**PULP platform**

Parallel Ultra Low Power (PULP) Platform bắt đầu như một nỗ lực chung giữa Phòng thí nghiệm Hệ thống Tích hợp (IIS) của ETH Zürich và nhóm Hệ thống nhúng tiết kiệm năng lượng (EEES) của Đại học Bologna nhằm khám phá các kiến ​​trúc mới và hiệu quả cho quá trình xử lý với công suất cực thấp.

PULP là nền tảng công suất cực thấp song song nhằm mục tiêu đạt hiệu quả năng lượng cao. PULP đang phát triển một nền tảng nghiên cứu phần cứng và phần mềm để phá vỡ rào cản năng lượng trong phạm vi công suất vài miliwatt, cũng như đáp ứng nhu cầu tính toán của các ứng dụng IoT yêu cầu xử lý linh hoạt các luồng dữ liệu được tạo ra bởi nhiều cảm biến.

**Processor cores**

*Ibex (formerly named Zero-riscy)*

Ibex is a 2-stage in-order 32b RISC-V processor core. Ibex has been designed to be small and efficient.

Ibex supports the following instructions:

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

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.

*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.

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
* PULP specific extensions
  + Post-Incrementing load and stores:
  + Multiply-Accumulate extensions
  + 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

RI5CY 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.

**Writeback (WB)**

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

*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 and privilege extension. It implements three privilege levels M (machine), S (supervisor), U (user) to fully support a Unix-like operating system.

Nó có cấu hình kích thước của TLB (translation lookaside buffer) riêng biệt, và phần cứng dự đoán chuyển nhánh

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.

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 provide minimal component for a microcontroller.

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.

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
  + 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.

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:
  + the OpenMP 4.5 Accelerator Model,
  + shared virtual memory (SVM),