Numerical Reproducibility based on Minimal-Precision Validation

Computational Reproducibility at Exascale Workshop (CRE2019)
Conjunction with SC19, at Denver, Colorado, the United States,
17 November 2019

RIKEN Center for Computational Science (R-CCS) (Japan)
Toshiyuki Imamura, Daichi Mukunoki, Yiyu Tan,

Sorbonne University, CNRS, LIP6 (France)
Fabienne Jézéquel, Stef Graillat, Roman Iakymchuk

What does Reproducibility refer to ?

In computational science, reproducibility is considered from several view-points depending on the context and demand. ***Reproducibility*** refers to **a capability of obtaining the identical result,** but it often means ``re-playability'' or ``re-traceability“.

- ***Bit-level reproducibility***
  is **the capability to reproduce the bit-wise identical result** with the same input on any HW/SW configuration. **No general approach** for any floating-point computation has been proposed yet. It is **non-realistic** to support bit-level reproducibility **on all floating-point computations** through the existing approaches.

- ***Weak numerical reproducibility***
  the reproducibility, (up to a high probability) of **the computation result with a certain accuracy demanded by the user**. The underlying numerical validation is performed using **a statistical approach** that estimates with a high probability **the number of correct digits** in the computation result.

  → ***The extension of our minimal-precision computing scheme, which validates the accuracy (demanded by the user) of the result through the minimal-precision use.***

Minimal Precision Computing

- ***The minimal-precision computing***

  high-performance and energy-efficient as well as reliable (accurate, reproducible, and validated) computations

- ***systematic approach combining internally***
  1. a precision-tuning method based on Discrete Stochastic Arithmetic (DSA),
  2. arbitrary-precision arithmetic libraries,
  3. fast and accurate numerical libraries, and
  4. Field-Programmable Gate Array (FPGA) with High-Level Synthesis (HLS)

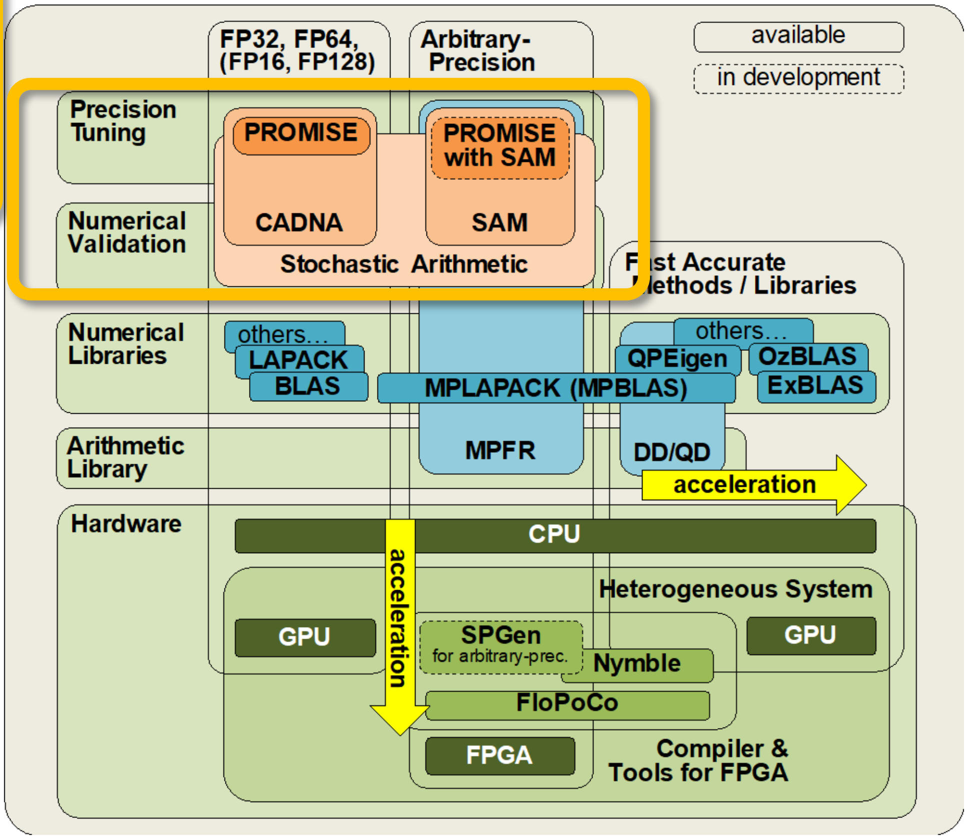
- Reliable, General, Comprehensive, High-performance,

  Energy-efficient, Realistic

System Overview

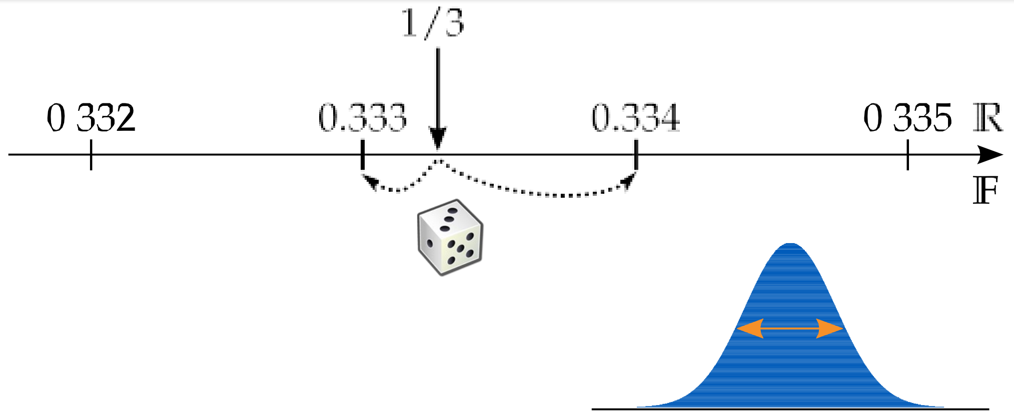
Main software/hardware components for minimal-precision computing system:

1. **Arbitrary-precision arithmetic library**
   - MPFR (GNU)
2. **Precision-tuning method based on stochastic arithmetic**
   - Stochastic libraries: CADNA & SAM (Sorbonne U.)
   - Precision-tuner: PROMISE (Sorbonne U.)
3. **Fast & accurate numerical libraries**
   - Accurate BLAS: ExBLAS (Sorbonne U.), OzBLAS (TWCU/RIKEN)
   - Quadruple-precision BLAS and Eigen solver: QPBLAS/QPEigen (JAEA/RIKEN)
   - Other open source (QD, MPLAPACK, etc.)
4. **Heterogeneous system with FPGA**
   - FPGA-GPU-CPU system: "Cygnus" (U. Tsukuba)
   - Compilers: SPGen (RIKEN), Nymble (TU Darmstadt/RIKEN)



Discrete Stochastic Arithmetic (DSA)

[Vignes, 2004]



- each operation executed **3 times** with a random rounding mode
- **number of correct digits in the results estimated using Student's test with the probability 95%**
- estimation may be invalid if both operands in a multiplication or a divisor are not significant.
  $\Rightarrow$ control of multiplications and divisions: *self-validation* of DSA.
- in DSA rounding errors are assumed centered.
  even if they are not rigorously centered, the accuracy estimation can be considered correct up to 1 digit.

Implementation of DSA

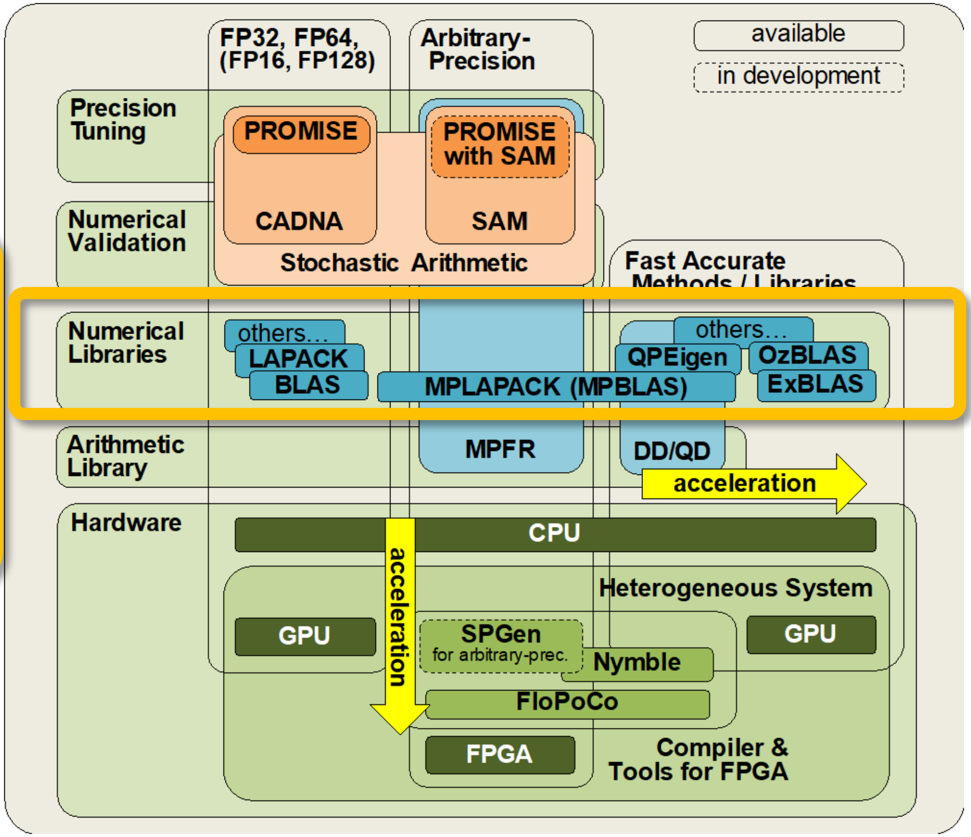
- CADNA: for programs in single and/or double precision
  http://cadna.lip6.fr
- SAM: for arbitrary precision programs (based on MPFR)
  http://www-pequan.lip6.fr/~jezequel/SAM

- estimate accuracy and detect numerical instabilities
- provide stochastic types (3 classic type variables and 1 integer):
  - `float_st` in single precision
  - `double_st` in double precision
  - `mp_st` in arbitrary precision
- all operators and mathematical functions overloaded
  ⇒ few modifications in user programs

System Overview

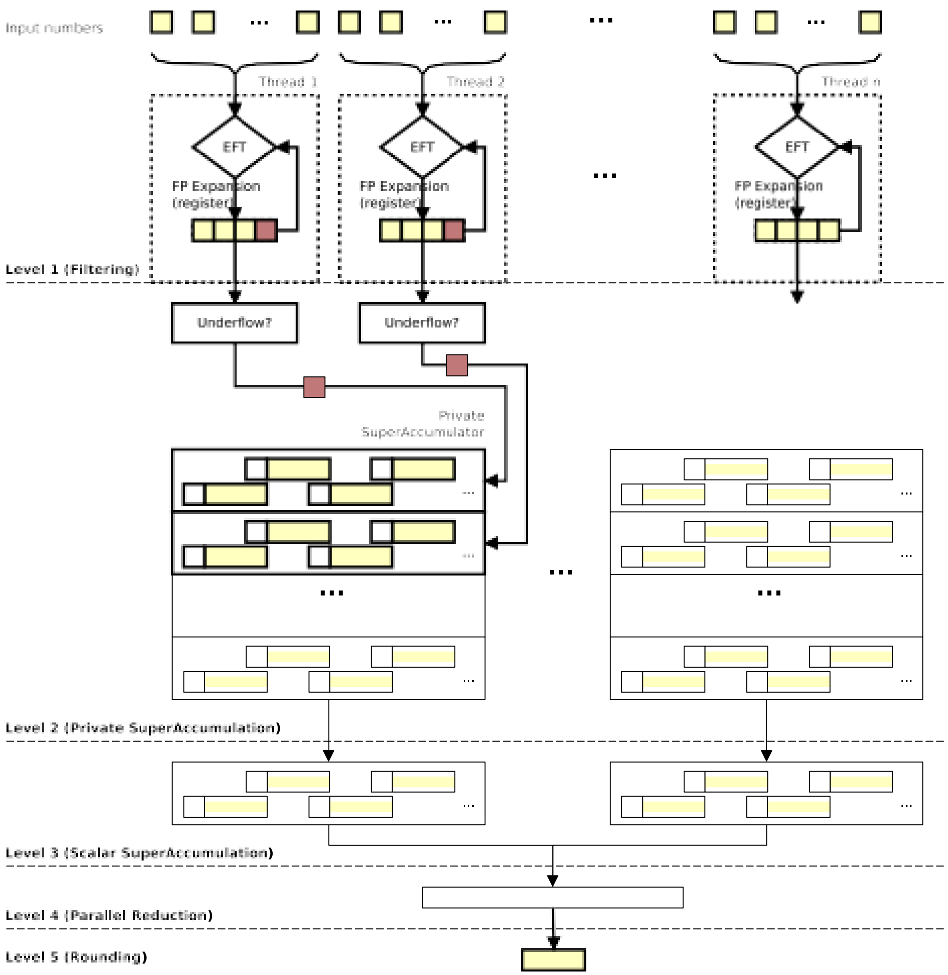
Main software/hardware components for minimal-precision computing system:

1. **Arbitrary-precision arithmetic library**
   - MPFR (GNU)
2. **Precision-tuning method based on stochastic arithmetic**
   - Stochastic libraries: CADNA & SAM (Sorbonne U.)
   - Precision-tuner: PROMISE (Sorbonne U.)
3. **Fast & accurate numerical libraries**
   - Accurate BLAS: ExBLAS (Sorbonne U.), OzBLAS (TWCU/RIKEN)
   - Quadruple-precision BLAS and Eigen solver: QPBLAS/QPEigen (JAEA/RIKEN)
   - Other open source (QD, MPLAPACK, etc.)
4. **Heterogeneous system with FPGA**
   - FPGA-GPU-CPU system: “Cygnus” (U. Tsukuba)
   - Compilers: SPGen (RIKEN), Nymble (TU Darmstadt/RIKEN)



Accurate/ Reproducible BLAS(ExBLAS)

Highlights of the Algorithm



- Parallel algorithm with 5-levels
- Suitable for today's parallel architectures
- **Based on FPE with EFT and Kulisch accumulator**
- Guarantees "inf" precision

→ **bit-wise reproducibility**

Accurate/ Reproducible BLAS(OzBLAS)

Accurate & reproducible dot-product ($x^{\mathrm{T}}y$)

The vectors can be split recursively until $\underline{x}^{(p)}$ and $\underline{y}^{(q)}$ become zero

$$x = x^{(1)} + x^{(2)} + x^{(3)} + \cdots + x^{(p\text{-}1)} + \underline{x}^{(p)}$$

$$y = y^{(1)} + y^{(2)} + y^{(3)} + \cdots + y^{(q\text{-}1)} + \underline{y}^{(q)}$$

$x^{\mathrm{T}}y$ is transformed to the sum of multiple dot-products

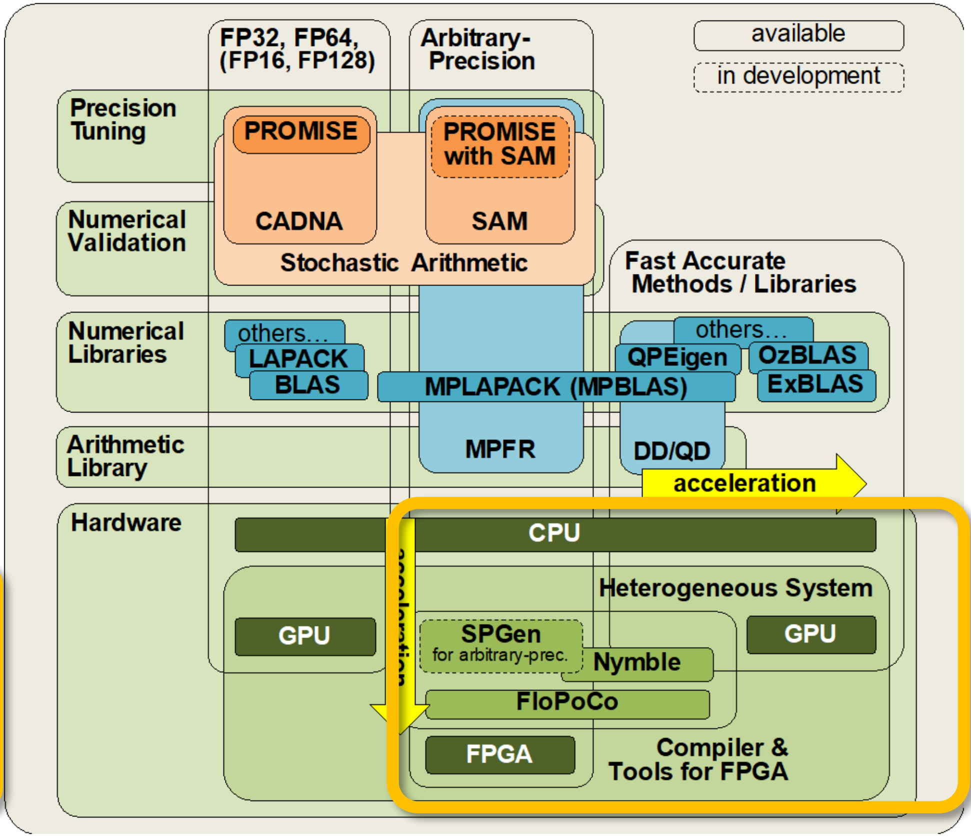
$$\begin{aligned} x^{\mathrm{T}}y = {} & (x^{(1)})^{\mathrm{T}}y^{(1)} + (x^{(1)})^{\mathrm{T}}y^{(2)} + (x^{(1)})^{\mathrm{T}}y^{(3)} + \cdots + (x^{(1)})^{\mathrm{T}}y^{(q\text{-}1)} \\ & + (x^{(2)})^{\mathrm{T}}y^{(1)} + (x^{(2)})^{\mathrm{T}}y^{(2)} + (x^{(2)})^{\mathrm{T}}y^{(3)} + \cdots + (x^{(2)})^{\mathrm{T}}y^{(q\text{-}1)} \\ & + (x^{(3)})^{\mathrm{T}}y^{(1)} + (x^{(3)})^{\mathrm{T}}y^{(2)} + (x^{(3)})^{\mathrm{T}}y^{(3)} + \cdots + (x^{(3)})^{\mathrm{T}}y^{(q\text{-}1)} \\ & + \cdots \\ & + (x^{(p\text{-}1)})^{\mathrm{T}}y^{(1)} + (x^{(p\text{-}1)})^{\mathrm{T}}y^{(2)} + (x^{(p\text{-}1)})^{\mathrm{T}}y^{(3)} + \cdots + (x^{(p\text{-}1)})^{\mathrm{T}}y^{(q\text{-}1)} \end{aligned}$$

Those computations can be performed using standard BLAS (e.g., MKL, OpenBLAS, cuBLAS) → **Productive & High-performance**

System Overview

Main software/hardware components for minimal-precision computing system:

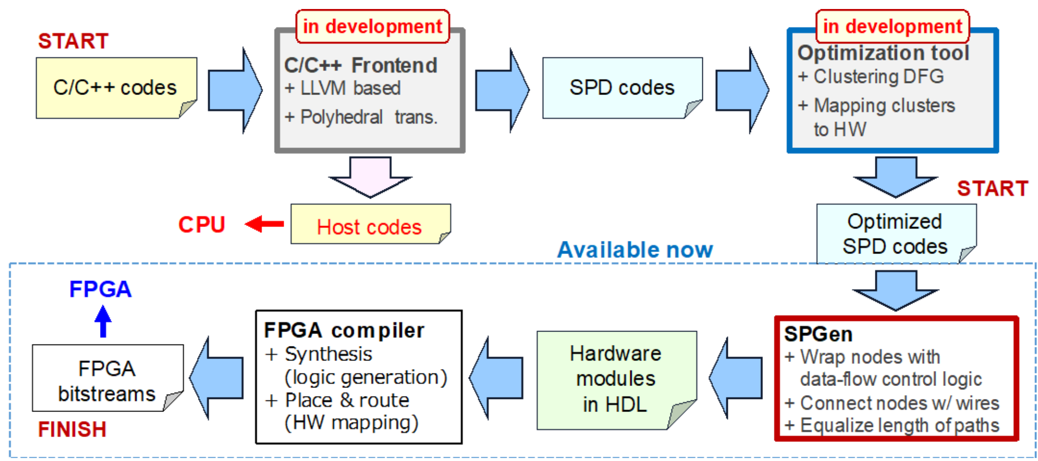
1. **Arbitrary-precision arithmetic library**
   - MPFR (GNU)
2. **Precision-tuning method based on stochastic arithmetic**
   - Stochastic libraries: CADNA & SAM (Sorbonne U.)
   - Precision-tuner: PROMISE (Sorbonne U.)
3. **Fast & accurate numerical libraries**
   - Accurate BLAS: ExBLAS (Sorbonne U.), OzBLAS (TWCU/RIKEN)
   - Quadruple-precision BLAS and Eigen solver: QPBLAS/QPEigen (JAEA/RIKEN)
   - Other open source (QD, MPLAPACK, etc.)
4. **Heterogeneous system with FPGA**
   - FPGA-GPU-CPU system: “Cygnus” (U. Tsukuba)
   - Compilers: SPGen (RIKEN), Nymble (TU Darmstadt/RIKEN)



FPGA Performance enhancement

- SPGen (RIKEN)
  - **a compiler to generate HW module codes in Verilog-HDL for FPGA** from input codes in Stream Processing Description (SPD) Format.
  - 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.**

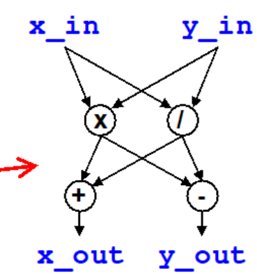
- Nymble (TU Darmstadt, RIKEN)
  - another compiler project for FPGA. It **directly accepts C codes** and has already started to support **arbitrary-precision.**



Module definition with data-flow graph
by describing formulae of computation

```
Name      PE;      ### Define pipeline "PE"
Main_In   {in:: x_in,  y_in};
Main_Out  {out::x_out, y_out};

EQU  eq1,  t1    = x_in * y_in;
EQU  eq2,  t2    = x_in / y_in;
EQU  eq3,  x_out = t1 + t2;
EQU  eq3,  y_out = t1 - t2;
```

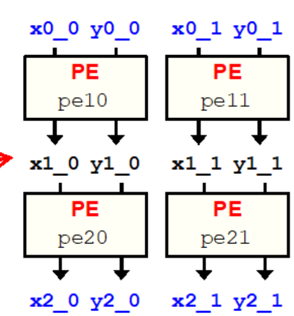


Module definition with hardware structure
by describing connections of modules

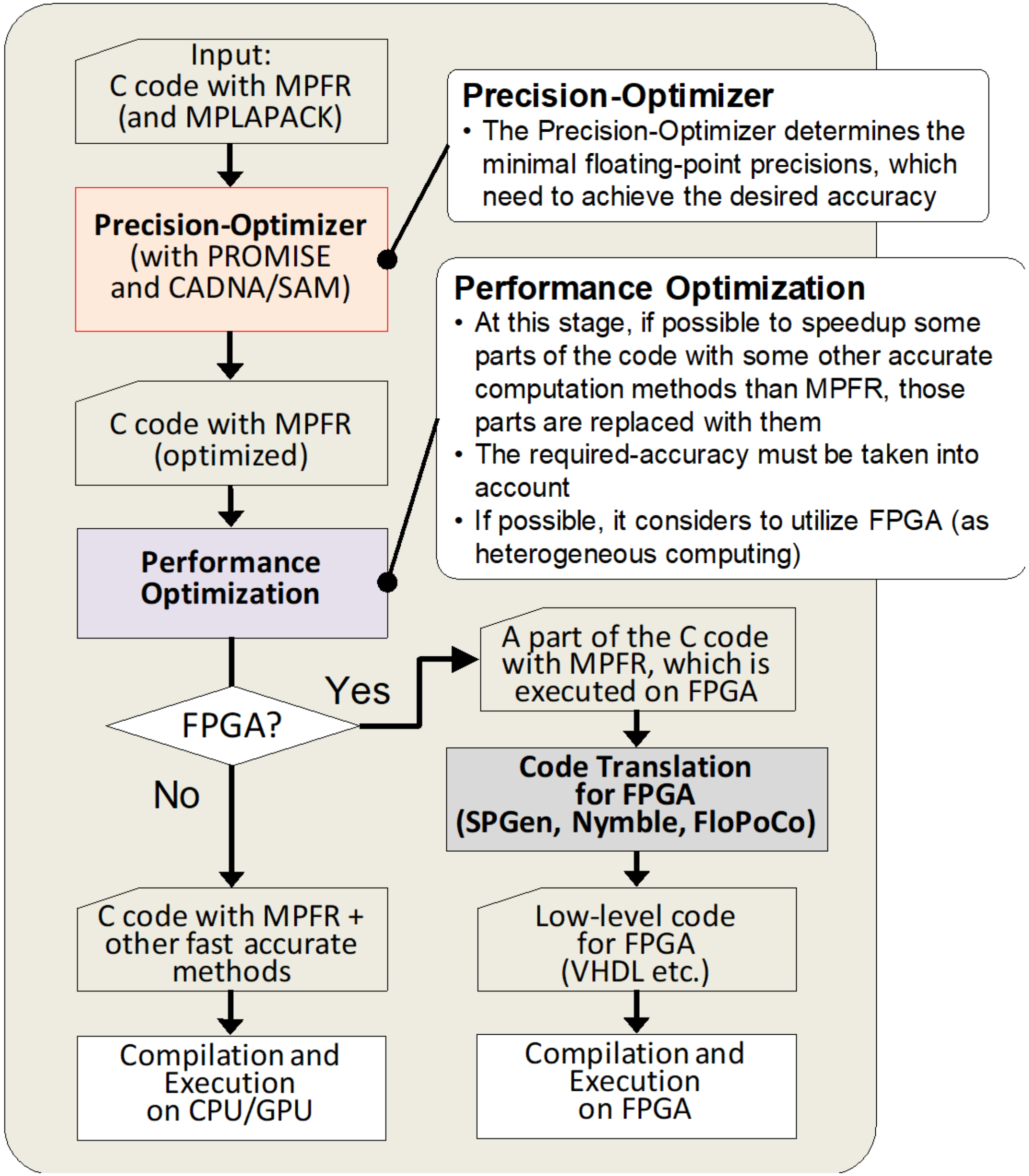
```
Name      Core;      ### Define IP core "Core"
Main_In   {in:: x0_0, x0_1, y0_0, y0_1};
Main_Out  {out::x2_0, x2_1, y2_0, y2_1};

### Description of parallel pipelines for t=0
HDL pe10, 123, (x1_0, y1_0) = PE(x0_0, y0_0);
HDL pe11, 123, (x1_1, y1_1) = PE(x0_1, y0_1);

### Description of parallel pipelines for t=1
HDL pe20, 123, (x2_0, y2_0) = PE(x1_0, y1_0);
HDL pe21, 123, (x2_1, y2_1) = PE(x1_1, y1_1);
```



Minimal-Precision Computing - System Workflow



Weak Numerical Reproducibility on Minimal-Precision Computing

1. **The minimal-precision computing system => a black box**
   - Though ***different paths for execution may be used*** either to speed up computations and/or ensure energy-efficiency, required precision is guaranteed.
2. **Validation of the requested accuracy of the computation demanded by the user**
   - If the computation method can achieve the required result, any methods, any computation environments, and any computation conditions can be accepted.
   - ***No longer need to develop some reproducible variant(s)*** for each computation method or mathematical problem.
3. **Comparing with re-playable and re-traceable methods**
   - easier to adapt to different (parallel) architectures.
   - Existing methods and software ***for ensuring bit-level reproducibility are still able to contribute to ensure the demanded accuracy***, if such method relies on some accurate method.

Conclusion

- A new concept of weak numerical reproducibility
  the reproducibility, (up to a high probability) of the computation result with a certain accuracy demanded by the user.
  A systematic approach for it with a support of minimal-precision tuning and validation.

- The concept of weak numerical reproducibility covers most of the demands for reproducibility in computational sciences.
- Besides, if it has been realized with new hardware like FPGAs, the minimal-precision computing system can address the demands for accuracy, high-performance, and energy efficient computation as well.
- Future work is ***Demonstration of weak numerical reproducibility.***

Please see also:

Poster 134: Minimal-Precision Computing for High-Performance, Energy-Efficient, and Reliable Computations

ACKNOWLEDGMENT

- ***The authors would like to thank***
  - Dr. Kentaro Sano and Dr. Yiyu Tan from RIKEN CCS
  - Prof. Taisuke Boku and Dr. Norihisa Fujita from CCS, University of Tsukuba

- ***Partially supported by***
  - the European Unions Horizon 2020 research, innovation programme under the Marie Skodowska-Curie grant agreement via the Robust project No. 842528,
  - the Japan Society for the Promotion of Science (JSPS) KAKENHI Grant No. 19K20286,
  - Multidisciplinary Cooperative Research Program in CCS, University of Tsukuba.