L'algorithme CORDIC dans les architectures spécialisées de traitement numérique du signal

CORDIC-based digital signal processing hardware algorithms

Gilles PRIVAT

Centre National d'Études des Télécommunications, CNET-Grenoble, BP n° 98, 38243 MEYLAN CEDEX.

Ingénieur de l'École Nationale Supérieure des Télécommunications, Paris, 1981. Docteur de l'École Nationale Supérieure des Télécommunications, spécialité Automatique et Traitement du Signal, Paris, 1986. Chercheur INPG-IMAG, 1982-1983. Ingénieur de recherche au CNET-Grenoble, division Conception de Circuits Intégrés, depuis 1983. Travaux en compilation de silicium, architectures VLSI pour le traitement numérique du signal, le codage d'images.

Marc RENAUDIN

Centre National d'Études des Télécommunications, CNET-Grenoble, BP n° 98, 38243 MEYLAN CEDEX.

Maîtrise d'automatique à l'Université Joseph-Fourier de Grenoble, ingénieur INPG en automatique et traitement du signal (Juin 1987), titulaire d'un DEA en traitement du signal (Sept. 1987, CEPHAG), Marc Renaudin effectue depuis octobre 1987 une thèse au sein de la division Conception de Circuits Intégrés au CNET-Grenoble. Son travail de recherche porte sur l'étude d'architectures VLSI pour l'intégration d'algorithmes avancés de traitements et de codages de séquences d'images animées.

RÉSUMÉ

Cet article présente une étude de synthèse sur une nouvelle interaction entre algorithmes et architectures résultant de l'utilisation de rotations vectorielles comme primitives de calcul. Après une présentation détaillée des différents aspects de l'algorithme CORDIC, la gamme des architectures possibles pour son implémentation est passée en revue. La puissance et la généralité de l'opérateur CORDIC sont illustrées par des exemples de nombreux algorithmes de traitement du signal et d'algèbre linéaire, en particulier toutes les versions normalisées d'algorithmes classiques, où il peut s'introduire.

MOTS CLÉS

CORDIC, architectures VLSI, transformations orthogonales, algorithmes normalisés.

SUMMARY

A survey of the new algorithm-architecture interaction arising from the use of vector rotations as primitive computational units is presented. After a thorough review of various features of the CORDIC algorithm, the full range of architectures allowing its implementation is examined. The CORDIC operator's versatility is illustrated through an overview of the numerous digital signal processing and linear algebra algorithms where it can be applied, with special emphasis on normalized variants of classical algorithms.

KEY WORDS

CORDIC, VLSI architectures, orthogonal transforms, normalized algorithms.

1. Introduction

L'algorithme CORDIC, (acronyme de « COordinate Rotation DIgital Computer ») fut introduit à l'origine [1], pour répondre à des besoins de calculs tels que les rotations de vecteurs ou les changements de coordonnées cartésiennes-polaires et polaires-cartésiennes dans le plan euclidien, utilisés dans le domaine de la navigation aérienne, et qui auraient auparavant réalisés au moyen de servo-systèmes analogiques.

L'algorithme recourt uniquement à des primitives élémentaires en base 2: additions, soustractions et décalages et permet accessoirement le calcul de racines carrées et de fonctions trigonométriques sinus, cosinus et arctangente.

Une généralisation [2] en a été proposée, par extension à des espaces pseudo-euclidiens dotés d'une métrique hyperbolique ou linéaire, permettant le calcul direct de la multiplication, de la division, des sinus, cosinus et argument-tangente hyperboliques, ainsi que d'exponentielles, et le calcul indirect (par composition des fonctions précédentes), de tangentes circulaires et hyperboliques et de logarithmes.

L'algorithme CORDIC a donc été conçu à l'origine essentiellement comme un outil pour le calcul numérique, et il a été effectivement utilisé en tant que tel sous forme programmée, ou cablée dans des calculatrices ou des coprocesseurs flottants pour le calcul de fonctions transcendantes [3].

Dans l'optique sensiblement différente, qui nous intéressera exclusivement ici, de l'implantation d'architectures VLSI spécialisées de traitement numérique du signal, l'algorithme CORDIC a fait l'objet depuis quelques années un important regain d'intérêt [12]. Il apparaît dans cette perspective plus large comme un nouveau paradigme en fonction duquel peut être repensée l'algorithmique d'un grand nombre de fonctions avancées du traitement numérique du signal, aux fins de l'utilisation de l'opérateur CORDIC intégré comme module de base dans une éventuelle réalisations sous forme de circuit spécifique.

On peut citer comme exemples d'applications: le filtrage complexe de signaux analytiques, la modulation en bande latérale unique, la transformée de Fourier discrète, directe et rapide [18, 19], le filtrage fréquentiel sur structures à faible sensibilité en bande-passante (treillis de Gray-Markel, filtres orthogonaux et filtres d'onde [25-27, 33]), la factorisation spectrale de processus stationnaires par l'algorithme de Schur normalisé [11, 12], la modélisation adaptative de processus non stationnaires (filtrage récursif optimal au sens des moindres carrés sur structure en treillis normalisé [28], filtrage de Kalman [23]). Plus généralement il intervient dans la résolution d'un grand nombre de problèmes d'algèbre linéaire, dont certaines fonctions précédentes apparaissent comme des cas particuliers: résolution exacte ou approchée au sens des moindres carrés de systèmes linéaires (algorithmes orthogonaux de Givens [9], Fadeeva [17]), équations aux valeurs propres, décomposition en valeurs singulières [20, 21], décomposition QR, de Cholesky [9, 24], le cas des matrices de Toeplitz ou quasi-Toeplitz (à rang de déplacement fini), donnant toujours lieu à des algorithmes spécifiques [22].

L'idée commune à tous ces algorithmes (à l'exception de ceux opérant directement sur des valeurs complexes), est de *normaliser* les calculs par l'utilisation exclusive à chaque étape de *transformations orthogonales*. Dans le cas du filtrage, on garantit ainsi une variance unitaire des variables de calcul intermédiaires, et cette mise à l'échelle correcte des signaux permet d'éviter d'éventuelles oscillations de dépassement. On peut montrer [27] que ces structures possèdent également de bonnes propriétés par rapport aux autres effets d'arithmétique finie: faible gain de bruit d'arrondi (décorrélé) [37], absence de cycles limites (bruit d'arrondi corrélé), faible sensibilité par rapport aux valeurs des coefficients. Cette dernière propriété se retrouve pour les algorithmes généraux d'algèbre linéaire (conditionnement).

L'utilisation de l'opérateur CORDIC présente également de nombreux avantages pour l'intégration au niveau purement architectural, par son excellente adaptation aux contraintes des VLSI: un même module de base, précaractérisé et éventuellement paramétré avec possibilité de choix des degrés de parallélisme, de pipelinage [30], pour une adaptation optimale aux débits requis par l'application, peut servir comme constituant élémentaire dans une grande variété d'architectures spécialisées, avec une complexité en matériel inférieure à ce qu'on obtiendrait par l'utilisation d'opérateurs classiques [14].

2. Principe de l'algorithme CORDIC

2.1. Formulation matricielle

On se place dans un espace vectoriel réel de dimension 2 où sont définies une forme bilinéaire,

$$[(x, y), (x', y')] \rightarrow (x, y)\, Q_m \begin{pmatrix} x' \\ y' \end{pmatrix},$$

avec $Q_m = \begin{bmatrix} 1 & 0 \\ 0 & m \end{bmatrix}$ et la semi-norme associée:

$$\| (x, y) \| = \sqrt{x^2 + m y^2}$$

Pour $m = 0, +1, -1$, la matrice de rotation généralisée suivante

$$\Theta_m(\alpha) = \begin{pmatrix} \cos(\sqrt{m}\,\alpha) & -\sqrt{m}\sin(\sqrt{m}\,\alpha) \\ \dfrac{1}{\sqrt{m}}\sin(\sqrt{m}\,\alpha) & \cos(\sqrt{m}\,\alpha) \end{pmatrix}$$

possède la propriété d'orthogonalité par rapport à Q_m: $\Theta_m Q_m \Theta_m^t = Q_m$.

Cas euclidien ($m = 1$): on a une matrice de rotation de Givens.

$$\Theta_1(\alpha) = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix}$$

Cas hyperbolique ($m=-1$): on a une matrice dite de Golub, qui est J-orthogonale (si l'on note $Q_{-1}=J$)

$$\Theta_{-1}(\alpha)=\begin{pmatrix}\mathrm{ch}(\alpha) & \mathrm{sh}(\alpha)\\ \mathrm{sh}(\alpha) & \mathrm{ch}(\alpha)\end{pmatrix}$$

du fait des égalités

$$\mathrm{Sh}(x)=-j\,\mathrm{Sin}(jx) \qquad \text{et} \qquad \mathrm{Ch}(x)=\mathrm{Cos}(jx),$$

avec $j=\sqrt{-1}$.

Cas linéaire ($m=0$):

$$\Theta_0(\alpha)=\begin{pmatrix}1 & 0\\ \alpha & 1\end{pmatrix}$$

Ces trois cas correspondent aux trois modes d'utilisation de l'algorithme CORDIC.

On cherche à faire subir une rotation généralisée d'angle α à un vecteur (x, y). Pour cela, on décompose l'argument angulaire en $\alpha=\sum \mu_i \alpha_i$, $\mu_i=\pm 1$, et on fait subir une séquence de rotations d'angle $\pm\alpha_i$ au vecteur (X, Y). Le principe de l'algorithme itératif réside dans la propriété (en fait triviale), que l'addition des arguments angulaires équivaut à une composition des rotations, donc au produit des matrices correspondantes. On peut donc calculer la rotation de départ par une séquence de rotations élémentaires d'angle $\pm\alpha_i$ supposées plus simples à calculer.

On effectue donc à chaque itération la transformation élémentaire $\Theta_m(\mu_i\alpha_i)$ suivante

$$\Theta_m(\mu_i\alpha_i)=\begin{pmatrix}\cos(\sqrt{m}\,\alpha_i) & -\mu_i\sqrt{m}\sin(\sqrt{m}\,\alpha_i)\\ \mu_i\dfrac{1}{\sqrt{m}}\sin(\sqrt{m}\,\alpha_i) & \cos(\sqrt{m}\,\alpha_i)\end{pmatrix}$$

avec $\mu_i=\pm 1$, définissant le sens de la rotation.

On exprime la matrice en mettant en facteur le terme en cos:

$$\Theta_m(\mu_i\alpha_i)=\cos(\sqrt{m}\,\alpha_i)\begin{pmatrix}1 & -\mu_i\sqrt{m}\,\mathrm{tg}(\sqrt{m}\,\alpha_i)\\ \mu_i\dfrac{1}{\sqrt{m}}\mathrm{tg}(\sqrt{m}\,\alpha_i) & 1\end{pmatrix}$$

$$\Theta_m(\mu_i\alpha_i)=\frac{1}{\sqrt{1+\mathrm{tg}^2(\sqrt{m}\,\alpha_i)}}\begin{pmatrix}1 & -\mu_i\sqrt{m}\,\mathrm{tg}(\sqrt{m}\,\alpha_i)\\ \mu_i\dfrac{1}{\sqrt{m}}\mathrm{tg}(\sqrt{m}\,\alpha_i) & 1\end{pmatrix}$$

$$\Theta_m(\alpha)=\Theta_m\left(\sum_{i=1}^{n}\mu_i\alpha_i\right)=\prod_{i=1}^{n}\Theta_m(\mu_i\alpha_i)=\prod_{i=1}^{n}\frac{1}{\sqrt{1+\mathrm{tg}^2(\sqrt{m}\,\alpha_i)}}\times\prod_{i=1}^{n}\begin{pmatrix}1 & -\mu_i\sqrt{m}\,\mathrm{tg}(\sqrt{m}\,\alpha_i)\\ \mu_i\dfrac{1}{\sqrt{m}}\mathrm{tg}(\sqrt{m}\,\alpha_i) & 1\end{pmatrix}$$

2.2. Formulation itérative

On peut écrire sous la forme suivante les itérations matricielles précédentes, le vecteur $(x_i y_i)$ étant transformée à la i-ième itération en:

$$\begin{pmatrix}x_{i+1}\\ y_{i+1}\end{pmatrix}=\Theta_m(\alpha_i)\begin{pmatrix}x_i\\ y_i\end{pmatrix}$$

En faisant provisoirement abstraction du facteur de normalisation (terme en cos), on peut écrire:

$$\begin{cases}\tilde{x}_{i+1}=\tilde{x}_i-\mu_i\sqrt{m}\,\mathrm{tg}(\alpha_i\sqrt{m})\,\tilde{y}_i\\ \tilde{y}_{i+1}=\tilde{y}_i+\dfrac{1}{\sqrt{m}}\mu_i\,\mathrm{tg}(\alpha_i\sqrt{m})\,\tilde{x}_i\end{cases}$$

Le principe de base de l'algorithme CORDIC est de prendre $(1/\sqrt{m})\,\mathrm{tg}(\alpha_i\sqrt{m})$ égal à une puissance de 2, ou plus généralement de la base de calcul de l'arithmétique utilisée, de telle sorte que les itérations ne comportent que des additions et des décalage. On obtient:

$$\begin{cases}\tilde{x}_{i+1}=\tilde{x}_i-m\,\mu_i 2^{-\sigma(m,i)}\tilde{y}_i\\ \tilde{y}_{i+1}=\tilde{y}_1+\mu_i 2^{-\sigma(m,i)}\tilde{x}_i\end{cases}$$

Cela revient à effectuer la décomposition de l'angle α sur une base constituée des $\alpha_i=(1/\sqrt{m})\,\mathrm{arctg}(\sqrt{m}\,2^{-\sigma(m,i)})$. On reviendra au paragraphe 2.5 sur les problèmes que pose le choix de cette décomposition.

2.3. Le facteur de normalisation

Dans l'écriture précédente des équations itératives, nous avons fait abstraction du facteur de normalisation, qui permet d'assurer la conservation de la norme entre l'entrée et la sortie de l'algorithme.

Son calcul peut être intégré à chaque itération [8], sous la forme suivante:

$$\begin{cases}x_{i+1}=(1-m\,\gamma_{m,i}2^{-\sigma(m,i)})\,(x_i-m\,\mu_i 2^{-\sigma(m,i)}y_i)\\ y_{i+1}=(1-m\,\gamma_{m,i}2^{-\sigma(m,i)})\,(y_i+\mu_i 2^{-\sigma(m,i)}x_i)\\ \text{avec } \gamma_{m,i}=1 \text{ ou } 0\end{cases}$$

les γ étant choisis de telle sorte que

$$K_m = \prod_{i=1}^{n} (1 - m\,\gamma_{m,i}\,2^{-\sigma(m,i)})$$

$$= \prod_{i=1}^{n} \frac{1}{\sqrt{1+\mathrm{tg}^2(\sqrt{m}\,\alpha_i)}}$$

$$= \prod_{i=1}^{n} \frac{1}{\sqrt{1+2^{-2\sigma(m,i)}}}$$

Il apparaît que K_m ne dépend pas de l'angle α, mais seulement de la base de décomposition choisie, et peut donc être calculé une fois pour toutes, la correction intervenant alors seulement en fin de calcul. La base de décomposition peut être choisie (§ 2.6) de manière à simplifier au maximum cette remise à l'échelle finale (facteur de normalisation égal à une puissance de 2).

2.4. Les différents modes de fonctionnement de l'algorithme

2.4.1. *Mode rotation*

Dans ce mode de base, l'algorithme converge en faisant tendre la variable auxiliaire z_i, initialisée au départ à α (argument de la rotation) vers zéro. Cette mise à jour est effectuée en parallèle avec les itérations de calcul des rotations, sous le forme. $z_{i+1} = z_i + \mu_i \alpha_i$. Le signe de μ_i est alors défini par celui de z_i, ce qui correspond à une commande par feedback asservissant les itérations de manière à faire tendre z_i vers 0.

2.4.2. *Mode évaluation*

On fait tendre la variable y_i vers zéro, le signe de μ_i étant défini à chaque itération par celui de y_i, ce qui correspond au calcul de arctg (y/x), qui est accumulé dans la variable z. Les deux premières colonnes

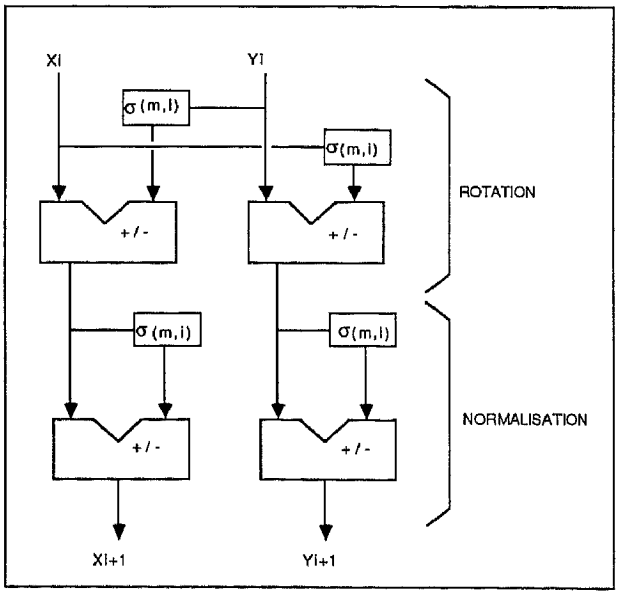

Fig. 1. — Structure d'une itération de l'algorithme CORDIC.

du tableau I résument les six modes de fonctionnement de base de l'algorithme, obtenus en combinant

TABLEAU I

Les différents modes de fonctionnement de l'algorithme CORDIC.

	EVALUATION	ROTATION	EVALUATION-ROTATION
CIRCULAIRE	$x \to (x^2+y^2)^{1/2}$	$x' \to x'\cos(z') - y'\sin(z')$	$x' \to (x^2+y^2)^{-1/2}(x'x - y'y)$
	$y \to 0$	$y' \to x'\sin(z') + y'\cos(z')$	$y' \to (x^2+y^2)^{-1/2}(x'y + y'x)$
	$z \to \mathrm{arctg}(y/x)$	$z' \to 0$	
HYPERBOLIQUE	$x \to (x^2-y^2)^{1/2}$	$x' \to x'\mathrm{ch}(z') + y'\mathrm{sh}(z')$	$x' \to (x^2-y^2)^{-1/2}(x'x + y'y)$
	$y \to 0$	$y' \to x'\mathrm{sh}(z') + y'\mathrm{ch}(z')$	$y' \to (x^2-y^2)^{-1/2}(x'y + y'x)$
	$z \to \mathrm{argth}(y/x)$	$z' \to 0$	
LINEAIRE	$x \to x$	$x' \to x'$	$x' \to x'$
	$y \to 0$	$y' \to y' + x'z'$	$y' \to y' + x'y/x$
	$z \to y/x$	$z' \to 0$	

ces deux modes de convergence avec les trois métriques utilisées. Les expressions données intègrent la normalisation du résultat.

2.4.3. *Mode évaluation-rotation*

Il existe une complémentarité fondamentale entre les deux modes de fonctionnement évaluation et rotation. Dans la majorité des applications, l'angle $\alpha = \mathrm{arctg}\,(y/x)$ est d'abord calculé par une première utilisation de l'algorithme en mode évaluation, pour être ensuite utilisé comme argument de calculs en mode rotation. C'est cet enchaînement de calculs qu'on a voulu montrer ci-dessus au travers du mode de fonctionnement «évaluation-rotation», qui est la composition des deux précédents : on effectue d'abord une évaluation sur les arguments x et y, puis une rotation de $z = \mathrm{arctg}\,(y/x)$ appliquée à x' et y'.

L'argument angulaire n'intervient donc que comme intermédiaire de calcul et il est alors inutile de l'exprimer par sa représentation externe, la représentation interne par la séquence de signature (μ_i) pouvant être transmise entre les deux phases du calcul. Une conséquence importante en est qu'il est possible dans ce cas de supprimer l'accumulation de la variable angulaire *z dans les deux modes de fonctionnement.*

Avec la métrique linéaire, on retrouve dans le mode évaluation un algorithme de division sans restauration, le résultat étant exprimé en chiffres signés sur une base qui est dans ce cas la base classique de la numération binaire.

2.5. Choix de la base de décomposition, propriétés de convergence et de résolution

Le choix de la séquence (α_i) des angles élémentaires qui permet de décomposer l'angle de rotation est un élément essentiel de l'algorithme car de lui dépendent le nombre d'itérations de l'algorithme, son domaine de convergence et la résolution angulaire.

Un cadre théorique général peut être donné à la décomposition utilisée dans l'algorithme CORDIC au travers de la notion de base discrète [10]. Étant donné une séquence infinie décroissante positive $B = (\alpha_i)$, B

est une base discrète sur l'intervalle I si quel que soit α appartenant à I, il existe une suite (μ_i), $\mu_i = \pm 1$, telle que:

$$\alpha = \sum_{i=1}^{\infty} \mu_i \alpha_i.$$

On montre que ceci est vérifié si et seulement si:

$$\alpha_n < \sum_{k=n+1}^{\infty} \alpha_k, \qquad \forall n.$$

On s'intéresse ici à la décomposition de l'argument sur une suite finie d'éléments. On fait alors une approximation et la condition devient:

$$\alpha_n - \sum_{k=n+1}^{p} \alpha_k < \alpha_p, \qquad \forall n \in [1 \ldots p-1].$$

Cette relation représente la *condition de convergence* de l'algorithme: à l'itération n, la somme des rotations restantes doit permettre de ramener l'angle à moins de α_p de zéro.

Le domaine de convergence est donné par la rotation maximale totale possible notée α_0:

$$|\alpha_0| = \sum_{k=1}^{p} \alpha_k + \alpha_p.$$

Pour les bases classiques [1], l'algorithme ne converge pas en général pour l'ensemble du domaine $-\pi < \alpha < +\pi$ (dans le cas circulaire), mais il est possible de ramener l'angle de départ dans le domaine de convergence par une rotation initiale.

La condition de convergence garantit également une *résolution angulaire* de α_p pour p itérations. On a pour un angle α et sa décomposition μ_i:

$$\left| \alpha - \sum_{i=1}^{p} \mu_i \alpha_i \right| < \alpha_p.$$

Les bases classiquement choisies dans l'algorithme CORDIC pour les différentes métriques sont les suivantes, pour trois métriques:

$m = 1$ (circulaire)

$$\alpha_i = \operatorname{arctg}(2^{-\sigma(1,\,i)})$$

$m = -1$ (hyperbolique)

$$\alpha_i = \operatorname{argth}(2^{-\sigma(-1,\,i)})$$

$m = 0$ (linéaire)

$$\alpha_i = 2^{-\sigma(0,\,i)}.$$

La séquence $\sigma(m, i)$ doit être choisie de manière à satisfaire la condition de convergence, et le cas échéant des contraintes sur le domaine de convergence et le facteur de normalisation.

Pour $m = 1$ ou 0, la condition de convergence est satisfaite avec la base «naturelle» $\sigma(m, i) = i$ [1]. Pour $m = -1$, il est nécessaire de répéter certaines puissances pour assurer la convergence [2].

Afin d'obtenir un facteur K qui soit une puissance de 2, il a été proposé [29] de systématiser cette répétition de termes de la base, dans le cas circulaire aussi bien qu'hyperbolique. On peut montrer que les bases ainsi obtenues satisfont toujours la condition de convergence, et que le domaine de convergence s'en trouve élargi, au prix bien sur d'un plus grand nombre d'itérations.

Il a également été proposé [4, 5], d'utiliser une base légèrement différente comportant des différences de puissances de 2, de la forme suivante:

$$\alpha_i = \frac{1}{\sqrt{m}} \operatorname{arctg}\left(\sqrt{m}\,(2^{-\sigma(m,\,i)} - \eta(m, i)\, 2^{-\sigma'(m,\,i)})\right),$$

avec $\eta(m, i) = 0$ ou 1.

On aboutit ainsi au même résultat (facteur K de normalisation puissance de 2), et ce sans avoir l'inconvénient, qui s'avère gênant pour les implémentations pipelinées de l'algorithme, d'un nombre d'itérations différent dans les cas circulaire et hyperbolique.

Exemples de séquences pour une résolution de 16 bits.

$m = 1$

$$\text{rot}\,\pi/2,\ 2^0 - 2^{-5},\ 2^0 - 2^{-2},\ 2^{-1} - 2^{-5},\ 2^{-2} - 2^{-8},\ 2^{-3} \ldots 2^{-14}.$$

$m = -1$

$$2^0 - 2^{-3},\ 2^0 - 2^{-2},\ 2^{-1} - 2^{-6},\ 2^{-1} - 2^{-3},\ 2^{-2} - 2^{-6},\ 2^{-3} - 2^{-8},\ 2^{-4} - 2^{-9},\ 2^{-5} \ldots 2^{-14}.$$

Il est possible [30] de combiner ces deux méthodes dans un logiciel général de construction des séquences $\{\sigma(m, i)\}$ pour les deux métriques circulaire et hyperbolique, en rendant toujours ainsi le facteur K égal à une puissance de deux. On peut prendre en compte, en plus du nombre de bits des données d'entrées, une condition sur la résolution, si celle-ci est inférieure à la précision des données d'entrée.

En général, les bases construites à partir de différences de puissances de deux seront à préférer: pour un nombre de bit, une résolution et un domaine de convergence donnés, la longueur de la séquence de calcul (nombre total d'additions et de décalages), est plus faibles pour ce type de base que pour les bases obtenues par simple répétition de puissances.

2.6. L'ALGORITHME CORDIC HYBRIDE ET PARALLÉLISÉ

Une formulation légèrement différente de l'algorithme a été proposée [16], dans une optique de parallélisation maximale de l'algorithme. Les itérations sur x et y sont découplées, la séquence de signature étant calculée séparément. Mais on perd évidemment la propriété essentielle de l'algorithme, qui est l'absence de multiplications. Compte tenu du fait que, par l'utilisation du pipelinage (*cf.* § 3.1.2), l'algorithme classique permet d'atteindre aussi élevés que nécessaire, le recours à cette forme très lourde semble *a priori* inutile.

Il est également possible, toujours dans un but d'accélération du calcul, d'avoir recours à des formu-

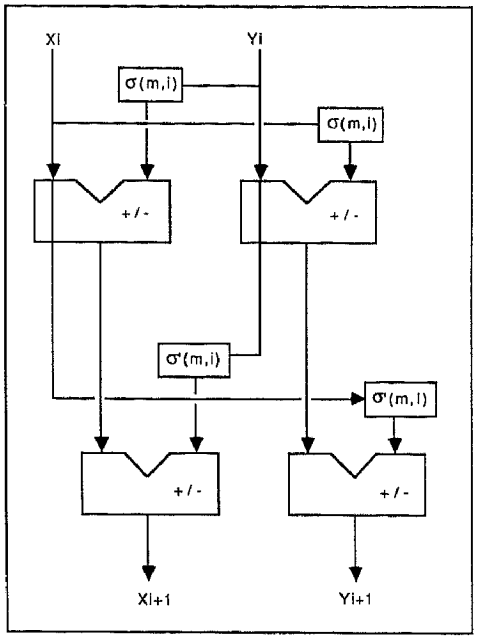

Fig. 2. — Structure d'une itération de l'algorithme CORDIC dans le cas où l'élément de base est construit à partir d'une différence de puissances de deux.

lations dites hybrides, sont intermédiaires entre l'algorithme original et une implémentation directe de la rotation de Givens au moyen de deux ou quatre multiplieurs parallèles utilisant comme coefficients de sin et cos stockés dans une table précalculée.

Afin de réduire la taille d'une telle table qui serait prohibitive avec les résolutions angulaires généralement utilisées, il est possible de n'effectuer de cette manière qu'une rotation initiale dégrossissant le calcul, et de terminer par un petit nombre d'itérations de l'algorithme CORDIC pour se rapprocher au plus près de l'argument.

Une autre technique hybride plus intéressante consiste à commencer le calcul par les itérations CORDIC, puis, quand l'argument angulaire résiduel est suffisamment petit, à approximer les tangentes au premier ordre: cette méthode ne nécessite pas de table de fonctions trigonométriques précalculées en plus de celles éventuellement utilisées par les itérations CORDIC.

3. Architectures de l'opérateur CORDIC

La gamme des différents choix possibles au niveau de l'architecture d'un opérateur CORDIC est représentée figure 3 sous la forme d'un diagramme synoptique en trois dimensions. Sur chacun des axes du repère est porté quantitativement le degré de parallélisme utilisable dans trois domaines distincts [13] largeur du chemin de données, pipelinage, parallélisme intrinsèque (sur les opérations et les itérations).

On peut agir sur la largeur du chemin de données des opérateurs élémentaires qui effectuent le calcul d'une itération (axe vertical) en utilisant des opérateurs arithmétiques en bit-série ou en bit-parallèle (avec éventuellement tous les intermédiaires envisageables).

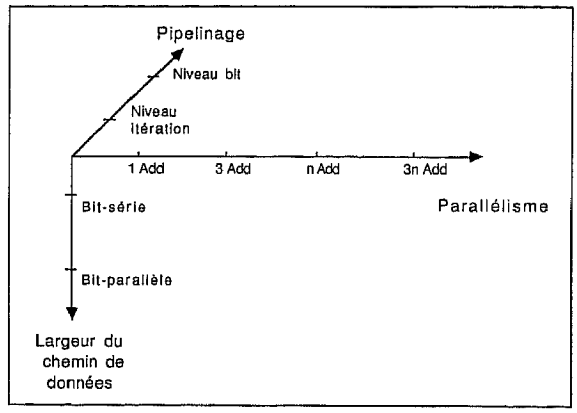

Fig. 3. — Diagramme synoptique des choix architecturaux pour l'algorithme CORDIC.

Pour obtenir le débit de calcul désiré, il est possible de pipeliner l'architecture (axe profond): du niveau bit en pipelinant internement le calcul des additions, jusqu'au niveau des itérations en insérant des registres de pipelinage entre les opérateurs réalisant le calcul d'une itération, si elles sont effectuées en parallèle.

Le troisième degré de liberté concerne les calculs des équations itératives qui peuvent être affectués en parallèle ou séquentiellement (axe horizontal) par uutilisation de trois unités arithmétiques ou d'une seule. On dénommera «à opérations parallèles» les architectures permettant le calcul simultané des trois équations de l'algorithme (ou deux si l'on supprime l'accumulation angulaire) et «opérations séries» celles ne permettant qu'un calcul séquentiel des équations. Les n itérations peuvent elles-mêmes être réalisées séquentiellement sur un à trois opérateurs, ou en parallèle sur n à trois n opérateurs. Il faut noter toutefois que ce parallélisme au niveau itérations n'est utilisable que par le biais d'un pipelinage «horizontal» minimum entre chaque itération, du fait de la dépendance entre les calculs des itérations successives.

3.1. Mise en oeuvre d'opérateurs bit-parallèles

Les additionneurs de l'architecture présentée sur la figure 1 sont des additionneurs parallèles.

Si le pipelinage «horizontal», c'est-à-dire le pipelinage entre étages de calcul, est possible et même nécessaire avec l'utilisation d'itérations parallèles pour avoir un taux d'accélération linéaire, le pipelinage «vertical» (au niveau bit) qui consisterait à pipeliner le report entre additionneurs complets est impossible du fait de la présence de décalages sur les opérandes des additionneurs. Si la variable Y est décalée de i bit, le i-ième bit constitue un des opérandes du premier additionneur complet et la retenue ne peut commencer sa progression que lorsque la retenue qui donne le résultat Y à l'étage précédent a atteint la position i. Il en est de même pour la variable X.

On obtiendra des architectures parallèles pipelinées dont la chaîne critique est constituée de «n» additionneurs complets juxtaposés où «n» est le nombre maximal de bit atteint par la représentation au cours de la chaîne de calcul.

Cette limitation due uniquement à la structure de l'algorithme CORDIC peut être contournée si on change la représentation des opérandes. On peut en effet résoudre le problème du pipelinage « vertical » des opérateurs bit-parallèles en utilisant une représentation redondante en chiffres signés.

Cette représentation permet de découper les opérandes en digits et d'effectuer les calculs sur les digits de façon indépendante et ainsi de supprimer la propagation entre l'opérateur élémentaire dans un même étage. Le problème de la décision de l'opération à effectuer à l'étage suivant se pose alors. Il faut extraire du digit de poids fort le signe de l'opérande afin de commander l'additionneur-soustracteur de l'étage suivant. Si ce digit est codé en représentation redondante il faut lui faire subir une transformation pour trouver le signe du nombre.

Un codage hybride des nombres a été proposé [31]. Le nombre est codé en représentation redondante sauf le digit de poids fort qui reste codé en complément à deux. Les opérateurs arithmétiques utilisent donc des additionneurs redondants et un additionneur en complément à deux pour le digit de poids fort. Le choix de la taille des digits fixe alors le degré de pipelinage « vertical ». On obtient ainsi des architectures dont le débit de calcul peut être relativement élevé.

3.1.1. *Architectures à itérations séries*

On met ici en œuvre des opérateurs arithmétiques parallèles qui réalisent simultanément le calcul des trois équations de l'algorithme, le calcul des n itérations étant enchaîné séquentiellement. La figure 4

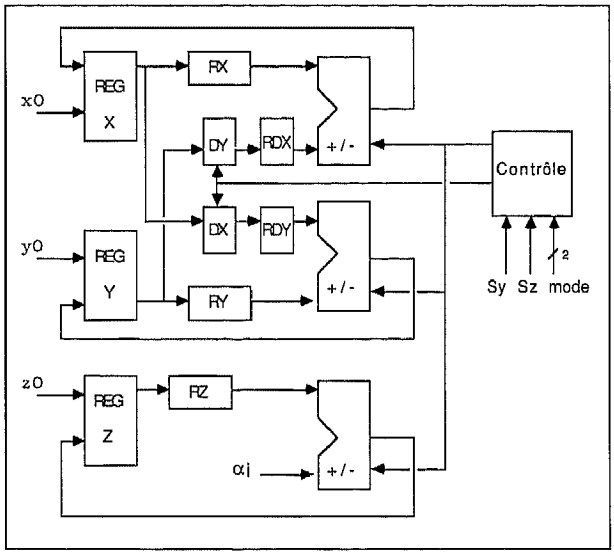

Fig. 4. — Architecture à « opérations parallèles » et itérations séries.

illustre ce type d'architecture. On utilise trois bus indépendants de largeur maximale pour réaliser en parallèle les calculs sur les trois variables.

Avant de commencer le calcul il faut réaliser la phase d'initialisation qui consiste à charger les registres X, Y et Z avec leur valeur respective x_0, y_0 et z_0. Tous les opérateurs dont la dénomination commence par un « R » sont des registres de pipelinage qui permettent d'isoler la partie calcul de la partie mémorisation des variables intermédiaires.

Les opérateurs DX et DY sont des décaleurs programmables dont les valeurs de décalage à appliquer aux variables X et Y à chaque itération dépendent de la base de décomposition. L'élément de base α_i est sélectionné en accord avec le rang de l'itération en cours de traitement et ajouté ou retranché à la variable courante z_i.

L'organe de contrôle élabore les signaux de commande des additionneur-soustracteurs (choix de l'opération), des décaleurs (valeur du décalage), en fonction du mode de fonctionnement de l'algorithme (mode) et des signes des variables courantes y_i et z_i (respectivement S_Y et S_Z).

Chaque itération est calculée en deux temps. Les données x_i, y_i, z_i sont d'abord transférées des registres X, Y, Z vers l'unité de calcul par l'intermédiaire des registres R_X, R_Y, RD_X, RD_Y et des décaleurs. Le deuxième temps comprend le calcul puis le stockage des variables résultats x_{i+1}, y_{i+1}, z_{i+1} dans les registres X, Y et Z. C'est cette dernière étape de calcul qui constitue la chaîne critique de l'architecture et qui fixe ses performances de vitesse. La fréquence de fonctionnement est limitée par le temps nécessaire à l'opérateur arithmétique pour réaliser une opération d'addition ou de soustraction.

La période de calcul correspond à n fois le temps caractéristique de la chaîne critique, où n est le nombre d'itérations.

Il faut noter qu'il est nécessaire de stocker les valeurs des arguments angulaires utilisés dans le calcul sur la variable auxiliaire z dans une mémoire ROM.

Architectures à opérations séries

Afin de limiter la complexité en matériel, on peut envisager de réduire le nombre d'opérateurs arithmétiques en n'utilisant qu'un seul additionneur-soustracteur bit-parallèle pour effectuer le calcul des trois équations. Le calcul des variables x, y et z est alors effectué séquentiellement par la même unité arithmétique et la structure est dite à « opérations séries » (cf. *fig.* 5). On a donc deux niveaux de séquencement. Le

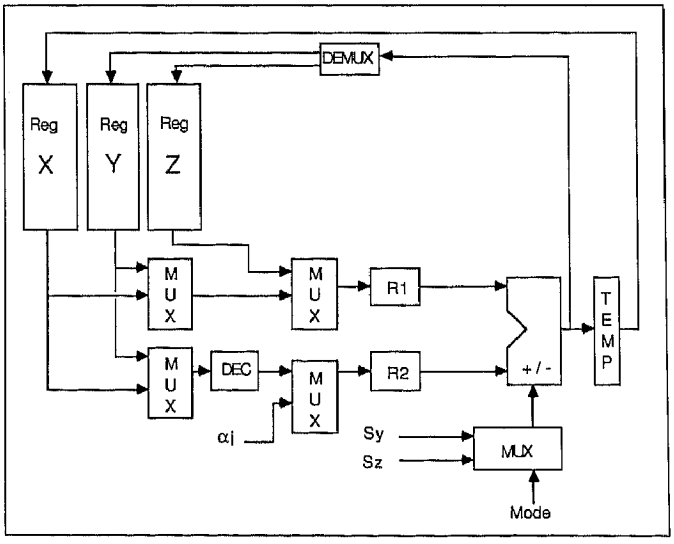

Fig. 5. — Architecture à « opérations séries » et itérations séries.

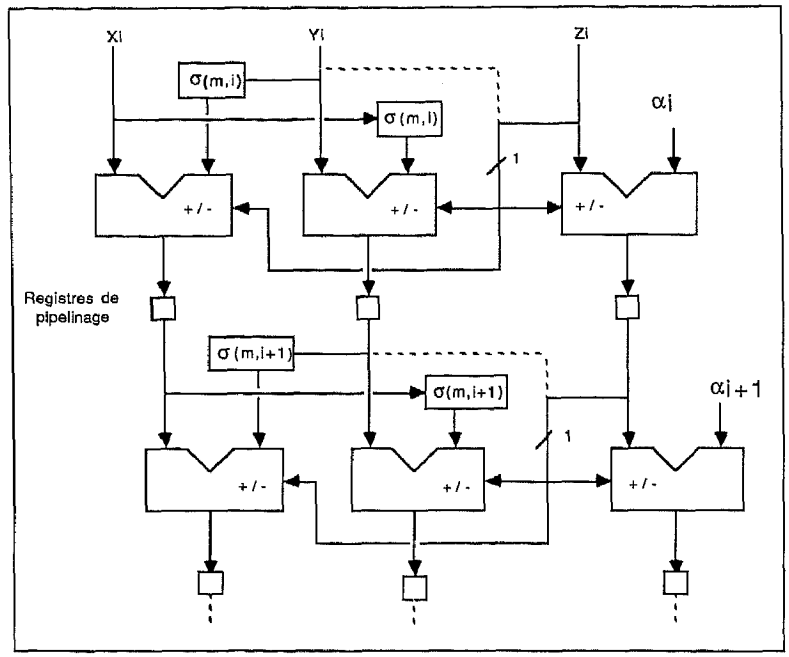

Fig. 6. — Architecture à « opérations parallèles » et itérations parallèles avec pipelinage « horizontal ».

niveau bas gère les calculs au sein d'une même itération, le niveau haut séquence le calcul des itérations conformément à la base de décomposition choisie. On accroît donc la complexité du séquenceur de la cellule CORDIC et on pert en partie les propriétés de modularité de l'opérateur.

Après initialisation des registres X, Y et Z le calcul itératif peut commencer. Le calcul d'une itération s'effectue en trois temps:

— chargement du registre R_1 avec la variable courante x_i issue du registre X et chargement du registre R_2 avec la variable courante y_i décalée issue du registre Y. Calcul par l'UA de la première équation et stockage du résultat x_{i+1} dans le registre Temp;

— chargement du registre R_1 avec la variable courante y_i issue du registre Y et chargement du registre R_2 avec la variable courante x_i décalée issue du registre X. Calcul par l'UA de la deuxième équation, stockage du résultat y_{i+1} dans le registre Y et transfert du registre Temp dans le registre X;

—chargement du registre R_1 avec la variable courante z_i issue du registre Z et chargement du registre R_2 avec l'élément de base α_i. Calcul par l'UA de la troisiéme équation et stockage du résultat z_{i+1} dans le registre Z.

Cette séquence est répétée à chaque itération et tous les calculs sont contrôlés par le mode choisi et les bit de signe des variables Y ou Z.

Une différence fondamentale avec la structure à « opérations parallèles » et itérations séries est que cette structure impose de stocker temporairement une des variables X ou Y lors des calculs d'une itération.

3.1.2. *Architectures à itérations parallèles*

Cette structure permet de traiter un flot continu de données avec un débit correspondant au calcul d'une seule itération. Le résultat du calcul est produit avec un retard dû à l'écoulement des données dans le pipeline.

L'importance de la base de décomposition apparaît nettement ici car c'est elle qui fixe le nombre d'itérations dont dépendent directement le délai de calcul et la complexité globale de l'architecture. On peut minimiser le nombre d'itérations en utilisant les méthodes décrites au paragraphe 2.5. La figure 2 illustre la structure d'un module de base (correspondant à une itération) d'une architecture qui utiliserait une base de décomposition construite à partir de différences de puissances de 2.

La chaîne critique de la structure à opérations parallèles et itérations parallèles est la même que celle de la structure à opérations parallèles et itérations séries. Les performances en vitesse sont limitées par le temps de calcul des opérateurs arithmétiques, qui est majoré par la propagation de la retenue dans un additionneur-soustracteur. Seule l'utilisation d'une représentation différente permet d'accroître le niveau de pipelinage et donc le débit de calcul.

Il a été proposé [5] un opérateur arithmétique qui permet d'unifier le temps de propagation dans les étapes de calcul. Le temps de la chaîne critique devient alors indépendant du rang de l'itération dans l'algorithme. Ceci permet d'optimiser l'écoulement des données dans les registres de pipeline car la chaîne critique est formée par les additionneurs complets dont les opérandes ne sont pas soumis à un décalage. Donc même si la largeur du chemin de données évolue au cours du calcul, la chaîne critique n'est pas modifiée.

Un avantage auxiliaire par rapport aux architectures à itérations série est l'absence de décaleur programmable. Les décalages sont ici réalisés par simples routages.

Il faut également noter l'adaptation excellente de cette architecture à un fonctionnement en mode

« évaluation rotation », puisqu'on peut dans ce cas supprimer à la fois l'accumulation de la variable angulaire z et la mémoire nécessaire pour stocker les arguments angulaires intermédiaires. Il est également possible de pipeliner conjointement un calcul d'évaluation et des calculs de rotation qui en utilisent le résultat (voir un exemple d'application au paragraphe 4.5).

3.2. Mise en œuvre d'opérateurs bit-séries

En bit série, les décalages ne sont plus réalisés par routage mais à l'aide de registres, ce qui diminue la complexité des décaleurs dans le cas programmable. Mais ici encore l'algorithme CORDIC impose une structure particulière des additionneur-soustracteurs lorsque les calculs des itérations sont effectués en parallèles. En effet si les calculs sont effectués séquentiellement, les opérandes sont mémorisés avant traitement et le signe est facilement accessible par l'opérateur arithmétique.

Pour les architectures où les calculs des itérations sont parallèles, le signe de la sortie de l'étage i détermine le type d'opération à effectuer à l'étage $i+1$ (addition ou soustraction). En complément à deux, le signe de l'opérande étant codé par le bit de poids fort, celui-ci doit être le premier présenté à l'opérateur arithmétique de l'étage suivant afin que le choix de l'opération à effectuer puisse être fait avant que le calcul ne commence. Il faut donc réaliser des additionneur-soustracteurs fonctionnant avec poids fort en tête, ce qui n'est possible, sans mémoriser un mot entier, qu'en utilisant une représentation redondante.

Une structure utilisant une telle arithmétique « digit-on-line » en chiffres signés a été proposée dernièrement [38], qui met en œuvre une variante légèrement modifiée de l'algorithme CORDIC standard.

Il faut remarquer que l'utilisation de la représentation redondante n'est réellement nécessaire que lors des calculs en mode évaluation. En mode rotation, on peut utiliser des opérateurs classiques sous réserve de connaître à l'avance la décomposition de l'argument sur la base.

3.2.1. *Architectures à itérations séries*

Ce type d'architecture utilise des opérateurs arithmétiques bit-série qui effectuent les calculs des itérations séquentiellement.

L'architecture décrite sur la figure 7 est à « opérations parallèles ». On retrouve les signaux de contrôle qui conditionnent l'opération à effectuer par l'unité arithmétique (mode, sens de rotation).

A l'initialisation les registres X, Y et Z sont chargés avec les valeurs initiales x_0, y_0 et z_0. Ces valeurs sont ensuite présentées en série à l'unité arithmétique. Les registres D_X et D_Y effectuent les décalages respectivement des variables X et Y en sélectionnant les bits à présenter en série à l'unité arithmétique. Ces opérateurs doivent également réaliser l'extension de signe des variables décalées.

Les résultats des opérations d'addition ou de soustraction sont stockés en série dans les registres R_X et R_Y. Le transfert de ces derniers dans les registres X et Y s'effectue en parallèle.

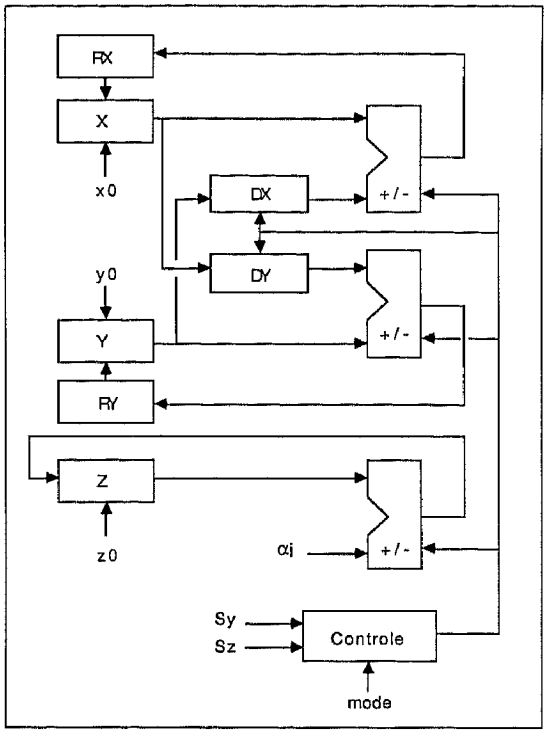

Fig. 7. — Architecture à « opérations séries » et itérations séries (les opérateurs effectuent les calculs en bit-série).

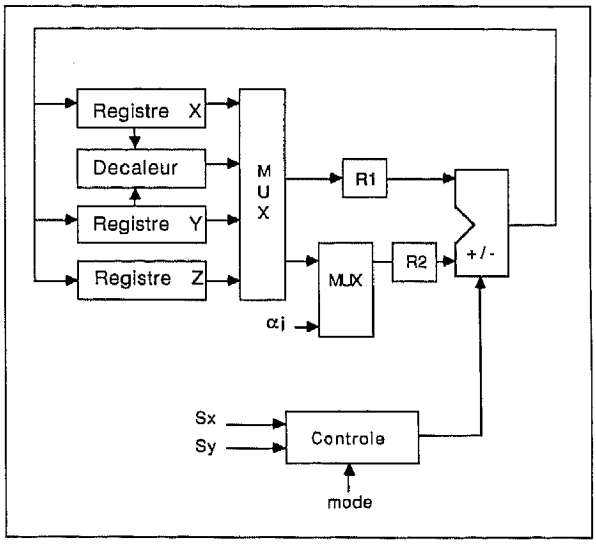

Fig. 8. — Architecture à « opérations parallèles » et itérations séries. Les unités arithmétiques travaillent en bit-série.

Le cycle de calcul de la variable Z est plus simple car aucun opérande n'est décalé. L'élément de base α_i est sélectionné en fonction du rang de l'itération en cours et est ajouté ou retranché à la variable z_i. Tous les transferts et calculs de la variable Z peuvent être réalisés en série.

Si cette structure utilise le moins de matériel et surtout présente la plus faible complexité au niveau routage des connexions, elle nécessite un séquenceur complexe à trois niveaux. Il faut gérer le séquencement d'une opération en bit-série, puis le calcul d'une itération et enfin le calcul des « n » itérations qui constituent l'algorithme complet.

Le débit maximal est limité par le temps nécessaire à l'unité arithmétique pour effectuer une opération d'addition-soustraction.

Il est encore possible de réduire la taille de l'architecture présentée ci-dessus en effectuant le calcul des équations d'une itération en séquence (architecture à « opérations séries », cf. *fig.* 8). Le séquencement du calcul d'une équation (x, y ou z) dans une itération est identique à celui de l'architecture à « opérations parallèles » et itérations séries présentée ci-dessus. Il faut ici gérer en plus le calcul séquentiel des trois équations au sein d'une itération de l'algorithme. Le multiplexeur placé en sortie des registres X, Y et Z permet de réaliser cette opération. Une telle architecture aboutirait en pratique à une complexité de contrôle beaucoup trop élevée, et surtout à des performances en débit de calcul inférieurs à ce qu'on recherche en général avec des architectures spécialisées.

3.2.2. *Architectures à itérations parallèles*

Dans ce type d'architecture les calculs des équations des « n » itérations s'effectuent en parallèle, mais en utilisant des opérateurs bit-série. Rappelons toutefois les problèmes arithmétiques posés par cette architecture. En mode évaluation les additionneur-soustracteurs doivent travailler en poids fort d'abord ce qui implique de changer la représentation des nombres. Le pipelinage est alors possible mais le prix à payer est une plus grande complexité des opérateurs arithmétiques qui se traduit par une surface plus importante. Les décalages doivent être réalisés à l'aide de registres à chaque étage ce qui augmente aussi la taille de la cellule. Il faut également noter la nécessité de générer des signaux d'initialisation ou de remise à zéro de ces registres.

L'architecture parallèle bit-série présentée ici est très intéressante de part son faible encombrement, pour une utilisation en mode « rotation » de l'opérateur CORDIC. En mode « évaluation » par contre, le gain relativement faible en surface justifie difficilement son utilisation par rapport à une architecture parallèle à opérateurs bit-parallèles qui permet d'atteindre des débits de calcul beaucoup plus élevés.

4. APPLICATIONS

Dans l'ensemble des algorithmes se prêtant à une utilisation potentielle de l'opérateur CORDIC, les deux premières applications mentionnées ci-dessous (§ 4.1 et 4.2) se distinguent fondamentalement des autres par le fait qu'il y est fait appel à des rotations, correspondant à des produits complexes, *où l'argument angulaire* θ *apparaît explicitement.* L'algorithme CORDIC est alors utilisé uniquement dans le mode rotation, et le contrôle par la variable dénotée par z ci-dessus, initialisée à θ et que l'on fait tendre vers 0 dans les itérations, y est nécessaire.

Dans toutes les autres applications, l'algorithme est utilisé pour le calcul de transformations planes orthogonales, où l'argument angulaire n'est qu'un intermédiaire de calcul pour l'opérateur CORDIC, utilisé ici dans le mode de fonctionnement composé évaluation-rotation.

4.1. Filtrage complexe de signaux analytiques, modulation a bande latérale unique

L'opérateur CORDIC permet de manière immédiate la manipulation directe de signaux complexes représentés par leur partie réelle et leur partie imaginaire, et leur produit par des coefficients complexes de module unité. L'algorithme pourrait aisément être adapté pour incorporer dans la normalisation le produit par un module fixe qui serait le cas échéant différent de 1.

L'intérêt de la manipulation de signaux complexes apparaît si on utilise le signal analytique (s. a.) correspondant à un signal réel donné (la composante imaginaire étant obtenue par passage au travers d'un filtre de quadrature effectuant une transformée de Hilbert discrète). On sait qu'un signal analytique peut être échantillonné à une fréquence moitié de celle qui serait nécessaire pour le signal réel correspondant. La représentation complexe aboutit donc à une utilisation du matériel équivalente en efficacité à ce qu'on a avec la représentation réelle.

Une utilisation très classique du signal analytique comme intermédiaire de calcul est la modulation en bande latérale unique, où l'on module le s.a. par une porteuse réelle pour aboutir à un signal modulé divisant par 2 l'encombrement en fréquence. On peut faire le produit du s.a. par la porteuse complexe $e^{2\pi jnf_0}$ au moyen d'un opérateur Cordic utilisé en mode rotation circulaire, le calcul de l'argument angulaire pouvant être effectué en parallèle soit par un multiplieur dédié, soit par un autre opérateur CORDIC fonctionnant en mode linéaire. Il suffit alors de conserver uniquement la partie réelle en sortie de l'opérateur pour retrouver le signal B.L.U.

4.2. Transformée de Fourier discrète

Le calcul direct de la transformée de Fourier en effectuant le produit matrice-vecteur correspondant peut être implémenté au moyen de l'opérateur CORDIC.

Il est théoriquement possible d'envisager une structure entièrement parallèle utilisant un tableau de N^2 processeurs CORDIC, chaque processeur étant dédié à un coefficient $e^{2\pi jnk/N}$, ce qui permet de le simplifier éventuellement. En pratique ceci ne serait réalisable avec une complexité en matéreil acceptable qu'au moyen d'opérateurs CORDIC bit-série (§ 3.2.2).

Il est classiquement possible de réaliser ce produit matrice N*N par le vecteur N*1 au moyen de N processeurs parallèles seulement, en effectuant une « projection » du calcul dans l'espace des processeurs. Si chaque processeur correspond à une valeur de sortie, l'accumulation est alors locale et les valeurs d'entrée diffusées globalement à tous les processeurs. De manière duale, si chaque processeur est associé à une valeur d'entrée, l'accumulation se fait en cascade entre les différents processeurs (il sera alors nécessaire de pipeliner le calcul par un décalage dans le temps des valeurs d'entrée, si l'on veut aboutir au même

débit). Les arguments angulaires doivent être mis à jour à chaque utilisation des processeurs, ou ils peuvent être précalculés.

L'utilisation d'un algorithme rapide se prête mal à une structure entièrement parallèle, les communications entre les différents processeurs étant alors dominantes. Une structure cascade pipeline est en général préférable. On peut montrer [19] que pour l'utilisation de l'opérateur CORDIC, une structure à base de transformées élémentaires d'ordre 4, qui se calculent uniquement par des additions, séparées par des rotations CORDIC, est optimale. Les rotations CORDIC peuvent éventuellement ne pas être normalisées, ce qui correspond au report en sortie d'un facteur de gain réel qu'il est possible de compenser en une seule fois.

Si on admet un facteur de gain complexe, une telle structure peut encore être simplifiée [18]. Il est alors possible d'utiliser des transformées élémentaires d'ordre 16, dont le nombre de notations total est minimisé par l'offset angulaire commun.

4.3. Structures en treillis

4.3.1. *Filtres à coefficients fixes*

La réalisation d'une fonction de transfert rationnelle par une structure en treillis correspond à une décomposition récursive en ordre sur la base orthogonale des polynômes de Szegö, les coefficients k_i fournissant directement une caractérisation de la stabilité. Le filtre normalisé de Gray-Markel est obtenu en imposant une condition supplémentaire d'orthonormalité, qui garantit de bonnes propriétés par rapport à tous les effets de longueur de mot finie [32].

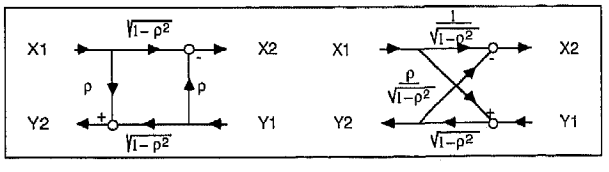

Fig. 9. – Cellule de filtre autorégressif en treillis normalisé de Gray-Markel.

La figure 9 illustre deux variantes (correspondant respectivement à une structure de type « ligne de transmission » et en treillis proprement dite) d'une cellule élémentaire de filtre normalisé purement récursif. Un tel filtre peut être obtenu par application de l'algorithme de Levinson normalisé au problème de la modélisation stochastique autorégressive d'un signal stationnaire.

Les équations correspondantes sont données ci-dessous.

$$X_2 = \sqrt{1-\rho^2}\, X_1 - \rho\, Y_1$$
$$Y_2 = \rho\, X_1 + \sqrt{1-\rho^2}\, Y_1$$

soit, avec $\rho = \sin\theta$

$$\begin{pmatrix} X_2 \\ Y_2 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} X_1 \\ Y_1 \end{pmatrix}$$

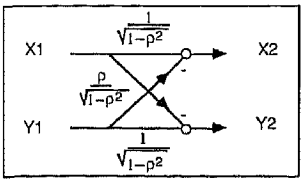

Fig. 10. – Cellule de fibre RIF en treillis normalisé.

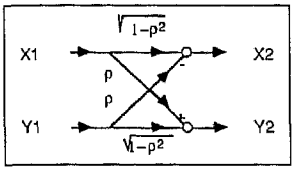

Fig. 11. – Cellule de filtre RIF en treillis passif.

On voit donc qu'une telle cellule élémentaire peut être directement réalisée au moyen d'un opérateur CORDIC.

La figure 10 donne la structure d'une cellule élémentaire du filtre inverse à moyenne adaptée, qui serait un filtre d'analyse (blanchissant) si le précédent était un filtre de synthèse. Les équations correspondantes sont dans ce cas :

$$X_2 = \frac{1}{\sqrt{1-\rho^2}} X_1 - \frac{\rho}{\sqrt{1-\rho^2}} Y_1$$
$$Y_2 = -\frac{\rho}{\sqrt{1-\rho^2}} X_1 + \frac{1}{\sqrt{1-\rho^2}} Y_1$$

soit, avec $\rho = \text{th}\,\theta$

$$\begin{pmatrix} X_2 \\ Y_2 \end{pmatrix} = \begin{pmatrix} \text{ch}\,\theta & \text{sh}\,\theta \\ \text{sh}\,\theta & \text{ch}\,\theta \end{pmatrix} \begin{pmatrix} X_1 \\ Y_1 \end{pmatrix}$$

Pour une implémentation de cette cellule au moyen de l'opérateur CORDIC, l'on utiliserait donc ici le mode de fonctionnement hyperbolique.

Ces cellules de base, qui correspondent à la réalisation numérique de quadripôles sans pertes, se cascadent entre elles (avec insertion de délais sur la branche inférieure) pour réaliser des filtres en treillis respectivement AR et MA.

4.3.2. *Algorithme de Schur*

Il est tout à fait remarquable qu'on puisse ramener à une structure de calcul analogue le calcul des coefficients k_i à partir de la matrice d'autocorrélation du signal (supposée connue *a priori* dans le cas stationnaire), si l'on utilise une version normalisée de l'algorithme de Schur (dénomination qui semble prévaloir désormais pour cette très importante variante de l'algorithme de Levinson, également connue comme algorithme de Le Roux-Gueguen, ou de Bareiss-Rissanen). Le principe est de normaliser les itérations de l'algorithme sous la forme suivante :

$$\frac{1}{\sqrt{1-k_i^2}} \begin{pmatrix} 1 & -k_i \\ -k_i & 1 \end{pmatrix} \begin{pmatrix} R_i^i & \dots & R_{n-1}^i \\ R_{i+1}^{*\,i} & \dots & R_n^{*\,i} \end{pmatrix}$$
$$= \begin{pmatrix} R_i^{i+1} & \dots & \dots \\ 0 & R_{i+1}^{*\,i+1} & \dots \end{pmatrix}$$

avec

$$k_i = \frac{R^{*i}_{i+1}}{R^i_i}$$

Les R^0_i étant initialisés à la valeur des coefficients d'autocorrélation du signal.

Du point de vue de l'algèbre linéaire, cet algorithme peut être vue comme une procédure pour la décomposition de Cholesky des matrices de Toeplitz, ou plus généralement pour la résolution de systèmes de Toeplitz [22, 34].

Le calcul pourrait s'effectuer sur une structure en treillis composée d'un tableau linéaire de cellules de base analogues à celle présentée à la figure 10, les k_i étant calculés (par leur représentation CORDIC hyperbolique) par une utilisation d'un des processeurs en mode évaluation.

Les k_i peuvent être directement utilisés dans un filtre d'analyse de même structure.

4.3.3. *Filtrage adaptatif*

Il est possible de généraliser l'utilisation de telles structures en treillis normalisé pour l'analyse de signaux non stationnaires, dans les cas classiques d'application des algorithmes des moindres carrés récursifs rapides (fenêtres antérieure et postérieure, et plus généralement matrice de covariance à rang de déplacement fini) [28, 35]. La normalisation apporte dans ce cadre une simplification importante sur les récurrences, en supprimant la nécessité de mettre à jour la variance des erreurs de prédiction directe et rétrograde, qui est unitaire. La mise à jour récursive des coefficients de treillis se fait par des calculs annexes qui peuvent tous être réalisés au moyen d'opérateurs CORDIC, la structure du filtre d'analyse lui-même étant identique au cass non adaptatif.

4.4. Filtres orthogonaux, filtres d'onde

Le filtre de Gray-Markel [32] permet de réaliser des fonctions de transfert comportant des zéros de transmission au prix de l'utilisation d'une sortie transversale qui lui fait perdre en partie ses bonnes propriétés numériques. Sa partie purement récursive représente un exemple simple de filtre orthogonal.

La propriété fondamentale caractéristique des filtres orthogonaux est le fait que toutes les variables internes sont décorrélées entre elles, et de variance unitaire. On les construit à partir de filtres multicanaux passe-tout, dont la matrice globale de représentation d'état est orthogonale, et la fonction de transfert (matricielle) est unitaire. La fonction de transfert désirée est obtenue par *insertion* dans un tel filtre passe-tout [27], par exemple par adjonction de terminaisons rebouclées sur les extrémités libres du filtre multicanal, ce qui le transforme en filtre scalaire.

Dans le cas du filtre de Gray-Markel purement récursif, le filtre passe-tout dont on part est obtenu par mise en cascade de quadripoles purement réactifs dont la matrice de transfert unitaire n'est autre qu'une matrice de rotation circulaire CORDIC.

Une structure possible de filtre orthogonal pour la réalisation d'une fonction de transfert quelconque est le double-treillis [25] dérivé du filtre de Gray-Markel. Un filtre d'ordre N est ainsi réalisé au moyen de 2N processeurs CORDIC fonctionnant en mode rotation circulaire.

Plus généralement, on peut montrer que l'opérateur CORDIC intervient dans la synthèse d'un filtre orthogonal quelconque [37], § 10.4.

Les filtres ainsi obtenus présentent essentiellement un intérêt pour leurs propriétés de faible sensibilité par rapport à la quantification des coefficients, de faible transmission du bruit d'arrondi interne, et l'absence d'oscillations de dépassement.

Il a été montré [26] que les filtres d'onde, qui partent d'un principe de synthèse *a priori* très différent (transposition de filtres analogiques LC sur terminaisons résistives), peuvent être obtenus par une synthèse directe en z, ce qui découvre leur parenté profonde avec les filtres orthogonaux, en particulier le fait qu'ils peuvent également être réalisés de manière modulaire en utilisant l'opérateur CORDIC.

4.5. Résolution de système linéaire

Il est connu que la méthode classique de Gauss pour la réduction d'une matrice à la forme triangulaire est inadaptée à la réalisation sous forme d'architectures hautement parallèles spécialisées du fait de la nécessité du pivotage. Cette procédure élémentaire est rappelée ci-dessous, pour une matrice (a_{ij}) d'ordre n :

$$\begin{pmatrix} a^{(r)}_{rr} & a^{(r)}_{rr+1} & \dots & a^{(r)}_{rn} \\ 0 & a^{(r+1)}_{ir+1} & \dots & a^{(r+1)}_{in} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \alpha^{(r)}_i & 1 \end{pmatrix} \begin{pmatrix} a^{(r)}_{rr} & a^{(r)}_{rr+1} & \dots & a^{(r)}_{rn} \\ a^{(r)}_{ir} & a^{(r)}_{ir+1} & \dots & a^{(r)}_{in} \end{pmatrix}$$

avec

$$\alpha^{(r)}_i = -\frac{a^{(r)}_{ir}}{a^{(r)}_{rr}}$$

Cette opération étant pour r variant de 1 à n, et i variant de $r+1$ à n pour chaque r. Une matrice de permutation est appliquée préalablement à chaque étape pour choisir le pivot a_{rr} de plus grande valeur. On peut noter que chaque itération ci-dessus correspondrait à l'utilisation de l'opérateur CORDIC dans le mode composé « évaluation-rotation » *linéaire*, chaque argument $\alpha^{(r)}_i$ évalué étant utilisé pour le calcul d'une ligne a_{ij}, j variant de $r+1$ à n.

L'utilisation de transformations orthogonales à chaque étape de la réduction supprime cette contrainte du pivotage. Les itérations deviennent :

$$\begin{pmatrix} a^{(r+1)}_{rr+1} & \dots & a^{(r+1)}_{m} \\ a^{(r+1)}_{ir+1} & \dots & a^{(r+1)}_{in} \end{pmatrix} = \begin{pmatrix} \cos(\theta^{(r)}_i) & -\sin(\theta^{(r)}_i) \\ \sin(\theta^{(r)}_i) & \cos(\theta^{(r)}_i) \end{pmatrix} \begin{pmatrix} a^{(r)}_{rr} & a^{(r)}_{rr+1} & \dots & a^{(r)}_{rn} \\ a^{(r)}_{ir} & a^{(r)}_{ir+1} & \dots & a^{(r)}_{in} \end{pmatrix}$$

avec

$$a^{(r+1)}_{rr} = \sqrt{(a^{(r)}_{rr})^2 + (a^{(r)}_{ir})^2}$$

$$\theta^{(r)}_i = -\operatorname{arctg} \frac{a^{(r)}_{ir}}{a^{(r)}_{rr}}$$

répétées de la même manière pour $r=1 \ldots n$, $i=r+1 \ldots n$.

On utilise ici l'algorithme CORDIC dans le mode de fonctionnement composé évaluation-rotation de la même manière que précédemment, mais avec la métrique circulaire.

Cet exemple illustre parfaitement la pertinence du mode de fonctionnement composé évaluation-rotation, avec les métriques circulaire et linéaire.

L'algotithme de Givens peut être implanté sur un tableau de processeurs CORDIC pipelinés entre eux, correspondant à une instantiation spatiale entièrement parallèle. La première colonne du tableau effectue les calculs en mode évaluation des arguments angulaires qui sont traonsmis sur chaque ligne aux processeurs réalisant le calcul en mode rotation.

On n'a traité ainsi que le problème de la triangulation, la résolution effective du système requérant l'étape supplémentaire de rétro-substitution, qui, à l'instar du pivotage, pose problème dans une implantation parallèle, car les calculs correspondants n'ont pas la propriété essentielle de localité spatiale.

C'est pourquoi il a été imaginé [17], d'utiliser un algorithme (attribué à Fadeeva) où la procédure de triangularisation précédente est appliquée à une matrice auxiliaire obtenue par adjonction de lignes et colonnes à la matrice du système, de telle sorte que la solution du système apparaît directement dans le résultat de la factorisation de cette matrice auxiliaire, supprimant ainsi l'étape de rétrosubstitution.

5. Conclusion

Cette revue des différentes applications et des implémentations architecturales possibles pour l'algorithme CORDIC était destinée à en illustrer toute la puissance et la généralité. Il apparaît que l'intérêt essentiel de l'opérateur CORDIC est de rendre possible l'implantation d'algorithmes normalisés par racines carrées dont les propriétés numériques sont supérieures à celles des algorithmes classiques, mais qui entraineraient une trop grande complexité additionnelle s'ils étaient réalisés sur du matériel classique à base de multiplieurs et additionneurs. C'est donc aux concepteurs de systèmes utilisant ces algorithmes qu'il revient de prendre conscience des possibilités nouvelles qui leur sont ainsi offertes, pour exploiter au mieux tout le potentiel de l'intégration à très grande échelle.

Manuscrit reçu le 15 *janvier* 1988.

BIBLIOGRAPHIE

[1] J. E. Volder, The CORDIC Trigonometric Computing Technique, *IRE Transactions on Electronic Computers*, EC-8, Sept. 1959, p. 330-334.

[2] J. S. Walther, A Unified Algorithm for Elementary Functions, *Proceedings Joint Spring Computer Conference*, 1971, p. 379-385.

[3] M. Cosnard *et al.*, The FELIN Arithmetic Coprocessor Chip, *Proceedings IEEE ARITH'8 Symposium*, Come, May 1987, p. 107-112.

[4] R. Udo et E. Deprettere, *On the Design of Pipelined Architectures for the CORDIC Algorithm*, Technical Report, Delft University of Technology.

[5] E. Deprettere, P. Dewilde et R. Udo, Pipelined CORDIC Architectures for Fast VLSI Filtering and Array Processing, *Proceedings ICCD'84*, p. 41.A.6.1, 41.A.6.4.

[6] Tze-Yun Sung et Tai-Ming Parng, High Performance VLSI CORDIC Algorithms, Architectures and Chip Design, *Journal of the Chinese Institute of Engineers.*

[7] K. Hwang, *Computer Arithmetic, Principles, Architecture, and Design*, Wiley, 1979.

[8] G. L. Haviland et A. L. Tuszynski, A CORDIC Arithmetic Processor Chip, *IEEE Transactions on Computer*, C-29, n° 2, Feb. 1980.

[9] H. Ahmed, J. M. Delosme et M. Morf, Highly Concurrent Computing Structures for Matrix Arithmetic and Signal Processing, *IEEE Computer Magazine*, Jan, 1982.

[10] J. M. Muller, Dicrete Basis and Computation of Elementary Functions, *IEEE Transactions on Computers*, C-34, n° 9, Sept. 1985.

[11] P. Dewilde, E. Deprettere et R. Nouta, *Parallel and Pipelined VLSI implementation of signal processing algorithms*, in *VLSI and Modern Signal Processing*, S. Y. Kung, H. J. Whitehouse et T. Kailath éd., Prentice Hall, 1985, p. 257-276.

[12] T. Kailath, *Signal Processing in the VLSI Era* in *VLSI and Modern Signal Processing*, S. Y. Kung, H. J. Whitehouse et T. Kailath éd., Prentice Hall, 1985.

[13] G. Privat, Architectures Spécialisées de Circuits VLSI pour le Traitement Numérique du Signal, *Thèse de Doctorat*, ENST, Paris, Déc. 1986.

[14] H. Ahmed, *Alternative Arithmetic Unit Architectures for VLSI Digital Signal Processors*, in *VLSI and Modern Signal Processing*, S. Y. Kung, H. J. Whitehouse et T. Kailath éd., Prentice Hall, 1985.

[15] J. M. Delosme, VLSI Implementation of Rotations in Pseudo-Euclidean Spaces, *Proceedings of ICASSP'83*, p. 927-930.

[16] A. Naseem et P. D. Fisher, The Modified Cordic Algorithm, *Proceedings of the 7th Symposium on Computer Arithmetic*, June 1985, p. 144-152.

[17] K. Jainandunsing, H. Nelis et E. Deprettere, Systematic Design of Fixed Size VLSI Systolic Array applied to the Orthogonal Fadeeva Equations Solver, *Proceedings ECCTD'87*, Paris, September 1987.

[18] A. M. Despain, Very Fast Fourier Transform Algorithms Harware for Implementation, *IEEE Transactions on Computers*, 28, n° 5, May 1979, p. 333-341.

[19] A. M. Despain, Fourier Transform Computers using CORDIC Iterations, *IEEE Transactions on Computers*, C-23, n° 10, October 1974, p. 993-1001.

[20] L. H. Sibul et A. L. Fogelsanger, Application of Coordinate Rotation Algorithm to Singular Value Decomposition, *Proceedings of ISCAS'84*, p. 821-824.

[21] J. R. Cavallaro et F. T. Luk, CORDIC Arithmetic for an SVD Processor, *Proceedings of the 8th Symposium on Computer Arithmetic*, Come, May 1987.

[22] I-Chang Jou, Yu-Hen Hu et W. S. Feeg, A Novel Implementation Of Pipelined Toeplitz System Solver, *Proceedings of the IEEE*, 74, n° 10, October 1986.

[23] Tze-Yun Sung et Yu-Hen Hu, VLSI Implementation of Real-Time Kalman Filter, *Proceedings of ICASSP'86*, Tokyo, p. 2223-2226.

[24] M. MORF, C. H. MURAVCHIK, P. H. ÅNG et J. M. DELOSME, Fast Cholesky algorithms and Adaptative Feedback Filbers, *ICASSP*'82, Paris, p. 1727-1731.

[25] S. K. RAO et T. KAILATH, Orthogonal Digital FIlters for VLSI Implementation, *IEEE Transactions on Circuits and Systems*, CAS-31, n° 11, November 1984.

[26] P. P. VAIDYANATHAN, A Unified Approach to Orthogonal Digital Filters and Wave Digital Filters, Based on LBR Two-Pair Extraction, *IEEE Transactions on Circuits and Systems*, CAS-32, n° 7, July 1985.

[27] P. DEWILDE, *Advanced Digital Filters*, in *Modern Signal Processing*, T. KAILATH éd., Hemisphere Publishing Corporation, 1985.

[28] H. M. AHMED, D. T. LEE, M. MORF et P. H. ANG, A VSLI Speech Analysis Chip Set based on Square Root Normalized Ladder Forms, *Proceedings of ICASSP*'81, p. 648-653.

[29] H. M. AHMED, Signal Processing Algorithms and Architectures, *Ph. D. Dissertation*, Department of Electrical Engineering, Stanford University, 1982.

[30] M. RENAUDIN, *Architectures d'un opérateur CORDIC*, Rapport de stage, CNET-Grenoble, CEPHAG, juin 1987.

[31] J. LIENARD et C. OLTAY, Distributed Digit-serial Architecture for Signal Processing, *Proceedings of ECCTD*'87, Paris, septembre 1987, p. 251-256.

[32] A. H. GRAY et J. D. MARKEL, A normalized Digital Filter Structure, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-23, n° 3, June 1975, p. 268-277.

[33] E. DEPRETTERE, Synthesis and fixed-point implementation of pipelined true orthogonal filters, *Proceedings of ICASSP*'83, Boston, p. 217-220.

[34] S. Y. KUNG et Y. H. HU, A Highly Concurrent Algorithm and Pipelined Architecture for Solving Toeplitz Systems, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-31, n° 1, Feb. 1983, p. 66-75.

[35] B. FRIEDLANDER, Lattice Filters for Adaptive Processing, *Proceedings of the IEEE*, 70, n° 8, Aug. 1982, p. 829-867.

[36] E. DEPRETTERE et K. JAINANDUNSING, Orthogonal and J-Orthogonal Matrix Inversion Techniques, *Proceedings of ISCAS*'87, May 87, p. 143-146.

[37] R. A. ROBERTS et C. T. MULLIS, *Digital Signal Processing*, Addison-Wesley, 1987.

[38] M. D. ERCEGOVAC et T. LANG, Implementation of Fast Angle Calculation and Rotation Using On-Line CORDIC, *Proceedings of ISCAS*'88, p. 2703-2706.