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'algorithme CORDIC

Proposé et dirigé par:

Mr Taghi Mohamed Oussaid

Mr Abdelouel Lahcene

Réalisé par:

Tiouidiouine zohra

2012

SOMMAIRE

1 -INTRODUCTION

Utiliser sa calculatrice pour déterminer la valeur d'un cosinus, d'un logarithme ou d'une racine carrée est un geste devenu tellement banal que plus personne ne se demande pourquoi et comment cela marche. Mais, pour reprendre une formule bien connue, a-t-on vraiment besoin de soulever le capot de sa voiture et de comprendre le fonctionnement du moteur pour s'en servir ? Evidemment non.

Pourtant, il nous arrive souvent de poser la question est ce que extraire la racine carrée à la main est le même principe qu'utilise une calculatrice ? On pose toujours cette question sans pouvoir y répondre. On pourrait même aller plus loin en se demandant comment la calculatrice peut donner la valeur d'un sinus ou d'un logarithme avec autant de précision.
Quand il s'agit d'opérations simples telles que l'addition ou la multiplication, on peut assez bien imaginer comment elle procède parce que nous savons faire ces opérations a la main.
Mais pour les autres fonctions, comment fait-elle réellement ? Utilise-t-elle des développements limites, des approximations de fonctions, ou d'autres mécanismes plus complexes ? C'est en fouillant a droite et a gauche que j'ai trouvé que la majorité des calculatrices commerciales utilise l'algorithme CORDIC. Alors la question qui se pose elle-même c'est quoi l'algorithme CORDIC ?

2-HISTORIQUE

A la fin des années 50, les calculateurs électroniques sont en plein essor. Les domaines ou ils interviennent sont de plus en plus varies et on les trouve par exemple dans les avions ou ils servent d'assistants à la navigation aérienne. De ce fait, les besoins de calculs en temps réel se font de plus en plus sentir, notamment pour les calculs trigonométriques. Ainsi, en 1959, Jack E.Volder met au point un algorithme qui permet d'approximer des fonctions trigonométriques à partir d'opérations élémentaires (additions, soustractions et multiplications). Cet algorithme appelé ≪ algorithme CORDIC ≫ (pour Coordinate Rotation DIgital Computer) repose, comme son nom l'indique, sur le calcul des coordonnées de vecteurs auxquels on applique une rotation bien choisie. Très vite, grâce a sa mise en œuvre matérielle très simple (l'ingéniosité de l'algorithme permet en effet un câblage électronique extrêmement simple), l'algorithme CORDIC est repris dans des domaines très diverses tels que : le traitement de signaux radars, les coprocesseurs mathématiques (intel i8087 par exemple) ou les calculatrices scientifiques (toutes marques confondues). Un autre avantage non négligeable de cet algorithme est qu'il permet d'obtenir une précision déterminée à l' avance en effectuant un nombre d'itération donné.

3- PRINCIPE DE L'ALGORITHME CORDIC :

L'algorithme CORDIC est basé sur des calculs des fonctions trigonométriques, son principe est d'effectuer des rotations sur un vecteur de base pour un angle donné. Si un vecteur V avec des coordonnées (x, y) est tourné par un angle Φ alors qu'un nouveau vecteur V' (x',y') où x' et y' peuvent être obtenus utilisant x, y et Φ par la méthode suivante.

$$X = r\cos\theta,\ Y = r\sin\theta \qquad \textbf{(3.1)}$$

$$V' = \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x.\cos(\Phi) - y.\sin(\Phi) \\ y.\cos(\Phi) + x.\sin(\Phi) \end{pmatrix} \quad \textbf{(3.2)}$$

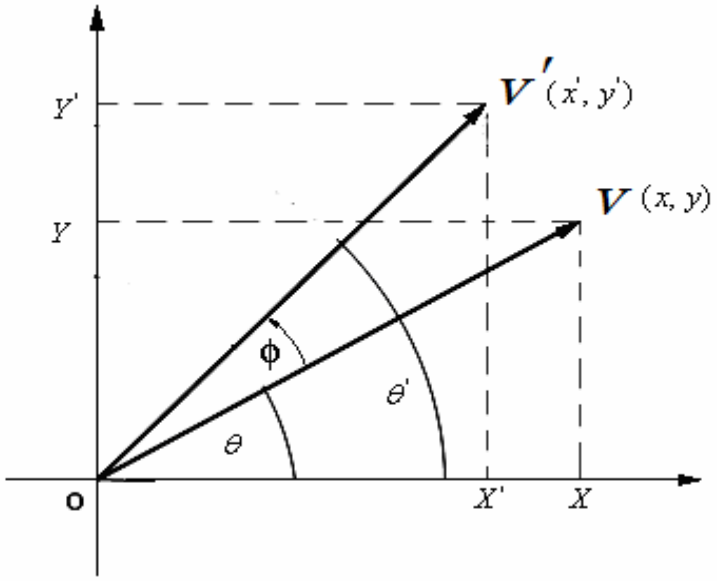

Figure 3.1: Rotation d'un vecteur V par un angle Φ

Dans la figure 3.1 le vecteur V (x, y) peut être décomposé en deux composantes suivant l'axe des x r cosθ et suivant l'axe des y r sinθ.

La figure 3.2 montre la rotation de vecteur V= $\begin{pmatrix} x \\ y \end{pmatrix}$ par l'angle Φ.

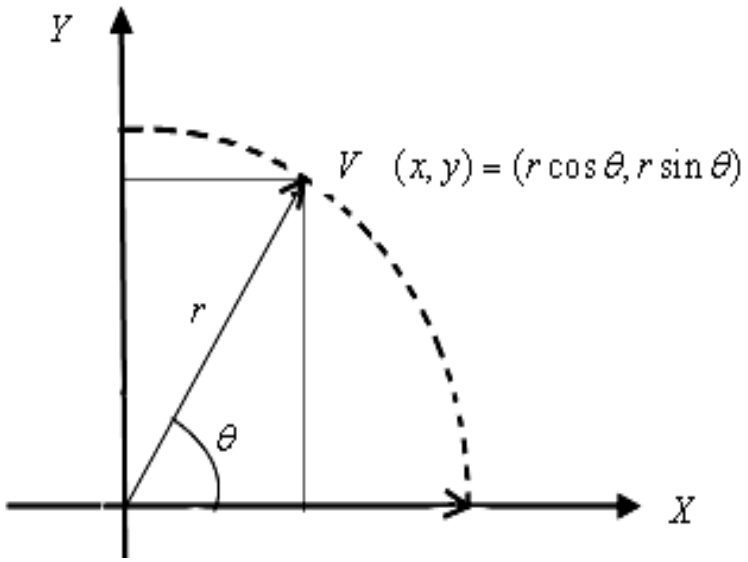

Figure 3.2: vecteur V d'amplitude r et d'argument θ

i.e. $\begin{cases} x = r\cos\theta \\ y = r\sin\theta \end{cases}$ **(3.3)**

Les deux vecteur V et V′ ont leurs amplitudes et arguments (r, θ) et (r,θ′) respectivement où V′ est la rotation de vecteur V par un angle Φ dans le sens inverse des aiguilles d'une montre. De la figure 2.1 on voit que

$\theta' - \theta = \Phi$ **(3.4)**

i.e. $\theta' = \theta + \Phi$ **(3.5)**

$$OX' = x' = r\cos\theta'$$
$$= r\cos(\theta + \phi)$$
$$= r(\cos\theta.\cos\phi - \sin\theta.\sin\phi)$$
$$= (r.\cos\theta)\cos\phi - (r.\sin\theta)\sin\phi \quad \textbf{(3.6)}$$

Utilisant la figure 3.2 et l'équation 3.3 OX' peut être représenté comme suit :

$OX' = x' = x\cos\phi - y\sin\phi$ **(3.7)**

La même chose pour OY'

$OY' = y' = y\cos\phi + x\sin\phi$ **(3.8)**

De la même façon si on fait tourner le vecteur V d'un angle ϕ dans le sens des aiguilles d'une montre, on obtient les équations suivantes :

$x' = x\cos\phi + y\sin\phi$ **(3.9)**

$y' = x\sin\phi - y\cos\phi$ **(3.10)**

Les équations (2.7), (2.8), (2.9), (2.10) peuvent être représentées dans la matrice suivante

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\Phi & \pm\sin\Phi \\ \pm\sin\Phi & \cos\Phi \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \quad \textbf{(3.11)}$$

Les équations séparées de x' et de y' peuvent être réécrite comme suit

$x' = x.\cos(\phi) \pm y.\sin(\phi)$ **(3.12)**

$y' = y.\cos(\phi) \pm x.\sin(\phi)$ **(3.13)**

On fait sortir cos Φ en facteur, ensuite on considère l'angle θ comme un ensemble de petits angles dont le tangent de chaque un est choisit comme une puissance de deux inverse.

Donc les équations peuvent être réécrites comme des formules itératives :

$x' = \cos\phi(x \pm y\tan\phi)$ **(3.14)**

$y' = \cos\phi(y \pm x\tan\phi)$ **(3.15)**

$z' = z \pm \phi$, ici ϕ est l'angle de rotation (le signe ± est la direction de rotation) et z est l'argument. Pour la facilité des calcules on prend d'abord la rotation dans le sens inverse des aiguilles d'une montre. Réarrangeant les équations (2.7) et (2.8).

$x' = \cos(\phi)[x - y.\tan(\phi)]$ **(3.16)**

$y' = \cos(\phi)[y + x.\tan(\phi)]$ **(3.17)**

Le tan (ϕ) est limité pour une valeur de 2^{-i}. Dans le digital hardware ceci indique une simple opération de décalage. En outre, si ces rotations sont exécutées itérativement et dans les deux sens chaque valeur de tan (Φ) est représentable. Avec $\phi = \arctan(2^{-i})$ le terme cosinus peut être également simplifié $\cos(\phi) = \cos(-\phi)$ c'est une constante pour un nombre fixe d'itérations. Maintenant cette rotation itérative peut être exprimée comme :

$$x_{i+1} = k_i[x_i - y_i\, d_i\, 2^{-i}] \quad \textbf{(3.18)}$$

$$y_{i+1} = k_i[y_i + x_i\, d_i\, 2^{-i}] \quad \textbf{(3.19)}$$

Où, i indique le nombre des rotations exigé pour atteindre l'angle demandé du vecteur demandé, $k_i = \cos(\arctan(2^{-i}))$ et $d_i = \pm 1$. Le produit des k_i représente ce qu'on appelle le facteur K :

$$K = \prod_{i=0}^{n-1} k_i \quad \textbf{(3.20)}$$

Où, $\prod_{i=0}^{n-1} k_i$= cosΦ_0 cosΦ_1 cosΦ_2 cosΦ_3…… cos Φ_{n-1}(Φ est l'angle de rotation ici pour n rotations).

Table 3.1: 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

K est le gain sa valeur change avec l'augmentation de nombre d'itération. Pour le hardware CORDIC de 8-bit la valeur approximative de K

$$K = \prod_{i=0}^{7} \cos\Phi_i = \cos\Phi_0 \cos\Phi_1 \cos\Phi_2 \cos\Phi_3 \cos\Phi_4 \cos\Phi_5 \cos\Phi_6 \cos\Phi_7$$
$$= \cos 45°.\cos 26.565°.....\cos 0.4469°$$
$$= 0.6073 \quad \textbf{(3.21)}$$

Du tableau ci-dessus on peut voir jusqu'à la précision 0.4469° pour le hardware CORDIC de 8- bits. Ces Φ_i sont enregistrés dans la ROM de hardware CORDIC comme un tableau de consultation (LUT look up table). Maintenant on prend l'exemple de balance on comprend comment l'algorithme CORDIC travaille.

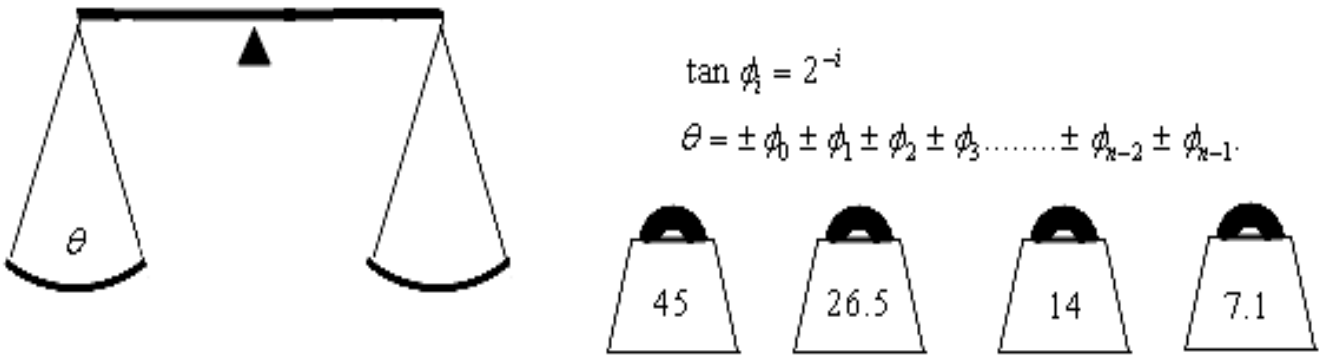

Figure 3.3: Une balance ayant θ sur un côté et des petits poids (angles) sur l'autre côté.

Dans la figure ci-dessus, tout d'abord, maintenir l'angle d'entrée θ sur le côté gauche de la balance et si la balance tourne autour en sens inverse des aiguilles d'une montre alors ajoute la valeur la plus élevée dans le tableau sur l'autre côté.

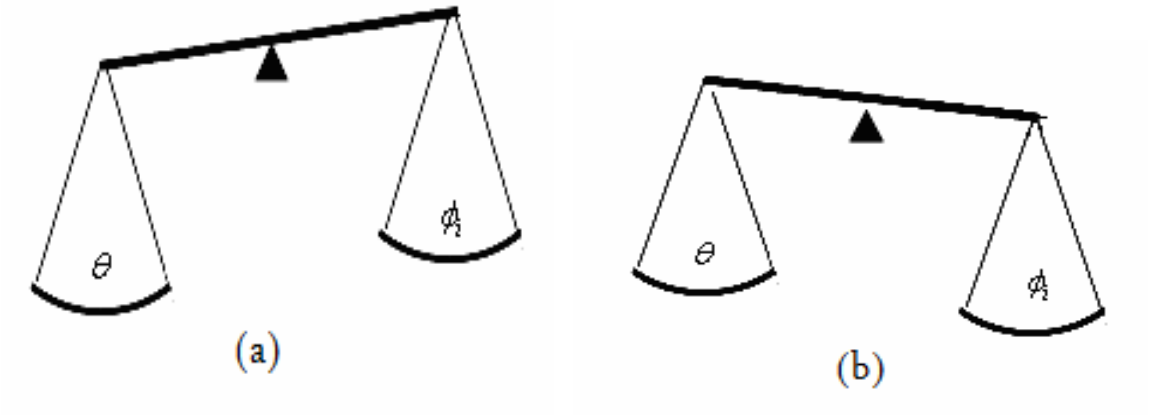

Figure 3.4: l'inclination de la balance due à la différence du poids des deux côté.

Donc, si la balance montre une inclination gauche comme sur le schéma 2.4 (a) puis d'autres poids sont exigés pour être ajouté dans le côté droit ou dans le terme de l'angle si le θ est plus grand que le total Φ_i donc ajouter des poids aussi possible pour atteindre θ mais dans l'autre main si la balance montre une inclination comme sur le schéma 2.4 (b) puis des poids exigés pour être retiré du côté droit ou dans le terme d'angle si le θ est moins que le total Φ_i le processus est répété aussi possible pour atteindre l'angle θ.
Une représentation matricielle de l'algorithme CORDIC pour un hardware de 8-bit :

$$\begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix} = \begin{pmatrix} \cos\Phi_i & \pm\sin\Phi_i \\ \pm\sin\Phi_i & \cos\Phi_i \end{pmatrix} \begin{pmatrix} xi \\ yi \end{pmatrix} \qquad \textbf{(3.22)}$$

$$\begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix} = \begin{pmatrix} \cos\Phi_0 & \pm\sin\Phi_0 \\ \pm\sin\Phi_0 & \cos\Phi_0 \end{pmatrix} \begin{pmatrix} \cos\Phi_1 & \pm\sin\Phi_1 \\ \pm\sin\Phi_1 & \cos\Phi_1 \end{pmatrix} \ldots\ldots$$
$$\ldots\ldots \begin{pmatrix} \cos\Phi_7 & \pm\sin\Phi_7 \\ \pm\sin\Phi_7 & \cos\Phi_7 \end{pmatrix} \begin{pmatrix} x0 \\ y0 \end{pmatrix} \qquad \textbf{(3.23)}$$

$$\begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix} = \cos\Phi_0 \cos\Phi_1 \ldots\ldots\ldots \cos\Phi_7 \begin{pmatrix} 1 & \pm\tan\Phi_0 \\ \pm\tan\Phi_0 & 1 \end{pmatrix} \begin{pmatrix} 1 & \pm\tan\Phi_1 \\ \pm\tan\Phi_1 & 1 \end{pmatrix}$$
$$\ldots\ldots\ldots \begin{pmatrix} 1 & \pm\tan\Phi_7 \\ \pm\tan\Phi_7 & 1 \end{pmatrix} \begin{pmatrix} x0 \\ y0 \end{pmatrix} \qquad \textbf{(3.24)}$$

Ainsi, le facteur d'échelle = $\cos\Phi_0 \cos\Phi_1 \ldots\ldots\ldots \cos\Phi_7$
$= 0.6073$
$= \frac{1}{1.6466}$ **(3.25)**

De l'équation (3.22) le cosinus et le sinus de l'angle θ peuvent être représenté sous la forme d'une matrice comme

$$\begin{pmatrix} cos\theta \\ sin\theta \end{pmatrix} = \begin{pmatrix} 1 & \pm\tan\ \Phi_0 \\ \pm\tan\Phi_0 & 1 \end{pmatrix}\begin{pmatrix} 1 & \pm\tan\ \Phi_1 \\ \pm\tan\Phi_1 & 1 \end{pmatrix}\ldots.$$
$$\ldots.\begin{pmatrix} 1 & \pm\tan\ \Phi_7 \\ \pm\tan\Phi_7 & 1 \end{pmatrix}\begin{pmatrix} 0.6073 \\ 0 \end{pmatrix} \quad \textbf{(3.26)}$$

D'une manière générale le facteur d'échelle K=$\prod_1^n Ki = 1/\prod_1^n \sqrt{(1+2\hat{}(-2i)}$

3.1 LES ITERATIONS FONDAMENTALES DE L'ALGORITHME CORDIC :

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

$$x_{i+1} = x_i - y_i\ d_i\ 2^{-i} \quad \textbf{(3.27)}$$

$$y_{i+1} = y_i + x_i\ d_i\ 2^{-i} \quad \textbf{(3.28)}$$

$$z_{i+1} = z_i - d_i\ tan^{-1}\ 2^{-i} \quad \textbf{(3.29)}$$

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 (Tableau 2.1) 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 ≈ 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 2.2 montre le procédé de sélectionner les signes des angles tournés pour une rotation désirée de +30 degrés. La Figure 2.5 dépeint les étapes premières du procédé de forcer z à zéro.

Tableau 3.2 : 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

Dans la terminologie CORDIC la règle de sélection précédente pour d_i, qui fait converger z vers zéro, est connue comme **mode de rotation**. Réécriture de l'itération CORDIC, où $\alpha_i = tan^{-1}\ 2^{-i}$:

$$x_{i+1} = x_i - y_i\ d_i\ 2^{-i} \quad \textbf{(3.30)}$$

$$y_{i+1} = y_i + x_i\ d_i\ 2^{-i} \quad \textbf{(3.31)}$$

$$z_{i+1} = z_i - d_i\ \alpha^i \quad \textbf{(3.32)}$$

Après m itérations en mode rotation, et quand z (m) tend vers zéro. Nous avons $\sum \alpha_i$ = z, et les équations CORDIC sont :

$$x_m = k\ (x\cos z - y\sin z) \quad \textbf{(3.33)}$$

$$y_m = k\ (y\cos z + x\sin z) \quad \textbf{(3.34)}$$

$$z_m = 0 \quad \textbf{(3.35)}$$

Règle : choisissant $d_i \epsilon$ {-1, 1} tel que $z \rightarrow 0$

La constante K dans les équations précédentes est k = 1.646760258121…..ainsi, pour calculer cos z et sin z, on peut commencer 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 z et sin z, respectivement.

Tableau 3.3 : Le choix des signes de la rotation des angles pour forcer z à zéro

i	$z_i - \alpha_i$	z_{i+1}
0	+ 30.0 - 45.0	-15
1	- 15.0 + 26.6	11.6
2	+ 11.6 - 14.0	-2.4
3	- 2.4 + 7.1	4.7
4	+ 4.7 - 3.6	1.1
5	+ 1.1 -1.8	-0.7
6	- 0.7 + 0.9	0.2
7	+ 0.2 - 0.4	-0.2
8	- 0.2 + 0.2	0
9	+ 0.0 - 0.1	-0.1

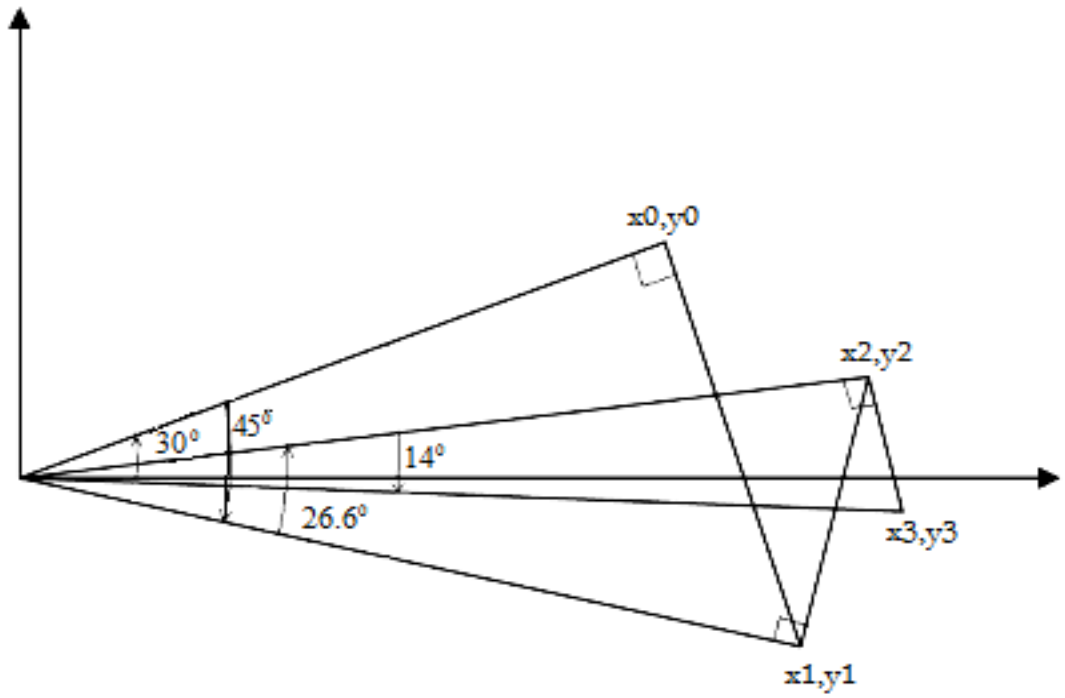

Figure 3.5: les trois premières itérations parmi les 10 principales dans la rotation par 30°,

mode Rotation.

Pour une précision de k bits des fonctions trigonométriques résultantes, on a besoin k itérations CORDIC. La raison est que pour un i grand on peut faire l'approximation $tan^{-1}\ 2^{-i} \approx 2^{-i}$. D'où, pour i > k,

En mode Rotation, la convergence de z à zéro est possible parce que chaque angle dans le Tableau 3.4 est plus que moitié de l'angle précédent ou, d'une manière équivalente, chaque angle est moins que la somme des angles qui le suivent. Le domaine de convergence est -99.7 <z <99.7, où 99.7 est la somme de tout les angles dans le Tableau 2.4. Heureusement, cette gamme comprend les angles de -90 à +90, où [-π/2 à +π/2] en radian. En dehors de la gamme précédente, des identités trigonométriques peuvent être converties en problème, à un qui est dans le domaine de convergence :

$$\cos (z \pm 2j\pi) = \cos z \qquad \textbf{(3.36)}$$

$$\sin (z \pm 2j\pi) = \sin z \qquad \textbf{(3.37)}$$

$$\cos (z - \pi) = - \cos z \qquad \textbf{(3.38)}$$

$$\sin (z - \pi) = -\sin z \qquad \textbf{(3.39)}$$

Notant que ces transformations deviennent particulièrement pratique si les angles sont représentées en multiple de π, de sorte que z = 0.2 réellement signifie que z = 0.2π ou converti en nombres dans le domaine bien facilement.

Dans la deuxième voie par l'utilisation des itérations CORDIC, connue sous le nom de «**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 tan($\sum \alpha_i$) = -y/x.

Ceci signifie que :

$$x_m = k\,[x\cos(\textstyle\sum \alpha_i) - y\sin(\sum \alpha_i)] \qquad \textbf{(3.40)}$$

$$x_m = k(x - y\tan(\textstyle\sum \alpha_i))/[1 + tan^2(\sum \alpha_i)]^{1/2} \qquad \textbf{(3.41)}$$

$$x_m = k(x + y^2 / x)/ (1+ y^2/x^2) \qquad \textbf{(3.42)}$$

$$x_m = k\,(x^2 + y^2)^{1/2} \qquad \textbf{(3.43)}$$

Les équations CORDIC sont ainsi :

$$x_m = k\,(x^2 + y^2)^{1/2} \qquad \textbf{(3.44)}$$

$$y_m = 0 \qquad \textbf{(3.45)}$$

$$z_m = z + \tan^{-1}(y/x) \qquad \textbf{(3.46)}$$

Règle : choisissant $d_i \epsilon \{-1, 1\}$ tel que $y \rightarrow 0$

On peut calculer $\tan^{-1} y$ dans le mode Vecteur en commençant par x = 1 et z = 0. Ce calcul converge toujours. [1]

4-ARCHITECTURES DE L'OPERATEUR CORDIC

4-1 ARCHITECTURE SERIE 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. La figure 2 illustre ce type d'architecture.

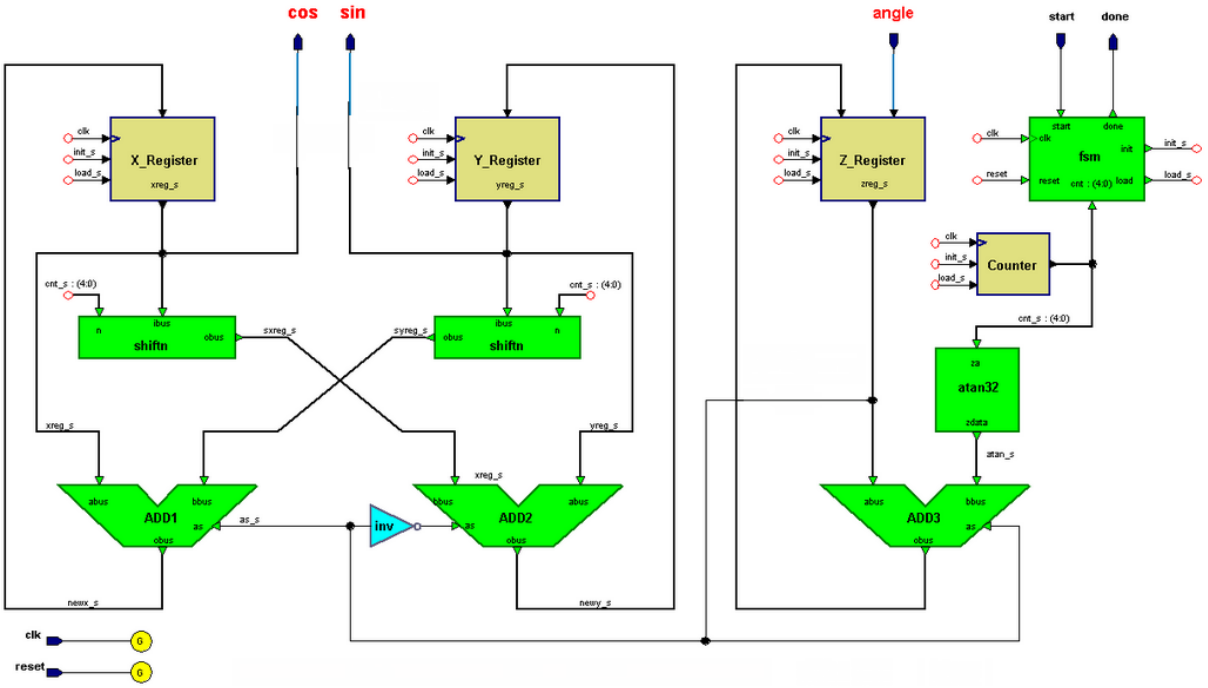

Figure 4.1 : Architecture série de l'algorithme CORDIC

4-2 ARCHITECTURE PARALLELE

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

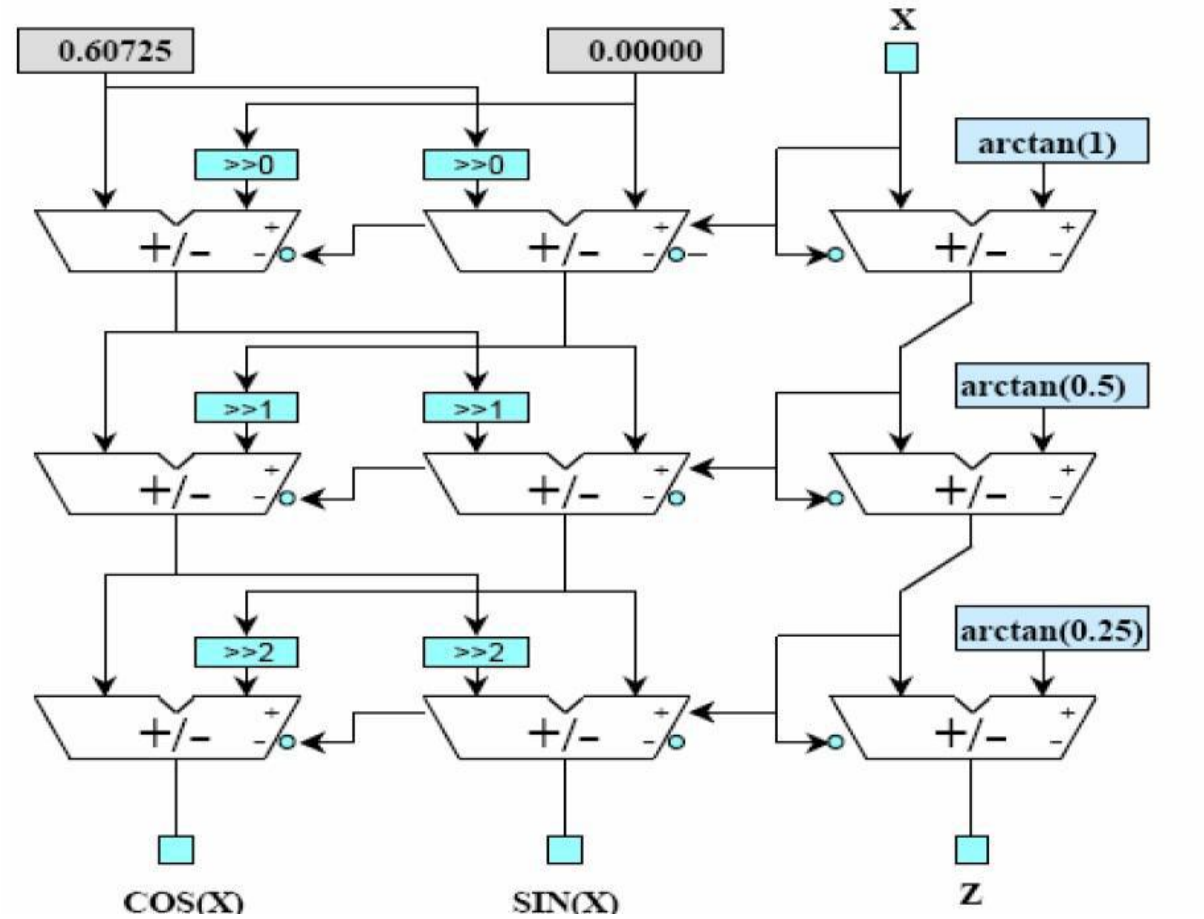

Figure 4.2 : L'architecture parallèle de l'algorithme CORDIC

4-3 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.

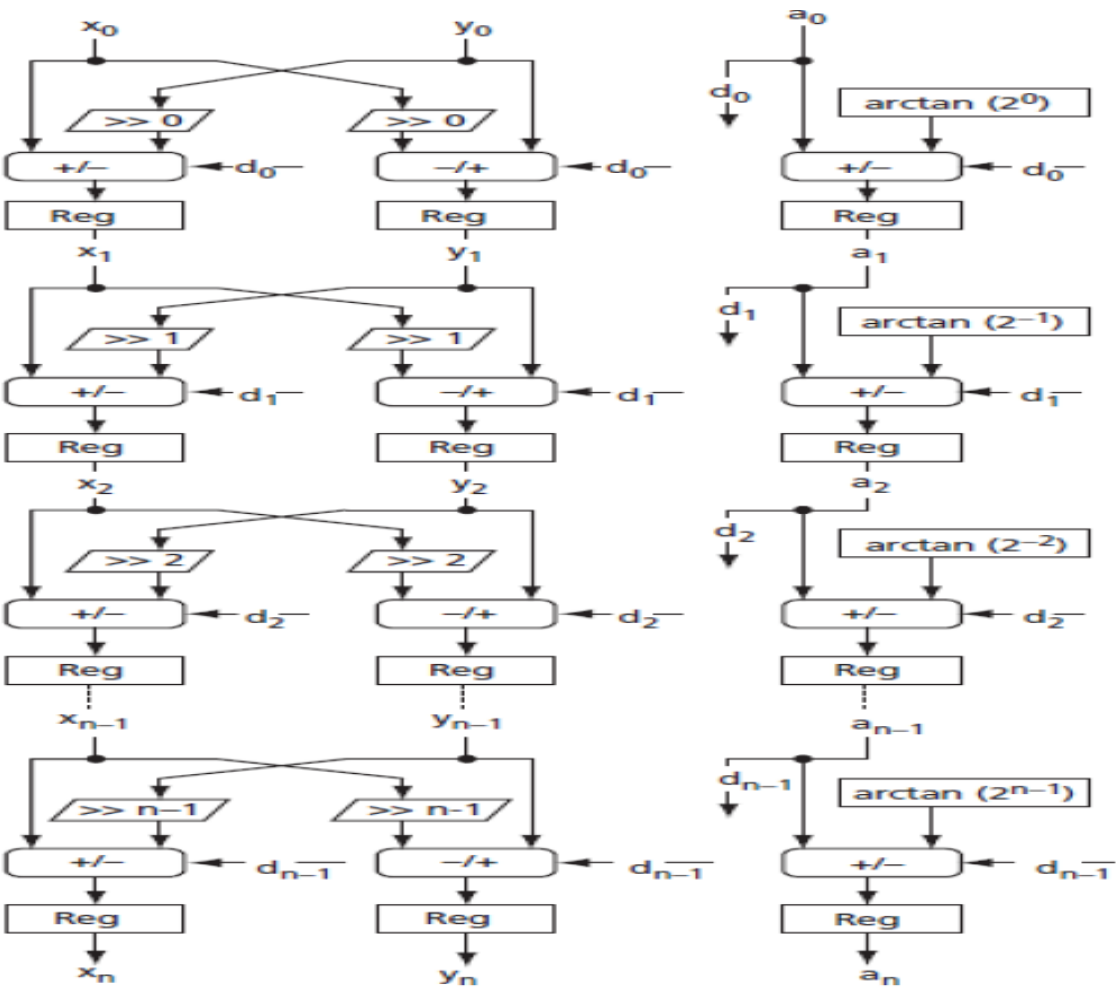

Figure 4.3 : L'architecture pipline de l'algorithme CORDIC

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 . [2]

5- LA METHODE 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]

5-1 ALGORITHME DE RECODAGE DES ANGLES :

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 6 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. [3]

$25° \approx 26.9° - 1.8° + 0.2° = 25.0200°$

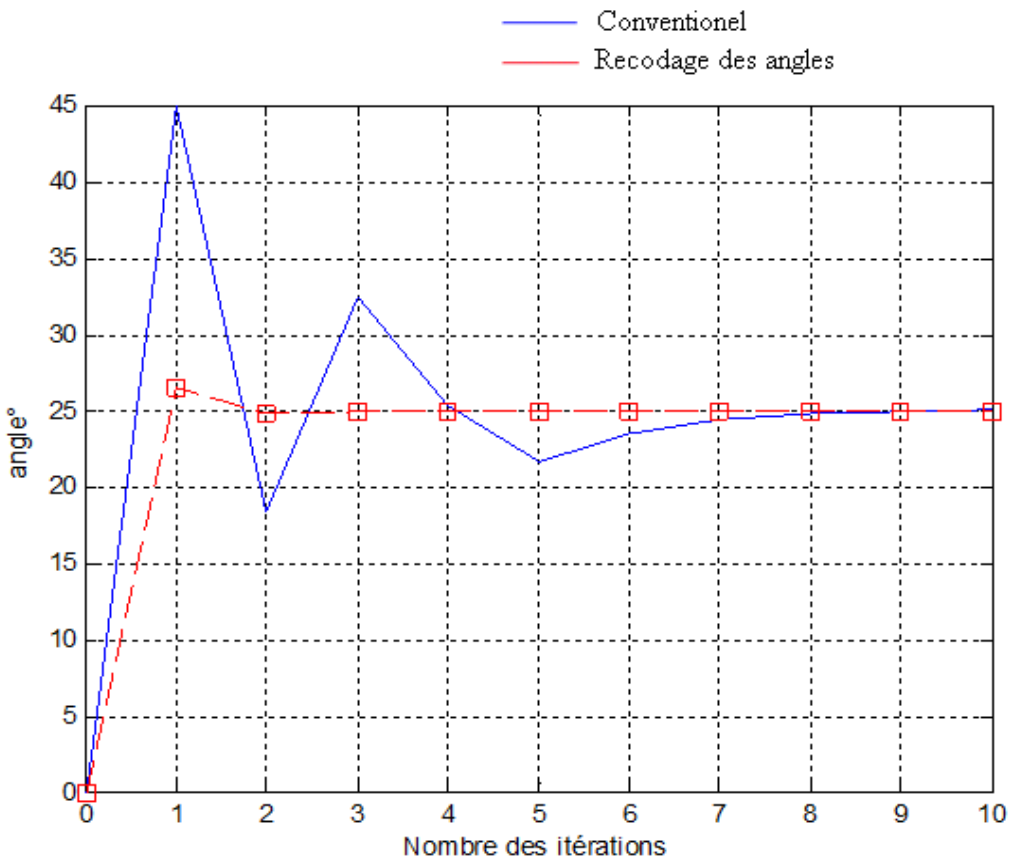

Figure 6 : comparaison entre CORDIC original et CORDIC adapté pour angle=25°

7- CONCLUSION

On voit qu'un besoin très simple de calculs en temps réel datant des années 50 a fini par déboucher sur la mise au point d'une multitude d'algorithmes permettant d'approcher la plupart des fonctions usuelles. Il est d'ailleurs étonnant, lorsqu'on se penche sur ce genre de problème, de constater la diversité des théories mathématiques mises en œuvre pour les résoudre. Par exemple, pour effectuer des calculs avec un grand nombre de chiffres (dont la précision est fixée à l' avance), certains calculateurs utilisent une approximation polynomiale ou rationnelle de la fonction calculée ou bien des algorithmes de type CORDIC. Mais si la précision doit être dynamique, les calculateurs vont plutôt utiliser des développements de Taylor ou des méthodes quadratiques utilisant la moyenne "arithmetico-geometrique" de Gauss−Legendre.

Finalement, on voit bien que ce sont les mathématiques qui ont permis le développement du calcul électronique et, plus encore aujourd'hui avec la généralisation du calcul formel et des calculs en grande précision, elles restent indispensables a la conception de logiciels pour les ordinateurs et les calculatrices.

BIBLIOGRAPHIE:

[1] THÈSE DE DOCTORAT:' FPGA IMPLEMENTATION OF DFT USING CORDIC ALGORITHM' Par 'Vikas Kumar'

[2]' Design and FPGA Implementation of CORDIC-based 8-point 1D DCT Processor' par

Rohit Kumar Jain

[3]' A VHDL Implementation of a CORDIC Arithmetic Processor Chip ' par Grant Hampson et Andrew Paplinski