ENHANCING SCIENTIFIC COMPUTATION USING A VARIABLE PRECISION FPU WITH A RISC-V PROCESSOR

Y.Durand, C.Fabre, A. Bocco, T. Trevisan | IMPRENUM Project | Oct 2019

USE CASES FOR (LARGE) VARIABLE PRECISION

Applications

- **Computational Physics**
- **Computational chemistry**
- **Computational statistics**
- **Computational geometry**
- **Large PDEs**
  - Finite elements, finite differences
- **ODE s**
- **optimization**

Techniques & Kernels

- **Dense/sparse linear algebra**
  - Solvers, eigenvalues
- **Numerical integration**
  - RK, but not only…
- **Monte Carlo**
- **Spectral techniques**
  - FFT and others
- **Interval arithmetics**

Our main focus today: ***linear algebra solvers***

However, there are many other area in scientific computing where variable precision is sought

VARIABLE PRECISION FOR SCIENTIFIC COMPUTATION
JACOBI

While error > tolerance

augment precision

while convergence not reached **do**

for i := 1:n **do**

$\sigma = 0$

for j := 1:n do

if j ≠ i then

$\sigma \mathrel{+}= a_{ij} x_j^{(k)}$

end

end

$x_i^{(k+1)} = \frac{1}{a_{ii}}(b_i - \sigma)$

end

$k=k+1$

end

end

Accumulation : **Requires max precision** should be done inside the FPU

Matrix coeffs: read-only, sparse doubles
Stay in remote memory

Vector update :

- dense
- Requires high precision
- should be kept in close memory

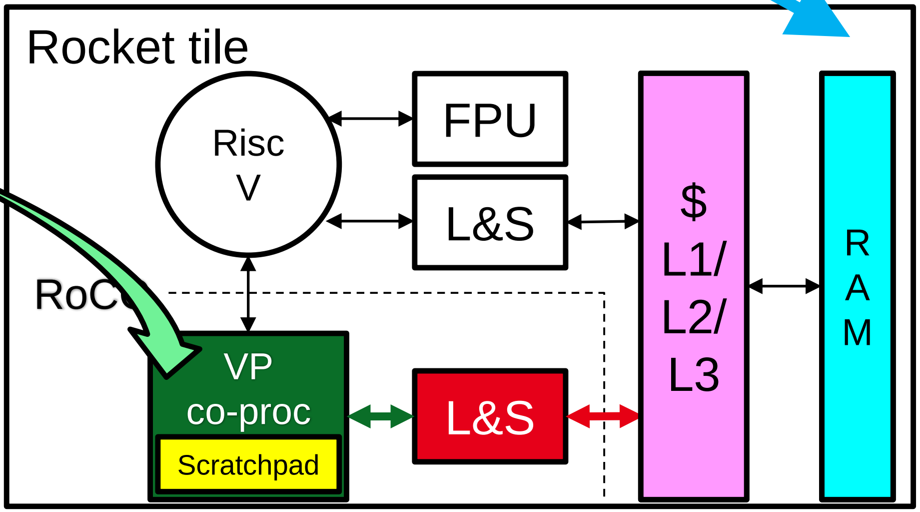
we need

1. ***extended precision operators,***
2. ***dedicated accumulators in registers inside the FPU,***
3. ***Extended precision storage in close memory***

MORE IN DEPTH WITH JACOBI : EXECUTING ON THE V1 ACCELERATOR

Input data, RO, in RAM, double format (sparse)

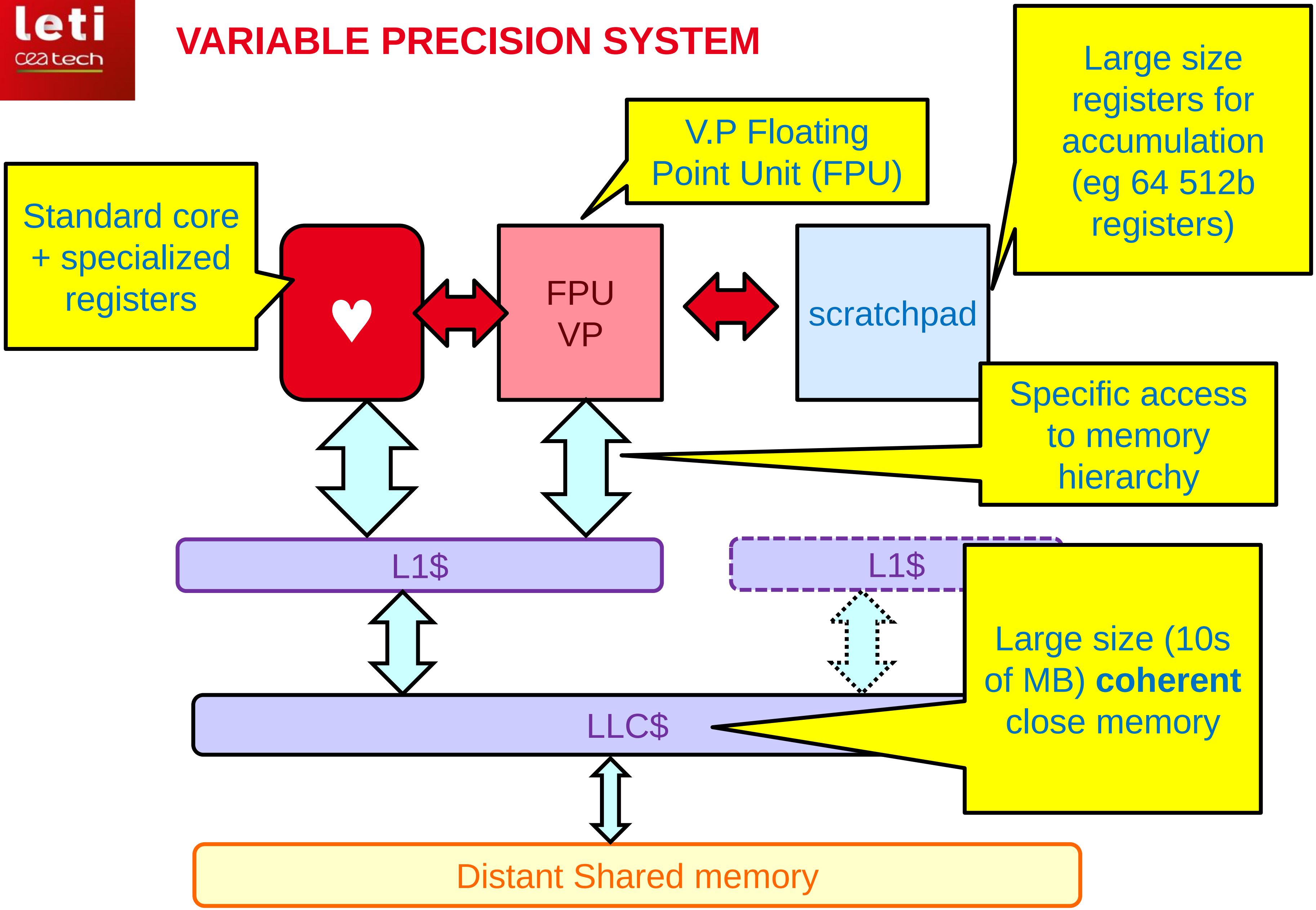
k = 0
while convergence not reached **do**
  for i = 1:n **do**
    $\sigma = 0$
    for j = 1:n do
      if j ≠ i then
        $\sigma \mathrel{+}= a_{ij} x_j^{(k)}$
      end
    end
    $x_i^{(k+1)} = \frac{1}{a_{ii}}(b_i - \sigma)$
  end
  k=k+1
end



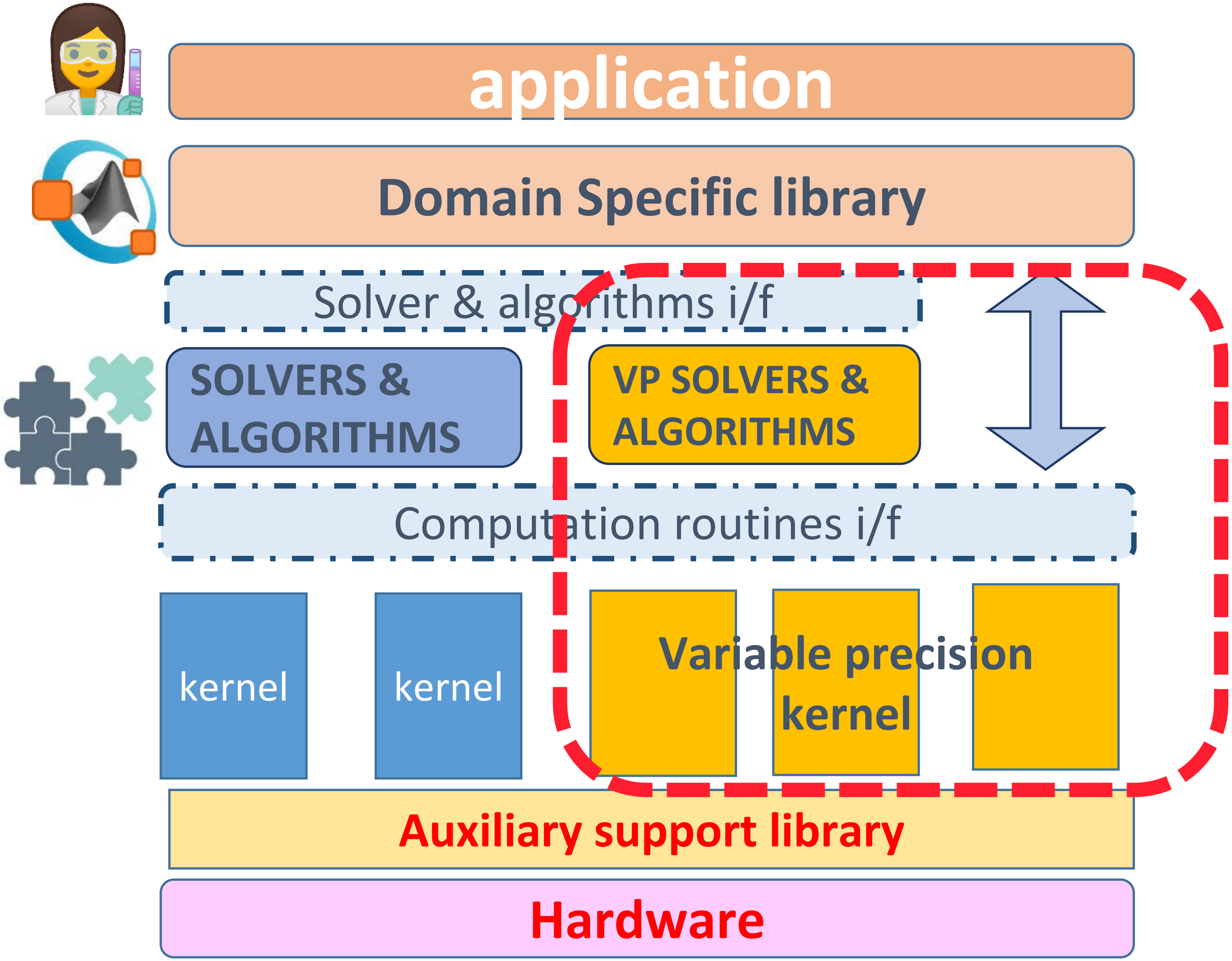
Internal format, for accumulation (high precision)

Intermediate vector, adjustable format (dense)

VARIABLE PRECISION SYSTEM



PROGRAMMING MODEL: HARDWARE & SOFTWARE LAYERS



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

RECAP: BENEFITS OF VARIABLE PRECISION

- **Augmenting accuracy inside the kernel reduces rounding errors → improves stability of the computation**
  - Augmenting the mantissa during accumulation is not sufficient

- **Usual solution is to tweak the solver (pre-conditioning, etc.) but this is costly, hazardous and very limited**
- **Another solution is to double precision *(→ quad !!)* in the intermediate calculation → huge impact in memory and in calculation time**
- **Using specialized data types (GMP, MPFR) has the same pitfalls**
  - At even higher cost in memory

- **Our solution:**
  - Variable precision, byte-aligned data format for intermediate data in memory
    - affordable memory footprint for intermediate data
  - Hardware support for variable precision in hardware co-processor
    - Up to 4x64 bits fractional part in internal accumulator

PERSPECTIVES

- **Early investigation carried on by CEA**
  - With support of other research projects
    - OPRECOMP, Imprenum, QUANTEX
  - First Use cases
  - Proof of concept = First FPGA prototype
  - Investigation on Compiler and library support

- **Mid-term Target : Proof of realization**
  - Re-engineering with actual memory subsystem & infrastructure
  - Improve co-processor integration with processor
  - SW integration (libraries, execution model ?)

- **Main publications**

  - Andrea Bocco, Yves Durand, and Florent de Dinechin. SMURF: Scalar multiple-precision unum Risc-V floating-point accelerator for scientific computing. In *Conference on Next-Generation Arithmetic*, March 2019
  - Tiago Trevisan Jost, Andrea Bocco, Yves Durand, Christian Fabre, Florent De Dinechin, Anca Molnos, Albert Cohen:Variable Precision Capabilities in RISC-V Processors, *RISC-V Workshop Zurich* (June 11 – 13, 2019)
  - Andrea Bocco, Yves Durand, and Florent de Dinechin. Dynamic precision numerics using a variable-precision UNUM type I HW coprocessor. In *26th IEEE Symposium of Computer Arithmetic (ARITH-26)*, June 2019.