Hardware implementation of discrete stochastic arithmetic

Roselyne AVOT-CHOTIN
LIP6/ASIM Laboratory, UPMC
Cuvier botanique - bureau 201
4 place Jussieu
F-75252 Paris cedex 05 - France
Email : Roselyne.Avot@lip6.fr

Habib MEHREZ
LIP6/ASIM Laboratory, UPMC
Cuvier botanique - bureau 203
4 place Jussieu
F-75252 Paris cedex 05 - France
Email : Habib.Mehrez@lip6.fr

Abstract. *In this paper we present a hardware implementation of the Discrete Stochastic Arithmetic (DSA) which is based on CESTAC (Controle et Estimation STochastique des Arrondis de Calculs), a method to control round-off errors in floating-point scientific computations.*
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 architecture is based on a Stochastic Floating-Point Unit (SFPU) which performs discrete stochastic operations. This SFPU has been integrated in a coprocessor, used in a complete System on Chip (SoC).

1 Introduction

The number of arithmetic operations in scientific computations increases with computers speed. But the use of floating-point numbers damages the results of these operations, due to a round-off error propagation. At the end of the computation, the result can be totally different from the expected result. The aim of the discrete stochastic arithmetic is to estimate the loss of accuracy of elementary operations.
Different methods to control this round-off error, such as interval arithmetic, discrete stochastic arithmetic, variable-precision arithmetic, already exist with 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 discrete stochastic arithmetic. Section 2 is dedicated to a short review of the discrete stochastic arithmetic. Section 3 details our hardware implementation of the SFPU. In section 4, the integration of the SFPU in a coprocessor as element of a SoC is described. Section 5 presents the performance of the complete hardware architecture. The conclusion is given in the section 6.

2 The Discrete 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 have been extensively proved in [6][7]. The idea

behind this method is to execute the same computation several times by running each arithmetical operation with a different rounding mode. 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 significant 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 perturbed. The theory of the CESTAC method resides on the study of the propagation of these perturbations.
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 number of significant digits of the true result is defined in [4] by :

$$C_{\overline{R}} = \log_{10} \frac{\sqrt{N}.|\overline{R}|}{\sigma.\tau_\beta} \text{ with } \overline{R} = \frac{1}{N}\sum_{i=1}^{N} R_i \text{ and } \sigma^2 = \frac{1}{N-1}\sum_{i=1}^{N}(R_i - \overline{R})^2 \qquad (1)$$

The result $\overline{R}$ of the N executions of the same computation, is the average of all the results (R_i) obtained with the CESTAC method, σ is the standard deviation, 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

The concept of stochastic zero was introduced in [8] to treat non-significant results. A result is a stochastic zero if all the results of the N runs (with different round-off error propagation) are null or if the number of significant digits is negative or null. In practice a stochastic zero is often considered as a mathematical zero.

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 rounding mode
- choose randomly the rounding mode
- compute the number of significant digits

- detect the stochastic zeros
- compute the average result

3 The Stochastic FPU

We have seen how to perform a computation with the CESTAC method. So all the steps have to be implemented. Stochastic operations are computed on a Stochastic Floating-Point Unit (SFPU). This unit has two parts : a standard floating-point unit and some dedicated hardwares to perform CESTAC method.

3.1 Floating-point unit

A floating-point unit has to be developed according to the IEEE-754 standard [9]. In particular it should be able to perform operation on special numbers such as infinity or NaNs. Furthermore it should detect all the standard floating-point exceptions.
This unit has been developed as a parameterizable generator. Its parameters are the width (32 bits, 64 bits or other), the number of pipeline stages (1, 2 or 3) and some architectural parameters such as not have dedicated hardware for the treatment of denormalized numbers or special numbers (infinity, NaNs)[1]. It performs floating-point addition, subtraction, multiplication, comparison and conversions between integer and float.

3.2 CESTAC hardwares

3.2.1 Random rounding mode

The random rounding mode was defined 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 rounding mode. 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 rounding mode is the opposite of rounding mode chosen for $i = 2$.

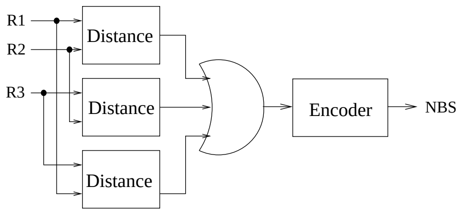
3.2.2 Number of significant digits

The number of significant digits is defined in equation (1). A hardware implementation requires a simplification of this formula. Other method to calculate the number of significant bits is done by :

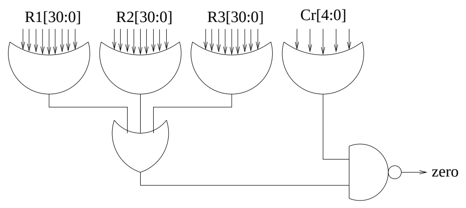
1. calculation of the distances $d_1 = |R_1 - R_2|$, $d_2 = |R_1 - R_3|$, $d_3 = |R_3 - R_2|$
2. for each distance d_i we search the position of the first true bit (p_i)
3. the number of significant bits is the minimal position $min(p_1, p_2, p_3)$

It has been proved that this method is a good approximation for the number of significant digit calculation[2].
This method is easy to implement in hardware with three operators of distance that

[1]Note that in this case the FPU does not respect the IEEE-754 standard
[2]This proof will be the subject of later publication



(a) Number of significant bits



(b) Stochastic zero

Figure 1: Stochastic arithmetic specific hardwares

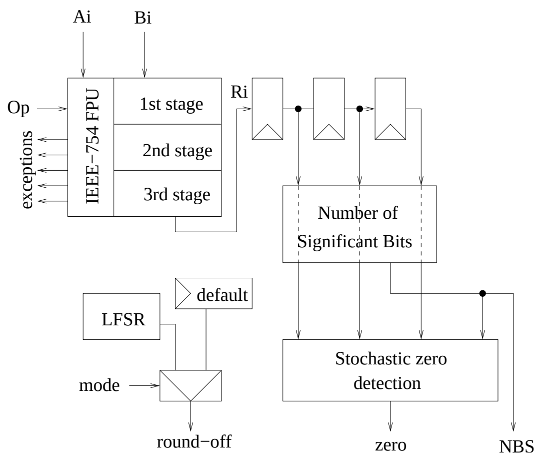


Figure 2: Stochastic unit architecture

calculate the absolute value of the difference, a comparater to obtain the greatest difference and a priority encoder to search the position of the first true bit. Figure 1(a) presents the hardware implementation of the number of significant bits calculation.

3.2.3 Average result

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

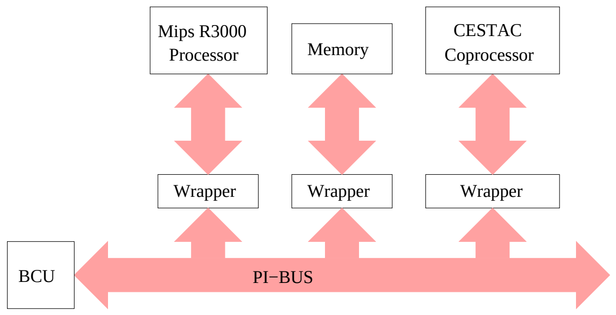
3.2.4 Stochastic zero

The stochastic zero has been defined in 2.3. So in hardware we test if all results are null or if the number of significant bits is null. Figure 1(b) presents the implementation of the detection of stochastic zero.

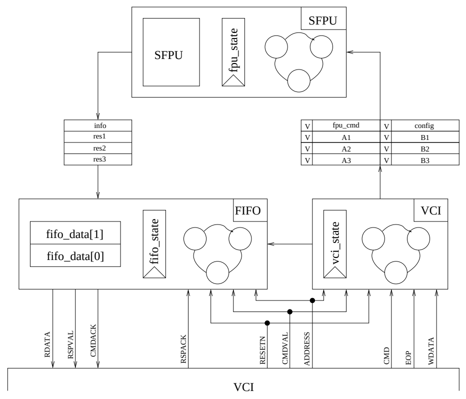
Figure 2 presents the architecture of the stochastic floating-point unit.

4 System on Chip integration

To use this SFPU, we have integrated it in a complete system to perform floating-point computations.



(a) System on Chip



(b) CESTAC coprocessor

Figure 3: System integration

4.1 Architecture

The proposed architecture is given on figure 3(a). The components are connected to a Pi-Bus [10]. They communicate through wrappers with the VCI protocol [11].

- The memory permits to store data and programs
- The computation is executed on a Mips R3000 processor
- The CESTAC coprocessor performs stochastic operations
- The BCU (Bus Control Unit) controls the bus accesses

The internal architecture of the CESTAC coprocessor is given figure 3(b). The coprocessor has three parts :

1. A VCI automaton that reads the data for the SFPU from the wrapper and writes the result to the wrapper.
2. A FIFO automaton that reads the results of the SFPU and store them in a FIFO.
3. A SFPU automaton that manages stochastic operations.

4.2 The VCI communications

4.2.1 Writing mode

When a write request appears on the bus, the VCI automaton writes the data in 8 registers. These 8 registers permit to store the 3 stochastic components of the two operands, the codop and the configuration of the SFPU. Each register has a validity bit that is set when the data has been written in the register and unset when the data is consumed by the SFPU.

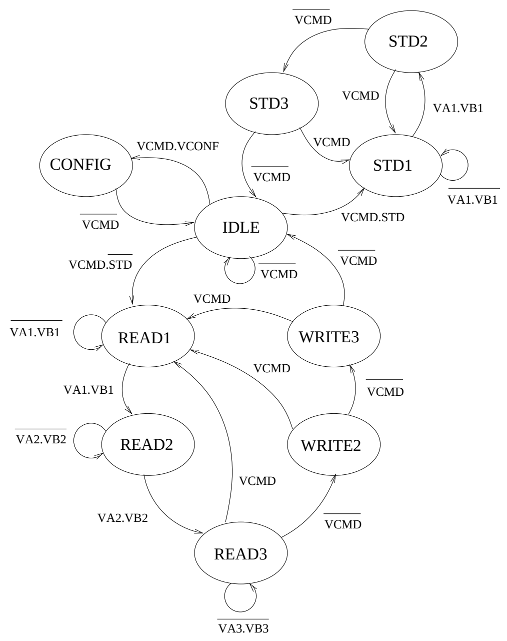


Figure 4: SFPU automaton

4.2.2 Reading mode

There are also 4 registers reading addressable by the FIFO automaton to get the 3 components of the result and the informations given by the SFPU (number of significant bits, stochastic zeros, exceptions). The results are written in a FIFO controlled by the FIFO automaton in order to avoid Mealy signals. The VCI automaton permits to read the results in the FIFO when a read request appears on the bus.

4.3 The SFPU automaton

The automaton that controls the SFPU is presented figure 4. An operation is performed on the 3 components of stochastic operands. When an operation is required, the validity bit of the CMD register is set (VCMD=1[3]), the automaton goes in the READ1 state and waits for the write of the first components of the stochastic operands (VA1=1 and VB1=1). When the components are ready, an operation begins in the SFPU. The next state is READ2 where the automaton waits for the second components to begin the second operation in the SFPU, and then READ3 for the third components. In this state, the first component of the result is ready and written in the RES1 register. The next state is WRITE2 where the second component of the result is written in RES2 register and then in the WRITE3 state the third component is written in the RES3 register. After if the automaton goes in the IDLE state to wait for the next operation. When the automaton consumes a data, the validity bit of the corresponding register is unset. In all the state of writing a result, if there is another operation requested, the automaton goes in the READ1 state to compute it due to the 3 pipeline stages of the SFPU.
The two another branches of the automaton permit :

- to configure the SFPU (CESTAC mode, standard mode, rounding mode in standard mode, accuracy in CESTAC mode).

[3]VX is the validity bit of the register X

Width	Techno	Area (mm^2)		Frequency (MHz)	
(bits)	(μm)	FPU	Cestac	FPU	Cestac
32	0.35	1.03	0.48	72	75
64	0.35	2.16	0.96	54	57
32	0.25	0.52	0.25	309	326
64	0.25	1.1	0.46	221	241

Table 1: Stochastic unit performances

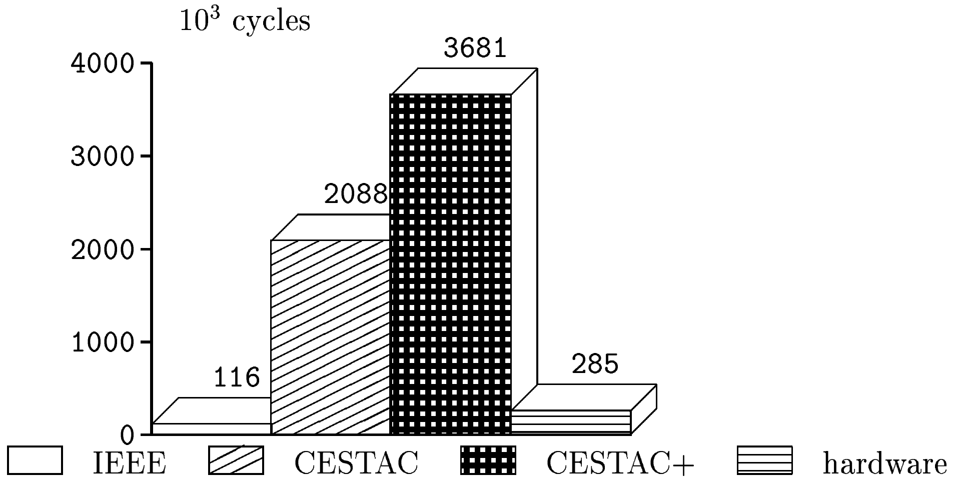


Figure 5: Average run time

- to execute floating-point operations in standard mode without stochastic operands and control of the accuracy.

5 Results

Table 1 presents the performances of the actual stochastic FPU with 3 stages of pipeline. The target standard cells library is Sxlib of the Alliance CAD system [12]. 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[4].

For the validation of our system on chip, the CASS simulator [13] has been used. It permits a cycle-true/bit-true simulation. We have run some programs on this system and the average performances in number of cycles are given figure 5, where :

- IEEE is the standard floating-point arithmetic (without estimation and control of accuracy)
- CESTAC is the software implementation of the stochastic arithmetic with accuracy estimation only for multiplication, division and comparison (as the software library)
- CESTAC+ is the software implementation of the stochastic arithmetic with accuracy estimation for all the operations

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

- hardware is our hardware implementation of stochastic arithmetic with accuracy estimation for all the operations

6 Conclusion

We have developed a stochastic FPU that implements addition/subtraction, multiplication, comparison of two floating-point numbers, float to integer conversion and vice-versa. In a technology of 0.25 μm, this 32 bits unit represents 0.77 mm^2 of silicon and has a frequency of 309 MHz with three stages of pipeline. The CESTAC block is not limiting in term of critical time, and its area represents about 33% of the total area of the SFPU.
A system on chip that implements the SFPU has been developed. This hardware architecture is significantly faster than the software version.
Some improvements are still possible to accelerate the computation by storing the intermediate results in order to minimize the bus access or by sending the data in burst mode since a stochastic number has 3 components that can be stored at contiguous addresses.
Another improvement is to integrate the SFPU in a processor.

References

[1] M.J. Schulte and E.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] ANSI/IEEE Std 754-1985, *IEEE Standard for Binary Floating-Point Arithmetic*, 1985.

[10] OMI 324, *OMI/PI-Bus Specification*, 1994, PI-BUS Rev. 0.3d.

[11] Virtual Sockets Interface Alliance, *VSI ALliance Virtual Component Interface Standard*, ocb design working group edition, Nov. 2000.

[12] 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.

[13] F. Pétrot, “Cycle Accurate System Simulation,” *Medea-Esprit Conference*, Nov. 1999.