



# **The European Processor Initiative: Platforms and tools for co-design with RISC-V accelerators**

Filippo Mantovani  
Barcelona Supercomputing Center (BSC)

# EPI MAIN OBJECTIVE

To develop European microprocessor and accelerator technology

- Strengthen competitiveness of EU industry and science

SiPearl

**Rhea**

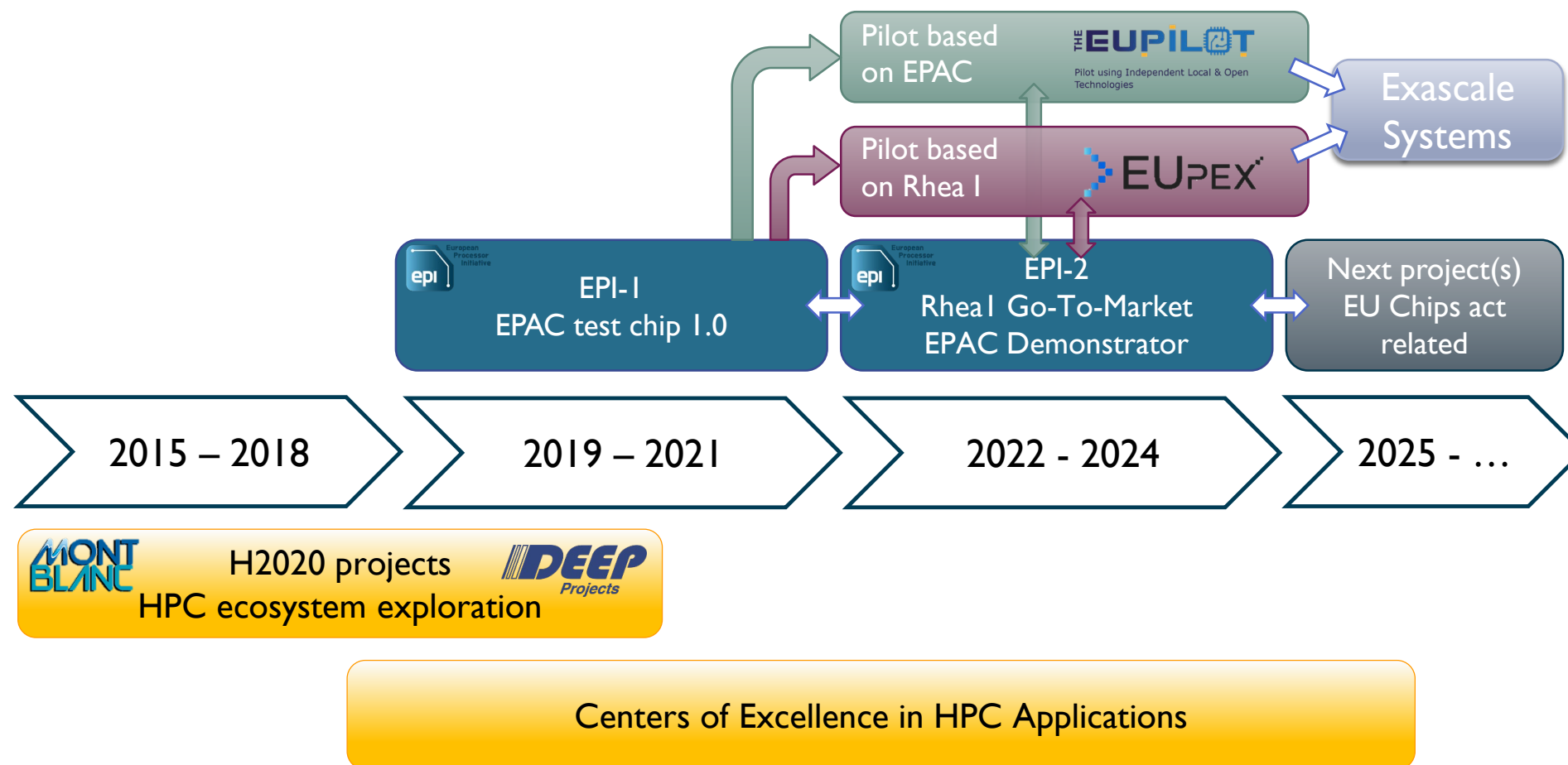
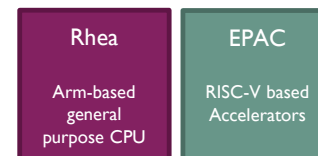
Arm-based  
general purpose  
CPU

**EPAC**

RISC-V based  
Accelerators

BSC, SemiDynamics,  
EXTOLL, FORTH, ETHZ,  
UniBo, UniZG, Chalmers,  
CEA, E4, Menta, ZPT, ...

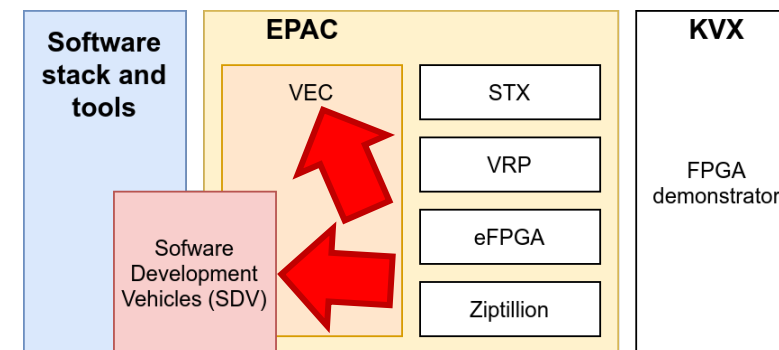
# OVERALL TECHNOLOGY ROADMAP



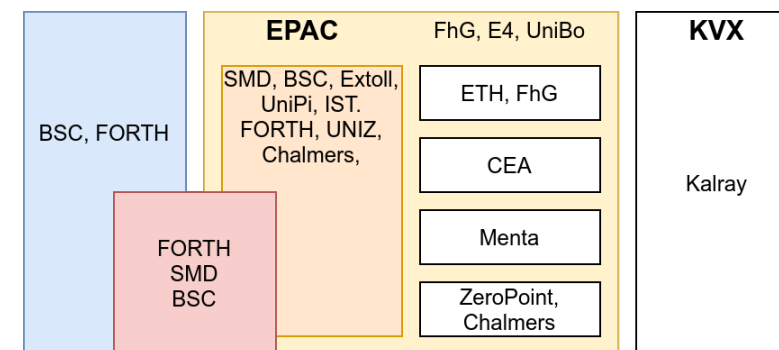
# VISIONS AND COLLABORATIONS

- **VEC** - Self-hosted RISC-V CPU + wide VPU (256 double elements) supporting RVV 0.7.1 / 1.0
- **STX** - RISC-V CPU + specific cores for stencil and neural network computation
- **VRP** - RISC-V CPU with support for variable precision arithmetic (data size up to 512 bit)
- **eFPGA** - On-chip reconfigurable logic
- **Ziptillion** - IP compressing/decompressing data to/from the main memory
- **KVX** - FPGA demonstrator of the Kalray RISC-V CPU targeting HPC and ML

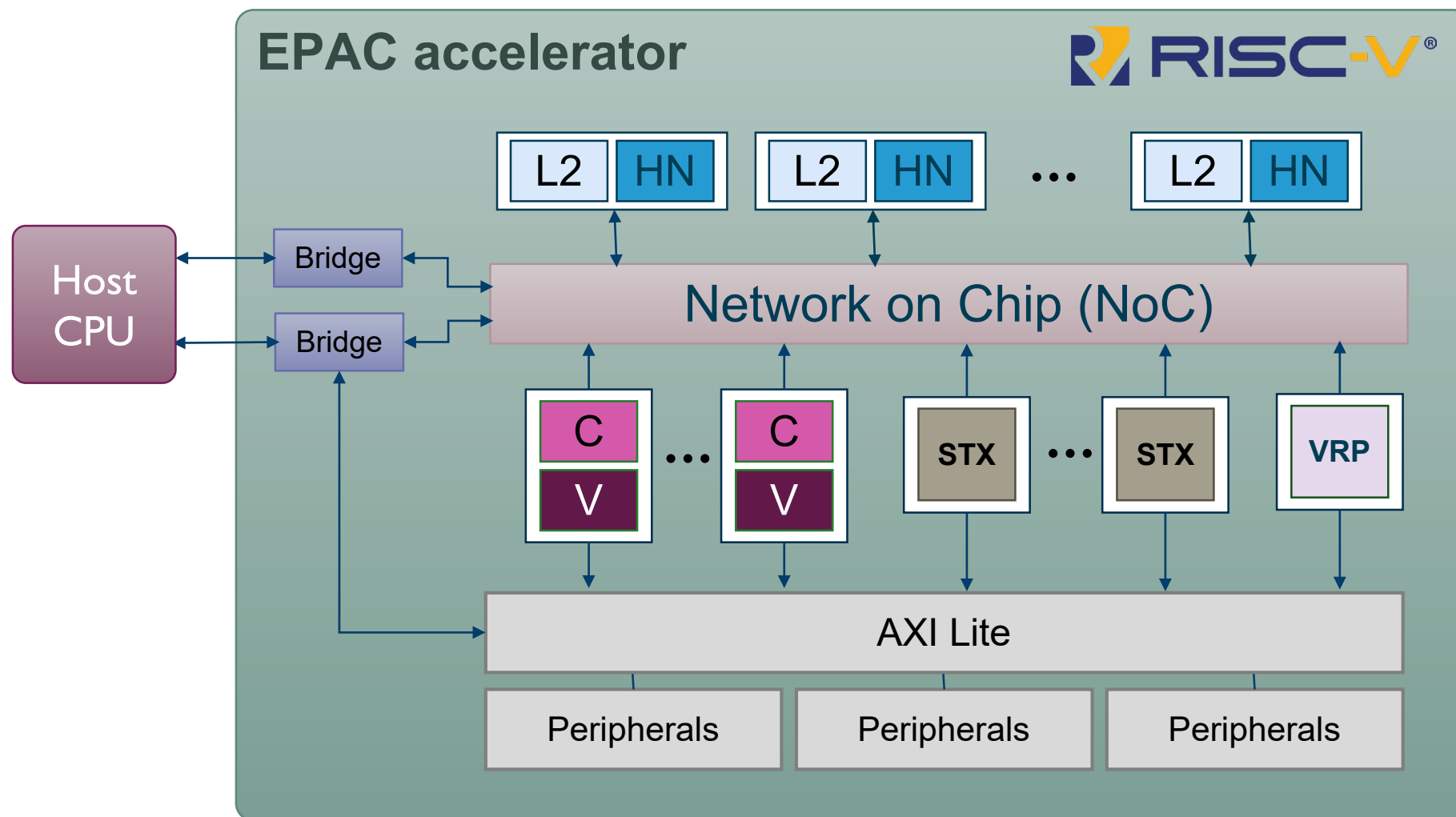
## VISION



## COLLABORATIONS



# EPAC: A RISC-V ACCELERATOR

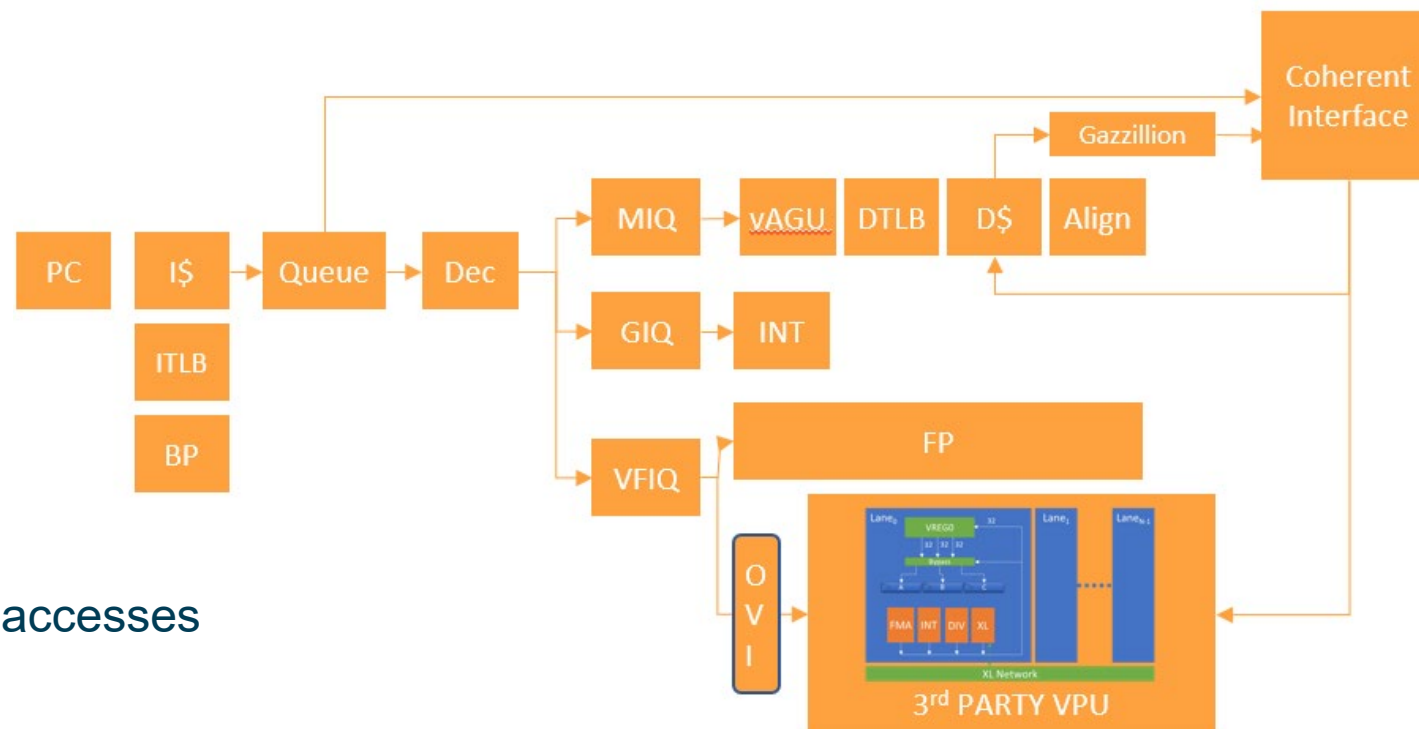


# AVISPADO 220 WITH VPU

## RISCV64GCV



- SV48
- 16KB I\$
- Decodes v0.7, v1.0 vector spec
- 32KB D\$
- Full hardware support for unaligned accesses
- Coherent (CHI)
- Vector Memory (vle, vlse, vlxe, vse, ...) processed by MIQ/LSU

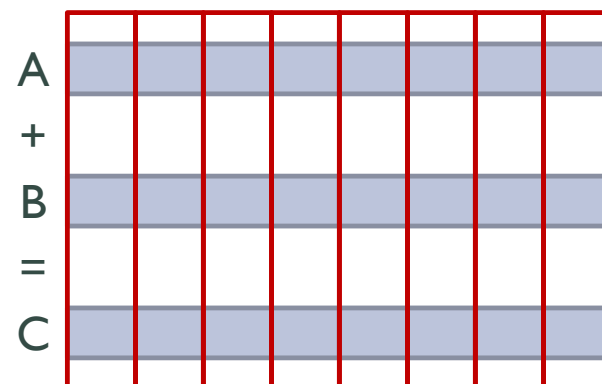


Courtesy Semidynamics

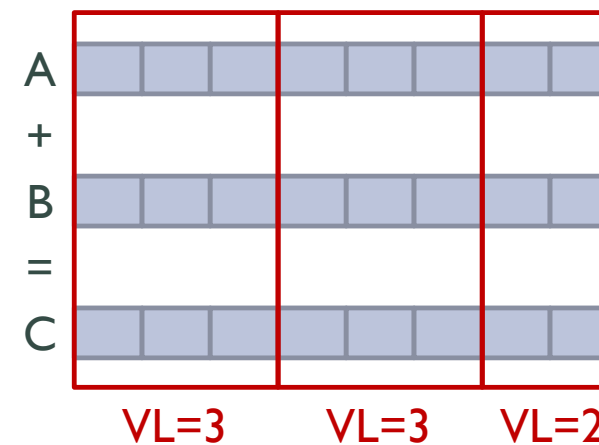


# EXAMPLE: $VL = 3$

Scalar

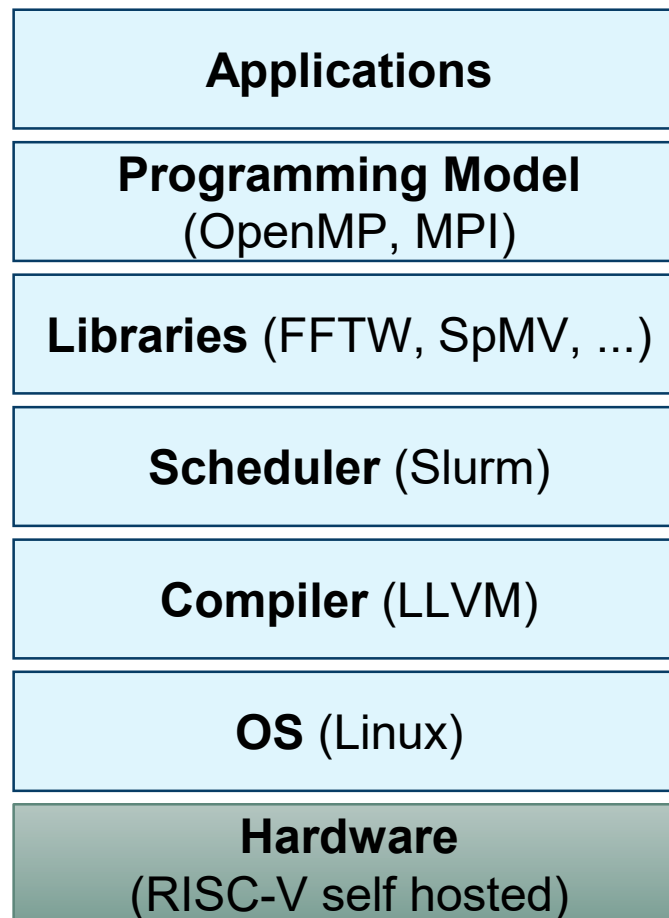


Vector





# EPAC PROGRAMMING ENVIRONMENT



# HOW TO USE THE V-EXTENSION?

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

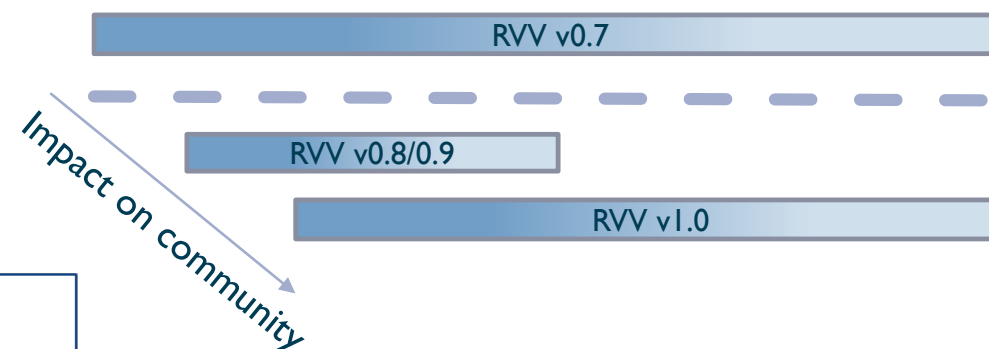
# RISC-V VECTOR EXTENSION (RVV) COMPILER

- LLVM support for the evolution of the RISC-V Vector (RVV) Extension
- Intrinsics
- Autovectorization

```
void SpMV_vec(double *a, long *ia, long *ja, double *x, double *y, int nrows) {
    for (int row = 0; row < nrows; row++) {
        int nnz_row = ia[row + 1] - ia[row];
        unsigned long gvl; // granted vector lengths
        int idx = ia[row];
        __epi_1xf64 v_a, v_x, v_prod, v_partial_res;
        __epi_1xi64 v_idx_row, v_three;
        y[row] = 0.0;
        v_partial_res = __builtin_epi_vbroadcast_1xf64(0.0, gvl);
        for (int colid = 0; colid < nnz_row; ) { // blocking on MAXVL
            gvl = __builtin_epi_vsetvl(nnz_row - colid, __epi_e64,
                v_a = __builtin_epi_vload_1xf64(&a[idx+colid], gvl);
                v_idx_row = __builtin_epi_vload_1xi64(&ja[idx+colid], gvl);
                v_three = __builtin_epi_vbroadcast_1xi64(3, gvl);
                v_idx_row = __builtin_epi_vsl_1xi64(v_idx_row, v_three, gvl);
                v_x = __builtin_epi_vload_indexed_1xf64(x, v_idx_row, g
                v_prod = __bu
                v_partial_res =
            }
            colid += gvl;
        }
        y[row] += __builtin_
    }
}

void target_inner_3d (...) {
    for (i) {
        for (j) {
            #pragma clang loop vectorize(enable)
            for (k) {
                lap = LAP;
                v[IDX3_1(i,j,k)] = 2.f*u[IDX3_1(i,j,k)]
                    -v[IDX3_1(i,j,k)]+vp[IDX3(i,j,k)]*lap;
            }
        }
    }

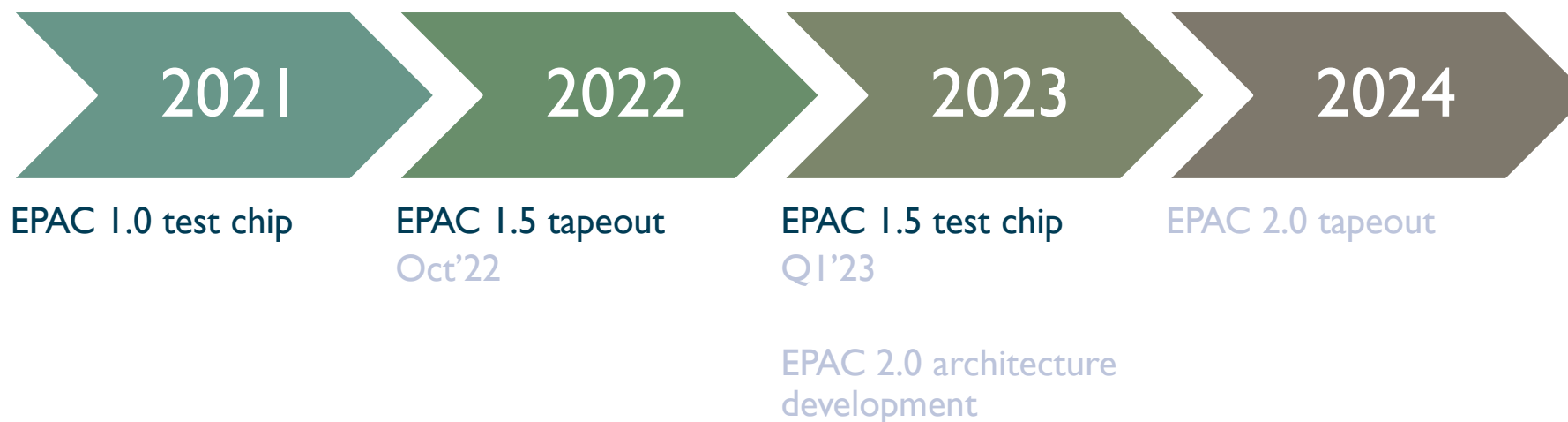
    ...
    float complex A[n][n];
    float complex templ, temp2;
    ...
    for (int j = 0; j < n; j++) {
        if (x[j] != ZERO || y[j] != ZERO) {
            templ = alpha * conjunction(creal(y[j]), cimag(y[j]));
            temp2 = conjunction(creal(alpha * x[j]), cimag(alpha * x[j]));
            #pragma clang loop vectorize(assume_safety)
            for (int i = 0; i <= j - 1; i++) A[i][j] = A[i][j] + x[i] * templ + y[i] * temp2;
            A[j][j] = creal(A[j][j]) + creal(x[j] * templ + y[j] * temp2);
        } else A[j][j] = creal(A[j][j]);
    }
}
```



Support EPAC 1.0  
and EPAC 1.5

Support EPAC 2.0

# EPAC TIMELINE





# **SOFTWARE DEVELOPMENT VEHICLES (SDV)**

# RISC-V PLATFORMS: COMMERCIAL AND FPGA-BASED



## Cluster of 10 HiFive Unmatched

- Operated as a standard HPC cluster (SLURM, modules, ...)



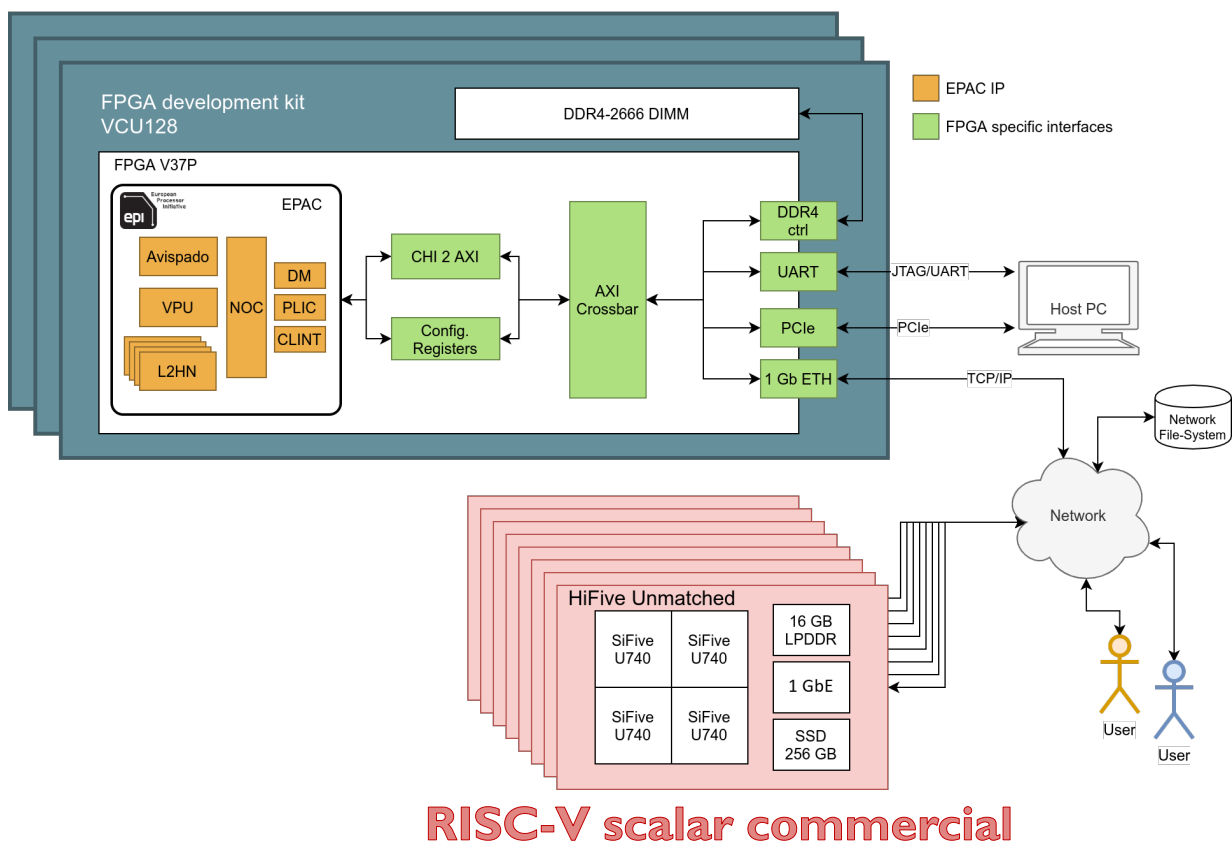
## Cluster of 3 host+FPGA (VCUI28)

- Operated as a standard HPC cluster once the FPGA is configured



# RISC-V PLATFORMS: COMMERCIAL AND FPGA-BASED

## Self hosted RISC-V vector node @ 50 MHz

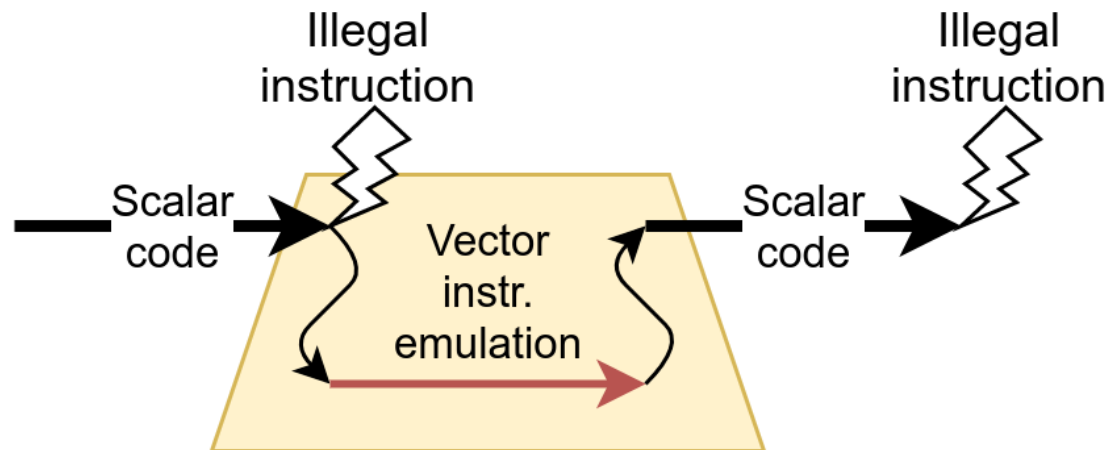


Scalar RISC-V commercial platforms coupled with EPAC development

- Hardware/Software infrastructure for Continuous Integration and RTL check
- Playground to demonstrate a full HPC software stack
  - Linux, compiler, libraries, job scheduler, MPI
- Platform to test latest RTL with complex codes
  - Advances performance analysis tools
  - Accurate timing available

# SOFTWARE EMULATOR: VEHAVE

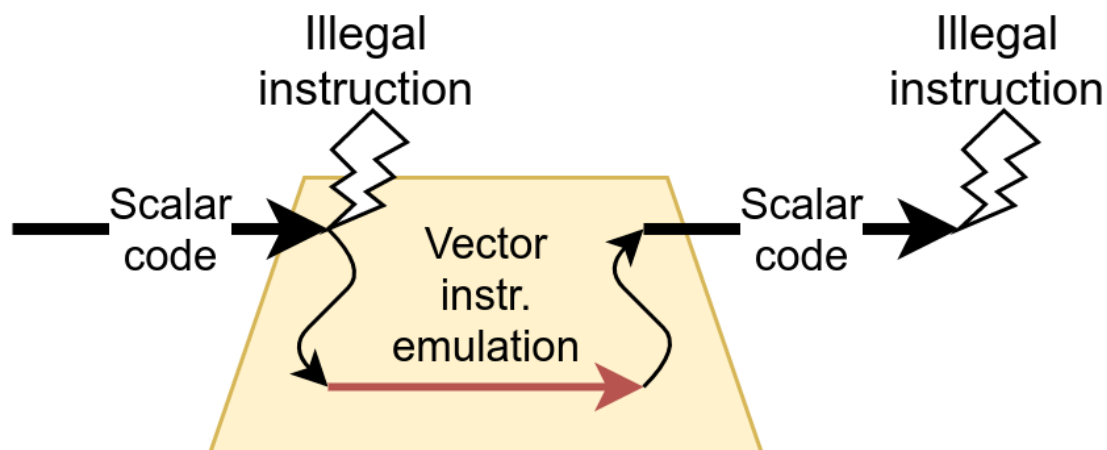
- Compile using the RISC-V Vector extension (RVV) Compiler
- Obtain a binary with vector instructions
- Run in a commercial RISC-V platform (scalar CPU)
- Obtain a trace with detailed information about the vectorization



Commercial RISC-V platform (scalar CPU)



# SOFTWARE EMULATOR: VEHAVE



## PROS:

- Useful to understand the level of vectorization achieved by the code
- Easy to use and accessible with no need of hardware infrastructure
- It supports RVV-0.7 and RVV-1.0
- Output compatible with Paraver

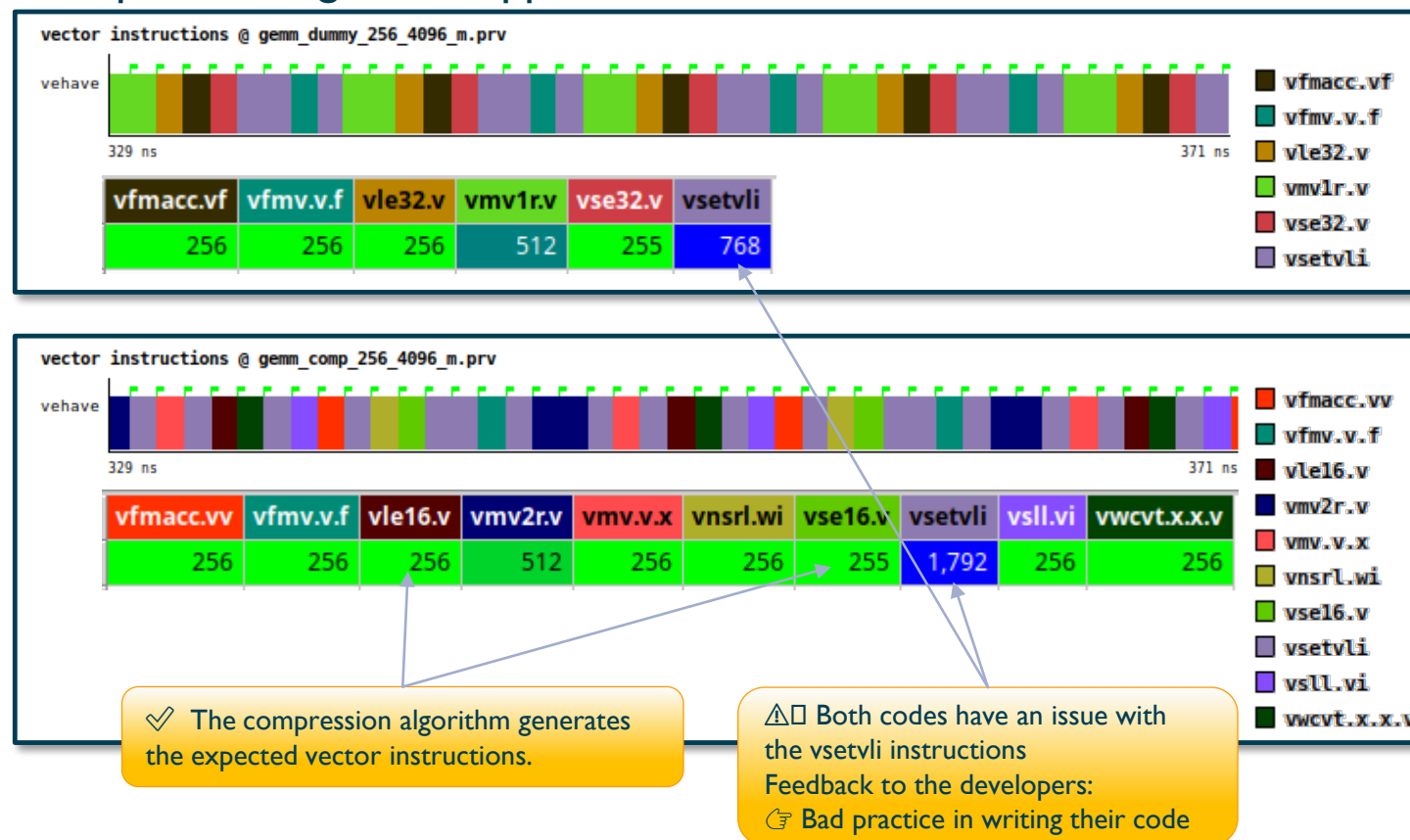
## CONS:

- Slow
- No information about performance (no timing)

# 1<sup>ST</sup> STEP: STUDY WITH VEHAVE

- Compile an application
  - Relying on autovectorization, with intrinsics, pragmas or assembly
  - Study the output of the compiler
- Run with emulation enabled
  - Collect execution traces
- Visualize and study traces
- How well the code is vectorized?
  - Feedback to the compiler team
  - Guidelines to the application developer

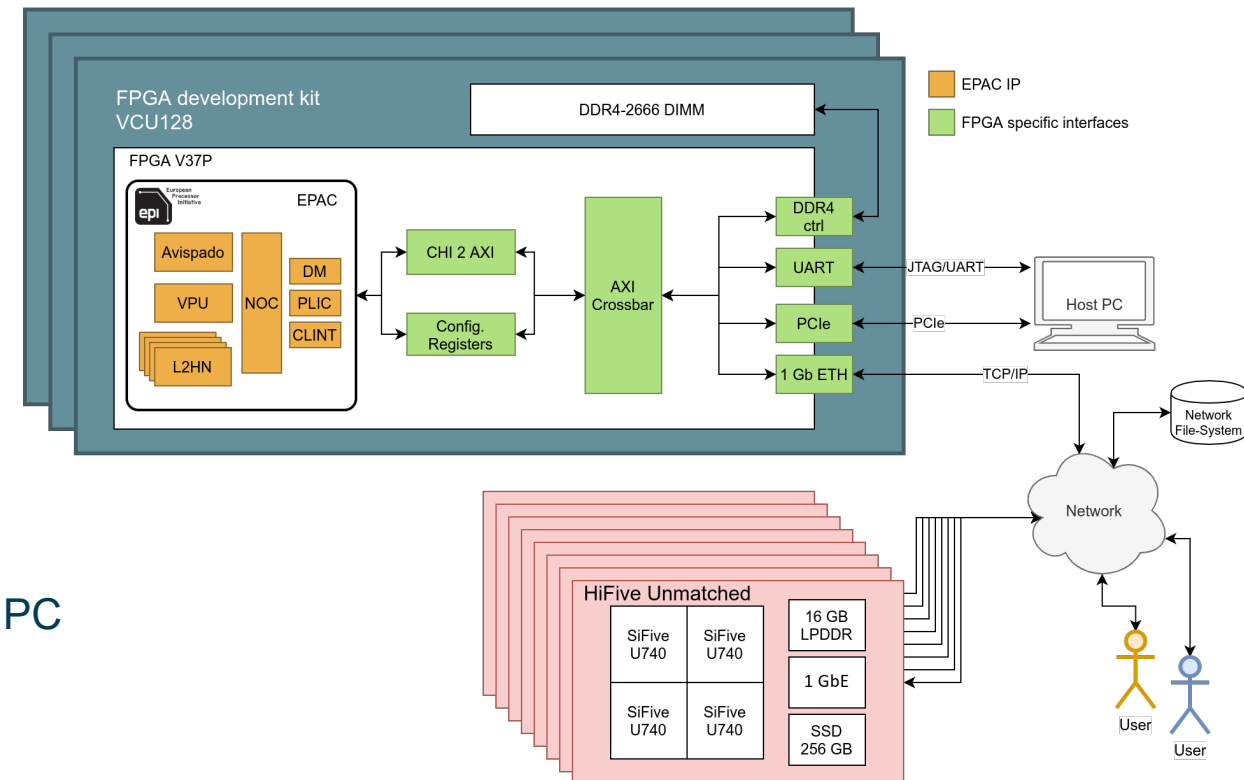
## Compression algorithm applied to a GEMM



# 2<sup>ND</sup> STEP: RUN ON FPGA-BASED SOFTWARE DEVELOPMENT VEHICLES

- Same RTL of EPAC mapped into FPGA
  - One tile (i.e., single core)
  - Running at 50 MHz
- Full HPC software stack and execution environment
  - Binary compatibility
  - Shared storage (NFS)
  - Multi-user / Multi-node (via MPI)
  - Standard debug and performance analysis tools for HPC

## Self hosted RISC-V vector node @ 50 MHz



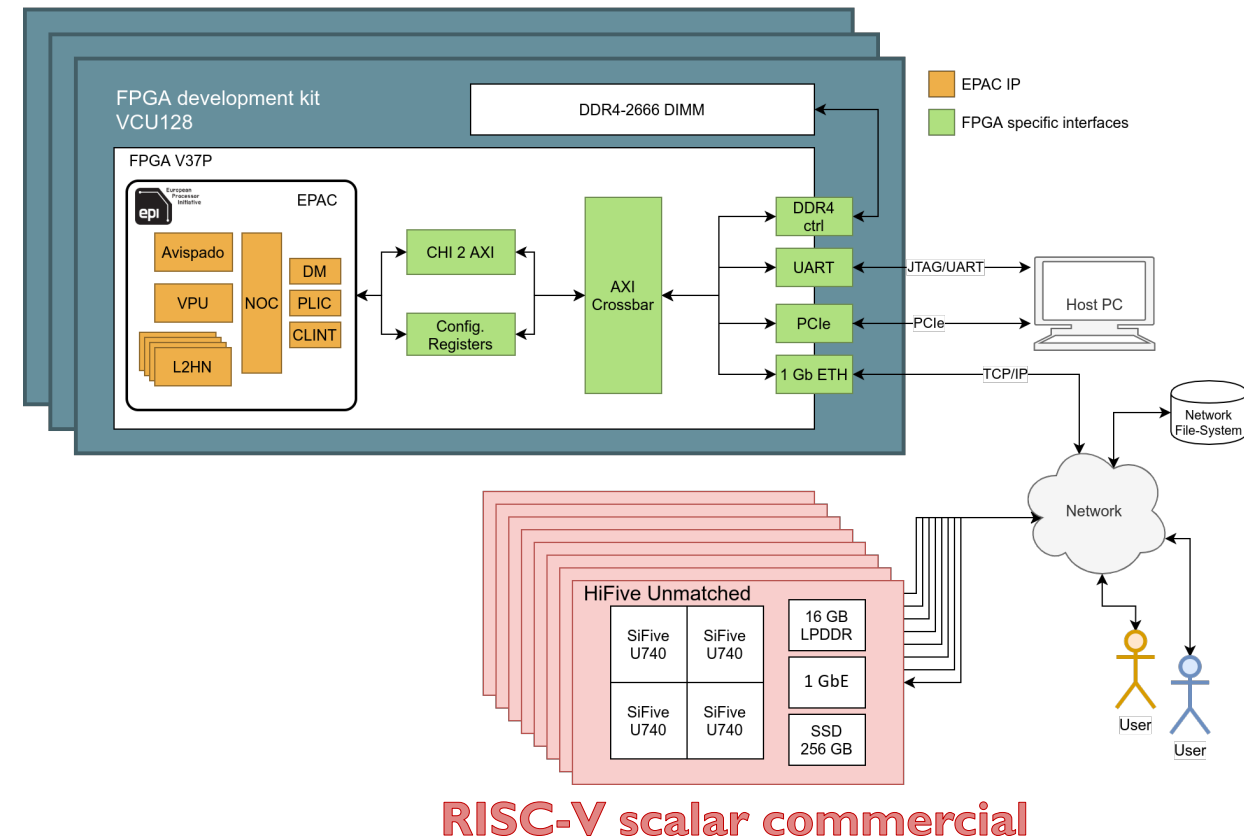
## RISC-V scalar commercial

# 2<sup>ND</sup> STEP: RUN ON FPGA-BASED SOFTWARE DEVELOPMENT VEHICLES

Added values:

- Test the platform before the hardware is available
- Test the effect of hardware design on complex codes before freezing the architectural specs
- Develop system-software, libraries and applications on top of a full HPC software stack
- Allow co-design between hardware and all software layers
  - In the end, SDV enables RISC-V in HPC!

## Self hosted RISC-V vector node @ 50 MHz

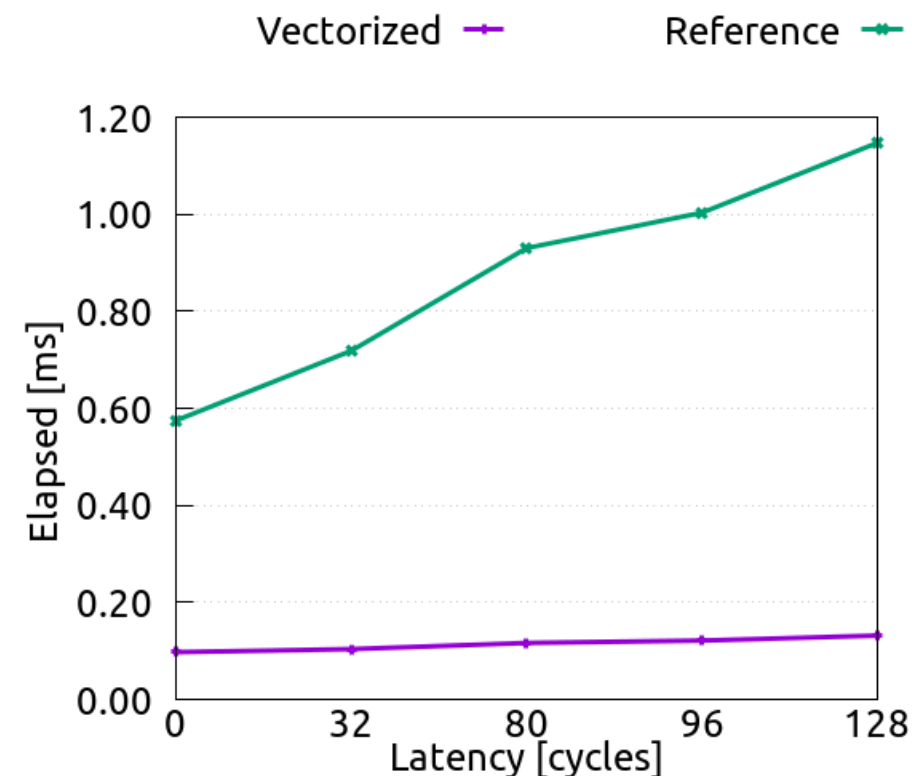
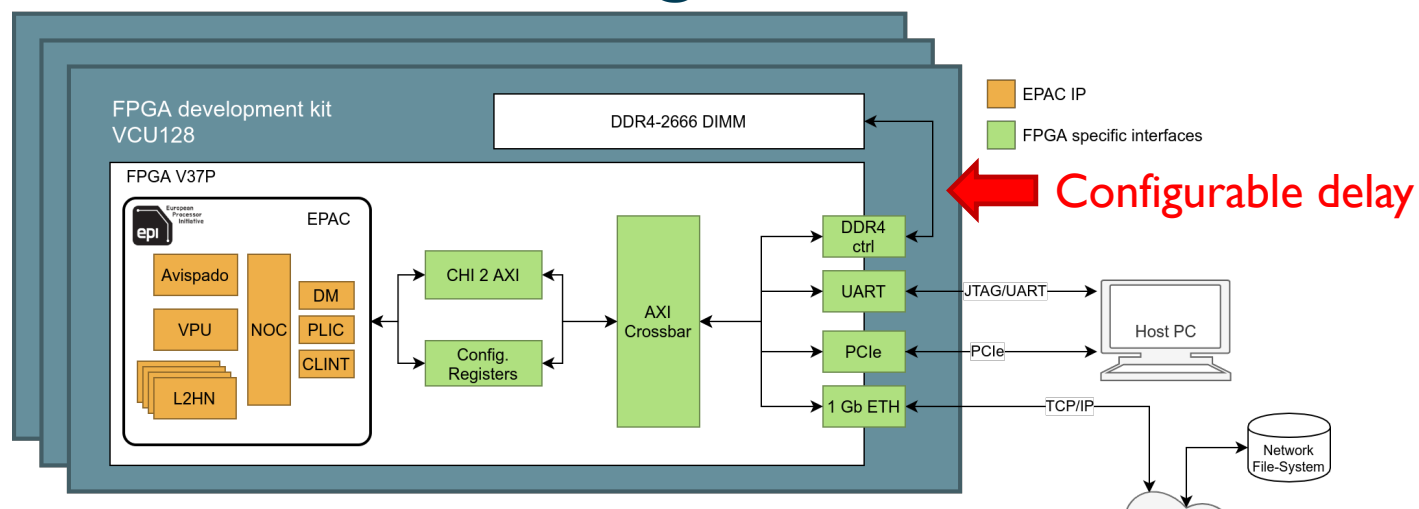


# EXAMPLE OF STUDY ON FPGA-SDV #1

## TEST OF HARDWARE CONFIGURATIONS

- How sensitive are vectorized/non vectorized codes to the latency accessing the memory?

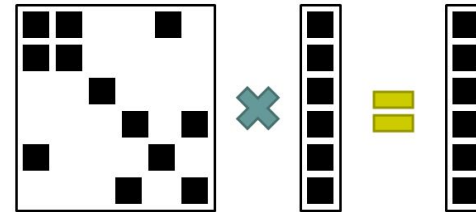
Self hosted RISC-V vector node @ 50 MHz



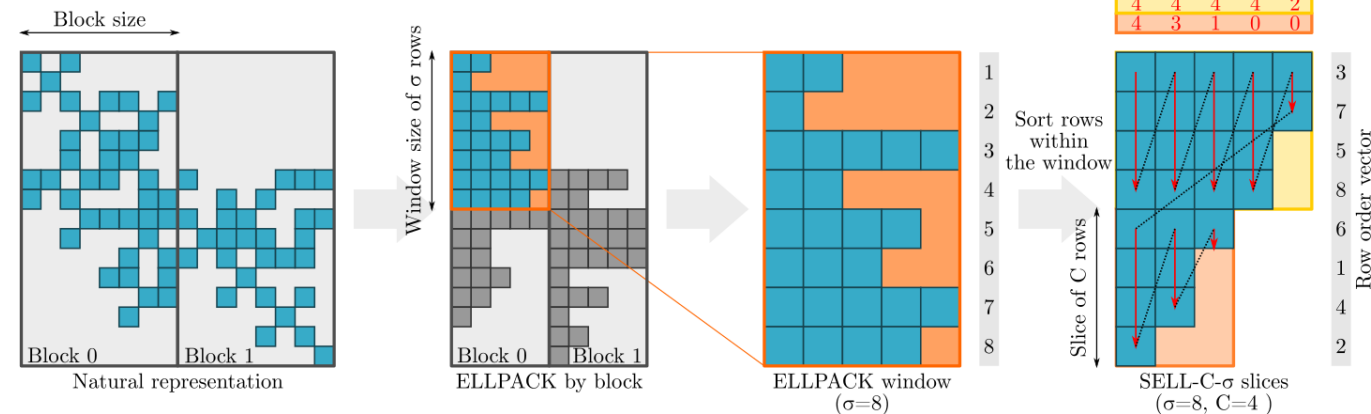
# EXAMPLE OF CO-DESIGN STUDY #2

## DEVELOPMENT OF OPTIMIZED LINEAR ALGEBRA LIBRARIES

- Sparse Matrix-Vector multiplication



- Naïve implementation struggles taking advantage of the vector unit
- Implementation of “vector friendly” SpMV algorithm: Sell-C-sigma



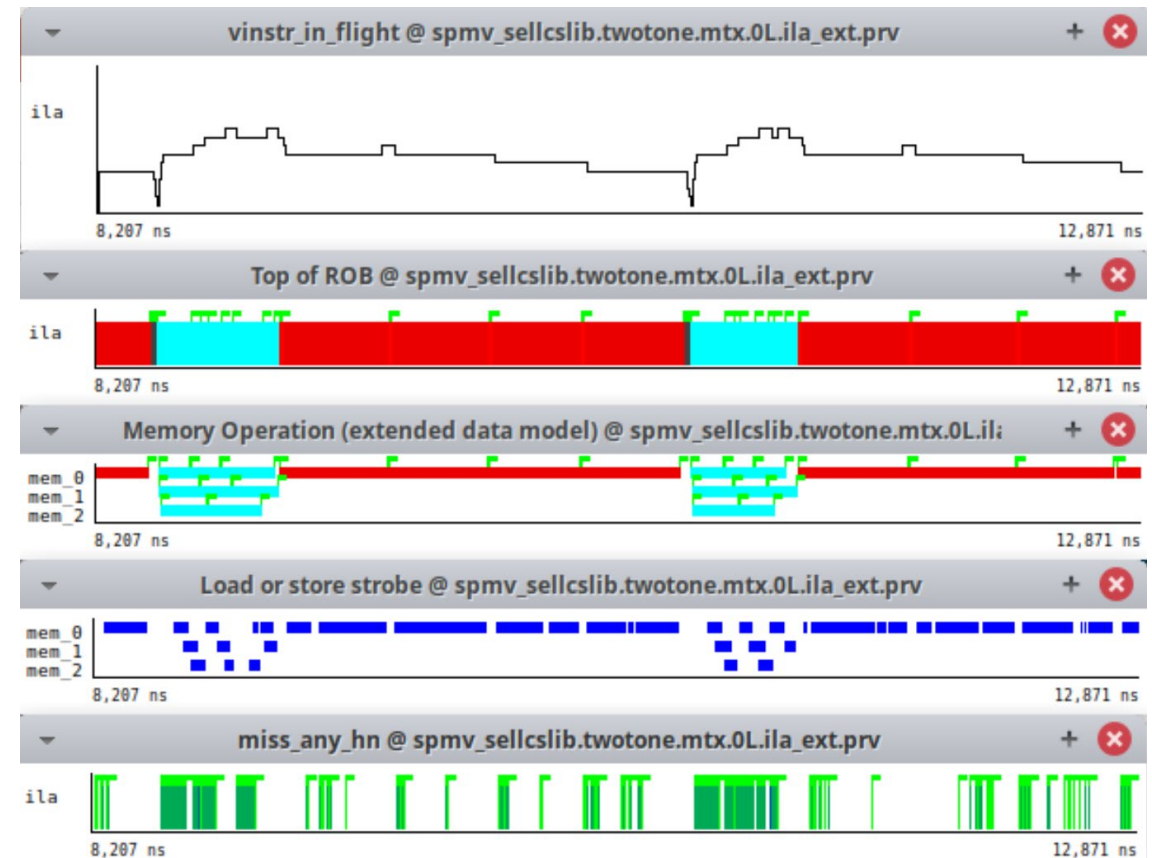
✎ Gómez, Constantino, Filippo Mantovani, Erich Focht, and Marc Casas. "Efficiently running SpMV on long vector architectures." In Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pp. 292-303. 2021.

✎ Vizcaino, Pablo, Filippo Mantovani, Roger Ferrer, and Jesus Labarta. "Acceleration with long vector architectures: Implementation and evaluation of the FFT kernel on NEC SX-Aurora and RISC-V vector extension." Concurrency and Computation: Practice and Experience (2022): e7424.

# 3<sup>RD</sup> STEP: SIGNAL ANALYSIS ON FPGA-SDV

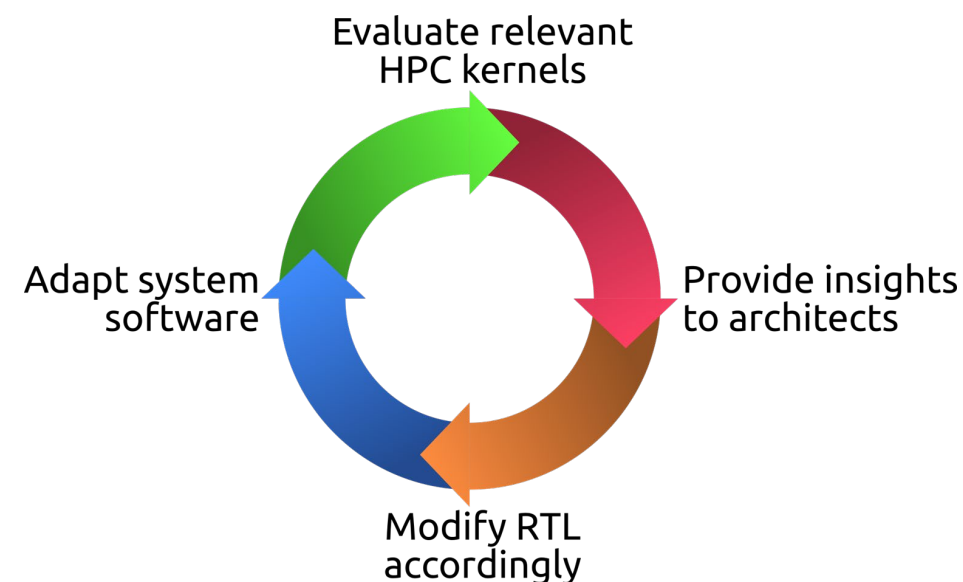
## INTEGRATED LOGIC ANALYSER

- Fine grained analysis (at level of instructions) is possible
- Graphical representation of timelines
- In depth study can help highlighting:
  1. Low usage of the vector unit
    - Feedback to the **code developer**
  2. Suboptimal saturation or resources (FU, mem)
    - Feedback to the **RTL implementation team**
  3. Suboptimal overlap of instructions
    - Feedback to the **compiler team** (improve scheduling)



# SUMMARY

- EPI develops the first RISC-V based accelerator targeting HPC leveraging the V-extension
  - RTL design of a Vector Unit
  - LLVM compiler support for the V-extension
- While RTL is becoming actual hardware, EPI develops tools for boosting the co-design cycle: Software Development Vehicles
  - Emulator of V-instructions (Vehave)
  - Commercial RISC-V-based clusters
  - FPGA-SDV: self-hosted RISC-V core coupled with a VPU on FPGA
  - Performance analysis tools
- We can leverage SDVs to:
  - Influence hardware design
  - Improve compiler autovectorization and support
  - Prepare and tune codes and libraries for long-vector architectures



Interested in testing your code on EPAC?  
✉ Filippo Mantovani [filippo.mantovani@bsc.es](mailto:filippo.mantovani@bsc.es)



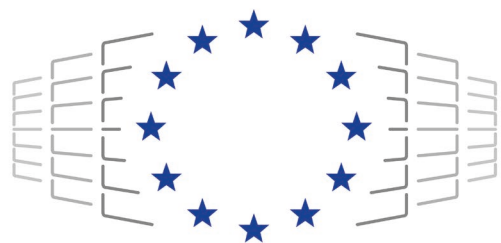
# CONTACTS

- Stream 3 leader and responsible for SDV:
  - Filippo Mantovani [filippo.mantovani@bsc.es](mailto:filippo.mantovani@bsc.es)
- EPI General Manager:
  - Etienne Walter [etienne.walter@atos.net](mailto:etienne.walter@atos.net)

# EPI PARTNERS



# EPI FUNDING



**EuroHPC**  
Joint Undertaking

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. The EPI-SGA2 project, PCI2022-132935 is also co-funded by MCIN/AEI /10.13039/501100011033 and by the UE NextGenerationEU/PRTR.



Federal Ministry  
of Education  
and Research



Fundação  
para a Ciência  
e a Tecnologia

**VINNOVA**  
Sweden's Innovation Agency



REPUBLIC OF CROATIA  
Ministry of Science and  
Education



Swedish  
Research  
Council



Financé  
par



