



**HAL**  
open science

## Optimizing Precision for High-Performance, Robust, and Energy-Efficient Computations

Roman Iakymchuk, Stef Graillat, Fabienne Jézéquel, Daichi Mukunoki,  
Toshiyuki Imamura, Yiyu Tan, Atsushi Koshiba, Jens Huthmann, Kentaro  
Sano, Norihisa Fujita, et al.

► **To cite this version:**

Roman Iakymchuk, Stef Graillat, Fabienne Jézéquel, Daichi Mukunoki, Toshiyuki Imamura, et al..  
Optimizing Precision for High-Performance, Robust, and Energy-Efficient Computations. International  
Conference on High Performance Computing in Asia-Pacific Region., Jan 2020, Fukuoka, Japan.  
hal-02401813

**HAL Id: hal-02401813**

**<https://hal.science/hal-02401813>**

Submitted on 15 May 2020

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Optimizing Precision for High-Performance, Robust, and Energy-Efficient Computations

\* All authors contributed equally to this research

## Introduction

In numerical computations, the precision of floating-point computations is a key factor to determine the performance (speed and energy-efficiency) as well as the reliability (accuracy and reproducibility). However, the precision generally plays a contrary role for both. Therefore, the ultimate concept for maximizing both at the same time is the **optimized/reduced precision computation through precision-tuning, which adjusts the minimal precision for each operation and data**. Several studies have been already conducted for it so far, but the scope of those studies is limited to the precision-tuning alone. Instead, we propose a more broad concept of the optimized/minimal precision and robust (numerically reliable) computing with precision-tuning, involving both hardware and software stack

### Available Components

#### (1) Precision-tuning with numerical validation based on stochastic arithmetic

- Rounding-errors can be estimated stochastically with a **reasonable cost** (for details, see "(A) Stochastic Arithmetic Tools" at the bottom left)
- General scheme applicable for **any floating-point computations**

**Tools:**  
• PROMISE [17] (based on a stochastic arithmetic library, CADNA [18]), Verrou [19], etc.

**Related works (not validation-based):**  
• Precimonious [20], GPUMixer [22] etc.

#### (2) Arbitrary-precision libraries and fast accurate numerical libraries

- Reduced-/mixed-precision with FP16/FP32/FP64 enables us to **improve performance & energy-efficiency**
- High-precision libraries and fast accurate computation methods have been developed for **reliable & reproducible computation**

**Tools:**  
• High-precision arithmetic: binary128 (intel, gcc), QD [1], MPFR [2], ARPREC [3], CAMPARY [4], etc.  
• Accurate sum/dot: AccSum/Dot [5], Ozaki-scheme [6], etc.  
• Numerical libraries: MPLAPACK [7], QPEigen [8], QPBLAS [9], XBLAS [10], ReproBLAS [11], ExBLAS [12], OzBLAS [13], etc.

#### (3) Field-Programmable Gate Array (FPGA) with High-Level Synthesis (HLS)

- FPGA enables us to implement any operations on hardware, including arbitrary-precision operations
- HLS enables us to use FPGAs through existing programming languages such as C/C++ and OpenCL
- FPGA can be used to perform **arbitrary-precision computations on hardware** efficiently (high-performance and energy-efficient)

**Tools:**  
• Compilers: SPGen [14], Nymbble [15], etc.  
• Custom floating-point operation generator: FloPoCo [16], etc.

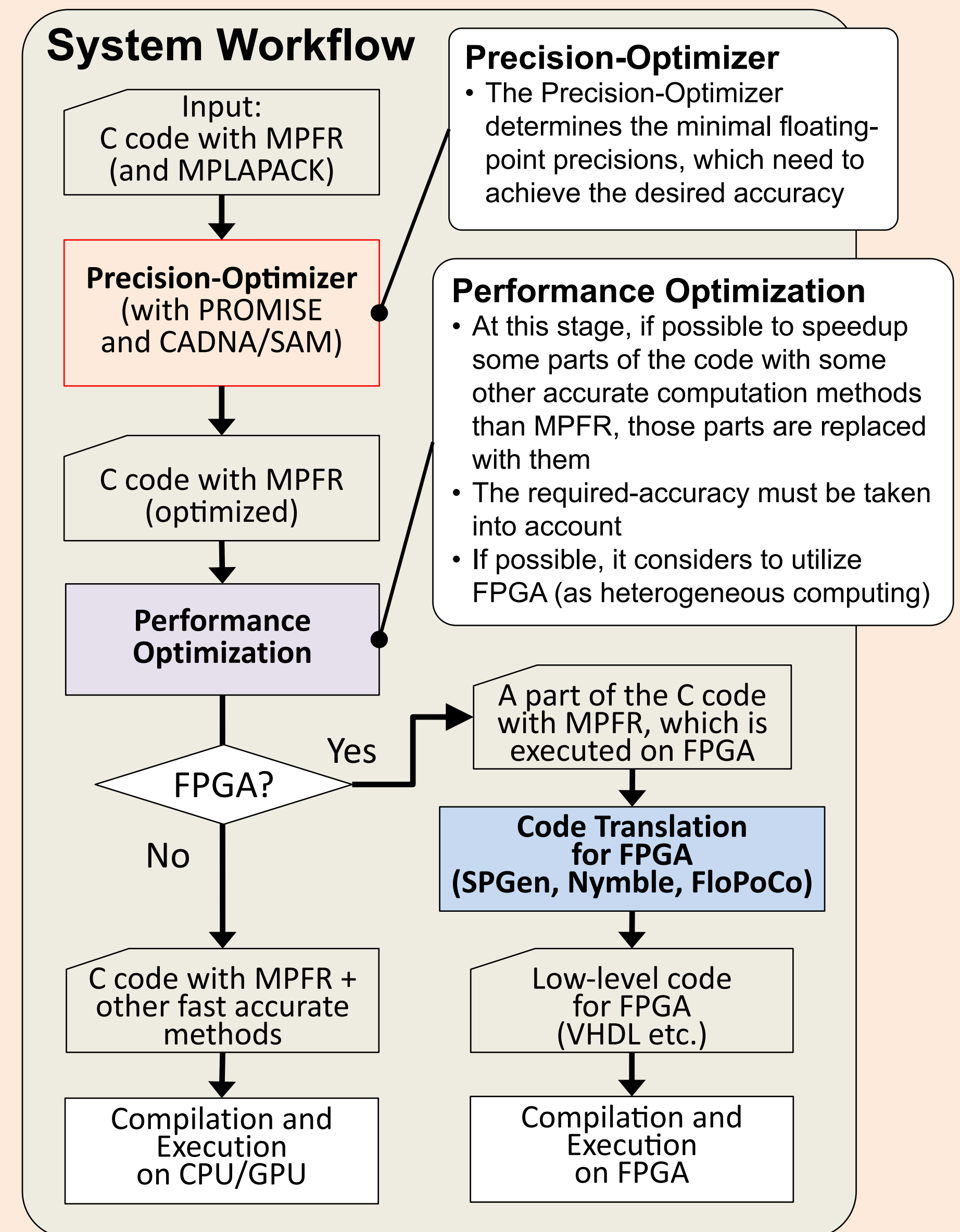
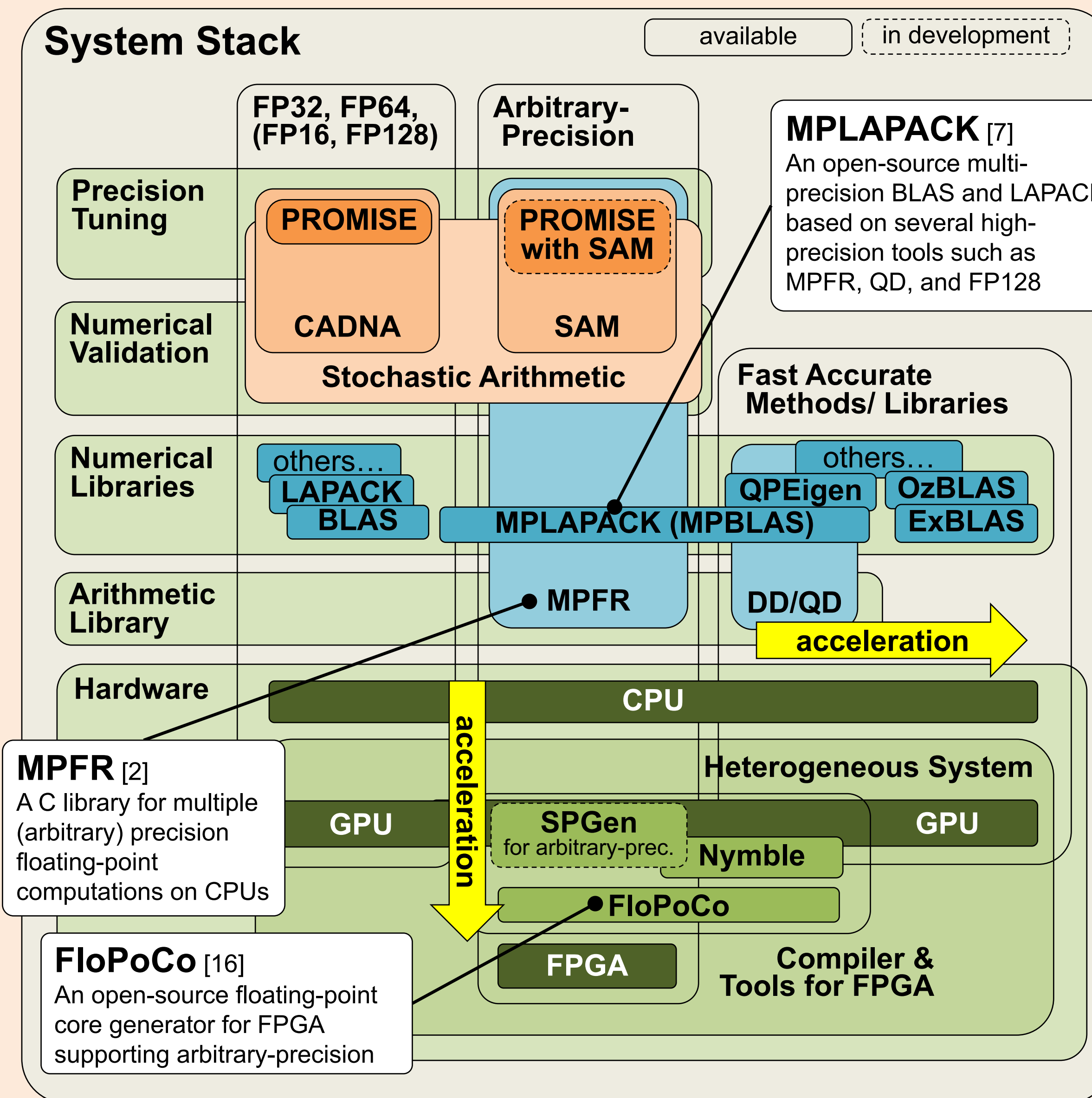
Red: Components developed by us

## Our Proposal

### Minimal-Precision Computing

Minimal-precision computing is both reliable (aka robust) and sustainable as it ensures the requested accuracy of the result as well as is energy-efficient

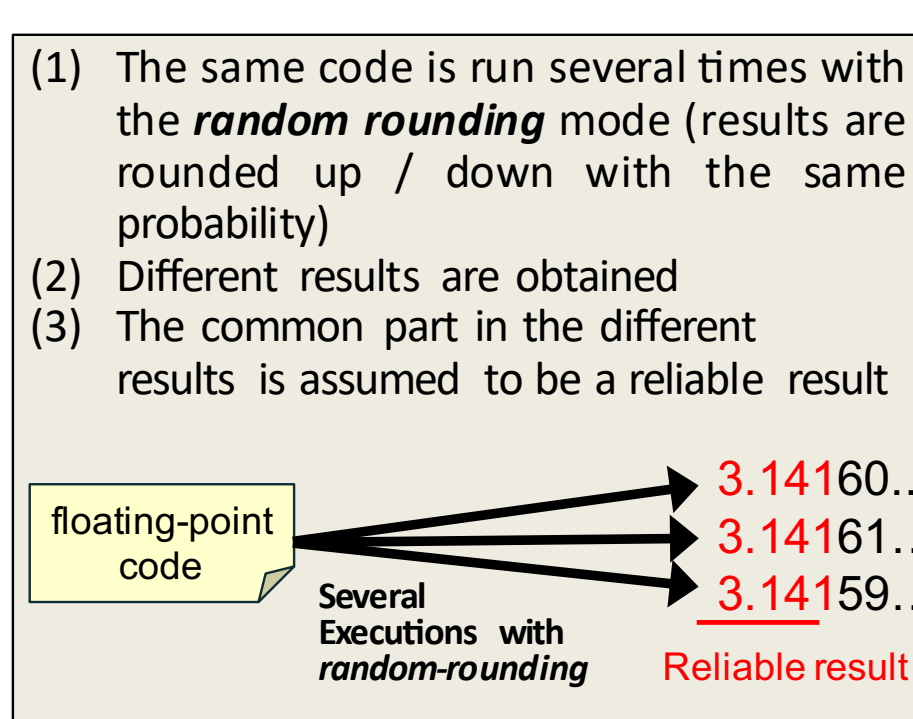
- High-performance**  
Performance can be improved through the minimal-precision as well as fast numerical libraries and accelerators
- Energy-Efficient**  
Through the minimal-precision as well as the energy-efficient hardware acceleration with FPGA and GPU
- Robust (Numerically Reliable)**  
To ensure the requested accuracy, the precision-tuning is processed based on numerical validation, guaranteeing also reproducibility
- General**  
Our scheme is applicable for any floating-point computations. It contributes to low development cost and sustainability (easy maintenance and system portability)
- Comprehensive**  
We propose a total system from the precision-tuning to the execution of the tuned code, combining heterogeneous hardware and hierarchical software stack
- Realistic**  
Our system can be realized by combining available in-house technologies



## Our Contributions

### A Stochastic Arithmetic Tools

Discrete Stochastic Arithmetic (DSA) [21] enables us to estimate rounding errors (i.e., the number of correct digits in the result) with 95% accuracy by executing the code 3 times with random-rounding. DSA is a general scheme applicable for any floating-point operations: no special algorithms and no code modification are needed. It is a light-weight approach in terms of performance, usability, and development cost compared to the other numerical verification / validation methods.

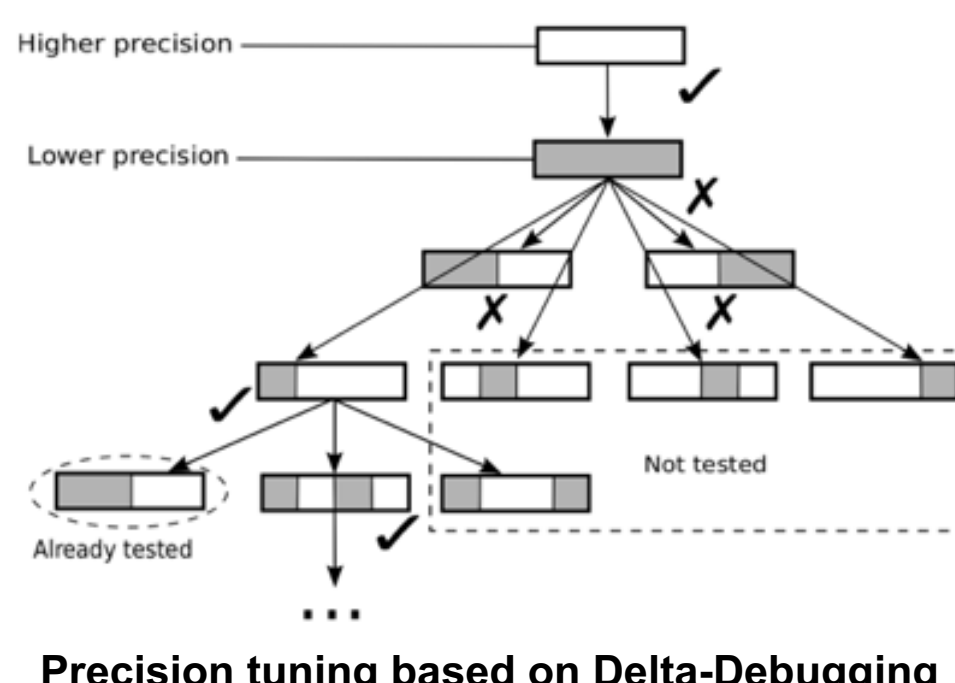


**CADNA & SAM (Sorbonne University)**  
• CADNA (Control of Accuracy and Debugging for Numerical Applications) [18] is a DSA library for FP16/32/64/128  
• CADNA can be used on CPUs in Fortran/C/C++ codes with OpenMP & MPI and on GPUs with CUDA.  
• SAM (Stochastic Arithmetic in Multiprecision) [23] is a DSA library for arbitrary-precision with MPFR.



**PROMISE (Sorbonne University)**

• PROMISE (Precision OptiMISE) [17] is a tool based on Delta-Debugging [24] to automatically tune the precision of floating-point variables in C/C++ codes  
• The validity of the results is checked with CADNA  
• We are going to extend PROMISE for arbitrary-precision with MPFR



### B FPGA as an Arbitrary-Precision Computing Platform

FPGA enables us to implement arbitrary-precision on hardware. High-level Synthesis (HLS) enables us to program it in OpenCL. However, compiling arbitrary-precision code and obtaining high performance are still challenging. Heterogeneous computing with FPGA & CPU/GPU is also a challenge

#### SPGen (RIKEN)

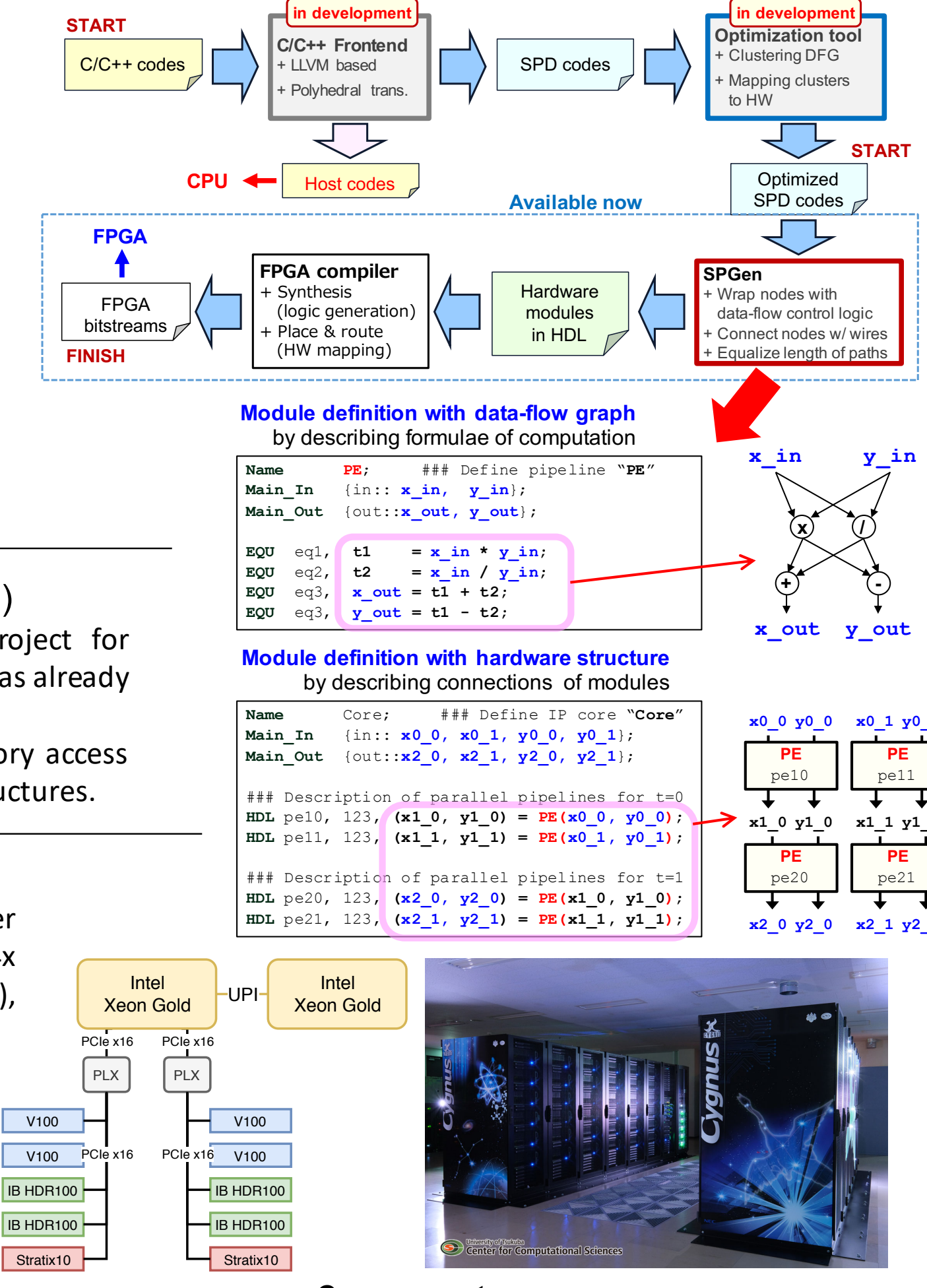
• SPGen (Stream Processor Generator) [14] is a compiler to generate HW module codes in Verilog-HDL for FPGA from input codes in Stream Processing Description (SPD) Format. The SPD uses a data-flow graph representation, which is suitable for FPGA.  
• It supports FP32 only, but we are going to extend SPGen to support arbitrary-precision floating-point. Currently, there is no FPGA compiler supporting arbitrary-precision.

#### Nymbble (TU Darmstadt, RIKEN)

• Nymbble [15] is another compiler project for FPGA. It directly accepts C codes and has already started to support arbitrary-precision.  
• It is more suited for non-linear memory access pattern, like with graph based data structures.

#### Cygnus (University of Tsukuba)

• Cygnus is the world first supercomputer system equipped with both GPU (4x Tesla V100) and FPGA (2x Stratix 10), installed in CCS, University of Tsukuba  
• Each Stratix 10 FPGA has four external links at 100Gbps. 64 FPGAs make 8x8 2D-Torus network for communication  
• This project targets such a heterogeneous system with FPGA.



### C Fast and Accurate Numerical Libraries

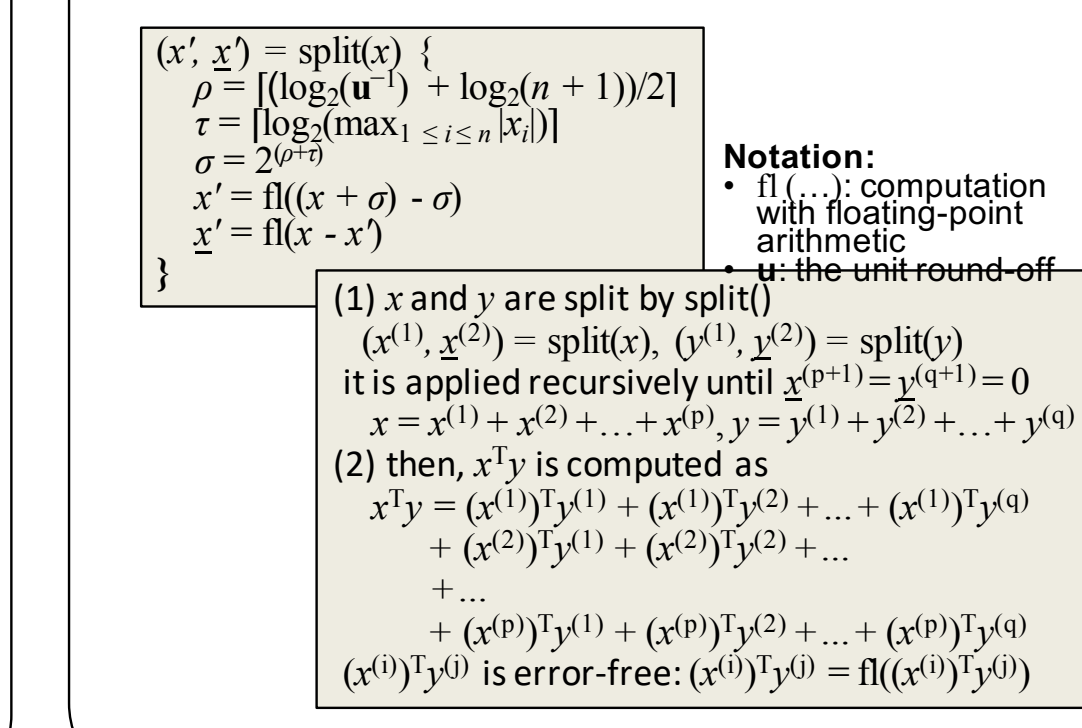
Arbitrary-precision arithmetic is performed using MPFR on CPUs, but the performance is very low. To accelerate it, we are developing several numerical libraries supporting accurate computation based on high-precision arithmetic or algorithmic approach. Some software also support GPU acceleration.

#### QPEigen & QPBLAS (JAEA, RIKEN)

• Quadruple-precision Eigen solvers (QPEigen) [8, 25] is based on double-double (DD) arithmetic. It is built on a quadruple-precision BLAS (QPBLAS) [9]. They support distributed environments with MPI; equivalent to ScaLAPACK's Eigen solver and PBLAS

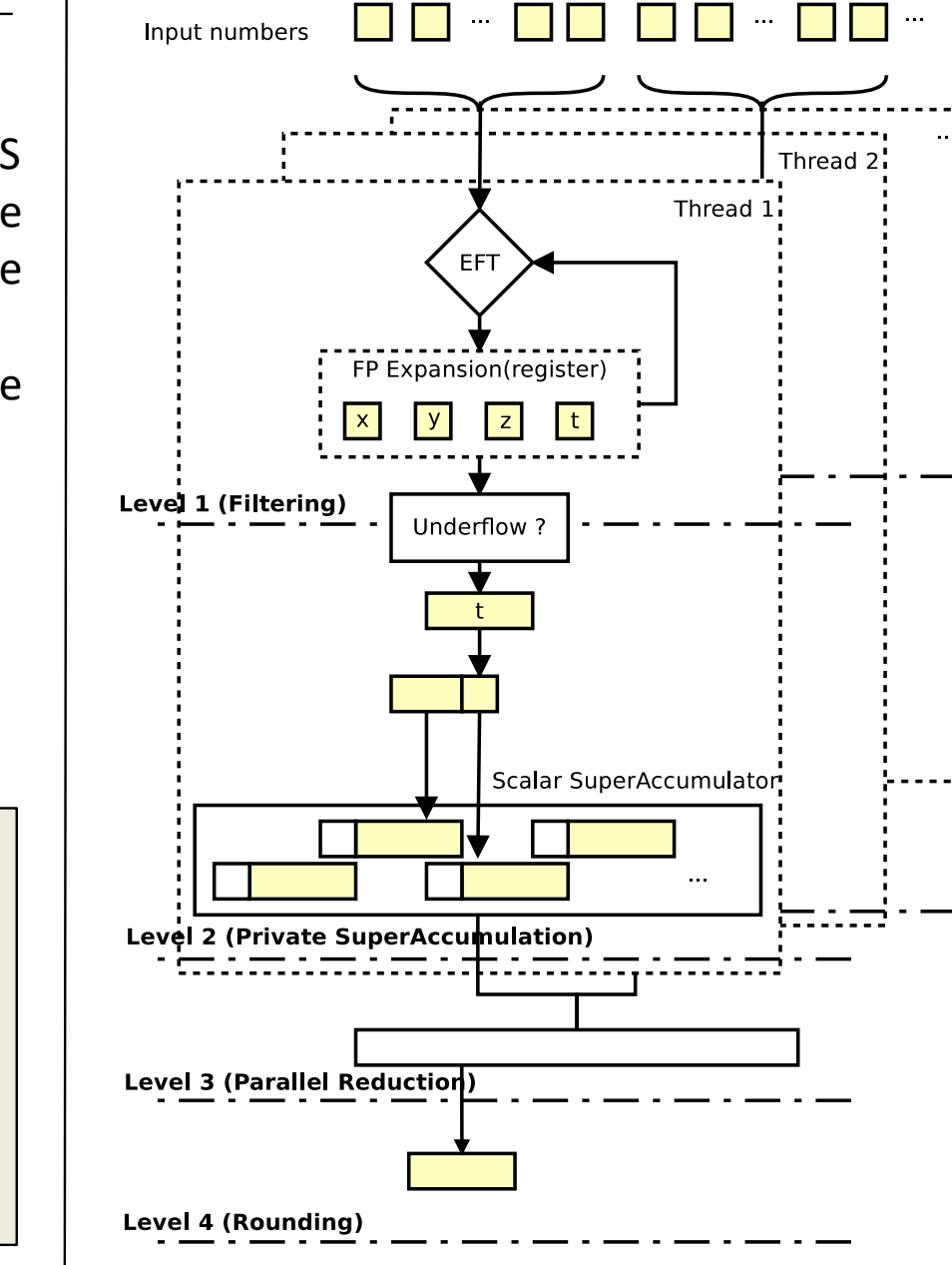
• OzBLAS [13] is an accurate & reproducible BLAS using Ozaki scheme [18], which is an accurate matrix multiplication method based on the error-free transformation of dot-product

• The accuracy is tunable and depends on the range of the inputs and the vector length  
• CPU and GPU (CUDA) versions



#### ExBLAS (Sorbonne University)

• ExBLAS [12] is an accurate & reproducible BLAS based on floating-point expansions with error-free transformations (EFT: twosum and twoprod) and super-accumulator  
• Assures reproducibility through assuring correct-rounding; it preserves every bit of information until the final rounding to the desired format  
• CPU (Intel TBB) and GPU (OpenCL) versions



## Conclusion & Future Work

We proposed a new systematic approach for minimal-precision computations. This approach is robust, general, comprehensive, high-performant, and realistic. Although the proposed system is still in development, it can be constructed by combining already available (developed) in-house technologies and extending them. Our ongoing step is to demonstrate the system on a proxy application

### Acknowledgement:

This research was partially supported by the European Union's Horizon 2020 research, innovation programme under the Marie Skłodowska-Curie grant agreement with the Robust project No. 842528, the Japan Society for the Promotion of Science (JSPS) KAKENHI Grant No. 19K20286, and Multidisciplinary Cooperative Research Program in CCS, University of Tsukuba.

### References

[1] D. H. Bailey, "QD (C++/Fortran-90 double-double and quad-double package)," <http://crd.lbl.gov/~dhbailey/mpdist>  
 [2] G. Hanrot et al., "MPFR: GNU MPFR Library," <http://www.mpfr.org>  
 [3] D. H. Bailey et al., "ARPREC: An Arbitrary Precision Computation Package," Lawrence Berkeley National Laboratory Technical Report, No. LBNL-53651, 2002.  
 [4] CAMPARY, <http://homepages.laas.fr/mjmmjides/campary>  
 [5] T. Ogita et al., "Accurate Sum and Dot Product," SIAM J. Sci. Comput., vol. 26, pp. 1955-1988, 2005.  
 [6] K. Ozaki et al., "Error-free transformations of matrix multiplication by using fast routines of matrix multiplication and its applications," Numer. Algorithms, vol. 59, No. 1, pp. 95-118, 2012.  
 [7] M. Nakata, "The MPACK (MPLAS/MPLAPACK): a multiprecision arithmetic version of BLAS and LAPACK," Ver. 0.6.7, <http://mplapack.sourceforge.net>, 2010.  
 [8] Japan Atomic Energy Agency, "Quadruple Precision Eigenvalue Calculation Library QPEigen Ver.1.0 User's Manual," [https://ccse.jaea.go.jp/ja/download/qpeigen\\_k/qpeigen\\_manual\\_en-1.0.pdf](https://ccse.jaea.go.jp/ja/download/qpeigen_k/qpeigen_manual_en-1.0.pdf), 2015.  
 [9] Japan Atomic Energy Agency, "Quadruple Precision BLAS Routines QPBLAS Ver.1.0 User's Manual," [https://ccse.jaea.go.jp/ja/download/qpblas/qpblas\\_manual\\_en-1.0.pdf](https://ccse.jaea.go.jp/ja/download/qpblas/qpblas_manual_en-1.0.pdf), 2013.  
 [10] X. Li et al., "XBLAS = Extra Precise Basic Linear Algebra Subroutines," <http://www.netlib.org/xblas>  
 [11] P. Ahrens et al., "ReproBLAS - Reproducible Basic Linear Algebra Sub-programs," <https://beob.cs.berkeley.edu/reproblas> in Asia Poster, International Supercomputing Conference (ISC'16), 2016.  
 [12] R. Iakymchuk et al., "ExBLAS: Reproducible and Accurate BLAS Library," Proc. Numerical Reproducibility at Exascale (NRE2015) workshop at SC15, HAL ID: hal-01202396, 2015.

[13] D. Mukunoki et al., "Accurate and Reproducible BLAS Routines with Ozaki Scheme for Many-core Architectures," Proc. PPAM2019, 2019 (accepted).  
 [14] K. Sano et al., "Stream Processor Generator for HPC to Embedded Applications on FPGA-based System Platform," Proc. First International Workshop on FPGAs for Software Programmers (FSP 2014), pp. 43-48, 2014.  
 [15] J. Huthmann et al., "Hardware/Software co-compilation with the Nymbble system," 2013 8th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), pp. 1-8, 2013.  
 [16] F. Dinechin and B. Pasca, "Designing custom arithmetic data paths with FloPoCo," IEEE Design & Test of Computers, Vol. 28, No. 4, pp. 18-27, 2011.  
 [17] S. Graillat et al., "Auto-tuning for floating-point precision with Discrete Stochastic Arithmetic," Journal of Computational Science, Vol.36, 101017, 2019.  
 [18] F. Jézéquel and J.-M. Chesneau, "CADNA: a library for estimating round-off error propagation," Computer Physics Communications, Vol. 178, No. 12, pp. 933-955, 2008.  
 [19] F. Fédotte and B. Lathuilière, "Debugging and optimization of HPC programs in mixed precision with the Verrou tool," hal-02044101, 2019.  
 [20] C. Rubio-González et al., "Precimonious: Tuning Assistant for Floating-Point Precision," Proc. SC'13, 2013.  
 [21] J. Vignes, "Discrete Stochastic Arithmetic for Validating Results of Numerical Software," Numer. Algorithms, Vol. 37, No. 1-4, pp. 377-390, 2004.  
 [22] I. Laguna, P. C. Wood, R. Singh, S. Bagchi, "GPUMixer: Performance-Driven Floating-Point Tuning for GPU Scientific Applications," Proc. IC2019, pp. 227-246, 2019.  
 [23] S. Graillat et al., "Stochastic Arithmetic in Multiprecision," Mathematics in Computer Science, Vol. 5, No. 4, pp. 359-375, 2011.  
 [24] A. Zeller and R. Hildebrandt, "Simplifying and isolating failure-inducing input," IEEE Trans. Softw. Eng., Vol. 28, No. 2, pp. 183-200, 2002.  
 [25] Y. Hirota et al., "Performance of quadruple precision eigenvalue solver libraries QPEigenK and QPEigenG on the K computer," HPC in Asia Poster, International Supercomputing Conference (ISC'16), 2016.