INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

# *Projet ARENAIRE*

*Arithmétique des Ordinateurs*

*UR Rhône Alpes, École Normale Supérieure de Lyon*

THÈME 2B

# Table des matières

*Le projet ARENAIRE est un projet commun au CNRS, à l'Ecole Normale Supérieure de Lyon et à l'INRIA. Il fait partie du Laboratoire de l'Informatique du Parallélisme (LIP, UMR CNRS-ENS Lyon-INRIA 5668) de l'École Normale Supérieure de Lyon. Ce projet est localisé à Lyon dans les locaux de l'ENS Lyon.*

**Note importante:** *Le projet CNRS/ENSL/INRIA Arénaire ayant été créé le 1er Octobre 1998, nous n'avons pas fourni à l'INRIA de rapport d'activité 1998. Aussi, nous incluons dans ce rapport 1999 la présentation de nos travaux effectués entre le 1er Octobre et le 31 décembre 1998.*

# 1 Composition de l'équipe

**Responsable scientifique**

Jean-Michel Muller [Directeur de Recherches au CNRS]

**Assistante de projet**

Sylvie Boyer [Contractuelle, 20% sur le projet]

**Personnel Inria**

Arnaud Tisserand [Chargé de Recherches stagiaire, dans le projet depuis le 1/10/99]

Philippe Langlois [en délégation sur un poste de Chargé de Recherches, dans le projet depuis le 1/10/99]

**Personnel CNRS**

Marc Daumas [Chargé de Recherches]

**Personnel ENS Lyon**

Florent Dupont de Dinechin [Maître de conférences]

**Chercheurs d'autres établissements**

Anne Mignotte [Maître de conférences UCB Lyon (IUT de Bourg en Bresse). Dans le projet jusqu'au 1/9/99. A depuis quitté le projet suite à son recrutement sur un poste de Professeur à l'INSA de Lyon]

**Chercheurs doctorants**

Claire Finot [Allocataire et Moniteur MENRT]

Antoine Fraboulet [Allocataire MENRT. Dans le projet, jusqu'au 1/9/99 (a suivi Anne Mignotte lors de son recrutement comme professeur)]

Vincent Lefèvre [Allocation couplée]

## 2 Présentation et objectifs généraux

**Mots clés** : arithmétique des ordinateurs, virgule flottante, fiabilité numérique, fonctions élémentaires, opérateurs asynchrones, circuits intégrés numériques, FPGA.

**Résumé :** *L'objectif du projet* Arénaire *est de contribuer à l'élaboration et à la consolidation des connaissances dans le domaine de l'arithmétique des ordinateurs. Nos travaux vont se situer sur deux axes privilégiés : d'une part l'utilisation avertie des normes existantes pour construire des algorithmes fiables et les prouver ; d'autre part la mise au point, plus en amont, des unités arithmétiques du futur. Fiabilité, précision, rapidité et (à plus long terme) faible consommation sont les principaux objectifs du projet.*

Le but scientifique du projet est de participer à l'amélioration de l'arithmétique disponible sur les microprocesseurs et les calculateurs dédiés. Cette amélioration peut porter aussi bien sur la rapidité des calculs que sur leur fiabilité et leur précision, ainsi que sur des paramètres plus technologiques tels que la consommation d'énergie que les dans les circuits intégrés.

L'arithmétique des ordinateurs a connu un changement majeur en 1985, avec l'apparition de la norme Ieee-754, qui spécifie les formats de représentation des nombres et les opérations arithmétiques en virgule flottante. Cette norme s'est rapidement imposée. Une des principales exigences de la norme est liée aux modes d'arrondi. La somme, le produit, le quotient de deux nombres qui s'écrivent exactement en virgule flottante (on dira des « nombres machine » pour simplifier) ne sont en général pas des nombres machine. Il faut donc les *arrondir*. La norme Ieee-754 demande que l'utilisateur puisse choisir entre 4 modes d'arrondi possibles : au plus près — c'est le mode par défaut, vers le haut, vers le bas ou vers zéro. Si $a$ et $b$ sont des nombres machine, le résultat d'une des quatre opérations arithmétiques $a \star b$ doit être le nombre machine qui serait obtenu si on avait d'abord fait le calcul avec une *précision infinie* avant d'arrondir avec le mode d'arrondi demandé par l'utilisateur. Cette exigence est appelée « arrondi exact ».

La plupart des micro-ordinateurs et des stations de travail actuels suivent la norme Ieee et les conséquences de cette adhésion sont importantes :

- on peut développer des *algorithmes* et des *preuves* de correction qui utilisent la spécification des unités de calcul comme cela a été fait par William Kahan et ses élèves à l'*University of California* de Berkeley [Kah99] ;
- en jouant sur les différents modes d'arrondi, il est possible pour l'utilisateur d'obtenir un minorant ou un majorant certifié du résultat et de faire ainsi de l'*arithmétique d'intervalles*.

L'exemple suivant illustre les avantages que l'on peut retirer de cette démarche. On prouve [CHGM99] que (sauf dépassements de capacité), si $x$ est exactement représentable en flottant et si $z$ est

[Kah99] W. Kahan, « Lecture Notes on the Status of the IEEE-754 Floating-Point Standard », 1999, Disponible à l'URL http://HTTP.CS.Berkeley.EDU/∼wkahan/.

[CHGM99] M. Cornea-Hasegan, R. Golliver, P. Markstein, « Correctness Proofs Outline for Newton-Raphson Based Floating-Point Divide and Square-Root Algorithms », *in : Proceedings of the 14th IEEE Symposium on Computer Arithmetic*, IEEE Computer Society Press, Los Alamitos, CA, Adelaide (Australie), Avril 1999.

une approximation au poids du dernier bit près de $1/x$, alors la suite d'opérations

$$\begin{aligned} \epsilon &= \circ_n(1 - xz) \\ z' &= \circ_n(z + \epsilon z) \end{aligned}$$

où $\circ_n(a + bc)$ signifie que le calcul $a + bc$ est fait en arrondi au plus près, donne une valeur $z'$ égale à l'arrondi au plus près de $1/x$. Ainsi, en utilisant l'arrondi correct de la multiplication-accumulation, on peut fournir des algorithmes de division fournissant un arrondi correct sans utiliser d'arithmétique à plus grande précision pour les calculs intermédiaires.

La plupart des travaux actuels en arithmétique des ordinateurs se situent autour des deux thèmes suivants :

- **Thème 1** : Les opérateurs arithmétiques étant donnés, les utiliser « au mieux ». Par exemple, les « axiomes » qui constituent les spécifications de la norme IEEE (ou d'autres spécifications) étant donnés, il faut construire des algorithmes et les preuves de correction de ces algorithmes qui utilisent au mieux chaque opportunité de performance. Claire Finot (étudiante en thèse au LIP et membre du projet) travaille activement à une arithmétique en précision multiple qui utilise, au lieu d'entiers, des flottants IEEE comme « briques de base ».

- **Thème 2** : Concevoir et/ou valider des opérateurs, ou de manière peut-être plus générale, concevoir d'autres « briques de base » que celles de l'arithmétique IEEE. La norme IEEE ne spécifie que les quatre opérations arithmétiques et la racine carrée et nous travaillons à une implantation rigoureuse et intelligente d'autres fonctions en virgule flottante. Nous travaillons également sur d'autres systèmes de numération. Il n'est pas utopique d'étudier ces systèmes : si pour des besoins « généralistes » comme pour un microprocesseur, l'arithmétique virgule flottante semble incontournable (même si on peut certainement en améliorer les implantations existantes), sur des systèmes dédiés à des applications particulières, d'autres modes de représentation des nombres peuvent s'avérer mieux adaptés. Un exemple classique est celui des systèmes de numération *redondants* (carry-save, borrow-save, etc.) qui sont utilisés à l'intérieur de nombreux multiplieurs et diviseurs[1]. Cette utilisation est transparente : en entrée comme en sortie de ces opérateurs, les nombres sont représentés dans un système usuel.

La recherche sur les algorithmes et les architectures pour la multiplication, pour la division, et pour le calcul des fonctions élémentaires est toujours très active[2]. Les membres du projet ont acquis une solide réputation dans ce domaine, et ont l'intention de poursuivre leurs travaux dans cette voie.

1. Par exemple, dans les diviseurs des processeurs Pentium et Pentium Pro, deux systèmes redondants différents sont utilisés. Le système « carry-save » pour représenter le reste partiel de la division, et un « système d'Avizienis » (base 4, chiffres $-2$, $-1$, 0, 1 et 2) pour représenter les chiffres du quotient. Le quotient est réécrit en notation binaire usuelle à la fin du calcul.

2. Juste pour donner un exemple : les algorithmes des diviseurs des processeurs HP PA-7100, HP PA-8000, AMD 29050, UltraSPARC, IBM RS/6000 sont *tous* différents. Ceci montre à l'évidence que les solutions sont encore très loin d'être figées dans ce domaine.

La maîtrise des erreurs d'arrondi dans les calculs (et de manière plus générale, la fiabilité numérique des systèmes) est un sujet de plus en plus important. On effectue des calculs considérablement plus volumineux que dans les années 70, alors que les formats de représentation des nombres (et par conséquent la précision de chaque opération prise individuellement) n'ont presque pas changé. Une conséquence est que les problèmes de précision se posent de plus en plus. Dans de nombreux domaines d'application, l'imprécision de l'arithmétique flottante peut avoir des conséquences tragiques. La maîtrise de la précision est un thème important. Notre équipe travaille déjà activement sur ce thème : nous avons participé à l'ARC Inria FIABLE. Nous avons également collaboré avec la société Aérospatiale dans ce domaine. Nous nous intéressons tout particulièrement à la précision du calcul des fonctions élémentaires.

L'amélioration automatique de la précision des résultats complète la maîtrise de la fiabilité numérique. Les utilisateurs qui rencontrent des problèmes de précision numérique n'ont souvent ni le temps ni le métier pour répondre à ces problèmes difficiles. Ce constat incite à proposer des approches automatiques qui permettent de contrôler et d'améliorer la qualité numérique d'un programme vu comme une « boîte noire ».

# 3 Fondements scientifiques

### 3.0.1 Le dilemme du fabricant de tables

Dans la norme IEEE 754, l'exigence d'« arrondi exact » n'apparaît que pour les 4 opérations arithmétiques et la racine carrée. Il n'y a aucune exigence de ce type pour les fonctions élémentaires, car on a longtemps cru qu'une telle exigence était impossible.

Lorsque l'on cherche à évaluer une fonction élémentaire (sinus, cosinus, logarithme, exponentielle, etc.), on calcule en fait une approximation intermédiaire de la valeur exacte sur une arithmétique à précision finie mais supérieure à la précision qui correspond au format dans lequel le résultat final sera retourné, appelé «format cible». Ensuite, on arrondit l'approximation, suivant le mode d'arrondi désiré, vers le format cible. Tout le problème est de savoir si la précision du calcul intermédiaire suffit pour que cet arrondi de l'approximation coïncide avec l'arrondi du résultat exact. Résoudre ce problème permettrait de fournir des environnements où toutes les primitives numériques (i.e. les opérations et les fonctions élémentaires) seraient complètement spécifiées. C'est un des objectifs majeurs de notre projet. Donnons un exemple. Supposons des mantisses de 24 bits (c'est le cas de la simple précision de la norme IEEE). Considérons le nombre $x$ qui s'écrit 1.11001100111000110011001. Son exponentielle s'écrit en base 2

$$\overbrace{110.000011010011110100110}^{24 \text{ bits}}011111111111111111111111101100100001\ldots$$

ce qui est extrêmement proche du milieu de deux nombres machine consécutifs. Si l'on désire calculer l'exponentielle de $x$ en arrondi au plus près, il faudra le faire en calculant tout d'abord une approximation très précise de cette exponentielle (sur au moins 50 bits), afin de savoir si elle est au-dessous ou au-dessus du milieu des nombres machine

$$A = 110.000011010011110100110$$

et

$$B = 110.00001101001111010011 1.$$

Si elle est en dessous (ce qui est ici le cas), le résultat fourni devra être $A$, et dans le cas contraire ce sera $B$.

### 3.0.2 Synthèse d'architectures intégrées

La synthèse d'architectures intégrées consiste à générer automatiquement un système intégré optimisé à partir de la description de son comportement, en appliquant principalement des techniques d'optimisation combinatoire. C'est un outil de la Conception Assistée par Ordinateurs (CAO) des Circuits et Systèmes Intégrés (C&SI). Avec l'augmentation de la complexité des architectures, la nécessité de rationaliser la conception devient évidente. Le niveau d'abstraction de la CAO de C&SI augmente donc sans cesse. Les systèmes contiennent désormais des parties programmables plus souples et des parties matérielles « dures » plus rapides. Les niveaux bas de la CAO dépendent de la *technologie visée* : on ne conçoit pas de la même façon un circuit intégré spécifique et un système constitué de circuits programmables (FPGA). A l'opposé, les hauts niveaux sont dépendants des *applications*. Il faudrait donc développer un outil de conception par domaine d'application.

On appelle parfois les outils de synthèse de haut niveau des « compilateurs sur silicium». En effet, l'analogie entre la synthèse et la compilation est forte. On veut concevoir une architecture spécifique qui réalise plusieurs processus en même temps pour une application dédiée. Les critères et les contraintes liés à la génération de circuits et systèmes intégrés sont particuliers. Il s'agissait essentiellement jusqu'ici de réaliser des compromis entre la surface du système et ses performances. Cependant, pour prendre en compte les évolutions récentes en termes d'applications et de technologies, il faut désormais se pencher également sur des critères tels que la consommation des circuits intégrés. Ce phénomène est renforcé par l'arrivée des technologies dites sub-microniques. Ces nouvelles technologies permettent d'intégrer sur une petite surface des applications complexes tout en améliorant les performances. Mais les besoins dans les applications embarquées obligent de plus en plus à réduire la consommation.

### 3.0.3 Autour de l'arithmétique à virgule flottante normalisée IEEE

Un opérateur arithmétique manipulant des nombres à virgule flottante est plus complexe que le même opérateur restreint aux seuls nombres entiers. Il faut en effet gérer correctement les divers modes d'arrondi, manipuler à la fois les mantisses et les exposants des opérandes, traiter les divers cas d'exception (infinis, nombres « dénormalisés », etc.). Paradoxalement, sur les microprocesseurs actuels, l'unité de calcul à virgule flottante est souvent plus puissante que l'unité de calcul entier. Il devient important de bien en maîtriser le fonctionnement. Une bonne connaissance des spécificités des standards actuels peut permettre d'obtenir des programmes très efficaces.

Quand le calcul sur ordinateur donne un résultat trop incorrect, il nous faut proposer des solutions alternatives raisonnables. Un de nos exemples de travail est le calcul d'un déterminant (ou parfois simplement du signe d'un déterminant) de taille 3 à 10. C'est une routine couram-

ment utilisée en géométrie algorithmique et clairement instable pour peu que le déterminant soit suffisamment petit devant les coefficients de la matrice.

Cette routine est à la base de la plupart des algorithmes de géométrie algorithmique. Les algorithmes sophistiqués utilisent des déterminants de grande taille. Il devient important de calculer très vite un déterminant robuste dans tous les cas. Dans la plupart des applications de géométrie algorithmique que nous visons, ce qui nous intéresse est le signe du déterminant. On ne veut plus d'information approchée (la valeur du déterminant à un seuil d'erreur près), mais une information certaine et prouvée (le signe). Une erreur sur ce signe peut entraîner des incohérences et donc un comportement imprévisible du programme.

#### 3.0.4 Correction linéaire des erreurs d'arrondi

Les erreurs d'arrondi introduites par chaque opération arithmétique modifient le comportement numérique de certains algorithmes. Elles peuvent être à l'origine d'une perte de précision sur un résultat final. Elles peuvent également faire prendre un « mauvais branchement » lors d'un test.

L'erreur globale peut être approchée par la linéarisation de l'influence de ces erreurs d'arrondi élémentaires sur le résultat final. Cette technique de linéarisation avait jusqu'à présent été utilisée pour majorer cette erreur globale. La correction linéaire automatique étend cette technique en *calculant* cette linéarisation par différentiation automatique et évaluation des erreurs d'arrondi élémentaires. On dispose ainsi d'un terme correctif qui, ajouté au résultat final, en améliore la précision. Cette amélioration est préférable à celle qui résulterait d'un passage en précision supérieure.

## 4 Résultats nouveaux

### 4.1 Dilemme du Fabricant de Tables

**Participants** : Vincent Lefèvre, Jean-Michel Muller, Arnaud Tisserand.

**Mots clés** : virgule flottante, arrondis, fonctions élémentaires.

**Résumé :** *Nous avons conçu des algorithmes et des programmes permettant de construire les pires cas pour le dilemme du fabricant de tables. Ces pires cas vont nous permettre de construire une bibliothèque calculant les fonctions élémentaires avec « arrondi exact ».*

Vincent Lefèvre [12] a mis au point un algorithme et une batterie de programmes permettant de construire les pires cas, pour le format virgule flottante « double précision » du dilemme du fabricant de tables. Son algorithme a permis de valider des estimations proposées par lui-même, Jean-Michel Muller et Arnaud Tisserand [13]. Ses programmes ont, entre autres, nécessité la construction d'algorithmes de multiplication par une constante [33]. En utilisant ces résultats, nous travaillons à la mise au point d'une bibliothèque de calcul des fonctions élémentaires avec arrondi correct [24].

Nous rendons publique une liste restreinte de pire cas à l'URL

www.ens-lyon.fr/~jmmuller/TMD.html.

Cette liste ne contient pas tous les pires cas trouvés, car nous désirons d'une part prendre un peu d'avance dans l'utilisation de nos pires cas pour la conception de fonctions, et d'autre part valoriser les résultats trouvés en ne les diffusant qu'à un «club de partenaires» de notre projet.

## 4.2 Adaptation de l'algorithme BKM

**Participant** : Jean-Michel Muller.

**Mots clés** : fonctions élémentaires.

**Résumé :** *Nous avons adapté l'algorithme BKM (qui calcule des logarithmes et des exponentielles complexes) à des architectures travaillant en base 10.*

L'algorithme BKM avait été introduit par Jean-Claude Bajard, Sylvanus Kla et Jean-Michel Muller en 1994 [2]. Cet algorithme permet d'évaluer rapidement des exponentielles et logarithmes complexes. Il est basé sur l'itération

$$\begin{cases} L_{n+1} = L_n - \ln(1 + d_n 2^{-n}) \\ E_{n+1} = E_n(1 + d_n 2^{-n}) \end{cases}$$

où $d_n$ est un nombre complexe de la forme $d_n^x + i d_n^y$, avec $d_n^x, d_n^y \in \{-1, 0, 1\}$ (ce qui implique qu'une multiplication par $d_n$ s'effectue trivialement).

En collaboration avec Fabien Rico (doctorant au LIRMM, Montpellier) et Laurent Imbert (doctorant au LIM, Marseille), Jean-Michel Muller a développé une variante de cet algorithme adaptée au calcul sur des machines utilisant la base 10 [10] (on vise ainsi l'implantation de ces fonctions sur des calculatrices).

## 4.3 Algorithmes de division

**Participant** : Jean-Michel Muller.

**Mots clés** : division, racine carrée, méthode de Newton, méthode de Goldschmidt.

**Résumé :** *De plus en plus souvent, l'implantation de la division et de la racine carrée se fait par appel du multiplieur flottant. Nous proposons une méthode utilisant au mieux les possibilités de pipe-line des multiplieurs présents dans les microprocesseurs actuels.*

Si $N$ et $D$ sont 2 nombres de $n$ bits, $1 \leq N, D < 2$, la méthode de Goldschmidt calcule $N/D$ comme suit. On construit une suite de facteurs $K_1$, $K_2$, $K_3$, ... tels que $r_i = DK_1K_2 \dots K_i$ tende vers 1. Ceci donne,

$$q_i = NK_1K_2 \dots K_i \to Q.$$

Le premier facteur $K_1$ est obtenu par lecture de table. Après ceci, si $r_i = 1 - \alpha$, on choisit $K_{i+1} = 1 + \alpha$, ce qui donne $r_{i+1} = 1 - \alpha^2$. Regardons ce qui se passe pour calculer $q_4$.

1. **Pas 1.** Supposons $D = 1.d_1d_2 \dots d_{n-1}$, et définissons $\hat{D} = 1.d_1d_2 \dots d_p$, avec $p << n$ (typiquement, $n = 53$ et $p \approx 10$). On lit $K_1 = 1/\hat{D}$ dans une table. Notons $\epsilon = 1 - K_1D$. On calcule
   - $r_1 = DK_1 = 1 - \epsilon$ (**mult. 1**);
   - $q_1 = NK_1$ (**mult. 2**).
2. **Pas 2.** A partir de $r_1$, on obtient $K_2 = 1 + \epsilon$. On calcule alors
   - $r_2 = r_1K_2 = 1 - \epsilon^2$ (**mult. 3**);
   - $q_2 = q_1K_2$ (**mult. 4**).
3. **Pas 3.** Comme précédemment, on obtient $K_3 = 1 + \epsilon^2$. On calcule alors
   - $r_3 = r_2K_3 = 1 - \epsilon^4$ (**mult. 5**);
   - $q_3 = q_2K_3$ (**mult. 6**).
4. **Pas 4.** Comme précédemment, on obtient $K_4 = 1 + \epsilon^4$. On calcule $q_4 = q_3K_4$ (**mult. 7**) qui vérifie $\frac{N}{D} = \frac{q_4}{1-\epsilon^8}$.

Cette itération s'implante en 17 cycles sur un multiplieur virgule flottante à 4 niveaux de pipe-line. Nous avons proposé [29, 9] des stratégies permettant d'effectuer des divisions plus rapidement en modifiant l'algorithme de Goldschmidt. Soit $\hat{\epsilon}$ le nombre constitué des $p$ bits les plus significatifs de $\epsilon$, et soit $\epsilon_r = \epsilon - \hat{\epsilon}$. Nous cherchons par exemple à accélérer les calculs en déduisant directement de $q_2$ une approximation de $q_4$. Au lieu de calculer $q_4 = q_2\,(1 + \epsilon^2)\,(1 + \epsilon^4)$, une de nos variantes consiste à calculer $q_4'' = q_2(1 + \epsilon^2 + \hat{\epsilon}^3(4\epsilon_r + \hat{\epsilon}))$, où le terme $\hat{\epsilon}^3$ est lu dans une table dès que $\epsilon$ est connu. L'erreur de calcul est d'environ $6 \times 2^{-6p}$. L'implantation sur un multiplieur pipe-line à 4 niveaux est décrite Figure 1.

### 4.4 Propriétés de la norme IEEE 754

**Participants** : Marc Daumas, Claire Finot, Jean-Michel Muller.

**Mots clés** : virgule flottante, preuve d'algorithmes, erreurs d'arrondi.

De récents problèmes liés à l'arithmétique virgule flottante (citons entre autres le fameux « bug» de la division du processeur du Pentium, et le fait que la soustraction virgule flottante du processeur spatial 1750 soit très imprécise dans certains cas) ont montré la nécessité de mieux contrôler la correction des opérateurs arithmétiques d'une part, et la qualité numérique des logiciels, d'autre part.

Nous désirons utiliser les spécifications de la norme IEEE 754 pour prouver le bon comportement de programmes usuels. Pour cela, il nous faut être capable de montrer la préservation d'un petit nombre de propriétés correctes en arithmétique « exacte ».

Nous avons déjà obtenu de tels résultats grâce aux travaux de Marc Daumas et Claire Finot sur les «expansions»[19, 8]. Nous préparons une réponse à l'appel d'offres INRIA concernant les Actions Coopératives Concertées. Nous préparons avec le projet LEMME (Sophia) et le projet POLKA (Lorraine) une ARC sur la certification d'algorithmes d'arithmétique des ordinateurs.

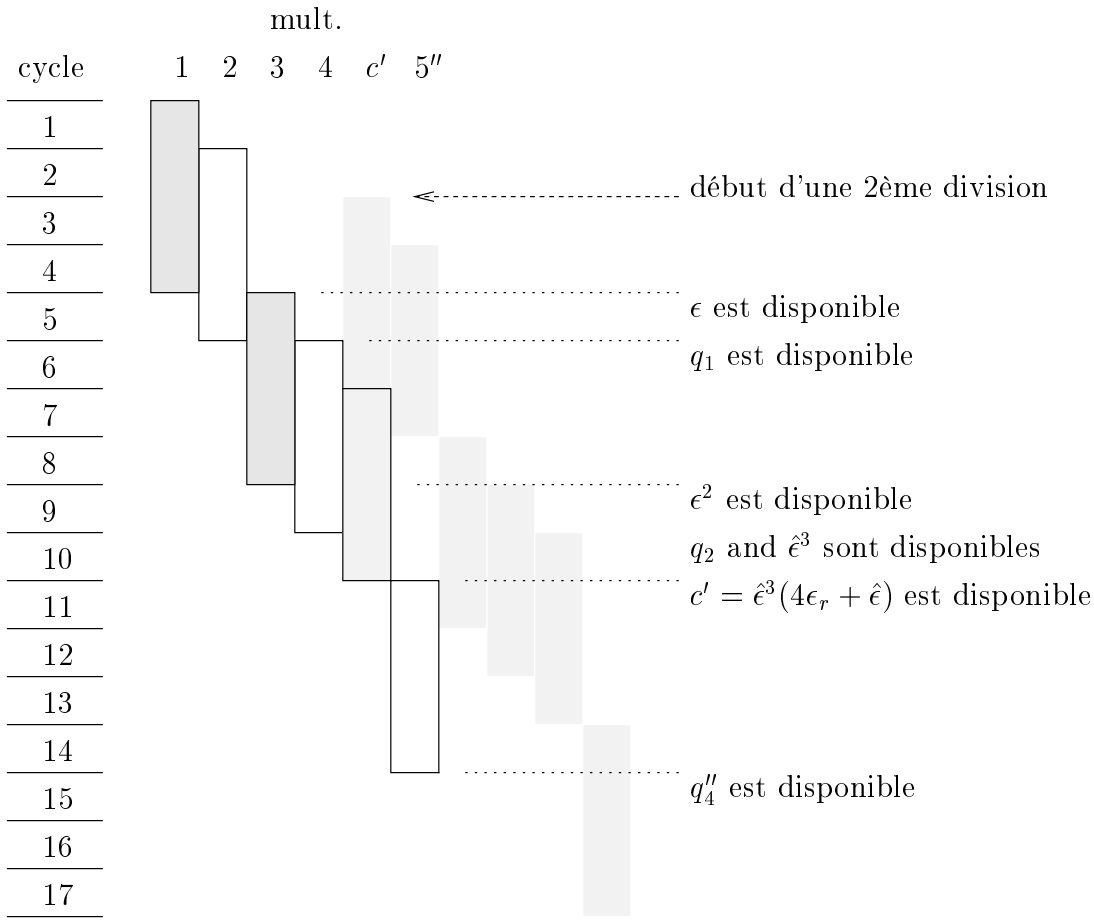


FIG. 1 – Accélération de l'algorithme de Goldschmidt utilisant notre méthode. Deux divisions entrelacées peuvent être effectuées en 17 cycles.

### 4.5 Expansions flottantes

**Participants** : Marc Daumas, Claire Finot.

**Mots clés** : arithmétique exacte, précision multiple, expansion, division, géométrie algorithmique.

**Résumé :** *Une expansion de nombres flottants est un nombre en précision multiple représenté par la somme d'un petit nombre de flottants. Manipuler des expansions permet parfois de rendre plus précise une portion critique d'un programme.*

Claire Finot et Marc Daumas on proposé [8] un nouvel algorithme de division sur les « expansions » de nombres flottants. Une expansion de nombres flottants est un nombre en précision multiple représenté par la somme d'un petit nombre de flottants. Pour manipuler ces expansions, on utilise l'unité de calcul en virgule flottante du processeur pour les calculs internes à la place de l'unité de calcul entier. Les recherches sur ce sujet se sont développées tout d'abord à la suite de l'observation que l'unité de calcul en virgule flottante devient une partie de plus en plus puissante des ordinateurs modernes. De nombreux opérateurs arithmétiques simples et quelques opérateurs géométriques utiles ont déjà été définis sur les expansions de nombres flottants. Nous avons étendu l'ensemble des opérations géométriques sur les expansions en proposant le calcul du déterminant d'une matrice de taille comprise entre 3 et 10 par la méthode de Bareiss.

Dans le cadre de l'ARC Fiable de l'INRIA, nous avons développé un ensemble de routines efficaces pour le calcul en multiprécision. Ces travaux vont être appliqués dans le développement de la bibliothèque quad-double. Ce développement fait partie d'une action du Fonds de collaboration avec l'University of California at Berkeley entre Marc Daumas et Jonathan Shewchuk.

### 4.6 La méthode CENA

**Participant** : Philippe Langlois.

**Mots clés** : Correction linéaire des erreurs d'arrondi, virgule flottante..

**Résumé :** *Nous proposons la méthode CENA qui automatise la correction linéaire des erreurs d'arrondi. Qu'on l'applique au résultat final d'un algorithme linéaire ou a des variables intermédiaires présumées sensibles, elle permet d'améliorer la précision du résultat et de corriger l'instabilité d'un algorithme due aux erreurs d'arrondi.*

Philippe Langlois [32] a amélioré l'efficacité et étendu l'application de la correction linéaire des erreurs d'arrondi, introduite par lui-même et Fabrice Nativel [11]. L'utilisation de la *running error analyse* de Wilkinson permet un calcul de la borne d'erreur résiduelle de la correction linéaire plus fin que les résultats *a priori* introduits auparavant. Les algorithmes itératifs mathématiquement stables mais numériquement instables nécessitent une utilisation élaborée de

la méthode CENA. Il s'agit d'identifier les variables intermédiaires sensibles qui sont alors automatiquement corrigées à chaque pas de calcul. Ainsi, par exemple, nous stabilisons l'itération de Newton appliquée au calcul de la racine multiple d'un polynôme.

## 4.7 Synthèse et Arithmétique

**Participants** : Anne Mignotte, Jean-Michel Muller, Olivier Peyran.

**Mots clés** : synthèse de haut niveau, arithmétique redondante.

Nous nous sommes intéressés à la synthèse d'architectures utilisant des représentations *redondantes* des nombres. Les arithmétiques redondantes permettent une addition extrêmement rapide. En revanche, elles peuvent impliquer une augmentation de la surface et de la consommation des circuits, car elles nécessitent *grosso-modo* plus de place pour la représentation des opérandes. L'arithmétique *mixte* (manipulant sur une même architecture à la fois des opérandes redondants et conventionnels) permet d'espérer un compromis surface/temps efficace. Nous avons développé une méthodologie de synthèse pour l'arithmétique mixte [14]. Elle nous a conduit à la formulation du problème d'ordonnancement avec sélection du codage des opérandes.

## 4.8 Arithmétique et FPGAs

**Participant** : Florent de Dinechin.

**Mots clés** : FPGAs, circuits reconfigurables.

**Résumé :** *Les circuits intégrés reconfigurables permettent la réalisation d'unités de calcul spécialisées à faible coût. On s'intéresse à l'implantation d'opérateurs arithmétiques sur de tels circuits.*

Florent de Dinechin a tout d'abord terminé des travaux commencés à Imperial College de Londres en collaboration avec Wayne Luk, et qui portaient sur l'expression et l'exploitation de contraintes de placement dans les langages ciblant les FPGAs, et ce d'une manière qui soit portable d'un FPGA à un autre. Ces travaux ont été présentés à la conférence FPGA99 [21]. Ces idées vont à présent être reprises dans un contexte différent, pour aider à la construction de bibliothèques arithmétiques VLSI/FPGA en collaboration avec Arnaud Tisserand. Florent de Dinechin a de plus réalisé une étude de complexité relativement simple (pour être générale, car les architectures de FPGAs sont nombreuses et variées) qui fixe des limites assez strictes à la puissance de calcul par unité de surface des FPGAs. La différence avec la complexité VLSI tient dans le routage programmable des FPGAs, qui occupe de la surface même lorsqu'il n'est pas utilisé, et a un coût opératoire très supérieur au routage des circuits VLSI: un signal entre deux blocs fonctionnels, dans un FPGA, doit passer par les nombreux transistors qui assurent la programmabilité du routage. Ainsi la comparaison avec la puissance théorique par unité de surface d'un circuit intégré dédié tourne rapidement au désavantage du FPGA. La conclusion en est que l'on ne pourra pas très longtemps concevoir – et programmer – des FPGAs comme

on le fait actuellement, ce qui ouvre de nombreux sujets de recherche. Ces travaux, présentés à la conférence Sympa'5 [20] ont été soumis à un journal.

# 5 Contrats industriels (nationaux, européens et internationaux)

## 5.1 Aérospatiale

**Participants** : Marc Daumas, Jean-Michel Muller, Claire Finot.

**Mots clés** : virgule flottante, fiabilité numérique.

La compagnie Aérospatiale nous a confié une étude confidentielle portant sur les problèmes liés à l'utilisation de l'arithmétique virgule flottante sur certains systèmes embarqués. Suite à cette étude, nous avons organisé deux semaines de formation à l'arithmétique virgule flottante des personnels d'Aérospatiale concernés.

# 6 Actions régionales, nationales et internationales

## 6.1 Collaboration avec UCLA et SMU

**Participants** : Marc Daumas, Jean-Michel Muller, Arnaud Tisserand.

**Mots clés** : arithmétique en ligne, multiprécision, virgule flottante.

**Résumé :** *Un projet PICS nous lie à l'*University of California at Los Angeles*, à la* Southern Methodist University *de Dallas et au Laboratoire d'Informatique de Marseille.*

Dans le cadre du projet PICS 479 « vers des arithmétiques plus fiables et plus rapides » (projet dirigé par M. Ercegovac du côté américain et J.M. Muller du côté français), nous collaborons avec les équipes d'Ercegovac à l'*University of California at Los Angeles*, de Matula à la *Southern Methodist University* de Dallas, et avec le groupe de J.C. Bajard au Laboratoire d'Informatique de Marseille. Ce projet PICS a démarré en 1997, et il a produit plusieurs résultats publiés ou sur le point de l'être (en 1999, une étude sur la division [9]). Nos principaux sujets de collaboration sont le calcul multiprécision, la conception d'opérateurs virgule flottante et le calcul « en ligne ».

## 6.2 Collaboration avec Berkeley

**Participants** : Marc Daumas, Claire Finot, Jean-Michel Muller.

**Mots clés** : multiprécision, virgule flottante, manipulation d'expansions.

**Résumé :** *Nous avons répondu à l'appel d'offres France-Berkeley 1999 et notre proposition a été acceptée. Nous collaborons dans ce cadre avec J. Shewchuk.*

Nous avons développé (voir section 4.5), en utilisant la notion d'expansion flottante, un ensemble de routines efficaces pour le calcul en multiprécision. Ces travaux vont être appliqués pour le développement d'une bibliothèque «quad-double» dans une action du Fonds de collaboration avec l'University of California at Berkeley (http://www.ias.berkeley.edu:80/cwes/fbf/) entre Marc Daumas et Jonathan Shewchuk (http://www.cs.berkeley.edu/ jrs/). Jean Michel Muller et David Bailey (http://www.nersc.gov/ dhb/) participent également à cette action.

# 7 Diffusion de résultats

## 7.1 Organisation de conférences, édition de numéros spéciaux de journaux

**Participants** : Marc Daumas, Florent de Dinechin, Anne Mignotte, Jean-Michel Muller.

**Mots clés** : organisation de conférences, numéros spéciaux.

Jean-Michel Muller a été président du 14th Ieee Symposium Arithmetic (ARITH-14) qui s'est déroulé à Adelaide, en Australie, en avril 1999. La conférence ARITH est la principale conférence dédiée à l'arithmétique des ordinateurs.

Christiane Frougny (LIAFA), Jean-Claude Bajard (LIM) et Jean-Michel Muller ont été éditeurs invités d'un numéro spécial *Real Numbers and Computers* de Theoretical Computer Science (TCS), paru en Janvier 1999 [6].

Anne Mignotte a co-organisé (avec Jean Mermet, de l'INP Grenoble) le *FDL'99 Forum on Design Languages* du 30 août au 3 septembre 1999. Ce Forum regroupe trois événements : VHDL User Forum (sur l'utilisation du langage VHDL), Design Reuse (colloque sur la propriété intellectuelle et la réutilisation de blocs), SSDL (System Specific Design Languages, sur la description de systèmes intégrés). Florent de Dinechin a été responsable de l'édition des actes de ce colloque.

Anne Mignotte a été responsable de l'édition des actes et des CD ROM pour la conférence DATE'99 qui regroupe depuis un an les conférences internationales ED&TC, EuroASIC et EuroDAC.

## 7.2 Enseignement de 3ème cycle

**Participants** : Anne Mignotte, Marc Daumas.

Anne Mignotte a enseigné au DEA d'Informatique de Lyon (DIL). A partir d'Octobre 1999, Marc Daumas enseigne l'Arithmétique des Ordinateurs au DEA d'Informatique Fondamentale de Lyon (les DEA lyonnais ont été refondus à la rentrée 1999).

## 7.3 Animation de la communauté

**Participants** : Marc Daumas, Anne Mignotte, Jean-Michel Muller.

Anne Mignotte a animé le séminaire du LIP jusqu'en juin 1999. Arnaud Tisserand va prendre cette responsabilité. Jean-Michel Muller est responsable du thème «Arithmétique» du PRC-GDR ARP.

# 8 Bibliographie

## Ouvrages et articles de référence de l'équipe

[1] J. Bajard, C. Frougny, J. Muller, G. Villard (éditeurs), *Special issue "Real Numbers and Computers" of Theoretical Computer Science, 162*, août 1996.

[2] J. C. Bajard, S. Kla, J. M. Muller, « BKM: A New Hardware Algorithm for Complex Elementary Functions», *IEEE Transactions on Computers 43*, 8, août 1994, p. 955–963.

[3] M. Daumas, J.-M. Muller (éditeurs), *Qualité des calculs sur ordinateur : vers des arithmétiques plus fiables*, Masson, 1997.

[4] T. Lang, J. Muller, N. Takagi (éditeurs), *Proceedings of the 13-th IEEE Symposium on Computer Arithmetic (ARITH-13)*, Asilomar, USA, IEEE Computer Society Press, Los Alamitos, CA, juillet 1997.

[5] J.-M. Muller, *Elementary functions, algorithms and implementation*, Birkhauser, 1997.

## Livres et monographies

[6] J.-C. Bajard, C. Frougny, J.-M. Muller (éditeurs), *Numéro spécial* Real Numbers and Computers *de Theoretical Computer Science*, Janvier 1999.

## Articles et chapitres de livre

[7] P. Y. Calland, A. Mignotte, O. Peyran, Y. Robert, F. Vivien, «retiming DAGs», *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 1999, à paraître.

[8] M. Daumas, C. Finot, « Division of floating point expansions with an application to the computation of a determinant», *Journal of Universal Computer Science 5*, 6, 1999, p. 323–338.

[9] M. Ercegovac, L. Imbert, D. Matula, J.-M. Muller, G. Wei, «Improving Goldschmidt Algorithm Division, Square Root and Square Root Reciprocal», *IEEE Transactions on Computers*, 1999, à paraître.

[10] L. Imbert, J.-M. Muller, F. Rico, «Radix-10 BKM Algorithm for Computing Transcendentals on Pocket Computers», *Journal of VLSI Signal Processing*, 1999, à paraître.

[11] P. Langlois, F. Nativel, « When automatic linear correction of rounding errors is exact », *CR. Acad. Sci. Paris, Série 1*, 328, 1999, p. 543–548.

[12] V. Lefèvre, «An Algorithm that Computes a Lower Bound on the Distance Between a Segment and $Z^2$ », *in : Developments in Reliable Computing*, T. Csendes (éditeur), Kluwer, Dordrecht, Netherlands, 1999, p. 203–212.

[13] V. Lefèvre, J.-M. Muller, A. Tisserand, « Toward Correctly Rounded Transcendentals», *IEEE Transactions on Computers 47*, 11, novembre 1998.

[14] A. Mignotte, J.-M. Muller, O. Peyran, « Synthesis for mixed arithmetics », *Design Automation for Embedded Systems*, 1999, à paraître.

[15] J.-M. Muller, «A Few Results on Table-Based Methods», *Reliable Computing 5*, 3, 1999.

[16] J.-M. MULLER, « Ordinateurs en quête d'arithmétique », *La Recherche (numéro hors-série)*, Août 1999, p. 90–96, Version revue et augmentée d'un article de 1995.

[17] J.-M. MULLER, « Vers des primitives propres en arithmétique des ordinateurs », *Technique et Science Informatiques*, 2000, à paraître.

## Communications à des congrès, colloques, etc.

[18] M. DAUMAS, A. NEGOÏ, J. ZIMMERMANN, « Un tout petit système pour la simulation numérique», *in : Sympa'5, 5ème Symposium En Architectures Nouvelles de Machines*, p. 27–36, Rennes, France, 1999.

[19] M. DAUMAS, « Multiplications of floating point expansions », *in : Proceedings of the 14th Symposium on Computer Arithmetic*, Adelaide, Australia, 1999.

[20] F. DE DINECHIN, « Le prix du routage dans les FPGAs », *in : Sympa'5, 5ème Symposium En Architectures Nouvelles de Machines*, Rennes, France, juin 1999.

[21] F. DE DINECHIN, « Towards Adaptable Hierarchical Placement for FPGAs », *in : FPGA'99*, Monterey, CA, février 1999.

[22] A. FRABOULET, G. HUARD, A. MIGNOTTE, «Loop Alignment for Memory Accesses Optimization», *in : International Symposium on System Synthesis (ISSS)*, novembre 1999.

[23] A. FRABOULET, G. HUARD, A. MIGNOTTE, « Optimisation de la consommation et de la place mémoire par transformations de boucles», *in : Colloque CAO de circuits intégrés et systèmes*, Aix en Provence, mai 1999.

[24] V. LEFÈVRE, J.-M. MULLER, « Table Methods for the Elementary Functions », *in : SPIE's International Symposium on Optical Science, Engineering, and Instrumentation*, Denver, Juillet 1999.

[25] J.-M. MULLER, « Calcul de fonctions à l'aide de tables », *in : Sympa'5, 5ème Symposium En Architectures Nouvelles de Machines*, Rennes, Juin 1999.

## Rapports de recherche et publications internes

[26] M. DAUMAS, C. FINOT, « Algorithm, Proof and Performances of a new Division of Floating Point Expansions», *Research report n° 3771*, INRIA, Le Chesnay, France, 1999.

[27] M. DAUMAS, C. FINOT, « Division of floating point expansions », *Research report n° 99-03*, Laboratoire de l'Informatique du Parallélisme, Lyon, France, 1999.

[28] F. DE DINECHIN, « The Price of Routing in FPGAs », *Research report n° RR-3772*, INRIA, septembre 1999.

[29] M. ERCEGOVAC, L. IMBERT, D. MATULA, J.-M. MULLER, G. WEI, « Improving Goldschmidt Algorithm Division, Square Root and Square Root Reciprocal », *rapport de recherche n° 3753*, INRIA, Septembre 1999, à paraître dans IEEE Transactions on Computers.

[30] A. FRABOULET, G. HUARD, A. MIGNOTTE, «Loop Alignment for Memory Accesses Optimization», *Research Report n° 1999–26*, École Normale Supérieure de Lyon, avril 1999, available at ftp://ftp.ens-lyon.fr/pub/LIP/Rapports/RR/RR1999/RR1999-26.ps.Z.

[31] L. IMBERT, J.-M. MULLER, F. RICO, «Radix-10 BKM Algorithm for Computing Transcendentals on Pocket Computers», *rapport de recherche n° 3754*, INRIA, Septembre 1999, à paraître dans Journal of VLSI Signal Processing.

[32] P. LANGLOIS, «Autimatic linear correction of rounding errors», *Research report*, INRIA, décembre 1999.

[33] V. LEFÈVRE, «Multiplication by an Integer Constant», *rapport de recherche n° 1999-06*, Laboratoire de l'Informatique du Parallélisme, 1999.