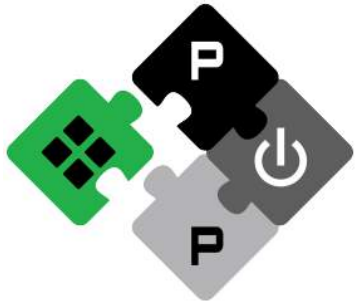


PULP PLATFORM

Open Source Hardware, the way it should be!

Working with RISC-V

Part 3 of 4 : PULP concepts

Luca Benini <lbenini@iis.ee.ethz.ch>

Frank K. Gürkaynak <kgf@ee.ethz.ch>

ETH zürich

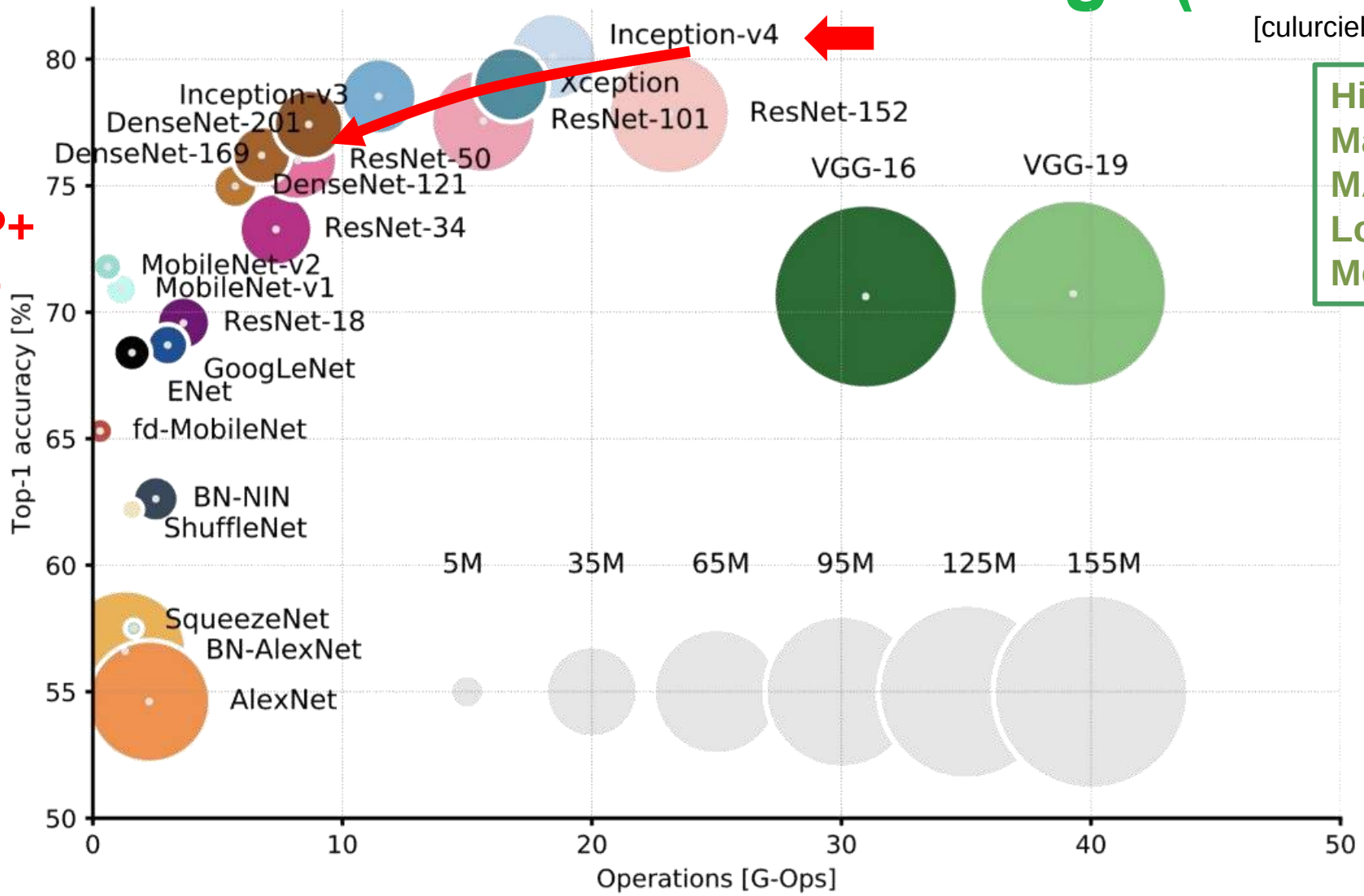
 http://pulp-platform.org  @pulp_platform  https://www.youtube.com/pulp_platform

Summary

- Part 1 – Introduction to RISC-V ISA
- Part 2 – Advanced RISC-V Architectures
- **Part 3 – PULP concepts**
  - Cores
  - Clusters
  - Heterogeneous systems
  - Accelerators
- Part 4 – PULP based chips

AI Workloads from Cloud to Edge (Extreme?)

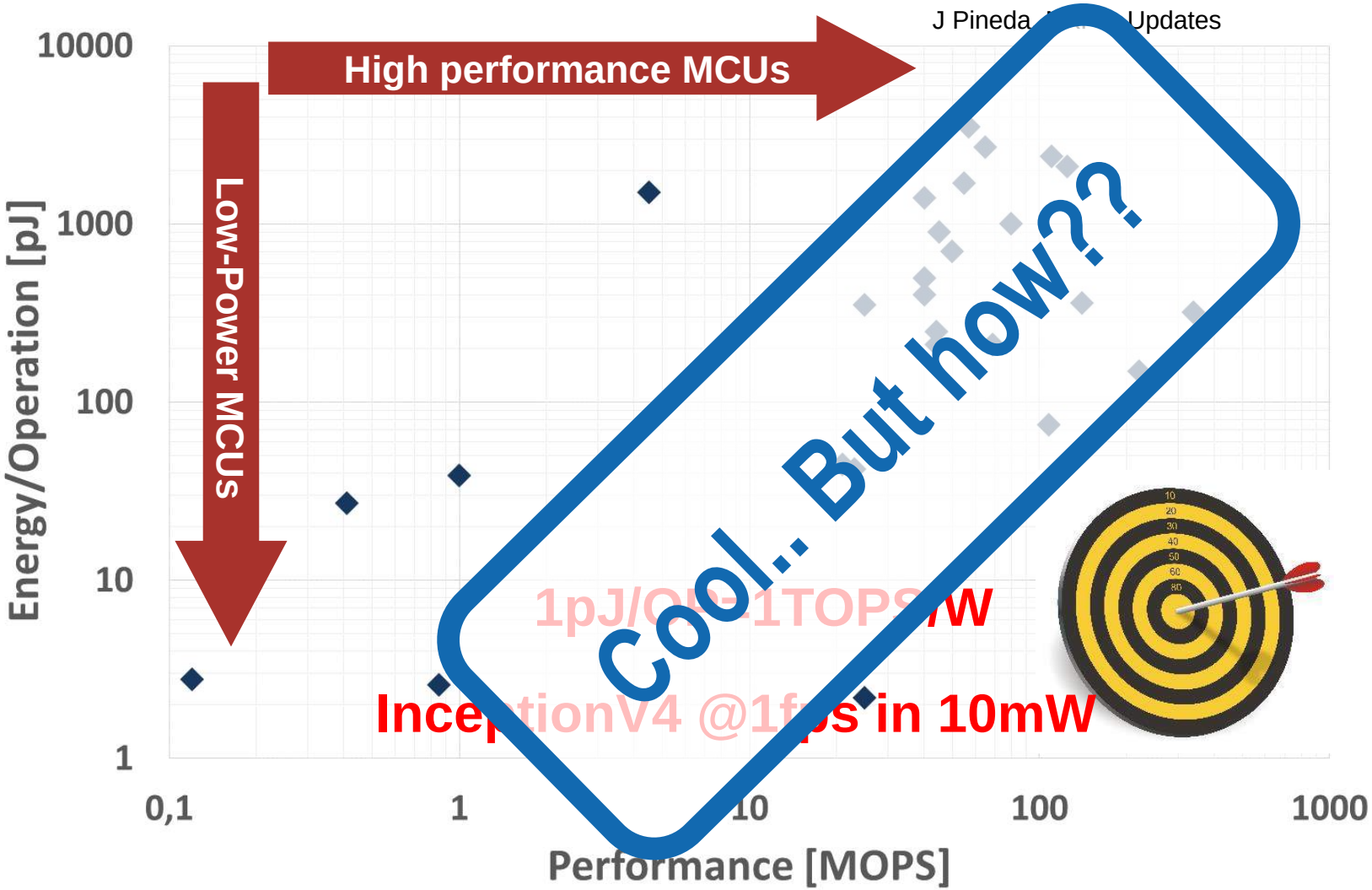
[culurciello18]



GOP+
MB+

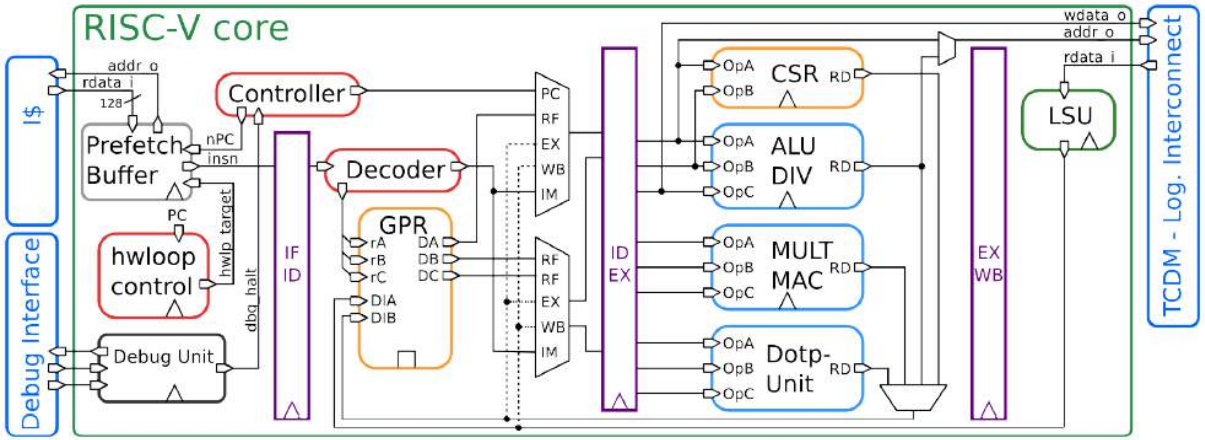
- High OP/B ratio
- Massive Parallelism
- MAC-dominated
- Low precision OK
- Model redundancy

Energy efficiency @ GOPS is THE Challenge



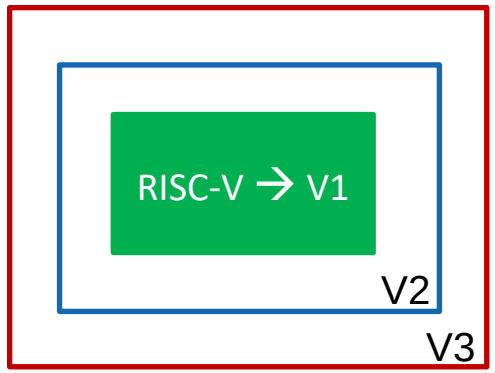
RI5CY – Recap from Part 1

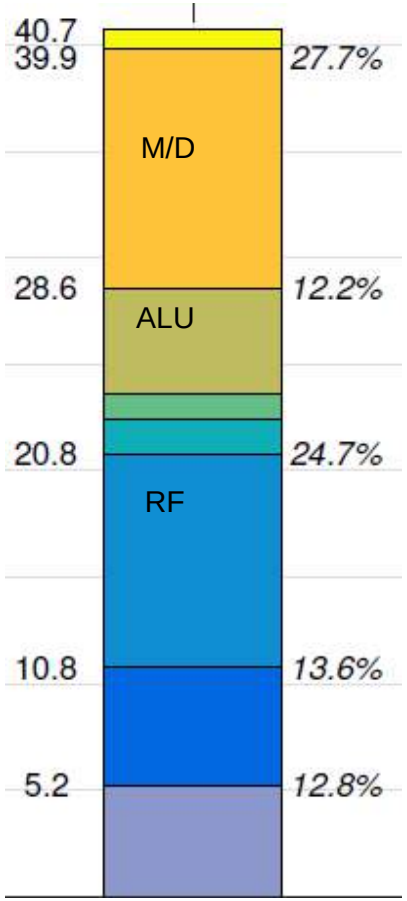
3-cycle ALU-OP, 4-cyle MEM-OP→IPC loss: LD-use, Branch



40 kGE
70% RF+DP

RISC-V core





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

V2 HW loops, Post modified Load/Store, Mac

V3 SIMD 2/4 + DotProduct + Shuffling
Bit manipulation, Lightweight fixed point

XPULP 25 kGE → 40 kGE (1.6x) but 9+ times DSP!

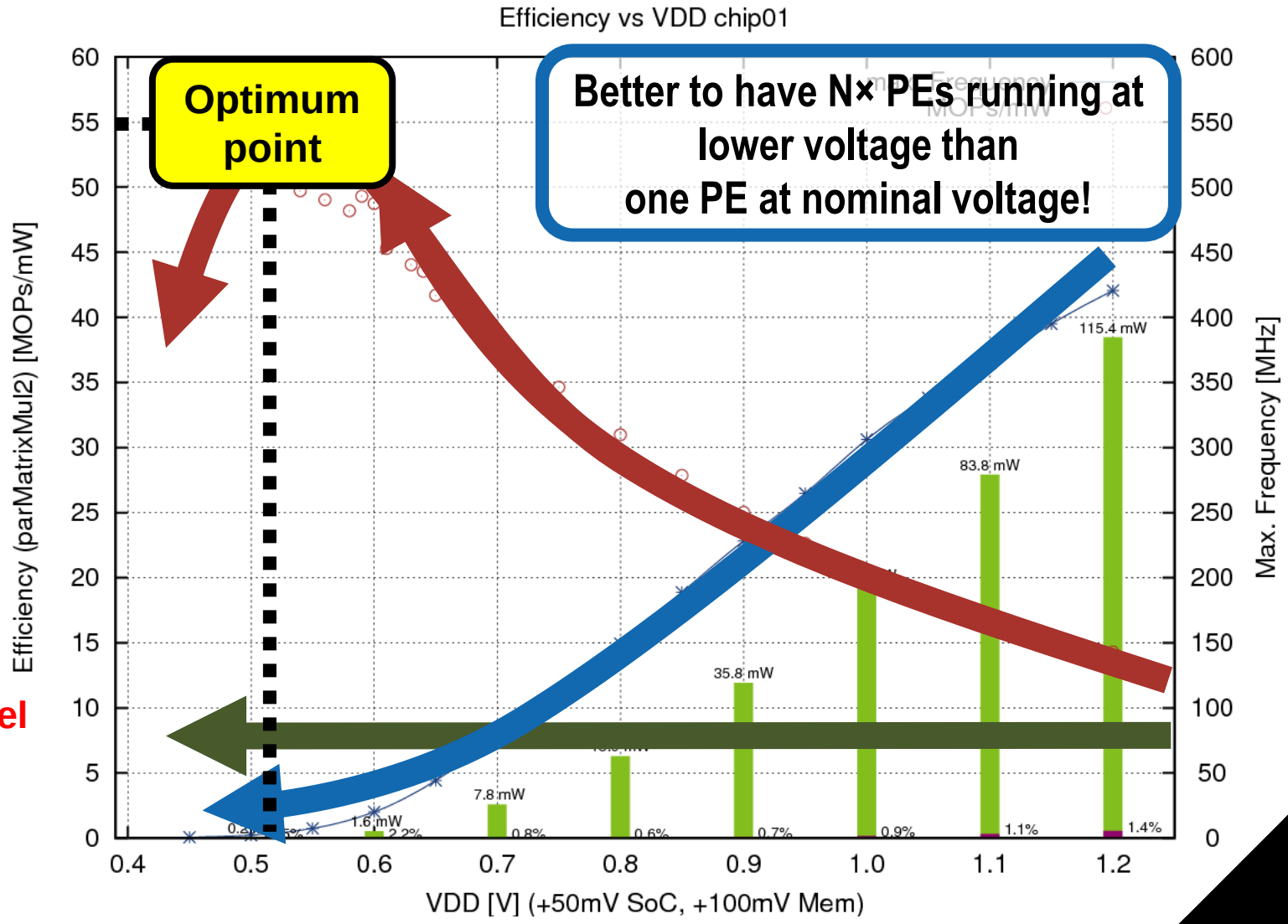
Nice – But what about the GOPS?
Faster+Superscalar is not efficient!

M7: 5.01 CoreMark/MHz-58.5 µW/MHz
M4: 3.42 CoreMark/MHz-12.26 µW/MHz

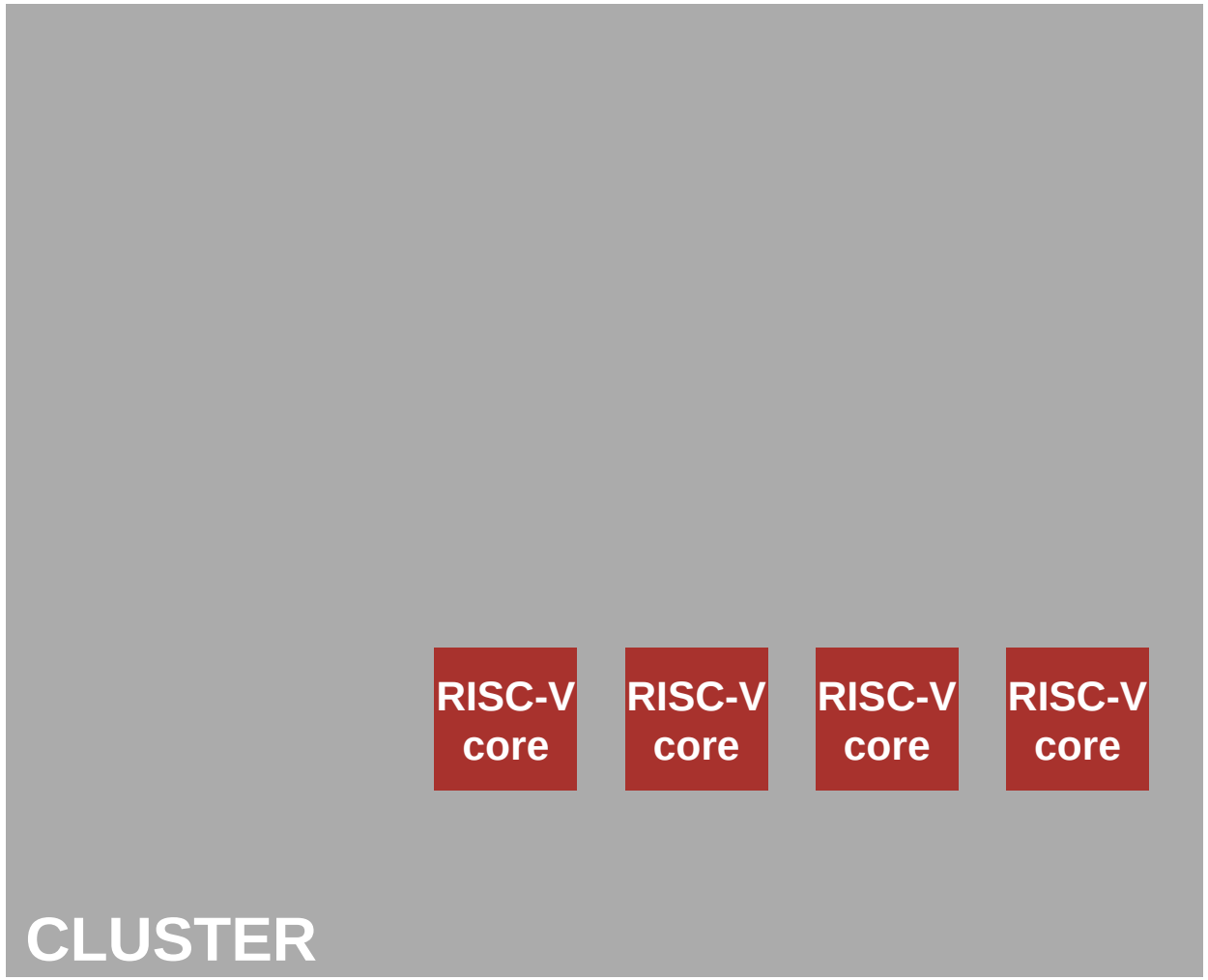
ML & Parallel, Near-threshold: a Marriage Made in Heaven

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

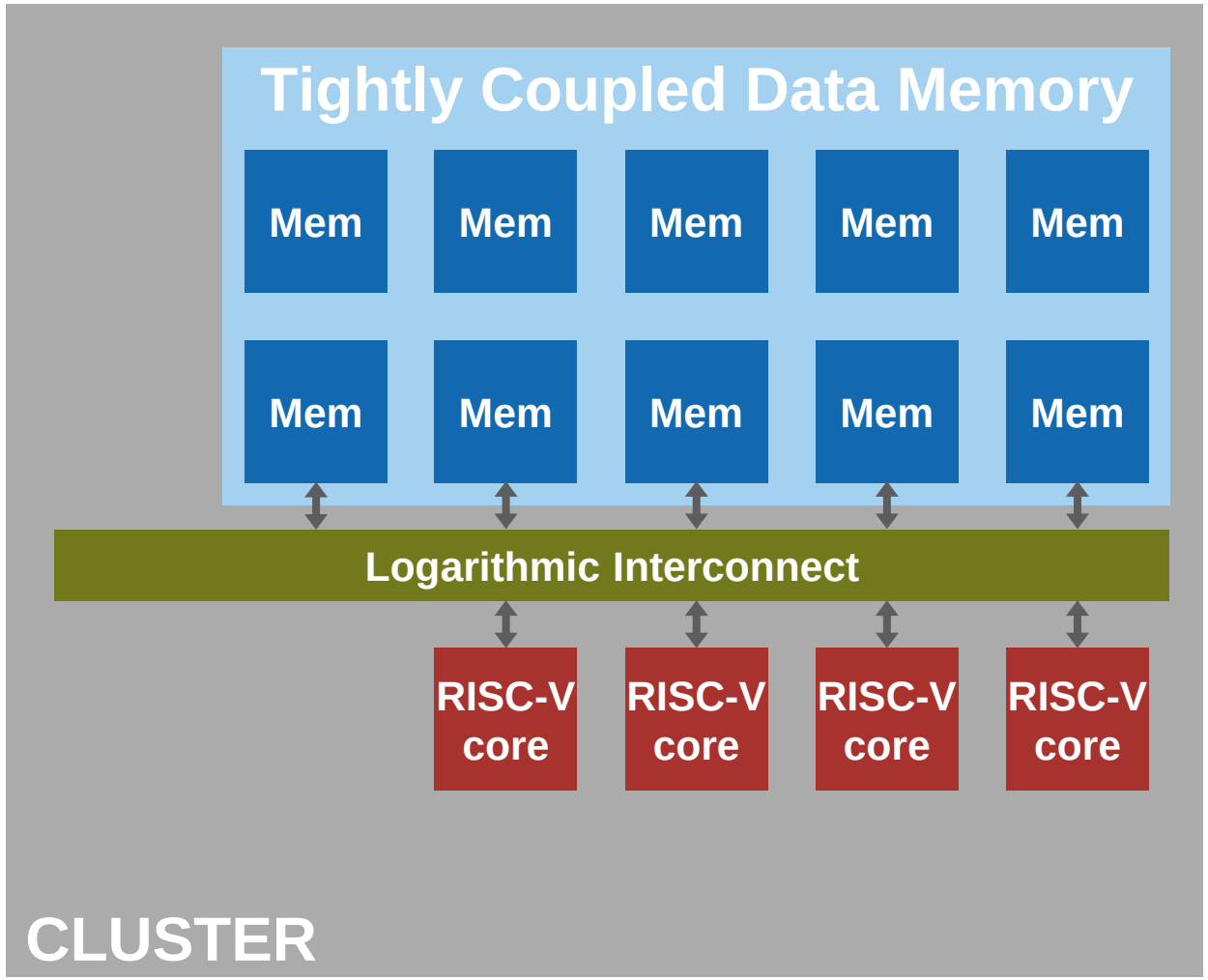
ML is massively parallel and scales well (P/S ↑ with NN size)



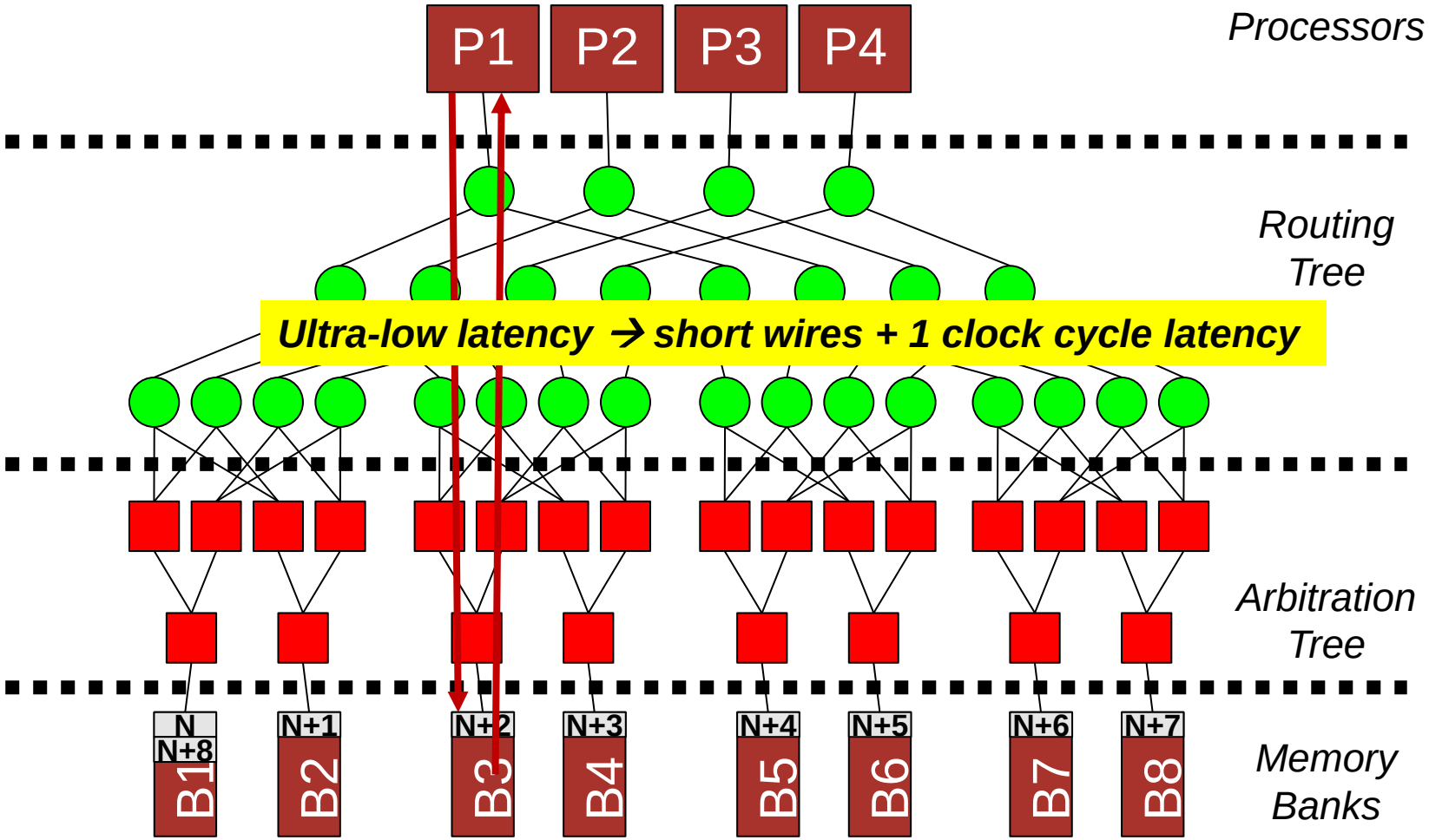
Multiple RI5CY Cores (1-16)



Low-Latency Shared TCDM

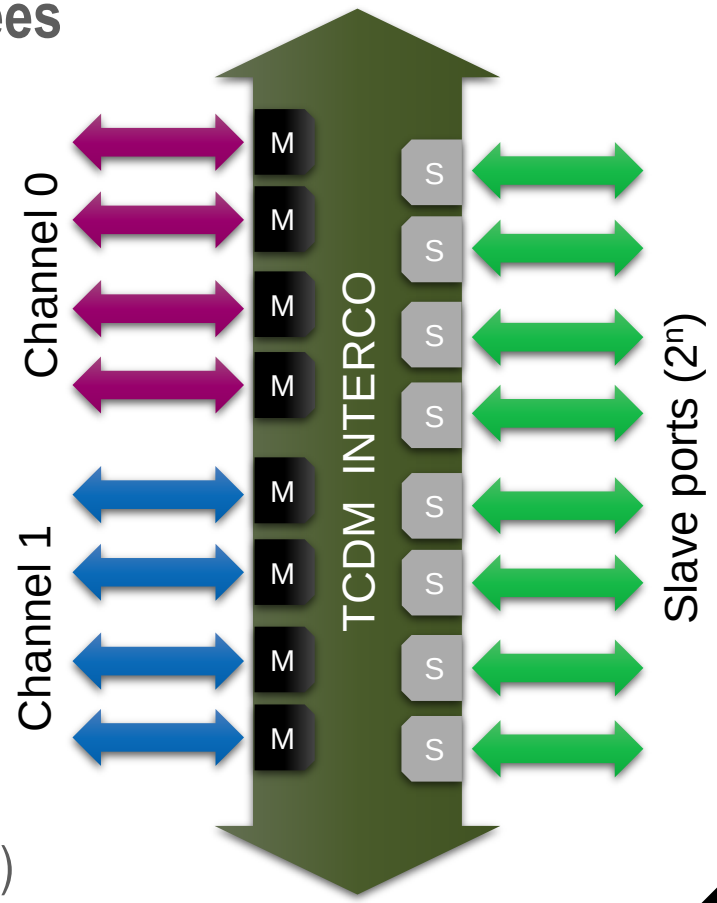


High speed single clock logarithmic interconnect



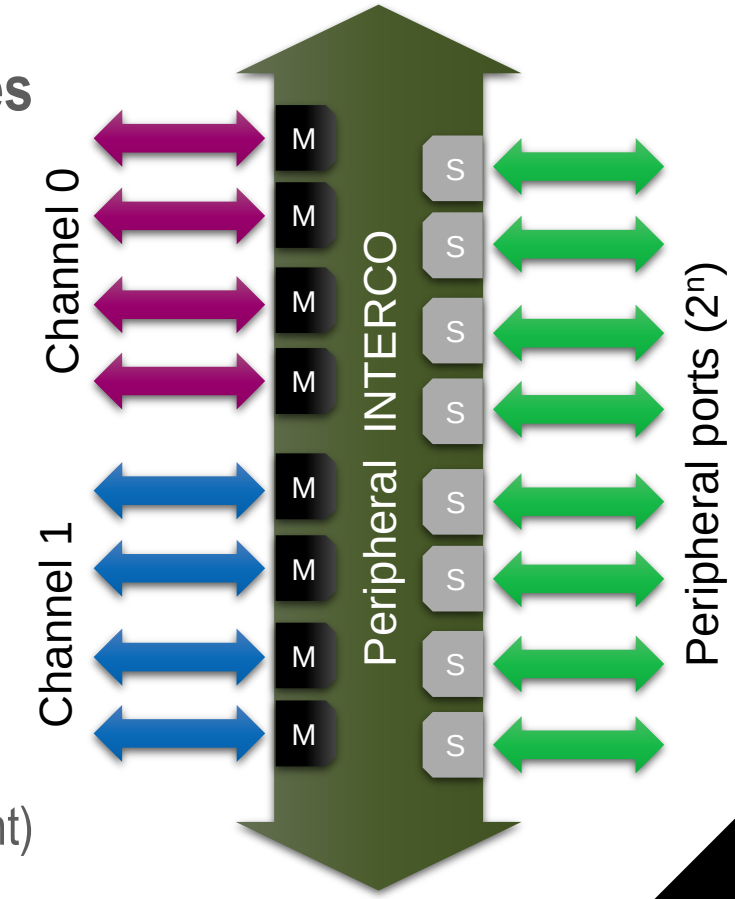
TCDM Interconnect

- **Synchronous low latency crossbar based on binary trees**
  - Single or dual channel (Each with 2^n master ports, n=[0,1,2 ...]
  - **Distributed Arbitration** (Round robin)
  - Combinational handshake (single phase)
  - Deterministic access latency (1 cycle) + response in second cycle
  - Test&Set Supported
  - Word-level Interleaving
- **Emulates Multiported RAMs**
  - Slaves are pure memories.
  - Bank conflicts can be alleviated → higher Banking Factor (BF=1,2,4)



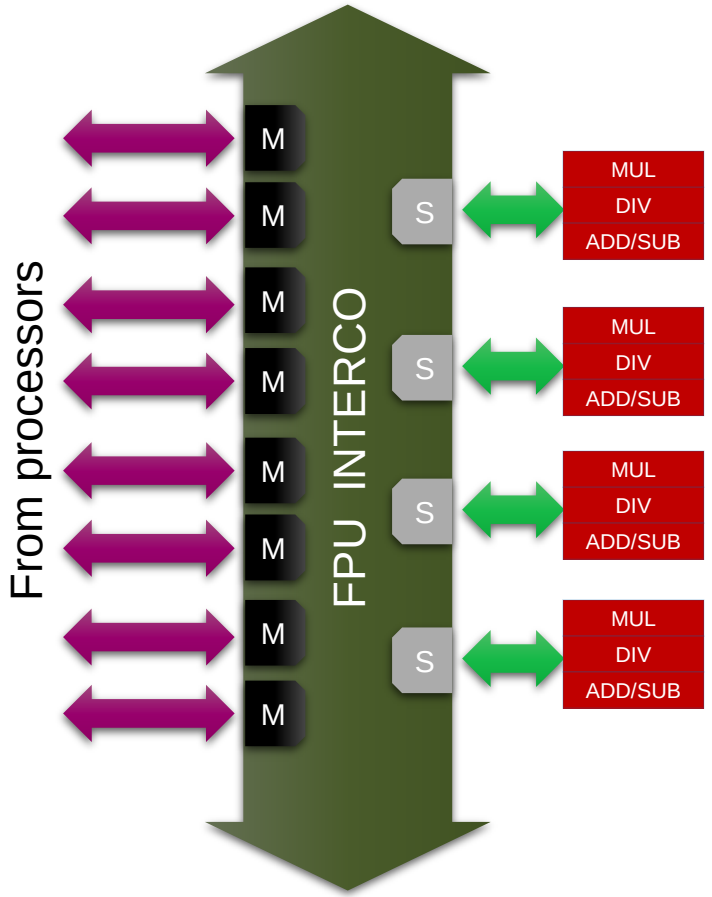
Peripheral Interconnect

- **Synchronous low latency crossbar based on binary trees**
  - Single or dual channel Each with 2n master ports, n=[0,1,2 ...]
  - Distributed Arbitration (Round robin)
  - Combinatorial handshake (single phase)
  - Custom port mapping (Address ranges)
  - Decoupled request and response path
    - Slaves have unpredictable latencies
- **Used to build Peripheral systems**
  - Slaves are pure generic peripherals like bridges , timers etc (Req/grant)
  - Mostly used to move data in and out from the cluster processors



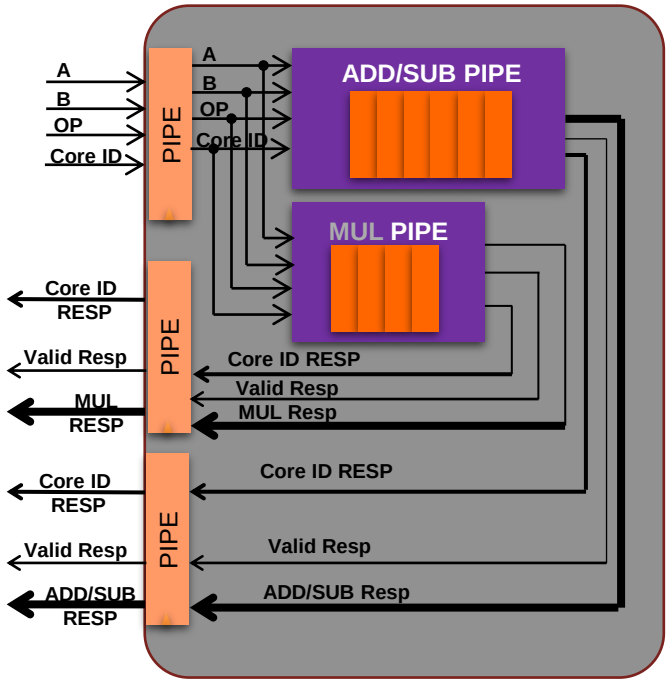
FPU Interconnect (1/2)

- **Synchronous low-power/low latency crossbar used to share several FPUs in a multicore system:**
  - 2^n master ports (n=[0,1,2 ...])
  - 2^m slave ports (m=[0,1,2 ...]) → General purpose FPUs(ADD/SUB, MUL etc)
  - Combinatorial handshake (single phase)
  - **Allocator**
    - **Random: given a request, a random fpu is choosen**
    - **Optimal: Maxime the utilization of FPUs**

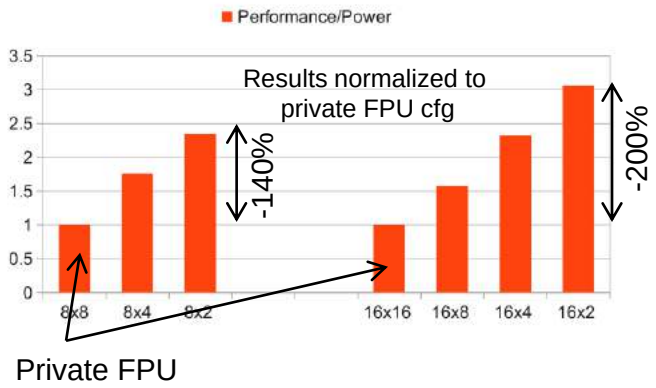


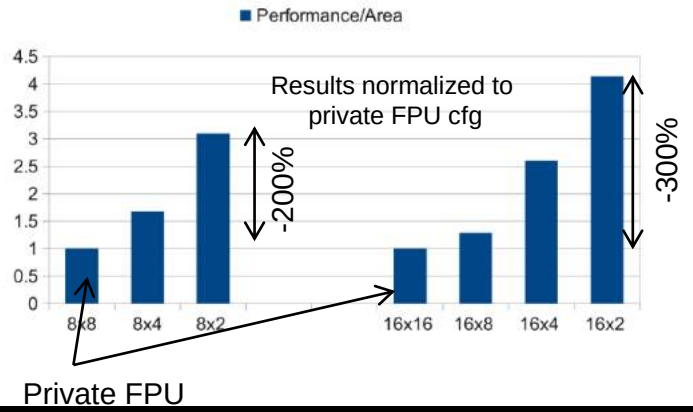
FPU Interconnect

- **Features:**
  - Independent response paths for each sub-FPU block
    - fully pipelined FPU sub-blocks with different latencies
  - Two Operators (A,B), one command (OP) and the ID are carried to the FPU.
  - No need of Flow control on the FPU side.
  - Flexible and parametrizable

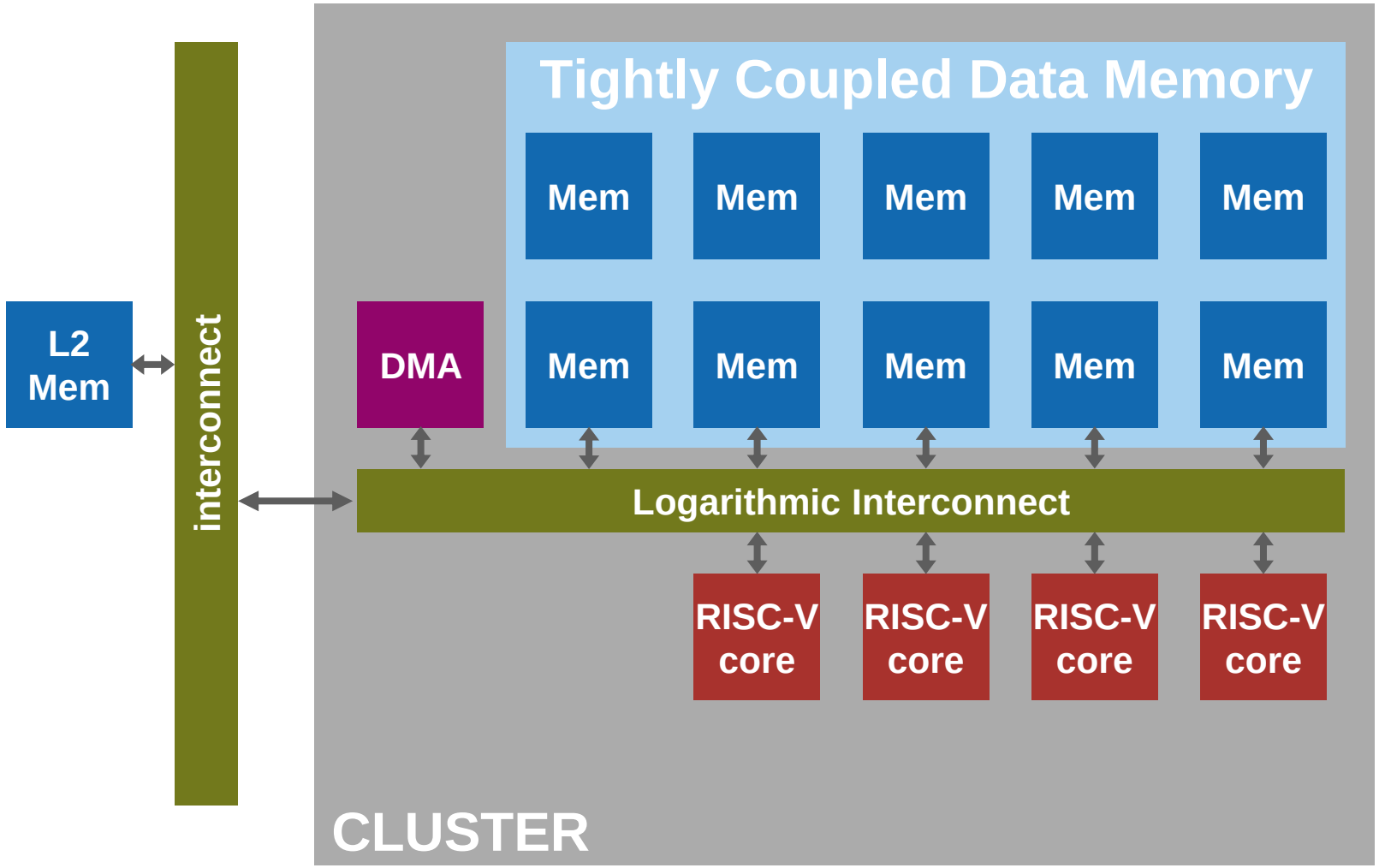


Example of FPU attached to the FPU interconnect



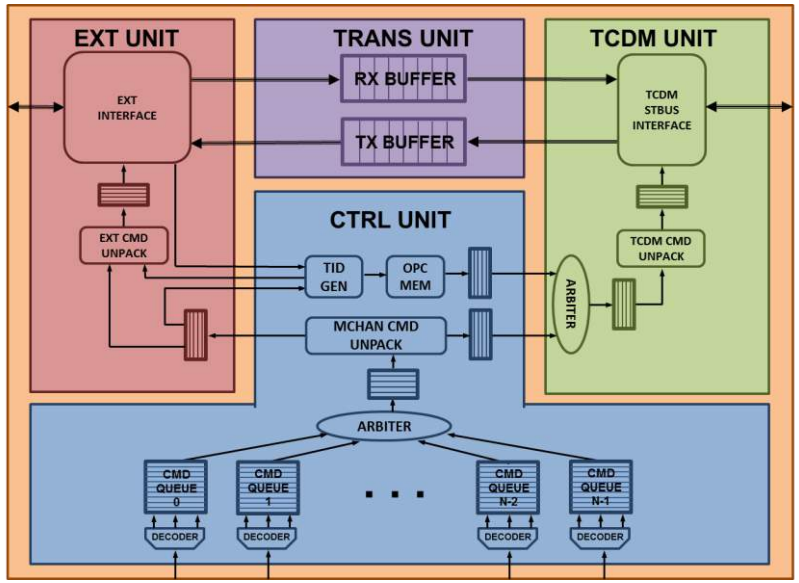


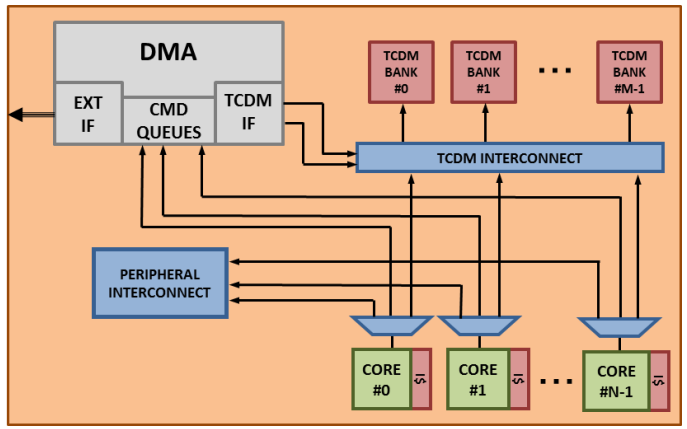
DMA for data transfers from/to L2



PULP MCHAN DMA Engine

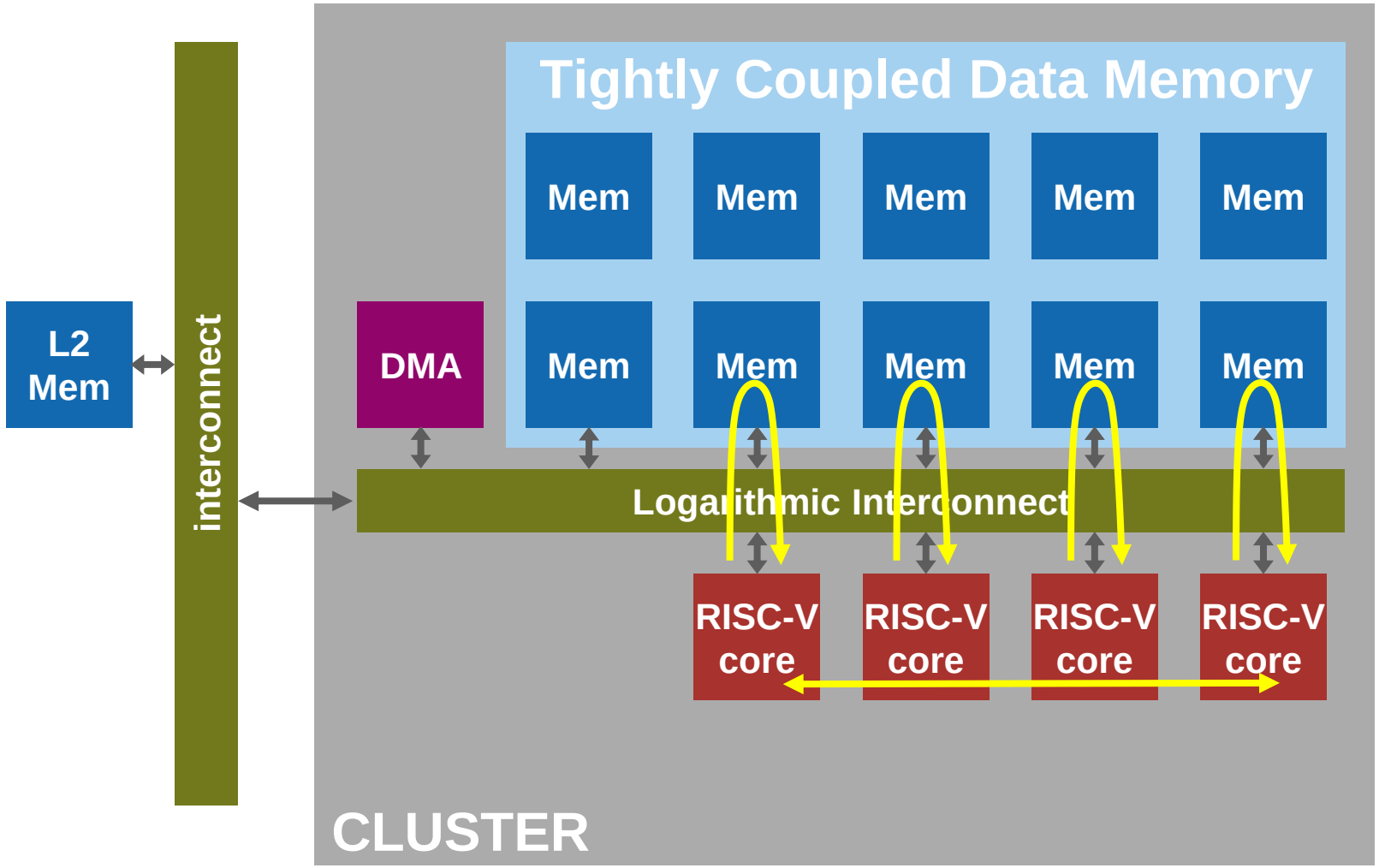
- **A DMA engine optimized for integration in tightly coupled processor clusters**
  - Dedicated, per-core non blocking programming channels
  - Ultra low latency programming (~10 cycles)
  - Small footprint (30Kgates) : avoid usage of large local FIFOs by forwarding data directly to TCDM (no store and forward)
  - Support for multiple outstanding transactions
  - Parallel RX/TX channels allow achieving full bandwidth for concurrent load and store operations
- **Configurable parameters:**
  - # of core channels
  - Size of command queues
  - Size of RX/TX buffer
  - # of outstanding transactions



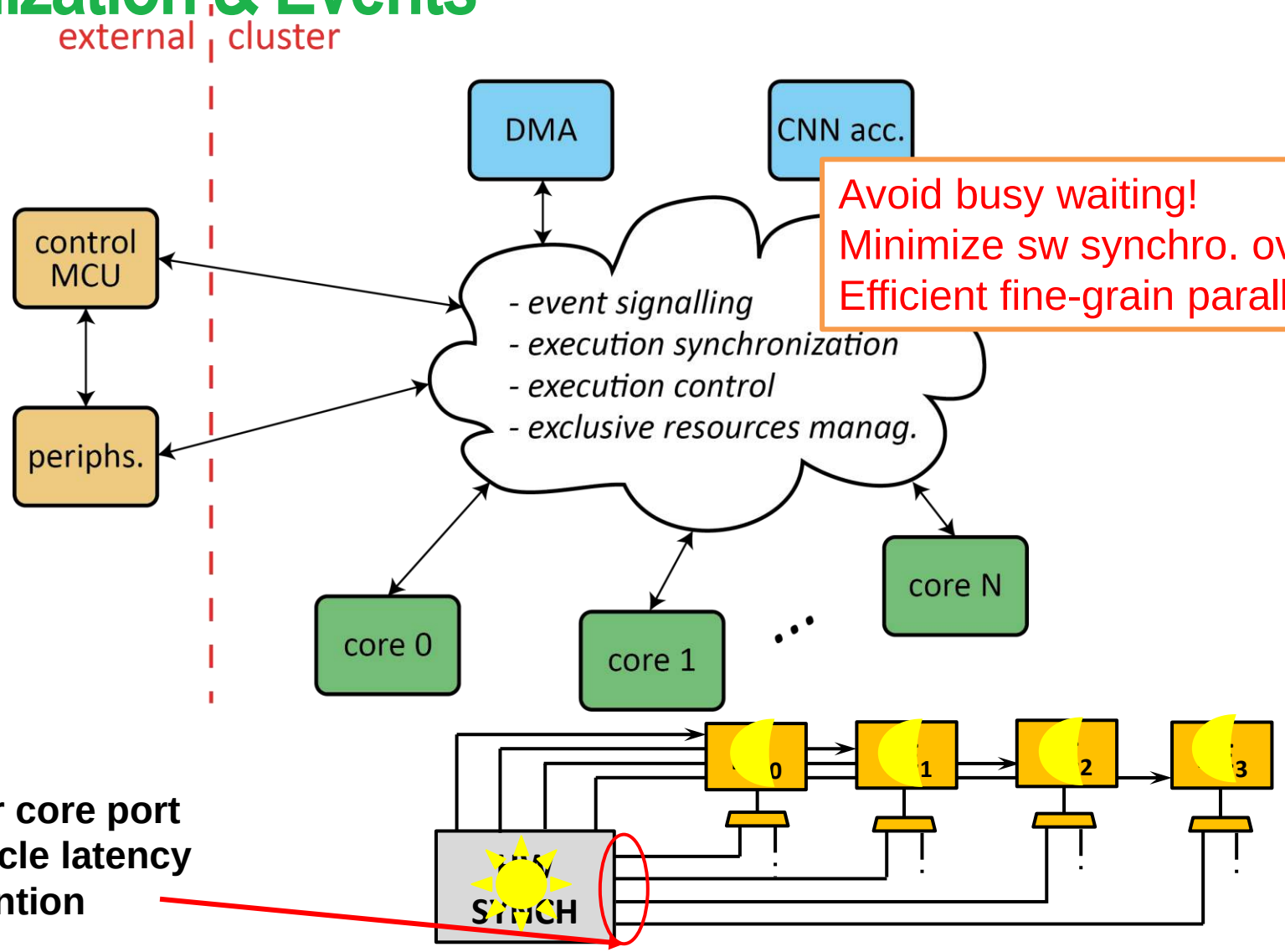


Integration in a PULP cluster

Fast synchronization and Atomics



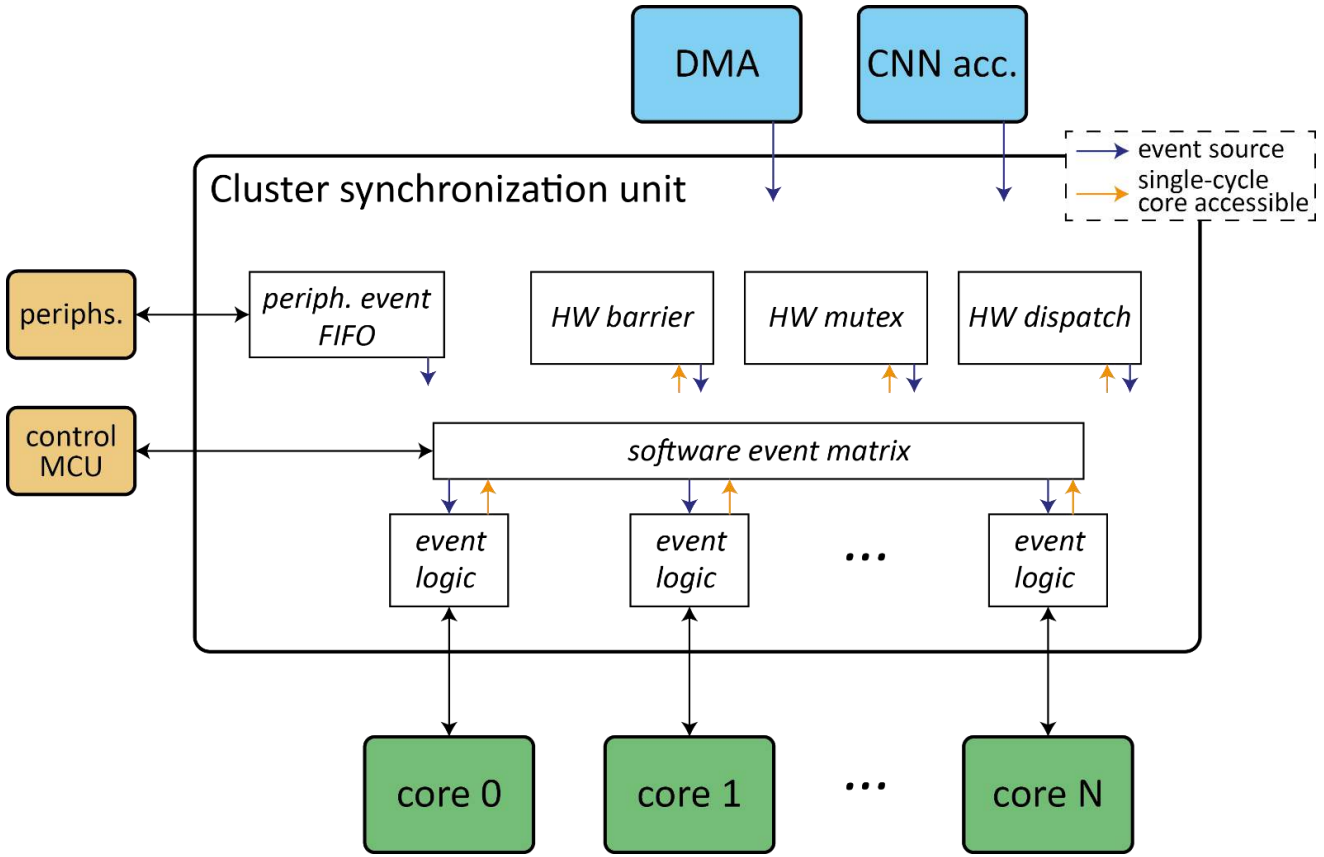
Synchronization & Events



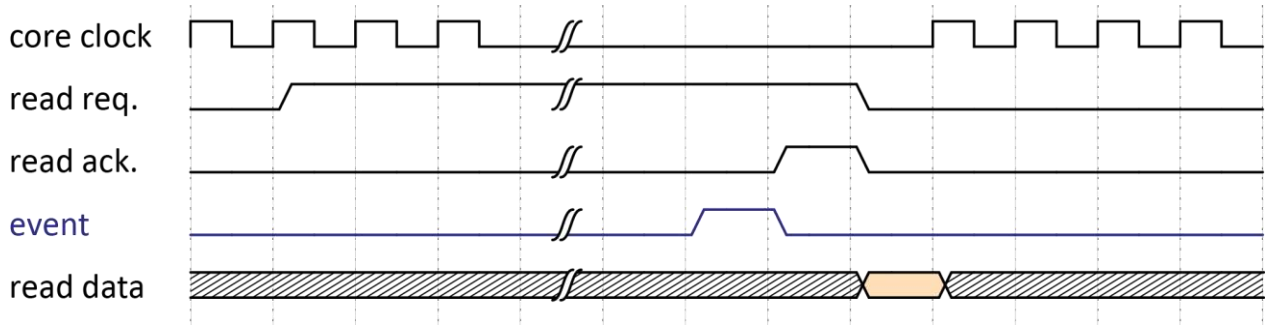
Avoid busy waiting!
Minimize sw synchro. overhead
Efficient fine-grain parallelization

Private, per core port
→single cycle latency
→no contention

PULP Cluster Event Unit



Energy-efficient Event Handling



instruction turns off core clock
→ NO pipeline flush!

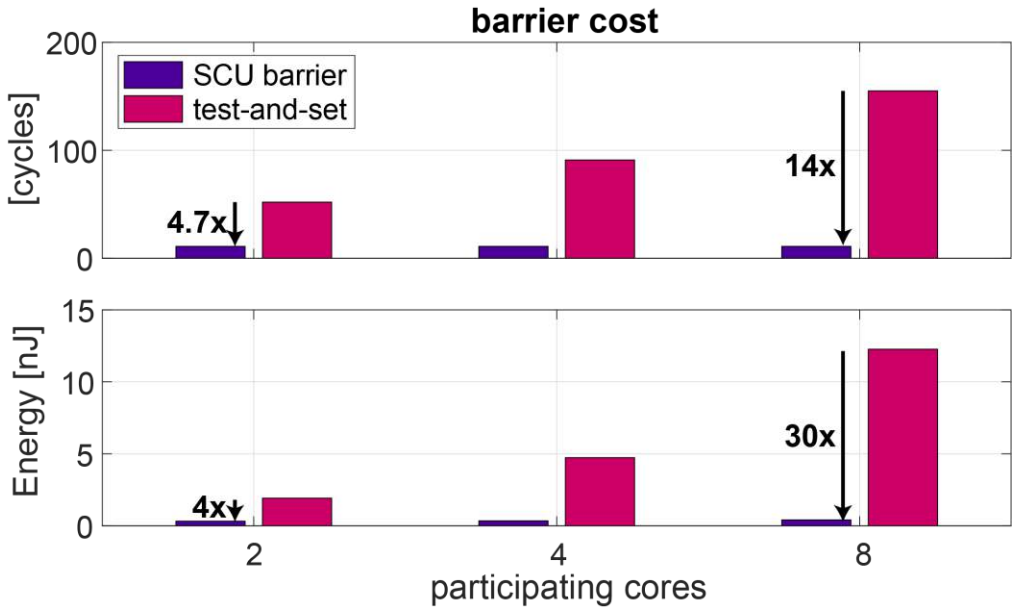
```
evt_data = evt_read(addr);
```

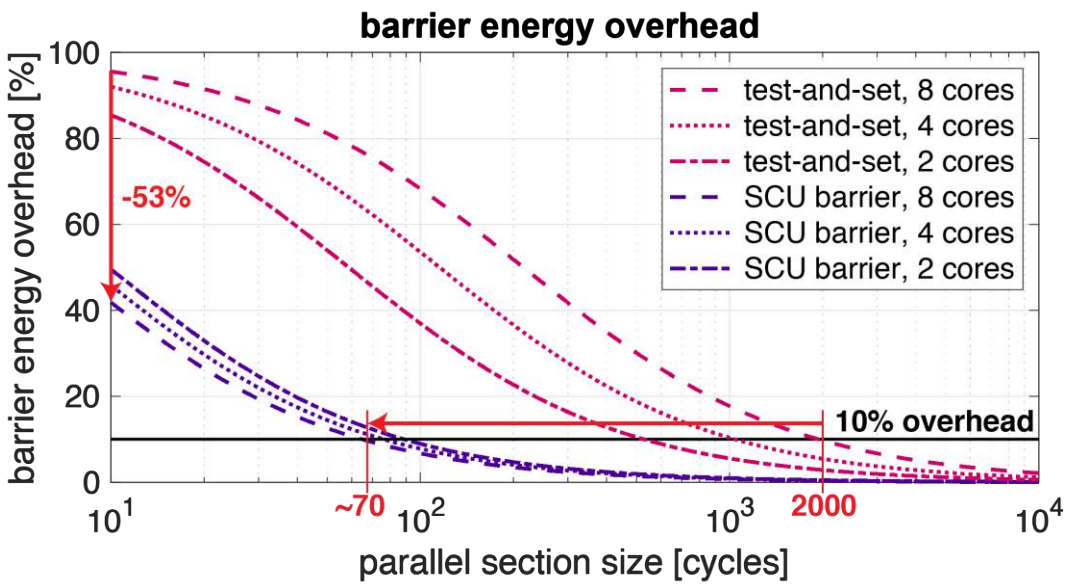
- event buffer content,
- program entry point
- mutex message
- triggering core id
- …

- trigger sw event,
- trigger barrier,
- try mutex lock,
- read entry point,
- auto buffer clear?
- …

- Single instruction rules it all:
  - address determines action
  - read datum provides corresponding information

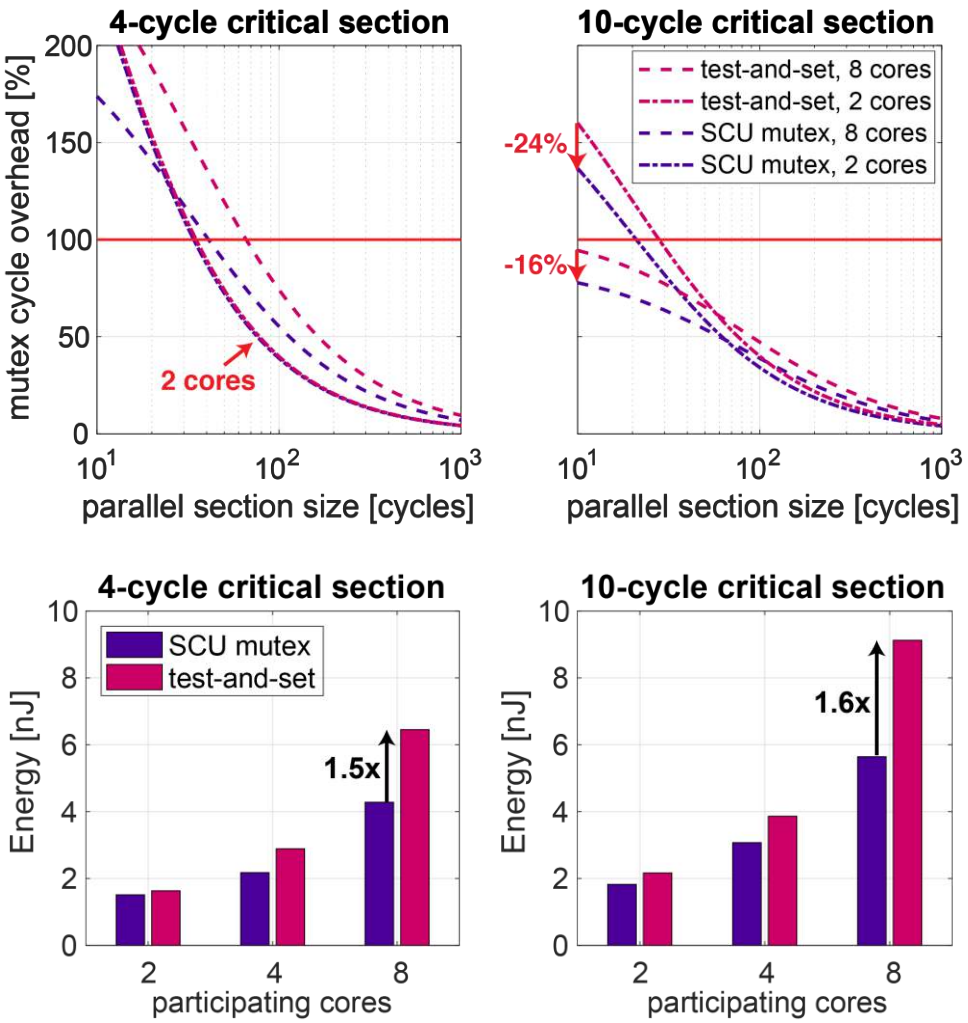
Results: Barrier





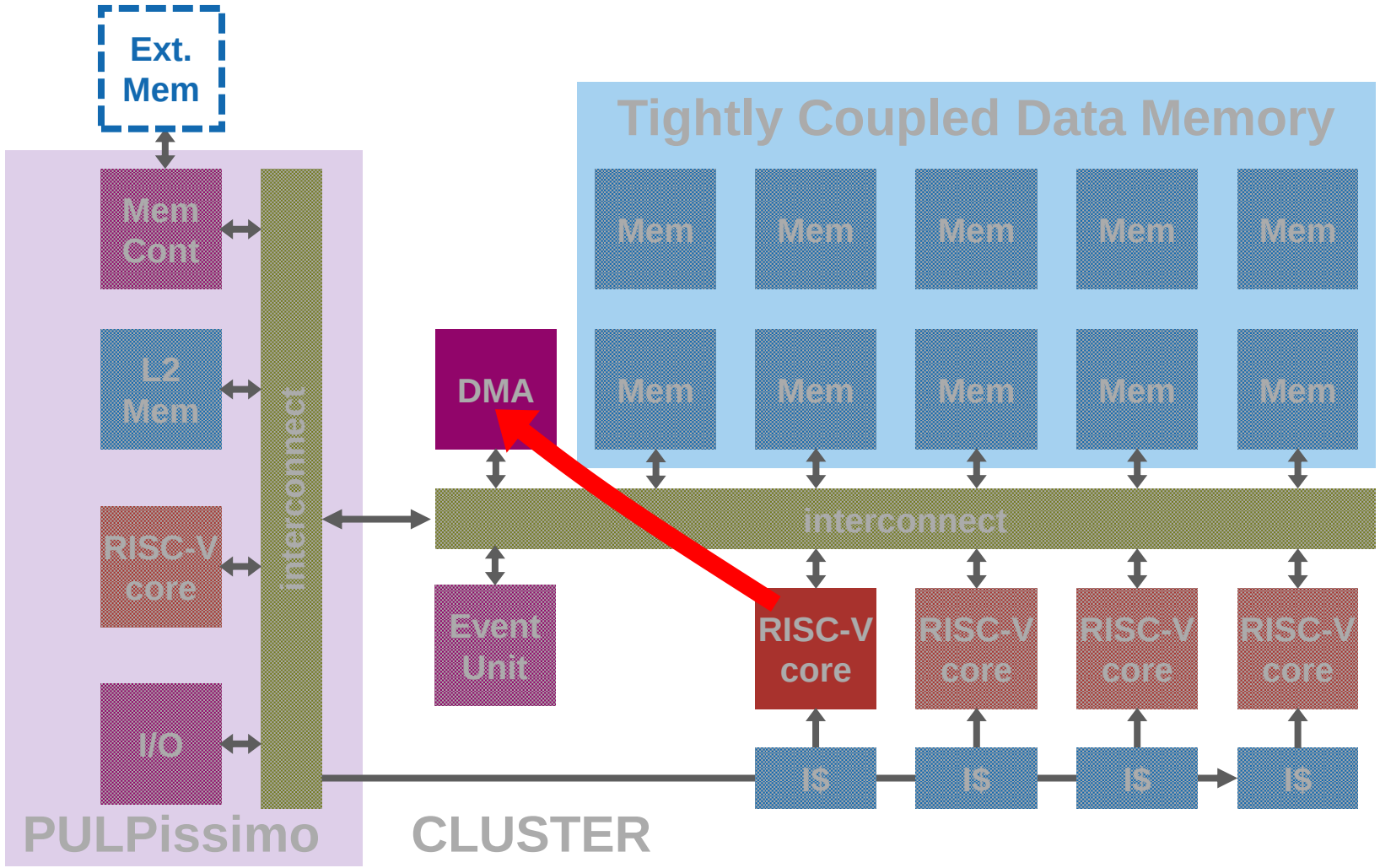
- Fully parallel access to SCU: Barrier cost constant
- Primitive energy cost: Down by up to 30x
- Minimum parallel section for 10% overhead in terms of …
  - … cycles: ~100 instead of > 1000 cycles
  - … energy: ~70 instead of > 2000 cycles

Results: Mutex

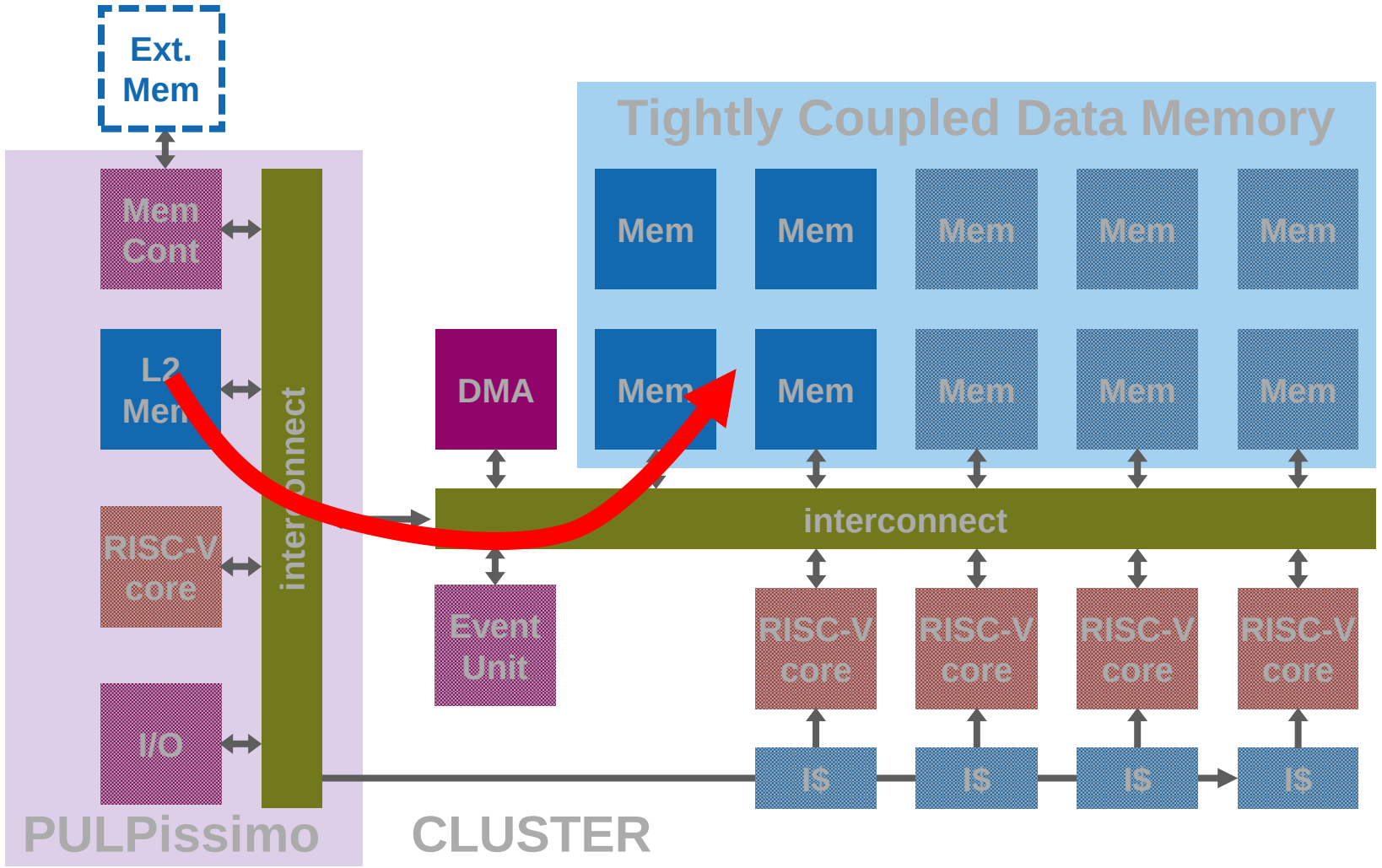


- Sequential execution: Cycle overhead always large
- TAS-variable inherently well-suited for mutex; lower cycle savings compared to barrier
- SCU still avoids L1 accesses: Energy of TAS mutex up to 1.6x higher
- Smallest parallel section for 10% energy overhead:
  ~1000 instead of 1600 cycles

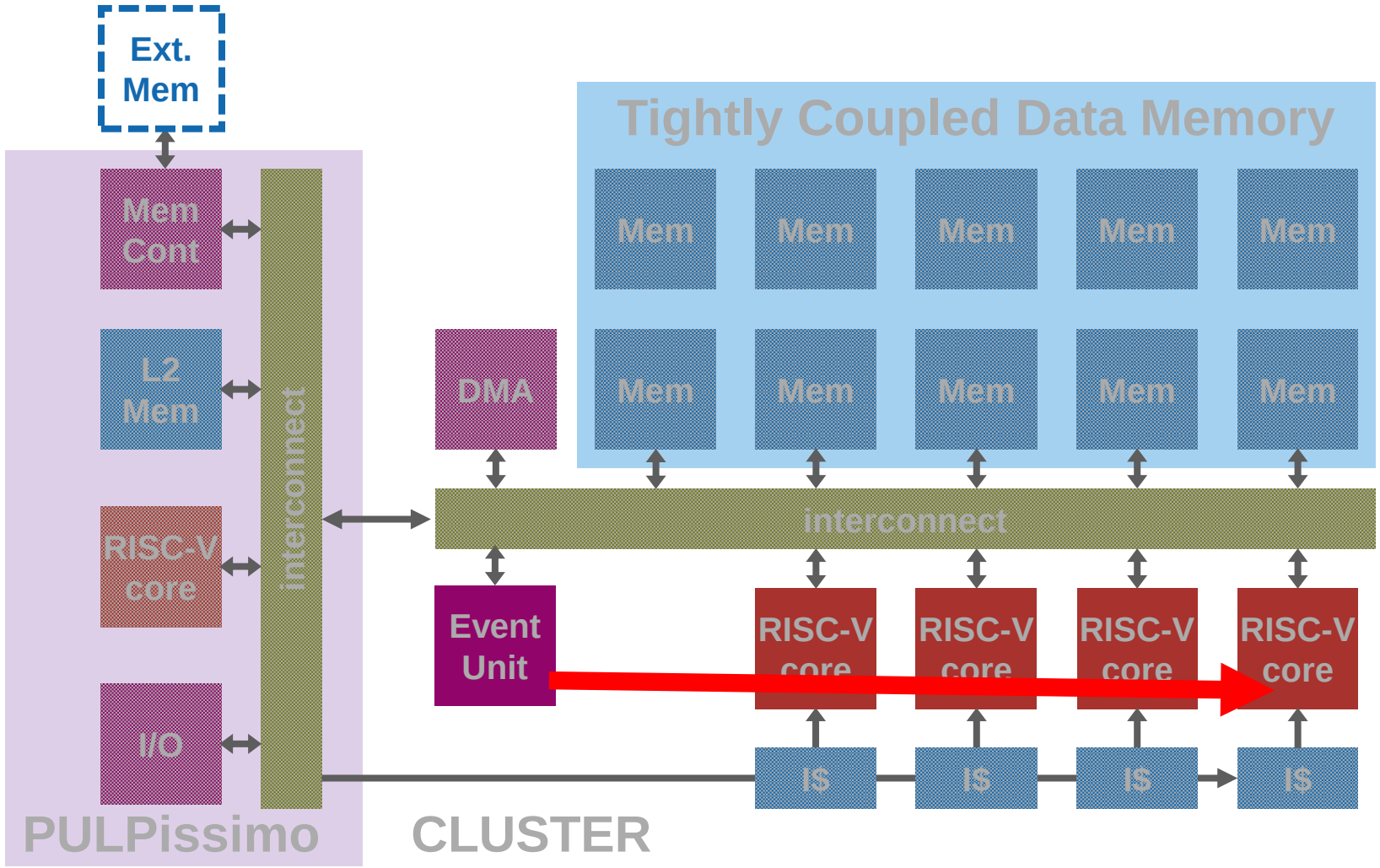
How do we work: Initiate a DMA transfer



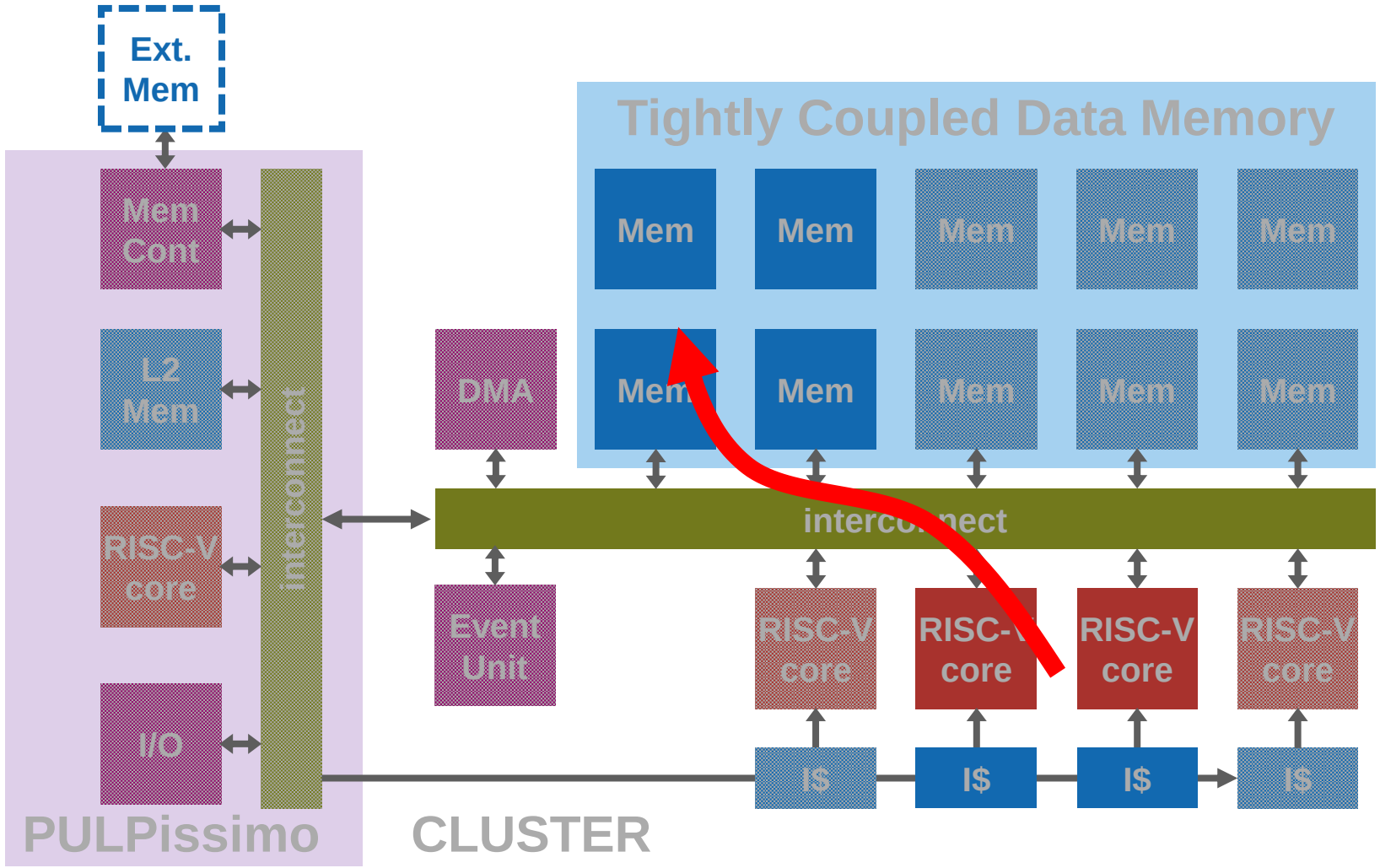
Data copied from L2 into TCDM



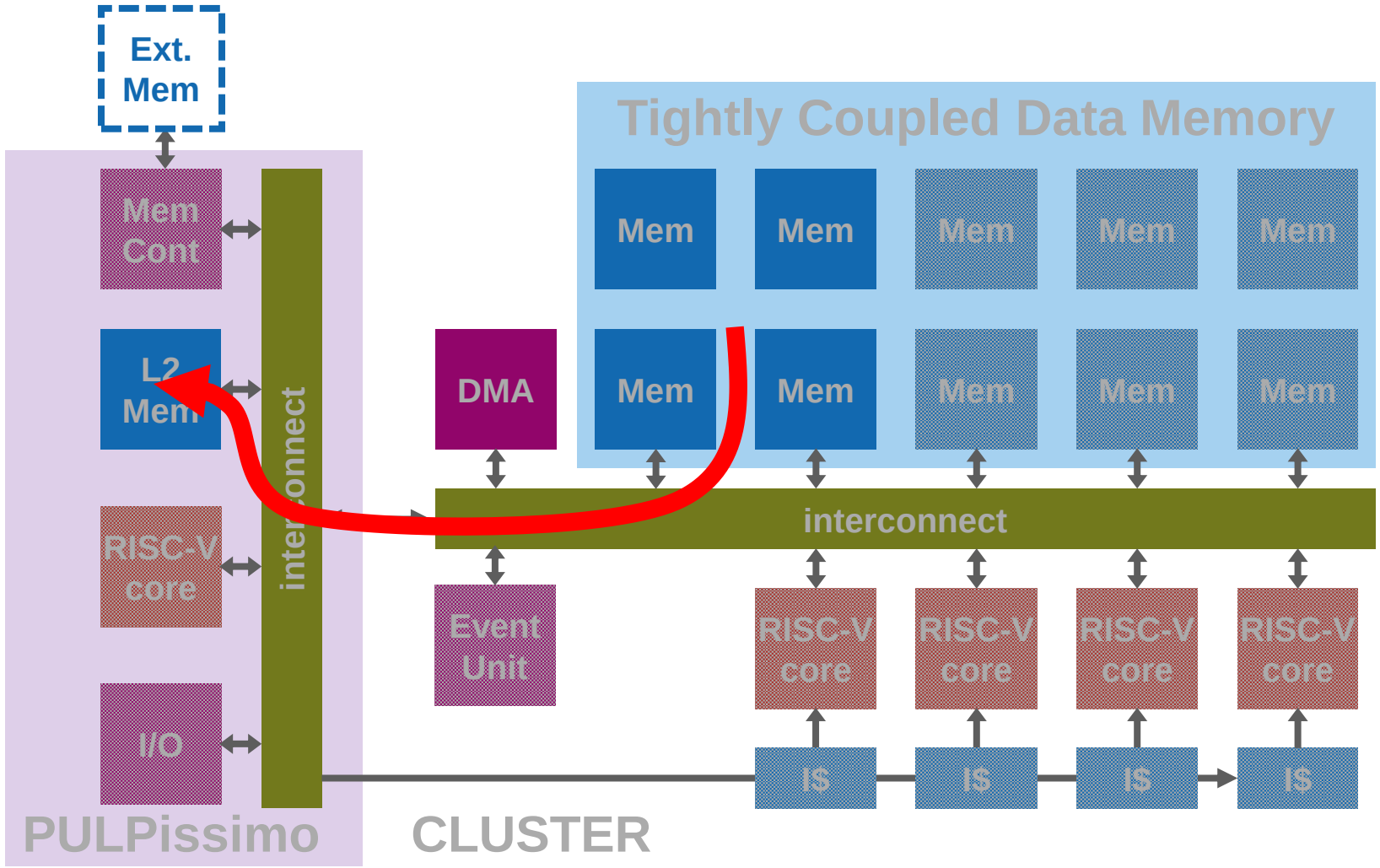
Once data is transferred, event unit notifies cores



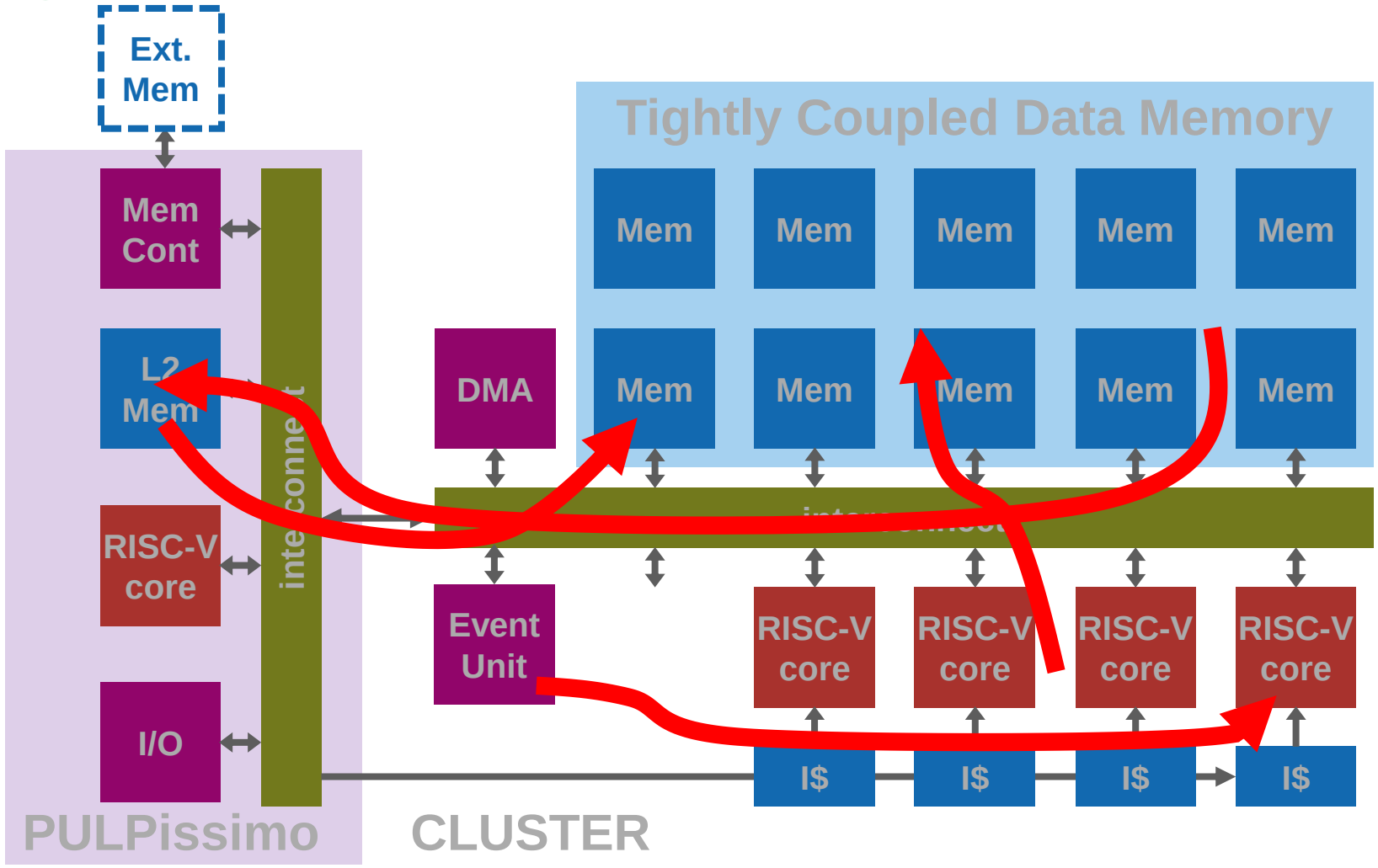
Cores can work on the data transferred



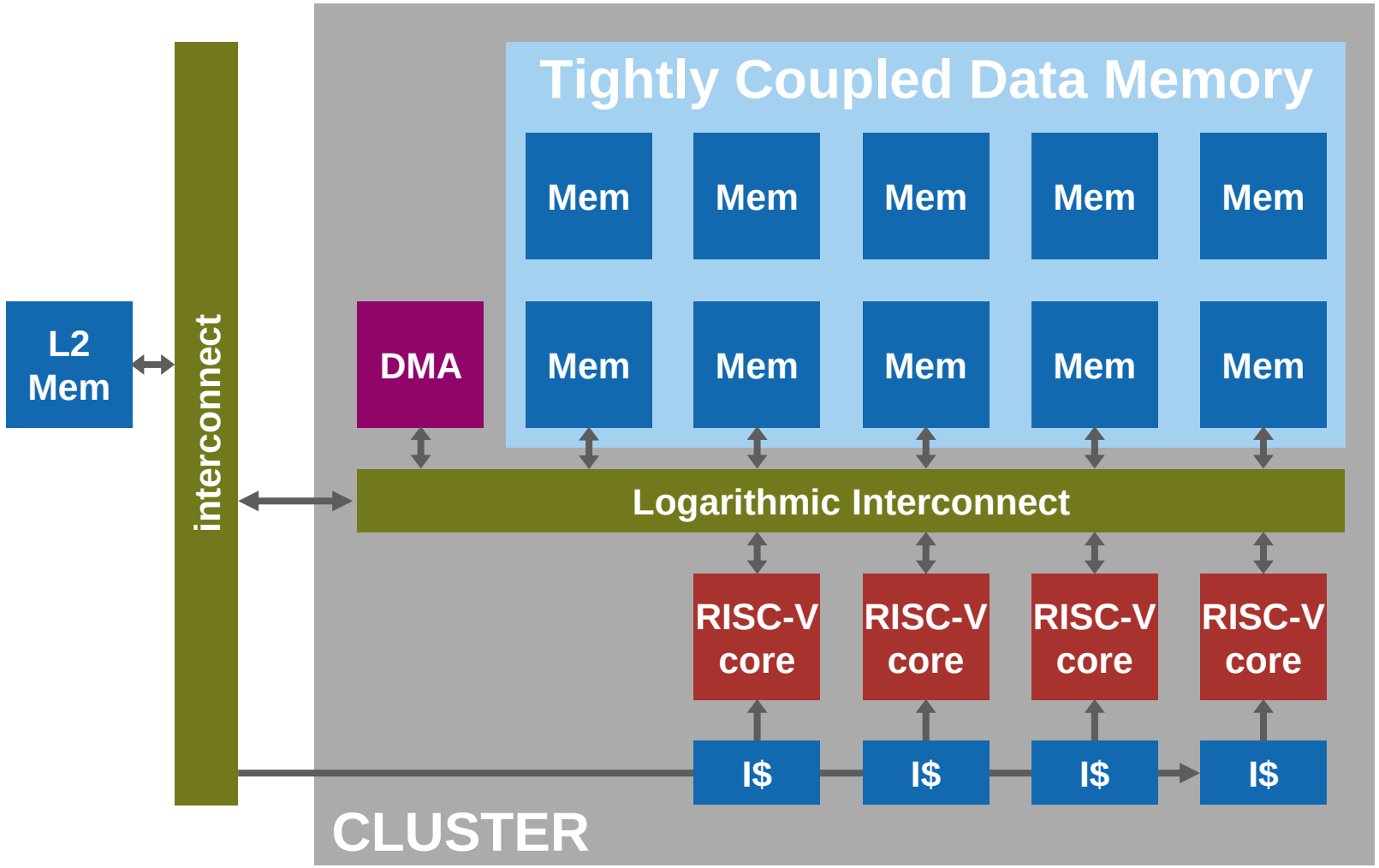
Once our work is done, DMA copies data back



During normal operation all of these occur concurrently



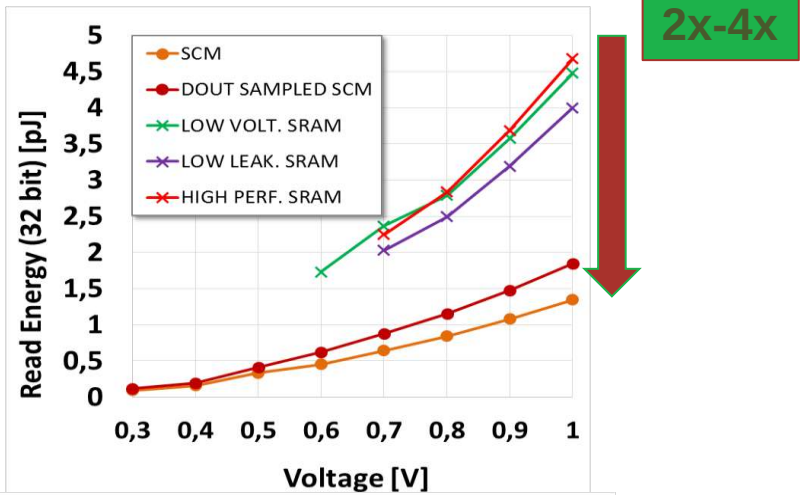
Shared instruction cache with private “loop buffer”

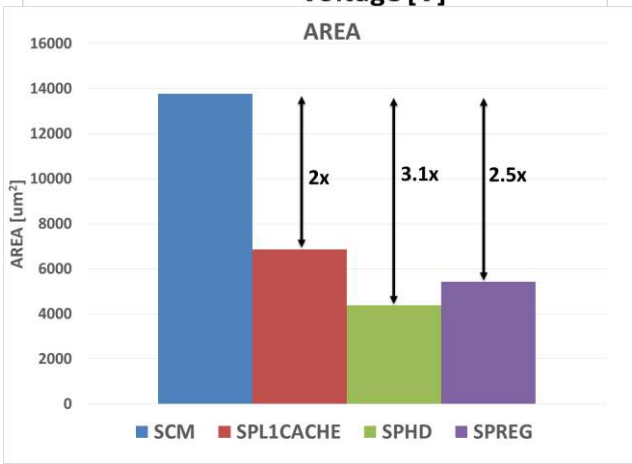


ULP (NT) Bottleneck: Memory

- **"Standard" 6T SRAMs:**
  - High VDDMIN
  - Bottleneck for energy efficiency
    - >50% of energy can go here!!!
- **Near-Threshold SRAMs (8T)**
  - Lower VDDMIN
  - Area/timing overhead (25%-50%)
  - High active energy
  - Low technology portability
- **Standard Cell Memories:**
  - Wide supply voltage range
  - Lower read/write energy (2x - 4x)
  - High technology portability
  - Major area overhead 4x → 2.7x with controlled placement

256x32 6T SRAMS vs. SCM





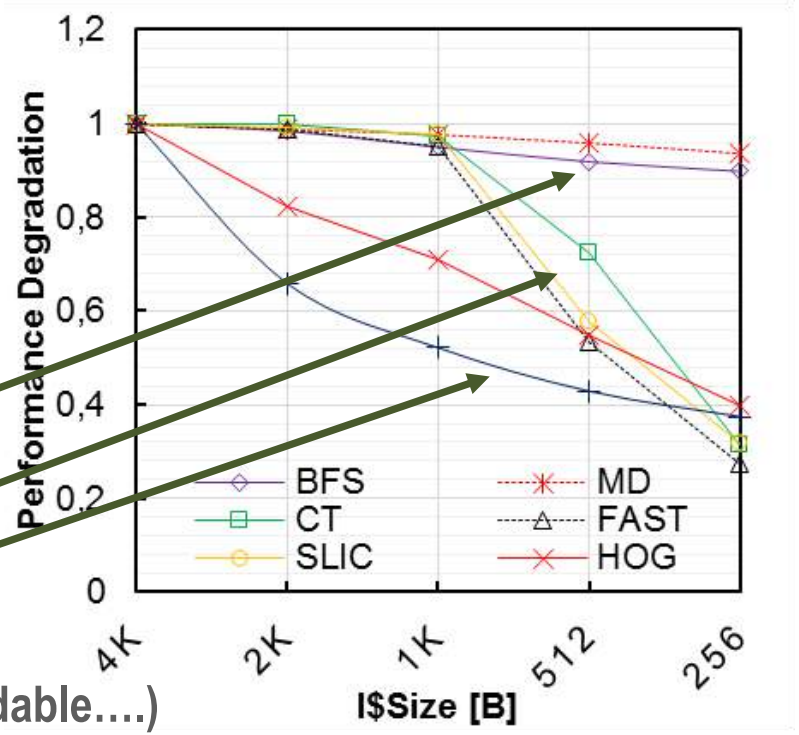
I$: a Look Into 'Real Life' Applications

SCM-BASED I$ IMPROVES EFFICIENCY BY ~2X ON SMALL BENCHMARKS, BUT…

Survey of State of The Art

Exixting ULP processors	Latch based I$
REISC (ESSCIRC2011)	64b
Sleepwalker (ISSCC 2012)	128b
Bellevue (ISCAS 2014)	128b

Applications on PULP



SHORT JUMP LOOP BASED APPLICATIONS

LONG JUMP APPLICATIONS

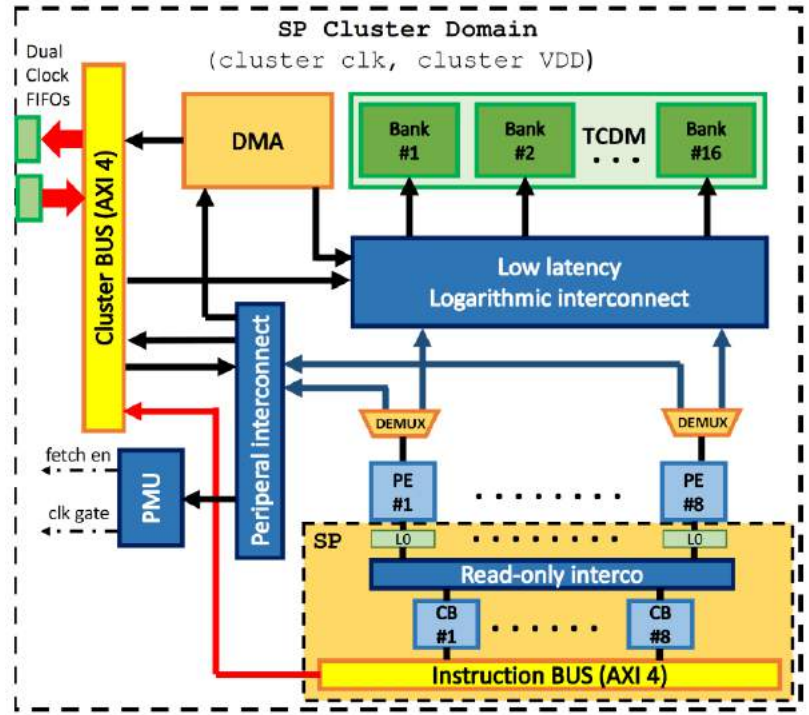
LIBRARY BASED

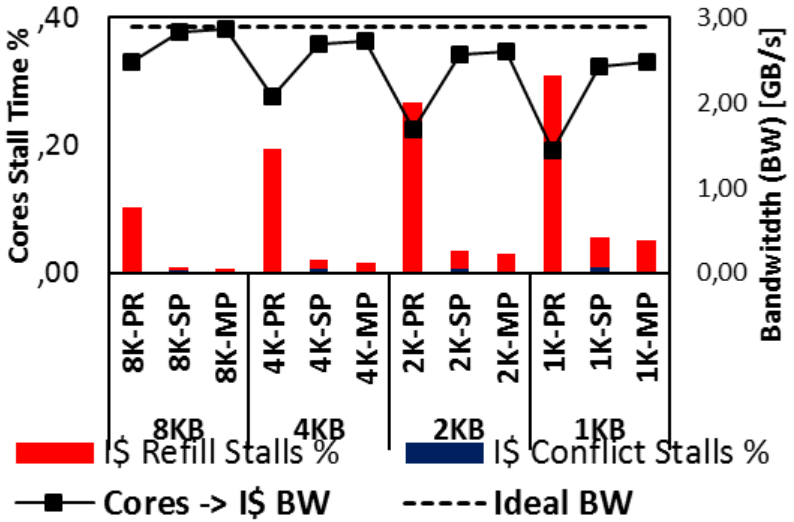
Issues:

1) Area Overhead of SCMs (4Kb/core not affordable….)
2) Capacity miss (with small caches)
3) Jumps due to runtime (e.g. OpenMP, OpenCL) and other function calls

Shared I$

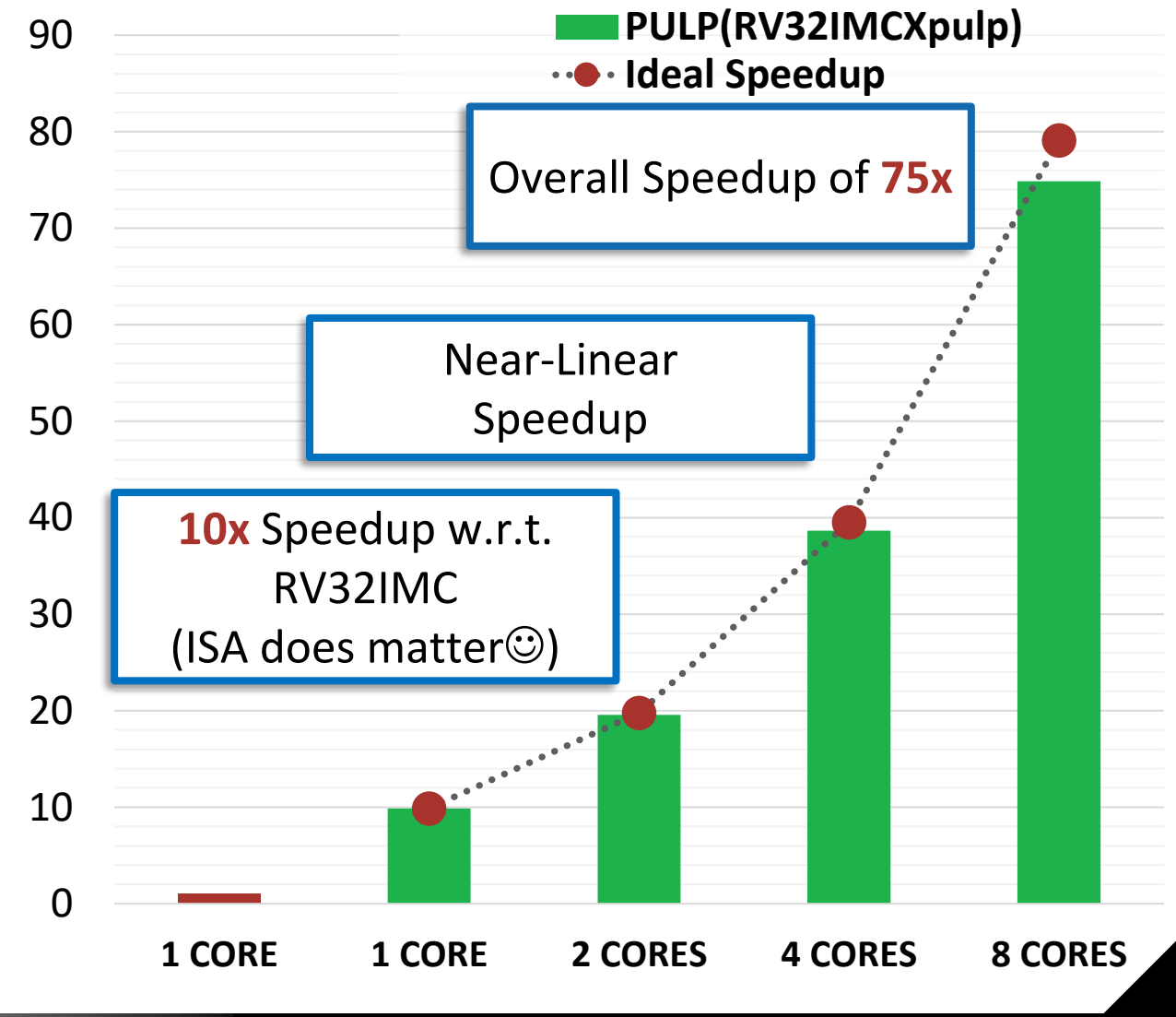
- **Shared instruction cache**
  - OK for data parallel execution model
  - Not OK for task parallel execution model, or very divergent parallel threads
- **Architectures**
  - SP: single-port banks connected through a read-only interconnect
    - Pros: Low area overhead
    - Cons: Timing pressure, contention
  - MP: Multi-ported banks
    - Pros: High efficiency
    - Cons: Area overhead (several ports)
- **Results**
  - Up to 40% better performance than private I$
  - Up to 30% better energy efficiency
  - Up to 20% better energy*area efficiency



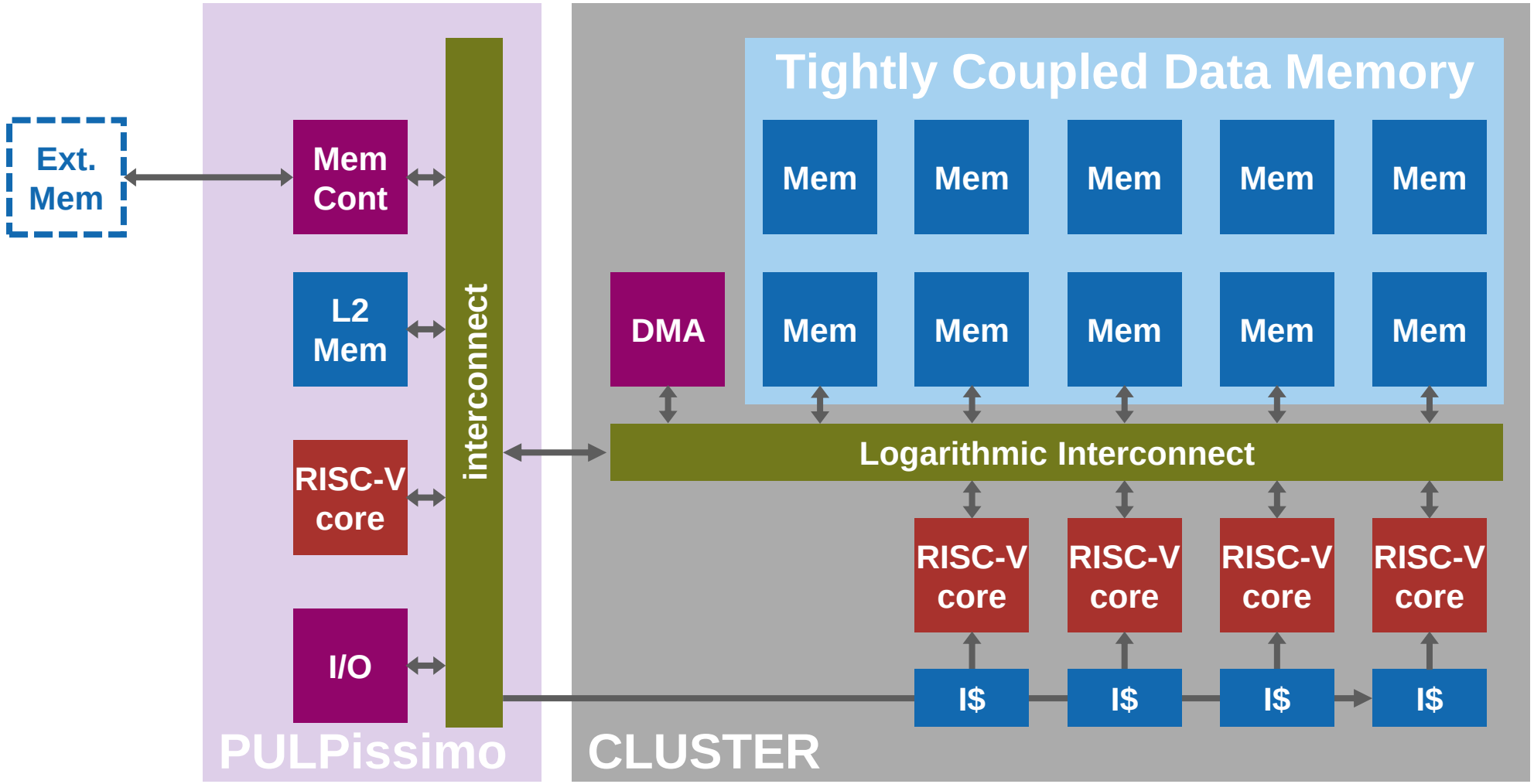


Results: RV32IMCXpulp vs RV32IMC

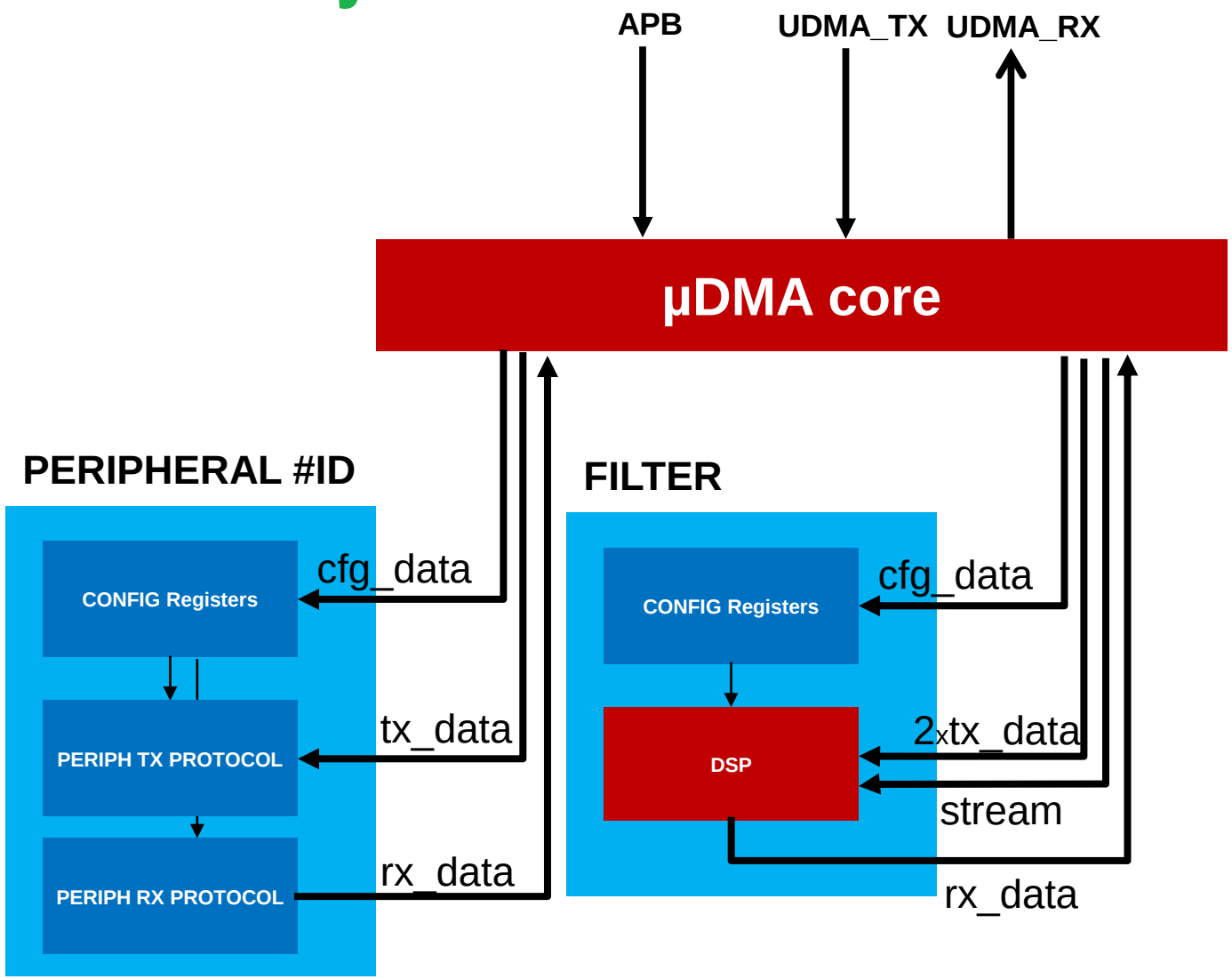
- **8-bit convolution**
  - Open source DNN library
- **10x through xPULP**
  - Extensions bring real speedup
- **Near-linear speedup**
  - Scales well for regular workloads.
- **75x overall gain**



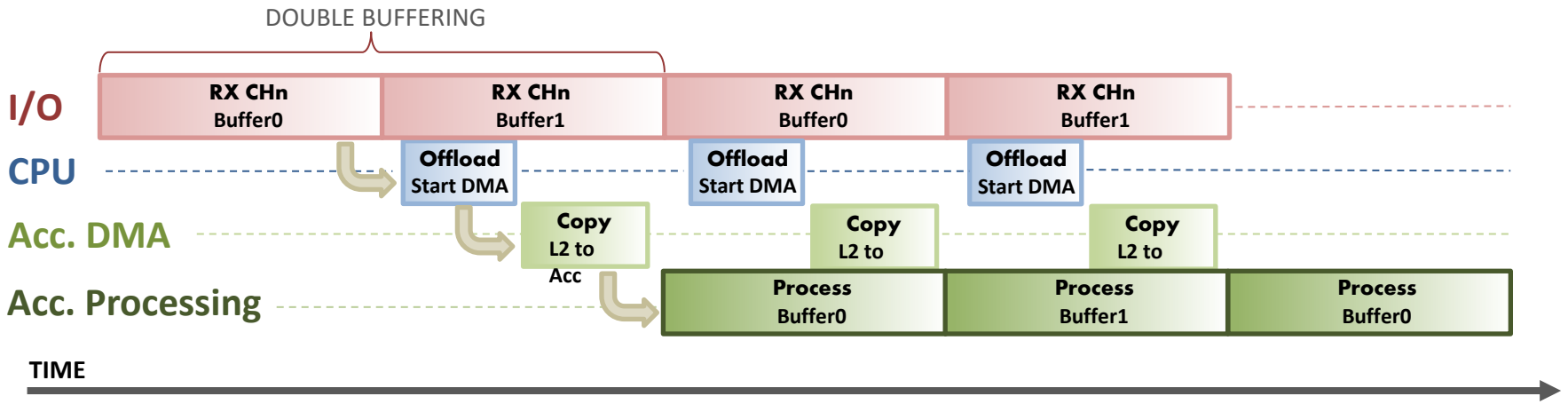
An additional I/O controller is used for IO



uDMA Subsystem



Offload pipeline

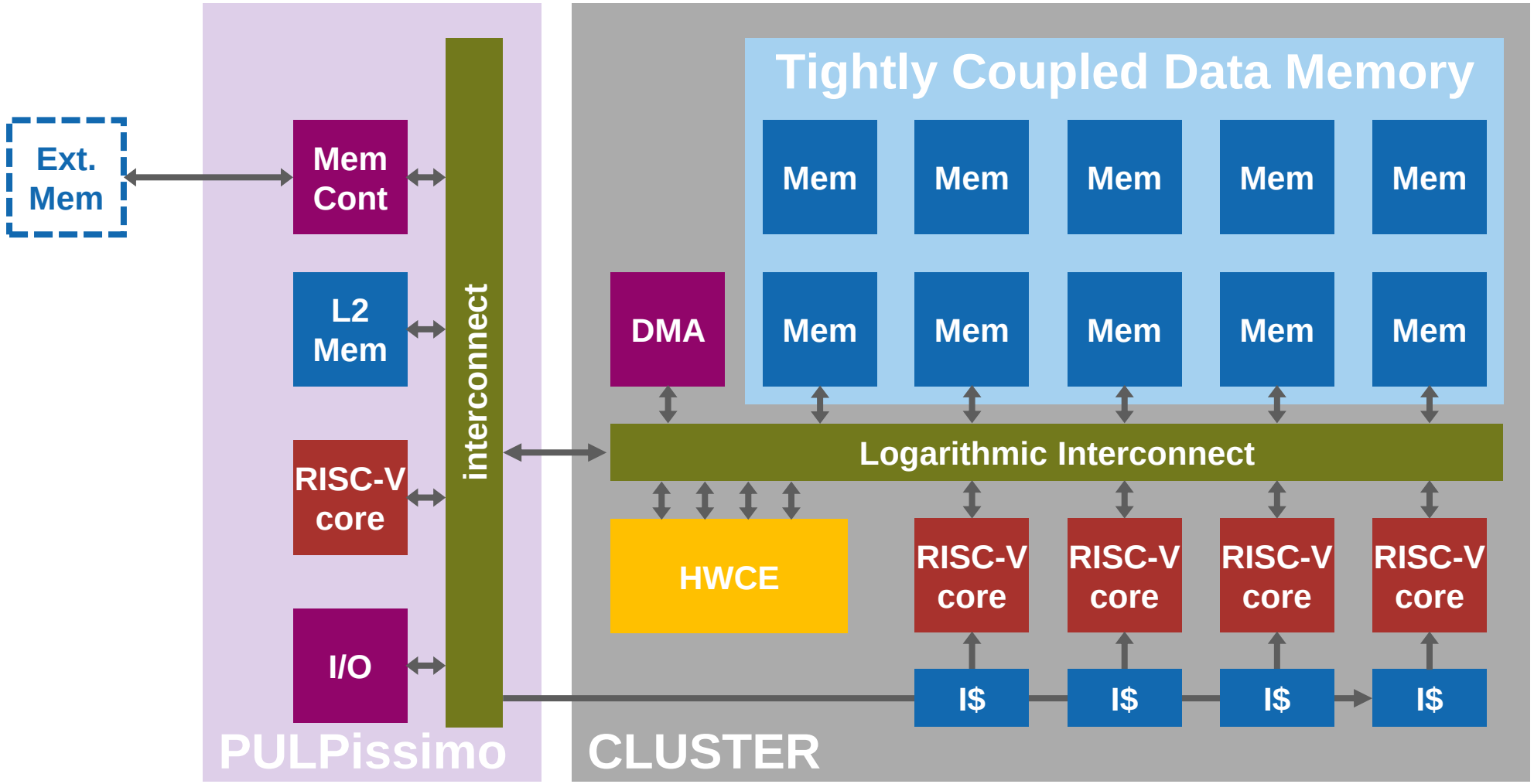


- Efficient use of system resources
- HW support for double buffering allows continuous data transfers
- Multiple data streams can be time multiplexed

PULP interrupts controller (INTC)

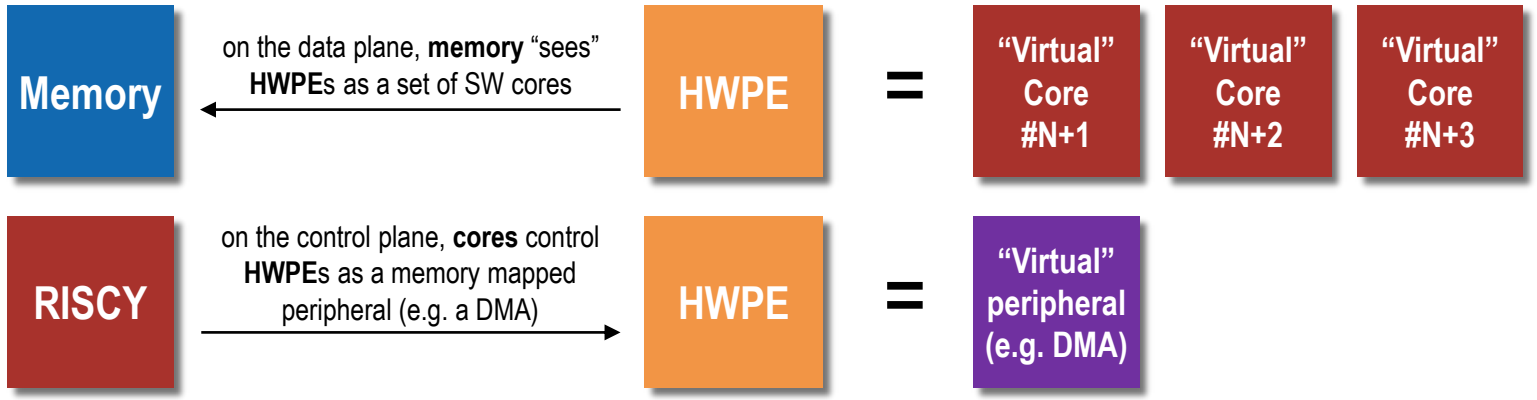
- It generates interrupt requests from 0 to 31
- Mapped to the APB bus
- Receives events in a FIFO from the SoC Event Generator (i.e. from peripherals)
  - Unique interrupt ID (26) but different event ID
- Mask, pending interrupts, acknowledged interrupts, event id registers
- Set, Clear, Read and Write operations by means of load and store instructions (memory mapped operations)
- Interrupts come from:
  - Timers
  - GPIO (rise, fall events)
  - HWCE
  - Events i.e. uDMA

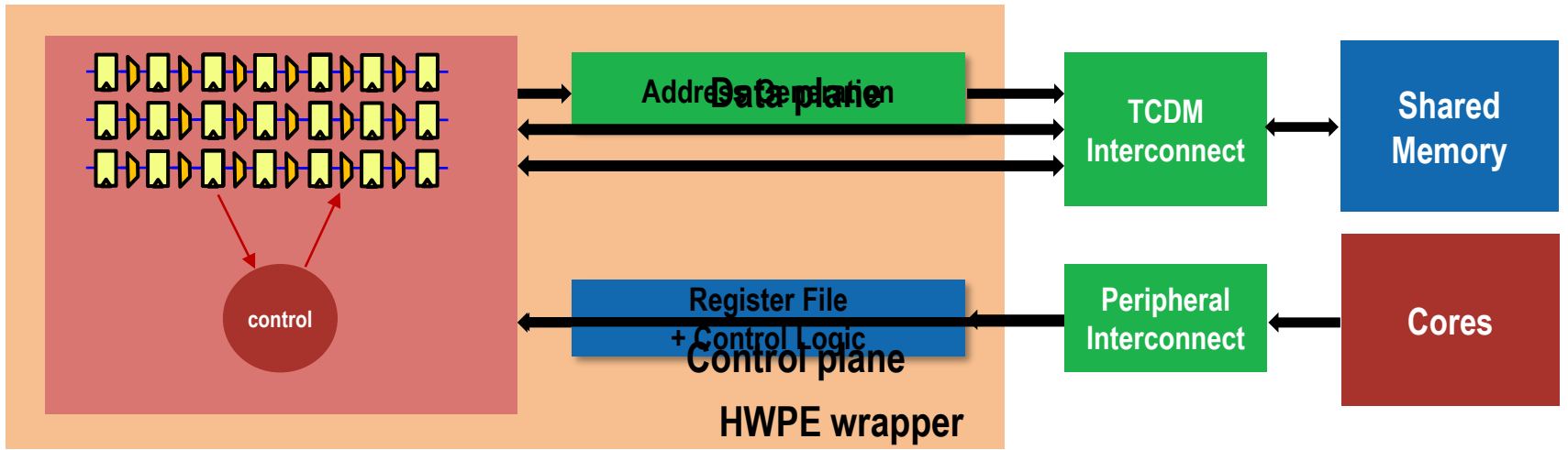
Tightly-coupled HW Compute Engine



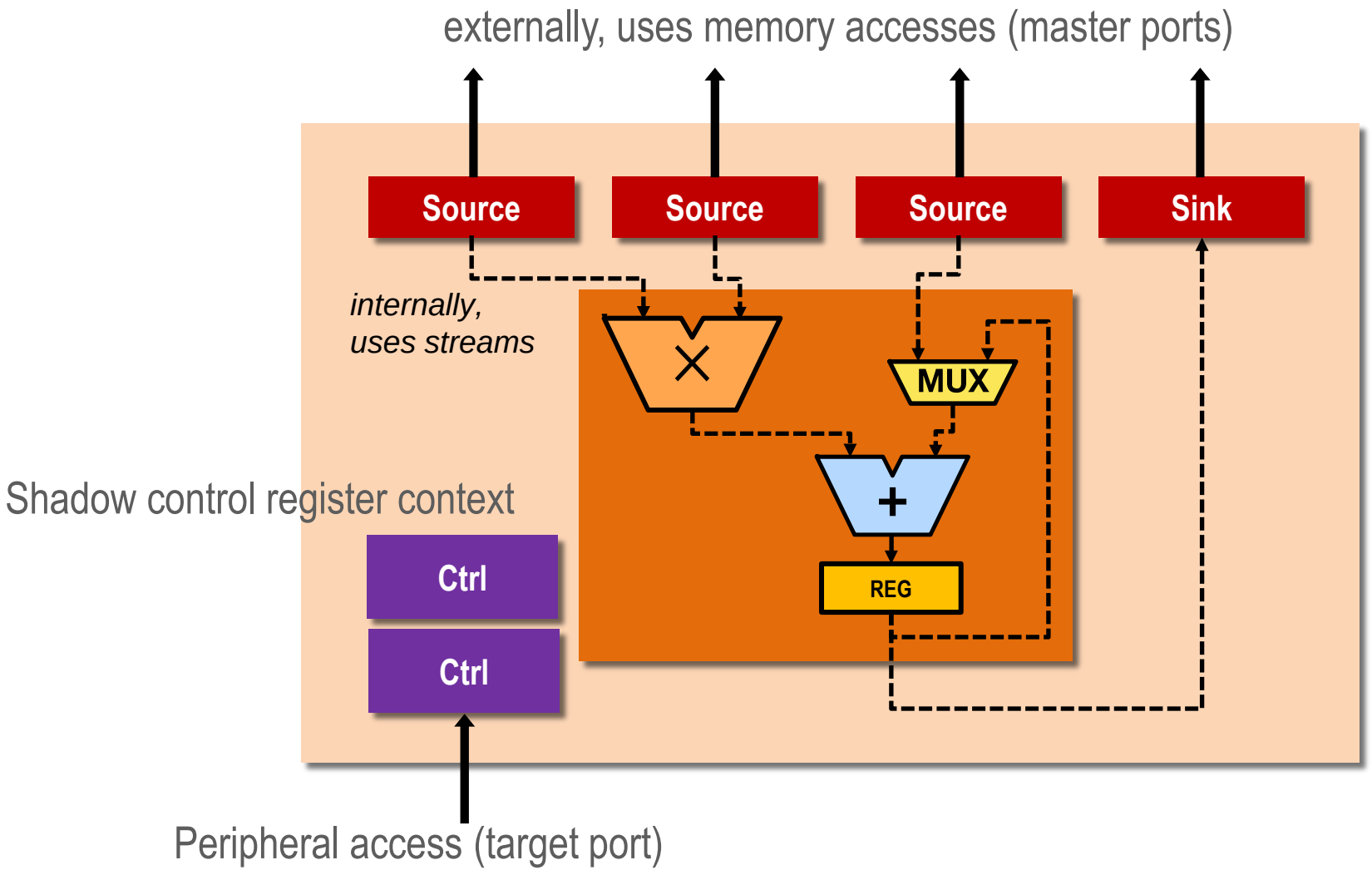
Acceleration with flexibility: zero-copy HW-SW cooperation

Hardware Processing Engines (HWPEs)





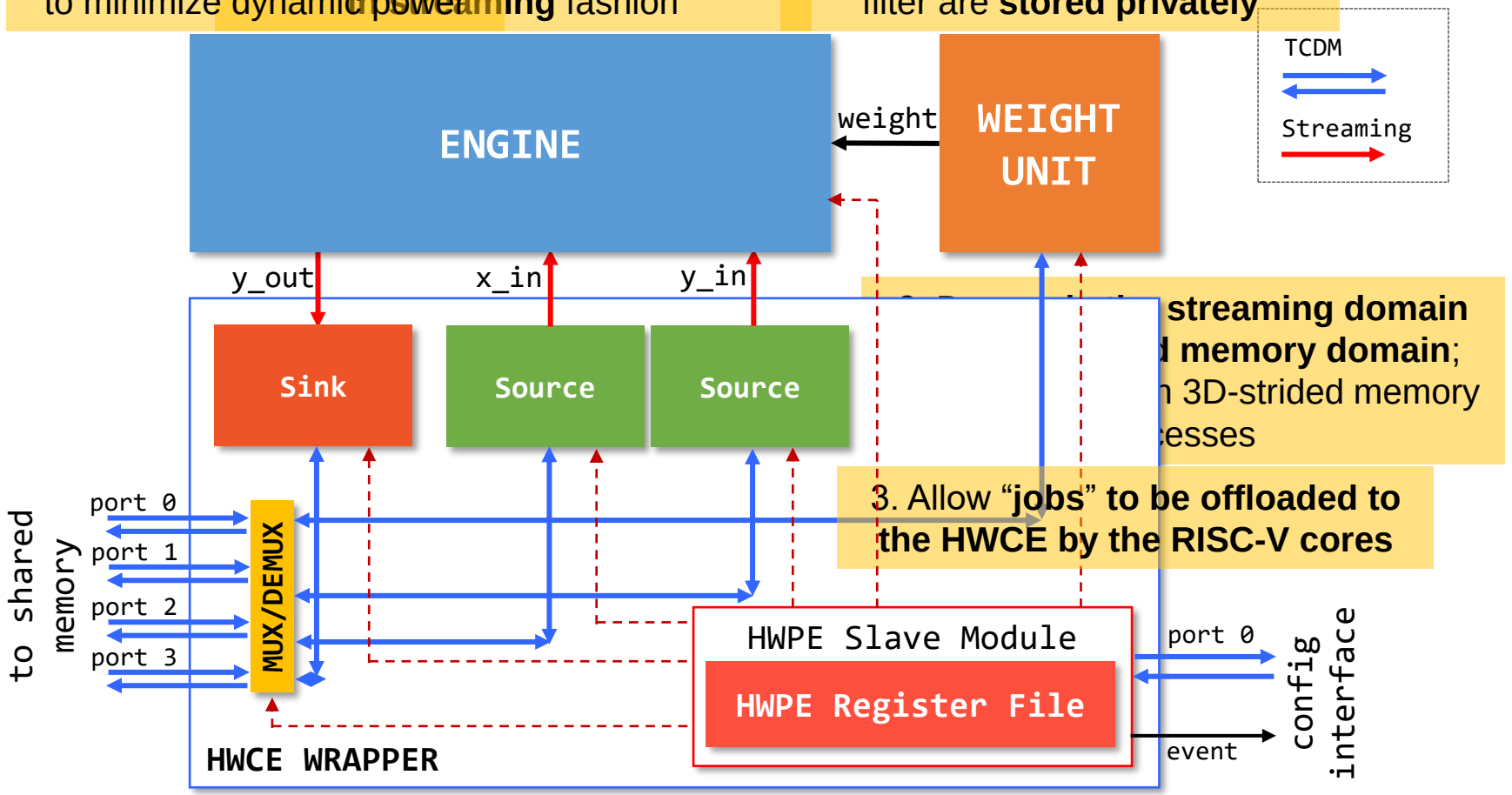
A toy HWPE


externally, uses memory accesses (master ports)
Source
Source
Source
Sink
internally,
uses streams
MUX
REG
Shadow control register context
Ctrl
Ctrl
Peripheral access (target port)

HW Convolution Engine

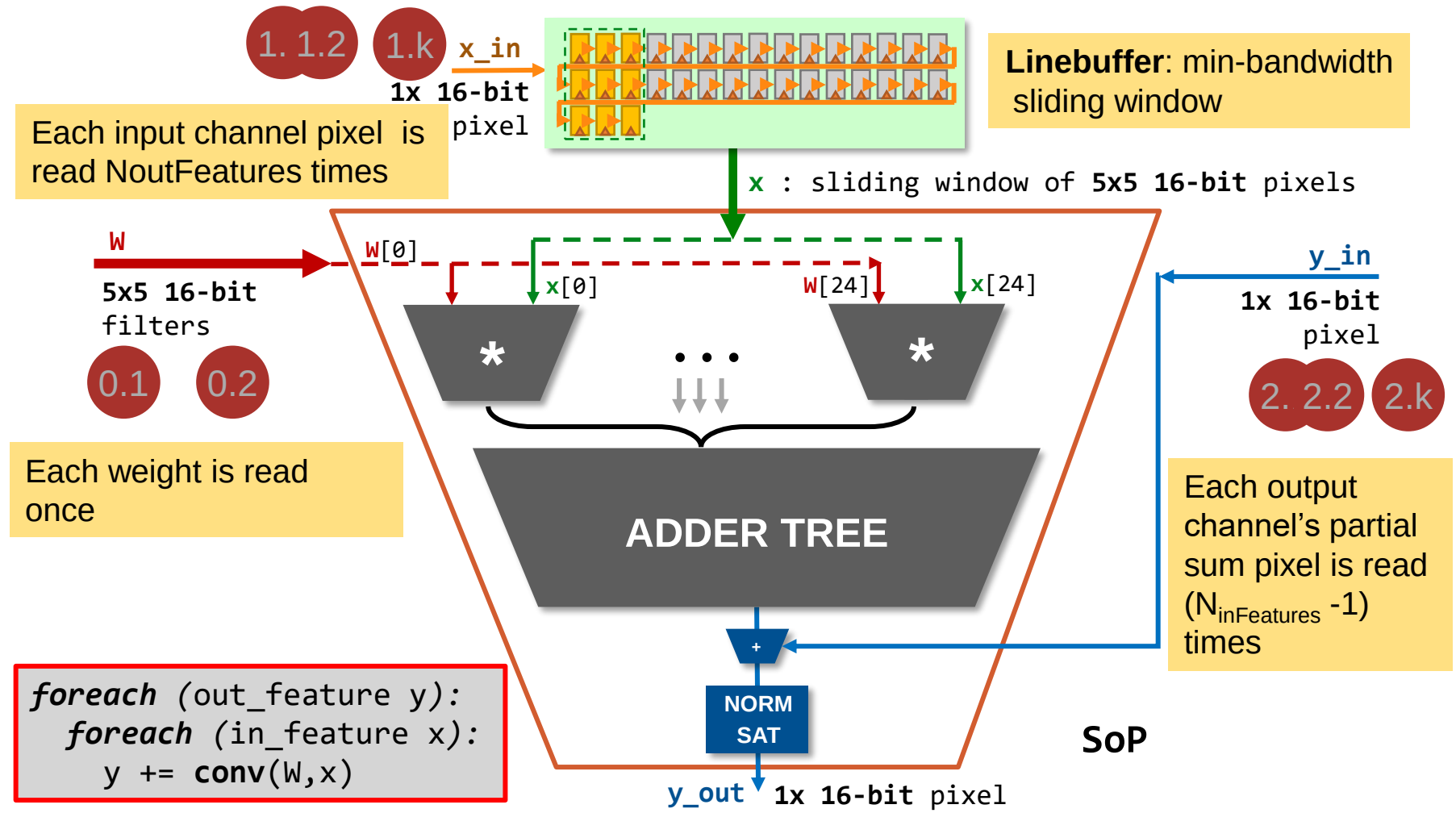
5. **Fine-grain clock gating** to minimize dynamic power

4. **Weights** for each convolution filter are **stored privately**



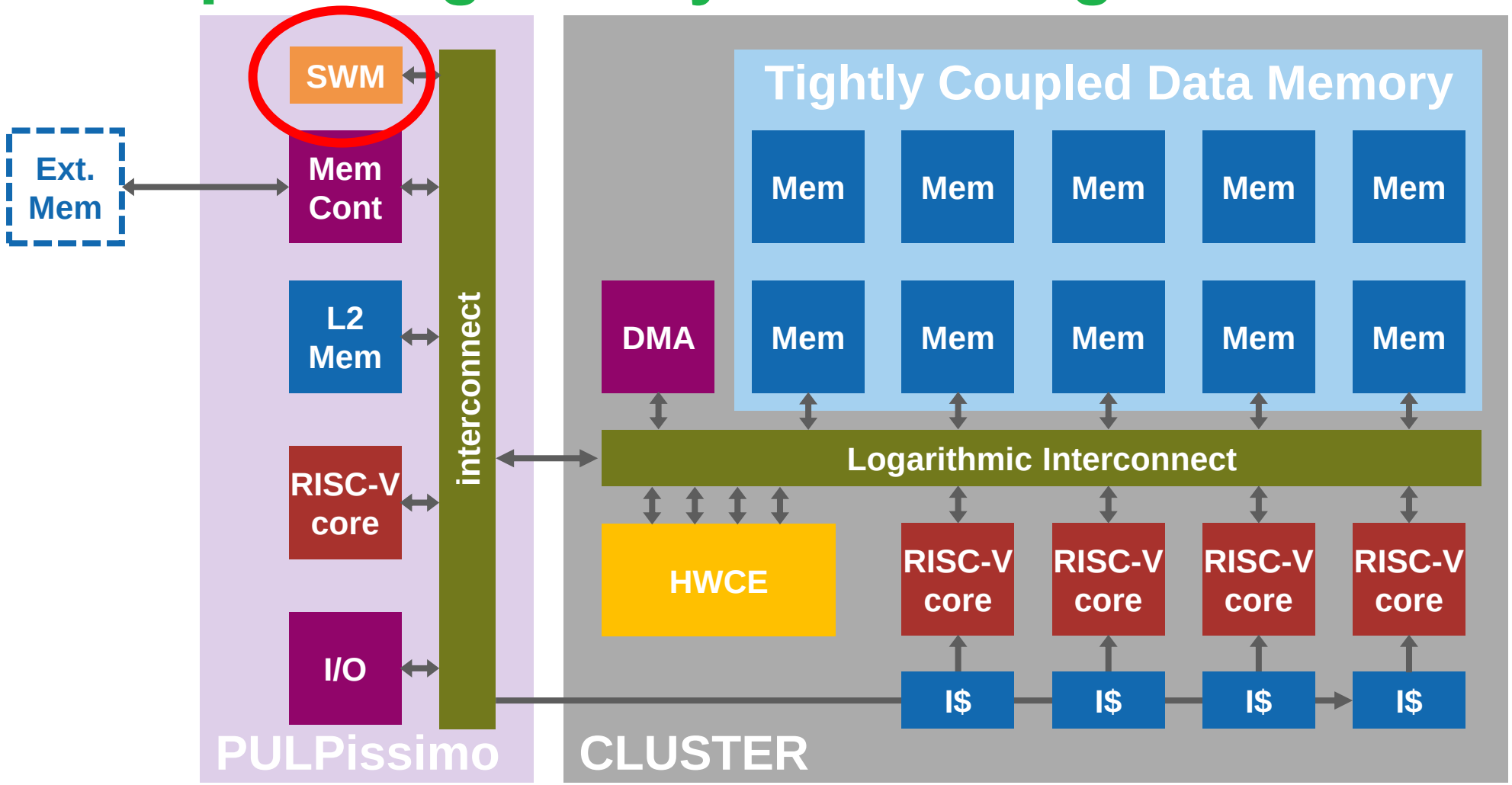
F. Conti and L. Benini, "A ultra-low-energy convolution engine for fast brain-inspired vision in multicore clusters," *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2015, pp. 683-688.

HWCE Sum-of-Products



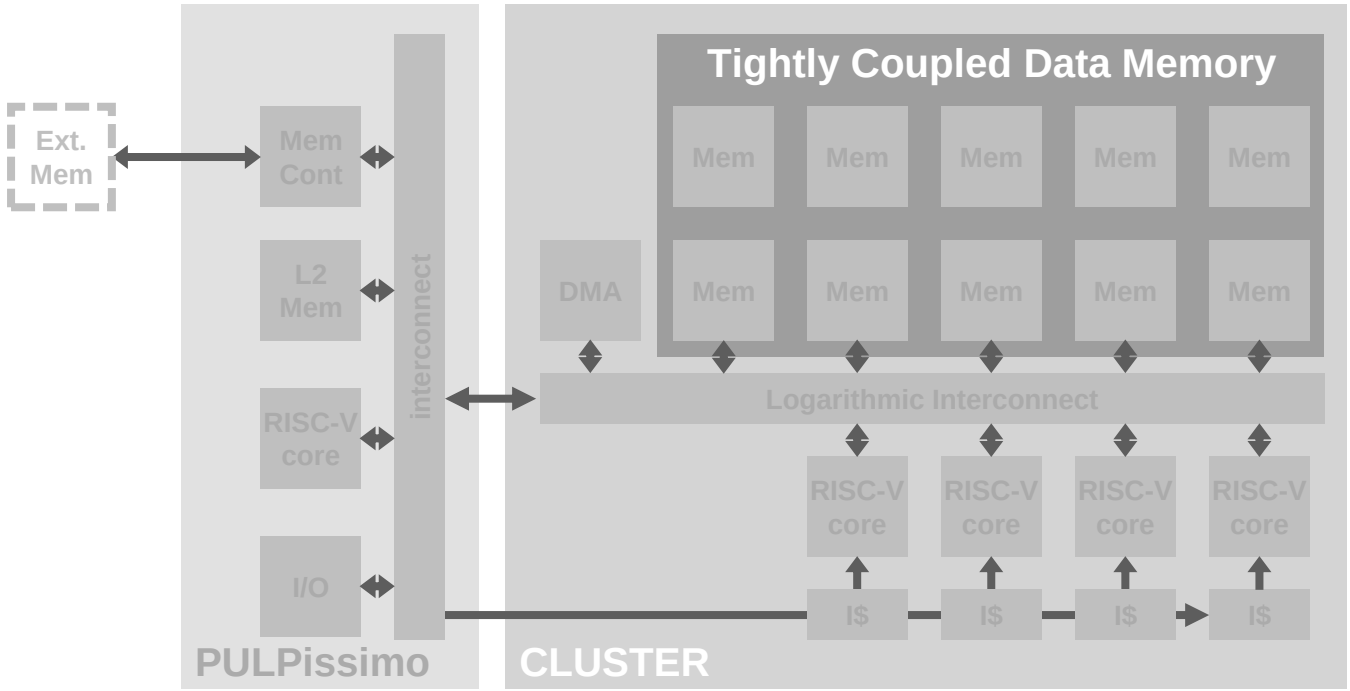

```
foreach (out_feature y):
  foreach (in_feature x):
    y += conv(W,x)
```

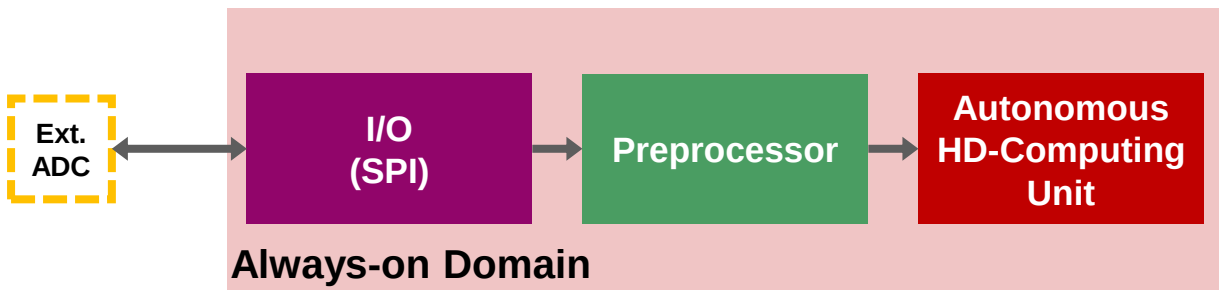
Need µW-range always-on Intelligence



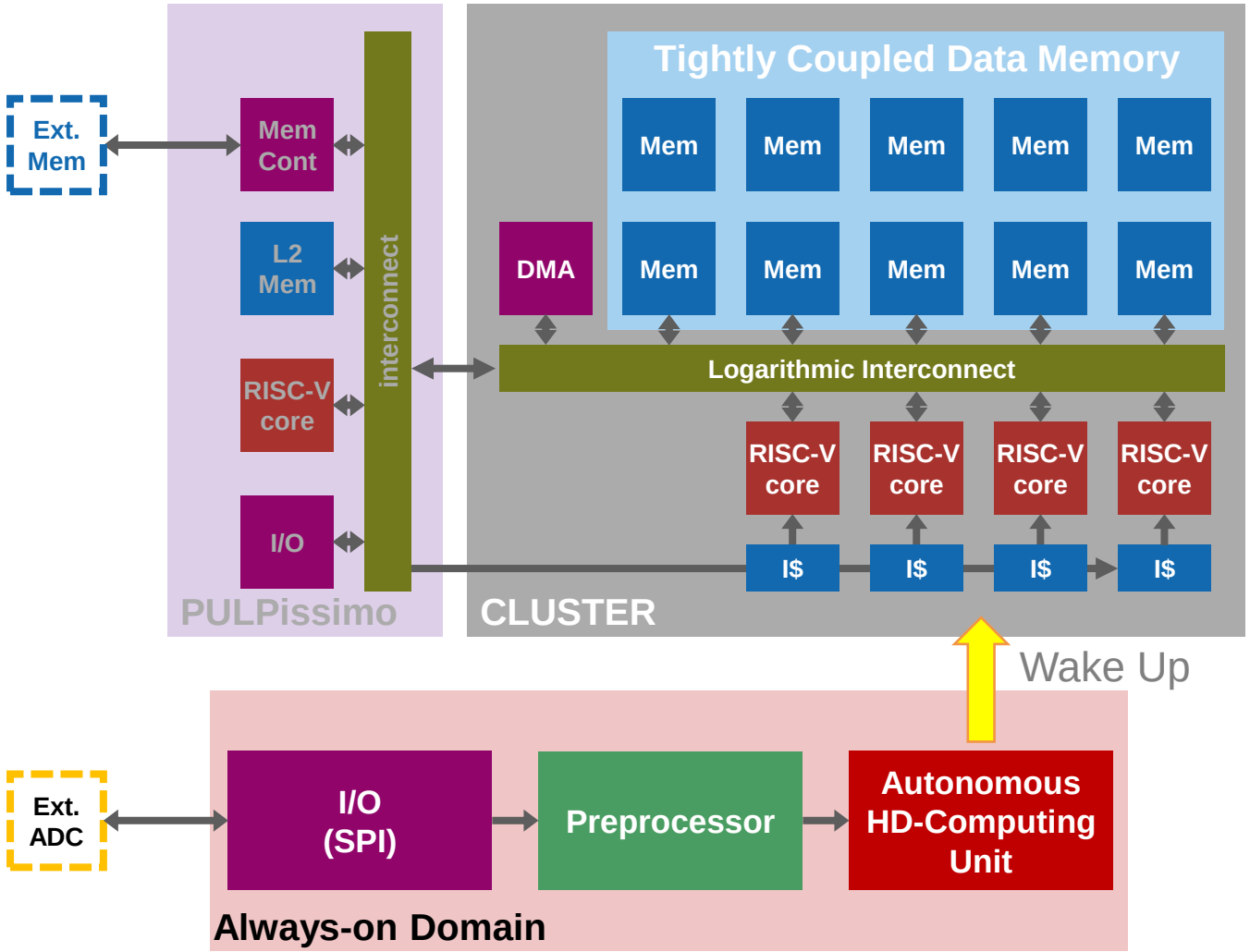
Smart Wakeup Module

HD-Based smart Wake-Up Module

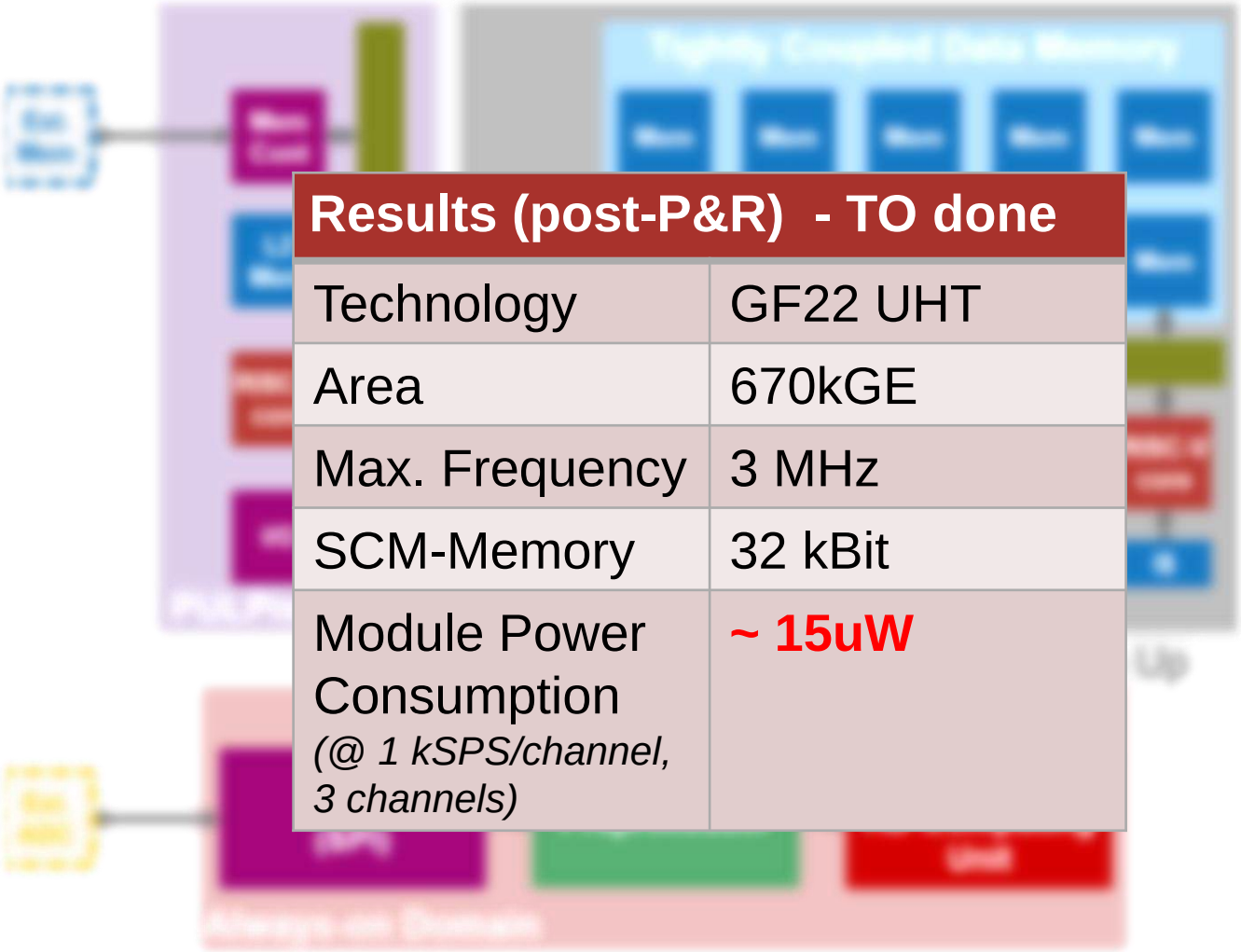




HD-Based smart Wake-Up Module



HD-Based smart Wake-Up Module



Results (post-P&R) - TO done	
Technology	GF22 UHT
Area	670kGE
Max. Frequency	3 MHz
SCM-Memory	32 kBit
Module Power Consumption *(@ 1 kSPS/channel, 3 channels)*	**~ 15uW**

PULP includes Cores+Interco+IO+HWCE → Open Platform

RISC-V Cores

RI5CY	Ibex	Snitch	Ariane + Ara
32b	32b	32b	64b

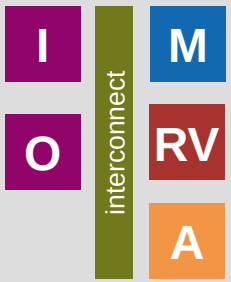
Peripherals

JTAG	SPI
UART	I2S
DMA	GPIO

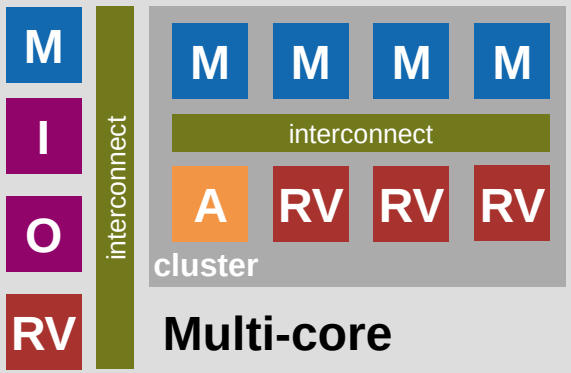
Interconnect

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

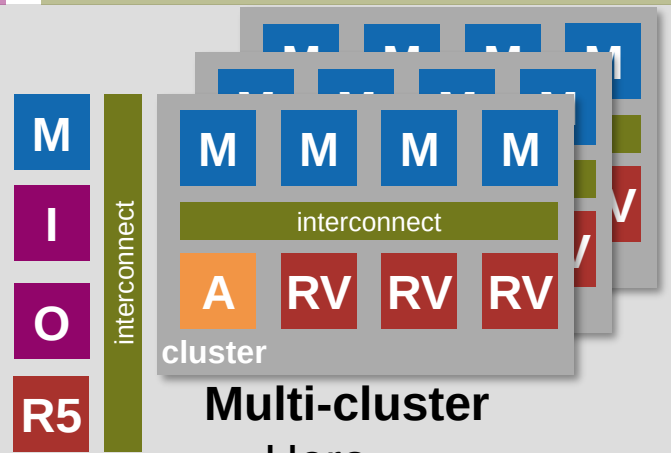
Platforms



Single Core
- PULPino
- PULPissimo



Multi-core
- Fulmine
- Mr. Wolf



Multi-cluster
- Hero
- Open Piton

IOT → HPC

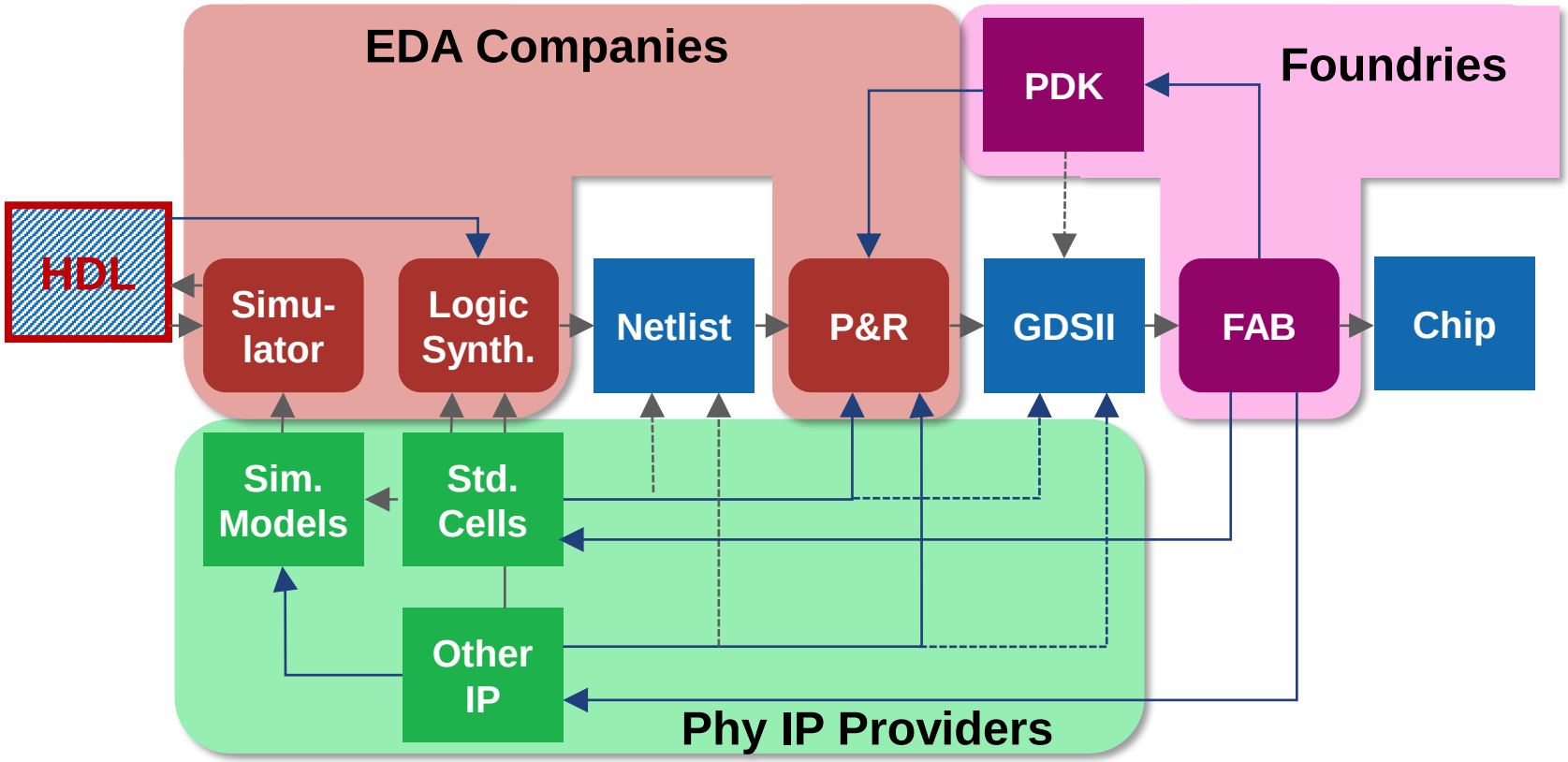
Accelerators

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

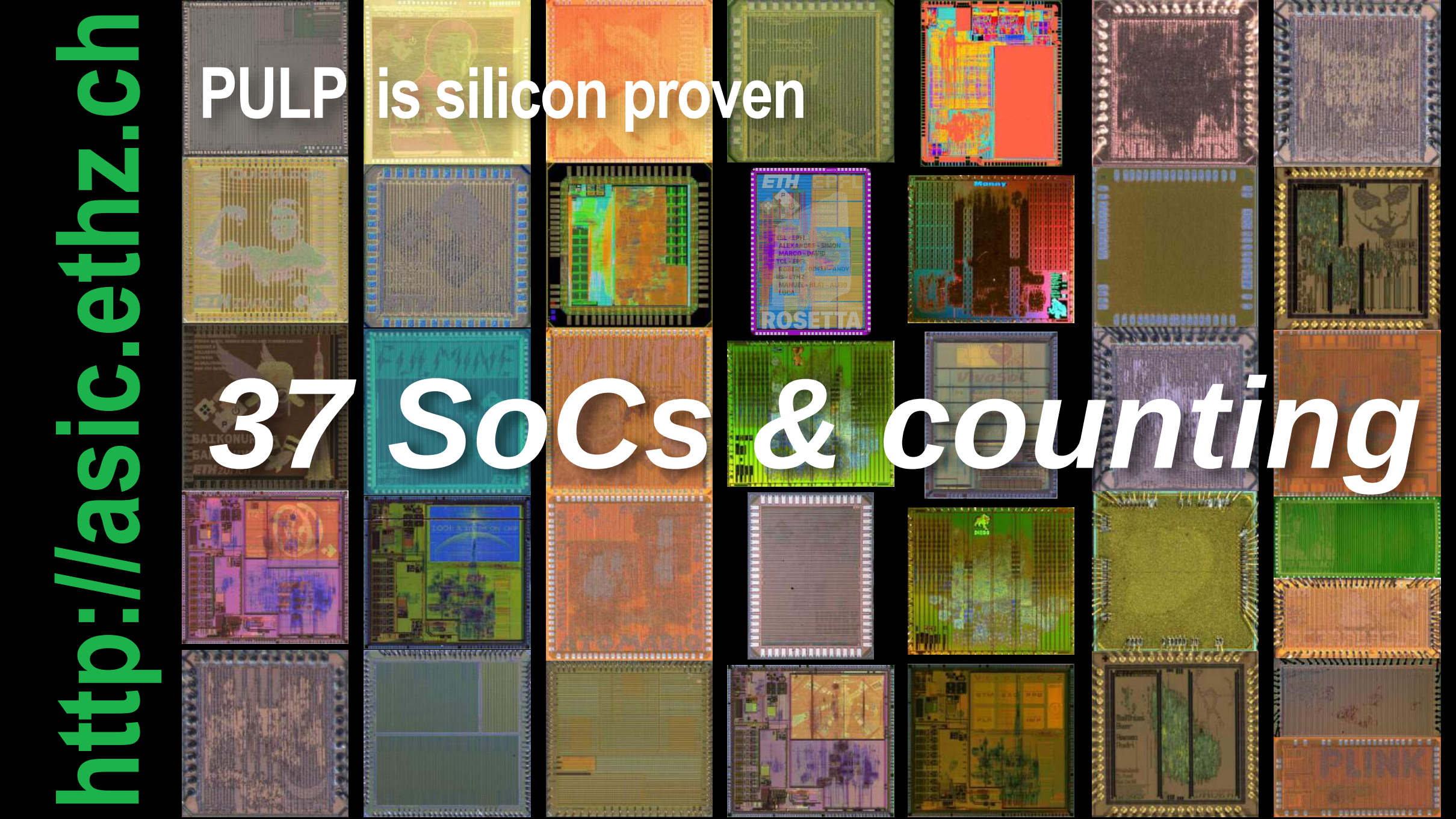
Nice, but what exactly is “open” in Open Source HW?

Nice, but what exactly is "open" in Open Source HW?

- Only the first stage of the silicon production pipeline can be open HW
  → **RTL source code** (in an HDL such as SystemVerilog)
- Later stages contain closed IP of various actors + tool licensing issues



Permissive, Copyright (e.g APACHE) License is Key for industrial adoption


PULP is silicon proven
37 SoCs & counting
http://asic.ethz.ch

PULP

Parallel Ultra Low Power

Luca Benini, Davide Rossi, Andrea Borghesi, Michele Magno, Simone Benatti, Francesco Conti, Francesco Beneventi, Daniele Palossi, Giuseppe Tagliavini, Antonio Pullini, Germain Haugou, Manuele Rusci, Florian Glaser, Fabio Montagna, Bjoern Forsberg, Pasquale Davide Schiavone, Alfio Di Mauro, Victor Javier Kartsch Morinigo, Tommaso Polonelli, Fabian Schuiki, Stefan Mach, Andreas Kurth, Florian Zaruba, Manuel Eggimann, Philipp Mayer, Marco Guermandi, Xiaying Wang, Michael Hersche, Robert Balas, Antonio Mastrandrea, Matheus Cavalcante, Angelo Garofalo, Alessio Burrello, Gianna Paulin, Georg Rutishauser, Andrea Cossettini, Luca Bertaccini, Maxim Mattheeuws, Samuel Riedel, Sergei Vostrikov, Vlad Niculescu, Hanna Mueller, Matteo Perotti, Nils Wistoff, Luca Bertaccini, Thorir Ingulfsson, Thomas Benz, Paul Scheffler, Alessio Burello, Moritz Scherer, Matteo Spallanzani, Andrea Bartolini, Frank K. Gurkaynak,

and many more that we forgot to mention

http://pulp-platform.org **@pulp_platform**