SMURF: SCALAR MULTIPLE-PRECISION UNUM RISC-V FLOATING-POINT ACCELERATOR FOR SCIENTIFIC COMPUTING

CoNGA'19 | **BOCCO Andrea** | 13 March 2019

INTRODUCTION: STATE OF THE ART

- Variable Precision (VP) computing has been investigated to improve convergence of algorithms based on the IEEE 754 [1] standard :
  - 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

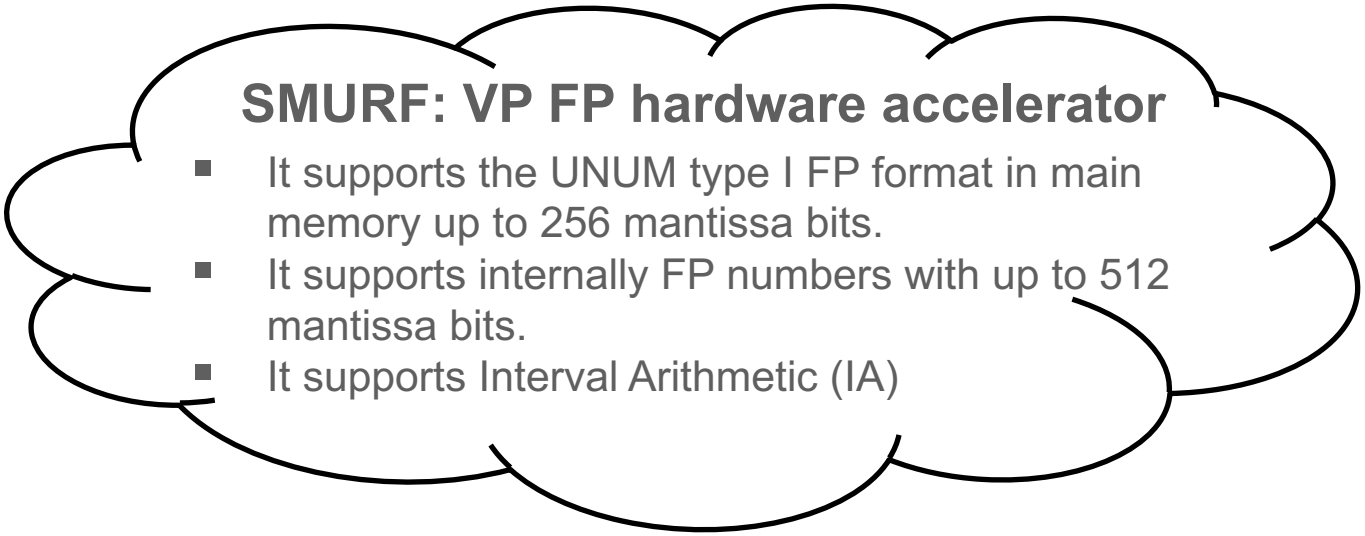
[4] Ulirich Kulisch. 2013. Computer arithmetic and validity: Theory, implementation, and applications

[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

In this work we present the **SMURF**:

- **S**calar
- **M**ultiple-precision
- **U**num
- **R**isc-V
- **F**loating-point Accelerator for Scientific Computing



OUTLINE

- Choice of the memory format: the UNUM type I
- The SMURF architecture
- The SMURF Instruction Set Architecture (ISA)
- The SMURF micro-architecture hardware implementation
- Validation, FPGA integration and Synthesis results
- An experimental software setup of the SMURF architecture.
- Conclusions

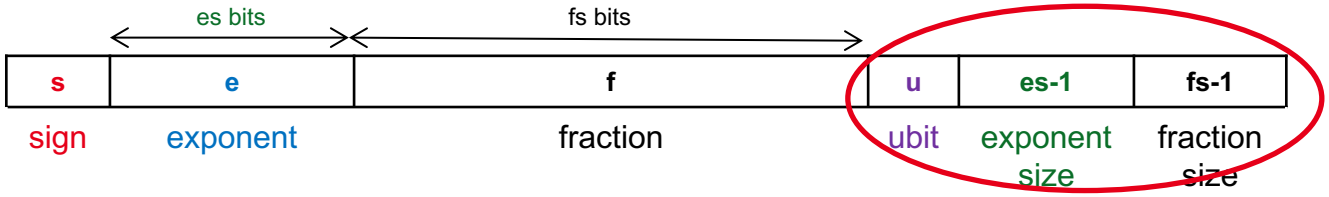
OUTLINE

- **Choice of the memory format: the UNUM type I**
- The SMURF architecture
- The SMURF Instruction Set Architecture (ISA)
- The SMURF micro-architecture hardware implementation
- Validation, FPGA integration and Synthesis results
- An experimental software setup of the SMURF architecture.
- 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

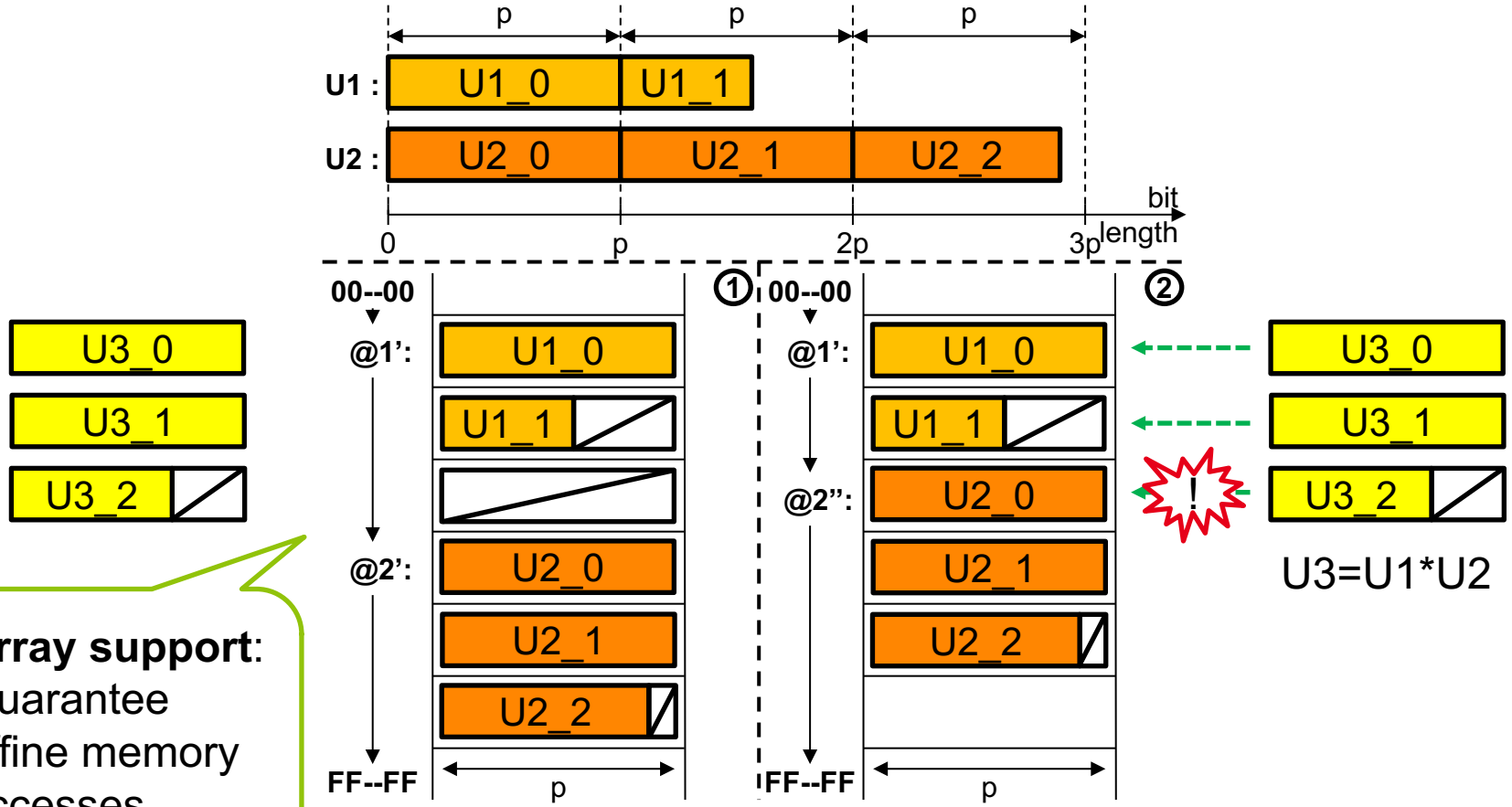
- **WHY?**
  - UNUM is a VP FP format
  - It self-handles the exponent and fraction field lengths

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

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

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

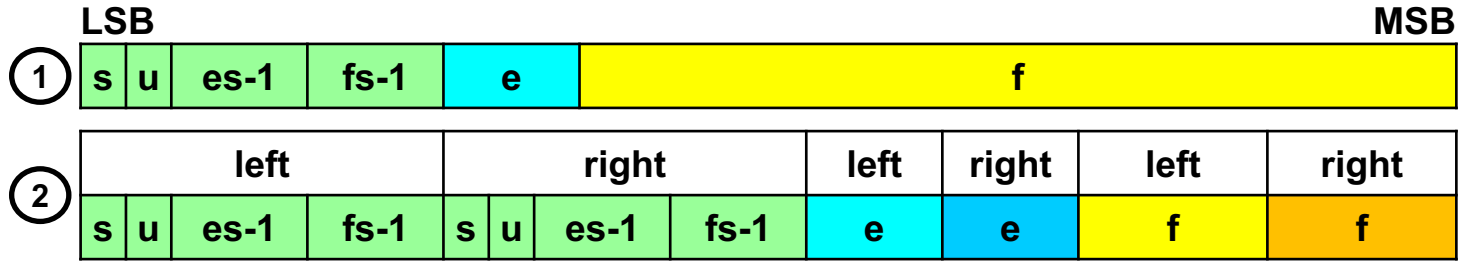


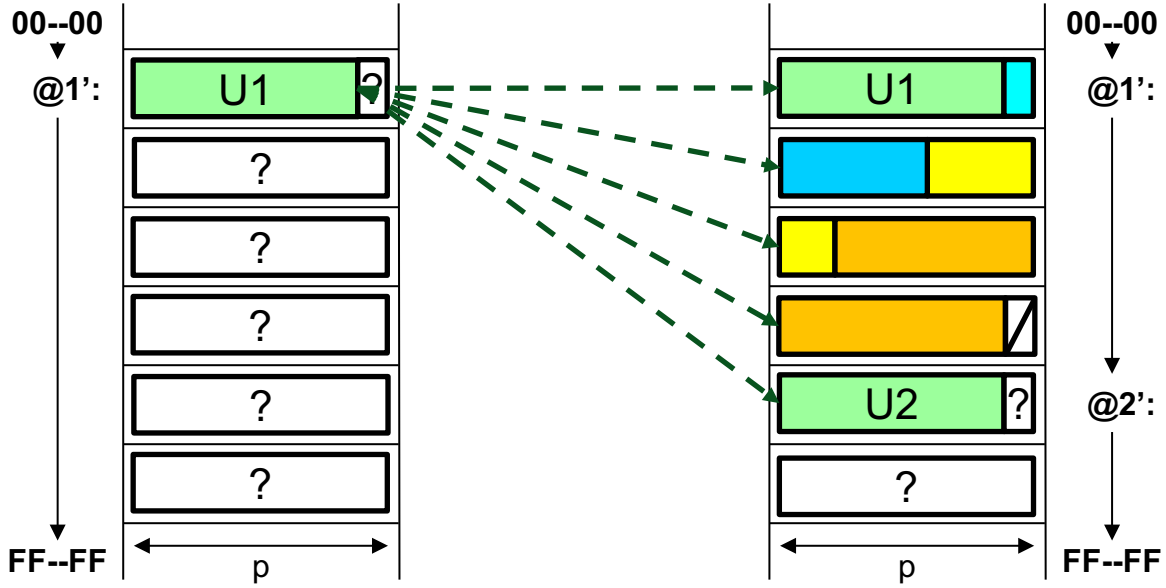
Array support: Guarantee affine memory accesses

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





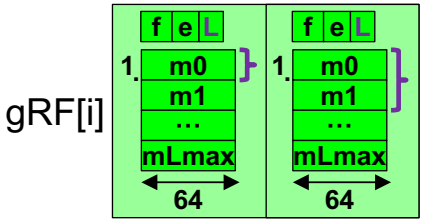
OUTLINE

- Choice of the memory format: the UNUM type I
- **The SMURF architecture**
- The SMURF Instruction Set Architecture (ISA)
- The SMURF micro-architecture hardware implementation
- Validation, FPGA integration and Synthesis results
- An experimental software setup of the SMURF architecture.
- Conclusions

THE SMURF ARCHITECTURE

Data organization:

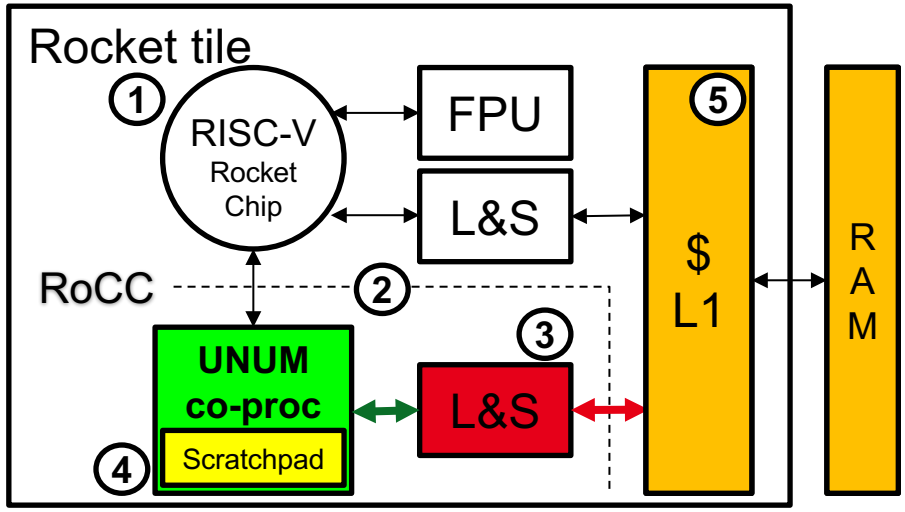
- **UNUMs/u-bounds** are strictly considered as memory formats
- Data inside the coprocessor **scratchpad** have dedicated FP format
  - Mantissa is normalized
  - Mantissa is divided in 8 chunks of 64 bits each
  - Exponent is explicit and signed



- **Conversions between formats are handled by a dedicated Load and Store unit (L&S)**

THE SMURF ARCHITECTURE

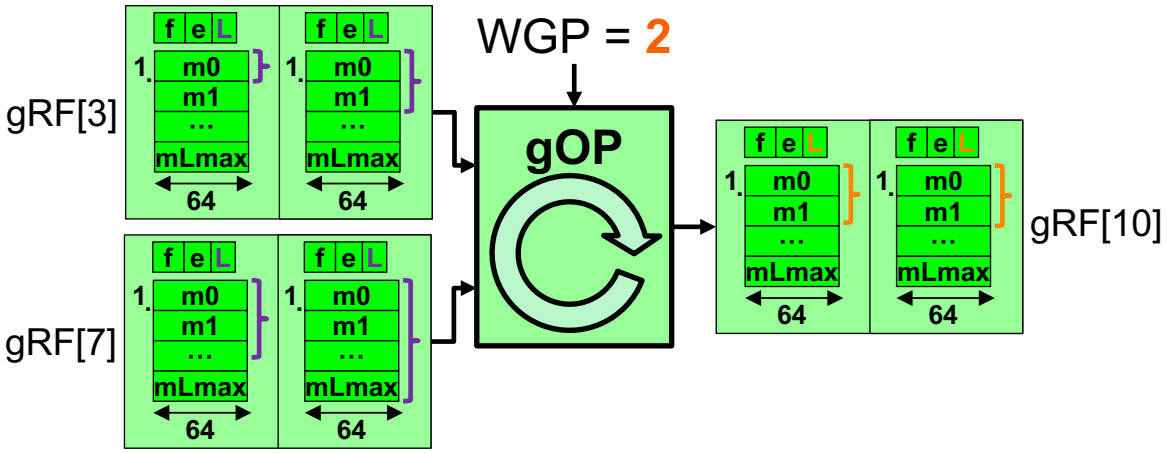
- 1 integer register file (iRF): 32 integer general purpose register (GPR) + pc, in the main processor.
- 1 g-bound register file (gRF): 32 entries, in the co-processor.
- UNUMs/u-bounds are strictly considered as memory formats:
  - Load operations:
    - Load UNUMs/u-bounds from the main memory, and 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
  - WGP



THE SMURF ARCHITECTURE

Mantissas are divided in chunks of 64 bits

- **The WGP status register defines the maximum mantissa chunks that an operator of the coprocessor can output.**



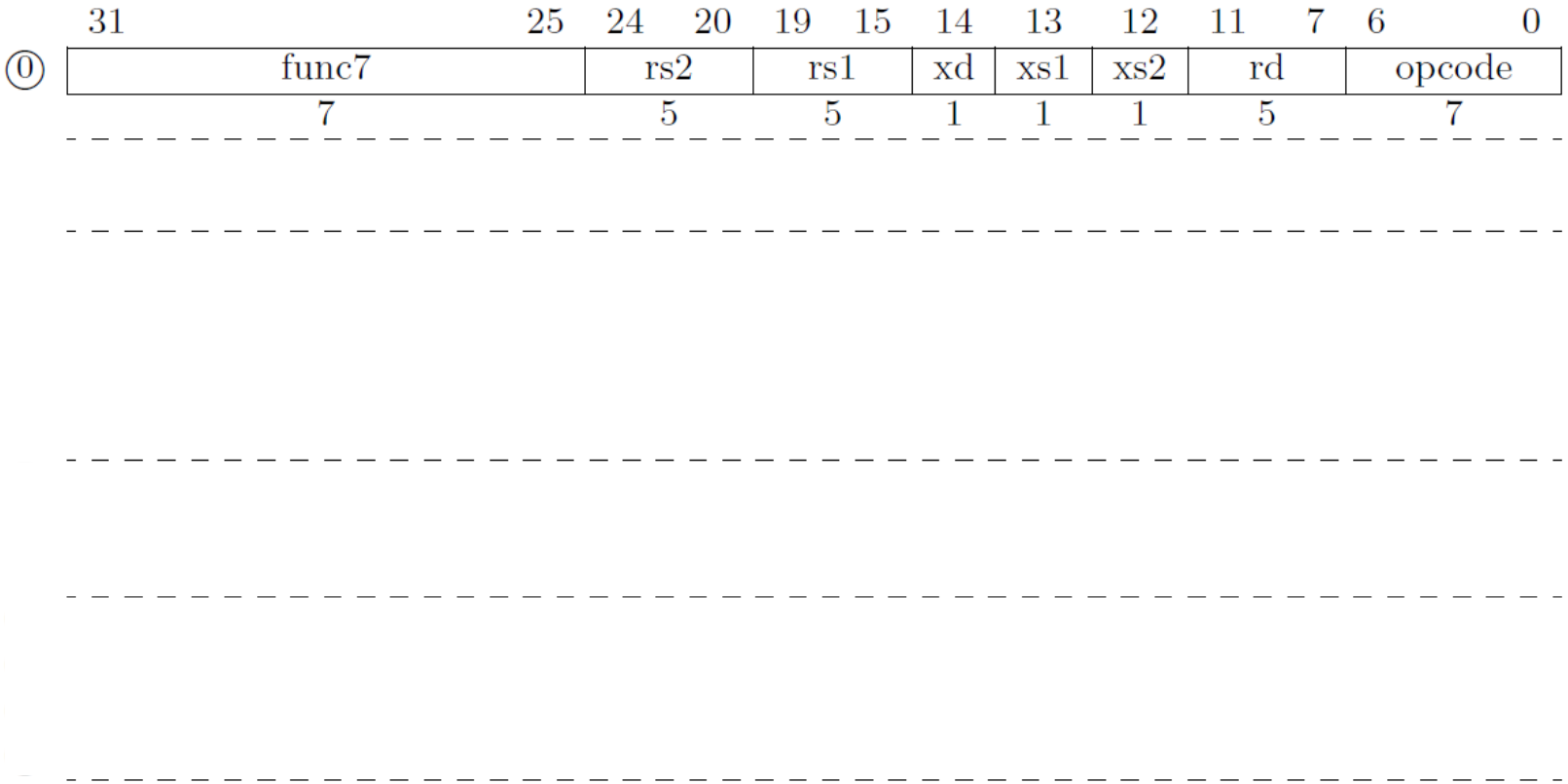
- **The user has a latency/precision tradeoff to exploit for his computation**

OUTLINE

- Choice of the memory format: the UNUM type I
- The SMURF architecture
- **The SMURF Instruction Set Architecture (ISA)**
- The SMURF micro-architecture hardware implementation
- Validation, FPGA integration and Synthesis results
- An experimental software setup of the SMURF architecture.
- Conclusions

THE SMURF INSTRUCTION SET ARCHITECTURE (ISA)

- **The SMURF ISA is divided in four groups:**



THE SMURF INSTRUCTION SET ARCHITECTURE (ISA)

- **The SMURF ISA is divided in four groups:**
  - Status Register settings

	31 – 25	24 – 20	19 – 15	14	13	12	11 – 7	6 – 0
⓪	func7	rs2	rs1	xd	xs1	xs2	rd	opcode
	7	5	5	1	1	1	5	7
①	SUSR	unused	Xs1	0	1	0	unused	OP-CUST
②	LUSR	unused	unused	1	0	0	Xd	OP-CUST

THE SMURF INSTRUCTION SET ARCHITECTURE (ISA)

- **The SMURF ISA is divided in four groups:**
  - Status Register settings
  - Mov operations

	31 25	24 20	19 15	14	13	12	11 7	6 0
⓪	func7	rs2	rs1	xd	xs1	xs2	rd	opcode
	7	5	5	1	1	1	5	7
①	SUSR	unused	Xs1	0	1	0	unused	OP-CUST
②	LUSR	unused	unused	1	0	0	Xd	OP-CUST
③	MOV_G2G	unused	gRs1	0	0	0	gRd	OP-CUST
④	MOVLL/MOVLR	unused	gRs1	0	0	0	gRd	OP-CUST
⑤	MOVRL/MOVRR	unused	gRs1	0	0	0	gRd	OP-CUST
⑥	MOV_X2G	#imm5	Xs1	0	1	0	gRd	OP-CUST
⑦	MOV_G2X	#imm5	gRs2	1	0	0	Xd	OP-CUST

THE SMURF INSTRUCTION SET ARCHITECTURE (ISA)

- **The SMURF ISA is divided in four groups:**
  - Status Register settings
  - Mov operations
  - Arithmetic operations

	31 25	24 20	19 15	14	13	12	11 7	6 0
⓪	func7	rs2	rs1	xd	xs1	xs2	rd	opcode
	7	5	5	1	1	1	5	7
①	SUSR	unused	Xs1	0	1	0	unused	OP-CUST
②	LUSR	unused	unused	1	0	0	Xd	OP-CUST
③	MOV_G2G	unused	gRs1	0	0	0	gRd	OP-CUST
④	MOVLL/MOVLR	unused	gRs1	0	0	0	gRd	OP-CUST
⑤	MOVRL/MOVRR	unused	gRs1	0	0	0	gRd	OP-CUST
⑥	MOV_X2G	#imm5	Xs1	0	1	0	gRd	OP-CUST
⑦	MOV_G2X	#imm5	gRs2	1	0	0	Xd	OP-CUST
⑧	GCMP	gRs2	gRs1	1	0	0	Xd	OP-CUST
⑨	GADD/GSUB/GMUL	gRs2	gRs1	0	0	0	gRd	OP-CUST
⑩	GGUESS/GRADIUS	unused	gRs1	0	0	0	gRd	OP-CUST

THE SMURF INSTRUCTION SET ARCHITECTURE (ISA)

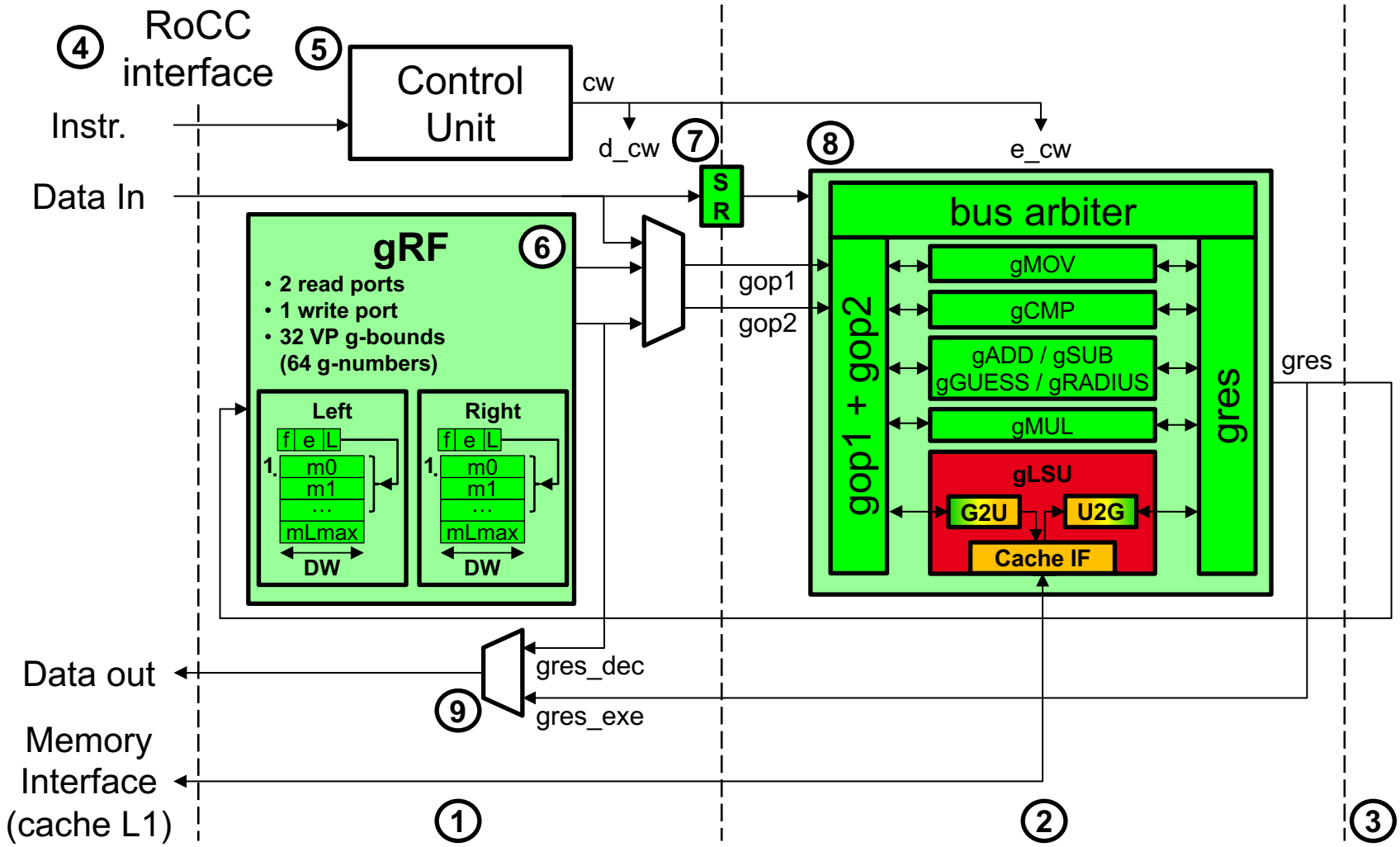
- **The SMURF ISA is divided in four groups:**
  - Status Register settings
  - Mov operations
  - Arithmetic operations
  - Load and Store operations

	31 25	24 20	19 15	14	13	12	11 7	6 0
⓪	func7	rs2	rs1	xd	xs1	xs2	rd	opcode
	7	5	5	1	1	1	5	7
①	SUSR	unused	Xs1	0	1	0	unused	OP-CUST
②	LUSR	unused	unused	1	0	0	Xd	OP-CUST
③	MOV_G2G	unused	gRs1	0	0	0	gRd	OP-CUST
④	MOVLL/MOVLR	unused	gRs1	0	0	0	gRd	OP-CUST
⑤	MOVRL/MOVRR	unused	gRs1	0	0	0	gRd	OP-CUST
⑥	MOV_X2G	#imm5	Xs1	0	1	0	gRd	OP-CUST
⑦	MOV_G2X	#imm5	gRs2	1	0	0	Xd	OP-CUST
⑧	GCMP	gRs2	gRs1	1	0	0	Xd	OP-CUST
⑨	GADD/GSUB/GMUL	gRs2	gRs1	0	0	0	gRd	OP-CUST
⑩	GGUESS/GRADIUS	unused	gRs1	0	0	0	gRd	OP-CUST
⑪	LDU/LDUB	unused	Xs1	0	1	0	gRd	OP-CUST
⑫	STUL/STUB	gRs2	Xs1	0	1	0	unused	OP-CUST
⑬	LDU_NEXT/LDUB_NEXT	gRs2	Xs1	1	1	0	Xd	OP-CUST
⑭	STUL_NEXT/STUB_NEXT	gRs2	Xs1	1	1	0	Xd	OP-CUST

OUTLINE

- Choice of the memory format: the UNUM type I
- The SMURF architecture
- The SMURF Instruction Set Architecture (ISA)
- **The SMURF micro-architecture hardware implementation**
- Validation, FPGA integration and Synthesis results
- An experimental software setup of the SMURF architecture.
- Conclusions

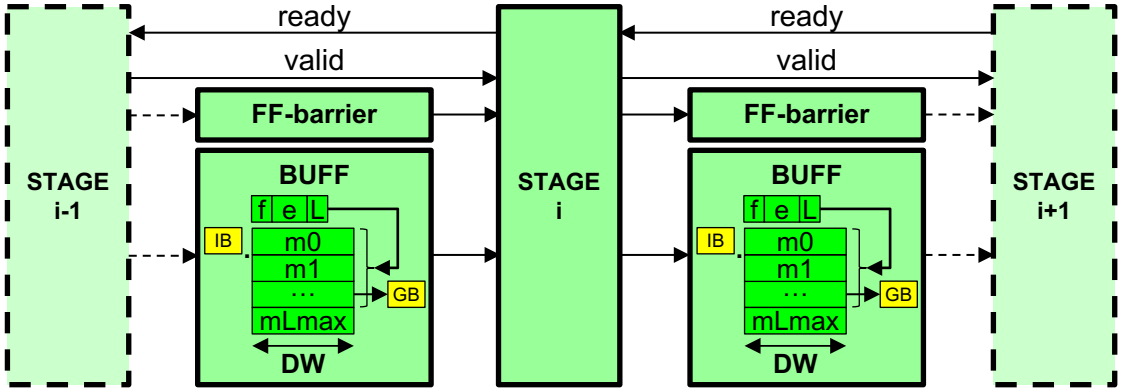
THE SMURF MICRO-ARCHITECTURE HARDWARE IMPLEMENTATION



THE SMURF MICRO-ARCHITECTURE HARDWARE IMPLEMENTATION

Internal operators pipelines are divided in "macro-stages":

- **A "macro-stage" is the logic which iterates on mantissas chunks:**
  - It implements a basic mantissa operation: mov, lzc, add, sub, shift, mul, …
  - It reads the input mantissa from an input buffer (BUFF)
  - It computes the resulting mantissa chunk-wise (64 bits per clock cycle)
    - **Variable latency**
  - It writes the computed result in an output buffer (BUFF)
  - It is synchronized with others macro-stages through a ready-valid protocol
    - **Stop and Wait protocol**



- **In this way, the complexity to do operations on multiple mantissas chunks is pushed inside each macro stage.**

OUTLINE

- Choice of the memory format: the UNUM type I
- The SMURF architecture
- The SMURF Instruction Set Architecture (ISA)
- The SMURF micro-architecture hardware implementation
- **Validation, FPGA integration and Synthesis results**
- An experimental software setup of the SMURF architecture.
- Conclusions

VALIDATION, FPGA INTEGRATION AND SYNTHESIS RESULTS

Validation: SMURF sub-components are validated against 50 millions pseudo-generated input vectors

FPGA integration: SMURF is integrated in a Xilinx Virtex 7 FPGA board @50 MHz

ASIC synthesis: working frequency 600MHz

① Unit	② stg	Area (μm^2) ③	Area (μm^2 buff) ④	Total Power (mW) ⑤	Total Power (mW buff) ⑥
Rocket_tile	-	1553 (100%)	n.a. (n.a.)	95.2 (100%)	n.a. (n.a.)
–RISC-V	-	23.09 (1.5%)	n.a. (n.a.)	0.78 (0.8%)	n.a. (n.a.)
–64bit_fpu	-	**53.1 (3.4%)**	n.a. (n.a.)	**1.43 (1.5%)**	n.a. (n.a.)
–d_cache	-	487.6 (31.4%)	n.a. (n.a.)	12.72 (13.4%)	n.a. (n.a.)
–i_cache	-	425.6 (27.4%)	n.a. (n.a.)	8.51 (8.9%)	n.a. (n.a.)
–if/periph	-	102.3 (6.6%)	n.a. (n.a.)	3.83 (4.0%)	n.a. (n.a.)
–coproc	3	**461 (29.7%)**	**307 (66.5%)**	**17.0 (17.9%)**	**4.2 (24.7%)**
—s_decode	-	131.1 (8.4%)	130.3 (99.3%)	2.29 (2.4%)	2.16 (94.5%)
——gRF	-	130.8 (8.4%)	130.3 (99.6%)	2.27 (2.4%)	2.16 (95.1%)
—s_execute	-	327.5 (21.1%)	176.5 (53.9%)	13.6 (14.3%)	2.03 (14.9%)
——gMOV	1	4.242 (0.3%)	3.061 (72.2%)	0.17 (0.2%)	0.02 (14.1%)
——gCMP	3	30.73 (2.0%)	24.97 (81.3%)	0.83 (0.9%)	0.25 (29.9%)
——gADD	11	66.77 (4.3%)	43.61 (65.3%)	2.3 (2.4%)	0.42 (18.3%)
——gMUL	10	143.4 (9.2%)	60.06 (41.9%)	6.76 (7.1%)	0.98 (14.4%)
——gLSU	3	81.35 (5.2%)	44.77 (55.0%)	3.5 (3.7%)	0.36 (10.4%)
———LD	3	29.96 (1.9%)	16.95 (56.6%)	1.51 (1.6%)	0.14 (9.3%)
————load	3	15.47 (1.0%)	9.344 (60.4%)	0.86 (0.9%)	0.08 (9.6%)
————u2g	4	10.53 (0.7%)	4.579 (43.5%)	0.56 (0.6%)	0.04 (6.6%)
———ST	3	51.03 (3.3%)	27.81 (54.5%)	1.95 (2.0%)	0.22 (11.5%)
————g2u	8	27.35 (1.8%)	16.96 (62.0%)	1.07 (1.1%)	0.11 (9.8%)
————store	4	17.29 (1.1%)	10.84 (62.7%)	0.6 (0.6%)	0.09 (14.9%)

VALIDATION, FPGA INTEGRATION AND SYNTHESIS RESULTS

Validation: SMURF sub-components are validated against 50 millions pseudo-generated input vectors

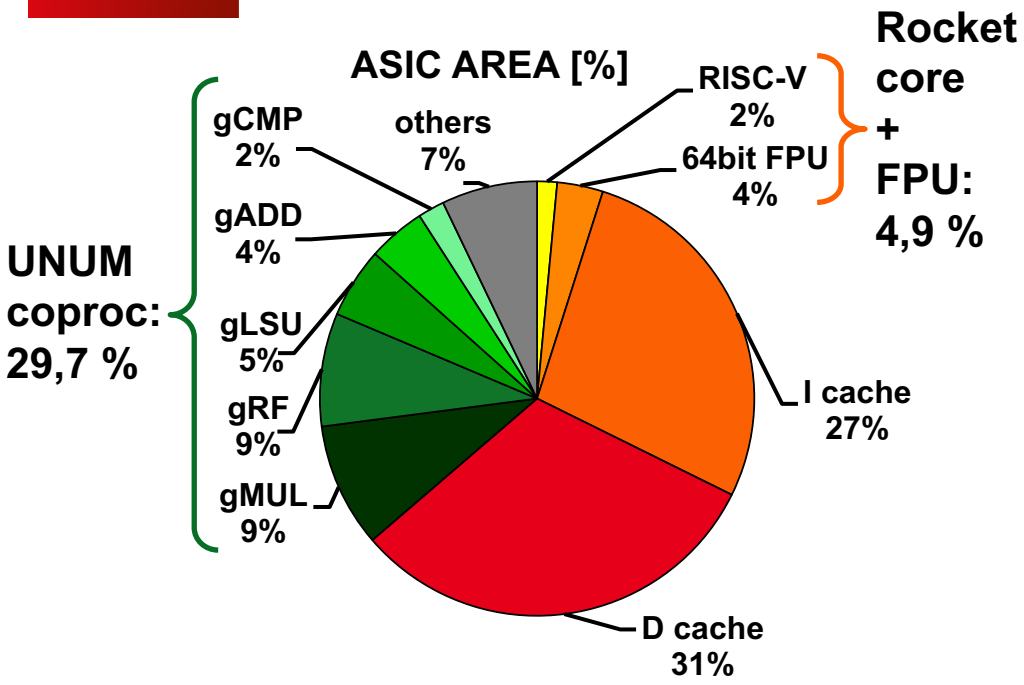
FPGA integration: SMURF is integrated in a Xilinx Virtex 7 FPGA board @50 MHz

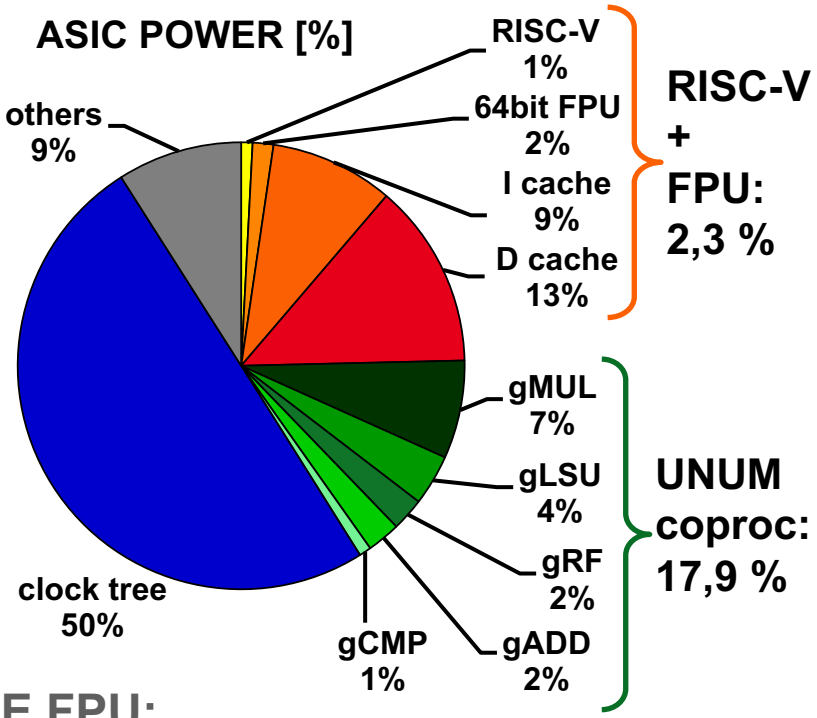
ASIC synthesis: working frequency 600MHz

① Unit	② stg	Area (μm^2) ③	Area (μm^2 buff) ④	Total Power (mW) ⑤	Total Power (mW buff) ⑥
Rocket_tile	-	1553 (100%)	n.a. (n.a.)	95.2 (100%)	n.a. (n.a.)
–RISC-V	-	23.09 (1.5%)	n.a. (n.a.)	0.78 (0.8%)	n.a. (n.a.)
–**64bit_fpu**	-	**53.1 (3.4%)**	n.a. (n.a.)	**1.43 (1.5%)**	n.a. (n.a.)
–d_cache	-	487.6 (31.4%)	n.a. (n.a.)	12.72 (13.4%)	n.a. (n.a.)
–i_cache	-	425.6 (27.4%)	n.a. (n.a.)	8.51 (8.9%)	n.a. (n.a.)
–if/periph	-	102.3 (6.6%)	n.a. (n.a.)	3.83 (4.0%)	n.a. (n.a.)
–**coproc**	3	**461 (29.7%)**	**307 (66.5%)**	**17.0 (17.9%)**	**4.2 (24.7%)**
——s_decod					4.5%)
——gRF					%)
——s_exe					%)
——gM					%)
——gC					%)
——gA					%)
——gM					%)
——gLS					%)
——LI					)
——l					)
——u2g					3.6%)
——ST	3	51.03 (3.3%)	27.81 (54.5%)	1.95 (2.0%)	0.22 (11.5%)
——g2u	8	27.35 (1.8%)	16.96 (62.0%)	1.07 (1.1%)	0.11 (9.8%)
——store	4	17.29 (1.1%)	10.84 (62.7%)	0.6 (0.6%)	0.09 (14.9%)

- Buffers have an important contribution in area (67%) and power (25%)
- We can reduce the buffer impact optimizing the design

VALIDATION, FPGA INTEGRATION AND SYNTHESIS RESULTS





The SMURF coprocessor vs. a standard IEEE FPU:

- 9 times bigger
- 12 times more energy consuming
- Flops performances have the same order of magnitude

➢ This unit is meant to be used only when VP is needed: the rest of the time it behaves as dark silicon

OUTLINE

- Choice of the memory format: the UNUM type I
- The SMURF architecture
- The SMURF Instruction Set Architecture (ISA)
- The SMURF micro-architecture hardware implementation
- Validation, FPGA integration and Synthesis results
- **An experimental software setup of the SMURF architecture.**
- Conclusions

AN EXPERIMENTAL SOFTWARE SETUP

We measured the SMURF **flop performance** on a **Newton-Raphson** (NR) division example:

The latency of a NR iteration

- Using the **SMURF** it varies between 33 and 362 clock cycles:
  54 – 5 Mflops @ 600MHz
- Using **MPFR** with 512-bits of accuracy
  1 Mflop @ 600MHz
- Using the **IEEE FPU** it takes 3 clock cycles:
  200 Mflops @ 600MHz

- **+5x performance with respect to MPFR**
- **-4x performance with respect to 64bits IEEE FPU**

```
1  #include "unum_rocc.h"
2  void nr_division(void *res, void *op1,
       void *op2, int wgp){
3    //Compute the Newton-Raphson reciprocal
4    //(1/op2) = Rn*(2-(D*Rn))
5    LDUB(G1, op2);
6    LDUB(G5, op1);
7    reciprocal_approx(G0, G1);
8    float2gbound(G2, 2.0, 4, 8);
9    int i;
10   for (i=0;i<wgp;i++){
11     GMUL(G3, G1, G0); //(D*Rn)
12     GSUB(G4, G2, G3); //2-(D*Rn)
13     GMUL(G0, G0, G4); //Rn*(2-(D*Rn))
14   }
15   //multiply the reciprocal with op1
16   GMUL(G5, G5, G0); //op1*(1/op2)
17   STUB(res, G5);
18 }
19 int main()
20 {
21   ubound_4_8_t op1, op2, res;
22   set_due(4,8);
23   set_sue(4,8);
24   set_wgp(7);
25   float2gbound(G10, 2.0, 4, 8);
26   float2gbound(G11, 3.0, 4, 8);
27   STUB(&op1, G10);
28   STUB(&op2, G11);
29   nr_division(&res, &op1, &op2, 7);
30   LDUB(G12, &res);
31   print_gbound(G12);
32   return 0;
33 }
```

AN EXPERIMENTAL SOFTWARE SETUP

We measured the SMURF **flop performance** on a **Newton-Raphson** (NR) division example:

The latency of a NR iteration

- Using the **SMURF** it varies between 33 and 362 clock cycles:
  54 – 5 Mflops
- Using **MPFR** v
  1 Mflop @ 60
- Using the **IEE**
  3 clock cycles
  200 Mflops @

Work in progress:

- We are working on a real programming model to program real VP FP kernels

- **+5x performance with respect to MPFR**
- **-4x performance with respect to 64bits IEEE FPU**

```
1  #include "unum_rocc.h"
2  void nr_division(void *res, void *op1,
       void *op2, int wgp){
3    //Compute the Newton-Raphson reciprocal
4    //(1/op2) = Rn*(2-(D*Rn))
5    LDUB(G1, op2);
6    LDUB(G5, op1);
7    reciprocal_approx(G0, G1);
8    float2gbound(G2, 2.0, 4, 8);
9    int i;
10   for (i=0;i<wgp;i++){
                                G0); //(D*Rn)
                                3); //2-(D*Rn)
                                ); //Rn*(2-(D*Rn))

                               ciprocal with op1
                                //op1*(1/op2)

                               , op2, res;
23   set_sue(4,8);
24   set_wgp(7);
25   float2gbound(G10, 2.0, 4, 8);
26   float2gbound(G11, 3.0, 4, 8);
27   STUB(&op1, G10);
28   STUB(&op2, G11);
29   nr_division(&res, &op1, &op2, 7);
30   LDUB(G12, &res);
31   print_gbound(G12);
32   return 0;
33 }
```

OUTLINE

- Choice of the memory format: the UNUM type I
- The SMURF architecture
- The SMURF Instruction Set Architecture (ISA)
- The SMURF micro-architecture hardware implementation
- Validation, FPGA integration and Synthesis results
- An experimental software setup of the SMURF architecture.
- **Conclusions**

CONCLUSIONS

This work proposes a Variable Precision (VP) Floating Point (FP) accelerator (SMURF) based on RISC-V ISA (Instruction Set Architecture) for high performance computing servers as an alternative to VP FP software routines.

- **SMURF is implemented as a RISC-V coprocessor**
- **SMURF supports UNUM/ubound format in main memory**
  - It supports several Unum Environments: from (1,1) to (4,8), up to 256 mantissa bits
- **SMURF supports a dedicated internal format in its Register File**
  - 32 intervals; Each interval endpoint can have up to 512 mantissa bits
- **SMURF is pipelined with an internal parallelism fixed at 64 bit**
  - Mantissas are threated iteratively 64 bits per clock cycle
- **SMURF has 9x area and 12x power with respect a standard IEEE 64bit FPU**
  - The SMURF behaves as **dark silicon** when is not used
- **SMURF 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

INSTITUT CARNOT CEA LETI