A Floating-Point Unit Using Stochastic Arithmetic Compliant With The IEEE-754 Standard

Roselyne CHOTIN and Habib MEHREZ

LIP6/ASIM Laboratory, UPMC
Tour 65/66 bureau 401
4 place Jussieu
F-75252 Paris cedex 05 - France
Email : Roselyne.Chotin@lip6.fr and Habib.Mehrez@lip6.fr

ABSTRACT

In this paper we present CESTAC, a method to control round-off errors in floating-point scientific computation, based on stochastic arithmetic. The real time use of this method suffers from bottleneck software calculation. This paper gives a hardware alternative that would significantly accelerate the computation. The proposed hardware architecture has two parts : a standard floating-point unit (FPU) and a unit dedicated to the control of round-off errors.

1. INTRODUCTION

Arithmetic operations in scientific computations increase with computers speed. But using floating-point numbers garbles the results of these operations, due to a round-off error propagation. At the end of the computation, the result can be totally different of the expected result. The aim of the stochastic arithmetic is to estimate the loss of accuracy of elementary operations.
Different methods to control this round-off error, such as interval arithmetic, stochastic arithmetic, variable-precision arithmetic, already exist in software implementation. Only variable-precision arithmetic has already been implemented in hardware [1][2].
The aim of this paper is to propose a hardware implementation of the stochastic arithmetic. A software implementation already exists but costs a lot of CPU load. The proposed architecture accepts standard floating-point operators and has a block that performs the stochastic arithmetic. Section 2 is dedicated to a short review of the stochastic arithmetic. Section 3 details our hardware implementation. Section 4 presents the performance of the complete hardware architecture. The conclusion is given in the section 5.

This paper presents the hardware point of view of the global stochastic arithmetic project. We acknowledge the collaboration of J. Vignes, J.-M. Chesneaux and J.-L. Lamotte for the complete definition of the theoretical method aspects.

2. THE STOCHASTIC ARITHMETIC

The CESTAC method was developed by M. La Porte and J. Vignes [3][4][5]. The validity and the efficiency of the method were extensively proven in [6][7]. The idea behind this method is to execute the same computation several times with different round-off error propagation. At the end of the computation, several results are available, each with a different round-off error propagation. CESTAC gives the number of significant digits of the result and the value of the true result.
A stochastic number has several floating-point components. A stochastic operation is performed by the execution of the corresponding floating-point operation on each component of the stochastic operands. Each result is rounded by using the random arithmetic. So the least significant bit of the mantissa is randomly disrupted. The theory of the CESTAC method resides on the study of the propagation of these disruptions. The CESTAC method is detailed in the following.

2.1. Random arithmetic

Each result in exact arithmetic is surrounded by two consecutive floating-point numbers, one rounded down F^- and the second rounded up F^+, each of them representing the exact arithmetical result. The random round-off consists in randomly choosing either F^- or F^+ as result with the same probability of $\frac{1}{2}$.

2.2. Estimation of the accuracy

The result $\bar{R}$ of the N executions of the same computation, is the average of each result (R_i) obtained with the CESTAC method :

$$\bar{R} = \frac{1}{N} \sum_{i=1}^{N} R_i$$

The number of significant digits of the true result is defined in [4] by :

$$C_{\bar{R}} = \log_{10} \frac{\sqrt{N}.|\bar{R}|}{\sigma.\tau_\beta} \quad (1)$$

Where σ is the standard duration :

$$\sigma^2 = \frac{1}{N-1} \sum_{i=1}^{N} (R_i - \bar{R})^2$$

And τ_β is the value of the Student's distribution for $N-1$ degrees of freedom and a probability level $1-\beta$. In practice we take $N = 3$, and then $\tau_\beta = 4.303$. Increasing the value of N is useless, because to increase of one the number of significant digits, it would be necessary to multiply the number of executions by 100 for the same probability.

2.3. Stochastic zero

Mathematically a result of a floating-point computation can be null, but due to the round-off error propagation it is not. So the concept of stochastic zero was introduced in [8]. A result is a stochastic zero if all the results of the N executions (with different round-off error propagation) are null or if the number of significant digits is negative. So a result is a stochastic zero if one of these conditions is true :

$$\begin{array}{ll} 1. & \forall\, i,\ R_i = 0, \\ 2. & C_{\bar{R}} \leq 0 \end{array} \quad (2)$$

2.4. CADNA software

A software implementation of the CESTAC method has been developed with a library named CADNA (*Control of Accuracy and Debugging for Numerical Applications*). This library permits to estimate the round-off error in scientific computations and a real debug [9][10].

Computation example

A program calculates the roots of the second degree equation : $0.3x^2 - 2.1x + 3.673 = 0$. This equation has a double root $x = 3.5$. When the program is executed on computer, the discriminant is negative and then there are two conjugated complex roots. The CADNA library detects that the discriminant is a stochastic zero and then the equation has a double root that is $x = 3.5$.

So to perform a computation with the CESTAC method, we have to :

- execute three times the floating-point operation on each component of the stochastic operands with a different round-off
- choose randomly the round-off
- calculate the number of significant digits
- detect the stochastic zeros
- calculate the average result

3. HARDWARE IMPLEMENTATION

We have seen how to perform a computation with the CESTAC method. So all the steps have to be implemented.

3.1. Floating-point unit

A library of floating-point operators has to be developed according to the IEEE-754 standard. In particular all operators should be able to perform operation on special numbers such as infinity or NaNs. Furthermore these operators should detect all the standard floating-point exceptions.
All operators are developed as parameterizable generators. Their parameters are the width (32 bits, 64 bits or other), the number of pipeline stages (1, 2 or 3) and some architectural parameters that depend of the operator.

3.2. Random round-off

The random round-off was definite in 2.1. This pseudo-random number generator is implemented with a Linear Feedback Shift Register (LFSR) that generates a sequence of pseudo-random 32 bits numbers. The least significant bit of the generated 32 bits number is the round-off. So this generator is used to choose either R_i^- or R_i^+ $(i = 1, 2)$ where i is the number of the running execution. For $i = 3$, the round-off is the opposite of round-off chosen for $i = 2$.

3.3. Number of significant digits

The number of significant digits is defined in (1). A hardware implementation requires a simplification of this formula. The number of significant digit is the number of common digits to the different R_i. The number of significant digits is therefore the position of the first no null digit of the different results R_i. Two close numbers could have no common digits (0.9999 and 1.0000 for example), so we prefer to search the position of the first no null digit on the absolute value of the R_i differences. So other method to calculate the number of significant bits is done by :

1. calculation of the distances $d_1 = |R_1 - R_2|$, $d_2 = |R_1 - R_3|$, $d_3 = |R_3 - R_2|$

Cycle	FPU			Cestac
	stage 1	stage 2	stage 3	
1	$A_1\ Op\ B_1$			- store A_1, B_1
2	$A_2\ Op\ B_2$	$A_1\ Op\ B_1$		- store A_2, B_2
3	$A_3\ Op\ B_3$	$A_2\ Op\ B_2$	$A_1\ Op\ B_1$	- store A_3, B_3 and R_1 - $NSB(A)$
4	$C_1\ Op\ D_1$	$A_3\ Op\ B_3$	$A_2\ Op\ B_2$	- store C_1, D_1 and R_2 - $NSB(B)$
5	$C_2\ Op\ D_2$	$C_1\ Op\ D_1$	$A_3\ Op\ B_3$	- store C_2, D_2 and R_3 - $NSB(R)$

Table 1. Operations scheduler

2. for each distance d_i we search the position of the first true bit (p_i)
3. the number of significant bits is the minimal position $min(p_1, p_2, p_3)$

We have compared the results obtained by this new method with those obtained with the CESTAC method (1) and the results are the same. It's easy to implement in hardware with three operators of distance that calculate the absolute value of the difference, a comparator to obtain the greatest difference and a priority encoder to search the position of the first true bit. Figure 1 presents the hardware implementation of the number of significant bits calculation.

Fig. 1. Calculation of the number of significant bits

3.4. Average result

The calculation of the average result is not done with a dedicated hardware because only one calculation is necessary at the end of the computation. So this calculation is assigned to the software.

3.5. Stochastic zero

The stochastic zero has been defined in (2). So in hardware we test if all results are null or if the number of significant bits is null. Then have a stochastic zero. Figure 2 presents the implementation of the detection of stochastic zero.

Fig. 2. Detection of stochastic zero

3.6. Putting it all together

Figure 3 presents the architecture of the stochastic floating-point unit. Every three cycles, the numbers

Fig. 3. Stochastic unit architecture

of significant bits of each operand and the result can be obtained. These three cycles are due to the computation of the three components of the stochastic number. The unit permits to :

- compute the floating-point operation
- setup the rounding mode
- calculate the number of significant bits of the operands and the result
- indicate if there is a sudden loss of accuracy that means that the difference between the number of significant bits of the operands and the result falls under the precision given by the accuracy signal
- signal if the result is a stochastic zero

The operations are scheduled as explain with table 1. This unit could be used in replacement of a standard FPU. It is necessary to modify the compiler so that it uses the stochastic numbers (three floating-point components) and the stochastic zero as a mathematics zero. Then the computation runs like a floating-point computation.

4. RESULTS

Width (bits)	Techno (μm)	Area (mm^2)		Frequency (MHz)	
		FPU	Cestac	FPU	Cestac
32	0.35	0.84	0.84	59	60
64	0.35	1.22	1.18	52	57
32	0.25	0.10	0.09	187	214
64	0.25	0.14	0.13	167	202

Table 2. Stochastic unit performances

Table 2 presents the performances of the actual stochastic FPU with addition/subtraction, comparison and conversions.
The target standard cells library is Sxlib of the Alliance CAD system [11]. The unit has been placed and routed with the Silicon Ensemble tool of Cadence and the timing analyzing was done with the timing analyzer Tas of Avertec[1].

[1] http://www.avertec.com

Furthermore a comparison of the number of significant bits calculation between the software and our hardware implementation has been done. Table 3 presents the performances in number of cycles of the two methods.

	Min (cycles)	Max (cycles)	Average (cycles)
CADNA library	856	46381140	1083
our method	133	1321980	280

Table 3. Comparison between software and hardware implementation

5. CONCLUSION

We have developed a stochastic FPU that implements addition/subtraction, test of two floating-point numbers, float to integer conversion and vice-versa. In a technology of 0.25 μm, this 32 bits unit represents 0.18 mm^2 of silicon and has a frequency of 187 MHz with two stages of pipeline. The CESTAC block is not limiting in term of critical time, but its area is the same as the FPU area. The area of the CESTAC block will not increase with more operators then that the area of the standard FPU will increase.
Furthermore, our implementation of the number of significant digits calculation is faster than the software. So the computation of the CESTAC method would be faster with our hardware implementation.
To use this stochastic FPU, we have to integrate it into a processor and to map it on a FPGA. Other operators, such as multiplication, division and square-root, are in development. A full comparison on the entire system between hard and soft implementations is also in progress.

6. REFERENCES

[1] Michael J. Schulte and Earl E. Swartzlander Jr., "Hardware Design and Arithmetic Algorithms for a Variable-Precision, Interval Arithmetic Coprocessor," *Proceedings of the 12th Symposium on Computer Arithmetic*, pp. 163–171, 1995.

[2] M.S. Cohen, T.E. Hull, and V.C. Hamarcher, "CADAC : A Controlled-Precision Decimal Arithmetic Unit," *IEEE Transactions on Computers*, vol. C-32, pp. 370–377, 1983.

[3] M. Pichat and J. Vignes, *Ingénierie du contrôle de la précision des calculs sur ordinateur*, Technip edition, 1993.

[4] J. Vignes, "Review on stochastic approach to round-off error analysis and its applications," *Math. Comp. Simul.*, vol. 30, pp. 481–491, 1988.

[5] J. Vignes and M. La Porte, "Error analysis in computing," in *Information Processing 74*, North-Holland, 1974.

[6] J.-M. Chesneaux, "Study of the computing accuracy by using probabilitic approach," in *Contribution to Computer Arithmetic and Self-Validating Numerical Methods*, pp. 19–30. C. Ulrich edition, 1990.

[7] J.-M. Chesneaux, *L'Arithmétique Stochastique et le Logiciel CADNA*, Habilitation à diriger des recherches, Université Pierre et Marie Curie, Nov. 1995.

[8] J. Vignes, "Zéro mathématique et zéro informatique," in *La vie des Sciences, C.R. Acad. Sci.*, number 1 in 4, pp. 1–13. Paris, Jan. 1987.

[9] R. Alt and J. Vignes, "Validation of results of collocation methods for ODEs with the CADNA library," *Applied Num. Math.*, vol. 20, pp. 1–21, 1996.

[10] Jean-Luc Lamotte, "A new approach for the study of surface interpolation with uncertain data using CADNA software," *workshop Reliable Computations and Interval*, 1999.

[11] A. Greiner and al ALLIANCE, "A complet set of CAD Tools for teaching VLSI Design," *Third EuroChip Workshop*, 1992, http://www-asim.lip6.fr/alliance.