DYNAMIC PRECISION NUMERICS USING A VARIABLE-PRECISION UNUM TYPE I HW COPROCESSOR

ARITH'26 | **BOCCO Andrea** | 11 June 2019

INTRODUCTION: STATE OF THE ART

- Variable Precision (VP) computing has been investigated to improve convergence of algorithms. It has been investigated in:
  - **Software** (SW): GMP[2] and MPFR[3]
    - Slow, they might not met requirements in high speed applications
  - **Hardware** (HW):
    - Kulisch[4] : large fixed point accumulator
    - Schulte and Swartzlander[5] : mantissas divided in multiple words

- None of the previous works show how to store efficiently VP Floating Point (FP) number in main memory
  - They support **IEEE 754 FP format** in main memory

[1] IEEE754-2008 2008. IEEE Standard for Floating-Point Arithmetic. IEEE 754-2008 https://doi.org/10.1109/IEEESTD.2008.4610935

[2] Torbjörn Granlund and the GMP development team. 2012. GNU MP: The GNU Multiple Precision Arithmetic Library. https://gmplib.org/

[3] Laurent Fousse, et al. MPFR: A Multiple precision Binary Floating-point Library with Correct Rounding. https://doi.org/10.1145/1236463.1236468

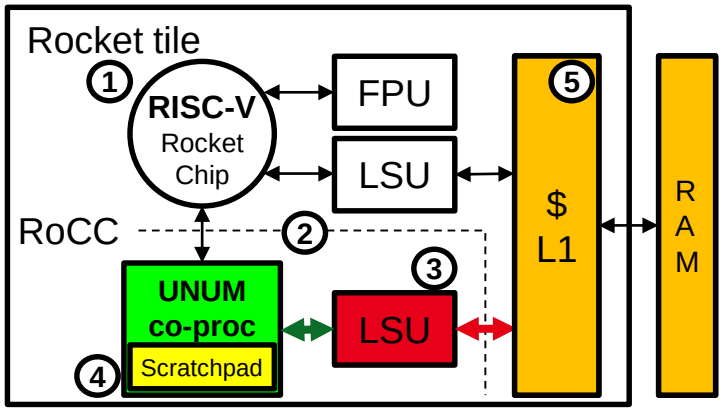
[4] Ulirich Kulisch. 2013. Computer arithmetic and validity: Theory, implementation, and applications

[5] M. J. Schulte and E. E. Swartzlander. 2000. A family of variable precision interval arithmetic processors. https://doi.org/10.1109/12.859535

INTRODUCTION: MY WORK

Our previous work[6]: a VP FP hardware accelerator:

- Supports the **UNUM** type I format in main memory
- Does computation internally with another (hardware friendly) FP format
- Supports **I**nterval **A**rithmetic (IA)



This work:

- Refines the UNUM type I FP format.
- Proposes a new VP FP architecture.
- Proposes a new programming model.
- Benchmarks our system.

[6] A. Bocco, Y. Durand, F. Dinechin, 2019, SMURF: Scalar Multiple-precision UNUM RISC-V Floating-point Accelerator for Scientific Computing.

OUTLINE

- Choice of the memory format: the UNUM type I
- Refinements on the UNUM type I FP format
- The adopted VP FP Architecture
- The programming model
- System benchmark: gauss elimination solver
- Conclusions

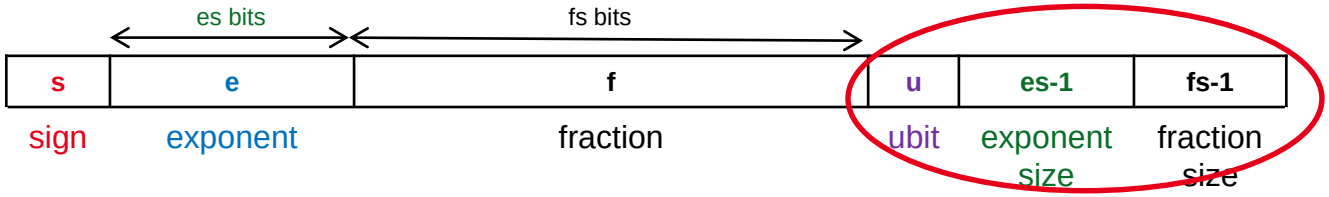
OUTLINE

- **Choice of the memory format: the UNUM type I**
- **Refinements on the UNUM type I FP format**
- **The adopted VP FP Architecture**
- **The programming model**
- **System benchmark: gauss elimination solver**
- **Conclusions**

CHOICE OF THE MEMORY FORMAT: THE UNUM TYPE I

We decided to use the UNUM type I FP format in main memory

- **It is 6 sub-fields self-descriptive FP format**



3 more that conventional IEEE 754 FP numbers

- **WHY?**
  - UNUM is a VP FP format
  - It self-encodes the exponent and fraction field lengths

However UNUM type I has some peculiarities to be fixed:

- How to organize UNUM arrays in main memory
- How to organize the UNUM fields in memory

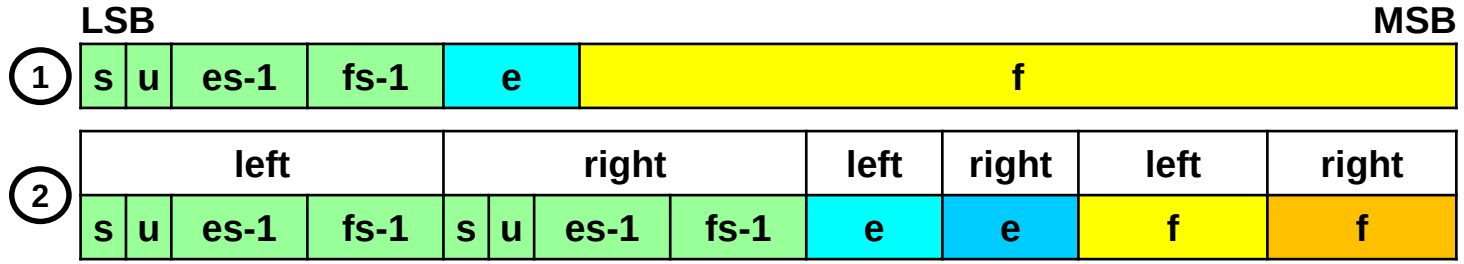
OUTLINE

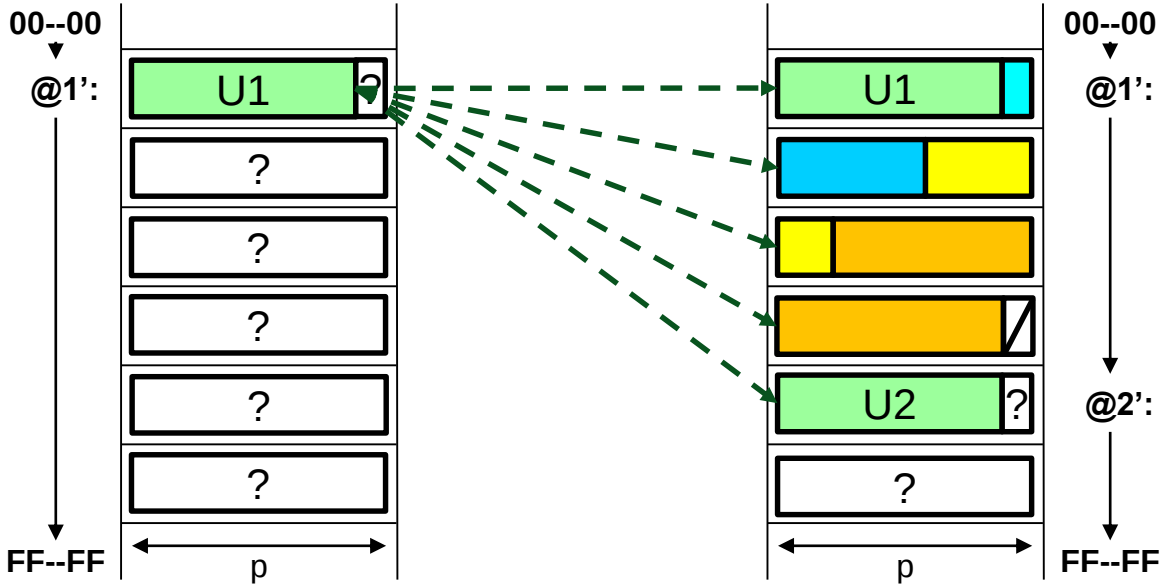
- Choice of the memory format: the UNUM type I
- **Refinements on the UNUM type I FP format**
- The adopted VP FP Architecture
- The programming model
- System benchmark: gauss elimination solver
- Conclusions

REFINEMENTS ON THE UNUM TYPE I FP FORMAT: - UNUM FIELD ORGANIZATION

For a UNUM/ubound which spans multiple addresses in main memory it is important to have the descriptor fields present in the lower addresses.

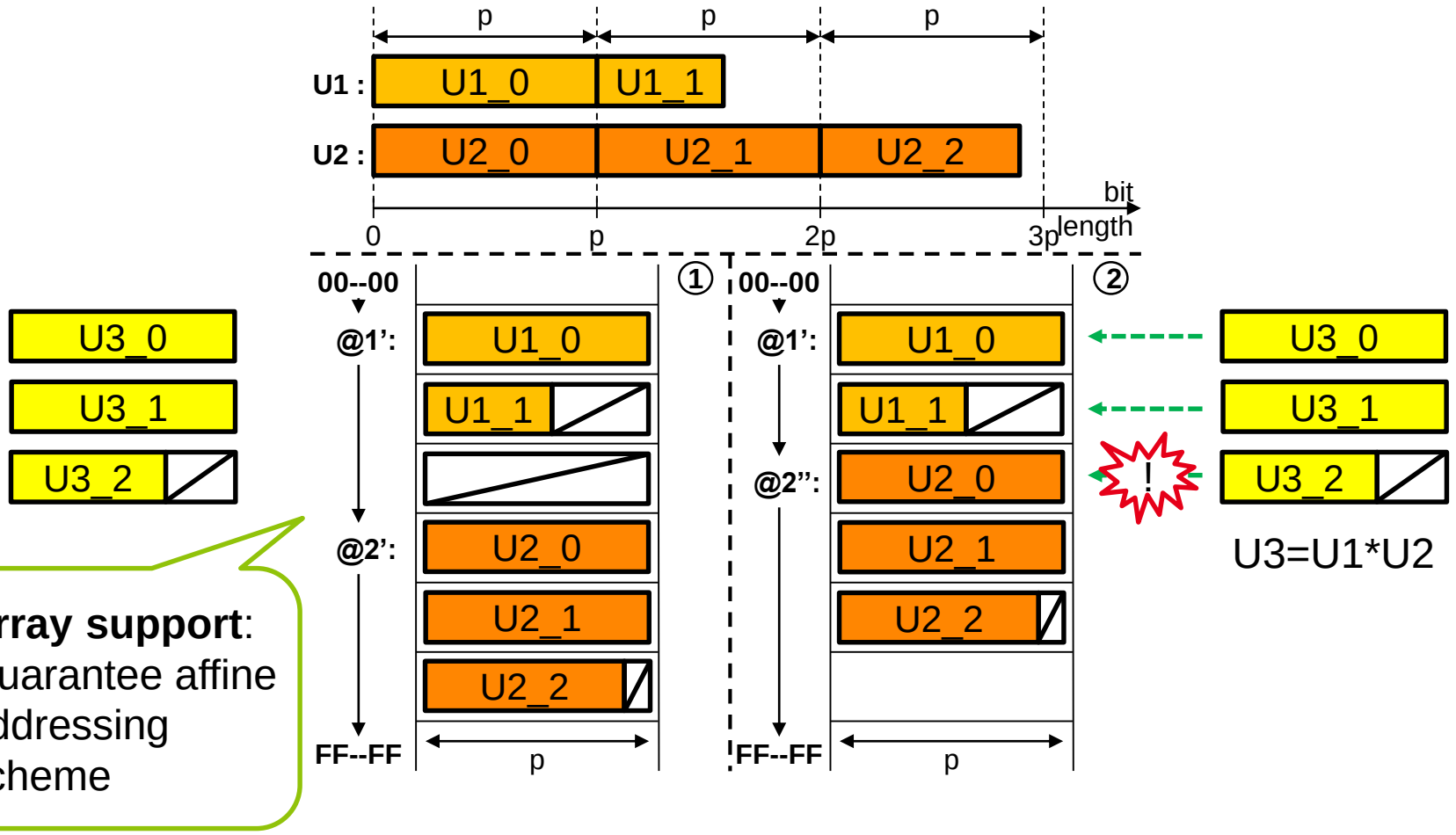
- We have re-organized the order of the fields for UNUM and ubound





REFINEMENTS ON THE UNUM TYPE I FP FORMAT: - UNUM ARRAY ORGANIZATION

Handling a two-element UNUM array on main memory with p bits parallelism



Array support: Guarantee affine addressing scheme

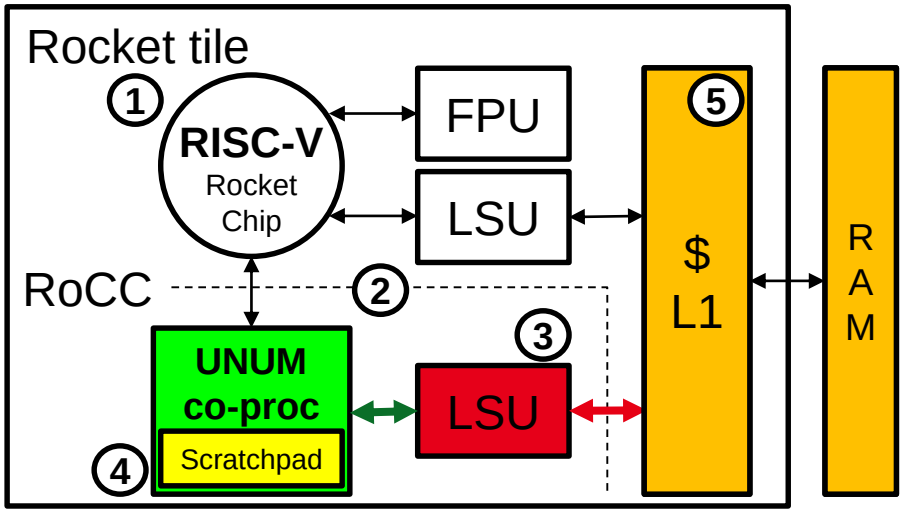
OUTLINE

- Choice of the memory format: the UNUM type I
- Refinements on the UNUM type I FP format
- **The adopted VP FP Architecture**
- The programming model
- System benchmark: gauss elimination solver
- Conclusions

THE ADOPTED VP FP ARCHITECTURE

- 1 integer register file (iRF): 32 integer general purpose register (GPR) + pc, in the main processor.
- 1 g-bound register file (gRF): 32 entries, in the co-processor.
- UNUMs/u-bounds are strictly considered as memory formats:
  - Load operations:
    - Load UNUMs/u-bounds from the main memory, and converts them into internal g-bounds.
  - Store operations:
    - Convert internal g-bounds (entries of the internal gRF) into u-bounds. Store the latter the main memory.
- The coprocessor internal parallelism is fixed to 64 bits
- Coprocessor's status registers:
  - DUE
  - SUE
  - MBB
  - WGP

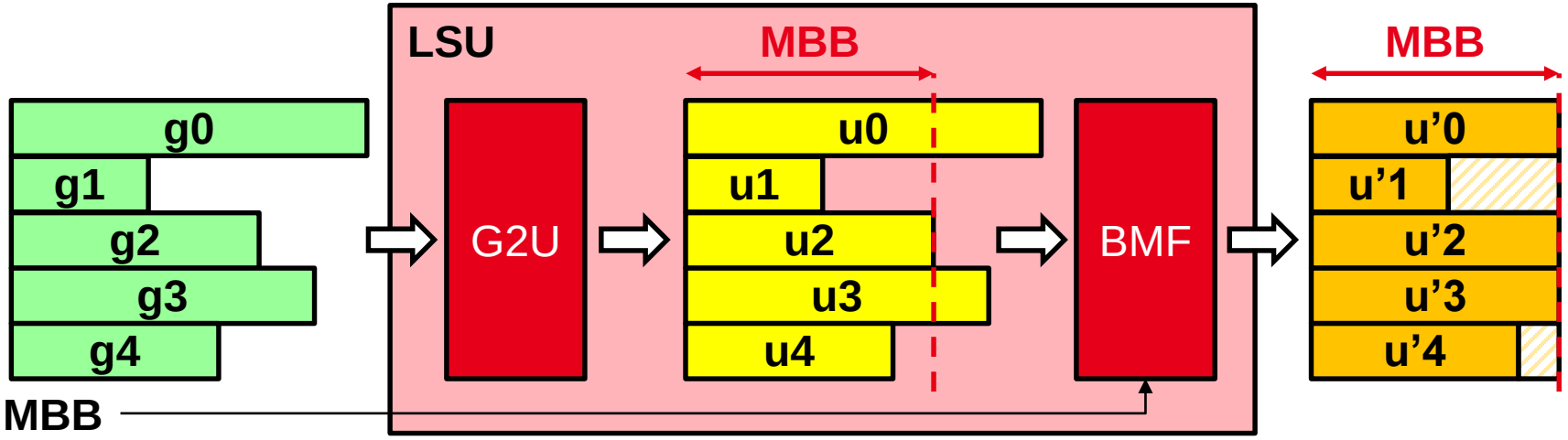
NEW!



THE MBB: MAXIMUM BYTE BUDGET

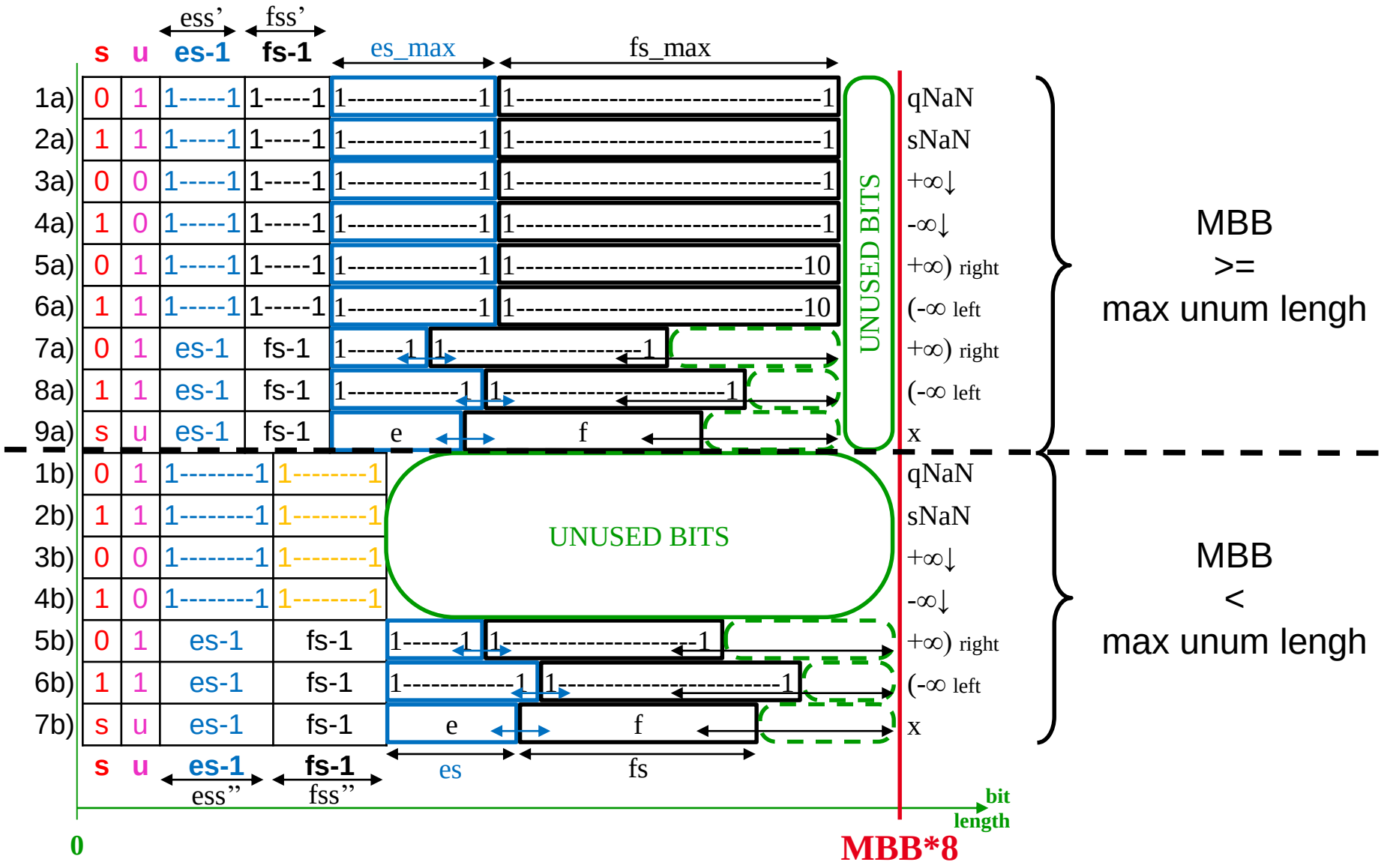
UNUM format is variable length (up to a maximum length)

- It is impossible to have compacted arrays having random access to its elements
- We define the **Maximum Byte Budget** (MBB) as the maximum length that a **UNUM** number can have in **main memory**



- The user can address **VP FP numbers** specifying their length with **Byte granularity.**

THE BMF: BOUNDED MEMORY FORMAT



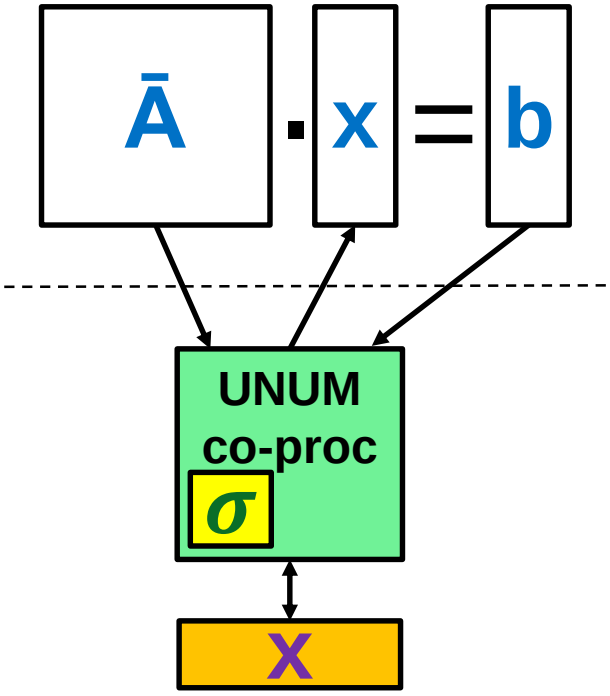
OUTLINE

- Choice of the memory format: the UNUM type I
- Refinements on the UNUM type I FP format
- The adopted VP FP Architecture
- **The programming model**
- System benchmark: gauss elimination solver
- Conclusions

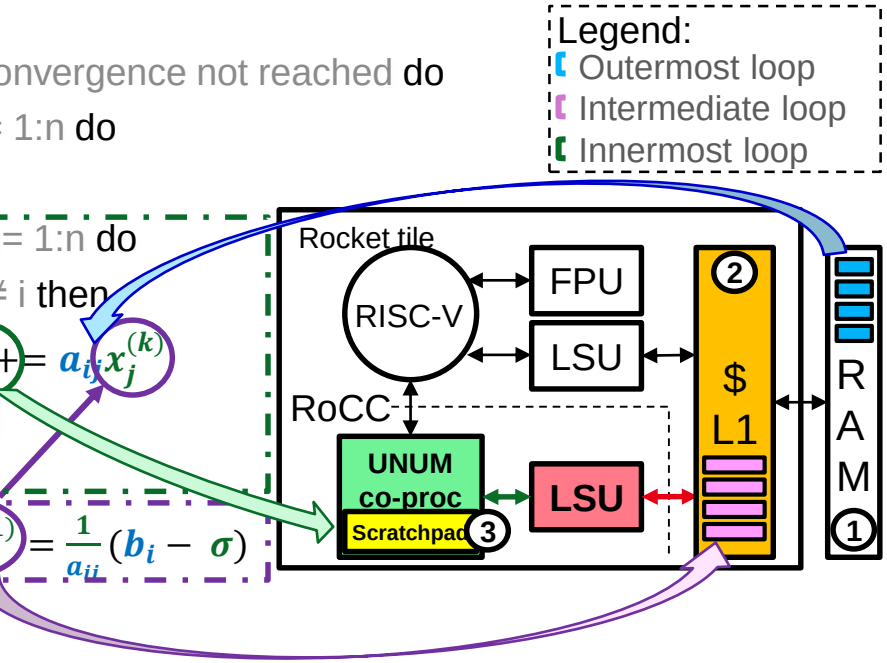
THE COPROCESSOR PROGRAMMING MODEL

Our hardware is best suited for VP kernels which exploit three different storage types:

- The external (main memory) storage
- The intermediate (L1 cache) storage
- The internal (register-level) storage




```
01: k = 0
02: while convergence not reached do
03:   for i := 1:n do
04:     σ =0
05:     for j := 1:n do
06:       if j ≠ i then
07:         σ += a_ij x_j^(k)
08:       end
09:     end
10:     x_i^(k+1) = 1/a_ii (b_i − σ)
11:   end
12:   k=k+1
13: end
```



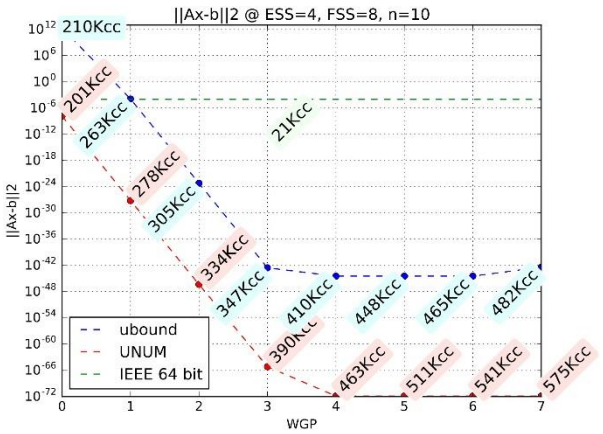
OUTLINE

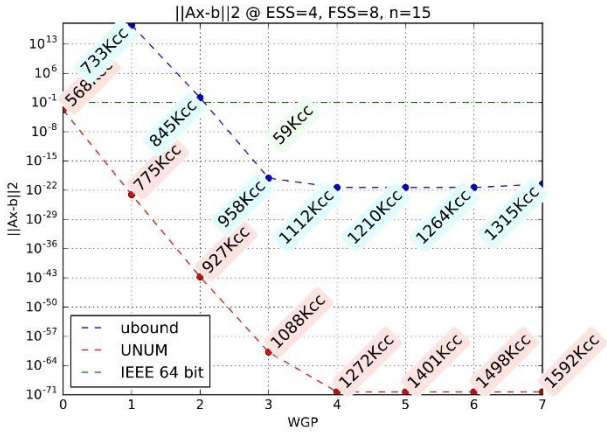
- Choice of the memory format: the UNUM type I
- Refinements on the UNUM type I FP format
- The adopted VP FP Architecture
- The programming model
- **System benchmark: gauss elimination solver**
- Conclusions

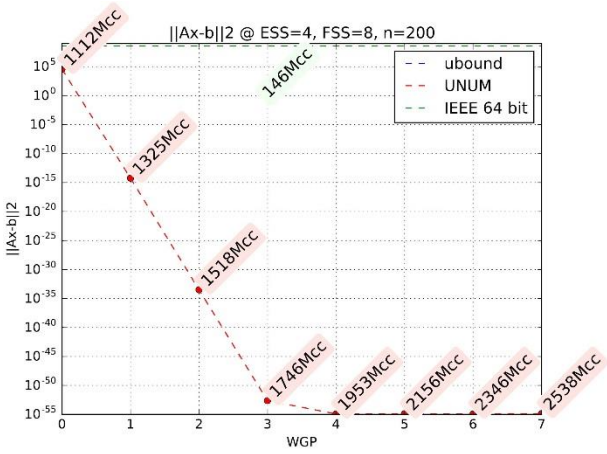
SYSTEM BENCHMARK: GAUSS ELIMINATION SOLVER

Our system benchmarked with a Gauss elimination solver, both in UNUM (scalar) and ubound (interval), showed:

- A gain of up to 65 decimal digits on IEEE double
- The result precision is constrained by the adopted precision in memory.
- Intervals do not converge always but it is useful in the computational error estimation (Ax-b).
- A speed up of 4-10x with respect to the MPFR software library







OUTLINE

- Choice of the memory format: the UNUM type I
- Refinements on the UNUM type I FP format
- The adopted VP FP Architecture
- The programming model
- System benchmark: gauss elimination solver
- **Conclusions**

CONCLUSIONS

This work proposes a Variable Precision (VP) Floating Point (FP) computing system, based on RISC-V, for high performance computing servers as an alternative to VP FP software routines.

- **It supports UNUM/ubound format in main memory**
  - It supports several Unum Environments: from (1,1) to (4,8), up to 256 mantissa bits
- **It supports a dedicated internal format in its Register File**
  - 32 intervals; Each interval endpoint can have up to 512 mantissa bits
- **With the adopted memory format (BMF) it supports VP FP in main memory**
  - User can decide the memory footprint of data with a Byte definition
- **With the adopted programming model, it is possible to extend VP FP high precision variables in main memory.**
  - The result precision can be significantly improved.
- **Its flops performances are better than software libraries (MPFR) and they stays within the same range of a regular fixed-precision IEEE FPU.**

THANK YOU FOR YOUR ATTENTION!

Contacts:
Andrea BOCCO
andrea.bocco@cea.fr

Leti, technology research institute
Commissariat à l'énergie atomique et aux énergies alternatives
Minatec Campus | 17 rue des Martyrs | 38054 Grenoble Cedex | France
www.leti.fr