N° Ordre…… /FSI/UMBB/2016

République Algérienne Démocratique et Populaire

Ministère De L'Enseignement Supérieur et De La Recherche Scientifique

Université Mhamed Bougara Boumerdes

Faculté des Sciences - Département de Physique

Structure Infotronique

Spécialité –Génie électrique

Mémoire de Fin d'Etudes pour l'Obtention du Diplôme

Master

en

Systèmes Electroniques Complexes

Thème

Implémentation d'un filtre orientable pour l'extraction des attributs des images cérébrales sur la carte ZedBoard

Par

Melle BENMAMMAR Lydia

Melle YACEF Soumeya

Promoteurs

M$^{r.}$ F. TALBI (CDTA)

M$^{r.}$ S.Seddiki (CDTA)

M^{me} H.BOUMERIDJA (UMBB)

Remerciements

Nous tenons à remercier tout d'abord « ALLAH » le tout puissant qui nous a donné le courage et la patience pour arriver à ce jour.

Nous tenons à remercier dans un premier temps, le Directeur du Centre de Développement des Technologies Avancées (CDTA) de nous avoir permis d'effectuer notre stage au sein de son établissement, ainsi que toute l'équipe de recherche, AC2 (Architectures pour la Classification et cryptographie), de la division ASM et à leur tête Mme F.ALIM, Chef d'équipe, pour nous avoir accueilli et de nous avoir permis de travailler dans de bonnes conditions et nous a conseillés tout le long de notre stage.

Nous adressons nos remerciements les plus sincères à notre encadreur Mr TALBI Farid, qui a accepté de nous encadrer et qui nous a fait profiter de ses larges connaissances ainsi que ses précieux conseils au cours de notre projet de fin d'études.

Nous exprimons nos profondes gratitudes et nos remerciements également à notre encadreur

Mr SEDDIKI Sofiane, pour son savoir faire professionnel et son aide, pour nous avoir dirigés au cours de ce travail ainsi son soutien tout au long de ce travail.

Nous exprimons nos remerciements à Mme H.BOUMERIDJA CHEBAANI, notre promotrice au Département d'Infotronique à la Faculté des Sciences de l'Université de Boumerdes pour ses conseils au cours de ce travail.

Sans oublier nos remerciements les plus sincères à nos parents.

Enfin nos remerciements vont aux membres du jury pour l'honneur qu'ils nous ont fait de participer à la soutenance de ce projet de fin d'études.

Dédicace

Je dédie ce modeste travail à :

A mes très chers parents qui n'ont pas cessé de prier pour moi et qui m'ont aidé durant toute la durée de mes études, que Dieu les garde pour nous .

A mon très cher frère Massinissa.

A ma chère sœur Chahrazed.

A toute la famille Benmammar et Abdelli.

A mon fiancé Kamel

A toutes mes amies : Houda, Imen, Amel, Sabrina, Nora, Chahrazed, Amina, Soumeya...

À toute la famille de mon binôme

A tous ceux qui j'aime et ceux qui m'aiment.

Lydia Ben

Dédicace

Je dédie ce modeste travail à mes très chers parents qui m'ont encouragé, guidé et conseillé durant les moments les plus difficiles de ce parcours.

À ma sœur Sarah et son mari.

À mes deux chers frères : AMINE et HAMZA.

À mon futur mari ANIS et toute sa famille.

À mes très chères cousines : FAIZA et FATIMA.

À tou(s)(tes) mes ami(e)s de l'UMBB.

À toute la famille de mon binôme

À tous ceux que j'aime et qui m'aiment.

Y.Soumeya

Résume

Résume

L’objectif principal de notre travail est l’implémentation sous l’environnement Xilinx ISE 14.7 sur un circuit FPGA (Field Programmable GateArray) d’un filtre gaussien orientable. Pour cela nous avons implémenté deux architectures : Gaussien 2D et Gaussien séparable, dont le but est de comparer entre ces deux architectures en termes de ressources occupées et de temps d’exécution. Une architecture du filtre orientable est appliquée au filtre séparable pour l'implémenter en utilisons un masque de 5x5.

Abstract

The main goal of our work is the implementation on the Xilinx ISE 14.7 environment with FPGA (Field Programmable Gate Array) circuit of a steerable Gaussian filter. For this we implemented two architectures: Gaussien 2D and Gaussien separable filters whose purpose is to compare between these two architectures in terms of the occupied resources and execution times. An architecture of the steerable filter is applied to the separable filter to implement it by using a mask 5x5.

الملخص

الهدف الرئيسي من هذا العمل هو زرع مرشحة موجهة في الدارة المنطقية المبرمجة (FPGA) من خلال استعمال برنامج Xilinx ISE 14.7 لهذا قمنا بزرع تصميمين الاول غوس ثنائي الابعاد و الثاني غوس قابل للانفصال و ذلك بهدف المقارنة بينهما من ناحية وقت التفعيل والموارد

اخترنا مرشحة غوس قابل الانفصال لزرع تصميم المرشح الموجه و ذلك باستعمال قناع 5*5.

Table des matières

Liste des Figures

Chapitre 1

Chapitre 2

Chapitre 3

Chapitre 4

Liste des tables

Liste des Abréviations

Abbreviation	***Sens***
ASIC	***A**pplication **S**pecific **I**ntegrated **C**ircuit*
CLB	***C**onfigurable **L**ogic **B**lock*
CPLD	***C**omplex **P**rogrammable **L**ogic**D**evice*
DSP	***D**igital **S**ignal **P**rocessor*
EPLD	***E**rasable **P**rogrammable **L**ogic**D**evice*
FPGA	***F**eild**P**rogrammable **G**ate **A**rrays*
HDL	***H**igh speed integrated circuits **H**ardware **D**escription **L**anguage*
IC	***I**ntegrated**C**ircuit*
IRM	***I**magerie par **R**ésonance **M**agnétique*
ISE	***I**ntegrated **S**ynthesis **E**nvironment*
LB	***L**ogic **B**locks*
LUT	***L**ook **U**p **T**able*
PS	***P**rocessing **S**ystem*
PL	***P**rogrammable **L**ogic*
PLD	***P**rogrammable **L**ogic **D**evice*
RAM	***R**andom **A**ccess **M**emory*
ROM	***R**ead **O**nly **M**emory*
SDK	***S**oftware **D**evelopment **K**it*
VHDL	***V**ery **H**igh speed integrated circuits hardware **D**escription **L**anguage*

Introduction générale

Introduction générale

Les années passées ont vu l'explosion du marché des systèmes embarqués dans de nombreux domaines industriels et grand public comme par exemple les télécommunications, les satellites, et l'imagerie médicale. Ces besoins de plus en plus importants génèrent une compétition industrielle où les facteurs comme le coût, les performances et surtout le «Time To Market» deviennent prépondérants pour le succès d'un produit.

Le traitement d'images médicales est d'une importance cruciale dans le diagnostic des tumeurs cérébrales. Ce traitement est un ensemble d'opérations qui permettent l'amélioration tel que le filtrage et le rehaussement de contraste, ainsi que l'extraction de l'information à partir des images médicales.

Ces transformations de l'image peuvent aussi être l'élément de base des systèmes d'aide au diagnostic qui se compose de blocs de segmentation et de classification. La fiabilité de ce dernier bloc dépend de la précision des paramètres extraits (texture, formes, etc.) de l'image. Plusieurs méthodes d'extraction des caractéristiques existent comme la méthode d'extraction des caractéristiques géométriques par le filtre orientable qui fera l'objet de ce mémoire.

Les filtres orientables sont utilisés dans des nombreuses tâches de vision et de traitement d'image, telles que l'analyse de la texture, la détection des contours, l'analyse du mouvement, et l'amélioration de l'image. Dans beaucoup de ces tâches, il est utile d'appliquer des filtres orientables, et d'examiner la sortie du filtre en fonction de l'orientation et la phase.

L'objectif du projet proposé par l'équipe de recherche AC2 (Architectures pour la classification et cryptographie) de la division ASM du Centre de Développement des Technologies Avancées (CDTA) est l'implémentation hardware du filtre orientable sur un circuit programmable de type **FPGA** dont le but est l'extraction des attributs des images cérébrales.

Afin d'atteindre notre objectif, nous avons organisé notre mémoire selon les étapes suivantes :

Le premier chapitre donne une description globale de l'outil FPGA et la nouvelle carte Zedboard.

Le deuxième chapitre présente les généralités sur les différentes méthodes d'extraction d'attributs.

Le troisième chapitre porte sur la théorie de fonctionnement des filtres étudiés dans ce mémoire (les filtres orientables), et leurs implémentations software sous l'outil MATLAB.

Le dernier chapitre est consacré à l'implémentation hardware sur un circuit programmable de type FPGA, les résultats de simulation ainsi que les performances temporelles et spatiales.

Nous terminons par une conclusion illustrant les perspectives de recherche et de développement du filtre orientable.

Chapitre 1

Généralités sur les FPGAs et la carte ZedBoard

1.1 Introduction

Les années passées ont vu l'explosion du marché des systèmes embarqués dans de nombreux domaines industriels et grand public comme par exemple les télécommunications, les satellites, et l'imagerie médicale. Ces besoins de plus en plus importants génèrent une compétition industrielle où les facteurs comme le coût, les performances et surtout le «Time To Market» deviennent prépondérants pour le succès d'un produit [13].

Depuis leur création, les technologies informatiques sont de plus en plus présentes et de plus en plus rapides, poussées par la création de circuits intégrés (IC). Ces circuits intégrés, sont beaucoup plus rapides que des lignes de codes s'exécutant sur un processeur car dédiés à une fonction définie. Cependant, le matériel de mise au point des IC est devenu de plus en plus performant et l'outil, le FPGA (Field Programmable Gate Array) dépasse désormais son simple rôle d'interface d'appoint. La technologie FPGA est devenue une cible viable pour la mise en œuvre des composants toujours plus rapides et à plus haute intégration, ce qui permet de programmer des algorithmes des applications importantes. Cette technologie permet d'implanter un grand nombre d'applications et offre une solution d'implantation matérielle à faible coût [13].

Dans ce chapitre, nous présentons quelques notions de base du domaine FPGA et une description générale sur la nouvelle carte ZedBoard.

1.2 La technologie FPGA

1.2.1Définition

Les FPGA (Field Programmable Gate Arrays ou "réseaux logiques programmables") ; littéralement traduit comme "Matrice de portes logiques programmable", sont des composants entièrement reconfigurables ce qui permet de les reprogrammer à volonté afin d'accélérer certaines phases de calculs. L'avantage de ce genre de circuit est sa grande souplesse qui permet de les réutiliser à volonté dans des algorithmes différents en un temps très court. Les FPGAs nous permettent de programmer du hardware à l'aide d'un langage de

description matérielle (VHDL). Ils permettent donc de concevoir et d'implémenter n'importe quelle fonction digitale [11].

1.2.2 Les composants d'un FPGA :

Tous les FPGA sont constitués de trois composants principaux, à savoir blocs logiques (LB), Les blocs d'entrée sortie I / O, et le routage programmable ou interconnexions comme le montre la figure 1.1

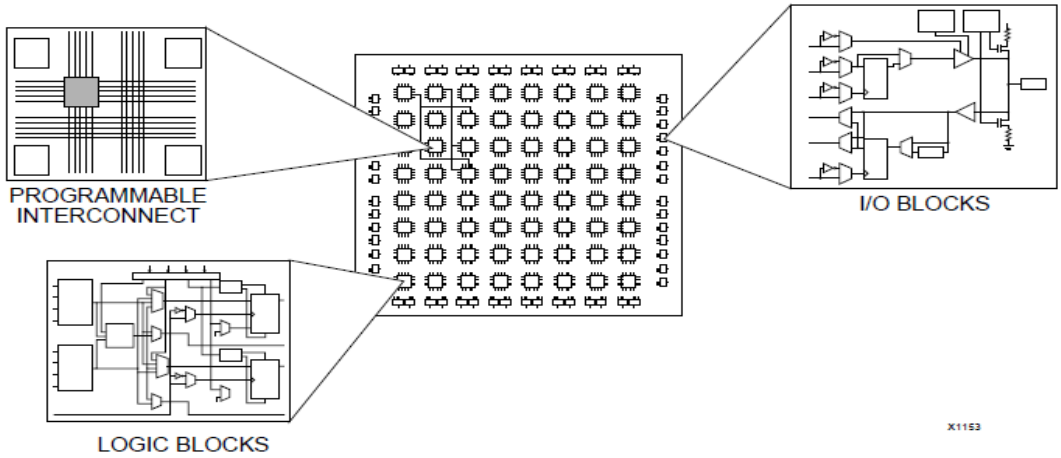

Figure1.1 : Architecture générale d'un FPGA.

Les FPGA utilisent des blocs logiques complexes les CLB et des commutateurs programmables d'interconnexion. Les CLB varient d'un circuit à un autre, cependant la majorité utilise des LUTs (Look Up Table) et des bascules (FlipFlops)[14].

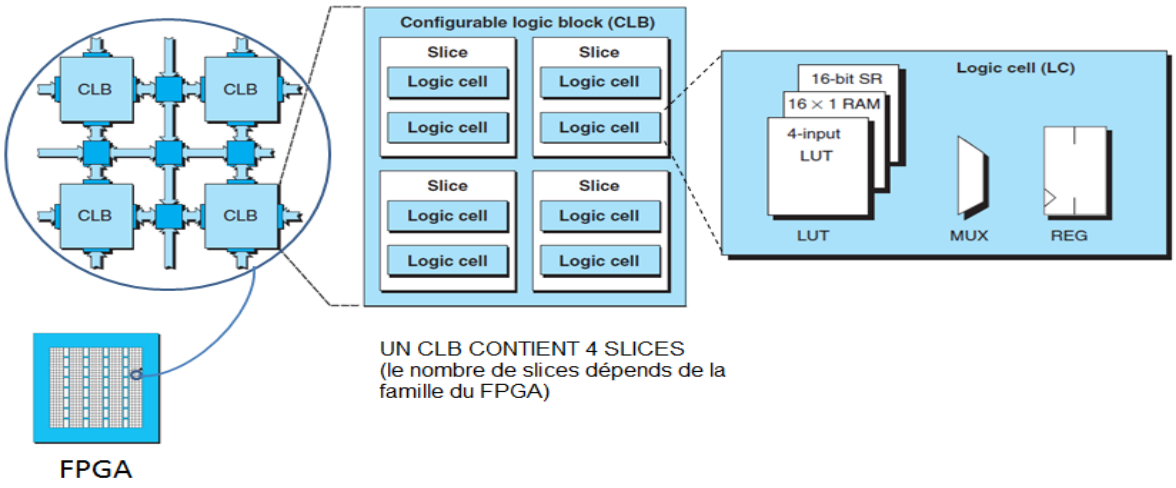

Figure 1.2: structure d'un CLB.

1.2.3 Les étapes du flot de conception

Afin de mettre en œuvre un circuit sur un FPGA, plusieurs étapes sont parcourues (figure1.3) qui consistent à concevoir et affiner la description du circuit du haut niveau (algorithme) jusqu'au bas niveau (dessin géométrique ou layout). Passer d'un niveau au suivant et vérifier le résultat obtenu se fait de façon semi-automatique en utilisant des outils de conception tels que ISE de Xilinx.

Ainsi, les étapes de flot de conception se résument comme suit (figure 1.3) :

- Entrée de conception (Design entry)
- Synthèse (Synthesis)
- Implémentation (Implementation)
- Vérification (Verification)
- La configuration du circuit (Device configuration)

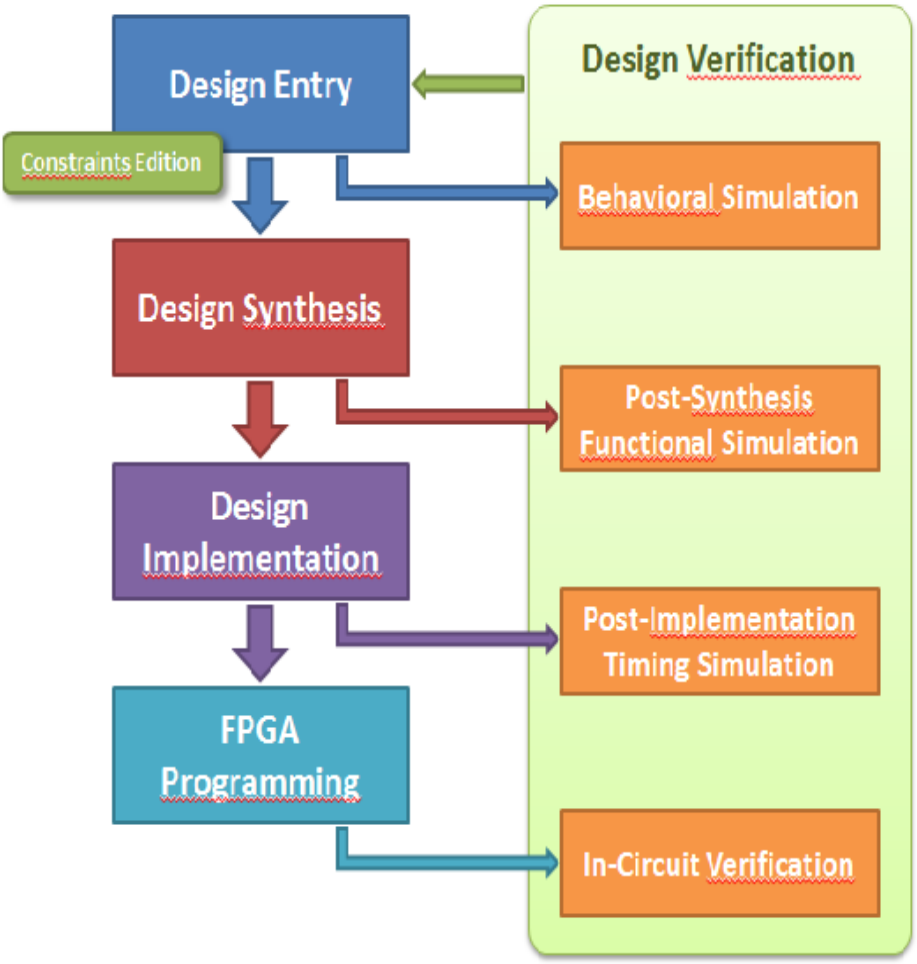

Figure 1.3: Les étapes du flot de la conception [15].

-Entrée de conception (Design entry) :

C'est la première étape de flot de conception d'ISE. Dans cette étape, le concepteur crée les fichiers sources en se basant sur l'architecture à concevoir. Ces fichiers sources vont être programmés en langage de description hardware de haut niveau tel que le VHDL, VERILOG, ABEL ou SCHEMATIC.

Après l'étape d'entrée, une simulation optionnelle peut être effectuée.

-Synthèse (Synthesis) :

Durant l'étape de synthèse le code VHDL (ou VERILOG, etc.) devient un fichier ***NETLIST***. Cette dernière est la donnée d'entrée pour la prochaine étape.

-Implémentation (Implementation)

Le fichier NETLIST (résultat après synthèse) (fichier logique) va être convertit en fichier physique durant l'étape d'implémentation. Ce qui nous permet de faire le téléchargement de la description dans le circuit cible. Les étapes d'implémentation dépendent du circuit choisi (FPGA ou CPLD).

-Vérification (Verification)

Nous pouvons vérifier la fonctionnalité de notre travail à plusieurs points de flot de conception. Nous utilisons le VHDL Test Bench ou Modelsim**.** La vérification peut se faire pour une portion de la conception ou pour toute la conception. La vérification englobe la fonctionnalité et le temps.

Le simulateur interprète le code VHDL ou Verilog en circuit fonctionnel et affiche les résultats de la description logique HDL pour déterminer le fonctionnement correct du circuit. La simulation nous permet de créer et vérifier des fonctions complexes dans un temps. Nous pouvons aussi exécuter la vérification après la programmation du circuit.

-La configuration du circuit (Device configuration)

Durant cette étape, la génération du fichier de configuration ainsi que le téléchargement du fichier de programmation du PC vers le circuit vont être exécutés.

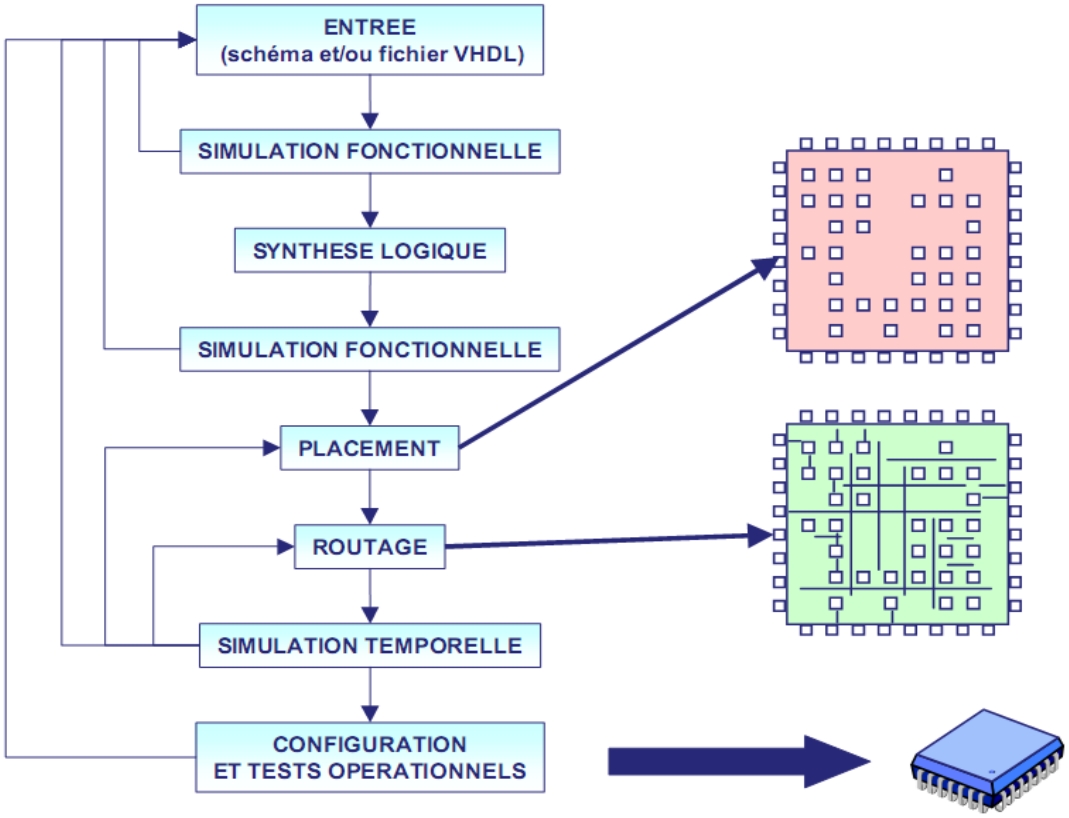

Figure 1.4: Flot de la conception d'un circuit FPGA.

1.2.4 Les avantages du FPGA

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

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

-Un investissement rentable dans la durée : Cela est dû à sa reprogrammation, ce qui implique une réutilisation à destination d'autres projets, malgré un prix à l'achat supérieur à un circuit ASIC.

-Une Reprogrammation quasi-instantanée du circuit : Une fois le programme validé cela ne prend que quelques minutes à l'implémenter. A titre de comparaison, la fabrication d'un circuit ASIC peut prendre plusieurs semaines.

1.2.5 Le langage VHDL

C'est un langage de description de matériel qui est utilisé pour la spécification, la simulation et la preuve formelle d'équivalence de circuits. Ensuite il a été utilisé pour la synthèse automatique. C'est un langage de description de matériel pour circuit à très haute vitesse d'intégration.

Aussi le langage VHDL permet la description des aspects les plus importants d'un système matériel (*hardware system*), à savoir son comportement, sa structure et ses caractéristiques temporelles. Par système matériel, on entend un système électronique arbitrairement complexe réalisé sous la forme d'un circuit intégré ou d'un ensemble de cartes.

- **Quelque caractéristique du langage VHDL**
  - **La portabilité**

La portabilité constituait un des objectifs principaux pendant la phase de définition du langage VHDL. L'utilisation largement répandue de ce langage est essentiellement due au fait qu'il soit un standard.

  - **La lisibilité**

La représentation schématique n'est pas toujours d'une grande lisibilité. Il est difficile de saisir rapidement le fonctionnement d'une machine d'état parmi un enchevêtrement de registres et de portes logiques.

  - **La modularité**

La modularité est une des caractéristiques essentielles de VHDL. Une description partitionnée en plusieurs sous-ensembles est dite modulaire.

  - **Le couple entité architecture**

L'élément essentiel de toute description en VHDL est formé par le couple entité/architecture qui décrit l'apparence externe d'un module et son fonctionnement interne.

- L'entité décrit la vue externe du modèle : elle permet de définir les ports par où sont véhiculés les informations (signaux) et les paramètres génériques.
- L'architecture définit la vue interne du modèle. Cette description peut être de type structurel, flot de données, comportemental ou une combinaison des trois.

On distingue 3 grandes classes de descriptions dans ce langage :

- **La description comportementale**: Elles spécifient le comportement au moyen de formules ou de programmes d'actions. Cela signifie que la sortie est en permanence donnée en fonction des entrées. Une telle description n'utilise aucun composant

interne : on considère que le composant est atomique, entièrement défini par sa spécification logique.

- **La description structurelle**: Elles énoncent une interconnexion de composants. Une description structurelle est la transcription directe d'un schéma. Dans cette description le comportement résulte du comportement des composants internes et du câblage réalisé, selon les règles usuelles de causalité physique.
- **La description flot de données:** Une description flot de données spécifie comment la donnée est transférée de signal à signal sans utiliser d'affectations séquentielles (comme dans la description comportementale). Cette description ne nécessite pas de déclaration de processus, et toutes les déclarations s'exécutent en même temps [11].

1.2.6 Performance de la technologie

Les systèmes électroniques modernes sont de plus en plus complexes, les contraintes de taille, de puissance dissipée et de performances sont de plus en plus sévères (téléphonie mobile, ordinateurs, traitement du signal, de l'image, etc.) d'où l'accroissement spectaculaire des densités.

Les progrès dans la capacité d'intégration des circuits électroniques ont ouvert de nouvelles perspectives pour le traitement d'images en temps réel sur des systèmes embarqués. D'un côté, des processeurs spécifiques peuvent couramment effectuer des milliards d'opérations par seconde. Ces circuits permettent de réaliser des applications avec des performances en termes de vitesse de traitement sans cesse croissantes.

la ZYNQ, une des dernières familles de Xilinx, un processeur ARM dual-core a été placé sur le circuit ZYNQ à côté de la logique reconfigurable [9].

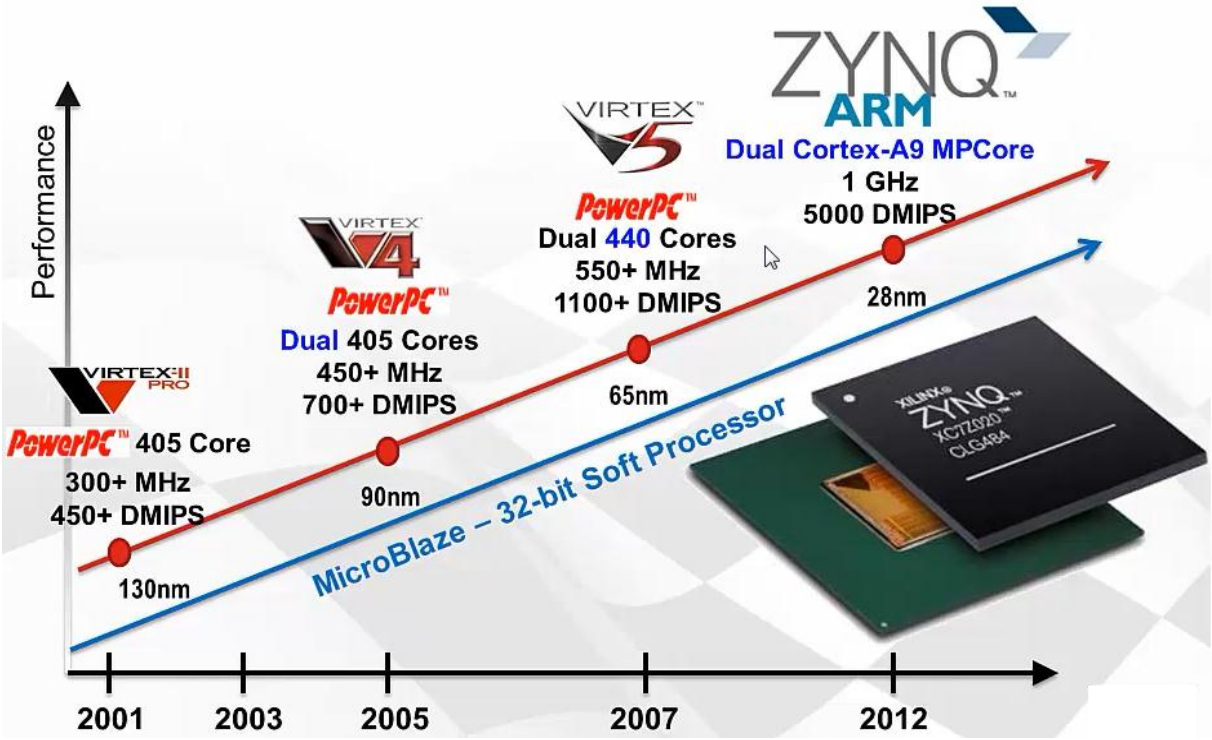

Figure 1.5 : Héritage de Xilinx.

1.3 la carte ZedBoard

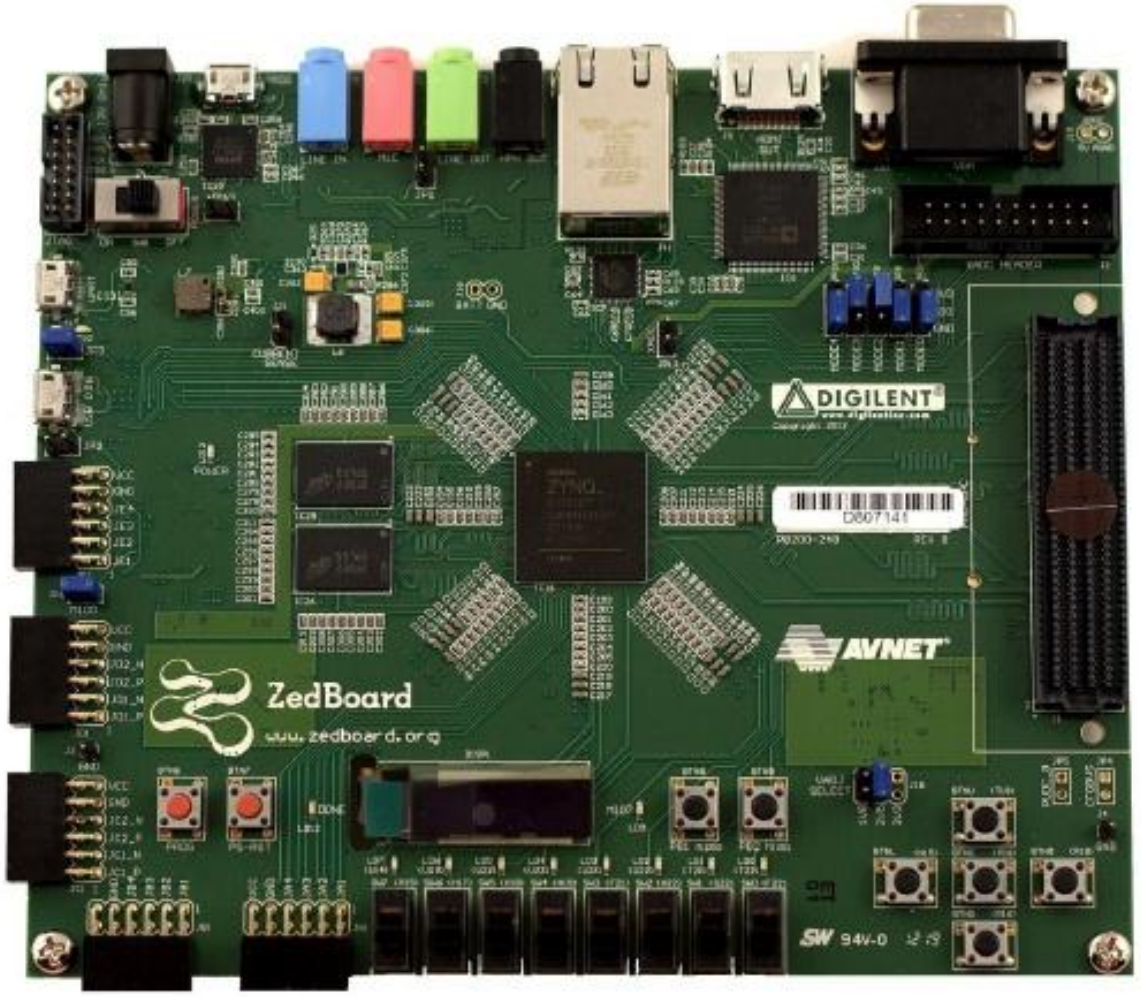

Figure 1.5 : La carte ZedBoard.

La carte ZedBoard est une carte de développement conçue par Digilent. Les systèmes on chip (SOC) ZYNQ de la société xilinx intègre :

-Un système de traitement PS (processing system) basé sur un processeur double cœur ARM cortex A9 capable d'accueillir un système d'exploitation comme LINUX .
On appelle PS la partie processeur et les périphériques associés cela inclus : les deux cœurs ARM cortex A9, les bus AMBA et AXI, le DMA, les GPIOs, I2C, UART, CAN, SPI, le contrôleur des mémoires QuadSPI, NAND et NOR et le contrôleur de mémoire vive [9].
-Un système de programmation logique PL (programmable logic) avec un FPGA de la série XILINX-7. La PL, contient les éléments logiques de base, la RAM, des DSP et les entrées sorties standard [9].

1.3.1 Flot de conception entre la partie PS et PL

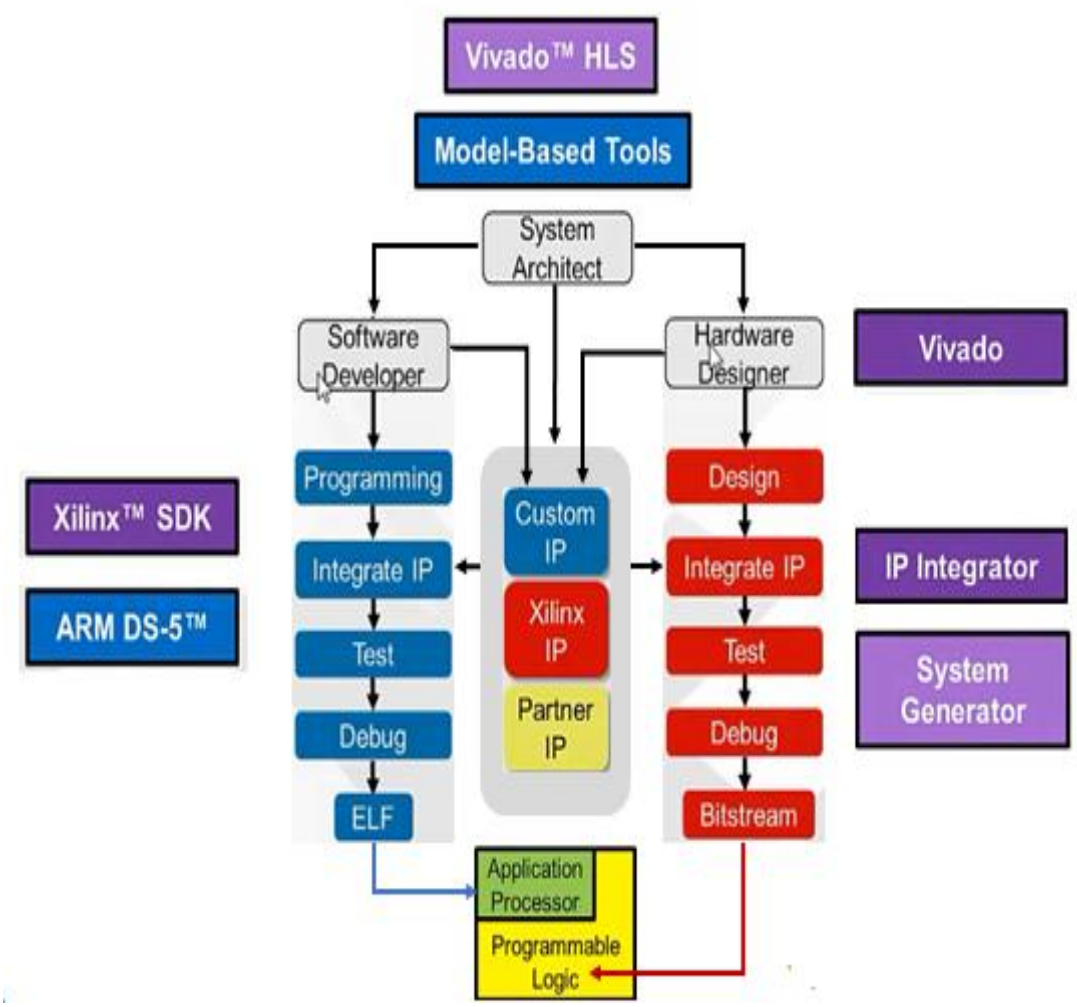

Figure 1.6: Flot de conception entre la partie PS et PL [15].

Nous pouvons voir sur la figure1.8 les différents périphériques présents sur la ZedBoard.

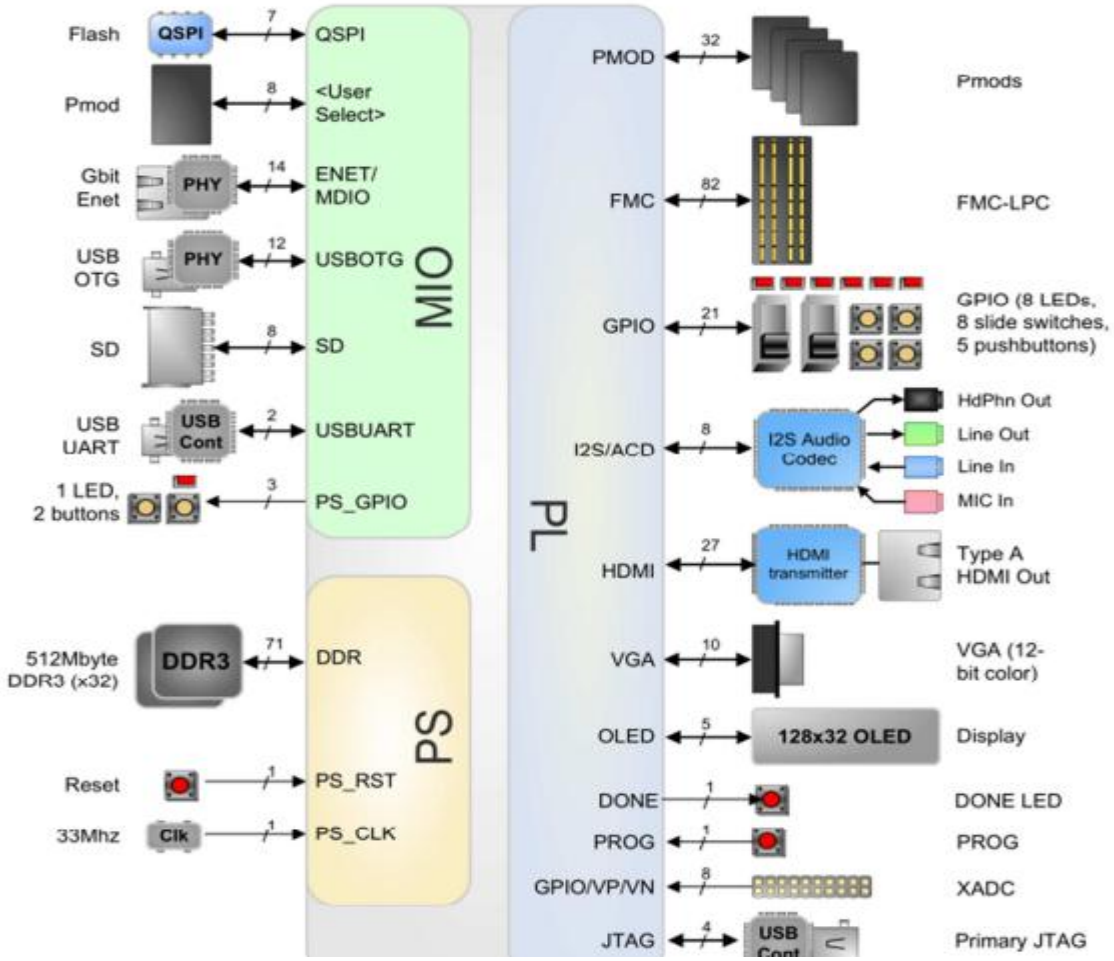

Figure 1.7 : Les Périphériques de la carte [9].

Au niveau de la connexion, la ZedBoard est assez complète et permet l'interfaçage de nombreux éléments:

- HDMI (Audio et vidéo)
- VGA (Vidéo)
- Ethernet (10/100/1000 Mbps)
- Port de débogage ARM Debug Access Port (DAP)
- Port USB 2.0 OTG (Device, Host et OTG)
- Port de programmation USB-JTAG
- Port USB-UART
- Connecteur FMC.

Les composants IHM sont également assez nombreux :

- 9 LEDs
- 8 interruptrices glissières

- 7 boutons poussoirs + 2 boutons Reset

Les mémoires intégrées, les entrées et sorties vidéo et audio, l'interface USB double rôle et Ethernet, ainsi que le slot SD simplifient au maximum la phase de conception sans matériel supplémentaire. En outre, six connecteurs Pmod facilitent toutes les conceptions [9].

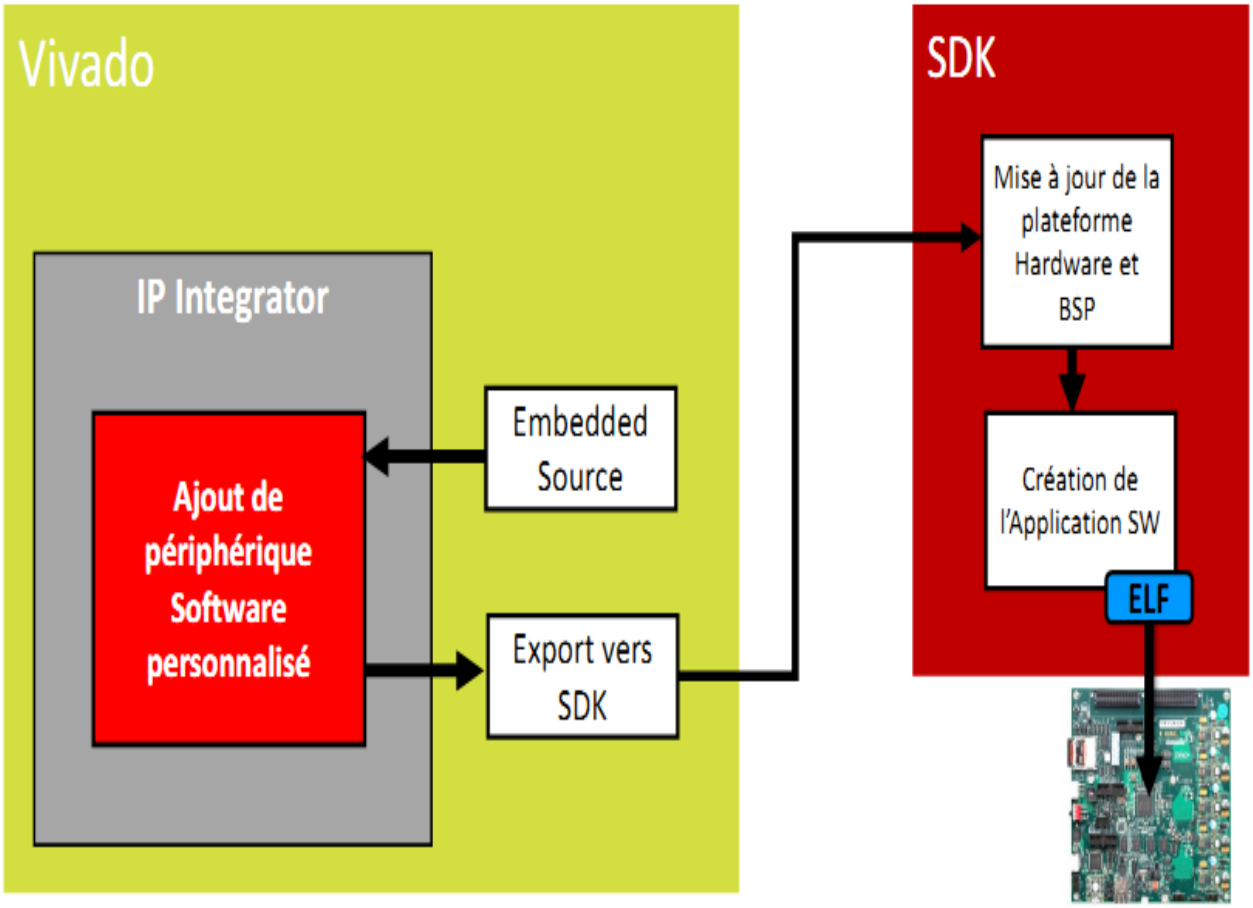

Figure 1.8 : Exemple d'ajout d'un périphérique PL custom [15].

Ils existent différents modèles de ZYNQ dont les caractéristiques sont résumées ci-dessous:

	Zynq-7000 AP SoC Devices	Z-7010	Z-7015	Z-7020	Z-7030	Z-7045	Z-7100
Processing System	Processor Core	Dual ARM® Cortex™-A9 MPCore™					
	Processor Extensions	NEON™ & Single / Double Precision Floating Point					
	Max Frequency	866 MHz			Up to 1 GHz		
	Memory	L1 Cache 32KB I / D, L2 Cache 512KB, on-chip Memory 256KB					
	External Memory Support	DDR3, DDR3L, DDR2, LPDDR2,2x QSPI, NAND, NOR					
	Peripherals	2x USB 2.0 (OTG), 2x Tri-mode Gigabit Ethernet, 2x SD/SDIO, 2x UART, 2x CAN 2.0B, 2x I2C, 2x SPI, 4x 32b GPIO					
Programmable Logic	Approximate ASIC Gates	~430K (28k LC)	~1.1M (74k LC)	~1.3M (85k LC)	~1.9M (125k LC)	~5.2M (350k LC)	~6.6M (444kLC)
	Block RAM	240KB	380KB	560KB	1,060KB	2,180KB	3,020KB
	Peak DSP Performance (Symmetric FIR)	100 GMACS	200GMACS	276 GMACS	593 GMACS	1334 GMACS	2662 GMACS
	PCI Express® (Root Complex or Endpoint)	-	Gen2 x4	-	Gen2 x4	Gen2 x8	
	Agile Mixed Signal (XADC)	2x 12bit 1Msps A/D Converter					
I/O	Processor System IO	130					
	Multi Standards 3.3V IO	100	150	200	100	212	250
	Multi Standards High Performance 1.8V IO	-	-	-	150	150	150
	Multi Gigabit Transceivers	-	4	-	4	16	16

Figure 1.9: Caractéristiques des différents modèles de la ZYNQ.

La carte est compatible avec la nouvelle solution haute performance, l'outil de programmation Vivado Suite de Xilinx, ainsi que le jeu d'outils ISE de Xilinx.

1.4 Conclusion

Nous avons revu dans ce chapitre une cible technologique pour une conception hardware. Les circuits FPGAs de Xilinx et la carte Zedboard ont été présentés vu que nous allons implémenter notre application sur ce matériel à l'aide des outils présentés à savoir Vivado et ISE de Xilinx.

Cependant, réussir une application à base de FPGA, c'est en premier lieu respecter un certain nombre de règles comme:

Bien connaître les caractéristiques du FPGA ciblé pour assurer son adéquation avec les besoins du projet.

Respecter une méthodologie de conception et un style d'écriture.

Opter pour des outils de synthèse de qualité.

Maîtriser les outils d'implémentation.

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

Chapitre 2

Généralités sur les différentes méthodes d'extraction des attributs

2.1 Introduction

Le traitement d'images est une discipline de l'informatique et des mathématiques appliquées qui étudie les images numériques et leurs transformations, dans le but d'améliorer leur qualité ou d'extraire de l'information.

L'attribut joue un rôle très important dans le domaine du traitement d'image. Avant d'obtenir les caractéristiques, différentes techniques de prétraitement des images comme binarisation, seuillage, redimensionnement, normalisation, etc. ces derniers sont appliqués sur l'image échantillonnée. Après cela, les techniques d'extraction des caractéristiques sont appliquées pour obtenir des caractéristiques qui seront utiles pour la classification et la reconnaissance des images.

Dans ce chapitre, nous allons expliquer les systèmes de traitement d'images, les principaux attributs et leurs différentes méthodes d'extraction des attributs.

2.2 L'image

Généralement, l'image est définie par le nombre de points qui la composent. En imagerie numérique, cela correspond au nombre de pixels qui compose l'image en hauteur (axe vertical) et en largeur (axe horizontal) par exemple 200 pixels par 450 pixels, par conséquent elle est considérée comme une image de taille « 200×450 ».

Une image est simplement la représentation graphique d'une scène ou d'un objet situé en général dans un espace tridimensionnel.

En imagerie médicale, une image est obtenue par la transformation d'une scène à l'aide d'un capteur [11].

2.3 Les systèmes de traitement d'images

En général, le système de traitement d'image contient quatre étapes qui sont illustrées par la figure suivante:

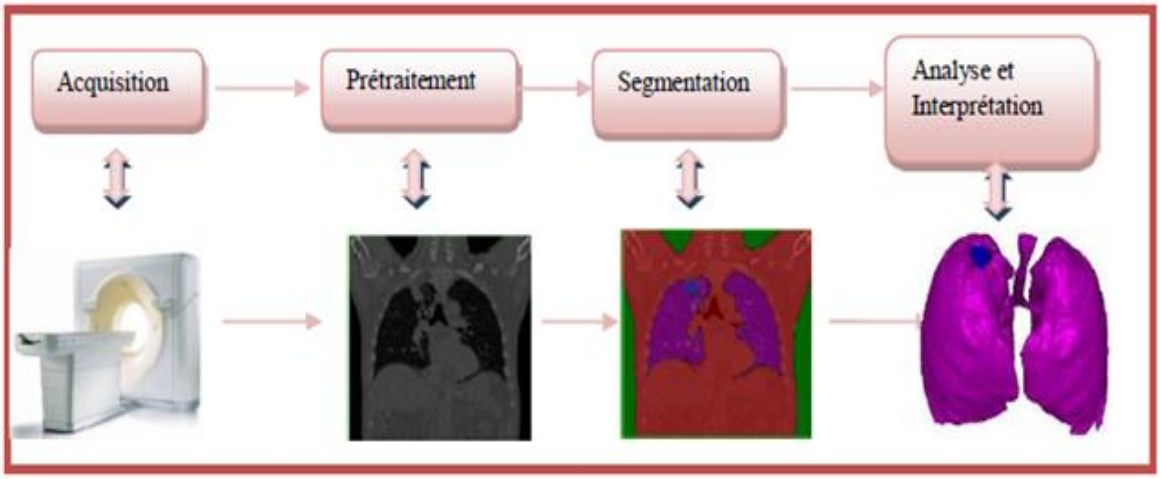

Figure 2.10 : Schéma d'un système de traitement d'image [11].

2.3.1 Acquisition d'une image

L'acquisition d'image est la première étape dans le système de traitement d'image et est considérée comme une partie essentielle de toute chaîne de conception et de production d'image. Pour pouvoir manipuler une image sur un système informatique, il est nécessaire de lui faire subir une transformation qui la rend lisible et manipulable par ce dernier.

2.3.2 Prétraitement des images

L'étape de prétraitement a pour but de faciliter la segmentation en renforçant la ressemblance entre pixels appartenant à une même région, ou en accentuant la dissemblance entre pixels appartenant à des régions différentes.

Il est souvent nécessaire d'améliorer la qualité de l'image ; et cela dans le but de faciliter l'extraction des différents objets de la scène.

2.3.3 Segmentation

La segmentation d'images est une opération de traitement d'images qui a pour but de rassembler les pixels qui possèdent une même propriété donnée afin d'obtenir des régions uniformes. Elle consistera à partitionner l'image en un certain nombre de régions disjointes et de séparer les différentes zones homogènes.

La segmentation a pour objectif l'extraction d'indices visuels dans une image. Elle permet de partitionner une image en ensembles de points appelés régions, homogènes pour une ou

plusieurs caractéristiques (intensité, couleur, texture, etc.) et sont différentes pour au moins une de ses caractéristiques des régions voisines [5].

Lorsque le prétraitement et le niveau souhaité de la segmentation (ligne, mot, caractère ou symbole) ont été atteints, une technique d'extraction de caractéristique est appliquée aux segments afin d'obtenir les attributs. C'est le but de ce chapitre puisque le travail demandé est basé sur les attributs géométriques et de texture.

2.3.4 Analyse et interprétation

A partir des données provenant de la scène analysée, l'interprétation de l'image nous a permet de reconnaitre les différentes entités qui composent l'image, et à comprendre leur organisation spatiale et construire une description de la scène [2].

2.4 Extractions des attributs

L'extraction de caractéristiques est effectuée après la phase de prétraitement et la segmentation et qui est suivi par la classification, Ainsi, il y aura une extraction des attributs de l'image à l'aide des fonctions mathématiques puis un regroupement sous la forme d'un représentant de l'image : le vecteur descripteur de l'image.

Il existe deux familles d'attributs :

1) **Les attributs locaux,** qui sont calculés sur une région de l'image considérée. Ils sont généralement géométrique (par exemple des parties concaves / convexes, le nombre de points d'extrémité, des branches, des articulations, etc.).
2) **Les attributs globaux,** qui sont calculés à partir de l'image entière. Ils sont généralement topologique (connectivité, profils de projection, le nombre de trous, etc.) ou statistiques (moments invariants, etc.) [3].

2.4.1 Les principaux attributs

Des attributs de différents types sont utilisés pour représenter le contenu de l'image. Ils sont classés en trois familles principales : la forme, la texture et la couleur.

2.4.1.1 La forme

La forme est un descripteur très important dans les bases d'images. La forme désigne l'aspect général d'un objet, son contour.

Les attributs de type forme sont capable de caractériser les différents objets contenus dans l'image. Généralement ce type d'attribut indique l'aspect général d'un objet, comme son contour donc une segmentation sous forme de traitement préliminaire de l'image est souvent nécessaire. Deux catégories d'attributs de formes peuvent être extraites : la première catégorie est basée sur la géométrie des régions de l'image. La deuxième est basée sur les statistiques des intensités de pixels de différentes régions dans l'image.

Une image peut subir des transformations géométriques comme la rotation, la translation et le changement d'échelle. Pour assurer une description assez robuste et efficace, les attributs de la forme couvrent, généralement, toutes les échelles de représentation (allant des détails jusqu'à la forme grossière) que comporte un objet. De plus, ils sont souvent insensibles aux différentes variations causées par des transformations géométriques.

La figure (2.2) présente un exemple des transformations géométriques que peut subir une image.

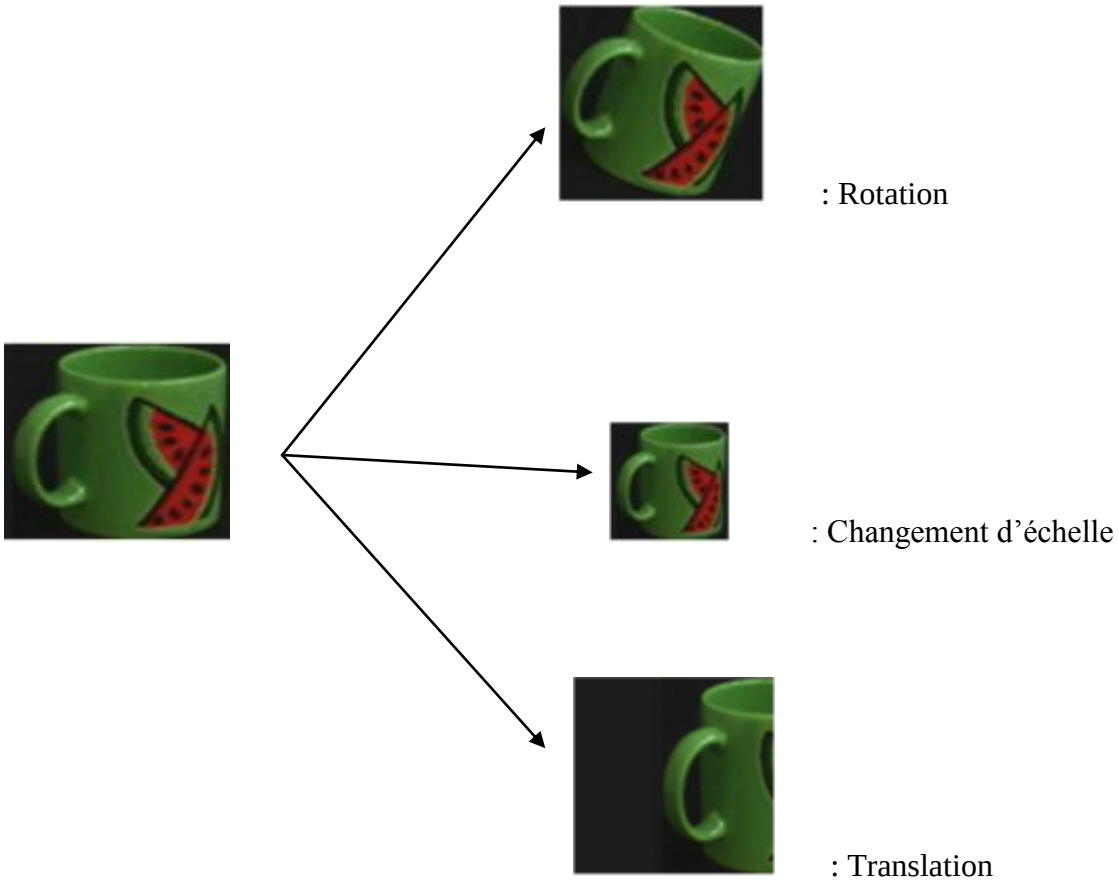

Figure 2.11 : Différentes transformations géométriques que peut subir une image.

Les principaux attributs de type forme sont les suivants :

A) Les attributs géométriques de région

Les attributs géométriques de forme permettent de distinguer les différents types de forme que peuvent prendre les objets d'une scène. Ils nécessitent une segmentation en région préalable de l'image. Ils sont ensuite calculés sur les différentes régions de l'image.

- La **surface,** relative (ou normalisée) d'une région S_K de l'image I est le nombre de pixels contenus dans cette région par rapport au nombre total de pixels de l'image :

$$\boldsymbol{S_K = \frac{\mathbf{card(R_K)}}{\mathbf{hauteur(I) * largeur(I)}}} \quad \textbf{(2.1)}$$

- Le **centre de masse,** des pixels de la région est défini par :

$$\mathbf{P = (Pi, Pj) = \left(\frac{\sum_{i \in R_K} i/card(Rk)}{largeur(I)}, \frac{\sum_{j \in R_K} j/card(Rk)}{largeur(I)}\right)} \quad \textbf{(2.2)}$$

- La **longueur du cône,** de la région est le nombre de pixels en bordure de la région :

$$\boldsymbol{I_K = card\ (contour(R_K))} \quad \textbf{(2.3)}$$

- La **compacité,** traduit le regroupement des pixels de la région en zones homogènes et non trouées :

$$\mathbf{C_K = \frac{l_K^2}{S_K}} \quad \textbf{(2.4)}$$

Ces attributs très simples permettent d'obtenir des informations sur la géométrie des régions de l'image. Il existe d'autres attributs de forme, basés sur des statistiques sur les pixels des régions de l'image [3].

B) Les moments géométriques

Les moments géométriques permettent de décrire une forme à l'aide de propriétés statistiques. Ils sont simples à manipuler mais leur temps de calcul est très long. La Formule générale des moments :

$$\boldsymbol{m_{p,q} = \sum_{p=0}^{m} \sum_{q=0}^{n} x^p y^q f(x,y)} \quad \textbf{(2.5)}$$

L'ordre du moment est p + q. Le moment d'ordre 0 m(0,0) représente l'aire de la forme de l'objet.

Les deux moments d'ordre 1 m(0,1) et m(1,0), associés au moment d'ordre 0 m(0,0) permettent de calculer le centre de gravité de l'objet [3].

2.4.1.2 La texture

Il existe deux types de définition de la texture:

– La première est **déterministe** et fait référence à une répétition spatiale d'un motif de base dans différentes directions. Cette approche structurelle correspond à une vision **macroscopique** des textures. La peau des reptiles est un exemple de texture macroscopique dans lequel la primitive est l'écaille.

– La seconde est **probabiliste** et cherche à caractériser l'aspect chaotique qui ne comprend ni motif localisable, ni fréquence de répétition principale. Cette approche est **microscopique** et l'herbe en est un excellent exemple.

Les méthodes d'extraction des attributs dépendent du modèle de texture recherché. Pour les textures macroscopiques, on va chercher des répétitions d'un motif dans une certaine direction. Pour les textures microscopiques qui sont probabilistes, les techniques de recherche sont basées sur des calculs statistiques sur les niveaux de gris des images.

2.4.1.2.1 Types de textures

A) les textures régulières :

Dans lesquelles la répétition spatiale d'un motif de base appelé « texton » est évidente, dans différents directions (Figure 2.3). La répétition spatiale de ces motifs obéit à des règles de direction et de placement. Ainsi, une région texturée constituée par un réseau bidimensionnel répétant le motif original selon une direction et une période particulière. La description du motif élémentaire, les dimensions du réseau et l'orientation du motif suffisent alors pour décrire complètement la texture, cette famille sera bien décrite par des approches fréquentielles ou structurelles [8].

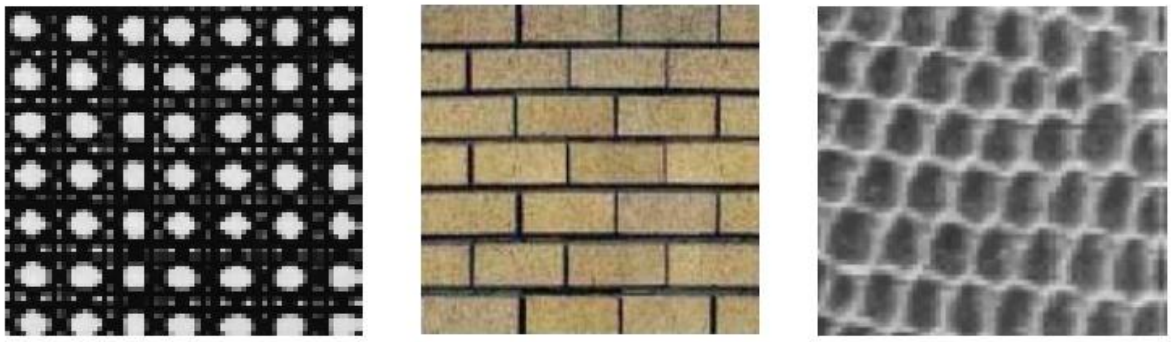

Figure 2.12 : Exemples de textures régulières.

B) Les textures aléatoires :

Dans ces textures, aucun motif particulier n'est localisable. Elles ont un aspect anarchique et désorganisé, tout en restant homogène, et ne répondent à aucune règle d'agencement particulière. La primitive est ramenée au niveau du pixel, ce qui vaut à ces textures le nom de textures « microscopiques » ou « micro-textures » (voir la Figure 2.4) [4].

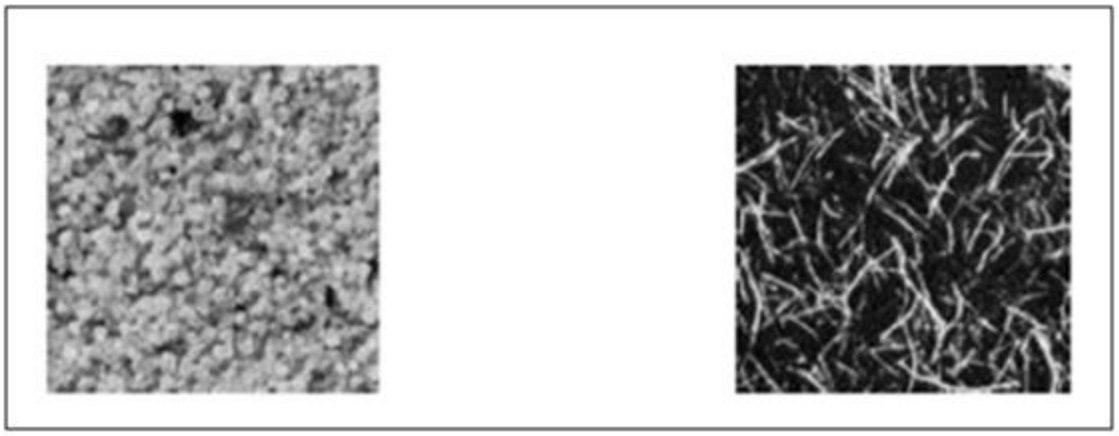

Figure 2.13 : Exemple de textures aléatoires.

2.4.1.2.2 Etat de l'art sur les différentes méthodes d'extraction d'attributs de texture

- **Statistiques :**

Étudient les relations entre un pixel et ses voisins et définissent des paramètres discriminants de la texture en se basant sur des outils statistiques. Généralement, elles sont utilisées pour caractériser des structures fines, sans régularité apparente.

Nous présentons quelques méthodes statistiques :

A) La matrice de cooccurrence

La texture d'une image peut être interprétée comme la régularité d'apparition de couples de niveaux de gris selon une distance donnée dans l'image. La matrice de cooccurrence est une matrice carrée n * n, où n est le nombre de niveaux de gris de l'image [3].

À partir de la matrice de cooccurrence, on extrait différents attributs de texture :

L'homogénéité h : $\mathbf{h= \sum_{i=1}^{n} \sum_{j=1}^{n} f(i,j|d,\theta)^2}$ **(2.6)**

L'entropie e : $\mathbf{e= \sum_{i=1}^{n} \sum_{j=1}^{n} f(i,j|d,\theta) \log_2 f(i,j|d,\theta)}$ **(2.7)**

Le contraste c : $\mathbf{c= \sum_{k=0}^{n-1} n^2 \left(\sum_{i=1}^{n} \sum_{j=1}^{n} f(i,j|d,\theta)\right) , |i-j| = n}$ **(2.8)**

B) Méthode de longueur de plage (Grey LevelRunLength Matrix-GLRLM)

Les longueurs de plage (ou iso-segments) est une méthode statistique d'ordre supérieur. On appelle une plage un ensemble de pixels consécutifs et dans une direction donnée ayant le même niveau de gris. La longueur d'une plage est alors le nombre de pixels pour une plage donnée.

A partir de la méthode de longueur de plage, diverses caractéristiques de texture peuvent être déduites, elles sont décrites comme suit [12]:

Nombre de longueurs de plage : $\boldsymbol{n_\theta = \sum_{n=1}^{N} \sum_{j=1}^{M_\theta} p_\theta(i,j)}$ **(2.9)**

Proportion de petites plages (Short Run Emphasis (SRE)) :

$$\mathbf{SRE= \frac{1}{n_\theta} \sum_{i=1}^{N} \sum_{j=1}^{M} \frac{p_\theta(i,j)}{j^2}.} \quad \textbf{(2.10)}$$

- ## Structurelles :

Les méthodes structurelles considèrent la texture comme un ensemble d'éléments de base qui permettent de décrire la texture en définissant les primitives et les "règles" d'arrangement qui les relient. Elles sont donc applicables sur les textures aléatoires.

- **Méthodes basées sur les modèles** :

Ces méthodes consistent à choisir un modèle pour représenter le signal. Ce modèle est défini par un certain nombre de paramètres qui, entre autre, peuvent caractériser la texture.

A) Méthodes markovienne

Les modèles de Markov s'intéressent à l'aspect aléatoire des textures en considérant que même structurées, les textures ne sont "jamais" parfaitement régulières. Dans ces modèles, l'analyse des textures se fait dans un cadre stochastique lié à la notion de voisinage, où le niveau de gris d'un pixel dépend uniquement des niveaux de gris des pixels voisins [10].

B) Méthodes fractales

La théorie fractale a été introduite par Mandelbort pour décrire la géométrie des objets naturels qui présentent une similarité statistique entre les parties constituant ces objets, c'est-à-dire que leurs formes géométriques restent statistiquement invariables par changement d'échelle [10].

- **Méthodes basées sur le filtrage** :

A) Théorie de la morphologie mathématique :

La morphologie mathématique s'intéresse à la forme des objets. Le principe de base de cette méthode consiste à comparer les objets que l'on veut analyser à un autre objet de forme connue appelé élément structurant .On peut le représenter comme une matrice binaire affichant un objet de forme généralement très simple, auquel les objets de l'image seront comparés.

B) Ondelettes :

Cette technique a été développée dans les années 90, testée pour l'analyse et la classification des textures des images satellitaires. L'intérêt pour les ondelettes s'est largement développé ces dernières années. Cette méthode se rapproche des filtres de Gabor [10].

Le filtre orientable :

Les filtres orientables sont utiles dans de nombreuses tâches de vision et de traitement d'image, tels que l'analyse de mouvement, et l'analyse de la texture, etc. La direction ou l'orientation implique que la sortie Oθ (x, y) d'une opération de filtrage utilisant un filtre orienté à un angle θ [7].

Le but de notre travail est d'appliquer le filtre orientable à des images IRM pour extraire les différents attributs géométriques et de textures.

2.4.1.3 La couleur

Le physicien James Clerk Maxwell (Pascale, 2003) a prouvé que toute couleur, étant initialement une sensation provoqué par la lumière avec l'œil, le cerveau, est une synthèse de trois couleurs seulement : le rouge, le vert, et le bleu. A partir de cette découverte en 1865, la colorimétrie (la science qui étudie la couleur) a vu le jour.

Les attributs couleurs sont toujours les plus utilisés grâce à leur simplicité d'extraction. Les auteurs dans (Bimbo, 2001) ont fourni une étude étendue de différentes méthodes employées pour l'extraction d'attributs couleurs. Un ensemble d'attributs couleurs a été testé pour être MPEG-7 (Manjunath et al., 2001). Ces attributs couleurs dépendent directement de l'espace couleur utilisé pour la représentation couleur de l'image. Dans la littérature plusieurs espaces couleurs sont étudiés (RVB,HSI ,HSV,HSB,HLS et XYZ) [6].

2.4.2 Le vecteur descripteur

À partir des attributs extraits de l'image, on construit un vecteur caractéristique de l'image, c'est le **vecteur descripteur** de l'image. Ce vecteur descripteur contient les attributs intéressants extraits de l'image. Il se présente en général sous la forme d'un vecteur à n composantes réelles.

Les attributs extraits des images sont de différents types et sont exprimés dans des unités différentes selon qu'ils appartiennent à la couleur, la texture, la forme. Une étape de **normalisation** est indispensable, elle va permettre de réajuster les valeurs des attributs pour les rendre commensurables [1].

2.5 Conclusion

Les attributs vont représenter l'image. Leurs choix est fortement dépendant des images de la base car il n'y a pas d'attributs universels donnant des bons résultats sur n'importe quelle base d'image.

Différentes approches ont été proposées dans la littérature pour extraire les caractéristiques d'une image, donc le chapitre suivant est consacré à l'étude des filtres destinés à l'extraction des attributs pour les images IRM cérébrales, Pour cela nous avons choisi les filtres orientables.

Chapitre 3

Implémentation software d'un filtre Gaussien orientable

3.1 Introduction

Les filtres orientables sont utilisés dans des nombreuses taches de vision et de traitement d'image, telles que l'analyse de la texture, la détection des contours, l'analyse du mouvement, et l'amélioration de l'image [1]

Dans ce chapitre nous présentons un aperçu théorique du filtre Gaussien orientable ainsi que son implémentation software sous MATLAB.

3.2 La convolution

La convolution est une opération de traitement d'image commune qui filtre une image en calculant la somme des produits entre l'image d'entrée et une image plus petite (le masque) appelé " noyau de convolution ou filtre de convolution " (figure 3.1). Une opération de convolution peut atteindre le flou, la netteté, le bruit, la réduction, la détection de bord et d'autres opérations d'imagerie utiles en fonction de la sélection des valeurs dans le noyau de convolution.

Mathématiquement, une convolution bidimensionnelle appliquée sur une image peut être représentée par l'équation :

$$\mathbf{h(m,n)= \sum_{i=0}^{hauteur-1} \sum_{j=0}^{largeur-1} g(i,j) f(m-i, n-j)} \qquad \mathbf{(3.1)}$$

D'où :

- f est l'image d'entrée.
- g est le filtre.
- h est l'image de sortie.

La double sommation est basée sur la largeur et la hauteur du noyau de convolution [2].

Par exemple, une convolution bidimensionnelle en utilisant une image de 3 x 3 d'entrée et un masque 3 × 3 ressemblerait comme suit:

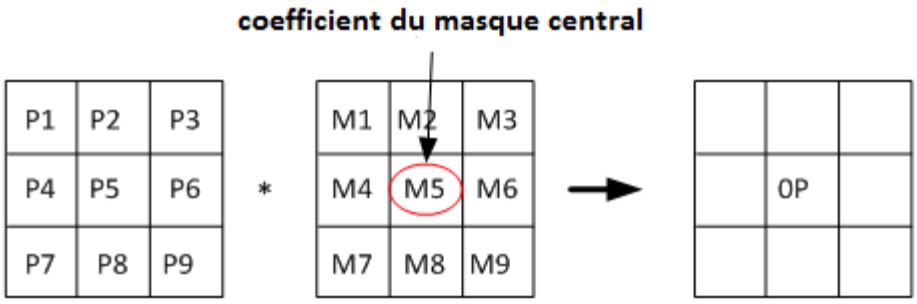

Figure 3.14: Opération de la convolution 2D.

OP= P1M9+P2M8+P3M7+P4M6+P5M5+P6M4+P7M3+P8M2+P9M1.

Afin de calculer un pixel de sortie pour un masque donné de taille m × n, mn multiplications et mn-1 additions sont nécessaires. Le masque Gaussien et l'une de ses propriétés importantes, à savoir la séparabilité, sont présentés plus en détail par la suite.

3.3 Le masque Gaussien

Le filtre Gaussien est un filtre de convolution. L'idée de convolution Gaussienne est d'utiliser cette enveloppe 2D en fonction d'étalement de point. Le degré de lissage est déterminé par l'écart-type σ de la Gaussienne. Depuis l'image est mémorisée en tant que collection de pixels discrets, une approximation discrète du Gaussien est nécessaire pour effectuer la convolution [2].

La distribution Gaussienne 1D est de la forme suivante:

$$g(x)=\frac{1}{\sqrt{2\pi\sigma^2}}e^{-x^2/(2\sigma^2)} \quad (3.2)$$

En 2D, La distribution Gaussienne est de la forme :

$$g(x,y)=\frac{1}{2\pi\sigma^2}e^{-(x^2+y^2)/2\sigma^2} \quad (3.3)$$

D'où :

- g est le poids du noyau Gaussien à l'emplacement des coordonnées x et y.
- Le paramètre σ est l'écart type de la distribution Gaussienne, qui détermine la netteté ou la douceur de la fonction Gaussienne.

- Le terme $\frac{1}{2\pi\sigma^2}$ est la constante de normalisation.

Une 2D Gaussienne générale est illustrée ci-dessous:

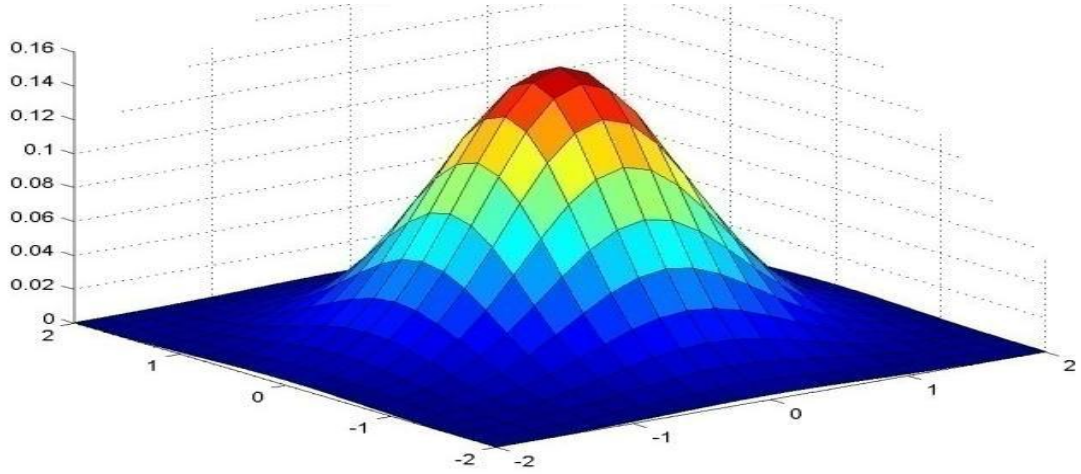

Figure 3.15 : Le masque Gaussien 2D.

L'intérêt de ce filtre est que l'on contrôle facilement le degré de filtrage à travers le paramètre $\boldsymbol{\sigma}$.

Le plus grand avantage des filtres Gaussien est qu'ils sont séparables.

3.4 Les filtres Gaussien Séparables et orientables

3.4.1 La séparabilité

Une application importante est que la convolution séparable avec un noyau Gaussien 2D peut être remplacée par une cascade de noyaux Gaussien 1D, ce qui rend l'ensemble du processus de convolution beaucoup plus efficace avec moins de multiplications.

En particulier, le produit de deux fonctions Gaussien 1D Gauss donne une Gaussienne de dimension supérieure, la fonction peut être représentée mathématiquement par:

$$\mathbf{g(x,y) = g(x)*g(y)} \qquad \mathbf{(3.4)}$$

Par conséquent, un filtre de convolution séparable est réalisé en deux étapes:

L'image d'entrée ou d'origine est convoluée avec un filtre de taille N x 1, puis le résultat est convolué avec un filtre de taille 1 × N.

Dans ce cas, la convolution séparable nécessite un total de 2N multiplications et 2N-2 additions, donc un nombre d'opération très réduit par rapport au cas non-séparables, particulièrement pour les filtres à grande échelle [2].

3.4.2 Les filtres orientables (directionnels)

Les Filtres directionnels ou orientables sont largement utilisés dans la vision par ordinateur et de traitement d'image, tels que l'analyse de mouvement, détection de bord et d'analyse de texture.

La direction implique que la sortie Oϴ (x, y) d'une opération de filtrage utilise un filtre orienté à un angle θ peut être calculé comme la combinaison linéaire d'un ensemble fini de M sorties $\{O_{\theta 0}(x, y), O_{\theta 1}((X, y) \ldots O_{\theta_{M-1}} (x, y)\}$ obtenu en appliquant le même filtre orienté selon les directions $\theta 0, \theta 1, \ldots\ldots\ldots, \theta M\text{-}1$.

3.5 Méthode de fonctionnement des Filtres Gaussien séparables et orientables

Un filtre séparable et orientable 2D peut être écrit comme [2]:

$$\mathbf{g_\theta(x,y)} = \sum_{r=-R}^{R} \mathbf{g_{iso}}(x - r\cos(\theta), y - r\sin(\theta))\, \mathbf{g^{1D}}(r) \qquad \textbf{(3.5)}$$

Où l'on a supposé que la taille de $g^{1D}(r)$ est égale à 2R + 1.

Le filtre décrit en (3.5) peut être appliqué à une image I (x, y) en deux étapes :

- la première étape : le Filtre $g_{iso}(x, y)$ est appliqué à l'image.

$$\mathbf{I_{iso}}(x, y) = I(x, y) * \mathbf{g_{iso}}(x, y) \qquad \textbf{(3.6)}$$

- la deuxième étape : l'opération suivante est appliquée à l'image $I_{iso}(x, y)$.

$$\mathbf{l_\theta}(x, y) = \sum_{r=-R}^{R} \mathbf{I_{iso}}\big(x - r\cos(\theta), y - r\sin(\theta)\big)\mathbf{g^{1D}}(r) \qquad \textbf{(3.7)}$$

L'opération décrite en (3.6) et (3.7) équivaut à l'opération dans laquelle l'image d'entrée I (x, y) est filtrée par un filtre Gaussien orientable orienté à la direction θ. La fonction $\mathbf{g_{iso}}(x, y)$ décrit un filtre séparable et peut donc être mis en œuvre de manière efficace. Plus spécifiquement, $\mathbf{g_{iso}}(x, y)$ peut être exprimée comme : $\mathbf{g_{iso}}x, y) = \mathbf{g_x}(x) * \mathbf{g_y}(y)$

Où :

$\mathbf{g_x}(x) = \frac{1}{2\pi\sigma_x^2} e^{-x^2/(2\sigma_x^2)}$ Et $\mathbf{g_y}(y) = e^{-y^2/(2\sigma_y^2)}$

Par conséquent, $g_{iso}(x, y)$ peut être appliquée à I (x, y) par un premier filtrage I (x, y) d'une manière horizontale à l'aide $g_x(x)$ puis en filtrant le résultat de manière verticale en utilisant $g_y(y)$.

L'équation (3.7) décrit une combinaison linéaire de versions décalées de l'image $\mathbf{I_{iso}}(\mathbf{x} - \mathbf{rcos}(\boldsymbol{\theta}), \mathbf{y} - \mathbf{rsin}(\boldsymbol{\theta}))$, qui dépendent de la direction θ de filtrage. Les coefficients de la combinaison linéaire sont égaux à la valeur de$\mathbf{g^{1D}}(\mathbf{r})$. L'image $\mathbf{I_{iso}}(x - rcos\ \theta, y - rsin\ \theta)$ peut être représentée comme la convolution entre l'image d'entrée I (x, y) et le filtre $\boldsymbol{g}\mathbf{iso}(x - rcos\ \theta, y - rsin\ \theta)$.

Ainsi, la mise en œuvre proposée est orientable dans le sens que le résultat final $I\theta(x,y)$, peut être exprimé comme une combinaison linéaire de la sortie de l'opération de filtrage $\mathbf{I_{iso}}(x - rcos\ \theta, y - rsin\ \theta)$ d'un ensemble de 2R + 1 des filtres fondamentaux $\mathbf{g_{iso}}(x - rcos\ \theta, y - rsin\ \theta)$, paramétré par r, appliqué sur l'image d'entrée I (x, y).

Le filtre isotrope $g_{iso}(x, y)$ est un filtre passe-bas et près de 100% de l'énergie du filtre est incluse dans la bande de fréquence [-3/σx, 3/σx]. Par conséquent, le $g_{iso}(x)$ de sortie, y obtenu par le filtrage de l'image d'entrée I (x, y) avec $g_{iso}(x, y)$ est une bande limitée dans la gamme de fréquences (-π, π] dans toutes les directions θ. Ainsi, l'équation (3.7) peut être modifiée :

$$\mathbf{I_\theta}(x, y) = \sum_{k=-[R/D]}^{[\frac{R}{D}]} \mathbf{I_{iso}}(\mathbf{x} - \mathbf{KDcos}(\boldsymbol{\theta}), \mathbf{y} - \mathbf{KDsin}(\boldsymbol{\theta}))\mathbf{g^{1D}}(\mathbf{KD}) \quad \textbf{(3.8)}$$

Avec :

$$\mathbf{g^{1D}(r) = g(KD)} = \frac{1}{\sqrt{2\pi(\sigma y^2 - \sigma x^2)}} e^{\frac{-(KD)^2}{2(\sigma y^2 - \sigma x^2)}} \quad \textbf{(3.9)}$$

$\mathbf{D} = \frac{\pi\sigma_x}{3}$, c'est le facteur de décimation **(3.10)**

3.6 Implémentation software d'un filtre Gaussien orientable

Pour extraire des différents attributs de l'image, le filtre orientable est utilisé. Nous menons plusieurs tests avec cette procédure pour plusieurs orientations sur une image IRM en utilisant l'outil **MATLAB.**

Le programme **MATLAB** (voir annexe) se déroule en 2 phases :

A) Phase n°1 : le programme (steerGauss.m) peut être utilisé pour évaluer la première dérivée directionnelle d'un filtre Gaussien orienté à la direction θ, se déroule en quatre étapes suivant le schéma suivant :

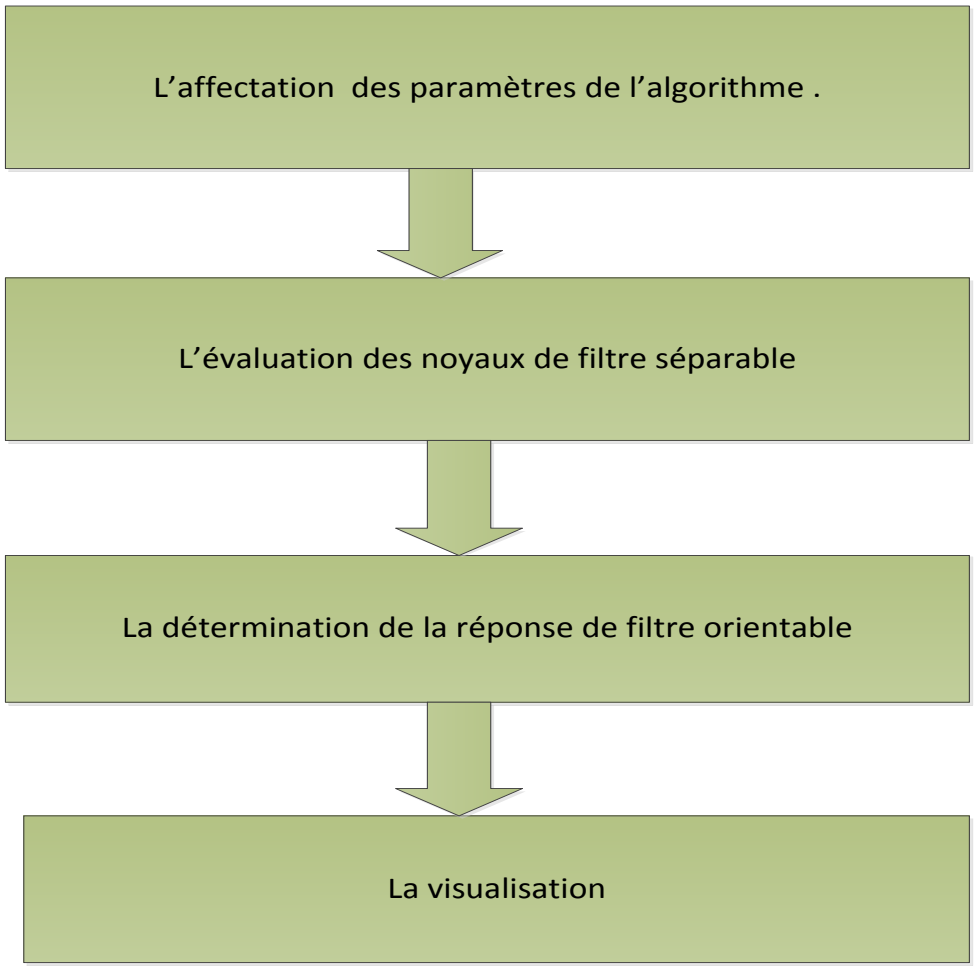

Figure 3.16 : Le schéma de la fonction Gaussienne orientable**.**

La première étape : L'affectation des paramètres de l'algorithme.

[J, H] = STEERGAUSE (I, thêta, SIGMA, VIS)

Evalue la dérivée directionnelle de l'image d'entrée I, orientée à thêta degrés par rapport aux lignes d'image. L'écart-type de Noyau Gaussien est donné par SIGMA (supposé être égal à L'unité par défaut). Les paramètres de filtre sont renvoyés à l'utilisateur dans la structure H.

[J, H] = STEERGAUSE (I, H, VIS)

Évalue la dérivée directionnelle de l'image d'entrée I, en utilisant le filtre calculé précédemment et stocké dans la note H.

H qui est une structure, avec les champs suivants:

H.g : la Gaussienne 1D.

H.gp: première dérivée de la gaussienne 1D.

H.theta: orientation du filtre.

H.sigma: dérivation standard Gaussienne.

On notera que le support du filtre est automatiquement ajusté (en fonction de la valeur de Sigma). En général, la visualisation peut être activée (ou désactivée) en définissant VIS = TRUE (ou FALSE). Par défaut, la visualisation est désactivée.

Cette étape du programme réalise les tâches suivantes :

- L'image de charge par défaut (si aucune condition), convertit en niveaux de gris.
- Les arguments d'entrées du processus (selon le mode d'utilisation).
- Attribuer l'orientation du filtre par défaut (si non fourni).
- Attribuer l'état de visualisation par défaut (si non fourni).
- Extraire les paramètres du filtre.

La deuxième étape : L'évaluation des noyaux de filtre séparable.

- Calculer le noyau du filtre (si non fourni par l'utilisateur).
- Déterminer le support du filtre nécessaire (pour le gaussien).
- Évaluer le filtre Gaussien 1D (et son dérivé).

Voici la fonction Gaussienne : **g =** $e^{(-\frac{(x^2)}{2*sigma^2})}$.

Non seulement la définition de la dérivée Gaussienne est erronée, le Gaussien lui-même est également faux: il doit être normalisé :

Gp = $-(\frac{x}{sigma^2})$ * g;

Pour notre cas, cette constante est considérée comme « 1 »

Et voici sa dérivée : **gp=** $-\frac{x}{sigma^2} * e^{-\frac{x^2}{2*sigma^2}}$ **.**

- Stoker le noyau du filtre (pour les exécutions ultérieures).

La troisième étape : La détermination de la réponse du filtre orienté.

- Calculer l'image gradients (en utilisant la séparabilité). Le filtrage en deux étapes (Ix et Iy) est que la Gaussienne est séparable.
- Évaluer la réponse du filtre orienté comme suit :
  J = cos(theta)*Ix+sin(theta)*Iy .

La quatrième étape : La visualisation.

- Création d'une fenêtre de figure et d'affichage des résultats.

B) Phase n°2 : le programme (runDemo.m) qui permettra de créer une animation montrant les dérivées directionnelles régulièrement espacées de 0 degrés jusqu'à 360 degrés (en incrémentant de 30 degrés).

3.6.1 Les résultats de MATLAB

Les figures ci-dessous montrent les résultats de l'application du filtre Gaussien orientable sur des images IRM.

- Pour thêta égale à « 0 degré » :

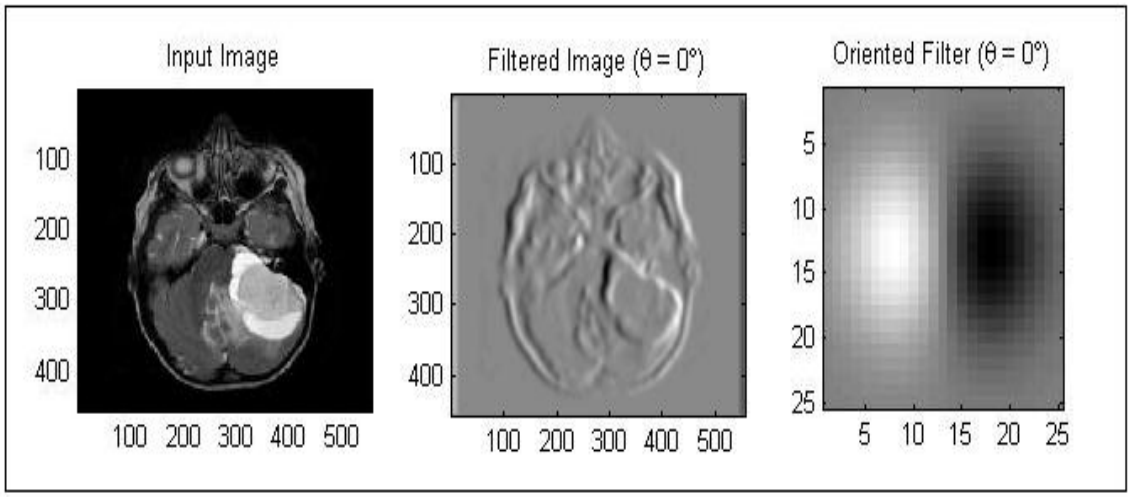

Figure 3.17: Application d'un filtre Gaussien orientable sur une image IRM pour thêta « 0 ».

- Pour thêta égale à « 80 degré » :

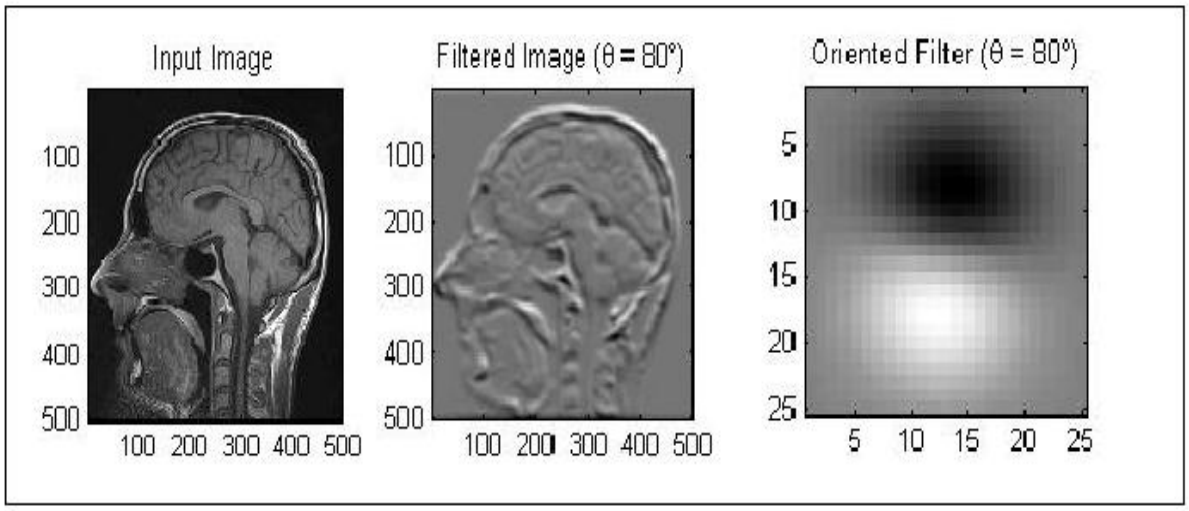

Figure 3.18: Application d'un filtre Gaussien orientable sur une image IRM pour thêta « 80 ».

- Pour thêta égale a « 200 degré » :

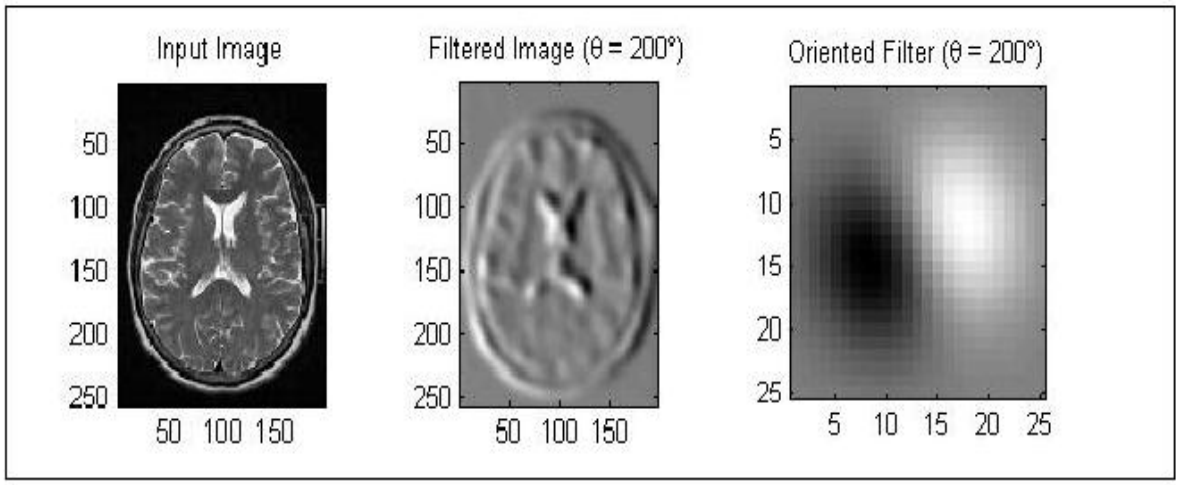

Figure 3.19: Application d'un filtre Gaussien orientable sur une image IRM pour thêta « 200 ».

Les 13 orientations de l'application d'un filtre Gaussien orientable sur une image IRM sont présentées par la figure suivante :

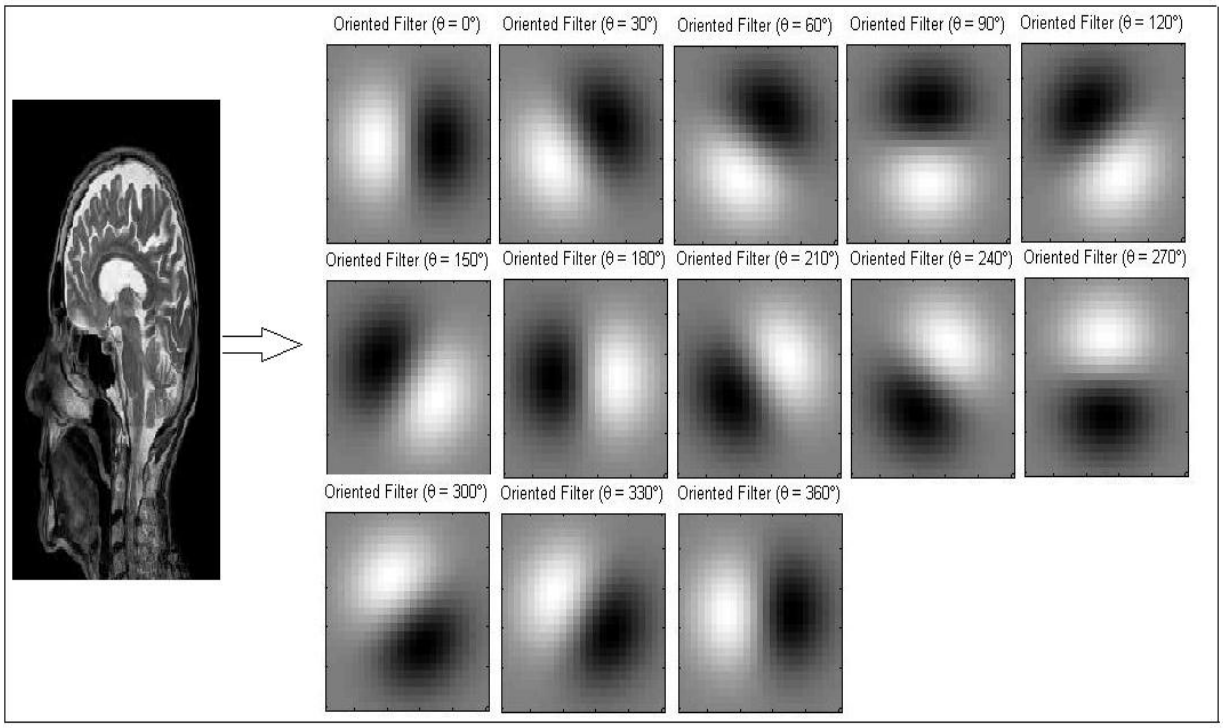

Figure 3.20: Les 13 orientations de l'application d'un filtre Gaussien orientable sur une image IRM.

3.7 Conclusion

Dans ce chapitre, nous avons donné une description théorique sur le produit de convolution ainsi que les filtres Gaussien, et le principe de fonctionnement des filtres séparables et orientables.

Les résultats de simulation obtenus sous l'outil **MATLAB** montrent bien l'efficacité de l'algorithme d'un filtre Gaussien orientable avec 13 orientations (0 jusqu'à 360 avec un pas de 30).

Le chapitre suivant sera consacré à l'étude comparative des deux architectures (le filtre Gaussien 2D et le filtre Gaussien séparable) ainsi que l'implémentation hardware du filtre orientable sur un circuit FPGA.

Chapitre 4

Implémentation hardware d'un filtre Gaussien orientable

4.1 Introduction

L'objectif principal de ce chapitre est l'implémentation sur un circuit FPGA (Field Programmable GateArray) d'une architecture d'un filtre orientable performante sur une plate-forme Zedboard de la famille Zynq.

Dans un premier temps, une architecture du filtre Gaussien 2D a été développée et simulée. Par la suite, une deuxième architecture du filtre Gaussien séparable a été développée et simulée sous deux environnements Xilinx ISE 14.7 et Vivado2014.4.
La troisième partie de ce chapitre sera consacrée à l'étude comparative des deux filtres. Lorsque la taille du masque augmente, une comparaison des performances des deux architectures montre que les filtres Gaussien séparables sont moins complexes et le nombre de multiplieurs est très réduit.
Le filtre Gaussien séparable est à la base d'une implémentation d'un filtre orientable qui sera détaillée dans la dernière partie de ce chapitre.

4.2 Implémentation d'un filtre gaussien 2D

Une RAM est utilisée pour stocker une image de test de taille 105 × 103.La sortie de la RAM est introduite à l'entrée du contrôleur qui génère les adresses pour choisir notre pixel et ses adjacents pour toute l'image.

Ces pixels sont introduits à l'entrée des multiplieurs (8 bits) avec les coefficients de masque de bloc de contrôleur. Les 9 sorties des multiplieurs (16bits) seront reliés à un additionneur dont la sortie (18 bits) est le résultat de convolution à deux dimensions entre l'image105 × 103 et le masque Gaussien de 3 × 3 (voir Table 4.1).

37	61	8
61	100	14
8	14	2

Table 4.1: Un masque Gaussien 2D avec une moyenne=0, σ=1 et N=0.0016.

La représentation d'un bloc diagramme de l'implémentation d'un filtre Gaussien 2D est illustrée ci-dessous :

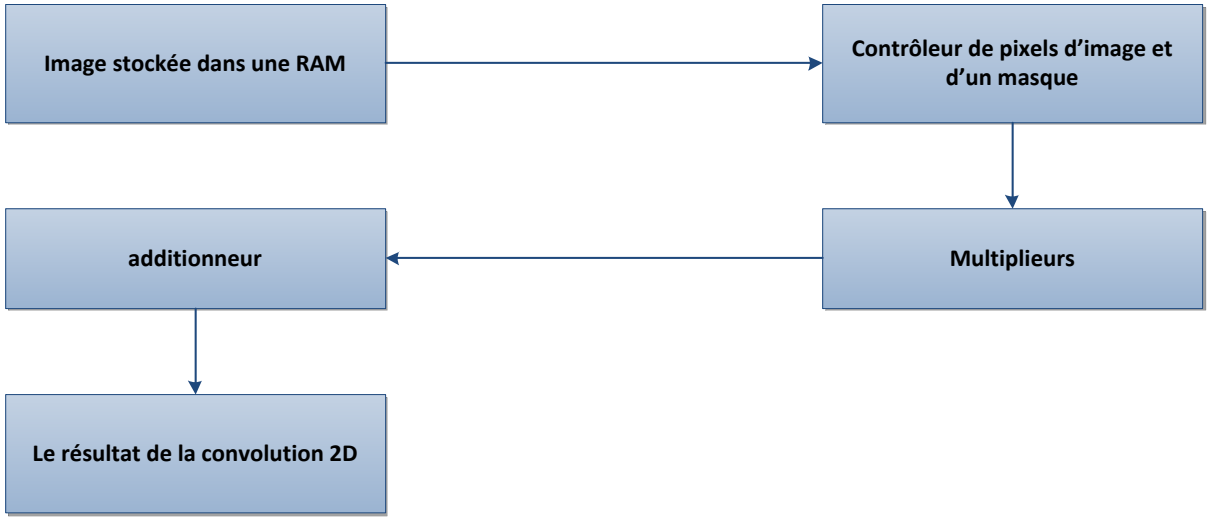

Figure 4.21: Bloc diagramme de l'implémentation d'un filtre Gaussien 2D.

L'architecture globale du procédé général de la convolution de deux dimensions est représentée ci-dessous :

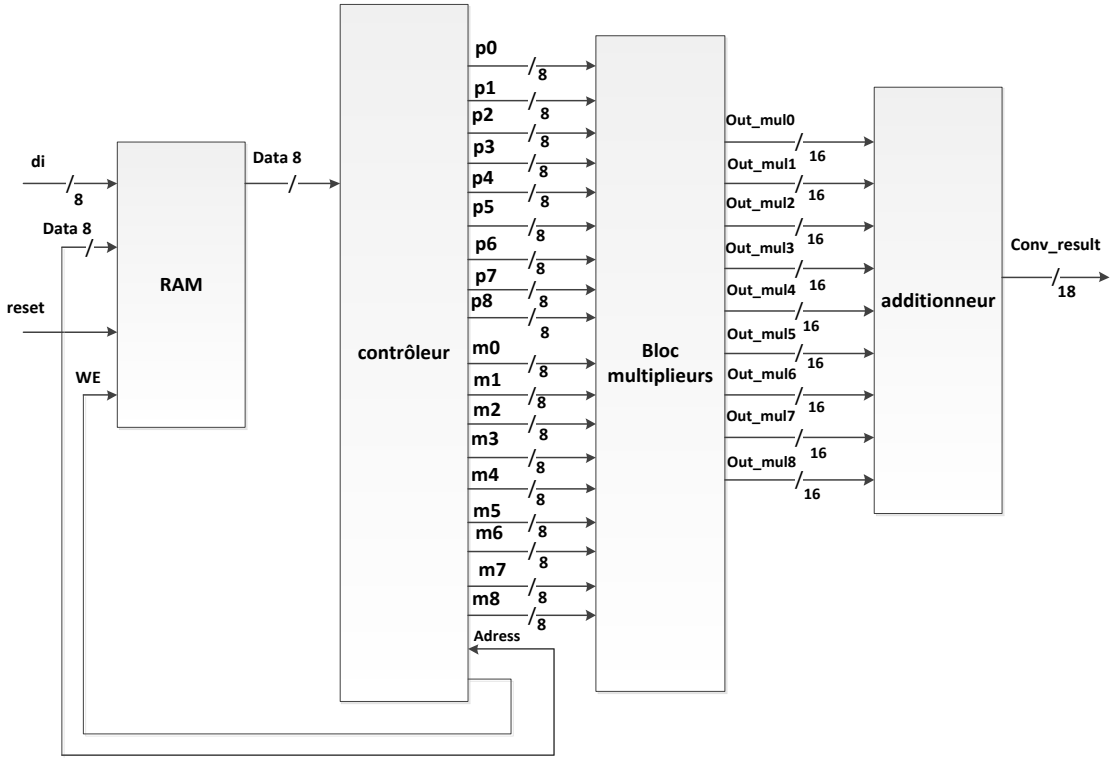

Figure 4.22: Schéma synoptique de l'implémentation d'un filtre Gaussien 2D.

4.2.1 Résultats de simulation de filtre Gaussien 2D

Les résultats indiqués ci-dessous sont obtenus en utilisant le simulateur de Xilinx à savoir ISIM :

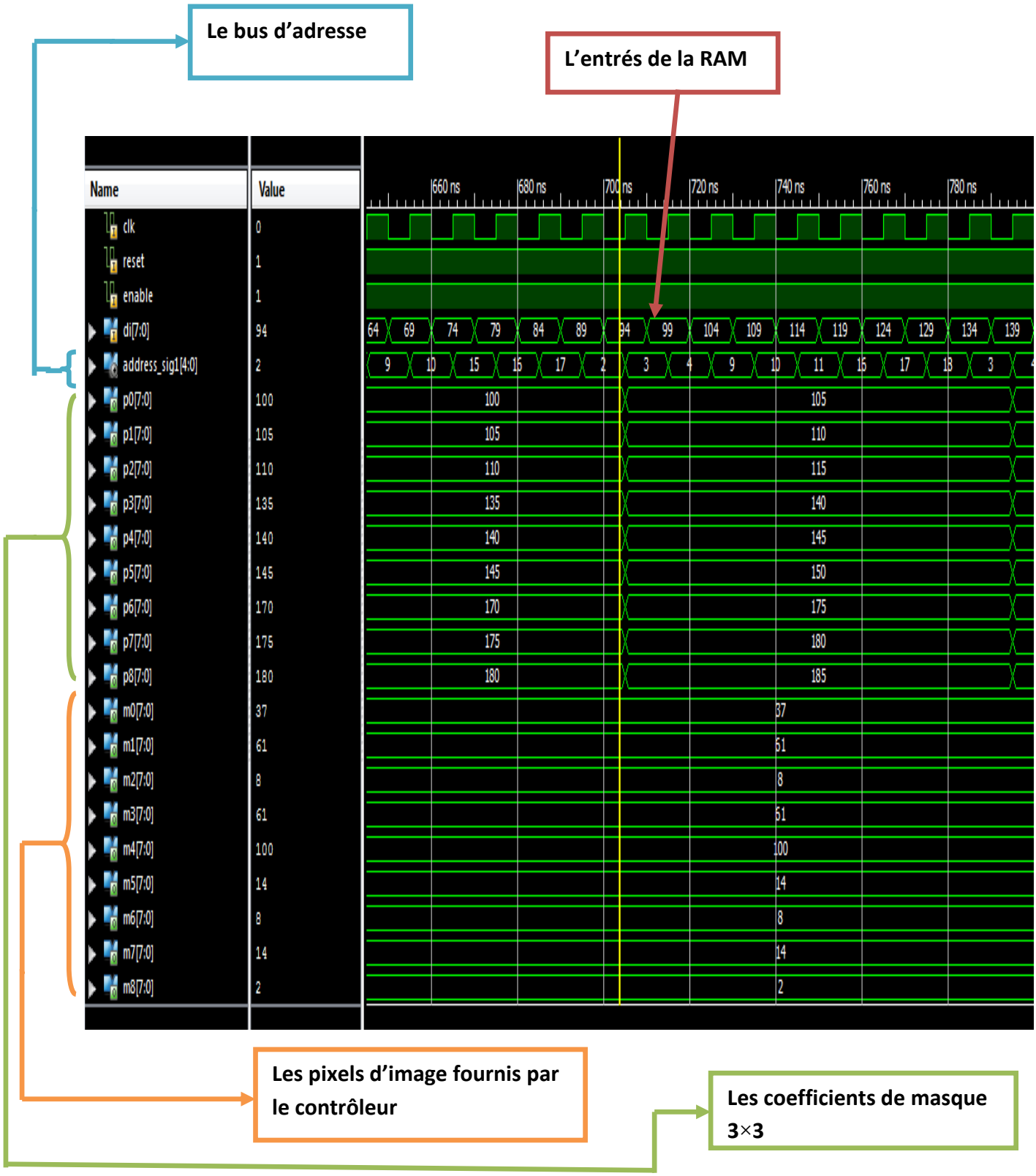

Figure 4.23: Simulation de bloc « RAM_controller » du filtre Gaussien 2D.

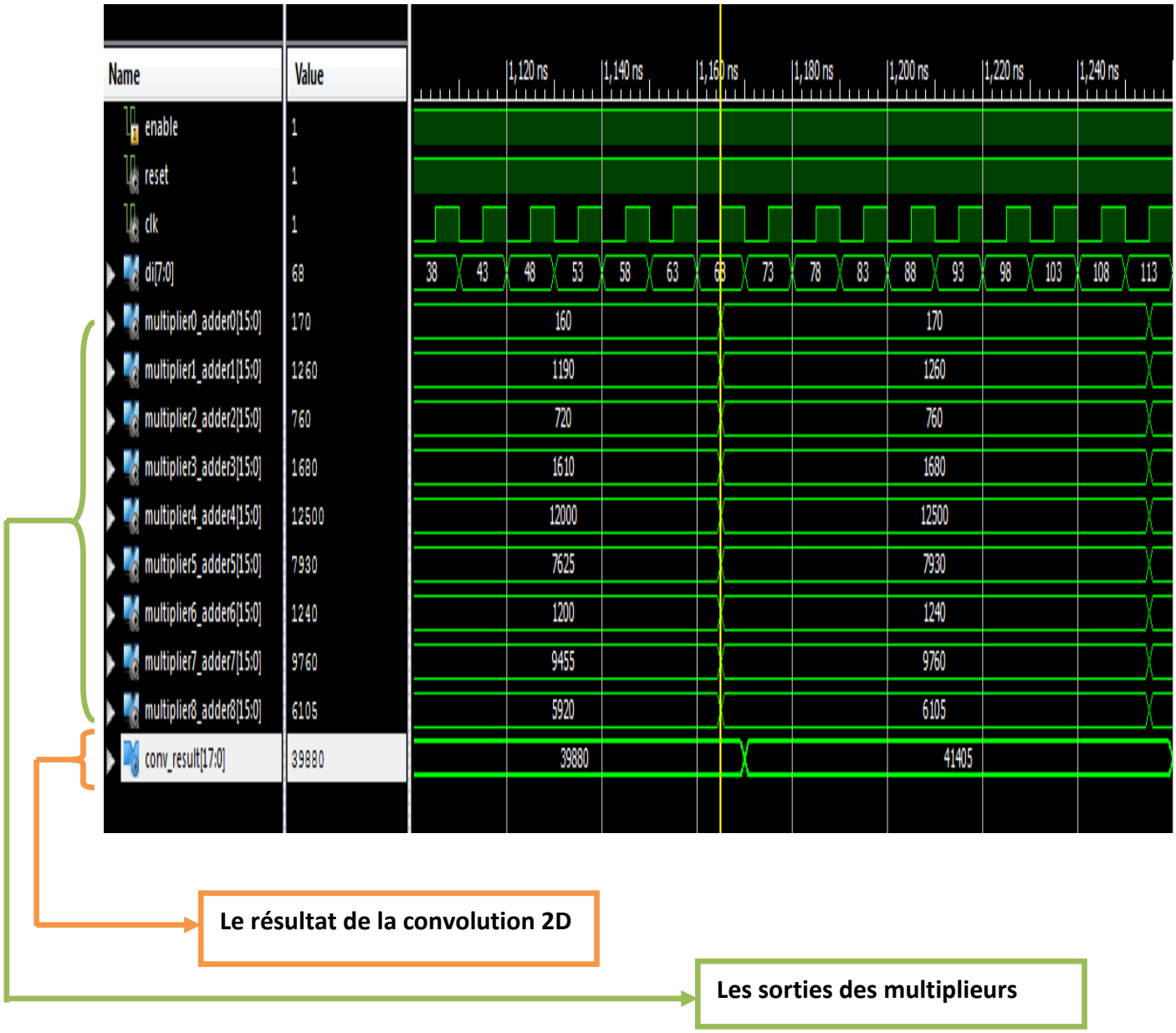

Figure 4.24: Simulation de bloc «final_convolution» de filtre Gaussien 2D.

4.2.2 Synthèse et implémentation de filtre Gaussien 2D

Les résultats de la synthèse sont obtenus avec l'outil Vivado2014.4, qui donne une estimation du taux d'occupation et du temps d'exécution de notre architecture.

Le tableau ci-dessous résume les ressources utilisées dans le circuit programmable XC7Z020 de la famille Zynq :

Figure 4.25 : Résultat de l'implémentation d'un filtre Gaussien 2D

4.3 Implémentation d'un filtre Gaussien séparable

Comme nous l'avons vu dans le chapitre précédent, le masque Gaussien séparable est dérivé en utilisant les équations (3.3) et (3.4), dont la moyenne=0, σ =1 et le facteur de normalisation N = 0,0016 sont montrés ci-dessous:

61	100	14

Table 4.2: Un masque Gaussien horizontal avec une moyenne=0, σ=1, N=0.0016.

61
100
14

Table 4.3 : Un masque Gaussien vertical avec une moyenne=0, σ=1, N=0.0016.

La représentation d'un bloc diagramme de l'implémentation d'un filtre Gaussien séparable est illustrée ci-dessous :

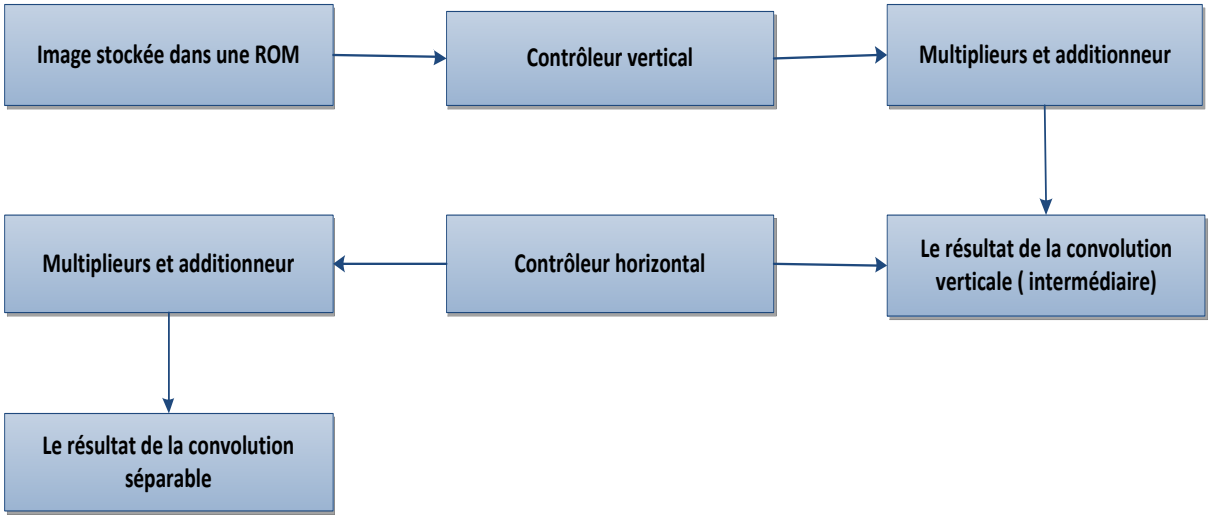

Figure 4.26: Bloc diagramme du filtre Gaussien séparable.

Le même principe que l'approche de convolution 2D présentée dans la section 4.1, la ROM est utilisée pour stocker une image de test 105 × 103. Les entrées des multiplieurs (8 bits) sont obtenues à partir des pixels d'image et les coefficients de masque fournis par le contrôleur.

Chaque multiplieur génère une sortie qui est représentée par 16 bits. Les sorties des multiplicateurs sont transmises à un additionneur qui fournit une sortie de 17 bits. La sortie de l'additionneur est le résultat de convolution verticale (intermédiaire), représentée sur la figure4.7.

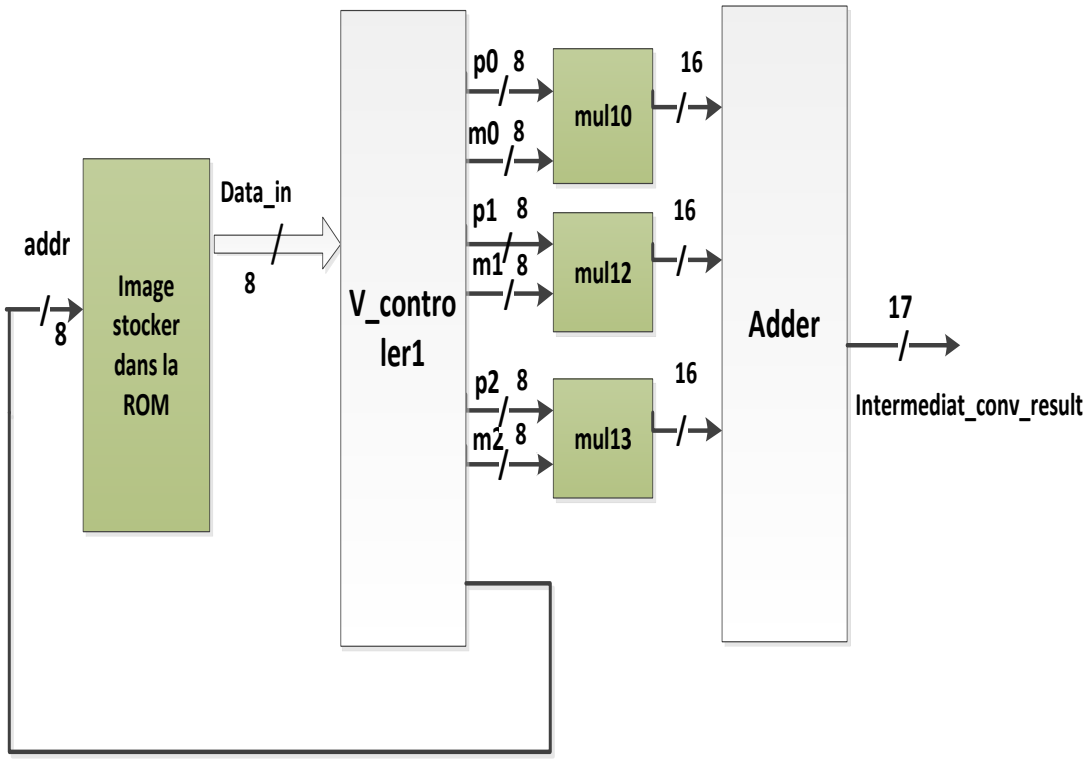

Figure 4.27: Architecture du bloc de calcul de la convolution horizontale « Intermediat_conv_result ».

La sortie du bloc de la convolution horizontale est transmise à l'entrée du bloc contrôleur.

Les coefficients de masque Gaussien horizontal (8 bits) et les pixels de résultat de convolution horizontale (17 bits) sont injectés à l'entrée des 3 multiplieurs. Les sorties des multiplieurs (25 bits) sont reliés à un additionneur de 27 bits appelé « adder_vertical ». La sortie de l'additionneur 27 bits est le résultat final de convolution séparable (voir figure 4.8).

L'architecture du bloc «separable_convolution_result» de filtre Gaussien séparable est illustrée comme suit:

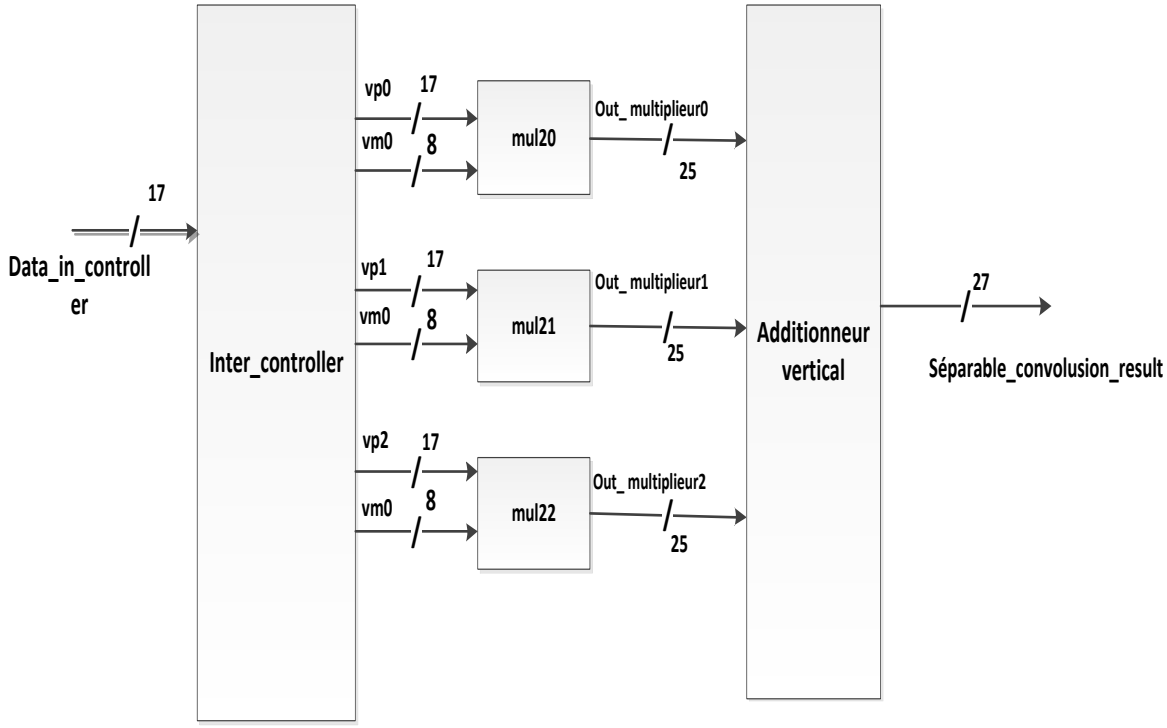

Figure 4.28: Architecture du filtre Gaussien séparable vertical.

L'architecture globale de filtre Gaussien séparable est représentée sur la figure ci-dessous :

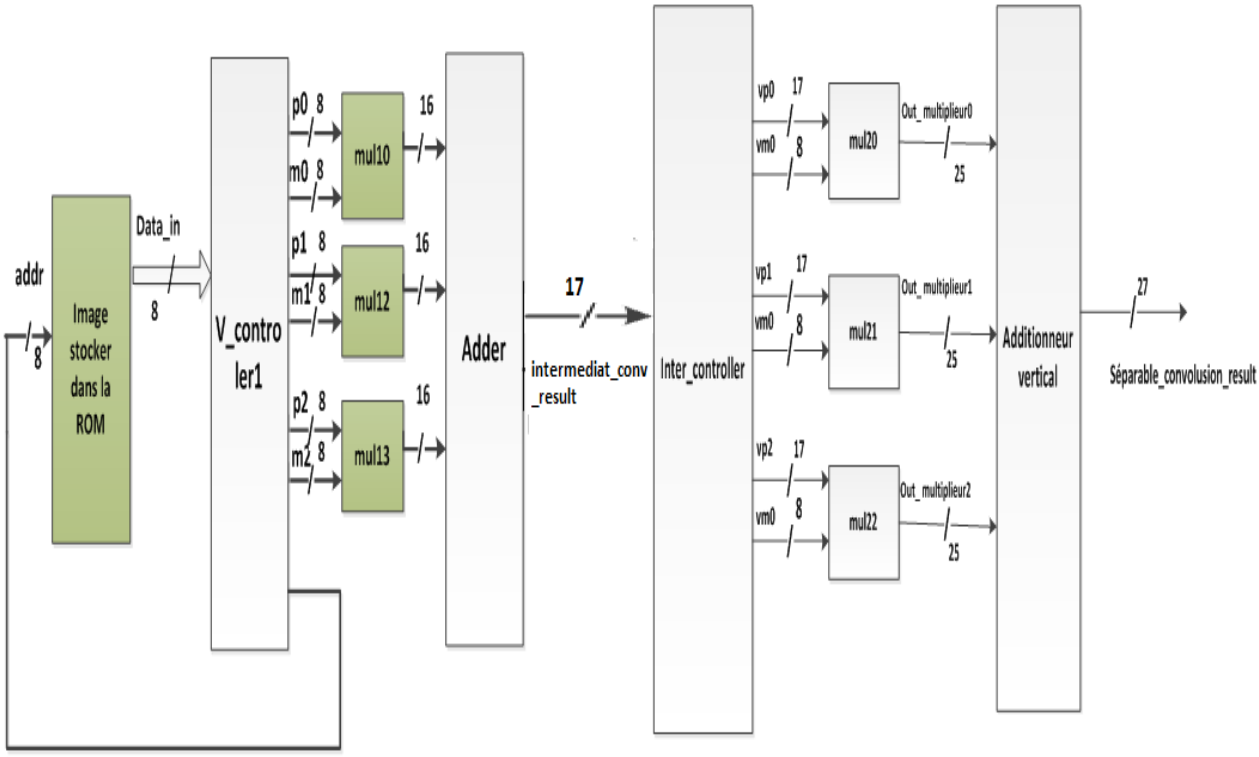

Figure 4.29 : Architecture globale du filtre Gaussien séparable (horizontale-verticale).

4.3.1 Résultat des simulations d'un filtre Gaussien séparable

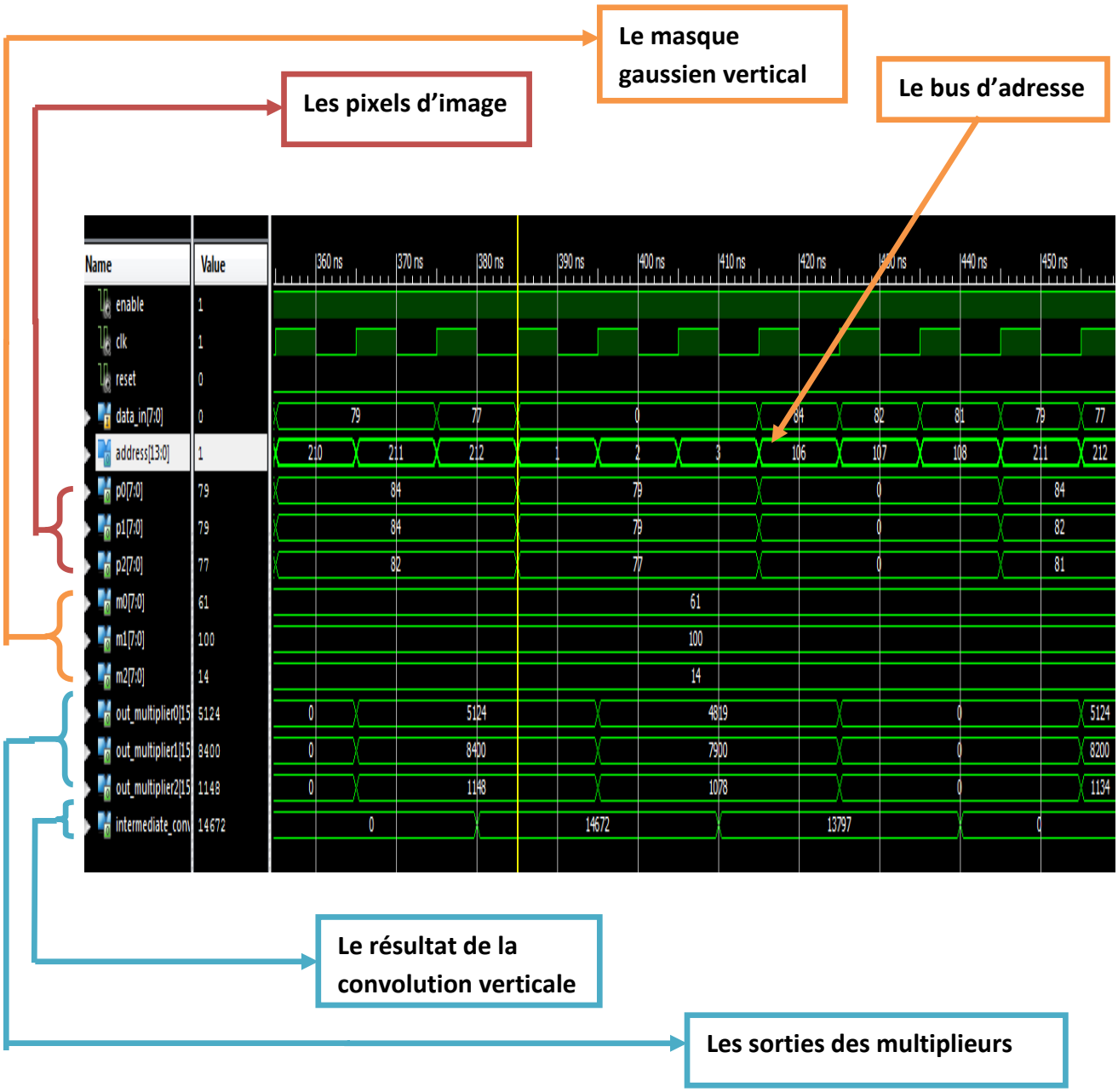

Figure 4.30: Simulation d'un filtre Gaussien séparable à la direction verticale.

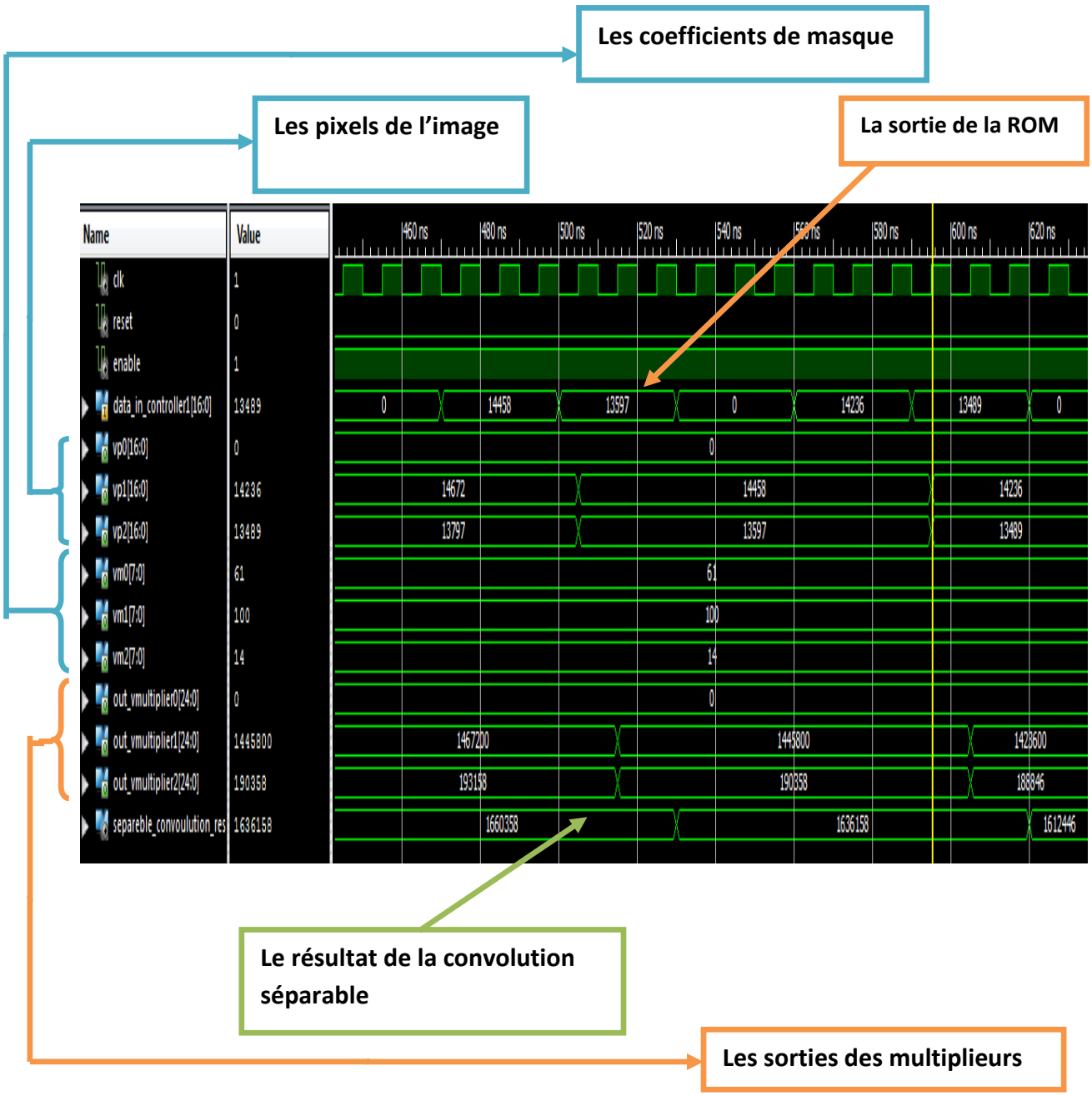

Figure 4.31: Simulation d'un filtre Gaussien séparable (verticale-horizontale).

4.3.2 Synthèse et implémentation d'un filtre Gaussien séparable

Le tableau ci-dessous résume les ressources utilisées dans le circuit programmable XC7Z020 de la famille Zynq :

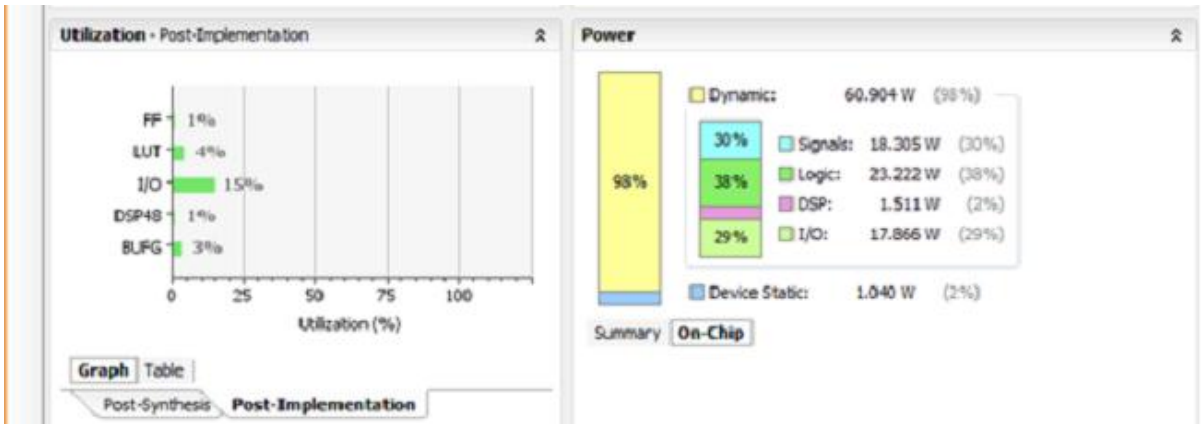

Figure 4.32 : Résultat de l'implémentation d'un filtre Gaussien séparable.

4.4 Comparaison entre les deux filtres Gaussiens (2D et séparable)

Les résultats des ressources sur le circuit FPGA pour les deux implémentations, du filtre 2D et filtre séparable, sont représentés par le tableau suivant :

	Number of Slice Registers	Number of Slice LUTs	Number of bonded'IOBs	Number of BUFG/ BUF GCTRLs	Temps d'exécution (hardware)
Filtre Gaussien 2D (3x3)	8 (0.01%)	10 (0.02%)	24 (12%)	1 (3.13%)	3.123 ns
Filtre Gaussien Séparable (3x3)	299 (0.28%)	1862 (3.50%)	30 (15%)	1 (3.13%)	3.769 ns

Table 4.4: les ressources occupées sur le circuit FPGA.

D'après les comparaisons ci-dessus, la méthode du filtre Gaussien 2D est préférable car elle utilise moins de ressources. Pratiquement, la mise en œuvre pourrait ne pas être possible si nous utilisons des masques de grandes tailles car le nombre de multiplieurs va

augmenter. D'où le filtre Gaussien séparable est plus favorable en termes de performances pour ces types de masques, car du point de vue ressources, il nécessite moins de multiplieurs.

4.5 Implémentation d'un filtre Gaussien séparable avec un masque (5x5)

La méthode de convolution choisie à savoir la méthode de convolution séparable est prolongée pour une image de 6 × 6 et un masque Gaussien séparable de 1 × 5 et 5 × 1 (table 4.5 et 4.6).

100	61	14	1	0

Table 4.5 : Le masque Gaussien séparable horizontal avec une moyenne=0, σ=1, N=0.0016.

100
61
14
1
0

Table 4.6: Le masque Gaussien séparable vertical avec une moyenne=0, σ=1, N=0.0016.

L'architecture globale de filtre Gaussien séparable avec un masque 5x5 est illustrée ci-dessous:

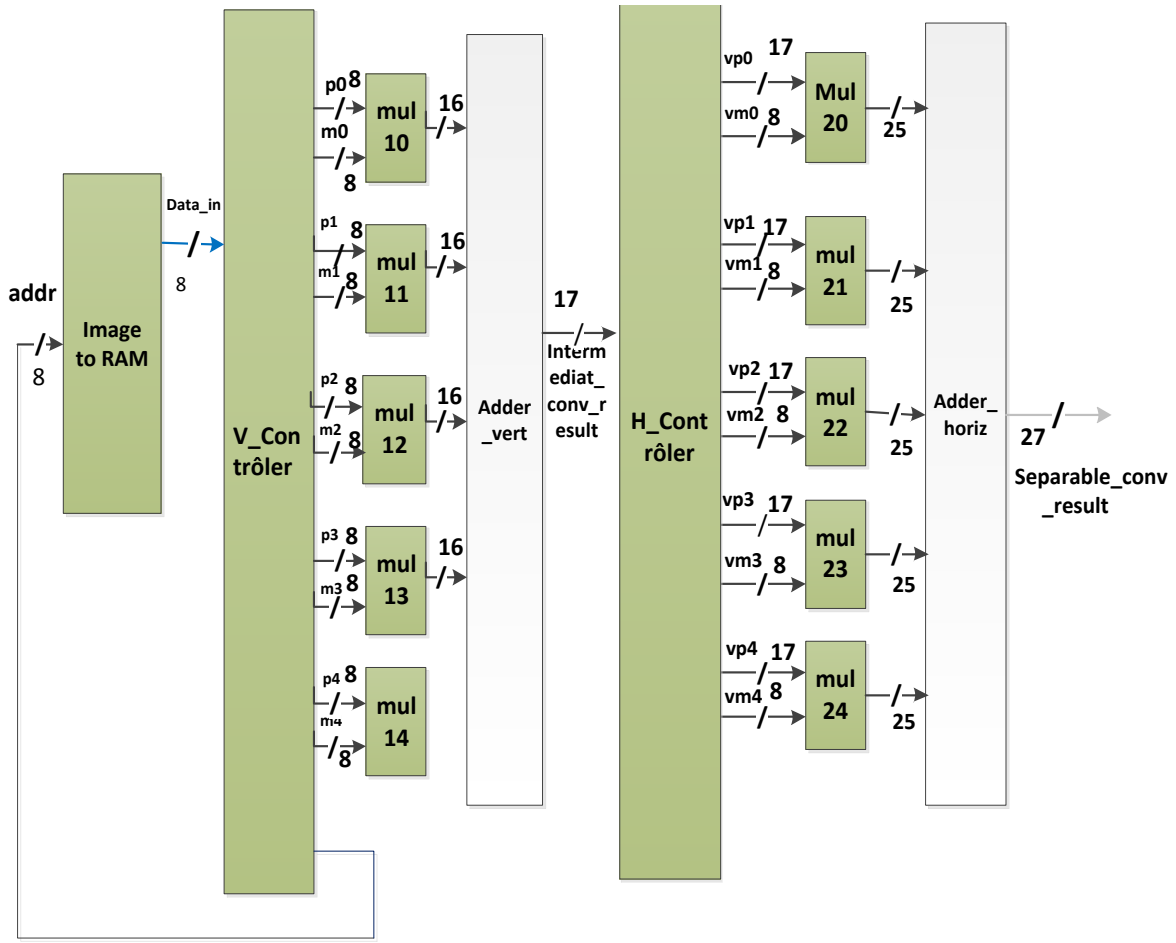

Figure 4.33: Architecture global d'un filtre Gaussien séparable avec un masque de 5x5.

4.5.1 Résultat de simulation d'un filtre Gaussien séparable avec un masque de 5x5

La simulation fonctionnelle du bloc « Intermediat_conv_result » et le bloc « Séparable_conv_result » du filtre Gaussien séparable est donnée par :

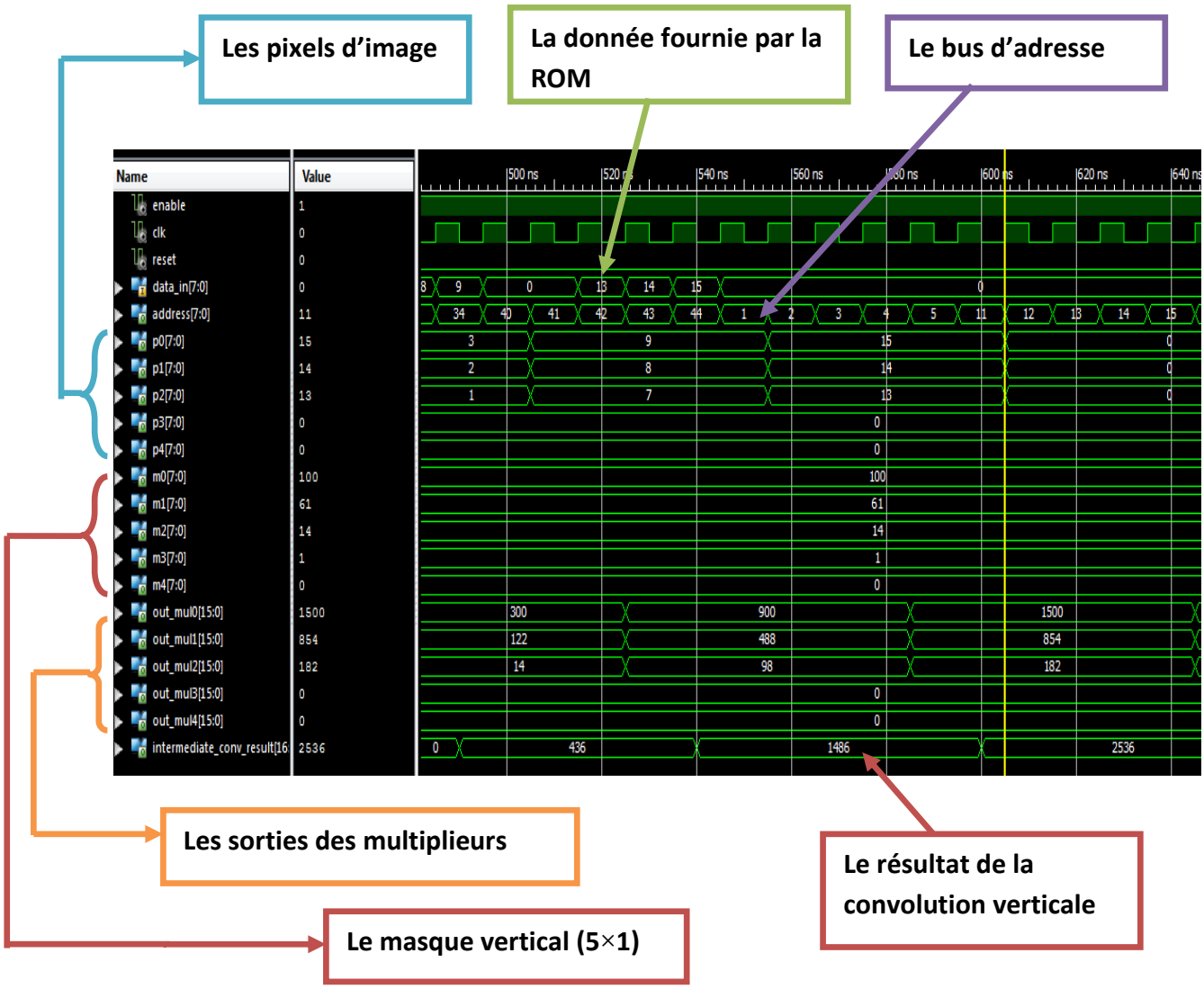

Figure 4.34 : Simulation du bloc « Intermediat_conv_result ».

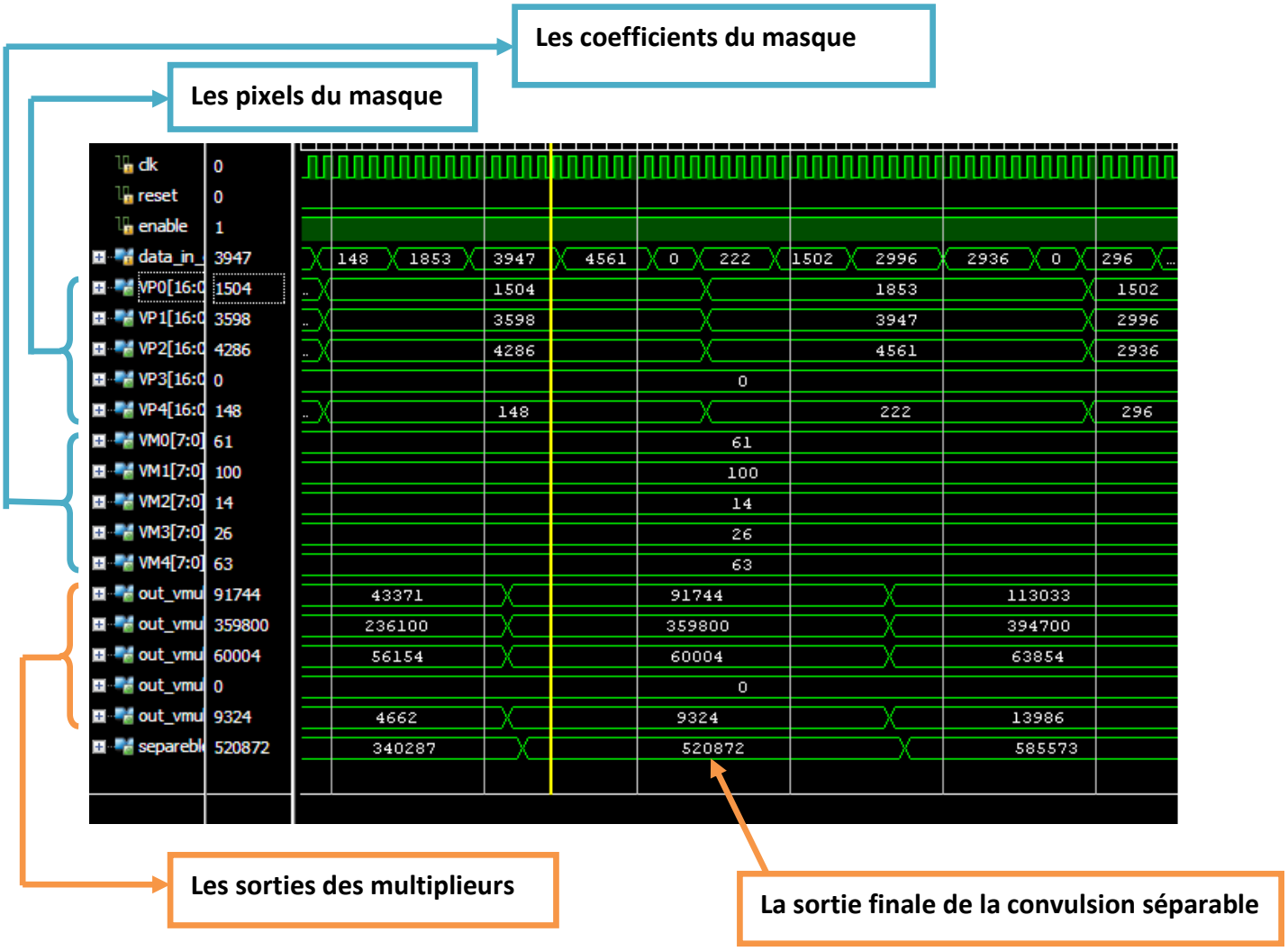

Figure 4.35: Simulation du bloc « Séparable_conv_result ».

4.5.2 Synthèse et implémentation d'un filtre Gaussien séparable (5x5)

Le tableau ci-dessous résume les ressources utilisées dans le circuit programmable XC7Z020 de la famille Zynq :

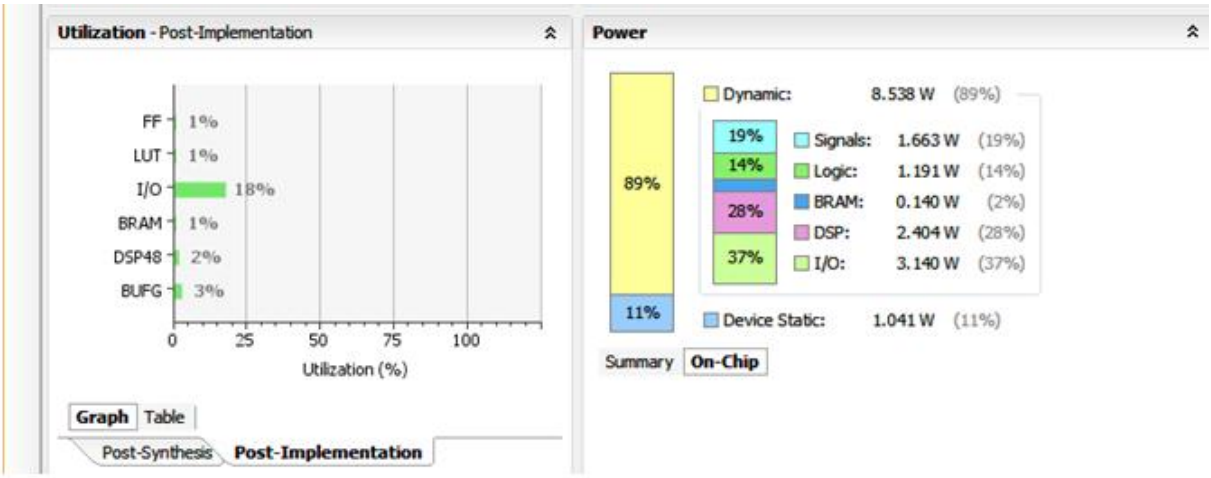

Figure 4.36: Résultat de l'implémentation d'un filtre Gaussien séparable (5x5).

4.5.3 Implémentation du filtre Gaussien séparable (5x5)

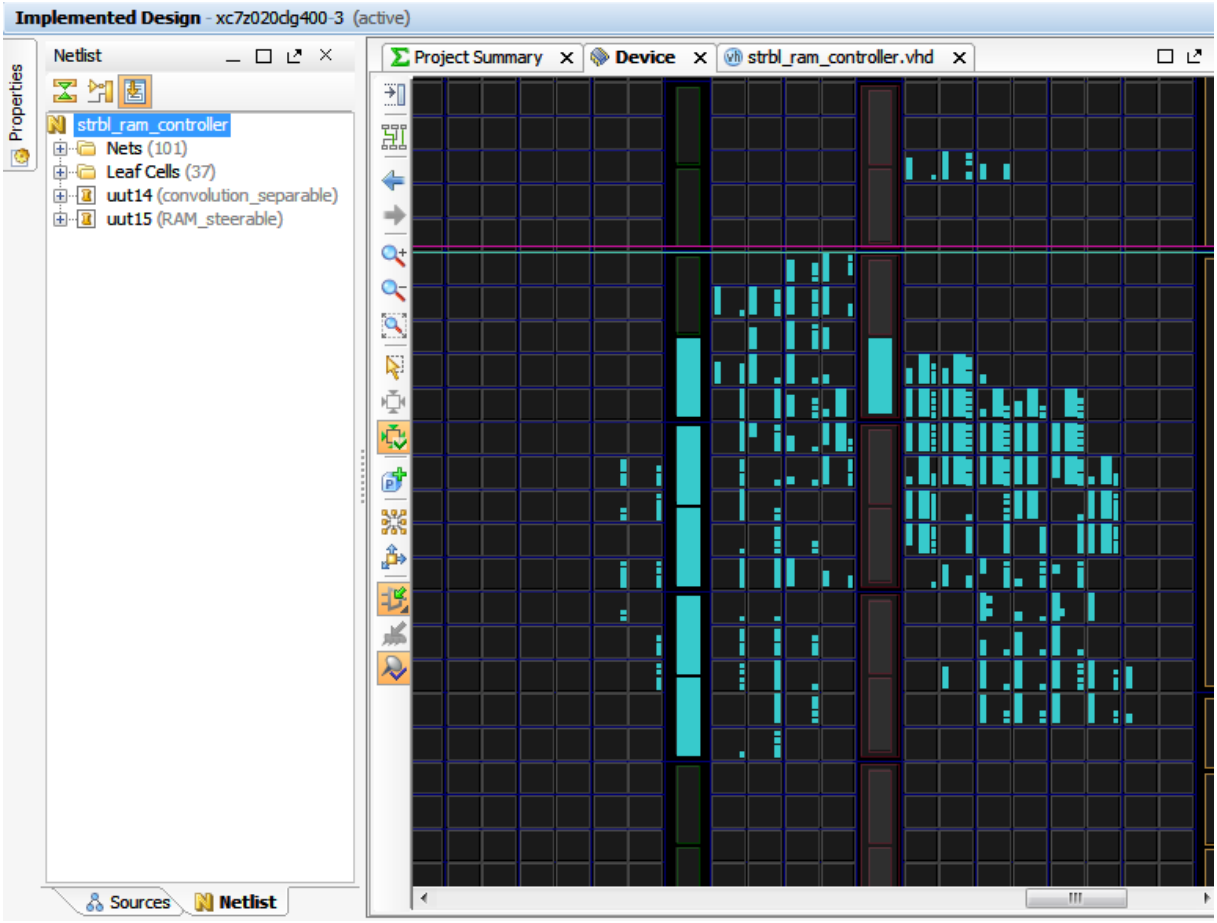

Figure 4.37: La surface occupée par l'architecture du filtre Gaussien séparable (5x5).

4.6 Implémentation d'une architecture du filtre orientable appliquée au filtre séparable avec un masque de 5x5

Le filtre orientable est appliqué sur les résultats obtenus de la convolution séparable entre une image 6 x 6 et un masque Gaussien 5 × 1 et 1 x 5. Pour l'application de l'orientabilité, un masque Gaussien orientable est dérivé en utilisant l'équation (3.9) et le facteur de décimation utilisant l'équation (3.10).

Le dérivé du masque Gaussien orientable de 3 × 1 et 1 × 3 pour une moyenne =0, σx =3, σy = 5, le facteur de Normalisation N = 0.001 et le facteur de décimation D = 3 sont présentés ci-dessous :

8	33	76

Table 4.7 : Un masque Gaussien horizontal avec une moyenne=0, σx=3, σy=5 et N=0.0016.

8
33
76

Table 4.8 : Un masque Gaussien vertical avec une moyenne=0, σx=1, σy=1 et N=0.0016.

Dans le concept orientable, nous avons accès à des pixels dans des directions différentes en fonction du facteur de décimation. Ensuite, les pixels sont multipliés par les coefficients de pondération du masque Gaussien, ces derniers sont injectés à l'entrée de l'additionneur pour obtenir des résultats orientables finaux dans une direction particulière. La représentation du bloc diagramme de l'architecture orientable appliquée sur les résultats de la convolution séparable en deux directions horizontale et verticale est indiquée ci-dessous :

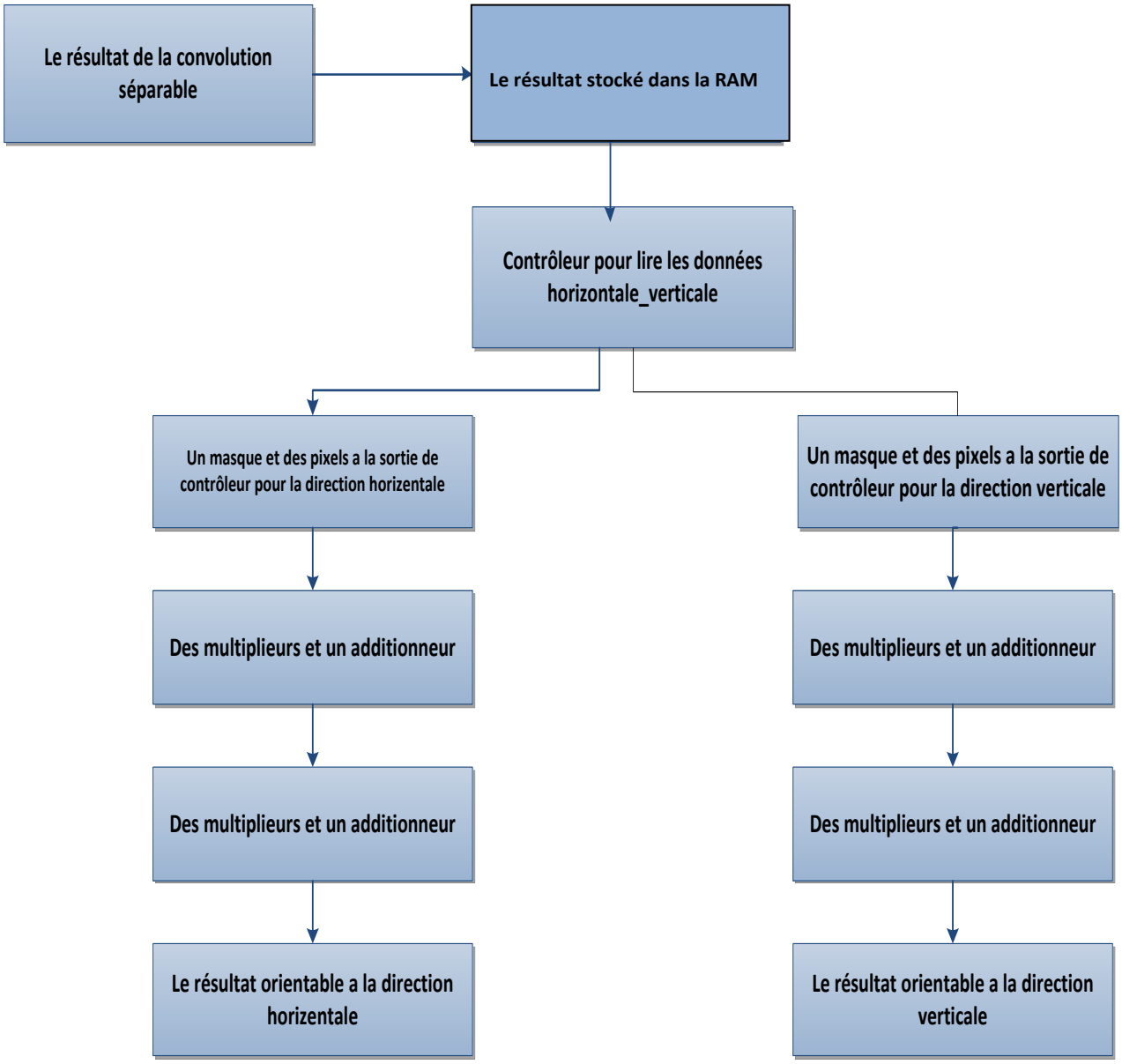

Figure 4.38: Bloc diagramme du filtre orientable.

L'architecture globale du filtre orientable sur une image 6× 6 en utilisant un masque Gaussien de 3× 1 et 1 x 3 dans des directions horizontales, verticales est illustrée ci-dessous:

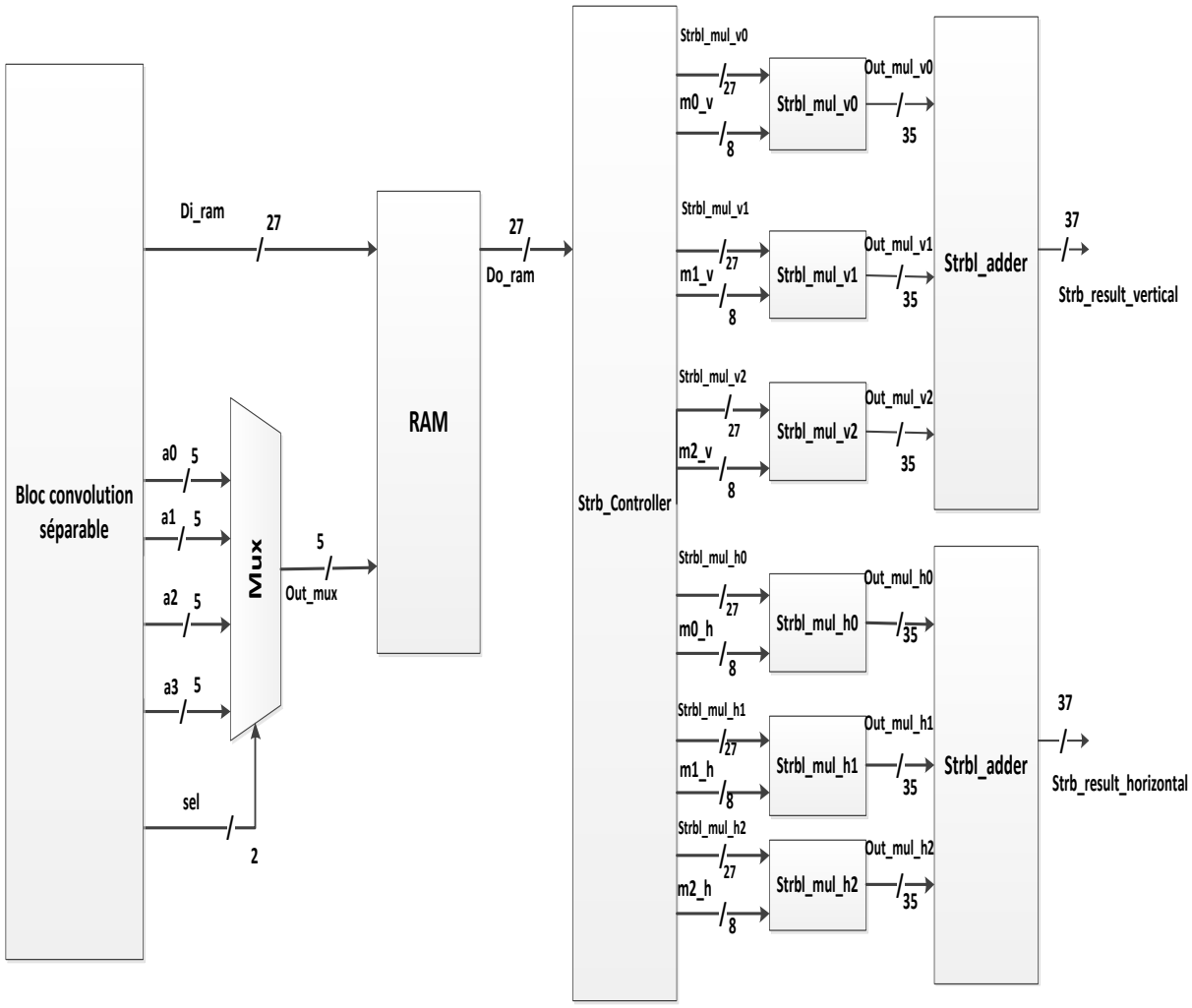

Figure 4.39 : Architecture globale du filtre orientable.

4.6.1 Bloc RAM

L'architecture du bloc« strbl_ram » est présentée par la figure suivante :

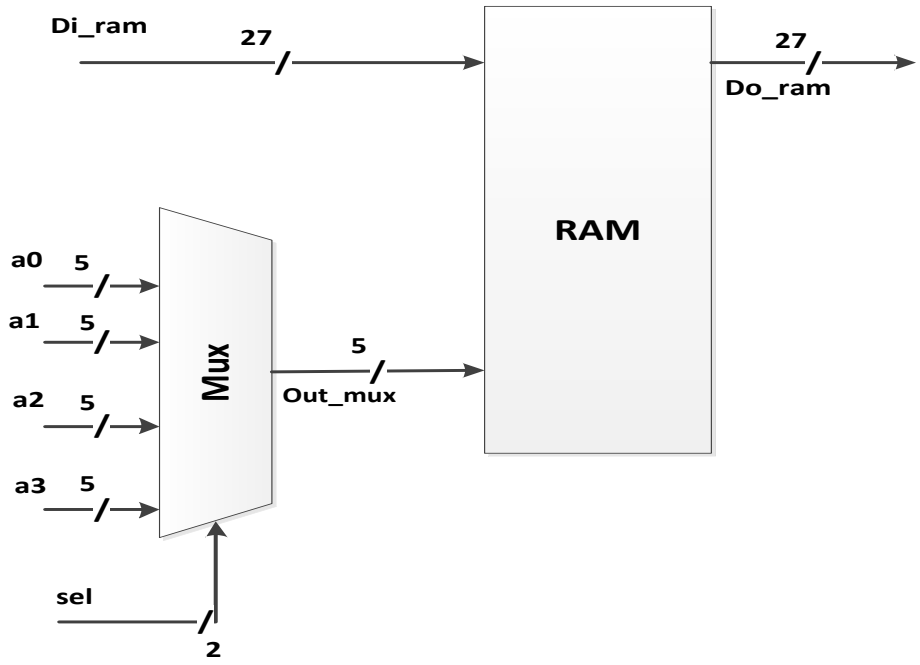

Figure 4.40 : Architecture de bloc « strbl_ram ».

4.6.1.1 Résultat de simulation de bloc RAM

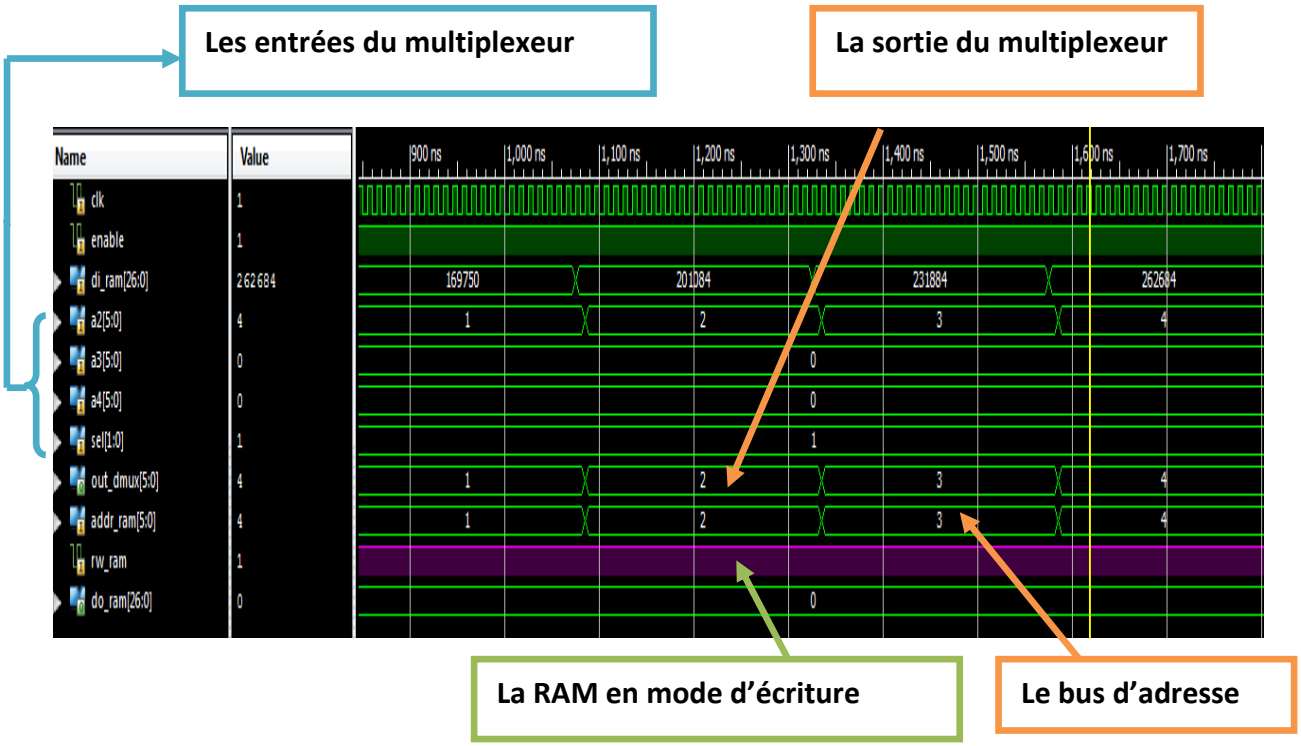

Figure 4.41 : Simulation du bloc « strbl_ram » en mode d'écriture.

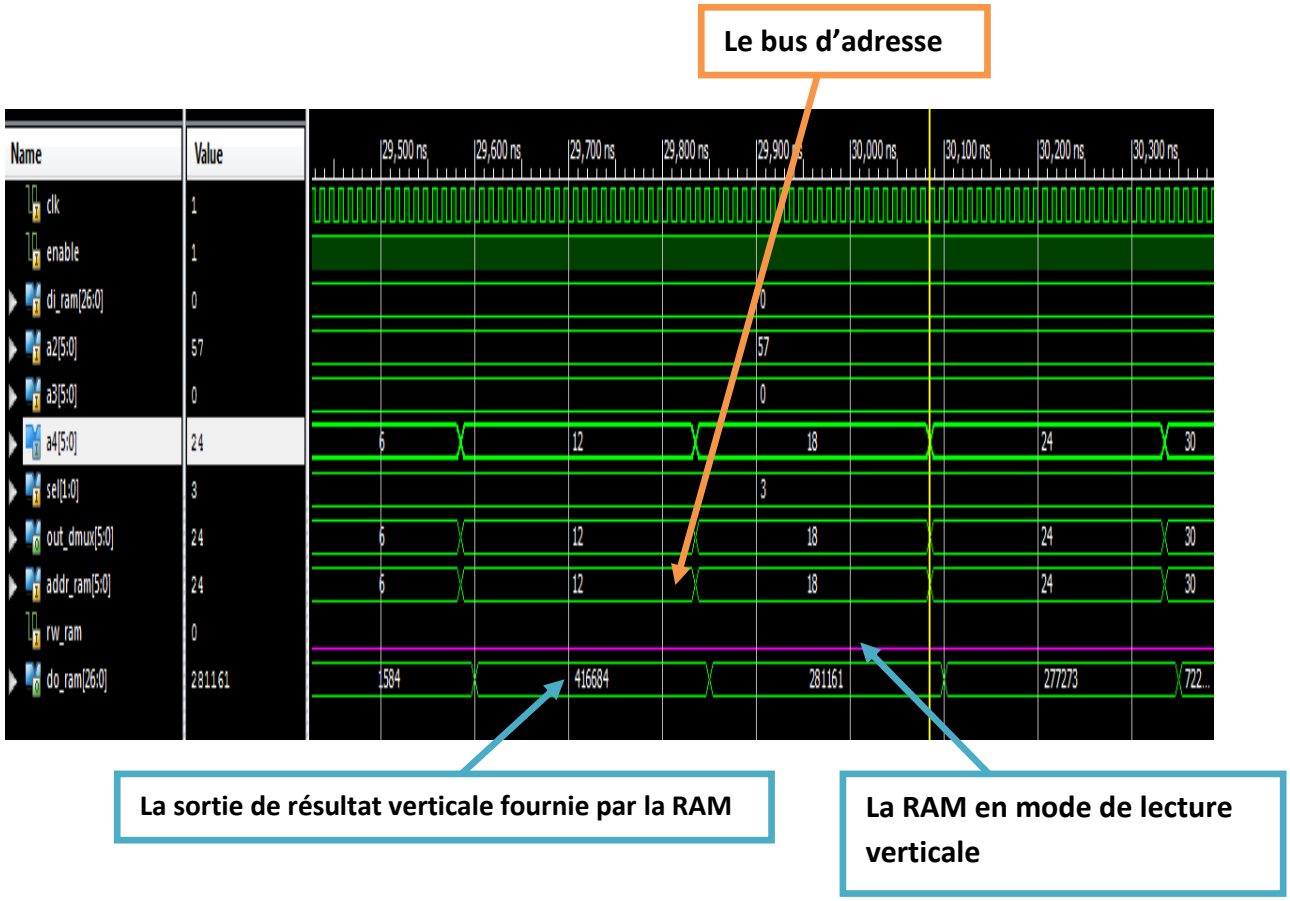

Figure 4.42 : Simulation de bloc « strbl_ram » en mode lecture des pixels vertical.

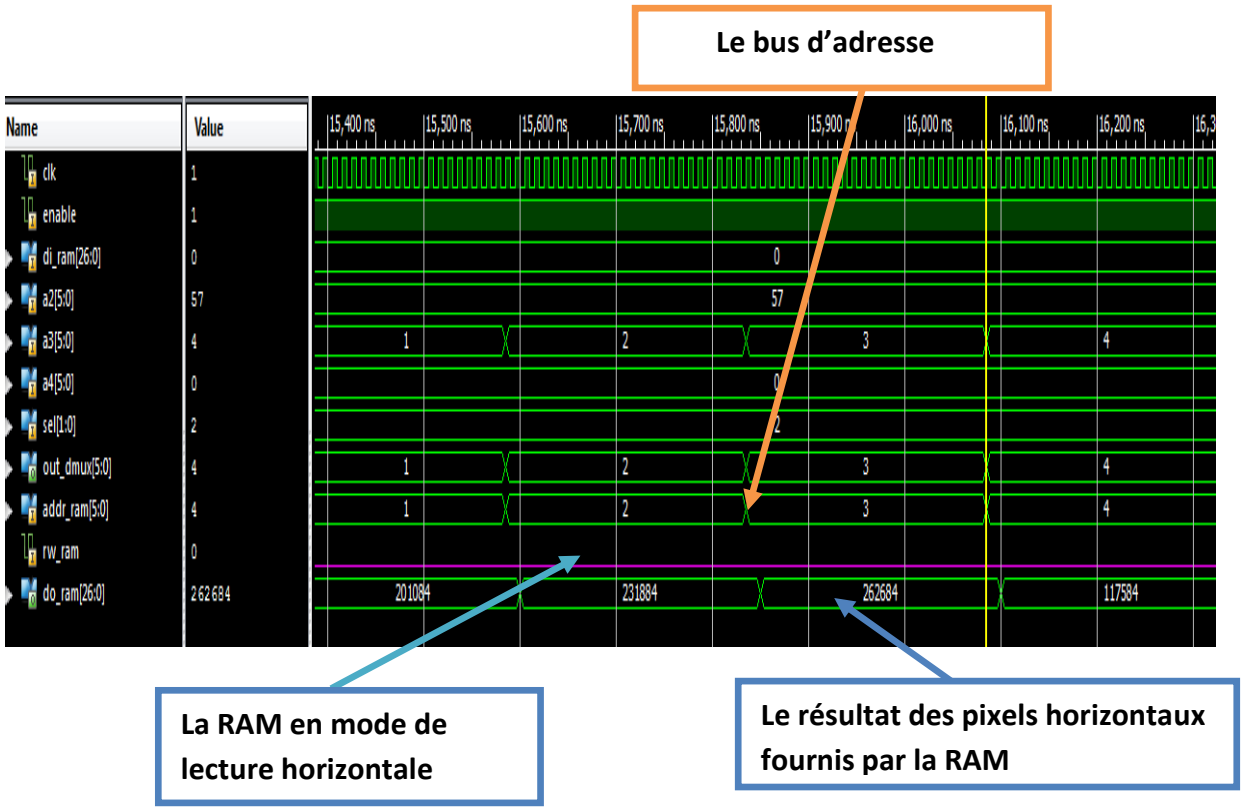

Figure 4.43 : Simulation de bloc strbl_ram en mode lecture des pixels horizontaux.

4.7 Conclusion

Dans ce chapitre, nous avons présenté la mise en œuvre d'un filtre gaussien 2D et d'un filtre gaussien séparable sur une plate-forme XC7Z020 de la famille Zynq. Apres l'étude comparative, nous constatons que les filtres gaussien 2D sont plus performants quand la taille du masque est inférieure à (3x3). Par contre, quand la taille du masque augmente (5x5 dans notre cas) les performances du filtre gaussien séparables sont nettement meilleures car le nombre des multiplieurs est très réduit.

Conclusion générale

Conclusion générale

L'objectif principal dans ce mémoire est de mettre en œuvre une implémentation hardware d'une architecture performante d'un filtre orientable sur une plate-forme **ZedBoard** XC7Z020 de la famille Zynq dont le but est l'extraction des attributs des images cérébrales.

La première étape a été consacrée à la présentation des circuits FPGAs de XILINX étant donné que nous allons implémenter notre application sur la carte **ZedBoard.**
La seconde partie était portée sur les généralités des différentes méthodes d'extraction d'attributs. Pour cela, nous avons présenté les différents filtres destinés à l'extraction des attributs pour les images médicales. Notre choix s'est arrêté sur les filtres gaussiens orientables.

Dans la troisième partie, nous avons présenté l'implémentation software sous l'outil MATLAB des filtres orientables. Les résultats obtenus au niveau de la simulation montrent l'efficacité de l'algorithme du filtre orientable pour les 13 orientations (0 jusqu'à 360 avec un pas de 30).

La dernière étape est dédiée à l'implémentation hardware sur un circuit programmable de type FPGA des filtres Gaussien 2D, des filtres Gaussien séparables et les filtres Gaussien séparables orientables.

Les résultats obtenus sont validés par plusieurs bancs de tests de simulation moyennant un environnement de simulation VHDL Test Bench. Le circuit XC7Z020clg 484-1de la famille Zynq a été utilisé pour l'implémentation hardware: les performances ont été justifiées en termes de ressources matérielles utilisées et du temps d'exécution.

Cependant, nous estimons par ce travail avoir apporté une modeste contribution dans les domaines respectifs de classification et de la segmentation, et nous souhaitons que dans le futur qu'il y ait un enrichissement du travail de point de vue implémentation en :
-Changement de type de carte (plus de ressources).
-Optimisation des ressources occupées par notre architecture.
Enfin, il serait intéressant d'intégrer ce circuit de base dans l'implémentation du filtre de Gabor utilisé pour la détection de la géométrie d'un objet.

Bibliographie

[1] W.T.Freeman, E.H. Adelson, «The Design and Use of Steerable Filters», Institut de la technologie, Cambridge, IEEE, September 1991.

[2] J-P.COCQUEREZ et S.PHILIP. « Analyse d'image, filtrage et segmentation », Edition Masson, Paris, 1995.

[3] J.Landré, « Analyse multirésolution pour la recherche et l'indexation d'images par le contenu dans les bases de donnée images-application a la base d'images paléontologique trans'tyfipal », Thèse de Doctorat, Université de Bourgogne, 2005.

[4] K.Albeau « Analyse à grande échelle des textures des séquences protéiques via l'approche Hydrophobic Cluster Analysis (HCA) », Thèse de Doctorat, Université de Versailles-Saint Quentin en Yve-lines, France, 2005.

[5] J.Ferayhia, « Traitement de données géophysiques à base de filtre non-linéaire et adaptatif », Thèse de Doctorat, Faculté des hydrocarbures et de la Chimie, Boumerdes, 2009.

[6] R.Kachouri « Classification multi-modèles des images dans les bases hétérogènes », Thèse de Doctorat, université d'Evry-Val d'Essonne, 2010.

[7] A.Jojinipelly « Implementation of Separable & Steerable Gaussian Smoothers on an FPGA», Thèse de Magistère, Université de New Orleans, 2010.

[8] M.Lehamel « Segmentation d'images texturées à partir des attributs Fractale », Thèse de Magistère en automatique, Université de Mouloud Mammeri Tizi-Ouzou, 2012.

[9] ZedBoard(Zynq™ Evaluation and Development) Hardware User's Guide,Version 1.1, 01-08-2012.

[10] H. Aissaoui, M.A.Belhabri, « Analyse de la texture: Filtrage et Matrice de cooccurrence », Mémoire de licence en informatique, Université de Abou BakrBelkaid, Tlemcen, 2013-2014.

[11] C. Ferhoum, H.Tariket, «Implémentation d'un filtre Médian directionnel pour la détection des régions d'intérêt sur les images médicales », Mémoire de Master en Imagerie et Appareillage Biomédical, Université de Boumerdes, 2015.

[12] N.Keddou , M.F. Zereg, « Extraction des paramètres caractéristiques des images IRM pour la classification des tumeurs cérébrales », Mémoire de Master en Imagerie et Appareillage Biomédical, Université de Boumerdes, 2015.

[13] D. Saptano, « Conception d'un outil de prototypage rapide sur le FPGA pour des applications de traitement d'images », Thèse de Doctorat, Université de Bourgogne, 2011.

[14] E.Sanchez, « Les circuits FPGA », Cours à Ecole Polytechnique Fédérale de Lausanne.

Webographie

[15] www.xilinx.com

Annexe

Code VHDL convolution 2D

Code VHDL du filtre 2D

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

entitystrbl_ram_controller is

port(clk,enable,reset : in std_logic;

out_addresse : out std_logic_vector (5 downto 0);

data_ram: out  STD_LOGIC_VECTOR (26 downto 0));

endstrbl_ram_controller;

architecture Behavioral of strbl_ram_controller is

componentconvolution_separable is

Port (      clk, reset,enable : in  STD_LOGIC;

           rw_ram: out std_logic;

sel : out std_logic_vector (1 downto 0);

           addresse_h,addresse_v,cmptr_ecriture: out std_logic_vector (5 downt
           separeble_convoulution_result : out  STD_LOGIC_vector(26 downto 0));

end component;

componentram_steerable is

port(clk,enable,rw_ram: in std_logic;

addr_ram: in std_logic_vector (5 downto 0);

di_ram: in std_logic_vector (26 dow,to 0);

do_ram: out std_logic_vector (26 downto 0));

end  component;

componentdmux is

port(

 a1,a2,a3,a4: in std_logic_vector(5 downto 0);
```

```
sel:instd_logic_vector(1 downto 0);
out_dmux: out std_logic_vector(5 downto 0));
end component;
begin
uut14: convolution_separable
port map (      clk =>clk, reset => reset, enable => enable,qq =>qq,sel =>sel,
separeble_convoulution_result =>data_ram_sig,
            cmptr_ecriture =>cmptr_ecriture,
            addresse_h =>addresse_h,
            addresse_v =>addresse_v,
rw_ram =>sig_rw);
uut15: ram_steerable
port map          ( clk =>clk, enable => enable, rw_ram     =>sig_rw, addr_ram =>sig_addr,
                    di_ram =>data_ram_sig, do_ram =>data_ram);
uut18:dmux
port map(a1=>qq, a2=>cmptr_ecriture, a3=>addresse_h, a4=>addresse_v, sel=>sel,
out_dmux=>sig_addr);
out_addresse<= sig_addr;
end Behavioral;
```

Code VHDL convolution séparable 3x3

Code VHDL du filtre Gaussien séparable (3x3)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entityseparable_convoulution is
Port ( clk : in  STD_LOGIC;
reset : in  STD_LOGIC;
enable : in  STD_LOGIC;
separeble_convoulution_result : out  STD_LOGIC_vector(26 downto 0));
endseparable_convoulution;

architecture Behavioral of separable_convoulution is
componentvcontroller_vmultipliers is
port(reset:instd_logic;
enable:instd_logic;
clk:instd_logic ;
out_enable_vmultiplier : out std_logic;
        out_vmultiplier0:outstd_logic_vector(24 downto 0);
        out_vmultiplier1:outstd_logic_vector(24 downto 0);
        out_vmultiplier2:outstd_logic_vector(24 downto 0));
end component;
componentadder_vertical_conv is
```

```
   Port ( clk : in  STD_LOGIC;
adder_vertical_enable:in  STD_LOGIC;
in0 : in  STD_LOGIC_VECTOR (24 downto 0); …in2 : in  STD_LOGIC_VECTOR (24 downto 0);
separeble_convoulution_result : out  STD_LOGIC_VECTOR (26 downto 0));
end component;
begin
vcontroller_vmultiplier_motif:vcontroller_vmultipliers
port map (reset=>reset,
enable=> enable,
clk=>clk,
out_enable_vmultiplier=>enable_vmultiplier_adder,
        out_vmultiplier0=> out_multiplier0_in0_adder_sig,
        out_vmultiplier1=>out_multiplier1_in1_adder_sig,
        out_vmultiplier2=>out_multiplier2_in2_adder_sig);
adder_vertical_conv_motif:adder_vertical_conv
port map(clk=>clk,
          adder_vertical_enable=>enable_vmultiplier_adder,
            in0=>out_multiplier0_in0_adder_sig, ……in2=>out_multiplier2_in2_adder_sig,
          separeble_convoulution_result=>separeble_convoulution_result);
endBehavioral;
```

Code VHDL convolution séparable 5x5

Code VHDL du filtre Gaussien séparable (5x5)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entityconvolution_separable is
Port (clk, reset, enable,rw_ram: in  STD_LOGIC;
sel : out std_logic_vector (1 downto 0);
addresse_h,addresse_v,cmptr_ecriture : out std_logic_vector (5 downto 0);
qq : out std_logic_vector (5 downto 0);
separeble_convoulution_result : out  STD_LOGIC_vector(26 downto 0));
endconvolution_separable;
architecture Behavioral of convolution_separable is
componentvcontroller_vmulls is
port(   reset, enable,clk:instd_logic;
out_enable_vmultiplier : out std_logic;
rw_ram: out std_logic;
            sel : out std_logic_vector (1 downto 0);
            addresse_h,addresse_v,cmptr_ecriture : out std_logic_vector (5 downto 0);
       out_vmultiplier0,……., out_vmultiplier4:out std_logic_vector(24 downto 0));
end component;
componentvertical_additionneur is
   Port ( clk : in  STD_LOGIC;
adder_vertical_enable:in  STD_LOGIC;
 in0,…..in4 : in  STD_LOGIC_VECTOR (24 downto 0);
```

```
separeble_convoulution_result : out  STD_LOGIC_VECTOR (26 downto 0));
end component;
begin
uut16: vcontroller_vmulls
port map (  reset=>reset, enable=> enable, clk=>clk,    rw_ram =>rw_ram,
          cmptr_ecriture =>cmptr_ecriture, addresse_h =>addresse_h,
addresse_v =>addresse_v, sel =>sel,
          out_enable_vmultiplier=>enable_vmultiplier_adder,
           out_vmultiplier0=>out_multiplier0_in0_adder_sig,
               .
               .
           out_vmultiplier4=>out_multiplier4_in4_adder_sig
                              );
 uut17: vertical_additionneur
port map(   clk=>clk,adder_vertical_enable=>enable_vmultiplier_adder,
          in0=>out_multiplier0_in0_adder_sig,
               ….
               …..
          in4=>out_multiplier4_in4_adder_sig,
          separeble_convoulution_result=>separeble_convoulution_result);
end Behavioral;
```

Code Matlab du filtre Gaussien orientable

“steerGauss.m”

```
function [J,h] = steerGauss(I,arg1,arg2,arg3)
% La première étape : L’affectation des paramètres de l’algorithme.
if ~exist('I','var') || isempty(I)
  I = imread('im IRM.jpg');
end
I = mean(double(I),3);
% Les arguments  d'entrées  du processus.
if ~exist('arg1','var') || ~isstruct(arg1)
   % Usage: [J,H] = STEERGAUSE(I,THETA,SIGMA,VIS)
usageMode = 1;
   % Attribuer  l’orientation du filtre par défaut
if ~exist('arg1','var') || isempty(arg1)
     arg1 = 0;
end
theta = -arg1*(pi/180);
if ~exist('arg2','var') || isempty(arg2)
arg2 = 1;
end
sigma = arg2;
   % Attribuer l’état de visualisation par défaut.
if ~exist('arg3','var') || isempty(arg3)
      arg3 = false;
end
vis = arg3;

else
   % Usage: [J,H] = STEERGAUSE(I,H,VIS)
```

```
usageMode = 2;
   % Extraction les paramètres de filtre.
h = arg1;
theta = -h.theta*(pi/180);
sigma = h.sigma;
   g = h.g;
gp = h.gp;
if ~exist('arg2','var') || isempty(arg2)
arg2 = false;
end
     vis = arg2;
end % End of input pre-processing.
% La deuxième étape : L'évaluation  des noyaux de filtre séparable.
% Calcule le  noyau du filtre.
ifusageMode == 1
  % Déterminer le supporte du filtre nécessaire
Wx = floor((5/2)*sigma);
ifWx< 1
Wx = 1
end
  x = [-Wx:Wx];
% la dérivé gaussienne
  g = exp(-(x.^2)/(2*sigma^2));
  gp = -(x/sigma^2).*exp(-(x.^2)/(2*sigma^2));
% stockage de filter.
h.g = g;
  h.gp = gp;
h.theta = -theta*(180/pi);
h.sigma = sigma;
end
% La troisième étape : La détermination de  la réponse de filtre orienté.
```

```
% utilize la séparabilité
Ix = conv2(conv2(I,-gp,'same'),g','same');
Iy = conv2(conv2(I,g,'same'),-gp','same');
% le filter orientable
J = cos(theta)*Ix+sin(theta)*Iy;
% La quatrième étape : La visualisation.
ifvis
figure(1); clf; set(gcf,'Name','Oriented Filtering');
subplot(2,2,1); imagesc(I); axis image; colormap(gray);
title('Input Image');
subplot(2,2,2); imagesc(J); axis image; colormap(gray);
title(['Filtered Image (\theta = ',num2str(-theta*(180/pi)),'{\circ})']);
subplot(2,1,2); imagesc(cos(theta)*(g'*gp)+sin(theta)*(gp'*g));
axis image; colormap(gray);
title(['Oriented Filter (\theta = ',num2str(-theta*(180/pi)),'{\circ})']);
end
« runDemo.m »
% Clear Matlab command window.
clc;
clear all;
% Example #1: usage principal .
% Note: Les filtres sont calculés pour chaque exécution.
disp('Example #1: usage principal);
theta = [0:30:360];
for i = 1:length(theta)
  [J,H] = steerGauss([],theta(i),05,true);
filters{i} = H;
pause(0.1);
end
disp('   c.'); pause;
I = imread('im IRM.jpg');
```

```
% Example #2: utilize le filtre pré-calculé
disp('Example #2: L'utilisation de filtres pré-calculées');
for i = [1:length(filters)]
  [J,H] = steerGauss(I,filters{i},true);
pause(0.1);
end
```