VPSDK : a portability library for extended arithmetic operations targetting a RISC-V Variable eXtended Precision accelerator.

Linear algebra kernels

Direct and iteratives linear algebra kernels, such as linear solvers or eigensolvers, are ubiquitous in both scientific and industrial applications. But they both encounter issues.

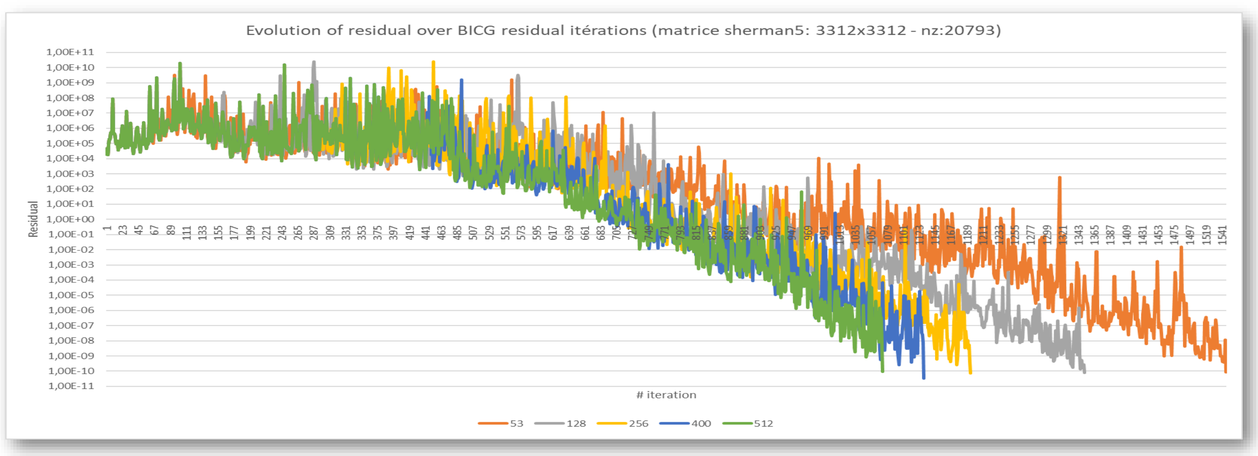
Iterative : $x_{k+1} = x_k + \alpha_k (b - Ax_k)$

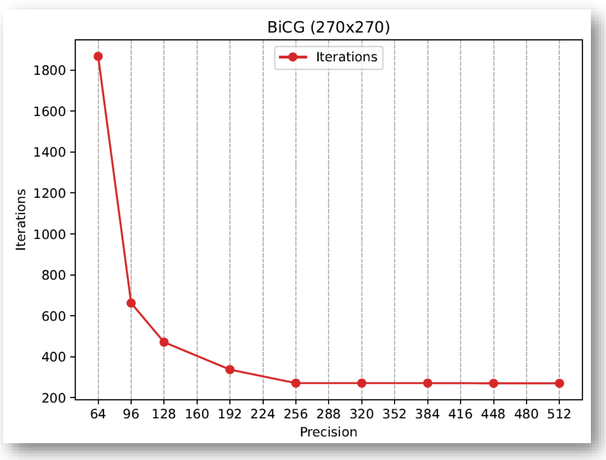
Direct : $A = LU; x = U^{-1}L^{-1}b$

	Direct	Iterative
Speed	Fast	Slow
Memory consumption	High	Low
Issues	Unusable for large problems	Sensitive to round-off errors May not converge

Impact of numerical precision on iterative solver

Extended precision limits the impact of round-off errors and thus speed-up, and in some case enable, the convergence of iterative methods.

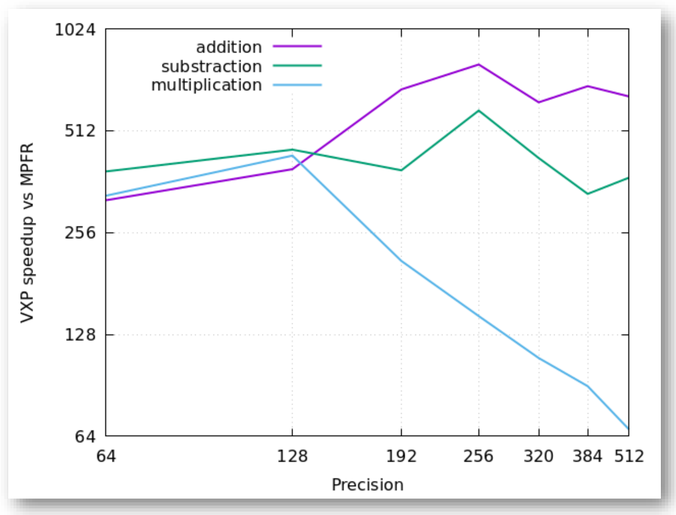




Variable precision solutions

Multiple solutions exists to handle variable precision in application code. Some are software only solutions and other are based on proprietary hardware accelerator. Each of them have their own advantages and limitations, but they all need specific programming models.

	MPFR	VXP accelerator
Supported langage	Standard C	RISC-V assembly
Syntax	mpfr_add,mpfr_sub, mpfr_mul, mpfr_div with mpfr_t struct	padd, psub, pmul
Max Precision	2^37 bits	512 bits
Portability	x86_64, aarch64, riscv64…	riscv64
Implementation	Software only	Hardware only

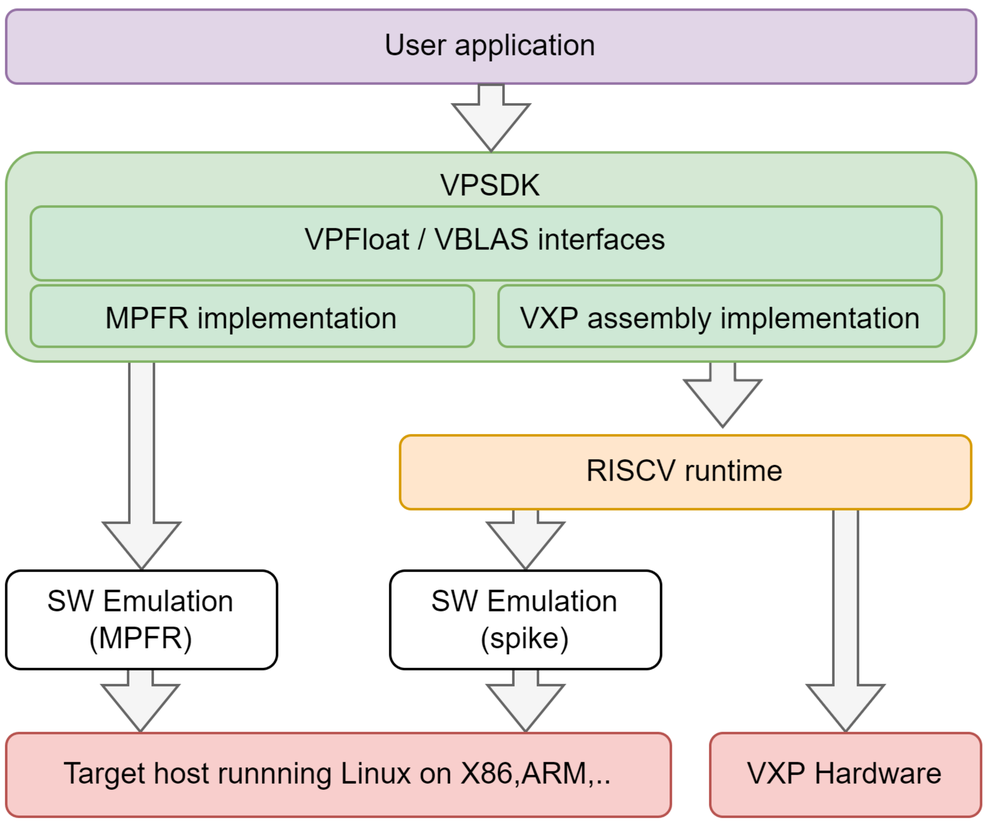


VPSDK

VPSDK provides

- A common interface for VPFloat numbers independent of VP backend choice
- Functions to manage dense and sparse matrices
- BLAS implementation for VPFloat numbers

VPSDK will be delivered in open-source




```
#include <VPFloat.hpp>
using namespace VPFloatPackage;

void main() {
  // Matissa size is 128 bits
  int precision=128;
  // Exponent size is 7 bits
  int exponent_size=7;
  // Stride size is 1 vpfloat elements
  int stride_size = 1;
  int n = 10;

  int vpfloat_memory_size = precision
                          + exponent_size + stride;

  // Set the precision to use for further code
  VPFloatComputingEnvironment::set_precision(precision);

  // VPFloat array declaration
  VPFloatArray r_k( exponent_size,
                    vpfloat_memory_size,
                    stride_size, n);

  // VPfloat scalar declaration
  VPFloat rs (exponent_size,
              vpfloat_memory_size, stride_size );

  rs = 10.0;

  for (int i = 1; i <= n; i++ ) {
      r_k[i-1] = rs / (double)i;
  }
}
```


Jerome Fereyre[1], Alexandre Hoffmann[1]

1 Univ. Grenoble Alpes, CEA, LIST, 38000 Grenoble

This work has been performed in the context of the EPI and TRISTAN projects. EPI has received funding from the European Union's Horizon research and innovation program under Grant Agreement EPI-SGA1: 826647. TRISTAN has received funding from the Key Digital Technologies Joint Undertaking (KDT JU) under Grant Agreement nr. 101095947