الجمهورية الجزائرية الديمقراطية الشعبية

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التعليم العالي و البحث العلمي

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFUQUE

جامعة سعد دحلب بالبليدة

UNIVERSITE SAAD DAHLAB DE BLIDA

كلية العلوم - دائرة الإلكترونيك

FACULTE DES SCIENCES DE L'INGENIEUR

DEPARTEMENT D'ELECTRONIQUE

Mémoire de projet de fin d'études

Pour l'obtention du titre master II microélectroniques microsystème et conception de circuits

Implémentation de la méthode des snakes pour la détection de contour actif sur FPGA

Encadrée par :

N. BOUGHERIRA

L. ABDELLI

Présentée par :

AMIMER KHEIREDINE

BENSETTI NABILA

Septembre 2013

ملخص

الأعمال المنجزة تقدم إسهاما في تنفيذ الكشف عن الحافة علىFPGA في الوقت الحقيقي، يتم التعامل على محورين المنهج التحليلي و منهج التتبع في محيط بالموقع.

درسنا و اقترحنا هيكل لتنفيذ منهج التتبع في محيط بالموقع في الوقت الحقيقي على FPGA

RESUME

Notre travail présente une contribution à l'implémentation sur FPGA de détecteurs de contours en temps réel. Deux axes sont traités l'approche analytique et l'approche du contour actif.

Nous avons étudié et proposé une architecture pour le contour actif temps réel implémentable sur FPGA.

Abstract

Our works presents a contribution to the implementation on FPGA of contour detectors in real time. Two axis are treated analytical approach and active contour approach.

We studied and proposed architecture for real time active contour implementable on FPGA.

Chapitre III

Contribution à l'implémentation du contour actif en temps réel sur FPGA

Chapitre IV

Simulation et résultats de l'implémentation

Remerciements

Nous tenons à remercier avec une chaleur particulière notre promotrice Madame N. BOUGHERIRA, enseignante à l'université de Saad Dahleb (Blida) ; en effet sa passion pour la recherche et l'enseignement, ses encouragements de tous les instants.

Nous tenons à exprimer également notre gratitude à L. ABDELLI copromoteur enseignant à l'université de Saad Dahleb (Blida), pour sa disponibilité, et leurs aides précieuses tout au long de ce travail de recherche.

Nous remercions Madame H.BOUGHERIRA, enseignante à l'université de Saad Dahleb (Blida) pour l'intérêt qu'elle a porté à notre travail.

Nous exprimons nos remerciements aux membres de jury, d'avoir fait l'honneur d'accepter d'examiner ce travail.

Pour finir, nous ne saurions exprimer à quel point nous remercions nos familles pour son soutien et ses encouragements depuis toujours. Ce sont nos parents qui nous ont appris à donner le meilleur et c'est cette exigence de soi qui nous a aidées tout au long de nos études. Nos amis sont également remercier, pour leur soutien inconditionnel, ainsi que l'ensemble des personnes qui, de près ou de loin, ont contribué à la réalisation de ce travail.

Remerciements.
Liste des figures.
Liste des tableaux.
Dédicaces
Table des matières
Introduction général 1

Chapitre I : Circuits logique programmables

Introduction 3

I. Qu'est-ce qu'un circuit programmable 3
II. Codage d'une fonction logique 4
II.1. Sommes de produits, produits de somme et matrice PLA (Programmable Logique Array) 4
II.2. Mémoires 5
II.3. Multiplexeur 6
III. Technologie d'interconnexions 7
III.1. Connexions programmable une seul fois (OTP : One Time Programming) 7
III.1.1. Cellules à fusible 7
III.1.2. Cellules à anti fusible 8
III.2. Cellules reprogrammables 9
III.2.1.Cellule à transistor MOS à grille flottante 9
III.2.2. Mémoires statiques 9
IV. Classification des technologies 10
IV.1. PLD (Programmable Logique Device) 11
IV.2. CPLD (Complex Programmable logiqc Device) 11
IV.3. FPGA (Field Programmable Gate Array) 12
IV.3.1. Les FPGA's de type SRAM 13
IV.3.2. Les FPGA's de type antifuse 14
IV.3.3. Comparaison entre les technologies Anti-fuse et SRAM 14
IV.3.4. Bloc d'entré sortie 16
IV. 4. ASIC (Application Specific Integrated Circuit) 16
V. Comparaison et évolution 17
VI. Les outils de développement 18

Conclusion 20

Chapitre II : Détection de contours

Introduction 21
I. La détection de contour 21
II. Extraction de contours 21
II.1. Les types de Contours 22
II.1.1 Contours de type marche 22
II.1.2. Contours de type crête 23
II.2. Les approches de détection de contour 23
II.2.1. Les approches classiques 23
II.2.2. Les approches analytiques 26
II.2.3. L'approche contour actif 32

Conclusion 40

Chapitre III : Contribution à l'implémentation du contour actif en temps réel sur FPGA

Introduction 41

I. Principe de fonctionnement de la convolution 2D sur FPGA 41
II. Gestion de flux de données 42
II.1. Génération des signaux de contrôle des RAM's 44
II.2. Sélection des RAM's 46
II.3. Ecriture des données sur la RAM 49
II.4. Architecture de calcul de la convolution 50
III. Architecture générale du système de convolution 52
IV. Implémentation de l'algorithme des snakes 52
IV.1. L'énergie interne 54
IV.2. L'énergie externe 54
IV.3. Utilisation des deux énergies 55
IV.4. Architecture proposé pour l'implémentation 55
IV.5. Acquisition d'image 56
IV.6. Acquisition du contour initial 56
IV.7. Sauvegarde des positions des pixels du contour 58
IV.8. Calcul des énergies 60
IV.8.1. Energie interne 60
IV.8.2. Calcul de l'énergie externe 60
IV.9. Observation des pixels 67
IV.10. Vérification de la condition d'arrêt 67
Conclusion 68

Chapitre IV : simulation et résultats de l'implémentation

Introduction 69

I. Présentation de l'environnement de développement XILINX ISE 69
I.1. Les étapes pour l'implémentation d'une spécification HDL sur un FPGA 69
I.2. synthèse et implémentation 72
I.2.1. xilinx Specific file au format .NGC 72
I.2.2. RTL (Registre Transfert Level) Shematic 72
I.2.3.Technologie Schématique 72
I.2.4. Le fichier LOG 73
I.2.5. Le rapport de synthèse 73
I.3. Simulation et résultats d'implémentation de l'algorithme de SNAKE 73
I.3.1. Schéma RTL du circuit détection de contour par masque de laplacien 73
I.3.2. Sauvegarde des positions des pixels du contour 74
Conclusion 82

Conclusion générale 83

REFERENCES

[1]- Weber Jacques, Meaurdre Maurice, *Le langage VHDL, cours et exercices*, paris , Ed Dunod 1997

[2]- Rabasté, Denis, *ASIC et composants à réseaux logique programmables PLA, PLD, CPLD, FPGA*, IUFM d'Aix-Marseille 2002

[3]- Etienne Messerli**,** Yves Meyer, *Cours Système numériques, Tome 1,* Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud suisse. Suisse. Septembre 2010.

[4]- Robert Michel, *Circuits et systèmes intégrés microélectroniques technologie conception*, cours l'Université Montpellier II

[5]- CONRAD Carl *et Alii, La logique programmable : architecture des FPGA et CPLD de conception le langage VHDL*, Paris, Ed Eyrolles, 1997.

[6]- N, Julien, *Circuits logiques programmables*, Cours, université de Morbihan, France. septembre 1999.

[7]-Karabernou ,Si Mahmoud, *FPGA* , cours, Université de BLIDA, Avril 2011

[8]-L.G.Robert. *Machine perception of tree-dimensional solids*.In L Clapp C.Koester J. Tippet, D.Berkowitzand A. Vanderburgh. Optical and electro-optical information processing, pages 159-179. MIT press, Cambridge,1965.

[9]-In B.S.Lipkin and A. Rosenfeld J.M.S. Prewitt. *Object enhancement and extraction*. editors. Picture processing and psychopictorics, pages 75-149.Academic press,New York.1970.

[10]-I.Sobel. *Neighbourhood coding of binary images for last contour following and general array binary processing*. Computer graphics and image processing 8, 127-135 (1978).

[11]-J. Canny,*A Computational Approach to Edge Detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 8, No. 6, Nov 1986.

[12]-J. Shen, S. Castan, *An Optimal Linear Operator for Step. Edge Detection*, IEEE conf Vision Pattern Recogn, pp. 109-114, 1986.

[13]-R. Deriche: *Fast algorithms for low-level vision*: IEEE Trans PAMI-12 pp 78–87, 1990.

[14]-A. Foulonneau, *Une contribution à l'introduction de contraintes géométriques dans les contours actifs orientés région*, Ph.D. thesis, Université Louis Pasteur - Strasbourg I, December2004.

[15]-S. Jehan-Besson, *Modèles de contours actifs basés régions pour la segmentation d'images et de vidéos*, Ph.D. thesis, Université de Nice-Sophia Antipolis, 2003.

[16]-M. Kass, A. Witkin, and D. Terzopoulos, *Snakes : active contour models*, 1st International Conference on Computer Vision, pp. 259-268 (1987).

[17]-C. Epstein et M. Cage. *The curve shortening flow*. Wave Motion :Theory, Modeling, and Computation, Springer-Verlag, 1987.

Introduction

Des technologies de fabrication des circuits configurables sont décrites dans ce chapitre.

On va diviser leur description en deux sections, la première est relative aux dispositifs programmables une seule fois (OTP one time programming) utilisant des fusibles ou des anti-fusibles, la seconde relative aux dispositifs programmables plusieurs fois, utilisant les différentes types d'EPROM ou SRAM.

On tente dans cette partie du mémoire de classifier les circuits configurables afin de pouvoir faire les différences qui les séparent.

Un outil de développement qui permet de programmer ces circuits est présenté à la fin de ce chapitre.

I. Qu'est-ce qu'un circuit programmable

Un circuit programmable est un assemblage d'opérateurs logiques combinatoires et de bascules dans lequel la fonction réalisée n'est pas fixée lors de la fabrication. *Il contient potentiellement la possibilité de réaliser toute une classe de fonctions, plus ou moins large suivant son architecture* [1].

La programmation du circuit consiste à définir une fonction parmi toutes celles qui sont potentiellement réalisables. Comme dans toute réalisation en logique câblée, une fonction logique est définie par les interconnexions entre des opérateurs combinatoires et des bascules (synchrones, cela va presque sans dire), et par les équations des opérateurs combinatoires. Il est important de relever que *ce qui est programmable dans un circuit concerne donc les interconnexions et les opérateurs combinatoires. Les bascules, sont le plus souvent, de simples bascules D, ou des bascules configurables en bascules D ou T.*

Ceci dit, la réalisation d'opérateurs combinatoires utilise des opérateurs génériques, c'est à eux qu'on va s'intéresser par la suite.

Le principe de base des circuits consiste à réaliser des connexions logiques programmables entre des structures présentant des structures de base. Avant tout chose on va rappeler comment coder une fonction logique.

II. Codage d'une fonction logique

La base d'une fonction logique, est toujours une fonction combinatoire. Pour obtenir une fonction séquentielle, il suffira en suite de réinjecter les sorties sur les entrées, ce qui donnera de facto un système asynchrone. Si on souhaite un système synchrone, on intercalera avant les sorties, une série de bascules à front, dont l'horloge commune synchronisera toutes les données et évitera quelques aléas de fonctionnement. En effet, pour coder une fonction combinatoire, trois solutions sont classiquement utilisées.

II.1. Sommes de produits, produits de somme et matrice PLA (Programmable Logique Array)

N'importe quelle fonction peut être codée par une somme de produit, par un produit de somme ou un mélange des deux. On peut immédiatement en déduire une structure de circuits, appelée matrice PLA (Programmable Logique Array). La figure **(**I.1**)** représente une matrice PLA à 4 entrée 4 sortie, constitué d'un réseau ET programmable et un réseau OU programmable.

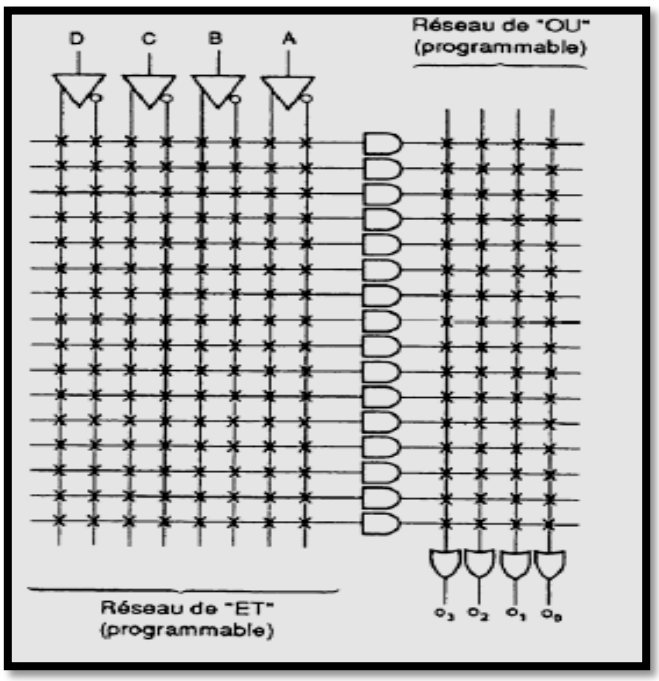


Figure (I.1) : Matrice PLA.

Ce type de structure est utilisé dans certains circuits ASIC (Application Specific integrated circuit) et demande une densité d'intégration importante.

La plupart des applications n'exigent pas une telle complexité et on peut se contenter d'une matrice ET programmable et d'une matrice OU figée. De même, il est peu probable d'utiliser tous les termes produits programmable, appelés au début PAL (Programmable Array Logic), plus communément désigné, aujourd'hui, sous le terme PLD (Programmable Logique Device).

La figure suivante représente schématiquement un PAL.

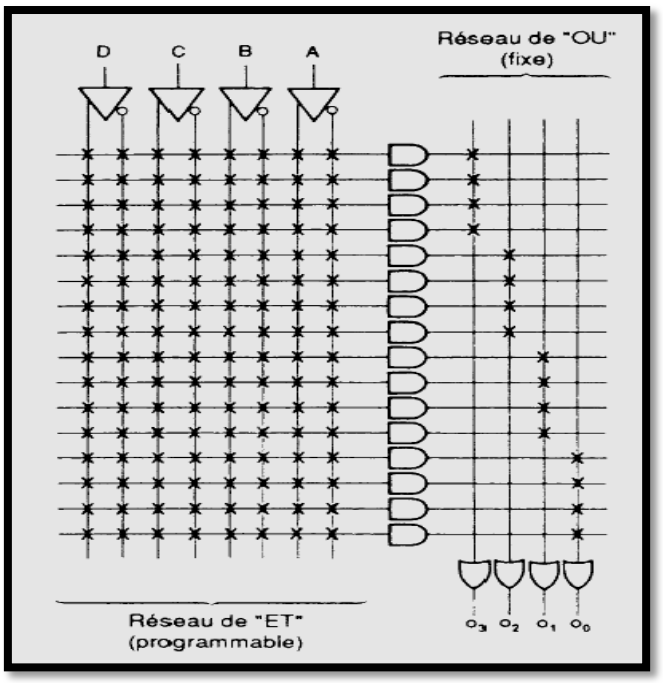


Figure (I.2) : Matrice PAL.

II.2. Mémoires

Une fonction combinatoire associe à chacune de ces combinaisons d'entrée une valeur en sortie décrite par sa table de vérité. C'est le principe de la mémoire ou pour chaque adresse en entrée, on associe une valeur en sortie, sur un ou plusieurs bits

La structure physique des mémoires fait appelle à une matrice PLA dont la matrice ET est figée et sert de décodeur d'adresse et dont la matrice OU est programmée en fonction de la sortie désirée.

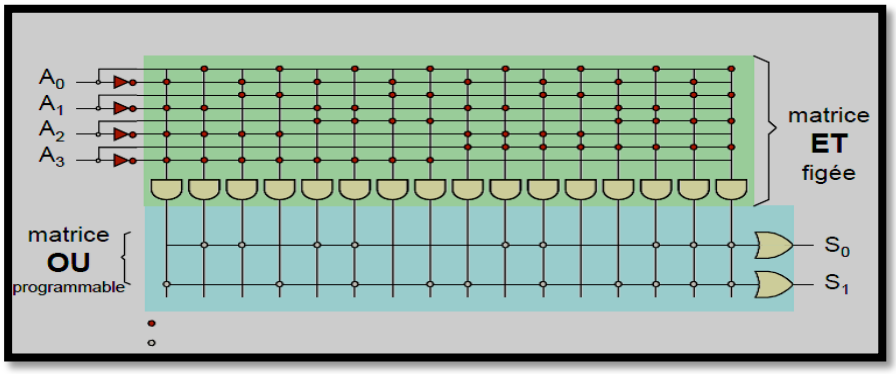


Figure (I.3) : Mémoire réalisée avec une matrice PLA.

II.3. Multiplexeur

Le multiplexeur permet également de coder une fonction combinatoire, comme le montre la figure ci-dessous.

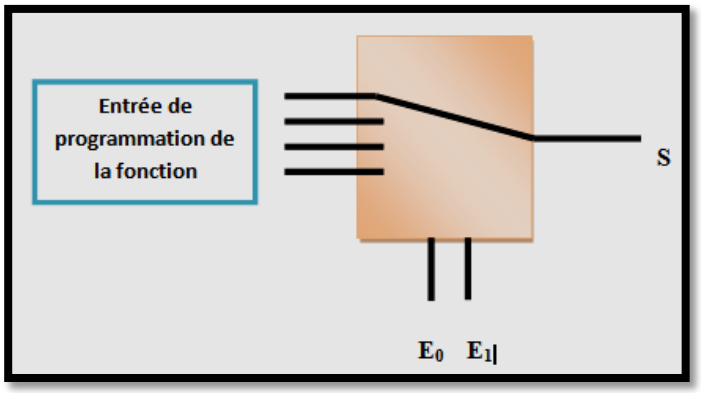


Figure (I.4) : Multiplexeur.

A chaque valeur des entrées de sélection E_0 et E_1, est associé à un niveau logique défini dans la table de vérité pour la sortie S. Ce niveau logique est imposé à l'entrée de la programmation correspondante ; ce principe est utilisé dans les FPGA's.

III. Technologie d'interconnexions

Comme on vient de le voir, l'un des éléments clé des circuits étudié est la connexion programmable. Le choix porté sur technologie dépendra essentiellement :

- La densité d'intégration.
- La rapidité de fonctionnement, une fois le composant programmé, fonction de la résistance à l'état passant et des capacités parasites.
- La facilité de mise en œuvre (programmation sur site, programmation etc.).
- La possibilité de maintien de l'information

Passons en revue, maintenant, quelques technologies classiquement utilisées et leurs caractéristiques.

Ces technologies sont, ou pourraient, être utilisées pour la réalisation de mémoires. Il faut cependant garder à l'esprit qu'hormis quelques cas particuliers (circuit programmés en cours d'utilisation), le temps d'écriture reste secondaire, le circuit étant habituellement programmé une fois pour toute avant utilisation.

III.1. Connexions programmable une seul fois (OTP : One Time Programming)

III.1.1. Cellules à fusible

Ce sont les premières à avoir été utilisées et elles ont aujourd'hui disparu au profit de technologies plus performantes. Leur principe consistait à détruire un fusible conducteur par passage d'un courant fourni par une tension supérieure à l'alimentation.

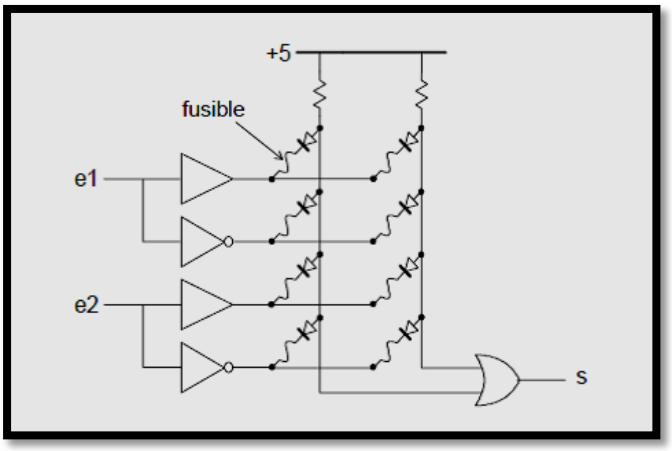


Figure (I.5) : Cellules à fusible.

III.1.2. Cellules à anti fusible

En appliquant une tension importante (16V pendant 1ms) à un isolant entre deux zones de semi-conducteur fortement dopées, ce dernier diffuse dans l'isolant et le rend conducteur. Chaque cellule occupe environ 1.8 um^2 (700 um^2 pour un fusible) ; cette technologie très en vogue permet une haute densité d'intégration.

Hormis la non-reprogrammabilité, c'est la meilleure technologie (vitesse et surtout densité d'intégration).

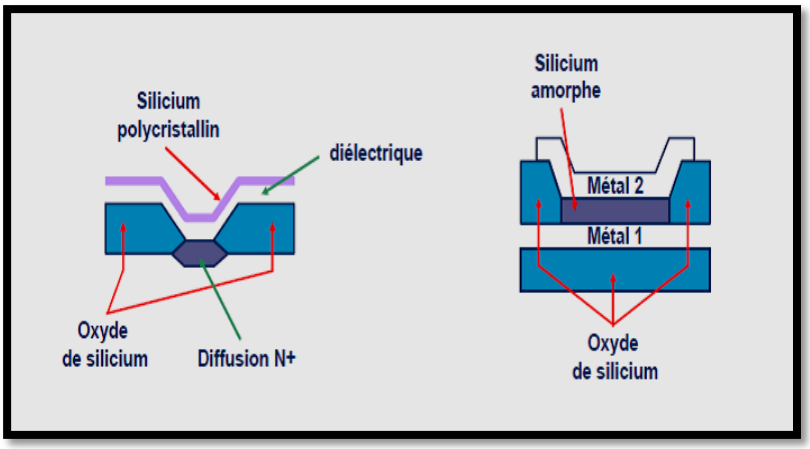


Figure (I.6) : Cellules anti-fusible.

III.2. Cellules reprogrammables

III.2.1.Cellule à transistor MOS à grille flottante

L'apparition du transistor MOS à grille flottante a permis de rendre le composant bloqué ou passant sans application permanent d'une tension de commande. Le principe consiste à piéger ou non (à l'aide d'une tension supérieure à la tension habituelle d'alimentation) des électrons dans la grille.

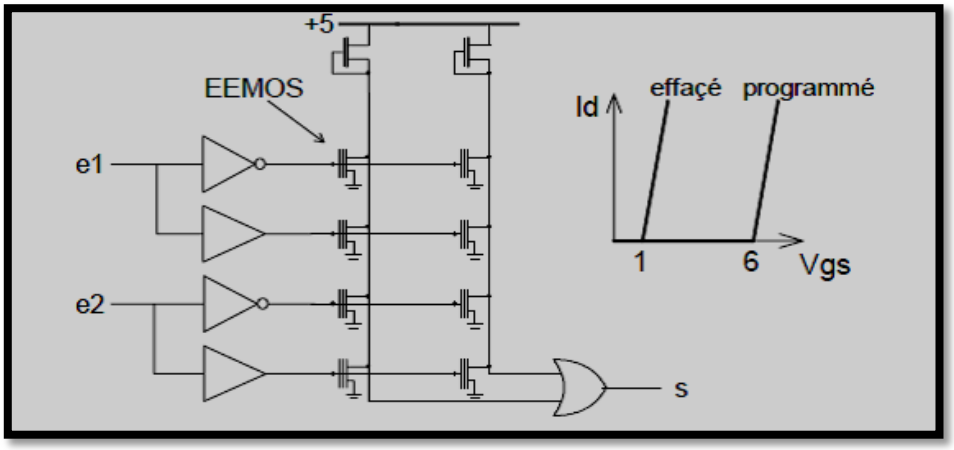


Figure (I.7) : Cellule à transistor MOS à grille flottante.

L'extraction éventuelle des électrons piégés permet le retour à l'état initial, plusieurs technologies EPROM sont en concurrence :

- UV-EPROM : les connexions sont rénitialisable par une exposition à un rayonnement UV.
- EEPROM : l'effacement et la programmation ce fait cette fois électriquement [2].

III.2.2. Mémoires statiques

Dans les circuits précédents, la programmation de l'état des interrupteurs est conservée en absence de tension d'alimentation, fait appel à un mode de fonctionnement électrique particulier. Dans les technologies à mémoire statique, l'état de chaque interrupteur est commandé par une cellule mémoire classique à quatre transistors (plus

un transistor de programmation). Le principe est explicité dans le schéma suivant. Figure (I.8).

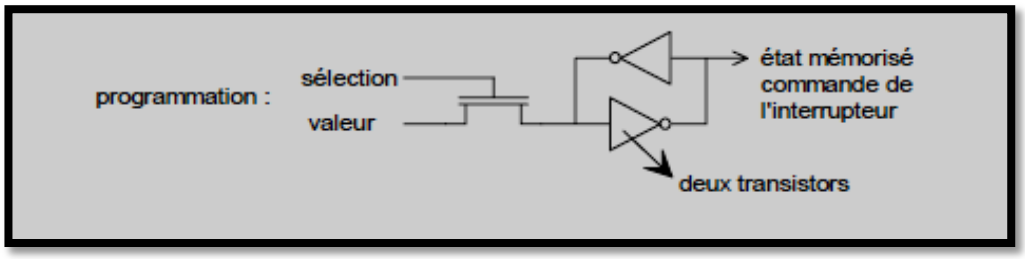


Figure (I.8) : Cellule SRAM

La modification d'un circuit devient alors une opération logique quasi ordinaire, qui ne nécessite pas d'opération électrique spéciale.

Ces circuits permettent des reconfigurations, partielles ou totales, en nombre illimité. Il est même envisageable de créer des fonctions dont certains paramètres sont modifiables en cours de fonctionnement.

Le prix à payer pour cette souplesse et flexibilité est que les cellules SRAM doivent être rechargées à chaque mise sous tension et que chaque interrupteur occupe plusieurs transistors : l'interrupteur lui-même et les transistors de la cellule mémoire [1].

IV. Classification des technologies

On va faire un survol, *ad et nunc*, afin de classifier les circuits programmables, et cela, par leur fonctionnement spécifique et le développement technologique des PLD (Programmable Logic Device).

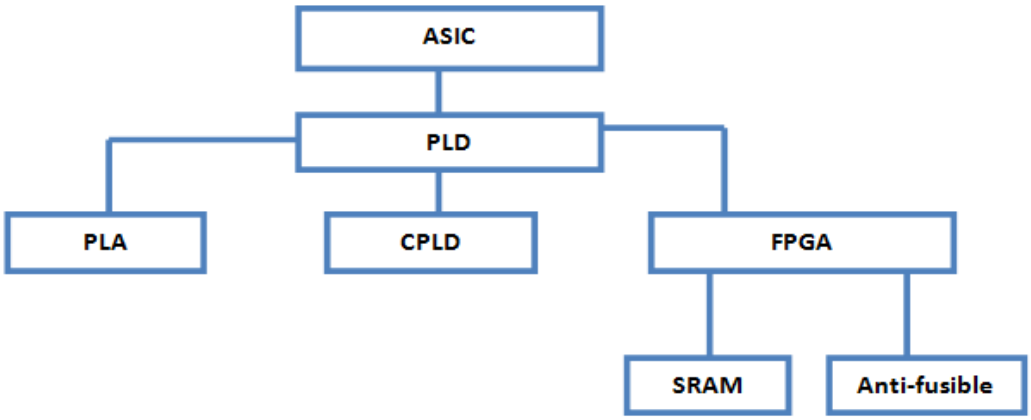


Figure (I.9) : Classification des circuits programmables.

IV.1. PLD (Programmable Logique Device)

Le fait que les PLD soient à la base de la conception des CPLD, très en vogue aujourd'hui, justifie cependant leur étude.

D'abord appelés PAL lors de sa sortie, ce circuit utilise le principe de la matrice PLA à réseau ET programmable. Bien que pas très anciens pour les dernières générations, les PLD ne sont presque plus utilisés pour une nouvelle conception. L'un de leur avantage qui était la rapidité a disparu, et s'en trouvent de facto caducs. Par ailleurs, les efforts de recherche des constructeurs portant plutôt sur les circuits à plus forte densité d'intégration que sont les CPLD et les FPGA.

Les cellules de connexions sont aujourd'hui réalisées en technologie MOS à grille flottante, la structure de base comprend un circuit PLA dont seule la matrice ET est programmable.

IV.2. CPLD (Complex Programmable logiqc Device)

La nécessité de placer de plus en plus de fonctions dans un même circuit a conduit tout naturellement à intégrer plusieurs PLD (bloc logique) sur une même pastille, relies entre eux par une matrice centrale.

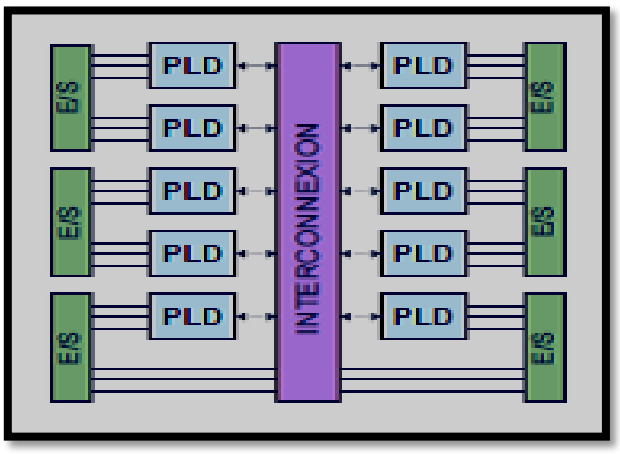


Figure (I.10) : Circuit CPLD.

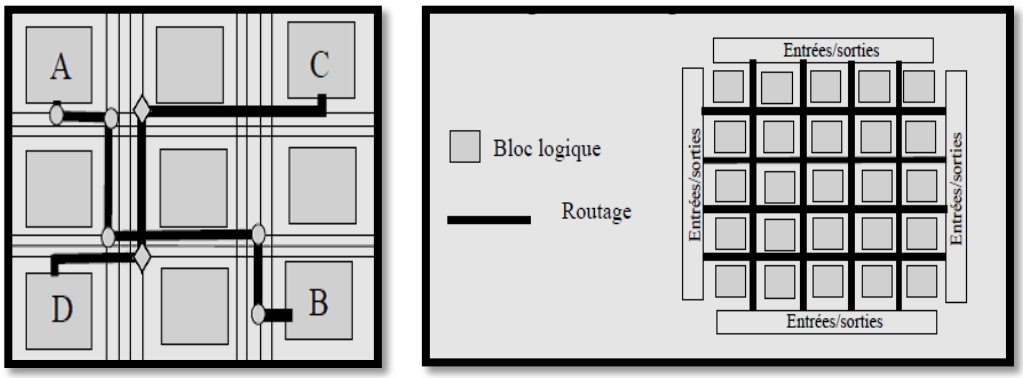
IV.3. FPGA (Field Programmable Gate Array)

Lancé sur le marché en 1984, par la firme **XILINX**, le FPGA (Field Programmable Gate Array) est un circuit pré-diffusé programmable.

Les blocs logiques sont plus nombreux et plus réduits que les CPLD, néanmoins avec les interconnexions entre les blocs logiques décentralisés.

Le passage d'un bloc logique à un autre se fait par un nombre de point de connexion (responsables des temps de propagation) : fonction de la position relative des deux blocs logiques et de l'état d'encombrement de la matrice.

De la phase de placement des blocs logiques et de routage des connexions dépendront, donc, considérablement les performances du circuit en termes de vitesse [3].



(a) **(b)**

Figure (I.11) : (a) Routage des connexions. (b) Blocs logique sous forme d'une matrice.

La technologie utilisée pour le plan de reconfiguration permet de déterminer deux catégories de FPGA :

IV.3.1. Les FPGA's de type SRAM

La structure d'un bloc de base d'un FPGA de type SRAM est complexe. Un registre est généralement associé à chaque LUT [20].

Ces blocs sont programmables (en plus la configuration des LUTs) c'est-à-dire, il est possible de modifier la structure du bloc logique. Les blocs logiques, relativement peu nombreux en comparaison avec FPGAs de type antifuse, sont capables de réaliser une fonction très complexe, un exemple de bloc logique de FPGA de type SRAM est représenté sur la figure (I.12) [4].

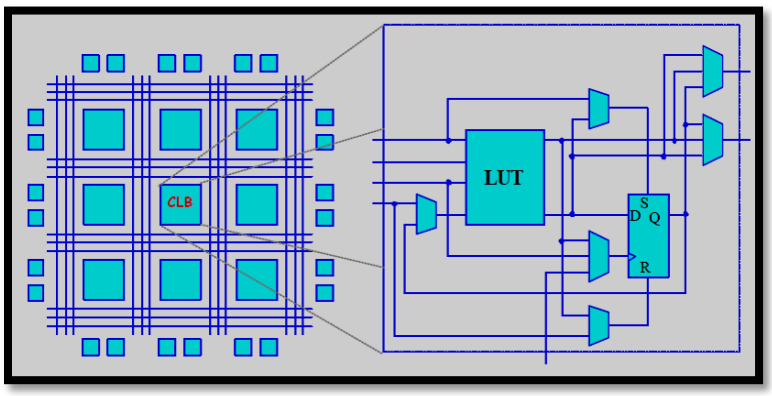


Figure (I.12) : Architecture d'un circuit FPGA- SRAM.

N.B : Le CLB (Configurable Logic Block), qui est au centre des blocs logique, est un bloc logique configurable construit à partir d'un générateur de fonction « LUT :Look Up Tabele».Il permet de réaliser des blocs combinatoires ou séquentiels.

Etant donné le caractère volatile d'une cellule SRAM, le chargement de la configuration est effectué après chaque mise sous tension. Les données de configuration sont disponibles dans un fichier généré par l'outil de développement.

IV.3.2. Les FPGA's de type antifuse

Les blocs logiques des FPGA du type antifuse sont généralement beaucoup moins complexes que ceux des SRAM. La principale différence réside dans le fait que ce type de cellule est OTP (One time programmable). La fonction est physiquement figée par des multiplexeurs. Ce composant est directement fonctionnel à la mise sous tension [4].

On notera que contrairement aux cellules de type SRAM, le nombre de sorties est réduit. Pour une fonction donnée, le nombre de blocs logiques nécessaire est supérieur au nombre de blocs nécessaire dans un FPGA de type SRAM. Conséquence directe de ce constat, le nombre d'interconnexions nécessaire est lui aussi largement supérieur. La compensation réside dans le fait qu'un anti-fusible est moins volumineux qu'une cellule SRAM [5].

IV.3.3. Comparaison entre les technologies Anti-fuse et SRAM

Pour une technologie donnée, il est possible d'établir un comparatif entre les technologies SRAM et anti-fuse. Le tableau ci-dessous en fait la synthèse.

caractéristiques	SRAM	Antifuse
Taille physique du système d'interconnexion	+	-
Taille physique de bloc logique	+	-
Quantité de blocs logiques	-	+
Capacité fonctionnelle d'un bloc logique	+	-
Quantité de canaux de routage	-	+
Efficacité de la technologie (vitesse)	-	+

Tab(I.1) : Comparaison caractéristiques des technologies Anti-fuse et SRAM.

Selon Carl Conrad, *la complexité d'une fonction réalisable au sein d'un bloc logique de type SRAM permet de réduire le nombre de liaisons nécessaires dans un composant*.

Ainsi, le manque d'efficacité en terme de vitesse d'une connexion de type SRAM n'est donc pas un critère trop pénalisant, le souligne l'auteur. En affirmant de même, *que la taille physique occupé par une connexion de type SRAM est importante et limite par la même le nombre de canaux, cette caractéristique n'est pas non plus pénalisante puisque de tout façon un nombre réduit de liaisons est nécessaire*. En d'autres termes dans une technologie de type SRAM la priorité est plutôt donnée aux blocs logiques qu'aux ressources d'interconnexion.

Les composants anti-fuse adoptent le principe contraire, ils exploitent simultanément le fait que la technologie occupe une place réduite sur le substrat et que la résistance engendré par une connexion de type antifuse est relativement faible, la quantité de liaison n'est donc pas un critère trop pénalisant pour les délais. Il en découle l'utilisation possible d'un grand nombre de blocs logiques de taille réduite.

Le terme *granularité* est donc souvent employé pour décrire le compromis (taille/nombre), les FPGAs de type SRAM sont dits à « gros » grain, ceux de type anti-fuse sont dit à grain « fin » [5].

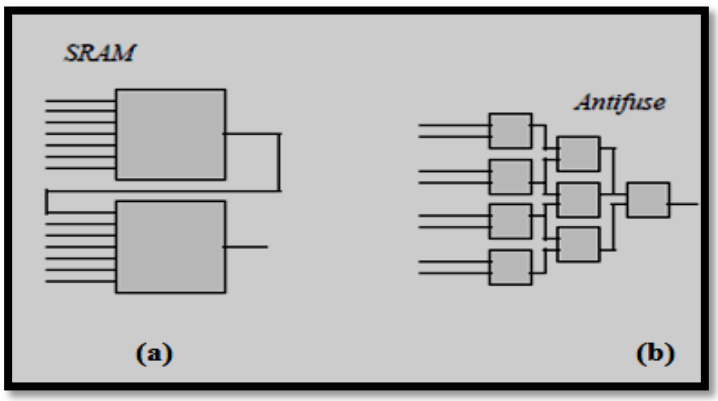


Figure (I.13) : Taille physique des blocs logiques. (a) Type SRAM. (b) Type anti fuse.

IV.3.4. Bloc d'entré sortie

Des blocs d'entrée-sortie (IOB) configurables sont répartis sur toute la périphérie du boitier. Chaque IOB assure l'interface entre une broche d'entrée/sortie du boitier et la logique interne. La figure suivante schématise un bloc d'entrée-sortie d'un FPGA de la famille SPARTAN II [6].

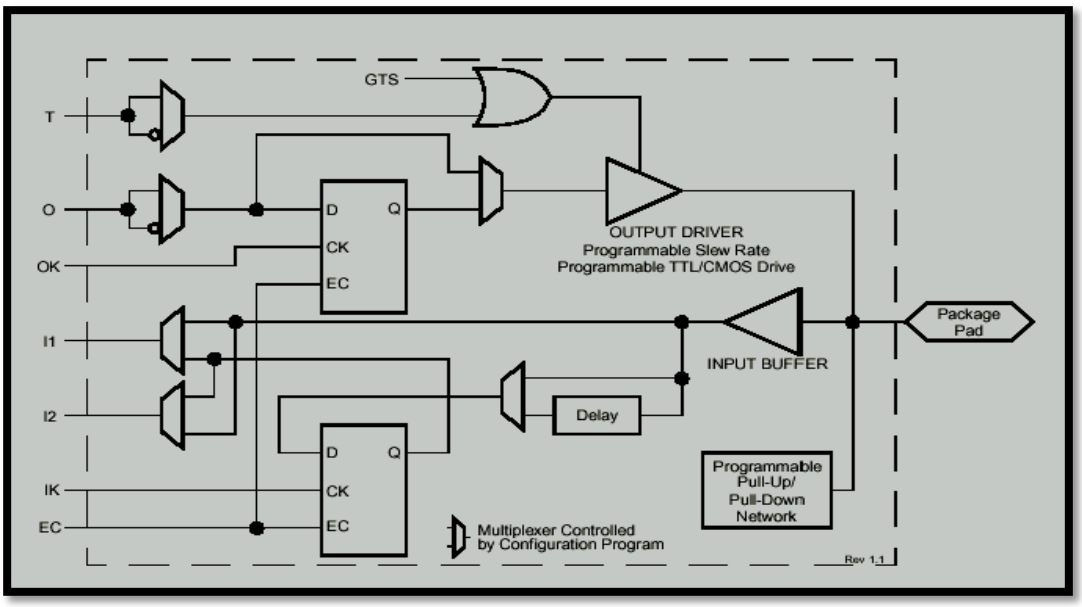


Figure (I.14) : Bloc d'entrée/sortie.

IV. 4. ASIC (Application Specific Integrated Circuit)

Si les composants précédents pouvaient être développés avec un simple ordinateur, ceux qu'on aborde maintenant nécessitent l'intervention d'un concepteur qui produira le circuit demandé à partir des masques fournis par son client [2].

Ici, le terme programmable n'est pas des plus judicieux, les connexions entre les éléments étant dessinées sur les masques.

Le temps et le coût de production sont importants. On distingue quatre types d'ASIC :

- **Les pré-diffusés (gate array)**
  Ils contiennent une nébuleuse de transistors ou des portes à interconnecter avec les problèmes de routage et de délais que cela comporte.

- **Les pré-caractérisés (standard cell)**
  Le circuit est un assemblage de cellules placées routé.

- **Les full customs**
  Ils sont entièrement définissables par le client. Les masques des transistors sont dessinés.

- **Les Embedded Gate array**
  C'est un Gate array avec des macro-blocs complexes (RAM).

V. Comparaison et évolution

Il est ardu de comparer les ASIC et les CPLD\FPGA. Les méthodes et temps de développement ne sont pas du tout les mêmes. L'utilisation des ASICs va surtout être justifiée par la production en grande série du circuit à réaliser. Il n'est pas à omettre que très marginaux en 1990, les CPLD et FPGA ont pris en 2000 près de 95% du marché des ASICs [2].

Les principaux critères de choix seront :

- La vitesse de fonctionnement.
- Le nombre de portes.
- la consommation.
- Le prix.

Caractéristiques	Circuits Configurables		ASIC	
	FPGA SRAM	CPLD	Standard Cell	Full Custom
Densité (portes/m²)	1500 à 6500 0,13µm 90 nm	Faible o,18 µm	Grande 90 nm	Grande 90 nm
Performance (Hrz, Bits/s)	500 MHz 10 Gbits/s	200 MHz	qq Ghz 12 Gbits/s	qq Ghz 20 Gbits/s
Consommation (Watt)	Grande	Grande	Faible	Faible
Flexibilité	Grande	Grande	Aucune	Aucune
Temps de conception	Court	Court	Long	Très Long
Utilisation des outils	Facile	Facile	Complexe	Complexe
Cout de conception	Faible	Faible	Elevé	Très élevé
Volume de production	Faible et moyen	Faible et moyen	> 1 000 000 de pièces	> 1 000 000 de pièces

Tab(I.2) : Comparaison des technologies.

VI. Les outils de développement

Ces outils vont permettre au concepteur de programmer le circuit à partir de la description de la fonction à réaliser. Cette description peut être textuelle (VHDL, Verilog) ou graphique (symboles de fonction, graphes des états, chronogrammes).

Dans cette partie on s'intéresse à la description VHDL, ce langage qui permet la description du circuit et son implantation à l'aide de l'environnement de développement l'ISE 12.3.

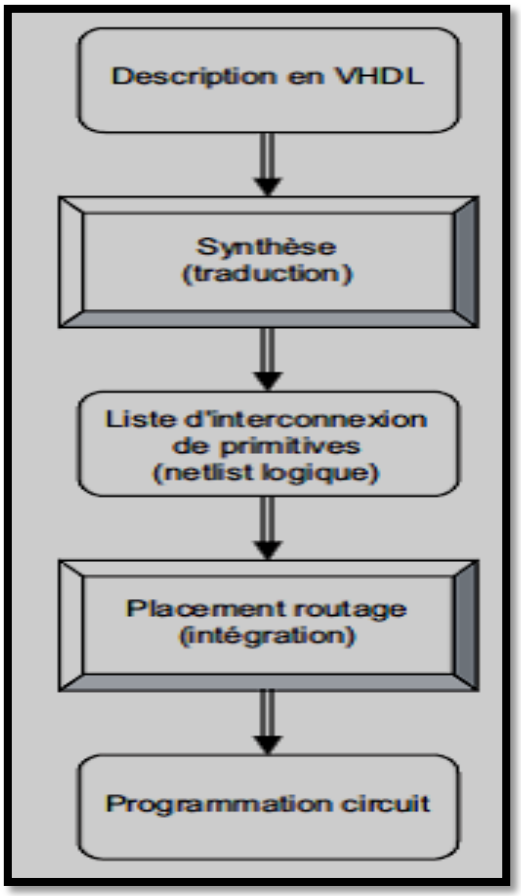


Figure (I.15) : Flot de conception.

- La description textuelle d'un circuit en langage VHDL comporte 3 niveaux de description :
  - Flot de données : exprimer les sorties en fonction des entrées et les opérateurs.
  - Comportementale : utilisation des structures conditionnelles et description séquentielle.
  - Structurelle : description en tenant compte de la structure interne (proche de la réalisation physique) [7].
- La synthèse est l'outil qui permet de transformer une description VHDL d'un circuit en logique.
- Le synthétiseur nous fournit une liste d'interconnexion de primitives.
- placement-routage : ces primitives sont ensuite placées à l'intérieur du composant (opération de placement) puis interconnectées entre elles (opération de routage).

- Vient enfin la programmation du circuit et la vérification du fonctionnement sur la carte. Si la simulation et la vérification ont été faites correctement, aucune erreur de fonctionnement ne doit apparaître [6].

Conclusion

L'intérêt suscité par les FPGAs de type SRAM est dû essentiellement à sa grande flexibilité qui offre l'aspect de la reprogrammation, en comparaison avec les ASICs, ils présentent une faible densité d'intégration de portes logiques et atteignent des fréquences de travail relativement faibles par rapport aux ASIC, on plus ces circuits sont totalement éliminé a cause de leur prix exorbitant et qui ont une architecture prédéfinie par le fabricant, c'est-à-dire l'aspect reconfiguration ne les convient pas.

Le choix du circuit programmable est fait à partir du claquement réalisé sur les différentes architectures des circuits présentés dans ce chapitre.

Le chapitre suivant va traiter les concepts de base de la détection de contour et ces différents types à fin de pouvoir les implémenté su FPGA.

Introduction

La détection de contour dans une image est un problème souvent posé avant toute phase de mesure ou d'interprétation.

Deux approches sont envisageables : l'approche contour ou l'approche région.

Le présent chapitre vise à introduire les concepts de base de la détection de contour en commençant par décrire un contour, ces différents types ainsi définir les différents approches utilisées pour la détection de contour.

I. La détection de contour

L'analyse d'image englobe l'ensemble des outils permettant la présentation de la scène perçue par le dispositif d'acquisition sous forme d'un signal 2D ou 3D ; la modélisation et le stockage de cette image sous forme numérique, le traitement, la compréhension et l'interprétation du contenue de l'image.

L'analyse d'image est née de la nécessité de remplacer ou de compléter l'observateur humain par la machine. A partir d'une image, il s'agit d'extraire les informations dont on a besoin, de les quantifier, de les traiter et de les interpréter.

La détection de contour est une étape préliminaire à de nombreuses applications de l'analyse d'images. Les contours constituent en effet des indices riches, pour toute interprétation ultérieure de l'image. Les contours dans une image proviennent des :

- discontinuités de la fonction de réflectance (texture, ombre).
- discontinuités de profondeur (bords de l'objet).

II. Extraction de contours

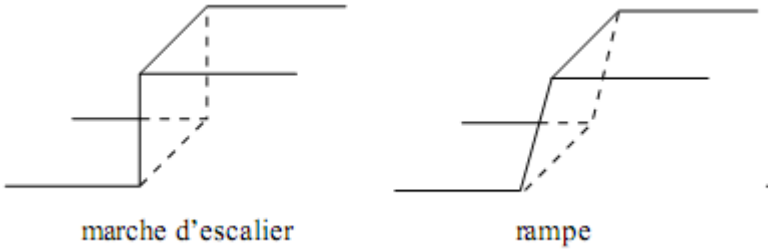
En général, les variations d'intensité dans une image correspondent à des zones pertinentes. Ces informations sont très importantes pour les opérations subséquentes à cette première phase.

Pour extraire de l'information de plus haut niveau, on doit d'abord avoir des informations de bas niveau (les contours).

Ces informations correspondent à des frontières de régions homogènes dans l'image.

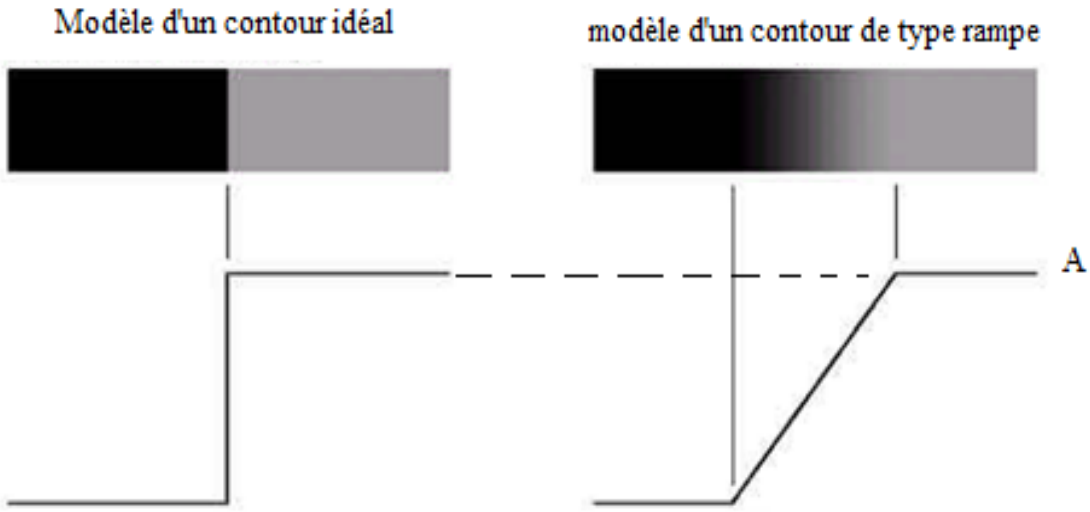
II.1. Les types de Contours

II.1.1 Contours de type marche



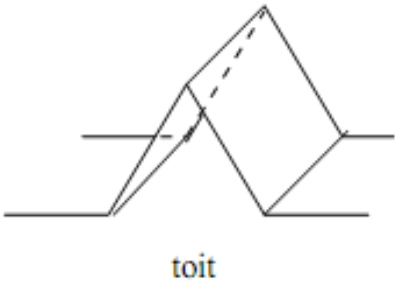
Figure(II.1) : Contours de type marche.

Ces contours existent généralement entre deux régions de niveaux de gris presque constants et différents; par exemple ce type de contours se produit lors du recouvrement d'un objet par un autre ou lors de la production d'ombre sur une surface. Nous pouvons caractériser ce contour par son amplitude (A) et par sa pente(p) comme illustrée dans la Figure (II.2)



Figure(II.2) : Contours de type marche caractérisé par son amplitude et sa pente.

II.1.2. Contours de type crête



Figure(II.3) : Contour de type toit

On distingue deux classes de contours :

- Les contours « toits »
- Les contours « arrêtes »

Nous trouvons ce type de contours dans les images contenant des lignes de faible ou forte intensité, telles que les images de caractères ou d'empreintes digitales. Le profil de la fonction des niveaux de gris perpendiculairement à la ligne de crête a la forme d'une cloche ou d'une vallée.

II.2. Les approches de détection de contour

II.2.1. Les approches classiques

Cette section présente un ensemble de méthodes qui ont eu historiquement une grande importance en traitement des images. Bien que certaines soient encore régulièrement employées parmi eux :

a). Les détecteurs de gradient par filtrage

Ces détecteurs reposent tous sur une recherche d'un extremum de la dérivée première ou d'un passage par zéro d'une dérivée seconde, celle-ci étant calculée de diverses manières, mais généralement par un filtrage passe-haut précédé d'un léger filtrage passe-bas pour s'affranchir des bruits.

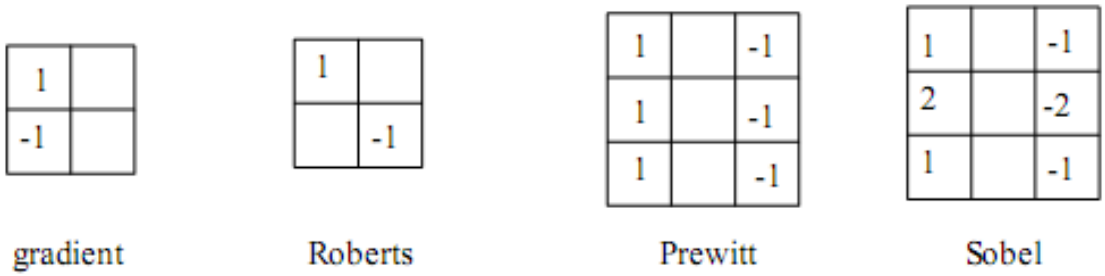
b). Les détecteurs de gradient par masques

A côté de ces approches très inspirés du traitement du signal, des filtres de dérivation plus empiriques ont été proposées à partir d'estimateurs locaux de l'image ou de ses dérivées.

Ces estimées sont obtenues à l'aide de masques présentés dans la figure 2.4appliqués sur des fenêtres de pixels2x2 ou 3x3 ou des fenêtres plus grandes en cas d'images très bruitées. La somme des coefficients de ces filtres est nulle (fonction de transfert nulle à la fréquence 0), et les coefficients sont antisymétriques.

Les filtres les plus utilisés sont :

Sobel, Roberts, Gradient, Prewitt[8][9][10]



Figure(II.4) Différents masques utilisés.

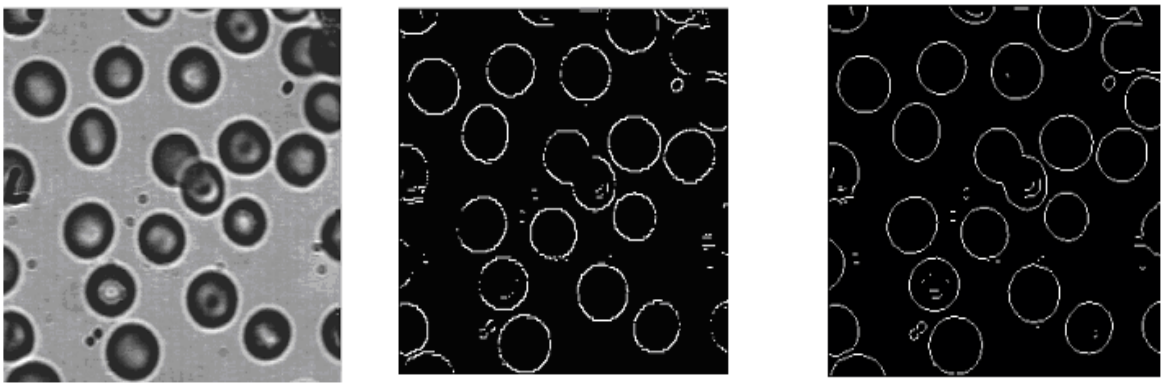


Image original operateur robert operateur sobel

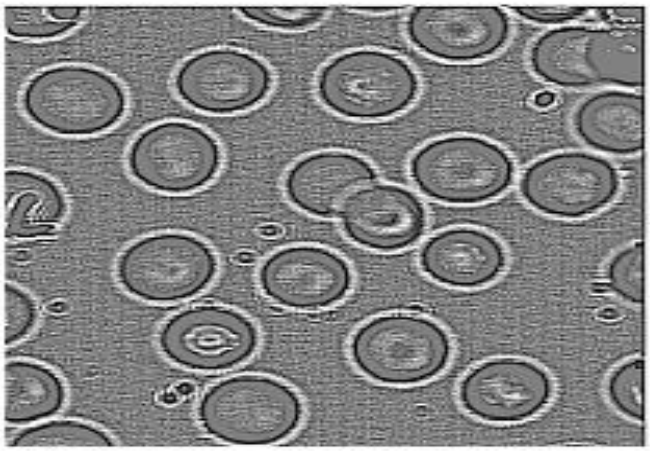
Figure(II.5) Image filtré.

c). Opérateur Laplacien

Cette méthode est basé sur le fait que le passage par zéro du Laplacient permet de bien mettre en évidence les extrémums de la dérivée, le passage par zéro du Laplacient correspond en effet bien au maximum du gradient dans la direction du gradient. Cette méthode tire en outre profit du fait que les zéros de la dérivée seconde constituent un réseau de lignes fermées.

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Figure(II.6) : Le masque (noyaux) de Laplacien.



Figure(II.7) : Détection de contour par Laplacien.

Le Laplacien possède cependant un inconvénient majeur qui est sa grande sensibilité au bruit. En effet cet opérateur réalise une dérivée seconde de l'image et est donc très instable.

II.2.2. Les approches analytiques

Nous allons voir maintenant une approche qui a permis une bien meilleure compréhension des conditions d'une bonne détection de contours et qui a ainsi conduit à des détecteurs de très bonne qualité.

a. Les critères de Canny

Canny a proposé [11] un filtre déterminé analytiquement à partir de trois critères :

- ✓ Bonne détection: la probabilité de ne pas réussir à marquer des points de contours réels doit être faible ainsi que la probabilité de marquer des points n'appartenant pas a des contours.
- ✓ Bonne localisation: Les points marques comme appartenant à un contour par l'operateur doivent être aussi proches que possible du centre du contour véritable.
- ✓ Une seule réponse à un seul contour: puisque s'il y a deux réponses à un même contour, l'une d'entre elle doit être considérée fausse.

Canny suit les étapes suivantes pour la détection de contour d'une image.

a.1. Réduction du bruit

Le bruit est tout phénomène imprévisible qui vient perturber le signal. Dans une image c'est un phénomène de brusques variations d'intensité d'un pixel par rapport à ses voisins.

Le bruit peut être causé par :

- Les événements inattendus lors de l'acquisition comme le bougé ou une modification ponctuelle des conditions d'éclairage.
- La mauvaise qualité des capteurs ou une mauvaise utilisation de ces derniers.
- Lors de l'échantillonnage. Le passage de la forme analogique à la forme numérique de l'image.

- Ou bien la nature de la scène elle même (poussières, rayures, perturbation atmosphérique,…).

Pour supprimer le bruit, le filtrage spatial est le plus souvent appliqué, c'est un filtrage qui s'appliqué sur un voisinage d'un pixel dans une image. Afin de pallier à ce bruit on opère un moyennage spatial à l'aide d'un filtre gaussien.

La fonction gaussienne en 2D est la suivante :

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

x : est la distance à l'origine sur l'axe horizontal

y : est la distance à l'origine sur l'axe vertical

σ : est l'écart type de la distribution gaussienne

Sa répartition est indiquée dans la figure (II.8)

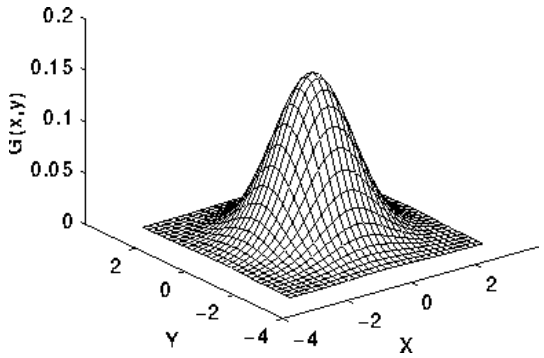


Figure (II.8) : Répartition de la fonction gaussienne

L'idée du lissage gaussien est d'utiliser cette distribution 2D comme une «fonction de point de propagation», ce qui est obtenu par convolution. Comme l'image est stockée comme une collection de pixels discrète nous avons besoin pour produire une approximation discrète de la fonction gaussienne avant que nous puissions effectuer la convolution. En théorie, la distribution gaussienne est non nulle par tout, ce qui exigerait un noyau de convolution infiniment grand, mais en pratique, lest effectivement nul plus que de trois écarts-types de la moyenne, et si nous pouvons tronquer le noyau à ce stade. La figure (II.9) montre un noyau de convolution approprié à valeurs entières qui se rapproche une gaussienne avec une σ =1.

$$\frac{1}{273}\begin{bmatrix}1 & 4 & 7 & 4 & 1\\ 4 & 16 & 26 & 16 & 4\\ 7 & 26 & 41 & 26 & 7\\ 4 & 16 & 26 & 16 & 4\\ 1 & 4 & 7 & 4 & 1\end{bmatrix}$$

Figure (II.9) : noyau de convolution qui se rapproche une gaussienne avec une σ =1.

a.2. Gradient d'intensité

La détection de contour par Canny est basé la ou il y a un changement brusque d'intensité, ces zones sont considérées par la détermination des gradients de l'image.

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

Le gradient à chaque pixel de l'image lissée sont déterminés en appliquant l'opérateur de Sobel.

Canny estime le gradient dans la direction x et y, en appliquant ces deux noyaux.

$$K_{GX} = \begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix}$$

$$K_{GY} = \begin{bmatrix}1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{bmatrix}$$

Il calcule le module et amplitude du gradient par les relations :

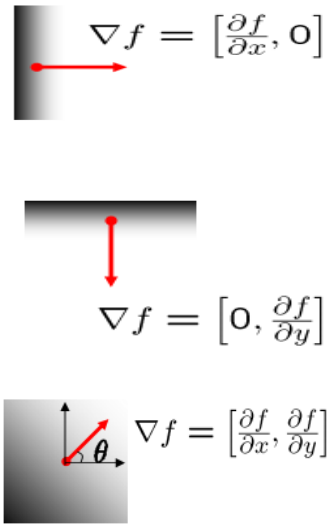
$$|G| = \sqrt{G_X^2 + G_Y^2} |G| = |G_x| + |G_y|$$

a.3. Direction des contours

Une fois que le gradient est déterminé à chaque pixel, les orientations des contours seront calculées par la formule :

$$\theta = \arctan\left(\frac{|G_Y|}{|G_X|}\right)$$

La figure (II.10) décrit les trois possibilités de la direction du gradient



Figure(II.10) : les trois possibilités de la direction du gradient

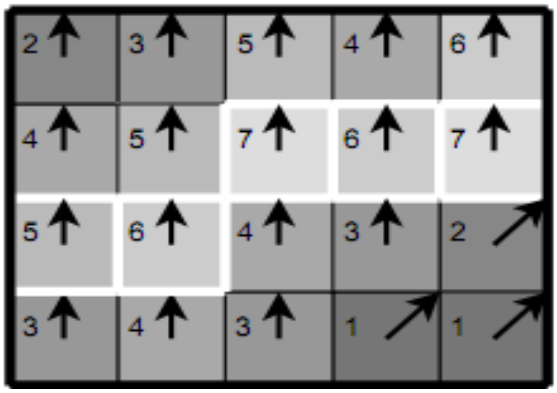
a.4. Suppression des non-maxima

Le but de cette étape est de convertir les bords "flous" à l'image du gradient aux bords «forts».

Cela se fait en conservant tous les maxima locaux de l'image gradient, et en supprimant tout le reste. L'algorithme est, pour chaque pixel de l'image gradient:

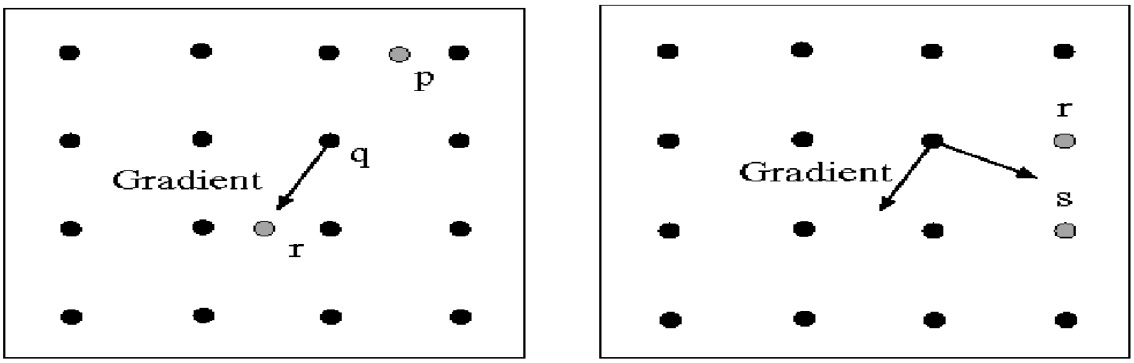
- Utiliser les 8 pixels voisin étant donnée une direction du gradient θ comme suit.
- Comparer la force du bord de pixel courant avec c'elle du pixel dont la direction du gradient positif et négatif .À savoir si la direction du gradient est au nord (θ =90 °), comparer avec les pixels au nord et au sud et ainsi de suite.
- si la force du bord du pixel courant est plus large que ses voisins on préserve la valeur si non on l'ignore

Les exemples suivants montrent la suppression des non-maxima



Figure(II.11) : exemple -1- Suppression des non-maxima.

Les résultats sont ceux marquées en blanc



Figure(II.12) : exemple -2- Suppression des non-maxima.

Au point q nous avons un maximum si la valeur du gradient est plus grande que celles de p et r.

a.5. Seuillage des contours

La différenciation des contours sur la carte générée se fait par seuillage à hystérésis. Cela nécessite deux seuils, un haut et un bas; qui seront comparés à l'intensité du gradient de chaque point. Le critère de décision est le suivant. Pour chaque point, si l'intensité de son gradient est:

- Inférieur au seuil bas, le point est rejeté
- Supérieur au seuil haut, le point est accepté comme formant un contour

- Entre le seuil bas et le seuil haut, le point est accepté s'il est connecté à un point déjà accepté.

b. Les critères de Shen-Castan

Shen et Castan ont proposé [12] un opérateur optimisant un critère incluant la détection et la localisation. Les critères qu'ils obtiennent correspondent aux critères de détection et de localisation de Canny et les filtres obtenus sont assez similaires dans la pratique. Le calcul du filtre de lissage optimal se fait par modélisation de la frontière par un échelon d'amplitude A noyé dans un bruit blanc stationnaire additif de moyenne nulle ; Le filtre de lissage obtenu par Shen et Castan s'écrit :

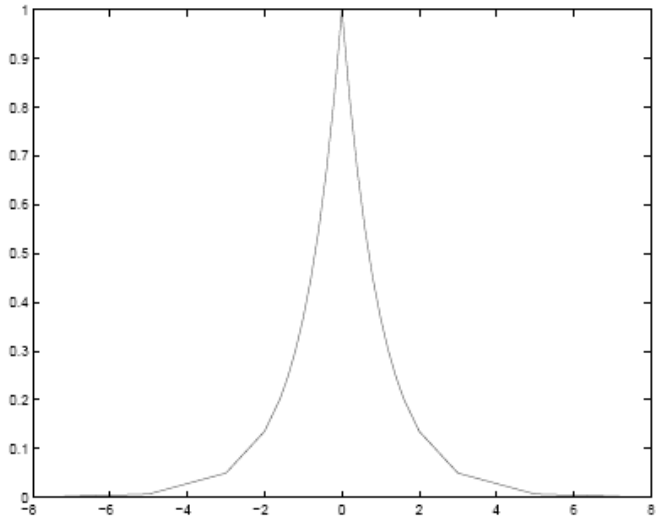
$$f(x) = ce^{-\propto|x|}$$

$$c = \frac{1 - e^{-\propto}}{1 + e^{-\propto}}$$

et le filtre de dérivation correspondant :

$$h(x) = \begin{cases} d.e^{-\propto x} six \geq 0 \\ d.e^{\propto x} six \leq 0 \end{cases}$$

Avec $d = 1 - e^{-\propto}$ Le paramètre α définit la ``largeur'' du filtre : plus α est petit, plus le lissage effectué par le filtre est important.



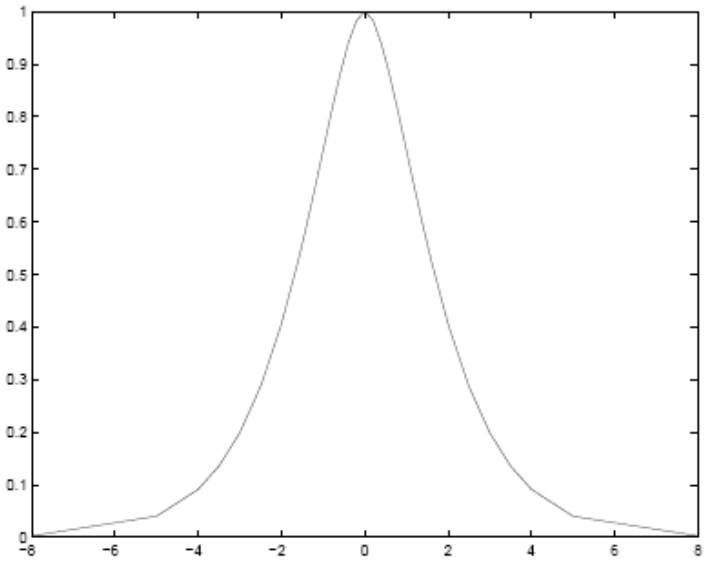
Figure(II.13) : Réponse impulsionnelle du filtre de shen-castan

c. Les critères de deriche

Deriche a proposé [13] un filtre de lissage dont la dérivée est la solution exacte de l'équation de Canny étendue aux filtres à supports infinis. Le filtre de lissage correspondant est :

$$h(x) = k(\propto |x| + 1)e^{-\propto|x|}$$

Avec $$k = \frac{(1-e^{-\propto})^2}{(1+2\propto e^{-\propto}-e^{-2\propto}}$$



Figure(II.14) : Réponse impulsionnelle du filtre de deriche.

II.2.3. L'approche contour actif

Les contours actifs sont des techniques de segmentation permettant d'extraire un objet d'intérêt et d'une image. Cette segmentation n'est pas immédiate, elle requiert une phase dynamique du contour (d'où la dénomination "actif") qui évoluera itérativement au cours du temps artificiel t, de sa position initiale vers les bords de l'objet à extraire. Une telle évolution temporelle peut se formaliser mathématiquement sous la forme d'une équation d'évolution exprimant explicitement ou implicitement la vitesse du contour actif. Il existe plusieurs manières d'obtenir une équation d'évolution. L'une consiste à dériver cette équation de la minimisation d'une fonctionnelle d'énergie, on parle alors d'approche variationnelle. Une alternative consiste à construire une équation d'évolution par analogie avec d'autres disciplines scientifiques comme la Physique, c'est l'approche

géométrique. Nous nous attacherons à décrire, puis utiliser l'approche variationnelle dans ce travail de thèse. Avec une telle approche, la fonctionnelle d'énergie peut être schématisée comme la somme de deux classes de termes énergétiques. La première classe concerne l'énergie interne du contour visant à contrôler des contraintes intrinsèques à celui-ci, comme par exemple sa régularité. La deuxième classe est relative au terme d'attache aux données qui fera interagir le contour actif avec des caractéristiques extraites de l'image. Dans les travaux de Foulonneau[14] et Jehan-Besson [15], ce terme d'énergie externe est appelé critère et est construit à partir de descripteurs. Un descripteur est une mesure faite sur l'image permettant de caractériser une frontière ou une région. Un descripteur de frontière serait par exemple la carte des gradients de l'image, un descripteur d'une région R pourrait être la moyenne des pixels de l'image inclus dans R. Selon le choix du descripteur adopté dans la fonctionnelle d'énergie, on dérivera différents types de contours actifs plus ou moins performants en fonction de la nature de l'image à analyser.

Donc il existe différentes catégories de contours actifs. Celles-ci se différencient par la façon avec laquelle on déduit l'équation d'évolution, par le mode de représentation du contour actif, et finalement le terme d'attache aux données qui peut être soit basé sur les frontières, les régions ou bien les deux.

Historiquement, c'est la représentation du contour actif a émergé en premier dans la communauté de Vision par Ordinateur grâce aux travaux pionniers de Kass, Witkin et Terzopoulosen 1987 [16]. Une telle représentation consiste à paramétrer le contour actif $\boldsymbol{\Gamma}$ par un paramètre arbitraire $\mathcal{P} \in$ [a, b] et le temps $\tau \in$ [0, T].

$$\begin{cases} [a,b] * [0,T] \to \mathbb{R}^2 \\ (\mathcal{P},\tau) \to \boldsymbol{\Gamma}(\mathcal{P},\tau) = X(\mathcal{P},\tau) = \begin{pmatrix} x(\mathcal{P},\tau) \\ y(\mathcal{P},\tau) \end{pmatrix} \end{cases}$$

L'évolution de ce contour actif $\boldsymbol{\rho}$ est régie par une équation aux dérivées partielles de la forme suivante :

$$\begin{cases} \dfrac{\partial \boldsymbol{\Gamma}(\mathcal{P},\tau)}{\partial \tau} = v(\mathcal{P},\tau) \\ \boldsymbol{\Gamma}(\mathcal{P},0) = \boldsymbol{\Gamma}_0(\mathcal{P}) \end{cases}$$

$\boldsymbol{\Gamma}_0(\mathcal{P})$ est le contour initial qui peut être déni manuellement par exemple et v est la vitesse d'évolution de la courbe. La figure (II.15) illustre cette évolution.

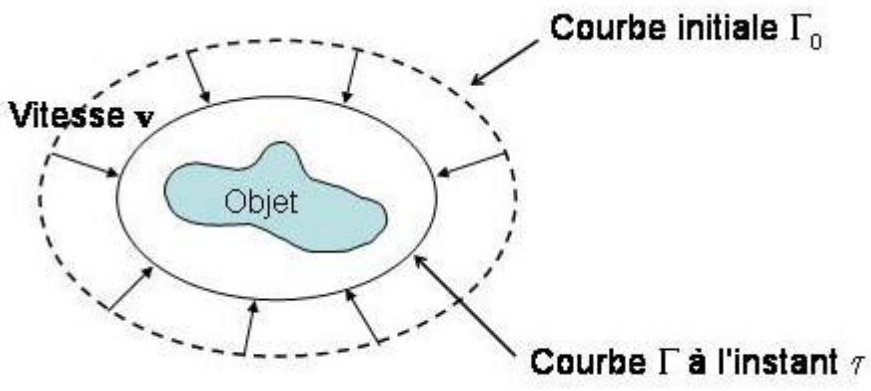


Figure (II.15) : Evolution du contour actif $\Gamma(\tau)$ avec la vitesse$v$ vers l'objet d'intérêt

La vitesse d'évolution v du contour actif a a priori une direction quelconque. Décomposons cette vitesse dans le repère de Frenet associé à la courbe. Nous obtenons deux composantes, une composante selon la tangente unitaire **T** et une composante selon la normale unitaire intérieure à la courbe **N**. Cela donne :

$$\frac{\partial \Gamma(\mathcal{P},\tau)}{\partial \tau} = v(\mathcal{P},\tau) = F_N(\mathcal{P},\tau)N(\mathcal{P},\tau) + F_T(\mathcal{P},\tau)T(\mathcal{P},\tau)$$

Où F_N est l'amplitude de la vitesse dans la direction normale au contour et $F_T$est l'amplitude de la vitesse dans la direction tangentielle au contour. La composante normale de la vitesse modifie la géométrie de la courbe tandis que la composante tangentielle de la vitesse modifie sa paramétrisation. L'équation d'évolution peut être simplifiée en considérant que nous nous intéressons uniquement à la déformation de la courbe et non à sa paramétrisation[17]. La partie de l'équation qui nous intéresse est donc uniquement la projection sur N car c'est celle qui régit le déplacement du contour, tandis que la projection sur T ne permet que de déterminer la paramétrisation du contour. L'équation d'évolution du contour actif s'écrit donc uniquement avec une vitesse normale au contour :

$$\begin{cases} \dfrac{\partial \mathbf{\Gamma}(\mathcal{P},\tau)}{\partial \tau} = v(\mathcal{P},\tau) = F_N(\mathcal{P},\tau)N(\mathcal{P},\tau) \\ \mathbf{\Gamma}(\mathcal{P},0) = \mathbf{\Gamma}_0(\mathcal{P}) \end{cases}$$

L'équation d'évolution de ce modèle de contours actifs est obtenue en minimisant la fonctionnelle suivante :

$$E_{image} = \alpha E_{ligne} + \beta E_{contour} + \lambda E_{term}$$

Avec :

$$E_{ligne} = I(x,y)$$

$$E_{contour} = -|\nabla I(x,y)|^2$$

$$E_{term} = \frac{C_{yy}C_x{}^2 - 2C_{xy}C_xC_y + C_{xx}C_y{}^2}{(C_x{}^2 + C_y{}^2)^{3/2}}$$

Avec $\alpha, \beta et \lambda$ des constantes positives.

Après cette rapide définition du principe des contours actifs, nous présentons maintenant les différents types de contours actifs existant : les contours actifs basés contour et les contours actifs basés régions. Un contour actif basé contour ne tient compte que des caractéristiques locales de l'image, à savoir le long du contour actif.

- **Approches variationnelles**

Les premiers contours actifs ont été intro duits par Kass et al. en 1988, sous le nom de snakes. Considérant une courbe $\boldsymbol{C^2}$ notée $\boldsymbol{\Gamma}: [\boldsymbol{a}, \boldsymbol{b}] \rightarrow \mathbb{R}^{\boldsymbol{2}}$, l'équation d'évolution de ce modèle de contours actifs est obtenue en minimisant la fonctionnelle suivante :

$$J(\Gamma) = \alpha \int_a^b |\Gamma'(p)|^2 \, dp + \beta \int_a^b |\Gamma''(p)|^2 \, dp + \lambda \int_a^b g(|\nabla I(\Gamma(p))|) \, dp$$

Avec $\alpha, \beta et \lambda$ des constantes positives.

Les deux premiers termes imposent une contrainte de régularité sur le contour et définissent son élasticité (premier terme) et sa rigidité (second terme) tandis que le troisième terme est un terme d'attache aux données qui va permettre d'attirer le contour actif vers les forts gradients d'image.

- **Contours actifs géodésiques**

La première approche de contours actifs a été améliorée par Caselles et al. Qui ont intro duit les contours actifs géodésiques. La fonctionnelle définie par Kass est modifiée en mettant β à zéro, et de ce fait, ce terme de régularisation jugé trop contraignant est abandonné. De plus, les auteurs démontrent que résoudre le problème des contours actifs classiques avec β à zéro revient à chercher une courbe géodésique dans un espace de Riemann, dans lequel la métrique dépend de l'image. Ils montrent que minimiser l'équation présidente lorsque β = 0 revient à minimiser :

$$J(\Gamma) = \int_a^b g(|\nabla I(\Gamma(p))|)|\Gamma'(p)| \, dp \, .$$

Ce qui peut s'écrire, puisque $|\Gamma'(p)|dp = ds$

$$J(\Gamma) = \int_0^{L(\Gamma)} g(|\nabla I(\Gamma(s))|)ds$$

Avec **L(Γ)** la longueur euclidienne de la courbe **Γ** et **s** l'abscisse curviligne de la courbe. La fonctionnelle est intrinsèque,i.e. qu'elle ne dépend pas de la paramétrisation. L'équation d'évolution suivante est obtenue :

$$\frac{\partial \Gamma}{\partial \tau} = (g(|\nabla I(\Gamma)|)\kappa - \nabla g(|\nabla I(\Gamma)|) \cdot \mathbf{N})\ \mathbf{N},$$

Avec ***κ*** la courbure du contour ***Γ*** . Dans l'équation suivante est ajoutée une force ballon ***c*** , intro duite par Cohen.

$$\frac{\partial \Gamma}{\partial \tau} = (g(|\nabla I(\Gamma)|)(c + \kappa) - \nabla g(|\nabla I(\Gamma)|) \cdot \mathbf{N})\ \mathbf{N}$$

Si la constante ***c*** est positive, le contour va rétrécir, et si elle est négative, le contour va s'élargir, tout en "combattant" la courbure pour permettre de segmenter des formes concaves et permettre au contour d'évoluer lorsque la courbure est nulle. Cette force rajoute une contrainte sur l'aire de la région. De plus, cette force ballon permet au contour d'éviter de rester "collé" dans des minima locaux de la fonctionnelle, notamment en présence de bruit. Elle permet aussi une plus grande liberté dans le choix du contour initial puisque celui-ci peut être à l'intérieur comme à l'extérieur de l'objet à segmenter (selon le signe choisi pour ***c***).

En suivant cette équation, le contour actif, dit *géodésique* aura un comportement différent selon qu'il se trouve dans une zone homogène ou proche d'un contour. Dans une zone homogène, le gradient de l'image est faible et l'équation se réduit à une évolution en fonction de la courbure. Près d'un contour, le gradient est plus fort et le contour actif évoluera vers le contour de l'objet, qu'il soit à l'intérieur ou à l'extérieur de l'objet. Grâce au dernier terme de l'équation, le contour actif est attiré aux ab ords de l'objet et stabilisé sur le contour.

- **Approches non variationnelles**

D'autres approches de contours actifs ont été proposées en parallèle en introduisant directement une équation d'évolution sans minimiser une fonctionnelle. Il s'agit dans ce cas d'envisager l'évolution de la courbe comme une propagation d'un front d'onde. C'est alors un modèle géométrique et non plus paramétrique. C'est le cas notamment de Caselles et al et Malladi et al qui ont présenté les contours actifs géométriques en s'inspirant des travaux de Osher et Sethian sur l'équation de la diffusion de la chaleur. L'équation d'évolution proposée se présente sous la forme suivante :

$$\frac{\partial \Gamma}{\partial \tau} = g(|\nabla I(\Gamma)|)\kappa \mathbf{N}$$

Avec $g(|\nabla I|)$ une fonction de détection. Elle est définie par :

$$g(|\nabla I|) = \frac{1}{1 + |\nabla (G_\sigma * I)|^p}, \quad p \in \{1, 2\}$$

Ou $G_\sigma * I$ est la convolution de l'image par une gaussienne de variance σ.

Cette fonctionnelle permet la segmentation de l'enveloppe convexe des objets. Pour parvenir à la segmentation d'objets non convexes, il a fallu ra jouter un terme semblable à la force ballon introduite par Cohen pour permettre au contour d'évoluer dans un sens ou dans l'autre. L'équation d'évolution proposée par Caselles est alors :

$$\frac{\partial \Gamma}{\partial \tau} = g(|\nabla I(\Gamma)|)(c + \kappa)\mathbf{N}$$

Nous pouvons remarquer que par rapport aux contours actifs classiques, nous n'avons conservé que le terme de régularisation du second ordre. De ce fait, la régularisation est moins stricte et son approximation numérique moins délicate. La même force ballon que dans les contours *géodésiques* est aussi introduite.

Tous ces modèles de contours actifs sont jusqu'à présent basés contours, c'est-à-dire qu'ils ne prennent en compte que des propriétés locales de l'image. Cependant une image est aussi caractérisée par des propriétés plus globales, à savoir les propriétés des régions comprises entre les contours de l'image. C'est pourquoi les contours actifs basés régions ont été intro duits.

- **Contours actifs basés régions**

Le principe des contours actifs basés régions est donc de prendre en compte des propriétés de l'image sur des régions et non plus localement. La fonctionnelle à minimiser dans le cas de termes régions s'écrit comme une intégrale sur la région et non plus sur le contour. Elle est de la forme :

$$J(\Omega) = \int_{\Omega} k(\mathbf{x}, \Omega) d\mathbf{x}$$

Où ***k*** est appelé descripteur de région. C'est une fonction qui dépend des propriétés de la région **Ω**. Il est important de noter que souvent les approches région combinent à la fois des termes basés régions et des termes basés contours, principalement pour prendre en compte les caractéristiques des régions et a jouter des contraintes sur le contour.
Les approches basées régions peuvent reposer sur une modélisation statistique des régions à segmenter. Un descripteur de moyenne ou bien de variance sur la région est, par exemple, parfois utilisé.

Les premiers à avoir utilisé des descripteurs de région sont Cohen et al et Ronfard Cohen et al présentent une méthode de reconstruction de surface en utilisant des contours actifs. Ils incluent un critère contenant des termes basés régions. Ronfard quant à lui propose une méthode de segmentation d'images en deux régions. Il définit la vitesse comme étant proportionnelle à la différence des descripteurs de la région des objets et de ceux de la région du fond. Dans de nombreux travaux, des critères statistiques sont utilisés. Nous pouvons notamment citer des méthodes développées dans le cadre du MDL (*Minimum Description Length*). Zhu et Yuille proposent un algorithme appelé *compétition de régions*. Leur idée est de présenter un cadre statistique pour combiner les caractéristiques géométriques des contours actifs et les techniques statistiques de croissance de régions. Dans leurs travaux, les descripteurs des régions sont les lois qui régissent les densités de probabilité. L'idée est de supposer connues les lois des densités de probabilité dont les paramètres sont à déterminer. Les paramètres des distributions sont estimés alternativement avec l'optimisation de la partition. Germain et al présentent un algorithme basé sur des contours actifs statistiques polygonaux permettant une bonne localisation des contours dans des images SAR (*Synthetic Aperture Radar, images issues d'un radar à synthèse d'ouverture*). Chesnaud et al utilisent des descripteurs statistiques dans un cadre de segmentation non supervisé. Les caractéristiques statistiques sont

estimées conjointement à l'étape de segmentation et cette méthode permet la segmentation d'images bruitées. Figueiredo et al proposent une méthode permettant de déterminer automatiquement l'ordre du modèle utilisé pour décrire le contour tandis que Ruch et al généralisent cette approche à des contours polygonaux et permettent ainsi une bonne régularisation du contour. Galland et al présentent un modèle de segmentation adapté à la segmentation de plusieurs régions d'intensité différente tandis que Martin et al proposent une méthode de segmentation où aucun paramètre n'est à régler et qui peut s'appliquer pour n'importe quelle distribution d'intensité. Nous pouvons aussi citer les méthodes utilisant les *graph cuts* (métho de de coup e minimale/ flot maximal dans un graphe) qui permettent d'effectuer une minimisation d'énergie sans tomber dans un minimum local, et ce en un temps polynômial. Ces méthodes connaissent un fort succès actuellement et sont utilisées dans de nombreuses applications de traitement d'images, et notamment en segmentation d'images, étendent quant à eux les résultats de Mumford et Shah en utilisant la moyenne comme descripteur pour la segmentation en deux régions, le descripteur utilisé est aussi la moyenne des régions mais la fonctionnelle est différente puisque c'est la distance quadratique entre les moyennes qui est maximisée. Paragios et al proposent d'utiliser des régions actives géodésiques dans un cadre supervisé où les paramètres des distributions sont déterminés au préalable. Jehan-Besson et al utilisent aussi des descripteurs de moyenne mais présentent surtout un cadre général de différenciation des fonctionnelles à l'aide d'outils inspirés de l'optimisation de domaines.

Les contours actifs basés régions sont aussi largement utilisés dans l'imagerie médicale. Debreuve et al introduisent un critère spatio-temporel pour la segmentation de séquences cardiaques. Montagnat et al utilisent des modèles déformables 3D pour la segmentation de volumes.

Du point de vue de l'implémentation, Precioso et al étendent le modèle des splines d'interpolation pour en proposer une implémentation paramétrique pour les contours actifs. Ils présentent les splines d'approximation qui permettent une robustesse au bruit intéressante. Les histogrammes peuvent être utilisés dans des critères de segmentation. Jehan-Besson et al et Freedman et al proposent d'utiliser des histogrammes comme descripteurs des régions. Des histogrammes connus sont pris en compte afin de servir de référence pour la segmentation. Kim et al proposent quant à eux d'utiliser des distributions non-paramétriques afin de minimiser l'information mutuelle entre l'intensité de l'image et un label définissant la position du pixel par rapport au contour.

Nous avons présenté les déférentes approches pour définir un critère de segmentation pour les contours actifs, de l'approche purement "contours" aux approches plus récentes qui intègrent des termes de régions dans les fonctionnelles à minimiser. Les approches contour ne prenant en compte que de l'information locale sont assez sensibles au bruit et lors de la minimisation de la fonctionnelle, il n'est pas rare de tomber dans un minimum local Quant aux approches régions, la plupart des méthodes sont efficaces dans des cas simples où l'objet à segmenter est facilement séparable du fond, de par sa couleur par exemple, mais la segmentation est plus difficile lorsque les objets à segmenter ne respectent pas les hypothèses faites sur leurs distributions

Conclusion

Dans l'approche contour, on trouve les méthodes dérivatives qui sont les plus immédiates pour détecter et localiser les variations du signal.

L'avantage des opérateurs de détection des contours est la simplicité d'utilisation. Par contre, ils sont très sensibles aux bruits et donnent des contours ouverts.

Contrairement aux approches classiques de détection de contours, les modèles déformables incorporent de la connaissance à priori sur la forme de l'objet recherché. Cette connaissance peut être issue d'une base d'apprentissage où une forme référence (forme prototype), ainsi des modes de déformation peuvent être extraits. Elle peut cependant être plus générale et se limiter à des propriétés comme la continuité ou la régularité qui caractérise le contour d'un objet.

Les contours actifs sont largement utilisés pour leur capacité à intégrer les processus de détection et de chaînage des contours en un seul processus de minimisation d'énergie; Toutefois l'estimation des paramètres et les problèmes d'initialisation font des contours actifs une méthode difficile à calibrer.

Introduction

Le but de ce chapitre est d'implémenter un contour actif pour la détection de contour en temps réel afin de sur FPGA.

L'algorithme choisit pour l'implémentation est celui des snake vu la simplicité de son principe et les avantage qu'il offre ainsi que c'est la base des contours actif.

Afin d'accomplir ce travail ; le présent chapitre a été structuré sur deux phases de la manière suivante :

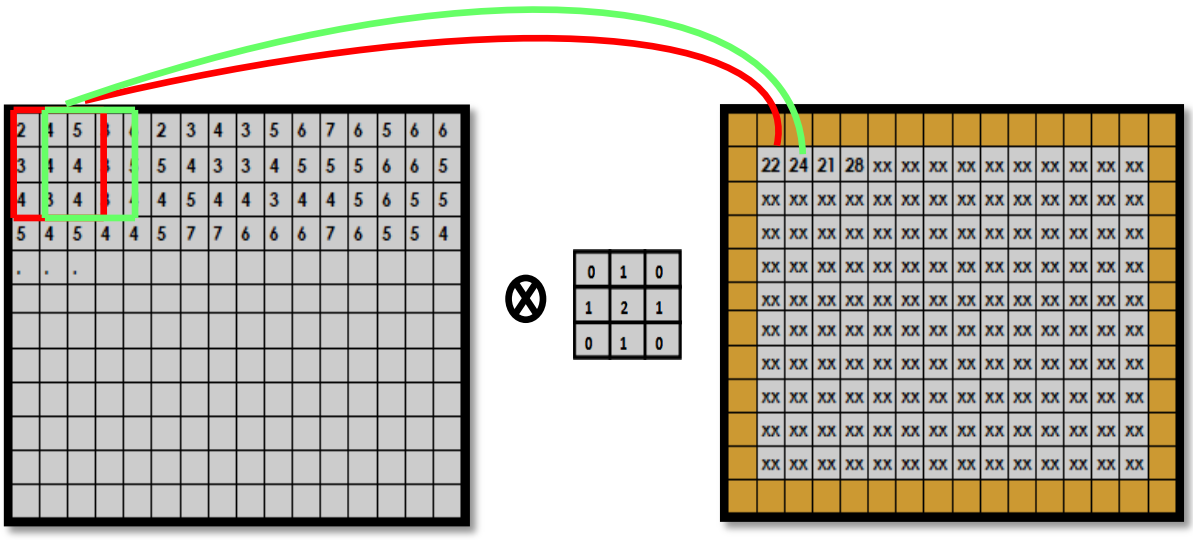
- Au cours de la première nous allons présenter l'architecture proposée et la description en VHDL de chaque module utilisé.
- La deuxième partie présente les résultats obtenus avec l'outil de simulation modelsim.

I. Principe de fonctionnement de la convolution 2D sur FPGA

La convolution multiplie le contenu de deux tableaux de nombres de tailles différentes entre eux, pour produire un troisième tableau de nombres.

L'idée de base est de balayer l'image avec une fenêtre de taille (2n+1)*(2n+1), la valeur du pixel de sortie est la somme pondérée des pixels d'entrée dans la fenêtre, ou les poids sont les valeurs du filtre assigné à chaque pixel de la fenêtre.

La figure (III.1) montre le déplacement horizontal du masque de convolution 3*3 pour chaque trois lignes de l'image en traitement, le pixel central est remplacé par le résultat du calcul de convolution.

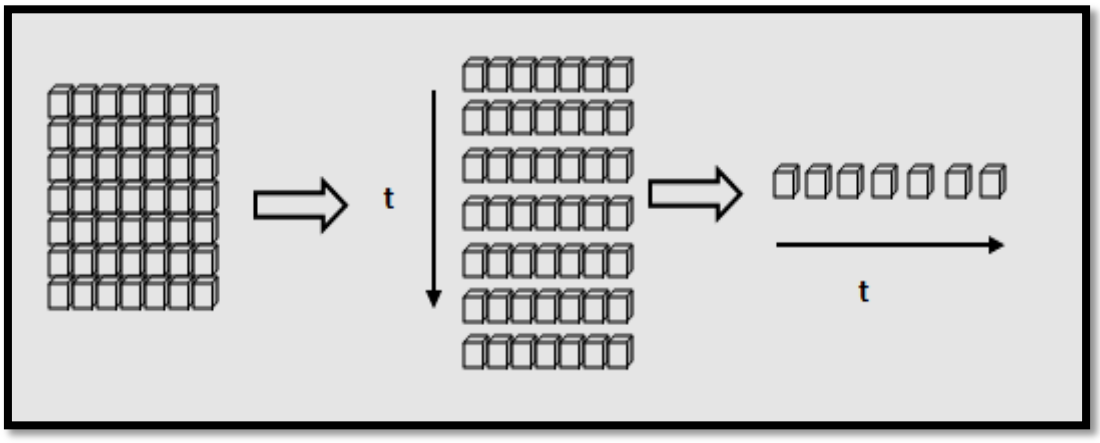


Figure(III.1) : Principe de convolution 2D.

II. Gestion de flux de données

La convolution utilisée traite l'image en utilisant un flux de pixel qui est générées à partir des lignes de l'image.

A partir d'une image 512*512 on extrait les lignes séquentiellement puis les pixels sont à leur tour extraits à partir des lignes. Le flux de données est synchronisé par l'horloge du FPGA.



Figure(III.2) : Balayage image.

L'extraction des lignes consiste à construire un module qui permet de marquer la succession de ces dernières. Un retard ligne de pixels est généré pour la réalisation de la convolution, il nécessite des mémoires de type FIFO.

Le FPGA spartan 3E offre une RAM bloc qu'on peut la considérer comme une FIFO, sa description VHDL est la suivante :

```
RAMB1: RAMB16_S9
generic map (
    INIT => X"000", --  Value of output RAM registers at startup
    SRVAL => X"000", --  Ouput value upon SSR assertion
    WRITE_MODE => "WRITE_FIRST", --  WRITE_FIRST, READ_FIRST or NO_CHANGE
    -- The following INIT_xx declarations specify the initial contents of the RAM
    -- Address 0 to 511
    INIT_00                                                                    =>
X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_01                                                                    =>
X"0000000000000000000000000000000000000000000000000000000000000000",

port map (
    DO => DO_1,      -- sortie de la donné sur 8-bits.
    DOP => DOP_1,    -- 1-bit sortie de parité.
ADDR => add,  -- addressagesur 11-bits.
CLK =>clk,    -- horloge.
    DI =>d_in,      -- entrée de la donné sur 8-bits.
    DIP => "0",    -- 1-bit entrée de parité.
    EN => '1',      --bit activation de la RAM.
    SSR => '0',    -- Reset entrée de synchronisation.
WE => we_1       -- 1 bit d'activation en écriture.
  );
```

Cette description se trouve dans la bibliothèque de l'outil de développement ISE 10.1.

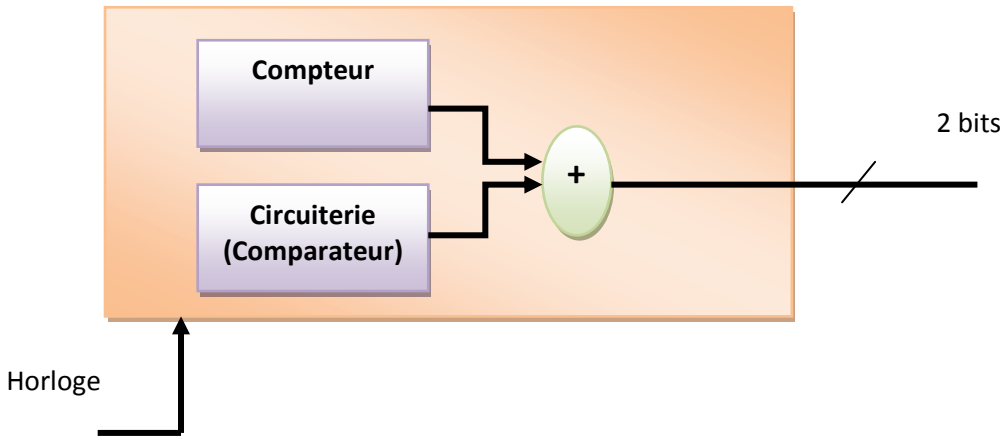
II.1. Génération des signaux de contrôle des RAM's

Les filtres adaptés pour notre traitement utilisent des masques defenêtre3x3, donc pour le calcul de la convolution on a besoin de stocker trois (03) lignes pour le balayage horizontal du masque.

Il est nécessaire de prendre une quatrième ligne pour considérer le déplacement vertical du masque.

L'alternance de la sélection des mémoires en lecture/écriture, est faite de telle sorte, que l'on ait trois signaux sélectionnés en mode lecture et un signal de sélection en mode écriture.

Le contrôle des signaux de sélection nécessite un système séquentiel qui est composé d'un compteur modulo 2048 et d'une circuiterie (comparateur) qui procure une sortie sur deux bits, figure(III.3).



Figure(III.3) : Circuit de génération des signaux de contrôle.

Ce système fonctionne à la cadence de l'horloge de FPGA, il offre en sortie tous les 512 tops d'horloge un code sur 2 bits qui contrôle la sélection des RAM's ; ces bits sont dit bits de contrôle ou d'état (état1, état2, état3 et état4).

L'organigramme suivant montre le séquencement de génération des signaux de contrôle des RAM's .

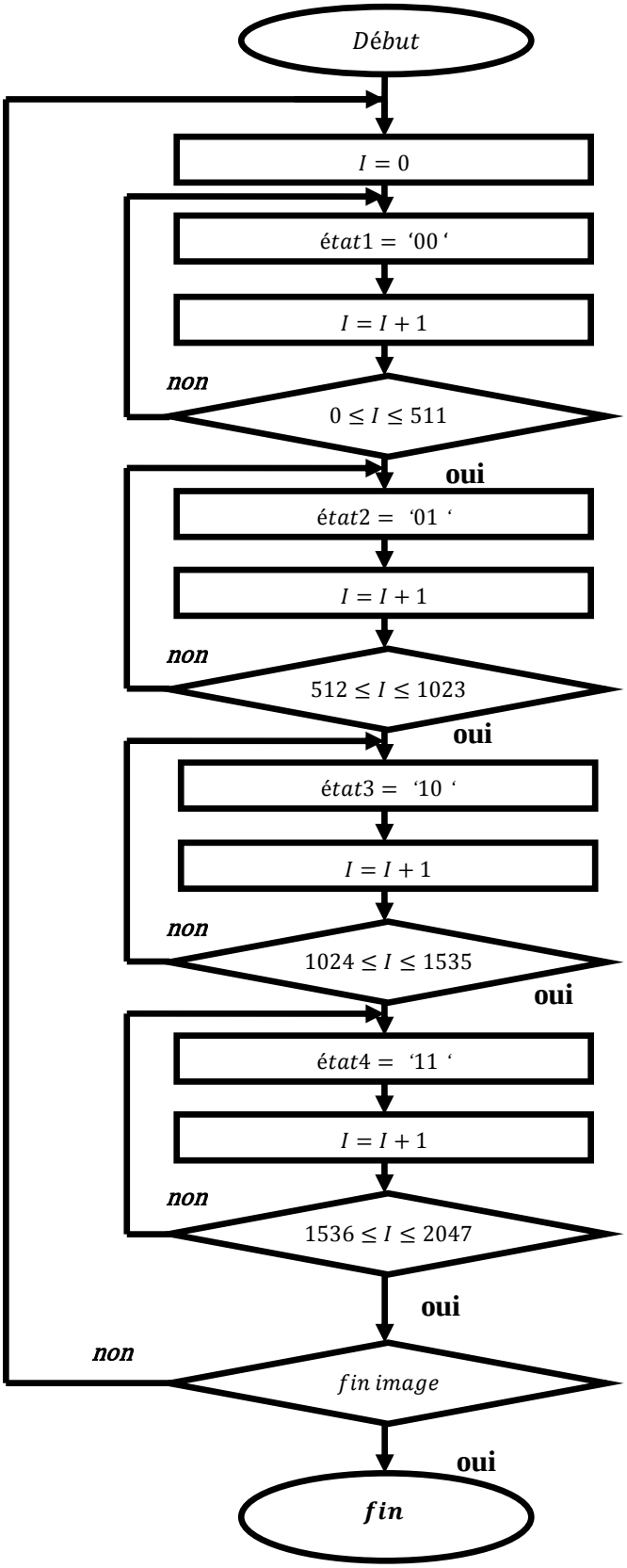


Figure (III.4) : Organigramme de génération des signaux de contrôle RAM

II.2. Sélection des RAM's

Les 4 états générés, permettent d'obtenir des signaux de sélection des RAM's.De chaque état, on génère 4 signaux qui définissent le mode de sélection des RAM's.

Chaque état, correspond à trois (03) sélections en mode lecture et une (01) en mode écriture.

les modes de selection des RAM's sont donnés par : we_1, we_2, we_3, et we_4.

La sélection est faite comme le montre la table de vérité suivante :

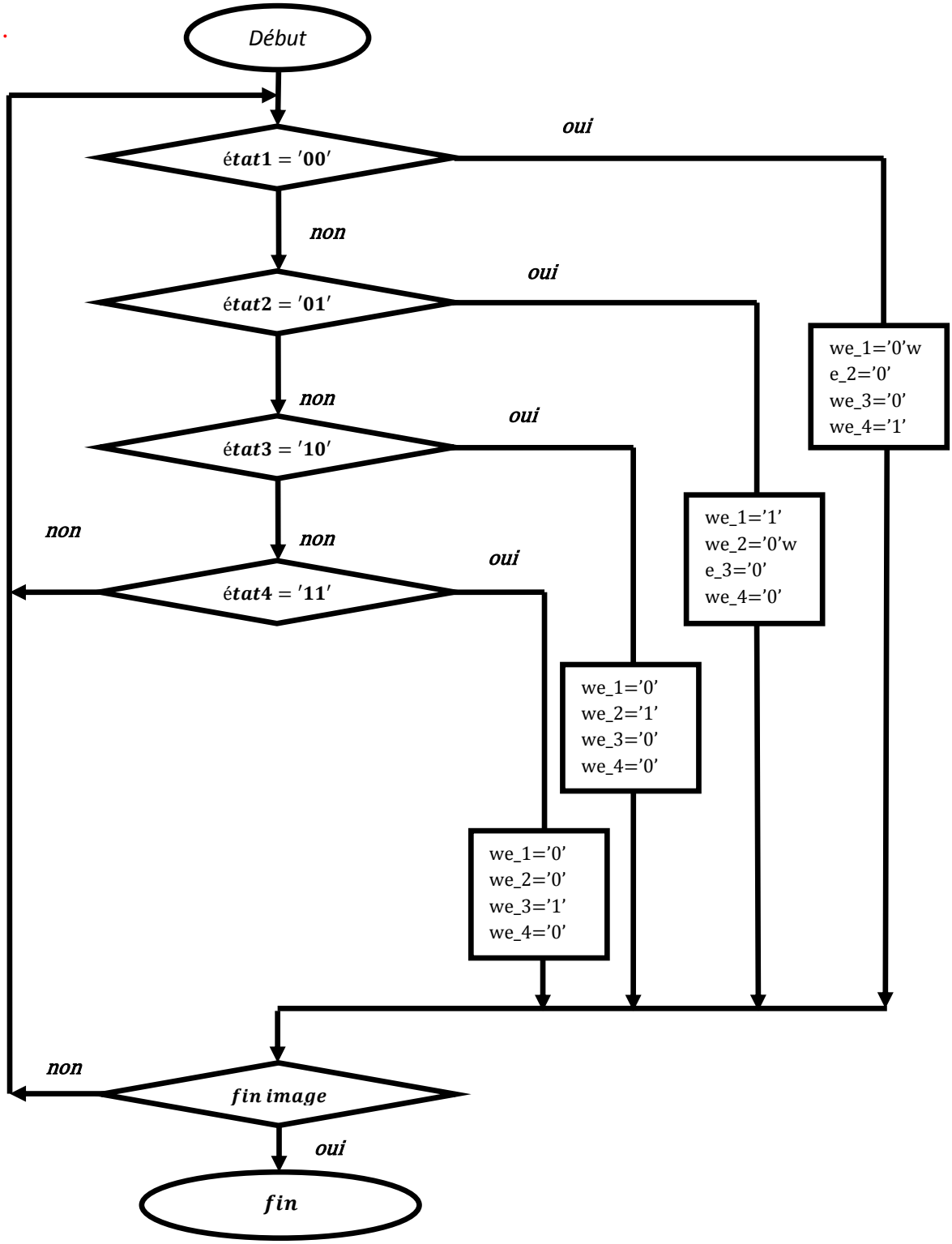
états	We_1	We_2	We_3	We_4
00	0	0	0	1
01	1	0	0	0
10	0	1	0	0
11	0	0	1	0

Tab(III.1) : Table de vérité de mode de sélection des RAM's

La sélection est donnée par :

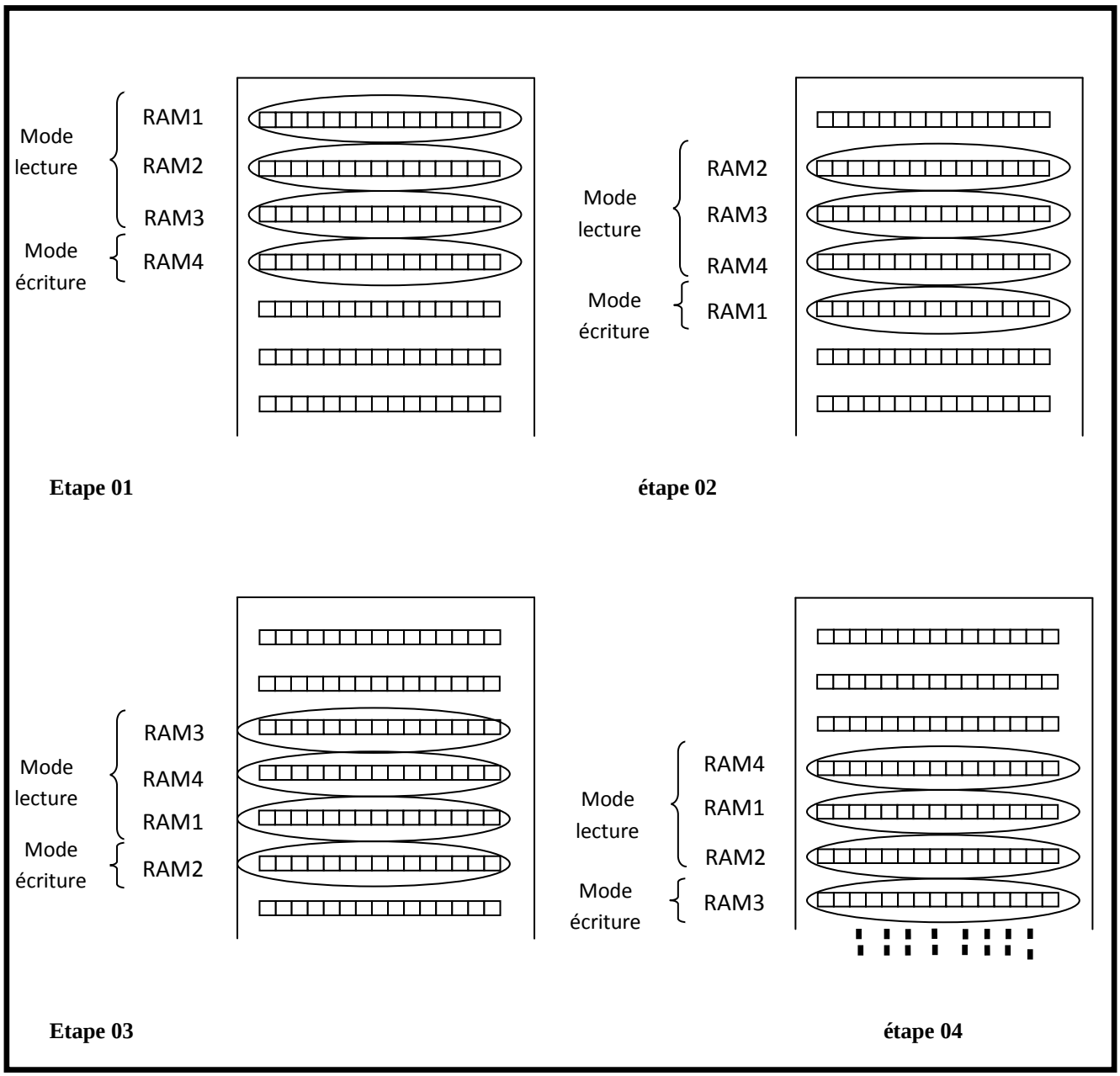
- we_x='1' en mode lecture
- par we_x='0' en mode écriture

L'organigramme suivant permet de voirles modes des sélections des RAM's en lecture/écriture.



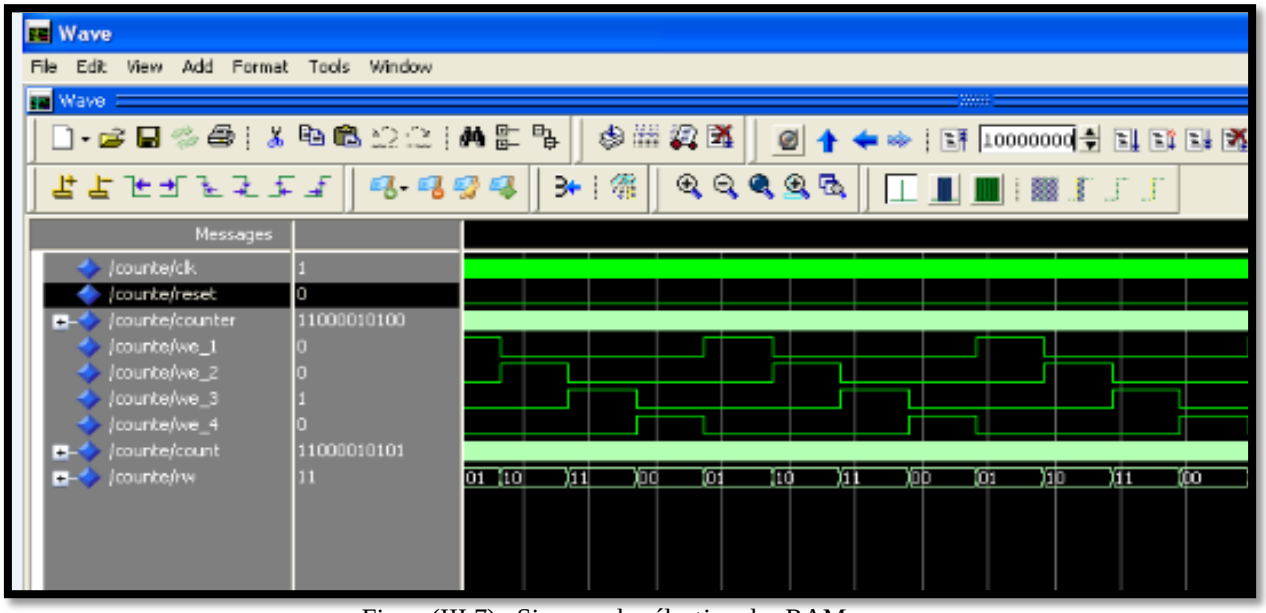
Figure(III.5) : Organigramme du mode de sélection RAM.

Le schéma de fonctionnement correspond à cette opération est donnée par la figure (III.6).



Figure(III.6) : Schéma fonctionnelle de sélection des RAM's.

La figure (III.7), représente les quatre signaux de sélection RAM's, ces signaux sont donnés par we_1, we_2, we_3, we_4.



Figure(III.7) : Signaux de sélection des RAMs

II.3. Ecriture des données sur la RAM

Un module d'adressage est nécessaire pour chaque RAM's, ce dernier est réalisé par un compteur synchrone modulo 512, l'incrémentation du compteur ce fait par pas de 1, ce dernier permet d'écrire les données sur la RAM, ces données mémorisées correspondent à une ligne de l'image par RAM.

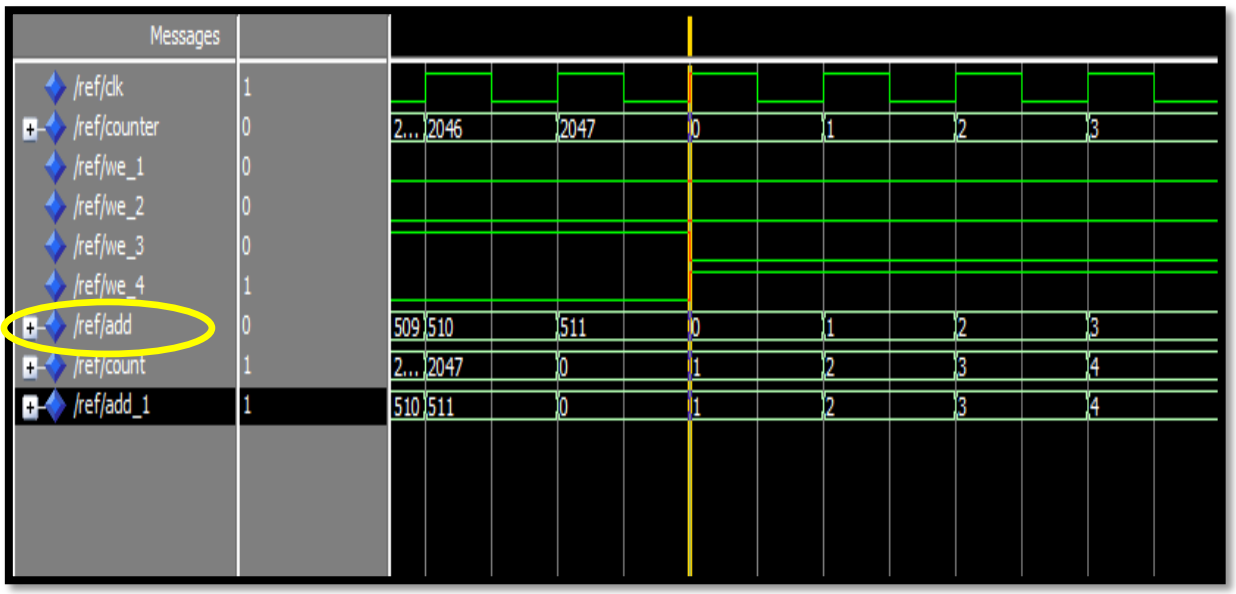


Figure (III.8) : Signaux d'adressage des RAM's.

La sortie du module d'adressage est mentionnée dans le chronogramme par « add ».

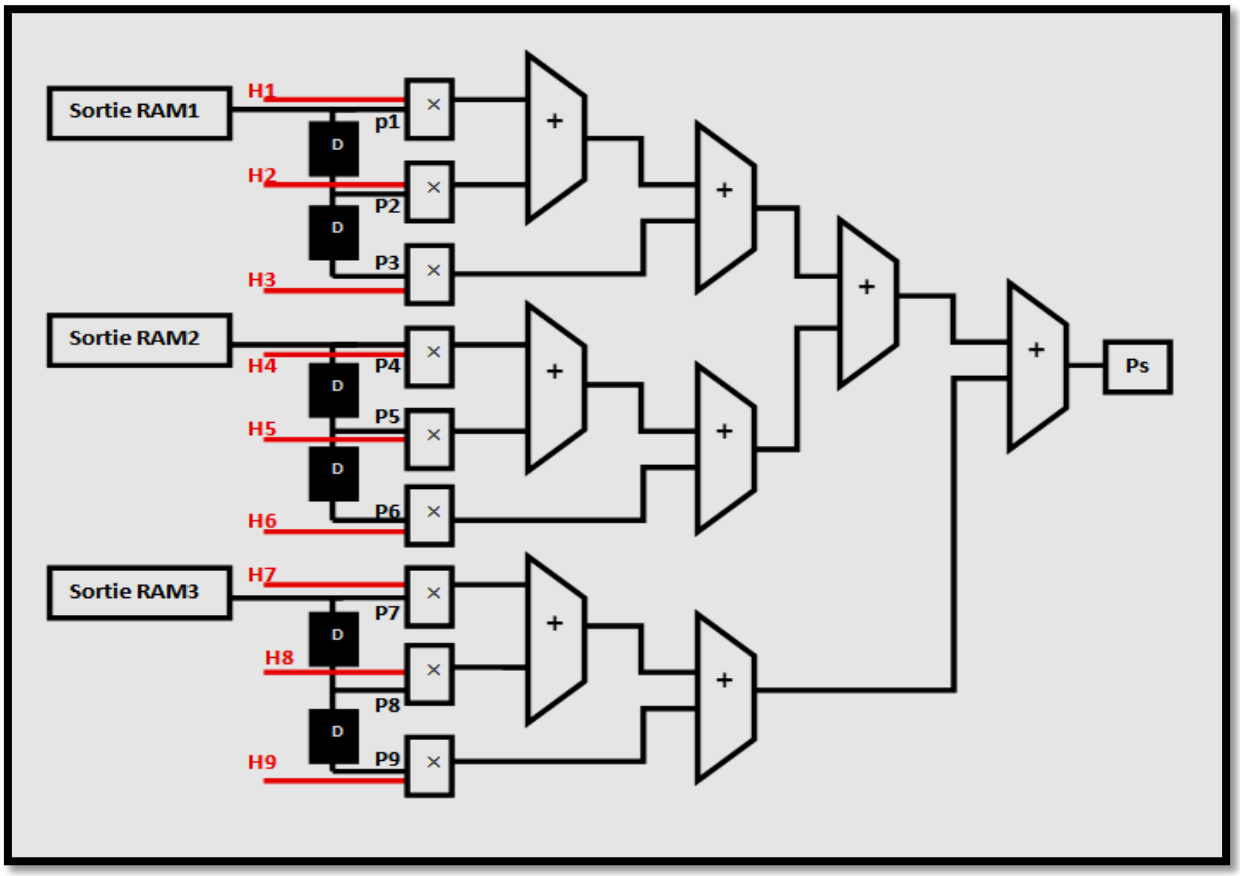
II.4. Architecture de calcul de la convolution

Deux bascules sont reliées à la sortie de chaque RAM, ceci dans un but de repérer le pixel en cours dans le flux.

Cette technique permet de considérer les pixels qui vont être utilisés pour le calcul de la convolution à chaque top d'horloge.

Une multiplication par le masque du filtre est réalisée, puis une suite d'addition est effectuée selon la figure (III.9).

Le masque est donnée par : $\begin{pmatrix} H1 & H2 & H3 \\ H4 & H5 & H6 \\ H7 & H8 & H9 \end{pmatrix}$



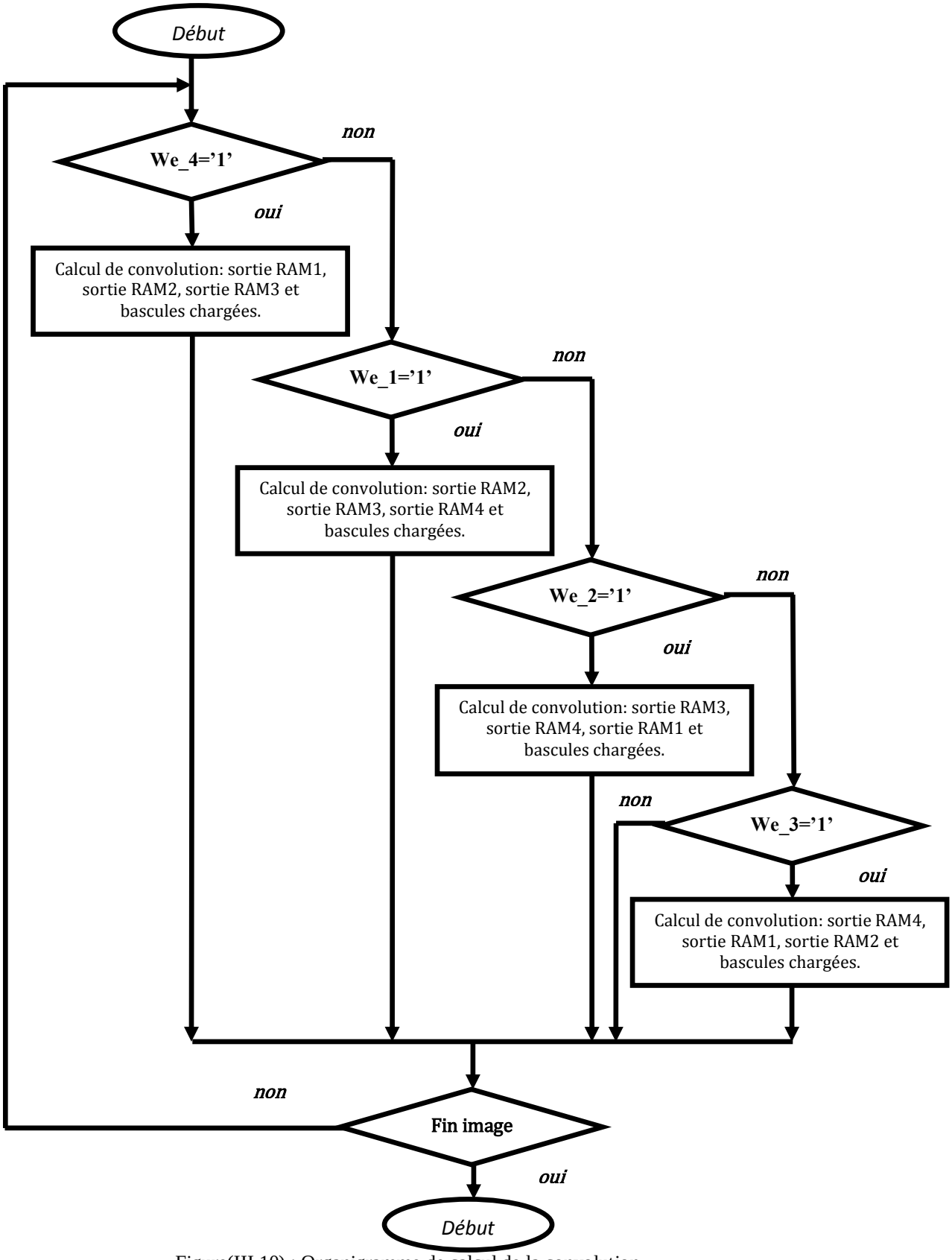
Figure(III.9) : Schéma bloc de la convolution.

Le résultat du pixel de sortie est donnée par :

$$Ps=(H1\times P1)+(H2\times P2)+(H3\times P3)+(H4\times P4)+(H5\times P5)+(H6\times P6)+(H7\times P7)+(H8\times P8)+(H9\times P9).$$

L'organigramme suivant illustre les différentes étapes de calcul de la convolution avec un masque de 3× 3

.



Figure(III.10) : Organigramme de calcul de la convolution.

III. Architecture générale du système de convolution

La figure(III.11), montre le schéma bloc de l'architecture de traitement et les différentes interconnexions entre les modules.

A noter, que le module d'adressage est le même pour les quatre RAM's. Ainsi, les modules sont synchronisés par la même horloge.

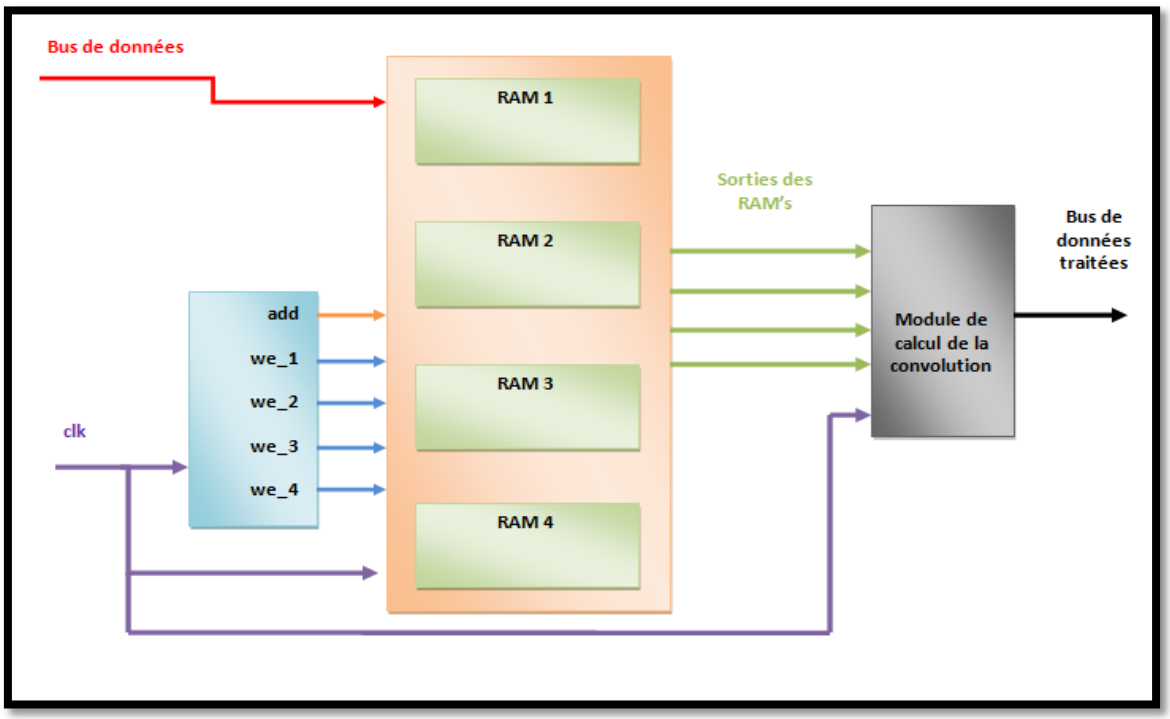


Figure (III.11) : Schéma bloc de l'architecture de traitement.

IV. Implémentation de l'algorithme des snakes

Un contour actif est un ensemble de points qu'on va tenter de déplacer pour leur faire épouser une forme. Il s'agit d'une technique d'extraction de données utilisée en traitement d'images. L'idée de cette méthode est de déplacer les points pour les rapprocher des zones de fort gradient tout en conservant des caractéristiques comme la courbure du contour ou la répartition des points sur le contour ou d'autres contraintes liées à la disposition des points.

Au démarrage de l'algorithme, le contour est disposé uniformément autour de l'objet à détourer puis il va se rétracter pour en épouser au mieux ses formes.

De la même manière, un contour actif peut aussi se dilater et tenter de remplir une forme, il sera alors situé à l'intérieur de celle-ci au démarrage de l'algorithme.

A chaque itération, l'algorithme va tenter de trouver un meilleur positionnement pour le contour pour minimiser les dérives par rapport aux contraintes utilisées. L'algorithme s'arrêtera lorsque le nombre maximum d'itérations aura été atteint. On utilise les notions d'énergies interne et externe pour caractériser respectivement la forme du contour et tous les éléments qui lui sont propres, et le positionnement du contour sur l'image.

La figure (III.12) illustre un exemple de déroulement d'une recherche en contour actif pour détourer un objet :

Figure (III.12) : exemple d'un contour initiale

Chaque itération peut se représenter de la manière suivante :

- calcul des énergies interne et externe, caractérisant le contour lui-même et son positionnement sur l'image.
- pour chaque point du contour, détermination d'une nouvelle position, sur laquelle le contour devrait mieux minimiser les écarts de contraintes.
- arrangement du contour pour qu'il respecte des contraintes d'écartement entre les points, de régularité de points ...

IV.1. L'énergie interne

L'énergie interne ne dépend pas de l'image ni de la forme à détourer, elle ne dépend que des points du contour.

Elle regroupe des notions comme la courbure du contour ou la régularité d'espacement des points. En effet, le contour doit conserver une forme arrondie en minimisant les dérivées d'ordre 1, 2, ... et doit empêcher un point de se détacher trop loin du reste du contour. Idéalement, l'énergie interne est minimale pour un cercle où tous les points sont régulièrement espacés.

Kass propose dans son article d'utiliser le masque laplacien pour déterminer l'énergie interne, ce qui impose de calculer le produit de convolution de l'image avec le masque laplacien

IV.2. L'énergie externe

L'énergie externe correspond à l'impact du contour sur l'image. Pour la calculer Kass a proposé :

$$E_{exter} = w_{ligne}\, E_{ligne} + w_{contour}\, E_{contour} + w_{term}\, E_{term}$$

Dont

$$E_{ligne} = I(x,y)$$

$$E_{contour} = -|\nabla I(x,y)|^2$$

$$E_{term} = \frac{C_{xx} C_y^2 - 2C_{xy} C_x C_y + C_{xx} C_y^2}{(C_x^2 + C_y^2)^{3/2}}$$

Avec

$$C_x = [-1 \quad 1] \otimes I(x,y)$$

$$C_y = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \otimes I(x,y)$$

$$C_{xx} = [1 \quad -2 \quad 1] \otimes I(x,y)$$

$$C_{yy} = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} \otimes I(x,y)$$

$$C_{xy} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \otimes I(x,y)$$

IV.3. Utilisation des deux énergies

Chaque position du contour actif donne une énergie interne et une énergie externe dont la somme doit être minimisée et influencera les mouvements des points du contour actif.

Après avoir calculé l'énergie globale dégagée par le contour et par son positionnement sur l'image, il convient de déterminer comment le faire évoluer pour minimiser cette énergie. Pour cela, une méthode simple et intuitive est d'observer les pixels voisins immédiats de chaque point du contour pour déterminer pour chacun d'eux l'énergie globale du snake, chaque meilleur voisin devenant un point du contour.

IV.4. Architecture proposé pour l'implémentation

L'organigramme de la figure (III.13) représente l'architecture proposé pour l'implémentation.

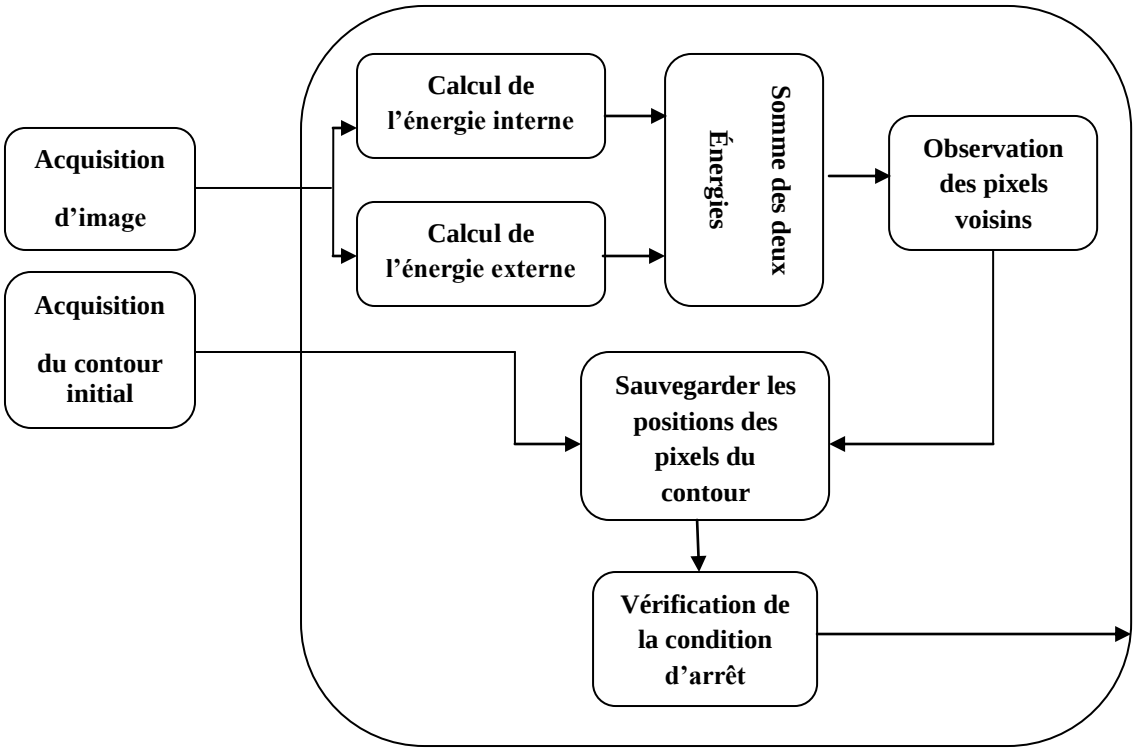


Figure (III.13) : Organigramme présentant l'architecture proposée pour l'implémentation du contour actif

IV.5. Acquisition d'image

Au cours de notre projet les images utilisé sont pris avec un appareil photo et sauvegarder dans le PC, ensuite on a changé leur format en RAW c'est-à-dire un format brut en code ascii à l'aide de l'outil adobe photo shop afin de les lire pixel par pixel dans l'outil de développement xilinxise 12.3.

La lecture de l'image pixel par pixel se fait avec une description VHDL de la manière suivante :

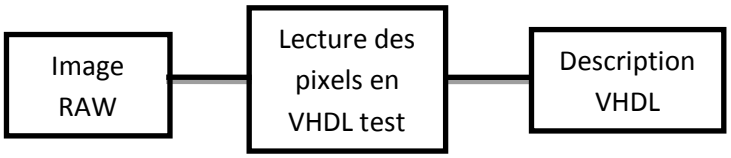


Figure (III.14) : procédure pour la lecture d'un fichier image en VHDL

IV.6. Acquisition du contour initial

Pour l'acquisition du contour initial on a développé un code source matlab qui permet d'extraire les positions du contour initial, de les trier par ordre croissant et de les sauvegarder dans un fichier texte en code ascii.

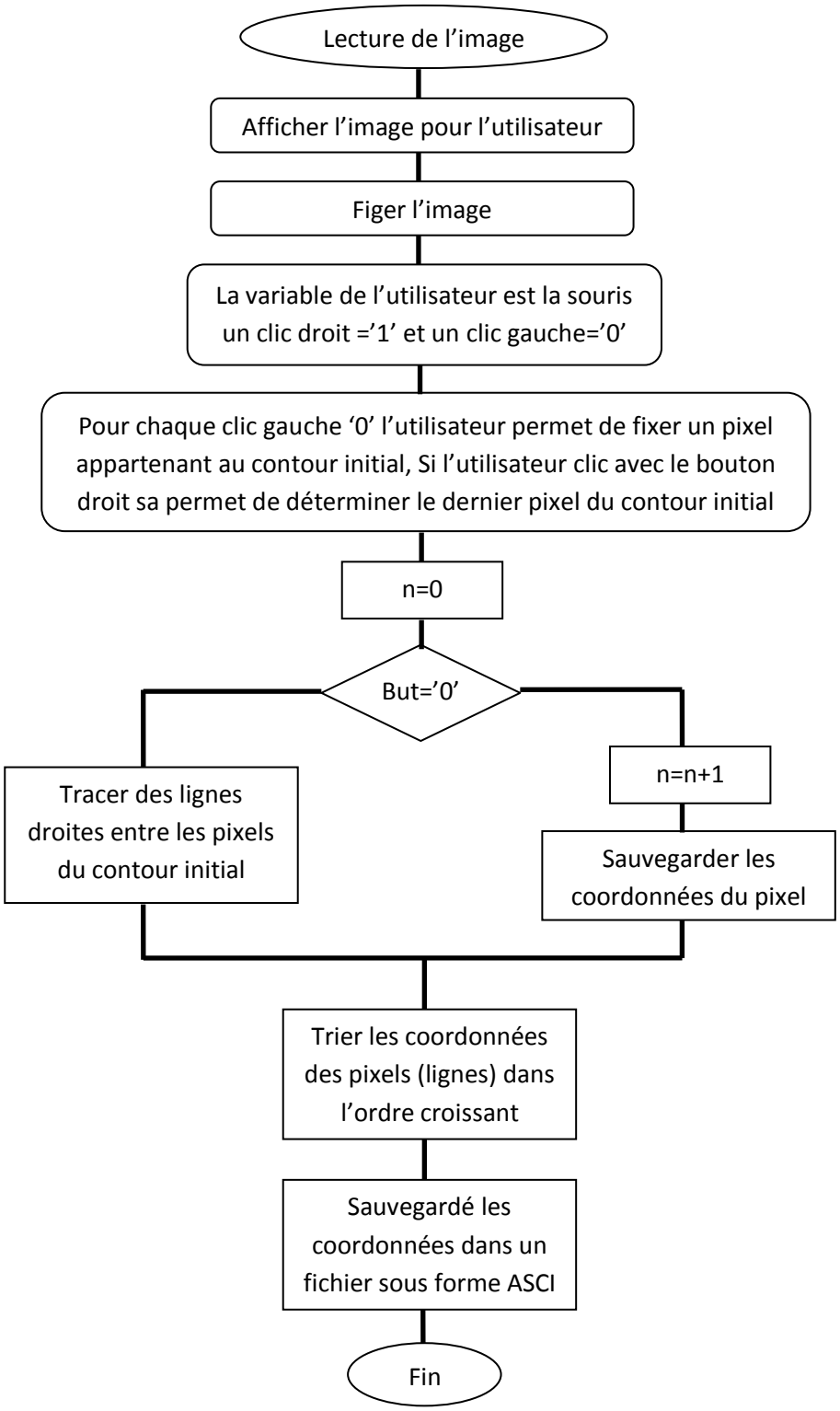


Figure (III.15) : Organigramme d'acquisition du contour initial

IV.7. Sauvegarde des positions des pixels du contour

Après avoir sauvegardé les positions des pixels du contour initial dans un fichier texte, on va lire ce dernier avec une description VHDL afin de sauvegarder les positions dans une RAM bloque.

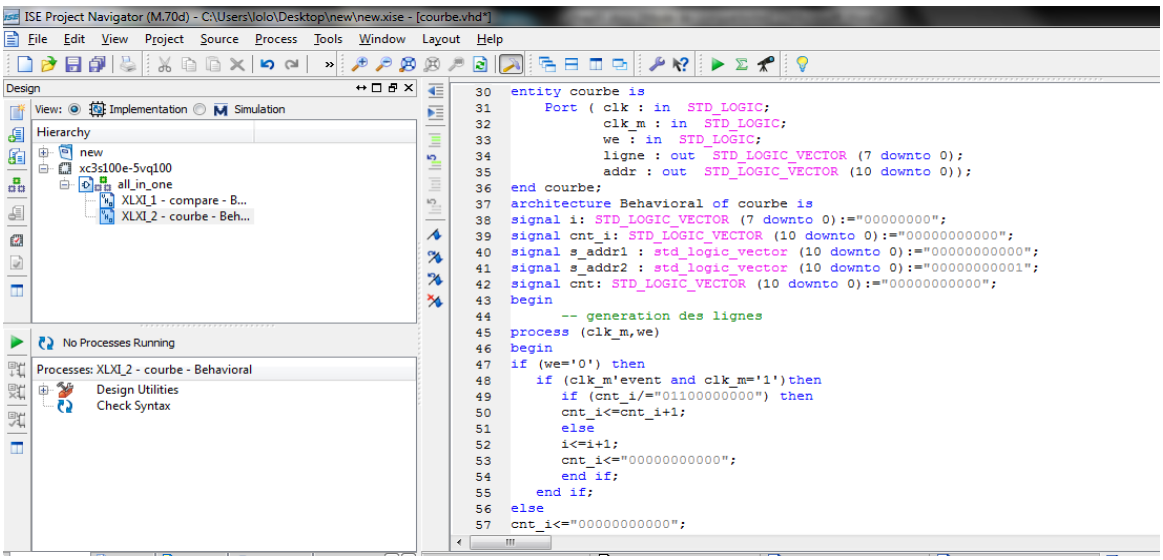
Lors de la lecture de l'image, on lit en parallèle le contenue de la RAM et détecter l'existence d'un pixel formant le contour initial.

Sachant que les positions des pixels formants le contour initial sont écrites dans la RAM de la manière suivante :

I
J
I+1
J+1
I+n
J+n

Avec « I » la position selon x du premier pixel formant le contour initial, et « J » la position selon y du premier pixel formant le contour initial.

Alors pour détecter la position selon x il faut lire la RAM avec un pas d'adresse de 1 et pour lire la position selon y il faut lire la RAM avec un pas d'adresse de 2 ; pour cela on a fait une description en VHDL assurant cette adressage, voir la figure (III.16).



Figure(III.16) : description VHDL d'adressage de la RAM de sauvegarde du contour initial.

Les sorties de ce module vont être les entrées d'un autre module décrit en VHDL permettant de la position lu de la RAM avec la position du pixel arrivé de l'image et d'indiquer si on est dans la présence un pixel formant le contour initial ; la figure (III.17) illustre la description VHDL de ce module.

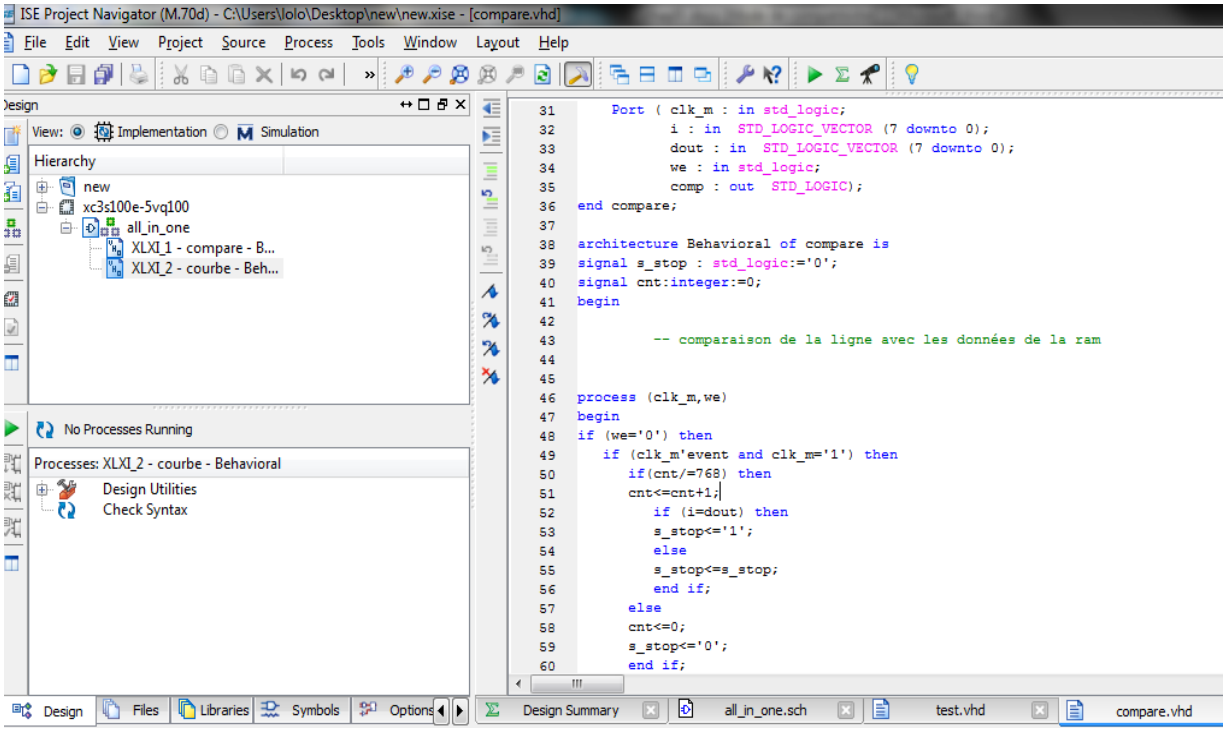


Figure (III.17) : description VHDL du comparateur

Pour la lecture et l'écriture dans la RAM on a utilisé deux fréquences différentes afin de lire les positions à partir de la RAM plus rapidement au début de chaque ligne de l'image ; pour cela on a utilisé un DCM (Digital Clock Manager) qui nous permet d'avoir une fréquence multiplier, et un deuxième DCM qui nous permet d'avoir une horloge déphasé de 90° afin de comparer les positions de la RAM avec les pixels de l'image pour qu'il n'y aura pas d'erreur.

On a rassemblé les deux modules décrit en VHDL avec la RAM et les deux DCM en schématique, voir la figure (III.18).

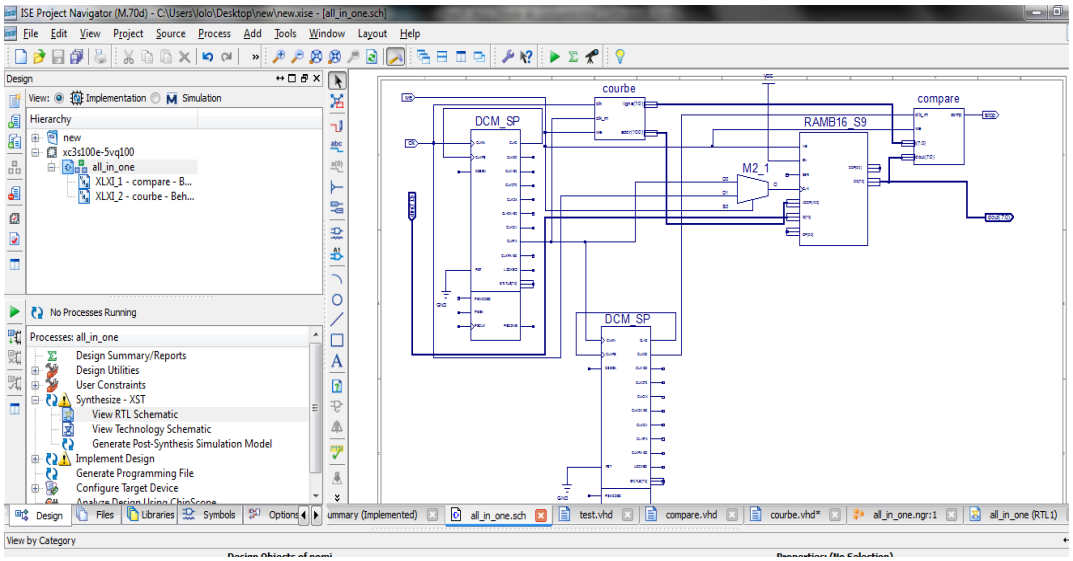


Figure (III.18) : schématique du module de détection des pixels du contour initial

IV.8. Calcul des énergies

IV.8.1. Energie interne

L'énergie interne correspond à la morphologie et aux caractéristiques de la courbe telles que la courbure, la longueur, etc. donc pour la calculer on aura besoin des données sur la courbe entière à chaque moment d'évolution de la courbe, Kass propose d'utiliser le masque laplacien pour calculer l'énergie interne.

Le masque laplacien possède deux masques, un d'ordre 1 et un d'ordre 2 qui sont respectivement :

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Pour son calcule on a utilisé l'architecture de convolution en 2D expliquer précédemment.

IV.8.2. Calcul de l'énergie externe

$$E_{exter} = w_{ligne} E_{ligne} + w_{contour} E_{contour} + w_{term} E_{term}$$

L'énergie ligne est tout simplement l'intensité du pixel

$$E_{ligne} = I(x, y)$$

L'énergie contour est l'opposé du gradient d'intensité élevé au carrée

$$E_{contour} = -|\nabla I(x,y)|^2$$

Le calcul du gradient d'intensité a été détaillé dans le chapitre III section 2.2

L'énergie terminaison :

$$E_{term} = \frac{C_{xx}C_y^2 - 2C_{xy}C_xC_y + C_{xx}C_y^2}{(C_x^2+C_y^2)^{3/2}}$$

Cette énergie est composé de cinq variable (C_x,C_y, C_{xx}, C_{yy} $et C_{xy}$), chaque variable est calculé séparément et en même temps que les autre afin de gagner du temps et d'avoir un résultat en temps réel.

Chaque variable est obtenu par un produit de convolution de l'image avec un masque comme suit :

$$C_x = [-1 \quad 1] \otimes I(x,y)$$

$$C_y = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \otimes I(x,y)$$

$$C_{xx} = [1 \quad -2 \quad 1] \otimes I(x,y)$$

$$C_{yy} = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} \otimes I(x,y)$$

$$C_{xy} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \otimes I(x,y)$$

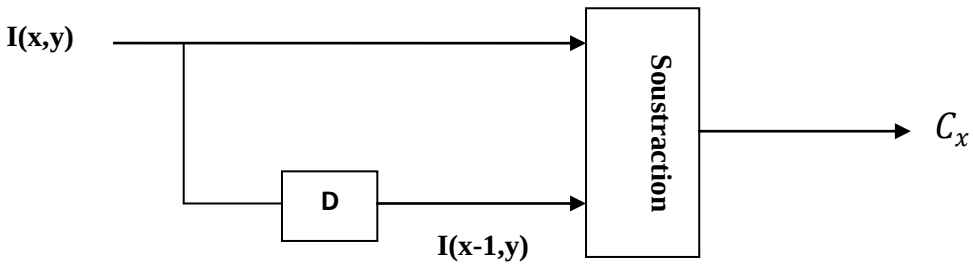


Figure (III.19) : schéma synoptique d'une convolution 1D selon x avec deux coefficients.

- Pour obtenir la variable C_x en temps réel on sauvegarde toujours un pixel en utilisant un latch, le résultat est obtenu après un top d'horloge.

Le produit de convolution 1D pour obtenir C_x ce fait de la manière présenté dans la figure (III.19).

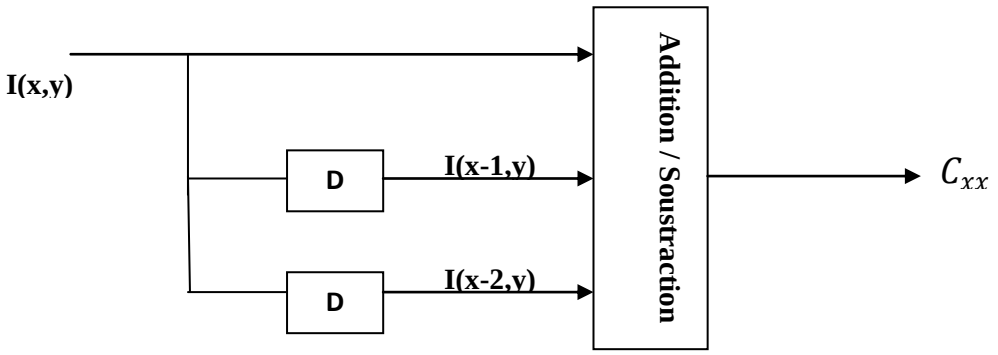


Figure (III.20) : schéma synoptique d'une convolution 1D selon x avec un filtre de trois coefficients.

➢ Pour obtenir la variable C_{xx} en temps réel on sauvegarde toujours deux pixels en utilisant des latch, le résultat est obtenu après deux tops d'horloge.

Le produit de convolution 1D pour obtenir C_{xx} ce fait de la manière présenté dans la figure (III.20).

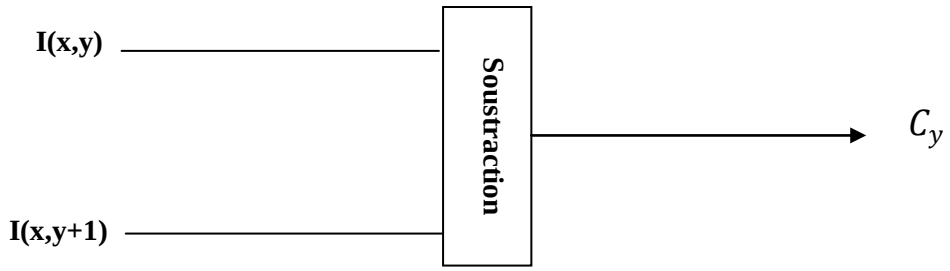
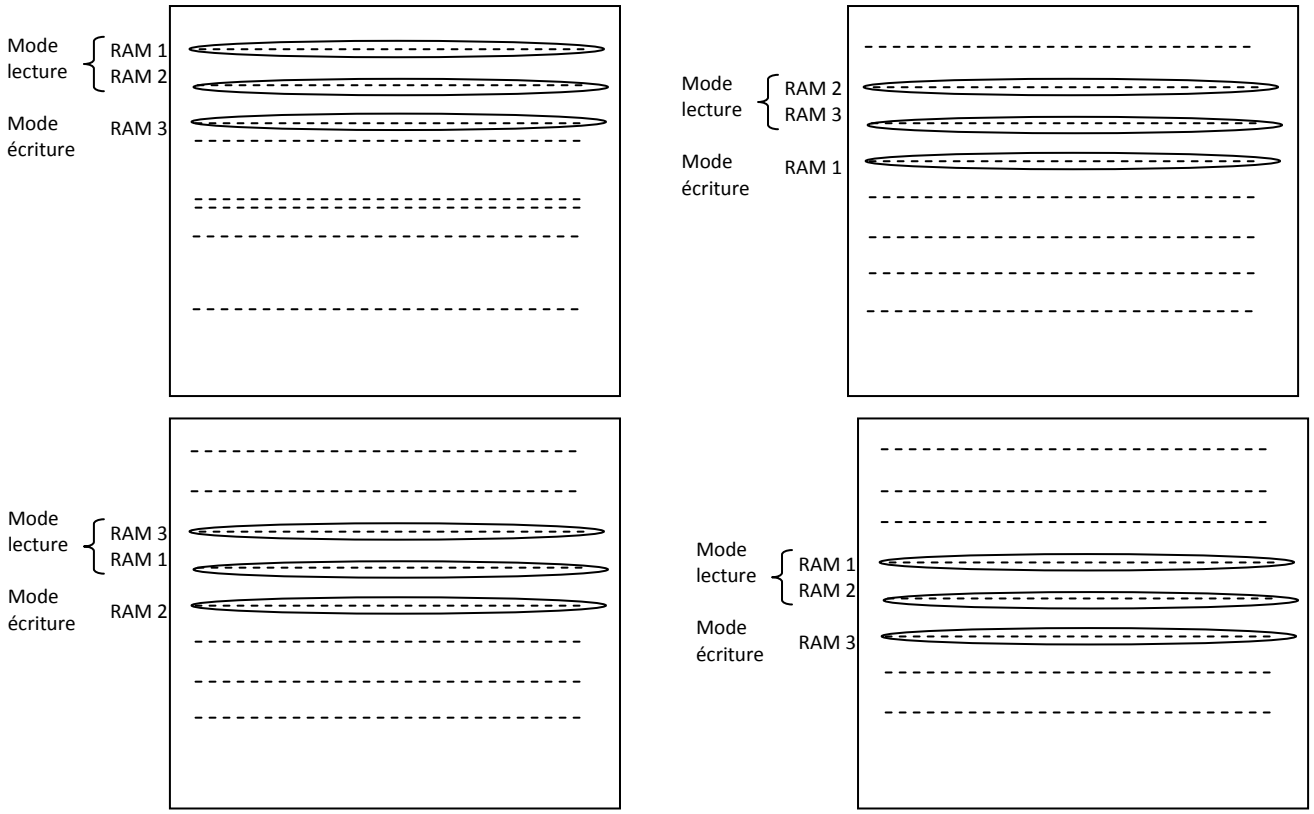


Figure (III.21) : schéma synoptique d'une convolution 1D selon y avec un filtre de deux coefficients.

Le produit de convolution 1D pour obtenir C_y ce fait de la manière présenté dans la figure (III.21).

On a utilisé dans ce module trois RAM bloque afin de sauvegarder au minimum trois lignes, les deux première RAM en mode lecture pour faire la convolution et la troisième en mode écriture pour l'utilisé lors de la prochaine convolution, la figure illustre le fonctionnement de cette architecture.



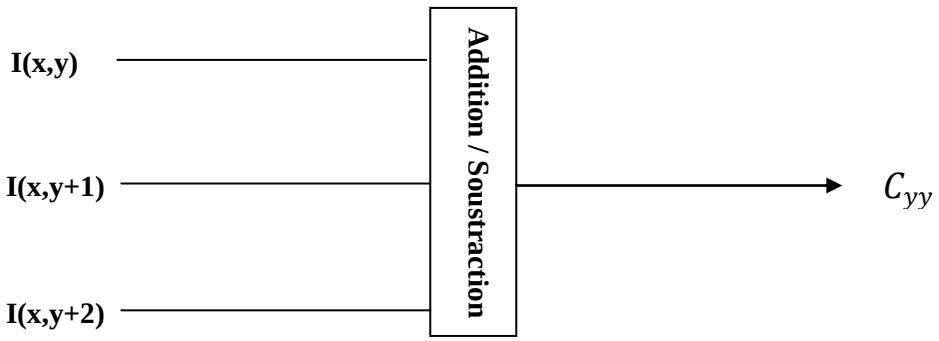


Figure (III.22) : schéma synoptique d'une convolution 1D selon y avec un filtre de trois coefficients.

- Le produit de convolution 1D pour obtenir C_{yy} ce fait de la manière présenté dans la figure (III.22).

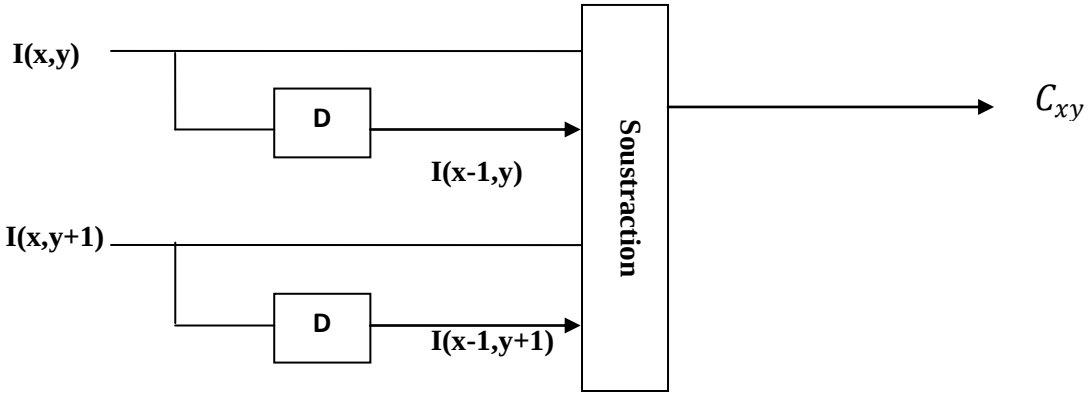


Figure (III.23) : schéma synoptique d'une convolution 2D selon y avec un filtre de quatre coefficients

La description VHDL de ce module est presque similaire à la précédente sauf qu'il y a une autre RAM bloque de plus dans ce module afin d'avoir trois RAM en mode lecture pour faire la convolution et une RAM en mode écriture.

- Le produit de convolution 2D pour obtenir C_{xy} ce fait de la manière présenté dans la figure (III.23).

Dans la description VHDL de ce module on a utilisé trois RAM bloque, deux en mode lecture pour faire la convolution et une en mode écriture.

Pour calculer le numérateur il suffit de multiplier et d'additionné le résultat obtenue des cinq variables sauf que un problème est apparue ?

Pour obtenir le premier résultat de la variable C_x il faut sauvegarder au moins un pixel alors le résultat est obtenue après un top d'horloge.

Pour obtenir le premier résultat de la variable C_{xx} il faut sauvegarder au moins deux pixels alors le résultat est obtenue après deux tops d'horloge.

Pour obtenir le premier résultat de la variable C_y il faut sauvegarder au moins une ligne et un pixel alors le résultat est obtenu après n+1 tops d'horloge (n représente la largeur de l'image).

Pour obtenir le premier résultat de la variable C_{yy} il faut sauvegarder au moins deux lignes et un pixel alors le résultat est obtenu après 2n+1 tops d'horloge (n représente la largeur de l'image).

Pour obtenir le premier résultat de la variable C_{xy} il faut sauvegarder au moins deux lignes et deus pixels alors le résultat est obtenu après 2n+2 tops d'horloge (n représente la largeur de l'image).

Pour résoudre ce problème on a réalisé un module de ligne à retard permettant de retardé $C_x, C_{xx}, C_y et C_{yy}$ afin d'avoir le résultat en même temps queC_{xy}, ce module a été réalisé en schématique sous l'environnement de développement xilinxise 12.3 avec des registre a décalage LUT existant déjà dans la bibliothèque, la figure (III.24) présente le schématique de la ligne à retard utilisé.

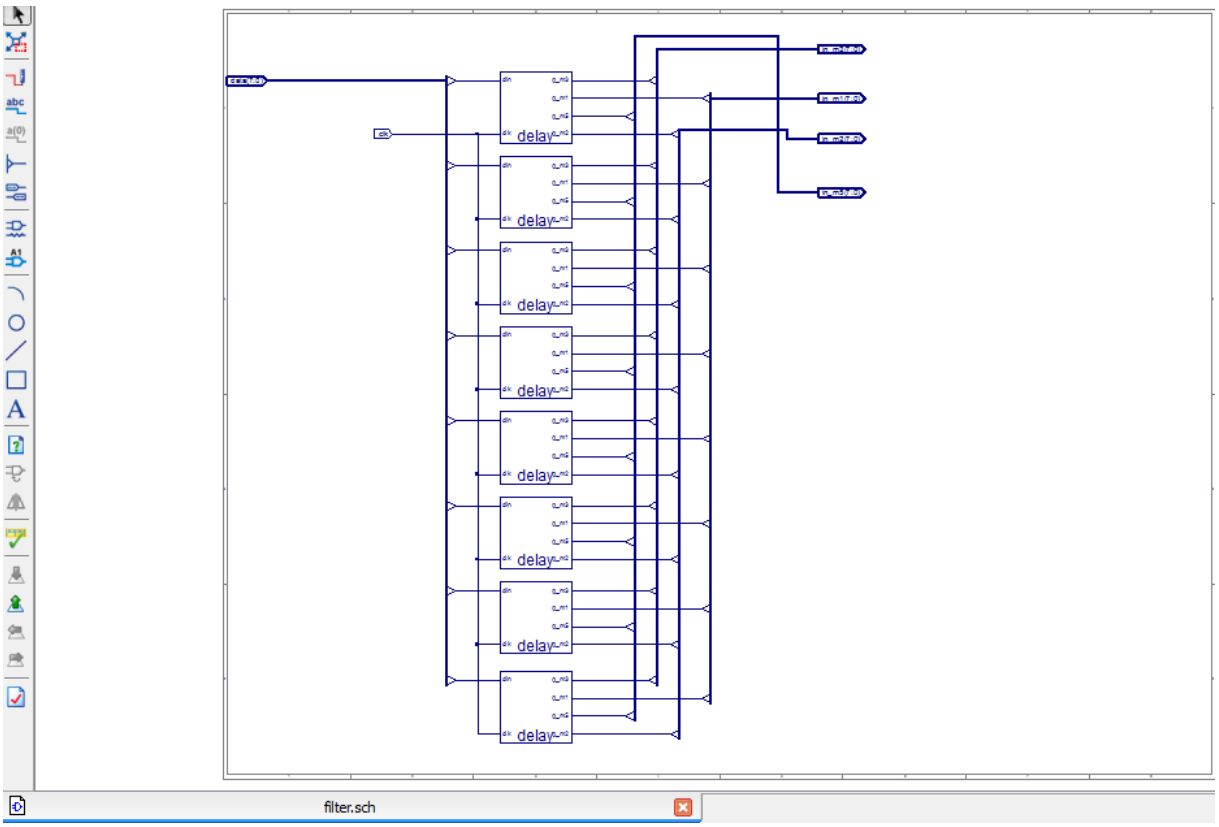


Figure (III.24) : Schématique de la ligne à retard

Après cet étage il suffit de calculer le numérateur avec des multiplications et additions des cinq variables pour cela on a créé un module assurant le calcul du numérateur.

Pour compléter le calcul de E_{term} on a calculé le numérateur et il reste le dénominateur.

Puisque l'implémentation d'une opération de division sur FPGA est une tâche difficile et pend trop de temps on a opté d'implémenter le résultat de $\frac{2^{26}}{(C_x^2+C_y^2)^{3/2}}$ sur une ROM afin de ne pas perdre le temps de calcul et d'économiser les ressources de notre FPGA.

On a multiplié par 2^{26} pour agrandir l'ordre du résultat.

Pour cela on calculer tous les cas possible sur matlab et on les a sauvegardé dans un fichier texte ensuite on a réalisé un mini programme en c++ permettant d'utiliser les données inscrites sur ce fichier et créer un autre fichier contenant la description VHDL de la ROM.

Pour avoir le résultat E_{term} il faut multiplier le résultat du numérateur avec le résultat obtenu de la ROM (sachant que la ligne d'adresse de la ROM est la concaténation de$C_x et C_y$) et en fin faire le décalage de 26 bit pour faire la division sur2^{26}.

Après avoir calculéE_{ligne}, $E_{contour}$ et E_{term} on a fixé dans notre approche

$w_{ligne} = w_{contour} = w_{term} = 1$et on additionné les énergies afin d'avoirE_{exter}.

La figure (III.25) présente le schématique réalisé pour calculéE_{exter}.

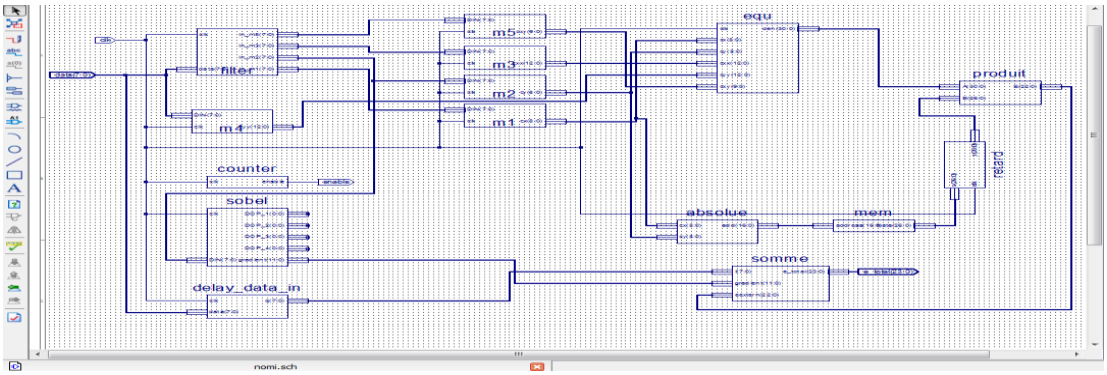


Figure (III.25) : Schématique du circuit réalisé pour le calcul deE_{exter}.

IV.9. Observation des pixels

Comme on a détaillé déjà précédemment dans la section 4.2.7, le module de sauvegarde des pixels permet aussi de détecter la présence d'un pixel formant le contour initial.

Dans le cas de la présence d'un pixel formant le contour, il faut comparer l'énergie de ce pixel avec celle des huit pixels de voisinage, et déplacer la position de ce pixel vers le pixel qui a l'énergie minimale.

A la fin de chaque itération on doit sauvegarder les nouveaux positions des pixels formant le contour, pour cela on a proposé d'utiliser une autre RAM bloque qui a le même rôle que celle utiliser dans la section 4.2.7 et on va les utiliser de la manière suivante :

- Les positions du contour initial sont sauvegardées dans la RAM_1.
- Les positions du contour après la 1ère itération sont sauvegardées dans la RAM_2.
- Les positions du contour après la 2ème itération sont sauvegardées dans la RAM_1.
- Les positions du contour après la n-1 itération sont sauvegardées dans la RAM_2.
- Les positions du contour après la nième itération sont sauvegardées dans la RAM_1.

IV.10. Vérification de la condition d'arrêt

Afin que l'algorithme s'arrête on vérifie la condition d'arrêt qui est le nombre d'itérations limité par l'utilisateur et on écrit le contenue de la RAM (les positions des pixels formant le contour) dans un fichier texte; après on lit ce dernier avec le matlab et on affiche le contour afin de vérifier s'il a convergé ou non.

Conclusion

Dans ce chapitre nous avons présenté le principe de fonctionnement des contours actifs, et nous avons détaillé l'architecture développée pour l'implémentation ainsi que les différents composants (avec leur description VHDL) que nous avons conçus et utilisées pour l'implémentation. Dans le chapitre suivant nous allons présenter les résultats de simulations obtenues et nous les commenterons.

Introduction

Pour implémenter un circuit sur un FPGA il faut qu'on dispose une description logique du circuit (schématique, diagramme d'état ou d'une description VHDL), et aussi d'un environnement de développement qui est choisi en fonction de composant sur lequel le circuit sera implémenté spécialement le logiciel du fabricant (ISE).

Des partie des descriptions VHDL ont été présenté dans le chapitre précèdent.

Dans ce chapitre, il sera présenté l'environnement de développement de XILINX et les résultats de l'implémentation et de simulation.

I. Présentation de l'environnement de développement XILINX ISE

C'est le logiciel de programmation produit par XILINX (CPLD, FPGA, Spartan et Virtex…) téléchargeable gratuitement sur le site internet de XILINX (dans sa version web pack). Il intègre différents outils permettant de passer à travers le flot de conception d'un système numérique. Il dispose de :

- Un éditeur de textes, de schémas et diagrammes d'états.
- D'un compilateur VHDL et verilog.
- D'un simulateur.
- D'outils pour la gestion des contraintes temporelles.
- D'outils pour la synthèse.
- D'outils pour la vérification.
- D'outils pour l'implémentation sur FPGA.

C'est un outil de développement complet pour toutes les gammes de produits XILINX

I.1. Les étapes pour l'implémentation d'une spécification HDL sur un FPGA

Pour implémenter une spécification HDL sur un FPGA, on suit quatre étape importante sur la plateforme ISE, qui son illustré dans la figure (IV.1).

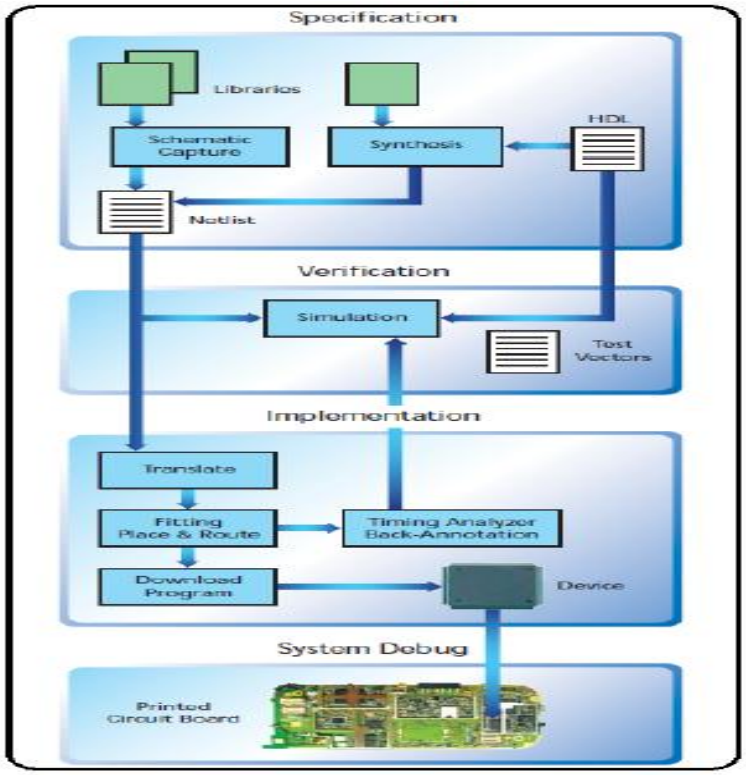


Figure (IV.1) : Les étapes pour l'implémentation d'une spécification HDL sur un FPGA

Spécification

Le terme de spécification est un terme qui regroupe les trois modes (schématique, diagramme d'état ou HDL) de saisie d'un circuit électronique. La spécification HDL est synthétisé pour générer un fichier appelé NETLIST qui décrit les interconnexions entre les registres.

Vérification

La vérification du design est une étape parallèle ou le concepteur observe le comportement du code et observe s'il se comporte tel qu'il est supposé. Un simulateur simule le circuit à condition de lui fournir les vecteurs de test. Les vecteurs de teste peuvent se présenter sous plusieurs formes, la plus courante est les TEST BENCHS rédigé dans un langage de description matériel pour entrer les instructions au simulateur. En appliquant les vecteurs de test sur le code, le simulateur fournit des sorties du circuit.

Implémentation

Une fois le Netlist (les interconnexions entre les portes logiques) décrit la conception en utilisant les portes logiques en tenant compte du fabricant et d'une famille de composants bien définie. Une fois la vérification est terminée, le circuit est

implémenter sur le composant en spécifiant les références exactes de celui-ci à savoir : le boitier, la fréquence de travail et les autres options spécifiques à chaque composant. Cette étape ce termine par un rapport de tous les sous-programmes exécutés, les warning, les erreurs, les I/O utilisés, des données qui permettent de savoir si le composant choisit et le mieux adapté pour l'application ciblée.

Débuggage du système

Après chargement des interconnexions sur le FPGA, des tests peuvent être effectué sur le circuit implémenté afin de voir les réactions du circuit et son comportement et détecter les anomalies afin de les corriger. En cas d'anomalie la spécification est revue et corrigée en conséquence. Si les tests physiques s'avèrent concluants, la spécification est validée et le prototype est considéré opérationnel.

Interface graphique de l'environnement ISE 12.3

L'ISE contrôle tous les aspects d'un design flow. L'interface Project Navigator donne accès à toutes les ressources d'un projet et aux outils de l'implémentation. Elle procure aussi un accès aux fichiers et documents associer au projet (voir figure IV.2).

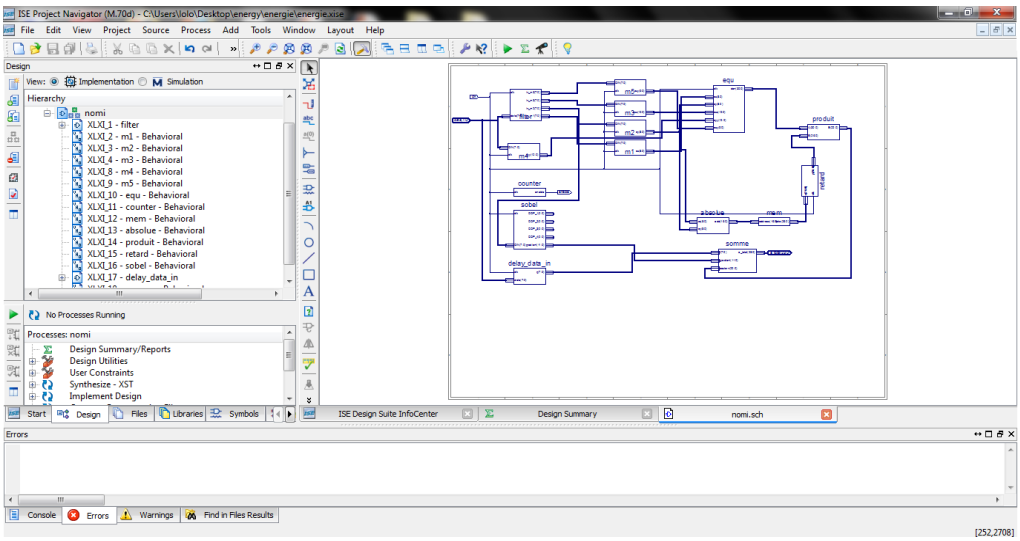


Figure (IV.2). Project Navigator de l'environnement de développement ISE 13.2

L'interface du Project Navigator se divise en quatre sous-fenêtres principales :

Source Windows: Elle affiche les éléments inclus dans le projet de façon hiérarchique

Process Windows: affiche les processes disponibles pour la source sélectionnée.

Transcript Windows: affiche les messages d'état, les erreurs et les warning.

MDI (multi documents interface) Windows : c'est l'espace de travail, elle permet de visualiser les rapports HTML, les fichiers textes ASCII, les schématiques et la fenêtre de simulation.

I.2. synthèse et implémentation

L'étape de synthèse reçoit en entrée le code VHDL ou le schématique après vérification et simulation et le fichier de contraintes.

L'étape de synthèse génère quatre fichiers :

I.2.1. xilinx Specific file au format .NGC

C'est le fichier principal de la synthèse et il comporte les données logiques qui constituent la conception et les contraintes.

I.2.2. RTL (Registre Transfert Level) Shematic

C'est un schéma représentatif de la conception pré-optimisée au niveau RTL. C'est une représentation en symboles génériques tels des additionneurs, multiplexeurs, compteurs ….etc. il est généré à la fin de l'étape de synthèse, il permet d'avoir un aperçu du circuit tout au début du processus d'implémentation.

I.2.3.Technologie Schématique

C'est une représentation schématique du fichier NGC, elle figure en termes d'éléments logiques optimisés pour une architecture ou une technologie bien définie. Par exemple, schématique avec des LUTs, buffers d'I/O …etc. Elle est générée après l'étape d'optimisation et la spécification de la technologie utilisée par le processus de synthèse. Ce fichier permet de visualisé la conception au niveau technologique du code VHDL. A ce niveau figure le schéma tel qu'il sera implémenté sur le composant FPGA.

I.2.4. Le fichier LOG

C'est un rapport qui contient les résultats de la synthèse. Il contient toutes les informations relatives au fichier d'entrée, le nom du fichier et la prise en charge des contraintes. Des informations relatives au fichier de sortie, le nom du fichier et son format NGC. Les messages d'erreurs et warning. Ainsi que les ressources internes nécessaires pour l'implémentation.

I.2.5. Le rapport de synthèse

L'environnement de développement ISE, fournit un rapport de synthèse sous forme de tableaux contenant les informations utiles liées au design.

I.3. Simulation et résultats d'implémentation de l'algorithme de SNAKE

Dans cette section nous allons présenter les schémas représentatifs RTL de chaque code VHDL ou schématique développé lors de la réalisation de notre projet.

Comme il a été détaillé dans le chapitre précèdent, afin d'implémenté l'algorithme des SNAKE on l'a décomposé en plusieurs modules comme suit :

I.3.1. Schéma RTL du circuit détection de contour par masque de laplacien

Nous avons utilisé pour le calcul de l'énergie interne le masque de laplacien. La figure (IV.3) montre le schéma RTL de la détection du contour 2D par ce masque.

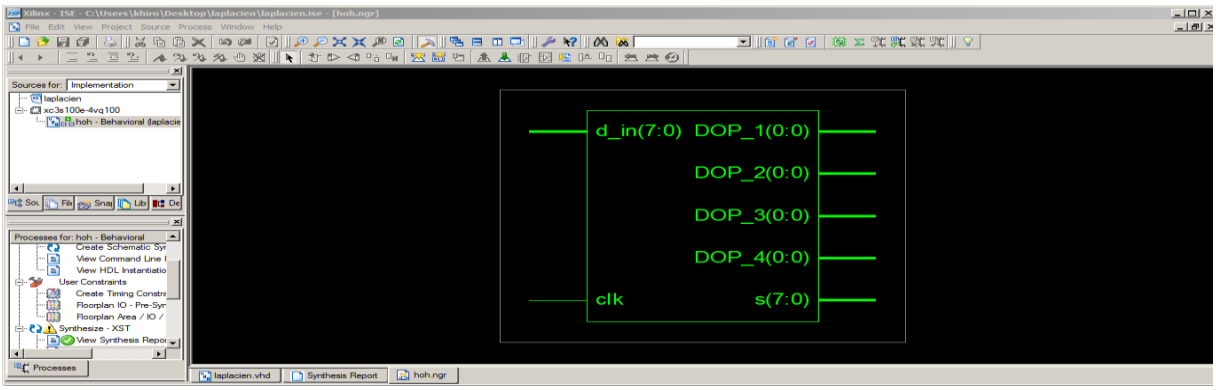


Figure (IV.3) : Schéma RTL du circuit détection de contour par masque de laplacien.

I.3.2. Sauvegarde des positions des pixels du contour

La figure (IV.4) représente le schéma RTL mis en place pour sauvegarder les positions des pixels dans une RAM et détecter la présence d'un pixel formant le contour initial.

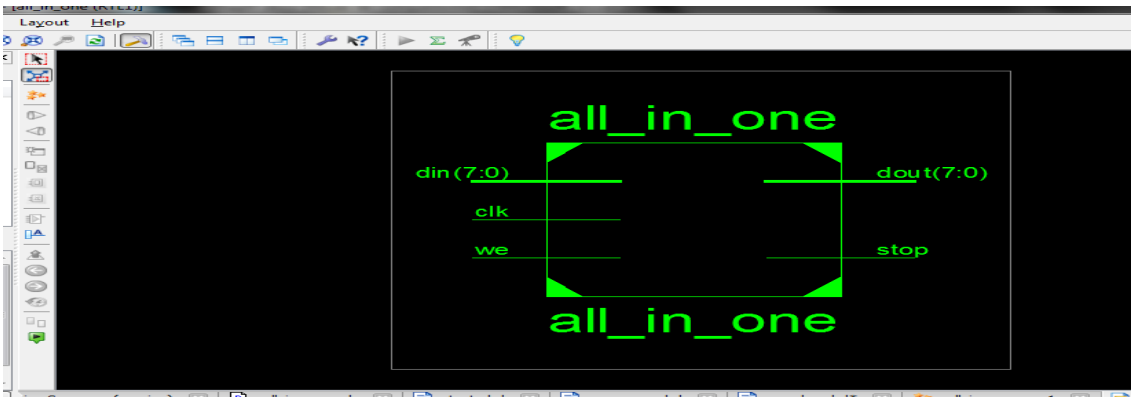


Figure (IV.4) : Schéma RTL du circuit de détection de contour initial

La figure (IV.5) illustre le résultat de la simulation de ce module sous modelsim.

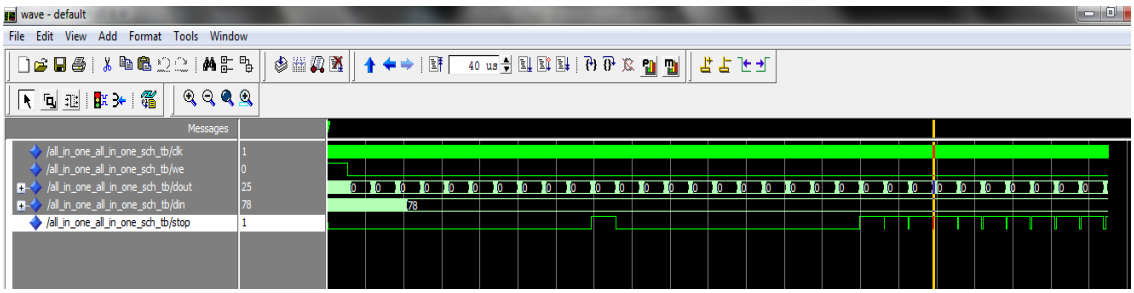


Figure (IV.5) : résultat de simulation du circuit de détection de contour initial

On remarque qu'à chaque présence d'un pixel formant le contour initial on aura un état haut pendant la durée de cette ligne.

Pour le calcul de l'énergie externe on a utilisé les modules suivants :

Calcul de C_x

La figure (IV.6) illustre le schéma RTL du module.

La figure (IV.7) illustre le résultat de simulation sous modelsim avec une entrée DIN qui provient d'une image brut de taille 512*512 sauvegardé dans le PC

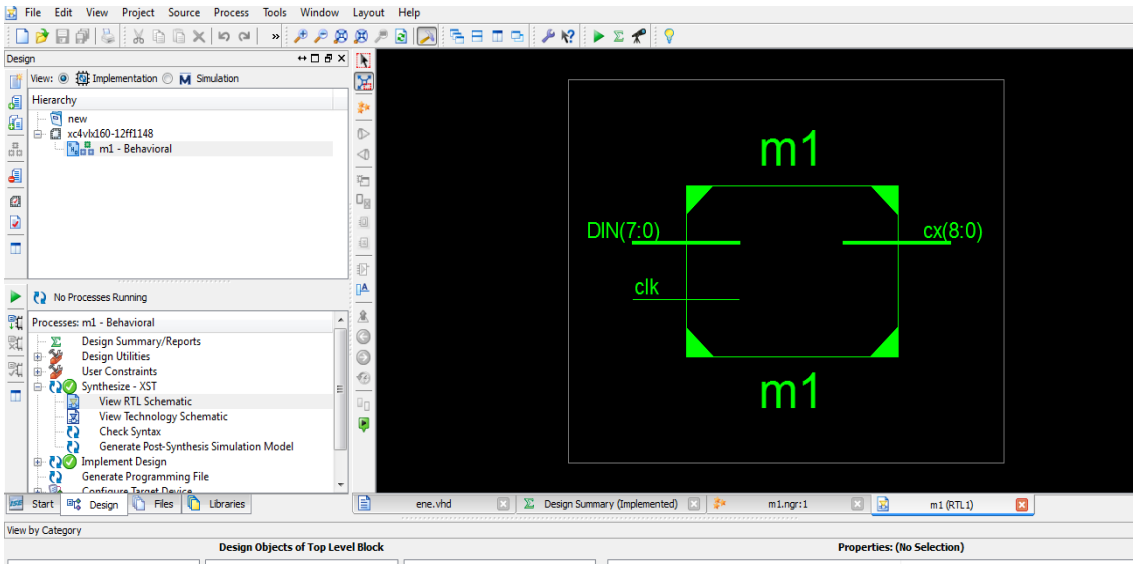
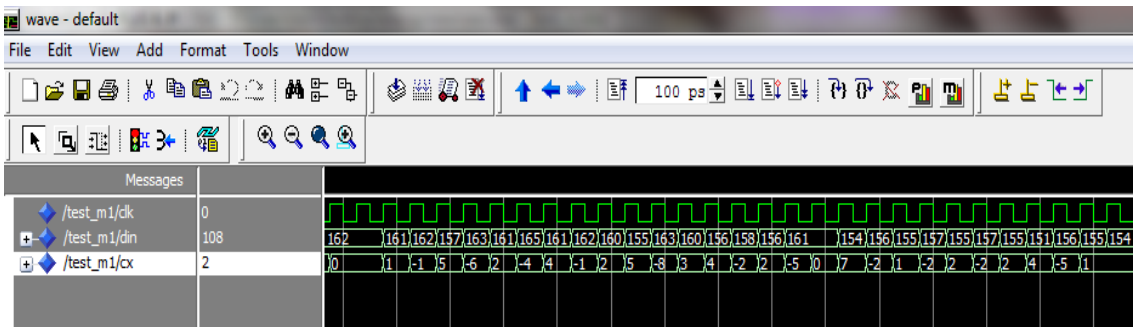


Figure (IV.6) : Schéma RTL du module de calcul de C_x



Figure(IV.7) : résultat de simulation du module de calcul de C_x

Calcul de C_y

La figure (IV.8) illustre le schéma RTL du module.

La figure (IV.9) illustre le résultat de simulation.

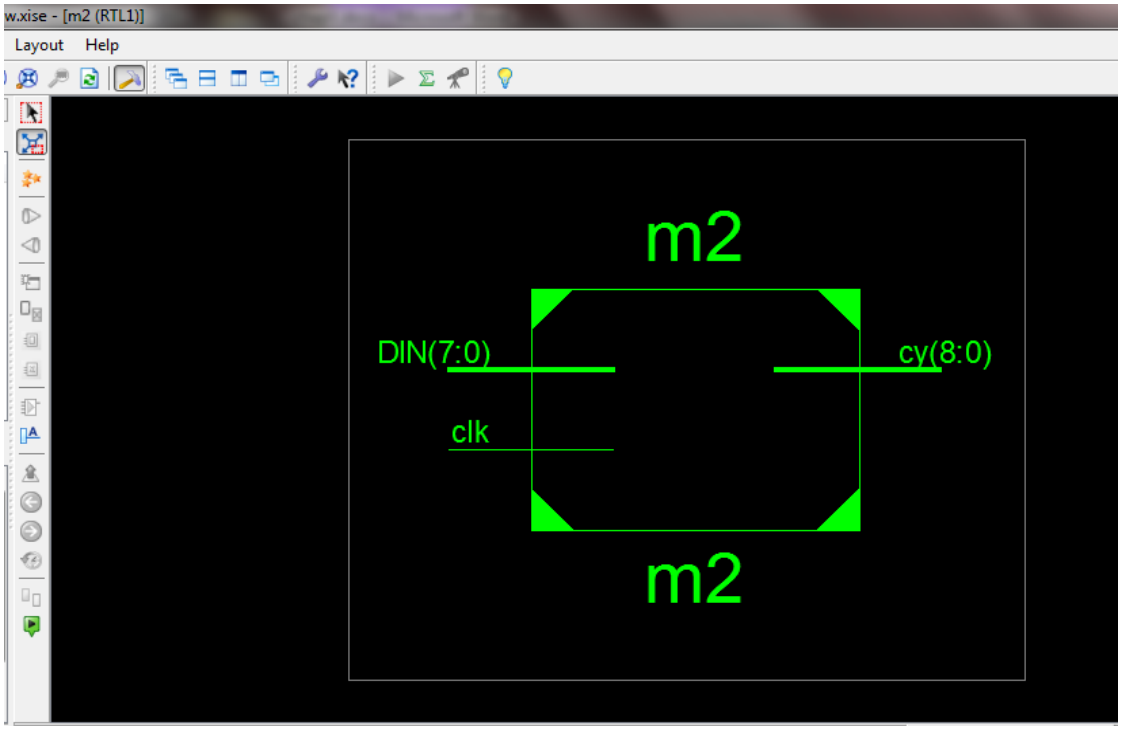


Figure (IV.8) : Schéma RTL du module de calcul de C_y

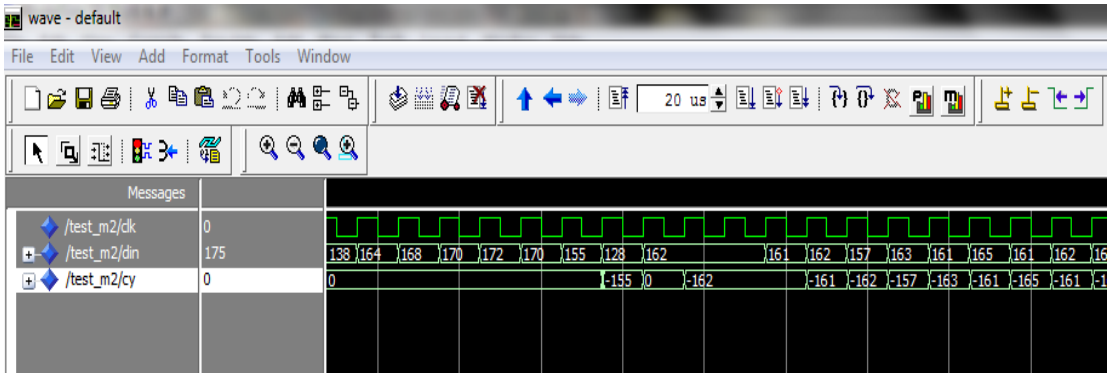


Figure (IV.9) : résultat de simulation du module de calcul de C_y

Calcul de C_{xx}

La figure (IV.10) illustre le schéma RTL du module.

La figure (IV.11) illustre le résultat de simulation.

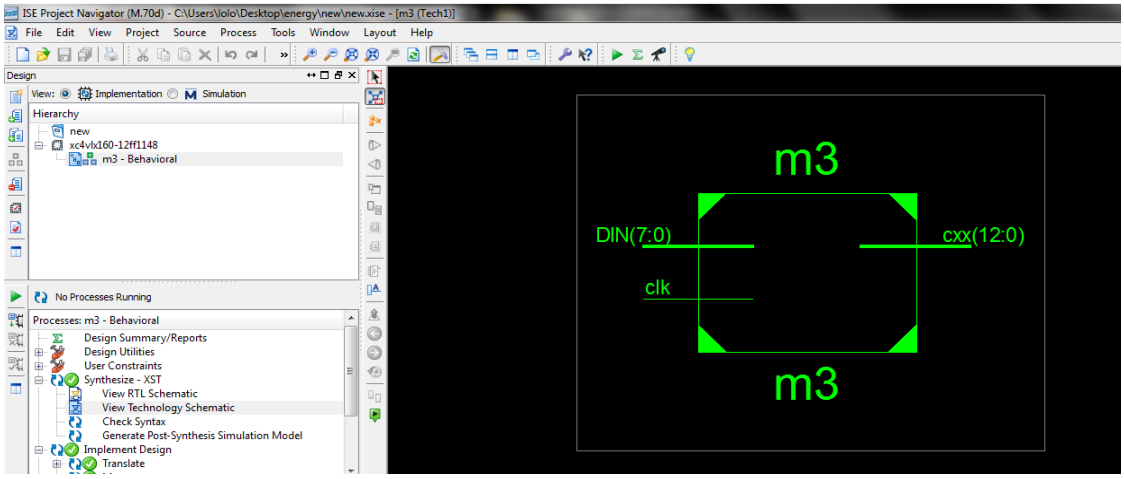


Figure (VI.10) : Schéma RTL du module de calcul de C_{xx}

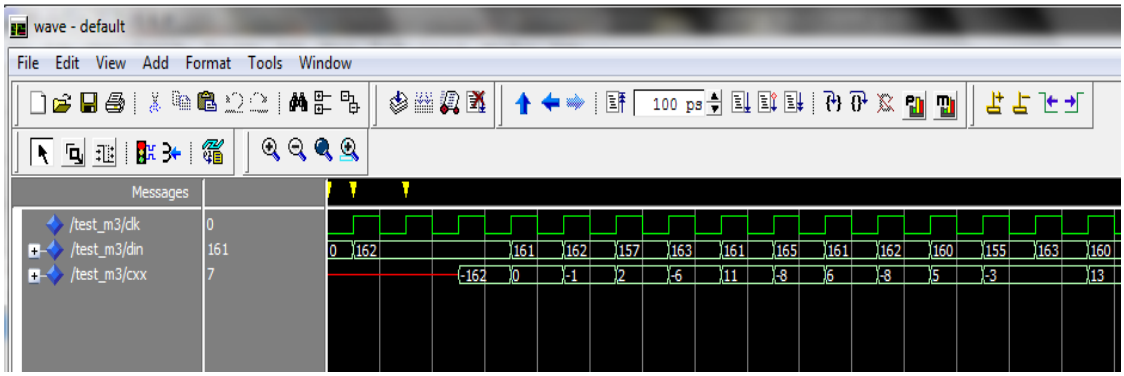


Figure (IV.11) : résultat de simulation du module de calcul de C_{xx}

Calcul de C_{yy}

La figure (IV.12) illustre le schéma RTL du module.

La figure (IV.13) illustre le résultat de simulation.

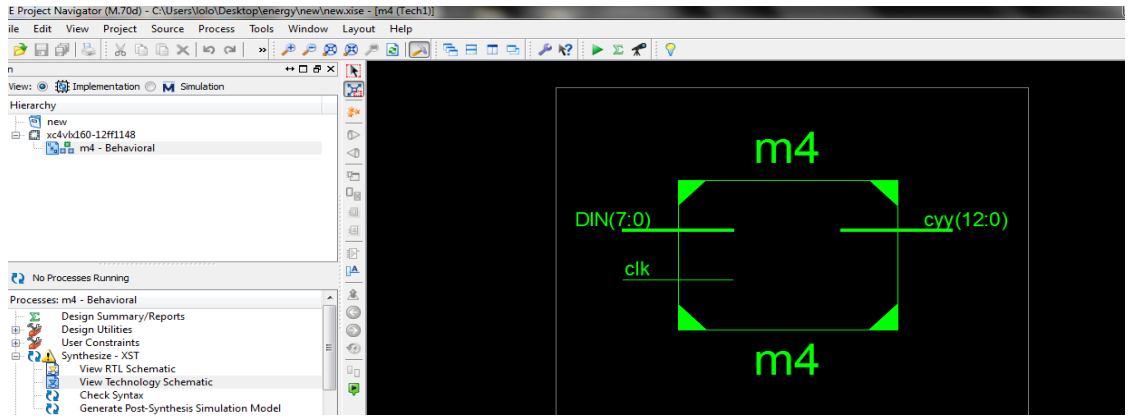


Figure (IV.12) : Schéma RTL du module de calcul de C_{yy}

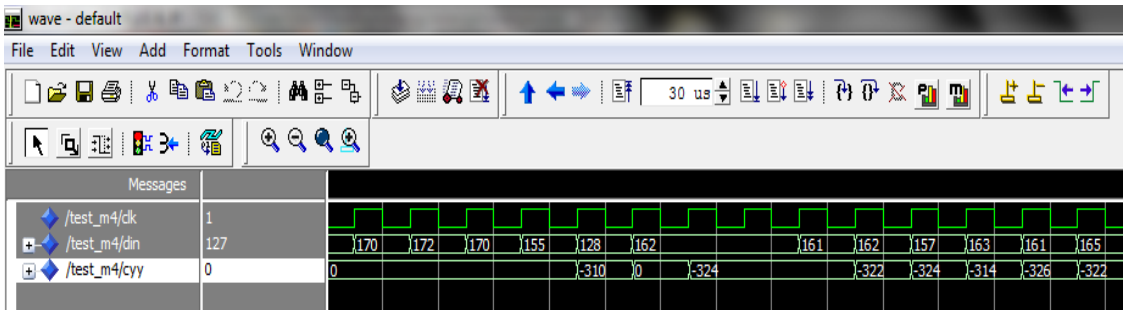


Figure (IV.13) : résultat de simulation du module de calcul de C_{yy}

Calcul de C_{xy}

La figure (IV.14) illustre le schéma RTL du module.

La figure (IV.15) illustre le résultat de simulation.

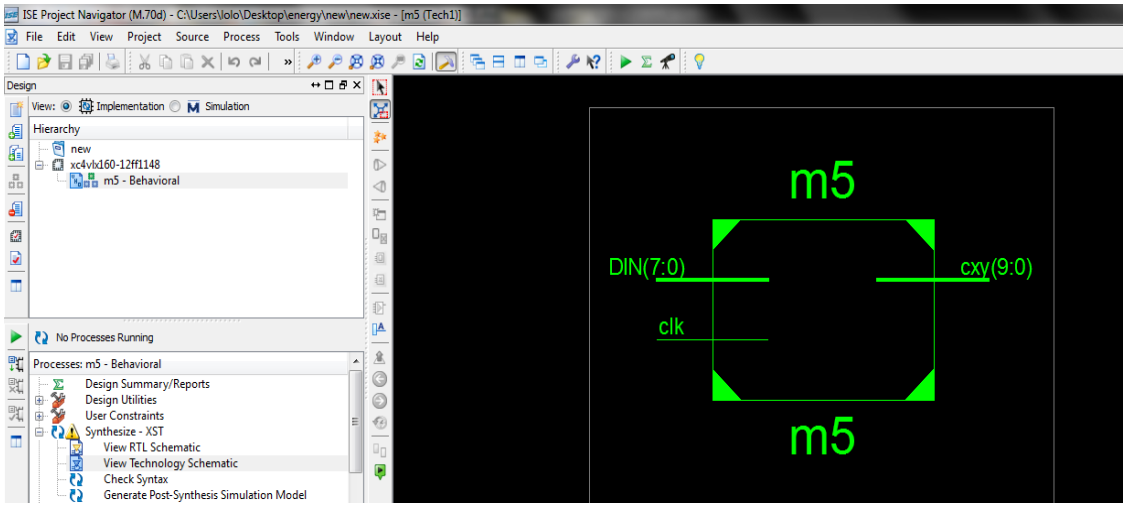


Figure (IV.14) : Schéma RTL du module de calcul de C_{xy}

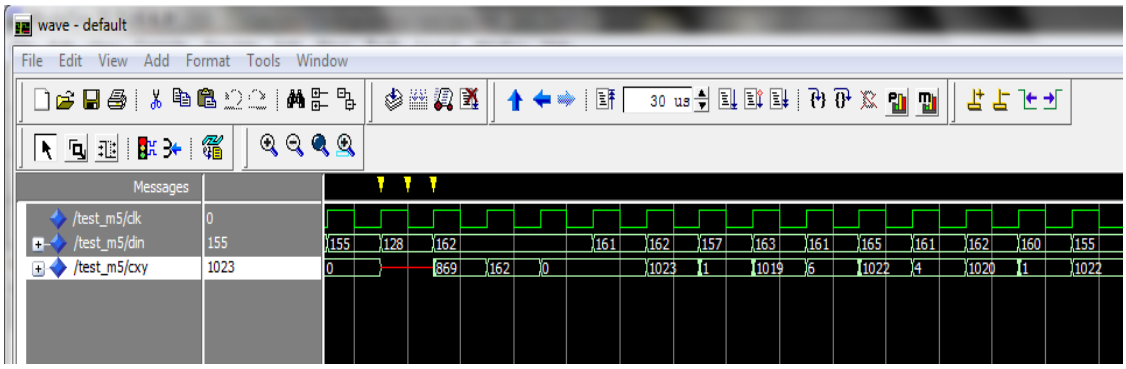


Figure (IV.15) résultat de simulation du module de calcul de C_{xy}

Et comme il a été détaillé dans le chapitre précédent on avait besoin d'une ligne à retard pour synchroniser les résultats obtenue parC_x, C_y, C_{xx}, C_{yy} et C_{xy}

La figure (IV.16) présente le schéma RTL du module de la ligne à retard quia été réalisé en schématique.

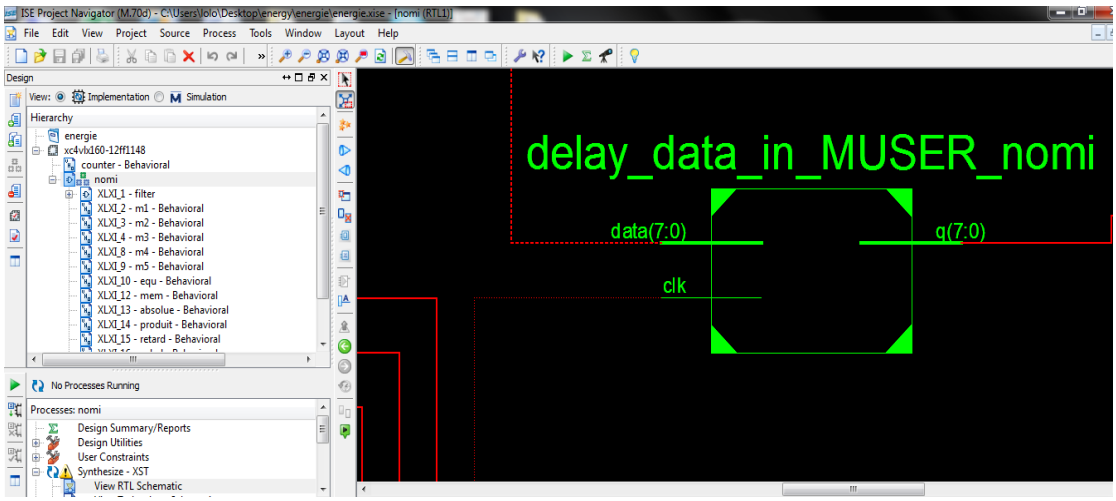


Figure (IV.16) : Schéma RTL du module de la ligne à retard

La figure (IV.17) illustre le schéma RTL de la ROM utilisé pour sauvegarder tous les cas possible du dénominateur.

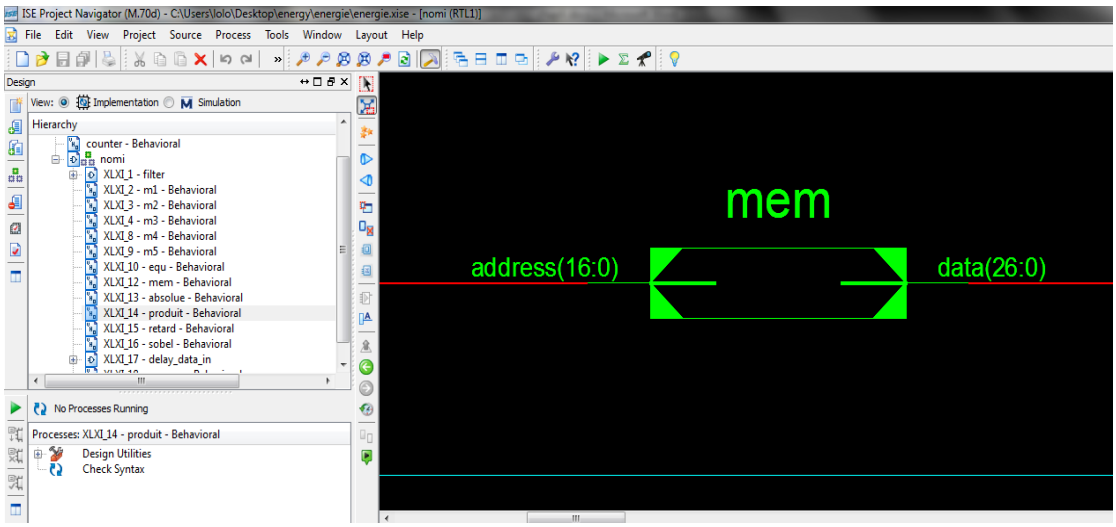


Figure (IV.17) : Schéma RTL de la ROM

La figure (IV.18) illustre le schéma RTL global qui calcul l'énergie terminaison.

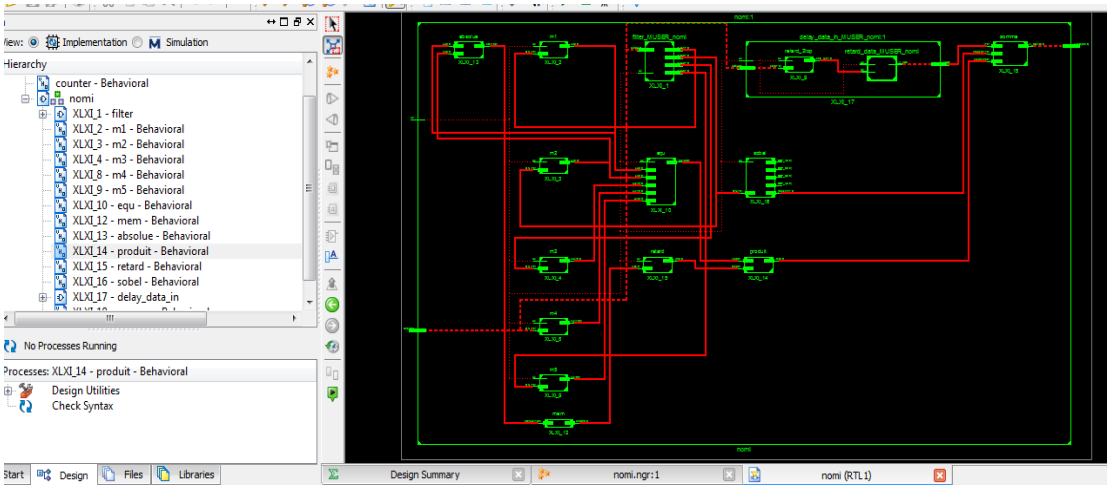


Figure (IV.18) : schéma RTL global pour le calcul de E_{term}

Pour le calcul de $E_{contour}$ on a réalisé un module composé de trois sous modules :

1- calcul du sobel horizontal

2- calcul du sobel vertical

3- calcul de la somme des valeurs absolue de sobel horizontal et sobel vertical.

La figure (IV.19) présente une partie de ce module d'où l'on peut visualiser l'utilisation des RAMs bloque.

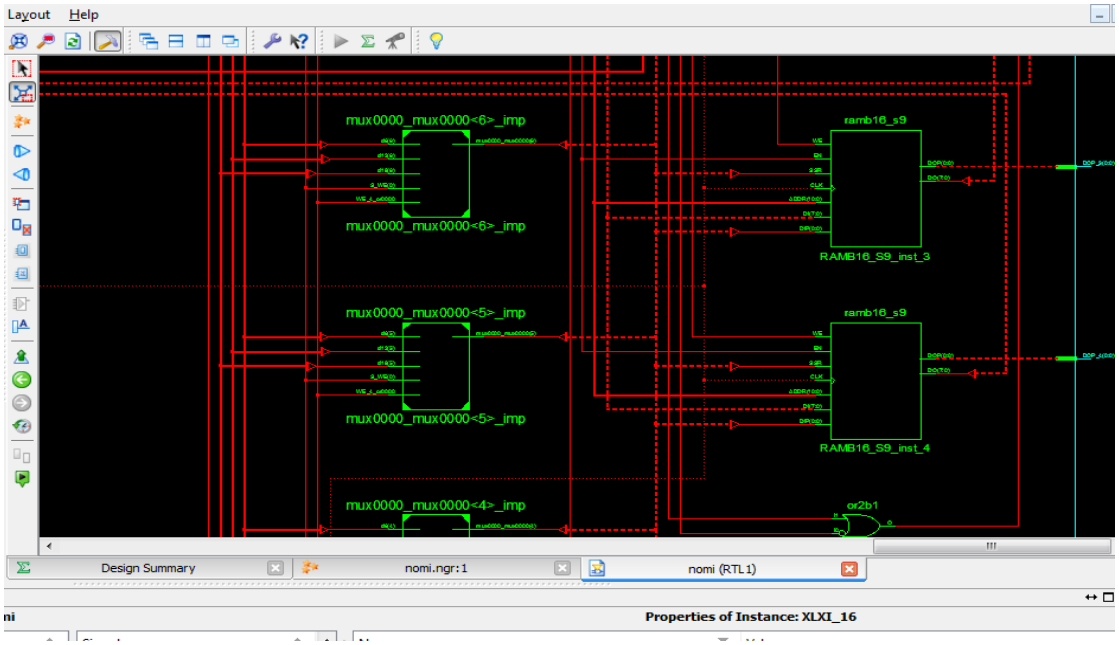


Figure (IV.19) : schéma RTL technologie du module pour le calcul de gradient

Pour calculer l'énergie externe reste à sommer les trois énergies dont E_{ligne} est l'image elle-même.

Ce dernier module de sommation, on la décrit en VHDL, la figure (IV.20) illustre le schéma RTL de ce module.

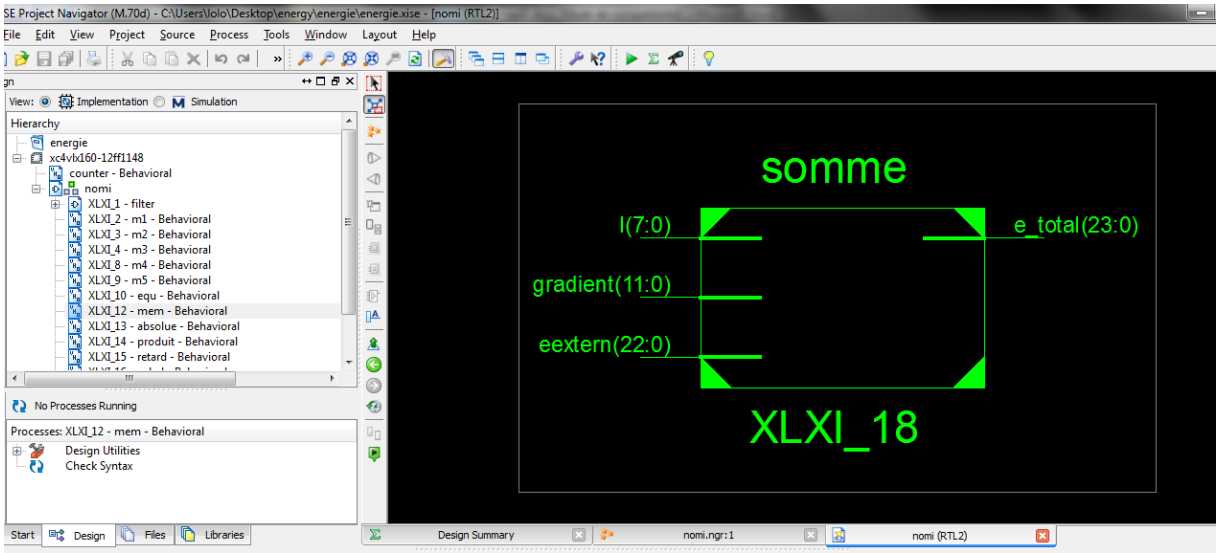


Figure (IV.20) : Schéma RTL du module de sommation

Tous les résultats obtenus ont été vérifiés et comparés avec les résultats obtenus sous matlab.

Pour la vérification de la condition d'arrêt le module n'a pas été réalisé vue le temps que prend une seul itération dans une simulation sur PC (environs 40 heures sous système d'exploitation linux scientifique et un processeur core2duo et 4GB de mémoire).

Conclusion

Ce dernier volet de cette étude nous permet de conclure que les résultats obtenus démontrent la justesse de l'architecture proposé et développé, et que cette méthode de conception permet d'économiser les ressources de notre FPGA par conséquent le cout est réduit, de même cette architecture permet d'avoir un contour en temps réel.

La conception et simulation du circuit numérique menée par l'appui d'outils informatiques spécialisé et l'utilisation du langage VHDL comme outil de description pour représenter le comportement et l'architecture du dispositif numérique nous a permis d'obtenir un certain niveau de réutilisabilité des différents blocs de l'architecture.

Conclusion générale

On a proposé une implémentation hardware pour la détection de contour par l'approche de snake en utilisant une architecture FPGA, cette solution est basée sur l'utilisation de plusieurs modules qui travaillent en parallèle afin de réduire le temps de calcul.

Les travaux menés au cours de ce mémoire constituent l'ensemble des étapes indispensables pour l'implémentation d'un circuit logique sur un FPGA de façon générale et l'implémentation d'un algorithme de traitement d'image en particulier.

Pour parvenir à réussir l'implémentation du circuit, on a suivi la méthode scientifique en commençant par la recherche bibliographique, l'étude des documents et l'assimilation des informations et par-dessus toute l'assimilation de la méthodologie propre à l'implémentation des circuits numériques sur un FPGA. Puis viens la partie expérimentale qui est la réalisation du circuit en langage VHDL, on teste le circuit en le simulant et on résout les erreurs obtenues après la simulation, tenant compte que le temps de la simulation été vraiment important.

En perspective, ce travail peut être poussé encore plus loin dans l'implémentation d'autres algorithmes de détection de contour par approche contour actif sur un traitement de vidéo afin d'aboutir à un suivi d'objet en temps réel.

Chapitre I

Circuits logiques programmables

Chapitre II

Détection de contours