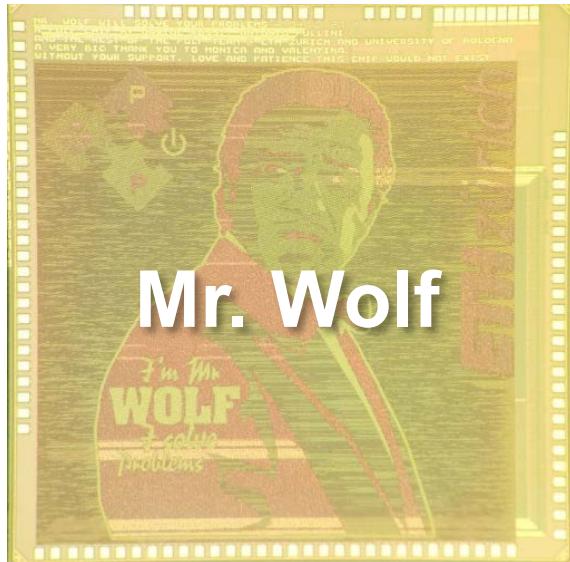
VivoSoC  
v2.001



VivoSoC  
v3

- 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

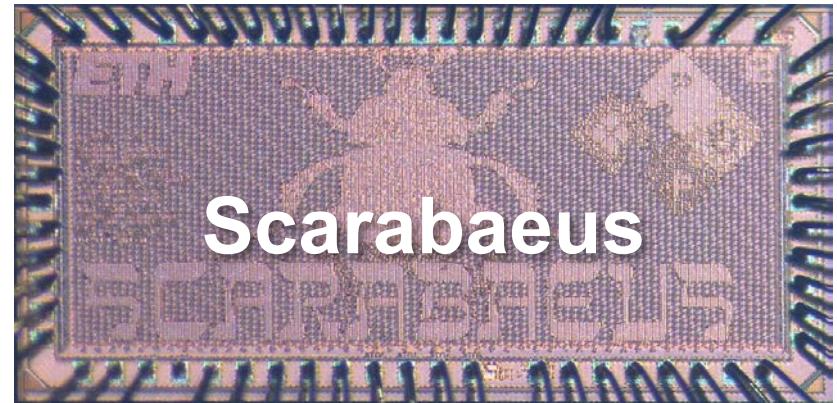
# The new generation chips from 2018



Mr. Wolf



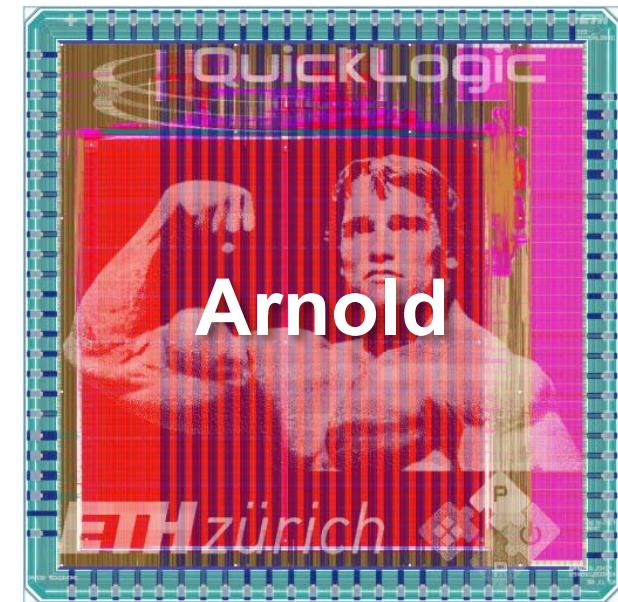
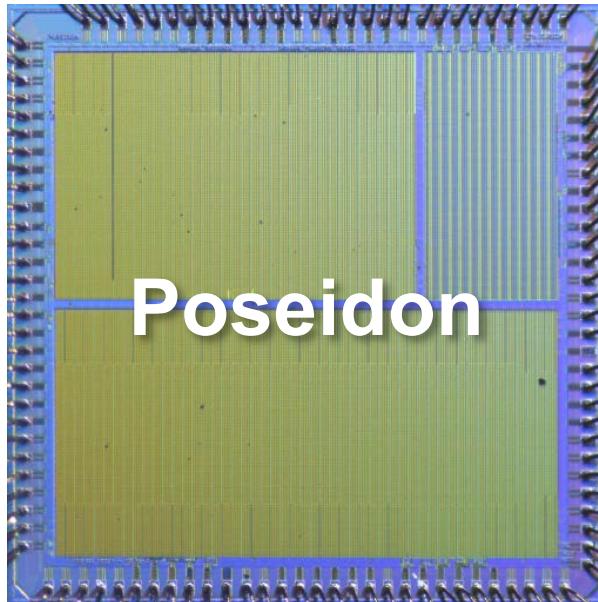
Atomario



Scarabaeus

- System chips in TSMC40 (Mr. Wolf) and UMC65
- **Mr. Wolf:** IoT Processor with 9 RISC-V cores (Zero-riscy + 8x RI5CY)
- **Atomario:** Multi cluster PULP system (2 clusters with 4 RI5CY cores)
- **Scarabaeus:** Ariane based microcontroller

# The large system chips from 2018



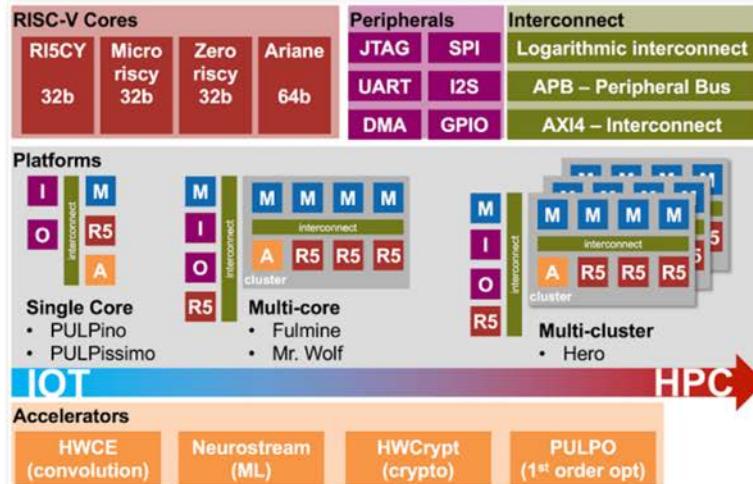
- All are 22nm Globalfoundries FDX, around 10 sqmm, 50-100 Mtrans
- **Poseidon:** PULPissimo (RI5CY) + Ariane
- **Kosmodrom:** 2x Ariane + NTX
- **Arnold:** PULPissimo (RI5CY) + Quicklogic eFPGA

# We firmly believe in Open Source movement

PULP Platform Resources ▾ About ▾ FAQ Privacy Policy Contact



144 days left until the RISC-V conference at ETH Zurich



Github

Twitter

## Get PULP now!

- You can get the source code for PULP-based systems released under a permissible SolderPad open-source license from [Github](#) now.
- If you want to program with PULP you can get the [SDK](#) and use the virtual platform.
- The following companies sell development boards with chips based on our PULP system:
  - GreenWaves sells a [development board](#) of their [GAP8](#) chip.
  - [open-isa.org](#) sells their [RV32M1-VEGA](#) development board.

First launched in February 2016 (github)



# 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.

# 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

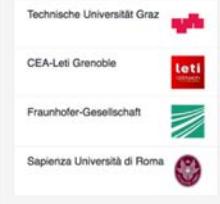
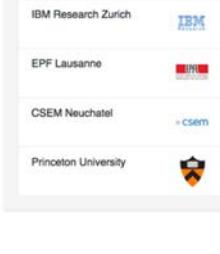
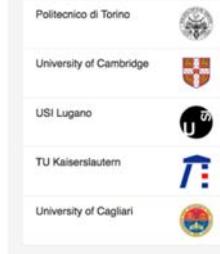
- **The following are ok:**
  - RTL code written in HDL, or a high-level language for HLS flow
  - Testbenches in HDL and associated makefiles, golden models
- **How about support scripts for different tools?**
  - Synthesis scripts, tool startup files, configurations
- **And these are currently no go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

# Silicon and Open Hardware fuel PULP success

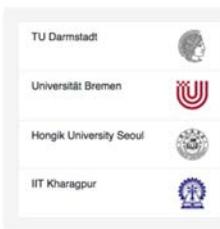
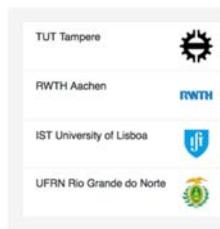
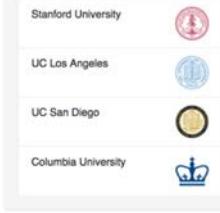
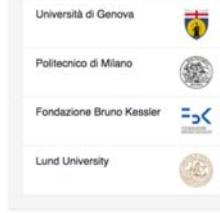
- Many companies (we know of) are actively using PULP
  - They value that it is **silicon proven**
  - They like that it uses a **permissive open source license**



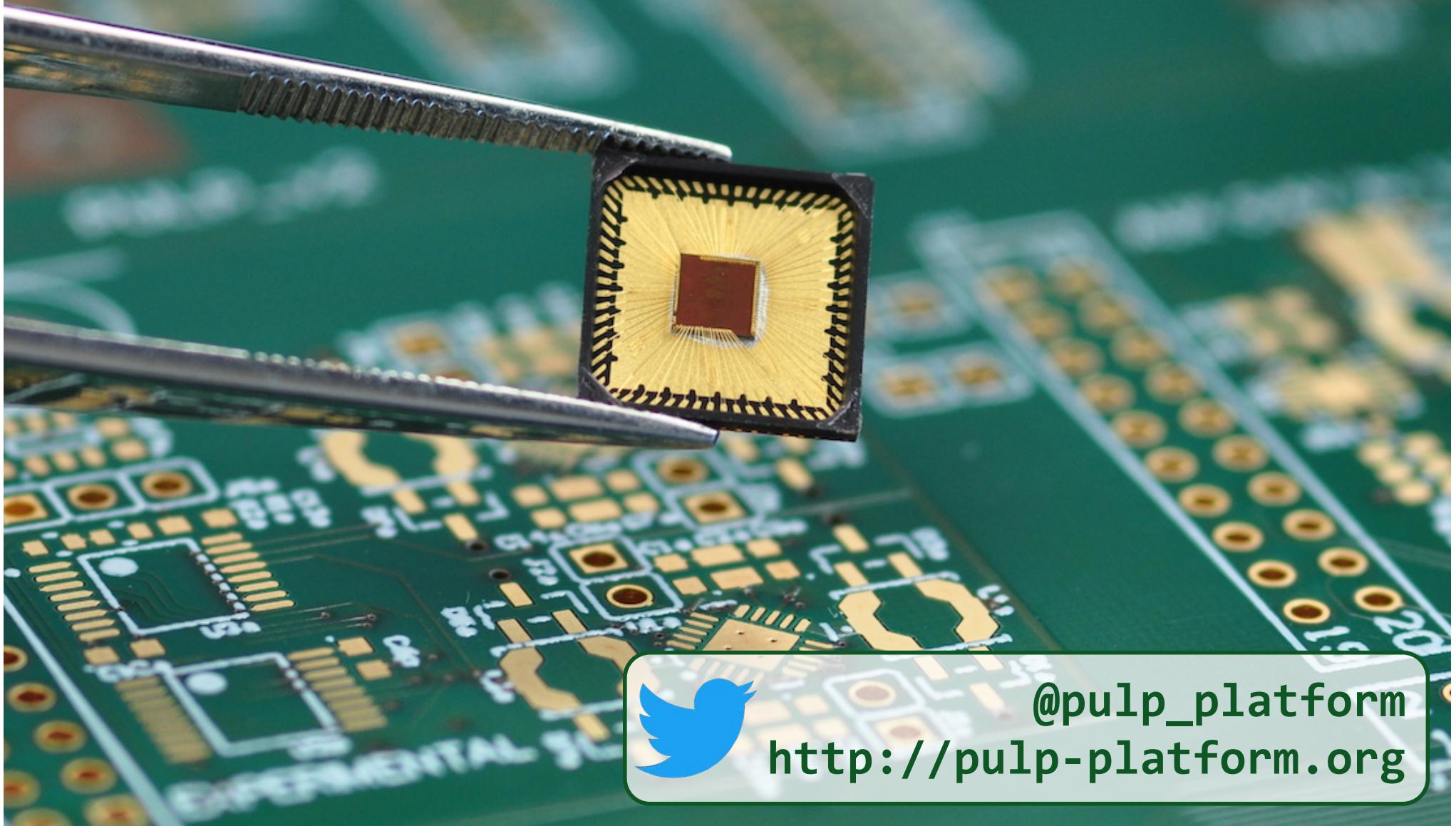
Direct research collaborators on PULP



Academic users we are aware of



# QUESTIONS?



@pulp\_platform  
<http://pulp-platform.org>

# Who says Open Source does not pay?

- List your open source HW/SW on the Eurolab4HPC www site and get
- Summer of Code activity on Transprecision Computing
- Up to **10x** projects will be supported

3000€

- For winner & **1000€** for two runner ups
- Register by 1<sup>st</sup> of March
- <https://www.eurolab4hpc.eu/open-source/call/>



Eurolab4HPC



6000€

- Register by 1<sup>st</sup> of March
- Has to be open source
- <http://oprecomp.eu/open-source>



Join us in Week of Open Source HW, June 11-14 Zürich

# WOSH

the Week of Open Source Hardware

June 11-14 Zürich, SWITZERLAND

- Official RISC-V Workshop (June 11-12)
- RISC-V foundation member meetings (June 13)
- Eurolab4HPC, Open Source Innovation Camp (June 13)
- Licensing and IP rights for Open source HW (June 13)
- FOSSI: Path to high quality IP, Open source EDA tools (June 14)
- Tutorials, demos, hackathons

