The Parallel Ultra Low Power Platform

RISC-V Tutorial at HotChips 2019

18 Aug 2019

Fabian Schuiki
and the entire PULP team

pulp-platform.org
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)
Motivation: Cloud → Edge → Extreme Edge AI

Latency, Privacy

Cost

Extreme edge AI challenge:
- AI capabilities below 1 pJ/op (MCU power envelope)
- Mops to Tops
- Beyond fp32/fp64

3x Cost reduction if data volume is reduced by 95%

Source: © Wikibon IoT Project. Reference Models AWS IoT Service & Pivot3 Server SAN. See Table 1 for Detailed Assumptions & Calculations
Near-Threshold Computing (NTC):
1. Don’t waste energy pushing devices in strong inversion
2. Recover performance with parallel execution
3. Core with ‘naked’ L1 interface to create cluster coupled at L1 level
4. Manage Leakage, PVT variability and SRAM limiting NT!

Need Strong ISA, Need full access to “deep” core interfaces, need to tune pipeline!
OPEN ISA: **RISC-V RV32IMC + New, Open Microarchitecture → RI5CY!**
Bespoke ISA needed! Enter Xpulp extensions

<32-bit precision \(\rightarrow\) **SIMD2/4** \(\rightarrow\) x2,4 efficiency & memory size

Risc-V ISA is extensible *by construction* (great!)

<table>
<thead>
<tr>
<th>V1</th>
<th>Baseline RISC-V RV32IMC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HW loops</td>
</tr>
<tr>
<td>V2</td>
<td>Post modified Load/Store Mac</td>
</tr>
<tr>
<td>V3</td>
<td>SIMD 2/4 + DotProduct + Shuffling</td>
</tr>
<tr>
<td></td>
<td>Bit manipulation unit</td>
</tr>
<tr>
<td></td>
<td>Lightweight fixed point <em>(EML centric)</em></td>
</tr>
</tbody>
</table>

25 kGE \(\rightarrow\) 40 kGE (1.6x)

---

RI5CY – are Xpulp ISA Extensions (1.6x) worthwhile?

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

**Baseline**

```asm
mv   x5, 0
mv   x4, 100
Lstart:
    lb   x2, 0(x10)
    lb   x3, 0(x11)
    addi  x10, x10, 1
    addi  x11, x11, 1
    add   x2, x3, x2
    sb   x2, 0(x12)
    addi  x4, x4, -1
    addi  x12, x12, 1
    bne  x4, x5, Lstart
```

**Auto-incr load/store**

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

**HW Loop**

```asm
lp.setupi 100, Lend
    lb   x2, 0(x10!)
    lb   x3, 0(x11!)
    addi  x4, x4, 1
    add   x2, x3, x2
    Lend:  sb   x2, 0(x12!)
```

**Packed-SIMD**

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

11 cycles/output 8 cycles/output 5 cycles/output 1,25 cycles/output

10x on 2d convolutions ...YES!
Results: RV32IMC\textsuperscript{X}pulp vs RV32IMC

8-bit Convolution Results

Overall Speedup of 75x

10x Speedup w.r.t. RV32IMC (ISA does matter)

PULP-NN: an open Source library for DNN inference on PULP cores
The Evolution of the ‘Species’

<table>
<thead>
<tr>
<th></th>
<th>PULPv1</th>
<th>PULPv2</th>
<th>PULPv3</th>
</tr>
</thead>
<tbody>
<tr>
<td># of cores</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>L2 memory</td>
<td>16 kB</td>
<td>64 kB</td>
<td>128 kB</td>
</tr>
<tr>
<td>Technology</td>
<td>FD-SOI 28nm</td>
<td>FD-SOI 28nm</td>
<td>FD-SOI 28nm</td>
</tr>
<tr>
<td></td>
<td>conventional-well</td>
<td>flip-well</td>
<td>conventional-well</td>
</tr>
<tr>
<td>Voltage range</td>
<td>0.45V - 1.2V</td>
<td>0.3V - 1.2V</td>
<td>0.5V - 0.7V</td>
</tr>
<tr>
<td>BB range</td>
<td>-1.8V - 0.9V</td>
<td>0.0V - 1.8V</td>
<td>-1.8V - 0.9V</td>
</tr>
<tr>
<td>Max freq.</td>
<td>475 MHz</td>
<td>1 GHz</td>
<td>200 MHz</td>
</tr>
<tr>
<td>Max perf.</td>
<td>1.9 GOPS</td>
<td>4 GOPS</td>
<td>1.8 GOPS</td>
</tr>
<tr>
<td>Peak en. eff.</td>
<td>60 GOPS/W</td>
<td>135 GOPS/W</td>
<td>385 GOPS/W</td>
</tr>
<tr>
<td>Status</td>
<td>post tape out</td>
<td>Silicon proven</td>
<td>silicon proven</td>
</tr>
<tr>
<td>Technology</td>
<td>FD-SOI 28nm</td>
<td>28nm</td>
<td>28nm</td>
</tr>
<tr>
<td></td>
<td>conventional-well</td>
<td>flip-well</td>
<td>conventional-well</td>
</tr>
<tr>
<td>Voltage range</td>
<td>0.5V - 1.2V</td>
<td>0.3V - 1.2V</td>
<td>0.5V - 0.7V</td>
</tr>
<tr>
<td>BB range</td>
<td>-1.8V - 0.9V</td>
<td>0.0V - 1.8V</td>
<td>-1.8V - 0.9V</td>
</tr>
<tr>
<td>Max freq.</td>
<td>200 MHz</td>
<td>1 GHz</td>
<td>200 MHz</td>
</tr>
<tr>
<td>Max perf.</td>
<td>1.9 GOPS</td>
<td>4 GOPS</td>
<td>1.8 GOPS</td>
</tr>
<tr>
<td>Peak en. eff.</td>
<td>60 GOPS/W</td>
<td>135 GOPS/W</td>
<td>385 GOPS/W</td>
</tr>
</tbody>
</table>

**2.6 pJ/op**
**Mr. Wolf Chip Results: Heterogeneous Computing Works**

<table>
<thead>
<tr>
<th>Technology</th>
<th>CMOS 40nm LP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chip area</td>
<td>10 mm²</td>
</tr>
<tr>
<td>VDD range</td>
<td>0.8V - 1.1V</td>
</tr>
<tr>
<td>Memory Transistors</td>
<td>576 Kbytes</td>
</tr>
<tr>
<td>Logic Transistors</td>
<td>1.8 Mgates</td>
</tr>
<tr>
<td>Frequency Range</td>
<td>32 kHz – 450 MHz</td>
</tr>
<tr>
<td>Power Range</td>
<td>72 µW – 153 mW</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Power Manager (DC/DC + LDO)</th>
<th>VDD [V]</th>
<th>Freq.</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Deep Sleep</strong></td>
<td>0.8</td>
<td>n.a.</td>
<td>72 µW</td>
</tr>
<tr>
<td><strong>Ret. Deep Sleep</strong></td>
<td>0.8</td>
<td>n.a.</td>
<td>76.5 - 108 µW</td>
</tr>
<tr>
<td><strong>SoC Active</strong></td>
<td>0.8 - 1.1</td>
<td>32 kHz - 450 MHz</td>
<td>0.97 - 38 mW</td>
</tr>
<tr>
<td><strong>Cluster Active</strong></td>
<td>0.8 - 1.1</td>
<td>32 kHz - 350 MHz</td>
<td>1.6 - 153 mW</td>
</tr>
</tbody>
</table>

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

**Packed-SIMD** support for all formats

<table>
<thead>
<tr>
<th>FP64</th>
<th>FP32</th>
<th>FP32</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP16</td>
<td>FP16</td>
<td>FP16</td>
</tr>
<tr>
<td>FP16</td>
<td>FP16</td>
<td>FP16</td>
</tr>
<tr>
<td>FP8</td>
<td>FP8</td>
<td>FP8</td>
</tr>
<tr>
<td>FP8</td>
<td>FP8</td>
<td>FP8</td>
</tr>
<tr>
<td>FP8</td>
<td>FP8</td>
<td>FP8</td>
</tr>
<tr>
<td>FP8</td>
<td>FP8</td>
<td>FP8</td>
</tr>
</tbody>
</table>

**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)
PULP cluster+MCU+HWCE(V1) → GWT’s GAP8 (55 TSMC)

Two independent clock and voltage domains, from 0-133MHz/1V up to 0-250MHz/1.2V

<table>
<thead>
<tr>
<th>What</th>
<th>Freq MHz</th>
<th>Exec Time ms</th>
<th>Cycles</th>
<th>Power mW</th>
</tr>
</thead>
<tbody>
<tr>
<td>40nm Dual Issue MCU</td>
<td>216</td>
<td>99.1</td>
<td>21 400 000</td>
<td>60</td>
</tr>
<tr>
<td>GAP8 @1.0V</td>
<td>15.4</td>
<td>99.1</td>
<td>1 500 000</td>
<td>3.7</td>
</tr>
<tr>
<td>GAP8 @1.2V</td>
<td>175</td>
<td>8.7</td>
<td>1 500 000</td>
<td>70</td>
</tr>
<tr>
<td>GAP8 @1.0V w HWCE</td>
<td>4.7</td>
<td>99.1</td>
<td>460 000</td>
<td><strong>0.8</strong></td>
</tr>
</tbody>
</table>

4x More efficiency at less than 10% area cost
New Application Frontiers: DroNET on NanoDrone

Pluggable PCB: PULP-Shield
- ~5g, 30×28mm
- GAP8 SoC
- 8 MB HDRAM
- 16 MB HFlash
- QVGA ULP HiMax camera
- Crazyflie 2.0 nano-drone (27g)

Only onboard computation for autonomous flight + obstacle avoidance
no human operator, no ad-hoc external signals, and no remote base-station!

https://youtu.be/57Vy5cSvnaA
The Cores
- 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** \(lp\text{.start}, lp\text{.end}, lp\text{.count}\)
- **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 (\(pv\text{.dotup.b}\))

\[
Z = D_1 \times K_1 + D_2 \times K_2 + D_3 \times K_3 + D_4 \times K_4
\]
Enter Zero/Micro-riscy (Ibex), small core for control

- 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
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.

Finally the step into 64-bit 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 GF 22FDX (Poseidon, Kosmodrom, Baikonur), 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
Extreme FP Performance: The “V” Extension

Ariane
1GHz
2 DP GFLOPS
8 GB/s

Instruction Queue

ARA
1GHz
8 DP GFLOPS
8 GB/s

I$, D$

64b

64b

64b

64b

Interconnect
Extreme FP Performance: The “V” Extension

Ariane
1GHz
2 DP GFLOPS
8 GB/s

Instruction
Queue

Instruction
Data

64b

64b

64b

64b

Interconnect

Instruction
Queue

Vector Register File

256b Wide Bank
256b Wide Bank
256b Wide Bank
256b Wide Bank
256b Wide Bank
256b Wide Bank
256b Wide Bank

VRF arbitration unit

64bit FP FMA
64bit FP FMA
64bit FP FMA
64bit FP FMA

Writeback

Load Store Unit

ARA

Vector Unit

I$, D$

The Platforms
# Making PULP: Cores

<table>
<thead>
<tr>
<th>RISC-V Cores</th>
<th>RI5CY</th>
<th>Ibex (MR)</th>
<th>Ibex (ZR)</th>
<th>Ariane</th>
</tr>
</thead>
<tbody>
<tr>
<td>32b</td>
<td>32b</td>
<td>32b</td>
<td>32b</td>
<td>64b</td>
</tr>
</tbody>
</table>
## Making PULP: Cores + Peripherals/Acc.

<table>
<thead>
<tr>
<th>RISC-V Cores</th>
<th>Peripherals</th>
<th>Interconnect</th>
</tr>
</thead>
<tbody>
<tr>
<td>RI5CY 32b</td>
<td>JTAG</td>
<td>Logarithmic interconnect</td>
</tr>
<tr>
<td>Ibex (MR) 32b</td>
<td>SPI</td>
<td>APB – Peripheral Bus</td>
</tr>
<tr>
<td>Ibex (ZR) 32b</td>
<td>UART</td>
<td>AXI4 – Interconnect</td>
</tr>
<tr>
<td>Ariane 64b</td>
<td>I2S</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DMA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GPIO</td>
<td></td>
</tr>
</tbody>
</table>

### Accelerators

- HWCE (convolution)
- Neurostream (ML)
- HWCrypt (crypto)
- PULPO (1st order opt)
Making PULP: Cores + Peripherals/Acc. = Platforms

RISC-V Cores
- RI5CY (32b)
- Ibex (MR) (32b)
- Ibex (ZR) (32b)
- Ariane (64b)

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

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

Platforms
- Single Core (IOT)
  - PULPino
  - PULPissimo
- Multi-core (IOT)
  - Fulmine
  - Mr. Wolf
- Multi-cluster (HPC)
  - Hero

Accelerators
- HWCE (convolution)
- Neurostream (ML)
- HWCrypt (crypto)
- PULPO (1st order opt)
The PULP platforms put everything together

<table>
<thead>
<tr>
<th>RISC-V Cores</th>
<th>Peripherals</th>
<th>Interconnect</th>
</tr>
</thead>
<tbody>
<tr>
<td>RI5CY 32b</td>
<td>JTAG</td>
<td>Logarithmic interconnect</td>
</tr>
<tr>
<td>Ibex (MR) 32b</td>
<td>SPI</td>
<td>APB – Peripheral Bus</td>
</tr>
<tr>
<td>Ibex (ZR) 32b</td>
<td>UART</td>
<td>AXI4 – Interconnect</td>
</tr>
<tr>
<td>Ariane 64b</td>
<td>I2S</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DMA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GPIO</td>
<td></td>
</tr>
</tbody>
</table>

**Platforms**

<table>
<thead>
<tr>
<th>I</th>
<th>M</th>
<th>O</th>
<th>R5</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Single Core**

- PULPino
- PULPissimo

**Accelerators**

- HWCE (convolution)
- Neurostream (ML)
- HWCrypt (crypto)
- PULPO (1st order opt)
- 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 (Ibex)
- Peripherals copied from its larger brothers
  - Any AXI and APB peripherals could be used
- 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 SoC/fabric controller in larger systems
The main PULP systems we develop are cluster based

**RISC-V Cores**
- RI5CY 32b
- Ibex (MR) 32b
- Ibex (ZR) 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)
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

Tightly Coupled Data Memory

- Mem
- Mem
- Mem
- Mem
- Mem
- Mem

L2 Mem

DMA

interconnect

CLUSTER

HW ACCEL

RISC-V core

RISC-V core

RISC-V core

RISC-V core

I$
Event unit to manage resources (fast sleep/wake up)
An additional microcontroller system (PULPissimo) for I/O
Finally multi-cluster PULP systems for HPC applications

### RISC-V Cores
- **RI5CY** 32b
- **Ibex (MR)** 32b
- **Ibex (ZR)** 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)**
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

- **Still work in progress**
  - Bare-metal released in Dec ’18
  - Update with support for SMP Linux will be released soon
OpenPiton+Ariane mapped to FPGA

**Digilent Genesys2**
- **Core**: 66 MHz
- Up to 2 cores
- 8 GiB DDR3
- 1 core config:
  - 85k LUT (42%)
  - 67 BRAM (15%)

**Xilinx VCU 118**
- **Core**: 100 MHz
- Up to 16 cores
- 32 GiB DDR4
- (Available soon)
The Chips
We have designed more than 25 ASICs based on PULP

ASICs meant to go on IC Tester
- Mainly characterization
- Not so many peripherals

ASICs meant for applications
- More peripherals (SPI, Camera)
- More on-chip memory
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 8+1 RI5CY cores
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
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.
- Designed in collaboration with the Analog group of Prof. Huang at ETH
- All chips with SMIC130 (because of analog IPs)
- First three with OpenRISC, VivoSoC3 with RISC-V
The new generation chips from 2018

- 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 (2x clusters with 4x RI5CY cores each)
- **Scarabaeus**: Ariane based microcontroller
The large system chips from 2018

- All are Globalfoundries 22FDX, around 10 mm\(^2\), 50-100 Mtrans
- **Poseidon**: PULPissimo (RI5CY) + Ariane
- **Kosmodrom**: 2x Ariane + NTX (FP streaming) accelerator
- **Arnold**: PULPissimo (RI5CY) + Quicklogic eFPGA
The next frontier from 2019

- UMC 65nm and Globalfoundries 22FDX
- **Billywig**: Streaming-enhanced RV32 cores for max. throughput, 3mm²
- **Urania**: Ariane+PULP Het. SoC, plus custom DRAM controller, 16mm²
- **Baikonur**: 2x Ariane + streaming-enhanced RV32 cores, 10mm²
We firmly believe in Open Source movement

First launched in February 2016 (Github)

All our development is on open repositories

Contributions from many groups
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 small enterprises

- Hardware is 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.
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

<table>
<thead>
<tr>
<th>University/Institution</th>
<th>Collaborator Logo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Politecnico di Torino</td>
<td><img src="https://example.com/politecnico-di-torino" alt="Logo" /></td>
</tr>
<tr>
<td>University of Cambridge</td>
<td><img src="https://example.com/university-of-cambridge" alt="Logo" /></td>
</tr>
<tr>
<td>ETH Zurich</td>
<td><img src="https://example.com/eth-zurich" alt="Logo" /></td>
</tr>
<tr>
<td>Technische Universität Graz</td>
<td><img src="https://example.com/technische-universitaet-graz" alt="Logo" /></td>
</tr>
<tr>
<td>Fraunhofer Gesellschaft</td>
<td><img src="https://example.com/fraunhofer-gesellschaft" alt="Logo" /></td>
</tr>
<tr>
<td>Sapere Aude University</td>
<td><img src="https://example.com/sapere-aude-university" alt="Logo" /></td>
</tr>
</tbody>
</table>

### Academic users we are aware of

<table>
<thead>
<tr>
<th>University/Institution</th>
<th>Collaborator Logo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Universiti di Genova</td>
<td><img src="https://example.com/universiti-di-genova" alt="Logo" /></td>
</tr>
<tr>
<td>Politecnico di Milano</td>
<td><img src="https://example.com/politecnico-di-milano" alt="Logo" /></td>
</tr>
<tr>
<td>KAIST</td>
<td><img src="https://example.com/kaist" alt="Logo" /></td>
</tr>
<tr>
<td>Leibniz University</td>
<td><img src="https://example.com/leibniz-university" alt="Logo" /></td>
</tr>
<tr>
<td>Silicon Labs</td>
<td><img src="https://example.com/silicon-labs" alt="Logo" /></td>
</tr>
<tr>
<td>AXILINK</td>
<td><img src="https://example.com/axilink" alt="Logo" /></td>
</tr>
<tr>
<td>TUM IT</td>
<td><img src="https://example.com/tum-it" alt="Logo" /></td>
</tr>
<tr>
<td>TIB Kaisersburg University</td>
<td><img src="https://example.com/tib-kaisersburg" alt="Logo" /></td>
</tr>
<tr>
<td>Chalmers University</td>
<td><img src="https://example.com/chalmers-university" alt="Logo" /></td>
</tr>
<tr>
<td>NTI Techonologies</td>
<td><img src="https://example.com/nti-techonologies" alt="Logo" /></td>
</tr>
</tbody>
</table>
Micro/Zero-riscy is now **Ibex**

- LowRISC has agreed to maintain micro/zero riscy
  - Interested in using the core in their projects
  - They have a team that can provide support
  - ETH Zürich and University of Bologna will continue to contribute to Ibex

- Our core has grown and left the house
  - Alpine Ibex (Capra Ibex) is a mountain goat that is typical in the mountains of Switzerland
Non-Profit Open Hardware Group
OpenHW Group Charter

OpenHW Group is a not-for-profit, global organization driven by its members and individual contributors where hardware and software designers collaborate in the development of open-source cores, related IP, tools and software such as the CORE-V Family of cores. OpenHW provides an infrastructure for hosting high quality open-source HW developments in line with industry best practices.

R. O’Connor (OpenHW CEO, former RISC-V foundation director)
Thanks!

@pulp_platform  pulp-platform.org  asic.ethz.ch