EPAC: the last dance

Experiences and learnings while building EU technology

Filippo Mantovani - Barcelona Supercomputing Center

European Processor Initiative (EPI)

VISION:

Geopolitical shifts and COVID-19 supply chain risks exposed Europe's dependency on non-European processor technologies.

⚒️ **EPI responds by building a sovereign HPC processor ecosystem.**

EPI-SGA1 €80M budget

EPI-SGA2 €70M budget

2018 — 2021 — 2022 — 2026

Dual track technology development following the host-design compute node paradigm:

CPU based on Arm and lead by SiPearl.

RISC-V accelerators driven by a consortium of academic and industrial partners.

EPAC: EPI Accelerator v1.5

GF22FDX, 27 mm^2, 0.3 Btr, Tape out: Mar 2023, Bring up: Oct 2023

VEC tile

General purpose RISC-V CPU
Avispado Core (16 kI$, 32 kD$)
with dedicated VPU.
Up to 256 DP element vector length.

VRP tile

General purpose RISC-V CPU
supporting variable precision
arithmetic up to 256 bit elements.

STX tile

RISC-V many-core machine learning
accelerator targeting stencil and tensor
arithmetics.

L2-HN tile

Distributed L2 cache (256 kB/slice) and
Coherence Home Node.

CHI NoC and SerDes

On-chip high-speed network based on
multiple CHI cross points (XP).
Off-chip link based on SerDes.

Physical design Fraunhofer

Prototype board integration E4 COMPUTER ENGINEERING

EPAC: EPI Accelerator v1.5

GF22FDX, 27 mm^2, 0.3 Btr, Tape out: Mar 2023, Bring up: Oct 2023

VEC tile

General purpose RISC-V CPU
Avispado Core (16 kI$, 32 kD$)
with dedicated VPU.
Up to 256 DP element vector length.

VRP tile

General purpose RISC-V CPU supporting variable precision arithmetic up to 256 bit elements.

STX tile

RISC-V many-core machine learning accelerator targeting stencil and tensor arithmetics.

L2-HN tile

Distributed L2 cache (256 kB/slice) and Coherence Home Node.

CHI NoC and SerDes

On-chip high-speed network based on multiple CHI cross points (XP).
Off-chip link based on SerDes.

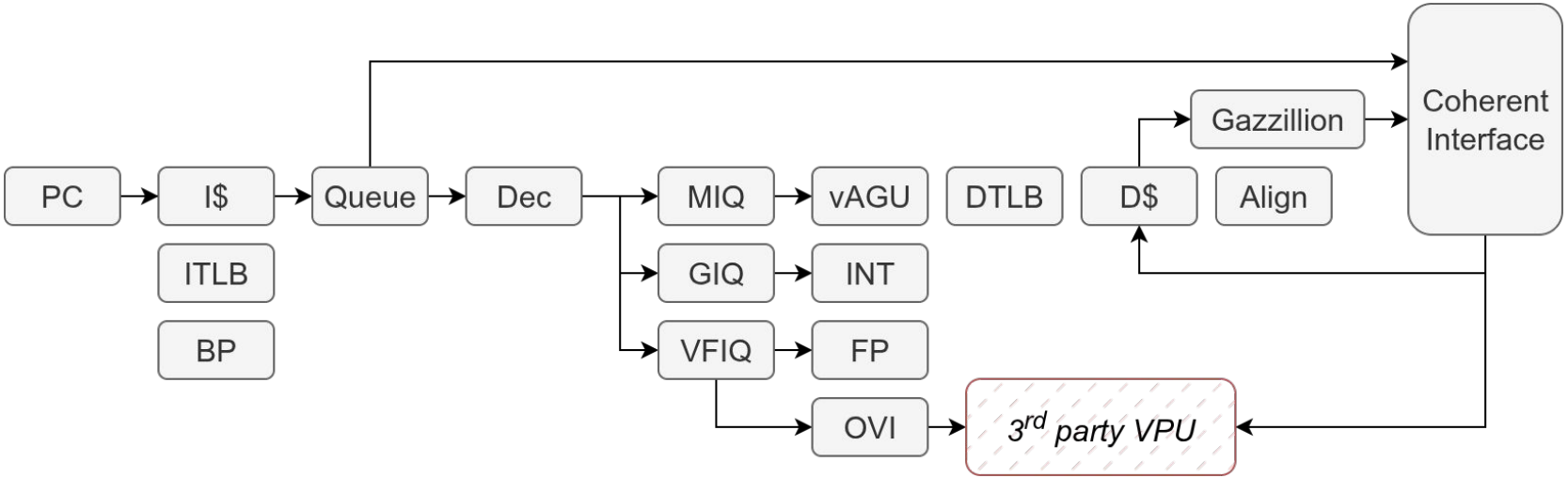
Physical design Fraunhofer

Prototype board integration E4 COMPUTER ENGINEERING

Scalar RISC-V core (Avispado)

- It boots Linux
- The scalar in-order RISC-V core releases several requests of cache lines to the memory
- The core is connected to a Vector Processing Unit (VPU) with very wide vector registers (16kb)

16 kB instruction cache
32 kB L1 data cache
1 MB L2 cache
Decodes RVV v0.7 vector extension
Cache coherent (CHI)
Vector memory accesses (vle, vlse, vlxe, vse, …) processed by a dedicated queue (MIQ/LSU)



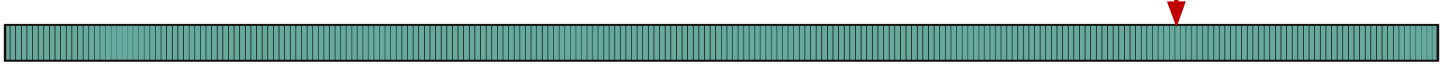
VPU with long Vector Length (VL) support

– AVX512 ← 512 bits per vector (8 DP elems)

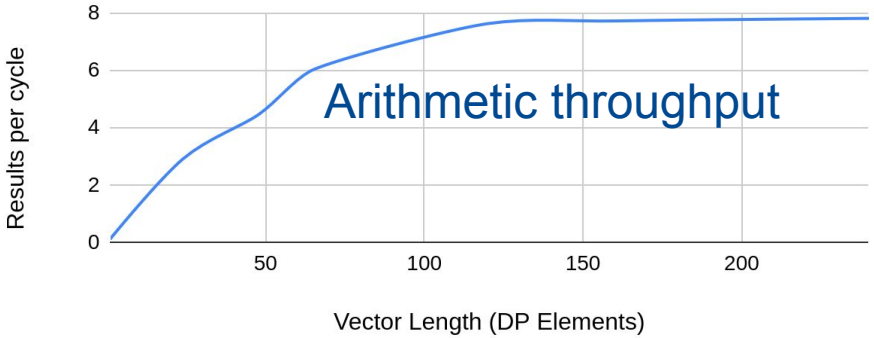
– SVE ← Up to 2048 bits per vector (16 DP elems)

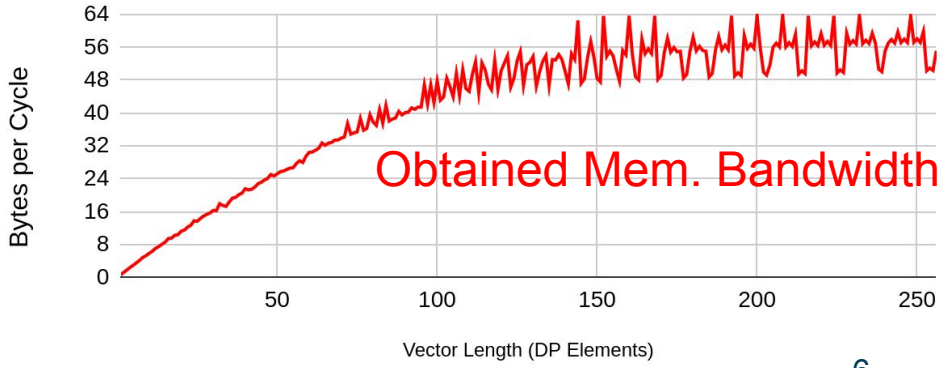
– RVV

16384 bits per vector (256 DP elems)

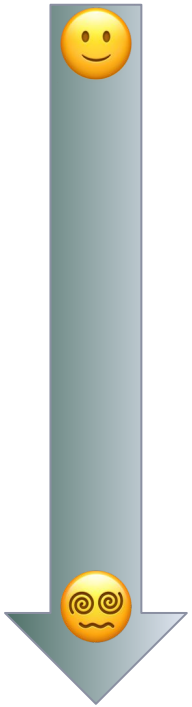


It's important to fill up the vectors!





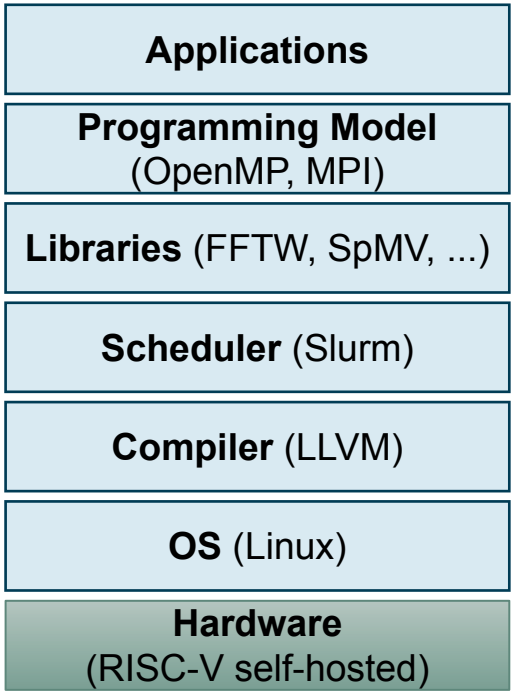
How do I program EPAC?



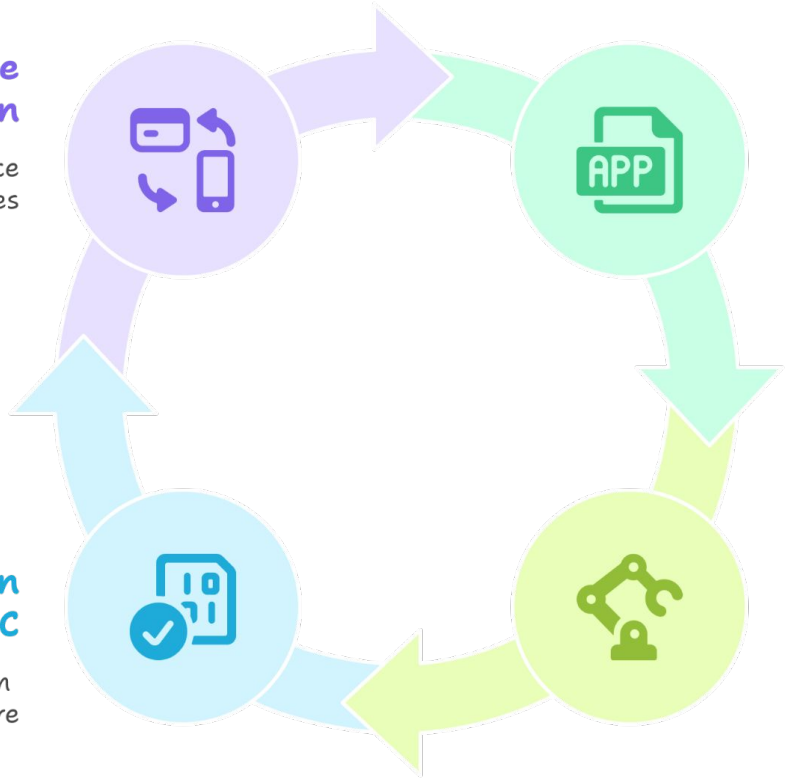
- **Auto-vectorization**
  - Leave it to the compiler
- **#pragma omp simd** (aka “Guided vectorization”)
  - Relies on vectorization capabilities of the compiler
    - Usually works but gets complicated if the code calls functions
  - Also usable in Fortran
- **C/C++/FORTRAN builtins** (aka “Intrinsics”)
  - Low-level mapping to the instructions
  - Allows embedding it into an existing C/C++ codebase
  - Allows relatively quick experimentation
- **Assembler**
  - Always a valid option but not the most pleasant

How do I use EPAC?

- Like a standard HPC system!
- Compile your code
  - We give you a compiler
- Link libraries
- Write/Submit a job script
  - SLURM
- Wait for the results
- Analyse execution traces and study how well your code is vectorized



Co-design methodology



Cross-Architecture Comparison

Comparing performance across architectures

App Preparation

Preparing and porting apps to RISC-V

Auto-Vectorization

Optimizing code for long vectors using emulators

Execution on EPAC-VEC

Running code on EPAC-VEC hardware

CEEC: Fluid dynamic

Application	1. App. preparation			2. Vector compilation & emulation			3. Execution on EPAC-VEC		4. Cross-architecture comparison
	Code	Input	Scalar RISC-V run	Compile w EPI-LLVM with auto-vectorization	Emulate (QEMU + RAVE)	Analyze & Optimize	Execute	Analyze & Optimize	
Alya (mini-app fluid matrix assembly)	✅	✅	✅	✅	✅	✅	✅	✅	✅
Alya (mini-app solid matrix assembly)	✅	✅	✅	✅	✅	✅	✅	✅	✅
FLEXI	✅	✅	✅	✅	✅	⚙️	✅		
WaLBerla	✅	✅	✅	✅	✅	✅	✅	✅	⚙️
SOD2D	✅	✅	✅	✅	✅	⚙️	✅	⚙️	
NEKO	✅	✅	✅	✅	⚙️				
NekRS	✅	✅	✅	🚧					

✅ Done ⚙️ Work in progres 🚧 Potential road block due to just in time compilation

Plasma-PEPSC: Plasma physics

Application	1-Scalar RISC-V	2-Emulated RISC-V Vector			3-FPGA RISC-V Vector (EPAC-VEC)		4-Study portability
	Compile & run on EPI clusters	Compile with LLVM/EPI compiler	Emulate (vehave / qemu)	Analyze & optimize	Execute	Analyze & optimize	Compare with other architectures
BIT1	✅	✅	✅	✅	✅	🏗️	
GENE	✅	✅	✅	🏗️	🚧(*)		
GENE-X	✅	✅	✅	🚧(**)	✅	🚧(**)	
PIConGPU	✅	✅	✅	🏗️			
Vlasiator	✅	✅	✅	✅	✅	✅	🏗️

(*): flang compilation errors

(**): major refactor of the main operator

ChEESE: Earth Science

App	1-Scalar RISC-V	2-Emulated RISC-V Vector			3-FPGA RISC-V Vector (EPAC-VEC)		4-Arch comparison
	Compile & run	Compile (Vectorize)	Emulate	Analyze & optimize	Execute	Analyze & optimize	
SPECFEM3D	✅ (*)						
HySEA	✅ (*)	✅ (*)	✅ (*)	🏗️	🏗️		
SeisSol	✅	✅	✅	🏗️	✅	🏗️	🏗️
FALL3D	✅ (*)	✅ (*)	✅ (*)	✅ (*)	✅ (*)	✅ (*)	✅ (*)
Tandem							
PTatin3D							
Xshells	✅ (*)	✅ (*)	✅ (*)	🏗️	✅ (*)	🏗️	

(*): on mini-app

Conclusions & Lessons Learned

Validated Silicon Platform

EPAC proves a distributed European consortium can deliver a full chip (architecture through tapeout and bring-up) with three silicon-proven RISC-V IPs.

Coordination Is a First-Class Constraint

Integrating RTL from multiple organizations required interface standardization, naming resolution, and toolchain alignment. As complexity grows, coordination overhead demands explicit planning.

Software Infrastructure Endures

LLVM toolchains, VBLAS libraries, and SDV emulation tools extend the project's value beyond the chip itself, feeding directly into future European processor developments.

EPI FUNDING

This project has received funding from the European High Performance Computing Joint Undertaking (JU) under Framework Partnership Agreement No 800928 and Specific Grant Agreement No 101036168 EPI-SGA2.

The JU receives support from the European Union's Horizon 2020 research and innovation programme and from Croatia, France, Germany, Greece, Italy, Netherlands, Portugal, Spain, Sweden, and Switzerland.

SPONSORED BY THE

With funding from the: