PULP

Parallel Ultra Low Power

# PULP: Open Hardware at the Edge of the IoT

14th HiPEAC Workshop on Reconfigurable Computing

20.01.2019

**Davide Rossi, University of Bologna**

[1]Department of Electrical, Electronic
and Information Engineering

ETH zürich

[2]Integrated Systems Laboratory

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

# Parallel Ultra Low Power (PULP)

- Project started in **2013**
- 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

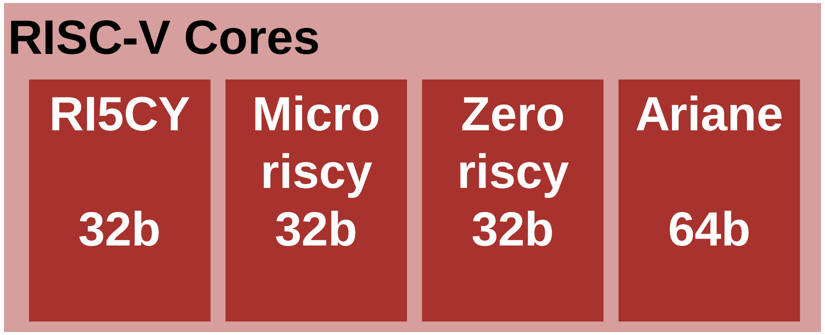
**How to get the most BANG for the ENERGY consumed in a computing system**

- 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)

# 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

- 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

## Accelerators

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

# All these components are combined into platforms

## RISC-V Cores

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

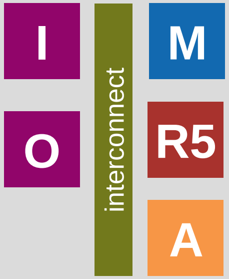
## Peripherals

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

## Interconnect

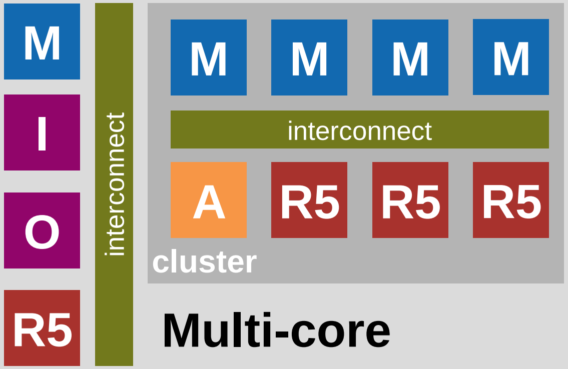
- Logarithmic interconnect
- APB – Peripheral Bus
- AXI4 – Interconnect

## Platforms



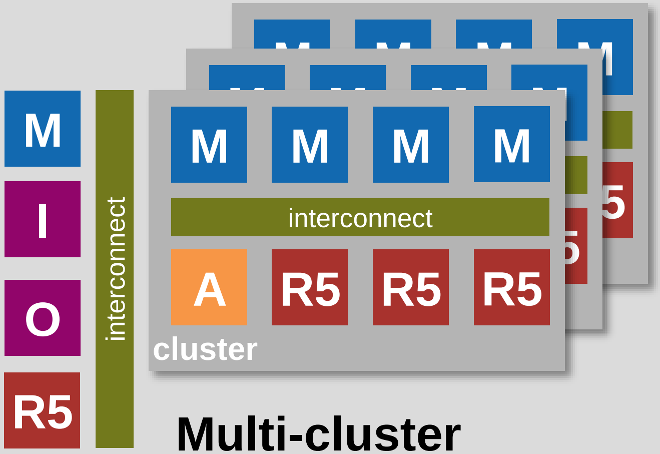
**Single Core**

- PULPino
- PULPissimo



**Multi-core**

- Fulmine
- Mr. Wolf



**Multi-cluster**

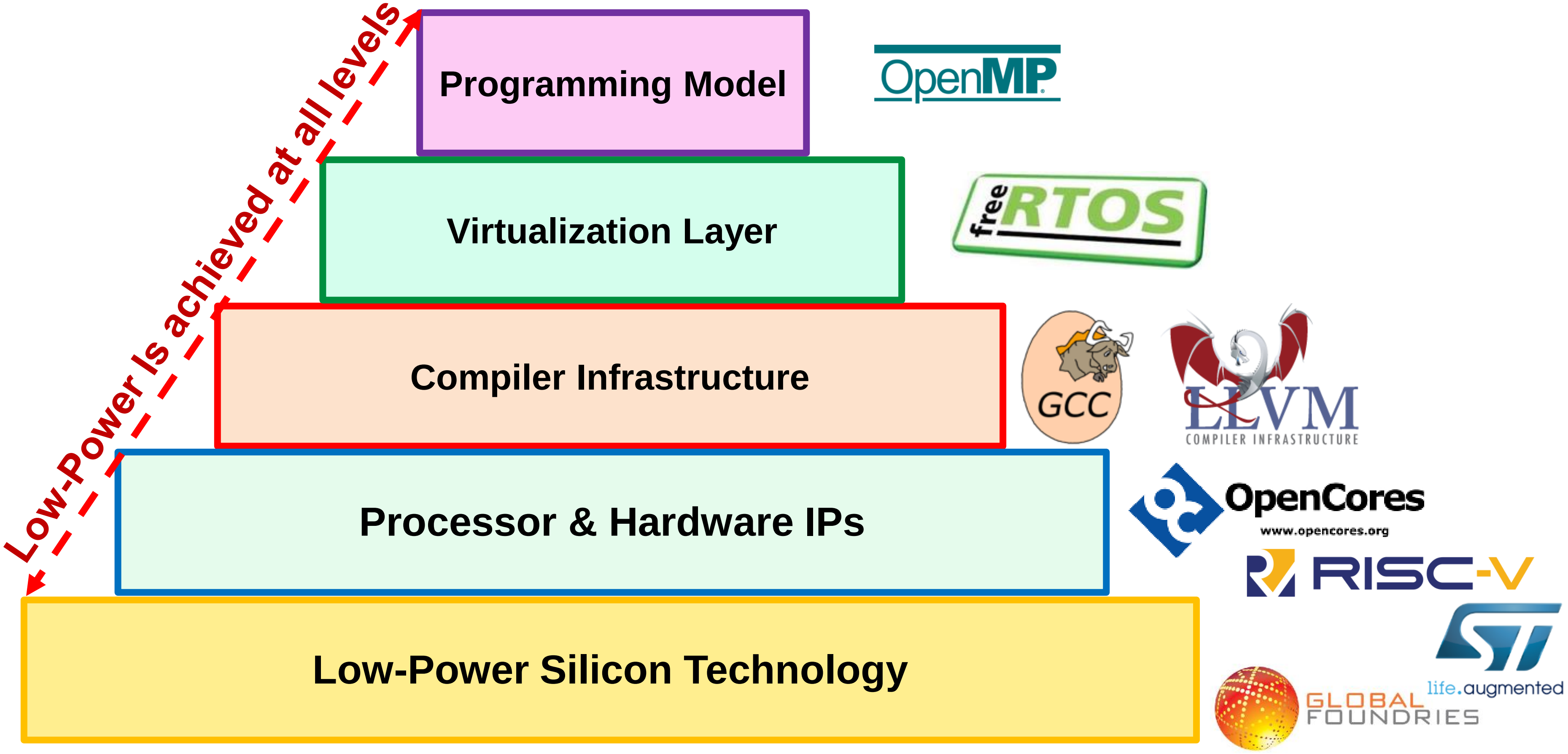
- Hero

IOT → HPC

## Accelerators

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

# Full stack including SW support is needed as well



Programming Model

Virtualization Layer

Compiler Infrastructure

Processor & Hardware IPs

Low-Power Silicon Technology

Low-Power Is achieved at all levels

**Parallel + Programmable + Heterogeneous ULP computing**

# RISC-V Instruction Set Architecture

- Started by UC-Berkeley in 2010
- RISC-V is an open standard governed by RISC-V 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
- The PULP project specializes in **efficient implementations of RISC-V cores and peripherals**

**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 family explained

## 32 bit

### Low Cost Core

- Zero-riscy
  - RV32-ICM
- Micro-riscy
  - RV32-CE

*ARM Cortex-M0+*

### Core with DSP enhancements

- RI5CY
  - RV32-ICMX
    - SIMD
    - HW loops
    - Bit manipulation
    - Fixed point

*ARM Cortex-M4*

### Floating-point capable Core

- RI5CY+FPU
  - RV32-ICMFX

*ARM Cortex-M4F*

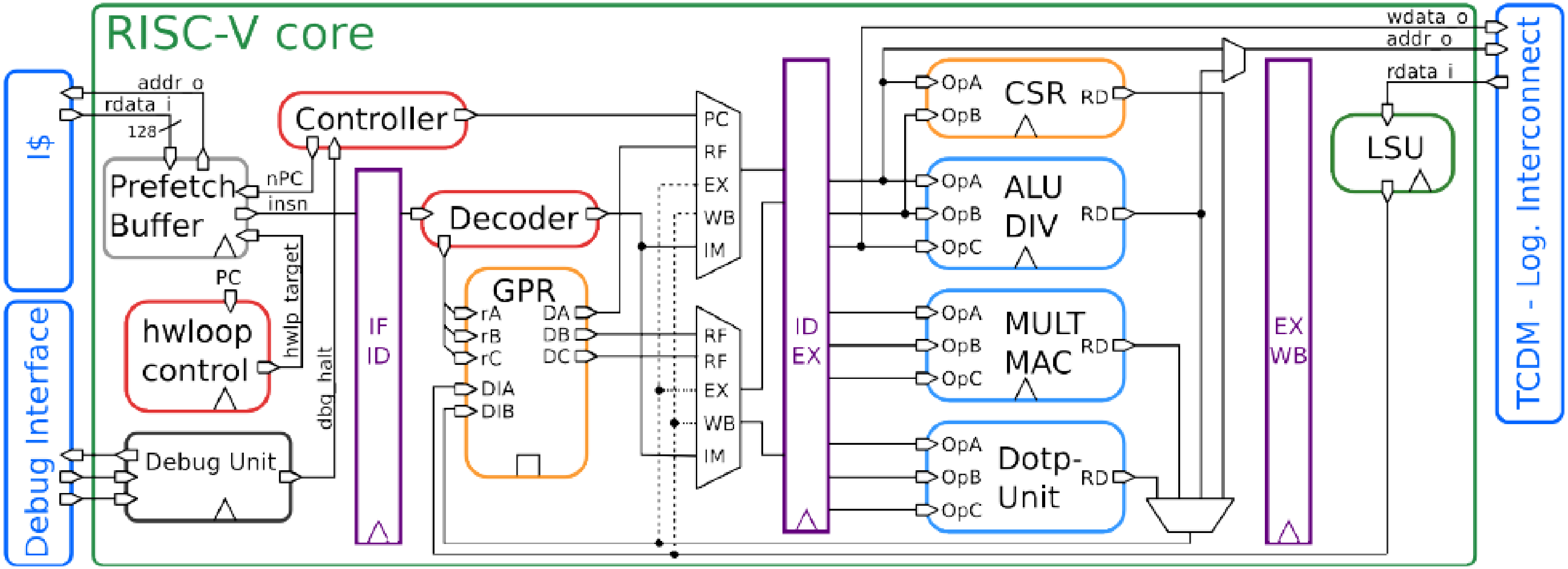
## 64 bit

### Linux capable Core

- Ariane
  - RV64-IMAFDCX
  - Full privilege specification

*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 (Xpulp) to RISC-V for DSP applications

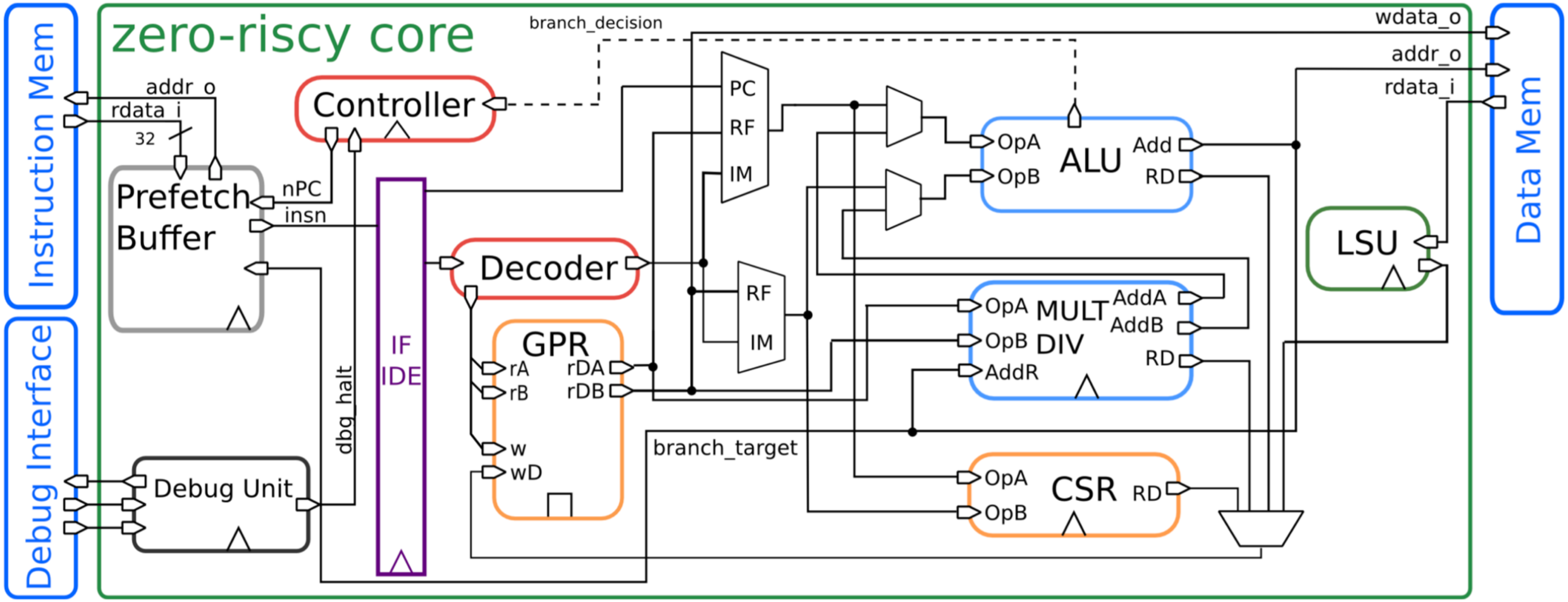
# Our extensions to RI5CY (with additions to GCC)

- **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`)

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

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



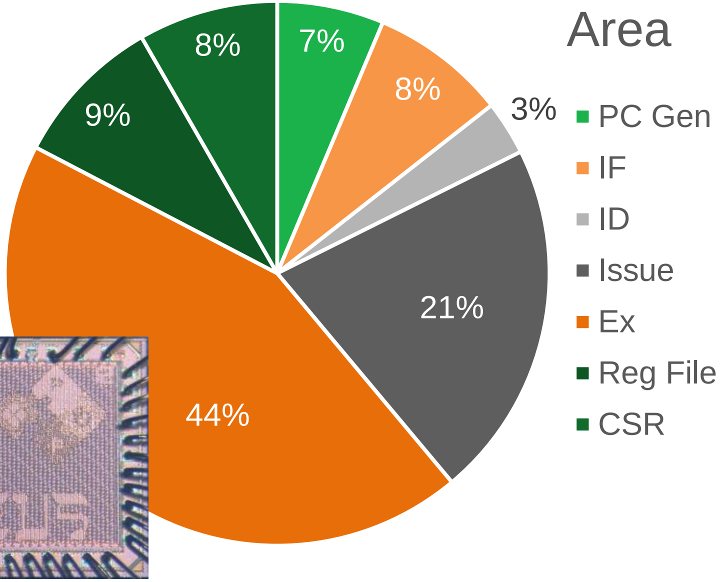
- Only 2-stage pipeline, simplified register file
- **Zero-Riscy** (RV32-ICM), 19kGE, 2.44 Coremark/MHz
- **Micro-Riscy** (RV32-EC), 12kGE, 0.91 Coremark/MHz
- Used as SoC level controller in newer PULP systems

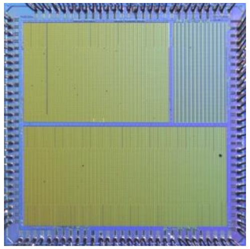
# Finally the step into 64-bit cores

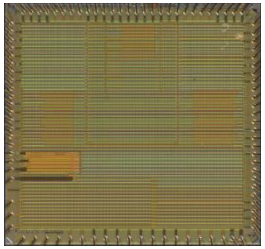
- **For the first 4 years of the PULP project we used only 32bit cores**
  - Most IoT applications work well with 32bit cores.
  - A typical 64bit core is much more than 2x the size of a 32bit core.
- **But times change:**
  - Using a 64bit Linux capable core allows you to share the same address space 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.

# Main properties of Ariane

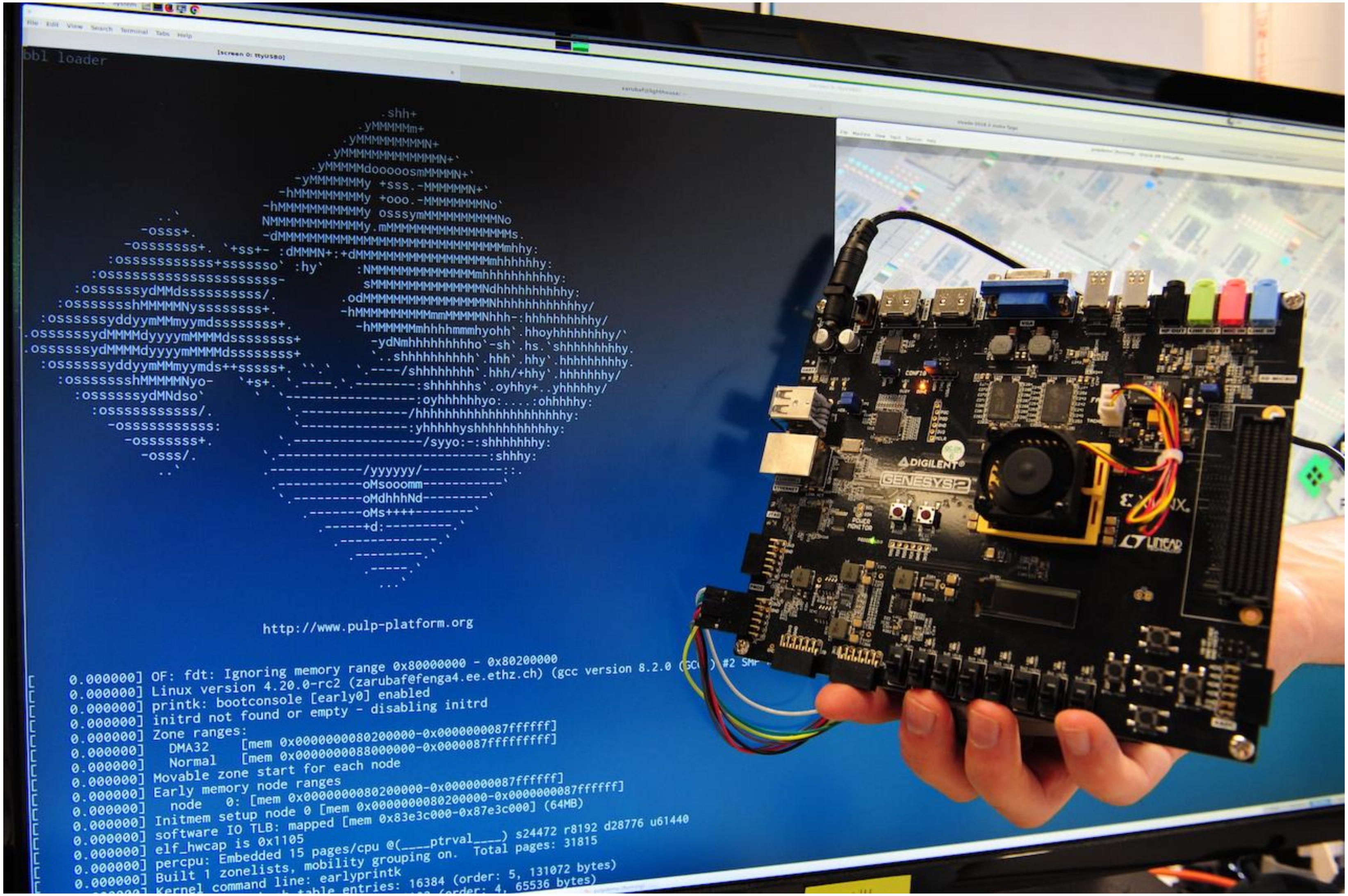
- Tuned for high frequency, 6 stage pipeline, integrated cache
  - In order issue, out-of-order write-back, in-order-commit
  - Supports privilege spec 1.11, M, S and U modes
  - Hardware Page Table Walker
- Implemented in GF22nm (Poseidon, Kosmodrom), and UMC65 (Scarabaeus)
  - In 22nm: ~1 GHz worst case conditions (SSG, 125/-40C, 0.72V)
  - 8-way 32kByte Data cache and 4-way 32kByte Instruction Cache
  - Core area: 175 kGE







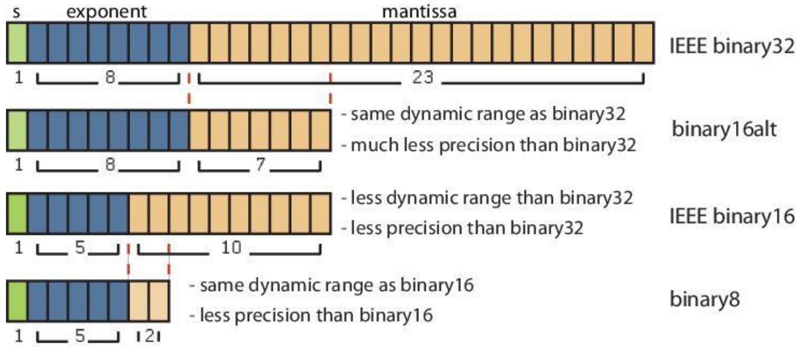
# Ariane booting Linux on a Digilent Genesys 2 board



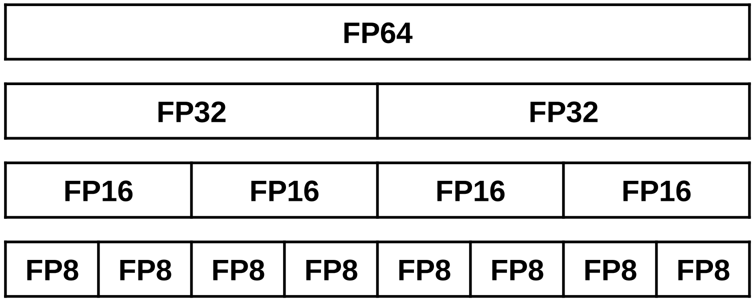
```
bbl loader

                    http://www.pulp-platform.org

[    0.000000] OF: fdt: Ignoring memory range 0x80000000 - 0x80200000
[    0.000000] Linux version 4.20.0-rc2 (zarubaf@fenga4.ee.ethz.ch) (gcc version 8.2.0 (GCC)) #2 SMP
[    0.000000] printk: bootconsole [early0] enabled
[    0.000000] initrd not found or empty - disabling initrd
[    0.000000] Zone ranges:
[    0.000000]   DMA32    [mem 0x0000000080200000-0x0000000087ffffff]
[    0.000000]   Normal   [mem 0x0000000088000000-0x0000087fffffffff]
[    0.000000] Movable zone start for each node
[    0.000000] Early memory node ranges
[    0.000000]   node   0: [mem 0x0000000080200000-0x0000000087ffffff]
[    0.000000] Initmem setup node 0 [mem 0x0000000080200000-0x0000000087ffffff]
[    0.000000] software IO TLB: mapped [mem 0x83e3c000-0x87e3c000] (64MB)
[    0.000000] elf_hwcap is 0x1105
[    0.000000] percpu: Embedded 15 pages/cpu @(____ptrval____) s24472 r8192 d28776 u61440
[    0.000000] Built 1 zonelists, mobility grouping on.  Total pages: 31815
[    0.000000] Kernel command line: earlyprintk
                        table entries: 16384 (order: 5, 131072 bytes)
                                       (order: 4, 65536 bytes)
```

# 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**
- **Alternative FP Format** support (<32 bit)


s
exponent
mantissa
IEEE binary32
1 8 23
- same dynamic range as binary32
- much less precision than binary32
binary16alt
1 8 7
- less dynamic range than binary32
- less precision than binary32
IEEE binary16
1 5 10
- same dynamic range as binary16
- less precision than binary16
binary8
1 5 2

**Packed-SIMD** support for all formats



FP64

FP32 FP32

FP16 FP16 FP16 FP16

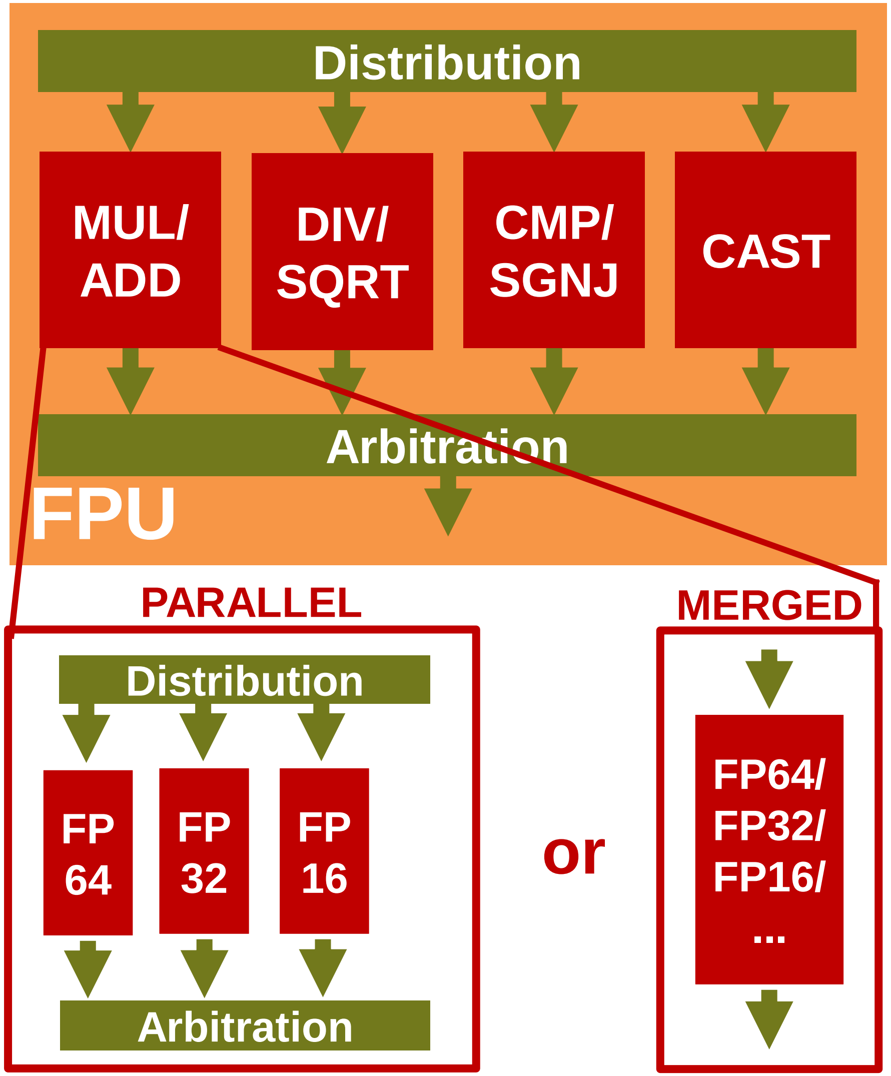
FP8 FP8 FP8 FP8 FP8 FP8 FP8 FP8

**Unified** FP/Integer register file

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

# Parametric Floating-Point Unit for Transprecision

- **Main FP operation groups**
  - **MUL/ADD**: Add/Subtract, Multiply, FMA
  - **CMP/SGNJ**: Comparisons, Min/Max etc.
  - **CAST**: FP-FP casts, Int-FP / FP-Int casts
- **Parametrizable**
  - Number & Encoding of **Formats**
  - Packed-SIMD **Vectors**
  - **# Pipeline Stages** (per Op and Format)
  - **Implementation** (per Op and Format)
    - **PARALLEL** for best Speed
    - **MERGED** (or Iterative) for best Area
- **Special Functions** for Transprecision
  - Cast-and-Pack 2 FP Values to Vector
  - Casts amongst FP Vectors + Repacking
  - Expanding FMA (e.g. FP32 += FP16*FP16)



# The pulp-platforms put everything together

## RISC-V Cores

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

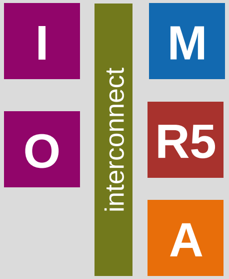
## Peripherals

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

## Interconnect

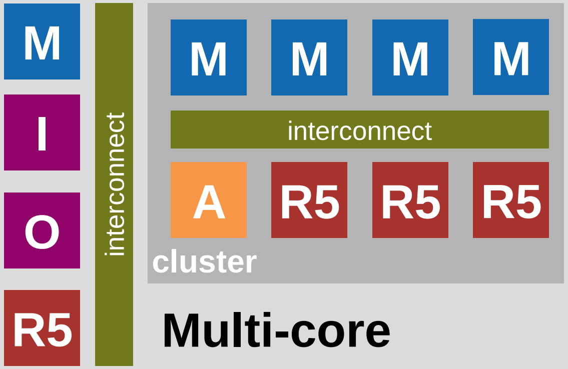
- Logarithmic interconnect
- APB – Peripheral Bus
- AXI4 – Interconnect

## Platforms



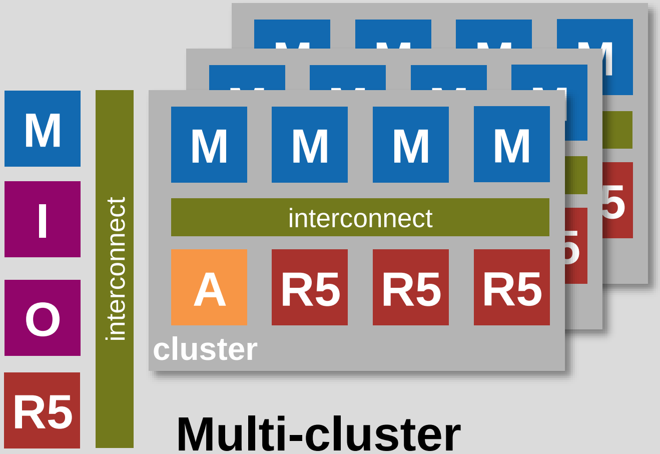
**Single Core**

- PULPino
- PULPissimo



**Multi-core**

- Fulmine
- Mr. Wolf



**Multi-cluster**

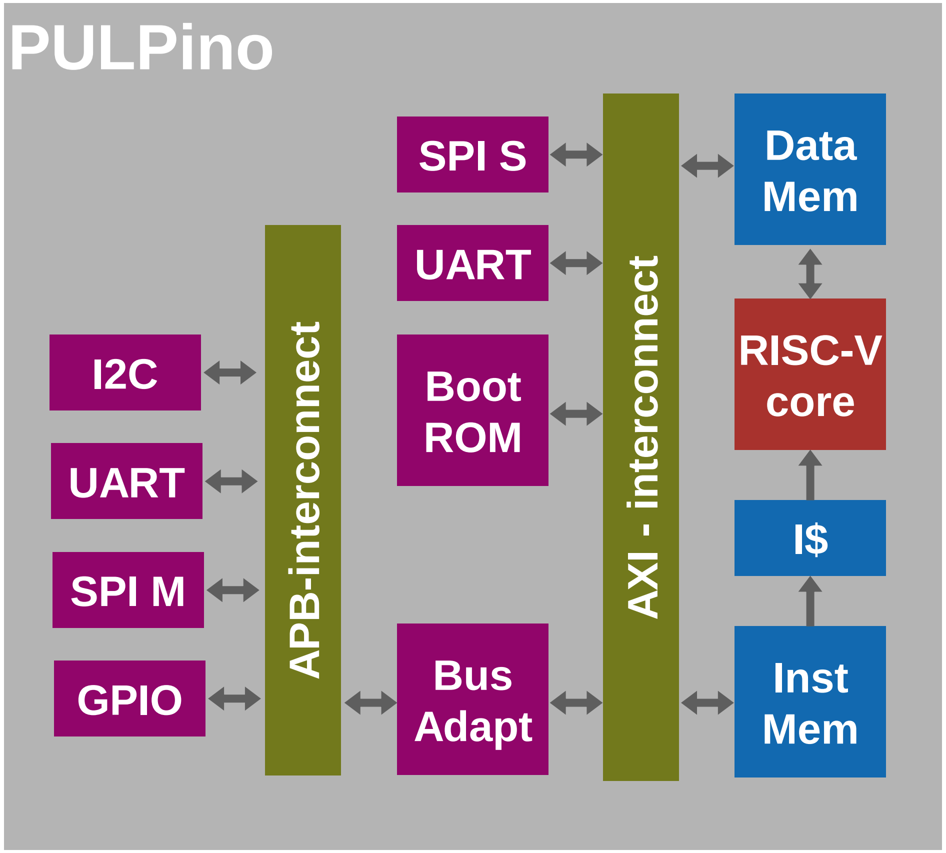
- Hero

## Accelerators

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

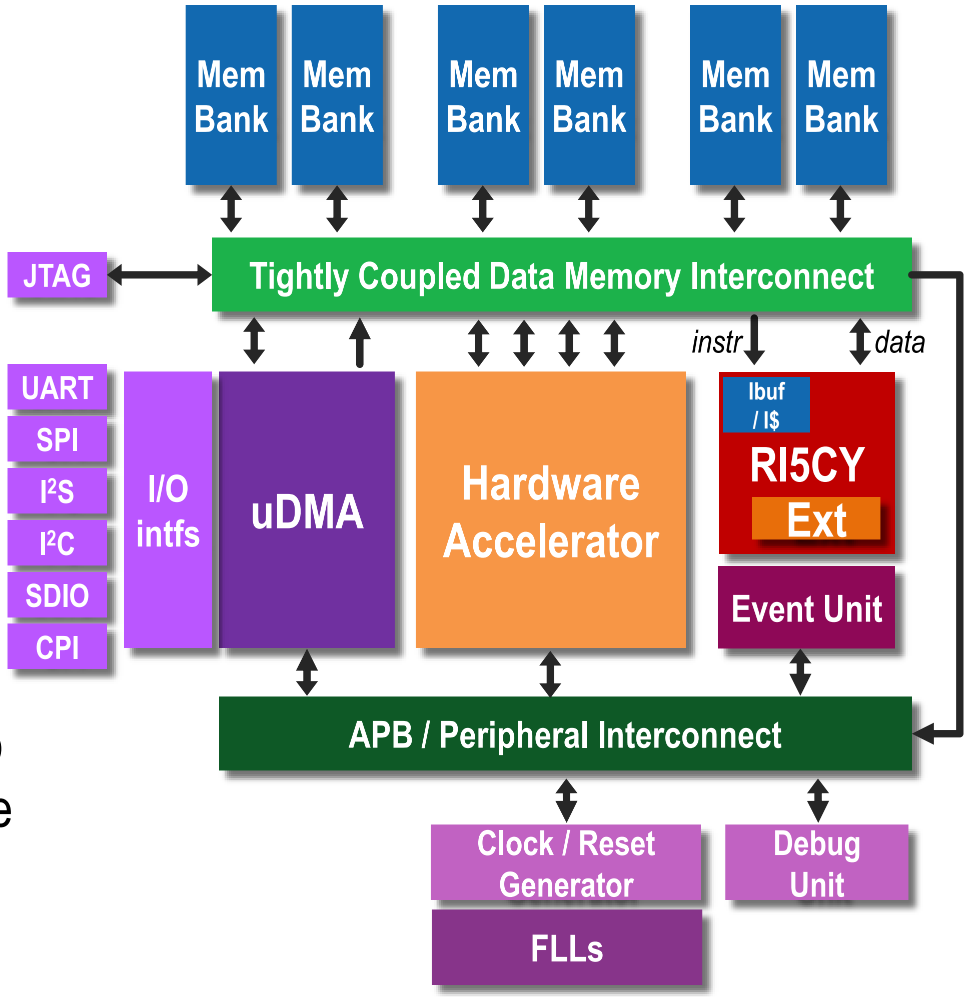
# PULPino our first single core platform

- **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

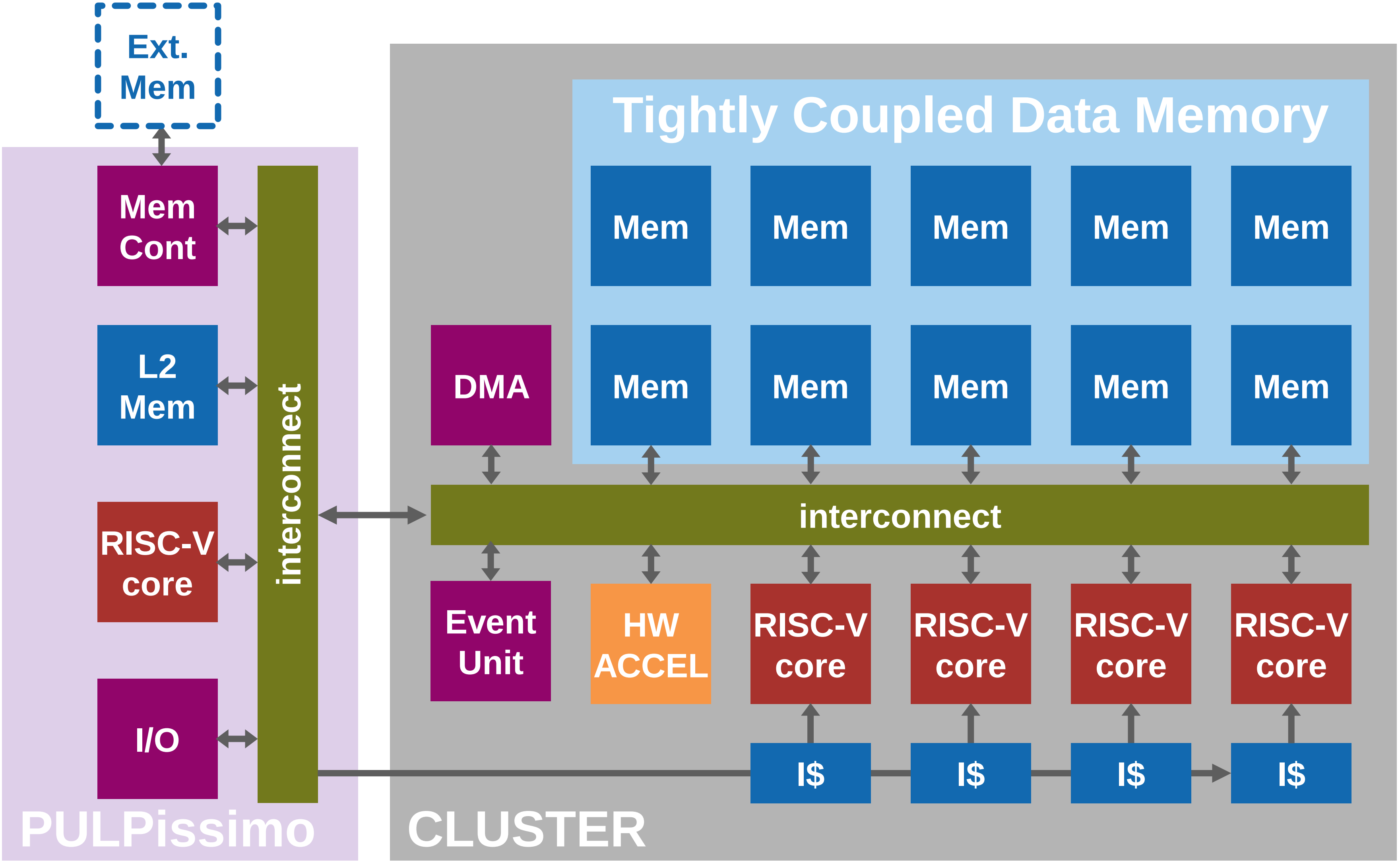


# PULPissimo the improved single core platform

- **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**



# The main PULP systems we develop are cluster based



# Multi-cluster PULP systems for HPC applications

## RISC-V Cores

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

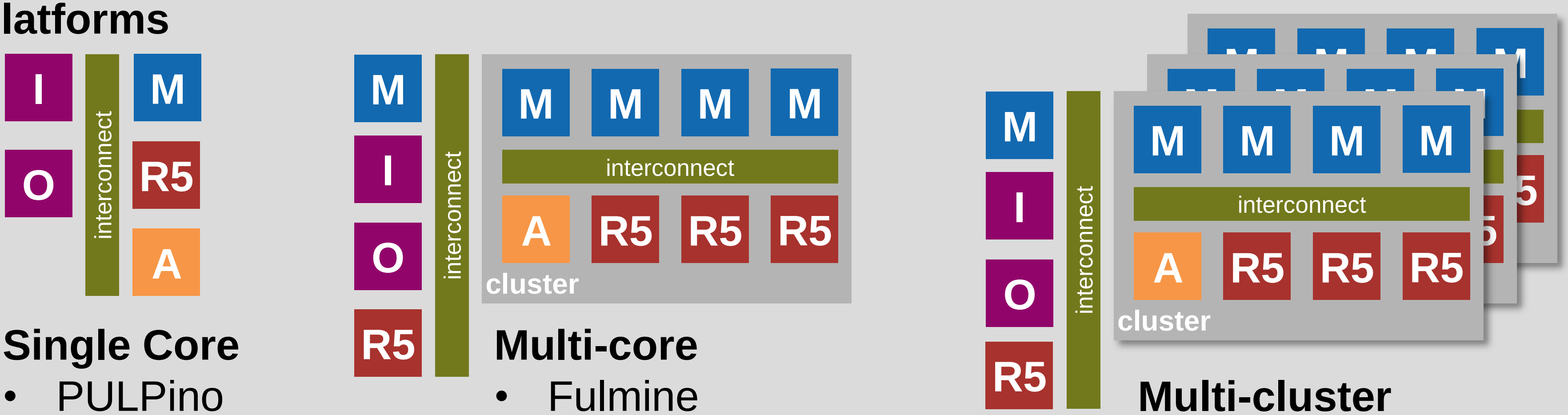
## Peripherals

- 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

IOT → HPC

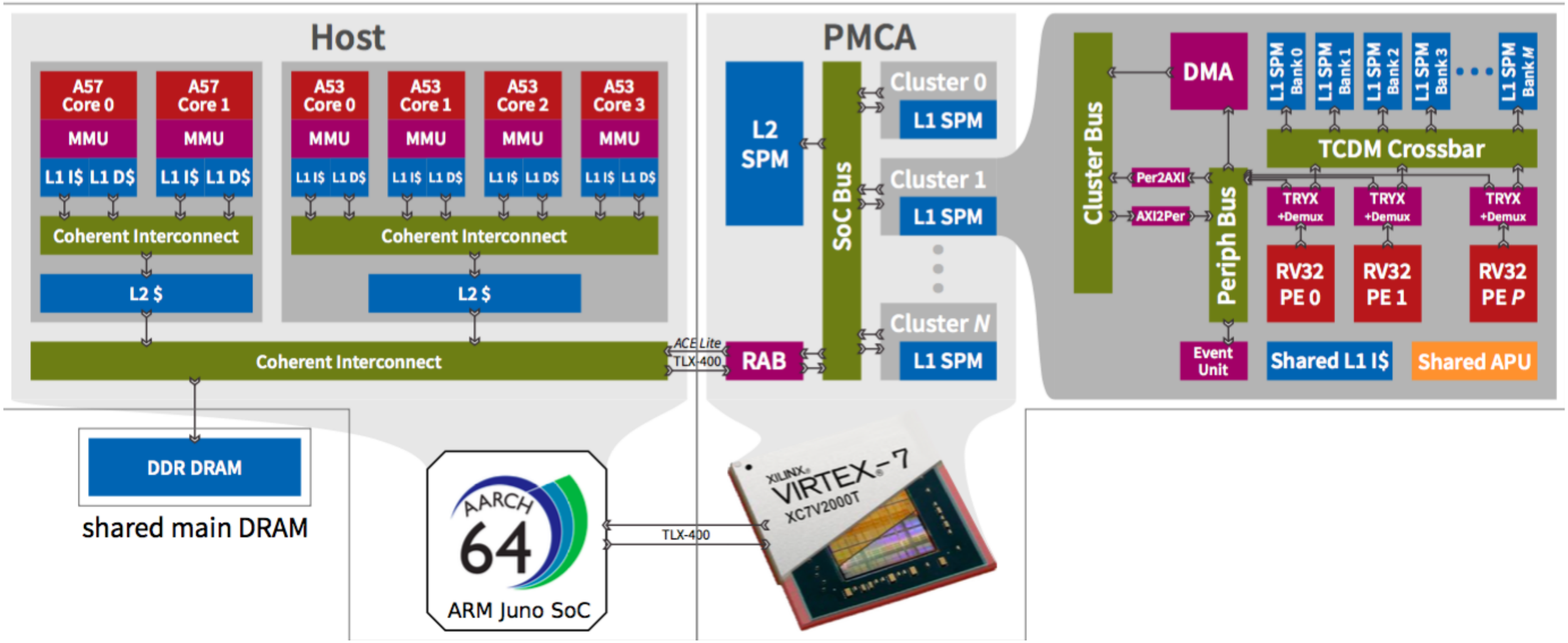
## Accelerators

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

# Heterogenous Research Platform

industry-standard, hard-macro
Arm Cortex-A Host processor

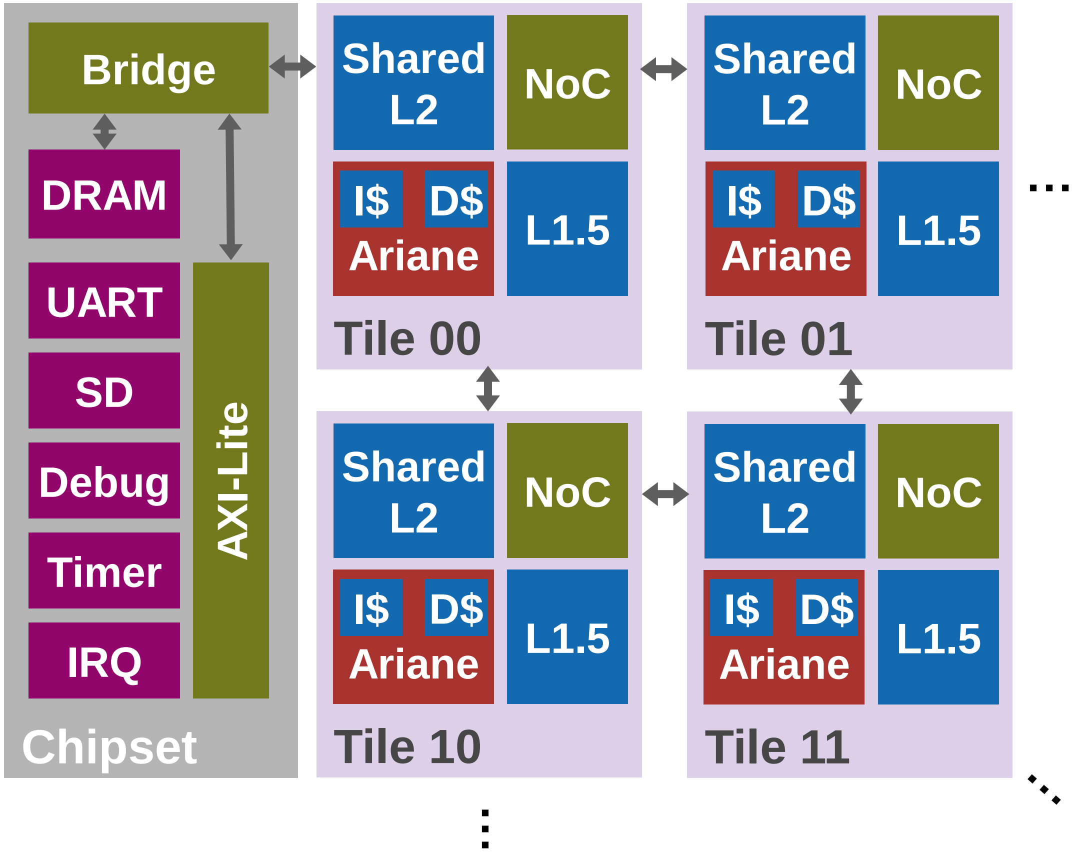
scalable, configurable, modifiable FPGA implementation
of PULP (silicon-proven, cluster-based PMCA with RISC-V PEs)



- First released in 2018
- Allows a PULP cluster to be connected to a host system

# OpenPiton and Ariane together, the many-core system

- **OpenPiton**
  - Developed by Princeton
  - Originally OpenSPARC T1
  - Scalable NoC with coherent LLC
  - Tiled Architecture



# PULP Accelerators for improved efficiency

## RISC-V Cores

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

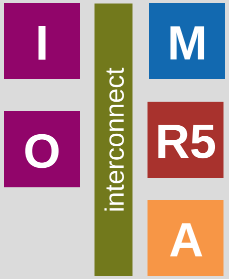
## Peripherals

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

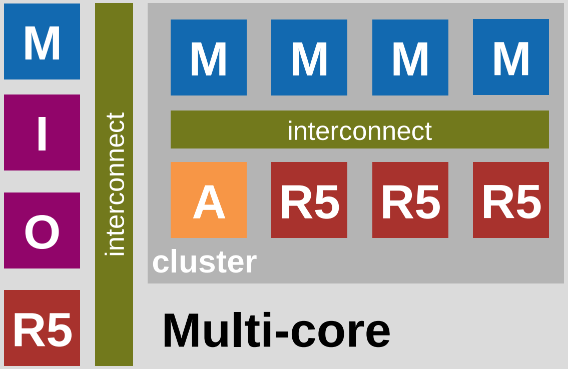
## Interconnect

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

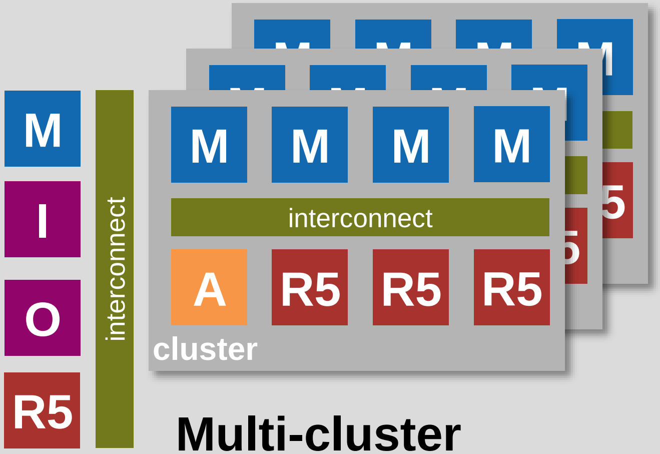
## Platforms



**Single Core**
- PULPino
- PULPissimo



**Multi-core**
- Fulmine
- Mr. Wolf



**Multi-cluster**
- Hero

## Accelerators

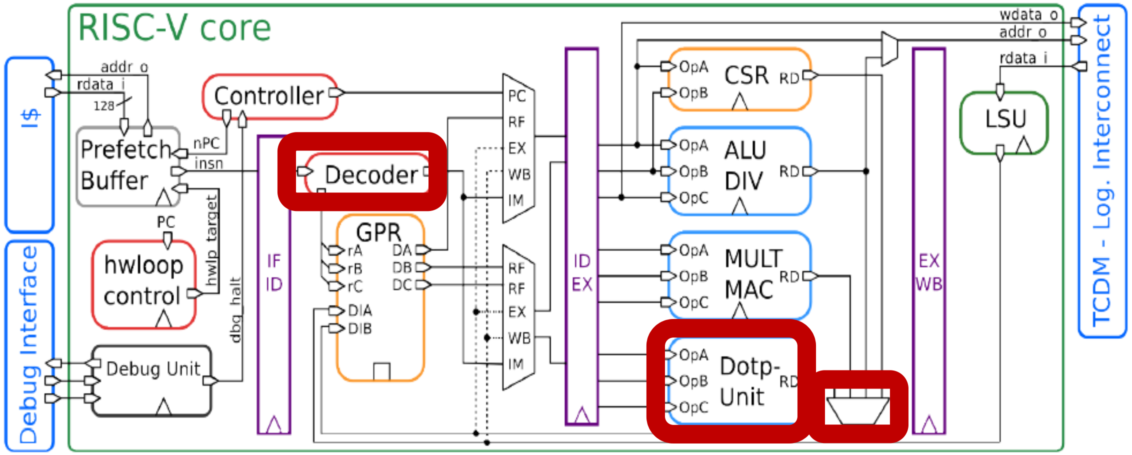
| HWCE (convolution) | Neurostream (ML) | HWCrypt (crypto) | PULPO ($1^{st}$ order opt) |
|---|---|---|---|

# How to accelerate processing in PULP systems

- **Standard peripheral talking over AXI/APB**
  - Common in practice, nothing new or special
- **Instruction set extensions**
  - Possible in RISC-V, RI5CY has many new DSP instructions
  - Goal is to get '**good instructions**' into standard extensions at some point
- **Shared functional units**
  - Amortizes expensive extensions (FPU/DIV) between multiple units
- **Additional pipeline stages**
  - Work in Patronus for Control flow Integrity
- **Shared memory accelerators**
  - Our bread and butter, PULPopen, NTX
- **Cluster as an accelerator**
  - HERO, BigPULP, etc

# What Kind of Acceleration:  ISA Extensions

- High Flexibility, relatively small performance boost
- Integrated in the Pipeline of Processors (ID Stage, EX Stage, WB Stage)
- Suffer from Register File Bandwidth Bottleneck (only 2-3 operands…)
- Require Adapting Compiler and Binutils
- Auxiliary Processing Units (APU), interface available in the RI5CY
- Examples: Dot product (already implemented), bit-reverse, butterfly…

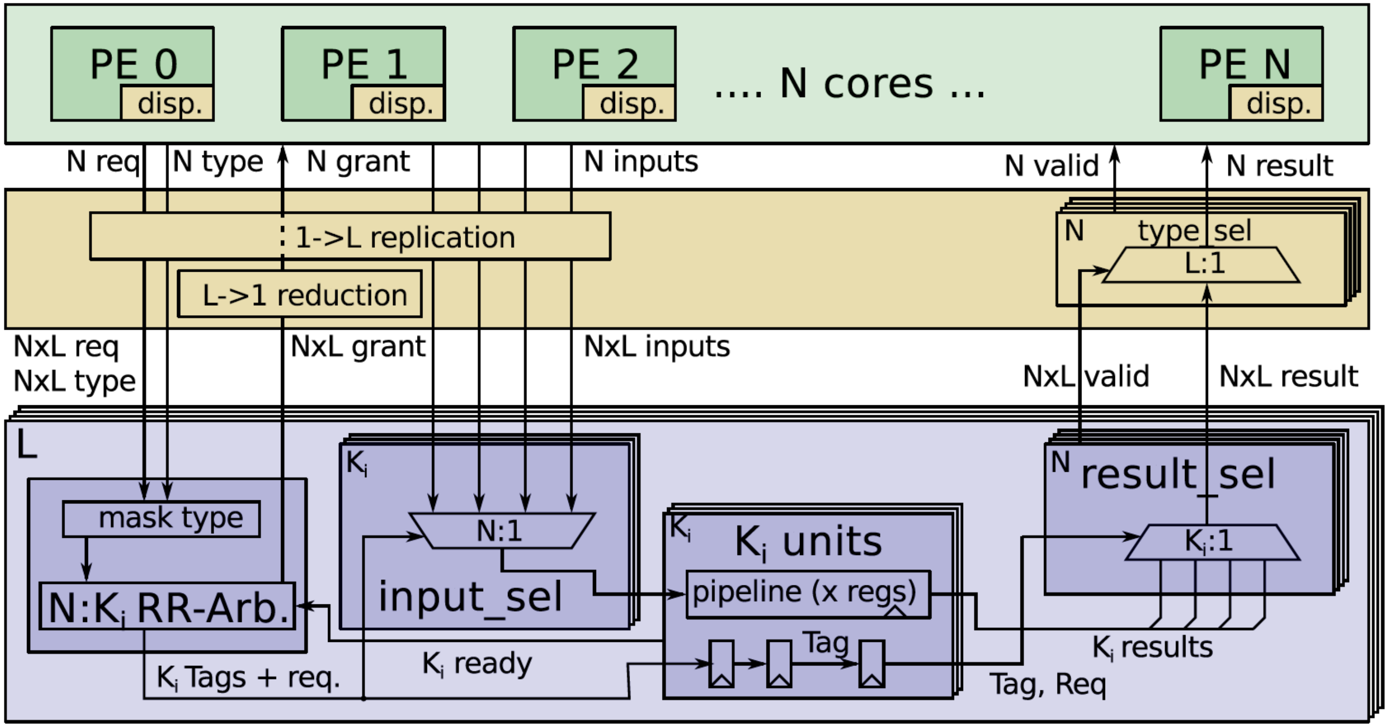


*Programming model*

```
#define SumDotp(a, b, c) \
__builtin_pulp_sdotsp2(a, b, c)

for (int k = 0; k < (N>>1); k++) {
    VA = VectInA[k];
    VB = VectInB[k];
    S = SumDotp(VA, VB, S);
}
```

# What Kind of Acceleration: Shared functional units

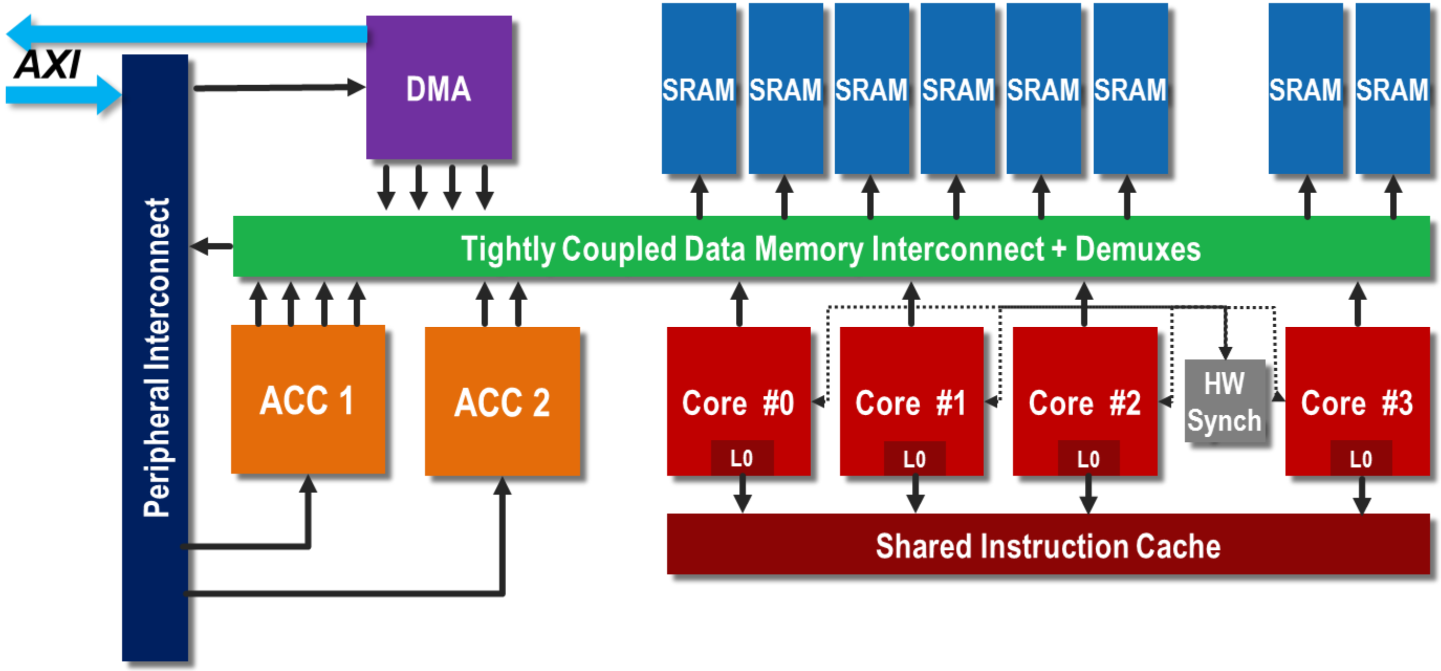
- The same as previous one, but one unit can be shared among multiple cores
- Useful to save area for low-utilization instructions (i.e. < 1/#cores%)
- Examples: Shared FPU (SQRT, DIV…)



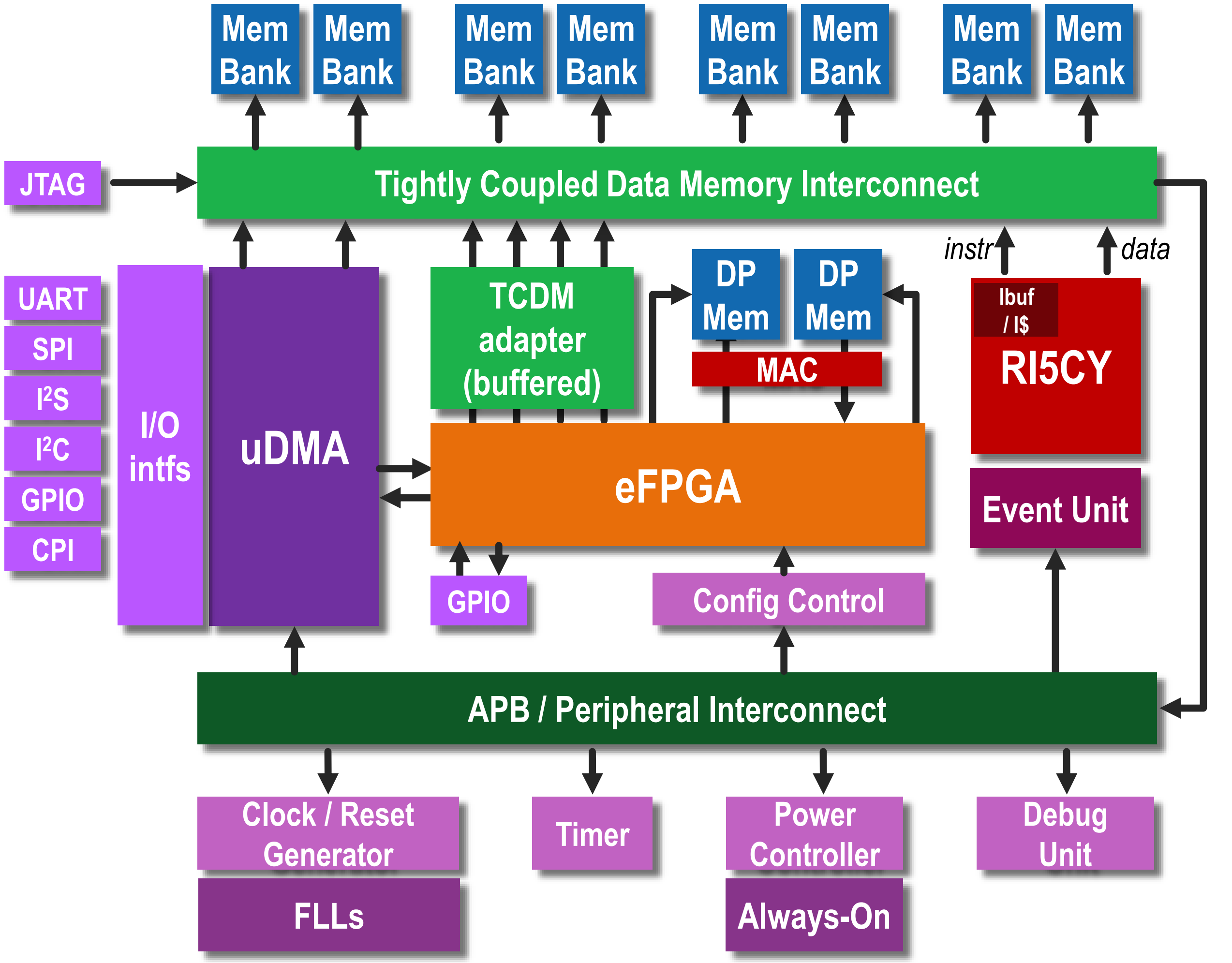
# What Kind of Acceleration: Shared memory accelerators

## Coarse-Grained Shared-Memory Accelerators

- DFGs mapped In Hardware (ILP + DLP) →Highest Efficiency, Low Flexibility
- Sharing data memory with processor for fast communication → low overhead
- Controlled through a memory-mapped interface
- Typically one/two accelerators shared by multiple cores



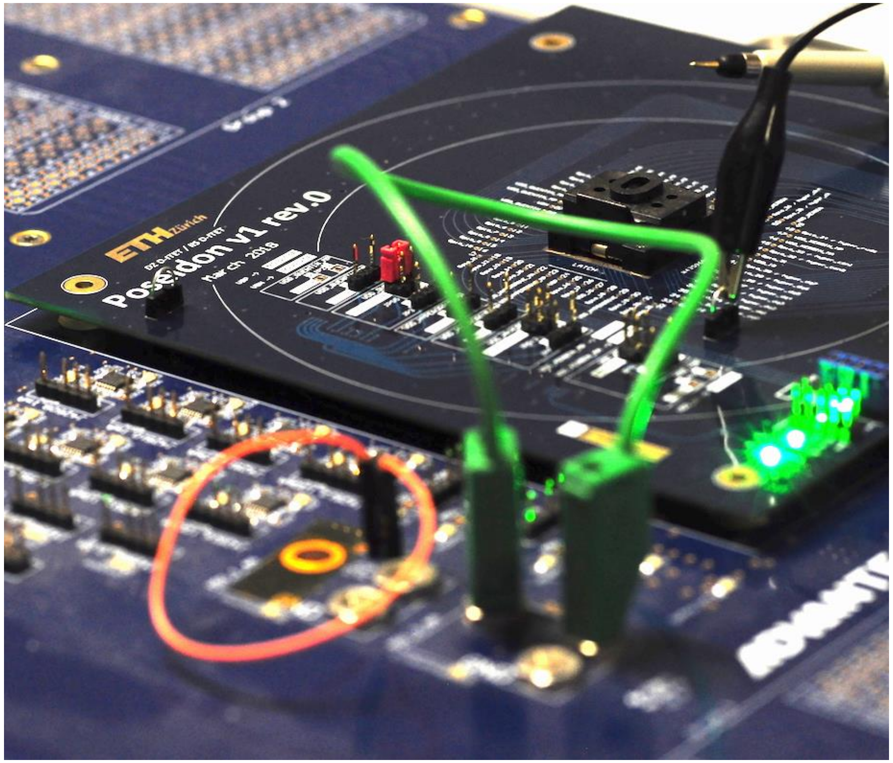
# The Quicklogic eFPGA integration in PULPissimo



*Blocks not drawn to scale*

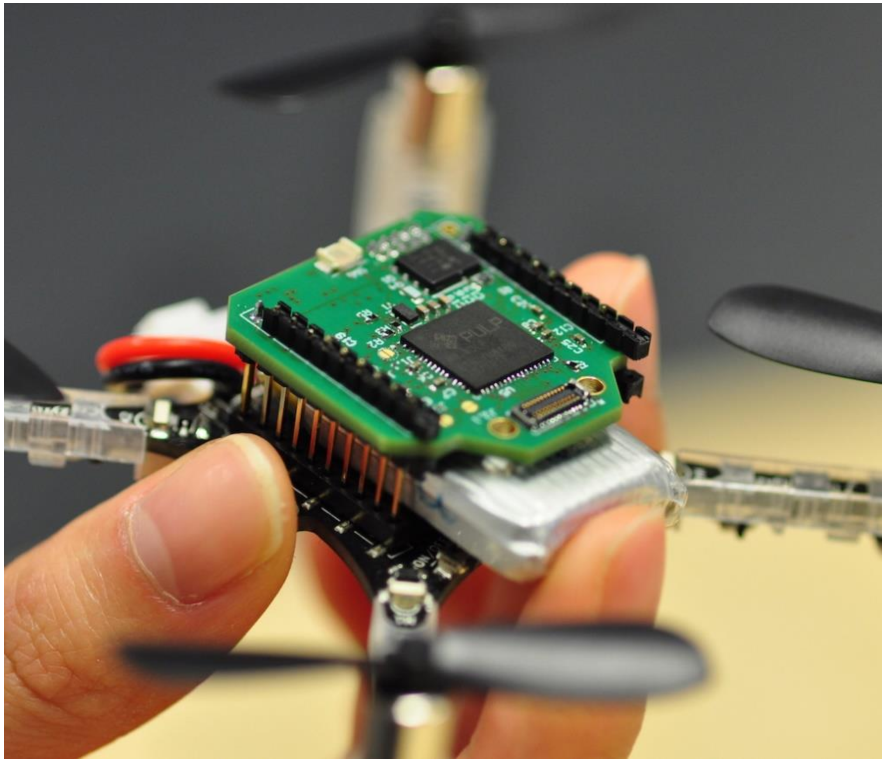
- APB port for configuration, programming and control
- Direct 4x TCDM access for eFPGA
  - 128 bits/cycle
  - 4 independent R/W
- Possibility to use uDMA

# We have designed more than 25 ASICs based on PULP



### ASICs meant for applications

- More peripherals (SPI, Camera)
- More on-chip memory



### ASICs meant to go on IC Tester

- Mainly characterization
- Not so many peripherals

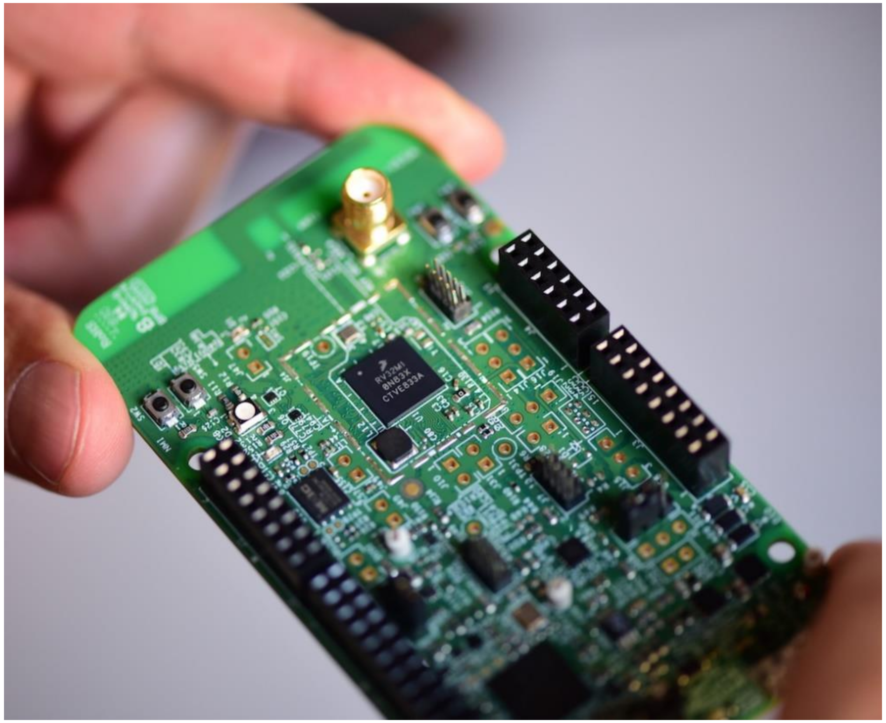
PULP

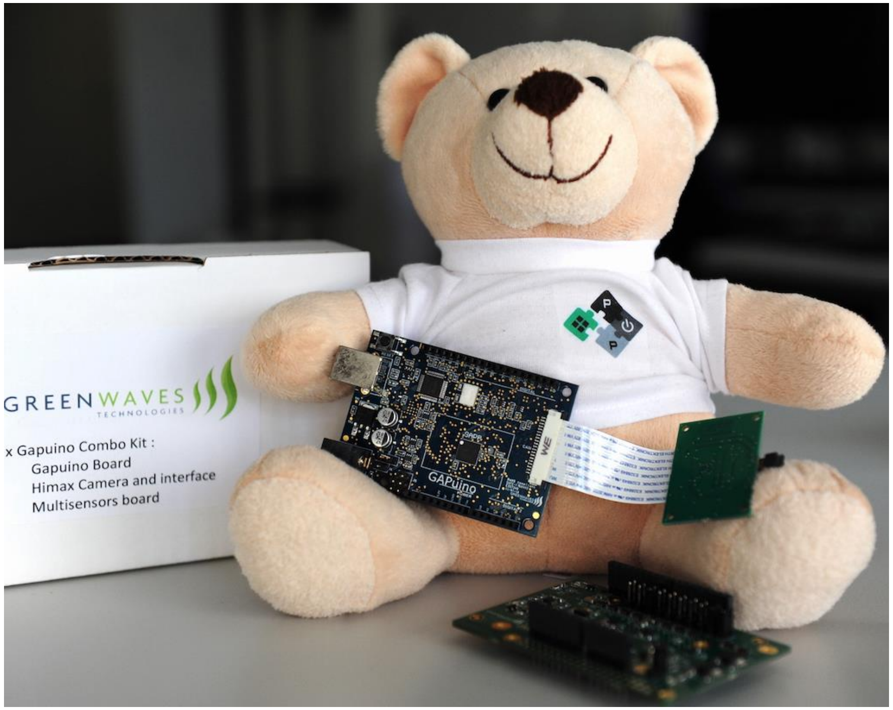
ETH

# You can buy development boards with PULP technology

## VEGA board from open-isa.org

- Micro-controller board with RI5CY and zero-riscy

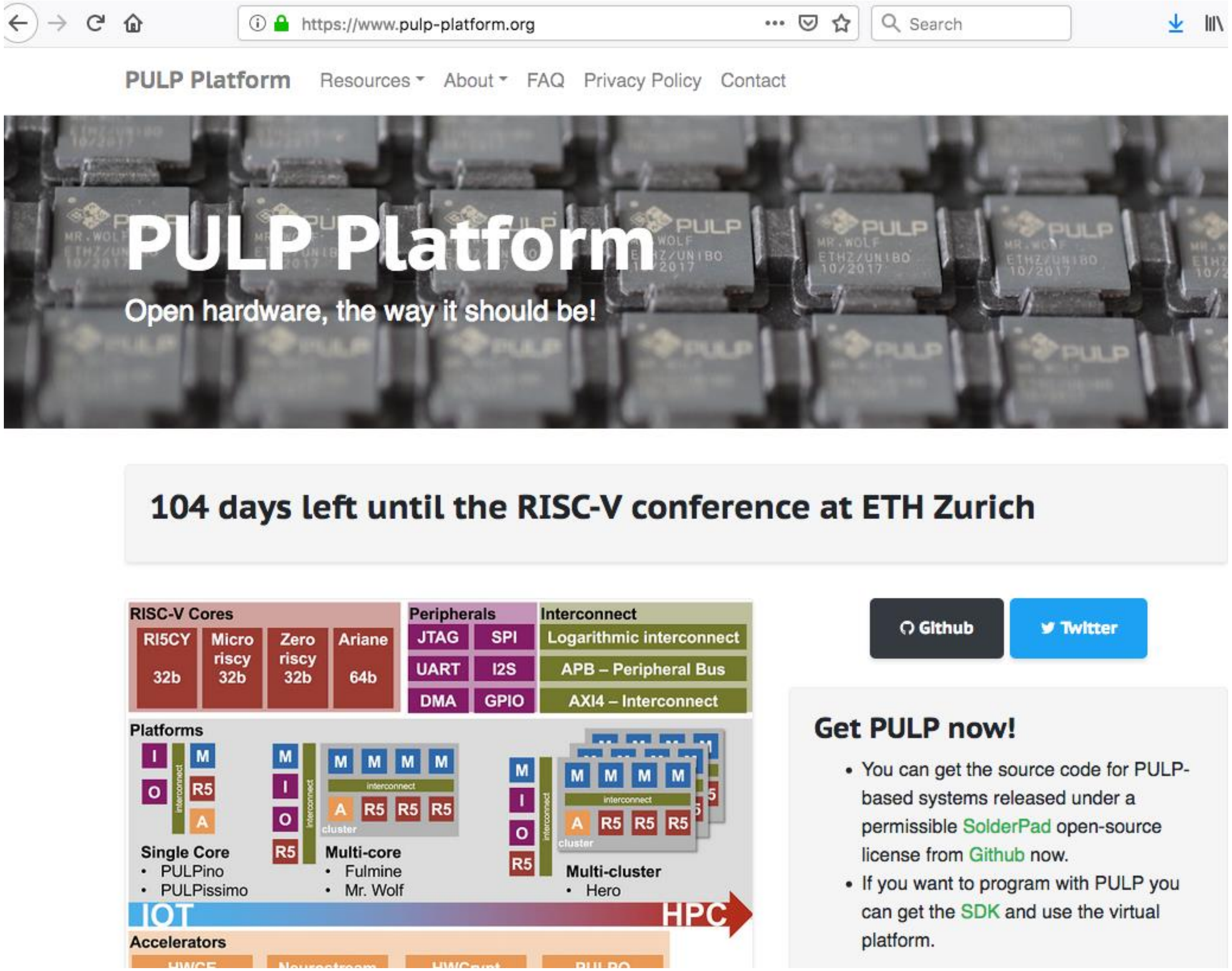




## GAPUINO from Greenwaves

- PULP cluster system with Nine RI5CY cores

# We firmly believe in Open Source movement



PULP Platform

Open hardware, the way it should be!

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

## 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.

First launched in February 2016 (Github)

All our development is on open repositories

Contributions from many groups

# 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.

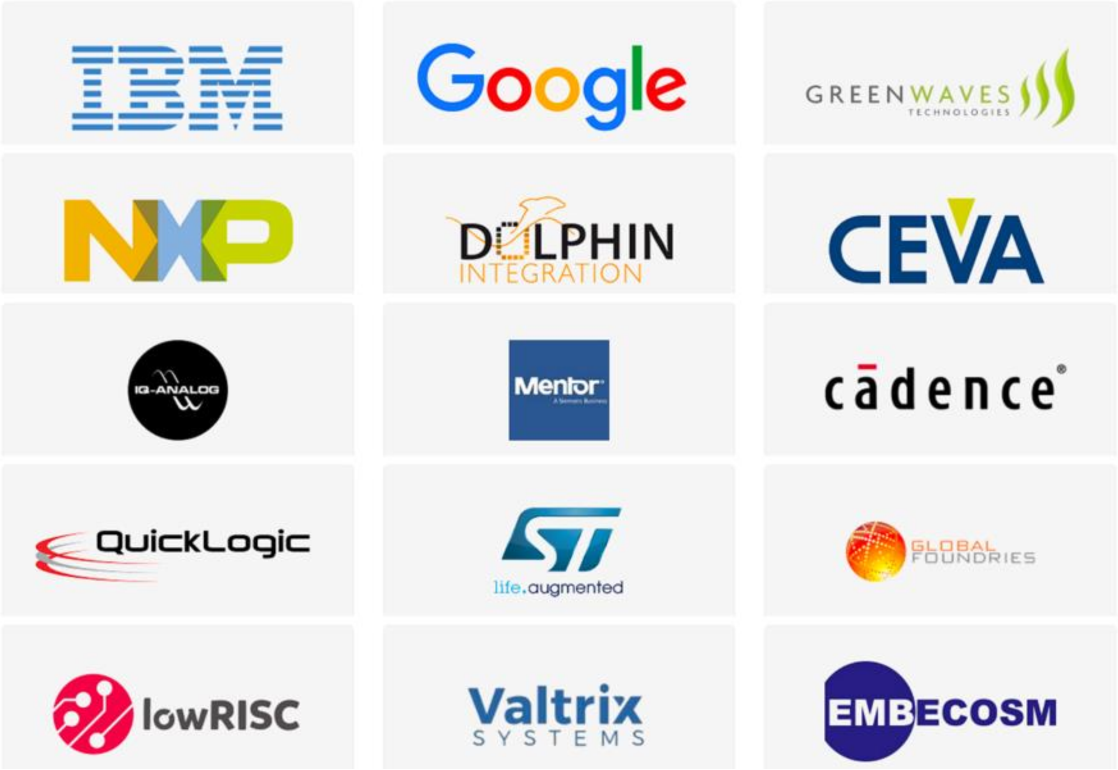
SOLDER Pad

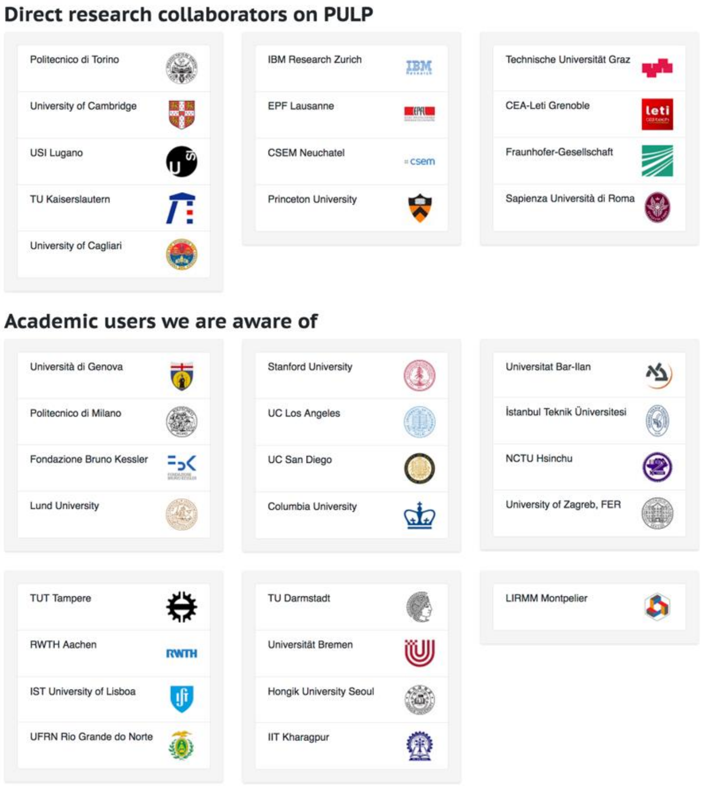
http://www.solderpad.org/licenses/

- Note the difference to **GPL**
  - Systems that include PULP do not have to be open source (Copyright not Copyleft)
  - They can be released commercially

# 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**

- Politecnico di Torino
- University of Cambridge
- USI Lugano
- TU Kaiserslautern
- University of Cagliari

- IBM Research Zurich
- EPF Lausanne
- CSEM Neuchatel
- Princeton University

- Technische Universität Graz
- CEA-Leti Grenoble
- Fraunhofer-Gesellschaft
- Sapienza Università di Roma

**Academic users we are aware of**

- Università di Genova
- Politecnico di Milano
- Fondazione Bruno Kessler
- Lund University

- Stanford University
- UC Los Angeles
- UC San Diego
- Columbia University

- Universitat Bar-Ilan
- İstanbul Teknik Üniversitesi
- NCTU Hsinchu
- University of Zagreb, FER

- TUT Tampere
- RWTH Aachen
- IST University of Lisboa
- UFRN Rio Grande do Norte

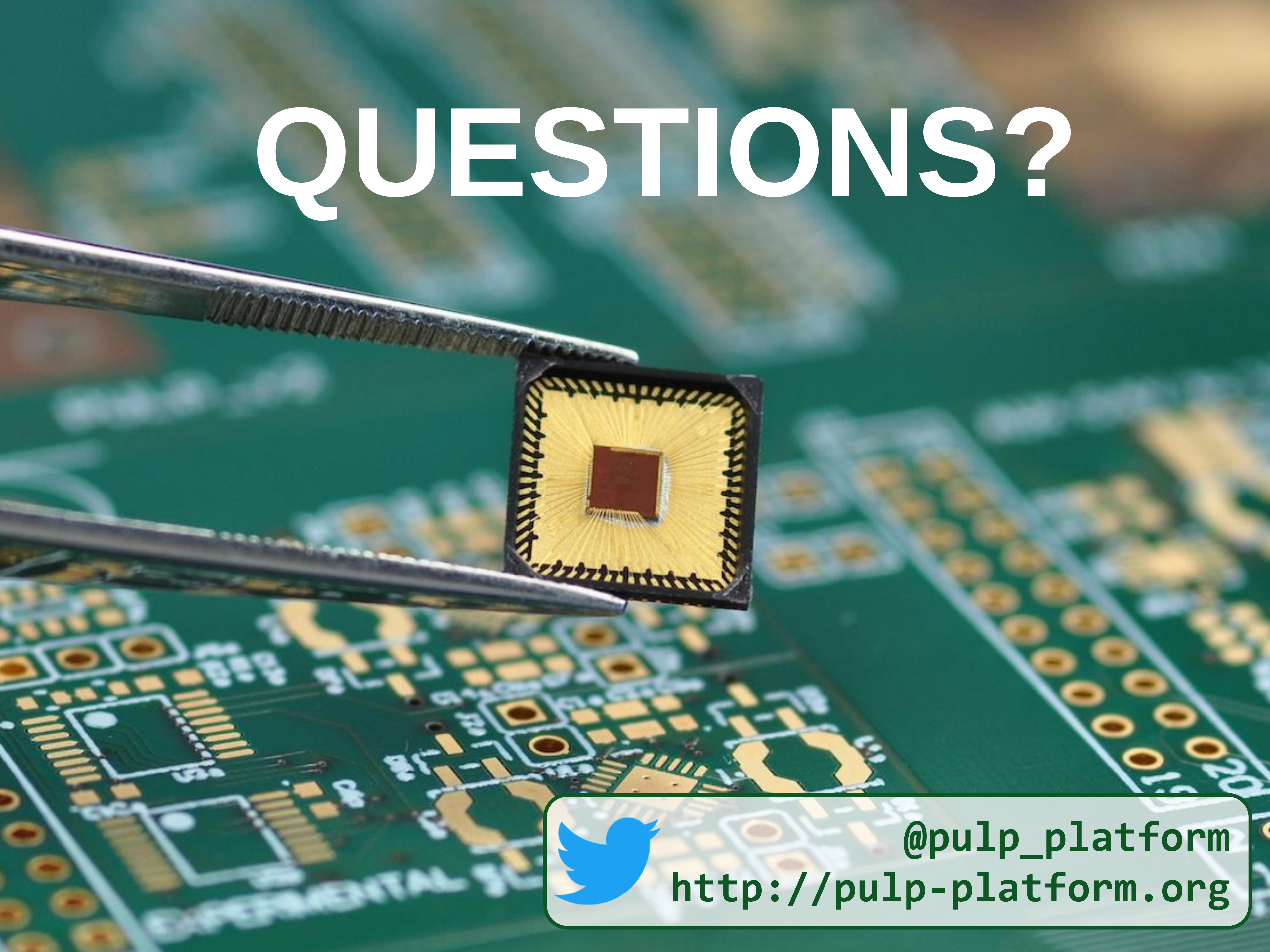
- TU Darmstadt
- Universität Bremen
- Hongik University Seoul
- IIT Kharagpur

- LIRMM Montpelier

# Last Words

- PULP provides high quality RISC-V based platforms
  - Permissive open source license
  - Written in SystemVerilog
  - Available from GitHub: https://github.com/pulp-platform
- Our research is developing energy-efficient systems
  - Wide range of applications from IoT to HPC
  - See our chips in our gallery: http://asic.ethz.ch
- Open to collaborations with industrial and academic partners
  - We have plenty of collaborations, as PULP allows us to quickly progress with projects

# QUESTIONS?



@pulp_platform
http://pulp-platform.org