ملخص:

الهدف الرئيسي من هذا العمل هو زرع في الدارة المنطقية المبرمجة FPGA Field Programmable GateArray) طريقة لكشف عن دوائر وخطوط مستقيمة منصورة واقعية،مستخدما تقنية تحويلة أو فالدائرية او الخطية.

في المرحلة الأول يتم إنجاز خوارزمية هذه التحويلة وبرمجتها منطر فبرنامج C++Builder لتأكيد من صلاحيتها فيما بعد ،عرضت وزرعت هندسة تحت برنامج ISE.7.1i.

Résumé :

L'objectif essentiel de notre mémoire est l'implémentation sur un circuit FPGA (Field Programmable GateArray) détection des lines et des cercles partir d'images réelles en utilisant la technique de la transformée de Hough circulaire et linaire . Dans un premier temps, un algorithme a été élaboré et validé par l'outil C++builder. Par la suite, une architecture a été proposée et implémentée sous l'environnement Xilinx ISE 7.1i

Abstract :

The main objective of our work, is the implementation of line and circle detection from reel images on FPGA (Field Programmable Gate Array) circuit by using a Circular and line Hough Transform . At first, an algorithm has been elaborate and tested under and C++builder software. Next, an architecture has been proposed implemented under Xilinx ISE 7.1i environment.

الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Projet de Fin d'Études

présenté par

AMRANE ABDELKADER

&

HAMDOUDE MUSTAPHA

pour l'obtention du diplôme Master2 en Électronique option système de vision et robotique

Thème

étude et implémentation d'une méthode de détection des formes

Proposé par :M^{me} Messaoudi khadidja & M^{r}.Mamoune M

Année Universitaire 2011-2012

Dédicace

Je dédie ce modeste travail

A dieu le tout puissant.

A mes très chers parents .

A mes chers frères et sœurs .

A mon binome Abdalkader qui sans il ce travail n'aurait pas été fait .

A tous mes amis et amies de l'université SAAD DAHLAB BLIDA.

Aux familles Hamdoud, Darmouni

Et à tous ceux qui me sont chers.

Mustapha

Dédicaces

Je dédie ce modeste travail à mes chers parents qui m'ont toujours entouré de leur amour et qui n'ont jamais cessé de m'encourager à aller de l'avant et progresser dans la vie.

A ma femme qui m'a crée le cadre de travail idéal

A mon adorable petit enfant Lokman Mohamed Amine

A mon binome Mustapha qui sans il ce travail n'aurait pas été fait.

A mes beaux parents.

A mes sœurs et belles sœurs.

A mes frères et beaux frères.

A mes nièces et à mes neveux.

Abd el kader

Remerciements

Nous remercions Dieu, le tout puissant de nous avoir gardé en bonne santé, de nous avoir donné le courage et la patience pour l'accomplissement de ce travail.

Nous tenons à remercier **M^{me} K.Messaoudi** qui nous a encadré, pour sin énorme effort fourni, sa confiance, sont dévouement constant constant et son entière disponibilité.

Nous remercions notre Co-promoteur **M^{r} M .Mamoun** pour ses conseils et orientations tant précieuse qu'il nous a prodigué durant ce travail.

Nous remercions chaleureusement tous ceux qui nous ont volontairement aidé à élaborer ce mémoire.

Nos remerciements, à **M^{r}.Bessalah** directeur de CDTA, de nos avoir accepté dans son établissement.

Entre autre **M^{me} .F .Alim**, chef d'équipe **AC2**, pour nous avoir accueillis au sein de son équipe. Nous tenons à la remercier bien vivement pour ses encouragements et aussi pour sa contribution à l'amélioration judicieuse de la qualité de ce mémoire. A toutes l'équipe de la division architecture des systèmes, pour leurs aides et surtout pour leurs accueils et la bonne ambiance au sien du groupe.

Plus spécialement à **M^{r} Sofaine Seddiki , M^{r} Oussama, M^{me} linda**, **M^{r} Oualid , M^{me} nadjia** et **M^{me} anane** pour leurs disponibilité, les échanges d'idées et leurs contributions judicieuse lors de l'implémentation HARDWARE et pour nous avoir fait profiter de leur expérience et leur savoir

Table des matières

Chapitre 3 Méthodologie de conception sur FPGA.

Chapitre 4 Conception et Implémentation.

Listes des acronymes et abréviations

TH : Transformée de Hough.

THC : Transformée de Hough circulaire.

ρ*:* la distance de l'origine.

θ: l'angle entre le vecteur distance et la direction positive des X.

r : est le rayon du cercle.

RTH : Randomized Hough transformé .

THH :Transformée de Hough Hiérarchique.

THB :Transformée de Houghprobiliste .

THI : Transformée de Hough Incrémental.

DSP : Digitale Signal Processing.

FPGA : Field Programmable Gate Array.

IOB : Input Output Bloc.

CLB: Configurable Logic Bloc.

RAM : Random Access Memory

LUT :Look Up Table.

DCM: Digital Clock Managers

PSM : Programmable Switch Matrix**.**

BRAM : Block RAM.

FIFO :First In First Out.

PLL : Phase Locked Loop ou boucle à verrouillage de phase.

SRAM : Static Random Access Memory

EPROM : Erasable Programmable Read Only Memory

EEPROM : Electrically Erasable Programmable Read Only Memory

ASIC : Application-specific integrated circuit.

CORDIC : COrdiante Rotation DIgital Computer.

ISE : Integrated Software Engineering .

CE : clock Enable.

CLK : clock

MUX : Multipléxeur.

Introduction générale

Le traitement d'images digitales n'est pas un domaine nouveau en soi. Depuis les trente dernières années, des applications concrètes naissant du traitement d'images, ont trouvé une place importante dans plusieurs domaines scientifiques et domestiques. Ainsi, la domotique, la robotique et la biométrie sont des exemples de champs d'applications reposant toutes sur le traitement et l'analyse d'images. Quelques applications de la robotique arborant un système de vision exploitant le traitement d'images digitales afin d'analyser un stimulus et d'en extraire une représentation symbolique. C'est souvent le cas de robots autonomes dotés d'une intelligent artificielle. D'autres cas concrets d'application directes du traitement d'images se retrouvent aussi en biométrie ,afin de produire un modèle mathématique de l'identité d'un sujet en fonction des caractéristique physionomiques d'une partie quelconque de son corps (emprunte, iris, main ,etc…..) .

La segmentation joue un rôle prépondérant dans le traitement d'images. Cette opération a pour but de séparer différentes zones homogènes d'une image, afin d'organiser les objets en groupes dont les membres ont en commun diverses propriétés (intensité, couleur, texture, etc. ……). Elle est réalisée avant les étapes d'analyse et de prise de décision dans plusieurs processus d'analyse d'image, tels que la détection et le suivi des objets. Elle aide à localiser et à délimiter les entités présentes dans l'image. Plusieurs techniques de segmentation existent, la segmentation par régions et la segmentation par contours. La transformée de Hough (TH) est une méthode robuste de détection de contours.

Cette transformée a été proposée par P.V.C. Hough dans un brevet déposé en 1960, afin de détecter des alignements à l'aide d'un oscilloscope, et de deux caméras vidéo. Depuis les années 80, elle a trouvé des champs d'applications divers vu sa robustesse de détecter des formes paramétriques dans l'image telle que la droite, cercle, ellipse, etc.

Dans notre travail, nous allons étudier et implémenter cette technique pour la détection de deux formes dans l'image : le cercle et les droites. Ce travail peut être utilisé dans la robotique pour la détection des obstacles en temps réel. Le mémoire est organisé de la manière suivante:

Dans le premier chapitre, nous présentons la transformée de Hough. Nous commençons par définir la transformée de Hough, ensuite nous exposons le principe et la méthode de détection de droite par la TH et par analogie, nous présentons la transformée de Hough Circulaire (THC). Pour la détection des formes aléatoires, la méthode de la transformée généralisée est présenté.

Dans le deuxième chapitre, nous présentons les résultats de l'exécution du programme développé sous C++Builder pour la détection de différentes formes par la TH.

Dans le troisième chapitre, nous exposons la méthodologie de conception sur les circuits FPGA. Comme nous possédons un circuit FPGA de xilinx, notre étude s'accentuera sur ce type de circuit.

Dans le quatrième chapitre, nous exposons les résultats de l'implémentation hardware. Nous présentons les principaux problèmes de l'implémentation hardware de la TH ligne et circulaire. Nous présentons aussi les différents blocs de l'architecture développée et leurs résultats de simulation. Les résultats de synthèse et d'implémentation sont aussi présentés dans ce chapitre

Nous terminons ce mémoire par une conclusion générale et des perspectives.

Liste des figures

Chapitre 1 Transformée de Hough

1.1 Introduction

La transformée de Hough, est une technique proposée par PVC Hough dans un brevet déposé en 1960. Depuis les années 70 à ce jour, cette dernière a donné ces preuves de robustesse pour la détection des courbes paramétriques dans une image.

Plusieurs domaines utilisent la TH pour la détection des formes (droite, cercle, ellipse, etc.). Par exemple dans le domaine de la biométrie, la TH est utilisée pour la segmentation de l'iris et en robotique pour affirmer la présence des obstacles. Vu son efficacité de détection, la TH est utilisée dans des domaines plus critique tels que l'imagerie médicales (la segmentation des images médicales) et dans le domaine militaire (la segmentation des plaques d'immatriculation, etc.).

Malgré la robustesse de la TH dans la détection des courbes paramétriques, son calcul requière de larges délais de traitement et de l'espace mémoire énorme utilisé. De ce fait, plusieurs variantes de la TH ont vu le jour tels que la TH probabiliste, la TH hiérarchique, et la TH incrémentale. Ces dernières ont pour but principal :
réduire le temps de calcul afin d'appliquer la TH dans les tâches de traitement d'image en temps réels.

1.2 Principe général de la transformée de Hough

Partant du fait que l'on cherche à détecter la présence d'une certaine structure (droite, cercle, ellipse…) dans une image. La première étape, dans la transformée de Hough, consiste à rechercher les paramètres de la structure à détecter.

La seconde étape vise à trouver les points de l'image susceptibles d'appartenir à la structure (typiquement les points où le gradient dépasse un certain seuil), en chacun de ces points on va y faire passer un représentant de la structure en calculant ses paramètres. Les valeurs de ces paramètres correspondent à a un point de l'espace de paramètre. [1]

Arrive la troisième étape qui consiste à marquer la courbe représentée par les paramètres calculés comme étant une représentation possible de la structure recherchée.

Pratiquement, nous allons voter pour des paramètres. La courbe dont paramètres obtiennent le plus de suffrages est donc celle qui décrit le mieux la structure recherchée.

Pour recueillir les voix, il faut disposer d'un tableau accumulateur, la dimension de ce tableau est égale au nombre de paramètres inconnus. La taille de chaque dimension est égale à la valeur maximale du paramètre correspondant. Une fois les paramètres calculés, ceux-ci représentent une case du tableau accumulateur, on va augmenter la valeur de cette case d'une unité.

A la fin du processus, on va explorer l'accumulateur pour trouver les cases qui ont obtenu le plus des voter, leurs paramètres correspondants décrivent les occurrences de la structure recherchée.

1.3 Détection des droites

Le problème de la détection des droites dans une image est un sujet encore d'actualité dans le domaine de recherche en traitement d'image et la vision par ordinateur. [2]

La méthode intuitive qui peut détecter une droite doit tester la présence des droites formées par toutes les paires de points de l'image, ce qui est extrêmement inefficace, vu que pour tester **n** point de l'image deux par on arrivera à un nombre exagéré d'itérations au moins supérieur à $\mathbf{n^2}$, dans une image **512×512** cette méthode devient prohibitive. [3]

La transformée de Hough est une technique optimale et robuste pour détecter les droites dans image des très bruitées assez rapidement, elle ne dépend pas de la Continuité des droites. Cependant, elle fournit en résultat des droites infinies et non des segments.

1.3.1 Principe

Une droite est décrite dans le plan cartésien (x, y) par l'équation suivante :

$$y=ax+b \quad (1)$$

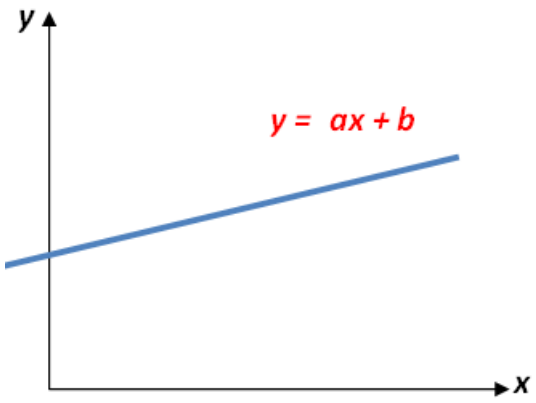


Figure 1.1 :Représentation d'un droit en espace cartésien.

Etant donné un ensemble de points contours d'un objet représenté par un ensemble de points discrets **M**font partie d'une courbe dont les paramètres **a** et **b**restent définir.

Hough puis Rosenfeld ont proposé une méthode pour détecter les droites l'aide des points du plan (x, y), son principe étant de calculer pour chaque point $\mathbf{M_i}$ de cordonnées ($x_{i,}$ $_{yi}$) de contours d'un objet, l'ensemble des paramètres **a**qui vérifient l'équation :

$$F(x_i,y_i,a,b)=0 \quad (2)$$

Avec **b** fixe.

Pour chaque point M_i (x_i, y_i) de l'image, il y a un ensemble de valeur possible des paramètres **a** et **b.** Cet ensemble forme une droite d'équation :

$$b=-axi+y_i \quad (3)$$

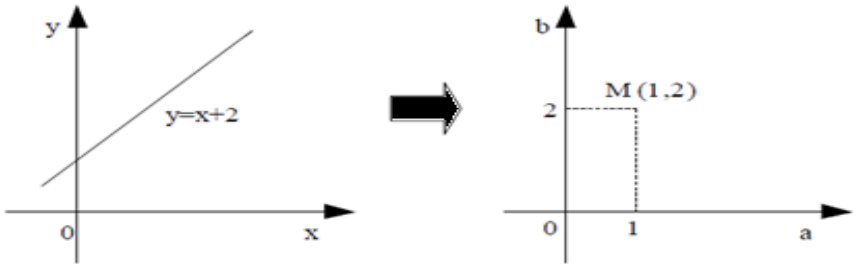


Figure1.2.a: Plan cartésien(x, y)***Figure1.2.b :*** plan des paramètres (a, b)

Figure1.2 : Transformation d'une ligne en un point dans

L'espace de paramètres.

Chaque droite dans l'espace (x, y) sera transformée en un point dans l'espace de (a, b) qui est l'espace de Hough (figure1.2).

Pour chaque point (x_i, y_i),on a un ensemble de droites qui le traverse (figure1.3).Cet droites vont être représentées par une droite passant par une droite dans l'espace (a, b).Pour trouver la droite passant par deux points **A** et **B**, on cherche pour commencer une droite dans l'espace (a, b)qui présent toutes les droites passant par le point **A**dans l'espace de (x, y),on l'appelle droite**D1**,puis, on cherche une

deuxième droite **D2**dans l'espace de(a, b**)**représentant les droites qui passent par le point **B.**L 'intersection de ces deux droites **D1** et **D2** donne le point contenant les paramètres de la droite recherchée.

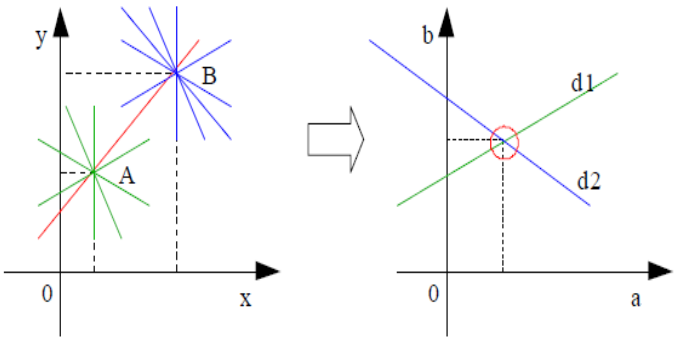


Figure1.3: rechercher la ligne qui passe par deux points en utilisant la transformée deHough.

1.3.2 La représentation polaire d'une droite

aPrincipe

En utilisant la méthode décrite précédemment on s'aperçoit rapidement que pour la plupart des points de l'espace (x, y),les pontes des droites obtenues sont très élevées, et que certains points l'ordonnée à l'origine est telle que la droite n'apparait pas forcement dans le tableau représentant l'espace (a, b). [3]

De tels problèmes deviennent gênants pour la droite verticale. Il peut

Paraitre suffisant d'agrandir la taille du buffer d'accumulation et le décentrer, mais ne sont réalité ce n'est pas suffisant (l'espace (a, b) est très inhomogène, les droite ne sont pas réparties uniformément). Ce genre de problème disparait en représentant droite de la façon suivante (figure 1.4) :

$$\boldsymbol{\rho = x\cos\theta + y\sin\theta}(4)$$

Ou $\boldsymbol{\rho}$:ladistance de l'origine

$\boldsymbol{\theta}$:L'angle entre le vecteur distance et la direction positive des X.

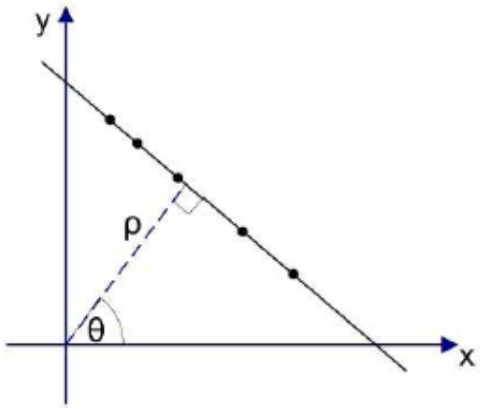


Figure 1.4: Représentation polaire d'une droite.

Une droite va être représentée dans l'espace des paramètres par un point (ρ, θ). L'ensemble des couples (ρ_i,θ_i)vérifiant l'équation :

$$\boldsymbol{\rho = x_A \cos\theta + y_A \sin\theta} (5)$$

Correspond à l'ensemble des droites qui passent par le point **A**dans l'espace image le même, l'ensemble des couple (x_i, y_i) vérifiant l'équation :

$$\boldsymbol{\rho_D = x\cos\theta + y\sin\theta} (6)$$

Correspond à l'ensemble des points qui appartiennent à la droite **(D).**Pour trouve la transformée d'un point certain (x_0, $y_{0)}$de l'espace image, nous devons résoudre l'équation :

$$\boldsymbol{\rho = x_0 \cos\theta + y_0 \sin\theta (7)}$$

$$\boldsymbol{\rho = \sqrt{x_0^2 + y_0^2}\left((x_0 / \sqrt{x_0^2 + y_0^2})\cos\theta + (y_0 / \sqrt{x_0^2 + y_0^2})\sin\theta\right)} (8)$$

En posant : $\boldsymbol{r_0 = \sqrt{x_0^2 + y_0^2}}$ et $\boldsymbol{\alpha = \arctan(y_0/x_0)}$**(9)**

On aura :

$$\rho = \sqrt{x_0^2 + y_0^2}\ (\cos\alpha_0 \cos\theta + \sin\alpha_0 \sin\theta) \quad (11)$$

$$\rho = r_0 \cos(\alpha_0 - \theta) \quad (12)$$

Pour cette représentation, un point de l'espace cartésien va donc correspondre à une sinusoïde dans l'espace (ρ, θ).

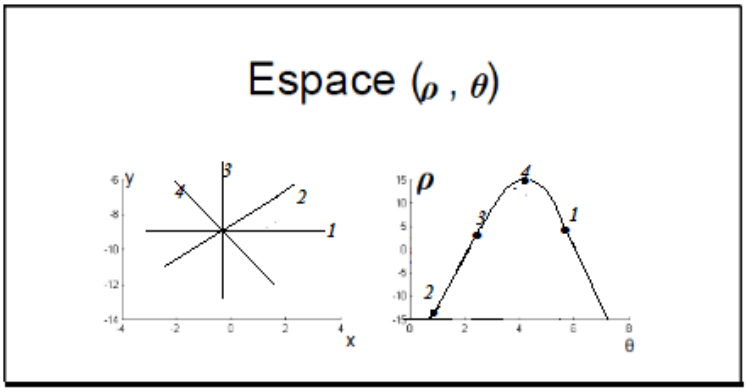


Figure 1.5: La transformée d'un point dans l'espace (ρ, θ).

Notre but est de rechercher le point d'intersection des sinusoïdes.

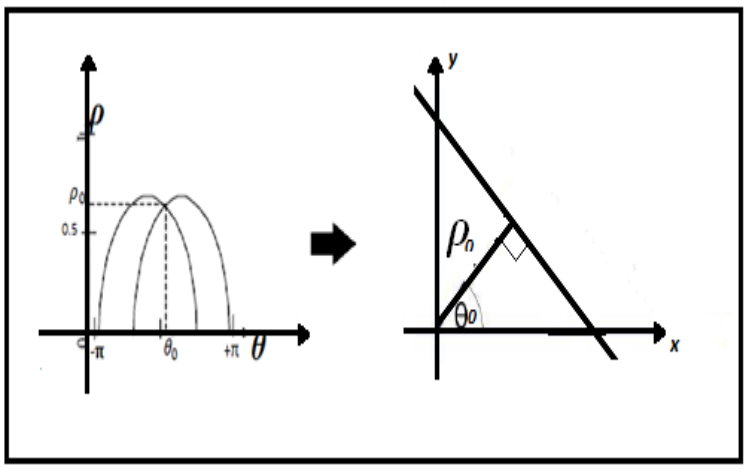


Figure 1.6 : Transformée inverse d'une droite

L'avantage principal de l'utilisation des (ρ, θ) est que la quantification sera réalisable**.** [4]

bPropriétés de la TH basée sur un paramétrage polaire

De l'étude précédente, il est facile de tirer quelques propriétés de la TH appliquée aux droites en utilisant les paramètres polaires [3]

- ✓ un point de l'espace image correspond à une sinusoïde dans l'espace paramètres.
- ✓ un point de l'espace paramètres correspond à une droite dans l'espace image.
- ✓ les points appartenant à la même droite dans l'espace image correspondent à des sinusoïdes qui s'interceptent dans un seul point dans l'espace des paramètres.
- ✓ les points appartenant à la même sinusoïde dans l'espace des paramètres correspondent à des droites qui passent par un même point dans l'espace image.

cAlgorithme de détection des droites

- **Trouver les contours de l'image originale**
- **Crée un tableau bidimensionnel H (ρ, Θ), discrétiser H en utilisant des pas ερ, εΘ Donnant une résolution optimale.**
- **Initialiser : H (ρ, Θ)=0**
- **Pour chaque point de contour I(x, y) dépassant un certain seuil**

Faire :

Θϵ [0, π] :

- évaluer ρ = x cosΘ + y sinΘ

-Incrémenter H (ρ, Θ)=0

- trouver les valeurs de H (ρ, Θ)=0 qui dépassent sur certain seuil S.

- les paramètres de la droite détectée dans l'image sont donnés par les ρ et θ trouvés

ces H (ρ, Θ)>S.

dDimensions des paramètres ρ et θ

d.1 Dimension de θ

Pour un angle **θ**qui varie de **0 à 2π**, toutes les droites s'expriment avec un **ρ** positif. Toutefois, nous remarquons que les droites avec **θ**appartenant à l'intervalle **[π ,2π]** peuvent être vues comme des droites à **ρ**négatif.

En effet, si nous considérons la droite **Q2** (figure 1.7) avec les paramètres polaires (ρ_2,θ_2), avec :

$$\theta_2 = \pi + \theta_1$$

On a : $\rho_2 = x \cos \theta_2 + y \sin \theta_2$

$$\rho_2 = x \cos (\pi + \theta_1) + y \sin (\pi + \theta_1)$$

$$\rho_2 = -(x \cos \theta_1 + y \sin \theta_1)$$

D'où : $\rho_2 = -\rho_1$

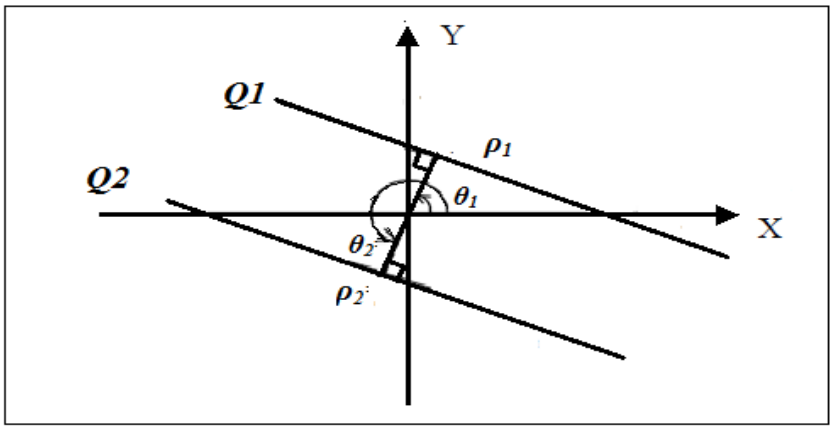


Figure 1.7:Paramètres polaires de deux droites opposées**.**

L'intervalle de **θ**peut donc être réduit de moitié, c'est-à-dire le champ de **θ**sera**[0, π [,** et pour chaque valeur **θ**appartenant à cet intervalle les droites s'expriment avec un **ρ**qui peut être négatif ou positif [1].

d.2 Dimension de ρ

Le champ de dimension de **ρ**est défini selon la taille d'image. Par exemple, pour une image carrée **N×N**(figure 1.7) le champ de dimension de **ρ**est **[0, N$\sqrt{2}$],**si l'origine de coordonnées est le coin inférieur gauche de l'image.

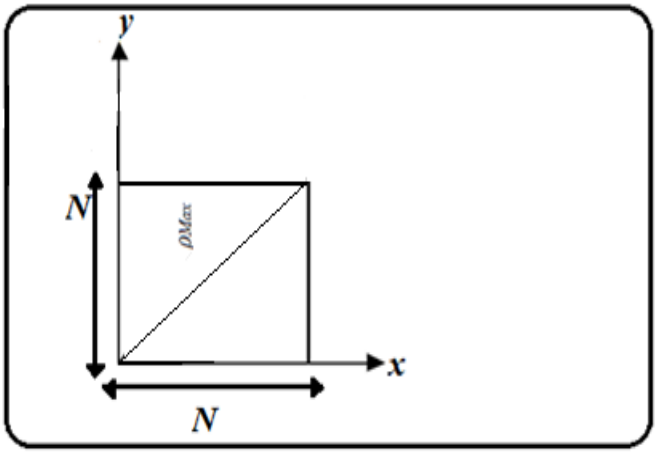


Figure 1.8: Le champ de dimension de ρ dans une imageN×N

Si nous déplaçons l'origine **0** du repère du plan (x y)au centre de l'image, puis examinons les valeurs **ρ_{min}**et**ρ_{max}**de la dimension de **ρ :**

- ✓ **ρ_{max}** est la plus grande distance positive de la droite par rapport à l'origine**0**. donc droite est représentée par la droite **Q1**dans la figure (1.9) **ρ_{max}= N /$\sqrt{2}$.**
- ✓ **ρ_{min}** est la plus petite distance négative de la droite par rapport à l'origine **0**. Elle est représentée par la droite **Q2** et **ρ_{min} = - N /$\sqrt{2}$.**

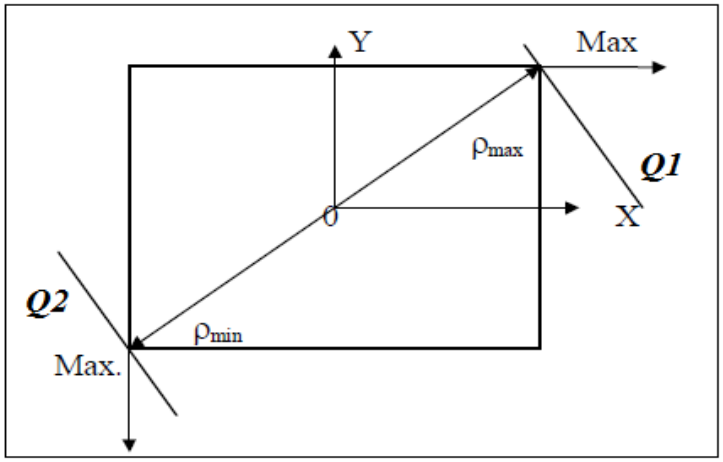


Figure 1.9: Le champ de dimension de **ρ**

1.4 Principe de détection des cercles

La transformée de Hough peut être aussi appliquée à sur la forme paramétriques tels que les cercles. Dans ce cas, le principe de la TH restera le même, mais la dimension du plan de Hough changera suivant le nombre de paramètres de l'équation analytique qui définit la forme paramétrique à détecter.

1.4.1 Détection de cercle

L'équation du cercle s'écrit comme suit :

$$(X-a)^2+(y-b)^2=r^2 \quad (13)$$

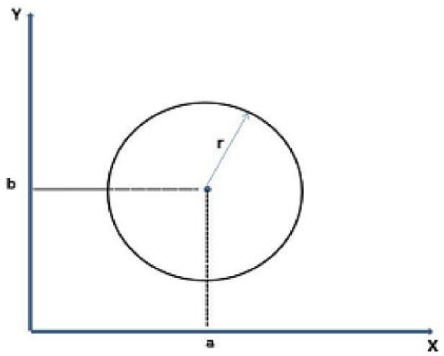


Figure 1.10: Représentation d'un cercle dans l'espace cartésien

Avec :

x et **y** sont les coordonnées d'un point sur le cercle dans le plan (x y).

r *est* le rayon du cercle.

aet **b** sont les coordonnées du centre du cercle(Figure 1.11).

Pour détecter un cercle, on doit déterminer trois paramètres; c'est-à-dire **3**dimensions (par rapport à**2**dimensions pour une droite) ce qui complique l'utilisation de la TH. Pour remédier à ce problème, on doit ce faire le traitement en deux étapes: Trouver d'abord tous les centres des cercles à **r**variant, puis trouver le rayon du cercle à détecter.

On considère l'espace de Hough (a, b, r) qui est dans ce cas un espace à trois paramètres, chaque point de coordonnées (x, y) de l'image correspond à un cône dans l'espace (a, b, r), et pour un rayon fixe, il correspond à un cercle dans l'espace (a, b) (Figure 1.12). Le principe de l'algorithme est devisé le traitement en deux :

- ✓ calculer **a**et **b** (les coordonnées du centre de tous les cercles) pour chaque**r**.
- ✓ tracer ces cercles dans l'espace deHough correspondant aux points de l'image. Lorsque tous les cercles se coupent en un même point, on a trouvé alors le rayon du cercle détecté et ces coordonnées sont(a, b).

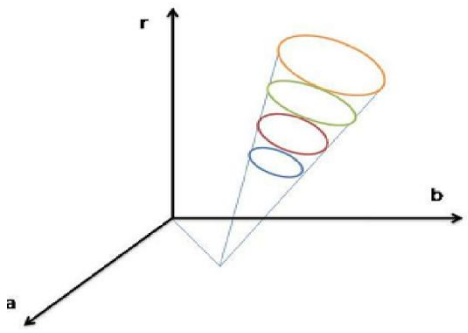


***Figure 1.11*:**plande Hough h (a, b, r)

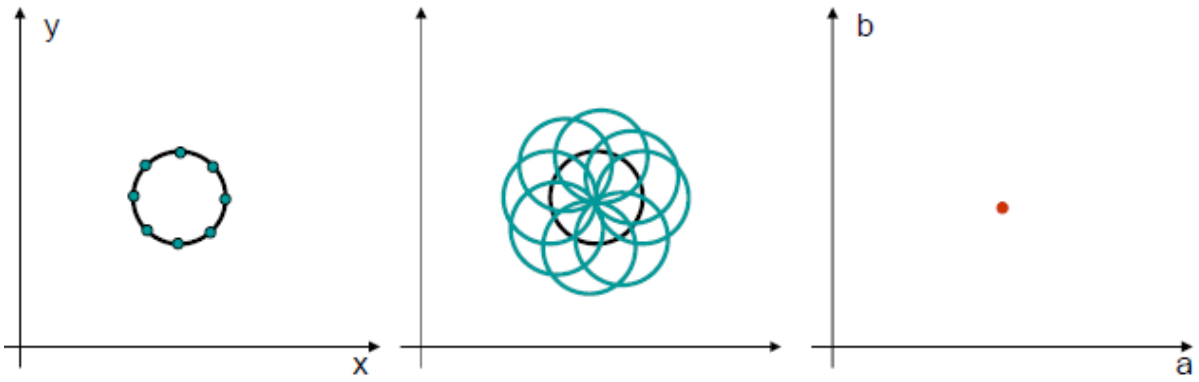


Figure 1.12 :plan de Hough avec**r** fixe.

1.4.2 Propriétés de la transformée de Houghcirculaire

- Chaque points dans l'espace cartésien (x y) correspond à une surface conique dans le plan des paramètres h (a, b, r).
- Un point dans le plan des paramètres h (a, b, r)correspond à un cercle dans le plan cartésien(x y).
- Les points appartenant au même cône du plan des paramètres correspondent à des cercles du même point du plan cartésien [6].
- Intersection des cercles dans l'espace (a, b) représenté les caractéristiques du cercle à détecter.

1.4.3 Algorithme de détection du cercle

- **Binaires et application une détection de contour.**
- **Initialise l'accumulateur à zéro.**
  - **Pour chaque point de contour de l'image incrémenter accumulateur si le point vérifier l'équation :**

$$(x-a)^2+(y-b)^2=r^2$$

- **Toutes les valeurs max dans l'accumulateur correspond aux centres des cercles.**
- **Pour chaque point de contour de l'image vote pour l'ensemble des centres des cercles de rayant r pouvant passer sur le centre.**
- **La valeur maximale de l'espace paramètre (a,b,r) représente les coordonnées de cercles détectés**
- **Lorsque tous les cercles se coupent en un même point, on a trouvé le rayon détecté.**

1.5 Différents types de la transformée de Hough

1.5.1 Transformée de Houghprobiliste (THB)

Une des faiblesses reconnues de la transformée de la Hough présenté ci –dessus est son large délai traitement, dans le cas où le nombre des points contours est élevé. Pour y remédier, une optimisation consiste à un tirage aléatoire des pointes contours. Cette approche originale, mise à jour par Lei Xu, Oja et Kultanen [7], et développée par la suite par Kàlviàinen et Hirvonen [13], est nommée Transformée de Hough

Probabiliste (RandomizedHough transformé (RTH)). Cependant, les résultats des travaux de Kiryati et al. [12] incitent à utiliser une proportion au moins comprise entre 10%et 20% des points contour possibles de l'image, pour produire dans l'espace de Hough des accumulations aussi significatives que celles obtenues par la TH.

1.5.2 Transformée de Hough Hiérarchique (THH)

Davis a proposé que l'on 'emboîte' les TH afin de détecter des structures de plus complexes[9] .L'idée de base d'appliquer la TH ,non pas à partir de points caractéristique de l'image, mais à partir d'objets plus structurés (lignes ,segments, courbes ,etc. ….) qui auront, bien sûr ,été détectés auparavant par la TH On est ainsi amené à rechercher dans l'espace des paramètres de la processus peut évidemment être itéré, pourvu que l'on sache, à chaque étape,dégager une para-métrisation satisfaisante.

1.5.3 Transformée de Hough Incrémental (THI)

L'utilisation de la TH pour des applications en temps réel nécessite son implémentation hardware. Les problèmes de cette dernière, dans le cas de la TH pour détection de droite, se résument à la complexité du calcul nécessaire pour générer toutes les distances **ρ** correspondant à toutes les valeurs **θ** pour chaque point contour de l'image et l'espace mémoire occupée. Comme le temps de calcul de la TH est proportionnel au nombre de la taille des ρ généré et l'espace mémoire dépends de la taille des multiplieurs et tables de transferts nécessaires pour le calcul des produits des coordonnées du point contour par les fonctions trigonométriques .C'est ainsi que M.H.Koshimizu et M.Numada ont proposée l'expression de la IHT [12] qui réduit le nombre des fonctions trigonométriques nécessaire pour le calcul de la TH. En ce basant sur les approximations sur les fonctions trigonométriques S.Tagzout et al [11]ont proposé une autre expression de la THI plus rapide que celle de proposé dans [10]vu qu'elle génère deux valeur de**ρ** à chaque itération. Les approximations sur les valeurs des fonctions trigonométriques produisent des erreurs de calcul après un certain nombre de calcul. Ces erreurs sont corrigées par les valeurs calculées par l'équation usuelle de la SHT.

En utilisant le même principe o.djkoune et al. Ont proposé dans [11], une équation plus généralisée de la THI qui augmente le nombre **ρ**généré par itération et ne pas fait pas appel à l'équation usuelle de la TH vu que les erreurs sont insignifiantes.

1.6 Lesavantages et les inconvénients de la transformée de Hough

1.6.1 Lesavantage

- On peut détecter importe quelle forme paramétriques même s'il est partiellement caché ou même et très robuste.
- On peut détecter une forme est très bruite

1.6.2Les inconvénients

- La transformée deHough nécessite le choix d'une méthode de détection de contours (Sobel, Canny, Deriche, etc…), et selon la méthode utilisée l'ensemble des points de contours ne sera pas le même. En effet, il est possible que des points critiques de l'objet à détecter puissent ne pas être considérés comme points de contours et n'interviennent donc pas dans le processus de vote.
- temps de calcul croît beaucoup trop lorsque le nombre de paramètres à traiter augmente, ou à cause du nombre de points à traiter.
- La TH est gourmande en espace de mémoire.

1.7 Conclusion

La transformée de Hough est un outil sûr et réputé pour sa capacité à détecter importe quel forme paramétrique droit, cercle, ellipse …….Ainsi, son utilisation dans sa forme théorique la plus simple est rarement envisagée, pour des causes de temps de calcul et de mémoire nécessaire.

De plus, elle doit être liée à un prétraitement efficace, ainsi qu'à un post-traitement intelligent et adapté (par exemple si on souhaite connaître les limites des segments).

Cependant il existe plusieurs variantes qui tâchent de résoudre ces problèmes afin d'optimiser les temps de calcul ou la mémoire nécessaire au calcul. Malgré ces défauts, elle reste une méthode exhaustive et, pour peu qu'on choisisse une discrétisation des paramètres assez fine, elle offre des résultats très précis.

Chapitre 2 Implémentation software

2.1 Introduction

Nous allons présenter dans ce chapitre, les étapes de traitement d'image nécessaires avant l'exécution de l'algorithme de la TH. Nous présentons aussi les tests et les simulations soft faites à l'aide de l'outil C++Builder. Nous avons choisi comme application de la TH : la détection des plusieurs formes. De ce fait, nous allons donner un aperçu sur cette application avant d'entamer la description des prétraitements**.** [24]

2.2 Prétraitement

En amont de la segmentation on retrouve l'étape du prétraitement. C'est une étape qui a pour but de faciliter la segmentation en accentuant les ressemblances des pixels d'une même région, ou les dissemblances entre les pixels appartenant à des régions distinctes. Dans ce qui suit nous allons présenter quelques méthodes de prétraitement.

2.2.1 Réduction de bruit

La diminution du bruit a pour but de réduire l'amplitude des variations d'intensités à l'intérieur d'une même région sans porter atteinte aux zones de transitions qui définissent les frontières entre les régions.

La réduction du bruit est souvent réalisée par une opération de filtrage, qui permet en même temps d'améliorer la qualité de l'image.

Le principe du filtrage est l'application d'une opération au niveau du pixel élémentaire en tenant compte de son environnement (pixels voisins).

C’est une opération qui consiste à déplacer un filtre (masque), qui est une matrice de dimension impaire par exemple 3*3, 5*5, sur l’image chaque pixel et de remplacer la valeur de ce dernier par le résultat de l’opération effectuée sur ses voisins. A cet effet ; plusieurs filtres ont été définis :

- Les filtres linéaires.
- Les filtres non linéaires.

a Filtres linéaires

Le filtre linéaire fait en sorte que chaque pixel de l’image soit remplacé par une valeur obtenue à partir de certaines opérations linéaires appliquées sur ses voisins. On utilise souvent des filtres :

- Les filtres « Passe-haut » qui appliquent une opération de dérivée sur l’image.
- Les filtres « Passe-bas » qui réalisent un lissage

a.1 Filtres passe-haut

Un filtre « passe haut » est un filtre qui favorise les hautes fréquences, d’où l’appellation de « passe-haut », comme les détails. De ce fait, il améliore le contraste. Un filtre « passe haut » est caractérisé par un masque comportant des valeurs positives et négatives autour du pixel central.

a.2 Filtres passe-bas

Les filtres « passe bas » agissent en sens inverse des filtres « passe haut » et le résultat est, un adoucissement des détails, ainsi qu’une réduction du bruit. Le filtre passe-bas se débarrasse des signaux ayant une haute fréquence et qui sont caractérisés par de grandes variations des niveaux de gris par rapport aux pixels voisins. Il est appelé aussi filtre de lissage car il adoucit considérablement les contours de l’image. Parmi les filtres passe-bas on peut citer les filtres :

- Filtre moyen
- Filtre gaussien

b Filtres non linéaires

Parmi les filtres non linéaires nous allons présenter le filtre médian donnant de bons résultats en imagerie.

b.1 Filtre médian

Le filtre médian est utilisé pour atténuer des pixels isolés, c'est-à-dire ayant une valeur très différente, de leur voisinage. Ce filtre est efficace sur des images

dégradées par une source de bruit impulsionnel, c'est-à-dire une source qui permet à quelques pixels de l'image de prendre des valeurs complètement aléatoires. Le principe de ce filtre est de :

- Considérer une fenêtre centrée sur un pixel ;
- Trier les pixels de cette fenêtre selon un ordre croissant de leur niveau de gris ;
- Affecter au pixel central la valeur médiane. La valeur médiane est celle qui sépare une séquence triée des valeurs des niveaux de gris en deux parties égales.

Cependant ce filtre possède un inconvénient car il affecte la géométrie des régions de l'image, de ce fait les zones comportant des angles aigus ont tendance à voir leurs angles arrondis par le filtrage.

b.2 Les filtres morphologiques

La morphologie mathématique a comme principe de base de comparer les objets d'une image **A** à un certain objet **B** pris comme référence (forme géométrique appelée élément structurant) [18]. Le filtre morphologique est utilisé pour éliminer les pixels isolés considérés comme un bruit .Les opérations de base est sont la dilatation et l'érosion, à partir desquelles on peut construire des outils plus avancés, tels que l'ouverture et la fermeture**.** [18]

Les différentes opérations de la morphologie mathématique sont la dilatation, l'érosion, l'ouverture, et la fermeture. Elles sont définies comme suit :

- ***Dilatation :*** Concerne les points noirs isolés au milieu des parties blanches. En pratique, ceci est réalisé en faisant passer sur l'image une fenêtre de taille fixe et en effectuant pour chaque pixel de l'image un OU logique des pixels formant la fenêtre, à l'exception du pixel traité (pixel central).[18].
- ***Erosion*** *:* L'érosion est l'opération duale de la dilatation. Les points blancs d'une image sont noyés par les zones noires qui se trouvent autour. Pratiquement, on effectue un *ET* logique entre les pixels contenus dans la fenêtre utilisée, sauf le pixel central. [18].
- ***Ouverture****:* L'ouverture est constituée par une opération d'érosion suivie d'une dilatation.
- ***Fermeture****:* La fermeture est constituée par une opération de dilatation suivie d'une érosion.

Sans toucher aux zones homogènes de l'image. Ainsi, on limite le risque de fusionner deux régions différentes.

2.2.2 Modification d'histogramme

Dans cette approche, on effectue une transformation d'intensité de chaque pixel. Cette transformation est choisie croissante de façon à conserver les liens relatifs entre les régions (une région claire sur un fond sombre apparaîtra plus claire que ce fond dans l'image transformée). Les méthodes de modification d'histogramme n'affectent pas la forme des régions, elles en modifient uniquement l'apparence visuelle. [19]

a Expansion de la dynamique

L'expansion de la dynamique permet de distribuer les pixels d'une manière uniforme selon une échelle de niveaux de gris. Si une image I possède des niveaux de gris entre Gmin et Gmax alors, elle deviendra I' comme suit:

$$\mathbf{I'(x,y)} = \frac{\mathbf{I(x,y)} - \mathbf{G_{min}}}{\mathbf{G_{max}} - \mathbf{G_{min}}} \qquad (1)$$

b Egalisation d'histogramme

Elle a pour but de rendre le plus plat possible l'histogramme des niveaux de gris dans l'image. Le niveau de gris de chaque pixel est calculé comme suit: [19]

$$\mathbf{G'} = \frac{\mathbf{255}}{\mathbf{nombre\_pixel\_total}} \mathbf{Histocumul\acute{e}(G)} \qquad (2)$$

Histocumulé (**G**) : le nombre total de pixels ayant le niveau de gris inférieur ou égal à **G**

2.2.3 Rehaussement du contraste

Le rehaussement de contraste a pour but de supprimer l'effet de flou, qui est caractérisé par une transition s'étalant sur plusieurs pixels entre deux régions différentes, dû par exemple à un bougé lors de la prise de vue ou de défocalisation de la caméra. Le principe consiste à essayer de restaurer les contours et de diminuer l'étendue de la zone de transition.

2.3 Segmentation

La segmentation, est la division d'une unité anatomique en plusieurs éléments [20].En traitement d'image, c'est le processus de partitionnement d'une image numérique en plusieurs régions homogènes ou ensembles de pixels. Mathématiquement, nous pouvons la modéliser [21] de la façon suivante:

Soit **I** une image composée d'un ensemble de pixels, la segmentation est le partitionnement de l'image *I* en une ou plusieurs régions $\mathbf{R_1, R_2,\ldots, R_n,}$ telle que:

$$\mathbf{I} = \cup_i \mathbf{R_i} \qquad (3)$$

$$\mathbf{R_i} \neq \forall_i \text{ et } \mathbf{R_i} \cap \mathbf{R_j} = \emptyset \; \forall_{i,j} \mathbf{i} \neq \mathbf{j} \qquad (4)$$

2.3.1 Objectif de la segmentation

L'objectif de la segmentation est de permettre l'exploitation du contenu de l'image pour l'interprétation : aide au diagnostic en imagerie aérienne, satellitaire et médicale. Pour une éventuelle localisation et reconnaissance (vidéo surveillance, contrôle robotique) une mesure des évolutions (contrôle de qualité, suivi thérapeutique, …etc.) La segmentation d'images est au cœur de nombreux problèmes en imagerie médicale puisque bien souvent elle constitue la première étape d'un véritable flux de traitements d'images.

2.3.2 Différentes approches de segmentation de l'image

Il existe plusieurs approches de segmentation de l'image :

- Approche contours
- Approche régions
- Approches mixtes régions / contours

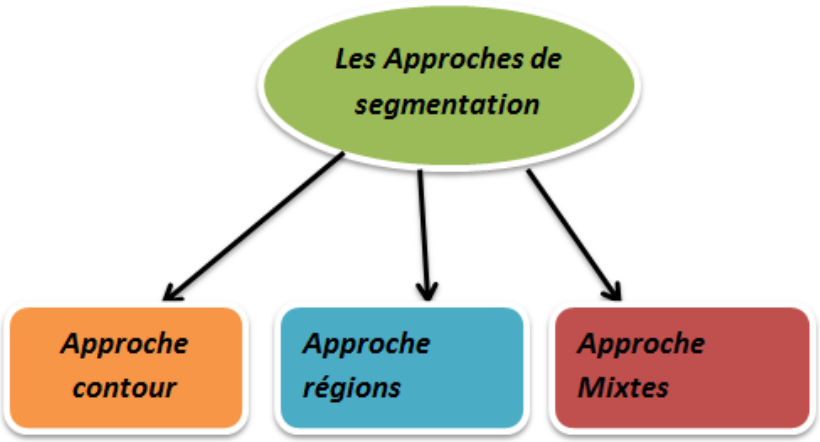


Figure 2.1 : Schéma représentant les approches de la segmentation

a Approche contours

Dans les techniques basées sur la détection de contours, la segmentation est réalisée par la recherche des frontières délimitant les régions dans l'image. Ceci est généralement réalisé par l'application de masque sur l'image afin de détecter les changements locaux dans l'intensité des pixels. Il existe différentes approches peuvent être réparties selon la manière d'estimer les dérivées de la fonction d'intensité [22]:

- **Filtrage par différences finies:** comme l'opérateur de gradient, l'opérateur Laplacien, Sobel, etc.
- **Filtrage optimal:** comme le filtre de Canny, filtre de Canny -Deriche, etc.

b Approche régions

La segmentation par régions est basée sur la définition formelle de la segmentation d'une image relative à un prédicat d'homogénéité. Etant donné un prédicat **P** pouvant s'appliquer aux attributs des régions, une segmentation est un partitionnement de l'image I en régions $\mathbf{R_i}$ telles que [21]:

$$\mathbf{R_i} \neq \emptyset \ \forall_I \qquad (5)$$

$$\mathbf{R_i} \cap \mathbf{R_j} = \emptyset \qquad (6)$$

$$\mathbf{I} = \cup_i \mathbf{R_i} \qquad (7)$$

$$\mathbf{R_i} \text{ est connexe } \forall_i \qquad (8)$$

$$P(R_i) = \text{vrai } \forall_i \quad (9)$$

$$P(R_i \cup R_j) = \text{faux si } i \neq j \text{ et } R_i \text{ et } R_j \text{ sont adjacentes} \quad (10)$$

Les méthodes de l'approche région cherchent à regrouper directement des pixels ayant une propriété commune; l'ensemble des regroupements de pixels définit à la fin une segmentation de l'image.

c Approches mixtes régions / contours

Comme nous l'avons vu précédemment, les approches régions et contours présentent toutes les deux des avantages et des inconvénients. Les chercheurs ont essayé de trouver un compromis entre les deux, et ont donné naissance à ce qu'on appelle la segmentation coopérative.

Cette approche combine l'approche par régions et l'approche par contour. Elle exploite conjointement l'information contour et l'information région afin de pallier les insuffisances de chacune des deux approches. La coopération peut être associative ou séquentielle.

2.3.3 Description de La chaine de traitement

Notre chaine de traitement comporte les étapes suivantes

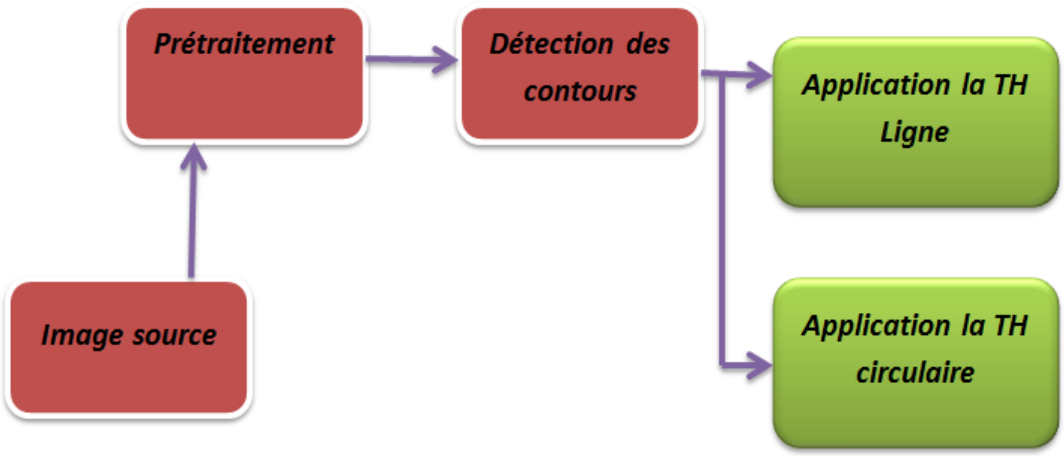


Figure 2.2 : Chaine de traitement

a Prétraitement.

b Détection des contours.

c Application de la transformée de hough ligne.

d Application de la transformée de hough circulaire

a Prétraitement

Les traitements préliminaires que nous avons effectués peuvent être résumés comme suit ***:***[23]

- La première étape consiste à réduire le bruit occasionné par le capteur et toute l'électronique de transfert associée. Pour ce faire, on appliquera un filtre GAUSSIAN sur l'image.
- La seconde étape du prétraitement est nécessaire afin de convertir l'image adoucit les composantes RGB en image avec des niveaux de gris. Cette conversion permet de tirer profit de la rapidité des différents algorithmes subséquents, car ils ne sont pas basés sur la couleur, mais bien sur les niveaux de gris. Puisque un pixel de teinte grise quelconque possède ces trois composantes RGB de même valeur, la conversion devrait plutôt se faire en utilisant la formule suivante:

$$\mathbf{NGris = 0.299R + 0.587G + 0.114B} \qquad (11)$$

Nous choisissons les images pour appliquer le prétraitement sur toutes les images comme exemple

Image 1 filtre gaussien niveau de gris

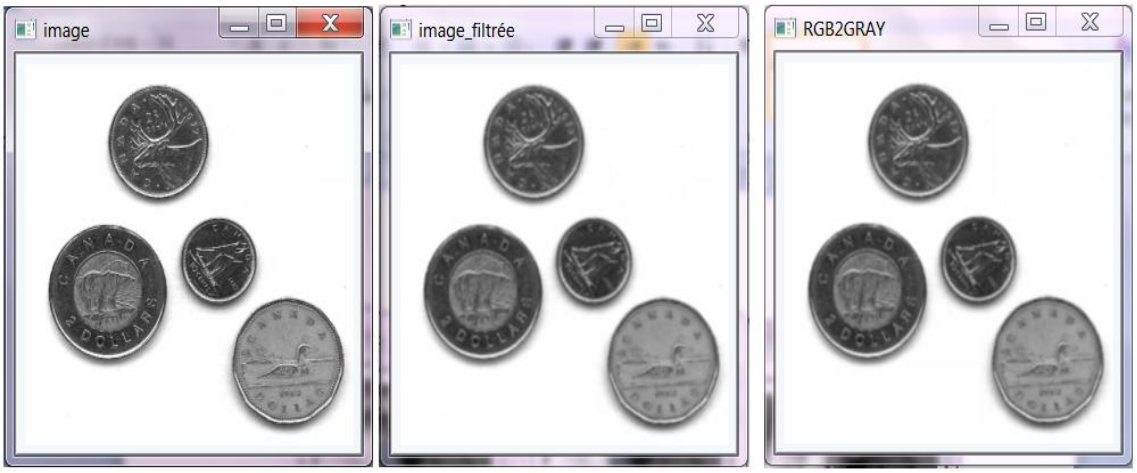


Image 2 filtre gaussien niveau de gris

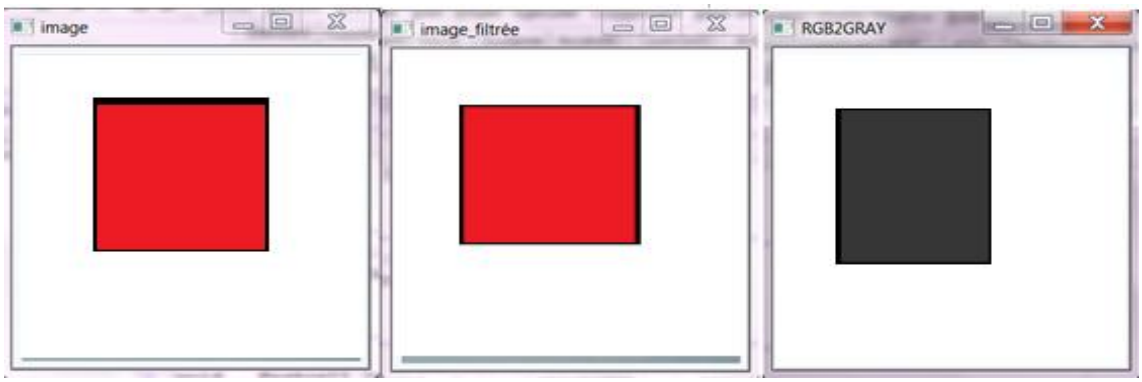


Image 3 filtre gaussien niveau de gris

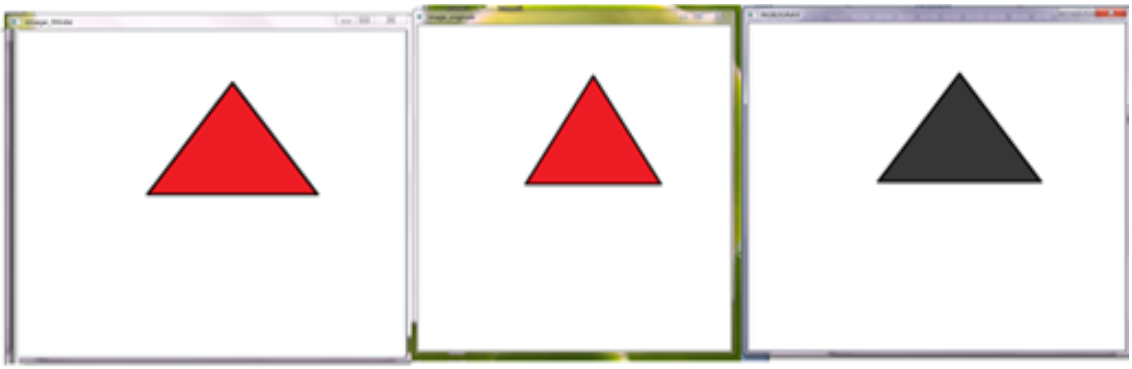


Image 4 filtre gaussien niveau de gris

Figure 2.3 : application les prétraitements sur différents images

b Détection de contours

La segmentation par la transformée de Hough nécessite le choix d'une méthode de détection de contour. Cette étape est primordiale car elle influe sur les formes recherchées.

Dans cette section, nous présentons les tests effectués sur l'algorithme de détection de contours par la méthode du filtre optimal. Ces différents tests ont été réalisés sur plusieurs images Les résultats obtenus sont présentés dans une perspective de comparaison visuelle.

b.1 Application du filtre de Canny

Le filtre de Canny fut le premier à spécifier à la fois la qualité de la détection et la précision de la localisation du contour sous une forme mathématique. [23]. L'algorithme du filtre de Canny est présenté comme suit:

1. Filtrer l'image par un filtre simple comme le filtre Gaussien par exemple.

2. Calculer l'image horizontale et verticale en utilisant un opérateur de convolution comme Sobel.

3. Calculer la magnitude du gradient **A** de l'image en prenant la somme des carrés des images obtenues en **2.**

$$\mathbf{A}=\sqrt{(\boldsymbol{GradX})^2+(\boldsymbol{GradY})^2} \qquad (12)$$

4. Calculer la direction du gradient qui est l'arc-tangentedu rapport entre les deux images horizontale et verticale.

$$\boldsymbol{\alpha = \mathbf{arctang}\left[\frac{\mathbf{GradX}}{\mathbf{GradY}}\right]} \qquad (13)$$

5. Supprimer les non-maximas du gradient : on met à **0** la valeur de tout pixel qui a un voisin de même direction de gradient et et de module supérieur.

6. On applique un seuillage par hystérésis pour mettre en évidence les points contours.

Les points dont le gradient est supérieur au seuil maximum est un point contour, ainsi que les points dont le gradient est entre les deux seuils et ayant un voisin déjà considéré comme contour.

b.2 le résultat obtenu

Pour montrer l'efficacité de l'algorithme donnée ci-dessus, on utilise une plusieurs image de test, les résultats obtenus sont illustrés dans la figure (2 .4)

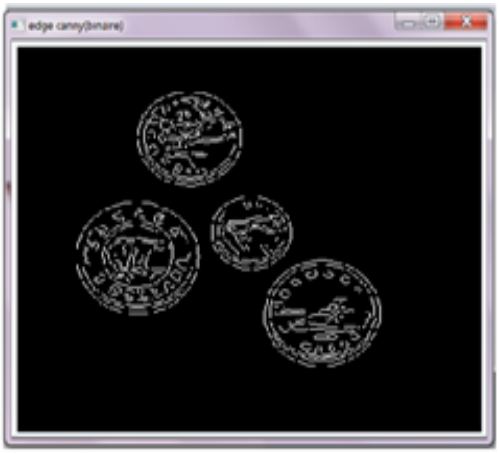


Image 1

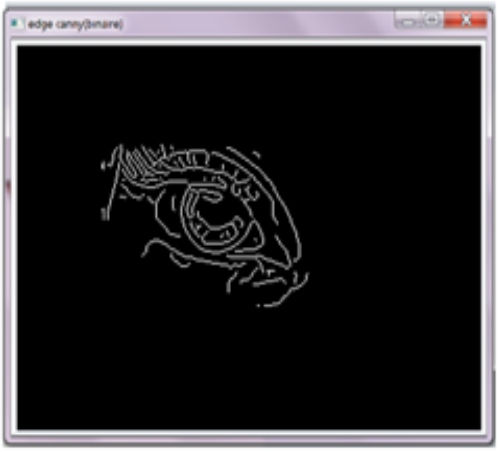
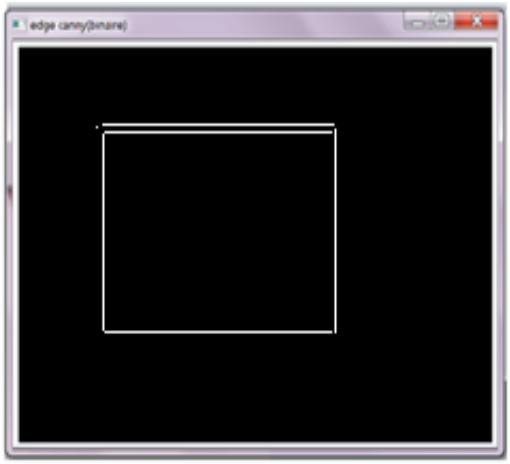


Image 2



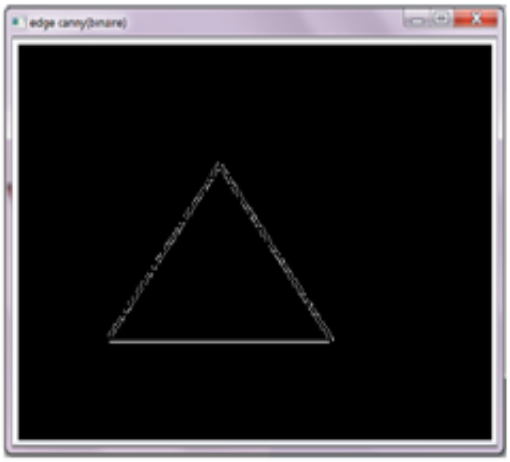


Image 3 Image 4

Figure 2.4 : application de filtre canny avec seuil

Nous avons applique l'algorithme de canny avec seuil sur plusieurs images en variant la valeur de cette seuil, nous constatons que la valeur de seuil la plus adapté pour avoir une bonne détection de contour est égale à **0.15.**

c Application de la transformée de Hough linéaire

Une fois l'image contour générée par le filtre de canny, on va appliquer la TH ligne pour détecter les formes linéaire, on fait varier la valeur du $\boldsymbol{\theta}$ et la valeur du seuil.

On utilisera la représentation en coordonnées polaires pour recherche dans l'image, les droites ou les forme linéaire recherchés étant décrits par l'équation des droites :

$$\boldsymbol{\rho = x\cos\theta + y\sin\theta} \qquad (14)$$

On applique l'algorithme de la droite sur les images contour, les résultats des tests sont illustrés dans les figures suivantes :

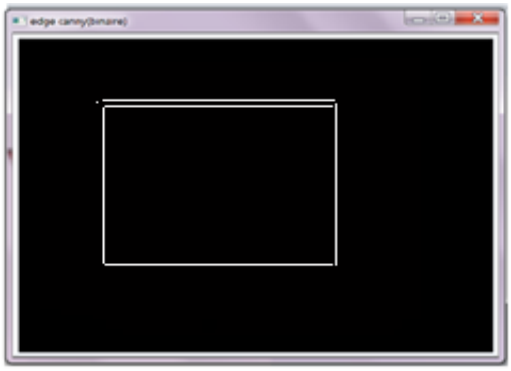
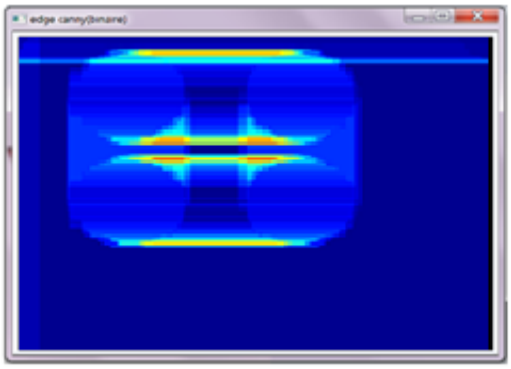
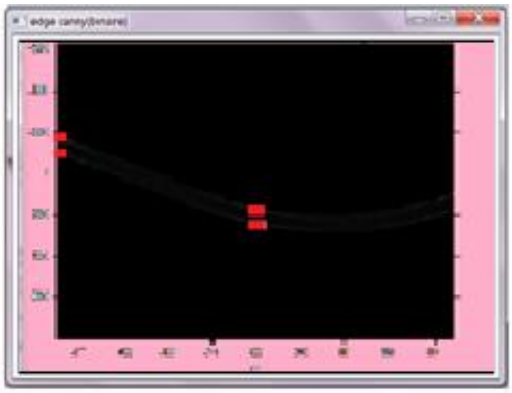


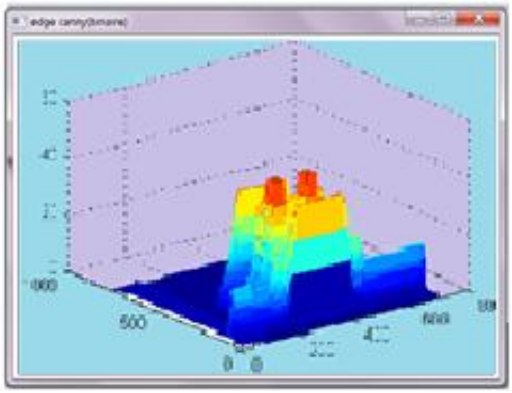
Image contour



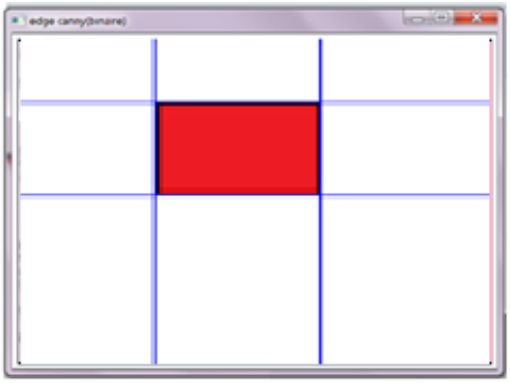
plans de hough 2D



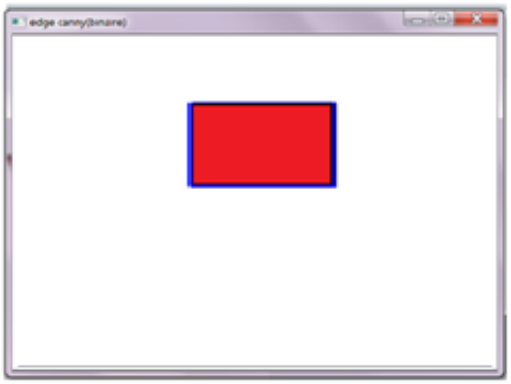
Les quatre pics



le plan de hough 3D



Détecter quatre linges par la TH linge



la forme à détecter

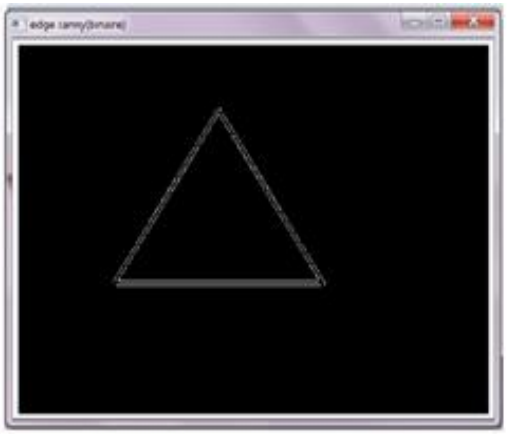
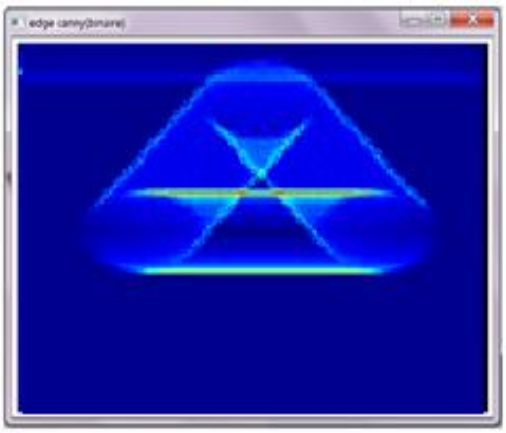
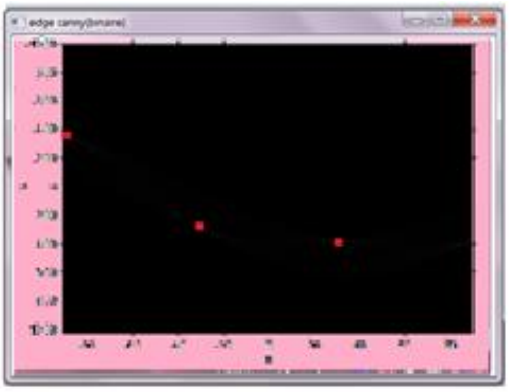


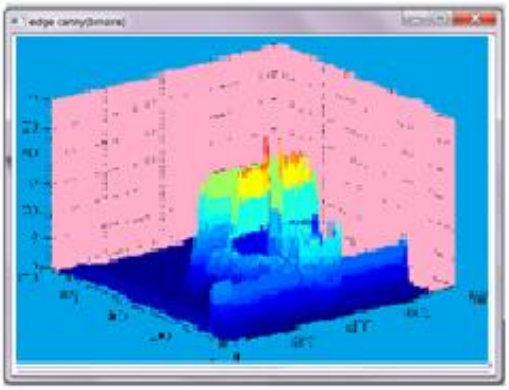
Image contour



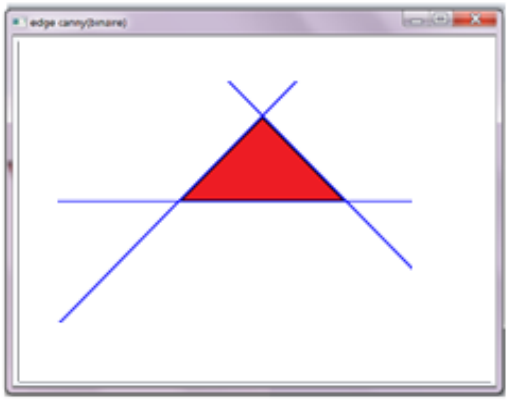
plans de hough 2D



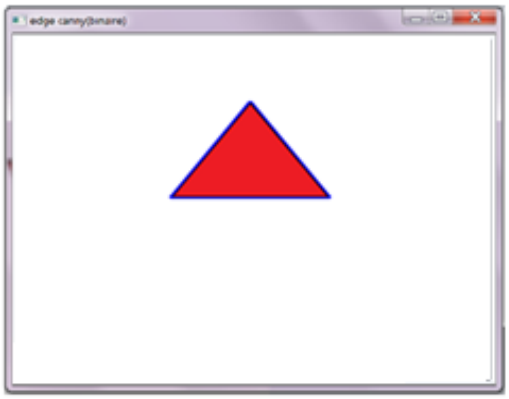
Les trois pics



plans de hough 3D



Détecter trois linges par la TH linge



la forme à détecter

Figure 2. 5 : applicationde la TH linéaire sur deux images

d Application de la transformée de Hough circulaire

Une fois l'image contour générée par le filtre de canny, on va appliquer la THC pour détecter les formes circulaires, on fait varier la valeur du seuil. On utilisera la représentation en coordonnées polaires pour le centre du cercle qu'on recherche dans l'image, le cercle ou les cercles recherchés étant décrits par l'équation du cercle:

$$(x - a)^2 + (y - b)^2 = r^2 \qquad (15)$$

En coordonnées polaires :

$$\mathbf{a} = \boldsymbol{\rho} * \mathbf{cos}(\boldsymbol{\theta}) \qquad (16)$$

$$\mathbf{b} = \boldsymbol{\rho} * \mathbf{sin}(\boldsymbol{\theta}) \qquad (17)$$

On applique l'algorithme de cercle sur les images contour, les résultats des tests sont illustrés dans la figure 2.6 suivante :

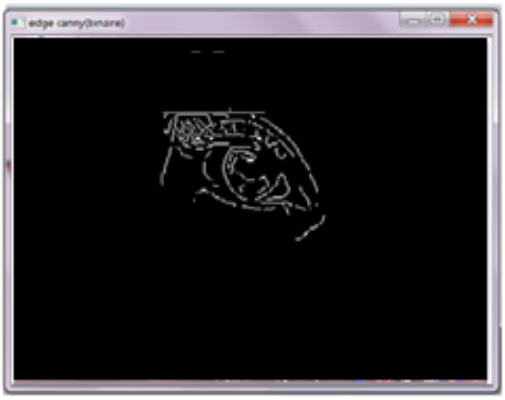
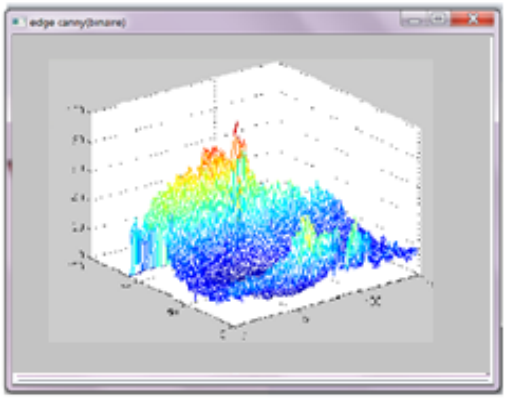


Image contour



plans de hough 3D

Affichage le pic de centre a détecté

détecter le cercle

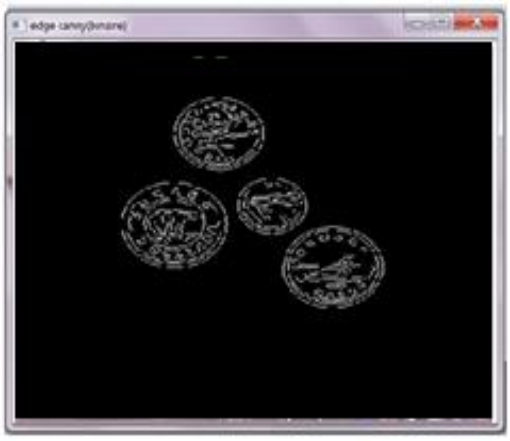
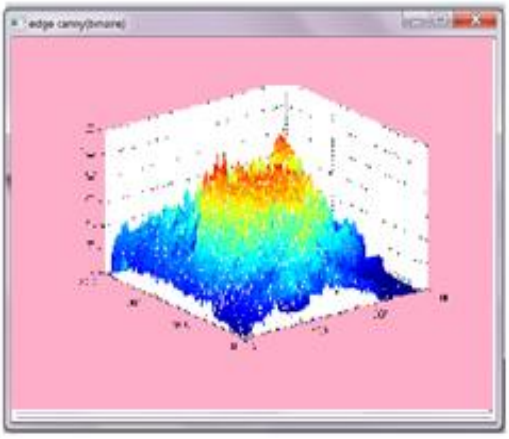
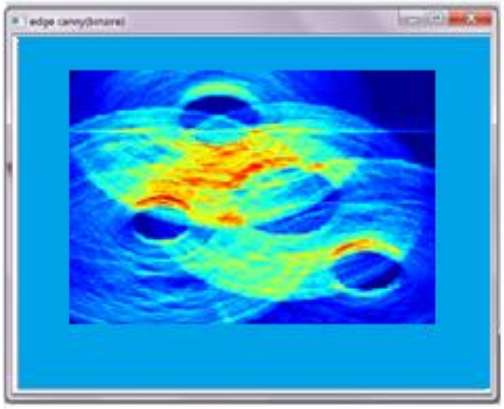


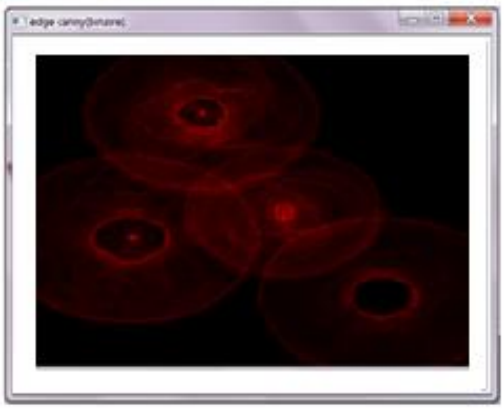
Image contour



plans de hough 3D



plans de hough 2D



Affichage les pics des centres a détecté

Détection les cercles dans l'image

Figure 2.6 : application la TH circulaire sur deux image

2.4 Conclusion

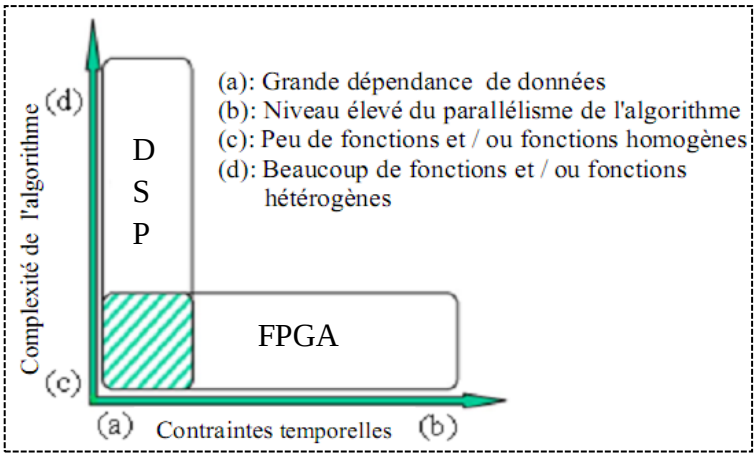
Dans ce chapitre, nous avons présenté les résultats obtenues après l'application de la transformée de Hough linéaire et la transformée de Hough circulaire pour la détection des plusieurs forme. Après l'élimination du bruit et la conversion de l'image en niveau de gris, le filtre de canny est appliqué pour obtenir une image contours la TH ligne et la TH circulaire appliqué à cette image. Le résultat obtenu au niveau de la simulation sous C++ Builder montre l'efficacité de l'algorithme de la TH ligne et TH circulaire dans la détection des formes.

Chapitre 3 Méthodologie de conception sur FPGA

3.1 Introduction

Les progrès technologiques continus dans le domaine des circuits intégrés ont permis la réduction des coûts et de la consommation. Les circuits intégrés spécifiques ont permis une réduction de la taille des systèmes numériques ainsi que la réalisation de circuits de plus en plus complexes, tout en améliorant leur performance et leur fiabilité. Aujourd'hui les techniques de traitement numérique occupent une place majeure dans tous les systèmes électroniques modernes grand public, professionnels ou militaires. De plus, les techniques de réalisation de circuits spécifiques, tant dans les aspects matériels (composants programmables, circuits pré caractérisés et bibliothèques de macro fonctions) que dans les aspect logiciels (placement-routage, synthèse logique) font désormais de la microélectronique une des bases indispensables pour la réalisation de systèmes numériques performants. Elle impose néanmoins une méthodologie de développement très structurée.

Lors de la conception des systèmes électroniques industriels (circuits) plusieurs critères doivent être pris en considération: le prix, la consommation de l'énergie (surtout dans le cas des systèmes embarqués), les performances demandées par l'application et avant tous est ce que le matériel est bien choisi et correspond l'algorithme à implémenter. Actuellement les deux moyens pour implémenter un contrôleur sont les DSPs et les FPGAs**.** Selon la nature de l'algorithme le concepteur peut faire un choix entre ces deux possibilités. La Figure (4.1). Illustre ce principe [34].



L'axe X de ce graphe représente les contraintes temporelles de l'algorithme, alors que

Figure 3.1 : Domaines d'utilisation des DSPs et des FPGAs

De nombreuses familles de circuits programmables et reprogrammables sont apparues depuis les années 70 avec des noms très divers suivant les constructeurs [35]. La figure suivante représente les différents circuits programmables et reprogrammables.

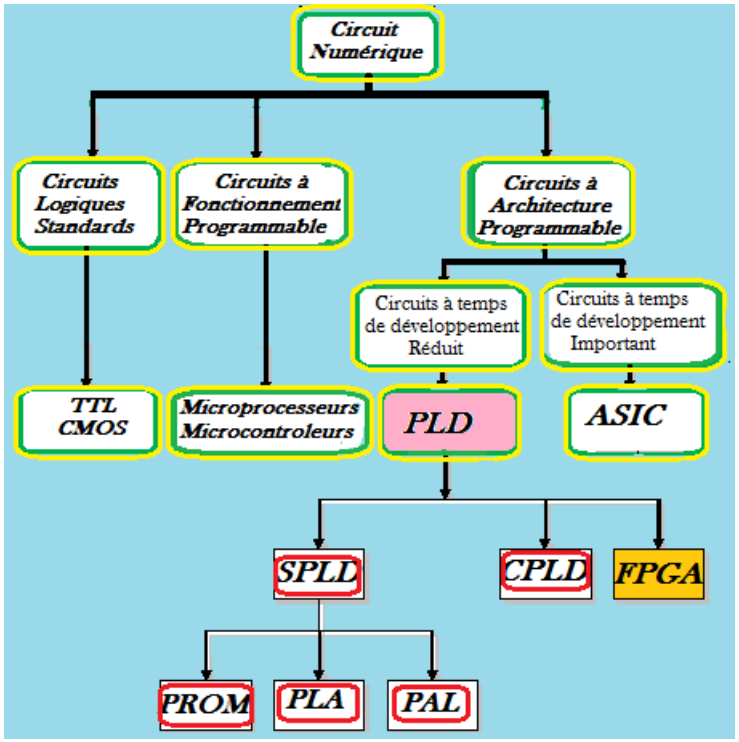


Figure 3.2: Classification des circuits numériques.

3.2 Circuits FPGA

Les FPGA (Field Programmable Gate Array) sont des circuits à architecture programmable qui ont été inventés par la société XILINX en 1985. Ils sont entièrement

reconfigurables et ne demandent donc pas de fabrication spéciale en usine, ni de systèmes de développement coûteux , ceci permet de les reprogrammer à volonté afin d'accélérer notablement certaines phases de calculs. Un autre avantage de ces circuits est leur grande souplesse qui permet de les réutiliser à volonté dans des algorithmes différents en un temps très court (quelques millisecondes).

Les FPGA sont utilisés dans de nombreuses applications, on en cite dans ce qui suit quelques-unes :

- Prototypage de nouveaux circuits.
- Fabrication de composants spéciaux en petite série.
- Adaptation aux besoins rencontrés lors de l'utilisation.
- Systèmes de commande à temps réel.
- Imagerie médicale.

3.2.1 Architecture des circuits FPGA

Les circuits FPGA possèdent une structure matricielle de deux types de blocs (ou cellules). Des blocs d'entrées/sorties (IOB : Input Output Bloc) et des blocs logiques programmables (CLB : Configurable Logic Bloc), comme le montre les figures (3.3 et 3.4). Le passage d'un bloc logique à un autre se fait par un routage programmable. Certains circuits FPGA intègrent également des mémoires RAM, des multiplieurs et même des noyaux de processeurs.

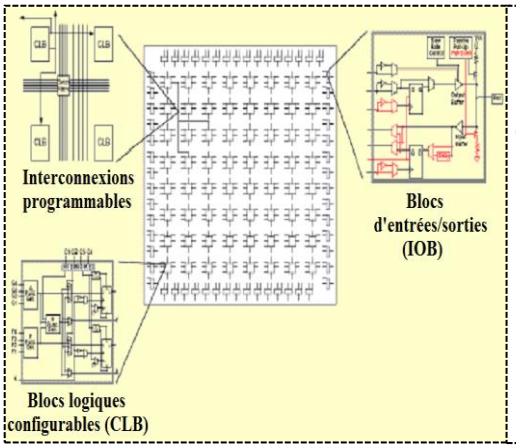


Figure 3.3 : Architecture interne d'un FPGA [35].

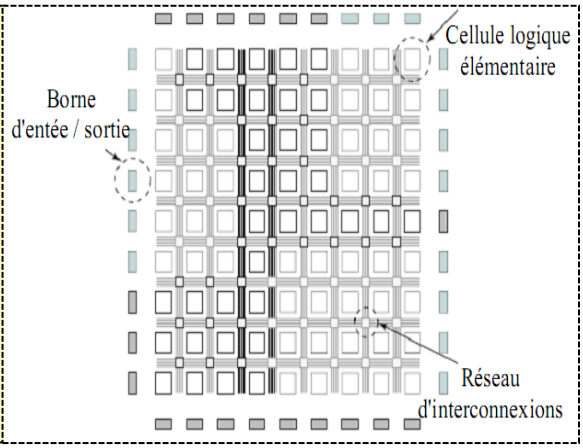


Figure 3.4 : Architecture des circuits FPGAs [36].

a Les blocs logiques configurables

Les blocs logiques configurables sont les éléments déterminants des performances du circuit FPGA figure (3.5). Chaque CLB est un bloc de logique combinatoire composé de générateurs de fonction à quatre entrées (LUT) et d'un bloc de mémorisation/synchronisation composé de bascules **D**. Quatre autres entrées permettent d'effectuer les connexions internes entre les différents éléments du CLB.

La LUT (Look Up Table) est un élément qui dispose de quatre entrées, il existe donc $\mathbf{2^4 = 16}$ combinaisons différentes de ces entrées. L'idée consiste à mémoriser la sortie correspondant à chaque combinaison d'entrée dans une petite table de **16** bits, la LUT devient ainsi un petit bloc générateur de fonctions. La figure (4.5) montre le schéma simplifié d'un CLB de la famille XC4000 de Xilinx.

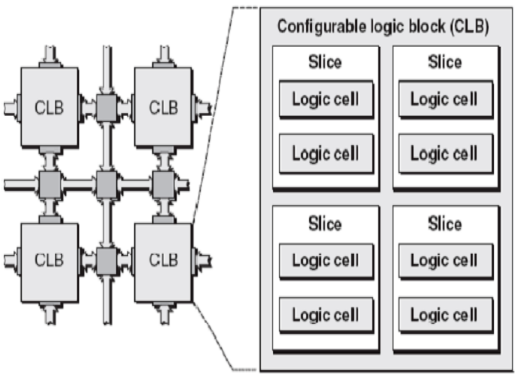


Figure 3.5 : Schéma simplifié d'un bloc logique configurable.

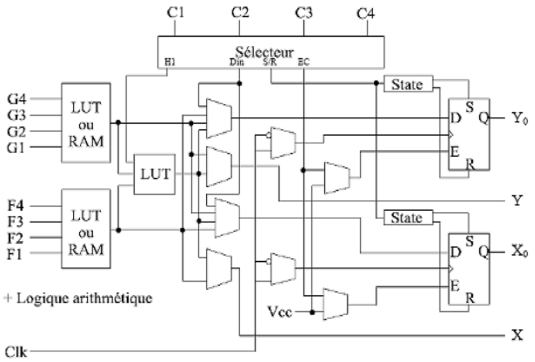


Figure 3.6 : Schéma d'une cellule

b Les blocs d'entrée/sortie

Les blocs d'entrée/sortie ou IOB (Input Output Bloc) permettent l'interface entre les broches du composant FPGA et la logique interne développée à l'intérieur du composant. Ils sont présents sur toute la périphérie du circuit FPGA. Chaque bloc IOB contrôle une broche du composant et il peut être défini en entrée, en sortie, en signal bidirectionnel ou être inutilisé (état haute impédance).

Les IOB peuvent s'adapter à un grand nombre de standards de communication: LVTTL, LVCMOS, PCI, LVDS, LVPECL, etc. Ils sont regroupés en banques, et chaque banque a sa propre tension d'alimentation. Ceci permet de combiner des standards incompatibles entre eux sur le même circuit FPGA en utilisant des banques différentes.

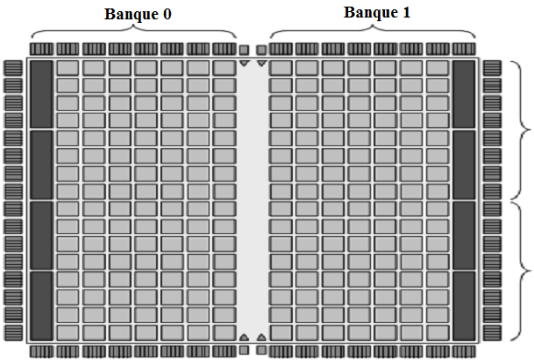


Figure 3.7 : Distribution des blocs d'entrée /sorti (IOB) en banque [39]

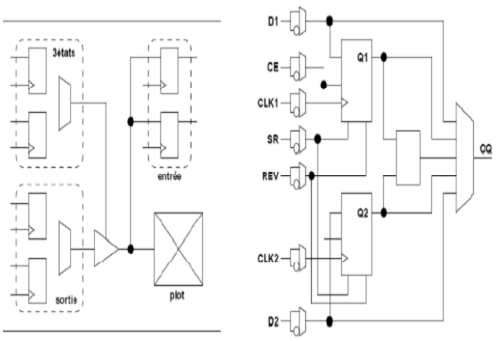


Figure 3.8 : Schéma d'un bloc d'entrée/sortie (IOB)[39]

c Le réseau d'interconnexions dans un circuit FPGA

Le réseau d'interconnexions permet la réalisation des connexions (i.e. le routage) entre les différentes tranches au sein d'un même CLB, entre les différents CLBs**,** les DCMs, les multiplicateurs, la mémoire RAM et les IOBs utilisés pour la construction d'un circuit donné. Au sein d'un même CLB, les différentes tranches peuvent communiquer via des connexions rapides. Une matrice des connexions programmables (Programmable Switch Matrix- PSM) est associée à chaque CLB, IOB, DCM, à la mémoire RAM et au circuit multiplicateur.

Le rôle de cette matrice est de réaliser une connexion programmée entre les conducteurs "entrants" et "sortants". La programmation de la connexion est établie à l'aide d'un mot de mémoire défini par la configuration. Grâce aux PSMs toute ressource peut donc accéder à toute autre ressource, quelque soit l'endroit où elle se trouve dans le circuit FPGA.

Les conducteurs "entrants" et "sortants" d'une matrice PSM sont disposés entre les lignes et les colonnes des CLBs [34].

Certains composants FPGA intègrent également des fonctionnalités particulières [40] :

- Blocs de mémoire supplémentaires (hors des LUT) comme les BRAM (BlockRAM), souvent double-port, parfois avec mécanisme de FIFO (First In First Out),
- Multiplieurs câblés (coûteux à implémenter en LUT),
- Cœur de microprocesseur enfoui (dit hard core),
- Blocs PLL (Phase Locked Loop ou boucle à verrouillage de phase) pour synthétiser ou resynchroniser les horloges,

- Cryptage des données de configuration.
- Sérialiseurs/Désérialiseurs dans les entrées-sorties, permettant des liaisons série à haut-débit.
- Impédance contrôlée numériquement dans les entrées-sorties, évitant l'utilisation de nombreux composants passifs sur la carte.

3.3 Méthodes de programmation

Il existe trois méthodes pour programmer un circuit FPGA [37] [38].

➢ ***SRAM:*** Les connexions sont réalisées en rendant les transistors passants. L'avantage de cette technologie est qu'elle permet une reconfiguration rapide au sein même du circuit. Le principal inconvénient est la surface nécessaire pour la SRAM. Elle nécessite l'utilisation d'une mémoire standard chargée à l'initialisation. XILINX et ALTERA utilisent cette technologie. Le composant peut être programmé infiniment.

➢ ***La technologie anti-Fuse*****:** Un état anti-fuse réside en un état d'une haute impédance. Il peut être programmé dans un état de faible impédance ou état "fused". Il s'agit d'une technologie moins chère que la SRAM, elle permet d'atteindre des vitesses plus élevées et occupe moins de place sur le circuit. Par contre, un tel FPGA ne peut être programmé qu'une seule fois. Les performances électriques sont supérieures à la technologie SRAM (minimisation des effets RC due à la faible surface). Actel et Quicklogic utilisent cette technologie.

➢ ***EPROM/EEPROM:*** Cette méthode est la même que celle utilisée dans les mémoires EPROM. L'EEPROM est reprogrammable. La puce fonctionne seule. La surface moyenne et les caractéristiques électriques sont semblables à la SRAM.

3.4 La famille Virtex-II

Dans les FPGAs de XILINX***, on*** distingue plusieurs familles de circuit à savoir XC2000, XC4000, XC5000, XC6000, XC3000, SPARTANN II, VERTEX-II, VERTEX-E, VERTEX-II-pro. La famille VERTE-II [40] a été conçue pour réaliser des conceptions à faible ou grande densité d'intégration et qui exigent des performances élevées (elle peut atteindre jusqu'à **10** millions de porte logique). La fréquence de son utilisation peut être portée à **420 Mhz**. Actuellement, cette famille est utilisée dans plusieurs

application tels que : les réseaux, la télécommunication, la vidéo et les application DSP (Digitale Signal Processing) .

Comme tout circuit FPGAs le VERTEXII est composé de :

- Bloc logique interne configurable
- bloc d'entrées/sorties (IOB).
- Routage globale et interconnexion.

3.4.1 Le bloc logique interne configurable

La logique interne configurable inclut quatre éléments majeurs organisés sous une forme régulière et un chemin dédié à la propagation de la retenue. Les éléments en question sont :

- ✓ Bloc logique configurable(CLB).
- ✓ Bloc Select RAM **18 K**bits à doubles ports.
- ✓ Bloc multiplieur **(18bit*18 bit).**
- ✓ DCM (Digital Clock Manager).

La connexion entre tous ces blocs est assurée d'une manière programmable grâce à des ressources d'interconnections configurables GRA '' Genral Routing Matrix''.

a Bloc Logique Configurable (CLB)

Les CLBs sont implémentés sous forme d'une matrice (N*M), ils incluent toute la logique nécessaire pour la conception des circuits combinatoires et séquentiels. Le CLB était l'élément déterminant des performances des premiers circuits FPGAs**.** Mais ce n'est plus le cas pour le VIRTEX-II**,** ou on ne parle plus de CLB mais de slice. Chaque CLB est composé de quatre Slices. Leur connexion au routage global est assurée par les matrices d'aiguillages GRM.

Les quatre slices sont implémentés en deux colonnes, chaque paire possède un chemin rapide dédié à la propagation de la retenue, leur interconnexion est assurée par un signal de liaison commun.

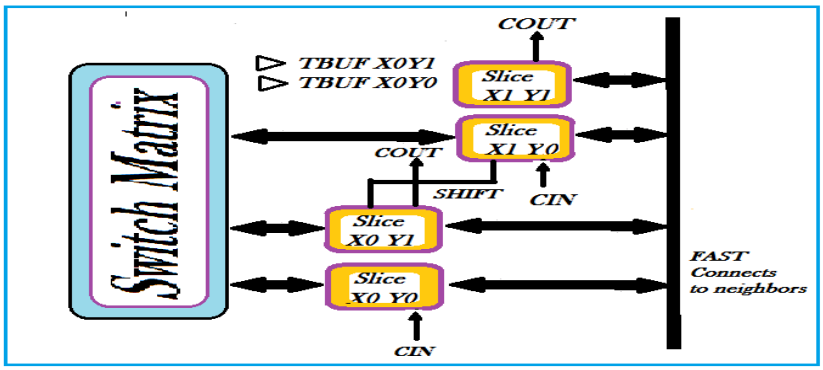


Figure 3.9 : Architecture d'un CLB du Virtex II

- Le CLB du Virtex II est structuré en vecteur.
- Il est configuré pour réaliser une fonction combinatoire ou séquentielle.
- Chaque CLB est attaché au switch matrice afin d'accéder au matrice de routage.
- Le CLB contient **4** slices, ces derniers sont répartis en deux colonnes de deux slices avec deux chaines de propagation de retenue et une ligne shift.

a.1 Description d'un Slice

L'architecture d'un slice se présente comme une structure symétrique. Cette caractéristique rend ce dernier flexible à l'optimisation des applications données. Chaque Slice est composé de :

- ✓ Deux fonctions génératrices **F** et **G** à quatre entrées et une sortie **.**
- ✓ Deux éléments de stockage (bascule **D**).
- ✓ Des portes logiques **(XOR).**
- ✓ une logique pour la propagation de la retenue.
- ✓ Des multiplexeurs.

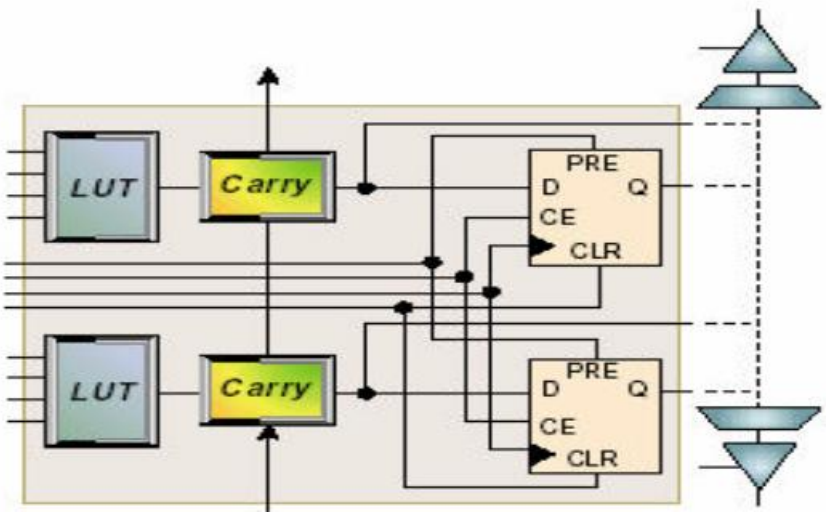


Figure 3.10: Architecture simplifié d'un Slice.

La figure (4.10), illustre l'architecture simplifiée d'un slice .La logique combinatoire est implantée grâce aux LUT (Look-Up Table) contenues dans chaque slice. Ces LUT peuvent également être configurés comme éléments de mémoire synchrones simple ou double –port de **16** bits, ou encore comme registres à décalage de **16** bits.Il existe donc trois modes de configuration de ces LUTs. Plus précisément, le fonctionnement en mode combinatoire est obtenu en lisant le contenu pointé par les signaux d'entrée. Autrement dit, les LUT sont des mémoires dont le contenu est initialisé lors de la configuration du FPGA. De ce fait, elles permettent à l'utilisateur d'en disposer d'un mode ‹élément mémoire › dans chacun des slices si nécessaire. La figure (4.11).

Décrit le mode de configuration particulier en registre à décalage de longueur programmable jusqu'à **16** bits. Par ailleurs, une logique supplémentaire utile pour la réalisation de fonctions implantées à raison de **2** bits par slice

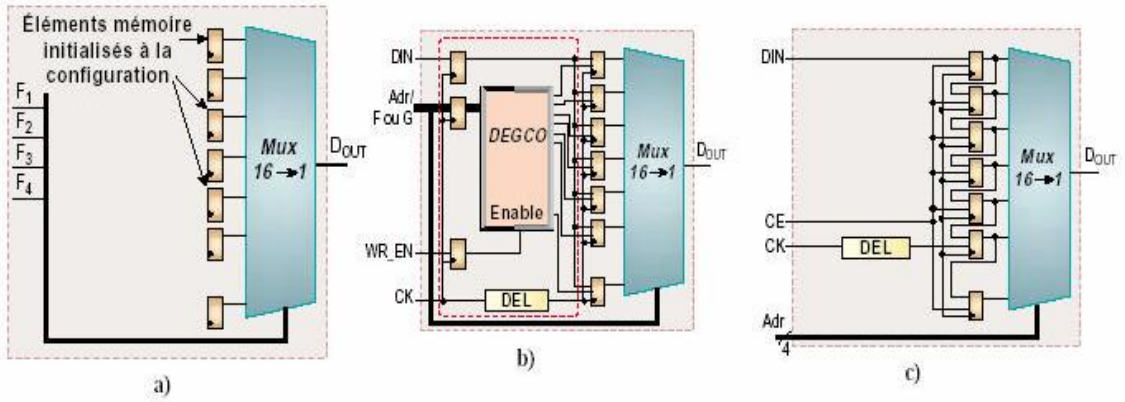


Figure 3.11: .Trois modes de configuration des LUTs.

b Bloc Select RAM à18 bits

Les blocs ‹SelectRam› dans l'architecture Virtex-II sont des mémoires Ram double port de **18K** bits, chaque port est totalement synchrone et indépendant. Ces mémoires sont configurables de différentes manières entre **18*1** Kbits et **512*36** bits.

Ces configurations sont déterminées par un compromis entre la taille de la donnée et le nombre d'adresses linges. Le compromis en question doit respecter la taille globale de la mémoire qui est de **18** Kbits. Les blocs «SelectRam ›› peuvent être mis en cascade pour réaliser des larges zones des stockages enfouies dans le FPGA.

Un module multiplieur **18*18** bits est disposé à proximité de chaque bloc «SelectRam ›› et optimisé pour opérer sur le contenu d'un port de la mémoire.

*c Bloc multiplieur 18*18 bits*

Ce bloc multiplieur du virtex II est un multiplieur complément à deux signé de **18**bits ***18**bits. Il est conçu d'une manière optimisé pour le calcul rapide une consommation de puissance faible par rapport au multiplieur configuré dans les slices.

La disposition des blocs multiplieur se trouve implémentée juste à côté des blocs selectRam. Ceci permet d'augmenter les performances d'une application si les opérandes proviennent d'une mémoire. Chaque Bloc SelectRam et un bloc multiplieur sont connectés au routage global grâce à quatre matrices d'interconnexion figure (3.12).

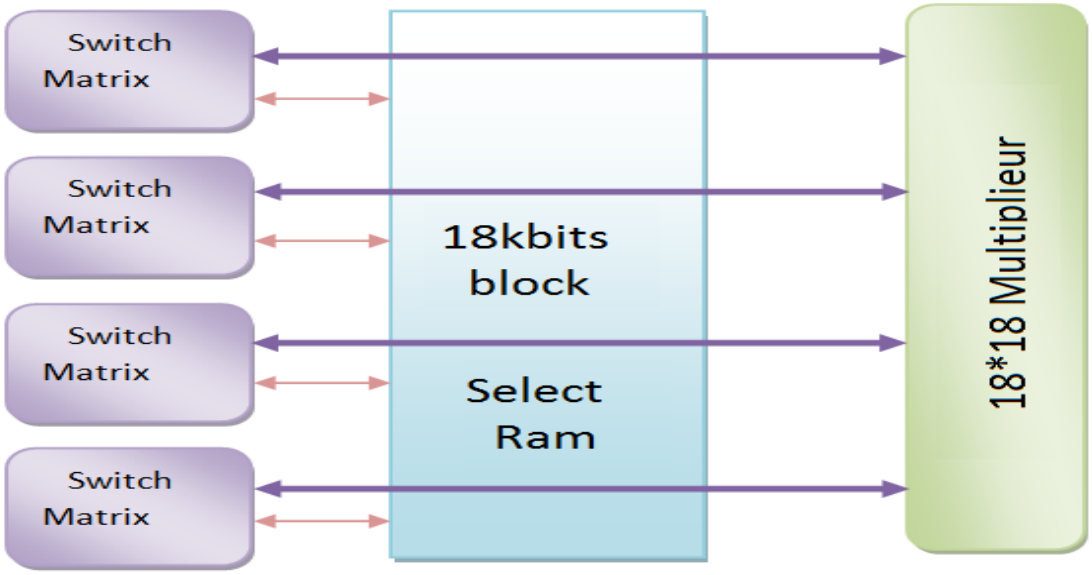


Figure 3.12 : bloc select RAM et bloc multiplieur.

d Les DCMs

Le signal d'horloge d'un FPGA est fourni de son voisinage. Il y a des entrées spécialisées pour recevoir les signaux d'horloge et les distribuer ensuit à l'intérieur du circuit. Le virtex II possède des blocs internes pour la gestion du signal d'horloge, les DCM (Digital Clock Managers). Ces derniers génèrent d'autres fréquences d'horloge à partir du signal externe, en éliminant le JITTER (retards entre les différents francs des signaux).

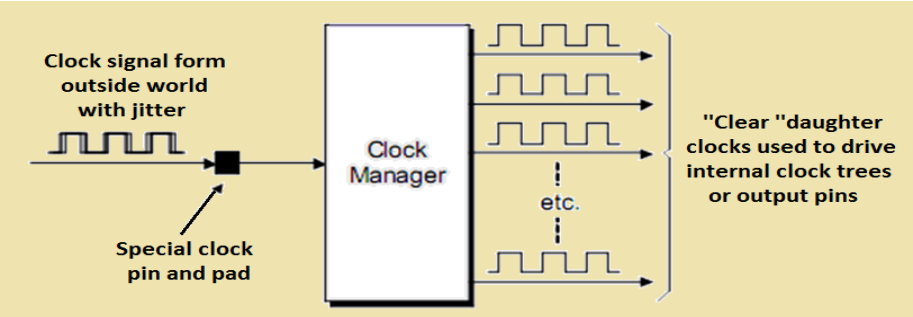


Figure 3.13 : Les DCM génèrent d'autres fréquences d'horloge

3.4 .2 le bloc d'entrées/sorties

Les blocs d'entrée/sortie (Figure 3.14) fournissent une interface entre les broches externes du circuit et la logique interne. Les IOBs sont fournis en groupe de deux ou quatre dans la périphérie du circuit. Ils peuvent être configurés en entre ou en sortie ou en entrée- sortie. Ils incluent six bascules **D**, leur utilisation est souvent recommandée pour la synchronisation des signaux de communication.

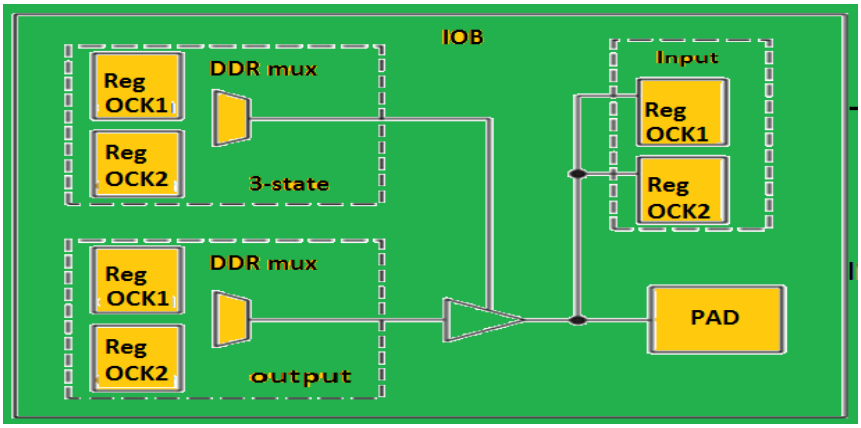


Figure 3.14 **:** Bloc IOB de VIRTEX II.

3.4. 3 Routage globale et interconnexion

Les ressources de routage global d'un circuit sont optimisées pour d'éventuelles prévisions des performances d'une application donnée l'accès au routage des différents blocs disponibles sur le circuit (IOB, CLB, Bloc RAM, Multiplieur et les DCMs) est réalisé d'une manière programmable grâce à des matrices d'aiguillage. (Voir figure 3.15).

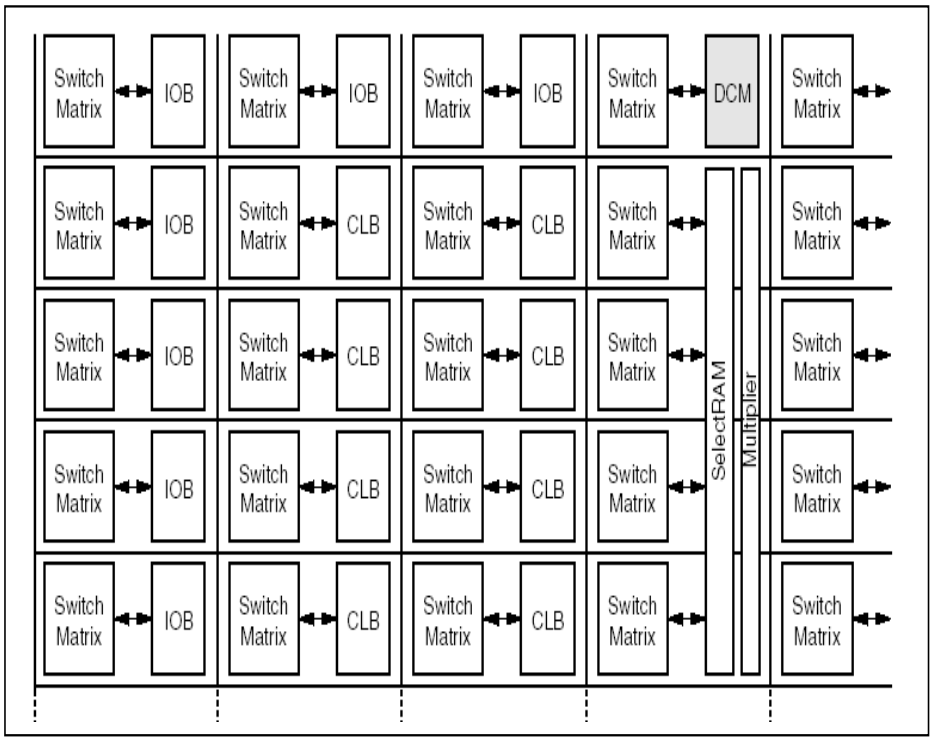


Figure 3.15 : les interconnexions des ressources interne du circuit FPGA.

3.5 Les avantages du FPGA

Ces notions sur le FPGA donnent des indications quant à l'intérêt de son utilisation dans notre travail. Les avantages de ce circuit se résument en :

- ✓ **un circuit reprogrammable :** l'avantage du FPGA est de pouvoir être reprogrammable contrairement aux circuits intégrés de type ASIC. Ce qui rend cette solution modulable et donne la possibilité de modifier le programme générique de base afin de le rende spécifique au circuit utilisé.

- ✓ **Un investissement rentable dans la durée :** Cela est dû à sa reprogrammation, Ce Qui implique une réalisation à destination d'autres projets, malgré un prix à l'achat supérieur à un circuit ASIC**.**
- ✓ **Une Reprogrammation quasi-instantanée du circuit :** Une fois le programme validé cela ne prend que quelques minutes à l'implémenter. A titre de comparaitrions, la fabrication d'un circuit ASIC peut prendre plusieurs semaines.

3.6 Etapes nécessaires au développement d'un projet sur circuit FPGA

Le développement en VHDL nécessite l'utilisation de deux outils : le simulateur et le synthétiseur. Le premier va nous permettre de simuler la description VHDL avec un fichier de simulation appelé « test-bench », cet outil interprète directement le langage VHDL et il comprend l'ensemble du langage. L'objectif du synthétiseur est très différent : il doit traduire le comportement décrit en VHDL en fonctions logiques de bases, celles-ci dépendent de la technologie choisie, cette étape est nommée « Synthèse ». L'intégration finale dans le circuit cible est réalisée par l'outil de placement et routage. Celui-ci est fourni par le fabricant de la technologie choisie. Le langage VHDL permet d'écrire des descriptions d'un niveau comportemental élevé.

La question est de savoir si n'importe quelle description comportementale peut être traduite en logique ?

Avec les outils actuels, il est possible de disposer de fichiers VHDL à chaque étape. Le même fichier de simulation (test-bench) est ainsi utilisable pour vérifier le fonctionnement de la description à chaque étape de la procédure de développement. La figure 3.16 donne les différentes étapes nécessaires au développement d'un projet sur circuit FPGA.

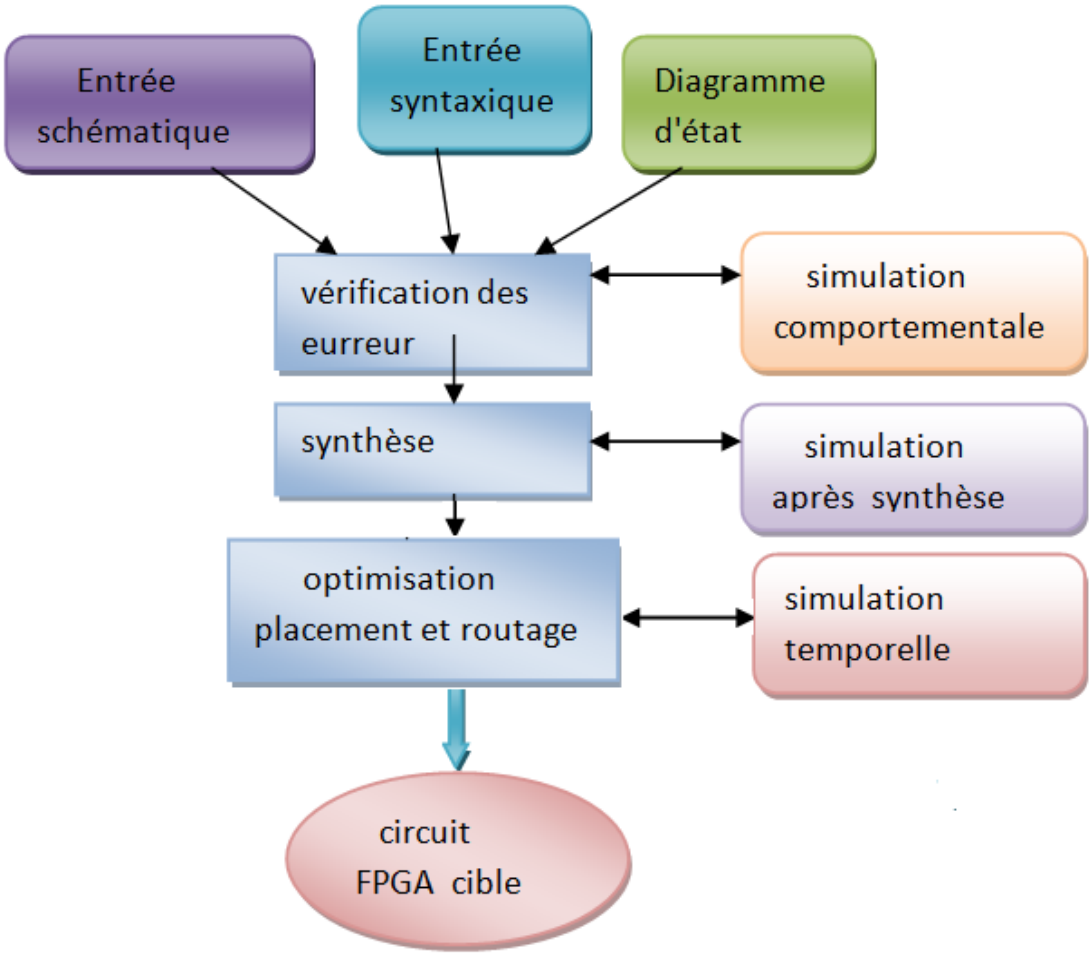
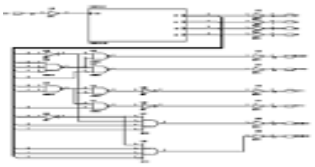
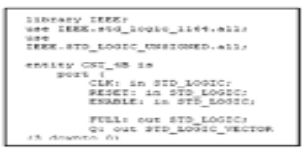


Figure 3.16 : Organisation fonctionnelle de développement d'un projet sur circuit FPGA



Entrée schématique



Entrée syntaxique

Diagramme d'états

3.6.1 Saisie du texte *VHDL*

La saisie du texte VHDL se fait sur le logiciel « ISE Xilinx Project Navigator ». Ce logiciel propose une palette d'outils permettant d'effectuer toutes les étapes nécessaires au développement d'un projet sur circuit FPGA. Il possède également des outils permettant de mettre au point une entrée schématique ou de créer des diagrammes d'état, qui peuvent être utilisés comme entrée au lieu du texte VHDL.

La figure 3.17 montre comment se présente le logiciel «ISE Xilinx Project Navigator».
La saisie du texte se fait sur la partie droite de l'écran, on voit en haut à gauche la hiérarchie du projet, et en bas à gauche les nombreux outils nécessaires tout au long du développement du projet.

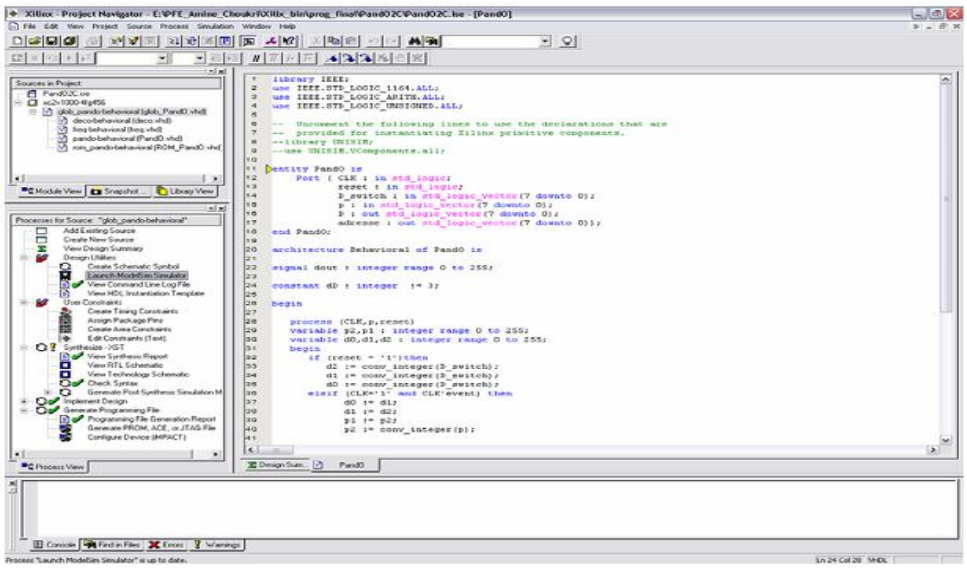


Figure 3.17: Vue d'ensemble du logiciel « Xilinx Project Navigator »

Il faut commencer par créer un projet, ensuite inclure des fichiers sources dans lesquels il faut saisir le texte VHDL désiré. On peut inclure autant de sources qu'on veut dans un projet.

3.6.2 Vérification des erreurs

Cette étape est effectuée en appuyant sur le bouton « check syntax ».
Elle permet de vérifier les erreurs (errors) de syntaxe du texte VHDL et d'afficher les différentes alarmes (warnings) liées au programme, par exemple des signaux déclarés mais non utilisés dans le programme. S'il y'a des erreurs dans le programme, il ne peut pas être synthétisé, mais la présence d'alarmes n'empêche pas de poursuivre normalement les autres étapes du développement.

a Synthèse

La synthèse permet de réaliser l'implémentation physique d'un projet.
Le synthétiseur a pour rôle de convertir le projet, en fonction du type du circuit FPGA cible utilisé, en portes logiques et bascules de base. L'outil « View RTL Schematic »

permet de visualiser les schémas électroniques équivalents générés par le synthétiseur (figure 3.18).

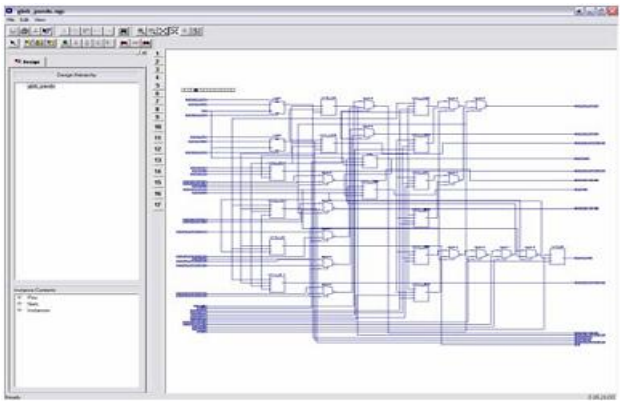


Figure 3.18 : Aperçu de l'outil « View RTL Schematic »

De plus, le synthétiseur permet à l'utilisateur d'imposer des contraintes de technologie (User constraints) : par exemple fixer la vitesse de fonctionnement (Create Timing Constraints), délimiter la zone du circuit FPGA dans laquelle le routage doit se faire (Create Area constraints) ou affecter les broches d'entrées/sorties (Assign Package Pins). La figure 3.19 montre un aperçu de l'outil d'assignation des broches d'entrées/sorties.

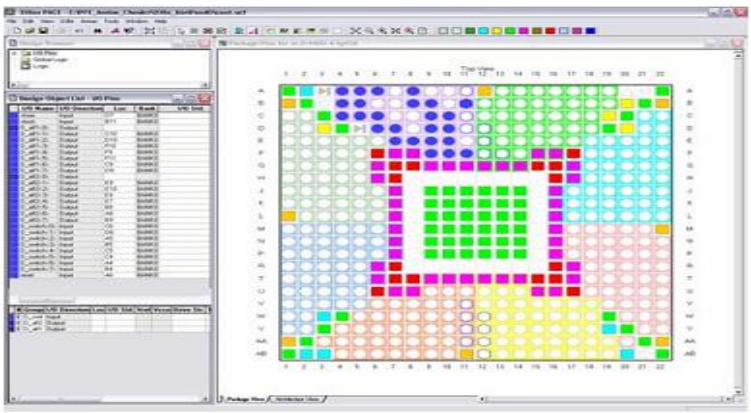


Figure 3.19 : Aperçu de l'outil d'affectation des broches d'entrées/sorties

b Simulation

Le simulateur utilisé est le « ModelSim Simulator » (figure 3.19). La simulation permet de vérifier le comportement d'un design avant ou après implémentation dans le composant cible. Elle représente une étape essentielle qui nous fera gagner du temps lors de la mise au point sur la carte. Il faut juste noter qu'un projet peut être simulé même s'il n'est pas synthétisable.

Lors de l'étape de simulation comportementale, on valide l'application indépendamment de l'architecture et des temps de propagation du futur circuit cible. La phase de simulation après synthèse valide l'application sur l'architecture du circuit cible, et enfin la simulation temporelle prend en compte les temps de propagation des signaux à l'intérieur du circuit FPGA cible.

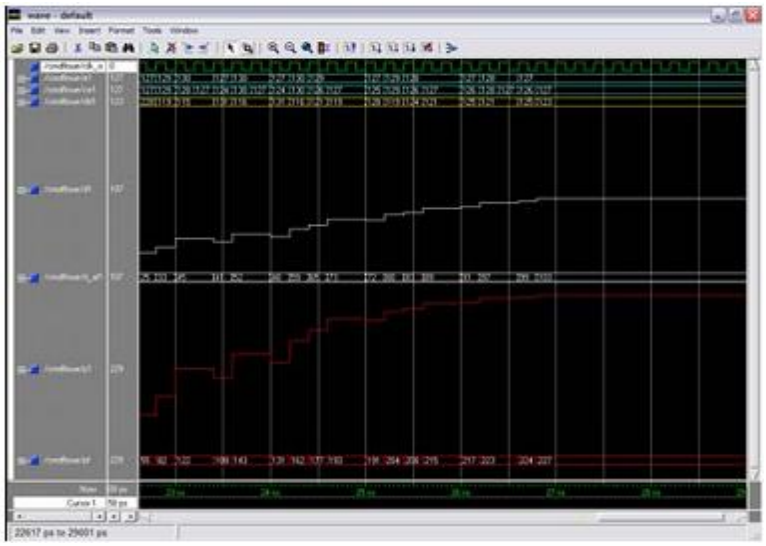


Figure3.20 : Présentation du simulateur ***«*** ModelSim Simulator »

c Optimisation, placement et routage

Pendant l'étape d'optimisation, l'outil cherche à minimiser les temps de propagation et à occuper le moins d'espace possible sur le circuit FPGA cible.

Le placement et routage permet de tracer les routes à suivre sur le circuit afin de réaliser le fonctionnement attendu. La figure (3.21) donne un aperçu de l'outil de placement et routage « FPGA Editor » qui permet de visualiser et d'éditer le circuit routé.

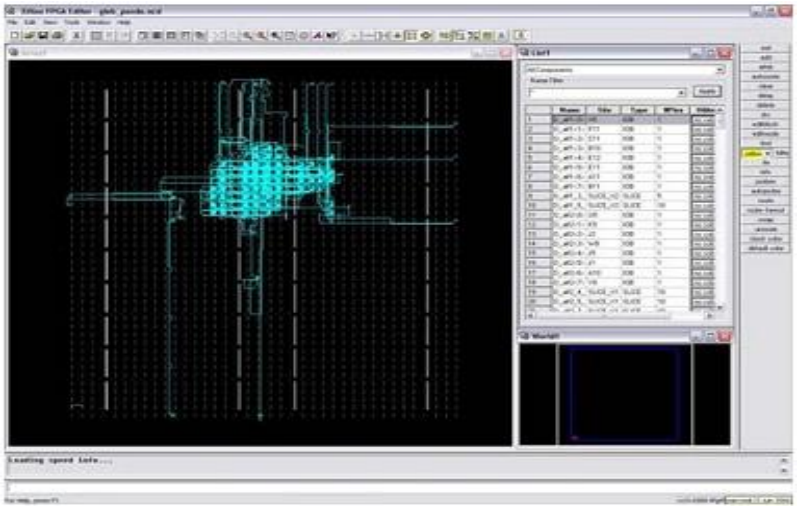


Figure 3.21 : Aperçu de l'outil « FPGA Editor »

d Programmation du composant et test

Dans cette dernière étape (Generate Programming Files), on génère le fichier à charger sur le circuit FPGA à travers l'interface JTAG. Une fois le programme chargé sur le circuit, on peut tester et visualiser les résultats directement sur la carte de développement Virtex-II à travers les nombreuses interfaces qu'elle offre, soit directement sur les deux afficheurs 7 segments, soit à travers l'interface RS232.

3.7 CONCLUSION

Nous avons exposé dans ce chapitre plusieurs cibles technologiques pour une conception hardware. Les circuits FPGAs de Xilinx ont été présentés plus en détails vu que nous allons implémenter notre application sur un circuit Virtex- II de Xilinx.

Autrement dit, il faut bien comprendre les possibilités offertes par le réseau logique programmable choisi ainsi que les moyens à mettre en œuvre pour en tirer le meilleur profil.

Chapitre 4 Conception et Implémentation

4.1 Introduction

Les simulations sous C++ Buider ont montré que la détection des formes par Transformée de Hough linéaire et Circulaire se fait avec une bonne précision.

La transformée de Hough est une technique bien connue pour sa robustesse dans le domaine de la reconnaissance de forme. La complexité de calcul et l'exigence excessive de la mémoire sont les principaux obstacles pour l'implémentation Hardware de la TH. Différents architectures et algorithmes [25], [26], [27], [28], [29] et [30] de la TH ont été proposés dans la littérature. Ces derniers proposent des solutions qui facilitent le calcul des fonctions trigonométrique.

Parmi les solutions proposées pour la transformée de Hough linéaire, c'est la transformée de Hough incrémentale [11]. Cette dernière simplifie l'expression de la TH ligne, en proposant une expression à base de fonctions d'additions et des décalages ce qui rend l'implémentation hardware simple et rapide.

D'autre part la TH circulaire (THC) a été modifiée par l'utilisation de l'algorithme CORDIC (***C****ordiante* ***R****otation* ***Di****gital* ***C****omputer*) pour le calcul des fonctions trigonométriques nécessaires dans le calcul de la THC. Sachant que l'algorithme de Cordic permet de calculer le Cos et Sin à base des primitives élémentaires en base 2.

Dans ce chapitre, il sera question de simuler et implémenter deux architectures : une pour la transformée de Hough incrémentale et l'autre pour la transformée de Hough circulaire à base de CORDIC.

Les architectures sont conçues dans l'environnement de Xilinx ISE 7.1i. Le processus d'implémentation convoite certaines phases de vérification. En premier lieu,

les architectures doivent être décrites en langage (VHDL). Après la description VHDL, nous utilisé le simulateur (ISE simulator) pour faire une simulation fonctionnelle des fichiers. Les résultats de simulation figurent sous forme de chronogramme. Une fois que la simulation fonctionnelle est validée, la phase de synthèse va nous permettre d'avoir une première estimation sur les ressources du circuit FPGA consommées, et sur la fréquence maximale de fonctionnement. Finalement, la phase de placement et routage éclate schématiquement la surface consommée par l'architecture globale et configure les routages d'interconnexion entre les différents blocs des ressources du circuit FPGA**.** Dans la simulation temporelle, on vérifie si le circuit obtenu respecte les contraintes temporelles et utilise les délais des portes et les délais dans les interconnexions pour calculer la vitesse maximale.

4. 2 Conception de la TH linéaire incrémental

4.2.1 Algorithme de S.Tagzout et AL

Afin d'accélérer le calcul de la TH linéaire, l'algorithme de S.Tagzout et Al [32], propose une solution qui règle deux problème: proposer une expression de la TH à base des additions et de décalage au lieu de cos et sin. De plus, accélérer la vitesse de génération des paramètres $\boldsymbol{\rho}$**.** Cet algorithme permet de générer les $\boldsymbol{\rho}$ correspondant à $\mathbf{0} \leq \boldsymbol{\theta} < \frac{\boldsymbol{\pi}}{\mathbf{2}}$ en même temps que les $\boldsymbol{\rho}$ correspondant à $\frac{\boldsymbol{\pi}}{\mathbf{2}} \leq \boldsymbol{\theta} < \pi$. En utilisant les mêmes notations de l'algorithme de H.Koshimizu et M.Numada [10], les nouvelles expressions de la TH incrémentale s'écrivent comme suit :

$$\begin{cases} \rho_{n+1} = \rho_n + \varepsilon . \rho_{n+\frac{K}{2}} \\ \rho_{n+1+\frac{K}{2}} = \rho_{n+\frac{K}{2}} - \varepsilon . \rho_n \end{cases} \quad \text{Avec} \quad \begin{cases} \rho_{n+\frac{K}{2}} = \rho'_n \ , \ \rho'_0 = x & (1) \\ \rho_{\frac{K}{2}} = \rho'_0 = y & (2) \end{cases}$$

La réalisation hardware de cet algorithme implique l'implémentation des deux expressions (1)(2) les valeurs $\boldsymbol{\rho_{n+1}}$ et $\boldsymbol{\rho_{n+1+\frac{K}{2}}}$ doivent être régulièrement substituées,

après un certain nombre d'itérations définies expérimentalement, par des paramètres ρ_u et ρ'_u obtenus à partir de l'expression usuelle (2) tels que :

$$\begin{cases} \rho_u = x\cos(\theta) + y\sin(\theta) & 0 \leq \theta < \frac{\pi}{2} \quad (3) \\ \rho'_u = -x\cos(\theta) + y\sin(\theta) & \frac{\pi}{2} \leq \theta < \pi \quad (4) \end{cases}$$

La figure (4.1) montre un module qui réalise les deux expression (1)(2) et qui donne la possibilité de faire introduire les paramètres ρ_u et ρ'_u obtenus à partir de l'expression usuelle (1). A l'entrée de ce module, on trouve des multiplexeurs à trois entrées qui servent à introduire régulièrement, les valeurs initiales ρ_0 *et* $\rho_{K/2}$, les valeurs ρ_n et $\rho_{n+K/2}$, issues de l'incrémentation de θ et les valeurs correctes ρ_u et ρ'_u, issues de circuit qui réalise l'expression usuelle .

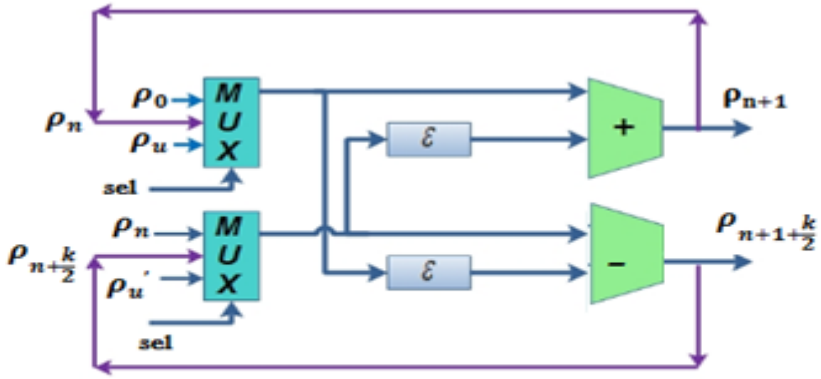


Figure 4 .1 : Génération des ρ incrémentale d'après S.Tagzout et Al.

L'implémentation de la TH incrémentale proposée par S.Tagzout et Al, conduit Nécessairement à implémenter une architecture qui calcule l'expression usuelle.
Cette expression est utilisée pour corriger les erreurs qui proviennent du processus de l'incrémentation, car les approximations prises sur **Cos** et **Sin** au cours de l'incrémentation des expressions (3) et (4) engendrent des résultats erronés (plus de détails dans [32]).

Cet algorithme diminue l'utilisation des LUT set réduit le temps global de calcul de TH puisqu'il peut générer deux valeurs de ρ en même temps, pour des valeurs de θ appartenant à $[0, \frac{\pi}{2}]$ l'intervalle. L'implémentation de cet algorithme donne

de bons résultats [32], au niveau de performance temporelle, mais elle ne résout pas le problème d'utilisation des LUTs qui occupent beaucoup d'espace.

4.2.2 Algorithme de TH incrémental généralisée

L'idée de base de cet algorithme [12] est de générer est **M** valeur de **ρ** dans un intervalle réduit de **θ**, où les approximations sur **Sin** et **Cos** n'engendrent pas d'erreurs significatives sur les résultats. Cet algorithme utilise les mêmes approximations sur **Cos** et **Sin** de l'algorithme précèdent, mais pour chaque valeur de θ_n, **M** valeur de **ρ** générées en même temps dans l'intervalle**] 0, π/M [**de **θ**. Cet algorithme est défini par l'expression générale suivante :

$$\begin{cases} 0 \leq m < M \quad \text{et} \quad 1 < M \leq K \\ 0 \leq n < \frac{K}{M} \\ \rho_{n+1+\frac{mK}{M}} = \rho_{n+\frac{mK}{M}} + \alpha . \varepsilon . \rho_{n+\frac{\alpha K}{2}+\frac{mK}{M}} \\ \rho_{\frac{mK}{M}} = x \cos\left(\frac{mK}{M}\ \varepsilon\right) + y \sin\left(\frac{mK}{M}\ \varepsilon\right) \\ \alpha = \begin{cases} 1 & si \quad m < \frac{K}{2} \\ -1 & si \quad m \geq \frac{K}{2} \end{cases} \end{cases} \qquad (5)$$

Avec

- **ε** est la résolution de **θ**.
- **M** Le nombre de valeurs de **ρ** générées en même temps.
- **K** représente le nombre de division de **θ**.
- ρ_i est la valeur de **ρ** obtenue pour un angle θ_i de l'axe de **θ**.

Cette expression peut être réarrangée pour aboutir au système d'équations suivant :

$$\begin{cases} 0 \le m < \frac{M}{2} \quad \text{et} \quad 1 < M \le K \\ 0 \le n < \frac{K}{M} \\ \rho_{n+1+\frac{mK}{M}} = \rho_{n+\frac{mK}{M}} + \alpha.\varepsilon.\rho_{n+\frac{K}{2}+\frac{mK}{M}} \\ \rho_{n+1+\frac{mK}{M}+\frac{K}{2}} = \rho_{n+\frac{mK}{M}+\frac{K}{2}} - \varepsilon.\rho_{n+\frac{mK}{M}} \\ \rho_{\frac{mK}{M}} = x\cos(\frac{mK}{M}\varepsilon) + y\sin(\frac{mK}{M}\varepsilon) \\ \rho_{\frac{mK}{M}+\frac{K}{2}} = y\cos(\frac{mK}{M}\varepsilon) - x\sin(\frac{mK}{M}\varepsilon) \end{cases} \quad (6)$$

Pour cette nouvelle expression, on note que :

- Les valeurs initiales sont calculées par les deux équations $\rho_{\frac{mK}{M}}$ et $\rho_{\frac{mK}{M}+\frac{K}{2}}$
- Pour calculer la prochaine valeur de $\rho(\rho_{n+1})$ dans l'intervalle $[\frac{mK}{M}, \frac{(m+1).k}{M}]$ ona besoin de l'ancienne valeur de $\rho(\rho_n)$ calculée dans le même intervalle et l'ancienne de $\rho(\rho_{n+K/2})$
  calculée dans l'intervalle $[\frac{mK}{M} + K/2, \frac{(m+1).k}{M} + K/2]$ en même temps et vice versa.
- Les valeurs de ρ calculées dans les intervalles $[\frac{mK}{M}, \frac{(m+1).k}{M}]$ et $[\frac{mK}{M} + K/2, \frac{(m+1).k}{M} + K/2]$ sont générées en même temps, ce qui réduit le temps de calcul globale de la TH.
- La valeur de **M** doit être paire.
- La valeur de **K/M** doit être entière.

a Implémentation

Nous allons implémenter l'algorithme de la TH incrémental généralisé (THIG). La figure 4.2 suivante montre une architecture qui peut générer les distances ρ de l'algorithme.

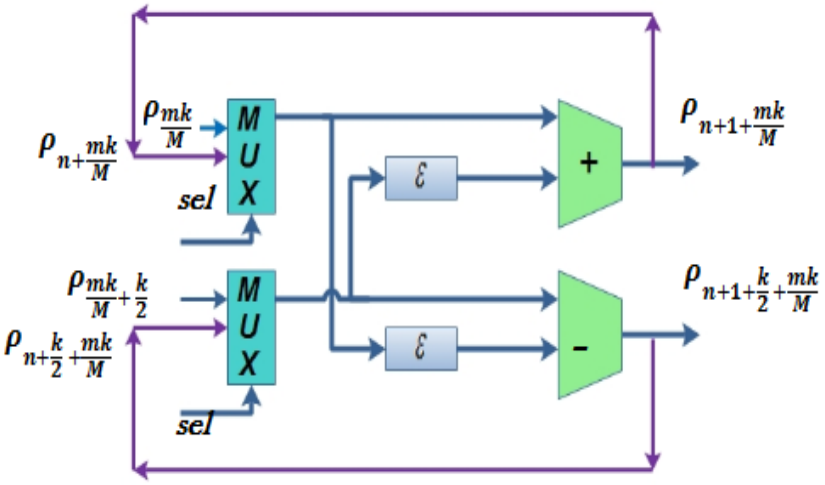


Figure 4.2 : architceture d'un module de générationin de **ρ** selon la THIG.

4.3 Description de l'arhitecture globale

Pour le cas particulier de **M=4**, l'expression générale (6) de l'algorithmede la THIG sera défini comme suit :

$m \in \{0,1\}$, $0 <= n < k/4$ ainsi on aura le système d'équation suivant :

$$\begin{cases} \rho_{n+1} = \rho_n + \varepsilon . \rho_{n+\frac{K}{2}} \\ \rho_{n+1+\frac{K}{2}} = \rho_{n+\frac{K}{2}} - \varepsilon . \rho_n \\ \rho_{n+1+\frac{K}{4}} = \rho_{n+\frac{K}{4}} + \varepsilon . \rho_{n+\frac{3K}{4}} \\ \rho_{n+1+\frac{3K}{4}} = \rho_{n+\frac{3K}{4}} - \varepsilon . \rho_{n+\frac{K}{4}} \\ \rho_0 = x \ , \ \rho_{\frac{K}{2}} = y \\ \rho_{\frac{K}{4}} = \frac{\sqrt{2}}{2} [x + y] \\ \rho_{\frac{3K}{4}} = \frac{\sqrt{2}}{2} [x - y] \end{cases} \qquad (7)$$

La réailisation hardware de cet algorithme impllique l'impliémentation de l'expersssion (7) sachant que ρ_0 ,$\rho_{k/2}$,$\rho_{k/4}$,$\rho_{3k/4}$ sont les valeurs initiales.l'architecture

proposé devient alors :

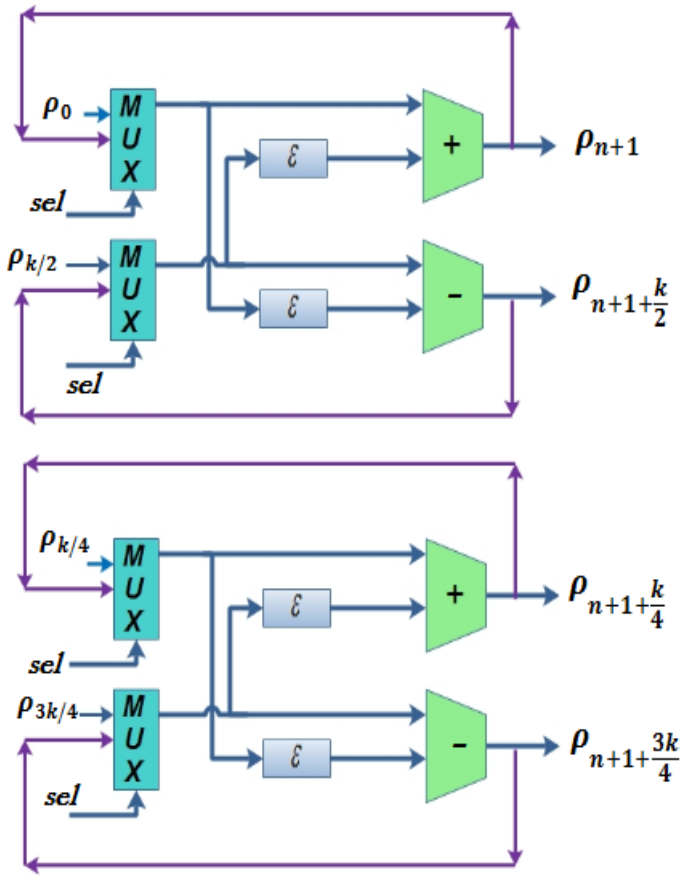


Figure 4.3: Architecture du bloc de génération des distances **ρ** pour M=4.

4.3.1 Choix de ε

La valeur **ε** peut être choisie comme très proche de **$1/2^p$,** sachant que la division par **2^p** est effectuée par un décalage de **p** bits vers la droite, alors la multiplication par **ε** dans l'équation (7) devient une simple fonction de décalage .pour notre calcul, on doit tenir compte que **K** doit être un multiple de **M.** Comme **ε = π/4,** on choisit le nombre **K** de division de l'angle égale à **48** car il est un multipe de **M=4 ,**et **$\pi/48 = 0.065 \approx 1/2^4$** donc la multiplication par **ε** est réalisée par un décalage de **4** bits vers la droite.

4.3.2 Choix de repère

Nous avons choisit l'origine de plans (x,y) au milieu de l'image pour calculer la TH dans l'intervalle [0,π] de θ au lieu de [0,2π], ainsi on réduit l'espace de Hough qui induit une réduction dans l'espace mémoire .dans ce cas, tout pixel de l'image de

cordonnées (i ,j) sera représenter dans le plans cartésien (x y) par les coordonnées (x_e ,y_e) tel que $\mathbf{x_e = i-T/2}$ et $\mathbf{y_e = j-T/2}$

Avec **T** la taille de l'image.

4.4 Description VHDL

En manipulant le langage de description hardware VHDL , à l'aide de l'outil de conception ISE Foundation**,** nous avons réalisé une bibliothèque contenant tous les éléments nécessaires pour l'élaboration d'une architecture optimale et fonctionnelle de notre algorithme .

Notre bibliothèque est composée de deux parties ''les cellules élémentaires'' et''les cellules complexes'' pour décrire les blocs principaux de l'architeture de notre algorithme. ces blocs sont :

- Bloc de génération des distances **.**
- Bloc de contrôle.
- Bloc de processus de votre.

3.4.1 Bloc génération des distances ρ

a architecture du bloc

le programme **‹‹gener››** dans la descriptionVHDL **,** génère les valeur de **ρ** calculés pour toutes les valeurs de n appartenant à l'intervalle [0 ,K/4]. Ce bloc est constitué de deux sous- blocs, l'un pour le calcul des valeurs initiales et l'autre pour la génération des **ρ** incrémentales. Ces sous –blocs sont.

- Bloc d'entrée.
- Bloc de régistre.

4.4.2 Bloc de contrôle

les signaux de contrôle qui gèrent l'agencement des donnée et tous les processus de calcul sont générés automatiquement à l'intérieur du circuit . ainsi le signal d'horloge et le signal d'initialisation **reset.** L'état du signal de contrôle **Sel** est déterminer par le signal de sortie du bloc de contrôle, ce dernier est réalisé par le programme contrôle. Ce bloc est constitué d'un compteur, de portes **ET** et portes **OR.** A l' état initial le signal d'initialisation du compteur AINIT est égale à **1** et le signal de sortie **s** du compteur est égale à $\mathbf{(0)_{8bits}}$, donc le signal **(s(i) and AINIT)** or **(s(i-1) and**

AINIT) pour **i** allant de **1 à 7**,est égal à zéro. Dés que le **reset** se met à zéro ce signal prendra **1**.

Le signal précédent va commander le signal de contrôle **Sel** puiusque pour **reset=1 Sel** est égale à zéro et pour **reset =0, Sel** prendra la valeur **1.**

4.4.3 Bloc de processus de vote

L'implimentation du processus de vote consiste à implémenter une architecture qui permet de charger les valeurs de θ **,**ρ dans le tableau accumulateur défini ci-dessus .elle calculera alors le nombre de possibilité d'avoir une valeur ρ_i correspondant à un angle θ_j **,** ce qui reparsent le nombre de points qui contituent la droite définie par les paramètres ρ_i **,** θ_j **.**

Pour notre algorithme, nous allons implémenter de la figure (4.4), cette dernière est constituée d'une mémoire RAM initialisation mise à zéro et un accumulteur ACUM.

On combine ρ_i **,** θ_j pour adresser la mémoire (adress bus), alors dés qu' on génère une valeur ρ_i ,on obtent l'adresse correspondante au couple (ρ_i , θ_j**),** à cette adresse la donnée (Dout) est accumulée dans ACUM et incrémentée **: acum=acum + 1**, puis récrite (Data bus) (Din) à la meme adresse, la mémoire peut basculer entre le mode lecture à l'aide de **W/R** qui sera commander par le bloc de contrôle. (pour les **M** valeurs de ρ_i générer en parallèle, on aura **M** modules à implèmenter).ce processus est répété pour tout les points de l'image, Ainsi à la fin du traitement, la donnée chaque casemémoire adressée par le couple (ρ_i , θ_j**) ,**indiquera le nombre de point qui consistituent la droite de paramètres ρ_i et θ_j**.**

a Bloc adressage

Dans chaque cellule accumulatrice est représentée par les valeur θ_n et ρ_n **,** qui représentera respectivement lesvaleurs discrétisées de θ et de ρ.

Puisque nous avons choisi de représenter chaque cellule par une case mémoire, nous avons alors adressé la mémoire par ρ_n et θ_n **,** ainsi chaque case mémoire adressée par ρ_n et θ_n représentera la celle A(n,q). Sachant que la valeur n indiquera la colonne tableau et le **q** représente la ligne tel que $\theta_n = n\varepsilon$ et **q=** $(\rho_n + R)/\rho_k$.

avec ε **:**la résolution de θ**.**

ρ_k *:* résolution de ρ**.**

R *:* la moitié de la diagonale.

Dans ce cas **M=4 , K=48** et ρ_k **=1** nous allons implémenter quatre mémoires, pour charger les résultas quatre valeurs générées en meeme temps. Nous avons utilisé les mémoires disponibles dans la virtex II, le nombre des lignes d'adresses est donné parla valeur n_{mex} **,** q_{max} avec la donnée **Max** de la valeur de **n.**

Avec n_{mex}**=K/M et** q_{max}**=2R/** ρ_k **=2R** qui correspond à ρ_{max} **=R=**$\sqrt{2}$ **T.** (la taille de l'image).

Pour une images carrée de **256*256,** la droite la plus longe est la diagonale, elle contient **362** points, donc la donnée de la mémoire sera sur **9** bits et le nombre de lignes d'adresses est de **8688** lignes **(**n_{max} , q_{max}**),** donc l'adresse sera codée sur **14** bits. le programme **''B_adressage ''** réalisé cet adressage. différents éléments du bloc d'adressage .

a Bloc_ Accumulateur

Le processus de vote est assuré par le Bloc_ Accumulateur. Ce dernier consiste à calculer le nombre de fois d'apparition de la valeur (ρ, n). Pour cela, nous avons utilisé les valeurs du bloc_ Registre comme adresse et quatre mémoires initialisés à zéro. A chaque fois que cette adresse apparaisse, la donnée est accumulée: **acum= acum+1.**

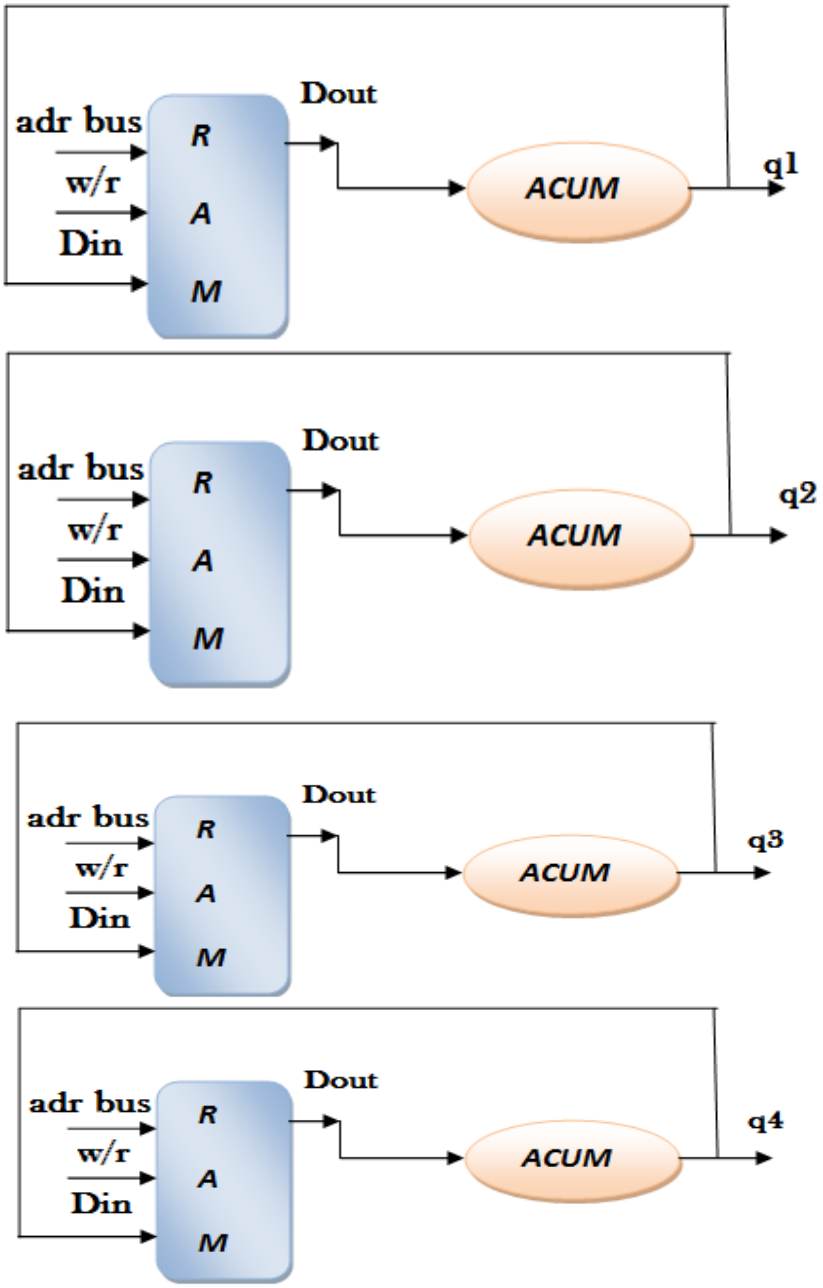


Figure 4.4: Architecture du bloc d'accumulateur

4.5 Synthèse et simulation

4.5.1 Résultat de simulation du bloc de génération des ρ_n et θ_n

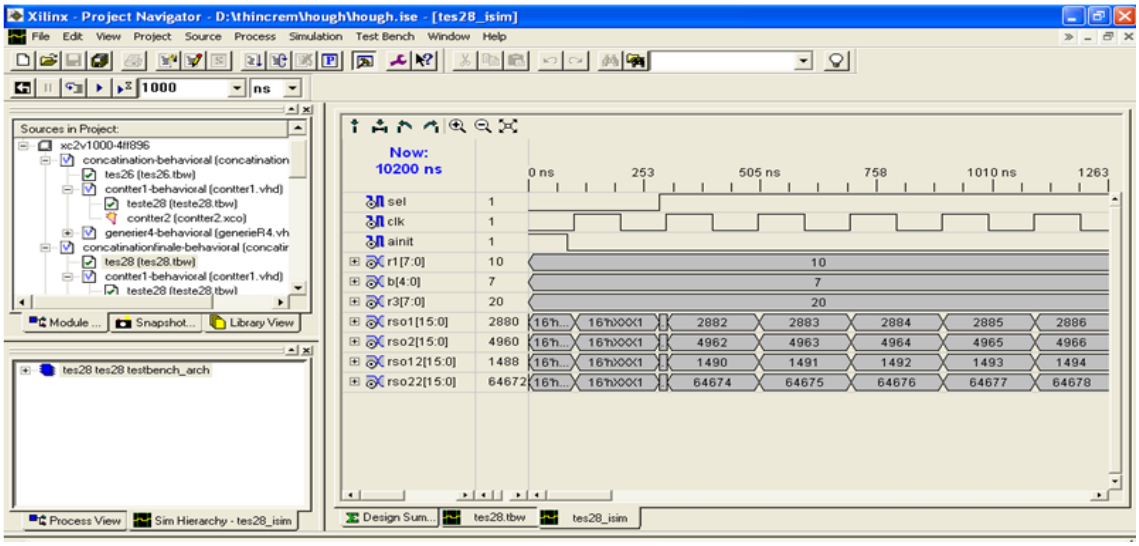


Figure 4 .5 **:** Chronogramme du bloc de génération des ρ_n et θ_n .

4.5.2 Résultat de simulation de bloc_Accumulateur

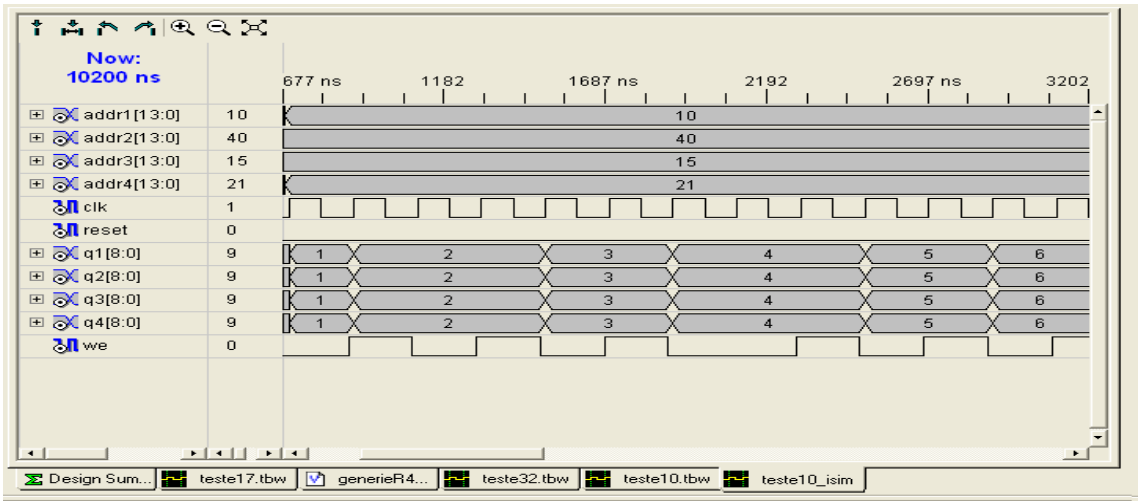


Figure 4.6 : Chronogramme du Bloc_ Accumulateur.

4.6 Conception de l'architecture globale de TH circulaire

4.6.1 Description de l'unité CORDIC

L'algorithme CORDIC 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. C'est une méthode simple et efficace pour le calcul d'une gamme de fonctions complexes qui s'appuie sur une technique d'additions et décalage entre vecteurs. L'algorithme calcule, par approximation de la plupart des fonctions trigonométriques. Il exécute des rotations sans utiliser d'opérations de multiplication. Un autre avantage de cet algorithme est que sa précision dépend du nombre d'itérations.

4.6.2 CORDIC modifié

Afin d'adapter le principe de CORDIC pour le calcul de la TH, plusieurs architectures ont été proposées [27], [28] et [29]. Dans [30] l'architecture de la TH pour la détection de ligne basé sur le principe de CORDIC à été implémenté. Sous cette base, nous avons proposé une architecture pour la THC. Le principe du CORDIC modifié est le suivant:

L'équation d'un cercle peut être définie comme:

$$(\boldsymbol{x}-\boldsymbol{a})^{\boldsymbol{2}}+\ (\boldsymbol{y}-\boldsymbol{b})^{\boldsymbol{2}}=\ \boldsymbol{r}^{\boldsymbol{2}} \qquad (8)$$

avec

x et **y**: sont les coordonnées d'un point sur le cercle dans le plan **(x y).**

r : le rayon du cercle.

a et **b :** représentent les coordonnées du centre du cercle.

Tous les points situés sur le même cercle donneront la même valeur du rayon pour différents**.** Considérons les coordonnées du centre du cercle à l'origine $(\boldsymbol{0},\boldsymbol{0})$ (Figure4.7) **,** l'équation (8) devient:

$$\boldsymbol{x}^{\boldsymbol{2}}+\ \boldsymbol{y}^{\boldsymbol{2}}=\ \boldsymbol{r}^{\boldsymbol{2}} \qquad (9)$$

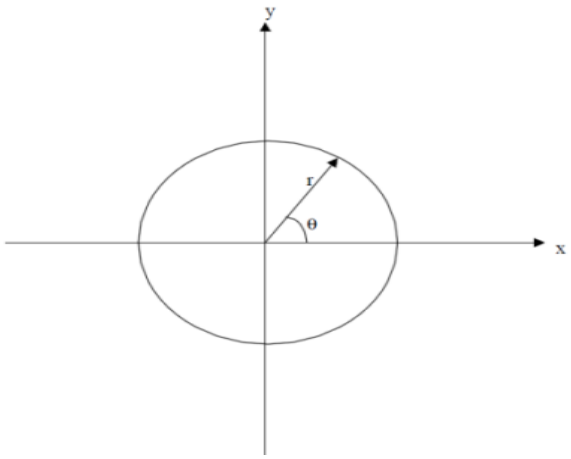


Figure 4.7: Représentation paramétrique d'un cercle.

Les coordonnées du vecteur r sont exprimées selon les équations:

$$\mathbf{x_r} = \mathbf{x}\cos(\boldsymbol{\theta}) - \mathbf{y}\sin(\boldsymbol{\theta}) \qquad (10)$$

$$\mathbf{y_r} = \mathbf{y}\cos(\boldsymbol{\theta}) + \mathbf{x}\sin(\boldsymbol{\theta}) \qquad (11)$$

où $\boldsymbol{\theta}$ est l'angle formé par le rayon et l'axe des **x.**

Les équations (10) et (11) sont exactement similaires à l'équation (Annexe) et peuvent être implémentées en utilisant le CORDIC. A partir de l'équation (Annexe), on obtient:

$$\begin{cases} x_a = x\cos\theta - y\sin\theta \\ y_a = x\sin\theta + y\cos\theta \end{cases} \qquad (12)$$

L'équation (12) montre que l'algorithme CORDIC permet le calcul des coordonnées du point (x_a, y_a) et de son argument $\pi/4$.

Le balayage de $\boldsymbol{\theta}$ sur l'intervalle $[\mathbf{0}, \mathbf{2\pi}]$ peut-être divisé en huit sous-espaces **a**$(\boldsymbol{\theta} \in [\mathbf{0}^\circ, \mathbf{45}^\circ])$, **b**$(\boldsymbol{\theta} \in [\mathbf{45}^\circ, \mathbf{90}^\circ])$, **c**$(\boldsymbol{\theta} \in [\mathbf{90}^\circ, \mathbf{135}^\circ])$, **d**$(\boldsymbol{\theta} \in [\mathbf{135}^\circ, \mathbf{180}^\circ])$, **e**$(\boldsymbol{\theta} \in [\mathbf{180}^\circ, \mathbf{225}^\circ])$, **f**$(\boldsymbol{\theta} \in [\mathbf{225}^\circ, \mathbf{270}^\circ])$, **g**$(\boldsymbol{\theta} \in [\mathbf{270}^\circ, \mathbf{315}^\circ])$ **et** **h**$(\boldsymbol{\theta} \in [\mathbf{315}^\circ, \mathbf{360}^\circ])$. Ainsi, le calcul en parallèle des équations (13), (14), (15), (16), (17), (18), (19), (20) avec$\boldsymbol{\theta} \in [\mathbf{0}^\circ, \mathbf{45}^\circ]$ couvre l'ensemble de balayage de l'intervalle $\boldsymbol{\theta}$. Les coordonnées du rayon **r** dans les différents sous-espaces peuvent être calculés comme suit:

$$\begin{cases} \mathbf{x_a} = \mathbf{x}\cos\boldsymbol{\theta} - \mathbf{y}\sin\boldsymbol{\theta} \\ \mathbf{y_a} = \mathbf{x}\sin\boldsymbol{\theta} + \mathbf{y}\cos\boldsymbol{\theta} \end{cases} \qquad (\boldsymbol{\theta} \in [\mathbf{0}, \mathbf{45}^\circ]) \qquad (13)$$

$$\begin{cases} \sqrt{\mathbf{2}}\,\mathbf{x_b} = \mathbf{x'_b} = \mathbf{x_a} - \mathbf{y_a} \\ \sqrt{\mathbf{2}}\,\mathbf{y_b} = \mathbf{y'_b} = \mathbf{x_a} + \mathbf{y_a} \end{cases} \qquad (\boldsymbol{\theta} \in [\mathbf{45}^\circ, \mathbf{90}^\circ]) \qquad \mathbf{(14)}$$

$$\begin{cases} \mathbf{x_c} = -(\mathbf{x}\sin\boldsymbol{\theta} + \mathbf{y}\cos\boldsymbol{\theta}) = -\mathbf{y_a} \\ \mathbf{y_c} = (\mathbf{x}\cos\boldsymbol{\theta} - \mathbf{y}\sin\boldsymbol{\theta}) = \mathbf{x_a} \end{cases} \quad (\boldsymbol{\theta} \in [\mathbf{90}^\circ, \mathbf{135}^\circ]) \qquad (15)$$

$$\begin{cases} \sqrt{\mathbf{2}}\,\mathbf{x_d} = \mathbf{x'_d} = -(\mathbf{x_a} + \mathbf{y_a}) = -\sqrt{\mathbf{2}}\,\mathbf{y_b} \\ \sqrt{\mathbf{2}}\,\mathbf{y_d} = \mathbf{y'_d} = \mathbf{x_a} - \mathbf{y_a} = \sqrt{\mathbf{2}}\,\mathbf{x_b} \end{cases} \quad (\boldsymbol{\theta}\,[\mathbf{135}^\circ, \mathbf{180}^\circ]) \qquad (16$$

$$\begin{cases} \mathbf{x_e} = -\mathbf{x_a} \\ \mathbf{y_e} = -\mathbf{y_a} \end{cases} \quad (\boldsymbol{\theta} \in [\mathbf{180}^\circ, \mathbf{225}^\circ]) \qquad (17)$$

$$\begin{cases} \mathbf{x_f} = -\mathbf{x'_b} \\ \mathbf{y_f} = -\mathbf{y'_b} \end{cases} \quad (\boldsymbol{\theta} \in [\mathbf{225}^\circ, \mathbf{270}^\circ]) \qquad (18)$$

$$\begin{cases} \mathbf{x_g} = -\mathbf{x_c} \\ \mathbf{y_g} = -\mathbf{y_c} \end{cases} \quad (\boldsymbol{\theta} \in [\mathbf{270}^\circ, \mathbf{315}^\circ]) \qquad (19)$$

$$\begin{cases} \mathbf{x_e} = -\mathbf{x'_d} \\ \mathbf{y_e} = -\mathbf{y'_d} \end{cases} \quad (\boldsymbol{\theta} \in [\mathbf{315}^\circ, \mathbf{360}^\circ]) \qquad (20)$$

Les coordonnées du rayon $(x'_b\,,\ y'_b)$ et $(x'_d\,,\ y'_d)$ dans les sous-espaces **b** et **d** sont considérées comme des paramètres modifiés. Il peut être observé que seulement l'équation (13) est suffisante pour calculer toutes les autres équations en utilisant le principe du CORDIC.

Les équations (14), (15) et (16) peuvent être dérivées de (13) par simple addition et soustraction. Les quatre autres équations peuvent être directement déterminées en ne changeant que les signes des équations (17), (18), (19) et (20).

4.6.3 Algorithme de la transformée de Hogh circulaire modifié

Soit $\mathbf{p} \in \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \dots, \mathbf{N}\}$ tel que p: représente l'indice de la micro-rotation et **N :** nombre total d'itération. On choisira **N= 25.**

Soit $\mathbf{R} \in \{\mathbf{4}, \mathbf{5}, \mathbf{6}, \dots, \mathbf{S}\}$tel que **R :** est le rayon et **S** la taille de l'image**/2**.

Soit $\mathbf{q} \in \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \dots, \mathbf{M}\}$ tel que **q :** représente l'indice du registre. On prendra **M=8.**

$\boldsymbol{\theta_0}$ est l'angle de rotation introduit pour chaque itération où $\boldsymbol{\theta_0} = {}^{\boldsymbol{\pi}}/_{\mathbf{4}}$.

I. Initialisation du tableau accumulateur à zéro.

II. Pour chaque valeur du rayon.

Pour **P** allant de **1 à N.**

a) Calculer en parallèle

$$\begin{cases} x_{pSa} = x(1 - 2^{-11}) - y\,2^{-5} \\ y_{pSa} = y(1 - 2^{-11}) + x\,2^{-5} \end{cases}$$

$$\begin{cases} x'_{pSb} = x_{pSa} - y_{pSa} \\ y'_{pSb} = x_{pSa} + y_{pSa} \end{cases}$$

$$\begin{cases} x_{pSc} = -y_{pSa} \\ y_{pSc} = x_{pSa} \end{cases}$$

$$\begin{cases} x'_{pSd} = -(x_{pSa} + y_{pSa}) \\ y'_{pSd} = x_{pSa} - y_{pSa} \end{cases}$$

b) Calculer en parallèle

$$\begin{cases} x_{pSe} = -x_{pSa} \\ y_{pSe} = -y_{pSa} \end{cases}$$

$$\begin{cases} x_{pSf} = -x_{pSb} \\ y_{pSf} = -y_{pSb} \end{cases}$$

$$\begin{cases} x_{pSg} = -x_{pSc} \\ y_{pSg} = -y_{pSc} \end{cases}$$

$$\begin{cases} x_{pSh} = -x_{pSd} \\ y_{pSh} = -y_{pSd} \end{cases}$$

c) Stocker les coordonnées dans les **M** registres référence.

III. Pour chaque point contour(x_c, y_c).

a) Calculer

$$\begin{cases} a = x_c + x_{pSM} \\ b = y_c + y_{pSM} \end{cases}$$

b) Stocker dans le tableau accumulateur, puis incrémenter la cellule correspondante.

IV. Chercher le pic dans le tableau accumulateur qui définit les coordonnées du centre du cercle recherché.

4.6.4 Description de l'architecture globale

L'architecture que nous avons proposée à base d'opérateurs CORDIC modifiés est composée de quatre définis comme suit:

1. Bloc_ Génération.
2. Bloc_ Registre.
3. Bloc_ Contrôle.
4. Bloc_ Accumulateur.

L'architecture globale est montrée par la figure 4.8

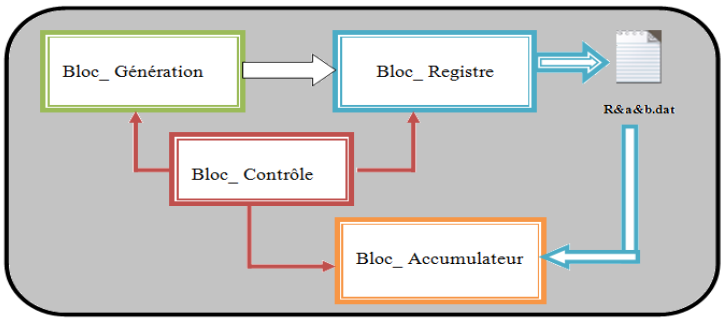


Figure 4.8 : Architecture globale.

Cette architecture reçoit en entrées le signal Reset, l'horloge CLK et les rayons. La donnée à l'entrée est codée sur **10** bits et le résultat en sortie est codé sur **46** bits.

a Bloc_ Génération

Le Bloc_ Génération permet de générer en parallèle **N** micro-rotations pour chaque intervalle $[\theta_o, \theta_0 + \pi/4]$. Ceci équivaut au calcul de tous les cercles possibles dont les coordonnées du centre sont à l'origine $(0, 0)$.

Ce bloc reçoit en entrée:

- L'horloge **CLK.**
- Le signal **CE** qui est actif à l'état haut pour l'initialisation (signal clock Enable).
- Le signal **Load** actif à l'état haut, permet de contrôler le chargement parallèle des rayons. Ces derniers permettent de calculer les différentes micro-rotations qui vont être stockées par la suite dans le Bloc_registre.
- **R [9 : 0].**
- **Y [9 : 0].**

En sortie, ce bloc génère **8** valeurs codées sur **46** bits ($\mathbf{x_{pS}}$ concaténé à $\mathbf{y_{pS}}$).

Pour l'initialisation on pose **X=R** et **Y=0**.

Ce bloc est constitué de:

a) Une unité CORDIC micro-rotation.

b) Un MUX50-2.

c) Une unité de liaison.

L'architecture de ce bloc est illustrée dans la figure (3.5).

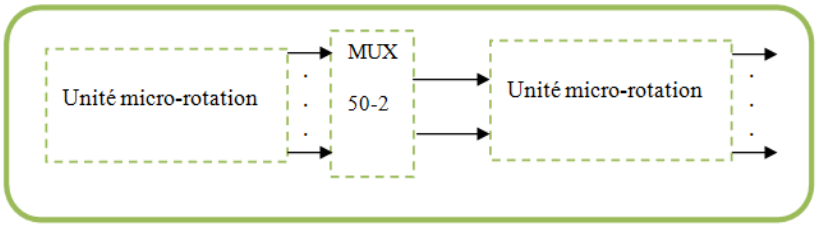


Figure 4.9 : Architecture du Bloc_ Génération.

a.1 Unité micro-rotation CORDIC

Cette unité est constituée de **N** bloc CORDIC. Ces derniers sont implémentés en cascade figure 3.10.

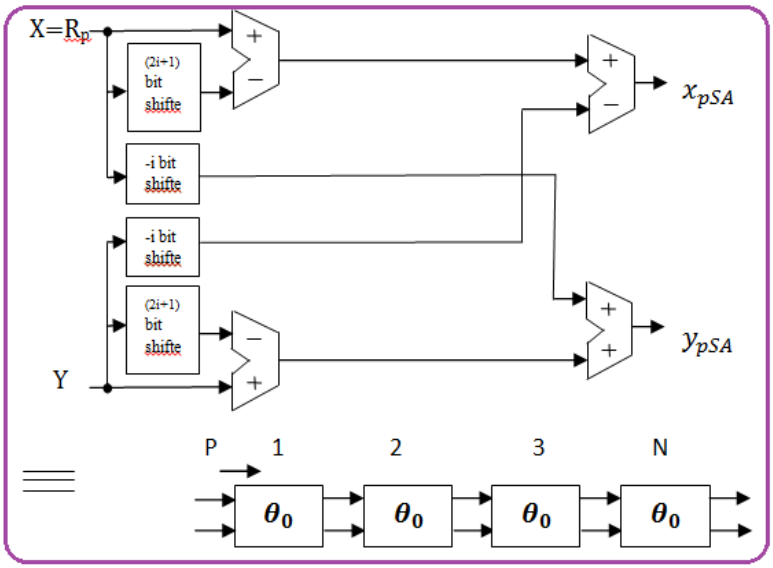


Figure 4.10 : Schéma synoptique de l'unité micro-rotation CORDIC.

a.2 MUX50-2

La fonction du MUX50-2 est la gestion des entrées $\mathbf{x_{pS}}$ **et** $\mathbf{y_{pS}}$ codées sur **23** bits par un sélecteur *Select*. Les variations de ce signal sont générées par un compteur codé sur **5** bits, comme illustré dans la figure 3.11:

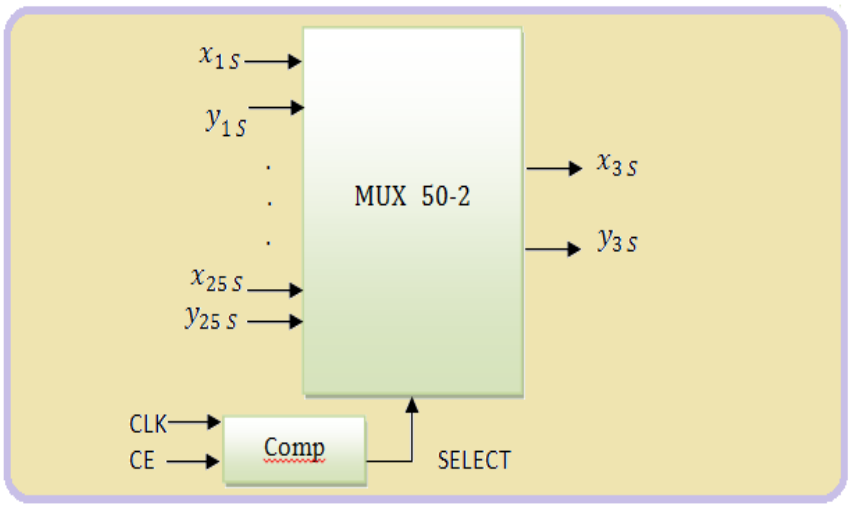


***Figure 4.11*:** Schéma synoptique du MUX 50-2.

a.3 Unité de liaison

Les couples $\mathbf{x_{pS}}$ et $\mathbf{y_{pS}}$ générés par MUX50-2 sont les entrées de l'unité liaison. Elle est constituée de deux modules:

a.3.1 Bloc_ Logique

Il permet de calculer en parallèle les équations (13), (14), (15), (16), des quatre premiers sous-espaces respectivement **a, b, c** et **d**. Les sorties sont codées sur **23** bits $(x_a, y_a, x_b, y_b, x_c, y_c, x_d, y_d)$

a.3.2 Bloc_ Inverse

Les équations des sous-espaces respectivement **e, f, g et h** sont générées par une simple soustraction par zéro des sorties du Bloc_Logique. Le schéma bloc des deux modules est illustré dans la figure 3.11.

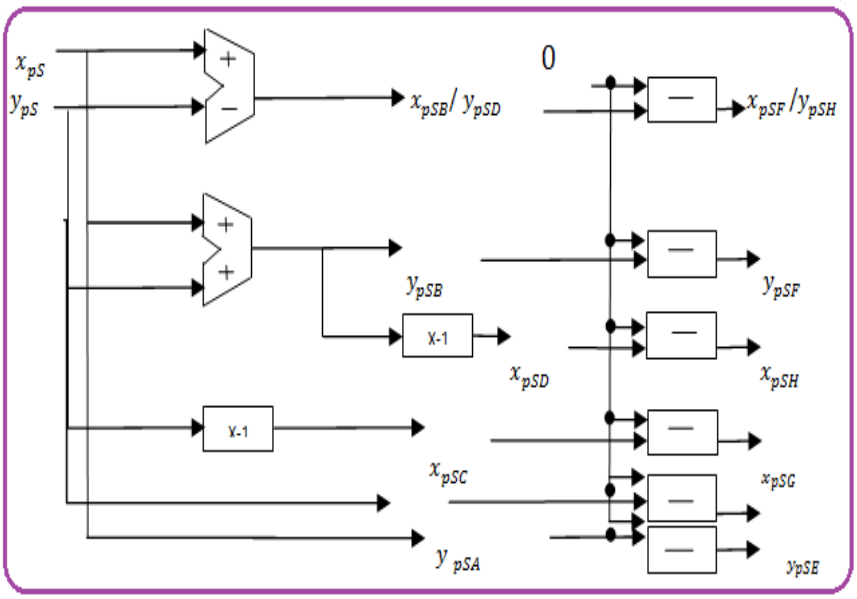


Figure 4.12 **:** architecture du Bloc_ logique _ inverse

Il est à noter que le déroulement des différentes itérations est semblable à la première itération.

b Bloc_ Registre

Nous avons généré dans l'unité de liaison huit valeurs ***(*** $\boldsymbol{x_{pS}}$ concaténé à $\boldsymbol{y_{pS}}$) qui correspondent aux coordonnées des points sur le cercle du rayon choisi à la première itération. Afin d'assurer la synchronisation du traitement, ces valeurs sont stockées dans des registres. Toutes les valeurs possibles des cordonnées des points contours (x_c, y_c) de l'image originale obtenue après traitement (conversion en binaire) et concaténation des deux vecteurs $(\boldsymbol{x_c}, \boldsymbol{y_c})$**,** sont aussi stocké dans un fichier ***data.coe***. Ce dernier sera utilisé pour le chargement de la ROM.

A chaque fois un registre est sélectionné pour additionner sa valeur avec les valeurs de la **ROM**. La sélection du registre se fait par un multiplexeur MUX8-1. Ce bloc est constitué alors de:

1. Une ROM de $\mathbf{13} \times \mathbf{2^{11}}$ bits.
2. Huit registres à ***46*** bits.
3. Un multiplexeur ***8_1***.

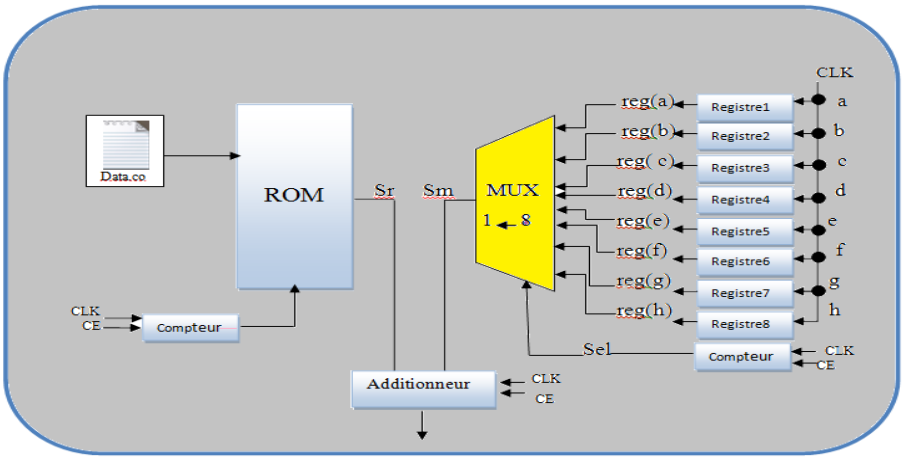


Figure 4.13: Architecture du Bloc_ Registre.

Les résultats d'addition seront stockés dans un fichier texte **R&a&b.dat**. Ce dernier servira d'adresse mémoire pour le bloc accumulateur.

c Bloc_ Contrôle

Ce bloc est conçu pour assurer la synchronisation des signaux de commandes. Il reçoit en entrée **CLK et CE** pour générer en sorties les signaux suivant:

- **Load**: ce signal est activé à l'état haut. Il est utilisé dans le registre d'entrée (les valeurs du rayon). Il est mit à l'état haut pendant un top d'horloge, puis remis à zéro pendant tout le traitement (**25** itérations). Il sera remis à un pour le chargement du prochain rayon.
- **Load 2**: ce signal commande le chargement du registre du bloc liaison. Il est mis à un pendant le chargement, puis remis à zéro jusqu'à la fin du traitement.

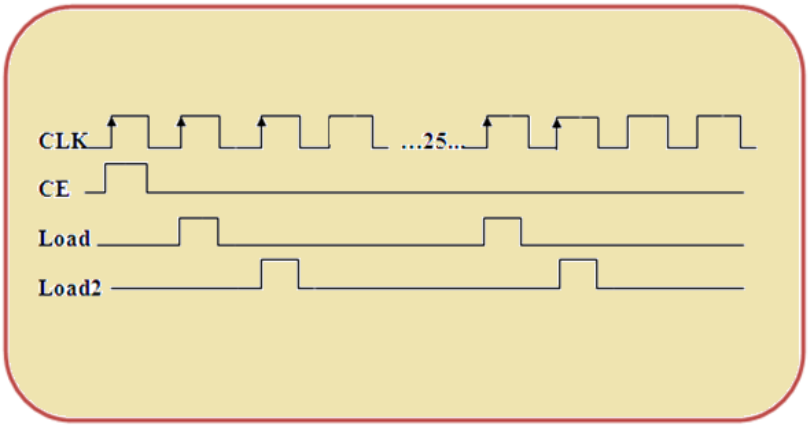


Figure 4.14: Chronogramme du des signaux de commande

d Bloc_ Accumulateur

Le processus de vote est assuré par le Bloc_ Accumulateur. Ce dernier consiste à calculer le nombre de fois d'apparition de la valeur $(\mathbf{R_s}, \mathbf{a_i}, \mathbf{b_i})$. Pour cela, nous avons utilisé les valeurs du bloc_ Registre comme adresse d'une mémoire initialisé à zéro. A chaque fois que cette adresse apparaisse, la donnée est accumulée:

acum= acum+1.

À la fin du processus, la donnée à l'adresse $(\mathbf{R_s}, \mathbf{a_i}, \mathbf{b_i})$ représente le nombre de son apparition.

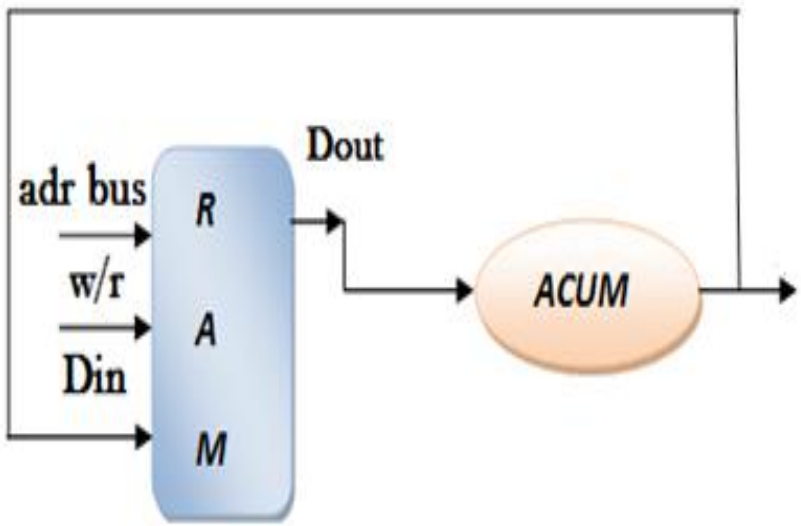


Figure 4.15 : Architecture du Bloc_ Accumulateur.

4.7 Synthèse et simulation

4.7.1 Résultat de simulation de N bloc Cordic

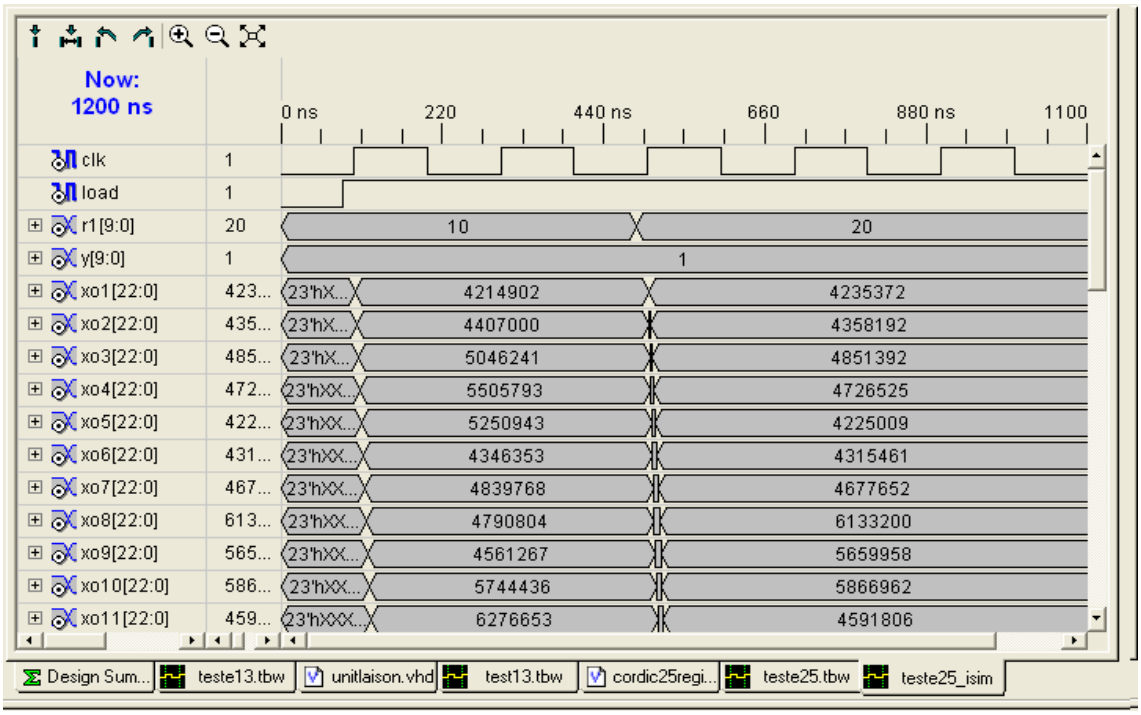


Figure 4.16: chronogramme du N bloc Cordic .

4.7.2 Résultat de simulation du bloc unité de liaison

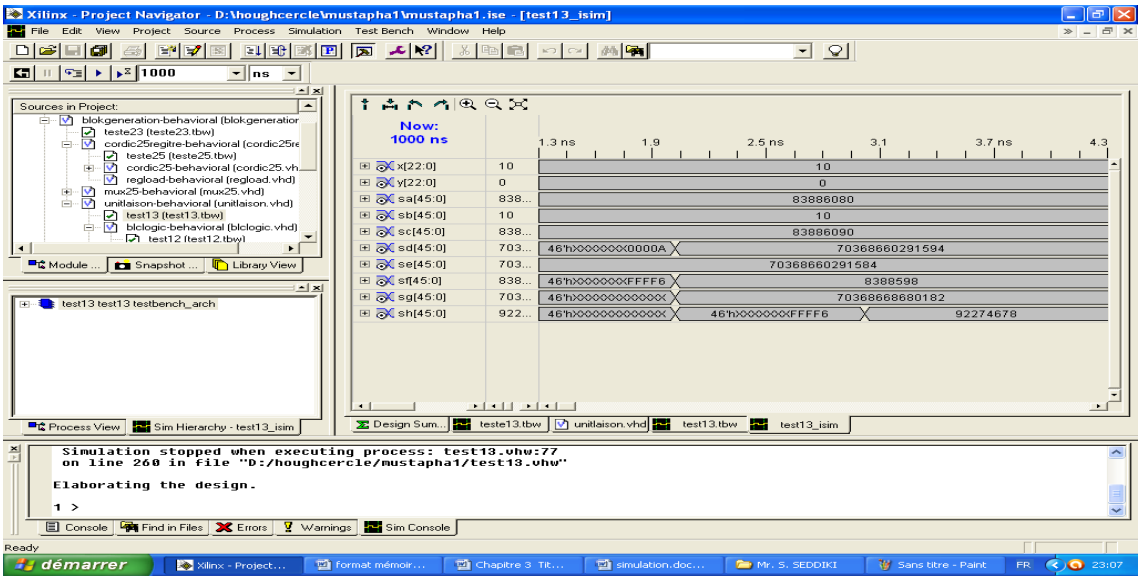


Figure 4.17 **:** chronogramme du bloc unité de liaison.

4.7.3 *Résultat de simulation de bloc_ Accumulateur*

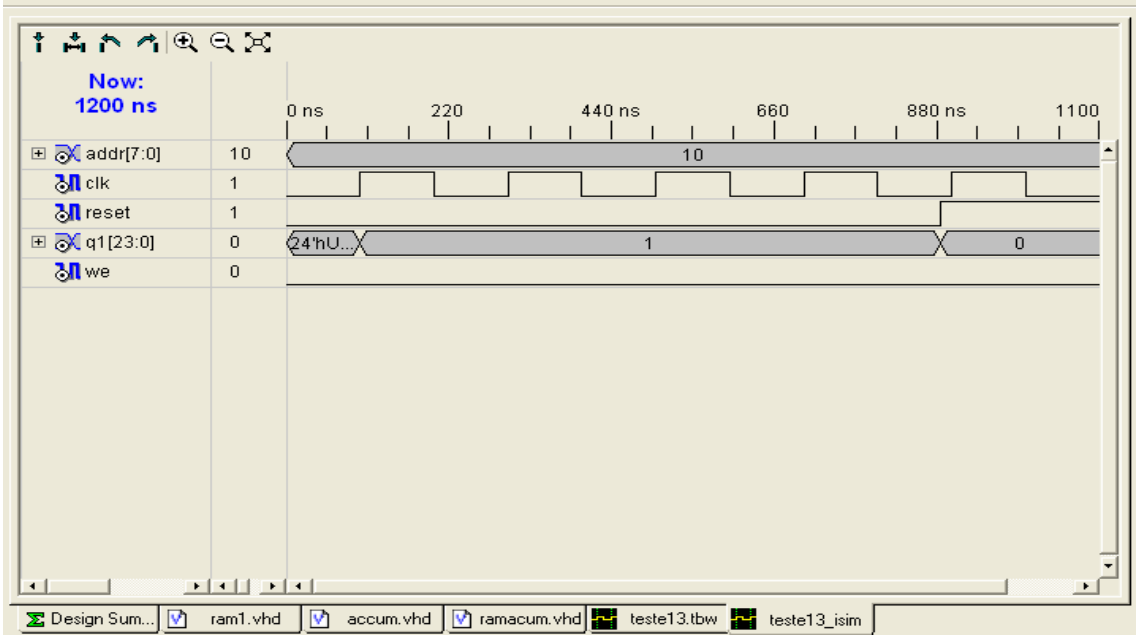


Figure 4.18 **:** chronogramme du bloc accumulateur.

4.8 Conclusion

Nous avons conçu les architectures pour la détection des formes dans une image, ce chapitre illustre les étapes principales de cette conception et décrit ces architectures an langage VHDL en adoptant. Nous avons également donné les résultats de simulation.

Les architectures ont été implémentés sue le circuit XC2V1000_4fg 456 de la famille VERTEX- II. le choix du circuit revient au faite que nous possédons une carte d'évaluation V2mb1000, qui possède ce circuit FPGA. les résultat d'implémentations montrent le bon fonctionnement du circuit réalisé.

Bibliographie

[1] M.NixonetA.Aguado : 'Feature extraction and image processing' ,Newness, 2002.

[2] J.Liados : 'The Hough transform as a tool for image analysis', Computer vision master, 2003.

[3] O.Duda et E.Hart : 'Use of the Hough transform to detect lines and curves in picutures' Newman,1972.

[4] W.Lam ,S.Yuen et Leung : 'An analysis on quantizing the Hough space'Elsevier,1994.

[5] k.MESSAOUDI : 'Imeplémentation de la transformée de Hough surXilinix' ,Octobre 2003 .

[6] Radu HORAUD et MONGA : 'Vision par ordinateur : outils fondamentaux ', Edition Hermès ,deuxième édition.

[9] L.S.DAVIS : 'Hierarchiclgenralized Hough and line segment based generalized Hough transform',s.Pattern Recognition ,15,(1982),pp.277-85.

[7] Xu ,L.Oja,E and Kultanen,P : 'A new curve detection : Randomized Hough transform(RTH)',Pattern Recognition ,vol 11 N0 5(1990) pp.331-338.

[8] N.Kiryati and Y.Eldar and . A.M.Bruckstien : 'A Proballisitic hough transform ', Pattern Recognition.24,pp.303-316.

[10] H.Koshimizu, M.Numada : 'FIHT2 Algorithm: a Fast incremental HoughTrasform',In IEICE Transactions,Vol 11 N0 10 , Oct.1991.

[11] S.Tagzout ,K. Achour, O.Djekoune : 'Hough Transform for FPGA implementation' Elsevier Journal, signal Processing, 81,pp.1295-1301.

[12] K. Achour,O.Djekoune : 'incremental Hough Trasnform: An Improvement Algorithmefor Digital Devices implementation',real-time imaging, volume 10 issus 6,December,2004.

[13] H.kälviäinen, N .Kiryati andS.Alaoutinen : 'Randomized Proballisitichough transform Unifed performance Evoluation' proceeding IAPR SCIA,Kanger lussuaq Greenland 1999,259-266.

[18] N. LASSOUAOUI : 'Segmentation des images par différentes approches et optimisation avec les algorithmes génétiques', Thèse de doctorat en électronique, ENP, 2004.

[19] J.P.COCQUREZ et s PHILIPP :' Analyse d'images filtres et segmentation',Edition Masson, Paris, 1995.

[20] Le petit Robert.

[21] AbdelhalimGabis : 'Compression et Segmentation des images ', Institut National de formation en Informatique (I.N.I), promotion 2004/2005.

[22] AKROUR Nawal et CHABI Lilia :'Une plateforme évolutive pour le développement d'approches biométriques en segmentation d'images ', Ecole nationale Supérieur d'Informatique (ESI), Promotion 2008/2009.

[23] Sofiane SEDDIKI : ' Implémentation d'une méthode de détection de l'iris sur FPGA', UNIVERSITE SAAD DAHLAB DE BLIDA , Juin 2011

[24] M^{me}Hamidatou eT Hamadi Fatma Zohra : 'Techniques de segmentation d'images par contours actifs et contribution au mouvement dans des séquences d'images', Ecole Nationale Polytechnique, 30 Octobre 2007.

[25] K. BENATCHBA et M.KOUDIL : 'Vérification de l'identité d'une personne par reconnaissance d'iris ', Institue National de formation en Informatique (I.N.I), promotion 2009/2010.

[26] K. MESSAOUDI :' Implémentation de la transformée de Hough sur circuit FPGA de Xilinix ', Octobre 2003.

[27] Radu HORAUD et Olivier MONGA :' Vision par ordinateur: outils fondamentaux', Editions Hermès, deuxième édition.

[28] J. Daugman: 'Biometric Personal Identification System Based on Iris Analysis',US Patent, vol. No. 5,291,560, 1994.

[29] S. Tim et al :'Efficient iris recognition through improvement of feature vector and classifier' ETRI J, vol. 23, pp. 61-70, 2001.

[30] J. Daugman:'High confidence visual recognition of persons by test of statistical independence',IEEE Trans. PAMI, vol. PAMI-15, pp. 1148-1161, 1993.

[31] D.Brlhandouz et A.Guessoum : ' Modélisation d'algorithmes machines de calcul en mode on line et halfa line ',mémoire de fin d'étude,Intitude D'électronique,USTHB 1984.

[32] S.Tagzout : 'Contribution à l'implémentation de la transformée de Hough sur un FPGA de Xilinx', Thèse de magister,Institut D'éléctronique,USTHB ,Alger ,Algérie,5 novembre 2000.

[33] Vertex II 1.5 V FiedProgrammable GateArrys.M.

[34] D. BENDIB : 'Etude et réalisation d'une commande MLI on-line sur circuit FPGA', Mémoire de Magister en Electronique, Ecole Nationale Supérieure Polytechnique,Juin 2009.

[35] A. Benmosbah : 'Implémentation sur FPGA d'un transpondeur à débit variable', Rapport du stage de fin d'étude, Ecole nationale supérieure des télécommunications, Paris, Septembre 2007.

[36] Dragomir Milojevic : 'Implémentation des filtres non-linéaires de rang sur des architectures universelles et reconfigurables',Thèse de Doctorat en Sciences Appliquées, Université Libre de Bruxelles, 2004.

[37] J P Deschamps, G JAntoineBioul:'Synthesis of Arithmetic Circuits FPGA, ASIC, And Embedded Systems' JOHN WILEY & SONS, 2006.

[38] Bob Zeidman:'Designing with FPGAs and CPLDs',ELSEVIER 2002.

[39] MVD Training : ' Support de cours de formation « Synthèse et simulation VHDL pour la conception de FPGA », 2006.

[40] Virtex™-II Platforme FPGAs: Introduction and Overview ,Xilinx tutorial.

Conclusion générale

Le travail présenté dans ce mémoire apporte une contribution dans la segmentation des images qui est celui de l'implémentation d'une méthode de détection des formes sur un circuit programmable. Les algorithmes de la TH ligne et circulaire ont été testé C++Builder et implémenter sur un circuit FPGA.

Nous avons étudié la transformée de Hough en détails, et en particulier la TH ligne et circulaire. L'application de cette transformée, ne peut se faire que sur des images contours, pour cela nous avons utilisé le filtre de Canny. Nous avons développé sous C++ Builder un programme pour la segmentation des formes linéaire et circulaire. Ce programme comporte une étape de prétraitement pour l'élimination du bruit, une étape de détection de contour par le filtre de Canny et une étape de l'algorithme de la TH (ligne et circulaire). L'excution de ce programme sur des images tests et réelles a donné de bon résultats.

Afin de régler les problèmes du large délai de traitement et l'espace important nécessaire pour l'implémentation de la TH, nous avons présentés deux architecture : la première pour la transformée de Hough incrémentale pour la détection de droites. La deuxième pour la THC à base de l'algorithme de CORDIC. Ce dernier permet de calculer les fonctions trigonométriques à base d'opérations simple ce qui facilite l'implémentation matérielle.

Les architectures développées ont été décrite par un langage de description hardware VHDL, sous l'environnement ISE 7.1i de Xilinx. Après, une étape de synthèse par l'outil XST et la simulation par ISE simulator, nous avons implémenté les architectures développées.

En perspectives, il serait intéressant d'implémenter un système pour la détection de plus formes paramétrique. Ce système peut être utilisé dans des applications qui nécessitent une détection de forme telle que la segmentation de l'iris, la détection d'une zone d'intérêt dans les images médicales, et la détection des obstacles, etc.

Annexes

L’algorithme CORDIC

L’algorithme CORDIC est basé sur des calculs de fonctions trigonométriques. Son principe est d’effectuer des rotations sur un vecteur de base pour un angle donné. Supposons la rotation du vecteur *V (x, y)* d’un angle ϕ telle qu’illustrée à la figure V.2. Les coordonnées du vecteur V' sont exprimées sous forme matricielle comme suit:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\,\varphi & -sin\,\varphi \\ \sin\varphi & cos\,\varphi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad \textbf{(V.1)}$$

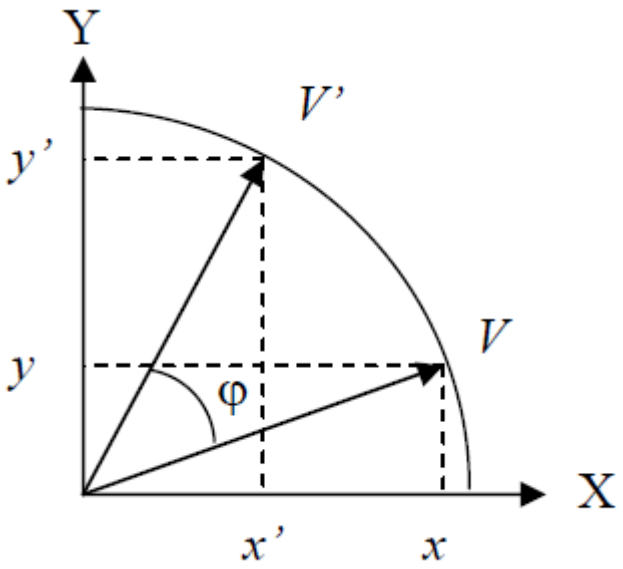


Figure V.2: Rotation du vecteur V dans le plan cartésien.

La rotation totale de l’angle φ peut être exprimée par des micro-rotations α_i tels que:

$$\varphi = \sum_{i=1}^{M} \alpha_i \qquad \textbf{(V.2)}$$

où M est un nombre entier.

L'équation (V.1) peut être calculée en cascade avec un certain nombre de micro-rotations:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \prod_{i=1}^{M} \begin{bmatrix} cos\ \alpha_i & -sin\ \alpha_i \\ \sin \alpha_i & cos\ \alpha_i \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \qquad \textbf{(V.3)}$$

Si les micro-rotations α_i sont très petites, telles que:

$\sin \alpha_i \cong \alpha_i = 2^{-i}$ et $\cos \alpha_i = (1 - 2^{-(2i+1)})$, pour notre cas on prendra i=5.

L'équation (V.3) peut être récrite comme:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \prod_{i=1}^{M} \begin{bmatrix} 1 - 2^{-(2i+1)} & -2^{-i} \\ 2^{-i} & 1 - 2^{-(2i+1)} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \qquad \textbf{(V.4)}$$