CERN Compute Accelerator Forum

VRP/VXP: Variable Extended Precision RISC-V Accelerator for HPC Scientific Applications

2024-10-09

Eric Guthmuller & Jérôme Fereyre

Contributors

Eric Guthmuller

Expert architect

Jérôme Fereyre

Lead software

Yves Durand

Numerical analyst

Tanuj-Kumar Khandelwal

Lead verification

Vincent Mengue

Lead frontend

Alexandre Hoffmann

Applications R&D

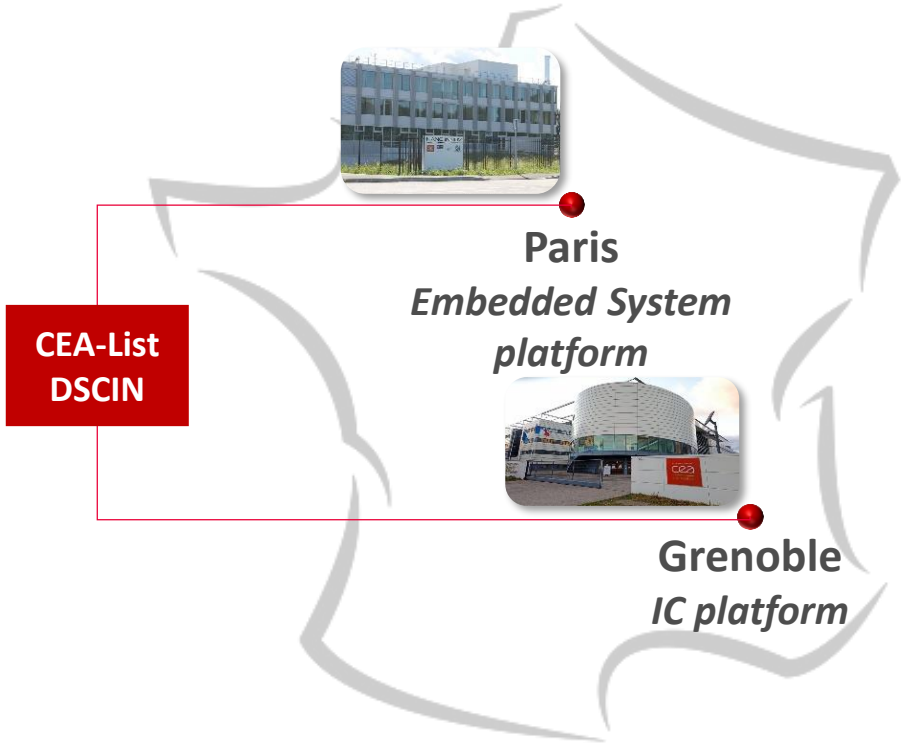
Andrea Bocco

Designer, arithmetic

Ihsane Tahir

Design/Verification

Our Department: System & Integrated Circuit



150 Staff members

2 500 m^2 of facilities

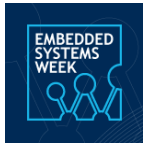
30 Industrial partners

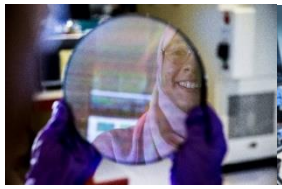
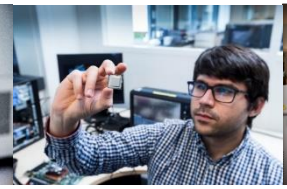
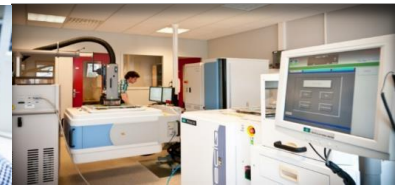
10 IC designs per year

25 Patents per year

100 Publications per year





Focus on Linear Solvers

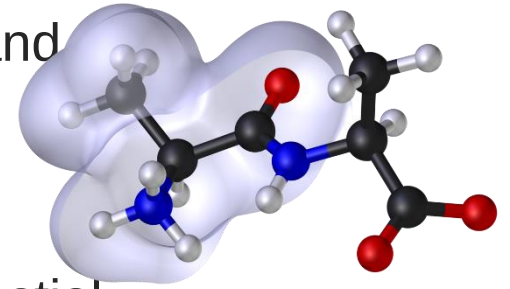
$$\tau_{yx} = -\mu \cdot \frac{dv_x}{dy}$$

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2 r}{\partial \theta^2} = -f(r,\theta)$$

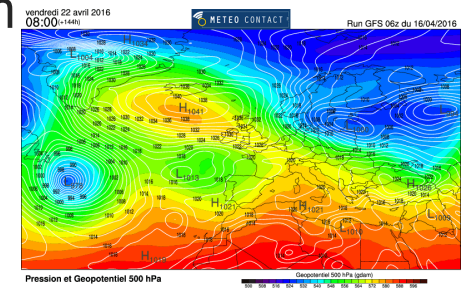
Starting point: symbolic manipulation of a set of PDE from CFD, convection/diffusion, Electromagnetism, etc.

e.g. variational formulation

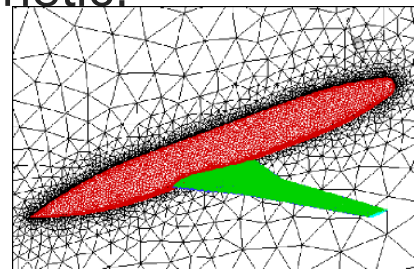
- Computational physics: Blend of physics and chemistry, with applied mathematics, numerical linear algebra, numerical optimization, and parallel computing.



- Climate model, weather prediction , celestial physics ,high energy physics, quantum mechanics

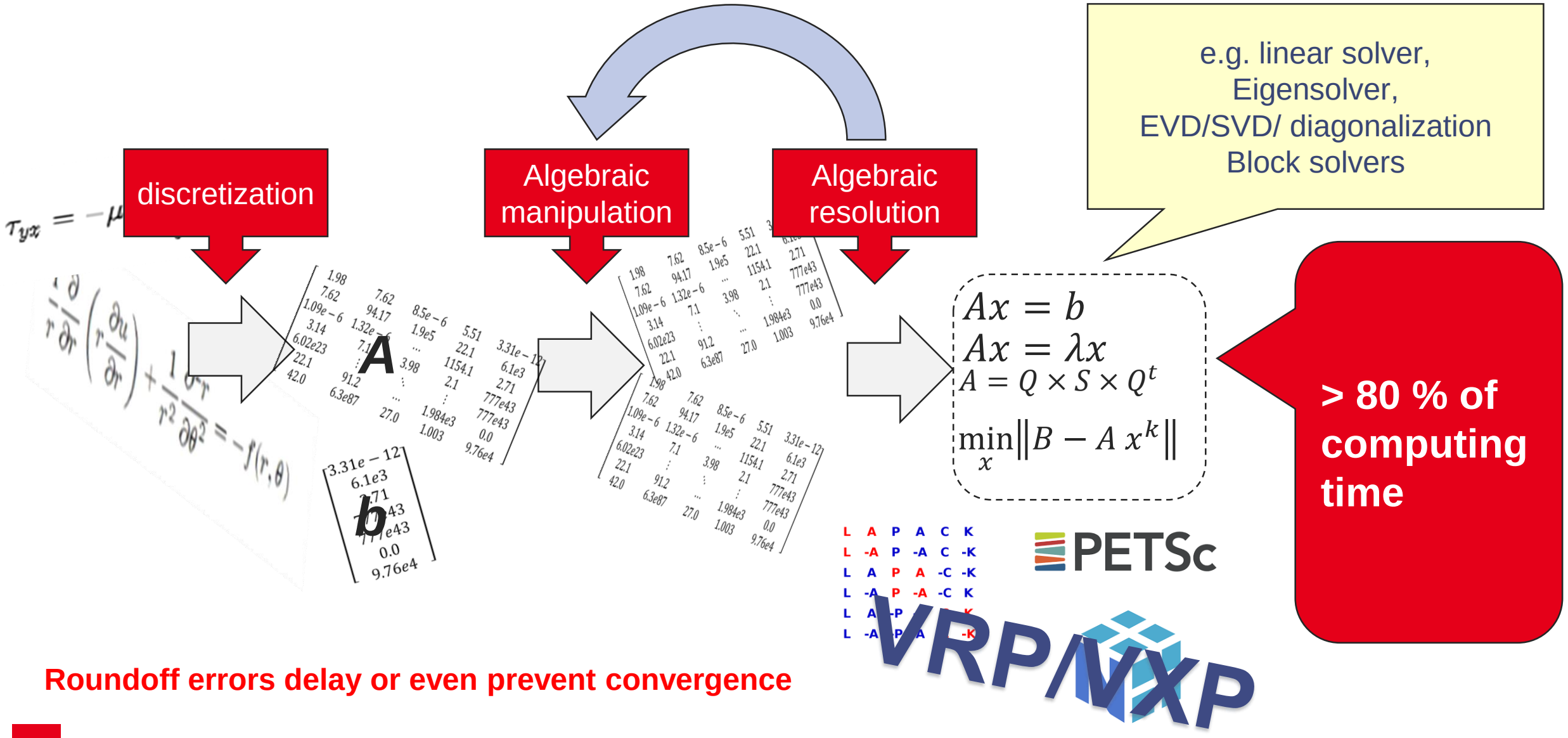


- More industrial topics such as electromagnetic, fluid dynamics.



- Some optimization algorithms

Focus on Linear Solvers



Roundoff errors delay or even prevent convergence

Why Extended Precision ?

- Radix-based floating point representation:

$$\left(\sum_{n=0}^{p-1} bit_n \times 2^{-n}\right) \times 2^{e}$$

k-bit exponent

p-bit mantissa

- VRP/VXP processor supports
  - p=*3*..*512* bits mantissa size (p=52 for double precision)
  - k=*2*..*18* bits exponent size (k=11 for double precision)
  - Specified at runtime (no recompilation)

- “1-bit LLMs Could Solve AI’s Energy Demands” ([1], IEEE Spectrum, 2024/05/30)

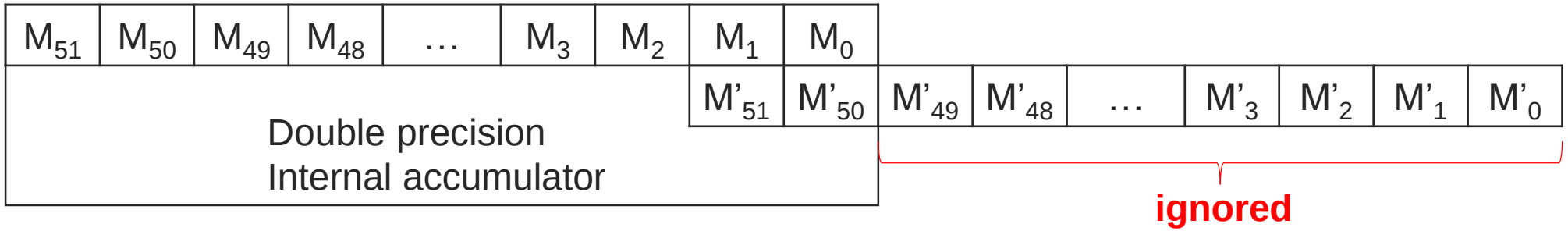
[1] https://spectrum.ieee.org/1-bit-llm

Motivation: Floating Point Issues

- Floating point arithmetic is done on fixed size hardware, which leads to rounding issues

- Example: alignment issues in floating point addition

$$\alpha = M \times 2^{E} \quad + \quad \beta = M' \times 2^{E-50}$$

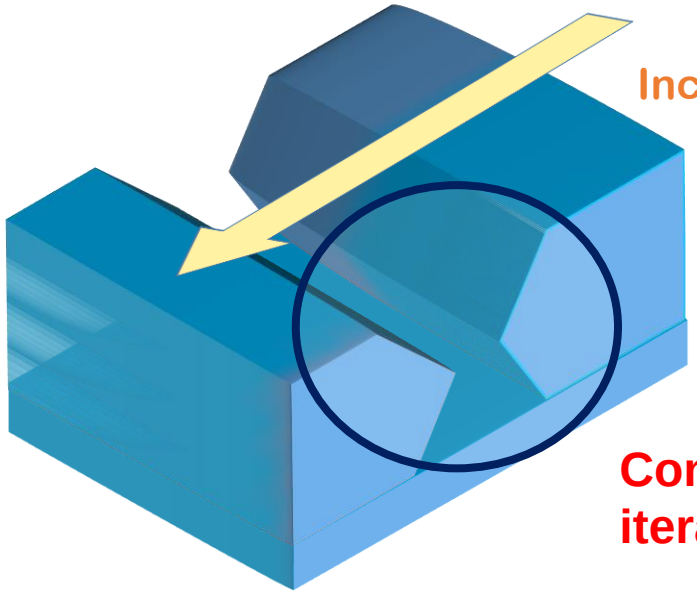


⇒ Floating point addition is not associative: $(\alpha + \beta) + \gamma \neq \alpha + (\beta + \gamma)$

⇒ Problematic for intensive computations, especially iterative methods applied on large problems

⇒ Extended precision can address these issues

Case Studies: Linear Solver on Fluidics



Incompressible flow

$$Ax = b$$

Conjugate gradient iterative solver

8K×8K matrix, **~10^{40}** condition number

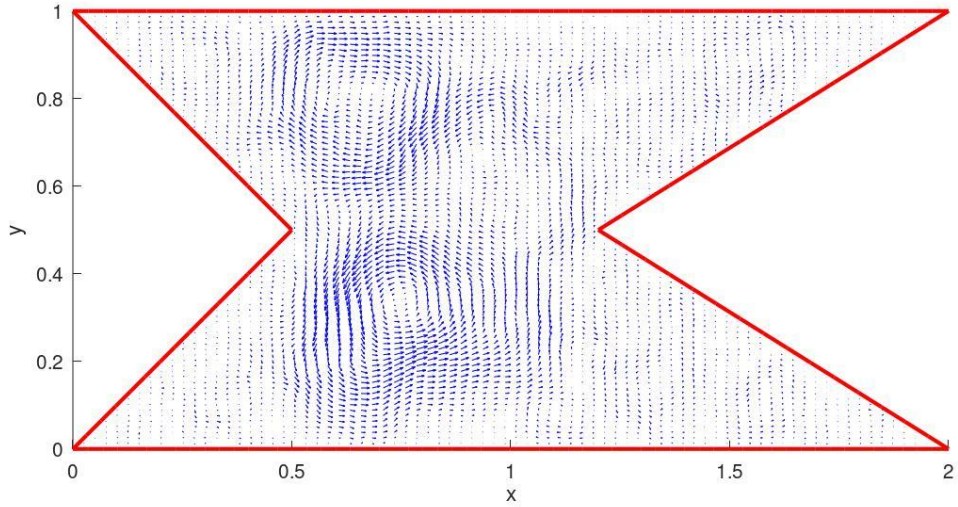
Comparison between:

1/ 64 bits precision with preconditioning (ref)

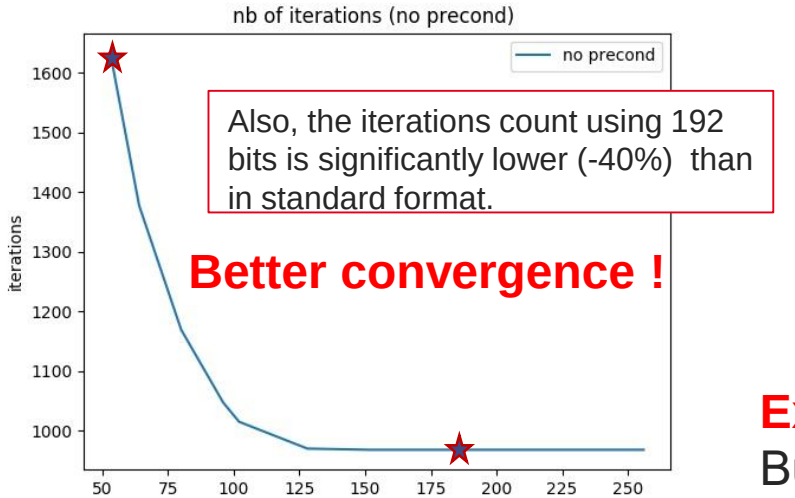
2/ 192 bits precision without preconditioning

Only intermediate vectors are stored in extended precision => *A* and *b* stay in 64b precision

Diff of velocity field @192b vs @64b



40% less iterations



Also, the iterations count using 192 bits is significantly lower (-40%) than in standard format.

Better convergence !

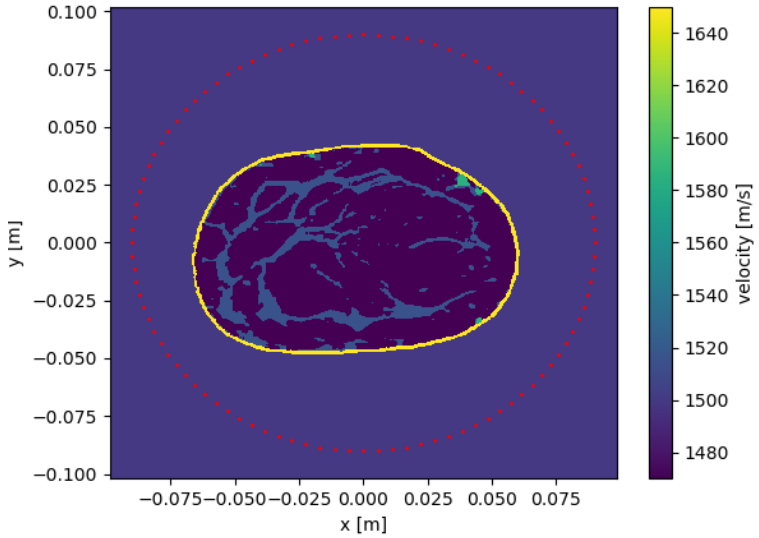
Precision (64->256)

Extended precision is not intended for more precise results

But we still get “better” results…

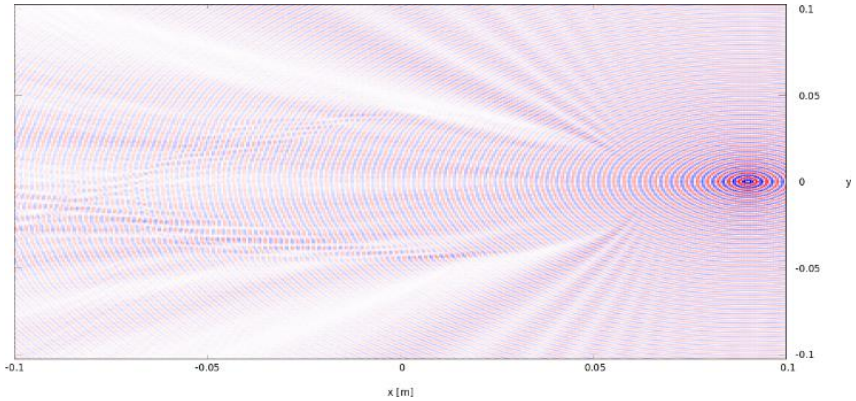
Case Studies: CG, BiCG, Block-BiCG Solvers

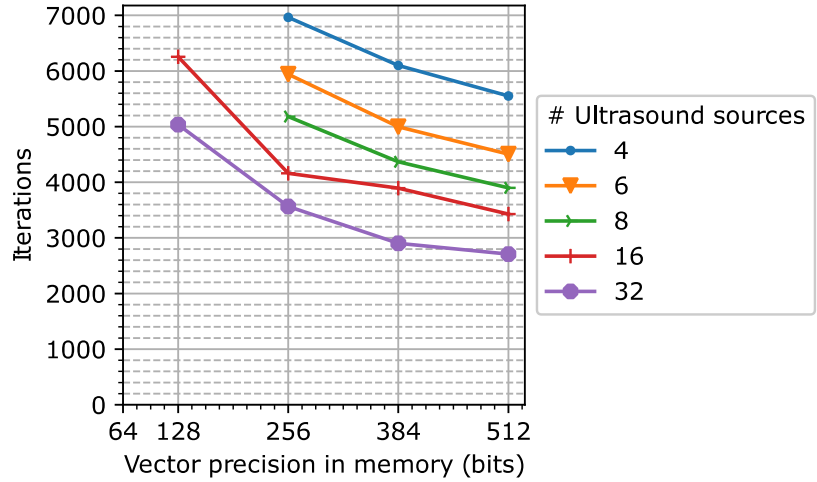
- Ultrasonic frequency-domain breast tomography (Block-BiCG) (SoA: time-domain) $AX = B$



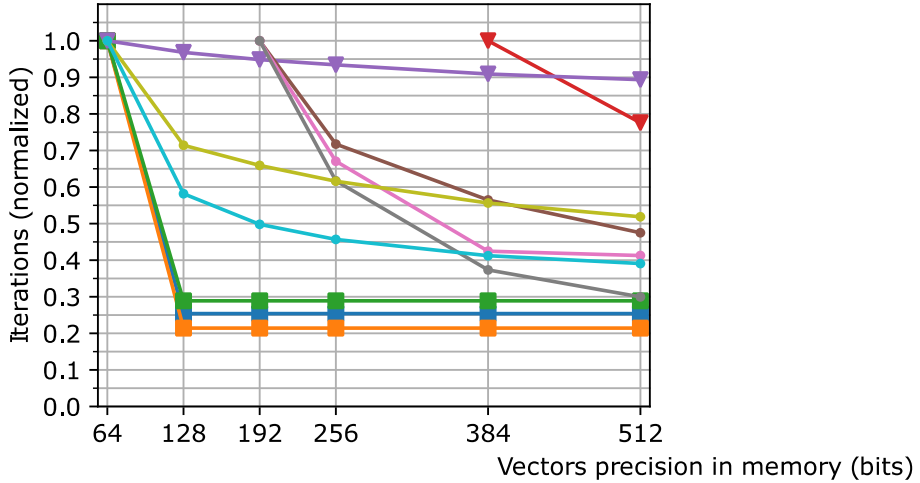
181k×181k matrix

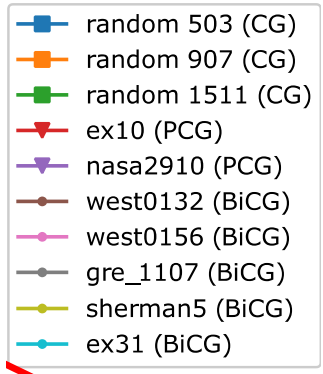
Source #1 solution





- SparseSuite matrices (benchmarking) solved on hardware prototype (SoA: complex preconditioners)

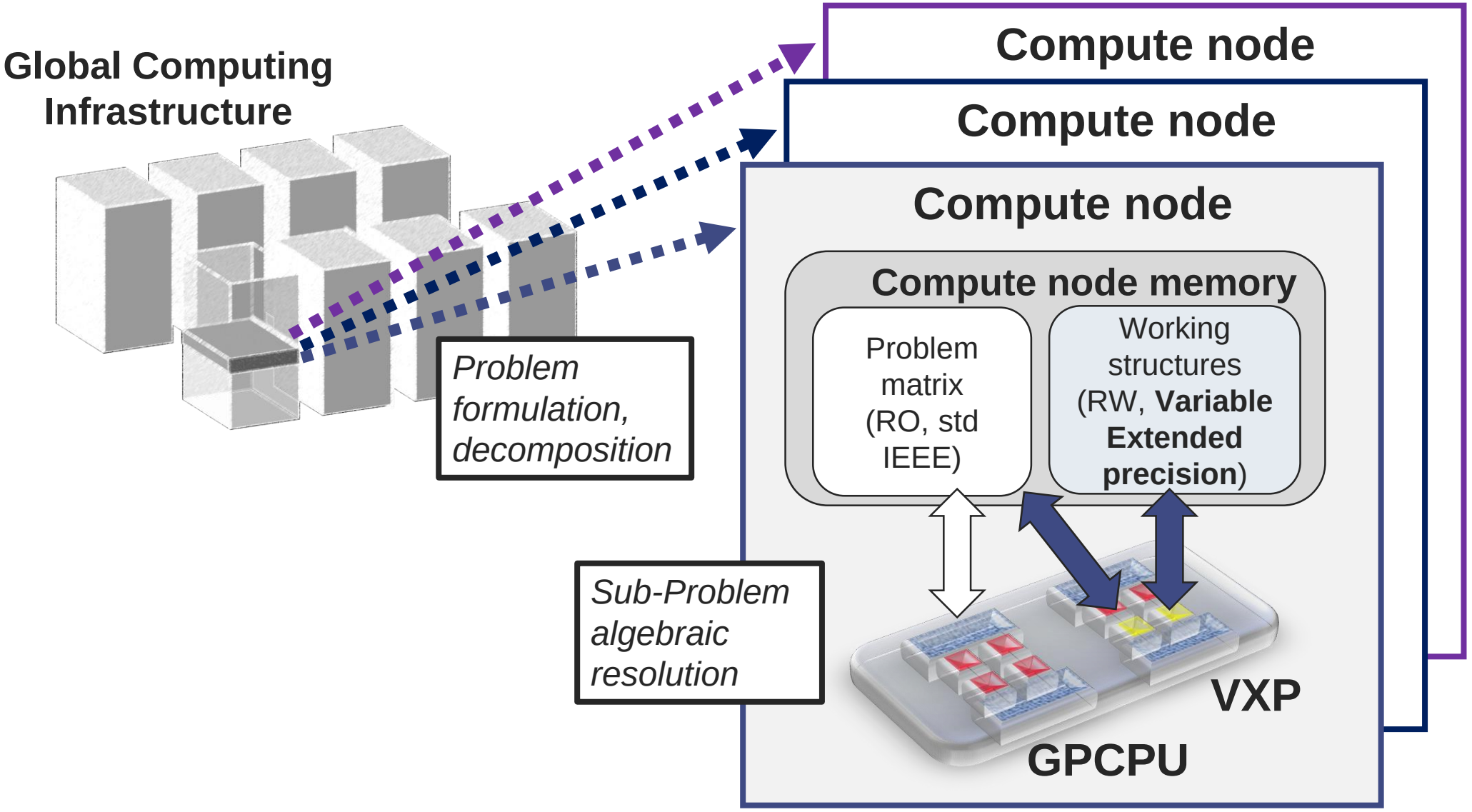




$Ax = b$

Precision sweet spots

VaRiable eXtended Precision Processor



Hardware Evolutions

Version	Features	Year	Technology
UNUM coprocessor [1]	RISC-V Rocket core, UNUM 256b FPU	2021	GF22FDX (EPI TC1.0)
VRP [2][3]	RISC-V CVA6, 512b FPU using IEEE extendable memory format, redesigned L1 cache	2022-2024	GF22FDX (EPI TC1.5), TSMC7 (Rhea)
VXP	RISC-V scalar OoO, 512b FPU with improved multiplier, streamer for sparse matrix-vector multiplication	*2026*	5-3nm (Rhea2)

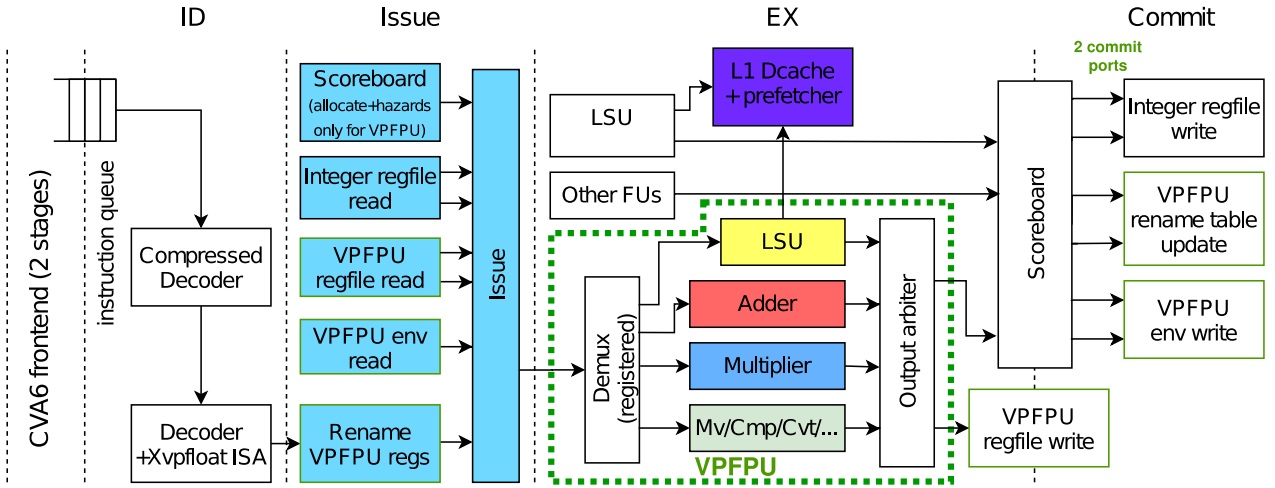
[1] A. Bocco, Y. Durand and F. de Dinechin, "Dynamic Precision Numerics Using a Variable-Precision UNUM Type I HW Coprocessor," *2019 IEEE 26th Symposium on Computer Arithmetic (ARITH)*, Kyoto, Japan, 2019, pp. 104-107, doi: 10.1109/ARITH.2019.00028.
[2] E. Guthmuller *et al.*, "Xvpfloat: RISC-V ISA Extension for Variable Extended Precision Floating Point Computation," in *IEEE Transactions on Computers*, vol. 73, no. 7, pp. 1683-1697, July 2024, doi: 10.1109/TC.2024.3383964.
[3] C. Fuguet et al., “A Variable and Extended Precision (VRP) Accelerator and its 22 nm SoC Implementation,” 2024 IEEE XXXIX Conference on Design of Circuits and Integrated Systems (DCIS), Catania, Italy (to be published)

VRP/VXP: RISC-V Accelerator for Variable Extended Precision Computing

Motivation: Software emulation (e.g. MPFR) too slow

Goal: be application-agnostic and limited by memory bandwidth instead of arithmetic

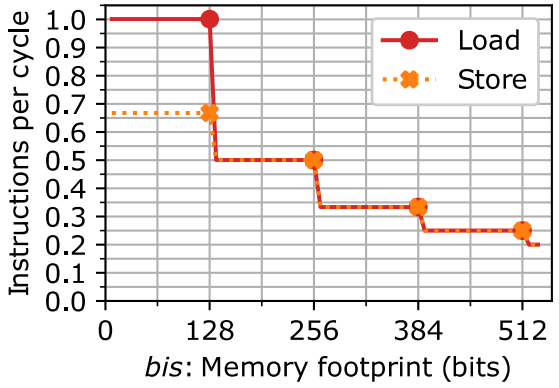
⇒ Variable extended Precision Floating Point Unit (VPFPU) integration in modified RISC-V CVA6 processor

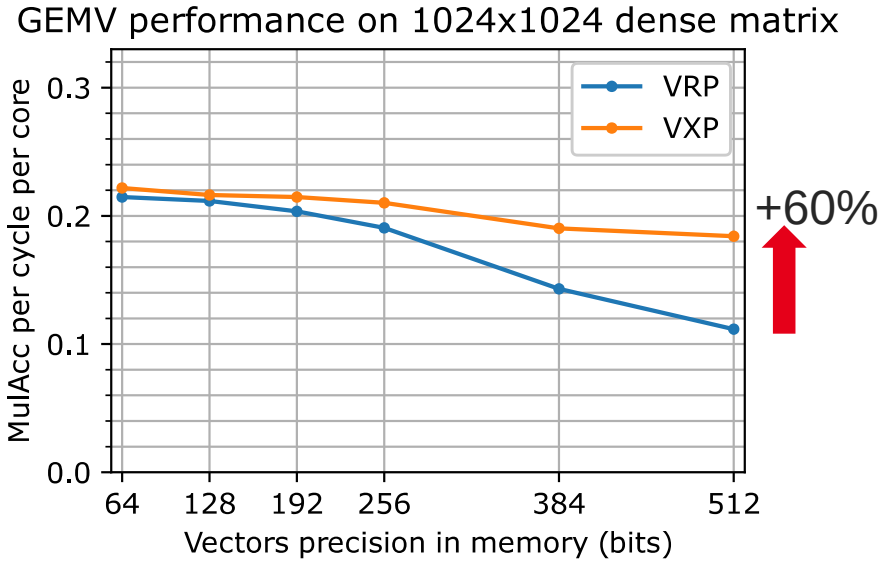


Main CVA6 modifications:

- 32 logical/64 physical 540-bit registers
- Register renaming (improved in VXP)
- OoO execution (new in VXP)
- New high-perf L1 cache (open sourced)
- 7 VPFPU functional units
  - Working iteratively on 64/128b chunks

  ⇒ Performance depends on precision





Xvpfloat RISC-V ISA Extension

- Opensource ISA extension for dynamically variable precision computation

⇒ **No recompilation**

- Instructions parametrized with in-memory and output precision

```
PMUL pt, pa, pb, eci          pt=pa*pb;
```

Env register includes:
- Output precision (in bits)
- Rounding mode

- Indexed load/store instructions

```
PLE pt, evpi, ra{,#index}     vpfloat<evpi> *tab=ra; pt=tab[index];
```

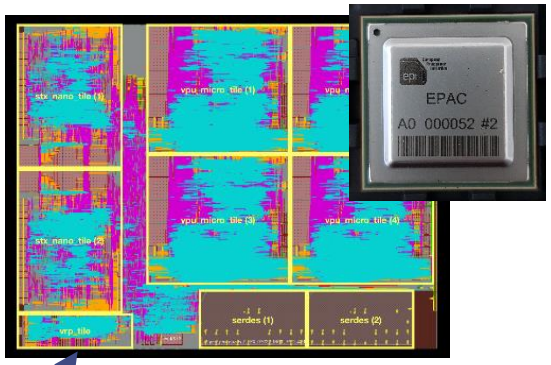
- In-memory IEEE-754 2008 extendable format

Env register includes:
- In-memory size (in bits)
- Exponent size (in bits)
- Rounding mode
- Stride

- \+ conversions, arithmetic (+,-,*), load/store, comparison

- Compiler WIP (Binutils ready, prototype LLVM+Clang with *vpfloat* C type)

Hardware Prototypes



EPACTC 1.0

VRP core GF22FDX

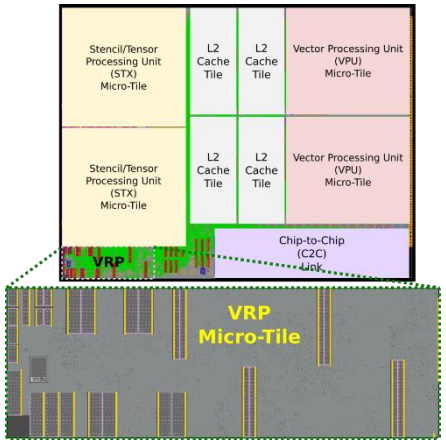
Q2 2021

VRP core (x2)

Dual-core VRP tile

AMD FPGA

Q3 2021



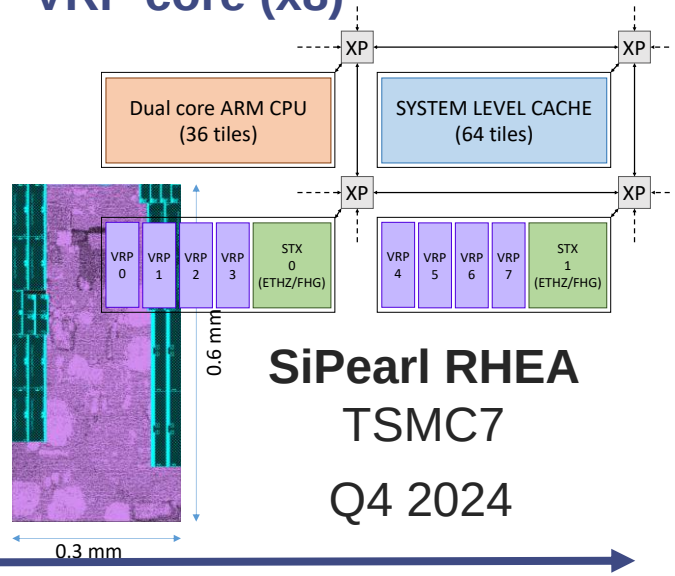
EPACTC 1.5

VRP core GF22FDX

Q3 2022

Jupiter HPC

VRP core (x8)



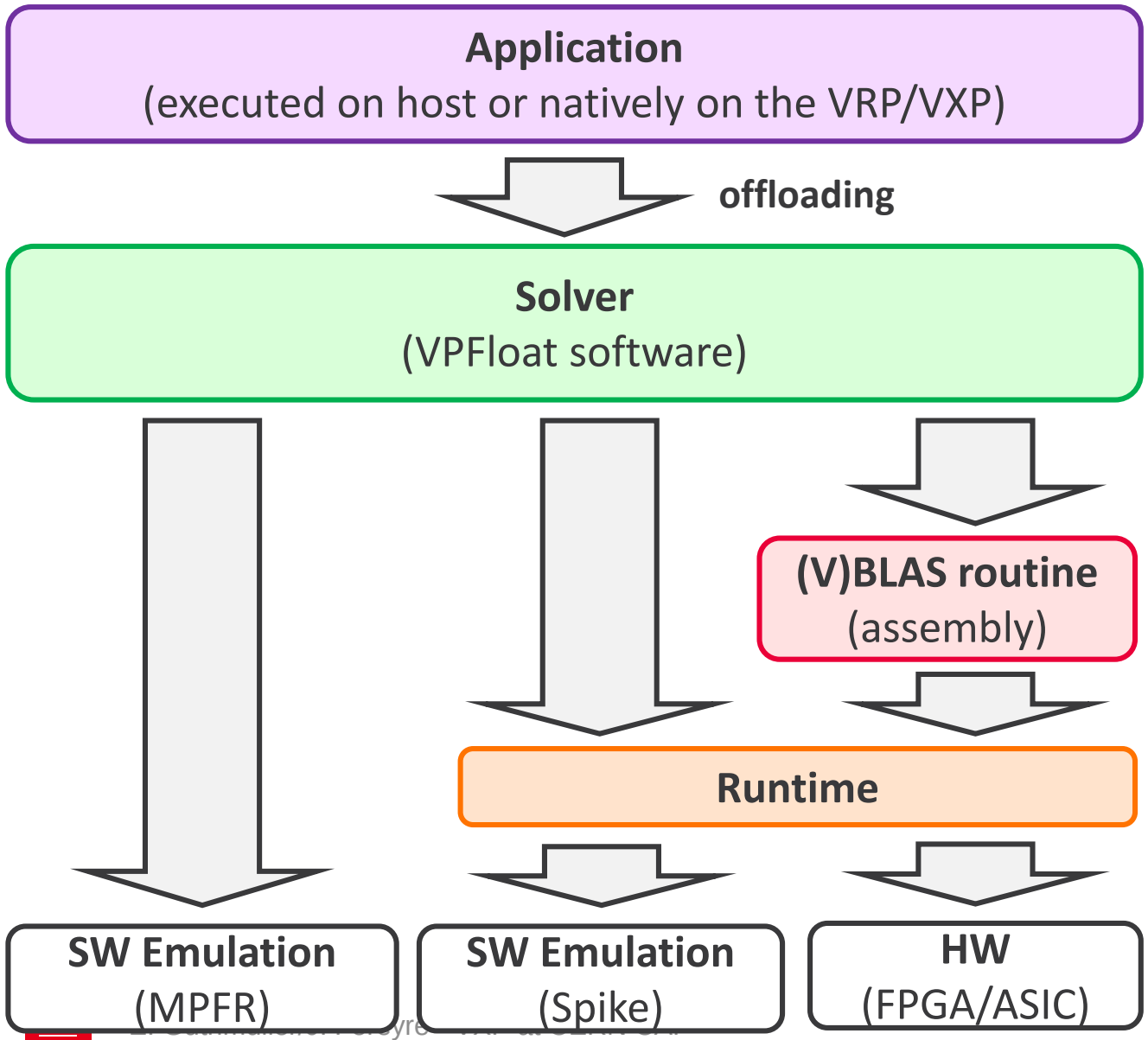
SiPearl RHEA

TSMC7

Q4 2024

FPGA prototype	EPACTC 1.5 (VRP features)	RHEA (VRP features)
2 VRP/VXP cores	1 VRP core	Two tiles with 4 VRP cores each
Clock frequency: 83 MHz	Clock frequency: 1 GHz	Clock frequency: 1.2 GHz

Software Stack – Overview

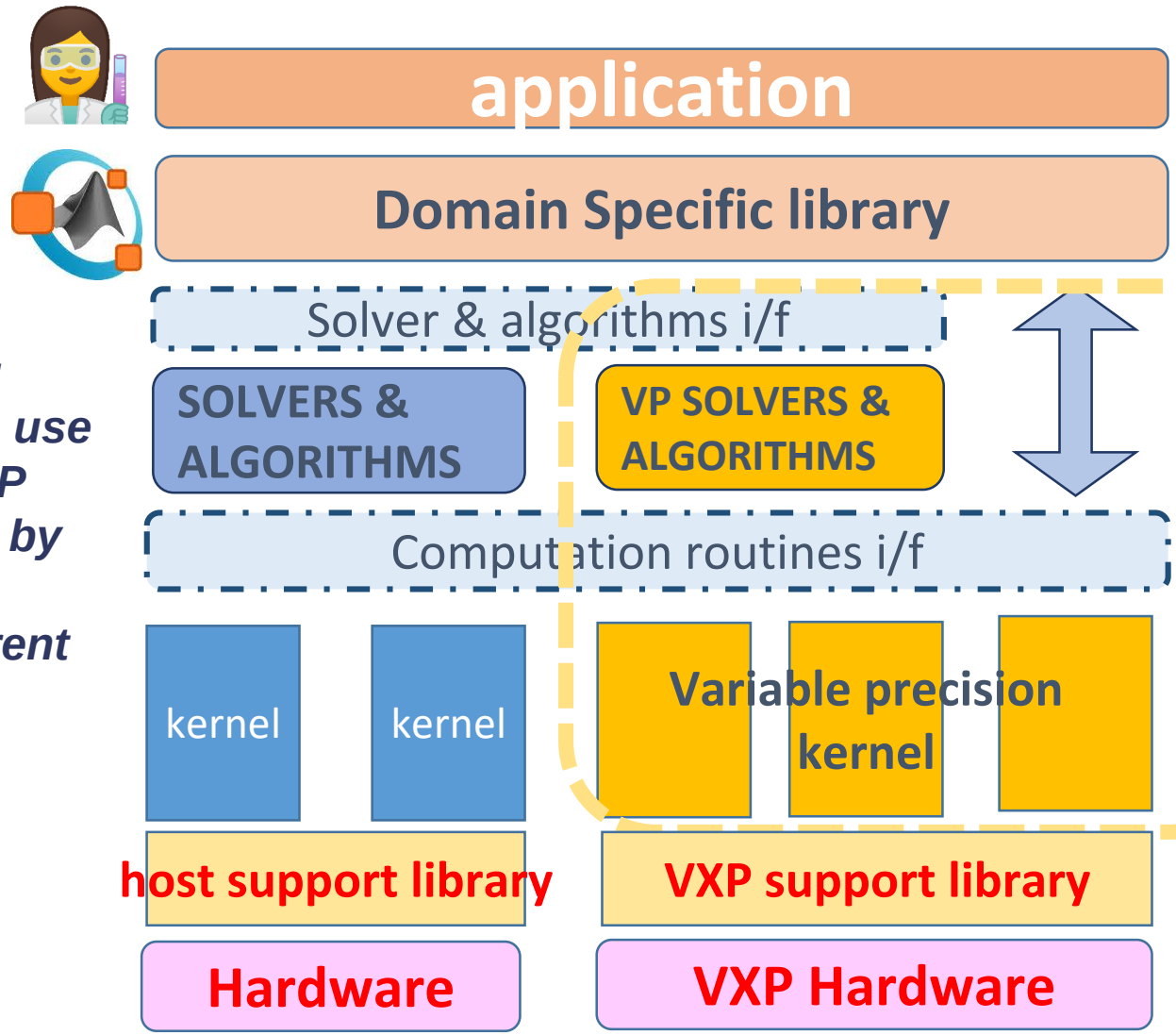



```
VPFloatArray X(EXP_SZ, FRAC_SZ, Ndiag);
…
Nbiter = cg_vp(precision, Ndiag, X, A, B, tol);
```

```
// A*x = b
int cg_vp(int precision, int Ndiag,
          VPFloatArray & x, double *A,
          VPFloatArray b, double tolerance) {
    …
    while (relerror < tolerance) {
        …
        VBLAS::vgemv(precision, n, n, A, p_k, Ap_k, …)
```

```
// y = A*x
void VBLAS::vgemv(int precision, …){
  pser_ec(precision, EC0);          // compute precision
  pser_evp(X.es(), X.fs(), EVP1); // memory precision
  for (int i = 0; i < m; i++) {
    pcvt_d_p(P24, 0);               // acc = 0
    for (int j = 0; j < n; j++) {
      ple(P0, a_ptr, 0, EVP0); // A(m)(n)
      ple(P1, x_ptr, 0, EVP1); // x(n)
      pmul(P0, P0, P1, EC0);   // A(m)(n)*x(n)
      padd(P24, P24, P0, EC0); // acc += A(m)(n)*x(n)
```

SW – VP and Legacy Software



Library based approach:
Application developpers use compatible alternative VP solvers when supported by the application
Developpers use a different type in C

Variable precision is contained within calls to kernel (BLAS level) and Solver (LaPack level) calls

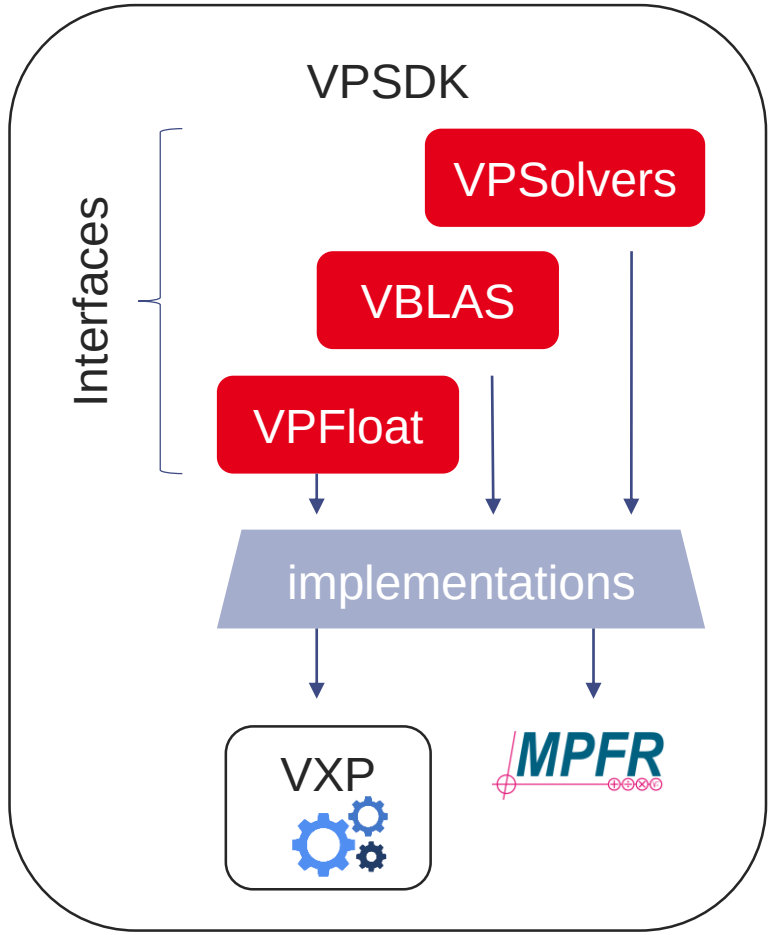
SW – Variable Precision SDK

VPSDK provides:

- A set of primitives to develop kernels using extended variable precision
  - Scalar operations +,-,*,/, sqr
  - BLAS operations: gemv, scal, axpy, dot, nrm2, copy, zero
- A set of kernels are available and ready to run :
  - BiCG, BiCGSTAB, CG, PCG, QMR

VPSDK supports two use cases:

- Create new applications for non accelerated platform to test variable precision benefits
- Call pre-compiled optimized kernels for VXP on different architectures



SW – VXP Platforms Supported

- Dedicated device memory architectures
  - Use PCIe bus to exchange between host and VXP
  - FPGA VXP implementation

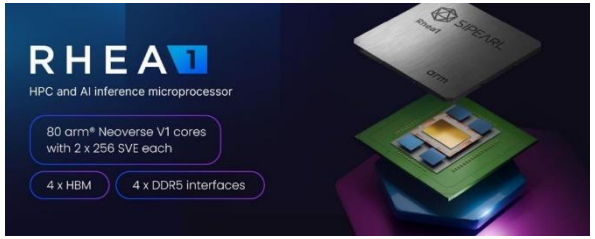
    ⇒ AMD/Xillinx FPGA (VCU128, Versal VCK190)

- Shared memory architectures
  - Use of IOMMU to exchange between host and VXP
  - VXP integration within General Purpose Processor

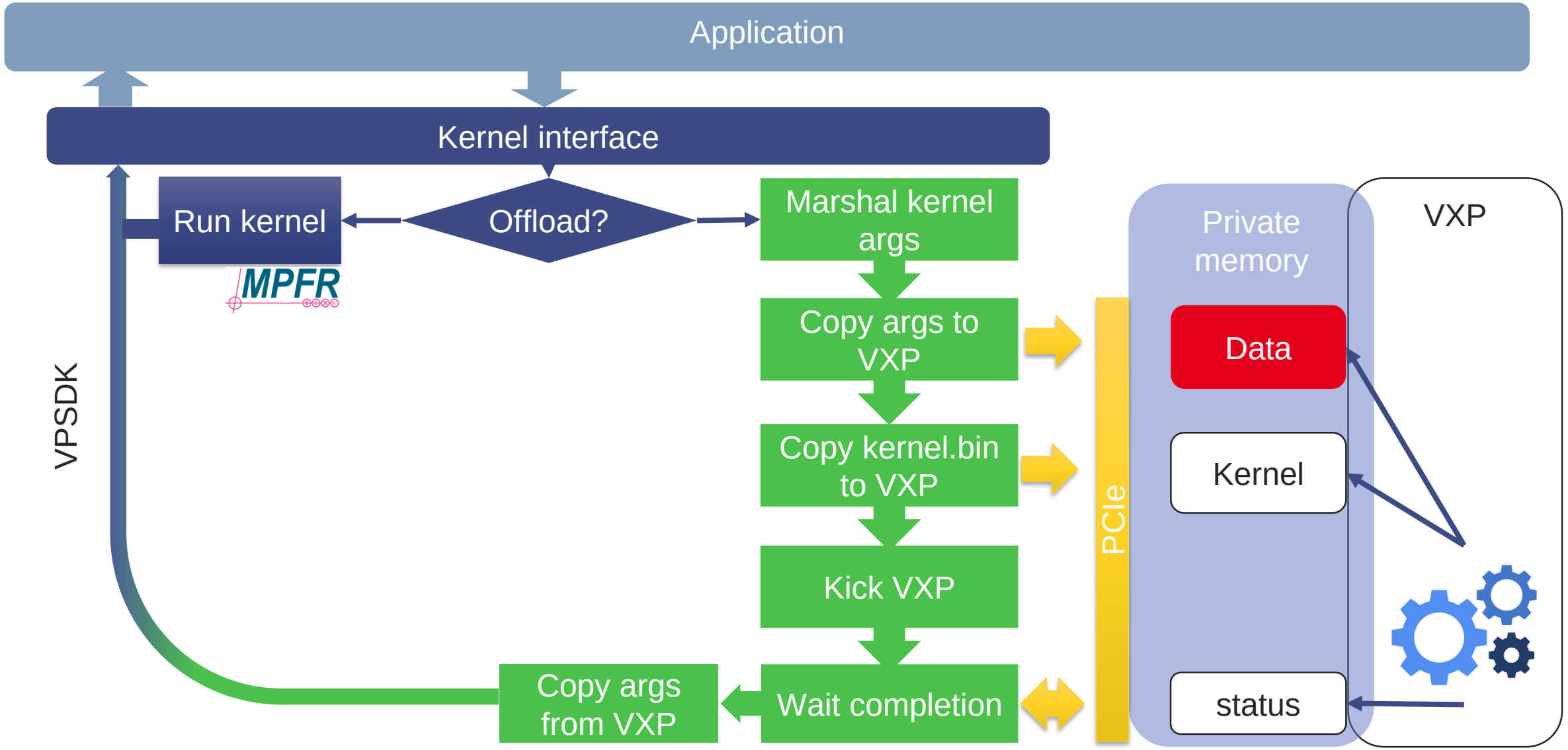
    ⇒ Sipearl RHEA1 processor

    ⇒ Integrated in JUPITER Exascale supercomputer JSC

SIPEARL



SW – VXP Kernel Offloading Through PCIe

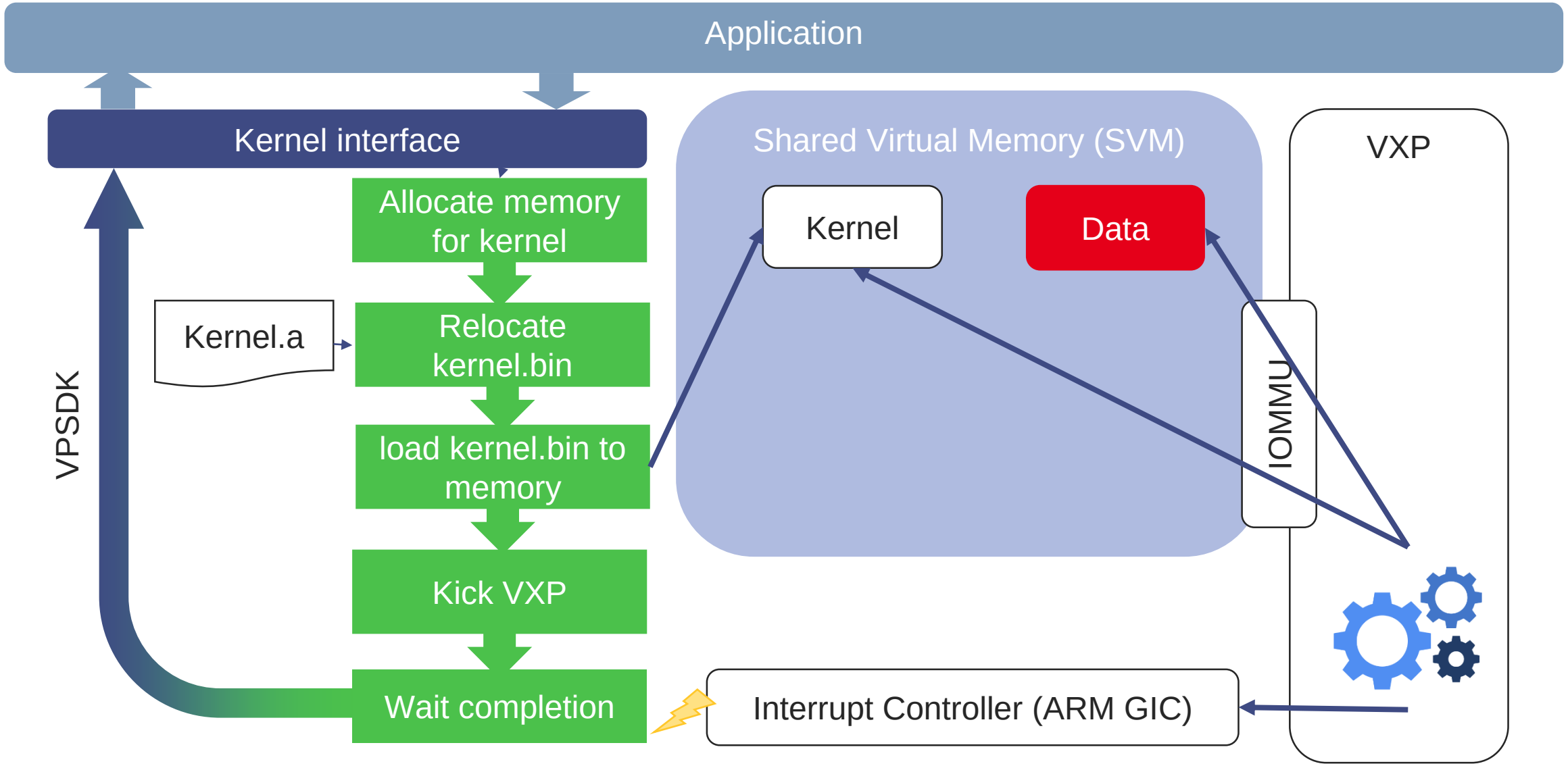


SW – VXP Kernel Offloading Through PCIe

Software components :

- VPSDK compiled for Linux or RISC-V bare metal platform
  - Run solvers either on host with MPFR or on VXP with native acceleration
- Dedicated VXP Linux driver
  - Used to transfer data and kernels to VXP memory and manage VXP run
  - Built on top of Xilinx XDMA DMA/PCI driver
- RISCV cross compiler to generate kernel RISCV binaries
  - Currently we only have binutils (GAS)

SW – VXP Kernel Offloading Using SVM



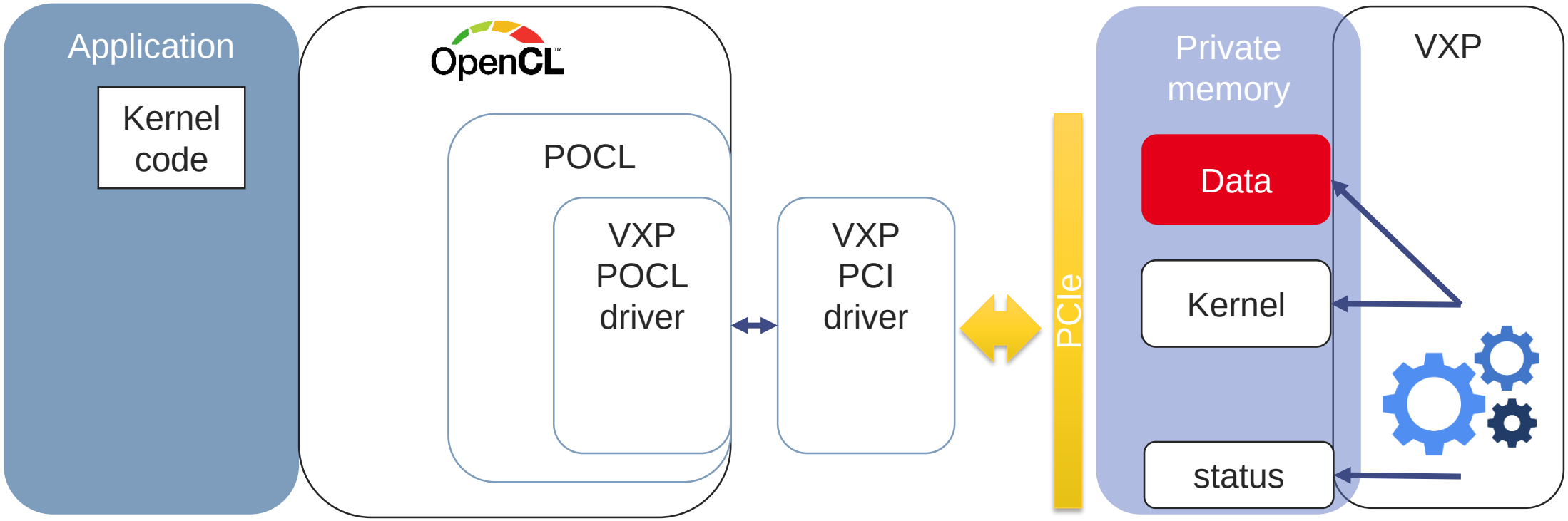
SW – VXP Kernel Offloading Using SVM

Software components :

- VPSDK & compiler same as previously

- User space driver used to map data and kernels in VXP virtual memory, and manage VXP execution
  - Built on top VFIO Linux driver to deal with IOMMU virtual memory mapping
  - Zero-copy offloading scheme

SW – Current Work : Dynamic Kernel Offloading



- Kernel compilation done at runtime as kernel arguments discovery
- Friendly integration with our Linux driver interface for kernel and data transfers

SW – Future Works

- Have a full compiler toolchain:
  - Stop writing assembly code for optimized kernels
  - Scalar operation will be more efficient (applications are not just about calls to BLAS)
  - Needed to provide VXP to end users

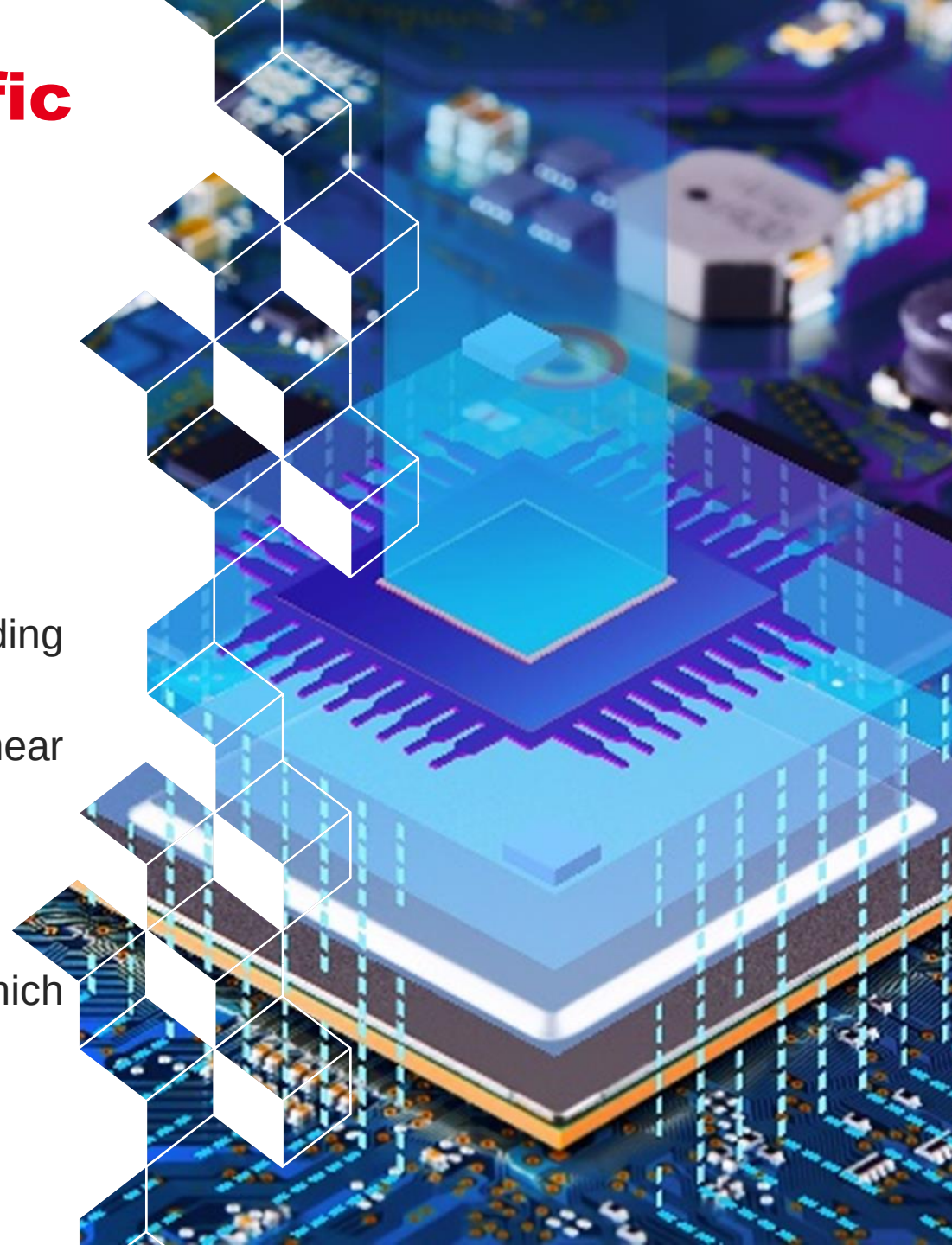
- Support OpenMP offloading
  - No more specialized code for kernels
  - One application code can run on all platforms

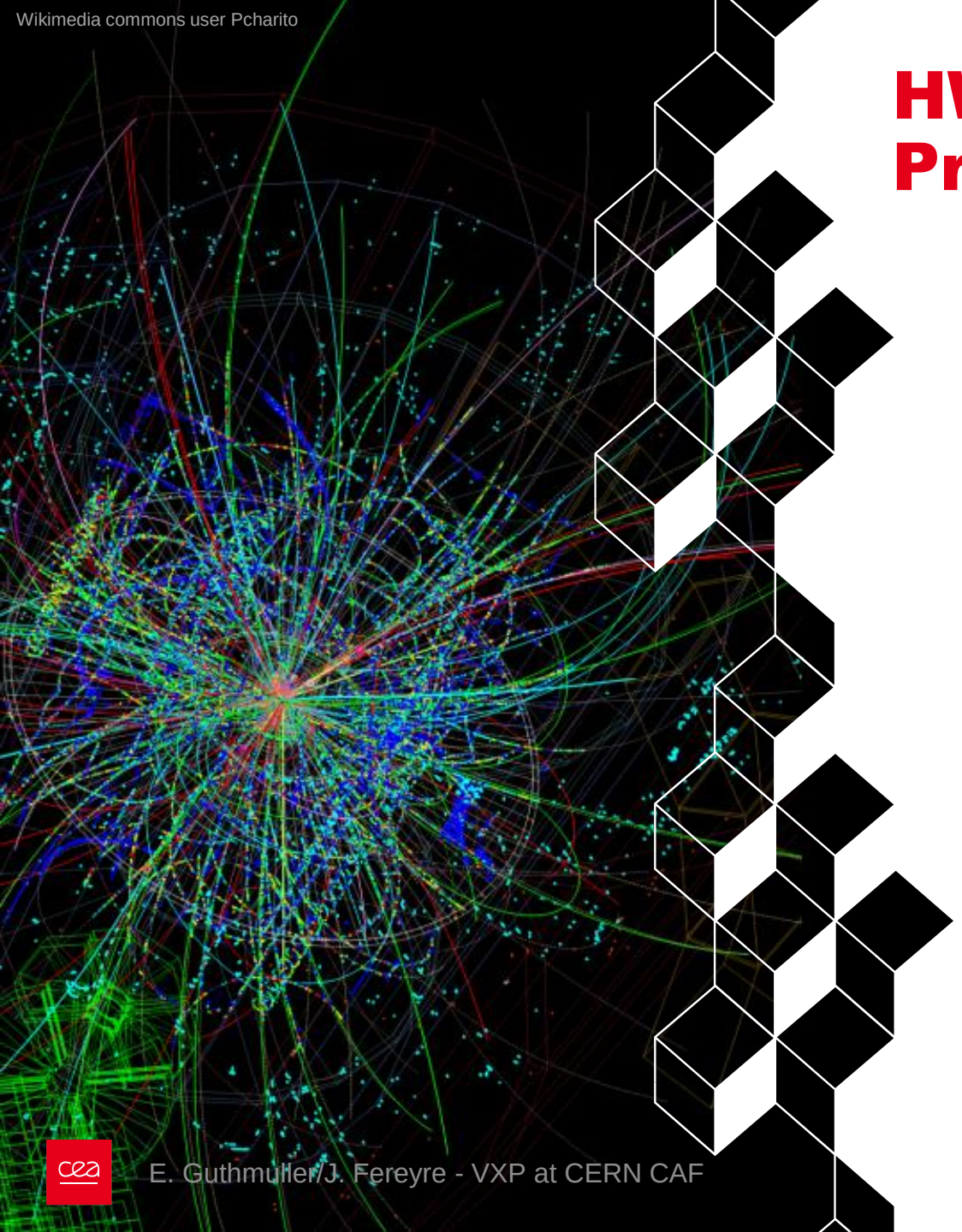
Extended Precision for Scientific Applications – Takeaways

- What we have already shown :
  - Improved convergence of iterative linear solvers
  - It is useful for other kernels
  - It reduces energy-to-solution in some cases

- We believe that extended precision
  - Reduces energy-to-solution in the general case, by avoiding costly preconditioning techniques
  - Reduces time-to-solution by improving reliability of linear solvers and eigensolvers

- Open questions
  - How to choose the right precision beforehand ? Which granularity (today bit-by-bit) ?
  - Impact of higher precision of result ?




Wikimedia commons user Pcharito

HW Implementation of Extended Precision – Takeways

- FPU implementation is “solved”
  - Iterative chunk-based design, performance depends on precision
  - Low arithmetic intensity of targeted applications => low area overhead of arithmetic units
- ISA & CPU integration demonstrated
  - Either wide OoO window or highly SMT system is needed to account for high latency instructions (10-20 cycles in our implementation)
- The challenge lies in the memory subsystem
  - High bandwidth caches, interconnects & memory
  - Streaming accelerators for sparse datasets

Wrapping Up!

Today we presented our activity on extended precision for scientific computing

→ our team has many other activities, including these ones relevant for CERN CAF:

- 128-bit RISC-V Extensions
  - Directly address all the memory in a full super-computer
- Heterogeneous CPU + GPU integration
  - Using open-source hardware (RISC-V, OpenPiton, Vortex)
  - Study relaxed coherency protocols for CPUs/GPUs
  - Collaboration with UCSB, GeorgiaTech
- Open to collaborations !

cea list

CERN Compute Accelerator Forum

THANKS FOR YOUR ATTENTION

INSTITUT CARNOT CEA LIST

université PARIS-SACLAY

UGA Université Grenoble Alpes

Publications

E. Guthmuller *et al*., "Xvpfloat: RISC-V ISA Extension for Variable Extended Precision Floating Point Computation," in *IEEE Transactions on Computers*, doi: 10.1109/TC.2024.3383964.

C. Fuguet et al., “A Variable and Extended Precision (VRP) Accelerator and its 22 nm SoC Implementation,” 2024 IEEE XXXIX Conference on Design of Circuits and Integrated Systems (DCIS), Catania, Italy (to be published)

Hoffmann, Alexandre & Durand, Yves & Fereyre, Jérôme. "Accelerating Spectral Elements Method with Extended Precision: A Case Study." 2024, *International Journal of Applied Physics and Mathematics*. 14. 45-58. 10.17706/ijapm.2024.14.2.45-58.

Y. Durand, E. Guthmuller, C. Fuguet, J. Fereyre, A. Bocco and R. Alidori, "Accelerating Variants of the Conjugate Gradient with the Variable Precision Processor," *2022 IEEE 29th Symposium on Computer Arithmetic (ARITH)*, Lyon, France, 2022, pp. 51-57, doi: 10.1109/ARITH54963.2022.00017.

T. T. Jost, Y. Durand, C. Fabre, A. Cohen and F. Pérrot, "Seamless Compiler Integration of Variable Precision Floating-Point Arithmetic," *2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, Seoul, Korea (South), 2021, pp. 65-76, doi: 10.1109/CGO51591.2021.9370331.

Tiago Jost, Yves Durand, Christian Fabre, Albert Cohen, and Frédéric Pétrot. "VP Float: First Class Treatment for Variable Precision Floating Point Arithmetic." 2020, in *Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques (PACT '20)*. Association for Computing Machinery, New York, NY, USA, 355–356. https://doi.org/10.1145/3410463.3414660

A. Bocco, Y. Durand and F. de Dinechin, "Dynamic Precision Numerics Using a Variable-Precision UNUM Type I HW Coprocessor," *2019 IEEE 26th Symposium on Computer Arithmetic (ARITH)*, Kyoto, Japan, 2019, pp. 104-107, doi: 10.1109/ARITH.2019.00028.

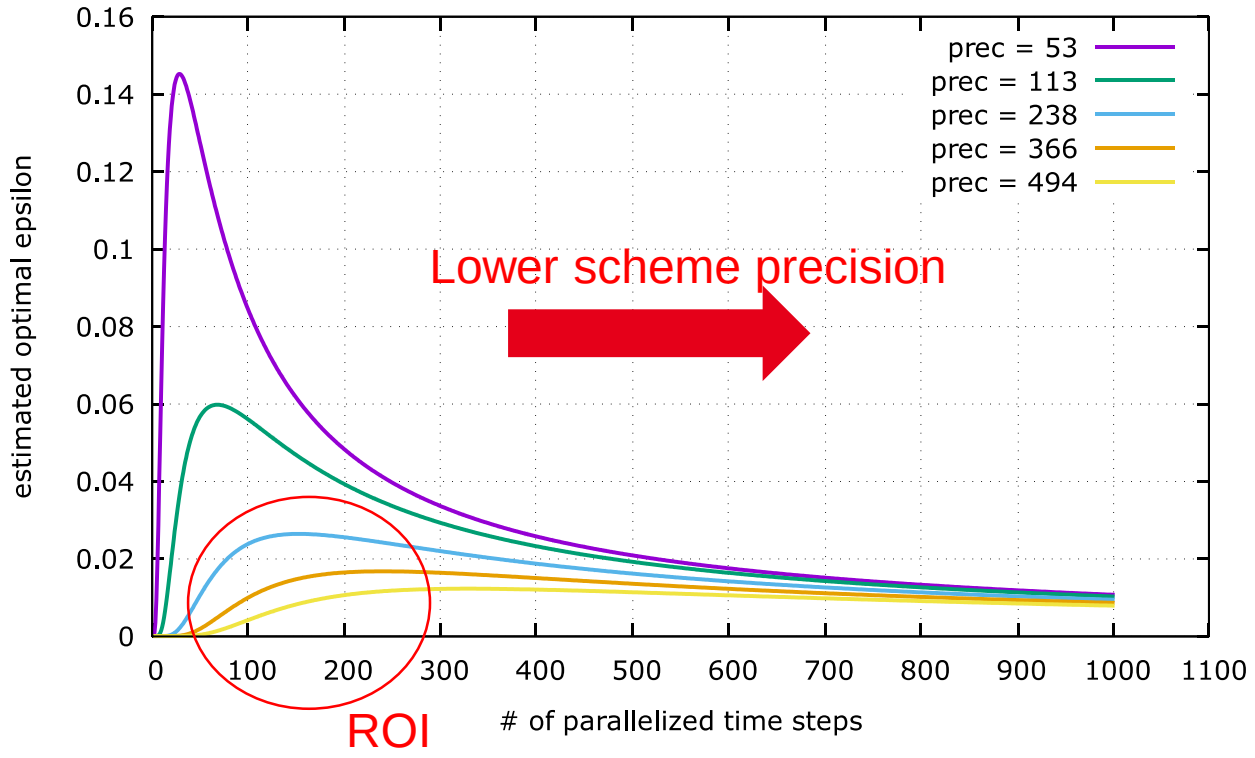
A. Bocco, T. T. Jost, A. Cohen, F. de Dinechin, Y. Durand and C. Fabre, "Byte-Aware Floating-point Operations through a UNUM Computing Unit," *2019 IFIP/IEEE 27th International Conference on Very Large Scale Integration (VLSI-SoC)*, Cuzco, Peru, 2019, pp. 323-328, doi: 10.1109/VLSI-SoC.2019.8920387.

Application results: Time-domain parallelization

- By transforming the initial problem, it is possible, under certain conditions, to solve multiple time-steps in parallel **=> useful for parallelization at cluster level**

$$M\frac{d^2u^h}{dt^2}(t) + Ku^h(t) = s^h(t) \in \mathbb{R}^n \quad \forall t \in (0,T)$$

- Ultrasonic frequency-domain breast tomography (Block-BiCG) (SoA: time-domain)

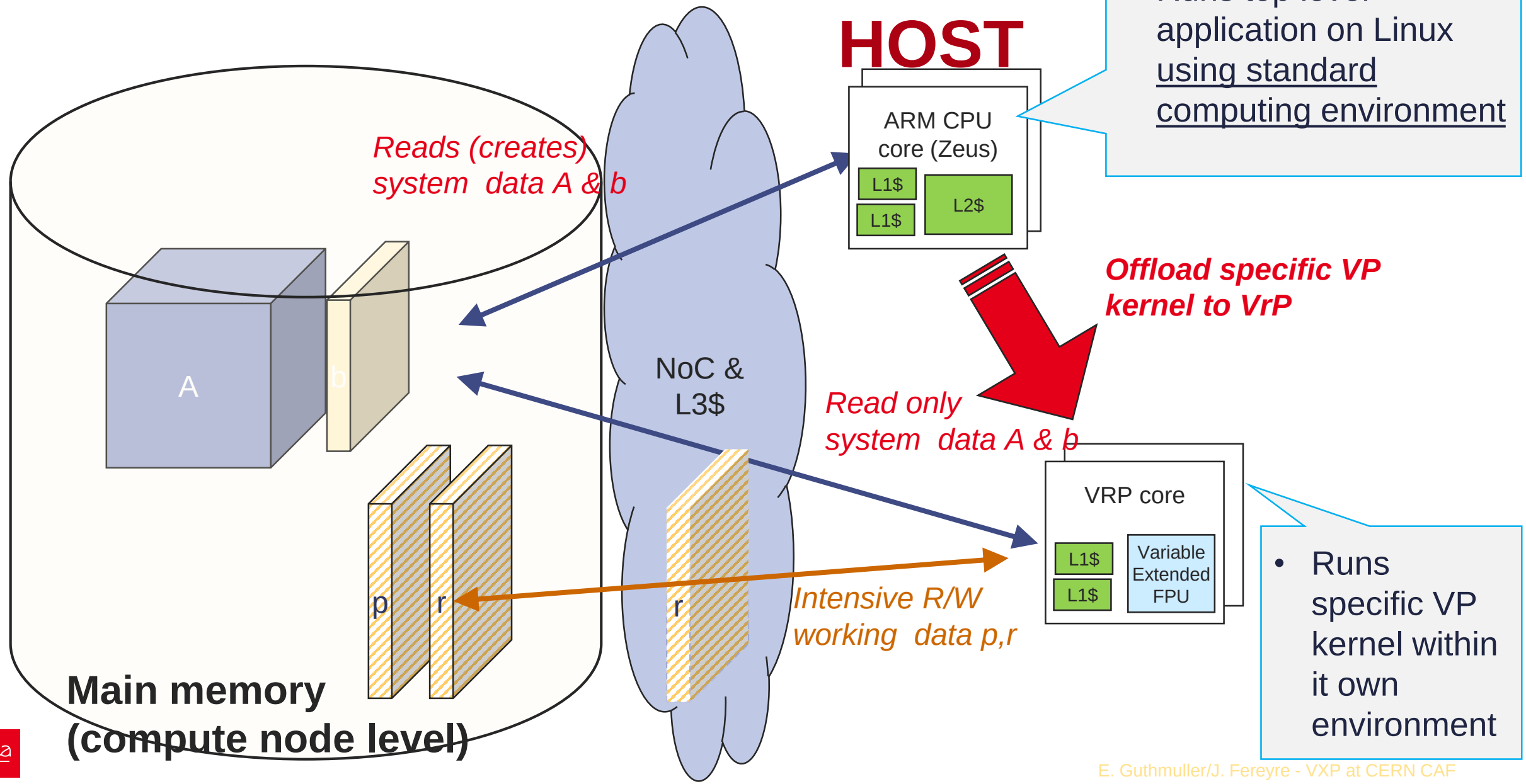


Case studies: Eigen solvers

- Eigen solvers are plagued by orthogonality issues while building the eigen-vectors basis
  - “Restarts” needed
  - Final result may still be wrong (reported by CEA DRF people working on quantum devices simulation)

- Increasing working precision improves orthogonality
  - Estimated reduction in number of operations proportional to number of eigenvalues being searched for

VRP OFFLOADING SCHEME



- Runs top level application on Linux using standard computing environment

- Runs specific VP kernel within it own environment