PULP and HERO Tutorial

at Week of Open-Source Hardware (WOSH) *14.06.2019*

Andreas Kurth

and the PULP team led by Prof. Luca Benini

[1]Department of Electrical, Electronic and Information Engineering

[2]Integrated Systems Laboratory

Agenda

09:00	Introduction to the **PULP Cluster** and its **Execution Model**: Software-Managed Scratchpad Memories and DMA Transfers
09:15	Introduction to the **PULP SDK**
09:30	*Break*
09:45	Introduction to **HERO**
10:00	**HERO Live Demo**
10:30	*Break*
10:45	**HERO Hands-On Programming**
11:45	**Q&A**
12:00	*Lunch Break*

Introduction to PULP Cluster and Execution Model

14.06.2019

Andreas Kurth

Frank K. Gürkaynak

and the PULP team led by Prof. Luca Benini

[1]Department of Electrical, Electronic and Information Engineering

[2]Integrated Systems Laboratory

The main PULP systems we develop are cluster based

RISC-V Cores

RI5CY 32b	Micro riscy 32b	Zero riscy 32b	Ariane 64b

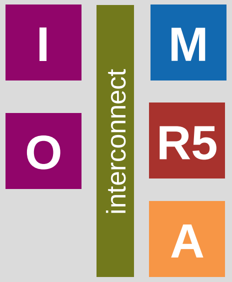
Peripherals

JTAG	SPI
UART	I2S
DMA	GPIO

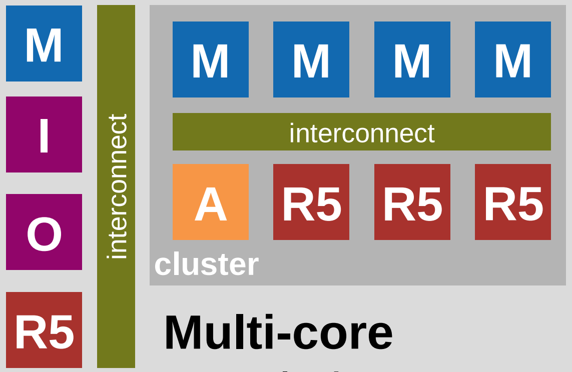
Interconnect

- Logarithmic interconnect
- APB – Peripheral Bus
- AXI4 – Interconnect

Platforms



Single Core
- PULPino
- PULPissimo



Multi-core
- Fulmine
- Mr. Wolf

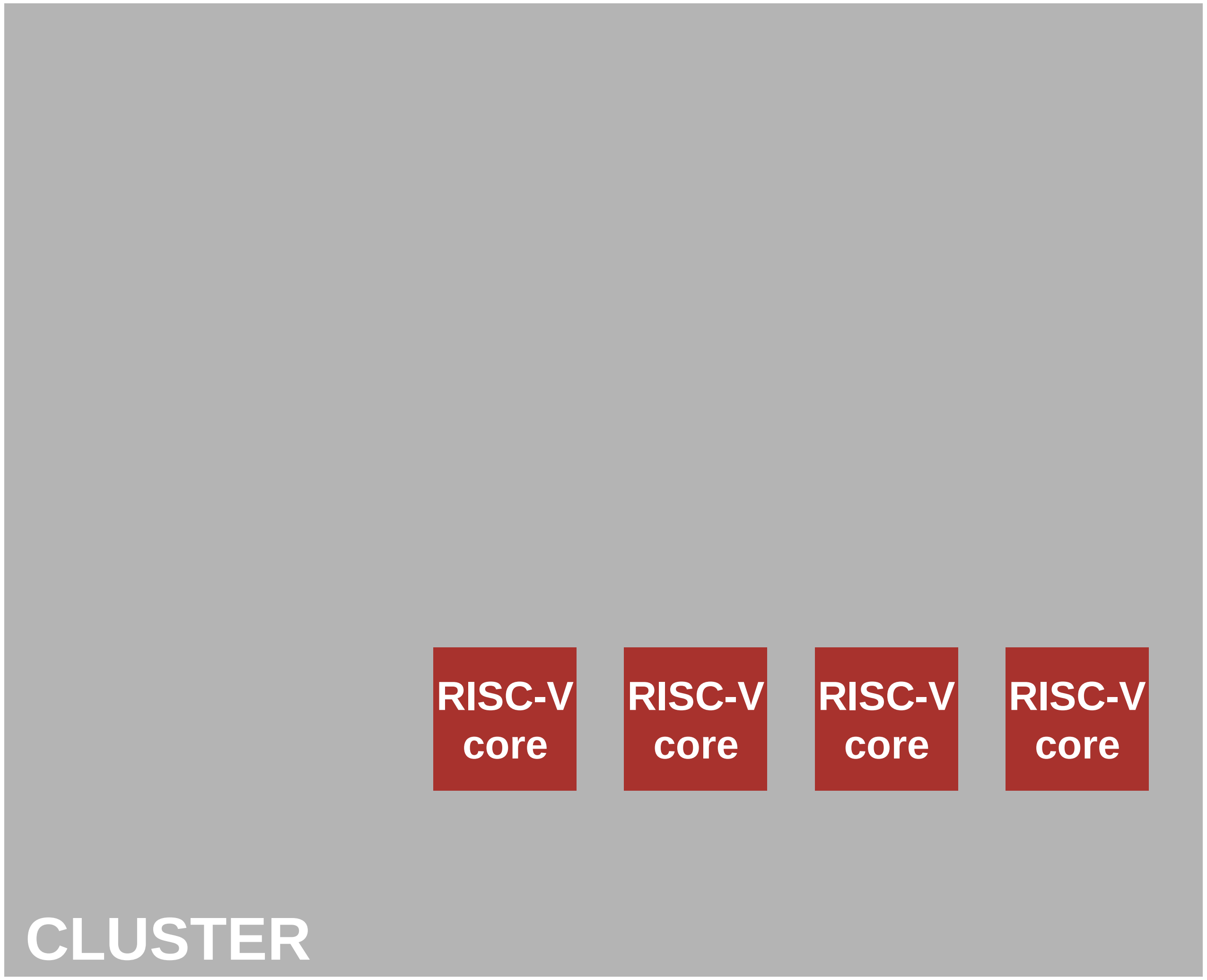
Accelerators

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

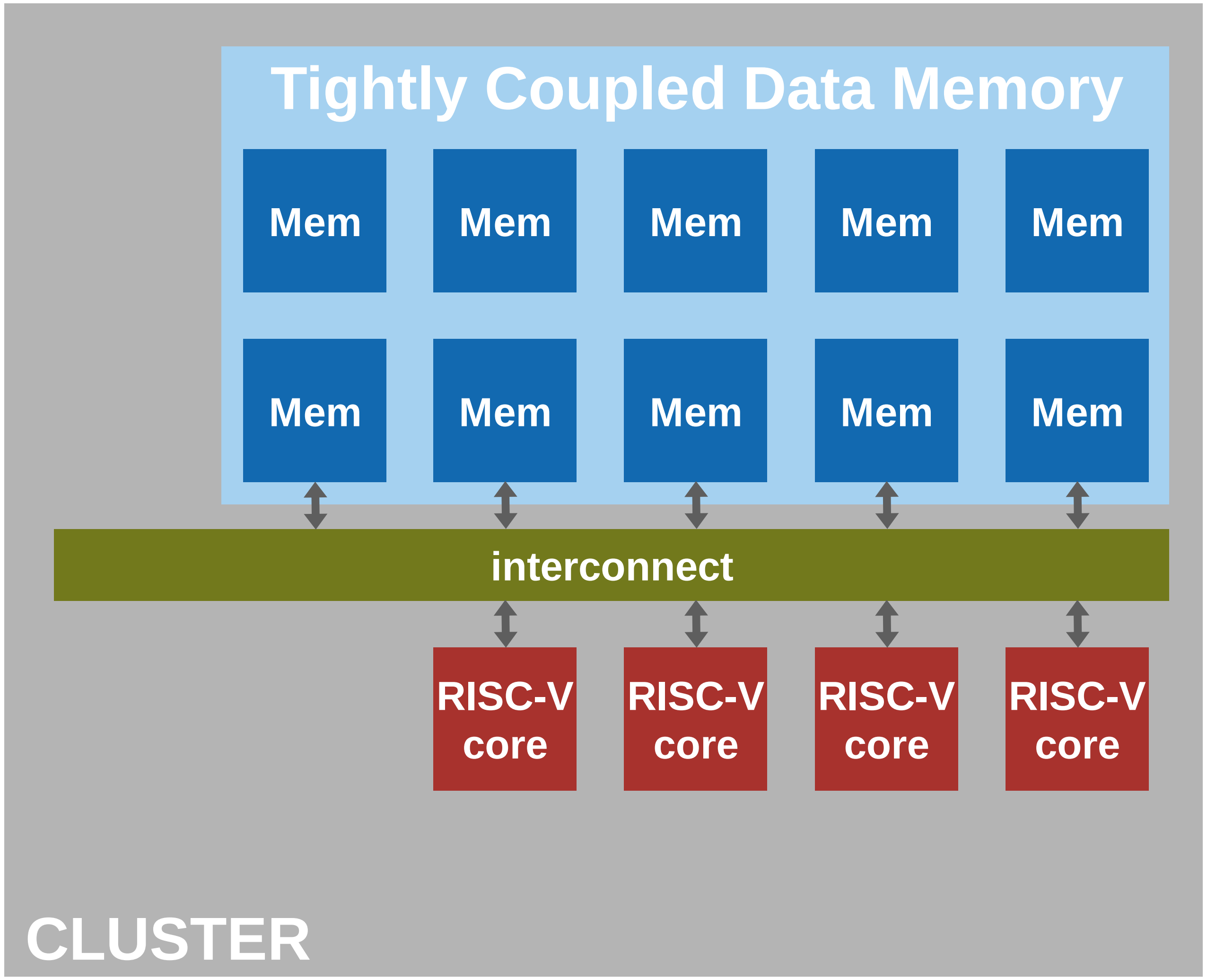
The main components of a PULP cluster

- 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.

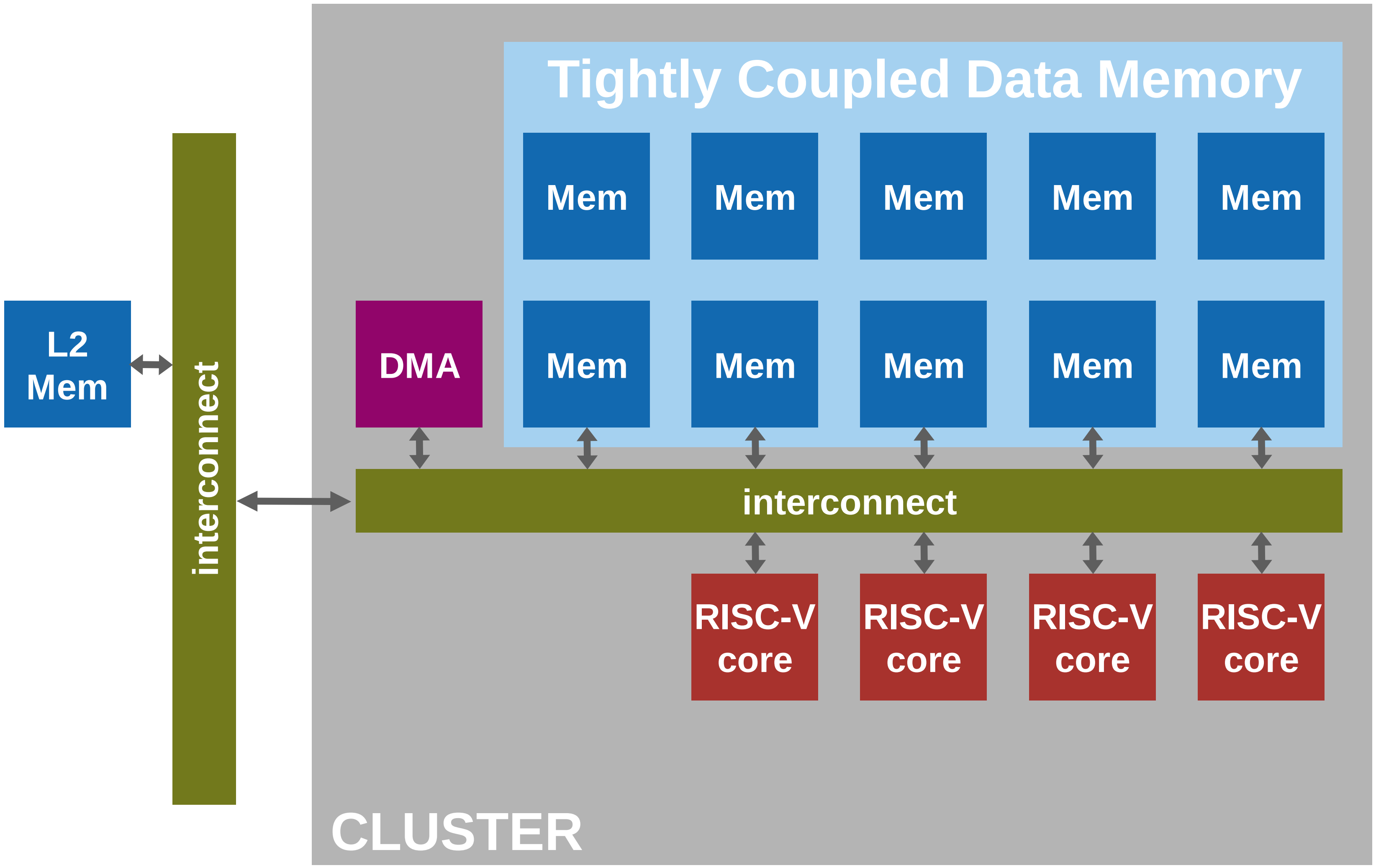
PULP cluster contains multiple RISC-V cores



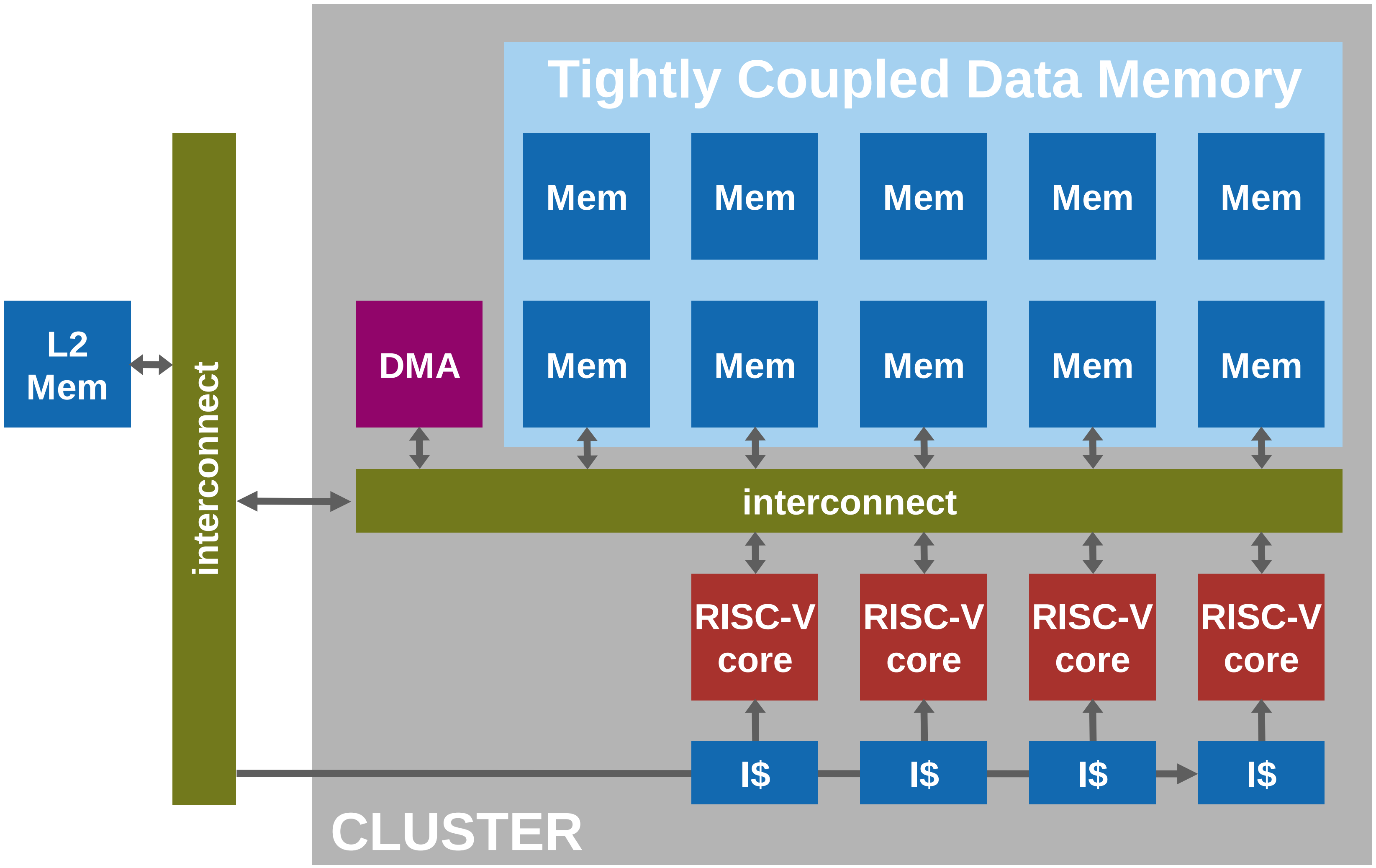
All cores can access all memory banks in the cluster



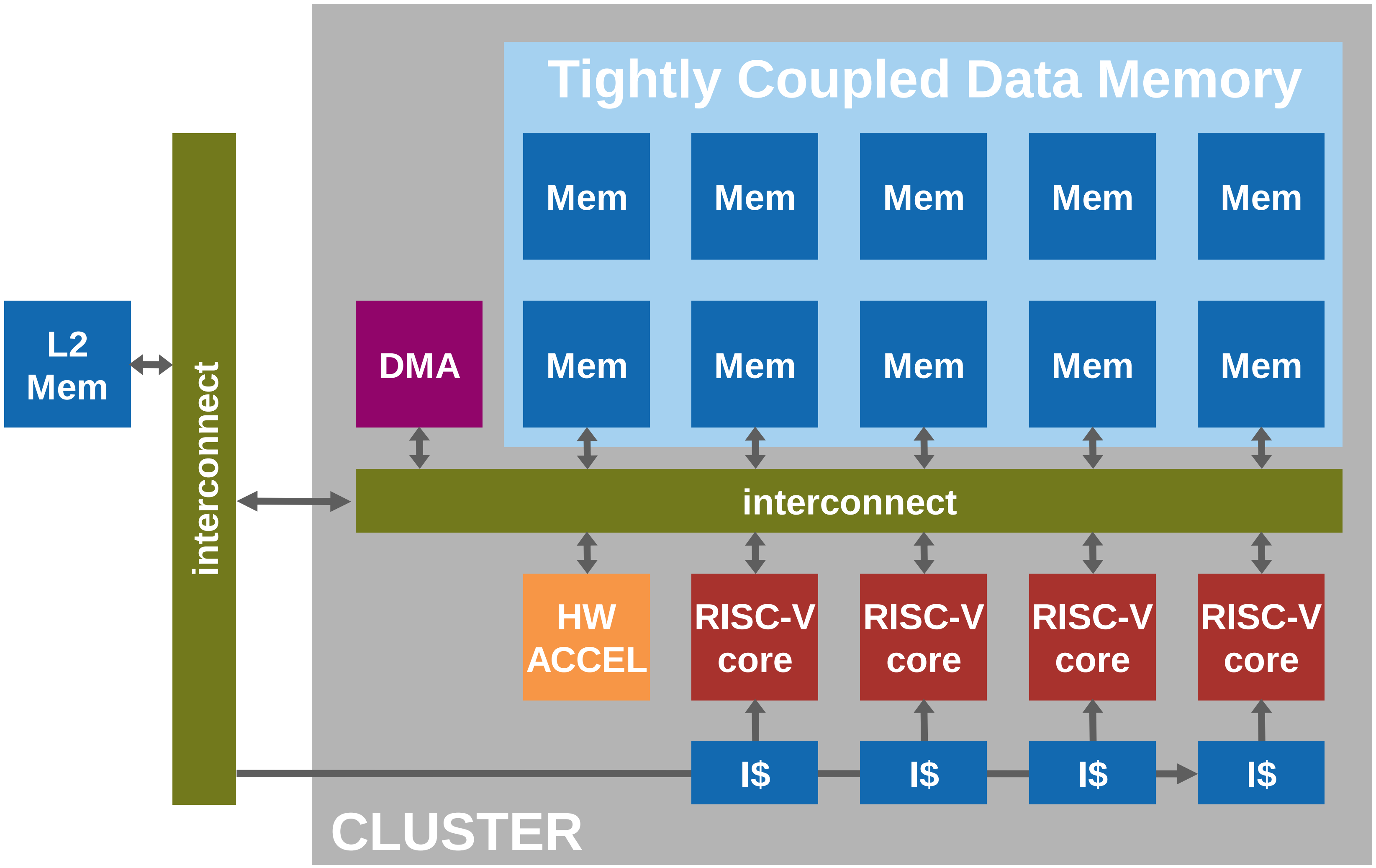
Data is copied from a higher level through DMA



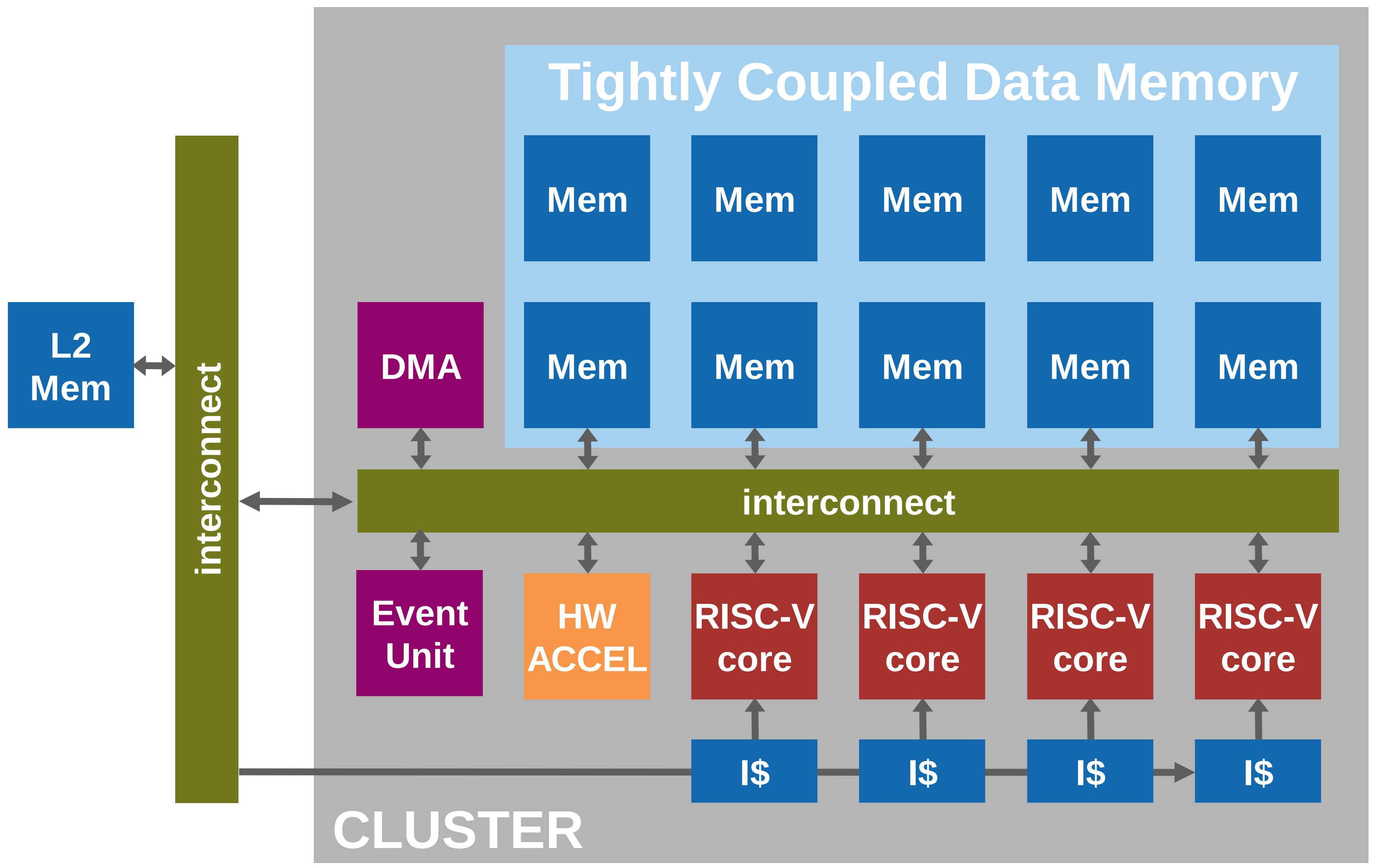
There is a (shared) instruction cache that fetches from L2



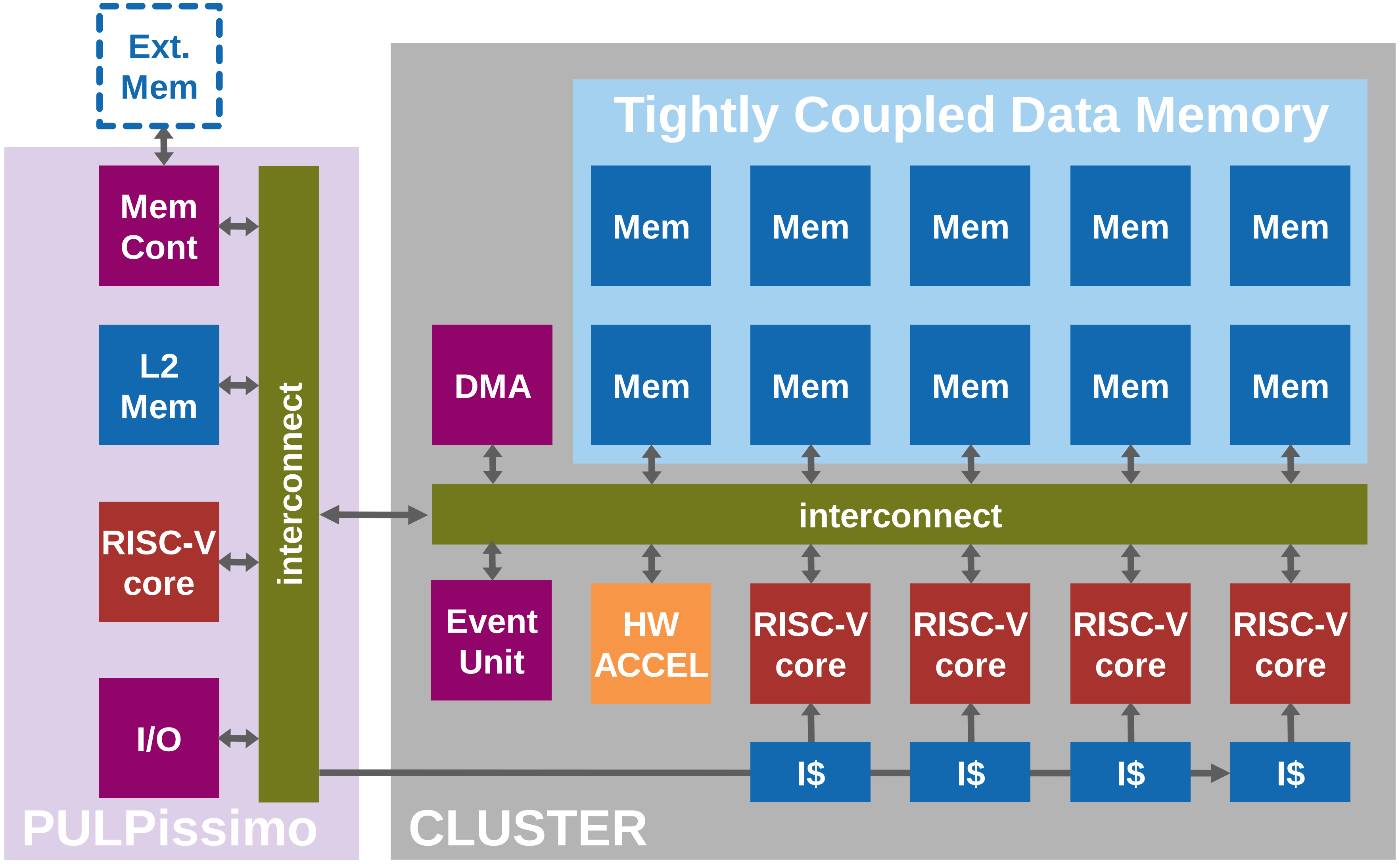
Hardware Accelerators can be added to the cluster



Event unit to manage resources (fast sleep/wakeup)

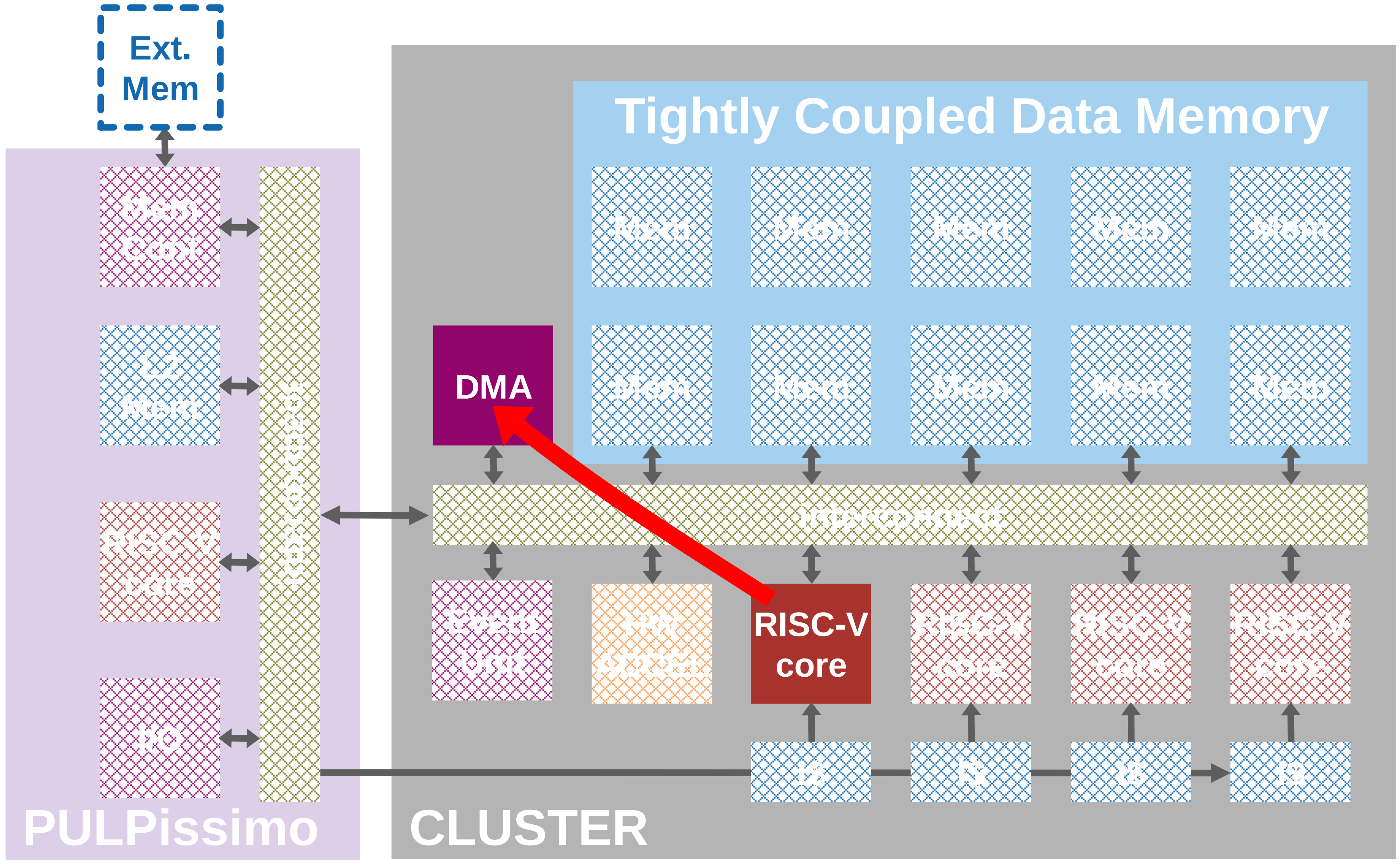


An additional microcontroller system (PULPissimo) for I/O

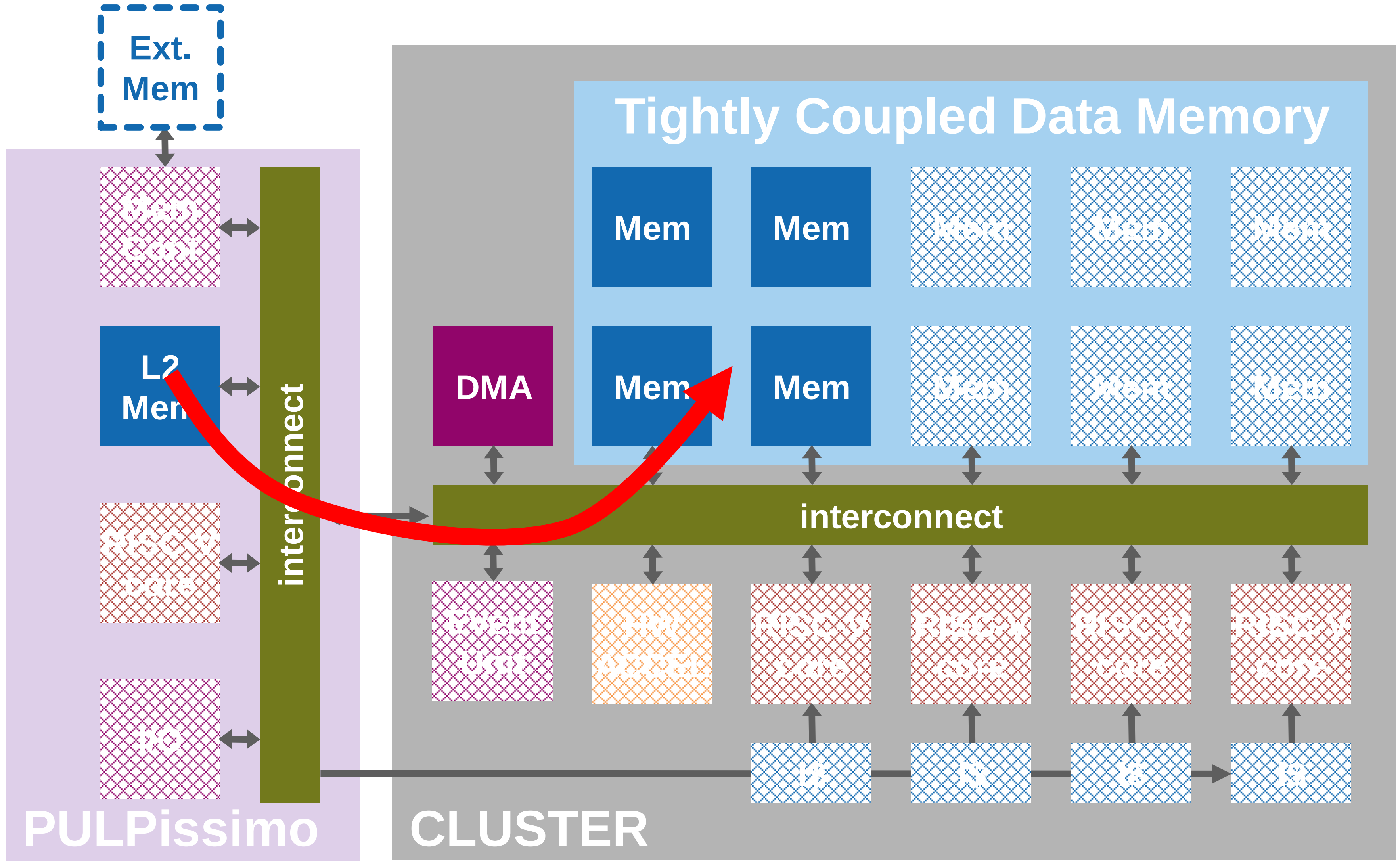


PULP Eurolab4HPC PRECOMP Open Transprecision Computing ETH

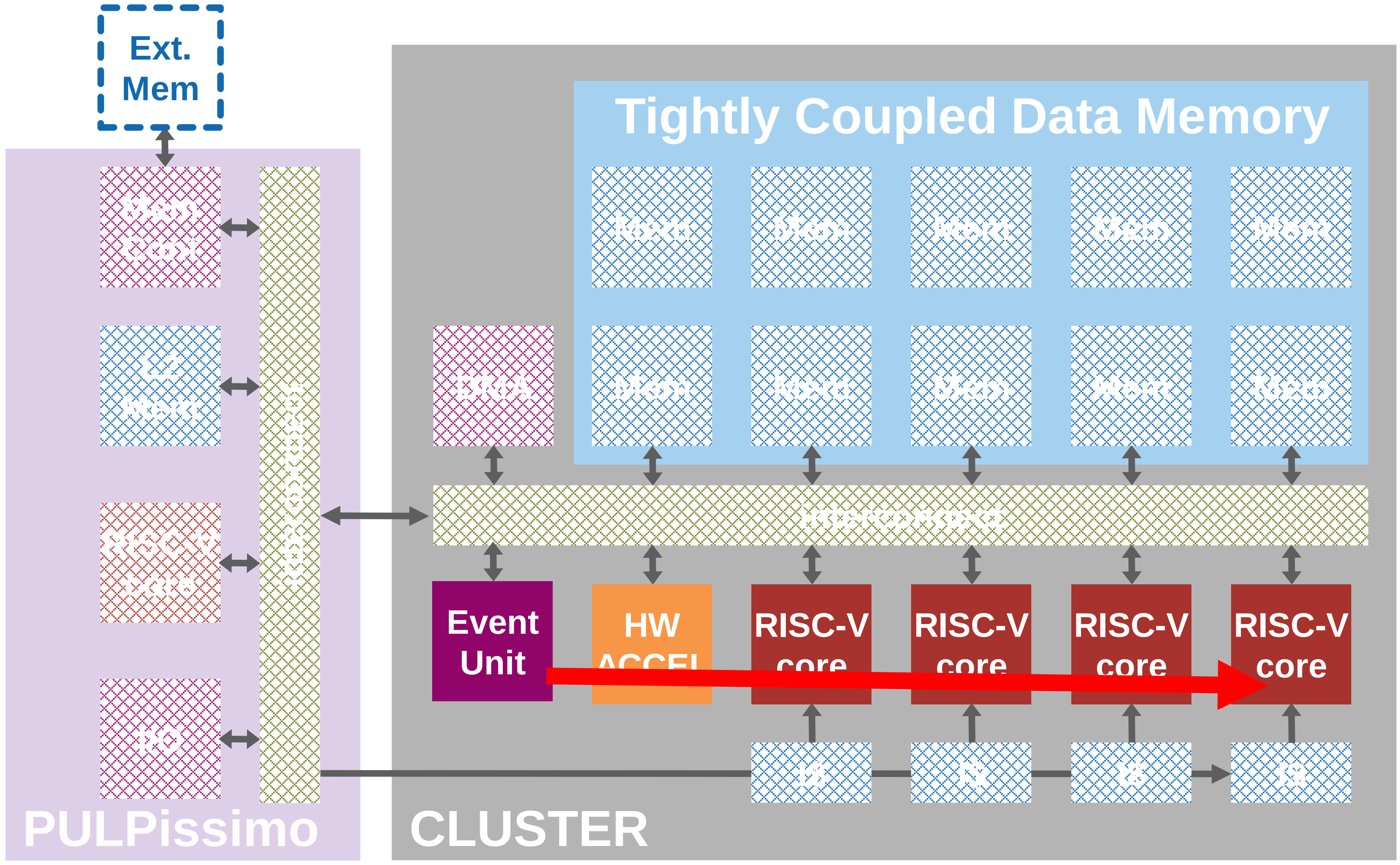
How do we work: Initiate a DMA transfer



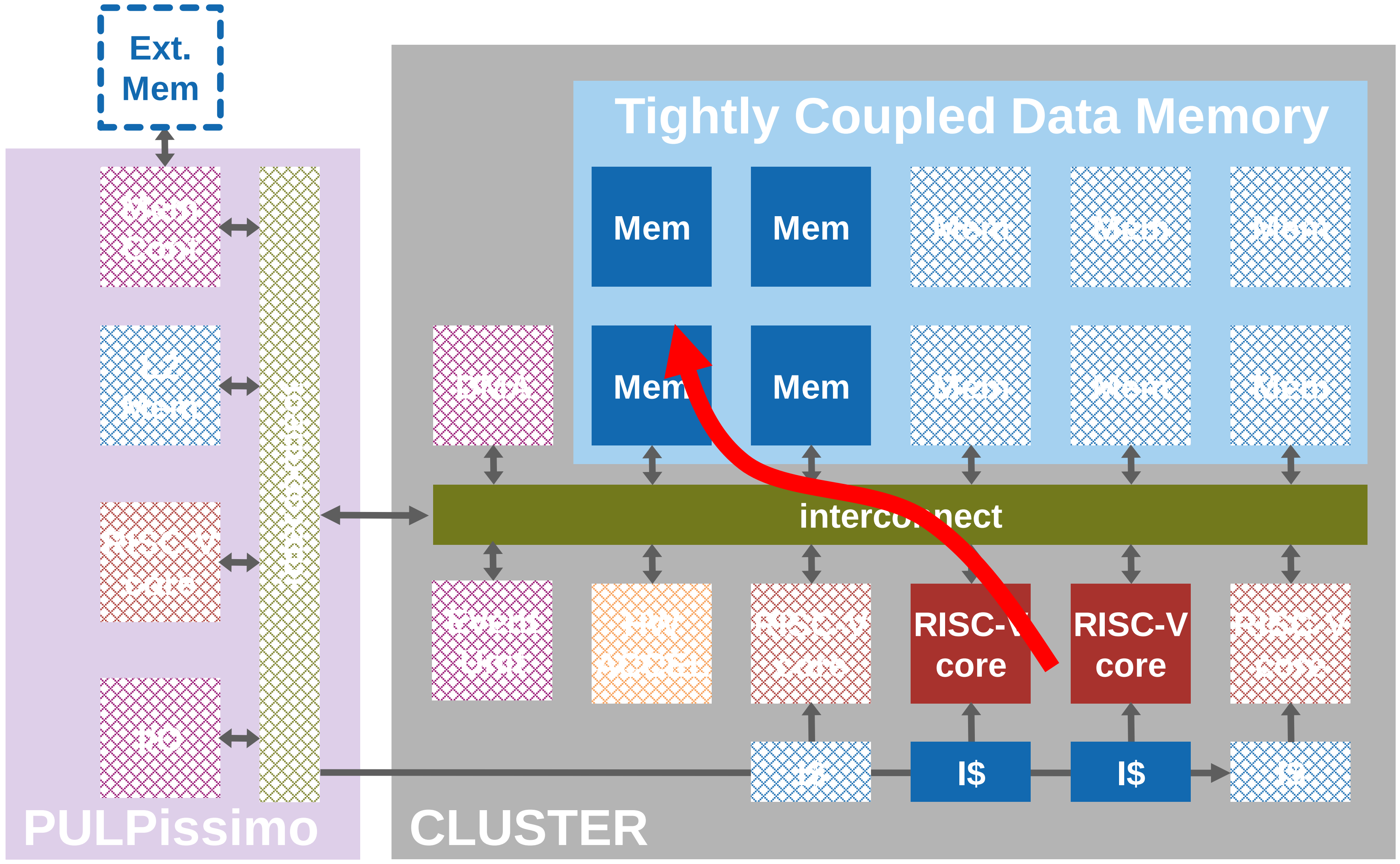
Data copied from L2 into TCDM



Once data is transferred, event unit notifies cores/accel



Cores can work on the data transferred



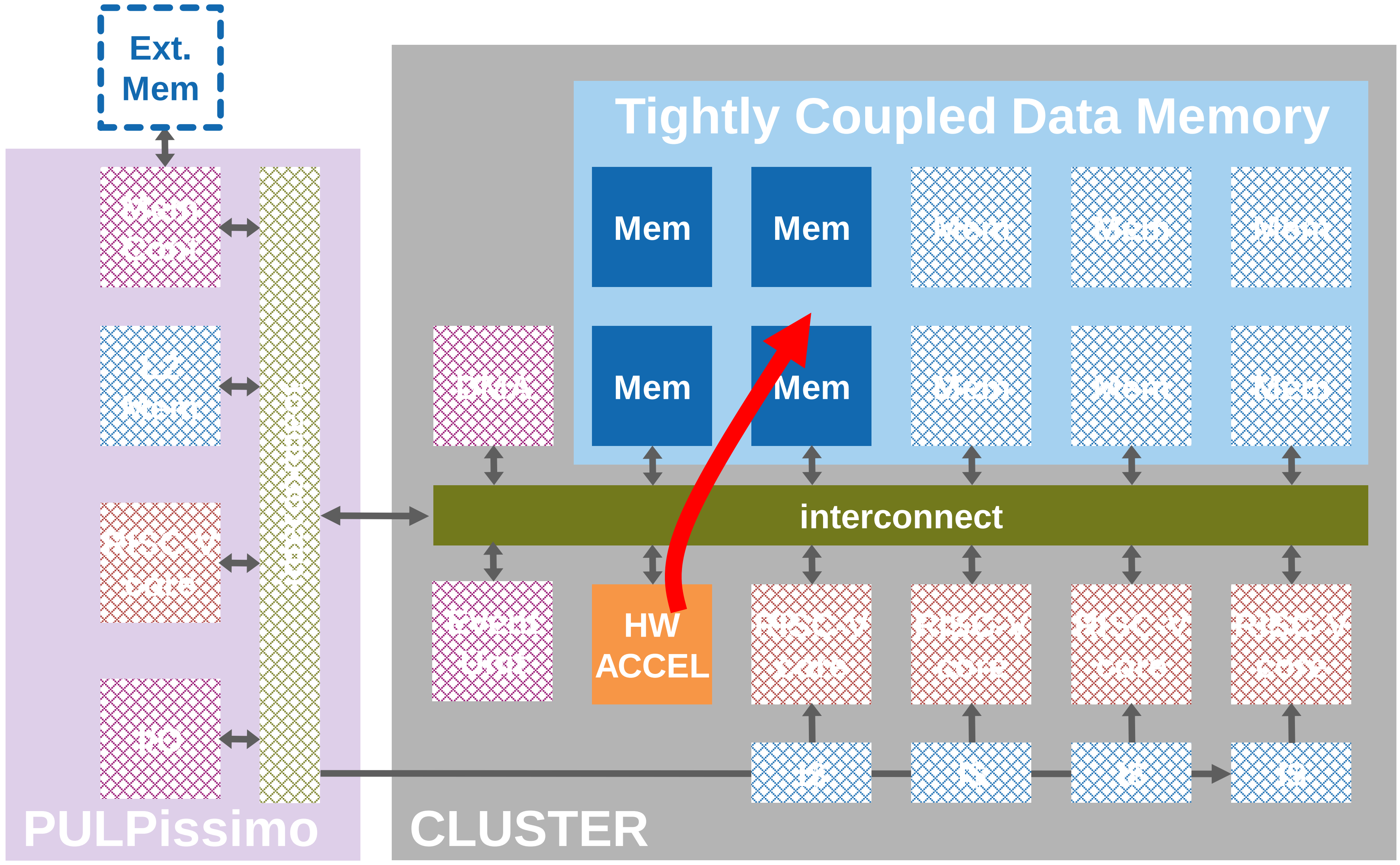
PULP

Eurolab4HPC

PRECOMP Open Transprecision Computing

ETH

Accelerators can work on the same data



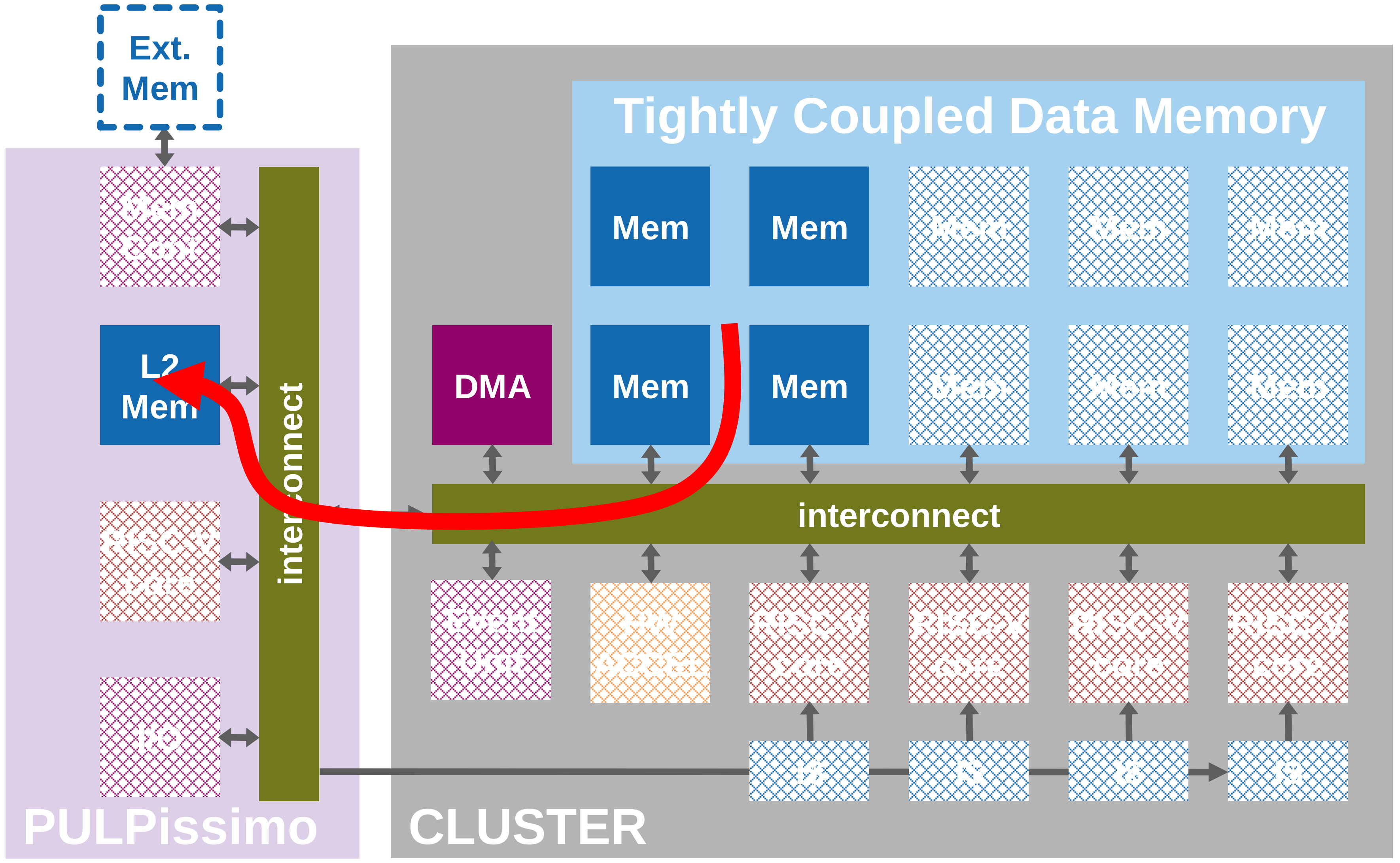
PULP

Eurolab4HPC

PRECOMP Open Transprecision Computing

ETH

Once our work is done, DMA copies data back



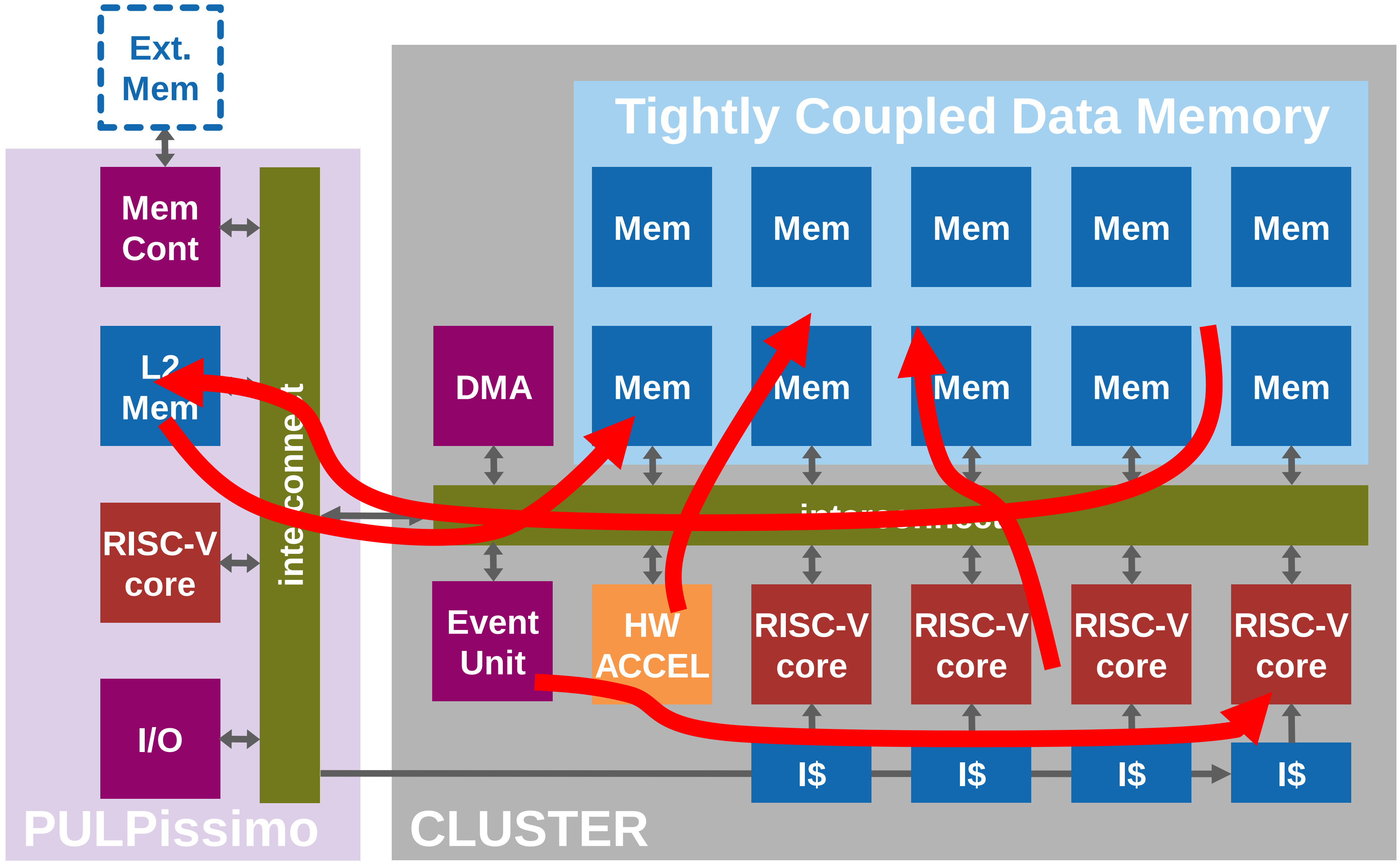
PULP

Eurolab4HPC

PRECOMP Open Transprecision Computing

ETH

During normal operation all of these occur concurrently



All these components are combined into platforms

RISC-V Cores

- RI5CY 32b
- Micro riscy 32b
- Zero riscy 32b
- Ariane 64b

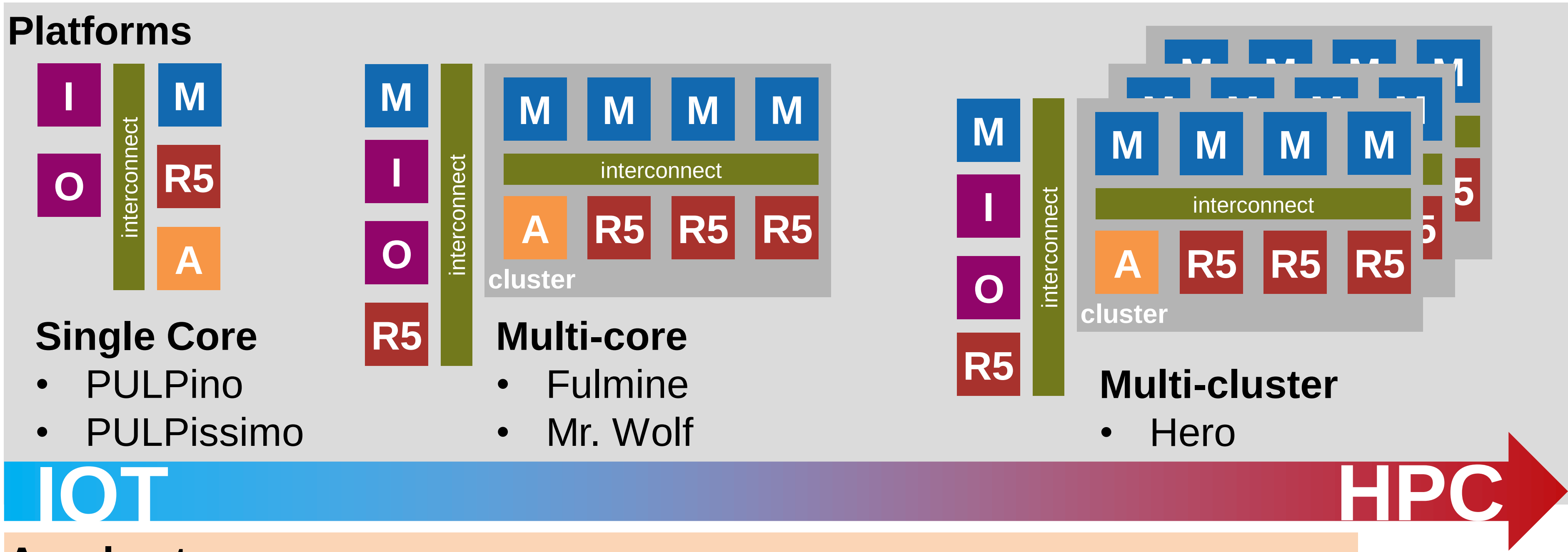
Peripherals

- JTAG
- SPI
- UART
- I2S
- DMA
- GPIO

Interconnect

- Logarithmic interconnect
- APB – Peripheral Bus
- AXI4 – Interconnect

Platforms



Single Core
- PULPino
- PULPissimo

Multi-core
- Fulmine
- Mr. Wolf

Multi-cluster
- Hero

Accelerators

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

Introduction to the PULP SDK

14.06.2019

Andreas Kurth

Germain Haugoug

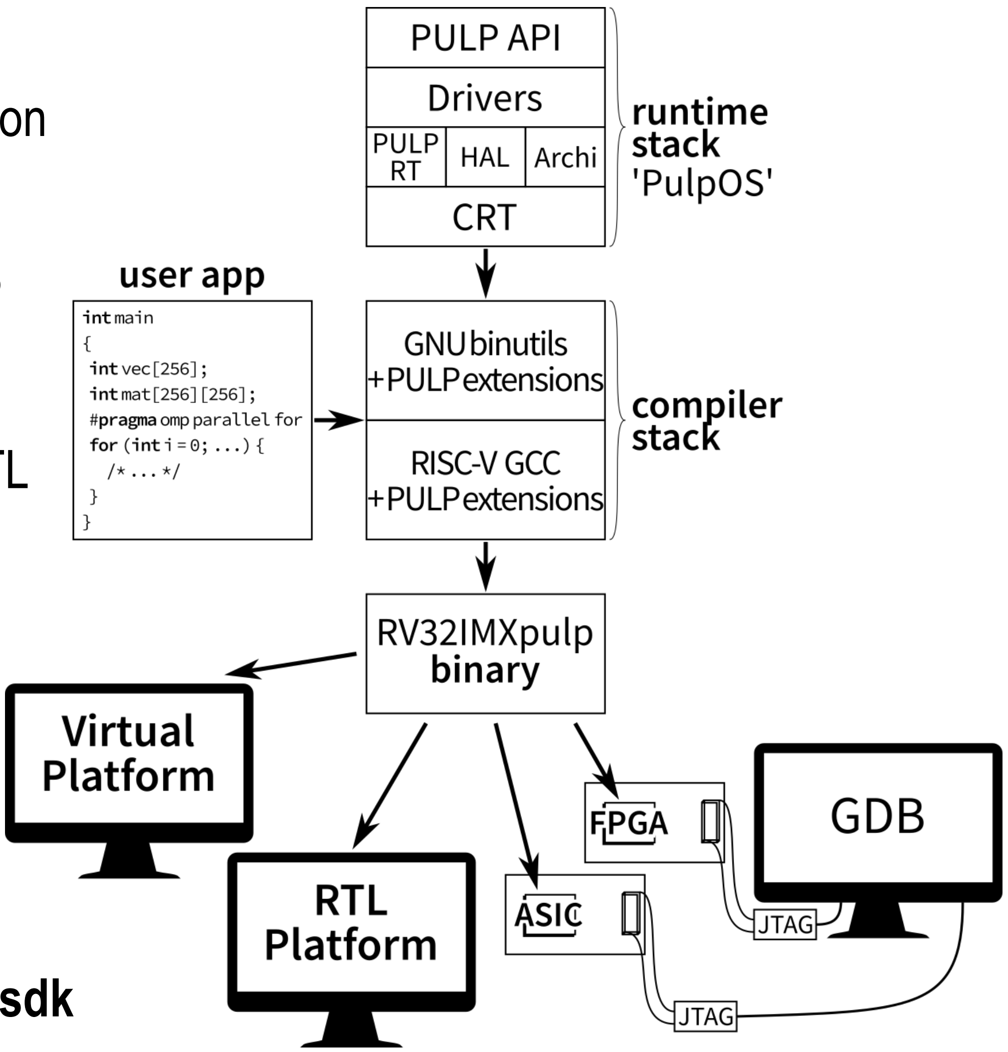
and the PULP team led by Prof. Luca Benini

[1]Department of Electrical, Electronic and Information Engineering

[2]Integrated Systems Laboratory

PULP Software Development Kit (SDK)

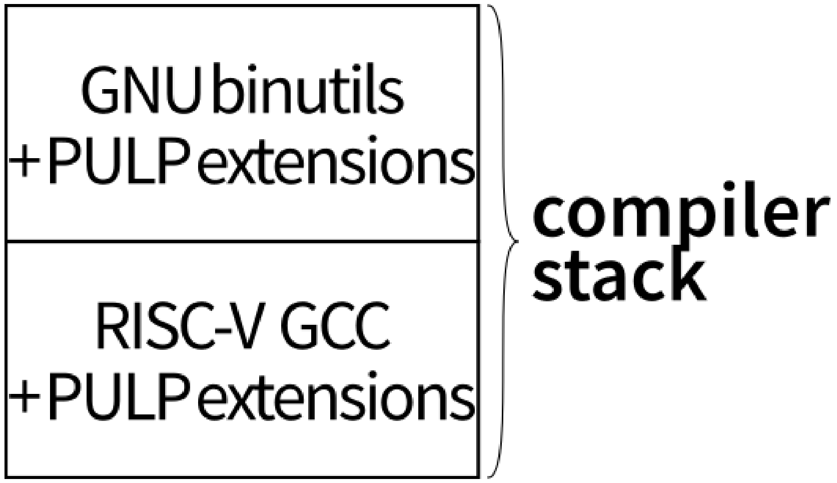
- Package for **compiling**, **running**, **debugging** and **profiling applications** on **PULP** platforms
- Supports all recent and upcoming PULP chips: Mr.Wolf, GAP, Vega, ...
- Supports all targets: Virtual Platform, RTL platform, FPGA, dev boards
- RISC-V GCC with support for PULP extensions
- Basic OpenMP support
- **Open-source**, available at https://**github**.com/**pulp-platform/pulp-sdk**



```
int main
{
 int vec[256];
 int mat[256][256];
 #pragma omp parallel for
 for (int i = 0; ...) {
   /* ... */
 }
}
```

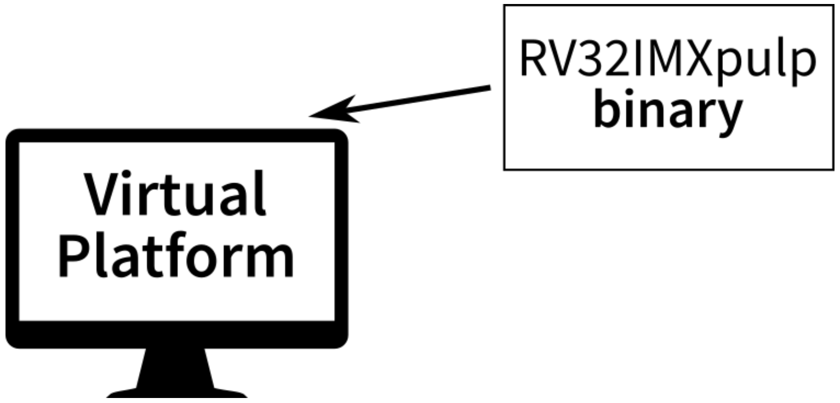
Compiler

- Forked **GCC 7.1**
- **Extended** with **all PULP custom instructions**
- Some **custom instructions instantiated by GCC** (e.g. bit manipulation instructions, auto-vectorization), others available through builtins
- **CoreMark 3.1** with RI5CY v2
- Extended binutils for **full GDB support** of **all custom instructions**



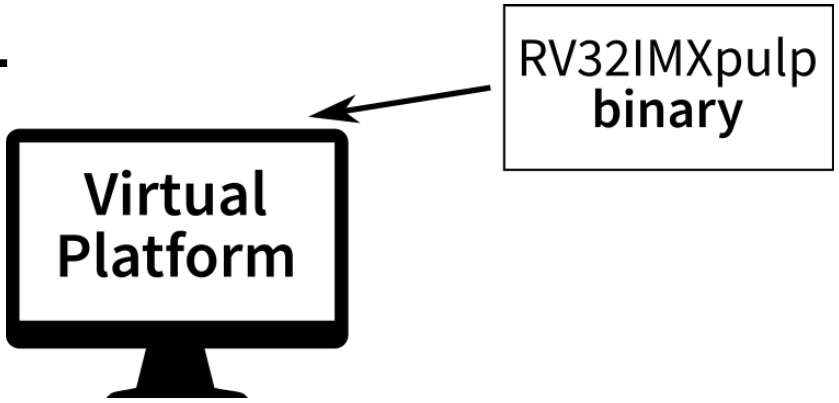
Virtual Platform

- **Simulate** the PULP architecture for **application development** with a good **trade-off** between **speed** and **accuracy** with performance feedback within a 20% error margin
- Mixed Python/C++ with component-based approach
- Event-based scheduler with cycle-based models
- Whole architecture described in JSON to ease multiple chips support
- 25000 lines of C++ (including ISS) and 4000 lines of Python



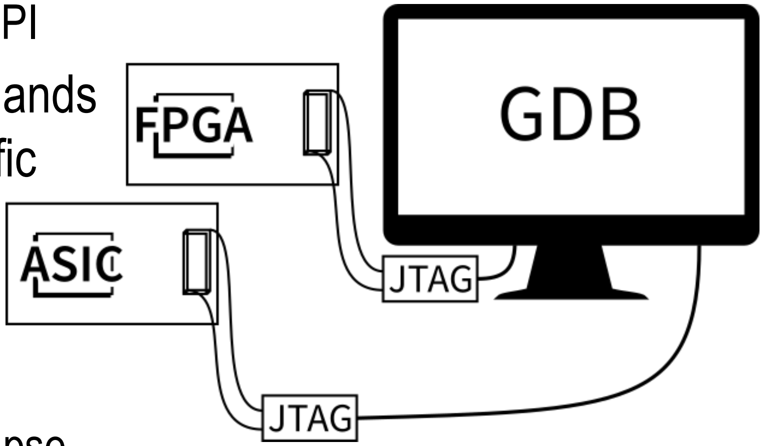
Virtual Platform: Features

- **100% functional equivalence to RTL** (or supposed to)
- **Performance estimation** (20% error margin)
- **Frequency scaling**
- Power on/off
- **Power consumption estimation**
- **Architecture traces**
- **VCD** traces
- **Peripheral** models (camera, ADC, microphone, etc)
- **GDB** connection



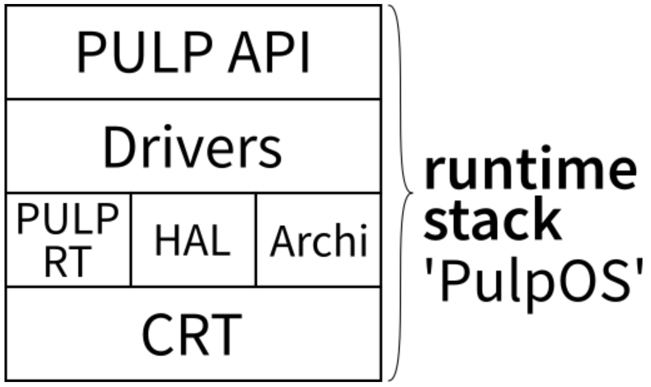
Debugger, GUI

- Access to the target is done through the **PULP debug bridge**
  - Uses memory-mapped accesses through JTAG or SPI
  - Receives **remote serial protocol** (RSP) commands from **GDB** and converts them to platform-specific commands (e.g. breakpoints)
  - Provides binary loading, chip reset and start
- For the GUI, we **support PlatformIO**
  - Plugin generator for lots of common editors/IDE (eclipse, sublime, visual studio, etc)
  - Platform described through JSON files (how to compile, connect, etc)
  - Connection done through GDB RSP protocol
  - Provides application build, code loading, target interactions, debug features, etc.



Runtime / OS

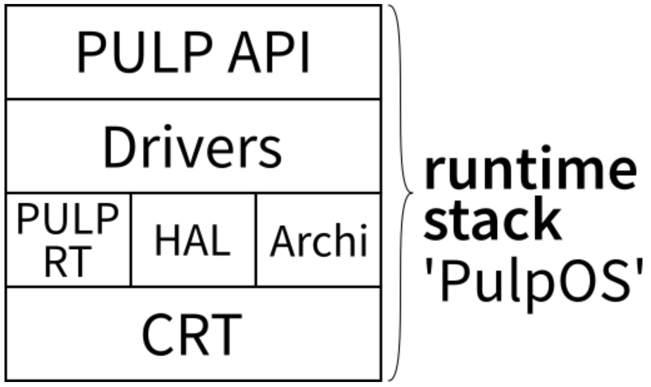
- **PulpOS**
  - Provides a simple OS for quick prototyping
  - Supports all PULP variants, with or without fabric controller (FC) and multiple clusters
  - Used for full applications including drivers, as well as basic tests
  - All APIs are asynchronous to support small reactive applications



- **Zephyr**
  - Just starting now
  - Plan is to port the kernel to PULP platforms, create new API for managing the cluster and port Zephyr drivers (SPI, etc)

PulpOS

- Features
  - **Multi-threading**: to get different priorities
  - **Event scheduler:** one per thread, to schedule run-to-completion tasks (all APIs are asynchronous)
  - **Memory allocators**: for all PULP memory levels (L2, L1)
  - **Cluster management**: cluster mount/unmount, remote cluster call, FC remote services for cluster
  - **Power management**: frequency scaling, deep sleep, voltage scaling
  - **Drivers**: SPI, I2S, I2C, CPI, etc.
  - **Cluster execution**: team fork / barriers / critical sections, DMA transfers



PULP SDK: Getting Started

```
git clone \
 https://github.com/pulp-platform/pulp-sdk
```

Check `README.md` for prerequisites and install them.

`Source` the configuration file of your target platform.

```
make all
```

Questions?

www.pulp-platform.org

@pulp_platform

PULP
Parallel Ultra Low Power

Florian Zaruba[2], Davide Rossi[1], Antonio Pullini[2], Francesco Conti[1], Michael Gautschi[2], Frank K. Gürkaynak[2], Florian Glaser[2], Stefan Mach[2], Giovanni Rovere[2], Igor Loi[1] Davide Schiavone[2], Germain Haugou[2], Manuele Rusci[1], Alessandro Capotondi[1], Giuseppe Tagliavini[1], Daniele Palossi[2], Andrea Marongiu[1,2], Fabio Montagna[1], Simone Benatti[1], Eric Flamand[2], Fabian Schuiki[2], Andreas Kurth[2], Luca Benini[1,2]

[1]*Department of Electrical, Electronic and Information Engineering*

ETH zürich
[2]*Integrated Systems Laboratory*

Agenda

- 09:00 Introduction to the **PULP Cluster** and its **Execution Model**: Software-Managed Scratchpad Memories and DMA Transfers
- 09:15 Introduction to the **PULP SDK**
- *09:30 Break*
- **09:45** Introduction to **HERO**
- **10:00 HERO Live Demo**
- *10:30 Break*
- **10:45 HERO Hands-On Programming**
- **11:45 Q&A**
- *12:00 Lunch Break*

HERO: Heterogeneous Research Platform

Open-Source HW/SW Platform for R&D of Heterogeneous SoCs

14.06.2019

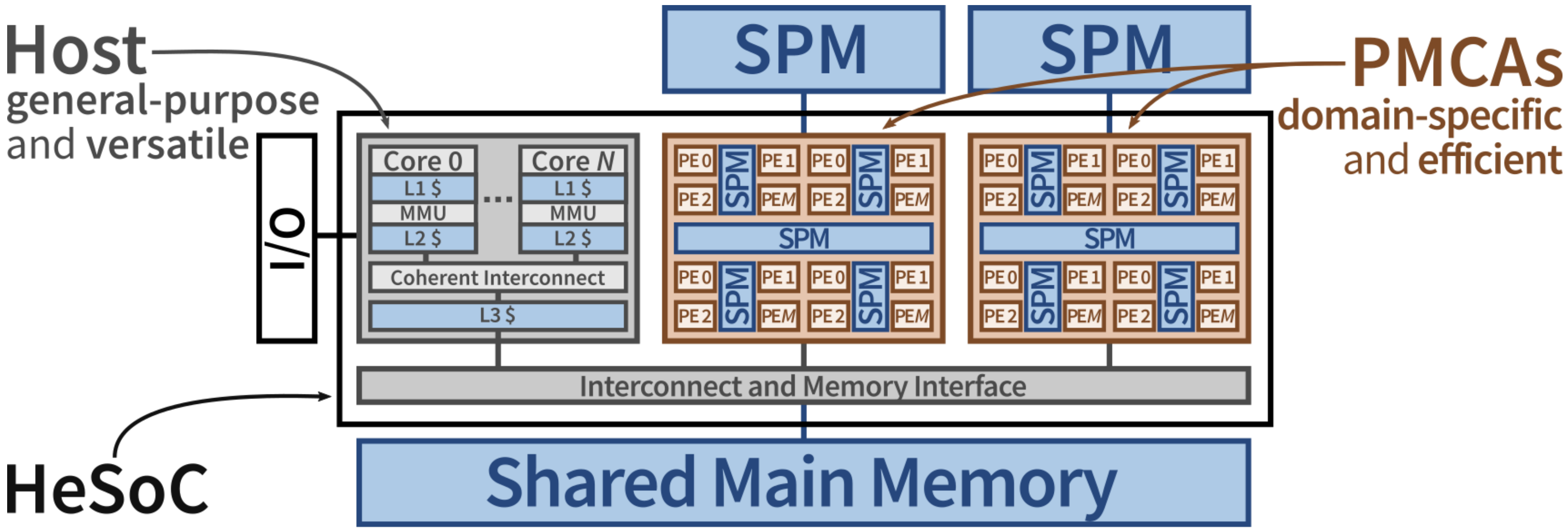
Andreas Kurth

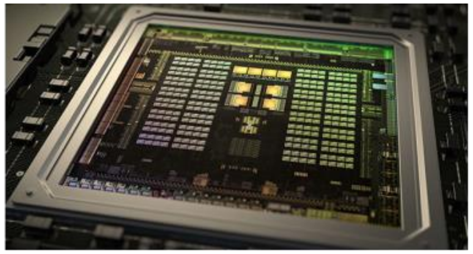
and the PULP Team led by Prof. Luca Benini

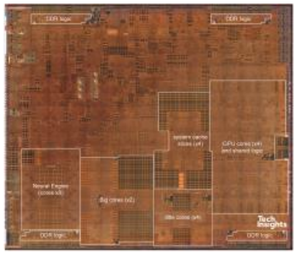
[1]Department of Electrical, Electronic and Information Engineering

[2]Integrated Systems Laboratory

Heterogeneous Systems on Chip (HeSoCs)




Nvidia Tegra X1 (source: Nvidia)


Apple A12 (source: TechInsights)

Research on Heterogeneous SoCs

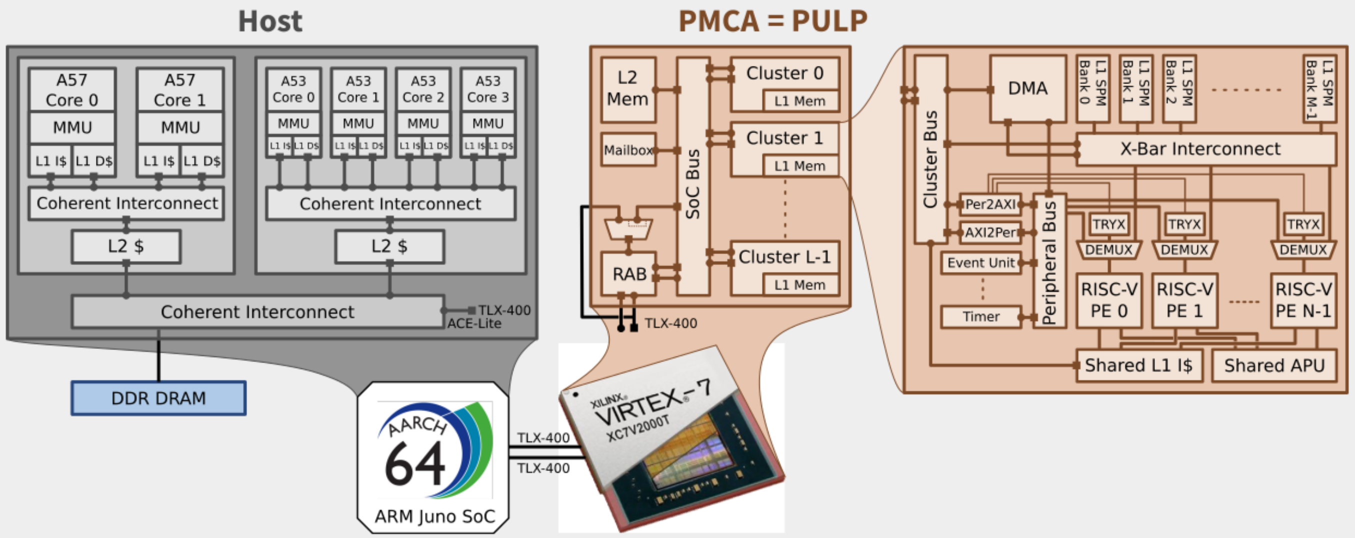
There are **many open questions** in various areas of computer engineering:

- programming models, task distribution and scheduling,
- memory organization, communication, synchronization,
- accelerator architectures and granularity, …

But there was no **research platform for heterogeneous SoCs**!

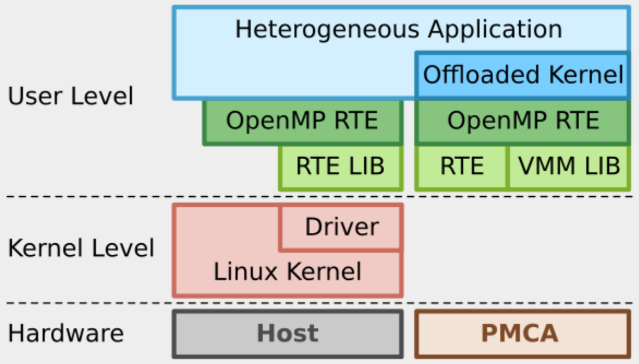
HERO: Heterogeneous Research Platform

Heterogeneous Hardware Architecture

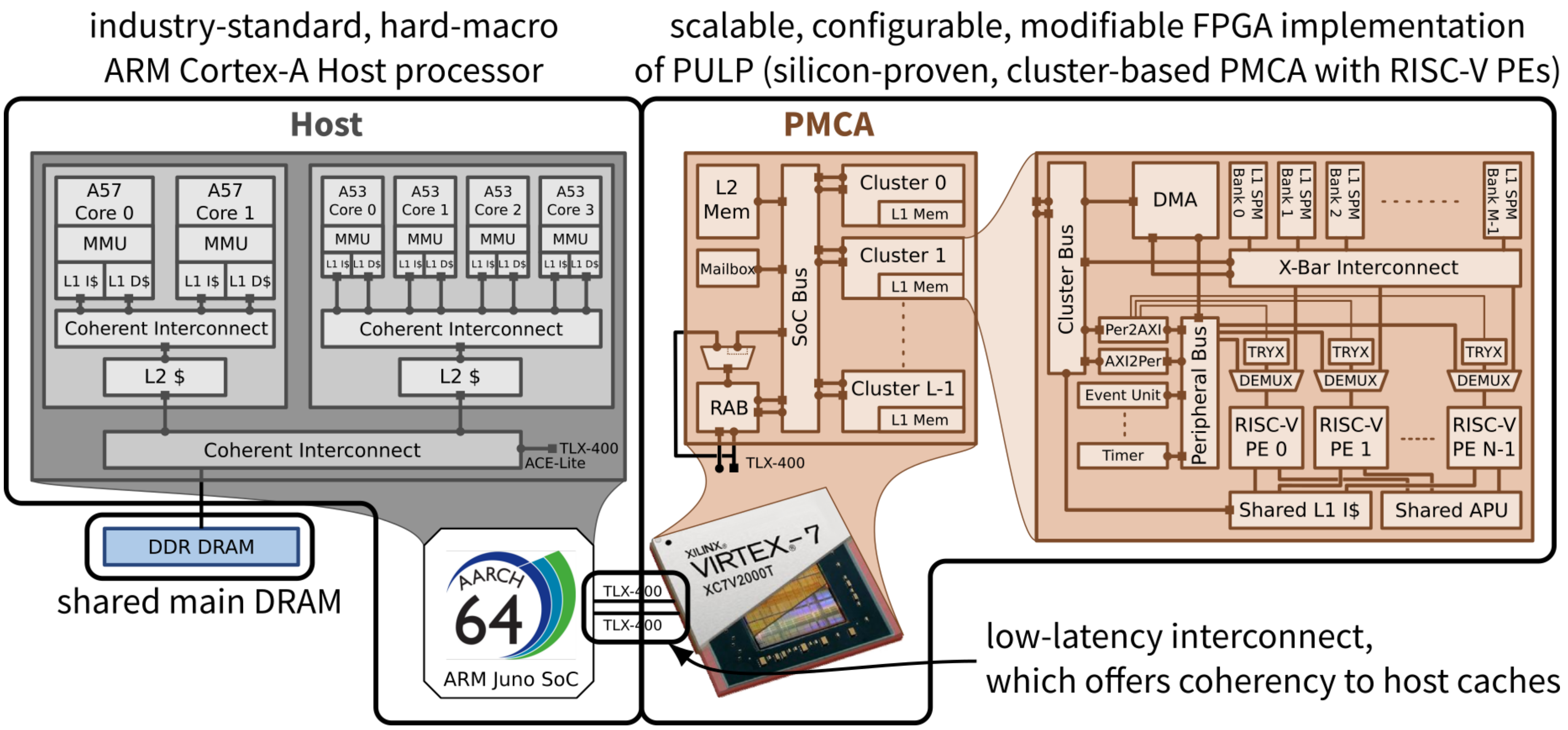


Heterogeneous Software Stack

- single-source, single-binary cross compilation toolchain
- OpenMP 4.5
- shared virtual memory for Host and PMCA

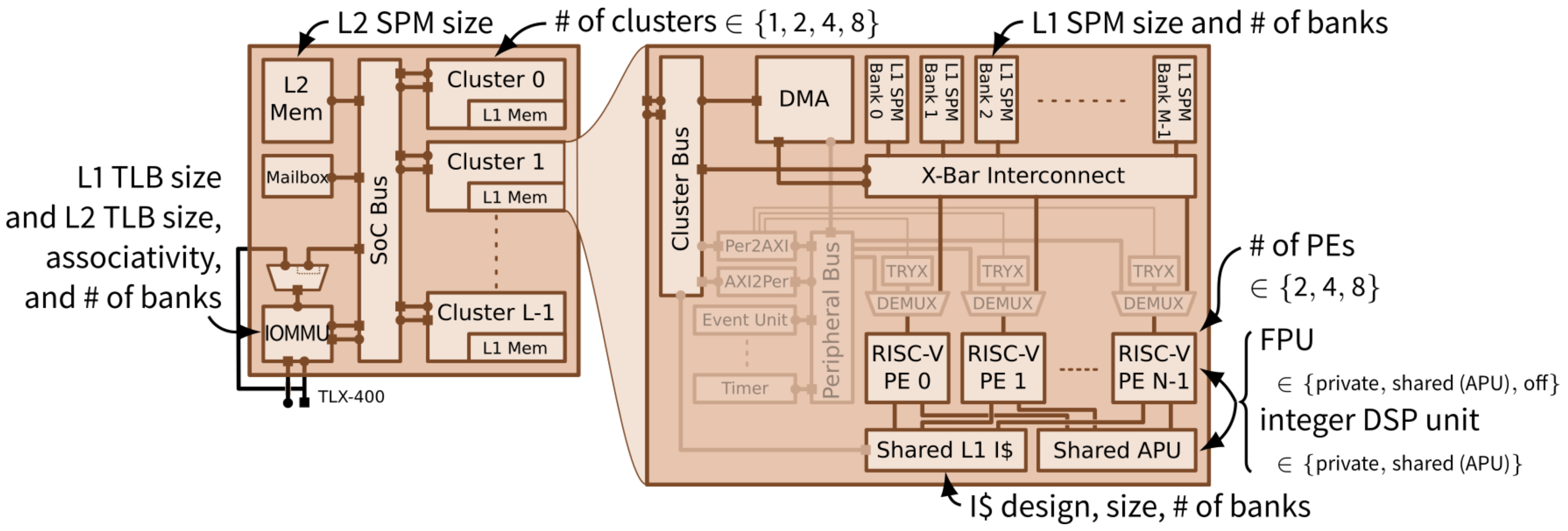


HERO: Hardware Architecture



bigPULP on FPGA: Configurable, Modifiable and Expandable

Configurable:

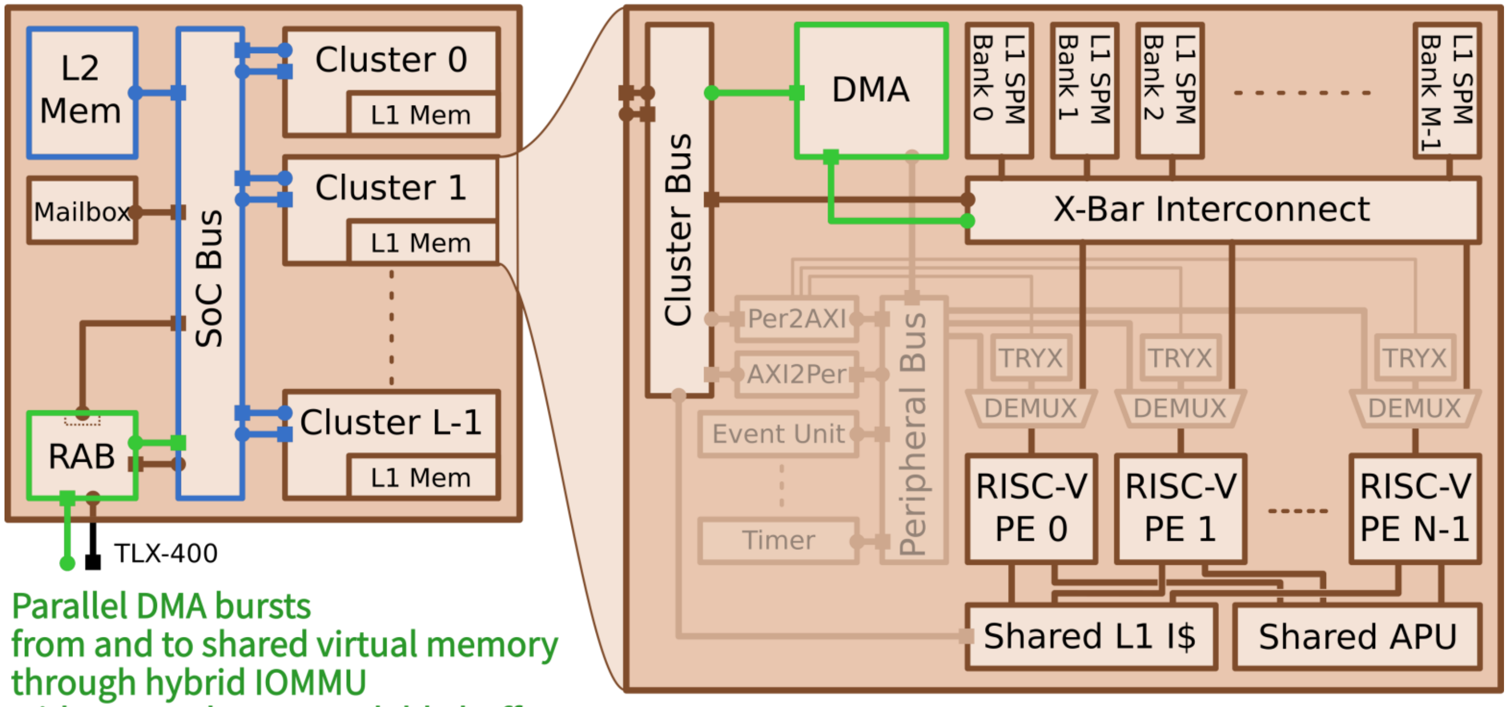


Modifiable and expandable:

- All components are open-source and written in industry-standard SystemVerilog.
- Interfaces are either standard (mostly AXI) or simple (e.g., stream-payload).
- New components can be easily added to the memory map.

bigPULP: Distinguishing Features

Scalable and efficient multi-cluster atomic transactions (RISC-V 'A' extension) to shared L2 memory



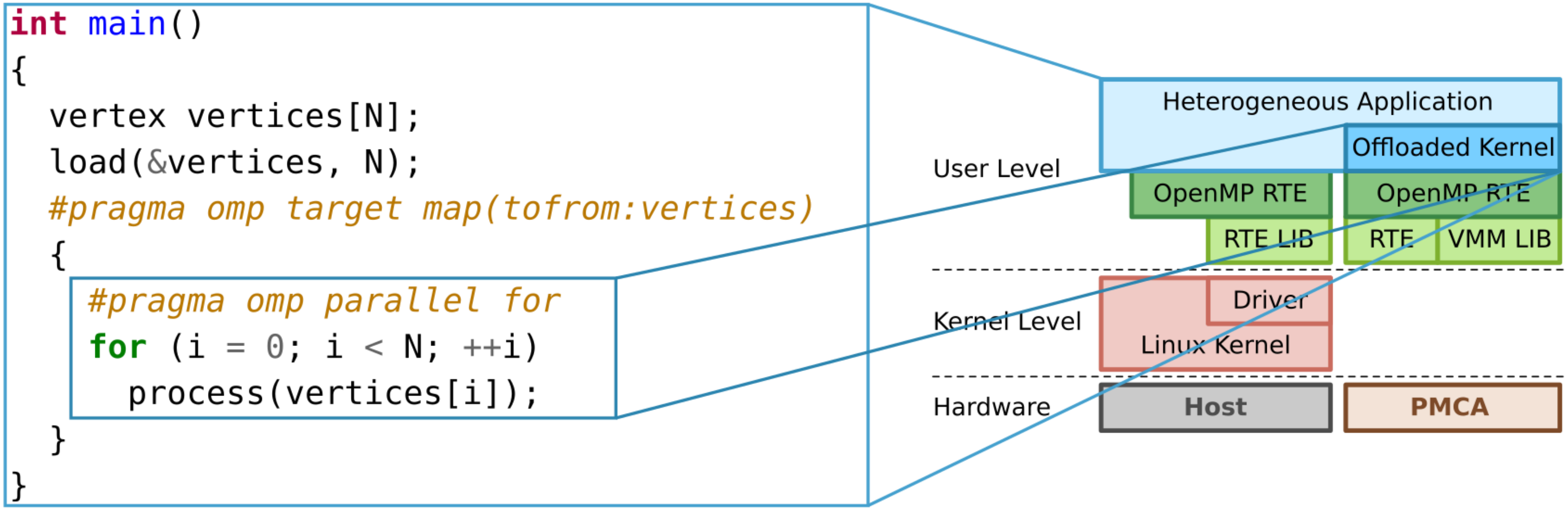
Parallel DMA bursts
from and to shared virtual memory
through hybrid IOMMU
without costly, non-scalable buffers

- Atomic transactions: RI5CY with 'A' decoder, additional signals through cluster and SoC bus, transactions executed atomically at L2 SPM
- Scalable SVM: Two-level software-managed TLB ("RAB"); TLB misses signaled back to RI5CY and DMA; handled in SW with lightweight HW support

HERO: Software Architecture

Allows to write programs that start on the host but seamlessly integrate the PMCAs.

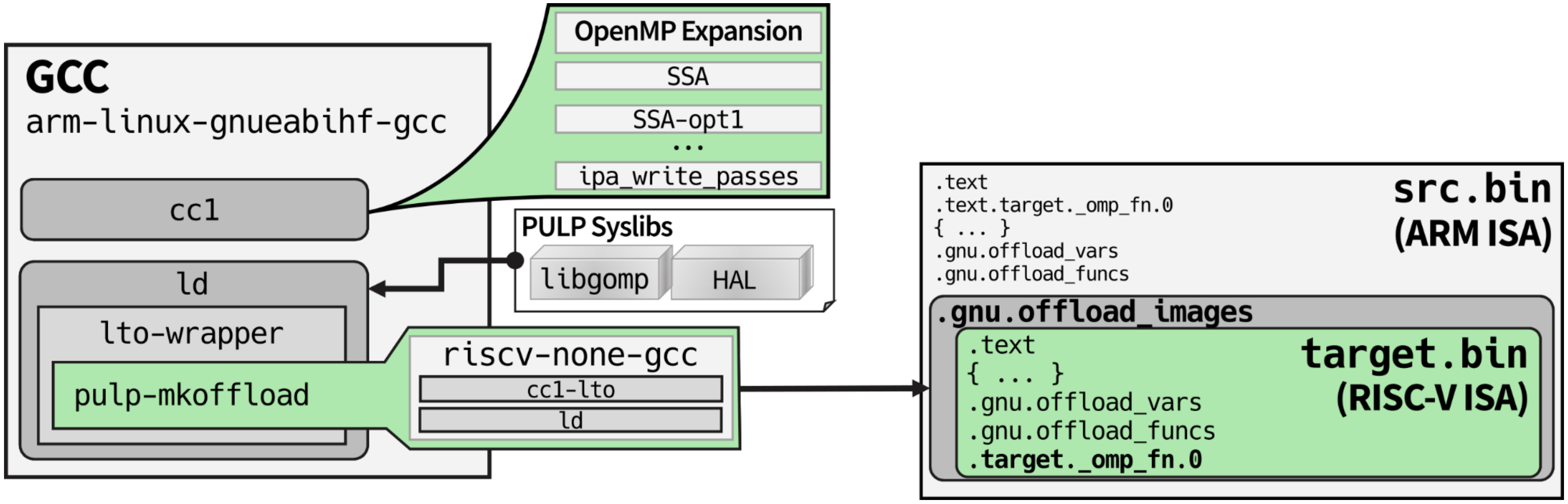
```
int main()
{
  vertex vertices[N];
  load(&vertices, N);
  #pragma omp target map(tofrom:vertices)
  {
    #pragma omp parallel for
    for (i = 0; i < N; ++i)
      process(vertices[i]);
  }
}
```



- Offloads with OpenMP 4.5 `target` semantics, zero-copy (pointer passing) or copy-based
- Integrated cross-compilation and single-binary linkage
- PMCA-specific runtime environment and hardware abstraction libraries (HAL)

HERO: Heterogeneous Cross-Compilation Toolchain

- OpenMP offloading with the GCC toolchain requires a **host compiler** plus **one target compiler for each PMCA ISA** in the system.



- A target compiler requires both **compiler and runtime extensions**.
- HERO includes the **first non-commercial** heterogeneous cross-compilation toolchain.

HERO: FPGA Platforms

Property	ARM Juno (with a Xilinx Virtex-7 2000T)	Xilinx Zynq UltraScale+ ZU9EG	Xilinx Zynq Z-4045
Host CPU	64-bit ARMv8 big.LITTLE	64-bit ARMv8 quad-core A53	32-bit ARMv7 dual-core A9
Shared main memory	8 GiB DDR3L	2 GiB DDR4	1 GiB DDR3
PMCA clock frequency	**30 MHz**	**150 MHz**	**50 MHz**
# of RISC-V PEs	**64 in 8 clusters**	**16 in 2 cluster**	**8 in 1 cluster**
Integer DSP unit	private per PE		
L1 SPM	256 KiB in 16 banks		
Instruction cache	8 KiB in 8 single-ported banks	4 KiB in 4 multi-ported banks	
Slices used by clusters	80%	63%	65%
Slices used by infrastructure	7%	15%	12%
BRAMs used by clusters	89%	55%	70%
BRAMs used by infrastructure	6%	12%	13%
Price	25 000 $	2500 $	2500 $

PULP

HERO: Roadmap

September 2018
v1.0 Public release of the world's first open-source heterogeneous hardware and software stack

October 2018
v1.1 Full support for OpenMP 4.5 API and release of example applications

February 2019
v1.2 Automatic compile-time insertion of SVM intrinsics

H1 2019
v2.0 US+ FPGAs with 64-bit hosts, support for 'F' extension with shared FPUs multi-cluster OpenMP RTE

H1 2020
v3.0 Replace ARM host processor with multi-core Ariane → world's first fully open-source HeSoC

HERO: Getting Started

```
git clone --recursive \
 https://github.com/pulp-platform/hero-sdk
cd hero-sdk; git checkout v1.1.0
```

Check `README.md` for prerequisites and install them.

```
./hero-z-7045-builder -A
```

Questions?

www.pulp-platform.org

@pulp_platform

PULP
Parallel Ultra Low Power

Florian Zaruba[2], Davide Rossi[1], Antonio Pullini[2], Francesco Conti[1], Michael Gautschi[2], Frank K. Gürkaynak[2], Florian Glaser[2], Stefan Mach[2], Giovanni Rovere[2], Igor Loi[1] Davide Schiavone[2], Germain Haugou[2], Manuele Rusci[1], Alessandro Capotondi[1], Giuseppe Tagliavini[1], Daniele Palossi[2], Andrea Marongiu[1,2], Fabio Montagna[1], Simone Benatti[1], Eric Flamand[2], Fabian Schuiki[2], Andreas Kurth[2], Luca Benini[1,2]

[1]*Department of Electrical, Electronic and Information Engineering*

ETH zürich
[2]*Integrated Systems Laboratory*

Agenda

09:00	Introduction to the **PULP Cluster** and its **Execution Model**: Software-Managed Scratchpad Memories and DMA Transfers
09:15	Introduction to the **PULP SDK**
09:30	*Break*
09:45	Introduction to **HERO**
10:00	**HERO Live Demo**
10:30	*Break*
10:45	**HERO Hands-On Programming**
11:45	**Q&A**
12:00	*Lunch Break*

Agenda

09:00	Introduction to the **PULP Cluster** and its **Execution Model**: Software-Managed Scratchpad Memories and DMA Transfers
09:15	Introduction to the **PULP SDK**
09:30	*Break*
09:45	Introduction to **HERO**
10:00	**HERO Live Demo**
10:30	*Break*
10:45	**HERO Hands-On Programming**
11:45	**Q&A**
12:00	*Lunch Break*

Agenda

- **09:00** Introduction to the **PULP Cluster** and its **Execution Model**: Software-Managed Scratchpad Memories and DMA Transfers
- **09:15** Introduction to the **PULP SDK**
- *09:30* *Break*
- **09:45** Introduction to **HERO**
- **10:00** **HERO Live Demo**
- *10:30* *Break*
- **10:45** **HERO Hands-On Programming**
- **11:45** **Q&A**
- *12:00* *Lunch Break*

HERO Hands-On Programming

- Log in to the workstation in front of you with the provided account.
- Open a shell and:
  - `cd /scratch/hero-sdk`
  - `source scripts/hero-z-7045-env.sh`
  - `cd hero-openmp-examples`
- There are a couple of example apps, but in particular:
  - **`mm-demo`** contains the final code that we discussed in the demo
  - **`sobel-filter`** is a sample application for you to work on

HERO Hands-On Programming: Sobel Filter

- `cd sobel-filter`
- Open `main.c` and `sobel.c` in the `src` directory with your favourite editor, e.g.,
  `atom src/main.c src/sobel.c`
- In the main function, you will find

  ```
  #pragma omp target map(...)
  sobelFilter(...);
  ```

  so the entry function for PULP is `sobelFilter`.
- Execute `make` then `make run` in the shell. Since we all share one HERO board, you have to wait to get a slot.
- It takes around 20s for PULP to process the image (do not despair, you will soon improve this by 100x). Compare `input.png` and `output.png`.

HERO Hands-On Programming: Sobel Filter Parallel & DMA

- All functions are currently executed by only one RISC-V core in the PULP cluster.
- Your first task is to parallelize execution:
  Find the three outer `for` loops executed in sobelFilter and parallelize them with
  `#pragma omp parallel for`
- All eight cores now work in parallel. What is the speed-up?
- The cores currently operate on data in the DRAM instead of the L1 memory.
- Your second task is to allocate local buffers and transfer data with the DMA to and from L1 memory. Use the following functions at the start and end of `sobelFilter`:
  `hero_l1malloc``(int unsigned n_bytes)`
  `hero_dma_memcpy``(void* dst, void* src, int unsigned n_bytes)`
- What is the speed-up by properly using local memories?

Questions?

www.pulp-platform.org

@pulp_platform

Florian Zaruba[2], Davide Rossi[1], Antonio Pullini[2], Francesco Conti[1], Michael Gautschi[2], Frank K. Gürkaynak[2], Florian Glaser[2], Stefan Mach[2], Giovanni Rovere[2], Igor Loi[1] Davide Schiavone[2], Germain Haugou[2], Manuele Rusci[1], Alessandro Capotondi[1], Giuseppe Tagliavini[1], Daniele Palossi[2], Andrea Marongiu[1,2], Fabio Montagna[1], Simone Benatti[1], Eric Flamand[2], Fabian Schuiki[2], Andreas Kurth[2], Luca Benini[1,2]

[1]*Department of Electrical, Electronic and Information Engineering*

ETH zürich
[2]*Integrated Systems Laboratory*