HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD

Prof. Yann Thoma and Alberto Dassatti

Laboratoire SCF

semestre de printemps 2024 - 2025

Calculateur CORDIC

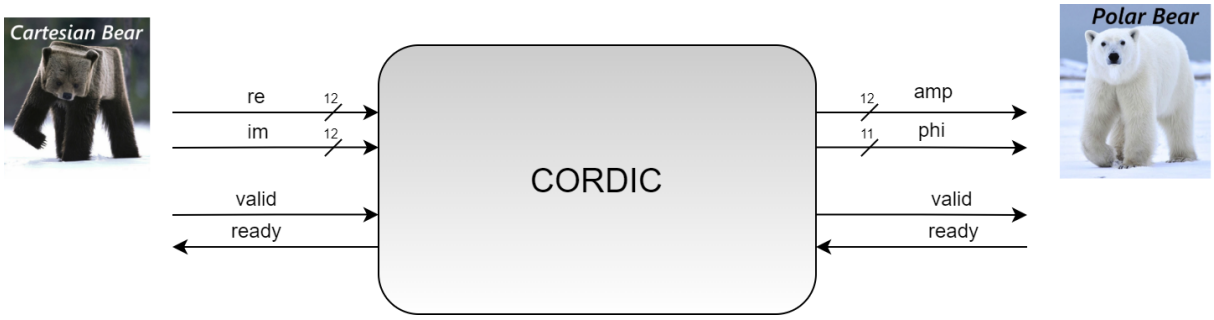
Objectifs pédagogiques

Ce laboratoire a pour but de réaliser un calculateur CORDIC permettant de transformer des coordonnées cartésiennes en coordonnées polaires. Plusieurs implémentations seront étudiée et réalisées (combinatoire, pipeline, séquentielle) afin de mettre en évidence leurs avantages et inconvénients en termes de débit, fréquence de fonctionnement et utilisation de la logique. Un banc de test SystemVerilog devra également être développé.

Cahier des charges

Il vous est demandé de réaliser un calculateur CORDIC (COordinate Rotation Digital Computer), très utile pour calculer entre autres des fonctions trigonométriques dans un système numérique sans exploiter des ressources matérielles spécialisées.

Dans le cadre de ce laboratoire, votre calculateur CORDIC doit calculer l'amplitude et la phase (aussi appelé module et argument) d'un point de coordonnées cartésiennes (re, im). L'entité à réaliser correspond au bloc suivant :



Les entrées/sorties sont :

Nom	Taille	Direction	Description
`clk_i`	1	in	Horloge du système
`rst_i`	1	in	Reset du système
`re_i`	12	in	Partie réelle de la coordonnée cartésienne, signée
`im_i`	12	in	Partie imaginaire de la coordonnée cartésienne, signée
`amp_o`	12	out	Amplitude calculée, non-signée
`phi_o`	11	out	Phase calculée, signée
`ready_o`	1	out	Indique que le calculateur est prêt à effectuer un nouveau calcul
`valid_i`	1	in	Indique l'arrivée d'une nouvelle coordonnée cartésienne à traiter
`ready_i`	1	in	Indique que l'entité en aval du calculateur est prête à accepter un nouveau résultat
`valid_o`	1	out	Indique que la coordonnée polaire disponible sur la sortie est valide

Le flux de données traité par le calculateur est de type "stream". Ainsi, des signaux *ready* et *valid* sont présents pour contrôler le flux de données en amont et en aval du calculateur.

Principe de fonctionnement

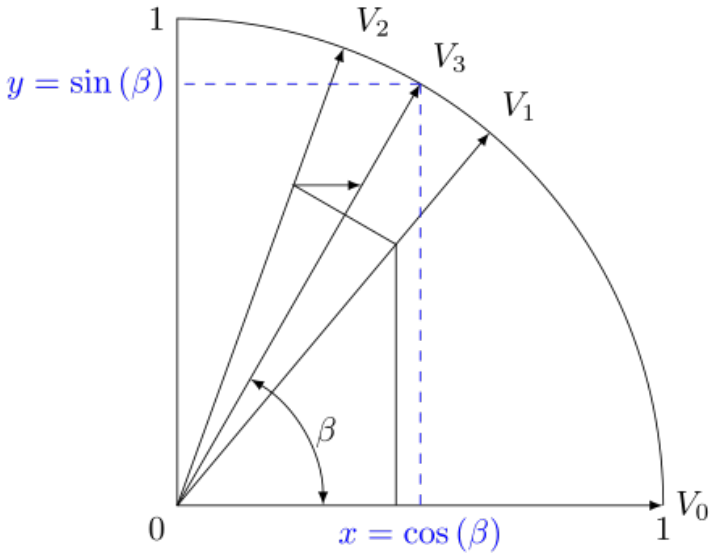
Il existe deux modes de fonctionnement de l'algorithme CORDIC : "rotation mode" permettant de transformer des coordonnées polaires en coordonnées cartésienne, et "vectoring mode" permettant de faire la transformée inverse. Le mode de travail utile pour ce laboratoire est donc "vectoring".

L'idée derrière ce mode CORDIC est de rabattre le vecteur d'entrée sur l'axe de réels par une succession de rotations par des angles de plus en plus petits. Il s'agit donc d'un algorithme de recherche itérative. A chaque itération, la nouvelle rotation appliquée est cumulée aux rotations précédentes. A la fin, le vecteur étant rabattu sur l'axe des réels, le cumul des angles de rotation est égal à la phase recherchée.

La rotation du vecteur d'entrée est mathématiquement obtenue grâce à l'opération matricielle suivante :

$$\begin{pmatrix} re_{i+1} \\ im_{i+1} \end{pmatrix} = \mathbf{R_i}\overrightarrow{v_i} = \begin{pmatrix} cos(\alpha_i) & -sin(\alpha_i) \\ sin(\alpha_i) & cos(\alpha_i) \end{pmatrix} \begin{pmatrix} re_i \\ im_i \end{pmatrix}$$

Voici un exemple de trois itérations, partent de $re_0 = 1$ et $im_0 = 0$:



La présence des fonctions trigonométriques n'est pas propice à une implémentation FPGA sans douleur. Les identités trigonométriques suivantes permettent d'obtenir une formulation plus utile de la matrice de rotation :

$$cos(\alpha_i) = \frac{1}{\sqrt{1 + tan^2(\alpha_i)}} \quad sin(\alpha_i) = \frac{tan(\alpha_i)}{\sqrt{1 + tan^2(\alpha_i)}}$$

$$\overrightarrow{R_i} = \frac{1}{\sqrt{1 + tan^2(\alpha_i)}} \begin{pmatrix} 1 & -tan(\alpha_i) \\ tan(\alpha_i) & 1 \end{pmatrix}$$

A première vue, cette réécriture ne nous avance pas car une fonction trigonométrique est toujours présente. Cependant, nous avons le loisir de choisir les angles α_i par lesquels tourne le vecteur à chaque itération. L'astuce de CORDIC consiste à se limiter à des angles de rotation dont la tangente vaut 2^{-i}. De plus, le facteur $\frac{1}{\sqrt{1+tan^2(\alpha_i)}}$ peut être omis pour obtenir :

$$\begin{pmatrix} re_{i+1} \\ im_{i+1} \end{pmatrix} = \mathbf{R_i}\overrightarrow{v_i} = \begin{pmatrix} 1 & -2^{-i} \\ 2^{-i} & 1 \end{pmatrix} \begin{pmatrix} re_i \\ im_i \end{pmatrix}$$

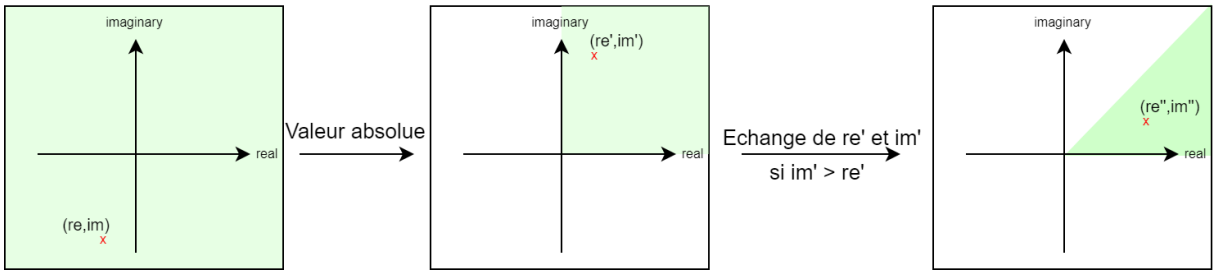
La formulation est nettement plus simple, et les seules opérations présentes sont l'addition, la soustraction et la division par une puissance de deux : un plaisir à implémenter en FPGA.

Description de l'algorithme

Etape 1 : réduction des coordonnées au premier octant

Afin de limiter le nombre d'itérations CORDIC, il est intéressant de projeter le vecteur d'entrée dans le premier octant du plan complexe. Pour ce faire, les opérations illustrées ci-dessous sont effectuées :

- Calcul de la valeur absolue de *re* et *im*. Ceci projette les coordonnées dans le premier quadrant.
- Comparaison entre *re* et *im*. Si *re* > *im* alors leurs valeurs sont échangées. Ceci projette les coordonnées dans le premier octant.



Etape 2 : itérations CORDIC

Les valeurs calculées à l'étape 1 sont utilisées en tant qu'entrées de la première itération CORDIC. Au total, votre implémentation réalisera 10 itérations. Voici les calculs à effectuer pour i allant de 1 à 10 :

- si la partie imaginaire à l'itération i est négative :
  - $re_{i+1} = re_i - im_i/2^i$
  - $im_{i+1} = im_i + re_i/2^i$
  - $phi_{i+1} = phi_i - alpha\_const_i$
- si la partie imaginaire à l'itération i est positive :
  - $re_{i+1} = re_i + im_i/2^i$
  - $im_{i+1} = im_i - re_i/2^i$
  - $phi_{i+1} = phi_i + alpha\_const_i$

D'une part, nous observons que le mécanisme décrit dans le principe de fonctionnement est respecté : seules des opérations d'addition/soustraction ou de division par une puissance de 2 sont présentes. Au fur et à mesure des itérations, l'angle phi enregistre les angles de rabattement successifs réalisés par CORDIC.
Les constantes pré-calculées $alpha\_const_i$ vous sont fournies sous forme de tableau dans le paquetage VHDL du laboratoire.

Etape 3 : projection de l'angle sur les 4 quadrants

L'angle phi ayant enregistré le cumul des rotations est disponible à la sortie de la dernière itération CORDIC. Cependant, l'angle varie uniquement au sein du premier octant. Il faut encore le transformer en tenant compte des projections ayant été faites à l'étape 1.

1. Projection sur le premier quadrant
Si les coordonnées re et im ont été échangées à l'étape 1, appliquer la correction $phi = PI/2 - phi$. Sinon laisser l'angle tel quel.

2. Projection sur les quatre quadrants
L'opération à effectuer dépend du quadrant d'origine :

- Premier quadrant : $phi = phi$
- Deuxième quadrant : $phi = PI - phi$
- Troisième quadrant : $phi = phi + PI$
- Quatrième quadrant : $phi = -phi$

Le quadrant d'origine peut aisément être déduit grâce aux signes des coordonnées non-réduites.

Les constante PI et PI/2 à utiliser sont fournies dans le paquetage du laboratoire. Ces dernières, combinées à l'usage des constantes $alpha\_const_i$ également fournies, ont été calculées de manière à produire un angle exploitant la plage binaire entière du vecteur de sortie.

Etape 4 : extraction de l'amplitude

L'algorithe CORDIC en mode "vectoring" fonctionnant par rabattement du vecteur d'entrée sur l'axe des réels, l'amplitude peut être extraite "gratuitement" en prenant tout simplement la valeur réelle de la dernière itération.

Travail à réaliser

1. Proposer trois schémas différents du calculateur CORDIC implémentées par une architecture combinatoire, pipeline et séquentielle. Choisissez une décomposition permettant de réutiliser une ou plusieurs parties du design dans les trois architectures.
2. Décrire vos trois architectures en VHDL dans des fichiers séparés. L'entité à réaliser est donnée dans le fichier *cordic.vhd*. Observez les constantes fournies dans le paquetage *cordic_pkg.vhd*.
3. Compléter le squelette du banc de test fourni et simuler vos trois architectures. Indications pour la génération des références :
   — La phase de sortie doit entièrement exploiter sa plage binaire. Ainsi, l'angle 0 doit correspondre à la représentation binaire "000000000000", l'angle PI à "011111111111" et l'angle -PI à "100000000000".

   Pour lancer le banc de test :
   — Dans le répertoire `sim`, lancez la commande `vsim`, puis tapez `do ../scripts/sim.do` depuis la console Questasim. Le script vous permettra notamment de tester les trois architectures. Observez sa construction pour savoir comment le lancer.
4. Créer un projet Quartus et compiler vos trois architectures. Observer et analyser la quantité de logique utilisée ainsi que la fréquence de fonctionnement maximale.

Travail à rendre

Une fois le travail achevé, rendez un mini rapport contenant une description de vos choix ainsi que des tests et une analyse des synthèses. Rendez également les sources, sous format électronique.