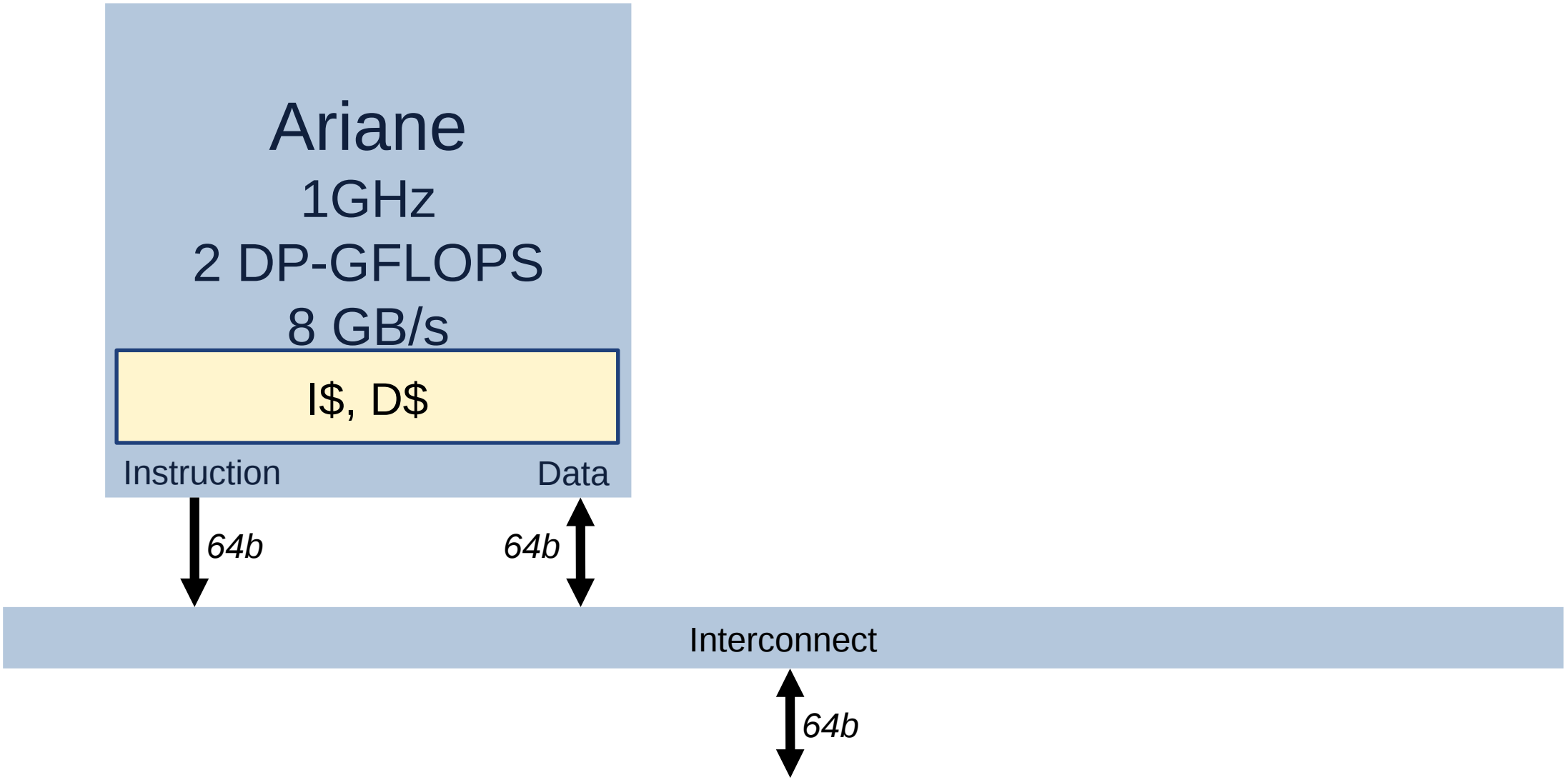
ETH zürich

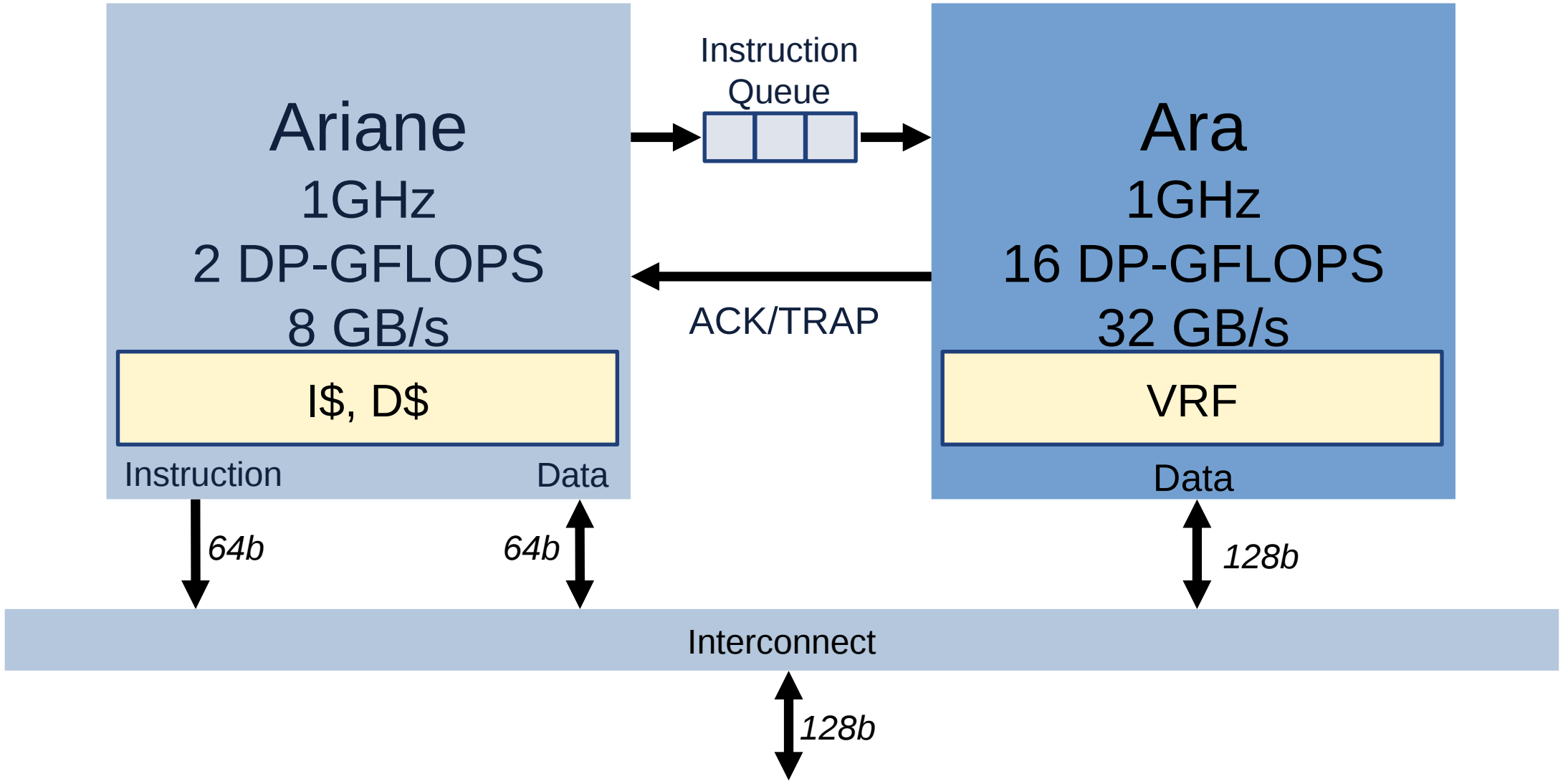
Ara: RISC-V Vector processor

Matteo Perotti
Matheus Cavalcante
PhD Students, Integrated Systems Laboratory
03 March 2021, Zurich

Ara: High-performance vector processor

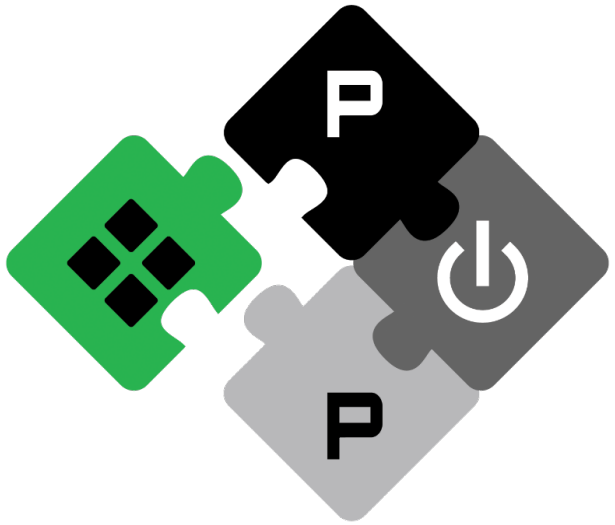


Ara: High-performance vector processor



Ara: High-performance vector processor

- **GlobalFoundries' 22FDX FD-SOI process**
  - Work initiated in my Master's thesis
  - First revealed during the 1st RISC-V Summit
  - Detailed paper available at arXiv:1906.00478



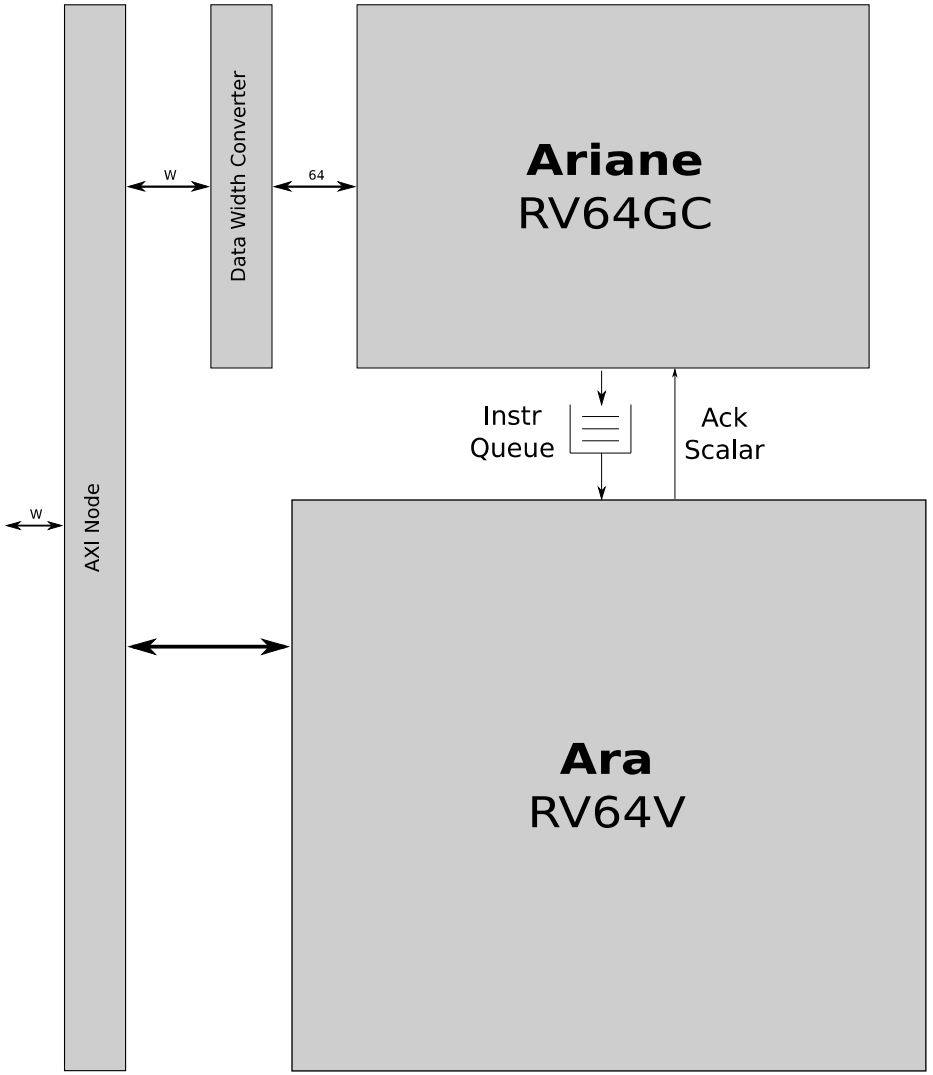
RISC-V Vector Extension

- **RVV: RISC-V "V" Extension**
  - Cray-like vector processing, opposed to packed-SIMD

- **Ara was originally based on the RVV version 0.5**
  - Ongoing Master's Thesis updating it to the version 0.9
  - Non-trivial changes regarding:
    - Encoding scheme
    - Masked instruction operation

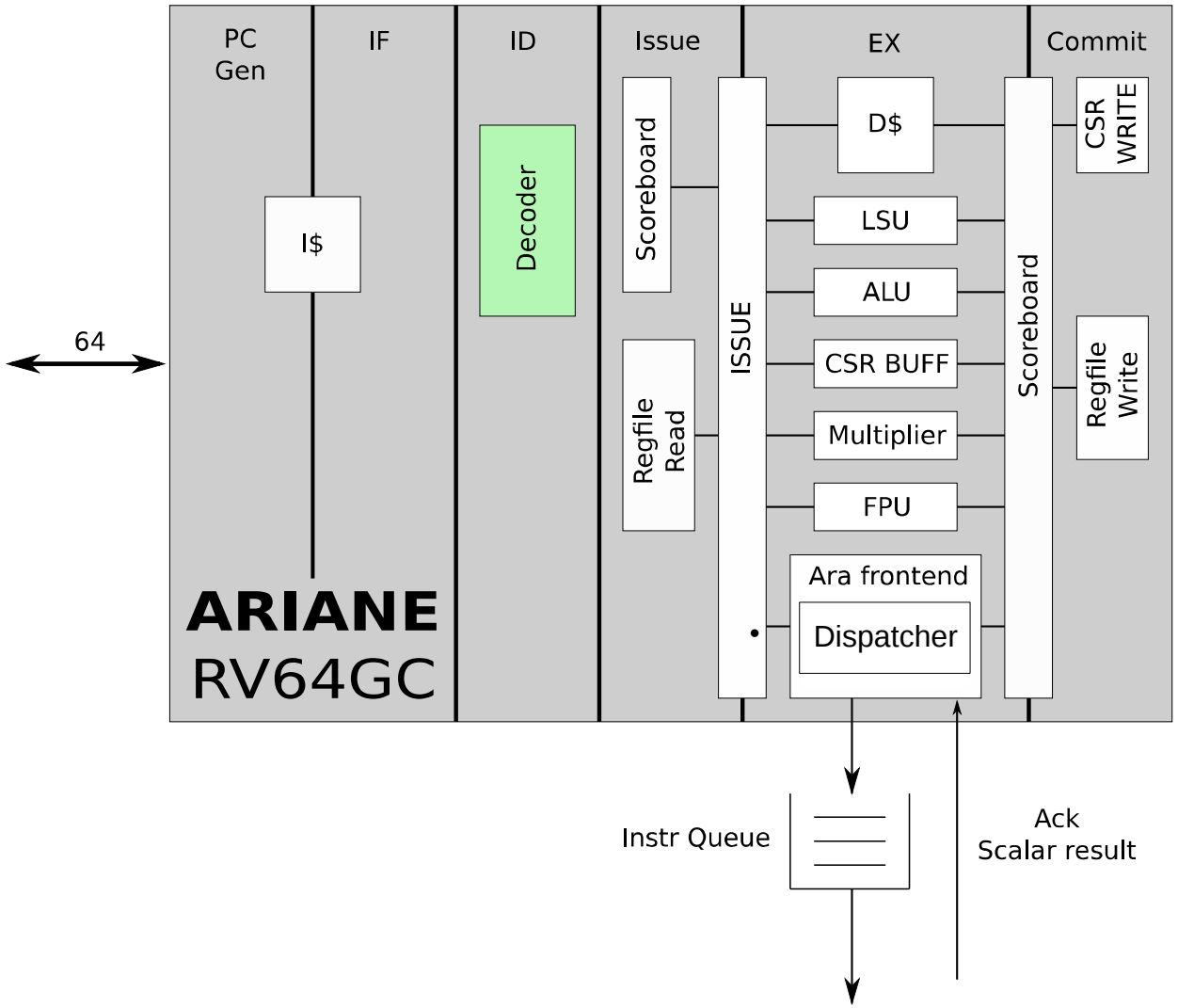
Ara - High level architecture

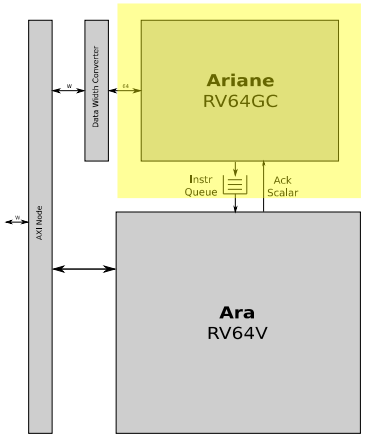
- **Coupled with Ariane**
- **Ariane dispatches RVV instructions to Ara**
- **Ara acknowledges Ariane** (Any errors? Scalar result?)
- Private memory access through AXI



Ariane – RVV instruction dispatch

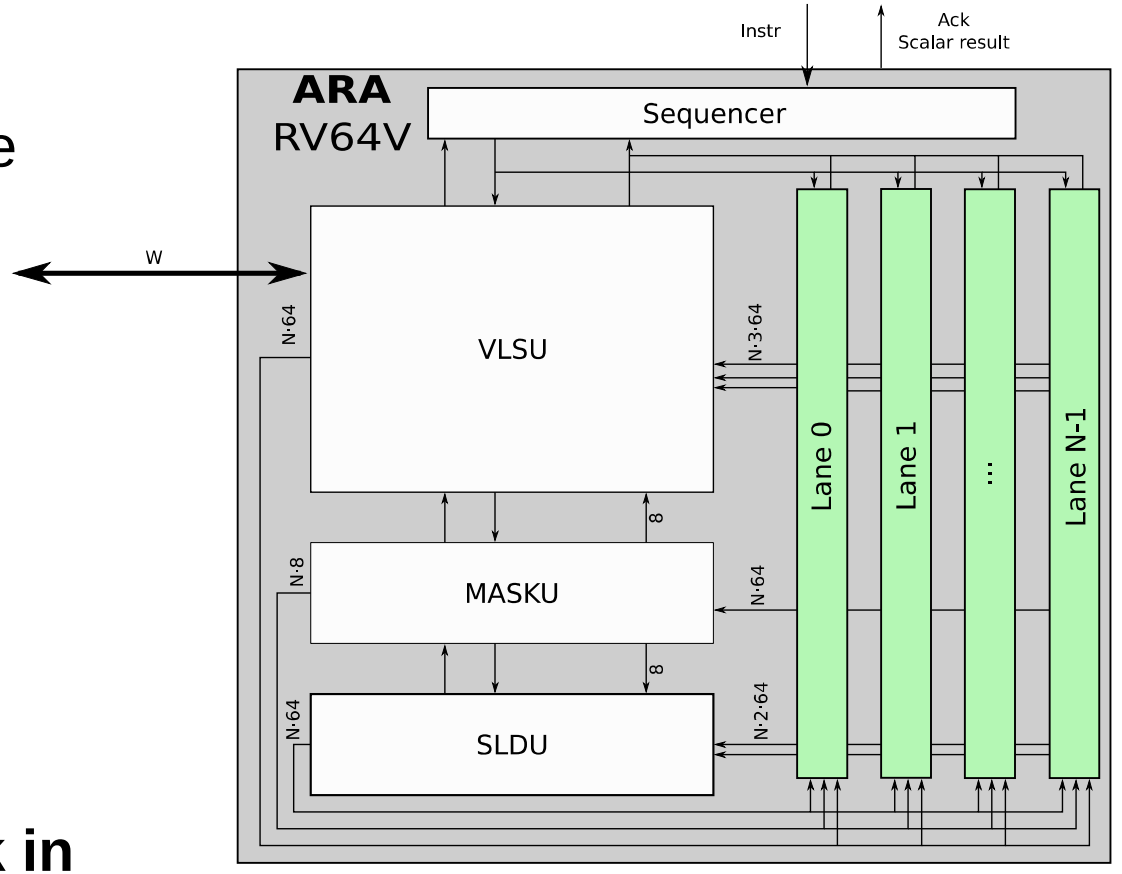
- Fetch + Decode in Ariane
- Scoreboard + **Issue to Ara frontend**
- Dispatch to Ara at **commit-time only**
- **Wait for Ara‘s answer to commit**

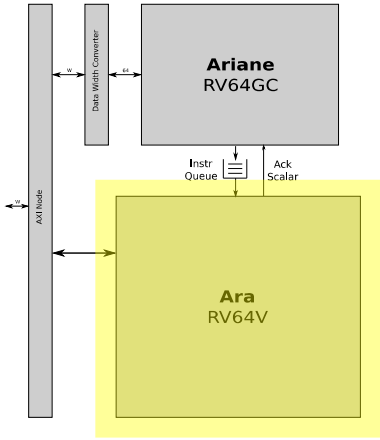




Ara – RVV instruction processing

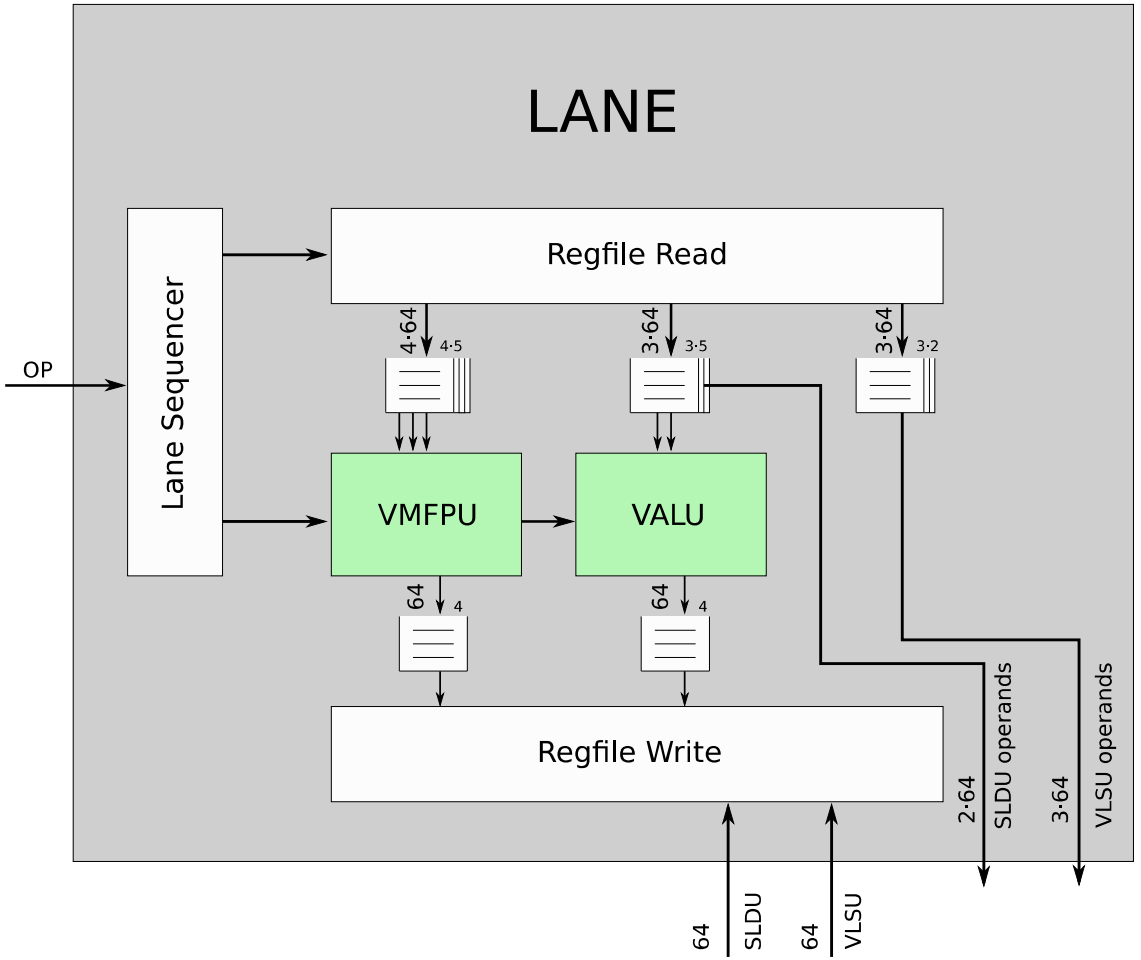
- **Sequencer**
  - Global view of Ara‘s state
- **VLSU**
  - **Memory** (AXI) ↔ **VRF** transfers
  - Address checks
- **VRF** split among the **lanes**
- **Lanes**: **VRF** + **processing**
- **MASKU and SLDU** → **work in progress**

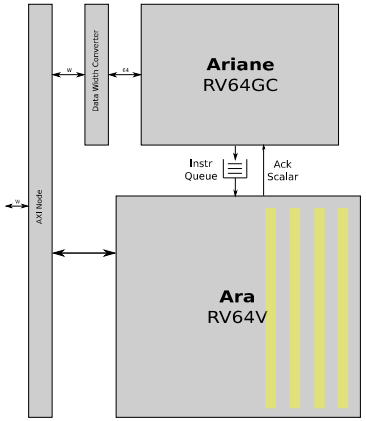




Ara Lane - RVV instruction core processing

- **Lane sequencer** tracks in-lane operations
- **8 banks per lane**
- **Multiple requests** to *VRF* → Need for an **arbiter**
- Multiple **operand queues** to feed the FUs
- **Direct write-back** from the FUs





Currently ongoing

- **Porting** to **RVV 0.9** and implementing:
  - **Mask Unit** (different mask layout from RVV 0.5 to 0.9)
  - **Slide Unit** (crucial for `conv2d`)
  - **Scatter/Gather** (exploit sparsity)
  - Remaining **floating-point** instructions
  - **Fixed-point** instructions

- Pushing **decoding** into **Ara** for increased flexibility
  - Increase decoupling between the core and Ara
  - More RV extensions-agnostic Ariane

- PPA results after P&R

SW environment – Current status

- **Compiler support**
  - GCC + Binutils + SPIKE support for *RVV* 0.8, going to 0.9
  - Compiler support is not optimized
- **Refactoring our benchmark kernels**
  - `Matmul, conv2d`
  - Manually vectorized
  - Intermix C + *RVV* assembly
- **Verification**
  - Extending *riscv-tests* with ad-hoc *RVV* tests and macros

- The Binutils port for v0.8
- The GNU toolchain port for v0.8
- The Spike simulator supports v0.10
- The RISC-V Proxy Kernel (to be use
  v0.8 capable GNU toolchain
- riscvOVPsim is a free RISC-V refere
  proprietary license, models are oper

- *[https://github.com/riscv/riscv-v-spec]*

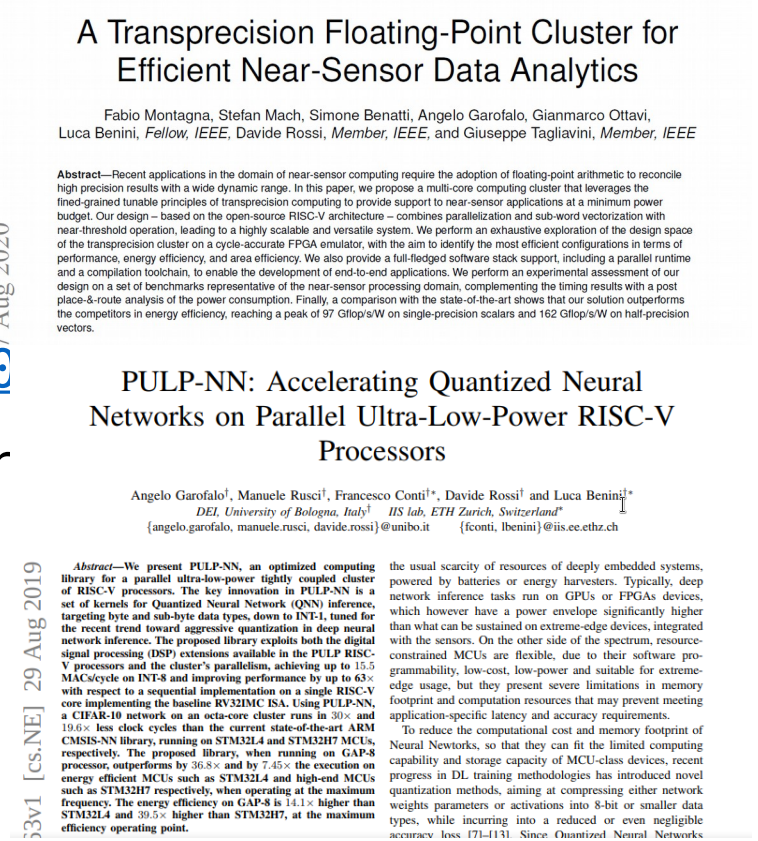
What‘s next? Benchmarks and ISA extensions

- Improve **benchmark pool**
  - FP Basic **Linear Algebra** Subprograms (**BLAS**)
    - AXPY, convolutions, gemv, gemm, …
  - **FP DSP programs**
    - FFT, DWT, FIR, IIR, K-Means, …

- Future **ISA extensions**
  - Fully support 16-bit and 8-bit SIMD FP computation
  - Xpulp-like extension, for fixed-point DSP

Going further – Benchmarking and comparisons

- **Compare Ara PPA with optimized DSP/ML systems**
  - PULP on FP benchmarks https://arxiv.org/pdf/200
    - Highly optimized FP benchmarks at various pr
  - PulpNN (ml workloads): https://arxiv.org/abs/1908.11263
    - Highly optimized library for QNN on 8-bit data
    - Experimental support for mixed precision on lower widths
  - PulpDSP (DSP workloads)
    - Highly optimized DSP library for PULP systems
    - Inspired by Arm CMSIS DSP
  - Baseline system: cluster with 8 Xpulpv2 RI5CY cores



A Transprecision Floating-Point Cluster for Efficient Near-Sensor Data Analytics

Fabio Montagna, Stefan Mach, Simone Benatti, Angelo Garofalo, Gianmarco Ottavi, Luca Benini, *Fellow, IEEE,* Davide Rossi, *Member, IEEE,* and Giuseppe Tagliavini, *Member, IEEE*

Abstract—Recent applications in the domain of near-sensor computing require the adoption of floating-point arithmetic to reconcile high precision results with a wide dynamic range. In this paper, we propose a multi-core computing cluster that leverages the fined-grained tunable principles of transprecision computing to provide support to near-sensor applications at a minimum power budget. Our design – based on the open-source RISC-V architecture – combines parallelization and sub-word vectorization with near-threshold operation, leading to a highly scalable and versatile system. We perform an exhaustive exploration of the design space of the transprecision cluster on a cycle-accurate FPGA emulator, with the aim to identify the most efficient configurations in terms of performance, energy efficiency, and area efficiency. We also provide a full-fledged software stack support, including a parallel runtime and a compilation toolchain, to enable the development of end-to-end applications. We perform an experimental assessment of our design on a set of benchmarks representative of the near-sensor processing domain, complementing the timing results with a post place-&-route analysis of the power consumption. Finally, a comparison with the state-of-the-art shows that our solution outperforms the competitors in energy efficiency, reaching a peak of 97 Gflop/s/W on single-precision scalars and 162 Gflop/s/W on half-precision vectors.

PULP-NN: Accelerating Quantized Neural Networks on Parallel Ultra-Low-Power RISC-V Processors

Angelo Garofalo†, Manuele Rusci†, Francesco Conti†*, Davide Rossi† and Luca Benini†*
DEI, University of Bologna, Italy† *IIS lab, ETH Zurich, Switzerland**
{angelo.garofalo, manuele.rusci, davide.rossi}@unibo.it {fconti, lbenini}@iis.ee.ethz.ch

***Abstract*—We present PULP-NN, an optimized computing library for a parallel ultra-low-power tightly coupled cluster of RISC-V processors. The key innovation in PULP-NN is a set of kernels for Quantized Neural Network (QNN) inference, targeting byte and sub-byte data types, down to INT-1, tuned for the recent trend toward aggressive quantization in deep neural network inference. The proposed library exploits both the digital signal processing (DSP) extensions available in the PULP RISC-V processors and the cluster's parallelism, achieving up to 15.5 MACs/cycle on INT-8 and improving performance by up to 63× with respect to a sequential implementation on a single RISC-V core implementing the baseline RV32IMC ISA. Using PULP-NN, a CIFAR-10 network on an octa-core cluster runs in 30× and 19.6× less clock cycles than the current state-of-the-art ARM CMSIS-NN library, running on STM32L4 and STM32H7 MCUs, respectively. The proposed library, when running on GAP-8 processor, outperforms by 36.8× and by 7.45× the execution on energy efficient MCUs such as STM32L4 and high-end MCUs such as STM32H7 respectively, when operating at the maximum frequency. The energy efficiency on GAP-8 is 14.1× higher than STM32L4 and 39.5× higher than STM32H7, at the maximum efficiency operating point.**

the usual scarcity of resources of deeply embedded systems, powered by batteries or energy harvesters. Typically, deep network inference tasks run on GPUs or FPGAs devices, which however have a power envelope significantly higher than what can be sustained on extreme-edge devices, integrated with the sensors. On the other side of the spectrum, resource-constrained MCUs are flexible, due to their software programmability, low-cost, low-power and suitable for extreme-edge usage, but they present severe limitations in memory footprint and computation resources that may prevent meeting application-specific latency and accuracy requirements.

To reduce the computational cost and memory footprint of Neural Newtorks, so that they can fit the limited computing capability and storage capacity of MCU-class devices, recent progress in DL training methodologies has introduced novel quantization methods, aiming at compressing either network weights parameters or activations into 8-bit or smaller data types, while incurring into a reduced or even negligible accuracy loss [7]–[13]. Since Quantized Neural Networks

63v1 [cs.NE] 29 Aug 2019

7 Aug 2020

PULP DSP: Digital Signal Processing on Parallel Ultra Low Power Platform

Introduction

This repository contains the work in progress of an optimized DSP library for PULP platforms.

It contains the kernel functions for different ISA extensions of RISC-V being developed for PULP platforms (RV32IM, XPULPV2, etc.).

Currently it's being developed and tested on Mr.Wolf (fabric controller (FC) with IBEX (previous zero-riscy), i.e. RV32IM, and cluster with 8 RISCY cores, i.e. RV32IM with XPULPV2 extentions.).

What‘s next? Vector-optimized library

- **Comparison** with the **optimized baseline**
- **Detect bottlenecks, improve performance + efficiency**
  - New RVV instructions, architectural changes
  - SIMD sub-byte computation
- **Develop** optimized **RVV library** for **DSP / ML** applications

Interrupts, Context switches

- **`mstatus.VS` CSR bit**
  - `Off/Initial/Dirty/Clean` → Reduce context switch overhead
- **Trap handling**
  - The `*epc` set to the faulty instruction address
  - `vstart` CSR updated to the offending element index, to restart from there
  - Some implementations: no interrupts during arith instructions
  - `vl` is not changed!
- **Precise traps**
  - Relaxed constraint on the elements following the offending one if idempotent.
- Use imprecise traps when resuming is not possible
- **Load/Store whole register** if `vl/vtype` are not known → They operates on `EEW=8`