REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

ECOLE NATIONALE SUPERIEURE POLYTECHNIQUE

Département d'Electronique

Projet Master

En vue de l'obtention du diplôme
Master 2 en Electronique

Thème

L'opérateur CORDIC : généralisation, architectures et méthodes d'amélioration

Proposé et dirigé par :

Mr Taghi Mohamed Oussaid

Mr Abdellouel Lahcen

Réalisé par :

Ben mesbah Wahiba

Promotion juin 2012

Sommaire

1 Introduction

L'algorithme CORDIC, (acronyme de COordinate Rotation DIgital Computing) a été créé à l'origine par J.E.Volder pour effectuer des calculs tels que les rotations de vecteurs ou les changements de coordonnées cartésiennes-polaires et polaires-cartésiennes dans le plan euclidien [7].
S'appuyant sur une technique décalage-addition et de rotation vectorielle, l'algorithme calcul par approximation la plupart des fonctions basées sur la trigonométrie. Beaucoup de calculatrices commerciales utilisent cet algorithme pour résoudre des fonctions telles que sinus, cosinus, arctangente et racine carré.
Comme la plupart des algorithmes performants l'efficacité de CORDIC repose sur sa simplicité et sa flexibilité. Tel qu'il le sera démontré un peu plus loin, l'algorithme se compose d'opérations mathématiques élémentaires qui, en plus d'offrir une grande efficacité de calcul, simplifie énormément l'implantation sur des plates formes logicielles ou matérielles.

2 Description de l'algorithme CORDIC

La résolution de fonction telle que sinus ou cosinus par l'algorithme de CORDIC s'appuie sur une méthode de rotation de vecteur dans le plan cartésien.
Supposons la rotation du vecteur V(x,y) d'un angle j tel qu'illustré à la figure1

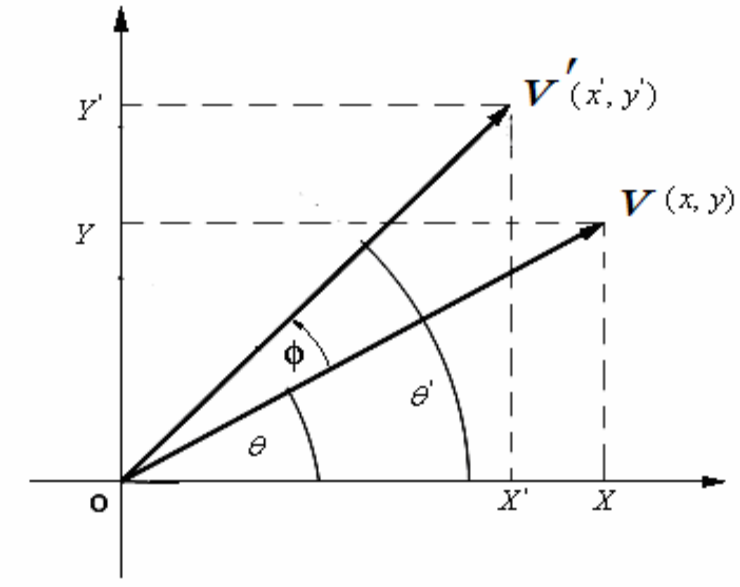

Figure 1: Rotation d'un vecteur V par un angle Φ

Les coordonnées du vecteur V' sont exprimées selon les équations

x' = x cos(j) - y sin(j)
y' = y cos(j) + x sin(j)

Que l'on peut réorganiser afin d'obtenir les relations suivantes :

$x' = \cos(j) [x - y \tan(j)]$
$y' = \cos(j) [y + x \tan(j)]$

Si on restreint l'angle de rotation à atan ($\pm 2^i$) où i = 0,1,2,3,…, on obtient alors φ par une série de rotations élémentaires successives de l'ordre de

$$z_{i+1} = z_i - d_i \times \tan\text{-}1(2^{-i})$$

Où di = ±1 .

L'indice di indique le sens de rotation de l'angle pour chaque itération. Cet indice est déterminé à chaque itération selon le résultat d'une comparaison.
Chaque vecteur itératif $V_{i+1}(x_{i+1}, y_{i+1})$ sont représenté par

$$x_{i+1} = Ki [xi - yi \times di \times 2^{-i}]$$
$$y_{i+1} = Ki [yi + xi \times di \times 2^{-i}]$$

Où

$$Ki = \cos (\tan^{-1} 2^{-i}) = (1 + 2^{-2i})^{-1/2}$$

Pour une suite d'itération, les facteurs Ki peuvent être mis en évidence et donner un produit de multiplication. Pour un nombre croissant d'itération le produit tend vers la valeur de 0.6073. Ainsi le gain de l'algorithme de rotation est d'approximativement 1.647. Le gain précis est fonction du nombre n d'itérations que l'on peut exprimer par

$$A_n = \prod_n \left(1 + 2^{-2i}\right)^{1/2}$$

Puisque pour un nombre relativement élevé d'itérations, le produit tend vers un résultat constant, il nous est possible d'appliquer ce dernier plus loin dans l'algorithme. Ainsi on obtient un ensemble d'équations simplifiées et effectives au calcul des opérations mathématiques recherchées :

$$x_{i+1} = x_i - di \times dyi$$
$$y_{i+1} = y_i + di \times dxi$$
$$z_{i+1} = z_i - di \times dai$$

Où

$$dyi = yi \times 2^{-i}$$
$$dxi = xi \times 2^{-i}$$
$$dai = \tan^{-1}(2^{-i})$$
$$di = \pm 1$$

Ces équations servent donc au calcul d'opérations. Le principe générale de l'algorithme CORDIC consiste à faire pivoter dans le sens approprié le vecteur de rotation par un angle de plus en plus petit jusqu'à ce que l'angle a ou les valeurs x et y, déterminés par l'usager, soient approximativement égales à 0.

Table 1: Pour un hardware CORDIC de 8-bit

i	tanΦ_i	$\Phi_i = \arctan(2^{-i})$	$\Phi_i$en radian
0	1	45°	0.7854
1	0.5	26.565°	0.4636
2	0.25	14.036°	0.2450
3	0.125	7.125°	0.1244
4	0.0625	3.576o°	0.0624
5	0.03125	1.7876°	0.0312
6	0.015625	0.8938°	0.0156
7	0.0078125	0.4469°	0.0078

3 Les itérations fondamentales de l'algorithme CORDIC :

Pour simplifier chaque rotation, choisissant α_i (l'angle de rotation de la iième itération) tel que $\alpha_i = d_i\ 2^{-i}$. d_i prend la valeur +1 ou -1 selon la rotation donc

$$x_{i+1} = x_i - y_i\ d_i\ 2^{-i}$$

$$y_{i+1} = y_i + x_i\ d_i\ 2^{-i}$$

$$z_{i+1} = z_i - d_i\ tan^{-1}\ 2^{-i}$$

Le calcul de x_{i+1} ou de y_{i+1} exige un registre à décalage à droite de i-bit et un additionneur/soustracteur. Si la fonction $tan^{-1}\ 2^{-i}$ est pré calculé et mémorisé dans un tableau pour les différentes valeurs de i, un simple additionneur/soustracteur suffit pour calculer z_{i+1} . Chaque itération CORDIC ainsi comporte deux registres à décalage, un tableau de consultation (LUT) et trois additionneurs.

Si la rotation est faite par le même ensemble des angles (avec un signe + ou -), alors le facteur K est une constante, et peut être pré calculer. Par exemple pour tourner par un angle de 30 degré, on suive la séquence des angles qui s'ajoute jusqu'à 30 degré.

$$30 \approx 45.0 - 26.6 + 14.0 - 7.1 + 3.6 + 1.8 - 0.9 + 0.4 - 0.2 + 0.1$$

$$= 30.1$$

En réalité, ce qui se produit réellement dans l'algorithme CORDIC est que z est initialisé à 30 degrés et puis, dans chaque opération, le signe du prochain angle de rotation est sélectionné pour essayer de changer le signe de z ; qui est, d_i=sign (z_i), où la fonction de signe est défini pour être -1 ou 1 selon si l'argument est négatif ou non négatif. Sa nous rappelle la division sans remise. Le Tableau 2 montre le procédé de sélectionner les signes des angles tournés pour une rotation désirée de +30 degrés.

Tableau 2 : les valeurs approximatives de la fonction α_i=arctan (2^{-i}), en degré,

Pour $0 \leq i \leq 9$.

I	α_i
0	45
1	26.6
2	14
3	7.1
4	3.6
5	1.8
6	0.9
7	0.4
8	0.2
9	0.1

Il existe deux modes pour l'algorithme CORDIC , mode rotation et mode vecteur :

3.1 Mode de rotation

Dons ce mode on suivre les équations de l'algorithme CORDIC citées avant et on initialise les coordonnés (x, y) par : x = 1/K = 0.607252935 et y = 0. Puis, comme z_m tend vers 0 dans les itérations CORDIC en mode rotation, x_m et y_m convergent vers cos a et sin a, respectivement.

Après m itérations

$$x_m = k\,(x_0 \cos a_0 - y_0 \sin a_0)$$

$$y_m = k\,(y_0 \cos a_0 + x_0 \sin a_0)$$

$$z_m = 0$$

3.2 Mode Vecteur

le y est rendu plus proche de zéro en choisissant d_i = - sign($x_i y_i$). Après m itérations dans le mode Vecteur donne comme résultats :

$$x_m = k\,(x^2 + y^2)^{1/2}$$

$$y_m = 0$$

$$z_m = a + \tan^{-1}(y/x)$$

Pour calculer $\tan^{-1}$y dans le mode Vecteur on commence par x = 1 et z = 0.

4 CORDIC généralisé :

En 1971, John Stephen Walther de Hewlett Packard, a présenté une généralisation de l'algorithme .Cette méthode permet de calculer notamment les fonctions hyperboliques mais également d'autres fonctions comme l'exponentielle, la division ou la multiplication. La généralisation s'exprime sous forme matricielle comme suit [7]:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos(\theta\sqrt{m}) & -\sqrt{m}\sin(\theta\sqrt{m}) \\ \frac{1}{m}\sin(\theta\sqrt{m}) & \cos(\theta\sqrt{m}) \end{pmatrix} * \begin{pmatrix} x \\ y \end{pmatrix}$$

Le paramètre m peut prendre les trois valeurs suivantes :

- m = 1 : cas circulaire, il permet de calculer différentes fonctions trigonométriques (sinus, cosinus, arctangente) ainsi que des racines carrées,
- m = -1 : cas hyperbolique, il permet de calculer les fonctions sinus, cosinus et argtangente hyperboliques ainsi que l'exponentielle,
- m = 0 : cas linéaire il permet également le calcul direct de multiplications et de divisions.

En faisant sortir le $\cos(\theta\sqrt{m})$ et en simplifiant les formules mathématiques, on trouvera les équations suivantes:

$$\begin{cases} x_{i+1} = x_i - m d_i y_i 2^{-i} \\ y_{i+1} = y_i + d_i x_i 2^{-i} \\ z_{i+1} = z_i - d_i a_i \end{cases}$$

Avec m$\in \{-1, 0, 1\}$, a_i : des constantes définies à l'avance et $d_i \in \{-1, 1\}$ (en fonction de la valeur de Z_k).

Le tableau 2 résume les modes de fonctionnement de base de l'algorithme CORDIC, obtenus en combinant les deux modes de convergence (mode rotation et mode vecteur) avec les trois métriques utilisées.

Tableau 2 : Modes de fonctionnement de CORDIC généralisé

operation	configuration	initialization	output	post-processing and remarks
$\cos\theta, \sin\theta, \tan\theta$	CC-RM	$x_0 = 1$ $y_0 = 0$ and $\omega_0 = \theta$	$x_n = \cos\theta$ $y_n = \sin\theta$	$\tan\theta = (\sin\theta/\cos\theta)$
$\cosh\theta, \sinh\theta$ $\tanh\theta, \exp(\theta)$	HC-RM	$x_0 = 1$ $y_0 = 0$ and $\omega_0 = \theta$	$x_n = \cosh\theta$ $y_n = \sinh\theta$	$\tanh\theta = (\cosh\theta/\sinh\theta)$ $\exp(\theta) = (\cosh\theta + \sinh\theta)$
$\ln(a), \sqrt{a}$	HC-VM	$x_0 = a + 1$ $y_0 = a - 1$ and $\omega_0 = 0$	$x_n = \sqrt{a}$ $\omega_n = \frac{1}{2}\ln(a)$	$\ln(a) = 2\omega_n$
$\arctan(a)$	CC-VM	$x_0 = a$ $y_0 = 1$ and $\omega_0 = 0$	$\omega_n = \arctan(a)$	no pre- or post-processing
division (b/a)	LC-VM	$x_0 = a$ $y_0 = b$ and $\omega_0 = 0$	$\omega_n = b/a$	no pre- or post-processing
polar-to-rectangular	CC-RM	$x_0 = R$ $y_0 = 0$ and $\omega_0 = \theta$	$x_n = R\cos\theta$ $y_n = R\sin\theta$	no pre- or post-processing
rectangular-to-polar $\tan^{-1}(b/a)$ and $\sqrt{a^2 + b^2}$	CC-VM	$x_0 = a$ $y_0 = b$ and $\omega_0 = 0$	$x_n = \sqrt{a^2 + b^2}$ $\omega_n = \arctan(b/a)$	no pre- or post-processing

5. Architectures de l'Opérateur CORDIC

5.1 Architecture série de l'algorithme CORDIC

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. 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.

Les opérateurs shiften 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 FSM (finite state machine) élabore les signaux de commande des additionneur-soustracteurs (choix de l'opération), des décaleurs (valeur du décalage), en fonction des signes des variables courantes y_i, et z_i.
La figure 2 illustre ce type d'architecture.

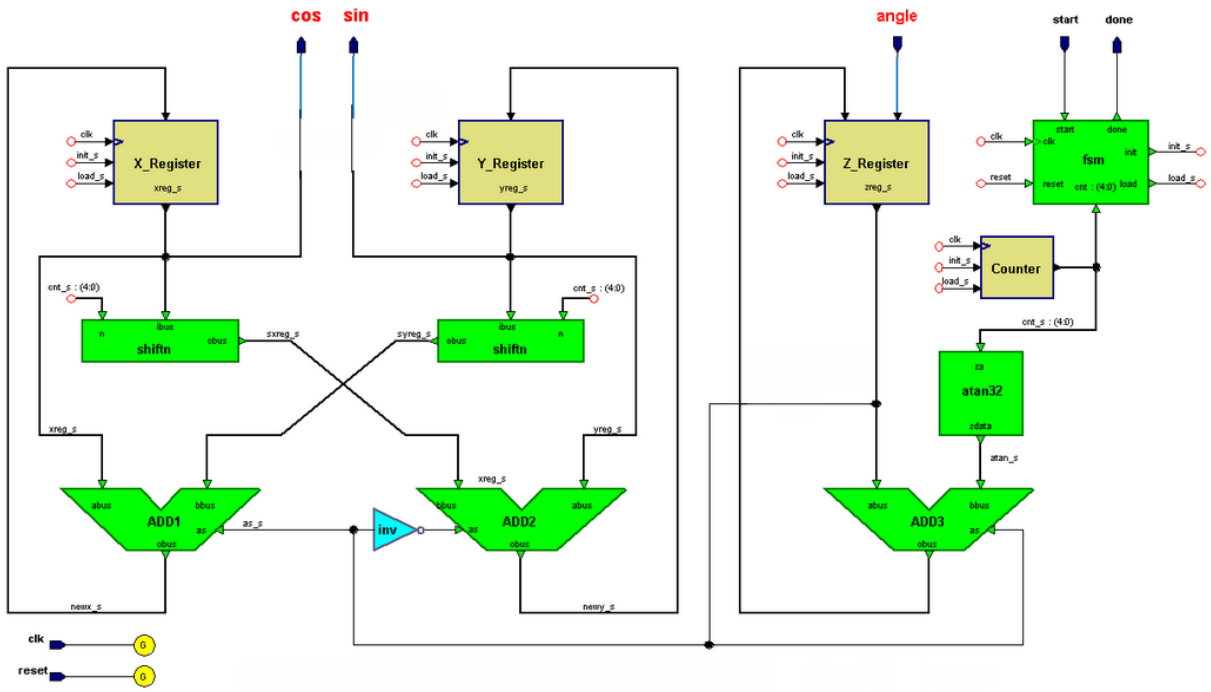

Figure 2 : Architecture série de l'algorithme CORDIC

5.2 Architecture parallèle

Au lieu de mettre en mémoire tampon la sortie d'une itération et d'employer les mêmes ressources de nouveau, on pourrait simplement monter en cascade le CORDIC itératif, qui signifie reconstruire la structure CORDIC fondamentale pour chaque itération. En conséquence, la sortie d'une étape est l'entrée de la prochaine, suivant les indications du la figure 3, et face aux étapes séparées deux simplifications deviennent possibles.

D'abord, les fonctionnements des décaleurs pour chaque opération peuvent être exécutés en câblant les connexions entre les étapes convenablement. En second lieu, il n'y a aucun besoin de changer les constantes et ceux peuvent pour cette raison être aussi bien câblés. Le modèle parallèle se compose seulement des composants combinatoires et calcule une valeur de sinus selon le cycle d'horloge.

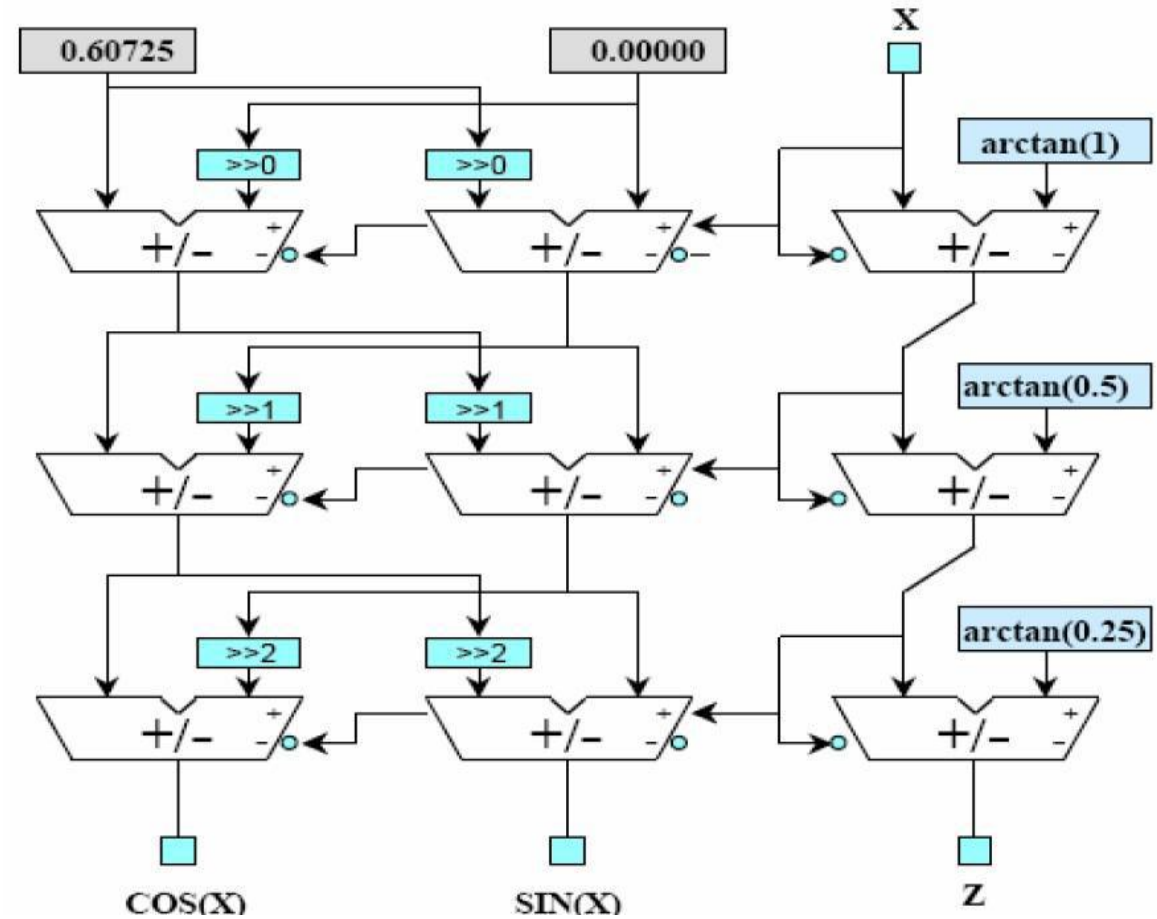

Figure 3 : L'architecture parallèle de l'algorithme CORDIC

5.3 Architecture pipeline de CORDIC :

Le même principe avec l'architecture parallèle de tel sorte quand 'on utilise plusieurs cellule CORDIC en cascade mais dans cette architecture entre chaque étage on insère des registres de piplinage qui permettent d'isoler la partie calcul de la partie mémorisation des variables intermédiaires.

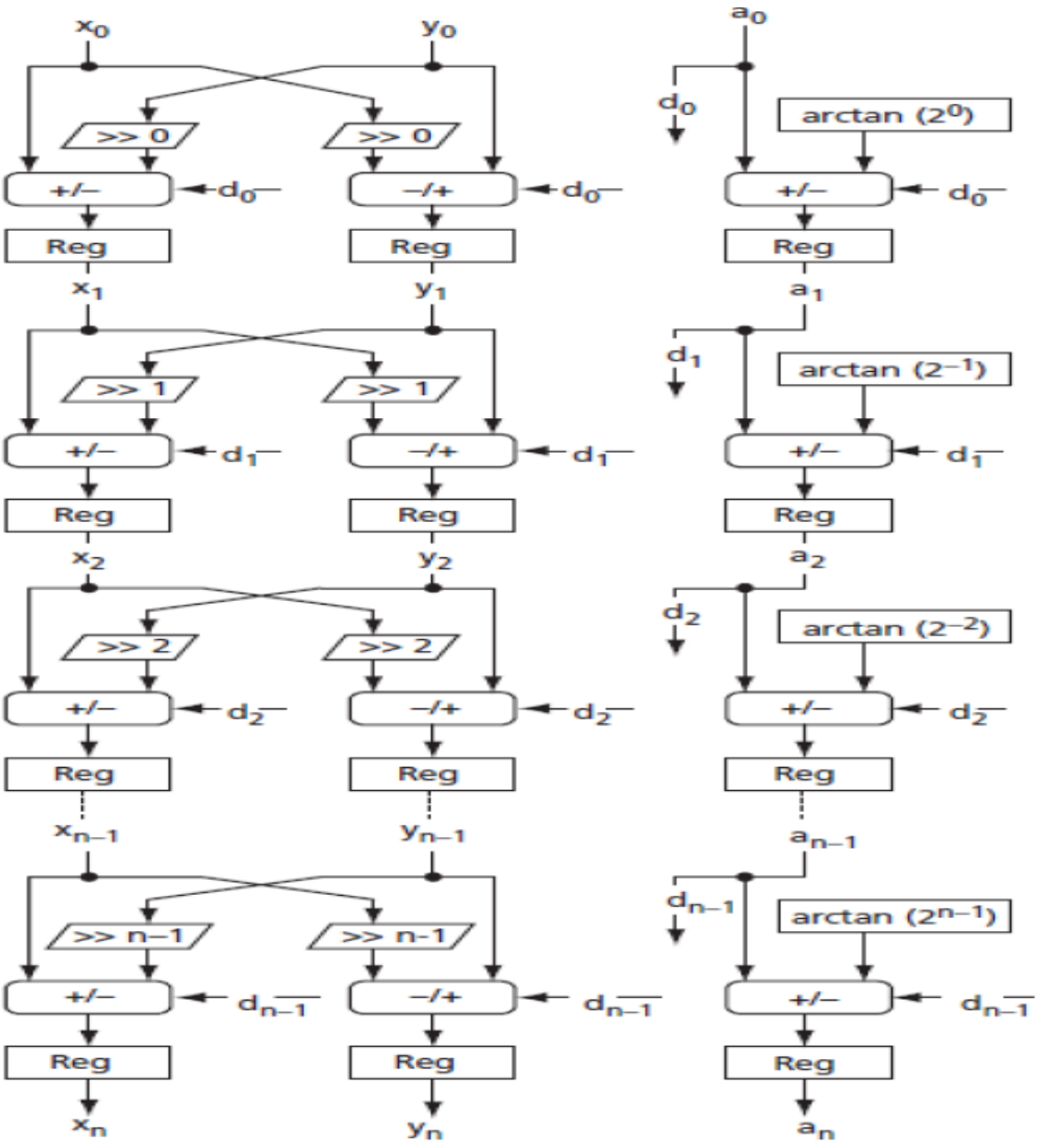

Figure 4 : L'architecture pipline de l'algorithme CORDIC

L’algorithme CORDIC présente donc un très gros avantage, puisque comme nous l’avons vu, il n’utilise aucune opération arithmétique complexe pour parvenir au résultat final puisque de la conception et l’implémentation d’un processeur CORDIC conventionnel est facilement réalisable mais le nombre des itérations reste un contrainte pour cet algorithme pour atteindre la précision désirée, de ce faite les chercheures ont pensé a trouver des méthodes efficaces pour réduire le nombre des itérations , dans ce qui suit on va faire une étude sur une de ces méthodes qui est la méthode de recodage des angles puis on va citer brièvement quelques méthodes similaires .

6. La méthode de recodage des angles

La méthode de recodage Angle a été proposée par Hu et Naganathan. Le recodage des angles utilise un algorithme glouton pour sauter certains angles de rotations, et peut réduire le nombre d'itérations nécessaires. Le nombre maximal d'itérations requis par cette méthode est N / 2, avec une valeur moyenne d'environ N / 3 itérations. La Méthode CORDIC qui rend l'utilisation de la méthode de recodage angle on l’appelle CORDIC adaptatif. [4]

Comme mentionné précédemment, dans les algorithmes conventionnels de CORDIC, d (i) = +/- 1. Par conséquent n itérations CORDIC sera toujours nécessaire, même si thêta = 0, parce que chaque fois d (i) = 1 ou -1 une itération CORDIC doit être calculé (en utilisant l'une opération décaleur et additionneur). Pour cela, nous proposons de relaxer cette contrainte en permettant d (i) = 0. Cela serait avantageux dans les applications où thêta est connue à l’avance. Si d (i) peut prendre +/- 1 ou 0, il serait souhaitable de réduire au minimum $\sum_{i=0}^{n} |d(i)|$ de sorte que le nombre total d'itérations CORDIC peut être réduit. Nous appellerons cette technique par le **recodage des Angles** car elle est similaire à la méthode de recodage multiplicateur employé dans la conception de multiplicateur moderne. Maintenant, le problème recodage angle peut être formellement déclaré [6]

6.1 Algorithme de recodage des angles [2]:

Initialisation : $\Theta_0 = \Theta$; d(i)=0 pour i=0 : n-1 , k=0,

Répéter tant que $|\Theta(k) < \text{atan}(2^{-n+1})|$ faire :

1 : choisir i_k , i_k= 0 :n-1 de tel sorte que :

Pour i=0 :n-1 , pour k=0 : n-1

$||\Theta_k| - \text{atan}(2^{-i_k})| = \min |(\Theta(k) - \text{atan}(2\text{^}-i))|$;

$\Theta(k+1) = \Theta(k) - d(i_k)a(i_k)$

Où $d(i_k) = \text{sign}(\Theta_k)$

Il s'agit d'un algorithme glouton, car à chaque étape, il tente de représenter l'angle restant (à faire tourner) en utilisant le plus proche d'un angle élémentaire CORDIC. Sans regarder devant étapes futures, ce choix est le plus raisonnable à l'itération courante.

Figure 5 montre la trajectoire angle de rotation tel qu'il s'approche de sa position cible finale de 25 ° à l'aide du procédé de recodage des angles. Il montre aussi la méthode originale de CORDIC pour la comparaison.

25°≈ 26.9°-1.8° +0.2°=25.0200°

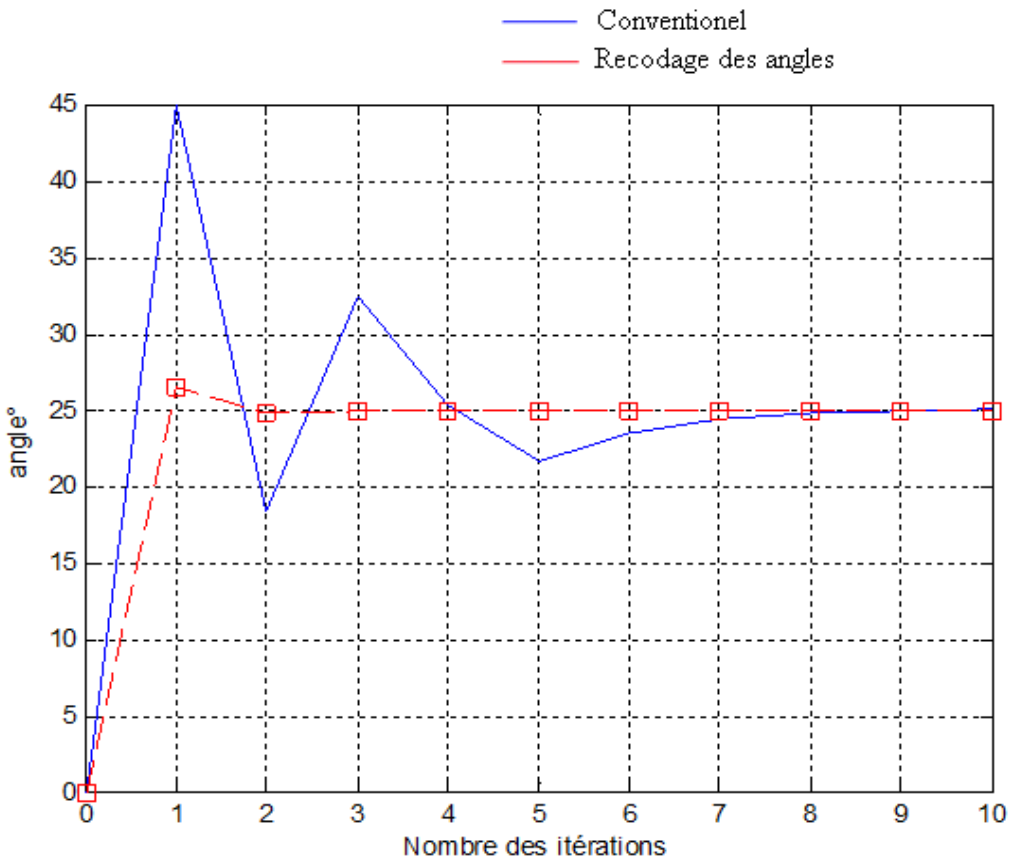

Figure 5 : comparaison entre CORDIC original et CORDIC adapté pour angle=25°

6.2 Architecture de l'algorithme CORDIC adaptatif

L'algorithme de recodage Angle par Hu et Naganathan est essentiellement un algorithme logiciel (software), et il doit être modifié pour le faire entrer dans une forme qui peut être mis

en œuvre dans le matériel, de manière à le rendre dynamique et capable de gérer n'importe quel angle de rotation. Pour ce faire, par substitution, l'analyse en série de constantes d'angle dans l'algorithme original par un test en parallèle à effectuer dans le matériel (hardware) [5] comme le montre dans la figure 6.

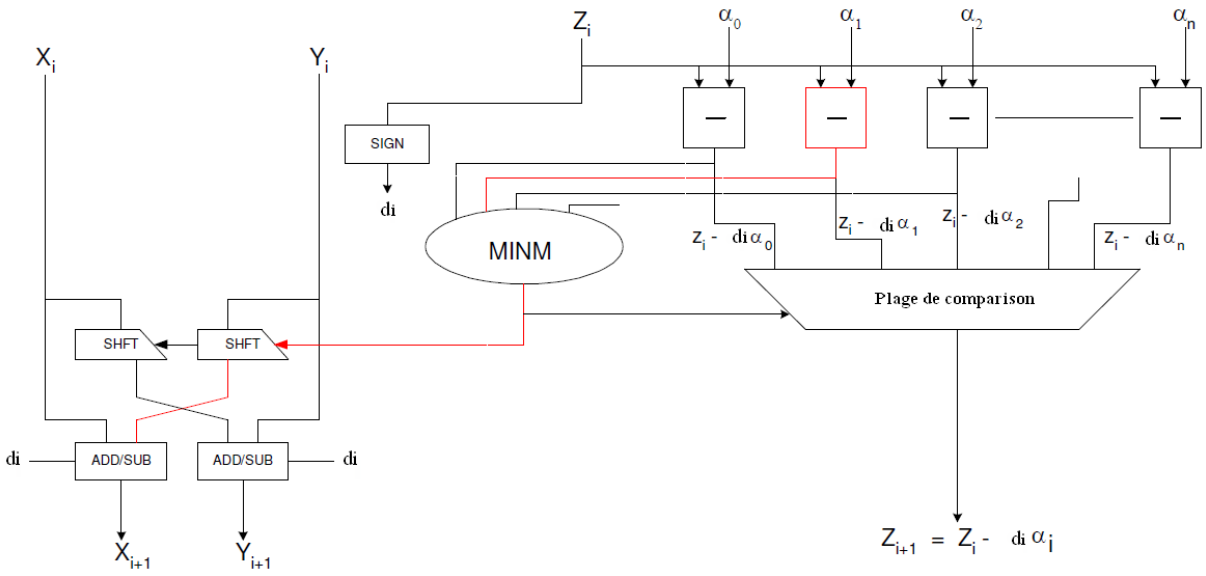

Figure 6 Architecture de CORDIC a recodage parallèle des angles

Son plus grand inconvénient, c'est que la fonction utilisée pour sélectionner le prochain constant d'angle est très complexe. Sa mise en œuvre dans le matériel pour une itération CORDIC, nécessite une augmentation de temps de cycle. Tout gain qui peut avoir été réalisés dans la latence globale de l'algorithme, grâce à l'utilisation de moins d'itérations, est donc annulé ou réduit par l'augmentation simultanée du temps de cycle qui est nécessaire pour l'opération de sélection angle dynamique [4].

Ce procédé est donc utilisé dans les cas statiques tels que l'angle de rotation (θ) est fixe et connue a priori de telle sorte que la sélection des constantes d'angle peut être fait hors ligne, et ceux constantes d'angle enregistrée pour une utilisation ultérieure dans un tableau look-Up , ou bien l'utilisation des additionneurs rapides ...etc.

7. CORDIC a Radix élevé :

L'algorithme CORDIC radix-r est donné par équations récursives ce qui suit [8]:

$$Xi = X_{(i-1)} - d_i . r^{-i} Y_{(i-1)}$$
$$Yi = Y_{(i-1)} + d_i . r^{-i} X_{(i-1)}$$
$$Zi = Z_{(i-1)} - a_i$$

$$a_i = \tan{-1}(d_i . r^{-i}) \quad i=1, 2,\ldots,n$$

La valeur de base «r» est supposé être une puissance de 2, soit $r = 2^m$, où «m» est un nombre entier qui n'est pas inférieure à 1.

'i' désigne l'étape d'itération, ai est la ième angle de rotation. Dans la rotation à High Radix CORDIC, Ki le facteur d'échelle donné par,

$$K=\prod_1^n Ki = 1/\prod_1^n \sqrt{(1 + di^2 r\text{^}(-2i))}$$

Il varie en fonction du choix de chiffres de rotation di.

Les coefficients di doivent être choisis de sorte que la coordonnée y devient plus petite.

7.1 CORDIC a radix 4 :

L'algorithme de CORDIC à radix 4 est donnée par

$$Xi = X_{(i-1)} - d_i . 4^{-i} Y_{(i-1)}$$
$$Yi = Y_{(i-1)} + d_i . 4^{-i} X_{(i-1)}$$
$$Zi = Z_{(i-1)} - a_i$$

$$a_i = \tan{-1}(d_i . 4^{-i}) \quad i=1,2,\ldots, n$$

Avec $d_i = \{-2,-1, 0, 1,2\}$

$$K=\prod_1^n Ki = 1/\prod_1^n \sqrt{(1 + di^2 4\text{^}(-2i))}$$

Le choix de di

Nous allons effectuer un changement de variable afin d'obtenir un système d'équations où CORDIC radix 4 peut être efficacement mises en œuvre. Nous définissons une nouvelle variable w_i pour trouver la fonction de sélection de di [8] :

$$Wi= 4^i y_i$$

On définit Pi (1) par le point comparaison utilisé pour discriminer les valeurs di = 0 et di = 1, et on définit Pi (2) que le point comparaison utilisé pour discriminer les di = 1 les valeurs et di = 2 (On définit Pi (-1) et Pi (-2) d'une manière similaire). Les points de comparaison que nous avons définis doivent appartenir à des intervalles se chevaucher et être facile à calculer et mettre en œuvre.

Deux bons choix pour les points de comparaison sont les suivants:

$$P_i (\pm 1) = \pm 1/2\ x_i$$

P_i (±2) = ± 3/2 x_i

Apres un calcul on constat que la fonction de sélection est la suivante :

$$\text{di} = \begin{cases} +2 & si \quad w_i > p_i(2) \\ +1 & si \quad p_i(1) < w_i \leq p_i(2) \\ 0 & si \quad p_i(-1) < w_i \leq p_i(1) \\ -1 & si \quad p_i(-2) < w_i \leq p_i(-1) \\ -2 & si \quad w_i \leq p_i(-2) \end{cases}$$

Pour une précision de sortie n bits, l'algorithme radix-4 CORDIC nécessite n/2 micro - rotations, ce qui est la moitié de celle de l'algorithme radix-2. Cependant, elle nécessite plus de temps de calcul pour chaque itération et implique plus de matériel par rapport à la CORDIC radix-2[2].

En 1996, E. Antelo et al ont proposé la méthode mixte radix-2 et radix-4 pour l'implémentation d'un processeur CORDIC dans l'architecture pipeline. La combinaison des deux radix peut réduire la latence et la superficie de pipeline [2]

La principale exigence pour cet algorithme, par rapport à la Radix 2 et Radix 4, correspond à une augmentation de la taille des tables pour stocker les angles élémentaires, la nécessité des tables pour les facteurs d'échelle ainsi que la nécessité de multiplicateurs rectangulaires au lieu d'additionneurs dans l'implémentation en série (pour l'implémentation en pipeline, nous estimons que le coût total des multiplicateurs rectangulaires et des additionneurs est comparable).

8. Conclusion

L’algorithme de CORDIC est une méthode simple et efficace pour le calcul d’une gamme de fonctions complexes. S’appuyant sur une technique décalage-addition et de rotation vectorielle, l’algorithme calcul par approximation la plupart des fonctions basées sur la trigonométrie.

Selon l’approches de la réduction de du nombre des itérations plusieurs méthodes existes pour l'algorithme de CORDIC haute performance, en réduisant la latence de calcul et en augmentant la précision du calcul, en plus de la méthode de recodage des angles et le CORDIC a radix élevé on cite:

- Méthode de rotation parallèle,
- La représentation des nombres redondants,
- Procédé de l'extension de rotation.

Bibliographie

[1]ELE4304 – Principe des Circuits Intégrés à Très Grande Échelle Projet : Algorithme de CORDIC

[2]Pramod K. Meher, Javier Valls, Tso-Bing Juang, K. Sridharan,et Koushik Maharatna: 50 Years of CORDIC: Algorithms, Architectures, and Applications

[4] Terence Keith Rodrigues, B.E.; M.S : Adaptive CORDIC: Using Parallel Angle Recoding to Accelerate CORDIC Rotations: These Doctorat.

[5] Terence Keith Rodrigues: Adaptive CORDIC: Using Parallel Angle Recoding to Accelerate CORDIC Rotations2007

[6] Yu Hen Hu et S. Naganathan : Angle recoding method for efficient Implementation of the CORDIC algorithm

[7]Hongzhi WANG : Architectures reconfigurables à base d'opérateur CORDIC pour le traitement du signal: Applications aux récepteurs MIMO

[8] J. Villalba, J.C. Arrabal, E. Antelo, J.D. Bruguera et E.L. Zapata Radix-4: Vectoring CORDIC Algorithm and Architectures.