Hardware implementation of a method to control round-off errors

ROSELYNE CHOTIN and HABIB MEHREZ
LIP6/ASIM Laboratory
University Pierre et Marie Curie
4 place Jussieu - 75252 PARIS cedex 05
FRANCE
Roselyne.Chotin@lip6.fr

Abstract: - This paper describes the hardware implementation of CESTAC, a method to control round-off errors in floating-point scientific computations. To overcome the speed limitation of CESTAC's software implementation, a hardware design has been developped. The proposed architecture is compliant with the IEEE-754 standard on floating-point arithmetic. It is described as a generator to take account of different designs, data-width, target library.

Key-Words: - Stochastic arithmetic, round-off errors, floating-point computation, floating-point architecture.

1 Introduction

Arithmetic operations in scientific computations increase with computers speed. But using floating-point numbers warps 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. So it is important to control errors in floating-point computations. 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][3].
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 unit (FPU) and has a bloc, which performs the stochastic arithmetic. Section 2 is dedicated to a short review of the stochastic arithmetic. Section 3 presents the architecture of some floating-point operators. Section 4 details our hardware implementation of CESTAC and section 5 our stochastic FPU. Section 6 presents the performance of the complete hardware architecture. The conclusion is given in the section 7.

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 [4][5][6]. 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.
For example a stochastic operation on two stochastic numbers with three components $A = (A_1, A_2, A_3)$ and $B = (B_1, B_2, B_3)$ is done by :

$$A \; op \; B = \begin{pmatrix} A_1 \; op \; B_1, \; round \; to \; +\infty \\ A_2 \; op \; B_2, \; round \; to \; +\infty \\ A_3 \; op \; B_3, \; round \; to \; -\infty \end{pmatrix}$$

The CESTAC method is detailed in the following.

2.1 Random arithmetic

Each result in exact arithmetic is surrounded by two floating-point numbers F^- and F^+. The round-off determines toward which number F^- or F^+ the result will be rounded. The random arithmetic permits to select F^- or F^+ with the same probability of $\frac{1}{2}$.

2.2 Estimation of the accuracy

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

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

The number of significant digits of the true result is defined in [5] by :

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

Where σ is the standard duration :

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

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 $N = 3$, $\tau_\beta = 4.303$

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 [7]. 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 :

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

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 [8][9][10][11].

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, which 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 Floating-point operators

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 which depend of the operator.
Actually only adder/subtracter, comparator and conversion between integer and float have been implemented because all are based on the floating-point adder, which is the most important and complex operator in a FPU. So most of the efforts have been devoted to develop an adder with good performances.

3.2 Floating-point adder/subtracter

An algorithm of the floating-point addition is given in [12]. It consists on : exponent substraction, alignment of the significand, significand addition, conversion, leading one detection (LOD), normalization and rounding. To reduce the delay, this algorithm can be split in two paths based on the exponent subtraction such in [13][14].

The main idea of this algorithm is that the alignment and normalization of the significand are mutually exclusive as well as rounding and conversion. Furthermore the LOD can be made in parallel with the significant adder as seen in [15]. Those architectural improvements are parameters of the generator, because they depend of the designer purposes. The adder architecture is given in the figure 1.

3.3 Floating-point comparator

The comparison operation is made with the same architecture as the addition. The two operands are subtracted and the sign of the result gives the result of the comparison.

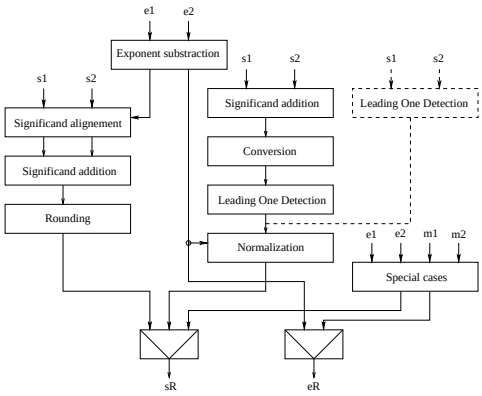

Fig. 1: Adder architecture

3.4 Format conversion

The float/integer and integer/float conversions are computed with the same architecture as the addition except an increase of the width of the normalization shifter in order to perform the integer/float conversion.

4 CESTAC in hardware

We have seen in 2 how to perform a computation with the CESTAC method. So all the steps have to be implemented.

4.1 Random round-off

The CESTAC method demands to choose a random round-off between -infinity ($-\infty$) or +infinity ($+\infty$) for the first and second execution. The third round-off is the opposite of the second. In order to choose randomly a round-off between $\pm\infty$, we use a Linear Feedback Shift Register of 32 bits.

4.2 Number of significant digits

The number of significant digits is defined in (1). A hardware implementation requires a simplification of this formula. Another method to calculate the number of significant bits has been developed :

1. calculation of the distances $d_1 = |R_1 - R_2|$, $d_2 = |R_1 - R_3|$, $d_3 = |R_3 - R_2|$
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 $min(p_1, p_2, p_3)$

Calculation validation

To validate the new calculation of the number of significant digits, we compare its results with the results given by the CADNA library [16].

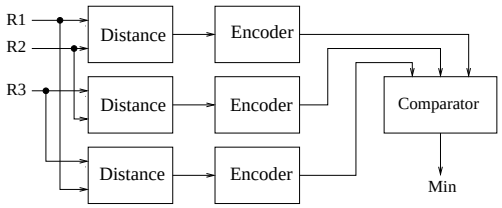

Fig. 2: Calculation of the number of significant bits

To do this we generate a random number R_1 in an interval $[j, 10 * j]$. Then R_2 is obtained with a perturbation ε of R_1 and R_3 with a random perturbation in $[0, \varepsilon]$. j and ε are integer variables.
So we compare the number of significant digits of R_1, R_2 *and* R_3 given by our calculation and by the CADNA library.
The algorithm of validation is given by :

While $j \leq interval$
  While $\varepsilon_1 \geq \varepsilon$
    For i=1 to N
      We generate randomly $R_1 \in [j, j * 10]$
      $R_2 = R_1 + \varepsilon_1$
      We generate randomly $\varepsilon_2 \in [0, \varepsilon_1]$
      $R_3 = R_1 \pm \varepsilon_2$
      $c1 = our\_calculation(R_1, R_2, R_3)$
      $c2 = cadna\_calculation(R_1, R_2, R_3)$
      If $(c1 \neq c2)$ Then error
      End If
    End For
    $\varepsilon_1 = \varepsilon_1/10$
  End While
  $\varepsilon_1 = 100$
  $j = j * 10$
End While

Where :

- interval is the interval where the numbers (R_1, R_2 and R_3) are randomly generated
- ε is the minimal perturbation
- N is the number of iterations

For these values our calculation gives the same results as the CADNA library. Furthermore our calculation has been tested in some examples of computation with the CADNA library and the results are the same. So on these samples, we have validated another method to calculate the number of significant digits.

Hardware implementation

It's easy to implement this calculation in hardware with three operators of distance, three priority encoders to search the position of the first true bit and a comparator to obtain the minimum. Figure 2 presents the hardware implementation of the number of significant bits calculation.

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

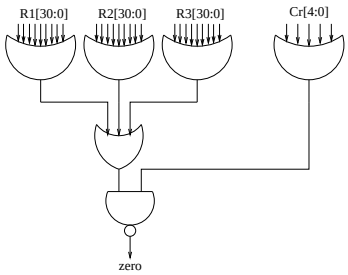

Fig. 3: Detection of stochastic zero

4.3 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.

4.4 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 3 presents the implementation of the detection of stochastic zero.

5 The stochastic FPU

Figure 4 presents the architecture of the stochastic floating-point unit. Every three cycles, the numbers 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 (round signal)
- calculate the number of significant bits of the operands and the result (signal NSB)

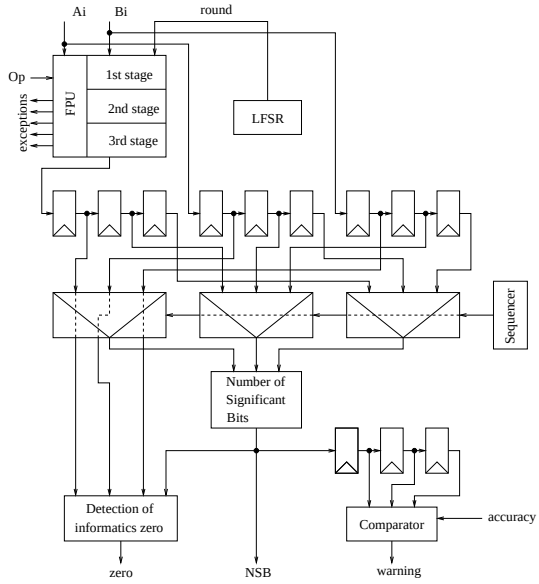

Fig. 4: Stochastic unit architecture

- indicate if there is a sudden loss of accuracy (warning signal), which 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 an stochastic zero (zero signal)

The operations are scheduled as explain in table 1. The A_i and B_i inputs are the components i of the stochastic numbers A and B and R_i is the result of the floating-point operation chosen by the signal Op. The exceptions are those of the IEEE-754 floating-point standard.

6 Results

A comparison of the number of significant bits calculation between the software and our hardware implementation has been done. Table 2 presents the per-

			Without LOD in parallel		With LOD in parallel			
Width (bits)	Pipeline stages	Techno (μm)	Area (mm^2)	Critical Time (ns)	Area (mm^2)		Critical Time (ns)	
32	1	0.35	0.71	48	0.84	+18%	39	-19%
32	2	0.35	0.77	24	0.90	+17%	24	- 0%
32	3	0.35	0.84	17	0.96	+18%	16	- 6%
64	1	0.35	1.05	53	1.23	+17%	43	-19%
64	2	0.35	1.12	28	1.30	+18%	26	- 7%
64	3	0.35	1.22	19	1.38	+17%	19	- 0%
32	1	0.25	0.08	15	0.10	+18%	13	-13%
32	2	0.25	0.09	8	0.11	+17%	7	-12%
32	3	0.25	0.10	5	0.11	+18%	5	- 0%
64	1	0.25	0.12	17	0.15	+17%	14	-18%
64	2	0.25	0.13	9	0.15	+18%	9	- 0%
64	3	0.25	0.14	6	0.16	+17%	6	- 0%

Table 3: Performances of the IEEE-754 FPU

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

Table 2: Comparison between software and hardware implementation

Width (bits)	Techno (μm)	Area (mm^2)	Critical Time (ns)
32	0.35	0.67	15
64	0.35	0.97	16
32	0.25	0.08	3.9
64	0.25	0.11	4.2

Table 4: Performances of the CESTAC unit

formances in number of cycles of the two methods. Our new method of calculation is significantly more efficient than the CADNA library. So our hardware implementation of CESTAC would be much more performant than the software.

The different hardwares have been placed and routed with the Silicon Ensemble tool of Cadence. The target standard cells library is Sxlib of the Alliance CAD system [17]. The timing analysis was done with the Tas tool of Avertec[1].

The table 3 presents the performances of the actual FPU with addition/subtraction, comparison and conversions. The chosen algorithm is to split the addition in two paths. Having a leading one detection in parallel with the significand addition is interesting in delay only if the FPU has no pipeline.

The table 3 presents the performances of the CESTAC unit. The area of the CESTAC unit represents between 30% and 50% of the stochastic FPU total area and the critical time is lower than the critical time of the FPU with three stages of pipeline. Then the CESTAC unit isn't a limiting factor for a delay consideration in the computation of floating-point operations, but represents an area increase.

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

7 Conclusion

We have developed a stochastic FPU which 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.

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.

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] D.M. Chiarulli, W.G. Rudd, and D.A. Buell, "DRAFT : A Dynamically Reconfigurable Processor for Ingeger Arithmetic," *Proceedings of the 7th Symposium on Computer Arithmetic*, pp. 309–318, 1985.

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

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

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

[7] 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.

[8] 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.

[9] S. Guilain, "Validation of thermodynamical computations using CADNA library," *Proc. CESA'96 IMACS Multiconference*, vol. 2, pp. 1139–1144, 1996.

[10] Jean-Marie Chesneaux and B. Troff, "Computational stability study using the CADNA software applied to the navier-stokes solver PEGASE," *Scientific Computing and Validated Numerics*, pp. 84–90, 1996.

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

[12] John L. Hennessy, David Goldberg, and David A. Patterson, *Computer Architecture : A quantitavtive Approach*, 2nd edition edition, Jan. 1996.

[13] S. F. Oberman and M. J. Flynn, "A Variable Latency Pipelined Floating-Point Adder," in *Proceedings of Euro-Par'96, Springer LNCS*, Aug. 1996, vol. 1124, pp. 182–192.

[14] D. Matula and A. Nielsen, "Pipelined Packet-Forwarding Floating Point: I Foundation and a Rounder," in *Proceedings of the 13th Symposium on Computer Arithmetic*, July 1997, pp. 140–147, IEEEE Computer Society Press.

[15] E. Hokenek and R. Montoye, "Leading-Zero Anticipator (LZA) in the IBM RISC System/6000 Floating-Point Execution Unit," *IBM Journal of Research and Development*, vol. 34, no. 1, pp. 71–77, Jan. 1990.

[16] J.-M. Chesneaux, "CADNA, an ADA tool for round-off error analysis and for numerical debugging," *Proceedings Congress on ADA in Aerospace*, 1990.

[17] 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.