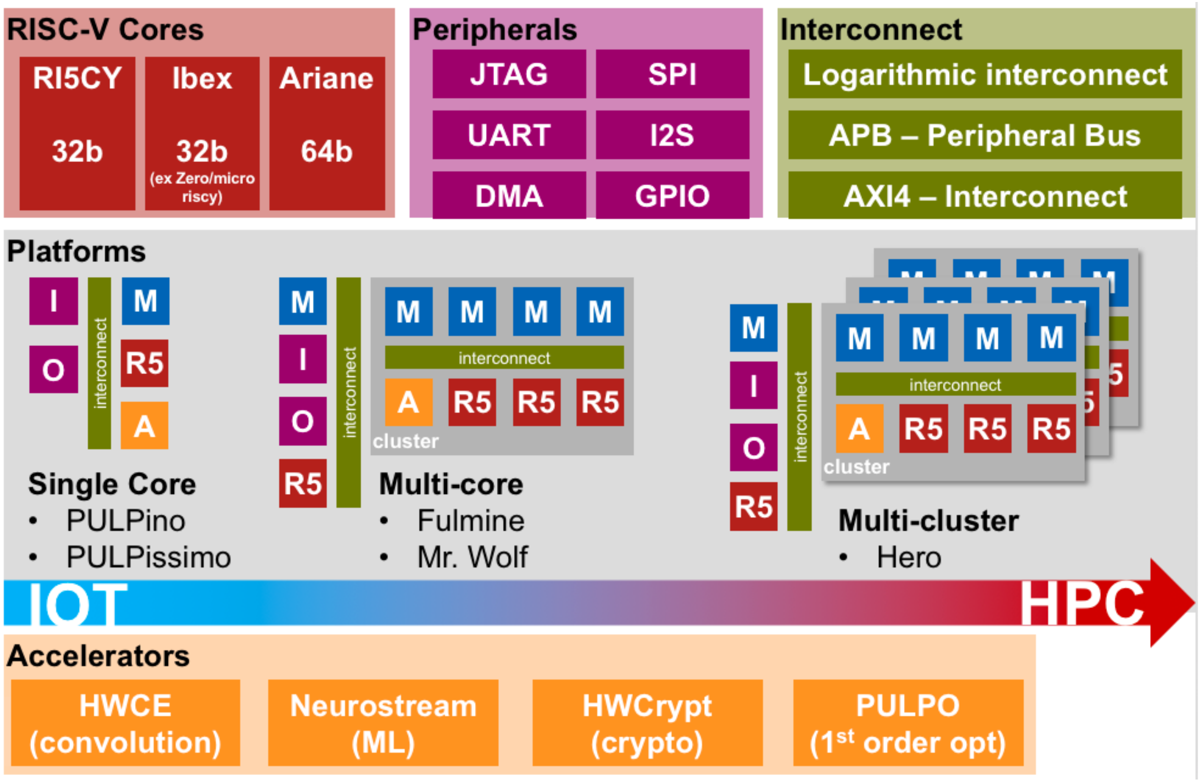
RISC-V cores in IoT Applications

PULP platform

The Parallel Ultra Low Power (PULP) Platform started as a joint effort between the Integrated Systems Laboratory (IIS) of ETH Zürich and the Energy-Efficient Embedded Systems (EEES) group of the University of Bologna to explore new and efficient architectures for ultra-low-power processing.

PULP is a silicon-proven Parallel Ultra Low Power platform targeting high energy efficiencies. The platform is organized in clusters of RISC-V cores that share a tightly-coupled data memory.

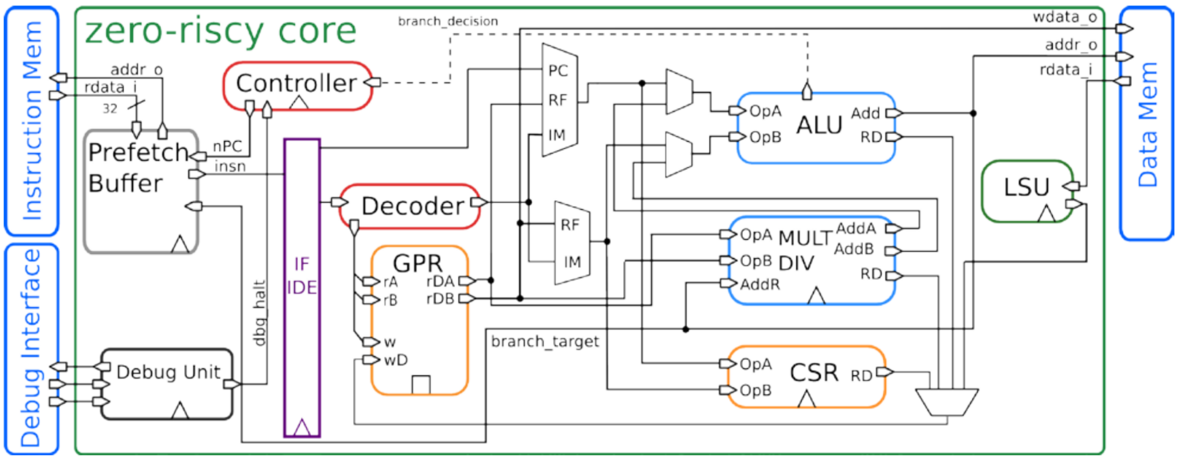
PULP is developing an open, scalable hardware and software research and development platform to break the energy efficiency barrier within a power envelope of a few milliwatts, as well as satisfy the computational demands of IoT applications requiring flexible processing of data streams generated by multiple sensors, such as accelerometers, low-resolution cameras, microphone arrays, and vital signs monitors.



Processor cores

1. Ibex (formerly named Zero-riscy)

ZERO-RISCY is a 2-stage in-order 32b RISC-V processor core. ZERO-RISCY has been designed to be small and efficient. Via two parameters, the core is configurable to support four ISA configurations.



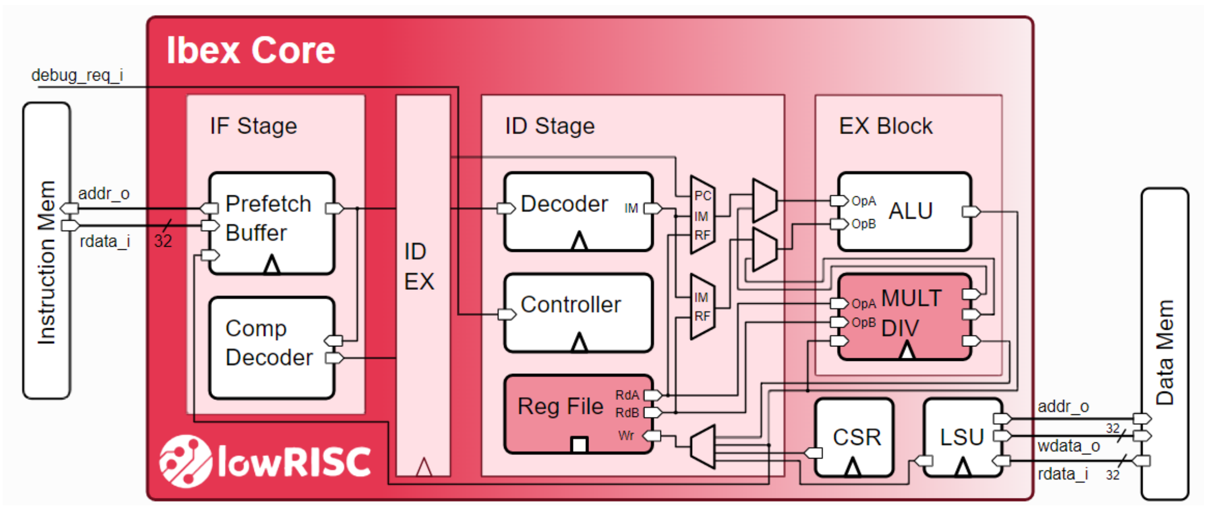
Block Diagram

ZERO-RISCY supports the following instructions:

- Full support for RV32I Base Integer Instruction Set
- Full support for RV32E Base Integer Instruction Set
- Full support for RV32C Standard Extension for Compressed Instructions
- Full support for RV32M Integer Multiplication and Division Instruction Set Extension

The RV32M and RV32E can be enable and disable using two parameters.

Pipeline in ZERO-RISCY:



ZERO-RISCY pipeline

Ibex has a 2-stage pipeline, the 2 stages are:

Instruction Fetch (IF)

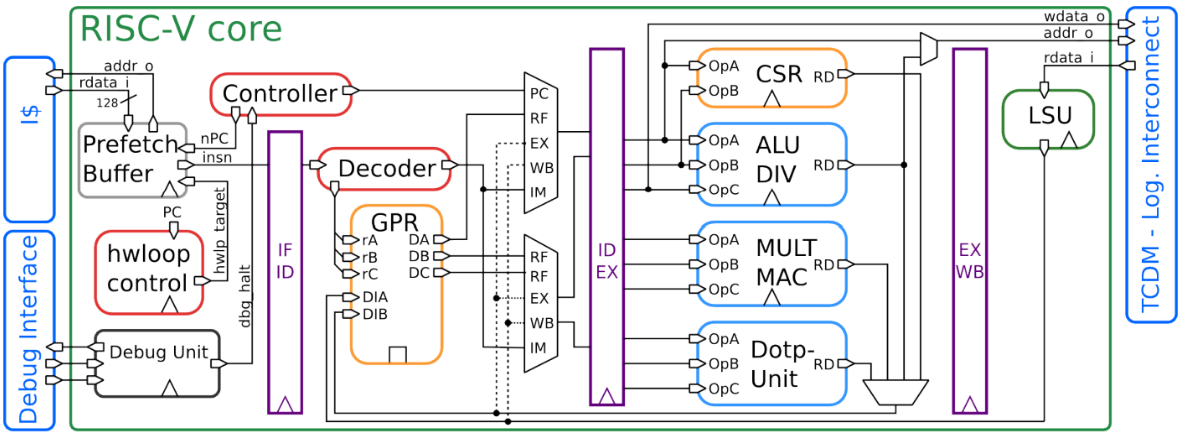
Fetches instructions from memory via a prefetch buffer, capable of fetching 1 instruction per cycle if the instruction side memory system allows.

Instruction Decode and Execute (ID/EX)

Decodes fetched instruction and immediately executes it, register read and write all occurs in this stage. Multi-cycle instructions will stall this stage until they are complete.

2. CV32E40P (formerly named RI5CY)

RI5CY is a 4-stage in-order 32b RISC-V processor core. The ISA of RI5CY was extended to support multiple additional instructions including hardware loops, post-increment load and store instructions and additional ALU instructions that are not part of the standard RISC-V ISA.



Block Diagram

RI5CY supports the following instructions:

- Full support for RV32I Base Integer Instruction Set
- Full support for RV32C Standard Extension for Compressed Instructions
- Full support for RV32M Integer Multiplication and Division Instruction Set Extension
- Optional full support for RV32F Single Precision Floating Point Extensions

  It is possible to extend the core with a private FPU, which is capable of performing all RISC-V floating-point operations that are defined in the RV32F ISA extensions.

  The FPU is divided into three parts:

  1. A simple FPU of ~10kGE complexity, which computes FP-ADD, FP-SUB and FP-casts.
  2. An iterative FP-DIV/SQRT unit of ~7 kGE complexity, which computes FP-DIV/SQRT operations.
  3. An FP-FMA unit which takes care of all fused operations. This unit is currently only supported through a Synopsys Design Ware instantiation, or a Xilinx block for FPGA targets.

- PULP specific extensions
  - Post-Incrementing load and stores

    Post-incrementing load and store instructions perform a load/store operation from/to the data memory while at the same time increasing the base address by the specified offset. For the memory access, the base address without offset is used.

    Post-incrementing load and stores reduce the number of required instructions to execute code with regular data access patterns, which can typically be found in loops. These post-incrementing load/store instructions allow the address increment to be embedded in the memory access instructions and get rid of separate instructions to handle pointers. Coupled with hardware loop extension, these instructions allow to reduce the loop overhead significantly.

  - Multiply-Accumulate extensions

RI5CY uses a single-cycle 32-bit x 32-bit multiplier with a 32-bit result. All instructions of the RISC-V M instruction set extension are supported.

The multiplications with upper-word result (MSP of 32-bit x 32-bit multiplication), take 4 cycles to compute. The division and remainder instructions take between 2 and 32 cycles. The number of cycles depends on the operand values.

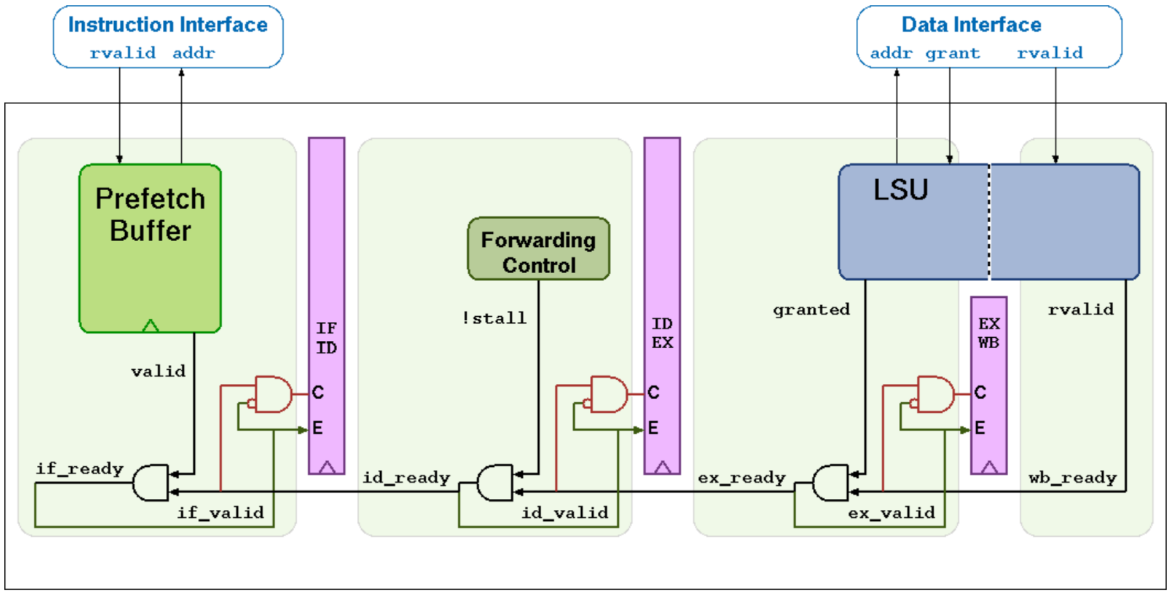
- ALU extensions

  RI5CY supports advanced ALU operations that allow to perform multiple instructions that are specified in the base instruction set in one single instruction and thus increases efficiency of the core.

- Hardware Loops

  A hardware loop is defined by its start address (pointing to the first instruction in the loop), its end address (pointing to the instruction that will be executed last in the loop) and a counter that is decremented every time the loop body is executed. RI5CY contains two hardware loop register sets to support nested hardware loops, each of them can store these three values in separate flip flops which are mapped in the CSR address space.

Pipeline in RI5CY:



RI5CY pipeline

CV32E40P has a 4-stage in-order completion pipeline, the 4 stages are:

Instruction Fetch (IF)

Fetches instructions from memory via an aligning prefetch buffer, capable of fetching 1 instruction per cycle if the instruction side memory system allows. The IF stage also pre-decodes RVC instructions into RV32I base instructions.

Instruction Decode (ID)

Decodes fetched instruction and performs required register file reads. Jumps are taken from the ID stage.

Execute (EX)

Executes the instructions. The EX stage contains the ALU, Multiplier and Divider. Branches (with their condition met) are taken from the EX stage. Multi-cycle instructions will stall this stage until they are complete. The ALU, Multiplier and Divider instructions write back their result to the register file from the EX stage. The address generation part of the load-store-unit (LSU) is contained in EX as well.

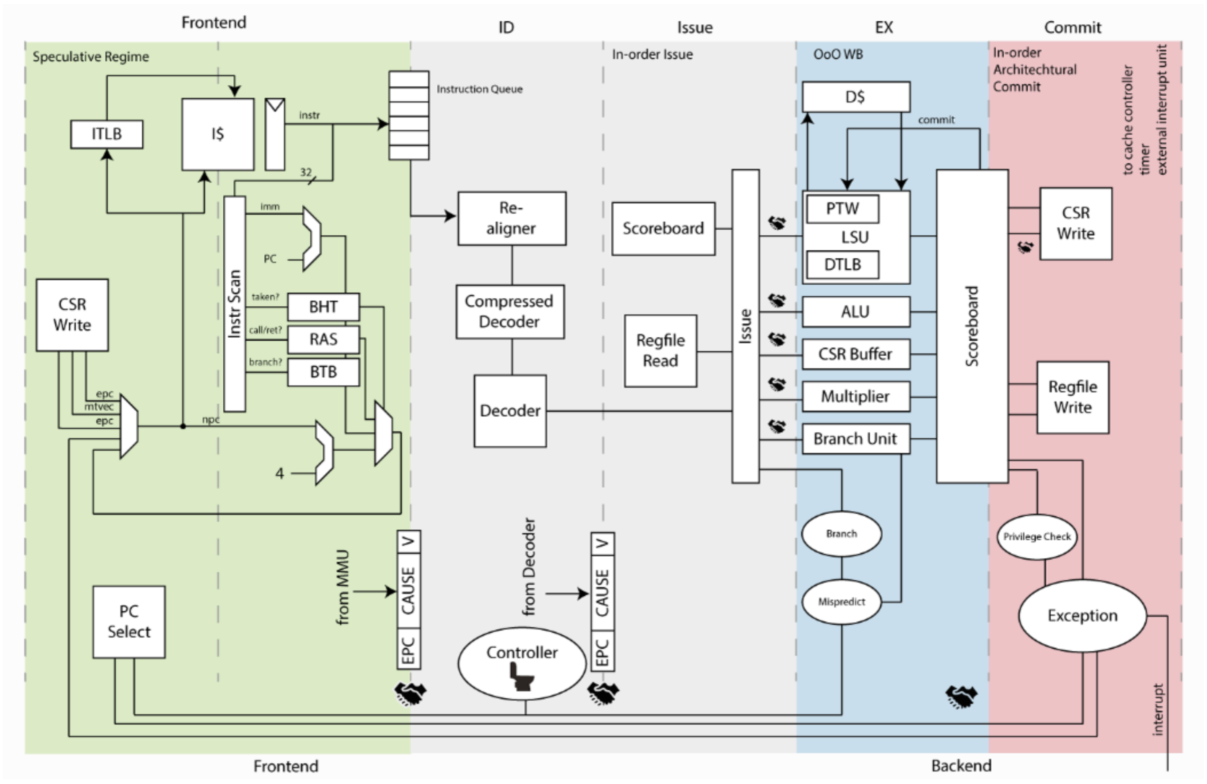
Writeback (WB)

Writes the result of Load instructions back to the register file.

3. CVA6 (formerly Ariane)

CVA6 is a 6-stage, single issue, in-order 64b RISC-V processor core. It fully implements I, M and C extensions as specified in Volume I: User-Level ISA V 2.1 as well as the draft privilege extension 1.10. It implements three privilege levels M, S, U to fully support a Unix-like operating system.

The purpose of the core is to run a full OS at reasonable speed and IPC. To achieve the necessary speed the core features a 6-stage pipelined design. In order to increase the IPC the CPU features a scoreboard which should hide latency to the data RAM (cache) by issuing data-independent instructions. The instruction RAM has (or L1 instruction cache) an access latency of 1 cycle on a hit, while accesses to the data RAM (or L1 data cache) have a longer latency of 3 cycles on a hit.



It has configurable size, separate TLBs, a hardware PTW and branch-prediction (branch target buffer, branch history table and a return address stack).

CVA6 has a 6-stage in-order completion pipeline, the 6 stages are:

PC Generation

PC gen is responsible for generating the next program counter. All program counters are logical addressed. This stage contains speculation on the branch target address as well as the information if the branch is taken or not. In addition, it houses the branch target buffer (BTB) and a branch history table (BHT).

Instruction Fetch

Instruction Fetch stage (IF) gets its information from the PC Gen stage. This information includes information about branch prediction, the current PC and whether this request is valid.

Instruction Decode

Instruction decode is the fist pipeline stage of the processor’s back-end. Its main purpose is to distill instructions from the data stream it gets from IF stage, decode them and send them to the issue stage.

Issue

The issue stage’s purpose is to receive the decoded instructions and issue them to the various functional units. The issue stage keeps track of all issued instructions, the functional unit status and receives the write-back data from the execute stage. Furthermore, it contains the CPU’s register file.

Execute

The execute stage is a logical stage which encapsulates all the functional units (FUs). Each functional unit maintains a valid signal with which it will signal valid output data and a ready signal which tells the issue logic whether it is able to accept a new request or not.

Commit

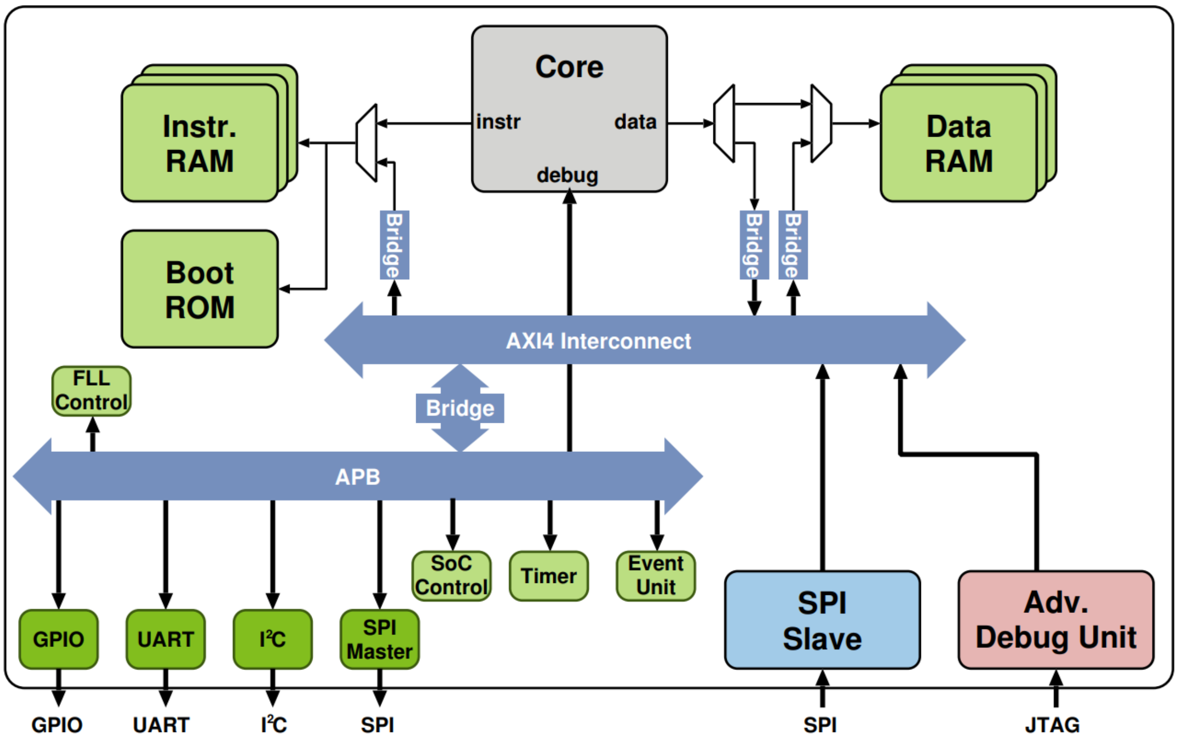
The commit stage is the last stage in the processor’s pipeline. Its purpose is to take incoming instruction and update the architectural state. This includes writing CSR registers, committing stores and writing back data to the register file.

SOC platforms

1. Single-core micro-controllers

a. PULPino

PULPino is a single-core System-on-a-Chip built for the RISC-V RI5CY and zero-riscy core. It uses separate single-port data and instruction RAMs. It includes a boot ROM that contains a boot loader that can load a program via SPI from an external flash device.



The SoC uses a AXI as its main interconnect with a bridge to APB for simple peripherals. Both the AXI and the APB buses feature 32 bit wide data channels. For debugging purposes the SoC includes an advanced debug unit which enables access to core registers, the two RAMs and memory-mapped IO via JTAG. Both RAMs are connected to the AXI bus via bus adapters.

PULPino feature:

- CPU Core

  PULPino supports both the RISC-V RI5CY and the RISC-V zero-riscy. The two cores have the same external interfaces and are thus plug-compatible.

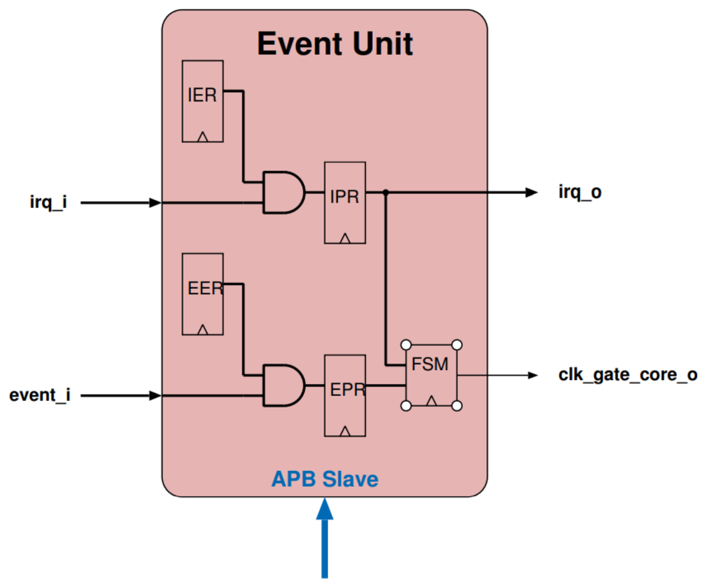
  The core use a very simple data and instruction interface to talk to data and instruction memories. To interface with AXI, a core2axi protocol converter is instantiated in PULPino.

- Advanced Debug Unit

  The advanced debug unit has an AXI master interface to access peripherals and memories.

  All core registers are now memory mapped which means that they can be read over the AXI interface. Hence, debugging is not only possible over JTAG, but also SPI or any other interface.

- Peripherals
  - UART
  - GPIO
  - SPI
  - I2C
  - TIMER
  - Event Unit



PULPino features a lightweight event and interrupt unit which supports vectorized interrupts of up to 32 lines and event triggering of up to 32 input lines.

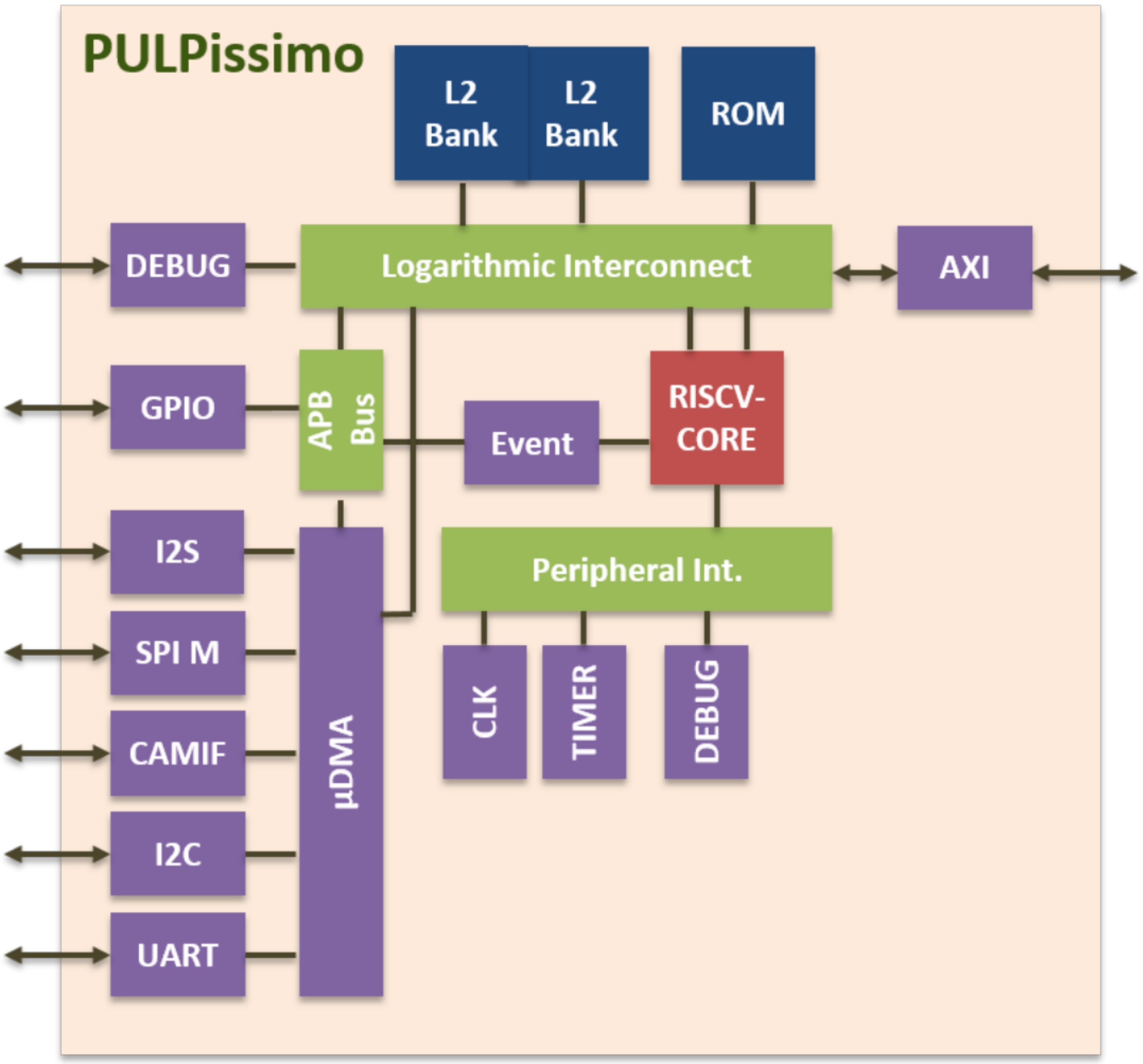
- SoC Control

  PULPino features a small and simple APB peripheral which provides information about the platform and provides the means for pad muxing on the ASIC.

b. PULPissimo

PULPissimo is a 32 bit single-core System-on-a-Chip. PULPissimo is the second version of the PULPino system and it can be extended with the multi-core cluster of the PULP project.

Differently from the simpler PULPino system, PULPissimo uses a more complex memory subsystem, an autonoumous I/O subsystem which uses the uDMA, new peripherals (eg the camera interface) and a new SDK.



The peripherals are connected to the uDMA which transfers the date to the memory subsystem efficiently. The JTAG and the AXI plug have also access to the SoC. The AXI plug can be used to extend the microcontroller with a multi-core cluster or an accelerator. As for PULPino, the advanced debug unit is used to access to system and core registers, memories and memory-mapped IO via JTAG. A logarithmic interconnect allows to link the core and the uDMA to the memory banks simultaneously.

PULPissimo features:

- CPU Core

  PULPissimo supports both the RISC-V and the zero-riscy RI5CY core. The two cores have the same external interfaces and are thus plug-compatible.

  For debugging purposes, all core registers have been memory mapped which allows to them to be accessed over the logaritmic-interconnect subsystem. The debug unit inside the core handles the request over this bus and reads/sets the core registers and/or halts the core.

- Peripherals
  - FLL

PULPissimo containts 3 FLLs. One FLL is meant for generating the clock for the peripheral domain, one for the core domain (core, memories, event unit etc) and one is meant for the cluster. The latter is not used.

- APB GPIO
- SoC Control

  PULPissimo features a small and simple APB peripheral which provides information about the platform and provides the means for pad muxing on the ASIC.

- Event/Interrupt Controller

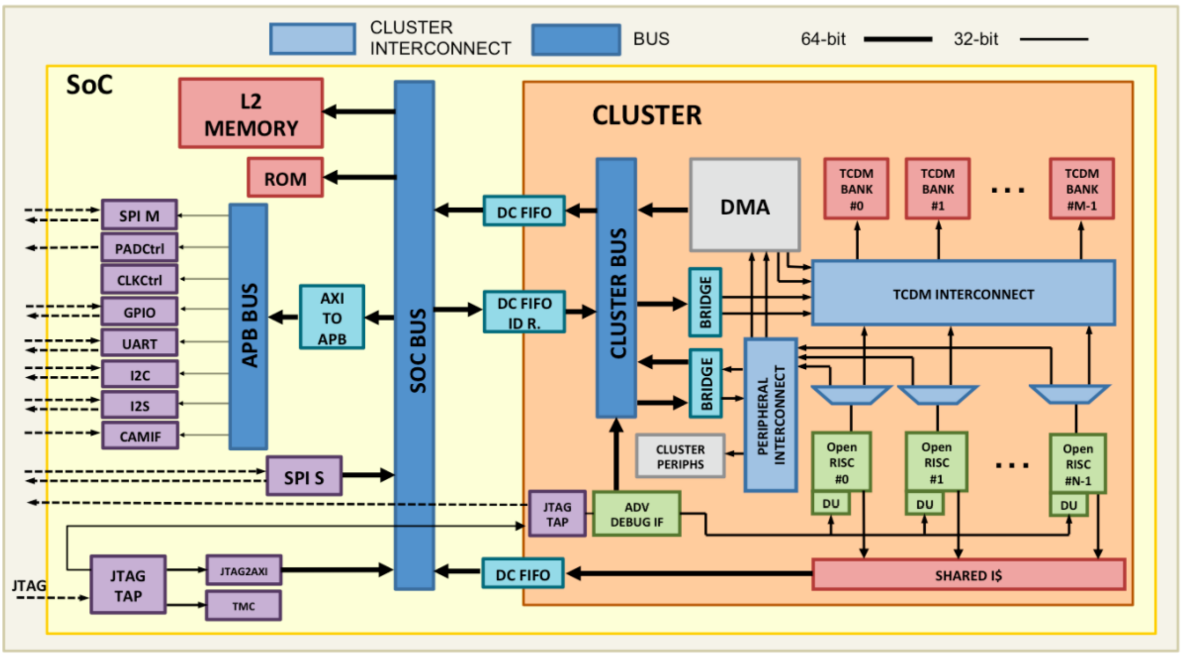
  PULPissimo features a lightweight event and interrupt controller which supports vectorized interrupts and events of up to 32 lines. It contains a FIFO of events from the peripherals or SW events.

- SoC Event Generator
- APB Timer
- APB Advanced Timer
- uDMA Subsystem

2. Multi-core IoT Processors

OpenPULP

OpenPULP is an open-source platform which itself implements an extended version of the open-source RISC-V instruction set. OpenPULP enables cost-effective development, deployment and autonomous operation of intelligent devices that capture, analyze, classify and act on the fusion of rich data sources such as images, sounds or vibrations. In particular, OpenPULP is uniquely optimized to execute a large spectrum of image and audio algorithms.



OpenPULP's hierarchical, demand-driven architecture enables ultra-low-power operation by combining:

- A fabric controller (FC) core for control, communications and security functions. This can be viewed like a classic MCU.
- A cluster of 8 cores with an architecture optimized for the execution of vectorized and parallelized algorithms.

All cores and peripherals are power switchable and voltage and frequency adjustable on demand. DC/DC regulators and clock generators with ultra-fast reconfiguration times are integrated. This allows OpenPULP to adapt extremely quickly to the processing/energy requirements of a running application. All elements share access to an L2 memory area. The cluster cores share access to an L1 (TCDM) memory area and instruction cache. Multiple DMA units allow autonomous, fast, low power transfers between cluster L1 and L2 memory and between L2 memory and external peripherals.

OpenPULP features:

- Cores

  The FC and cluster cores in PULP are based on the PULP RI5CY core.

  All 8 cores of the cluster share the RV32IMFCXpulp instruction set architecture, while the fabric controller can be configured as either RV32IMC or RV32IMFCXpulp. The I (integer), C (compressed instruction), M (Multiplication and division) and a portion of the supervisor ISA subsets are supported.
- Memory areas

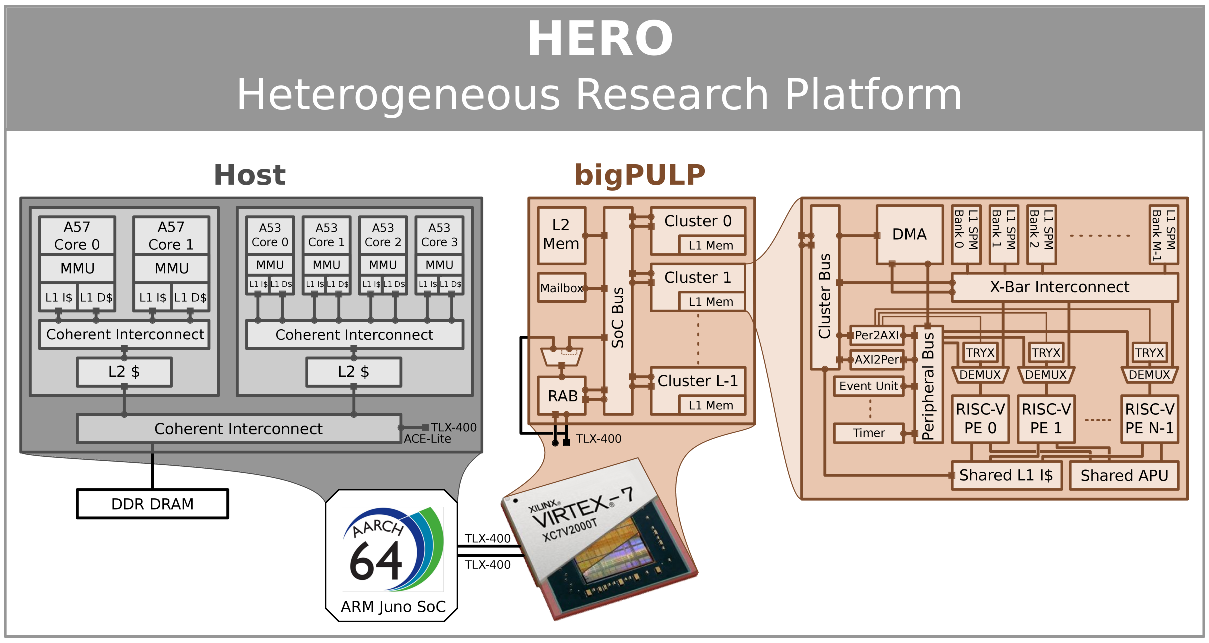
  There are 2 different levels of memory internal to PULP. A larger level 2 area of 512kB which is accessible by all processors and DMA units a smaller (Tightly Coupled Device Memory - TCDM) level 1 area shared by all the cluster cores (128kB).
- Event Units

  Two event units (EU) are available in PULP. One for the FC and one for the cluster.
- DMA
- Micro DMA
- SPI
- UART
- I2C
- I2S
- CPI
- GPIOs

3. Multi-cluster heterogeneous accelerators

HERO

HERO combines a PULP-based open-source parallel manycore accelerator implemented on FPGA with a hard ARM Cortex-A multicore host processor running full-stack Linux. HERO is the first heterogeneous system architecture that mixes a powerful ARM multicore host with a highly parallel and scalable manycore accelerator based on RISC-V cores.



HERO features:

- HERO combines:
  - a hard ARM Cortex-A multicore host processor with
  - a scalable, configurable, and extensible FPGA implementation of an open-source, silicon-proven, cluster-based manycore accelerator.
- The fully open-sourced, heterogeneous software stack of HERO supports:which allows for transparent accelerator programming and thereby tremendously simplifies the porting of existing applications to create heterogeneous implementations.
  - the OpenMP 4.5 Accelerator Model, and
  - shared virtual memory (SVM),
- The base configuration of HERO for the Xilinx Zynq ZC706 Evaluation Kit features the following accelerator configuration:
  - 1 PULP cluster comprising 8 32-bit RI5CY cores,
  - 256 KiB of shared L1 scratchpad memory,
  - 4 KiB of shared L1 instruction cache,
  - 256 KiB of shared L2 scratchpad and instruction memory,
  - our brand-new IOMMU with
    - an L1 TLB of 32 variable-sized entries, and
    - an L2 TLB of 1024 page-sized entries.